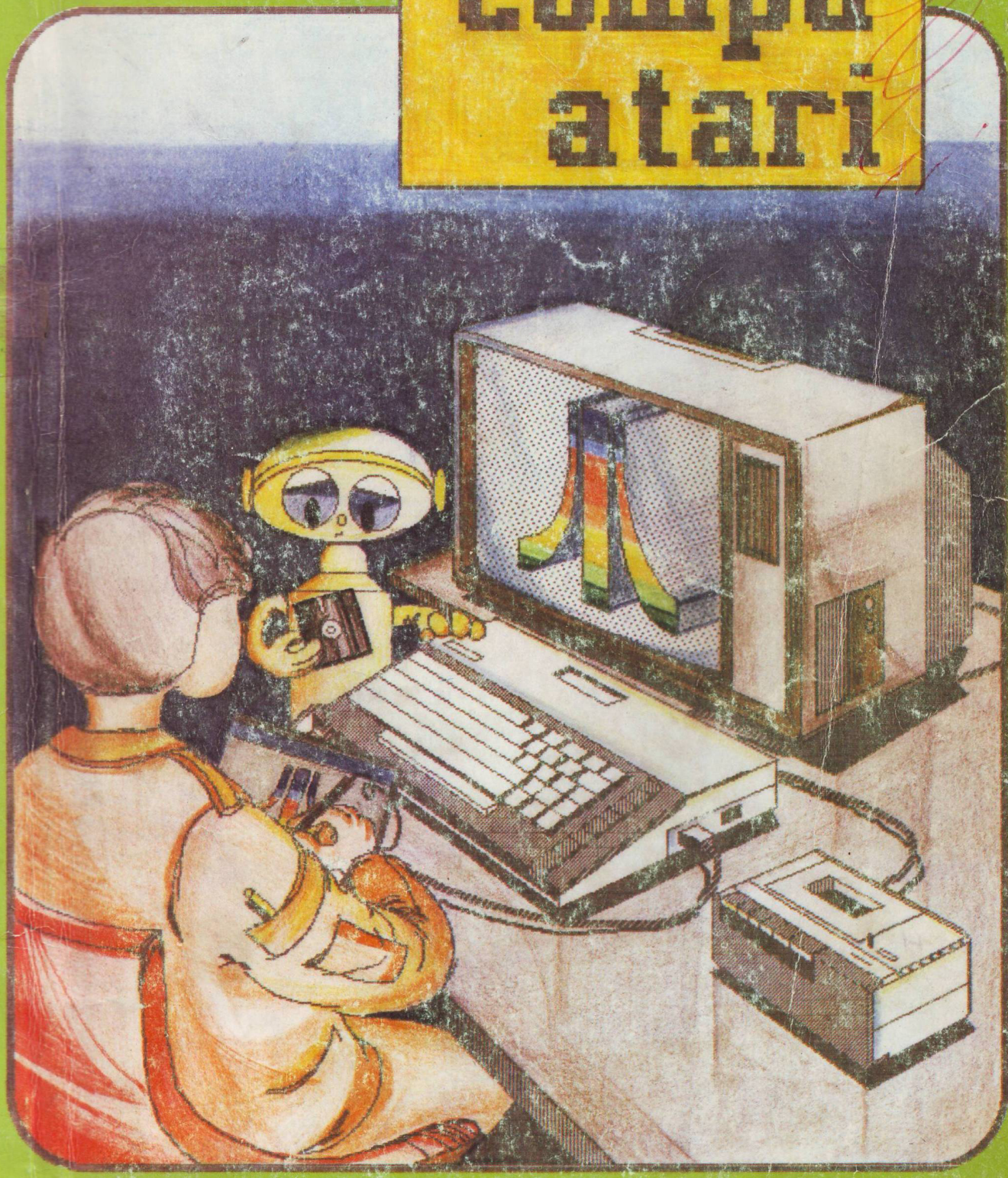
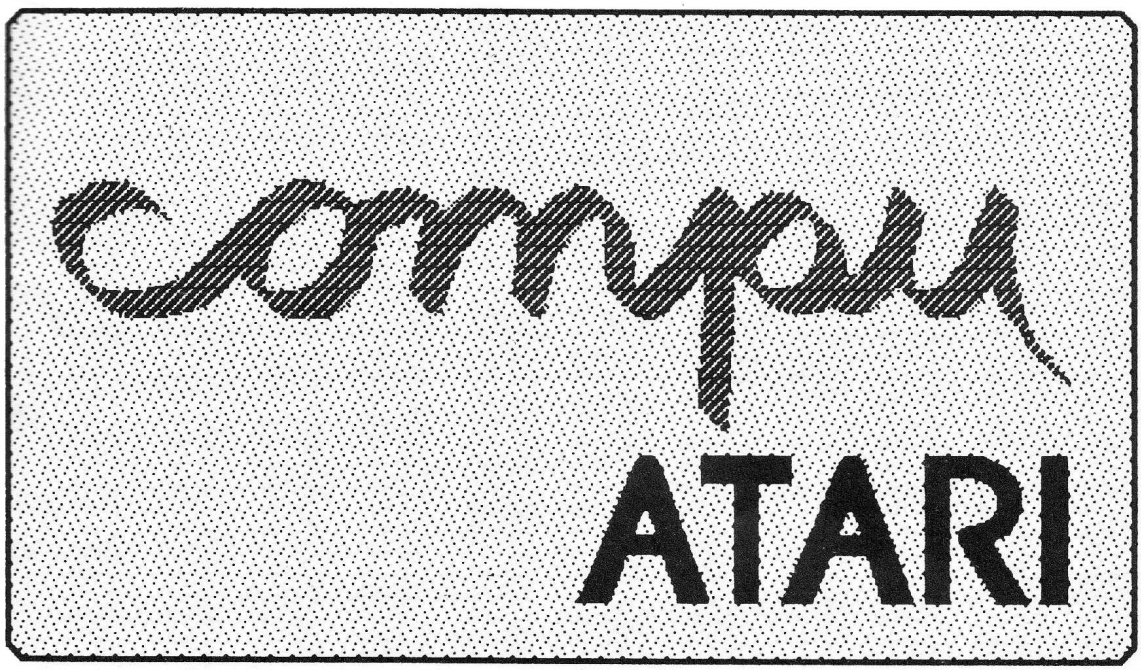
compu atari



EDICIONES COMPUGRAFICA



AUTORA:

VERONICA ESPINOSA H.

PORTADA E ILUSTRACIONES:
LUIS CORVALAN VELIZ

EDICIONES COMPUGRAFICA

INTRODUCCION

¿Y esa cara tan triste? ¿Qué pasa?

ES QUE QUIERO HACER TANTAS COSAS CON MI ATARI, PERO NO SE POR DONDE EMPEZAR. SE QUE SE PUEDEN HACER TODAS LAS COSAS QUE YO QUIERO ¡PERO NO SE COMO!

¡Calma! Recuerda que si quieres ser un experto con tu Atari debes ser muy organizado.
Veamos, ¿qué te gustaría hacer?

BUENO. ME GUSTARIA HACER MIS PROPIOS PROGRAMAS, Y USAR LOS JUEGOS Y PROGRAMAS EDUCATIVOS. ME GUSTARIA HACER HARTOS GRAFICOS Y PODER DEJAR GRABADO TODO LO QUE HAGA.

Bien. Poco a poco iremos revisando todas las cosas que quieres aprender y muchas otras que cuando las conozcas te encantarán. Es importante que tengas claro que este no es un manual de operación, sino una guía en todos los aspectos que te interesan y no aparecen en los manuales del equipo.

AH! MUCHO MAS ENTRETENIDO ENTONCES.

Ojalá así sea. Primero me interesa conocer cual es el nivel de tus conocimientos así es que me gustaría que conversáramos un poco.

¿ME VAS A HACER UNA PRUEBA?

Algo así. Me gustaría saber que es para ti un computador.

BUENO, UN COMPUTADOR ES .... ES UNA HERRAMIENTA A LA QUE YO LE PIDO QUE HAGA ALGUNAS COSAS Y SIEMPRE LAS HACE BIEN.

Tienes razón. El computador es una herramienta, una máquina que tiene memoria y puede realizar operaciones aritméticas y lógicas. Para poder hacer todo esto necesita varias partes o componentes ¿sabes cuales son?.

¡HUMM! CREO QUE SI. AQUI TENGO MI ATARI Y TE VOY A IR ENUMERANDO SUS PARTES:

- TENGO UN TECLADO QUE USO PARA ESCRIBIR.

Así es. El teclado se conoce como "Unidad de Entrada" y sirve para ingresar información al computador.

- TENGO TAMBIEN UNA PANTALLA EN LA CUAL VEO TODO LO QUE HACE MI ATARI.

La pantalla o monitor es la "Unidad de Salida" más usada en los microcomputadores y puede ser un televisor corriente.

- Y UNA IMPRESORA. ES BIEN CHIQUITITA PERO ES RAPIDA.

La impresora, igual que la pantalla, se utiliza como unidad de salida. La diferencia es que con la impresora obtienes lo que se conoce como "copia dura" del trabajo en cambio en la pantalla tienes una "copia blanda".

- LO ULTIMO QUE TENGO ES UNA DISKETTERA. AL PRINCIPIO TENIA UNA CASSETTERA PERO EN LOS CASSETTES NO ME CABIA MUCHO Y SE DEMORABA DEMASIADO.

Las cassetteras y disketteras son los medios de almacenamiento auxiliar más utilizados en los microcomputadores. Y hablando de memoria, creo que se te olvida un componente.

PERO SI LOS DIJE TODOS.

¿Seguro? Fíjate en el teclado.

AH DE VERAS. UN SEÑOR DE ATARI ME LO EXPLICO. EN LA MISMA CAJA DONDE ESTA EL TECLADO VIENE LO MAS IMPORTANTE DEL EQUIPO: EL PROCESADOR Y LA MEMORIA PRINCIPAL.

Así es. La memoria del computador sirve para almacenar datos.

SI, ME ACUERDO QUE EXISTEN DOS TIPOS DE MEMORIA PRINCIPAL:
- LA RAM QUE SE UTILIZA COMO MEMORIA DE TRABAJO PARA IR DEJANDO LOS PROGRAMAS QUE HAGO.
- LA ROM TRAE GRABADOS LOS PROGRAMAS BASICOS DE OPERACION QUE PROPORCIONA EL FABRICANTE DEL EQUIPO.

Me alegra saber que conoces bien el equipo, así podremos comenzar

rápidamente a utilizarlo.

¡QUE RICO! PORQUE LA VERDAD ES QUE ABURRE UN POCO TODA LA PARTE TEORICA.

De acuerdo, pero tienes que tener en cuenta que es indispensable conocerlo. En la medida que domines los aspectos teóricos podrás hacer un mejor uso de tu equipo.

TU DIJISTE QUE ESTE NO ES UN MANUAL DE OPERACION. ENTONCES ¿QUE VAMOS A ESTUDIAR?.

Lo mejor es que revisemos nuestro índice punto por punto. Primero veremos algunas técnicas de programación.

¿TECNICAS DE PROGRAMACION? AH! APUESTO QUE SE TRATA DE ALGUNOS TRUCOS PARA PROGRAMAR EN MI ATARI.

Algo así. Para esto se supone que tu sabes BASIC.

BUENO. NO SOY EXPERTO PERO TENGO EL MANUAL Y CUANDO TENGO DUDAS RECURRO A EL.

Me parece muy bien, esa es la idea. Por eso en el capítulo de Técnicas de Programación veremos algunas instrucciones de BASIC avanzadas, que solo tienen los equipos más modernos y por supuesto, muchos ejemplos que te servirán de guía para hacer tus propios programas.

SI, ME INTERESA. ¿Y DESPUES?

El Segundo capítulo lo dedicamos exclusivamente a las variables del Atari BASIC.

AH! SEGUIMOS CON BASIC. MENOS MAL QUE ESTE NO ES UN MANUAL DE BASIC .....

Es verdad, no es un manual de BASIC. Lo que pasa es que, como ya debes saber, el BASIC es el lenguaje más utilizado en los microcomputadores y viene con tu equipo. Además es impresionante la variedad de aplicaciones que tiene.

SI, A MI ME ENCANTA. UN AMIGO ME DIJO QUE CONOCIENDO BIEN UN LENGUAJE ES MUCHO MAS FACIL APRENDER OTROS.

Correcto, porque todos, o casi todos, tienen las mismas bases.
Y así llegamos al tercer capítulo que también tiene relación con BASIC. Esta vez se trata del uso de los Peeks y Pokes en el Atari.

¡LOS FAMOSOS PEEKS Y POKES! POR FIN PODRE SABER DE QUE SE TRATAN.

Ojalá sea así. El cuarto capítulo es un poco más técnico, se trata del uso de la diskettera y cassettera en el Atari.

¿PODRE CONOCER BIEN LAS VENTAJAS DE ESOS MEDIOS DE ALMACENAMIENTO?

Y más que eso: podrás sacarle el máximo provecho. Lo mismo al Joystick, del que hablamos en el capítulo quinto.

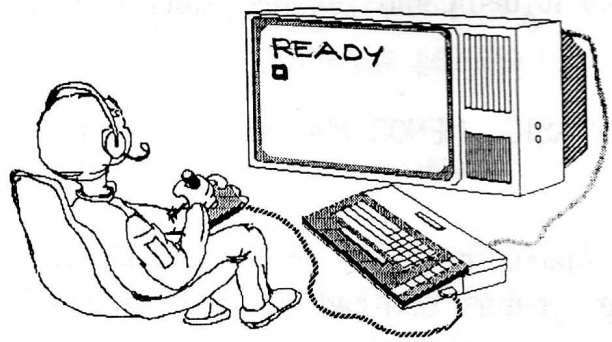
¡ME MUERO DE GANAS DE EMPEZAR LUEGO! IMAGINATE CON TODAS ESTAS COSAS NUEVAS LOS PROGRAMAS QUE VOY A PODER HACER.

Y para darte aún mas ayuda, en la última parte de este libro encontrarás un Anexo con algunos conceptos básicos de utilización del Atari. Así puedes recordar aspectos de tu Atari que se te han olvidado, o aprenderlos si no los conoces.

¡FANTASTICO! ¿EMPEZAMOS?

Una última cosa. Me interesa saber si conoces bien tu Atari. ¿lo encendemos?

¡YA! ¡ENCENDI MI EQUIPO! MIRA LO QUE APARECE EN LA PANTALLA



Cuando ves la palabra READY en la pantalla, significa que el computador está listo para que lo utilices.
El cuadrado verde es el cursor. A medida que tú ingresas las instrucciones, el cursor se mueve dentro de la pantalla y te indica donde aparecerá la próxima letra que escribas.
¡TANTAS TECLAS! ES VERDAD QUE EL TECLADO ES MUY PARECIDO AL DE UNA MAQUINA DE ESCRIBIR, PERO HAY OTRAS TECLAS DIFERENTES.

Así es. Dentro de las teclas diferentes, existen dos que son muy importantes: RETURN y RESET.

¿Y QUE SIGNIFICAN ESAS TECLAS?

La tecla RETURN marca el final de un ingreso de información a través del teclado. Ya la veremos en detalle más adelante.

¿Y EN BASIC ES DIFERENTE, VERDAD?.

Sí, ahí marca el final de una línea que puede ser:

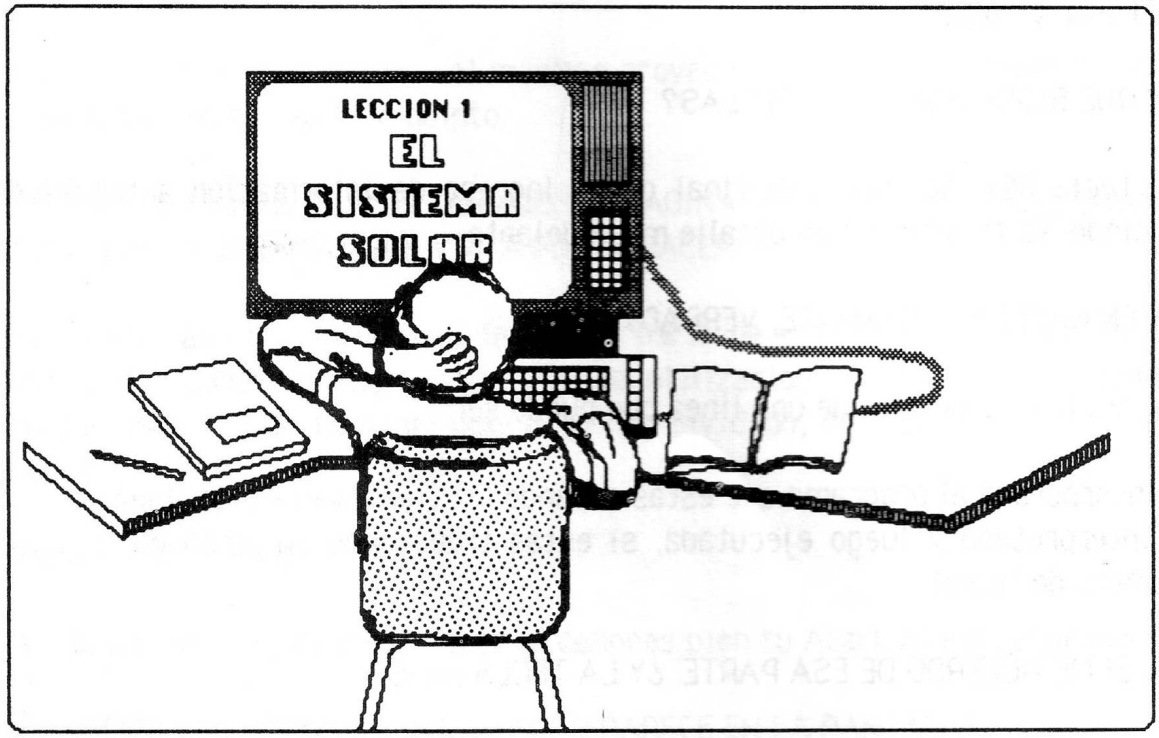
- incorporada al programa que estás creando (si tiene número de línea).
- interpretada y luego ejecutada, si estás trabajando en modo directo (sin número de línea).

SI, SI ME ACUERDO DE ESA PARTE. ¿Y LA TECLA RESET?.

La presión de la tecla RESET deja al computador en su condición inicial normal, con la pantalla limpia y modo texto.

TODO CLARO. ¿EMPEZAMOS?.

Empecemos .....


LECCION 1
EL
SISTEMA
SOLAR

CAPITULO 1

TECNICAS DE PROGRAMACION

Como mencionamos en la Introducción, comenzaremos viendo algunas Técnicas de Programación en BASIC.

YA, PERO RECUERDA QUE YA TENGO UN MANUAL DE BASIC. ESO SI, NO ES MUY COMPLETO. A VECES QUIERO HACER ALGUNAS COSAS Y NO ENCUENTRO COMO.

De eso trata este capítulo. BASIC es un lenguaje con un conjunto pequeño de instrucciones y una sintáxis muy simple. Entonces ahora trataremos de abordar diferentes problemas con estas herramientas.

DEBE SER MUY DIFICIL HACER UN LENGUAJE DE PROGRAMACION. ¿A QUIEN SE LE OCURRIO EL BASIC?.

El lenguaje BASIC fue creado por John Kemenny y Thomas Kurtz en 1965 inicialmente orientado a aplicaciones numéricas.

¿SOLO APLICACIONES NUMERICAS? PERO YO PENSE .....

Lo que pasa es que se ha difundido y mejorado de tal forma que hoy permite resolver diversas aplicaciones, transformándose en uno de los lenguajes más usados en mini y microcomputadores, por su facilidad de uso, aprendizaje y versatilidad en el manejo.

ALGO HE ESCUCHADO DE DISTINTOS BASIC, YO PENSABA QUE HABIA UNO SOLO.

El BASIC ha ido evolucionando y han ido apareciendo distintas versiones, primero para computadores grandes y actualmente para los microcomputadores. La más popular de estas versiones para minicomputadores es la familia de BASIC de la Digital Equipment Corporation (DEC).

¿COMO FAMILIA? SI ESTAMOS HABLANDO DE UN LENGUAJE, NO DE PERSONAS.

Es una forma de expresarse. Existen muchas versiones de BASIC, con

distintos orígenes. Las versiones más importantes de la familia DEC son: BASIC - 11, BASIC - PLUS y BASIC - PLUS 2.

¿Y POR QUE SON MAS IMPORTANTES? SUPONGO QUE TENDRAN ALGUNAS VENTAJAS FRENTE A OTRAS VERSIONES .

Estas versiones de BASIC poseen instrucciones especiales que no tienen otras.

BUENO, PERO SUPONGO QUE HABRA INSTRUCCIONES COMUNES A TODAS LAS VERSIONES.

Claro que sí. Las instrucciones más comunes son:

- **PRINT** que despliega información en la pantalla o en otro dispositivo de salida.

GOTO ¡YO SE! GOTO PERMITE SALIRSE DE LA SECUENCIA NORMAL DE UN PROGRAMA E IR A LA INSTRUCCION SEÑALADA A CONTINUACION DE LA INSTRUCCION GOTO.

Me olvidaba que sabes BASIC ¿sabes entonces para qué sirven las instrucciones **END**, **LET**, **LIST**, **RUN**, **REM**, etc.?

SI. LAS USO MUCHO EN MI ATARI. .

Pues muy bien. Aparte de estas instrucciones comunes hay otras especiales.

¿COMO CUALES?.

Por ejemplo, están las que se conocen como "funciones especiales de manipulación de strings".

¿Y QUE SON LOS STRINGS?

Un string es un conjunto de letras o caracteres alfanuméricos (letras, números y signos combinados) tratados como una unidad.

DAME UN EJEMPLO, POR FAVOR.

Tu nombre es un string, o sino se consideraría cada letra por separado, como

caracteres independientes. Esta es la manera como el computador ve los textos.

AH! YA ENTIENDO ¿Y CUALES SON ESAS FUNCIONES ESPECIALES DE MANIPULACION DE STRINGS?

MID$, LEFT$ y RIGHT$ y la posibilidad de crear "arreglos de Strings".

HE ESCUCHADO UN PAR DE VECES LA PALABRA ARREGLO, PERO NUNCA RELACIONADA CON STRINGS.

Un arreglo es un conjunto cualquiera de elementos, en este caso de Strings. Puedo tener, por ejemplo, un arreglo con el nombre de los meses del año, los días de la semana, colores, etc.

ES IGUAL A LOS ARREGLOS DE NUMEROS. PERO SABES UNA COSA, AHORA QUE RECUERDO, YO HE TRABAJADO CON MICROSOFT BASIC Y TE PERMITE HACER COSAS ESPECIALES CON LOS STRINGS.

Es que Microsoft BASIC es uno de los descendientes de esta familia. Otros productores, como Data General, usan una sintaxis diferente para manejar substrings.

¿Y COMO ES?

Así: A$ (X,Y) elimina el trabajo innecesario que debe hacerse con (MID$). Atari BASIC es de este tipo.

PERO ES MAS CONOCIDA LA FUNCION MID$ AUNQUE ES MAS COMPLICADA.

Por eso, es tan importante tener siempre presente que una sintaxis simple no significa necesariamente menos poder. En efecto, una sintaxis BASIC más simple, puede significar que el intérprete BASIC usa menos recursos de Hardware.

SI, ES INTERESANTE TU TEORIA.

¿Te gustaría aprender algunos trucos de programación?

¡CLARO QUE SI!

Una característica especial del Atari, es la capacidad que tiene de direccionar Strings muy largos.

¿Y QUE PUEDO HACER CON STRINGS LARGOS?

Bueno, quién te dice que no puedes construir un Procesador de Textos o un Editor, en BASIC, cuando tienes el poder para controlar y manipular un gran número de palabras.

PERO DEBE SER MUY LARGO Y DIFICIL.

Uno puede inicializar un gran string de una longitud difícil de manejar, sin ocupar instrucciones largas y complicadas.

```
1000 E$ = " " : TRAP 1010: FOR J = 1 TO 15 : E$ (LEN(E$)+1)=E$
1010 NEXT J
```

VEAMOS ¿QUE OTRAS COSAS PODRIA HACER?

Para aplicaciones de Bases de Datos, un administrador generalizado de "formatos de pantalla", podría construir y almacenar formatos de pantalla de entrada como grandes "imágenes tipo string". Uno podría incluir información especifica en una imagen tipo string con una sola instrucción.

```
1200 REC$ (J +1 - LEN (EX$), J) = EX$
```

¿Y QUE HACE ESA INSTRUCCION?

Lo que hace es colocar el extremo derecho de EX$ en la posición J-ésima de REC$, ajustando a la derecha la información en REC$.

HAGAMOSLO PARA AJUSTAR A LA IZQUIERDA.

Para eso habría que colocar el extremo izquierdo de EX$ en la posición J de REC$.

```
1300 REC$ ( J, J - 1 + LEN (EX$)) = EX$
```

SABES. SIEMPRE HE TENIDO UNA DUDA CON LOS PROCESADORES DE TEXTO. NO SE COMO HACEN PARA ALINEAR A AMBOS LADOS UN TEXTO.

Ah! Muy sencillo. Simplemente, agregan blancos entre las palabras.

¿Y COMO SABEN CUANTOS BLANCOS?

Porque uno sabe cuantas letras caben en una línea. Revisa las siguientes líneas del programa:

```
1400 LN = LEN (EX$)
1410 IF LN < 25 THEN EX$ (LN + 1) = BLANK$ (1,25 - LN)
```

¿Y QUE HACEN ESTAS LINEAS?

Muestran como se "agranda" EX$ agregándole blancos al final hasta que su longitud sea 25. Por supuesto, la primera vez EX$ debe tener menos de 25 posiciones y BLANK$ debe inicializarse como un string de blancos.

PERO NO ME RESULTA PARA STRINGS MAS LARGOS QUE 25.

Cambia el 25 por una variable del tamaño que desees y las mismas líneas en una subrutina te servirán para strings de cualquier largo.

SUPONGO QUE DESPUES PODRE SACAR LOS BLANCOS QUE SOBRAN EN EL STRING.

Prueba estas líneas.

```
1500 LN = LEN (EX$)
1510 IF LN>1 AND EX$ (LN,LN) = " " THEN EX$= EX$ (1,LN-1)
1520 GOTO 1500
```

¡UN LOOP! Y ASI SACO TODOS LOS BLANCOS QUE SOBRAN.

Y puedes hacer lo mismo con los blancos a la izquierda de un string.

```
1600 IF EX$ (1,1)= " " THEN EX$ = EX$ ( 2,LEN(EX$)) : IF LEN (EX$) > = 2 THEN 1600
```

Y SI SUSTITUYO EL BLANCO (" ") POR UNA VARIABLE QUE CONTENGA UN BLANCO, ESO HACE MAS RAPIDA LA EJECUCION DEL PROGRAMA.

¡Qué bien que te guste optimizar tus programas! Es muy importante que lo hagas así, aunque se trate de programas pequeños.

ESO ES LO QUE ME GUSTA DE LA PROGRAMACION: UNO PUEDE CREAR SUS PROPIOS PROGRAMAS Y LUEGO MODIFICARLOS PARA QUE NOS SATISFAGAN CIENTO POR CIENTO.

Así es. A medida que vamos aprendiendo, nos damos cuenta de la infinidad de cosas que se pueden hacer con el Atari.
Y también es importante ir aprendiendo nuevas técnicas, no tan sólo para desarrollar programas más eficientes, sino que también para sacar más provecho del equipo.

A VER, COMO SERIA ESO.

Mira. Revisa el programa que aparece en las páginas siguientes y verás que es muy útil para ingresar una pantalla completa de texto. El computador ubicará automáticamente esta pantalla en un arreglo.

AH! TE REFIERES A LO QUE SE CONOCE COMO EDICION DE PANTALLA.

Así es. La Edición de Pantalla, es un medio muy conveniente para el ingreso de datos y texto en archivos computacionales.

EN REALIDAD ES MUY ABURRIDO INGRESAR DATOS EN UN ATARI. HAY QUE DISEÑAR LA PANTALLA Y LUEGO INGRESAR UN REGISTRO A LA VEZ.

Existen varias soluciones para este problema y una de las mejores se presenta para el Atari 800XL.

¿EN SERIO? ¿Y DE QUE SE TRATA?

La clave es la capacidad de controlar la pantalla y "verter" su contenido rápidamente con la técnica del "teclado dinámico".

¿TECLADO DINAMICO? NO LO HABIA ESCUCHADO NUNCA.

El siguiente programa es un sistema que demuestra este principio.

¿ES MUY COMPLICADA LA RUTINA?

No, es muy simple. Se establecen primero, los márgenes de la pantalla,

haciendo una pantalla más chica. En este ejemplo, las dimensiones de la pantalla son 16 líneas verticales y 35 horizontales. La columna de números que aparece a la izquierda de la pantalla, se supone que está fuera de la pantalla.

Y DESPUES DE PONER LOS MARGENES ¿QUE PASA?

Se toma los caracteres del teclado y se muestran en la pantalla, luego la rutina verifica si se sobrepasó el borde de la pantalla, haciendo un PEEK en las localizaciones 84 y 85.

¿QUE PASA CON LOS RETORNOS DE CARRO (RETURN)?

No tienen otro efecto que imprimir el carácter que lo representa.

¿Y COMO VUELVO AL PRINCIPIO?

En el programa Editor de Pantalla, CONTROL J se usa como una señal. El programa pide una confirmación y si se escribe una "S", se pasa a la rutina del "teclado dinámico".

¿QUE PASA CON ESA RUTINA?

En la rutina del teclado dinámico, el cursor se ubica o coloca (POSITION) y se pide una entrada (INPUT) desde el editor, y el teclado dinámico se activa y desactiva a medida que se reconocen las 16 líneas.

¡YA! QUIERO VER EL PROGRAMA.

Está bien. Aquí lo tienes, úsalo:

```
0   REM SAVE "D:EDITOR"
10  REM *EDITOR DE PANTALLA*
100 DIM B$(600), A$(40)
110 ? CHR$(125)
120 GOSUB 500
130 OPEN #1,4,0,"K:"
140 REM ************************************************
150 POSITION 3,0:? :GOSUB 1000
160 GET #1,T
165 IF  T=10  THEN 230:REM 10=CTRL"J"
170 IF  T=20  THEN 150:REM 20=CTRL"T"
180 IF  PEEK(85)=36  THEN  GOSUB 1000
```

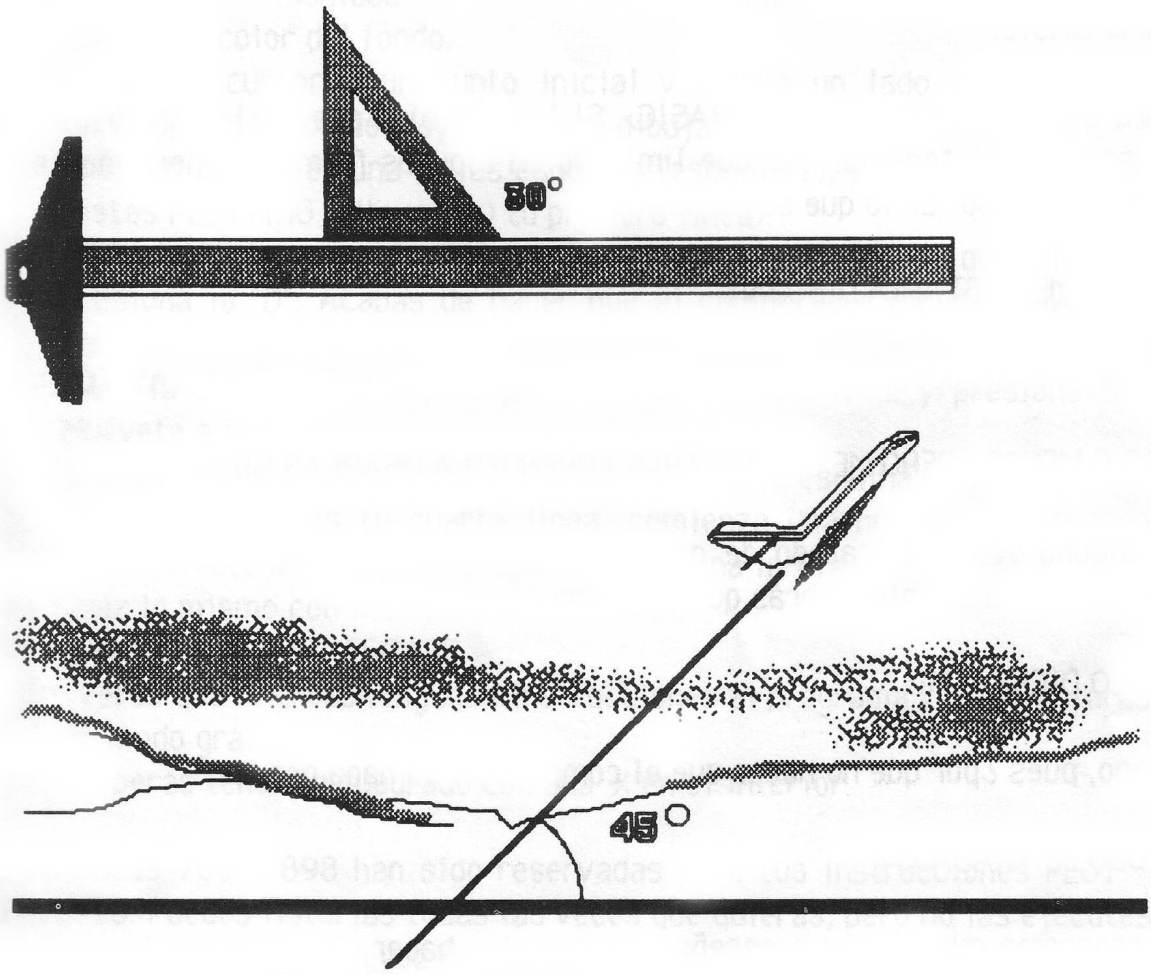
```
190 IF  PEEK(85)>38 THEN POKE 85,38:GOSUB 1000:GOTO 160
200 IF PEEK(84)>16 OR PEEK(84) <1  THEN 150
210 IF  T=156 OR T=157 OR T=125  THEN 160
220 ? CHR$(T);:GOTO 160
230 POSITION 3,18
235 ? "ESTA SEGURO?";:GET #1,X
236 POSITION 3,18
237 ? "                                "
240 IF X<>ASC("S") THEN 150
250 REM ********************************************************
260 FOR I=1  TO  16
262 POSITION 3,I
264 POKE 842,13
266 INPUT A$
270 POKE 842,12
280  B$(I*35-34,I*35)=A$
290 NEXT I
300 REM ********************************************************
310 STOP
315 ? CHR$(125)
320 FOR  I=1  TO  16
330 ? B$(I*35-34 ,I*35)
340 NEXT I
350 GOSUB 500
355 GOTO 150
360 STOP
500 B$(1)="     "
510 B$(600)="          "
520  B$(2)=B$
600 POKE 82,0
602 POSITION 0,0:?
604 FOR  I=1  TO 16:? I:NEXT I
606 POKE 82,4
610 POSITION 3,0
620 ? "USE CTRL J PARA ACEPTAR" :RETURN
1000 FOR  I=1 TO 5
1010 SOUND 0,50,10,10
1020 NEXT I
1030  SOUND 0,0,0,0:RETURN
```

¡TANTOS POKES!

No son tantos. Sólo los que aparecen entre las líneas 260 y 270. Sirven para leer desde la pantalla a la variable llamada A$. Ya aprenderemos más adelante como usar bien las instrucciones PEEK y POKE.

EN REALIDAD ESTA ES UNA RUTINA, SUPONGO QUE DEBE FORMAR PARTE DE ALGUN PROGRAMA.

Así es. Generalmente, A$ se escribe a un archivo en disco. En este programa


30°
45°

de ejemplo, simplemente lo copio a otro string (B$).

¿Y PARA QUE LA INSTRUCCION STOP?

Este programa de demostración se detiene cuando se transfiere el contenido desde A$ a B$. Si se usa CONT se repite el ciclo completo.

ME SORPRENDE BASTANTE QUE SE PUEDA HACER ALGO ASI CON UN LENGUAJE COMO BASIC.

Para que no menosprecies el BASIC. Si lo usas bien puedes lograr cosas más sorprendentes todavia. Aunque limitado en algunas formas, generalmente es más "amistoso" de lo que se piensa.

AHORA ME GUSTARIA DIBUJAR.

¿Dibujar?. Si, puede ser. ¿Has usado las instrucciones **PLOT** y **DRAWTO**?

MAS O MENOS. PERO ME ENCANTARIA APRENDER A USARLAS BIEN.

Si alguna vez haz tratado de graficar alguna figura complicada, e incluso alguna figura simple, sabrás que consume mucho tiempo.

¡ME LO DICES A MI!

Bueno, pues ¿por que no hacer que el computador lo haga por ti?

¿TODO EL TRABAJO?

El programa que te voy a enseñar, te permite hacer el dibujo en la pantalla usando el joystick, y luego el computador lo grafica bien con las instrucciones PLOT y DRAWTO.

¡FANTASTICO! ¿QUE TENGO QUE HACER?

Escribe cuidadosamente el programa, sin tratar de acortarlo para que no te equivoques.

NO TE PREOCUPES, SIEMPRE SIGO TUS INSTRUCCIONES AL PIE DE LA LETRA.

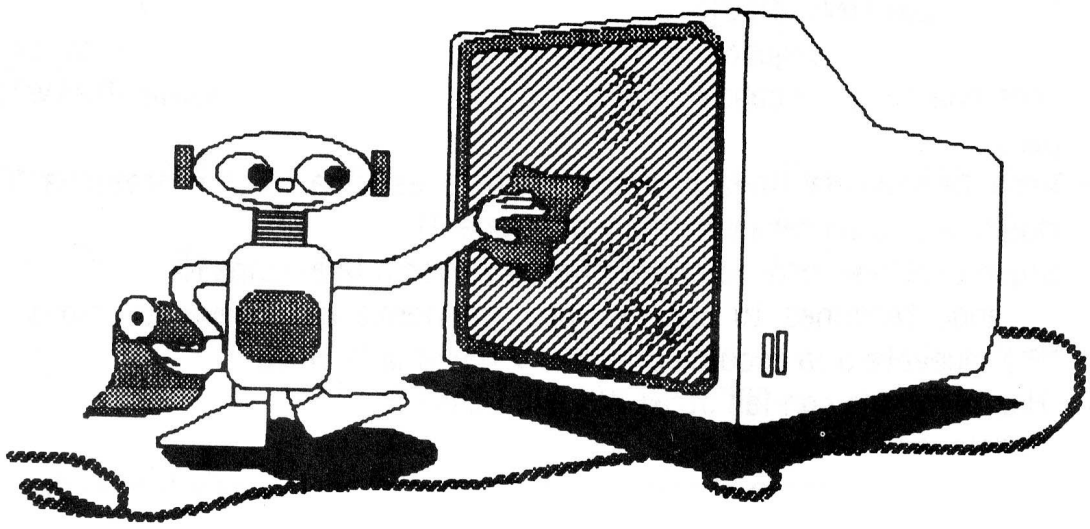
Los siguientes pasos, te muestran como hacer un cuadrado con una equis (x), en su interior. A primera vista, te parecerá un poco confuso, pero a medida que vayas aprendiendo, te saldrá mucho más fácil.

1.- Escoge el modo gráfico que quieras. Te sugiero el modo gráfico 3 para empezar.
2.- Escoge el color del cursor.
3.- Escoge la luminosidad del cursor (con 7 obtendrás una buena sombra)
4.- Escoge el color del fondo.
5.- Mueve al cursor a un punto inicial y dibuja un lado del cuadrado (presiona el botón del joystick para dibujar).
6.- Ubica el cursor en una de las esquinas (esquina 1), y luego presiona "P" (estás PLOTTING (dibujando) tu primera línea).
7.- Muévete a la segunda esquina, no importa el camino que sigas y presiona la "D". Acabas de hacer que el computador dibuje (DRAWTO) para tí.
8.- Para la segunda línea permanece en la esquina Nº 2 y presiona "P". Muévete a la tercera esquina y presiona "D".
9.- Sigue el mismo procedimiento hasta volver a la esquina Nº 1.
10.- Cuando termines tu cuarta línea, comienza una nueva (presionando "P"). Muévete a la esquina diagonal y termina la línea.
11.- Haz lo mismo con las otras dos esquinas.
12.- Presiona la tecla ESCAPE.
13.- Veras que ingresan algunas instrucciones al programa y luego estarás en modo gráfico.
14.- Deberás tener tu cuadrado con una X en el interior.

Las líneas 700 - 898 han sido reservadas para tus instrucciones PLOT - DRAWTO. Puedes listarlas todas las veces que quieras, pero no las ejecutes para volver al dibujo. En vez de eso escribe "G.4000".
Te darás cuenta que el cursor se mueve lentamente en modo gráfico 8. Si puedes dibujar más rápido, presiona la "F". Cuando quieras dibujar mas lento, deberás disminuir la velocidad del cursor, presionando la "S".
Si quieres comenzar una nueva figura o dibujar debes sacar las líneas 700-898. Asegúrate que la línea 700 sea:

```
700 RETURN
```

COMPU ATARI



Aqui tienes el Programa.

```
  0   REM SAVE "D:DIBUJA"
  1   REM  *PROGRAMA QUE DIBUJA*
  2   REM  *** ------------>  *
  3   DIM AN$(1):LI=699:GRAPHICS 0:SETCOLOR 2,0,0:? :? :? "MODO GRAFICO (3 AL 8)";:INPUT
      MODO:IF MODO=8THEN 5.
  4   ? :? "COLOR DEL CURSOR (0 A 14)";:INPUT CC:? :? "LUMINOSIDAD DEL CURSOR (0 A
      14)";:INPUT CL
  5   ? :? "FONDO DE COLOR (SI O NO)";:INPUT AN$:IF AN$="N" THEN 7
  6   ? :? "COLOR DEL FONDO (0 A 14)";:INPUT BC:? :? "LUMINOSIDAD DEL FONDO (0 A 14)";:INPUT
      BL
  7   IF MODO=8 THEN  HRNG=319:VRNG=159
  8   IF MODO=6 OR MODO=7  THEN HRNG=159:VRNG=79
  9   IF MODO=4 OR MODO=5 THEN HRNG=79:VRNG=39
 11   IF MODO=3  THEN HRNG=38:VRNG=19
 14   GRAPHICS  MODO:SETCOLOR  2,BC,BL:SETCOLOR  0,CC,CL:SETCOLOR    4,BC,BL:COLOR  1:POKE
      752,1:L=500:GOSUB  700
 17   ? "HORIZONTAL - ":? "VERTICAL - "
 20   GOSUB 100:IF STRIG(0)=0  THEN PLOT H,V:GOTO 20
 30   PLOT H,V:FOR X=1 TO 5:NEXT X:POSITION  H,V:? #6;"  ":GOTO 20
100   S=STICK(0):IF S=11  THEN H=H-1
105   IF S=5 THEN H=H+1:V=V+1
110   IF S=7 THEN H=H+1
115   IF S=6 THEN H=H+1:V=V-1
120   IF S=14 THEN V=V-1
125   IF S=13 THEN V=V+1
130   IF S=9 THEN H=H-1:V=V+1
135   IF S=10 THEN H=H-1:V=V-1
140   IF H<0 THEN H=0
142   IF H>HRNG THEN H=HRNG
144   IF V>VRNG THEN V=VRNG
150   IF V<0 THEN V=0
155   IF  MM=0  THEN  POKE  656,0:POKE  657,18:?  "MEM.DISP.:";FRE(0):MM=1
160   IF PEEK(764)=56 THEN GOSUB 2000:POKE 656,2:? "RAPIDEZ(DIBUJA MAS LENTO)
      ":FT=1:GOSUB  440
165   IF PEEK(764)=62 OR OK=0 THEN GOSUB 2000:POKE 656,2:? "***  DIBUJANDO
      FACILMENTE***";FT=0:GOSUB  440:OK=1
170   IF FT=1 THEN RETURN
180   IF PEEK(764)=10 THEN GOSUB 2000:POKE 656,1:POKE 657,15:? "GRAF.";H;",";V;"    ":GOSUB
      L:GOSUB 440
185   IF PEEK(764)=28 THEN  1010
190   IF PEEK(764)=58 THEN GOSUB 2000:POKE 656,1:POKE 657,27:? "DIB.";H;",";V;  "    ":GOSUB
      L:GOSUB 440
210   POKE 656,0:POKE 657,13:? H;"  "
230   POKE 657,11:? V;"  "
300   RETURN
440   POKE  764,255:RETURN
500   L=503:A=H:B=V:RETURN
503   L=510:C=H:D=V:RETURN
510   L=513:E=H:F=V:RETURN
513   L=520:G=H:I=V:RETURN
```

```
 520   L=523:J=H:K=V:RETURN
 523   L=530:M=H:N=V:RETURN
 530   L=533:O=H:P=V:RETURN
 533   L=540:Q=H:R=V:RETURN
 540   L=543:SS=H:T=V:RETURN
 543   L=550:U=H:W=V:RETURN
 550   L=553:Y=H:Z=V:RETURN
 553   L=560:AA=H:AB=V:RETURN
 560   L=563:AC=H:AD=V:RETURN
 563   L=570:AE=H:AF=V:RETURN
 570   L=573:AG=H:AH=V:RETURN
 573   L=580:AI=H:  AJ=V:RETURN
 580   L=583:AK=H:AL=V:RETURN
 583   L=590:AM=H:AN=V:RETURN
 590   L=593:AO=H:AP=V:RETURN
 593   L=600:AQ=H:AR=V:RETURN
 600   L=603:AS=H:AT=V:RETURN
 603   L=650:AU=H:AV=V:RETURN
 650   GOTO 1010
 700   RETURN
 899   RETURN
 950   IF DD=0  AND  EE=0 AND FF=0 AND GG=0 THEN RETURN
 951   LI=LI+1
 952   ? LI;" PLOT  ";DD;",";EE;":DRAWTO  ";FF;",";GG:RETURN
1010   GRAPHICS 0
1012   ? :? :?
1015   DD=A:EE=B:FF=C:GG=D:GOSUB  950:DD=E:EE=F:FF=G:GG=I:GOSUB  950:DD=J:
       EE=K:FF=M:GG=N:GOSUB  950
1020   DD=O:EE=P:FF=Q:GG=R:GOSUB  950:DD=SS:EE=T:FF=U:GG=W:GOSUB  950:
       DD=X:EE=Z:FF=AA:GG=AB:GOSUB  950
1030   DD=AC:EE=AD:FF=AE:GG=AF:GOSUB  950:DD=AG:EE=AH:FF=AI:GG=AJ:  GOSUB
       950:DD=AK:EE=AL:FF=AM:GG=AN:GOSUB  950
1040   DD=AO:EE=AP:FF=AQ:GG=AR:GOSUB  950:DD=AS:EE=AT:FF=AU:GG=AV:  GOSUB  950:? "GOTO
       4000"
1100   POSITION 0, 0:POKE 842,13:END
2000   SOUND 0,17,10,10
2001   FOR X=1 TO 7:NEXT X
2002   SOUND 0,0,0,0
2003   RETURN
4000   A=0:B=0:C=0:D=0:E=0:F=0:G=0:I=0:J=0:K=0:L=0:M=0:N=0:O=0:P=0:Q=0:R=0
       :SS=0:T=0:V=0:W=0
4001   Y=0:Z=0:AA=0:AB=0:AC=0:AD=0:AE=0:AF=0:AG=0:AH=0:AI=0:AJ=0:AK=0:
       AL=0:AM=0:AN=0:AO=0:AP=0:POKE  842,12
4002   AQ=0:AR=0:AS=0:AT=0:AU=0:AV=0:LI=LI+1:MM=0:OK=0:GOTO  14
```

¡ME ENCANTA HACER DIBUJOS Y GRAFICOS! LO QUE ME DA UN POCO DE RABIA ES TENER QUE HACER TANTAS VECES EL MISMO DIBUJO.

¿Cómo? No te entiendo.

CLARO. SI TENGO UN DIBUJO Y LO QUIERO OCUPAR EN OTROS PROGRAMAS TENGO QUE VOLVER A DIBUJARLO. Y PARA ESO HAY QUE USAR UN MONTON DE INSTRUCCIONES: **POSITION**, **PRINT**, ETC.

Tengo un programa que creo te puede servir para eso.

¿EN SERIO? ¿ ES MUY COMPLICADO?

No, no mucho. Este programa permite hacer cualquier tipo de caracteres gráficos en una pantalla, con **GRAPHICS 0**.

¿Y COMO LOS USO DESPUES?

La pantalla, tal como queda finalmente, se graba en un diskette con números de línea que puedan agregarse a un programa principal por medio de la instrucción **ENTER**.
Es importante que sigas algunas instrucciones.

MIENTRAS NO SEAN DEMASIADAS .....

No te preocupes. Después de ejecutar el programa y de darle un nombre para grabar los resultados, no debes hacer nada para mover la pantalla.

¿QUE SIGNIFICA ESO?

Que lo que dibujas tiene que ser hecho con el cursor y no debes presionar nunca la tecla **RETURN**.
Es recomendable que la primera cosa que hagas sea borrar el mensaje que aparece al iniciar el programa. No borres la palabra CONT que aparecerá en la línea 22 de tu pantalla. Puedes usar las teclas de movimiento del cursor y cualquier otra función de edición del Atari.

¿LISTO?

Cuando termines, mueve el cursor a la línea que tiene el CONT y presiona la tecla RETURN.

¿Y QUE PASA AHI?

Se ejecutará el programa. Cuando la disketera termine de funcionar, el programa ha finalizado.

TODO CLARO. ¿VEAMOS LAS OPCIONES DEL PROGRAMA?

Este programa tiene muchas opciones. La única opción que no tienes es mover el CONT a la última línea.

¿POR QUE NO?

Porque si lo haces la pantalla se moverá (SCROLL) cuando presiones la tecla RETURN.

¿Y PUEDO HACER VARIAS PANTALLAS DIFERENTES?

Por supuesto. Solo tienes que ejecutar varias veces el programa, una para cada pantalla. Tienes que usar una especificación de archivo diferente y cambiar el valor de la línea 2020 (30000). Como se necesitan exactamente 40 líneas para grabar una pantalla, incrementa 30000 en bloques de 50 y así tendrás líneas disponibles para instrucciones RETURN.
Al poner juntos todos tus gráficos, simplemente haz un ENTER de todas las especificaciones de archivo que usaste y listalas bajo una sola nueva especificación.

¿Y COMO FUNCIONA EL PROGRAMA?

El programa simplemente construye un arreglo R$ para que sea igual a una línea de BASIC. Concatena un número de línea (3000 + I), un comando de impresión (el signo de interrogación) y luego agrega una comilla (con el CHR$(34)).

¿Y YO PUEDO PONER UNA COMILLA COMO CARACTER GRAFICO EN LA PANTALLA?

¡Buena pregunta! Lamentablemente la respuesta es NO, ya sabes, se puede confundir con la otra comilla.
Luego, usando la instrucción **LOCATE**, se examina cada posición en la pantalla y se agrega a R$ con la instrucción **CHR$**(x).

ME PARECE RARO QUE EN LA LINEA 2035 J COMIENCE CON UN VALOR IGUAL A 2, SIEMPRE ES 1.

Sí, pero recuerda que puedes inicializar un contador con el valor que tu quieras.

¡CIERTO! SI PUEDO IR DISMINUYENDO ESE VALOR.

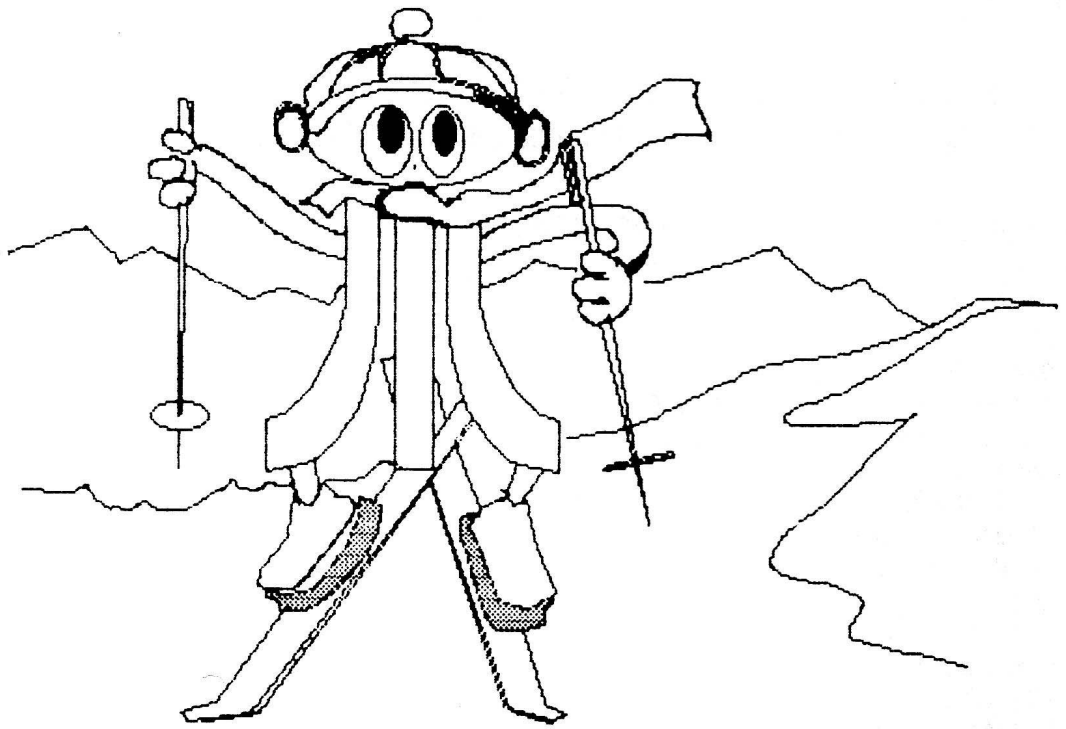
Bueno, la línea 2035 inicializa J con un 2 para que coincida con los valores por omisión en la pantalla.
Si tratas de usar las líneas resultantes con diferentes anchos de pantalla, entonces se cambia este valor para que coincida con ellos.

¿Y NUNCA CIERRAS LA COMILLA?

Claro que sí. R$ se completa con un CHR$(34) terminal, una coma para evitar un "scrolling" (CHR$(59)) o un retorno de carro (CHR$(155))

Mira. Aquí tienes el listado del Programa.

```
0     REM SAVE "D:ARCHIVA"
100   GRAPHICS 0
110   DIM  R$(80),F$(17),I$(10)
120   ? "INGRESA EL NOMBRE DEL ARCHIVO";
130    INPUT I$:F$="D:":F$(LEN(F$)+1)=I$
140   GRAPHICS 0
150   POSITION 4,22
160   ? "CONT       ":STOP
1900  OPEN #1,8,0,F$
2000  J=0
2010  FOR I=0  TO 23
2020  R$=STR$(30+I)
2025  R$(LEN(R$)+1)="b?"
2030  R$(LEN(R$)+1)=CHR$(34)
2035  FOR J=2 TO 39
2040  LOCATE  J,I,X
2045   R$(LEN(R$)+1)=CHR$(X)
2050  NEXT J
2055  R$(LEN(R$)+1)=CHR$(34)
2060  R$(LEN(R$)+1)=CHR$(59)
2065  R$(LEN(R$)+1)=CHR$(155)
2070  PRINT #1;R$
2075  R$="      "
2080  NEXT I
2090  CLOSE #1
2200  POSITION 0,22:?
```



CAPITULO 2

VARIABLES EN ATARI BASIC

ME DA LA IMPRESION QUE ESTE CAPITULO NO VA A SER MUY ENTRETENIDO. ¿SIGAMOS CON LOS GRAFICOS MEJOR?

Para muchos programadores, las variables no son tan entretenidas. Pero serás capaz de crear programas mucho más eficientes si entiendes como se almacenan y como pueden ser manipulados.

SUENA INTERESANTE. EXPLICAME UN POCO MAS.

Existen tres tipos de variables que pueden ser utilizadas por los programadores Atari.

¿TRES? ¿CUALES SON?

Los numéricos, los arreglos y los strings.

¡QUE NOMBRES TAN RAROS! BUENO, YA ME EXPLICASTE LOS STRINGS, PERO LOS OTROS PARECE QUE NO LOS CONOZCO.

Sí, si los conoces. Las variables numéricas comunes, representadas con un nombre como X, I, TOTAL, SUMA, etc.

AH! SON LAS VARIABLES QUE USO CON LA INSTRUCCION **LET**, QUE NO ES NECESARIO ESCRIBIR.

Exacto. El valor de una variable numerica se asigna dentro del programa.

ASI COMO X = 7, SUMA = A + B

y cada variable numérica ocupa 6 bytes de memoria.

¿Y CUAL ES LA DIFERENCIA CON LOS ARREGLOS?

Los arreglos son conjuntos de números representados por un nombre de variable seguido de un elemento numérico entre paréntesis.

UN EJEMPLO, POR FAVOR.

Por ejemplo, A(3), MES(7), etc. Antes de usar un arreglo, debes dimensionarlo con el tamaño máximo estimado.

¡LA INSTRUCCION **DIM**!.

Claro. Tienes que escribir por ejemplo, DIM MES(12).

¿Y CUANTA MEMORIA OCUPAN LOS ARREGLOS?

Reservan seis bytes de memoria por cada elemento del arreglo.

EN EL CASO DEL EJEMPLO SI TENGO 12 ELEMENTOS OCUPARIA 12 * 6 = 72 BYTES ¡QUE HARTO!

Por eso es muy importante que dimensiones muy bien tus arreglos.

YA SOLO NOS QUEDAN LOS STRINGS.

Los strings son variables extremadamente versátiles, de ahí su gran utilidad.

LO QUE RECUERDO BIEN ES QUE LOS STRINGS SE REPRESENTAN CON UN NOMBRE DE VARIABLE SEGUIDO DE UN SIGNO PESOS (COMO A$, BLANK$).

Otra característica importante es que también deben ser dimensionados.

¿Y PARA QUE SE DIMENSIONAN LOS STRINGS?

Para que el computador reserve memoria para el string. Cada caracter en una variable tipo string se almacena como un byte.
¿Sabes lo que es la Tabla de Nombres de Variable?

NO TENGO LA MENOR IDEA.

Es una lista de todos los nombres de variable que se han ingresado, almacenados en números de código ATASCII en la misma secuencia en la cual fueron encontrados dentro del programa.

¿Y QUE MAS INDICA ESTA TABLA?

Cada nombre de variable identifica el tipo de Variable. Las variables numéricas se almacenan con 128 agregado al valor ATASCII del último byte del nombre.

AHI SI QUE NO TE ENTENDI NADA.

Por ejemplo, el nombre de la variable HIT se describe en tres bytes con los valores 72, 73 y 212 (212 = 84 + 128).

UN POCO MEJOR. ¿Y LOS ARREGLOS?

Los arreglos se almacenan con un paréntesis a la izquierda y también se le suma 128 como último byte en el nombre.

SUPONGO QUE PARA EL ULTIMO BYTE EN LOS STRINGS TAMBIEN SE LE SUMA 128.

Sí, y se le agrega además el signo pesos. La dirección de memoria para el inicio de una tabla de nombres de variable puede determinarse con PEEKS en las localizaciones 130 (LO) y 131 (HI) o con NTAB = PEEK (130) + 256 * PEEK (131).

¿VAMOS A VER ESTO CON ALGUN PROGRAMA?

Ejecuta el programa siguiente para ver las entradas en la tabla de nombres de variables.

¿Y QUE HACE EL PROGRAMA ESPECIFICAMENTE?

Imprime el valor almacenado en los primeros 24 bytes de la tabla, un byte por cada caracter en cada nombre de variable. Chequea los resultados con el listado de los ATASCII que aparece en tu manual y no olvides que se le agregó 128 al último caracter.
El primer número impreso es 216, que es la primera variable (*) (88 en código ATASCII) más 128, ya que X es también el último caracter. La línea 50 imprime los nombres con el último caracter en inverso.

```
  0  REM SAVE "D:NOMVAR"
  1  REM PROGRAMA PARA IMPRIMIR TABLA DE
  2  REM NOMBRES DE VARIABLES
 10  X=12:Y=35:ZZ=12345:DIM  A(12),  HIT(4,9),SMALL$(35),BIG$(612)
 20  NTAB=PEEK(130)+256*PEEK(131)
 30  FOR BYTE=0 TO 23
 32  ? PEEK(NTAB+BYTE);",";
 34  NEXT BYTE
 40  ? :? :?
 50  FOR BYTE=0 TO 23
 52  ? CHR$(PEEK(NTAB+BYTE));
 54  NEXT BYTE
```

Para que te quede más claro, escribe y ejecuta el siguiente programa.

¿Y HACE LO MISMO QUE EL ANTERIOR?

No. Con este programa podrás ver el número de la Variable, el nombre y 8 bytes de data para cada variable dentro del programa.

PERO LA LINEA 10 ES IGUAL A LA DEL PROGRAMA ANTERIOR.

Si, esa línea contiene variables de ejemplo para experimentar.

ME GUSTARIA AGREGARSELO A MIS PROGRAMAS PARA QUE ME LISTE LAS VARIABLES.

Lo puedes modificar y te sirve para tus programas. Revisa el siguiente listado.

```
0    REM SAVE "D:VALVAR"
1    REM  PROGRAMA PARA IMPRIMIR LA
2    REM  TABLA DE VALORES DE VARIABLES
5    GRAPHICS 0
10   X=12:Y=35:ZZ=12345:DIM    A(12),HIT(4,9),SMALL$(35),BIG$(612)
15   REM BUSCA LA DIRECCION INICIAL DEL NOMBRE DE LA TABLA
20   NTAB=PEEK(130)+256*PEEK(131)
25   REM BUSCA LA DIRECCION INICIAL DEL VALOR DE LA TABLA
27   VTAB=PEEK(134)+256*PEEK(135)
30   ? "VAR  NOMVAR   DATA VTAB":?
35   I=0
40   FOR VARNUM=0 TO 12
45   ?  "#";VARNUM; "    ";
50   POSITION 9,VARNUM+2
55   FOR BYTE=0 TO 100
60   VARCHR=NTAB+BYTE
65   IF PEEK(VARCHR)>128 THEN ? CHR$(PEEK(VARCHR)-128),:NTAB= VARCHR+1:POP :GOTO 80
70   ? CHR$(PEEK(VARCHR));
```

```
75  NEXT BYTE
80  POSITION 17,VARNUM+2
85  FOR BYTE=0 TO 7
90  ? PEEK(VTAB+VARNUM*8+BYTE);
95  IF BYTE<7 THEN ? ",";
100  NEXT BYTE
105  ?
110  I=I+1
115  NEXT VARNUM
```

ES BIEN ELEVADO EL NIVEL DE TODO LO QUE HEMOS VISTO. ¿TE ENOJAS SI TE PREGUNTO QUE PODEMOS HACER CON TODO ESTO?

No, por el contrario. Una primera aplicación es realizar PEEKS sobre la tabla y determinar el estado de las distintas Variables.

DE VERAS, ASI PUEDO VERIFICAR QUE EL PROGRAMA ESTE HACIENDO LO QUE QUIERO QUE HAGA.

Como ya te dije, los strings son muy versátiles. Ya sabes que los strings son una serie de números ATASCII y cada número tiene un valor entre 0 y 255 y ocupa un byte en la memoria.

SI, LO RECUERDO. UN STRING DE MIL CARACTERES OCUPA MIL BYTES ADYACENTES EN MEMORIA.

Y esto sugiere una aplicación interesante. Una manera útil de reservar una cantidad determinada de bytes de memoria, por ejemplo 1000, es dimensionar el string.

COMO A$ (1000)

Puedes manejar tu propia tabla de valores dentro del string indexando cada 10 o 100 direcciones. La dirección inicial de tu string puede encontrarse fácilmente con la función ADR(A$).

¿QUE HACE ESA FUNCION?

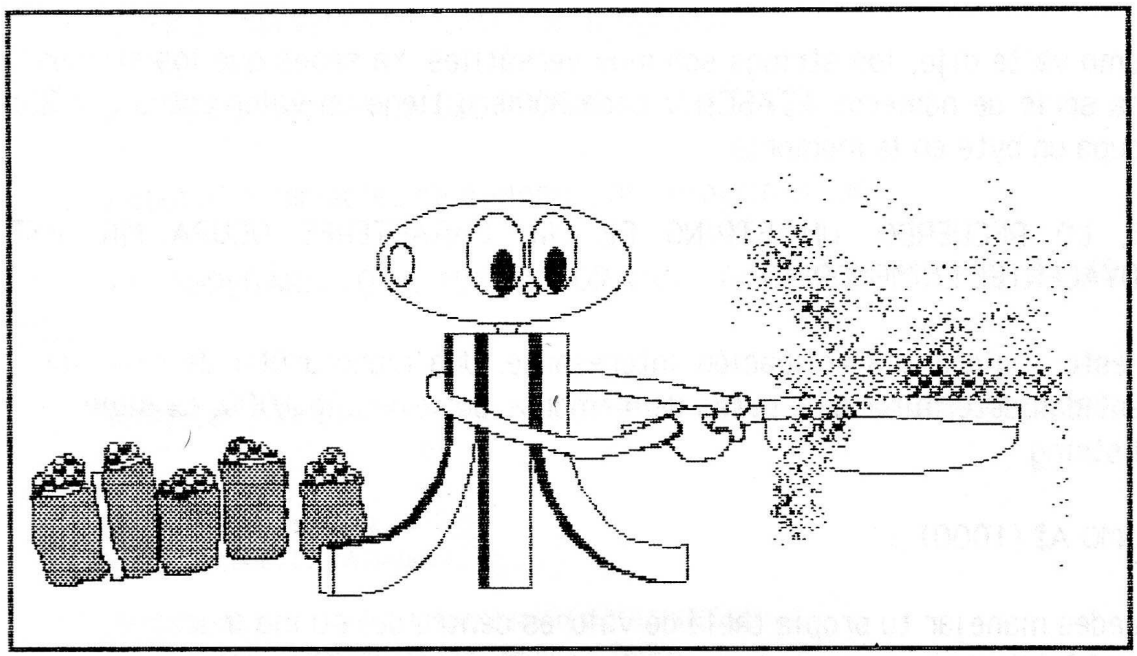
La instrucción **?PEEK** (ADR(A$)+99) te entregará el valor del centésimo byte en A$. El uso de los strings de esta manera es una forma común de almacenar datos enteros (menores que 256) y subrutinas en lenguaje de máquina.
Ahora si que veremos algo sumamente interesante.

¿EN SERIO? ¡YA ERA HORA! ¿QUE ES?

¡Como que ya era hora!. A lo mejor la primera vez que leas estas cosas no te parecerán demasiado interesantes, pero no te darás ni cuenta y ya estarás usando estas ideas.

MENOS PALABRAS Y MAS ACCION. ¿DE QUE SE TRATA?

Bueno, vamos a seguir con nuestro tercer capítulo con el que entenderás mucho mejor los programas anteriores, ya verás.



CAPITULO 3

PEEKS y POKES en el ATARI

Si eres un programador Basic que recién empiezas, seguramente aún no te haz dado cuenta de las capacidades adicionales que tienes al usar Basic en tu Atari.

¿POR QUE? ¿ACASO ES DIFERENTE AL BASIC DE OTROS EQUIPOS?

Así es y más que eso, no es el Basic sino el Atari. De hecho, el Atari Basic utiliza algunas capacidades que tiene Atari y no otros equipos..

¿EN SERIO?

Sí, es en serio. Por ejemplo, puedes redefinir el conjunto de caracteres de tal forma, que la letra A aparezca como un marciano invasor.

¡AH! ASI SE DEBEN HACER LOS JUEGOS.

Correcto. Si estás programando, puedes hacer que el programador tenga el completo control sobre la pantalla de gráficos. Y como éstas, muchas otras características adicionales de tu Atari.

PERO A LO MEJOR NO PUEDO USAR TODAS ESTAS COSAS ESPECIALES CON BASIC.

Es posible usar la gran mayoría en Basic. Algunas requieren que se trabaje en lenguaje de máquina.

...Y UN PRINCIPIANTE COMO YO NO TIENE IDEA DE UTILIZARLAS.

Sin embargo, para utilizarlas directamente, sólo necesitas dos palabras especiales en Basic: PEEK y POKE.

¿PEEK y POKE? ¿QUE SIGNIFICAN?

Ya lo descubrirás. Primero vamos a hacer un pequeño repaso.

Veamos: ¿Sabes cuál es el procesador de tu Atari?

BUENO. EL PROCESADOR, ES EL CEREBRO DE MI ATARI.

Correcto, y el modelo del procesador es el 6502. ¿Y cuántos kilobytes puede direccionar?

¿DIRECCIONAR?

Si, cuántos kilobytes puede manejar. Es decir, en que lugares puede guardar información y de cuales otros, puede sacar información.

¡AH! SUPONGO QUE 64 Kb, POR LO MENOS EL MANUAL DICE QUE EL EQUIPO TIENE 64 Kb DE MEMORIA RAM.

Casi, casi. Puede direccionar directamente 65.536 ubicaciones de memoria en RAM y, además tiene otra memoria en ROM.

¿Y PARA QUE DIRECCIONA LA RAM?

Para poder grabar allí los datos y programas.

¿Y LA ROM?

Como en la ROM se guarda el Basic y el sistema operativo, necesito poder accesarlos y para eso, tengo que direccionarla.
Además, hay una tercera clase de memoria.

¡NO PUEDE SER! LOS COMPUTADORES SOLO TIENEN MEMORIA RAM Y ROM.

Tienes razón. En realidad, no es memoria propiamente tal. Se trata de chips, utilizados para las operaciones de Entrada y Salida.

MEJOR ME EXPLICAS MAS CLARAMENTE, PORQUE NO ENTIENDO NADA.

Empecemos de cero. ¿Sabes qué es un chip?

SI. SON LOS CIRCUITOS INTEGRADOS.

Ya. ¿Y una operación de Entrada y Salida?

SI. SE REFIERE A GRABAR O RECUPERAR INFORMACION, DESDE O HACIA EL COMPUTADOR.

Bueno, resulta que tu Atari, tiene algunos chips de entrada y salida con nombres bien especiales. Ellos son:

- **GTIA** , un chip gráfico
- **ANTIC**, un chip que direcciona el GTIA para producir modos gráficos
- **POKEY**, un chip que lee el teclado y maneja las operaciones de entrada y salida, usadas para comunicarse con disketteras e impresoras.

¿Y COMO FUNCIONAN ESTOS CHIPS?

Estos chips necesitan información, como por ejemplo, qué caracteres poner en la pantalla y, pueden producir información, por ejemplo: saber que tecla se está presionando.
Para hacer las cosas más simples, tú manejas estos chips como localizaciones de memoria.

AH! POR ESO DIJISTE QUE ERA UN TIPO DE MEMORIA.

Correcto. Al principio se definen distintas direcciones, en las cuales se les puede informar algo a estos chips.

COMO CARTAS EN EL CORREO.

Así es. En este caso, lo haces con las instrucciones PEEK y POKE.
POKE, sirve para modificar la acción de un chip y PEEK, entrega información desde el chip.
Desde un punto de vista computacional, éstas son localizaciones de memoria. En efecto, algunas se comportan como la RAM.

¿DE QUE FORMA?

Por ejemplo, puedes colocar (POKE) un número en una localización, y luego

leer (PEEK) la localización para obtener el número de vuelta.

O SEA, QUE YO GUARDO UN NUMERO EN CUALQUIER LOCALIZACION DE MEMORIA Y LUEGO, DOY LA LOCALIZACION Y ME APARECE EL NUMERO.

La mayoría de las direcciones de Entrada/Salida, son sólo de lectura o sólo de escritura. La ROM, es sólo de lectura.
Las localizaciones de memoria sólo de escritura, pueden ser cambiadas.

¡AH! TANTAS COSAS.

Así es tu Atari, muy completo. Te lo explico: si tú escribes (POKE) en una dirección, sucede una cosa; mientras que si lees (PEEK), sucede otra.

A VER, RESUMAMOS. POKE SE USA PARA..., PARA...

POKE, se usa para cambiar el contenido de la memoria. El formato es:

POKE dirección, dato

¿CUALES PUEDEN SER LAS DIRECCIONES?

La dirección es un número entre 0 y 65535. Cada número accesa una celda de memoria, que puede contener un número entre 0 y 255.

LAS CELDAS EN LA MEMORIA, SON COMO LAS CASILLAS DEL CORREO. ¿VERDAD?

Sí, así se representan esquemáticamente. Más técnicamente, se puede pensar en una celda de memoria como un grupo de 8 interruptores.

¿INTERRUPTORES?

¡Claro! un interruptor puede tomar 2 valores.

¡ENCENDIDO Y APAGADO!

Correcto. Y si decimos "encendido=1" y "apagado=0", cada interruptor puede tener sólo dos valores: 0 y 1.

AH! ME ACORDE! ESTAS HABLANDO DE LOS NUMEROS BINARIOS.

Muy bien. Entonces, tenemos este grupo de 8 interruptores con dos estados cada uno, y con los cuales se puede representar distintos números.
¿Cuál es el 0 en binario?

BUENO, ES IGUAL QUE EN LOS NUMEROS NUESTROS: 0

Es decir, todos los interruptores deben estar en 0. ¿Y el 255?

SUPONGO QUE SERAN OCHO UNOS.

Así es. 1 1 1 1 1 1 1 1 = 255

ME DISTE EL FORMATO DEL POKE, PERO DEL POBRE PEEK NUNCA MAS SE SUPO.

PEEK es el inverso de POKE, pero es más una función que una instrucción.

¿POR QUE?

Porque PEEK entrega como resultado un valor, y cualquier instrucción que use un valor, puede usar un PEEK.

POR EJEMPLO...

Veamos la instrucción **PRINT**. Tú puedes decir PRINT 4, lo que te muestra un 4. PRINT TOTAL, te muestra el contenido de la variable TOTAL y PRINT PEEK (53076), te muestra el valor almacenado en la localización 53076.

ENTONCES YO PODRIA USAR UN PEEK DENTRO DE UN POKE.

Correcto. Está bien si escribes POKE 106, PEEK (30865).

CUANDO MENCIONASTE LA INSTRUCCION PEEK, DIJISTE ALGO DE QUE SE PODIA REDEFINIR EL CONJUNTO DE CARACTERES DE MI ATARI.

Así es. Veamos ¿cómo lees un carácter desde el teclado, usando BASIC?

CON LA INSTRUCCION **GET**

Correcto. Pero que espera GET.

NO ENTIENDO TU PREGUNTA.

Qué tienes que hacer, para que el computador sepa la tecla que presionaste.

NO SE. BUENO, YO PRESIONO UNA TECLA Y EL RESTO LO HACE EL COMPUTADOR.

Eso es el computador espera que presiones una tecla. Si no presionas ninguna, el programa se detendrá hasta que presiones alguna.

¡ES VERDAD! NO ME HABIA DADO CUENTA

En vez de usar GET, puedes chequear la localización 764 con PEEK (764).

¿POR QUE LA 764?

Porque en esa localización, el computador guarda la tecla que presionaste.

¿Y QUE PASA SI NO PRESIONO NADA?

Si PEEK (764) = 255, significa que no se ha presionado ninguna tecla. Si el valor que te entrega es diferente a 255, puedes usar la instrucción GET para saber que tecla fue presionada.

YA ME PARECE MAS UTIL LA INSTRUCION PEEK.

A ver, ¿y qué tienes que hacer antes de usar la instrucción GET?

AL PRINCIPIO DEL PROGRAMA, DEBO ABRIR UN ARCHIVO QUE DIRECCIONE EL TECLADO, LO HAGO CON LA INSTRUCCION OPEN.

Bueno, si tú sólo quieres esperar que una tecla sea presionada, puedes usar algo como:

```
10 PRINT "PRESIONE CUALQUIER TECLA PARA CONTINUAR
20 IF PEEK (764) = 255 THEN 20
30 POKE 764, 255
```

¿Y QUE PASA CON EL POKE DE LA INSTRUCCION 30?

Te entrega un número, que es la representación de la tecla expresado en términos de la fila y columna, en las cuales está la tecla.
Ejecuta este pequeño programa, para ver que valores te entrega.

```
10 PRINT PEEK (764) : GOTO 10
```

¿Y COMO VEO LOS VALORES?

Cuando presionas una tecla, te muestra el valor.

APARECEN PUROS 255

Así es. Observa que mientras no presiones una tecla, te aparecerá el 255.

¡ES VERDAD! CUANDO PRESIONO OTRAS TECLAS, APARECEN OTROS NUMEROS. LO QUE NO TENGO CLARO, ES EL POKE 764, 255.

Poke 764, 255 elimina cualquier valor almacenado en esa localización.

¿Y PUEDO TENER POKE 764 CON OTROS VALORES?

Por supuesto. Y así puedes ver como esta instrucción hace que el computador escriba un carácter, automáticamente.

AH, YA. Y AHORA SUPONGAMOS QUE YO QUIERO SABER SI LA TECLA SE PRESIONO UNA SOLA VEZ O SE MANTIENE PRESIONADA.

Mira, este pequeño programa que encontrarás a continuación, simula la acción de un órgano. Cuando presionas una tecla, emite un sonido tanto tiempo como la tecla sigue presionada.

¿Y NO PARA?

Sí, para cuando dejas de presionar la tecla.

```
10 IF PEEK (53775) = 255 THEN SOUND 0,0,0,0 : GOTO 10
20 SOUND 0, 100, 10, 8 : GOTO 10
```

¿QUE TIENE LA LOCALIZACION 53775?

La localización 53775 tiene el valor 255, si no se ha presionado ninguna tecla.

¿Y QUE OTROS VALORES PUEDE TENER?

Un 251 si una tecla se mantiene presionada y un 247, si la tecla SHIFT se está presionando.

Ya está bueno, ahora aprendamos otras cosas. ¿Conoces la tecla INVERSE?

¡UF! ¡NO LA VOY A CONOCER! ES SUMAMENTE FACIL PRESIONARLA POR EQUIVOCACION, EN LOS MODELOS 600 Y 800.

Bueno, si tú estás aceptando o ingresando información a través del teclado, seguramente no quieres que el usuario ingrese caracteres en inverso o en minúsculas.

SI. UNA VEZ HICE UN PROGRAMA PARA COMPARAR UNOS NOMBRES, Y NO SABIA PORQUE NO ME RESULTABA. AL FINAL, DESCUBRI QUE ESTABA COMPARANDO UNA MINUSCULA CON UNA MAYUSCULA.

Claro que también querrás, que a veces el usuario presione la tecla INVERSE, de tal forma que todo lo que el ingrese, quede en modo inverso.

SI, PUEDE SER.

Pues bien, la localización 694 controla ésto.

¿COMO?

Tienes que hacerle un POKE con 128 para forzar los caracteres inversos y con 0 para deshabilitarlos.

¡ESTO ME PUEDE SERVIR PARA LOS LISTADOS Y PANTALLAS!

No, no te equivoques. Esta localización no afectará la forma en que se imprime el texto, sólo la forma en que se recibe la información del teclado.

YA, TE PROMETO QUE NO SE ME OLVIDA. ¿Y EL 694, TAMBIEN CONTROLA LAS MAYUSCULAS/MINUSCULAS?

La localización 702, se preocupa de las mayúsculas y minúsculas, almacenando el estado de la tecla CAPS/LOCK. Si la tecla CAPS está presionada, es decir mayúsculas, la localización 702 guarda un 64.

¿Y CUANDO ESTA EN MINUSCULAS, CUAL ES EL VALOR DE LA LOCALIZACION 702?

Un 0. Además, puede tomar el valor 128 si está en modo CTRL.

USEMOS EL POKE PARA PROBAR TODO ESTO.

Puedes usar POKE para obligar al teclado a que esté en un modo determinado. Prueba ésto.

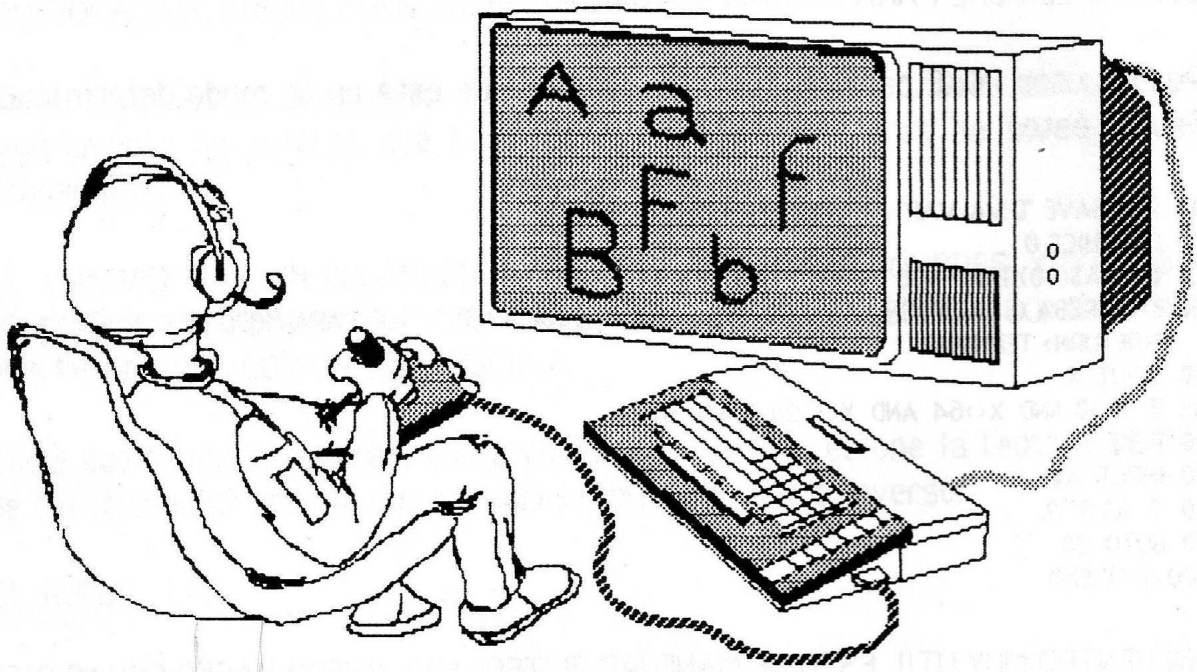
```
0  REM SAVE "D:VALVAR"
5  GRAPHICS 0
10 DIM A$(10):TRAP 100
20 ? "INGRESA 0 PARA MINUSCULAS"? "64 PARA MAYUSCULAS":? "128 PARA MODO DE CONTROL":?
   "<RETURN> TERMINA"
30 INPUT X
35 IF X<>0 AND X<>64 AND X<>128 THEN 30
40 POKE 702,X
50 INPUT A$
60 ? A$:? :?
70 GOTO 20
100 ? "]":END
```

ENCUENTRO MUY UTIL ESTO DE MANEJAR EL TECLADO, PUEDO HACER COSAS BIEN INTERESANTES.

Hay algunas teclas con las que las cosas son un poco diferentes. Las teclas START, OPTION y SELECT, no se pueden leer como otras teclas del teclado.

¿COMO LAS LEO?

Puedes leerlas fácilmente, revisando la localización 53279. Aquí tienes una lista de los posibles valores que te entrega:


A a
B f
f
b

7 no se ha presionado ninguna tecla
6 sólo la tecla START
5 sólo la tecla SELECT
3 sólo la tecla OPTION
0 START, SELECT y OPTION simultáneamente
1 OPTION y SELECT juntas
2 OPTION y START juntas
4 SELECT y START juntas

QUE TAL SI ME HACES UN PROGRAMA PARA DEMOSTRAR LO ANTERIOR.

Aquí lo tienes.

```
0   REM SAVE "D:STARTOP"
5   GRAPHICS 0
10  ? "PRESIONA ALGUNA TECLA DE FUNCIONES"
15  A=PEEK(53279)
20  IF  PEEK(53279)=A THEN  20
30  ON  A+1 GOSUB  50,60,70,80,90,100,110
40  GOTO 15
50  ? "START+SELECT+OPTION":RETURN
60  ? "OPTION+SELECT+OPTION":RETURN
70  ? "OPTION+START+OPTION":RETURN
80  ? "OPTION":RETURN
90  ? "SELECT+START":RETURN
100 ? "SELECT":RETURN
110 ? "START":RETURN
```

¿Recuerdas que algunas localizaciones tienen diferentes funciones, cuando se lee o se escribe en ellas?

SI, ALGUNA VEZ LO MENCIONASTE.

La localización 53279 es una de ellas.

¡COMO SI TUVIERA DOS PERSONALIDADES!

Más o menos. Cuando la lees, te indica que teclas de la consola están siendo presionadas en forma continua.

¿Y CUANDO PONGO ALGUN VALOR EN ESA LOCALIZACION?

Si le haces un POKE 53279,0 se genera un suave "click".

¿POR QUE?

El cero hace que el parlante suene, pero el sistema operativo se encarga de hacerlo, poniendo un 8 en la localización 53279 cada 1/60 de segundo. Haciendo varios POKE 53279,0 en esta localización, creas un ruido como un zumbido o alarma.

EN REALIDAD NO TIENE NADA QUE VER EL EFECTO DEL POKE CON EL DEL PEEK SOBRE ESTA LOCALIZACION.

Y para que veas lo útil que son estas instrucciones, te voy a enseñar como anular la tecla BREAK para evitar que te listen tus programas BASIC.

AH! QUE INTERESANTE. ASI MI PROPIEDAD INTELECTUAL SE MANTENDRA INTACTA. NADIE PODRA COPIAR MIS PROGRAMAS.

Simplemente usa estas dos instrucciones POKE:

```
POKE 16,64
POKE 53774, 64
```

¿Y COMO PUEDO VOLVER A UTILIZAR LA TECLA BREAK?

La tecla BREAK se rehabilita, cambiándote a modo gráfico o presionando System Reset.

O SEA, QUE SI QUIERO QUE LA TECLA BREAK PERMANEZCA ANULADA, ¿TENGO QUE PONER LAS 2 INSTRUCCIONES POKE DESPUES DE CADA INSTRUCCION GRAFICA?

Correcto. Así es, y también después de cada instrucción OPEN.

¿Y PUEDO ANULAR EL SYSTEM RESET?

Aunque la tecla SYSTEM RESET no puede anularse, puedes hacer algo similar. Si haces POKE 580 con un valor distinto de 0, la tecla SYSTEM RESET actuará como si hubieses apagado y prendido el equipo. Esto se conoce como "Partida en frío".

¿PARTIDA EN FRIO?

Claro, igual que en un auto, sin calentar el motor. La "partida sobre caliente", es la que se ejecuta generalmente con esta tecla, cuando el computador ya ha estado funcionando.

NO LE VEO MUCHA UTILIDAD, PERO COMO MUCHAS OTRAS COSAS, ALGUN DIA PUEDE SERVIRME.

¡Qué bueno que pienses así! Ahora sigamos viendo aplicaciones de POKE. Revisemos su uso en las instrucciones de gráficos.

¡ME FASCINAN LOS GRAFICOS!

Puedes usar **POKE** directamente sobre los registros de color para seleccionarlos, aunque es más fácil con la instrucción **SETCOLOR**.

ALGUNA VENTAJA TENDRA...

POKE es más rápido y compacto, ya que sólo debe evaluar un número en vez de cuatro.

¿SOLO UNO?

Así es. Las localizaciones 708-712, corresponden directamente desde SETCOLOR 0 a SETCOLOR 4. Cada localización, guarda tanto el color como la luminosidad. Simplemente, multiplica el número del color (0-15) por 16 y súmale la luminosidad (0-15).

COMO QUEDA ENTONCES.

SETCOLOR a, b, c corresponde a POKE 708 + a, b * 16 + c.
Y eso no es todo, en cuánto a gráficos. La localización 559, puede, entre otras cosas, encender o apagar la pantalla.

¿Y PARA QUE ME PUEDE SERVIR?

Para hacer más rápidos los programas. Como toma algún tiempo desplegar la pantalla, el Atari puede correr un 30% más rápido con la pantalla apagada.

AH! ENTONCES PODRIA APAGAR LA PANTALLA CUANDO TENGA QUE HACER UN CALCULO LARGO.

Sí, pero si es otra persona la que va a usar el programa, debes advertirle para que no se asuste.

SI, POR SUPUESTO. TAMBIEN PODRIA APAGAR LA PANTALLA CUANDO TENGA QUE HACER ALGUN GRAFICO COMPLICADO Y PRENDERLA, PARA HACER QUE EL GRAFICO APAREZCA INSTANTANEAMENTE.

Y esta maravilla la logras con POKE 559, 0.

¿Y PARA VOLVER A NORMAL?

Con POKE 559, 34.
Y seguramente, haz escuchado de las localizaciones 82 y 83.

LA VERDAD ES QUE NUNCA.

Bueno, estas localizaciones se usan principalmente para ajustar el ancho de la pantalla, ya que algunos televisores no pueden desplegar el ancho total de la pantalla.

¿Y QUE SE HACE CON LA LOCALIZACION 82 y 83?

La localización 82, controla el margen izquierdo.

¿QUE ES EL MARGEN IZQUIERDO?

El margen izquierdo, es el número de espacios en blanco desde el borde de la pantalla.
PEEK, te muestra el valor que tiene y POKE, le asigna el valor que quieres.

¿CUAL ES EL ANCHO MAXIMO DE LA PANTALLA?

40 columnas, y eso lo logras con POKE 82, 0.

LA LOCALIZACION 93, SEGURO QUE ES PARA EL MARGEN DERECHO.

Así es. El margen derecho es un número entre 0 y 39, y representa el número de espacios desde el lado izquierdo de la pantalla.
Después que cambies los márgenes, las instrucciones **PRINT** siguientes se ajustan a los nuevos márgenes.
No hagas nunca el margen izquierdo más grande que el derecho.

¿POR QUE NO?

¡Pruébalo y te darás cuenta! Además, ten presente que si dejas el ancho de la pantalla muy pequeño, no podrás tipear ningún comando.

AH! PERO ALGUNA FORMA HABRA DE VOLVER A LA PANTALLA NORMAL.

En cualquier caso, al presionar SYSTEM RESET, el valor del margen izquierdo vuelve a 2 y el derecho a 39.

SABES LO QUE TE QUERIA PREGUNTAR. ES SOBRE EL CURSOR, NO ME GUSTA QUE ESTE SIEMPRE EN LA PANTALLA.

Es sumamente fácil deshabilitarlo. Simplemente, haz un POKE 752, 1.

¿Y HABILITARLO?

Con un 0 en la misma localización. Después que hagas un POKE a esta localización, el cursor no cambiará hasta que la próxima instrucción PRINTlo mueva o hasta que limpies la pantalla.

¿Y ESTA VEZ AFECTAN LOS GRAFICOS?

Cualquier cambio en los modos gráficos re-almacena el cursor, lo mismo que **SYSTEM RESET**.

¡EL FAMOSO SYSTEM RESET!

Además puedes controlar la forma en que aparecen los caracteres inversos.

¿EN SERIO?

Sí. Con un número 2 en la localización 755 los caracteres aparecen inversos.

¿COMO?

Cada carácter, es decir letra, número o signo, se compone de una serie de puntos. En el estado normal, todos estos puntos aparecen en el color opuesto al del fondo de la pantalla y en el inverso.....

¡YO SE! HE VISTO ALGUNAS LETRAS QUE APARECEN EN INVERSO. POR EJEMPLO, SI LA PANTALLA ES NEGRA, APARECE UN CUADRADO VERDE Y DENTRO DE ESE CUADRADO, LA LETRA EN NEGRO ¡SE VE BONITO!

Un número 0 en la localización 755, devuelve los caracteres a su estado normal y adivina, que pasa con un 1.

¿UN 1? NO SE. PARA MI QUE NADA, YO SOLO CONOZCO DOS ESTADOS: INVERSO Y NORMAL.

No, hay muchos más. Un 1 los hace invisibles. Un 3 hace que todos los caracteres inversos aparezcan como espacios inversos (opacos).

¡TANTAS COSAS!

Y más. Súmale 4 a cualquiera de los valores anteriores y todo el texto aparecerá como si lo estuvieras proyectando en 1 espejo.

¿EN SERIO? SERIA IGUAL QUE EN LOS JUEGOS DE VIDEO.

Además, como el cursor también se puede considerar como un carácter, la localización 755, también afecta la apariencia del cursor. Prueba este programa cortito, para ver como puedes usar el 755, para que el texto parpadee:

```
100 PRINT "LOS CARACTERES INVERSOS PUEDEN PARPADEAR"
110 POKE 755,2 - PEEK (755)
120 FOR W = 1 TO 50 : NEXT W
130 GOTO 110
```

¿Y COMO SE DONDE ESTA EL CURSOR?

Puedes leer fácilmente la posición del cursor, verificando las localizaciones 84 y 85. La localización 84 guarda la fila.

AH! LA POSICION HORIZONTAL

Correcto, y la localización 85 la columna, o posición vertical.

...O SEA QUE EN LA POSICION 84 PUEDE HABER UN NUMERO DEL 0 AL 23 Y EN LA 85 UN NUMERO DEL 0 AL 39.

Y puedes usar la instrucción **POSITION**, para mover directamente el cursor a una posición determinada. Con **POKE**, puedes cambiar la fila o columna en forma separada.

O SEA QUE PUEDO MOVER EL CURSOR COMO YO QUIERA: ARRIBA, ABAJO, AL LADO, ETC.

Así es, pero el cursor no se mueve hasta que no uses una instrucción Print.

HMM! ¿SABES LO QUE SE ME OCURRE? LA POSICION 85, LA DE LA COLUMNA, SE PUEDE USAR COMO TABULADOR, PORQUE EL TECLADO NO TIENE Y EL BASIC TAMPOCO.

¡Qué buena idea! Así puedes hacer formatos de pantalla en forma mucho más fácil, por ejemplo la línea:

```
10 Z = Z + 6 : PRINT TAB (20-19*SIN (Z) ) ; CHR$ (42) : GOTO 10,
```

escribe una curva sinusoidal en Basic Microsoft.

¿Y SE PUEDE HACER EN ATARI BASIC?

Por supuesto:

```
10 Z = Z + 6 : POKE 85, 20 - 19 * (SIN (Z) ) : PRINT CHR$ (42) : GOTO 10
```

¡CASI IGUAL! YO PENSE QUE ERA MAS LARGO.

Si estamos trabajando en un Atari, no en cualquier cosa.
En modo gráfico, las localizaciones 84 y 85 controlan la posición del cursor gráfico, no el cursor de texto.

¿Y COMO CAMBIO EL CURSOR DE TEXTO EN LOS MODOS GRAFICOS?

Con las localizaciones 656 (fila) y 657 (columna). También puedes cambiar la localización 201, que guarda el número de espacios entre las comas.

¿DE QUE ME HABLAS?

Mira: cuando tú imprimes una lista de variables (como PRINT A,B,C$) cada vez se tabula dentro de una zona separada de 10 espacios de ancho. Si lo que estás imprimiendo sobrepasa la zona que tiene asignada, se pone en la siguiente y todos los datos que siguen se corren.

AH, YA VOY ENTENDIENDO.

Puedes cambiar el ancho de la zona haciendo un POKE 201, ancho.
Nunca pongas un 0 en esa localización porque el computador queda dando vueltas y más vueltas, forzándote a presionar SYSTEM RESET o cortar el poder del equipo para que puedas retomar el control.

PERO DEBE SER UTIL, CUANDO SON NUMEROS PEQUEÑOS, PARA QUE NO QUEDEN MUY SEPARADOS.

Sí, para eso se usa, y ahora, pasando a otra cosa ¿suena tu diskettera cuando lees o grabas?

¡TERRIBLE! PARECE QUE ESTUVIERAN CORTANDO CARNE O RALLANDO ALGO. CLARO QUE SIRVE PARA CERCIORARSE DE QUE ESTA LEYENDO O GRABANDO.

Hay personas que pueden distinguir el ruido que hace al escribir del que hace al leer, e incluso puden decir si hubo algún error.

COMO ALGUNOS MEDICOS. LO QUE ES YO, DE REPENTE PIENSO QUE ME VOY A VOLVER LOCO CON TANTOS RUIDOS.

La solución: POKE 65,0 inhabilita el parlante, de modo que no se note ningún ruido.

¡SENSACIONAL! PERO ¿Y SI SUBO EL VOLUMEN?

Aún así, claro que en ese caso escucharás un leve sonido.

PERO IMAGINATE NO SENTIR LA DISKETTERA, NI LAS TECLAS...

¡Un momento! Este POKE no tiene ningún efecto sobre las teclas y tampoco sobre la instrucción **SOUND**.

NO PODRIA SER CIERTO TANTA MARAVILLA. MEJOR QUE SUENE TODO.

Para eso, haces un POKE 65 con cualquier valor distinto de cero.
No te he preguntado ¿qué modelo es tu Atari?

YO TENGO EL 600XL Y EN EL COLEGIO USAMOS UN 800XL, SON BIEN PARECIDOS.

Así es. El Atari 600XL y el 800XL usan el Sistema Operativo XL, que tiene considerable poder y flexibilidad.

¿Y QUE SIGNIFICA ESO?
En pocas palabras, que tienes muchos más POKE'S para probar. Recuerda, eso sí, que ninguno de estos POKES funcionará en los computadores Atari 400 y 800 corrientes. (No pertenecientes a la línea XL).

A VER, CUALES SON ESOS NUEVOS POKE

El POKE más novedoso, permite un movimiento de líneas (scrolling) en modo gráfico 0.
Simplemente ingresa `POKE 622,255 : GRAPHICS 0`

ME TINCA QUE PASA MUY RAPIDO. LE VOY A AGREGAR:

```
FOR I = 1 TO 1000 : PRINT I : NEXT I
```

PARA FIJARME BIEN.

A diferencia del movimiento normal, que mueve una línea de texto a la vez, el scrolling va moviendo los caracteres en forma más suave.

UN AMIGO MIO LO HIZO EL OTRO DIA ¡ES BIEN BONITO EL EFECTO! CLARO QUE DESPUES ME IBA A MOSTRAR EL PROGRAMA Y TUVO PROBLEMAS CON EL EQUIPO.

Lo que pasa, es que este movimiento afecta algunos programas. Para inhabilitarlo, escribe POKE 622,0 : GRAPHICS 0

¿SOLO SIRVE CON MODO GRAFICO 0?

Sí, así es.
Los modelos Atari XL, tienen todos una tecla de ayuda (HELP). Aunque no la usa ni el sistema operativo ni el Basic, puedes leer esta tecla en tus propios programas y actuar sobre ella.

VEAMOS, VEAMOS. ESTO SE PONE INTERESANTE.

Una vez que se presiona la tecla HELP, la localización 732 guarda un 17 y continuará con este valor hasta que hagas un POKE 732,0. Debes verificar que la localización 732 tenga un valor distinto de 0, y luego haces POKE 732,0 una vez que has actuado sobre la tecla.
Si se presiona SHIFT junto con HELP, la localización 732 tendrá un 81, y un 145, indica que se ha presionado CTRL-HELP.

DIME UNA COSA. EN ESTOS MODELOS QUE SON MAS AVANZADOS ¿SE PUEDE

SUPRIMIR EL RUIDO DE LAS TECLAS? ALGUNAS PERSONAS LO ENCUENTRAN UTIL PARA LA FAMOSA "DIGITACION AL TACTO", PERO YO, FRANCAMENTE, LO ENCUENTRO ABURRIDOR.

En realidad, en los modelos 600 y 800, la única forma es cortando los alambres que van al parlante, pero con los modelos XL puedes hacerlo. Simplemente, con POKE 731,255. Un valor 0 te permite seguir escuchando el click de las teclas.
Otra cosa interesante es la posibilidad de presionar una tecla y que esta se repita. Puedes cambiar el lapso de tiempo que transcurre entre una y otra letra.

¿QUE ES ESO POR FAVOR?

Existen dos factores en la repetición de las teclas, cuando presionas una tecla, no quieres que se repita en forma instantánea.

SI, ASI ES.

Por eso, el Sistema Operativo espera 4/5 de segundo antes de comenzar a repetir; y una vez que comienza la repetición, el otro factor de tiempo es la velocidad con la que la tecla se repite, que es alrededor de 10 veces por segundo.
En el sistema operativo, estos tiempos de espera se expresan en múltiplos de 1/60 por segundo. 1 valor de 60 es un segundo completo, 30 es medio segundo, y así sucesivamente.

AH, LO QUE TU ME QUERIAS DECIR ES QUE SE PUEDEN CAMBIAR ESOS LAPSOS DE ESPERA.

Correcto. Para cambiar el tiempo, antes que la tecla se empiece a repetir, se usa POKE 729, y para especificar el tiempo entre repeticiones POKE 730.

Y CUALES SON LOS VALORES ESTANDAR.

Para el 729, un 48 y para el 730 un 6.

¡VEAMOS UN PROGRAMA! ¡ME ENCANTAN LOS PROGRAMAS! ¡QUE TE CUESTA!

Está bien, está bien. Haremos un programa bien completo, que te hará fácil mirar dentro de la memoria de tu Atari. También podrás cambiar la memoria y hasta convertir números decimales, hexadecimales y binarios.
Pero antes, es importante que aclaremos algunos conceptos:

- el contenido de todas las direcciones de memoria, está formado de un byte que corresponde a...

¡8 BITS!

Correcto. Estos 8 bits o dígitos binarios, sólo pueden contener números binarios.

SI, SOLO 1's y 0's (Unos y ceros).

Y el 1 indica que el bit está "encendido" (ON) y el 0 que el bit está "apagado" (OFF). Los bits se numeran del 0 al 7, de derecha a izquierda.
Tu Atari sólo puede entender lenguaje binario.

¿Y CON LOS CARTRIDGES?

Cuando insertas un cartridge de lenguaje en tu equipo, le estás proporcionando al Atari un interprete que te permite comunicarte con él, a través de un lenguaje como el BASIC, en alguna forma útil y significativa.

¡YA, VAMOS AL PROGRAMA!

Más adelante, encontrarás el listado del programa que te mencioné. Escríbelo y guarda una copia antes de usarlo.

Cuando ejecutas el programa, te aparecerá un menú con nueve opciones.

¿NUEVE? ¡QUE HARTAS!

Así es. La opción 1 te permite examinar cualquier localización de memoria y ver su contenido en hexadecimal, decimal y binario.
La opción 2 te permite cambiar la memoria al usar esta opción. Si haces un POKE de un número equivocado en una localización equivocada, corres el riesgo de echar a perder tu sistema.

¡HUY, QUE SUSTO! PERO HABRA ALGUNA SOLUCION SI ME EQUIVOCO.

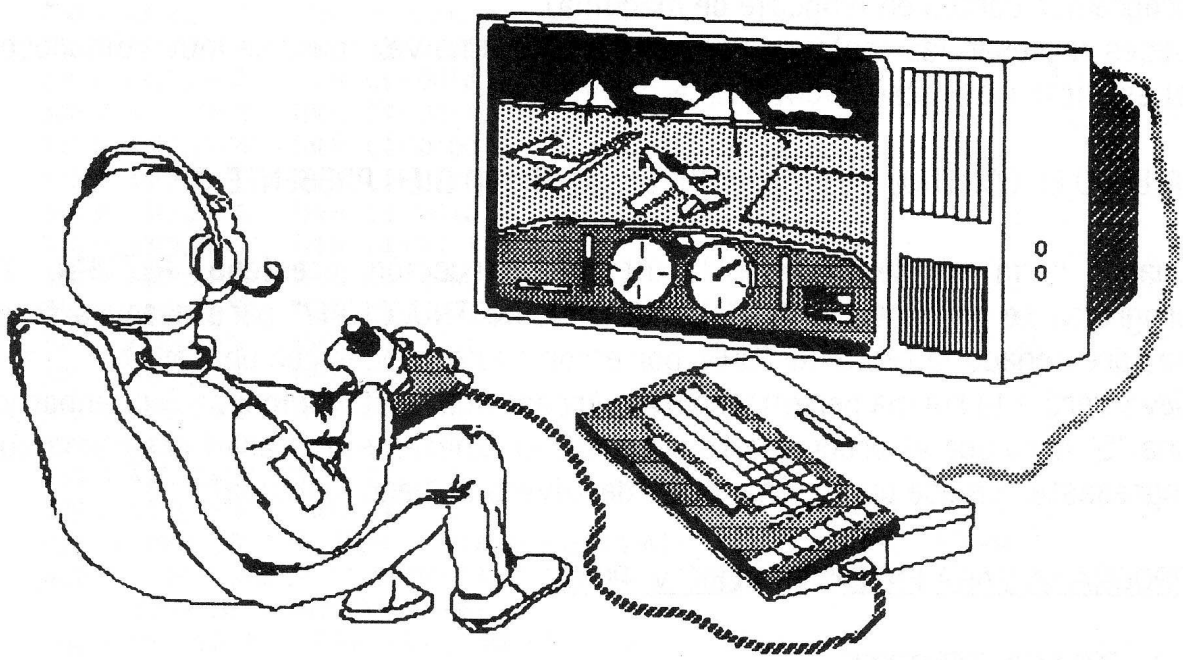
Sí, tendrás que apagar y encender el equipo para recobrar el control.

A VER, DAME EJEMPLOS DE LAS LOCALIZACIONES DONDE PUEDA HACER CAMBIOS.

Algunos lugares interesantes, donde hacer cambios, son las localizaciones 53760-53768 (registros de sonido), 53266-53274 (los registros de color para los juegos con gráficos), y 53248-53255 (la posición de los registros gráficos para los juegos).

Puede que aparezcan extrañas cosas en la pantalla cuando pruebes con estos registros, pero presionando SYSTEM RESET te sacará de cualquier lío en el que te hayas metido.

BUENO, Y QUE PASA CON EL RESTO DE LAS OPCIONES.



SOLO FLIGHT

Las opciones 3 a 8, son rutinas de conversión. Oberva que si estás ingresando un número hexadecimal para convertir, siempre debes ingresar dos dígitos, aunque el primero sea un 0.

¿Y SI SE ME OLVIDA?

Tendrás una respuesta errónea.
La opción 9, te permite poner un Programa en lenguaje de máquina en la memoria.

Puedes hacerlo en forma hexadecimal o decimal y puedes especificar la dirección de comienzo (1536 es generalmente un buen punto de comienzo para programas cortos en lenguaje de máquina).
Debes ingresar las instrucciones una a una. Una vez más, se muy cuidadosos al escribir las instrucciones, un error y ...

¡PIERDO EL CONTROL DE MI ATARI! SI, LO TENGO BIEN PRESENTE.

Cuando termines de ingresar la última instrucción, presiona RETURN. El programa te preguntará "¿ES LA ULTIMA INSTRUCCION?" para asegurarte si haz presionado la tecla RETURN por error Si respondes con una "N", te devolverá a la rutina permitiéndote ingresar más instrucciones. Sin embargo, una "S" hará que el programa te pregunte si quieres ejecutar el programa que ingresaste. En ese punto, una "N" te devolverá al Menú Principal

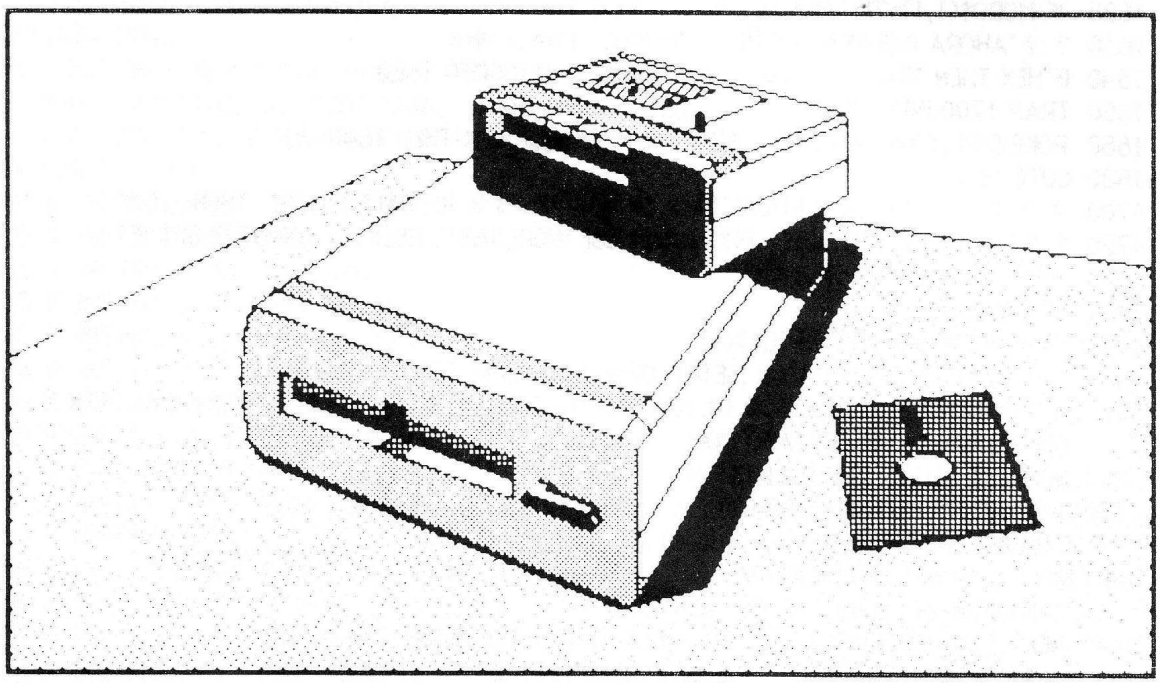
PROGRAMA PARA ENTENDER PEEK y POKE

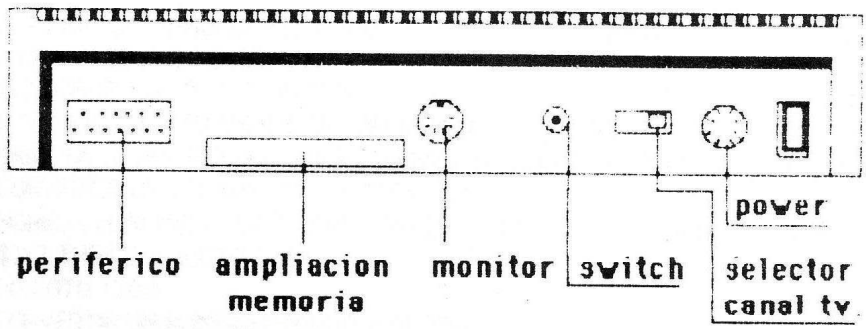
```
0  REM SAVE "D:PEEKPOKE"
5  DIMA$(2),AD$(1),B$(4),C$(5),RESP$(1),BIN$(8),MODO$(3),INST$(6),TD(2),
   N$(8),TITULO$(20),LINEA$(22),NOMBRE$  (22)
6  MENU=600:CLICK=6000:B$="0000"
7  GOTO 5000
39 REM SUBRUTINA DE CONVERSION DE DECIMAL A HEXADECIMAL
40  N=PEEK(DIRECCION):N1=PEEK(DIRECCION)
60 I=2.08
70 TEMP=N
71  N=INT(N/16):TEMP=TEMP-N*16
72 IF TEMP<10 THEN A$(I,I)=STR$(TEMP):GOTO 90
80  A$(I,I)=CHR$(TEMP-10+ASC("A"))
90 IF N<>0 THEN I=I-1: GOTO 70
91 IF M=3 OR M=8 THEN ? "HEX=";A$(I,2):A$="[,]":B$="    ":C$="    ":RETURN
95 IF M>4 THEN RETURN
```

```
100 REM SUBRUTINA DE CONVERSION DE HEXADECIMAL A BINARIO.
110 IF A$(1,1)="[,]" OR A$(1,1)="0" THEN B$="0000"
120 IF A$(1,1)="1" THEN B$="0001"
130 IF A$(1,1)="2" THEN B$="0010"
140 IF A$(1,1)="3" THEN B$="0011"
150 IF A$(1,1)="4" THEN B$="0100"
160 IF A$(1,1)="5" THEN B$="0101"
170 IF A$(1,1)="6" THEN B$="0110"
180 IF A$(1,1)="7" THEN B$="0111"
190 IF A$(1,1)="8" THEN B$="1000"
200 IF A$(1,1)="9" THEN B$="1001"
210 IF A$(1,1)="A" THEN B$="1010"
220 IF A$(1,1)="B" THEN B$="1011"
230 IF A$(1,1)="C" THEN B$="1100"
240 IF A$(1,1)="D" THEN B$="1101"
250 IF A$(1,1)="E" THEN B$="1110"
260 IF A$(1,1)="F" THEN B$="1111"
270 IF A$(2,2)="0" THEN C$="0000"
280 IF A$(2,2)="1" THEN C$="0001"
290 IF A$(2,2)="2" THEN C$="0010"
300 IF A$(2,2)="3" THEN C$="0011"
310 IF A$(2,2)="4" THEN C$="0100"
320 IF A$(2,2)="5" THEN C$="0101"
330 IF A$(2,2)="6" THEN C$="0110"
340 IF A$(2,2)="7" THEN C$="0111"
350 IF A$(2,2)="8" THEN C$="1000"
360 IF A$(2,2)="9" THEN C$="1001"
370 IF A$(2,2)="A" THEN C$="1010"
380 IF A$(2,2)="B" THEN C$="1011"
390 IF A$(2,2)="C" THEN C$="1100"
400 IF A$(2,2)="D" THEN C$="1101"
410 IF A$(2,2)="E" THEN C$="1110"
420 IF A$(2,2)="F" THEN C$="1111"
421 IF M=4 OR M=7 THEN ? "BINARIO=";B$;C$:A$="[,]":B$="  ":C$=" ":RETURN
422 ? "  HEX:";A$(1,2):? "DECIMAL:";N1:? " BINARIO: ";B$:C$:? "PEEK(";
    DIRECCION;")=";PEEK(DIRECCION)
440 IF M=1 OR M=2 THEN A$="[,]":B$ = "  ":C$="  ":RETURN
450 IF M=3 THEN ? "HEX=";A$(1,2):? :A$="[,]":RETURN
460 IF M=4 THEN ? "BINARIO=";B$;C$:B$="  ":C$="  ":RETURN
499 REM SUBRUTINA DE CONVERSION DE BINARIO A HEXADECIMAL
500 TRAP MENU
501  B7=VAL(BIN$(1,1)):B6=VAL(BIN$(2,2)):B5=VAL(BIN$(3,3)):B4=VAL(BIN$ (4,4))
505 IF BIN$="00000000" THEN PKR=0: GOTO 526
510  B3=VAL(BIN$(5,5)):B2=VAL(BIN$(6,6)):B1=VAL(BIN$ (7,7)):B0=VAL (BIN$(8,8))
520 PKR=INT(B7*2^7+B6*2^6+B5*2^5+B4*16+B3*8+B2*4+B1*2+B0)
525 IF B7 AND NOT B6 AND NOT B5 AND NOT B4 AND NOT B3 AND NOT B2 AND NOT B1 AND
    NOT B0 THEN PKR=PKR-1
526 IF M=6 THEN ? "DECIMAL=";PKR:RETURN
527 IF M=8 THEN RETURN
530 POKE DIRECCION,PKR:RETURN
599 REM MENU PRINCIPAL
600  ? CHR$(125):POKE 752,1:POSITION 16,4:? "MENU":POSITION 16,5:? "----":?
```

```
610 POSITION 6,8:? "1 REVISAR CONT.DE MEMORIA":POSITION 6,9:? "2 CAMBIAR CONT.DE MEMORIA"
612 POSITION 6,10:? "3 CONV. NRO. DECIMAL A HEXADEC.":POSITION 6,11:? "4 CONV. NRO. DECIMAL
    A BINARIO"
615 POSITION 6,12:? "5 CONV. NRO. HEXADEC. A DECIMAL":POSITION 6,13:? "6 CONV. NRO. BINARIO A
    DECIMAL"
616 POSITION 6,14:? "7 CONV. NRO. HEXADEC. A BINARIO":POSITION 6,15:? "8 CONV. NRO. BINARIO A
    HEXADEC."
617 POSITION 6,16:? "9 INGRESAR POKES SUCESIVOS"
618 POSITION 5,17:? "10 TERMINA"
620 TRAP MENU
630 POSITION 6,22:? "ELIJA OPCION ";:INPUT M:? "}":ON M GOTO
    1000,1050,1100,1100,1200,1300,  1400,1500,1600,6010
699 REM SUBRUTINA DE CONVERSION DE HEXADECIMAL A DECIMAL
700 FOR Q=1 TO 2
701 IF INST$(Q,Q)="0" THEN TD(Q)=0
705 IF INST$(Q,Q)="1" THEN TD(Q)=1
710 IF INST$(Q,Q)="2" THEN TD(Q)=2
715 IF INST$(Q,Q)="3" THEN TD(Q)=3
720 IF INST$(Q,Q)="4" THEN TD(Q)=4
725 IF INST$(Q,Q)="5" THEN TD(Q)=5
730 IF INST$(Q,Q)="6" THEN TD(Q)=6
735 IF INST$(Q,Q)="7" THEN TD(Q)=7
740 IF INST$(Q,Q)="8" THEN TD(Q)=8
745 IF INST$(Q,Q)="9" THEN TD(Q)=9
750 IF INST$(Q,Q)="A" THEN TD(Q)=10
755 IF INST$(Q,Q)="B" THEN TD(Q)=11
756 IF INST$(Q,Q)="C" THEN TD(Q)=12
757 IF INST$(Q,Q)="D" THEN TD(Q)=13
758 IF INST$(Q,Q)="E" THEN TD(Q)=14
759 IF INST$(Q,Q)="F" THEN TD(Q)=15
760 NEXT Q:INST=TD(1)*16+TD(2):IF M=5  THEN ? "DECIMAL=";INST
770 RETURN
999 REM "1" REVISAR LOS CONTENIDOS DE MEMORIA
1000 ? :? :?" ¿QUE DIRECCION QUIERES VER?":INPUT DIRECCION:GOSUB 40: GOTO 1000
1040 TRAP 40000
1049 REM  "2" CAMBIAR LOS CONTENIDOS DE MEMORIA
1050 ? :? "POR FAVOR INGRESA LA DIRECCION A LA CUAL DESEAS  CAMBIAR EL CONTENIDO";:INPUT
     DIRECCION
1060  ? :? "AHORA INGRESA EL NUMERO EN BINARIO QUE DESEAS PONER EN        EL REGISTRO(LOS
     BITS SE NUMERAN DE 7 A 0 DESDE LA I"
1070  INPUT BIN$:GOSUB 500:? :? "PEEK(";DIRECCION;")=";PEEK(DIRECCION):? :? :GOTO 1050
1099  REM "3" Y "4"  CONVIERTEN UN NUMERO DECIMAL A HEXADECIMAL O A BINARIO"
1100  ? :? :? "INGRESA EL NUMERO QUE QUIERE CONVERTIR";:INPUT N:GOSUB 60:GOTO 1100
1199  REM "5" CONVERTIR UN HEXADECIMAL A DECIMAL"
1200  ? :? :? "INGRESA EL NUMERO QUE QUIERE CONVERTIR(2 DIGITOS)";:  INPUT INST$:IF
     LEN(INST$)>2 THEN ? "}":GOTO 1200
1210  GOSUB 700:GOTO 1200
1299  REM "6" CONVERTIR UN NUMERO BINARIO A DECIMAL
1300  ? :? :? "INGRESA EL NUMERO QUE QUIERES CONVERTIR";:INPUT BIN$:  GOSUB 500:GOTO 1300
1399  REM "7" CONVERTIR UN HEXADECIMAL A BINARIO
1400  ? :? :? "INGRESA EL NUMERO QUE QUIERES CONVERTIR(2 DIGITOS)";:  INPUT A$:GOSUB
     110:GOTO 1400
```

```
1499 REM "8" CONVERTIR UN BINARIO A HEXADECIMAL
1500 ? :? :? "INGRESA EL NUMERO QUE QUIERAS CONVERTIR";:INPUT BIN$: GOSUB 500:N=PKR:GOSUB
     60:GOTO 1500
1599 REM "9" INGRESAR UNA SERIE DE POKES EN LOCALIZACIONES DE MEMORIAS SUCESIVAS
1600 ? :? :? "INGRESAR INSTRUCCIONES EN FORMA DECIMAL(D) O HEXADECIMAL(H)";:INPUT MODO$
1610 ? :? "CUAL ES LA DIRECCION INICIAL(EN DECIMAL, POR FAVOR)";: INPUT
     DIRECCION:DIRECCION1=DIRECCION
1620 IF MODO$(1,1)="H" THEN HEX=1
1625 IF MODO$(1,1)="D" THEN HEX=0
1630 ? :? "AHORA INGRESA LAS INSTRUCCIONES UNA A UNA"
1640 IF HEX THEN TRAP 1700:INPUT INST$:GOSUB 700:GOTO 1660
1650 TRAP 1700:INPUT INST
1660 POKE DIRECCION,INST:DIRECCION=DIRECCION+1:IF HEX THEN 1640
1680 GOTO 1650
1700 ? :? "ES LA ULTIMA INSTRUCCION?";:INPUT RESP$:IF RESP$(1,1)<>"S" THEN 1640
1720 ? :? "DESEAS EJECUTAR EL PROGRAMA QUE INGRESASTE RECIEN";: INPUT RESP$:IF
     RESP$(1,1)="S" THEN 1750
1740 GOTO MENU
1750 TRAP MENU:X=USR(DIRECCION1)
4999 REM ***** TITULO INICIAL,SETCOLOR=FONDO NEGRO LETRAS BLANCAS
5000 SETCOLOR 2,0,0:SETCOLOR 1,10,10:? CHR$(125):POKE 752,1:TITULO$= "JUEGA AL PEEK Y
     POKE":LINEA$="PRESENTADO POR"
5001 NOMBRE$="EDICIONES COMPUGRAFICA"
5370 LN=INT(LEN(TITULO$)/2):FOR Z=1 TO LN:POSITION 17-Z,7:? "*";
     TITULO$(LN-Z+1,LN+Z);"*":GOSUB CLICK
5400 NEXT Z:LN=INT(LEN(LINEA$)/2):FOR Z=1 TO LN:POSITION 17-Z,9 :? "*";
     LINEA$(LN-Z+1,LN+Z);"*":GOSUB CLICK
5440 NEXT Z:LN=INT(LEN(NOMBRE$)/2):FOR Z=1 TO LN:POSITION 17-Z,11:? "*";
     NOMBRE$(LN-Z+1,LN+Z); "*":GOSUB CLICK:NEXT Z
5500 FOR ESPERA=1 TO 500:NEXT ESPERA:GOTO MENU
6000 FOR TICK=0 TO 3:POKE 53279,0:NEXT TICK:RETURN
6010 POKE 752,0:GRAPHICS 0:END
```




power
periferico
ampliacion
memoria
monitor
switch
selector
canal tv

CAPITULO 4

USO DE CASSETTERA Y DISKETTERA

AH! AQUI SI QUE TE VOY A COMER A PREGUNTAS HACE HARTO TIEMPO QUE TENGO DUDAS SOBRE LAS CASSETTERAS Y DISKETTERAS.

¿En serio? No me lo habías dicho

SE DICEN MUCHAS COSAS DEL ATARI, QUE PUEDE HACER ESTO Y LO OTRO PERO NO SE CUALES SON CIERTAS. POR EJEMPLO, 4 VECES HE TENIDO PROBLEMAS CON LOS ARCHIVOS QUE GRABO EN CASSETTES. ES TERRIBLE BUSCAR UN ARCHIVO EN UN CASSETTE A VECES SE TRANSFORMA EN UNA HAZAÑA

No exageres. Una forma fácil de recordar es escribir la descripción del archivo y su ubicación en un papelito.

Y SI SE TE PIERDE EL PAPELITO ¡SERIA TODO! Y NO ES DE LO MAS ENTRETENIDO CARGAR UN ARCHIVO TRAS OTRO HASTA LLEGAR AL APROPIADO

Verás que los programas que vienen más adelante son fantásticos. Los debes usar de la siguiente forma:
- Comienza cargando un cassette y poniendo en 0 el contador
- Avanza la cinta hasta que el contador sea 20.
- Graba hasta seis archivos en la cinta, cuidando de anotar el nombre (hasta 16 caracteres), la ubicación del comienzo del archivo y el comando que usaste para grabar el archivo.

¿Y QUE PASA DESPUES?

Cuando estés listo, rebobina la cinta y ejecuta el Programa 1. Te preguntara el nombre que quieres ponerle al cassette.

¿Y?

Y debes responderle con un nombre de hasta 16 caracteres. El programa luego te preguntará por los nombres, localizaciones y comandos que usaste para grabar los 6 archivos

Y YO LE RESPONDO CON EL NOMBRE, EL VALOR QUE TENIA EL CONTADOR AL EMPEZAR CADA ARCHIVO Y LA PRIMERA LETRA DEL COMANDO QUE USE PARA GRABAR CADA ARCHIVO.

¿Como? ¿Acaso conoces los programas?

NO, PERO ES LO TIPICO. SI A MI TAMBIEN SE ME OCURREN COSAS DE REPENTE. OYE .... ¿Y SI TENGO MENOS DE 6 ARCHIVOS?

Si tienes menos de 6 archivos, responde al mensaje que te pide el nombre del último archivo presionando dos veces ESC y luego RETURN. Asi el programa termina de preguntarte y almacena lo que se podria llamar "el archivo del sistema de cinta" en el cassette.

SUPONGO QUE ESE ARCHIVO CONTIENE LA INFORMACION QUE RECIEN INGRESE.

Así es, toda la información necesaria para localizar y cargar todos los archivos queda grabada en forma segura en el mismo cassette.

AHORA SIGAMOS CON EL PROGRAMA 2.

El Programa 2 lee este archivo del sistema, escribe un menú en la pantalla y te pide que selecciones el número del archivo que quieres cargar.

Y NO ME DIGAS QUE CARGA ESE ARCHIVO.

No seas flojo, algo de trabajo para tí también. El programa te indica el número del contador en el que encontrarás el archivo. Avanza la cinta y simplemente presiona RETURN para cargar la cinta.

¿Y SI ME EQUIVOCO?

El programa te da otra oportunidad para cargar el archivo.

ME GUSTARIA QUE ME EXPLICARAS UN POCO MAS EN QUE CONSISTEN ESTOS PROGRAMAS.

La clave de estos programas reside en entender el manejo de strings del Atari BASIC.

AH! ENTONCES NO TENDRE NINGUN PROBLEMA. YA, SIGAMOS.

Como sabes, un string es una secuencia de 1 ó más caracteres. En el Programa 1, que encontrarás más adelante, la instrucción 40 es una instrucción de dimensionamiento (DIM) define las variables tipo string que se usan en el programa.
CBUF$ es el string que se pondrá en el buffer del cassette y luego se escribe en la cinta.

¿QUE COSA ES ESE BUFFER DEL CASSETTE?

Es un área de memoria RAM desde la cual Atari graba a la cassettera.

BUENO, ¿Y EL PROGRAMA 1 USA MAS STRINGS?

Si, varios más. Los usa para guardar los inputs (ingresos) desde el teclado.

TE REFIERES A LAS RESPUESTAS QUE UNO ESCRIBE EN EL TECLADO ¿VERDAD?

Así es. Como los nombres de archivo, las localizaciones y comandos se ingresan a CBUF$ uno tras otro. S se usa como una medida para guardar espacio.

NO ME IMAGINO COMO VAN QUEDANDO TANTAS COSAS EN CBUF$.

Para agregar caracteres a CBUF$, como ocurre en la linea 110, se usa la siguiente expresión:

```
CBUF$ (principio, fin) = TN$
```

¿QUE ES PRINCIPIO Y FIN?

principio y fin son las posiciones de inicio y término en el string CBUF$.

¿Y TN$?

TN$ contiene el string que tiene que ubicarse en CBUF$ en las posiciones indicadas por principio y fin.

AH. YA ENTENDI. MANEJANDO LAS POSICIONES DE INICIO Y FIN DE LOS DATOS DENTRO DE CBUF$, SE LLENA EL STRING CON LOS DATOS, UNO CADA VEZ.

Muy bien. Con esta explicación y tu Manual de Referencia del Atari serás capaz de descifrar el resto del programa.

¿DESCIFRAR? ESO SUENA COMO QUE EL PROGRAMA ES MUY DIFICIL.

No, en absoluto. Verás lo fácil que es. Ah! antes que se me olvide, para apresurar la carga de estos programas, puedes hacerlos más pequeños.

Y ASI HAY MENOS QUE CARGAR. DIME, ¿QUE PUEDO SUPRIMIR?

Puedes suprimir las instrucciones **REM**, sustituir las variables por constantes, como se explica más abajo, poniendo dos líneas de código en una sola, separadas por dos puntos y sustituyendo la palabra PRINT por un signo de interrogación. (?).

¡LISTO! ¿ME DAS EL PROGRAMA?

PROGRAMA 1

```
  0   REM SAVE "D:ARCHCAS
 10   REM **PROGRAMA QUE CREA UN ARCHIVO DE DIRECTORIO
 15   REM **EN CASSETTE
 20   ? CHR$(125)
 30   REM **DEFINICION DE VARIABLES Y CONSTANTES
 40   DIM   CBUF$(128),FN$(12),CT$(3),TN$(16),SM$(1):S=16
 50   REM **COMPLETA EL BUFFER DEL CASSETTE CON BLANCOS
 60   FOR N=1 TO 128:CBUF$(N,N)=" ":NEXT N
 70   REM **INGRESO DEL NOMBRE DEL VOLUMEN DEL CASSETTE
 80   ? "INGRESA EL NOMBRE DEL CASSETTE(MAXIMO 16 CARACTERES)"
 90   INPUT TN$
100   REM **PONE EL NOMBRE DEL CASSETTE EN EL BUFFER
110   CBUF$(1,S)=TN$
120   REM **INGRESA INFORMACION DE HASTA 6 ARCHIVOS
130   FOR N=1 TO 6
140   ? "INGRESE NOMBRE DEL ARCHIVO ";N
150   INPUT FN$
160   IF FN$(1,1)=CHR$(27) THEN GOTO 270:REM **CUANDO SE PRESIONA LA TECLA ESCAPE SE
      TERMINA EL INGRESO
180   CBUF$(N*S+1,N*S+13)=FN$
190   ? 'LECTURA DEL CONTADOR PARA ESTE ARCHIVO(3 DIGITOS)"
200   INPUT CT$:CBUF$(N*S+13,N*S+15) = CT$
210   REM **FORMA DE ALMACENAR EL ARCHIVO
220   ? "QUE COMANDO SE USO PARA GRABAR?"
```

```
230   ? "CSAVE/SAVE C/LIST C-(C,S,L)"
240   INPUT SM$
250   CBUF$(N*S+S,N*S+S)=SM$
260   NEXT N
270   REM **GRABA EL ARCHIVO DIRECTORIO EN EL CASSETTE
280   ? "PRESIONA PLAY/RECORD-RETURN "
290   OPEN #1,8,0, "C:"
300   ? "GRABANDO EL DIRECTORIO "
310   ? #1;CBUF$
320   CLOSE #1
```

PROGRAMA 2

```
 10   REM **DESPLIEGA UN MENU PARA LEER LOS ARCHIVOS
 20   ? "}"
 30   REM **DEFINICION DE VARIABLES Y CONSTANTES
 40   DIM CBUF$(128), FN$(12), CT$(3), TN$(16), SM$(1):S=16
 50   REM **OPEN CASSETTE
 60   ? "PRESIONA PLAY Y LUEGO RETURN"
 70   OPEN #1,4,0,"C:"
 80   REM **LLENA EL BUFFER DEL CASSETTE CON BLANCOS
 90   FOR N=1 TO 128:CBUF$(N,N)="":NEXT N:REM **LEE E IMPRIME EN LA PANTALLA EL NOMBRE
      DEL CASSETTE
100   ? CHR$(125);"LEYENDO EL DIRECTORIO DEL CASSETTE"
110   INPUT #1;CBUF$:? "}"
120   ? CBUF$(1,15): ?
130   REM **IMPRIME EL MENU DE ARCHIVOS EN LA PANTALLA
140   FOR N=1 TO 6
150   FN$=CBUF$(N*S+1,N*S+13)
160   CT$=CBUF$(N*S+13,N*S+15)
170   SM$=CBUF$(N*S+S,N*S+S)
180   ? N;" ";FN$;" "; CT$;" ";SM$
190   ?
200   NEXT N
210   REM **PREGUNTA POR EL PROGRAMA QUE SE QUIERE CARGAR
220   ? :? "INGRESA EL NUMERO DEL PROGRAMA QUE QUIERES EJECUTAR "
230   INPUT N
240   FN$=CBUF$(N*S+1,N*S+13)
250   CT$=CBUF$(N*S+13,N*S+15)
260   ? "AVANZA LA CINTA HASTA EL ";CT$
270   TRAP 350
280   REM **DECIDE CON QUE COMANDO CARGARLO
290   IF CBUF$(N*S+S,N*S+S)="C" THEN 330
300   IF CBUF$(N*S+S,N*S+S)="E" THEN 340
310   REM **CARGA EL PROGRAMA
320   LOAD "C:":? "PRESIONA RETURN "
330   CLOAD :? "PRESIONA RETURN"
340   ENTER "C:"
350   ? "RETROCEDE LA CINTA AL ";CT$;
360   GOTO 290
```

¿Te gustaron?

CLARO QUE SI. ES MUY INTERESANTE ESTA MATERIA.

Ahora veremos algo parecido, siguiendo la misma idea de ahorrar el tiempo que se pierde buscando archivos o programas en diskettes y cassettes.

SI, A VECES UNO NO SABE EN QUE LUGAR GRABO UN PROGRAMA Y HAY QUE EMPEZAR A PROBAR CON TODOS.

En efecto, para muchos usuarios de computadores, una de las tareas que mas tiempo consume es buscar entre los diskettes en cual está un programa especifico. Veremos un programa que te da una posible solución.

.....HUM ¿SERA ALGO COMO UN INDICE?

Exacto!. ¿Acaso no sería fantástico tener un índice de todos tus diskettes, almacenado en otro diskette?

ISENSACIONAL! DE ESA FORMA NUNCA HABRIA QUE ESTAR CAMBIANDO DISCOS.

Y con algunas pocas facilidades, como capacidad de búsqueda e impresión, tendrías un utilitario verdaderamente "útil".

BUENO, PERO EN REALIDAD HAY HARTOS SISTEMAS DE CATALOGACION DE DISCOS.

El problema de la mayoría de ellos es que hacen mucho más de lo que realmente se necesita y eso hace que sean más complicados.

O SEA QUE ES UN PROGRAMA BIEN SIMPLE ENTONCES.

El programa "Utilitario de Catalogación de Discos" (UCD) es simplemente un directorio masivo, crea archivos separados de todos los directorios de tus discos, en un disco designado para ello.

¿Y QUE SIGNIFICA ESO?

Esto significa que puedes obtener un directorio de cualquier disco sin tener que recurrir al DOS y cambiar diskettes.

SI. EN REALIDAD ES UTIL.

Otras tres características hacen que UCD sea aún mas útil. Primero, tienes la opción de hacer un disco-directorio de todos tus discos, uno solo, o todos los discos en un cierto rango. Segundo, UCD utiliza totalmente lo que se conoce como "Comodín de Atari" (Atari's wild card)

¿COMODIN? ¿COMO EN LOS NAIPES?

En concepto, sí. Se trata de un símbolo que puedes poner en el nombre de un programa y te permite ver todos los programas que tienen el string especificado en su nombre.

COMO QUE TE ENREDASTE UN POCO CON LA EXPLICACION.

Más fácil un ejemplo. Si quieres ver todos los archivos que tengan la terminación. LST escribes *.LST y te muestra la lista de programas que tienen esa terminación.

ME HUBIERAS DICHO QUE SE TRATABA DEL ASTERISCO Y LISTO.

Bueno, el asterístico es el "comodín" más conocido pero tú puedes elegir uno particular cuando uses UCD.

¡QUE SIMPATICO!

Una especificación de archivo no tiene porque tener un comodín. Si tú sabes exactamente que programa o archivo estas buscando, escribe su nombre tal cual y UCD identifica todos los discos que tienen ese nombre.

YA. ¿Y CUAL ES LA TERCERA CARACTERISTICA?

Para aquellos que poseen impresora, UCD también entrega una "copia dura" del índice de los directorios.

HAY QUE SER BIEN ORDENADO CON LOS DISCOS PARA QUE RESULTE ESTE SISTEMA.

Así es. Primero debes numerar todos tus discos comenzando con el 1 y poner este número en su etiqueta.

YA, LO VOY A HACER EN SEGUIDA

Luego ejecuta el programa. Lo primero que te pide es que elijas el símbolo "comodín". Si presionas RETURN, UCD usará automáticamente el asterisco como comodín.

BUENO, Y CUANDO EMPIEZO A CATALOGAR LOS DISCOS?

En este punto, estas listo para hacerlo. Corre el UCD y escribe "C" para catalogar. El programa te pregunta que disco quieres catalogar.

A LO QUE DEBO RESPONDER CON EL NUMERO DEL DISKETTE.

luego RETURN y tienes que insertar el diskette correspondiente y volver a presionar RETURN. UCD leerá el directorio del disco. Luego inserta el disco de datos.

¿CUAL ES ESE?

Es el disco con el programa UCD y todos los archivos creados con el UCD.
Bueno, ibamos en que insertabas este disco, luego RETURN y UCD grabará el directorio del disco que habia leido antes. Lo graba como un nuevo archivo que se llamará " DISCO n" donde n corresponde al número del diskette que pusiste.

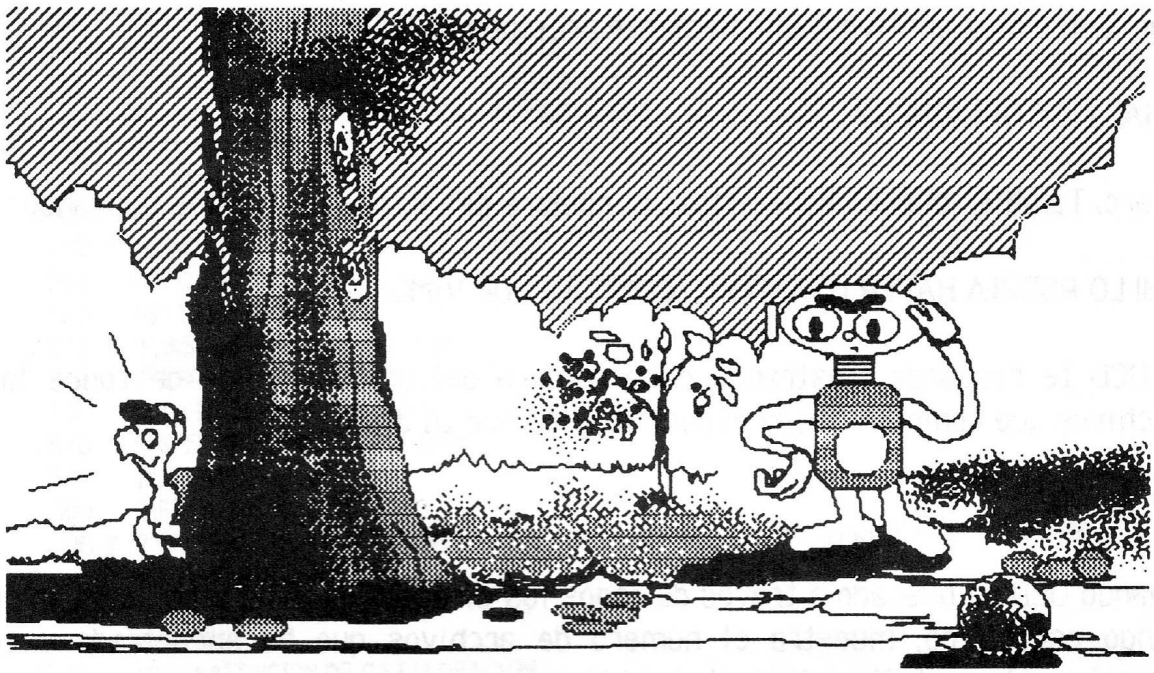
O SEA QUE SI PONGO EL DISKETTE Nº 1 GRABA UN ARCHIVO QUE SE LLAMARA DISCO1.

Por eso es que tienes que asegurarte de escribir el número correcto cuando estés catalogando.
Tienes que seguir este mismo procedimiento para todos tus diskettes.

Y DESPUES SUPONGO QUE PUEDO VER UN DIRECTORIO DE TODOS LOS DIRECTORIOS.

Para obtener un directorio completo, escribe "D" y te aparecerán varias opciones:

- si tienes una impresora y quieres un listado de tus directorios, escribe S cuando te aparezca la pregunta si es que quieres un listado.



SI NO TENGO IMPRESORA, LA UNICA OPCION QUE ME QUEDA ES UNA N ¿VERDAD?

Sí, pero tienes otras opciones. Por ejemplo, tienes la opción de buscar dentro de un rango determinado de discos. Escribe S si quieres un rango.

A VER SI TE ENTIENDO. SI ESCRIBO S Y LUEGO 3,8 EL PROGRAMA BUSCARA ENTRE LOS DIRECTORIOS 3, 4, 5, 6, 7 Y 8.

Exacto. Y si quieres que UCD comience con el disco 1 y continúe hasta que no encuentre más archivos, simplemente presiona RETURN.

¿HAY MAS OPCIONES?

Claro. También puedes especificar el nombre del archivo que estás buscando.

AH! LO PODRIA HACER USANDO EL COMODIN QUE VIMOS RECIEN.

Y UCD te responde mostrando cada número del disco seguido de todos los archivos que cumplen con la especificación que tu diste.

¿Y AL FINAL?

Cuando UCD ha buscado a través de todos los directorios, o ha terminado en un rango especifico, muestra el número de archivos que ha encontrado y te devuelve a la pantalla principal. Aqui tienes el programa.

UTILITARIO DE CATALOGACION DE DISKETTES. (UCD).

```
0   REM SAVE "D:UTCATDIS"
10  GRAPHICS 0:SETCOLOR 2,12,4:SETCOLOR 1,0,15
12  DIM  D(1000),ESPEC$(17),F$(17),FN$(15)
14  DSK=1
15  OPEN  #1,4,0,"K:"
17  ? "UTILIT.PARA CATALOGACION DE DISKETTES"
18  ? "SIMBOLO COMODIN?":GET #1,CO:IF CO=155 THEN CO=ASC("*")
19  FOR I=1 TO 38:? CHR$(CO);:NEXT I
20  CLOSE #2:? CHR$(125):? "UTIL. DE CATALOGACION DE DISKETTES[UCD] ":? :? "[C]
     CATALOGAR DISKETTE":?
22  ? "[D] DIRECTORIO COMPLETO"
40  POSITION 2,22:? "TU ELECCION=>";:GET #1,K
45  IF CHR$(K) <>"C" AND CHR$(K) <>"D" THEN ? CHR$(125);CHR$(253);"D o C":GOTO 40
50  IF CHR$(K)="C" THEN 3000
60  REM **FUNCION DE DIRECTORIO
65  ?  CHR$(125);CHR$(29);"[UCD]DIRECTORIO":FLS=0:SETCOLOR  2,15,4
```

```
 70  ? "INSERTE DISKETTE DE DATOS, <RETURN>":GET #1,K:GOTO 80
 75  ? "SALIDA POR IMPRESORA(S/N) ";:GET #1,K:IF CHR$(K)="S" THEN FLAG=1:GOTO 80
 77  FLAG=0
 80  ? "QUIERES ESPECIFICAR RANGO(S/N)":GET #1,K:IF CHR$(K)="S" THEN ? "RANGO";:INPUT
     PPIO,FIN:GOTO 100
 90  PPIO=1:FIN=99999
100  ? "ARCHIVO DESEADO";:INPUT ESPEC$
105  IF ESPEC$=" " THEN ESPEC$(1)=CHR$(CO):ESPEC$(2)=".":ESPEC$(3)= CHR$(CO)
110  F$=ESPEC$:ESPEC$(1,2)="   ": ESPEC$(3)=F$:F$=" "
120  FOR DSK=PPIO TO FIN
130  FN$="D:DISCO":FN$(7)=STR$(DSK)
140  TRAP 605:OPEN #2,4,0,FN$:? "DISCO ";DSK:TRAP 605:IF FLAG  THEN LPRINT "DISCO ";DSK
150  INPUT #2; F$:I=3
152  REM ** BUSQUEDA Y COMPARACION DE ARCHIVOS
155  IF F$(I,I) <> " " AND F$(I,I) <> "*" THEN 600
160  IF ASC(ESPEC$(I,I))= CO THEN I=I+1:GOTO 500
170  IF ESPEC$(I,I)="." THEN 500
180  IF ESPEC$(I,I) <> F$(I,I) THEN 150
200  IF I<LEN(ESPEC$) THEN I=I+1:GOTO 160
210  FLS=FLS+1:? F$:IF FLAG THEN LPRINT F$
215  GOTO 150
500  I=I+1:L=LEN(F$)-6
510  IF ASC(ESPEC$(I,I))=CO THEN 210
520  IF ESPEC$(I,I) <>F$(L,L) THEN 150
530  IF I<LEN(ESPEC$) THEN I=I+1:L=L+1:GO TO 510
540  FLS=FLS+1:? F$:IF FLAG THEN LPRINT F$
542  GOTO 150
600  ? F$:CLOSE #2:NEXT DSK
605 ? :? FLS;" ARCHIVO(S) ENCONTRADO(S) ":IF FLAG THEN LPRINT :LPRINT FLS;" ARCHIVO(S)
     ENCONTRADO(S) "
606  FOR P=1 TO 500:NEXT P
610  GOTO 20
2999 REM **FUNCION DE CATALOGACION
3000 ? CHR$(125):SETCOLOR 2,1,4:SETCOLOR 1,0,15:? "CATALOGO[UCD]":? :? "NRO DEL DISCO
     A CATALOGAR";:INPUT DSK
3015 FN$="D:DISCO":FN$(7)=STR$(DSK)
3020 ? :? "INSERTA DISCO ";DSK;:? " <RETURN> ";:GET #1,A
3045 OPEN  #2,6,0,  "D:*.*":I=0
3050 TRAP 3500:GET #2,K
3052 D(I)=K:I=I+1:GOTO  3050
3500 D(I+1)=0:CLOSE #2:PRINT :? "INSERTA DISKETTE DE DATOS-LUEGO RETURN":GET #1,A:OPEN
     #2,8,0,FN$:I=0
3505 TRAP  3600:PRINT CHR$(D(I));:PUT  #2,D(I):I=I+1
3510 IF D(I)=0 THEN 3600
3515 GOTO 3505
3600 ? "CATALOGACION FINALIZADA":CLOSE #2:GOTO 20
```

ME ENCANTO ESTE PROGRAMA. LO ENCUENTRO SUPER UTIL. EN REALIDAD ES UNA GRAN PERDIDA DE TIEMPO ESTAR PONIENDO UN DISKETTE TRAS OTRO PARA VER SU DIRECTORIO. LO UNICO MALO ES QUE EL DIRECTORIO NO TE MUESTRA MUCHA INFORMACION, PERO ¡BUENO! NO HAY QUE SER TAN EXIGENTE.

¿TAN COMPLICADO PARA UN PROBLEMA QUE PARECE TAN SIMPLE?.

Lo que pasa es que además te solucionará muchos otros problemas. Revisemos algunos conceptos del **DOS** primero.
Como debes saber, el DOS de Atari accesa los discos por sectores: uno a la vez. Sin embargo, esto también puede hacerse en forma independiente, sin usar DOS, modificando ciertos punteros en la sección de memoria conocida como "bloque de control de disco" (DCB, DISK CONTROL BLOCK) y llamando al Sistema Operativo (OS, OPERATING SYSTEM), para que haga el resto del trabajo.

PERO .....

Lo siento, no hay preguntas entre medio. Deja explicarte todo bien y después me preguntas lo que quieras. ¿Ya?.

..... SI NO HAY OTRA ALTERNATIVA.

Bueno, DCB y OS no necesitan que esté cargado el DOS en este método. Fíjate después en las líneas 8010 y 8100 del programa.
Esa es la subrutina para accesar directamente el disco. Ahora te explico como funciona el programa.
Primero necesitamos definir un área "buffer" para que guarde el contenido del sector del disco que leemos o para que guarde alguna información que deberá grabarse en el disco ¿está claro?.

HASTA AQUI VAMOS BIEN. ENTIENDO QUE TENGO QUE RESERVAR UN AREA AUXILIAR EN LA MEMORIA QUE GUARDA LO QUE SE LEYO EN EL DISKETTE O LO QUE TENGO QUE ESCRIBIR EN EL.

Correcto. Este buffer deberá tener 128 bytes de longitud ya que cada sector del diskette tiene 128 bytes. Para esto usamos BUF$ en el programa.
Como te dije antes, la línea 8010 identifica el comienzo del DCB que ocupa las direcciones de memoria 768-779. Haciendo un POKE con valor 1 al segundo byte (BLK + 1) se especifica acceso al drive 1. La variable FUN de la línea 8020 guarda el número de la función para la rutina de acceso al disco; 82 significa un comando de lectura del sector y 87 es un comando de escritura al disco.
Para poder leer lo que tiene el area buffer, o bien escribir en esta area, la dirección de este buffer debe ubicarse en BLK + 4 y BLK + 5 en formato "byte

menor/byte mayor" (lowbyte/highbyte).

PERDONA QUE TE INTERRUMPA, PERO MEJOR ME EXPLICAS ESO O SI NO NO ENTENDERE ABSOLUTAMENTE NADA DE AQUI EN ADELANTE.

Lo anterior significa que BLK + 5 contendrá la parte superior de la dirección, el número de veces que cabe 256 en la dirección. Esto se expresa como INT (ADR (BUF$) /256.
BLK + 4 contendrá la parte inferior de la dirección, el resto de la división por 256, dada por: ADR(BUF$) - 256 (parte superior). Este formato de dirección low byte / high byte es standard en el Atari y otros microcomputadores basados en el microprocesador 6502.
Luego, el número del sector que debe ser accesado debe expresarse en este mismo formato low/high y se ubica en BLK + 10 y BLK + 11. Los sectores se enumeran desde 1 a 720. Algunos de estos sectores se usan para el directorio y otras cosas, por lo que el DOS mostrará solo 707 sectores disponibles en un diskette vacio, aunque tu puedes accesar todos los sectores con este programa.

¡QUE FRESCURA LA DEL DOS! SE QUEDA CON 13 SECTORES EN CADA UNO DE MIS DISCOS.

Bueno, ya es hora de llamar al Sistema Operativo. La subrutina que escribe el contenido del buffer en un sector del disco especifico y que escribe el contenido del sector en el buffer se localiza en la dirección hexadecimal $E453 (decimal 58451). Para llamar a esta subrutina necesitas hacer un pequeño programita en lenguaje de máquina. La línea 8080 inicia este programa que las líneas 205 y 210 ponen en DSKINV$. Después de volver, si hubo algún problema al leer o escribir un sector, se coloca un código de error en BLK + 3, el valor 1 indica que no hubo ningún error. La línea 8080 además pone este código en la variable NR para usarlo en las líneas 8086 y 8090.

¡TANTAS VARIABLES! ESPERO QUE NO SE ME CONFUNDAN.

No te preocupes, después te daré una lista de todas las variables y su significado. Ahora que ya sabes como se lleva a efecto el acceso directo al disco, veamos el programa principal. Cuando hayas escrito el programa, grábalo en un disco que tenga diferentes archivos del DOS, como el AUTORUN.SYS, el DOS.SYS, DUP.SYS, algún programa de LIST/ENTER y otros de SAVE/LOAD, que será muy interesante revisarlos.
Cuando corras el programa te aparecerá un menú. Presiona el botón SELECT y

si todo va bien, podrás cambiar el fondo de la pantalla de azul a verde (o azul y azul oscuro, dependiendo de tu televisor). Cuando la pantalla esté verde (o azul oscuro) puedes imprimir todo lo que aparece en ella, asumiendo que tienes una impresora conectada. En cualquier punto de la ejecución del programa, puedes pasar del modo de impresión al de no - impresión y vice versa.
Seleccionemos una pantalla de no-impresión por ahora. Las opciones son:

1.- Revisar un sector
2.- Revisar un sector del directorio
3.- Rastreo sector por sector
4.- Revisar un bloque de sectores
5.- Mapa de utilización del disco
6.- Tabla de sectores utilizados
7.- Copiar un sector
8.- Respaldar el directorio
9.- Proteger / desproteger sectores
10.- Revisar la memoria.

Escoge la opción 2 y presiona RETURN. Esta opción te permite revisar el contenido de un sector en forma especial, el programa asume que estás buscando uno de los sectores entre el 361 y 368 y te mostrará los datos de acuerdo a esto. Cuando te pide el número del sector escribe 361, para revisar el primer sector del directorio. Escucharás un "beep" mientras se lee el sector y luego los bytes en el sector se despliegan en la pantalla como caracteres. Los sectores del directorio están organizados en grupos de 16 bytes y cada grupo contiene información de uno de los archivos que aparece en el directorio del DOS. DOS se refiere a los programas y archivos a través de números, comenzando desde 0 y con un máximo de 64 archivos.

EL ULTIMO ARCHIVO TENDRIA ENTONCES EL NUMERO 63. ¡POBRES ARCHIVOS! LOS TRATAN COMO A LOS PRESOS.

La pantalla se divide en 8 líneas de 16 bytes cada una, así cada directorio aparece en una línea separada. Fíjate en el primer byte de un directorio típico. Una B mayúscula indica un archivo normal listo para ser usado. De esta forma un primer byte igual a 66 (el valor ATASCII de B) le indica al DOS que ese archivo está bueno y operativo. Un archivo protegido tiene el bit 5 del primer byte en "ON" (valor 1). Si el bit 7 del primer byte está en "ON" DOS considera este archivo eliminado del directorio. Ese espacio está entonces disponible y será re-escrito por el próximo archivo que se grabe. Si se borró un archivo con la opción D del DOS y no se han grabado nuevos archivos en el disco, el archivo puede recuperarse.

¡FANTASTICO! NO TENIA IDEA QUE SE PODIA HACER ESO. ME SIGO ASOMBRANDO CON MI ATARI. ¡ES UNA MARAVILLA! ME DA RABIA ACORDARME DE TODOS LOS PROGRAMAS QUE HE BORRADO POR EQUIVOCACION Y CREIA PERDIDOS PARA SIEMPRE. DE HABER SABIDO LOS HUBIESE RECUPERADO Y ..... ¡BUENO! YA APRENDI Y YA NO ME PASARA MAS.

Ya que te interesó tanto esta parte, probemos con un ejemplo, a ver si realmente ingresaste el programa en forma correcta. Escribe algún programa tonto, como 10 REM y grábalo en diskette. Recurre al DOS y lista el directorio. Ahora usa la opción D para eliminar el programa. Un nuevo directorio del disco te servirá para verificar que tu programa no aparece. Vuelve al BASIC y carga el programa DESCUBRIDOR. Ejecútalo y elige la opción 2 nuevamente. Revisa el sector 361. Si tienes muchos archivos en el disco, puede que tengas que revisar más sectores, posiblemente hasta el 368, para ver el programa que borraste; presionando RETURN un par de veces puedes volver al menú principal para seleccionar el próximo sector. Repite esto hasta que encuentres tu programa y verás que aparece a pesar de que tu lo eliminaste.
Presiona RETURN una vez más para que te aparezca el mensaje "CAMBIOS EN EL SECTOR?".
Responde S, luego escribe C en el próximo mensaje, de manera que puedas agregar bytes dentro del sector en forma de caracteres y no de números. Escribe 1 para el número de caracteres que quieres cambiar. Determina cual es el primer byte del programa que eliminaste para responder al siguiente mensaje. Este debe ser un corazón invertido (CHR$(128)) que indica el archivo eliminado. El número del byte aparece al lado izquierdo de la pantalla.
Escribe el número del byte inicial. El programa te pedirá el byte que quieres insertar: escribe una B, luego S para corroborar que si quieres hacer el cambio. Escucharás un sonido en la diskettera cuando se esté
modificando el sector del disco y luego un "beep" cuando se vuelve a leer y a desplegar en la pantalla. Tendría que aparecer la B como el primer byte de tu programa, lo que indica que lo recuperaste. Para verificarlo trata de cargarlo o vuelve al DOS y mira un listado del directorio.

¡NO LO PUEDO CREER! ES MUCHO MAS SENCILLO DE LO QUE YO SUPONIA.

Ah!, un detalle. Aunque ya puedes cargar y grabar tu programa, sus sectores no estan protegidos por el DOS. Esto significa que si grabas un nuevo programa te puede quedar sobre el tuyo. Para protegerlo, grábalo con un nombre nuevo y DOS marcará sus sectores como si estuviesen en uso.

TENGO UNA IDEA! PERO NO, NO CREO QUE SIRVA.

Pero dímela primero, ahí vemos.

YO SE QUE SE PUEDE HACER, PERO NO SE SI CON ESTE PROGRAMA. ME HAN CONTADO Y LO HE VISTO EN ALGUNOS DISCOS, QUE UNO PUEDE DEJAR INVISIBLES ALGUNOS PROGRAMAS, O SEA, ESTAN EN EL DISCO PERO OTRA PERSONA NO LOS PUEDE VER EN EL DIRECTORIO.

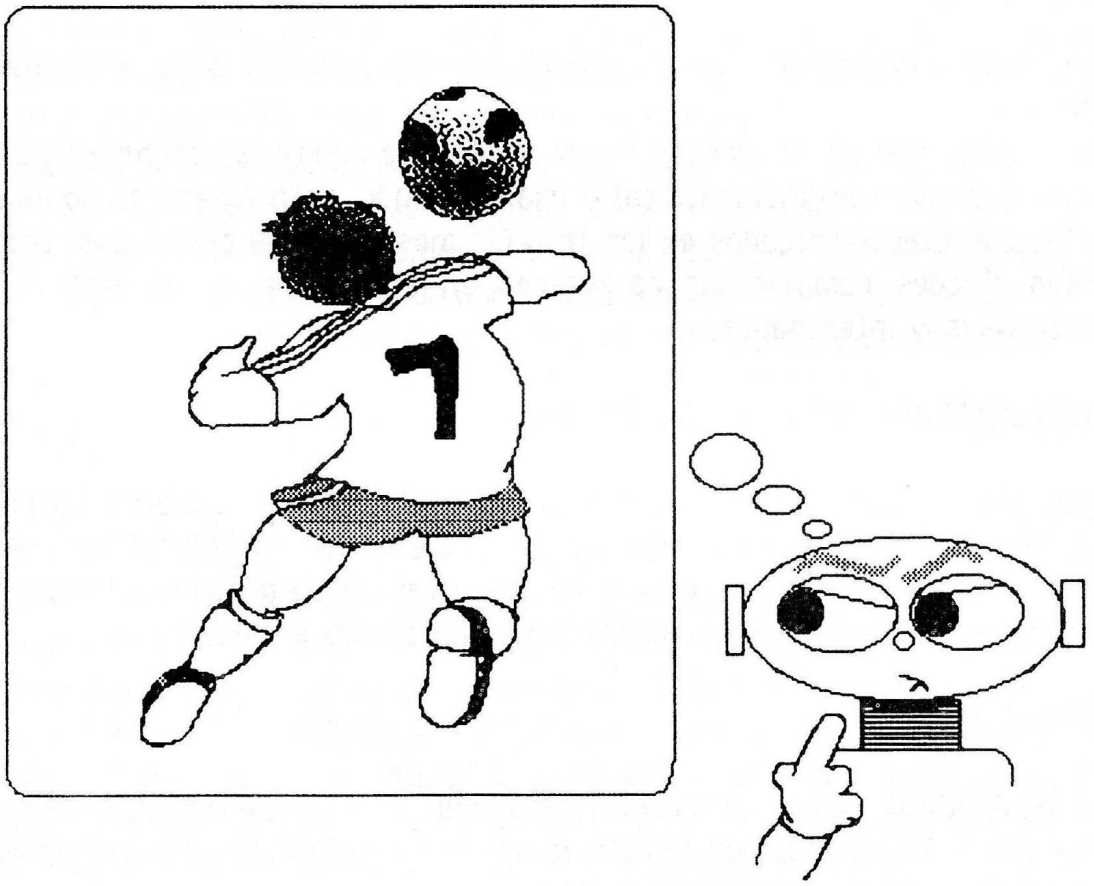
Por supuesto que sirve este programa. Es muy sencillo dejar invisible un archivo.
Como puedes ver en el directorio, el nombre de todos los archivos tiene 2 partes: el nombre propiamente tal y una extensión. Esta extensión no es más que los caracteres ubicados en los tres últimos espacios disponibles para el archivo. Puedes cambiar los caracteres en el nombre de un archivo con resultados muy interesantes.

¡OTRO EJEMPLO! ¿YA? ASI ME QUEDA MUCHO MAS CLARO.

Bueno, elige la opción 2. Escoge N esta vez, para insertar números. Escribe 1 para insertar solo un número. Usa nuevamente el programa "tonto" que usamos en el ejemplo anterior y elige uno de los bytes de su nombre para cambiar. Cambialo al valor 125, luego escribe S para modificar el sector.

¿POR QUE 125?

El valor ATASCII 125 es el caracter para limpiar la pantalla (CLEAR SCREEN). Si vuelves al DOS y pides un directorio del disco, cuando llegues al nombre de tu programa, la pantalla quedará en blanco.
No puedes cargar el programa con su nombre modificado porque CHR$(125) no es un caracter legal en un nombre de archivo. Sin embargo DOS no considera disponibles estos sectores, estan protegidos y no puede escribirse nada en ellos. Creaste, entonces, un archivo que no puede ser cargado o ejecutado sin modificar su nombre. Con una elección apropiada de los caracteres de control (por ejemplo CTRL <--), puedes nombrar un archivo y hacer que ni siquiera aparezca en la lista.
Mira, aqui tienes otro ejemplo para probar. Haz otro programa cortito y grabalo como el primer programa en un diskette vacio. Luego graba otros programas con otros nombres. Ejecuta el programa "DESCUBRIDOR", usa la opción 2 para revisar el sector 361 y usa la opción de cambio con inserción de



números ocho veces para insertar los siguientes números al principio del area de entrada de tu programa (comienza al principio, no en el nombre): 66, 1, 0, 4, 0, 88, 156, 155. Luego cambia los ocho bytes restantes de ese directorio por espacios. Puedes usar la opción de inserción de números, con inserción de bytes múltiples, ya que son todos el mismo número. El número 32 representa el caracter ATASCII del espacio en blanco.
Vuelve al DOS. Lista el directorio; no aparece nada. Todavía puedes cargar, ejecutar y grabar los programas en forma normal, pero solo si conoces sus nombres correctamente.

¡QUE ENTRETENIDO! TODO EL MUNDO ALEGA QUE LOS MICROCOMPUTADORES NO TRAEN BUENAS FUNCIONES DE PROTECCION Y COSAS ASI, PERO CON TODAS ESTAS POSIBILIDADES EL QUE NO PROTEGE SUS PROGRAMAS ES PORQUE NO QUIERE.

Así es. Y ya vez que no es demasiado complicado.

SABES, ME QUEDO UNA DUDA CUANDO MODIFICAMOS EL NOMBRE DEL PROGRAMA Y LE METIMOS UN CARACTER PARA BORRAR LA PANTALLA DENTRO DEL NOMBRE ¿TE ACUERDAS?

Si. ¿Cuál es tu duda?

CUANDO LO LISTO CON D.O.S. SE LIMPIA LA PANTALLA PERO EN EL DESPLIEGUE POR PANTALLA APARECE UNA FLECHA.

Existe un POKE para hacer que el computador despliegue caracteres de comandos en vez de ejecutar los comandos. Ese POKE lo encuentras en la línea 1047 (POKE 766,1). POKE 766,0 te devuelve al modo normal. Prueba POKE 766,1 en modo directo y trata de editar la pantalla con las flechas de control.
Con la opción 2 además se puede ver más información de un archivo. Entre el primer byte y el primer byte del nombre hay 4 bytes que contienen el número del primer sector del archivo y la longitud de éste en sectores. Esta información está almacenada en el formato low/high que vimos hace un rato. Con la opción 2, la interpretación de estos bytes se despliega al lado derecho de la pantalla.

Y TODAS LAS OTRAS OPCIONES DEL PROGRAMA ¿NUNCA SE OCUPAN?

Entretengámonos un poco con las otras opciones. Examina el sector del

directorio que contiene el archivo DUP.SYS, el DISK UTILITIES PACKAGE (Paquete de utilitarios de disco) y fíjate en que sector comienza. Ahora vuelve al menú principal y escoge la opción 1. Escoge el sector donde comienza DUP.SYS. Ingresa la opción de despliegue en caracteres y verás el primer sector de DUP.SYS desplegado en forma de caracteres. En alguna parte en ese sector verás el mensaje que aparece en la parte superior de la pantalla cuando pones el DOS. Con la opción de cambio de sectores puedes poner tu propio mensaje y así cada vez que pongas el DOS se desplegará.

AH! LE PUEDO PONER MI NOMBRE Y ASI SI ALGUIEN ME LO COPIA LO PILLO. TAMBIEN PUEDO PONERLE ALGUNA EXPLICACION O UN NOMBRE SIMPATICO.

Fíjate además que el computador emite un ruido (beep) varias veces durante el despliegue. Esto ocurre porque hay varios bytes en el archivo que tienen valor 155, el código de la tecla RETURN presionada.

BUENO, LO PUEDO ELIMINAR CON EL POKE QUE USAMOS RECIEN.

No, esto no tiene representación en caracteres, ni siquiera con el POKE 766,1 así que normalmente rompe la imagen en la pantalla con saltos de línea.

¿Y NO SE PUEDE EVITAR?

Para evitar esto, el programa los detecta, hace sonar el "beep" para que tu sepas que existen y pone un asterisco en inverso en la pantalla para mostrar su ubicación. Esto lo hace la línea 1067.
Cada sector en el disco tiene 128 bytes de largo. Sin embargo, si el sector fue llenado con el Atari DOS, solo quedan disponibles 125 bytes para datos.

¡YA ESTA EL FRESCO DEL DOS OCUPANDO MIS SECTORES!
¿QUE PASO CON LOS OTROS TRES RESTANTES?

Los otros 3 bytes contienen información "doméstica". El byte 126 tiene en 6 de sus bits, el número del archivo con el cual se asocia el sector. En los otros dos bits del byte 126 y en los 8 bits del byte 127 está el próximo puntero.

¿EL PROXIMO PUNTERO? SI NO TIENE QUE VER CON FUTBOL, NO TENGO IDEA DE LO QUE ME ESTAS HABLANDO.

El próximo puntero es un indicador. En este caso indica el próximo sector de

ese archivo específico del DOS. Lo que pasa es que todos los sectores asociados a un cierto archivo del DOS no tienen porque estar contiguos, uno al lado del otro, en el disco. Por eso necesitamos un puntero que nos indique cual es el próximo sector.
El byte 128 tiene un contador de la cantidad de bytes sin uso que quedan en el sector.

¿Y COMO PUEDEN QUEDAR BYTES SIN USO EN UN SECTOR?

Por ejemplo, si ese sector es el último de un archivo, puede que no esté lleno. DOS necesita saber cuantos bytes están en desuso para no cargar esta parte del sector a la memoria principal como parte del archivo.

BUENO, EN REALIDAD ESTA INFORMACION ES BIEN UTIL. POR ESTA VEZ CREO QUE EL DOS TIENE RAZON DE OCUPARLOS.

La opción 3 del Menú principal permite rastrear automáticamente a través de un archivo de DOS sector por sector, siguiendo los punteros. Encuentra el sector inicial del archivo DUP.SYS. Escoge esta opción con su primer sector como el inicial. Ahora te aparece un menú con dos opciones: auto trace y pausa.

¿Y DE QUE SE TRATAN?

Pausa indica que el computador esperará que presiones una tecla después que se despliega cada sector, para seguir con el siguiente. Auto trace te permite recorrer los sectores uno tras otro hasta el final del archivo.

BUENO, PERO SIGAMOS CON EL EJEMPLO.

Ya. Escoge la opción de pausa. Luego escoge despliegue en forma de caracteres. Verás los sectores del archivo DUP.SYS que se despliegan sucesivamente. Parte de la información, como el menú del DOS será ilegible. La opción Trace (rastreo) también se puede usar para revisar programas almacenados.

¿Y ES LEGIBLE UN PROGRAMA DE ESTA FORMA?

Si un programa ha sido salvado (SAVE), la mayor parte de él lucirá como garabatos. Sin embargo, si un programa fue listado (LIST) al disco, será posible leerlo ya que todavía está en caracteres ATASCII.

Al mirar un programa listado verás las líneas de programa corrientes, seguidas de un "beep" y un asterisco en inversa.

¡AH! ESE ASTERISCO CORRESPONDE AL RETURN QUE SE PRESIONA DESPUES DE CADA LINEA.

La Opción 6 te entrega una tabla de todos los sectores utilizados por un archivo dado, si tu le das el sector inicial. Con esta opción te darás cuenta que en muchos casos los sectores no son contiguos.

VAMOS MAS LENTO. TE SALTASTE LA OPCION 4.

La Opción 4 es muy similar a la 3. Se usa para revisar un bloque de sectores contiguos, ingresando el sector inicial y el número de sectores que quieres ver. Esta opción no sigue el método "encadenado" del trace, ignora por completo los punteros.

AHORA NOS TOCA LA OPCION 5.

Como todos los sectores creados con DOS contienen el número de la posición que ocupan en el directorio es posible hacer un mapa del disco, marcando cada sector con el archivo al cual está encadenado. Y esto es exactamente lo que hace la opción 5.
Ejecútala, con un disco que tenga muchos archivos y haya sido muy usado cargando y regrabando estos archivos. Toma solo unos pocos minutos ..... el mapeo de los 720 sectores. Si tienes una impresora seguramente te interesará cambiar al modo de impresión para tener una copia del mapa para referencias futuras.

SABES? SE ME OCURRE QUE NO ES MUY VENTAJOSO TENER UN PROGRAMA TAN SEPARADO EN EL DISCO, SE DEBE PERDER TIEMPO EN JUNTAR TODOS LOS SECTORES PARA CARGAR UN PROGRAMA. ¿NO SE PUEDEN JUNTAR TODOS LOS SECTORES QUE CORRESPONDEN A UN MISMO PROGRAMA?

Claro que sí. Veámos un ejemplo. Supongamos que escribiste tres programas A, B y C en ese orden. Estan almacenados en forma secuencial en el disco, con 10 sectores por programa. Luego modificas el programa A, agregándole algunas líneas con lo que ocuparía 15 sectores. Al salvarlo, el DOS sigue usando los 10 sectores del principio pero ahora necesita 5 más, los tiene que

dejar después del programa C. Ahora el mapa del disco sería A-B-C-A.
Luego puedes modificar el programa C y tendrías A-B-C-A-C y así sucesivamente.

¿VERDAD QUE SE PONE MAS LENTO?

Y más que eso es mucho más fácil que ocurran errores con los comandos **LOAD** y **SAVE**.
Si Duplicamos el disco igual sigue con la misma organización, pero usando la opción **COPY** del DOS, con nombre de archivos *.* los sectores de un mismo programa quedarán contiguos en el disco, lo que lo hace mucho más confiable.

CON TANTOS SECTORES DISPERSOS ¿COMO SABE EL DOS QUE SECTORES ESTAN EN USO Y NO PUEDEN UTILIZARSE?

DOS mantiene un mapa de los sectores del disco, que se actualiza después de cualquier modificación. Este mapa lo guarda en el sector 360. Por esto el sector 360 es un sector especial reservado que se crea cuando formateas un disco.

¿Y POR QUE EL 360?

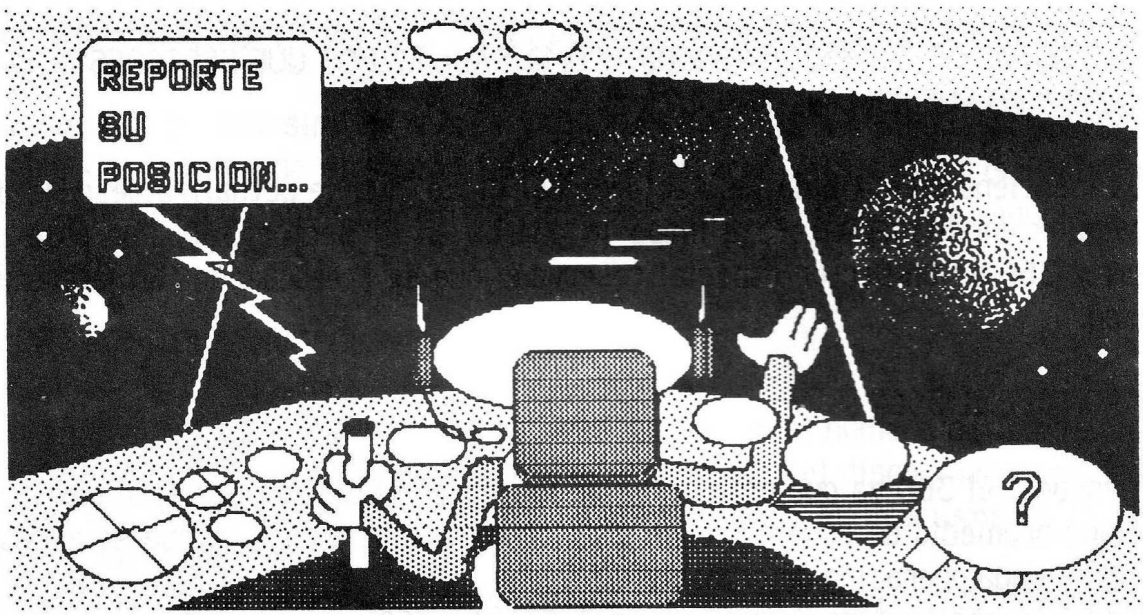
Se escogió el 360 ya que está cerca de la mitad del disco y por esto tiene un tiempo promedio de acceso menor. Esto es muy importante, ya que se accesa durante cada operación de grabación.
Ejecuta la opción 1 del Menú Principal. Escribe 360 para el sector. Escoge la opción 4, que solo aparece cuando eliges el sector 360, un mapa hexadecimal del sector 360. Después de un momento verás el contenido del sector 360 desplegado en hexadecimal. Probablemente la mayoría serán F's y 0's.

¿Y QUE SIGNIFICA ESTO?

El cero indica los sectores en uso. Cada 0 indica 4 sectores en uso.

NO ENTIENDO MUCHO. ¿POR QUE 4 SECTORES?

Recordemos un poco los números binarios y hexadecimales. Cada conjunto de 2 dígitos hexadecimales representa un byte. Un byte son 8 bits, por eso dos dígitos hexadecimales representan 8 sectores en el mapa. F es el dígito hexadecimal de 15, que en binario es 1111. Los sectores libres en un disco se


REPORTE
SU
POSICION...
?

muestran como 1's, por eso FF indica 8 sectores disponibles.

AHORA SI ENTENDI. ¿Y QUE PASA CON LOS OTROS CARACTERES HEXADECIMALES?

C en hexadecimal es un 12 en decimal, lo que en binario es 1100. Por eso una C en el mapa indica 2 sectores disponibles, seguidos por 2 sectores en uso. Los sectores en uso estan protegidos. Esta protección puede hacerse con DOS o con algún otro método, por ejemplo con el programa "DESCUBRIDOR". De esta forma puedes reservar espacio en el disco que jamás será ocupado por el DOS.

ES UN POCO COMPLICADO EL MAPA DE LOS SECTORES. UNO NUNCA UTILIZA LOS NUMEROS HEXADECIMALES.

Como ayuda, los números decimales que muestran qué sector se representa por el último bit del último número en esa línea, se listan al lado derecho de la pantalla.

BUENO. AHORA ME GUSTARIA VER EL PROGRAMA.

Primero, como ya te lo había dicho, te daré una lista con todas las variables que usa el programa, que son bastantes.

LISTA DE VARIABLES DEL "DESCUBRIDOR"

BIT$ Tabla de búsqueda de números decimales con sets de bits sucesivos.
BLA$ String de blancos para borrar textos de la pantalla.
BUF$ Area buffer usada para almacenar información leída de un sector, o información que tiene que ser grabada en un sector.
DSKINV$ Llama a la subrutina OS para acceso al disco. (Sistema Operativo).
G$ Guarda una S o N para las respuestas.
H$ Almacena el contenido de BUF$ después de la conversión hexadecimal.
HEX$ Tabla de búsqueda de dígitos hexadecimales para conversión.
KNOT$ Rutina en lenguaje de máquina para negar (NOT) un número.
ND$ Hace un AND lógico entre dos números.
OAR$ Hace un OR lógico entre dos números.
VB$ Rutina en lenguaje de máquina para inicializar la rutina vertical con blancos.
AP Decisión de Auto Trace o Pausa.
B Variable temporal para leer datos a strings.

BIT Valor ATASCII de BIT$(BITMOD).
BITMOD Indice de BIT$.
BLK Comienzo del bloque de control del disco en memoria.
BN Bytes del mismo número que se tienen que insertar en BUF$
CN Número de caracteres que se tienen que insertar en BUF$
FN Número del archivo de los datos del sector
FP Puntero al próximo sector del archivo.
FUN Número de la función para la rutina de acceso al disco.
H Valor ATASCII de los caracteres de BUF$ para convertir a hexadecimal.
HH Byte mayor de H
HL Byte menor de H.
I Variable de índice de loops.
J Almacenamiento temporal e índice de loops.
K Byte desde la pantalla en la rutina de vaciado a la impresora.
LK Variable de elección para opciones de protección/desprotección (lock/unlock).
M Almacenamiento temporal de valores PEEK.
MEM Puntero a bloques de memoria.
N Variable índice a H$
NB Valor del nuevo byte a insertar en BUF$
NFRE Número de sectores disponibles desde el Sector 360.
NFREH Byte mayor de NFRE
NFREL Byte menor de NFRE
NR Número del error en lectura de sectores.
NSL Número de sectores para proteger/desproteger
NUM Número de bloques o sectores para revisar
P Byte desde el teclado para continuar.
PK Variable para almacenamiento temporal de valores PEEK.
Q Para usar con funciones USR
RSLT Resultado de subrutina de operaciones lógicas
SEC Sector a accesar.
SECHI Byte mayor de SEC
SECLO Byte menor de SEC.
SS Sector inicial para operaciones
SSH Byte mayor de SS.
SSL Byte menor de SS
TY Tipo de vaciado requerido (impresora - pantalla).
WH Opción en el Menú principal

LISTA DE SUBRUTINAS

100-260	Inicialización
300-770	Menú principal y submenus.
660-670	Revisar la memoria en vez del disco.
780-1000	Manejo de algunas opciones de menú.
1040-1105	Vaciado de caracteres.
1106-1160	Cambio de un sector.
1200	Borra el texto.
1400-1440	Opción de copia de un sector.
1600-1640	Opción de respaldo del directorio.
1700-1799	Proteger y desproteger sectores del bit map y actualizar el contador de sectores disponibles.
1800-1840	Opciones de mapeo de archivo.
2010-2200	Opción de mapeo de sectores encadenados.
3000-3060	Vaciado de la pantalla a la impresora.
4040-4060	Conversión hexadecimal.
4205-4240	Vaciado hexadecimal.
5015-5060	Vaciado decimal
7010-7100	Determina número del archivo y al próximo puntero.
8010-8100	Lee/escribe un sector del archivo.

PROGRAMA "DESCUBRIDOR"

Precaución: El mal uso de este programa puede destruir programas de gran valor. Por favor lee cuidadosamente las instrucciones de las páginas anteriores antes de usarlo.

```
0       REM SAVE "D:DESCUBRE"
100     REM **PROGRAMA DESCUBRIDOR  **
105     CLOSE #2:OPEN #2,4,0,"K:"
107     CLOSE #4:OPEN #4,12,0,"S:"
109     POKE 766,0
200     DIM   BUF$(130),DSKINV$(10),G$(1),ND$(12),HEX$(16),H$(1155),
        BLA$(38),VB$(11),BIT$(8),OAR$(12),KNOT$(12)
203     HEX$="0123456789ABCDEF"
204     BLA$="
205     DATA  104,32,83,228,96
210     RESTORE 205
211     FOR I=1 TO 5
212     READ B:DSKINV$(I)=CHR$(B)
213     NEXT I
215     DATA    104,104,104,133,209,104,104,37,209,133,209,96
220     RESTORE 215
221     FOR I=1 TO 12
222     READ B:ND$(I, I)=CHR$(B)
223     NEXT I
225     OAR$=ND$:OAR$(8)=CHR$(5)
226     KNOT$=ND$:KNOT$(8)=CHR$(69)
```

```
230     DATA    72,169,8,141,31,208,173,31,208,201,5,208,8,173,255,6,
        73,255,141,255,6
231     DATA    173,255,6,201,255,240,7,169,148,141,198,2,208,5,169,
        162,141,198,2,104,76,95,228
235     RESTORE 230
236     FOR I=1 TO 44
237     READ B:POKE 1535+I,B
238     NEXT I
240     DATA    104,162,6,160,0,169,6,32,92,228,96
245     RESTORE 240
246     FOR I=1 TO 11
247     READ B:VB$(I)=CHR$(B)
248     NEXT I
250     Q=USR(ADR(VB$))
260     J=128
270     FOR I=1 TO 8
280     BIT$(I)=CHR$(J):J=J/2
290     NEXT I
300     LM=23:POSITION 0,0:? CHR$(125);"(EN UN DISKETTE DOS    STANDARD"
305     ? "LOS SECTORES 361-368 CORRESPONDEN AL                 DIRECTORIO)":? :?
310     ? "1.-REVISAR UN SECTOR "
315     ? "2.-REVISAR UN SECTOR DENTRO DEL DIRECTORIO"
320     ? "3.-RASTREAR UN ARCHIVO"
325     ? "4.-REVISAR UN CONJUNTO DE SECTORES"
330     ? "5.-MAPA DE SECTORES ENCADENADOS EN EL DISCO"
335     ? "6.-MAPA DE UN SOLO ARCHIVO"
340     ? "7.-COPIAR UN SECTOR"
345     ? "8.-RESPALDAR EL DIRECTORIO"
350     ? "9.-PROTEGER/DESPROTEGER SECTORES"
355     ? "10.-REVISAR LA MEMORIA"
356     ? :INPUT WH
360     ON WH GOTO 400,400,450,480,2010,1800,1400,1600,1700,375,
        1900
375     ? CHR$(125):? :? :?
376     ? "LOCALIZACION DE MEMORIA INICIAL";:INPUT MEM
377     ? :? "NUMERO DE BLOQUES A REVISAR";:INPUT NUM:GOTO 500
400     ? CHR$(125):? :?
405     ? "(SECTORES DEL DIRECTORIO=361-368)"
410     ? "QUE SECTOR";:INPUT SEC
430     IF WH=2  THEN TY=1:GOTO 720
440     GOTO 600
450     ? CHR$(125) :? :?
455     ? "CON QUE SECTOR COMENZAR EL RASTREO";
460     INPUT SEC:GOTO 500
480     ? CHR$(125) :? :?
485     ? "NUMERO DE SECTORES";:INPUT SEC2
490     ? :? "SECTOR INICIAL ";:INPUT SEC1
500     ? CHR$(125):? :?
510     ?  "1=AUTOTRACE":?  "2=PAUSE":?
520     INPUT AP
600     ? CHR$(125):? :?
```

```
602       ? "TIPO DE DESPLIEGUE:":?
604       ? "1= CARACTERES":? :? "2=HEXADECIMAL":? :? "3=DECIMAL"
605       IF SEC<>360 THEN ? :INPUT TY:GOTO 650
610       ? :? "4=MAPA HEXADECIMAL DEL SECTOR 360":?
615       INPUT TY
650       IF WH<>10 THEN 700
660       LM=12:?  CHR$(125):SEC=1
662       FOR  M=MEM TO MEM+128*(NUM-1) STEP 128
664       POSITION 1,0:? "MEMORIA COMENZANDO EN ";M
670       FOR PK=1 TO 128
672       BUF$(PK)=CHR$(PEEK(M+PK-1))
674       NEXT PK:GOTO  735
700       IF WH=4  THEN FOR SEC=SEC1 TO SEC1+SEC2-1
720       BUF$(128)="    ":FUN=82
730       GOSUB 8010
735       REM GUARDA LUGAR PARA FUTUROS ENCADENAMIENTOS
740       ON TY GOSUB 1040,4040,5015,4040
765       IF WH=10 THEN 840
770       IF WH<>2 THEN 800
780       ? :? "BYTES DE ENTRADA DEL DIRECTORIO:":?
785       ? "START+0=FLAG":?  "(66/NORMAL;128/ELIMINADO;98/    PROTEGIDO;ETC.)"
790       ? "START+1,2=CONTADOR DE SECTORES LO/HI":? "START+3,4=
          SECTOR INICIAL LO/HI"
800       GOSUB 7010
810       POSITION 2,20:? "NRO DEL ARCHIVO=";FN
815       POSITION 2,21:? "PROXIMO PUNTERO AL SECTOR=";FP
820       POSITION 2,22:? "SECTORES EXTRA=";125-ASC(BUF$(128))
840       IF PEEK(1791)=255  THEN GOSUB 3000
845       IF AP=1 THEN GOTO 920
850       POSITION 2,23:? "PRESIONA CUALQUIER TECLA PARA          CONTINUAR";:GET
          #2,P:GOSUB 1105
920       IF WH=3 AND FP=0 THEN POSITION 2,23:? BLA$;:POSITION   2,23:? "FIN DEL
           RASTREO";:AP=2:WH=1:GOTO 850
930       IF AP<>1 THEN POSITION 2,21:? BLA$:POSITION 2,21:?       "PRESIONA CUALQUIER
           TECLA PARA CONTINUAR";:GET #2,P
935       IF WH=10 THEN NEXT M
940       IF WH=3 THEN SEC=FP:GOTO 720
950       IF WH=4 THEN NEXT SEC
1000      GOTO 300
1040      IF WH=10  THEN 1047
1045      POSITION 0,0:? CHR$(125);"NRO SECTOR";SEC:?
1047      POKE 766,1
1060      ? "BYTE"," 02468ACE   SSTLTH":?
1061      FOR I=1 TO 8
1065      ? A6*(I-1);16*I-1,:FOR J=0  TO  15:K=16*(I-1)+1+J
1067      IF ASC(BUF$(K,K))=155  THEN POKE 766,0
1068      ? CHR$(253);"*";:POKE  766,1:GOTO  1070
1069      ? BUF$(K,K);
1070      NEXT J
1071      IF SEC<361 OR SEC>368 THEN ? :GOTO 1075
1072      POSITION  30,3+1:?  256*ASC(BUF$(16*(I-1)+5))+ASC(BUF$(16*  (I-1)+4))
```

```
1073    POSITION 35,3+I:? 256*ASC(BUF$(16*(I-1)+3))+ASC(BUF$(16*
        (I-1)+2))
1075    NEXT I:POKE 766,0
1078    RETURN
1103    IF PEEK(1791)=255 THEN GOSUB 3000
1105    IF WH=10  THEN POP :GOTO 935
1106    FOR I=1 TO 3:POSITION 2,20+I:? BLA$;:NEXT I
1107    GOSUB 1200:? "DESEAS CAMBIAR EL SECTOR(S/N)";:INPUT G$:
        IF G$<>"S" THEN RETURN
1110    GOSUB 1200:? "INSERTA NUMERO (N) O CARACTERES (C)";:  INPUT G$
1112    IF G$="N" THEN 1130
1114    IF G$<>"C" THEN 1110
1116    H$=" ":GOSUB 1200:? "CUANTOS CARACTERES QUIERES      CAMBIAR";:INPUT  CN
1118    GOSUB 1200:? "EN QUE BYTE COMENZAR";:INPUT BYTE
1120    GOSUB 1200:? "ESCRIBE LOS ";CN;" CARACTERES A INSERTAR
        .....":INPUT H$
1122    BYTE=BYTE+1:BUF$(BYTE,BYTE+CN-1)=H$:GOTO  1150
1130    GOSUB 1200:? "CUANTOS BYTES CON EL MISMO NUMERO";:INPUT
        BN
1142    GOSUB 1200:? "NRO DEL BYTE PARA COMENZAR";:INPUT BYTE:B=
        BYTE+1
1144    GOSUB  1200:? "BYTE  ANTIGUO ";BYTE;"=";ASC(BUF$(B,B));
1146    ? "BYTE NUEVO ";BYTE;"=": INPUT NB
1148    FOR I=B TO B+BN1
1149    BUF$(I,I)=CHR$(NB):NEXT  I
1150    G$=" ":GOSUB 1200:? "ESCRIBE S PARA MODIFICAR EL      SECTOR";:INPUT G$:IF
        G$<>"S" THEN 1105
1160    FUN=87:GOSUB 8010:POP :GOTO 720
1200    POSITION 2,21:? BLA$:POSITION 2,21:RETURN
1400    POSITION 0,0:? CHR$(125);:?
1405    ? "NRO. DEL SECTOR A COPIAR";:INPUT SEC
1407    ? : ? "INSERTA DISKETTE FUENTE, PRESIONA CUALQUIER TECLA"
1410    GET#2,P:BUF$(128)="  ":FUN=82:GOSUB  8010
1420    SECT=SEC:? :? "NRO. DEL SECTOR DE DESTINO";:INPUT SEC
1430    ? :? "INSERTA DISKETTE DE DESTINO, PRESIONA CUALQUIER  TECLA":GET #2,
        P:FUN=87:GOSUB 8010
1435    LK=1:SS=SEC:NSL=1:GOSUB  1710
1440    ? :? " SECTOR ";SECT;" ANTIGUO";:? "GRABADO COMO EL SECTOR";
        SEC;"NUEVO":? :? "CUALQUIER TECLA PARA EL MENU"
1445    GET #2,P:GOTO 300
1600    POSITION 0,0:? CHR$(125);:? :? "INSERTA DISCO CON      DIRECTORIO A GRABAR"
1605    ? "Y PRESIONA CUALQUIER TECLA PARA CONTINUAR":GET #2,P
1610    FUN=82:BUF$(128)=" ":FOR I=0 TO 8
1615    SEC=360+I:GOSUB  8010:N=128*I+1:H$(N,N+127)=BUF$:NEXT  I
1620    ? :? "NRO DEL SECTOR INICIAL PARA GRABAR DIR.";:INPUT SS
1625    ? :? "INSERTA DISCO DE DESTINO, PRESIONA CUALQUIER TECLA":
        GET #2,P
1630    FUN=87:FOR I=0 TO 8:SEC=SS+I:N=128*I+1
1635    BUF$=H$(N,N+127):GOSUB  8010:NEXT I:LK=1:NSL=9:GOSUB  1710
1640    ? :? "DIR GRABADO EN EL SECTOR";SS;"AL";SS+8:? :? "CUALQUIER
        TECLA PARA EL MENU"
1650    GET #2,P:GOTO 300
```

```
1700    POSITION 0,0:? CHR$(125):? "1= PROTEGER":? "2=DESPROTEGER":
        ? :INPUT LK
1705    ? :? :? "COMENZAR CON EL SECTOR";:INPUT SS
1707    ? :? "CUANTOS SECTORES";:INPUT NSL
1710    SEC=360:FOR I=0 TO NSL-1:BUF$(128)=" ":FUN=82:GOSUB 8010
1720    S=SS+I:SSH=INT(S/8):SSL=S-8*SSH
1721    BYTE=ASC(BUF$(11+SSH)):BITMOD=SSL+1:BIT=ASC(BIT$      (BITMOD))
1722    Q=USR(ADR(ND$),BIT,BYTE):IF (LK=2 AND PEEK(209)>0) OR (LK=1
        AND PEEK(209)=0) THEN 1795
1725    IF LK=2 THEN Q=USR(ADR(OAR$),BIT,BYTE):RSLT=PEEK(209):
        GOTO 1735
1730    Q=USR(ADR(KNOT$),BIT,255):RSLT=PEEK(209):Q=USR(ADR(ND$),
        RSLT,BYTE):RSLT=PEEK(209)
1735    BUF$(11+SSH)=CHR$(RSLT):NFRE=ASC(BUF$(4))+256*ASC(BUF$
        (5)):IF LK=1 THEN NFRE=NFRE-1:GOTO 1745
1740    NFRE=NFRE+1
1745    NFREH=INT(NFRE/256):NFREL=NFRE-256*NFREH
1750    BUF$(4)=CHR$(NFREL):BUF$(5)=CHR$(NFREH)
1790    FUN=87:GOSUB 8010
1795    NEXT I:IF WH=7 OR WH=8  THEN RETURN
1799    GOTO 300
1800    BUF$(128)=" ":N=0:POSITION 0,0:? CHR$(125):? "EL ARCHIVO
        COMIENZA EN EL SECTOR";:INPUT SEC:POKE 82,1:POKE 201,5
1810    FUN=82:GOSUB 8010:GOSUB 7010:N=N+1:IF N=9 THEN N=1:? :?
1820    ? SEC:SEC=FP:IF FP <> 0 THEN 8010
1830    POKE 82,2:POKE 201,10:? :? "FIN DEL MAPA...":IF PEEK(1791)=
        255 THEN GOSUB 3000
1840    POSITION 2,23: ? "CUALQUIER TECLA";:GET #2,P:GOTO 300
2010    ? CHR$(125):SEC=0:FUN=82:BUF$(128)=" ":POSITION 1,23:?
        "CUALQUIER TECLA PARA TERMINAR";
2020    POKE 764,255:POKE 766,1:FOR J=0 TO 23:POSITION 1,J:?  J*30+1;"-";(J+1)*30;
2025    POSITION 9,J:FOR I=1 TO 30:SEC=SEC+1
2030    IF PEEK(764)=255 THEN GOSUB 8010:GOSUB 7010:GOTO 2038
2035    POKE 764,255:POP :POP :GOTO 2150
2038    IF NR <>1 THEN  ? "*";:GOTO 2100
2040    IF FN<10 THEN ? CHR$(FN+48);:GOTO 2100
2050    IF FN>9 THEN IF FN<36 THEN ? CHR$(FN+55);:GOTO 2100
2060    IF FN>35 THEN ? CHR$(FN+61);
2100    NEXT I:NEXT J
2150    IF PEEK(1791)=255 THEN GOSUB 3000
2160    POSITION 2,23:? BLA$;:POSITION 1,23:? "PRESIONA CUALQUIER
        TECLA";
2200    POKE 766,0:GET #2,P:GOTO 300
3000    POKE 764,255:POKE 766,1:TRAP 3060:CLOSE #3:OPEN #3,8,0,
        "P:"
3020    FOR J=0 TO LM:FOR I=0 TO 39
3030    POSITION I, J:GET #4,K
3035    IF K>127  THEN K=K-128
3036    IF K<32 OR K=127 THEN ? #3;"*";:GOTO 3045
3040    ? #3;CHR$(K);
3045    NEXT I:LPRINT :IF PEEK(764)=255 THEN NEXT J:GOTO 3050
```

```
3047    POKE 764,255:POP :GOTO 3050
3050    LPRINT:LPRINT:LPRINT
3060    POKE 766,0:TRAP 40000:RETURN
4040    FOR I=1 TO 382 STEP 3:H=ASC(BUF$(INT(I/3)+1)):HH=INT
          (H/16):HL=H-16*HH:HH=HH+1:HL=HL+1
4050    H$(I,I)=HEX$(HH,HH):H$(I+1,I+1)=HEX$(HL,HL):H$(I+2,I+2)="":NEXT
        I
4060    IF WH=10 THEN ? :GOTO 4210
4205    ? CHR$(125):? :? "SECTOR ";SEC;"EN HEXADECIMAL"
4210    FOR I=1 TO 16:? 8*I-1;"-";8*I-1,H$(24*(I-1)+1,24*I-1):NEXT I
4220    IF TY=4 THEN J=47:FOR I=5 TO 16:POSITION 36,I:? J;:J=J+64:
        NEXT I
4240    RETURN
5015    POKE 82,1:POKE 201,5
5017    IF WH=10 THEN GOTO 5025
5020    POSITION 2,0:? CHR$(125);"SECTOR ";SEC;" EN DECIMAL "
5025    ?  :FOR I=0 TO 7:? I,CHR$(30);:NEXT I:?
5030    FOR I=1 TO 16:? 8*(I-1),
5040    FOR J=1 TO 8:? ASC(BUF$(8*(I-1)+J)),CHR$(30);:NEXT J:?
5050    NEXT I:POKE 201, 10:POKE 82,2
5060    RETURN
7010    FP=ASC(BUF$(126)):Q=USR(ADR(ND$),FP,3):FP=PEEK(209)*256:
        FP=FP+ASC(BUF$(127))
7050    FN=ASC(BUF$(126):Q=USR(ADR(ND$),FN,152):FN=INT(FN/4)
7100    RETURN
8010    BLK=768
8020    POKE BLK+1,1:POKE BLK+2, FUN
8040    ABUF=ADR(BUF$):AHI=INT(ABUF/256):ALO=ABUF-256*AHI
8050    POKE BLK+4, ALO:POKE BLK+5,AHI
8060    SECHI=INT(SEC/256):SECLO=SEC-256*SECHI
8070    POKE BLK+10,SECLO:POKE BLK+11,SECHI
8080    Q=USR(ADR(DSKINV$)):NR=PEEK(BLK+3)
8085    IF WH=5 THEN 8100
8086    IF NR=1 THEN 8100
8090    POSITION 0,0:? CHR$(125) :? :? :? "SECTOR ";SEC;"=SECTOR
        MALO":IF AP=1 THEN 8100
8095    ? :? "PRESIONA CUALQUIER TECLA PARA CONTINUAR":GET #2,P:
        POP :GOTO 300
8100    RETURN
```

COMPU ATARI

CAPITULO 5

USO DEL JOYSTICK

¡YA ME SIENTO UN EXPERTO EN ATARI BASIC!. LO UNICO QUE ME FALTA SABER ES ALGO QUE ME PROMETISTE ¿NO TE ACUERDAS?

..... algo que te prometí. ¿Será el uso del Joystick?

SI! ESO ES! ME GUSTARIA PODER HACER UN PROGRAMA QUE UTILICE EL JOYSTICK.

Primero te voy a dar algunas indicaciones para que no cometas los errores que suelen cometer las personas que hacen rutinas para usar el joystick dentro de sus programas.

LISTO TRATARE DE NO COMETER LOS ERRORES QUE ME ADVIERTAS.

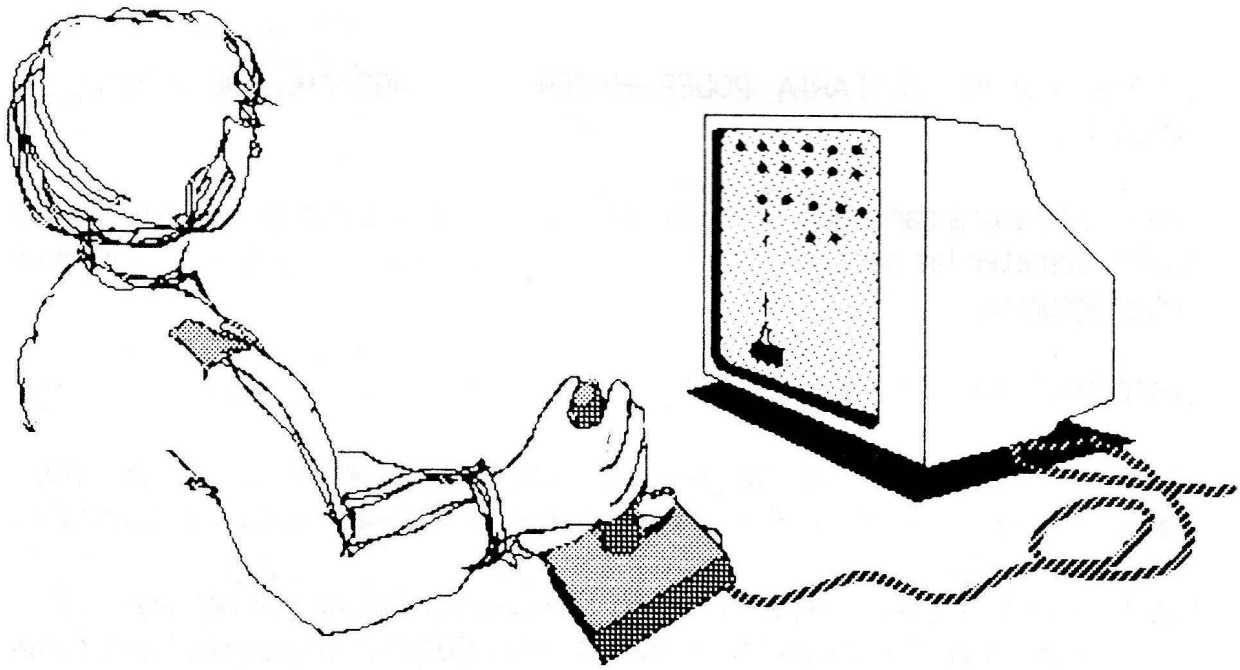
La mayoría de las rutinas de joystick escritas por principiantes contienen numerosas instrucciones IF - THEN que hacen que el joystick responda en forma muy lenta.
Los principiantes, además, a menudo ubican las rutinas de uso del joystick en números de línea muy altos (por ejemplo, 100000) y usan instrucciones como GOSUB 100000 para las rutinas del joystick.

PERO ESO NO TIENEN NADA DE MALO, FUNCIONA.

Sí, funciona, pero es demasiado lento. Es mejor poner estas rutinas al comienzo del programa y comprobarás que aunque sean las mismas instrucciones funciona más rápido.

¿Y CUAL ES LA RAZON DE ESTO?

Es muy simple. Para ejecutar una instrucción como GOSUB 100000 el computador tiene que verificar línea por línea hasta que encuentra la



USO JOYSTICK

100000. Si pones la misma rutina en la línea 10 el computador la encontrará mucho más rápido. Más adelante encontrarás dos rutinas que puedes usar en tus programas.

¿DE QUE SE TRATAN ESAS RUTINAS? ¿COMO FUNCIONAN?

El programa 1 coloca toda la información necesaria para manejar el joystick en dos arreglos ¿sabes para qué?.

BUENO, SUPONGO QUE PARA TENER TODOS LOS DATOS EN UN MISMO LUGAR.

Y además así tienes la información siempre en memoria ¿cierto?. Entonces colocamos un 0, 1 o -1 en cada uno de los elementos del arreglo. X = STICK(0) le dará a X un valor relacionado con la posición del joystick en el slot 1. Si el joystick está en la posición central, X será igual a 15. Cuando los arreglos fueron creados, el quinceavo elemento tanto para el arreglo de filas como de columnas se inicializaron con valor cero (COL(15) = 0; FILA(15) = 0). Por esto, cuando a la posición actual le agregas COL(15), no hay ningún cambio, como tampoco habrá cambio en la fila. En forma similar, si mueves el joystick hacia arriba, la fila irá disminuyendo de a uno pero la columna permanece igual.

A VER, ESTO QUIERE DECIR QUE SI MUEVO EL JOYSTICK UN LUGAR HACIA ARRIBA, EL NUEVO VALOR DE X SERA 14. POR ESTO, COL(14) = 0 y FILA(14) = -1 ¿VERDAD?

Así es, está muy bien tu razonamiento. Aquí tienes la lista con todos los valores de cada uno de los dos arreglos:

COL(1) = 0	FILA(1) = 0
COL(2) = 0	FILA(2) = 0
COL(3) = 0	FILA(3) = 0
COL(4) = 0	FILA(4) = 0
COL(5) = 1	FILA(5) = 1
COL(6) = 1	FILA(6) = -1
COL(7) = 1	FILA(7) = 0
COL(8) = 0	FILA(8) = 0
COL(9) = -1	FILA(9) = 1

COL(10) = -1	FILA(10) = -1
COL(11) = -1	FILA(11) = 0
COL(12) = 0	FILA(12) = 0
COL(13) = 0	FILA(13) = 1
COL(14) = 0	FILA(14) = -1
COL(15) = 0	FILA(15) = 0

NO ENTIENDO MUCHO COMO FUNCIONA ESTO.

Veamos un poquito más en detalle la rutina. Las líneas 100 y 110 comienzan la creación del arreglo ubicando ceros en los elementos del 1 al 4, ya que esos números no se utilizan para la lectura del joystick. La línea 120 usa las sentencias DATA de las líneas 140 y 150 para colocar los valores apropiados en los elementos 5 al 15 de cada uno de los arreglos.

¿PARA QUE SE USA LA INSTRUCCION **TRAP**?

En las líneas 40 y 13 aparece la instrucción TRAP 40. TRAP se usa aquí para evitar la verificación de los límites de la pantalla cada vez que se recorre la rutina. TRAP 40 mueve el programa a la línea 40 cada vez que ocurre una condición de error. Cuando la instrucción **PLOT** de la línea 30 trata de graficar en una posición que está fuera de los límites, TRAP detecta el error. Sin esta instrucción TRAP el programa se detendría. Pero en esta rutina el programa se traslada a la línea 40, verifica los valores de

la columna (C) y de la fila (F) y se resetean dentro de los límites legales.

YA ENTENDI. ¿EL OTRO PROGRAMA FUNCIONA IGUAL?

El programa 2 usa algunas de las mismas técnicas usadas en el programa 1. La diferencia principal es que las respuestas necesarias para el movimiento del joystick no se almacenan en arreglos. En vez de esto, las instrucciones que indican como mover el cursor se ubican en la línea a la cual se vuelve leyendo el joystick.

¿Y COMO SE HACE ESO?

En otras palabras, GOTO STICK (0) lee el joystick y va a la línea que corresponde al movimiento del joystick. Si el joystick está en la posición central, la rutina va a la línea 15 y se repite.
Para que este programa funcione, se tienen que usar números de línea apropiados. La línea 14 moverá el cursor hacia arriba; la línea 13 lo hará hacia abajo; la línea 11 a la izquierda; la línea 7 a la derecha y así sucesivamente.

NO CREO QUE EL NUMERO DE LAS LINEAS SEA UNA LIMITACION TAN GRANDE. PODRIA ESCRIBIR: GOTO STICK(0) * 10 Y USAR LAS LINEAS DE LA 50 A LA 150.

Por supuesto. GOSUB STICK(0) también es una instrucción aceptable. Bueno, estos programas no necesitan tanta presentación, así que aquí los tienes:

PROGRAMA 1 LECTURA DEL JOYSTICK USANDO ARREGLOS.

```
  0     REM SAVE "D:JOYARR"
 10     GOSUB 100
 20     X=STICK(0):C=C+COL(X):F=F+FILA(X):IF X=15  THEN 20
 30     COLOR 0:PLOT A,B:COLOR 1 :PLOT C,F:A=C:B=F:GOTO 20
 40     TRAP 40:IF C=-1 THEN C=0:GOTO P
 50     IF C=80 THEN C=79:GOTO P
 60     IF F=-1 THEN F=0:GOTO P
 70     IF F=48 THEN F=47:GOTO P
100     DIM FILA(15),COL(15)
110     FOR X=1 TO 4:FILA(X)=0:COL(X)=0:NEXT X
120     FOR X=5 TO 15:READ C,F:COL(X)=C:FILA(X)=F:NEXT X
130     GRAPHICS 21:P=30:TRAP 40:POKE 712,132:POKE 709,122:RETURN
140     DATA  1,1,1,-1,1,0,0,0,-1,1
150     DATA  -1,-1,-1,0,0,0,0,1,0,-1,0,0
```

PROGRAMA 2 LECTURA DEL JOYSTICK CON LA INSTRUCCION GOTO

```
  0     REM SAVE "D:JOYGOTO"
  2     GOTO 70
  5     C=C+1:F=F+1 :GOTO P
  6     C=C+1:F=F-1 :GOTO P
  7     C=C+1:GOTO P
  9     C=C-1:F=F+1:GOTO P
 10     C=C-1:F=F-1:GOTO P
 11     C= C-1:GOTO P
```

```
13      F= F+1:GOTO P
14      F= F-1:GOTO P
15      GOTO STICK(0)
20      COLOR 0:PLOT A,B:COLOR 1:PLOT C,F:A=C:B=F:GOTO 15
30      TRAP 30:IF C=80 THEN C=79:GOTO P
40      IF C=-1  THEN C=0:GOTO P
50      IF F=48  THEN F=47:GOTO P
60      IF F=-1  THEN F=0:GOTO P
70      P=20:GRAPHICS 21:POKE 712,116:POKE 709,16:TRAP 30:GOTO 15
```

ES INTERESANTE IR BUSCANDO FORMAS DE OPTIMIZAR EL USO DE LA MEMORIA Y LOS PROGRAMAS MISMOS, ASI SE HACEN CADA VEZ MAS RAPIDOS.

Desgraciadamente ciertas instrucciones y métodos utilizan mucha memoria. Un ejemplo de esto, son las rutinas que hacen llamados (CALL) o rutinas assembler a través de la función **USR**. Este método requiere que la rutina en lenguaje de máquina sea incluida en el programa codificada con instrucciones DATA que tienen que ser leídas (READ) a ingresadas a la memoria con los POKES.
Aparte de la gran cantidad de memoria que esto consume, como son rutinas en assembler, para hacerles cualquier modificación es necesario conocer el Assembler 6502 de Atari.

Y PARA ESO ME FALTA MUCHO, TRATEMOS DE HACER TODO CON EL BASIC. ¿YA?.

Bueno. Tratemos de hacer una rutina de uso del joystick totalmente en BASIC y tan compacto como sea posible, para equipos que tengan poca memoria. Además tiene que ser rápida para que los juegos realmente tengan acción.

LO QUE ME ENSEÑASTE HACE UN RATO TAMBIEN ES IMPORTANTE. EVITAR LAS TRANSFERENCIAS A OTRAS LINEAS YA QUE SI NO ES A LA LINEA SIGUIENTE, EL PROGRAMA COMIENZA A BUSCAR DESDE EL PRINCIPIO.

Correcto. Para empezar revisa esta rutina muy simple:

```
20      S=STICK (S)
30      DX=(S=5 OR S=6 OR S=7)-(S=9 OR S=10 OR S=11)
40      DY=(S=5 OR S=9 OR S=13)-(S=6 OR S10 OR S=14)
50      RETURN
```

¡QUE CORTA! ¿Y COMO LA USO?

Para usar esta rutina, tienes que hacer la variable S igual al número del joystick que quieres leer (0 - 1). Luego, llamar a la subrutina, en este caso GOSUB 20. Los valores Delta-X y Delta-Y se obtienen en las variables DX y DY, respectivamente. Observa que el valor de entrada (S) se pierde. Si llegases a necesitar este valor, cambia el nombre de la variable paramétrica de índice del joystick. Por ejemplo, si tu rutina necesita leer el joystick para cada uno de los cuatro jugadores en un juego, podría contener la siguiente secuencia:

```
FOR I=0 TO 3
GOSUB 20
NEXT I
```

Y la línea 20 podría cambiarse de la siguiente forma:

```
S=STICK(I)
```

En algunas aplicaciones, puede que quieras la lectura del joystick solamente para los cuatro puntos cardinales, es decir, N - S - E - O, e ignorar los movimientos diagonales. Si es así modifica las líneas 30 y 40

de la rutina de la siguiente forma:

```
30      DX=(S=7)-(S=11)
40      DY=(S=13)-(S=14)
```

QUE TAL SI LO PROBAMOS

Claro, porqué no. Escribe la rutina y el siguiente programa de demostración:

```
 10     GOTO 100
100     S= 0:GOSUB 20
110     IF NOT (DX OR DY) THEN 100
120     PRINT ? "DX=";DX,"DY=";DY:GOTO 100
```

Conecta el joystick en el Slot 1, luego ejecuta el programa. Observa los

valores de DX y DY para cada una de las posiciones del joystick.

¡ESTO ME GUSTO! ¿SEGUIMOS PROBANDO?

Prueba el pequeño programa de demostración que encontrarás a continuación. Estudia este programa para que veas como puedes usar la rutina del joystick en tus programas.

RUTINA DE USO DEL JOYSTICK EN ATARI BASIC.

```
  0   REM SAVE " D : USOJOY"
 10   GRAPHICS 23:X=79:Y=47:COLOR 1:GOTO 160
 20   S=STICK(S)
 30   DX=(S=5 OR S=6 OR S=7)-(S=9 OR S=10 OR S=11)
 40   DY=(S=5 OR S=9 OR S=13)-(S=6 OR S=10 OR S=14)
 50   RETURN
100   S=0:GOSUB 20
110   IF NOT (DX OR DY) THEN 100
120   X=X+DX:IF X>159  THEN X=0
130   IF X<0 THEN X=159
140   Y=Y+DY:IF Y>95  THEN Y=0
150   IF Y<0 THEN Y=95
160   PLOT X,Y
170   N=255-INT((X/159+Y/95)*125)
180   SOUND 0,N,10,8
190   GOTO 100
```

NO SE LO QUE LE PASA A MI JOYSTICK. DE REPENTE FUNCIONA, DE REPENTE NO FUNCIONA. NO PUEDE ESTAR MALO EL PROGRAMA PORQUE SI NO, NO ME FUNCIONARIA NUNCA.

Bueno, los joysticks de Atari se producen en masa y a veces pueden fallar. Te voy a enseñar un programa que revela rápidamente cualquier mal funcionamiento en el joystick. Deberás ser capaz de poner la pantalla en rojo cuando presionas el botón del joystick. Deberás, además poder dibujar nueve puntos azules en la pantalla, correspondientes a las nueve posiciones del joystick. Para limpiar la pantalla, presionas cualquier tecla. Puedes controlar también cuán sensible es un joystick controlándolo solo con un dedo, uno sensible es muy bueno para programas como Star Raiders.

ME GUSTARIA QUE ME EXPLICARAS BIEN ESTE PROGRAMA ¿PODRIAS HACERLO PASO A PASO?

Son muy poquitas líneas, así es que te diré que va haciendo cada una de ellas:
La línea 2 escoge GR.3 y hace desaparecer el cursor (POKE 752)
Las líneas 5 a la 15 registran la posición del joystick.
La línea 200 borra la línea antigua y dibuja una nueva línea.
La línea 230 dibuja el punto central y el exterior.
La línea 300 cambia el color de la pantalla a rojo cuando se presiona el botón del joystick.
Las líneas 500 a 520 escriben el valor de STICK(0) y STRIG(0); los PEEKS 656 y 657 mantienen la escritura en solo un lugar.
La 540 limpia la pantalla cuando se presiona cualquier tecla.
La 999 tiene C y D con la última posición del joystick (C y D se usan para minimizar el parpadeo en la pantalla), A y B son las nuevas posiciones del joystick y crean un "loop" continuo.

PRUEBA DEL JOYSTICK

```
  0   REM SAVE " D : PTRUEBAJ"
  2   GRAPHICS 3:POKE 752,1:GOTO 999
  5   A=24:B=14:GOTO 200
  6   A=24:B=6:GOTO 200
  7   A=25:B=10:GOTO 200
  9   A=16:B=14:GOTO 200
 10   A=16:B=6:GOTO 200
 11   A=15:B=10:GOTO 200
 13   A=20:B=15:GOTO 200
 14   A=20:B=5:GOTO 200
 15   A=20:B=10
200   IF C<>A OR D<>B THEN COLOR 4:DRAWTO 20,10:COLOR 2:DRAWTO A, B
230   COLOR 3:PLOT 20,10:PLOT A, B
300   POKE 712,66-STRIG(0)*66
500   POKE 656,1:POKE 657,5
510   ? "STICK(0)="; STICK (0),
520   ? "STRIG(0)="; STRIG(0)
540   IF PEEK(764)<>255 THEN POKE 764,255:RUN
999   C=A:D=B:GOTO STICK(0)
```


NUMERAU !!
NUMERAU !!

ANEXO

COMO USAR LOS COMPUTADORES ATARI

En este pequeño anexo te explicamos algunas características más de los computadores personales ATARI. Además examinaremos muchos de los numerosos productos hechos por Atari y otras compañías para incrementar su poder y aplicaciones.
Como ya lo has escuchado muchas veces, un computador personal es una máquina con la cual ejecutar pequeñas tareas.
Puede realizar una amplia variedad de tareas extremadamente rápido. Esta es una de las principales características de los computadores, aunque la más importante es que tiene una memoria para almacenar información.
Una lista de estas tareas es lo que se conoce como programa. Puedes crear tus propios programas o comprarlos. En jerga computacional, los programas son el software y el resto de los equipos computacionales es el hardware.

1.- COMPONENTES DEL SISTEMA COMPUTACIONAL ATARI.

HARDWARE

Revisemos el hardware incluído en un sistema computacional Atari típico.

a) UCP: El corazón del Atari es el microprocesador o UCP (Unidad Central de Proceso), un pequeño chip localizado en el interior del equipo.
Este microprocesador es el cerebro y centro de control, responsable de todos los cálculos y manejos realizados por el Atari. Debe cumplir las instrucciones que tú le das y controla todas las demás partes de tu equipo. El Atari utiliza un microprocesador 6502, utilizado comúnmente en gran variedad de computadores personales.

b) Memoria: La segunda parte importante del computador es la memoria que sirve para almacenar programas y datos. El almacenamiento se realiza en forma de pulsos eléctricos en el interior de los chips de memoria que están conectados al microprocesador. Como sabes existen dos tipos de memoria, RAM (Random Access Memory) y ROM (Read Only Memory). Para entender la

diferencia entre ellas, pensemos en la memoria como en un gran pizarrón de colegio. A medida que la profesora pasa la materia, va escribiendo en el pizarrón. Puede borrarlo y volver a escribir. En forma similar la memoria RAM puede llenarse de datos y luego ser borrada para poner cosas nuevas.
Ahora imagínate un alumno que ingresa a la sala fuera de la hora y escribe con pintura indeleble en el pizarrón. Para la profesora será imposible borrar esto y ahora el pizarrón no sirve para escribir en él, sólo se puede leer lo que escribió el alumno. Así funciona la memoria ROM.
Discutamos otro punto que es muy importante en relación a la memoria: su limitación en capacidad. Puede recordar solo un número determinado de cosas y existe formas de medir su capacidad en unidades. La unidad de memoria más pequeña es el bit, que solo puede recordar si algo está "ON" o "OFF". Ocho bits se combinan para formar 1 byte que es la memoria necesaria para almacenar un caracter del alfabeto o un número. Mil bytes forman un kilobyte o K, y un millón un megabyte. La cantidad de memoria que tengas en tu computador determina el tamaño de los programas que puedes usar en él. Los computadores Atari vienen con memoria desde 16 K a 64 K RAM.
Una diferencia importante entre los modelos 600XL y 800XL es la cantidad de memoria RAM que tiene cada uno. El Atari 600XL viene con 16K y si tu quieres expandirla, tienes que comprar una expansión que amplía su capacidad hasta 64K. El Atari 800XL se vende con 64K.

c) Teclado: La tercera y última parte del computador Atari es el teclado, nuestro principal canal de comunicación con el microprocesador. Si estás familiarizado con las máquinas de escribir, verás sus similitudes. Esto hace que sea muy fácil para los dactilógrafos digitar.
Tiene además una serie de teclas extra que explicaremos más adelante.
Existe una gran diferencia entre los teclados de los dos modelos Atari que hemos nombrado.
Ahora que ya hemos visto las partes del computador, veamos el resto del hardware necesario para completar el sistema. Estos otros elementos son los periféricos.

d) Monitor: Es probablemente el periférico más crucial del sistema. Permite que el computador se comunique con nosotros, mostrando letras o dibujos en la pantalla. El micrófono del monitor se usa para crear música y sonidos controlados por el Atari. El Atari está diseñado para usar cualquier Televisor como monitor y además pueden usar monitores especialmente diseñados para ser usados con computadores, con lo que se obtiene una imagen más clara.


estamos revisando la imagen con detenci
y...,caramba,no me parece en absoluto.
Ya teníamos referencias del juez,el año
72 ya le anuló un gol legitimo al elenco
albo....vean como hay un hombre que
habilita toda la jugada.El guardalineas
no levanta su banderola!...a ver si podem
repetir esa acción,vean..nnno me parece,
Pedro,que haya estado acertado Arfifilo.

e) Cassettera y Diskettera: Otro importante periférico es el "dispositivo de almacenamiento de datos". Esto podría parecer innecesario, ya que tenemos toda la memoria RAM para almacenar datos. Sin embargo, esta memoria tiene un gran problema: cuando el computador se desconecta, todo lo que está en ella se borra. La solución es leer el contenido de la RAM y pasarlo a otro periférico que pueda almacenar datos sin necesidad de estar conectado a la electricidad constantemente.

¿Entonces para que usar la memoria RAM si es volátil?. Bueno, su facilidad para borrar y aceptar nuevos datos hace que sea muy fácil para el microprocesador usarla. Es mucho más rápida que el mejor dispositivo de almacenamiento.

Un típico sistema Atari usa uno o ambos dispositivos de almacenamiento: cassettera y diskettera. La cassettera es mucho más económica que la diskettera. Los datos son convertidos desde la RAM a señales eléctricas, que se registran en un cassette, de la misma forma que la música.

La diskettera almacena datos en un disco pequeño flexible cubierto de plástico conocido como "floppy disk" o diskettes. Los diskettes están hechos del mismo material que las cintas de grabación pero son un poco más gruesos. Los diskettes vienen protegidos con un sobre para que no se dañen.

La diskettera tiene una cabeza lecto-grabadora que se mueve sobre la ranura ovalada que tiene el sobre del diskette. Esta cabeza grabadora puede registrar señales eléctricas en el disco de la misma forma que las cabezas grabadoras de las cassetteras lo hacen con los cassettes.

Las mayores ventajas de las disketteras frente a las cassetteras son su velocidad y accesibilidad a los datos. Puede cargar y grabar datos de 10 a 20 veces más rápido que una cassettera. Además puede almacenar y recuperar datos en cualquier orden. ¿Y cómo nos ayuda esto?. Hagamos una analogía. Considera el tiempo que gastas buscando una canción en un cassette: tienes que ir deteniendote cada cierto tiempo para ver si llegaste a la canción que buscabas. Si tienes un disco, en cambio, ves en el índice la ubicación y colocas la aguja directamente sobre el comienzo de la canción. En forma similar, los datos almacenados en un diskette pueden encontrarse casi instantáneamente. Cada disco tiene un directorio que informa todos los programas e información contenidos en el disco.

El computador puede revisar ese directorio e ir directamente a los datos requeridos.

Existe una tercera manera de almacenar datos, aunque si no tienes equipamiento especial, no puedes hacerlo.

Este método es grabar los datos en un chip ROM como una serie de circuitos.

Una vez que se ha creado el software se inserta en el chip y queda allí para siempre sin sufrir daños. Puedes comprar programas almacenados en ROM, los que generalmente vienen en un cartridge. La ventaja principal del cartridge es que carga en forma instantánea simplemente instalándolo y encendiendo el equipo. Los cartridges además son muy durables.
Hemos visto el hardware incluído en un sistema de computador Atari típico: el computador propiamente tal, un monitor para ver la información y un dispositivo de almacenamiento (diskettera o cassettera). Aunque cualquier sistema Atari funcional necesita a lo menos estos elementos para obtener resultados, se le puede agregar muchos otros periféricos para aumentar su poder y versatilidad.

SOFTWARE

Un computador sin software no puede hacer nada. El Software viene en varios medios de almacenamiento: diskettes, cassettes y cartridges. Además puede venir escrito en diferentes lenguajes de programación. Revisemos los dos más comunmente usados en Atari.

a) Lenguaje de Máquina. El lenguaje de máquina es el lenguaje interno del computador: todos los comandos que recibe el microprocesador deben estar en lenguaje de máquina. Es un lenguaje muy rápido y eficiente usado por programadores avanzados, pero no es sencillo. Se escribe como una serie de números, los que son convertidos a impulsos eléctricos que le indican al microprocesador que hacer.

b) BASIC. Ya que a la mayoría de la gente no le gusta hablar en números, afortunadamente existe gran variedad de lenguajes de programación mucho más fáciles de usar, que se conoce como "amistosos". Uno de los lenguajes más amistosos es el BASIC que utiliza palabras en inglés para dar instrucciones al computador. Para poder usar el BASIC en el Atari, debemos tener un programa "intérprete" que traduce las instrucciones en inglés a lenguaje de máquina.
BASIC puede venir incluído como un lenguaje con los computadores Atari o

como alternativa en un cartridge que se instala en el slot que está en frente del computador. El Atari 800 XL tiene BASIC en ROM.
Los computadores Atari se comercializan con gran variedad de Software adicional. Por ahora, revisaremos el software que por lo general se incluye en un sistema computacional simple.

c) El Sistema Operativo. Un programa muy importante que se incluye en todos los Atari es el Sistema Operativo (SO), escrito en lenguaje de máquina que ejecuta todas las funciones fundamentales del computador. Sin Sistema Operativo, el computador, aunque lo enciendas, no haría nada. El Sistema Operativo le indica que responda al teclado, como mostrar las cosas en la pantalla, como recibir los programas de la cassettera o diskettera y miles de otras cosas. El Sistema Operativo está contenido en una ROM que está bajo la cubierta del computador y empieza a funcionar en el instante en que el computador se enciende.

d) El Sistema Operativo de Disco (D.O.S.). Si compras una diskettera Atari, esta incluye un programa que se llama Sistema Operativo de Disco (D.O.S.). DOS viene en un diskette y se carga en el computador cuando éste se enciende. DOS le da al Atari la capacidad para intercambiar datos con la diskettera, revisar el directorio del disco y controlar el motor del drive. Sin DOS no puedes usar la diskettera Atari.

2.- PUESTA EN MARCHA DEL EQUIPO

Conectar y hacer funcionar un computador Atari no es muy difícil. El computador y sus periféricos han sido diseñados para conectarse unos con otros fácilmente. La única herramienta que podrías necesitar es un destornillador. Solo sigue las instrucciones del manual del usuario, cualquiera sea el modelo que tengas.
No necesitas condiciones ambientales especiales, pero es bueno que busques un lugar agradable donde quepa tu equipo y periféricos sin problemas.
Lo primero que debes hacer es conectar el monitor (o TV) al equipo y luego conectarlos a la electricidad, siguiendo las instrucciones del manual. Una vez hecho esto enciende el computador (POWER) para ver como funciona. Si no pasa nada, verifica las conecciones. Ahora que funciona podemos desconectarlo e instalarle el dispositivo de almacenamiento. Nuevamente, sigue las

instrucciones de los manuales, ya sea para cassettera o diskettera.
Simplemente enchufa el conector del joystick o del paddle al jack Nº 1; si necesitas conectar otro hazlo en seguida.

3.- USO DEL TECLADO

Ya sabemos que el corazón del Atari es un microprocesador, un pequeño chip localizado dentro del computador. El teclado nos permite comunicarnos con el microprocesador. Cada vez que presionas una tecla, aparece un caracter en la pantalla y además es enviada al microprocesador. Hay una parte especial en el Sistema Operativo, el "editor de pantalla", que maneja lo que se escribe en la pantalla. Para poder "conversar" con nuestro computador, necesitamos saber las reglas por las cuales se rige el editor de pantalla.
A continuación veremos como funciona cada una de las teclas y como el editor de pantalla tiene cuidado al desplegar nuestros mensajes en la pantalla. Notarás varias cosas interesantes. Por ejemplo: si mantienes presionada una tecla se repetirá el caracter o función que representa en la pantalla. Si dejas el teclado por más de 8 minutos, notarás que la pantalla cambiará de color, este es un método de protección de la pantalla que termina apenas presionas una tecla.
Algo muy importante es el pequeño cuadrado blanco que se mueve en la pantalla que se conoce como "cursor". Aparece para indicarte donde estás escribiendo. Salta automáticamente de una línea a otra sin necesidad de presionar RETURN, en inglés el término que describe esta característica es Wraparound.
Puedes moverte a cualquier posición de la pantalla sin necesidad de escribir. En el borde izquierdo del teclado hay una tecla marcada CTRL. Presiona esta tecla simultáneamente con una de las teclas de la derecha marcadas con pequeñas flechas en cuadrados blancos. Es interesante ver que si pasas sobre caracteres que ya escribiste en la pantalla no los borra, simplemente los deja en modo "inverso" cuando está sobre ellos y sigue de largo. En cambio si escribes un caracter, eliminas el anterior.
Una vez que haz llegado al final de la pantalla el editor automáticamente sube todas las filas un lugar más arriba y deja una línea en blanco en la parte inferior, esto se llama "scrolling". Si quieres borrar todo lo que aparece en la pantalla, presiona CLEAR.

Dividiremos las teclas en 4 tipos diferentes para entenderlas mejor.

a) **Teclas de Caracteres** --> cuando son presionadas producen caracteres en la pantalla.

b) **Teclas de Edición** --> no producen caracteres pero mueven el cursor, limpian la pantalla, desplazan caracteres dentro de la pantalla y otras funciones.

c) **Teclas de Programación** --> es un pequeño grupo de teclas que se usa para ejecutar tareas especiales cuando se programa.

d) **Teclas de Control de Programa** --> estas teclas se usan para controlar un programa cuando se está ejecutando.

a) Teclas de Caracteres: Son las más simples: letras, números, puntuaciones y símbolos matemáticos que aparecen en las teclas centrales. Cuando las presionas aparece el caracter en la pantalla. La barra espaciadora en la parte inferior del teclado también es una tecla de caracteres, produce espacios. Algunas teclas son de caracteres múltiples, tienen varios símbolos que se usan con las teclas SHIFT y CTRL.

b) Teclas de Edición: Generalmente se ubican a la derecha y a la izquierda del teclado. Revisemoslas una a una.

Tecla **SHIFT** --> permite obtener un segundo caracter de la mayoría de las teclas. Hay dos teclas SHIFT, una a cada lado del teclado.
Por lo general, entrega la mayúscula de la letra o el símbolo que aparece en la parte superior de la tecla.

CAPS--> Habrás notado que todas las letras que aparecen en la pantalla son mayúsculas. La tecla CAPS te da la posibilidad de escribir con minúsculas. Solo afecta a las teclas con letras. Para poner un "*", por ejemplo, de todas maneras hay que apretar SHIFT y un 3. Para volver a mayúscula se presiona SHIFT y CAPS juntos (solo se presiona por segunda vez CAPS).

CTRL --> CTRL es como un segundo SHIFT que te permite usar caracteres gráficos (puedes mirar tu manual para ver la localización de cada figura en el teclado). Además en las teclas que tienen 3 símbolos nos entrega esa tercera función (como las flechas).
Puede ser bloqueada presionando juntas CTRL y CAPS. Para volver al modo

mayúsculas se presiona SHIFT y CAPS.
El uso de la tecla CTRL con los números da resultados bien interesantes. Presiona CTRL y el número uno juntos y trata de escribir en el teclado; ¡está bloqueado!, no puedes escribir nada. Vuelves al estado normal con CTRL 1.
CTRL 2 hace sonar una alarma y CTRL 3 crea una señal EOF (fin de archivo). Con el resto de los números no pasa nada.

TECLA DE **VIDEO INVERSO** está localizada a la derecha del teclado, presionarla una vez aparecen todos los caracteres en inverso. Al presionarla por segunda vez, los caracteres que se digiten aparecen en forma normal.

RETURN --> En una máquina de escribir, la tecla RETURN sirve generalmente como avance de carro, terminando la línea que estás escribiendo y pasando al inicio de la siguiente, una línea más abajo. La tecla RETURN del Atari hace esto mismo en la pantalla, pero no es exactamente como la tecla RETURN de la máquina de escribir. Piensa en la tecla RETURN como un mensaje al computador que le dice "ya escribí, ahora te devuelvo el control para que puedas leer mi mensaje".

TAB/SET/CLEAR --> esta tecla es realmente tres teclas en una. Al presionarla sola actúa como tabulador. Con SHIFT se habilita la función SET que sirve para definir los distintos puntos de tabulación. Con CTRL te permite eliminar un punto de tabulación.

TECLAS DE MOVIMIENTO DEL CURSOR (->, |, <- y |) --> estas teclas normalmente son teclas de caracteres, cuando las usas solas o con SHIFT. Con CTRL mueve el cursor en la dirección de la flecha. Son muy útiles.

DELETE/ BACK S --> esta es una tecla con tres funciones que te permite borrar caracteres de la pantalla. Al presionarla sola mueve el cursor un espacio hacia atras (izquierda), borrando todo lo que encuentre. Esta es la función BACK S.
La función DELETE puede usarse con SHIFT y con CTRL. SHIFT DELETE hace que desaparezca completa la línea en la cual está el cursor. CTRL DELETE hace que el caracter que está bajo el cursor desaparezca.

INSERT --> es parte de una tecla de caracteres. La tecla INSERT solo funciona con CTRL o SHIFT. Piensa en INSERT como en el antónimo de DELETE. SHIFT INSERT hace que aparezca una línea en blanco donde está el cursor y

todas las demás se mueven hacia abajo. CTRL INSERT agrega un espacio corriendo los caracteres bajo el cursor a la derecha.

CLEAR --> también es parte de una tecla de funciones. Da lo mismo usarla con CTRL o SHIFT y hace que se borre todo lo que hay en la pantalla y el cursor vuelva a la esquina superior izquierda.

c) Teclas de Programación.

Las teclas programables se usan cuando se está escribiendo o ejecutando un programa.

ESC -- esta tecla, abreviatura de ESCAPE, le indica al Atari que ignore la siguiente tecla de edición que se presione e imprima un símbolo especial que represente esa función. Presiona ESC una vez y luego DELETE / BACK S. Verás una flecha en la pantalla pero no se ejecuta la función. Lo mismo pasa con CLEAR, INSERT y el mismo ESC. No funciona con RETURN, SHIFT, CTRL, CAPS, BREAK y la tecla Atari.

BREAK --> la tecla BREAK se usa para indicarle al computador que termine lo que está haciendo. Puedes usarla cuando el computador esté ejecutando algún programa largo que quieres detener, ya sea para cambiar algo en el programa o para cargar un nuevo programa.

d) Teclas de Control de Programa.

Hay cinco teclas "especiales", de control de programa o de "consola":

- **HELP**
- **START**
- **SELECT**
- **OPTION**
- **RESET**

Las teclas START, SELECT y OPTION no nos serán útiles por ahora, porque sus funciones dependen de un programa ejecutándose. START puede usarse para comenzar un juego o para hacer que el programa calcule algo. SELECT puede

elegir entre varias versiones del programa. Es importante recordar que estas teclas no tienen funciones establecidas y a menudo no se usan en algunos programas que se ejecutan. RESET es una tecla de emergencia. La gran diferencia entre presionar esta tecla y apagar y encender el equipo es que RESET no borra un programa en la memoria del computador.

4.- EL EDITOR DE PANTALLA.

Seguramente habrás notado que a veces el editor de pantalla hace cosas inesperadas. Por ejemplo, a veces suena una "alarma" cuando escribes. Si usas SHIFT DELETE para borrar una línea borra varias líneas. ¿Qué está pasando?.
Cuando miramos la pantalla vemos caracteres en distintas líneas y columnas. El editor ve las cosas diferentes. Lo que vemos como una línea puede ser simplemente parte de una línea más larga. Para diferenciar entre nuestra línea y la línea del editor de pantalla, necesitamos nombrar dos tipos diferentes de líneas. Estas son la línea física y la línea lógica (lo que el editor ve).

Cuando queremos enviar un mensaje al microprocesador lo escribimos y luego presionamos RETURN que marca el fin de un mensaje y el inicio del otro. Como es solo un pequeño microprocesador no puede recibir mensajes muy largos. El editor de pantalla limita los mensajes a tres líneas físicas. Este mensaje sería una línea lógica.
Entonces cuando el computador hace sonar su alarma es porque te encuentras 8 caracteres antes del final de una línea lógica. Al presionar **SHIFT DELETE** borrar una línea lógica, no física.

5.- PROGRAMACION BASIC.

Como todos los programas que te entregamos en este libro estan escritos en BASIC, es interesante hacer algunas acotaciones al respecto.
Existen varias versiones de BASIC de disco disponibles para Atari: entre ellas una de Microsoft, que también se usa en muchos otros computadores. Además del BASIC Atari, incluido en ROM en el computador.
Como sabes hay dos formas de utilizar BASIC:
- **Modo inmediato** -> tipeas la instrucción y la ingresas presionando RETURN. El computador ejecuta el comando inmediatamente.

- **Modo diferido** -> podemos ingresar una larga lista de instrucciones en el orden que deseemos. Primero se escribe un número positivo entre 0 y 32767, un espacio y tu instrucción. Al final un RETURN. El computador almacena estas instrucciones y las ejecuta solo cuando se lo indicas con la instrucción RUN.
Cuando el computador ejecuta un programa, verifica los números al comienzo de cada línea en el programa y los va ejecutando en orden numérico, del número de línea más pequeño al mayor. Los números no tienen que ser consecutivos.
Hay algunas instrucciones que alteran el orden en el cual el computador ejecuta el programa. GOTO, por ejemplo es una instrucción que va seguida de un número de línea. Cuando el computador ejecuta esa instrucción, salta al número de línea mencionado y comienza a ejecutar las instrucciones desde allí.
Si tienes una cassettera y quieres grabar tus programas en cassette, esto es lo que tienes que hacer.

1.- Pon un cassette rebobinado (al principio) en la cassettera.
2.- Ingresa el comando **CSAVE** que significa CASSETTE SAVE. Escucharás dos zumbidos del computador. Es la señal para presionar dos botones en la cassettera.
3.- Presiona simultáneamente los botones de **PLAY** y **RECORD** y deberán quedar bloqueados.
4.- Presiona RETURN en el computador. Escucharás un ruido proveniente del monitor y el cassette comenzará a moverse. Después de unos pocos segundos, oirás algunos sonidos desagradables, ¡no te asustes! es el sonido de tu programa que se está transfiriendo al cassette. Cuando termina, se acaba el ruido y aparece la palabra **READY** en la pantalla.
5.- Presiona el botón **STOP** de la cassettera y rebobina el cassette.
Tu programa ya ha quedado grabado.
Si vas a almacenar tu programa en una diskettera, necesitas ponerle un nombre al programa. **DOS** te lo pide para poder llamarlo después. DOS es muy especial con respecto a los nombres, deben cumplir estas reglas:
- Máximo 8 caracteres.
- Solo letras de la A - Z y números del 0 - 9.
- El nombre debe comenzar con una letra.
- No acepta minúsculas, caracteres inversos, ni espacios.
Si necesitas más de 8 caracteres puedes agregar 3 más si pones un punto antes. Esto constituye la extensión.

Se utiliza el comando **SAVE** para grabar seguido de 'D:' para indicar que será en disco y luego el nombre del programa. Esta información debe ir entre comillas, por ejemplo, SAVE "D : TEXTO". Después de ingresar este comando la diskettera comienza a funcionar y el parlante del monitor emite una serie de ruidos. Al final aparece READY que te indica que tu programa fue grabado.
Si quieres recuperar un programa que haz grabado en un cassette, sigue estas instrucciones:
1.- Pon el cassette.
2.- Escribe **CLOAD** (CASSETTE LOAD) y escucharás un zumbido del computador. Esto te indica que presiones una tecla en la cassettera.
3.- Presiona el botón **PLAY**.
4.- Presiona **RETURN**. Se traspasa el programa del cassette a la memoria del computador. Al final aparece READY.
5.- Presiona **STOP** en la grabadora.
6.- Rebobina el cassette para la próxima vez que lo uses.

Cargar un programa desde un diskette es muy sencillo :

1.- Inserta el diskette.
2.- Escribe **LOAD** "D : seguido por el nombre.

Cuando finaliza aparece la palabra READY. Escribe **RUN** para ver si el programa funciona.
Si estás usando una diskettera para cargar el programa, puedes acortar el proceso escribiendo: RUN "D : TEXTO" y con esto combinas el comando LOAD y RUN.

6.- ALMACENAMIENTO DE ARCHIVOS: CASSETTES Y DISKETTES.

A medida que comiences a trabajar con tu computador, comenzarás a acumular programas y datos, llamados por lo general "archivos".
Cuando grabas información en un cassette, es conveniente grabar solo un archivo por lado y así no tienes problemas para encontrar el inicio de tu archivo. De otra forma deberás usar el Contador de la cassettera.
La primera vez que usas un cassette asegúrate que esté al principio y el contador en 0. Después que hayas grabado un archivo deja por lo menos 10

espacios libres y escribe el número del contador. Estás en condiciones de grabar tu segundo archivo. Al cargar un archivo siempre deja el cassette al principio y avanza hasta la posición donde está el archivo deseado y sigue el procedimiento normal. No es raro que desaparezca algún archivo por arte de magia, sobre todo si el cassette está casi completo.
Otro punto importante es que los contadores varían de cassettera en cassettera por lo que puedes tener problemas al buscar un archivo que fué grabado en otro equipo.
Como puedes ver, los cassettes no son siempre el medio de almacenamiento más confiable. Es importante respaldar, hacer copias de los archivos importantes. Además, cuando tengas archivos importantes que no quieres que se te borren jamás, hay una forma de proteger el cassette, quebrando una de las pequeñas patitas que tiene atrás (la del lado derecho). Para proteger el otro lado, quiebras la del otro lado.
Notarás que es imposible grabar en este cassette.
Como regla, el almacenamiento de archivos en diskettes es mucho más conveniente y eficiente que el almacenamiento en cassette.
Los diskettes estandar que usa Atari son de 5 1/4" de densidad simple, un lado y sectorizados por software.
Es necesario formatear los diskettes antes de usarlos, por esta característica de ser sectorizados por software. Esto se puede hacer a través del menú del DOS.
Obtendrás el Menú del DOS cada vez que enciendas el equipo sin cartridge (por supuesto, con el diskette de DOS en la diskettera). Si estás en BASIC simplemente escribe DOS. Notarás que aparecen varias opciones después de las letras A a la O. Cada una de estas opciones ejecuta algún tipo de tarea que te permite copiar archivos, moverlos de disco a disco, renombrarlos, borrarlos, etc. Aprender a usar todas estas opciones toma su tiempo, no las revisaremos todas.

- **Opción de Formateo**: Para formatear diskettes nuevos debes insertarlos en la diskettera y seleccionar la Opción I del menú DOS. Cuando el computador te pregunte que drive formatear, pon 1, a menos que tengas más de una diskettera. Luego Atari te pide la confirmación y escucharás el sonido de la diskettera. Una cosa muy importante que hay que saber del formateo es que BORRA TODO LO QUE HAYA EN EL DISKETTE. La opción de formateo puede ser muy útil como borrador de diskettes, pero úsala con cuidado.

- **Opción de Directorio del Diskette**: Algún día puede que no sepas que contiene un diskette. Ponlo en la diskettera y utiliza la opción A, el directorio del disco. Escribe A seguida de dos RETURN. En la pantalla aparecerá una lista de todos los archivos del diskette.
Cada archivo va seguido de un número que nos indica cuantos sectores ocupa para almacenarlo. Un sector es un área del diskette que almacena 128 bytes de información.
Al final de la lista aparece una línea que te indica la cantidad de sectores disponibles.

- **Opción de Escribir los Archivos del DOS**: Puede ser muy aburrido tener que estar cambiando diskettes cada vez que quieres cargar el DOS. Es más fácil tener el DOS. Es más fácil tener el DOS en el disco que estas usando, la opción H te permite hacer esto. Una vez que tengas el diskette en el drive, escribe H y el computador te preguntará en que drive está el diskette, luego te pide la confirmación y el DOS será copiado a tu diskette.

- **Opción de Cargar un Cartridge**: La última opción del DOS que veremos es la B, Run Cartridge. Si quieres volver al BASIC o cualquier otro lenguaje en un cartridge, escribe B. El menú del DOS desaparece y estas de vuelta en el BASIC en el Cartridge.
Como veíamos recién, cada vez que formateas un diskette se borra todo su contenido por lo que es posible que por equivocación, borres archivos importantes. Por esto podemos proteger los diskettes para que no se pueda escribir encima.
Si te fijas, al lado derecho del diskette hay una ranura por la cual pasa un rayo de luz cuando el diskette está en la diskettera y quieres escribir en él. Si este rayo de luz no pasa, no puedes formatear ni escribir en ese diskette. Para protegerlo debes ponerle una de las etiquetas plateadas o doradas que vienen en las cajas de diskettes. Si no las tienes, puedes usar un pedazo de cinta adhesiva negra o de algún color que no deje pasar la luz.
Si quieres escribir de nuevo en el diskette, simplemente retira las protecciones.
Cuando se fabrican los diskettes, se revisa cada lado para ver si permite una correcta grabación. Si un lado no cumple las normas de confiabilidad, ese diskette será un diskette de un lado. En la mayoría de los casos, el lado no certificado se puede utilizar en el Atari. La única razón por la cual no se usa es que no tiene la ranura para poder escribir en él.

Si le haces una ranura al lado opuesto de la ranura original, podrás escribir en el reverso del diskette. Asegúrate de que la ranura quede a la misma altura de la otra para que puedas grabar. Debes tener en cuenta que al hacer esto pierdes la garantía del diskette y no siempre funciona.
Sabrás si el lado opuesto funciona si es que puedes formatearlo.

7.- PERIFERICOS

Existe gran cantidad de periféricos que agregarle a tu Atari, así que antes de ponerte a gastar plata en hardware para tu equipo, revisa bien tus necesidades. Respondiendo satisfactoriamente a las 4 preguntas siguientes te puede ayudar a evitar la compra de un periférico que solo sirva para acumular polvo.
¿El periférico hace lo que quieres que haga?
La impresora COLLOSO 8000 escribe 8000 caracteres por segundo, pero sólo si el texto es en Japonés; seguramente no la necesitas. Tienes que ser muy específico acerca de lo que quieres que haga un periférico antes de comprarlo.
¿Es compatible el periférico con tu equipo?
Asegurate primero de todo lo que puede conectarse al Atari. Asegúrate también que tengas la memoria necesaria para soportar el periférico. Algunos periféricos necesitan cables de conexión y equipamiento especial antes de instalarlos. Asegúrate de que estén disponibles.
¿Puedes comprar o escribir software para utilizar el periférico?. A menos que tu Atari tenga un programa que te indique como dar ordenes a este extraño aparato, no hará nada. Algunos proveedores venden el software con sus periféricos, otros te permiten escribir tus propios programas o encontrarlos en el mercado.
¿Existe algún Manual claro o algún experto que te asegure que sabrás como usar el periférico?. Nunca olvides que la mayoría de los equipos computacionales son complicados y necesitan instrucciones claras para ser operados apropiadamente. Si tienes un amigo experto o el vendedor lo es, fantástico. Si no, asegurate que entiendes el Manual.
Ahora echemos un vistazo a los productos más populares. Como la industria de los microcomputadores está cambiando constantemente, tienes que estar alerta a los nuevos productos que aparecen en el mercado y los que se discontinúan.
a) Impresoras -> te permiten obtener una "copia dura" en papel, que no se borra cuando apagas el equipo.

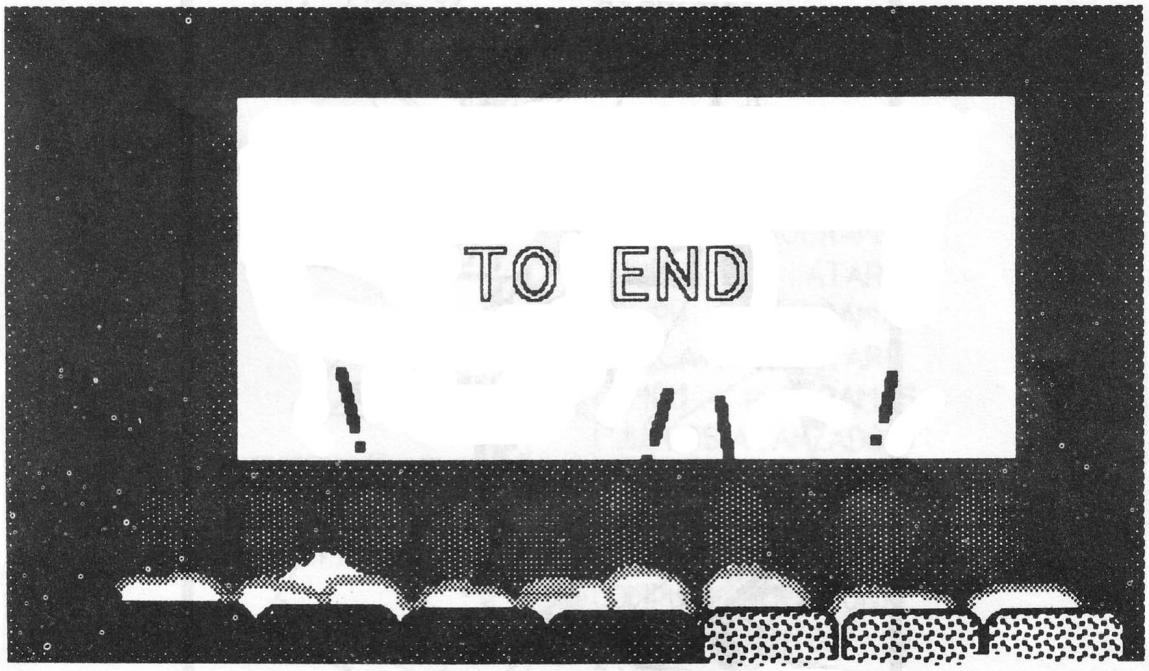
Uno de los primeros aspectos a considerar es la calidad de la impresión. La mayoría de las impresoras son de matriz de puntos: forman los caracteres con un pequeño cabezal con puntas que barren el papel cada vez que imprime una línea. Las puntas chocan con una cinta entintada en una combinación correcta para crear cualquiera que sea el caracter deseado. Mientras más puntas tengan, más complejos y bien formados serán los caracteres que imprime.
Un tipo más económico de impresoras de matriz de punto se llaman impresoras térmicas, que usan un papel especial e imprimen usando puntas calientes que "queman" los caracteres en el papel químico. Su mayor desventaja es que deben usar papel especial y la calidad de la impresión no siempre es buena.
La mejor calidad se obtiene con las impresoras de "margarita" que escriben letras igual que una máquina de escribir. Tienen una pequeña rueda que gira (la margarita) con muchas "patitas" flexibles.
Otros aspectos a considerar son:
- Si es bidireccional (capaz de imprimir de ida y vuelta, lo que la hace mucho más rápida).
- Si permite imprimir 80 columnas (carro angosto) u 132 columnas (carro ancho).
- Si tiene alimentación por fricción (para hojas sueltas), alimentación por tractor (para papel computacional, con "hoyitos") o ambos.
- El tamaño del buffer. El Atari envía el texto a la impresora muy rápido y ésta lo usa lentamente, lo que hace que el computador tenga que esperar. Un buffer es memoria en la impresora que almacena el texto del computador y se lo va entregando a la impresora.
- La capacidad gráfica, etc.

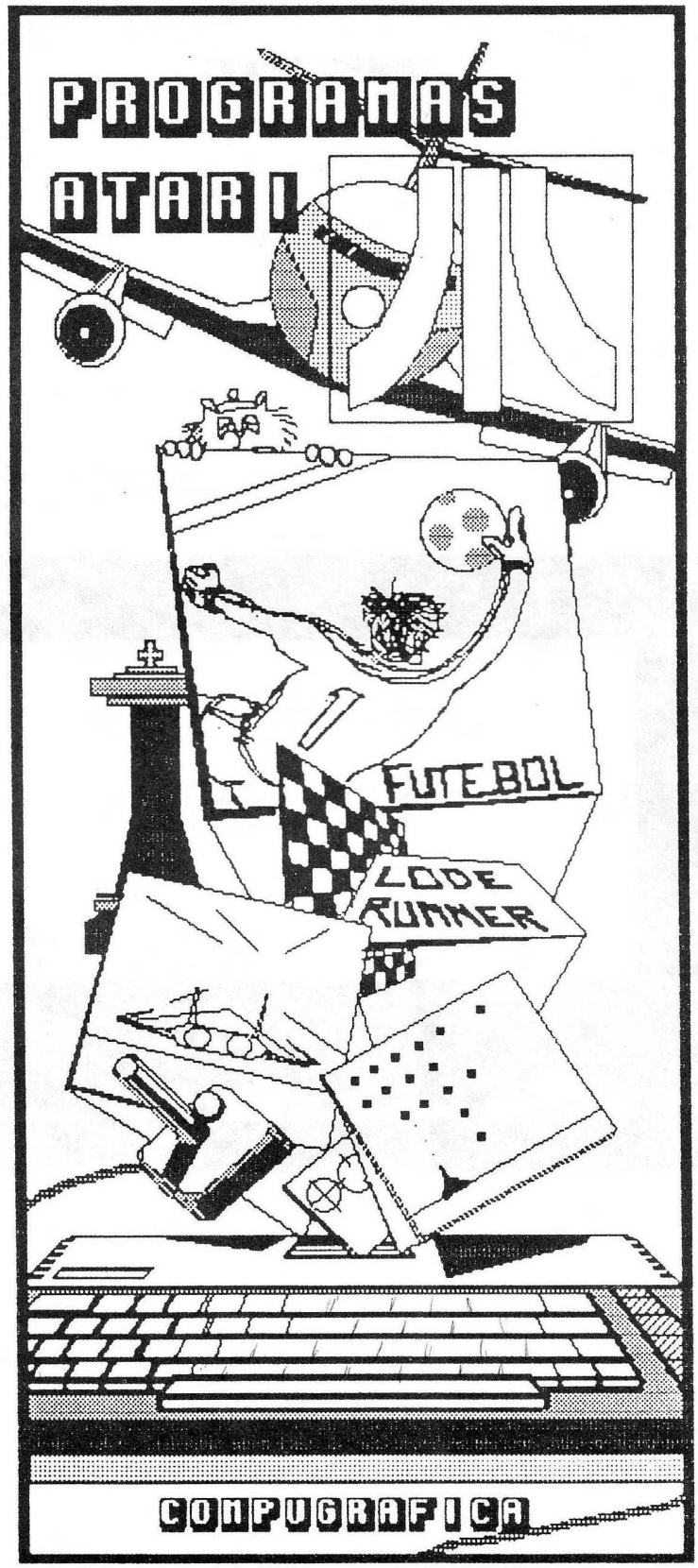
b) Modems -> un modem es un periférico que te permite enviar datos de tu computador vía teléfono y recibir datos de otros computadores.
Sus características más importantes son la velocidad (medida en bps, bits por segundo), la frecuencia de la señal (en baudios), el tipo de transmisión (full duplex o que permite recibir y enviar información a la vez o half duplex, que permite una cosa a la vez).

Otros periféricos importantes son:

- **Controladores para juegos.**
- **lápices luminosos.**
- **plotters, etc.**

INDICE


TO END
! ! !



ANUNCIAMOS LA PROXIMA PUBLICACION DE TRES TEXTOS DE PROGRAMAS ATARI

*** COMPU PROGRAMAS ATARI JUEGOS**

*** COMPU PROGRAMAS ATARI EDUCATIVOS**

*** COMPU PROGRAMAS JUEGOS AVANZADOS**

EDICIONES COMPU GRAFICA

TITULOS PUBLICADOS

1 COMPU DICCIONARIO
2 COMPU BASIC
3 COMPU ESCOLAR
4 COMPU LENGUAJES
5 COMPU PASCAL
6 PROCESAMIENTO DE PALABRAS
7 COMPU BASIC AVANZADO
8 ROBOTS
9 COMPU CURSO
10 COMPU ATARI

EN PRENSA

11 COMPU PROGRAMAS (JUEGOS ATARI)
12 COMPU PROGRAMAS (EDUCATIVOS ATARI)
13 COMPU PROGRAMAS (JUEGOS AVANZADOS)

EN PREPARACION

14 COMPU GRAFICOS
15 COMPU PROFESIONAL
16 PROCESAMIENTO DE DATOS

OTRAS PUBLICACIONES

☐ Salvemos la Vida de un niño del Dr. Profesor
Rodrigo Miranda C.

☐ Siempre en la Luna de Enrique Araya.

☐ VENTAS POR MAYOR Y PEDIDOS DE PROVINCIA

A

C O M P U G R A F I C A

Av. 11 de Septiembre 1480 Ofic. 71
Fono 741278 Casilla 16158 Santiago Chile

Computadoras Atari
Impresoras
Programas Escolares
Literatura
Programas Culturales
Capacitación y Entrenamiento
Boletín Informativo a Domicilio
Diskettera

CENTRO ATARI®

Cursos de Programación
Asesoría Técnica
Cassettera
Información
Programas Profesionales
Programas de
Educación Audiovisuales

Acérquese a su Centro Atari:

Los Centros Atari fueron creados para dar respuestas a cualquier inquietud que usted tenga con respecto al uso y utilidad que brinda su computador.

Para esto, cada Centro Atari cuenta con un equipo humano especializado, que le hará demostraciones para que usted y sus hijos conozcan lo fácil que es operar su computador Atari. Ellos le brindarán apoyo, asesoría y asistencia técnica permanente. Ademas, le entregarán información actualizada de todas las novedades de la computación Atari.

Los Centros Atari cuentan con una amplia biblioteca especializada, a la que usted y sus hijos tienen libre acceso. En ellos encontrará un extenso material de lectura para complementar sus conocimientos.

Algo importante. TODOS ESTOS SERVICIOS SON TOTALMENTE GRATI.

CURSOS DE PERFECCIONAMIENTO

Normalmente, los Centros Atari dictan cursos de programación, y ofrecen diversos seminarios, con el fin de que las familias que ya cuentan con un computador Atari, aprovechen al máximo las posibilidades que éste ofrece y se integren al fascinante mundo de la computación.

Un encuentro con la computación.

ARICA	: Santa María 2119	**TALCA**	: 1 Sur 1639 Edificio Lircay L. 7
IQUIQUE	: Baquedano 898	**CHILLAN**	: Edificio Salman L. 14
ANTOFAGASTA	: Baquedano 477	**CONCEPCION**	: Rengo 281
LA SERENA	: Cordovez 499	**LOS ANGELES**	: Colón 523
VIÑA DEL MAR	: San Martín 545	**TEMUCO**	: Montt 730
VALPARAISO	: Blanco 1131	**VALDIVIA**	: Independencia 555
SAN ANTONIO	: Barros Luco 1413	**OSORNO**	: Eleuterio Ramírez 870
SANTIAGO	: Augusto Leguía Sur 75 Andrés de Fuenzalida 79	**PUERTO MONTT**	: Urmeneta 529
CURICO	: Peña 800	**PUNTA ARENAS**	: Rocca 886 L. 23

COMPUTADORES • PROGRAMAS
ATARI EDUCACIONAL
CENTROS ATARI

CON RESPALDO COELSA COMPUTACION