QL MAGAZINE

Suplemento especial Nov. 1985

Discos: más memoria a menos precio

nuevas utilidades: BCPL, Assembler, Monitor y Toolkit

Introduccion al SuperBasic

ENSAMBLADOR

Es curioso que, en cualquier ordenador personal que se precie, uno de los primeros lenguajes que se implementan sea el ensamblador, cuando es ciertamente uno de los menos y peor usados. Sin embargo, la versión que vamos a comentar llega con un cierto retraso al mercado, ya que existen al menos dos versiones anteriores. A cambio, sale bajo el apoyo de Sinclair Research Ltd., lo que siempre es una garantía de un mínimo número de ventas.

El sistema ensamblador al que nos referimos ha sido desarrollado por **GST Computer Systems** para Sinclair, y se engloba dentro del primer conjunto de aplicaciones comercializadas bajo su nombre para el QL.

Se trata de un ensamblador sencillo, ya que no dispone ni de posibilidad de definición de macros ni de ensamblado condicional, pero, por otra parte, dispone de tres características de interés.

La primera de ellas es el conjunto de comandos para el listado del ensamblador, que permiten con facilidad documentar de un modo estético el desarrollo de un programa, lo que nunca es despreciable.

Otra característica tiene que ver con el uso de librerías de dos tipos. Por un lado, las clásicas librerías de rutinas en ensamblador que se leen desde el código fuente principal y se ensamblan en el momento y que permiten la creación de programas sumamente largos de los que normalmente no hubiera cabido todo el fuente ni en el editor ni en el ensamblador, y por otro lado, las librerías precompiladas, donde podemos disponer de rutinas muy útiles y frecuentemente utilizadas, que no ralentizan el tiempo de ensamblado, ya que ya han sido ensambladas previamente.

La última de estas ventajas es la existencia de mensajes de aviso, además de los de error, ya que sirven para indicar al programador dónde pudiera haber un error por haber otro modo mejor de realizar una operación, pero que no consisten en un error de concepto.

Hay, sin embargo, un gran inconveniente, y es la carencia de definiciones externas, que generan un tipo de archivos que han de ser reprocesados junto con otros del mismo tipo por medio de un programa especial llamado LINKER.

En resumen, se trata de un ensamblador sencillo y, por tanto, es para gente con ideas sencillas que no tenga ganas de complicarse la vida.

HERRAMIENTAS DE TRABAJO

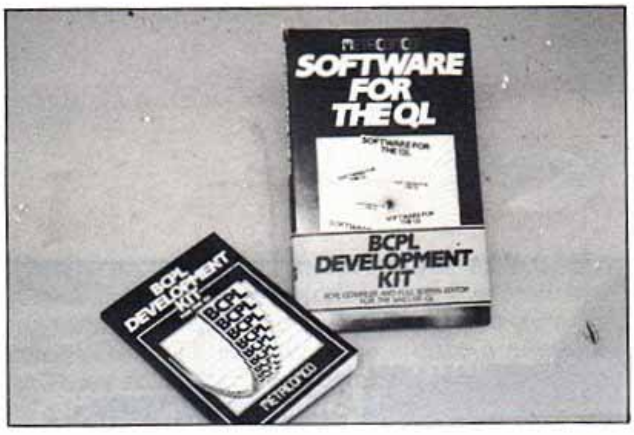
Cuando empezamos a programar en el Superbasic de QL, echamos en falta gran cantidad de comandos y/o utilidades que nos harían la vida más fácil. Como la casa Sinclair, ciertamente, no es tonta, se ha preocupado de lanzar dentro de su paquete comercial inicial un conjunto de rutinas bajo el nombre de **Toolkit,** cuyo principal interés radica en que contiene precisamente todo lo que le falta al QL (o casi todo).

Las rutinas incluidas se pueden dividir en tres bloques. Un primer bloque es el de las extensiones al BASIC. Dentro de éstas cabe destacar el tan deseado editor de pantalla completa, que hasta ahora llevaban todos los ordenadores de precio medio menos el QL. Otros añadidos de interés son los que controlan el funcionamiento en multitarea, añaden el acceso aleatorio a los archivos, copian archivos sin interrumpir al BASIC, cambian de base numérica y otra serie de funciones interesantes.

El segundo bloque está formado por las rutinas de multitarea, entre las que se encuentran varios modelos distintos de relojes, digitales y analógicos, y una alarma horaria (para los que se despistan cuando tienen un teclado delante). Otras rutinas cambian el formato de los archivos, paginándolos o poniéndolos en mayúsculas, por ejemplo.

El último bloque es el formado por varios programas útiles escritos en Superbasic. Unos se encargan de la copia repetida de uno o más archivos a uno o más destinos distintos, otro de la generación de juegos de carácteres nuevos (para el que no guste de los actuales) y otro de la alteración de archivos que hayan sido creados incorrectamente.

Como conclusión, más que un añadido que pudiera interesarnos ponerle al QL, es una prolongación del propio QL que nunca debió ser amputada.



BCPL, LENGUAJE PARA CONNOISEURS

El BCPL, cuyas siglas corresponden al inglés BASIC COMPUTER PROGRAMMING LANGUAGE (Lenguaje Básico de Programación de Ordenadores), no es un lenguaje en absoluto similar al BASIC, nuestro viejo amigo, sino que se trata de una herramienta creada para el desarrollo de sistemas operativos, compiladores e intérpretes de otros lenguajes. Se engloba, pues, en la familia a la que pertenecen los lenguajes P (usado para la compilación de PASCAL) y C (usado en el desarrollo del sistema operativo UNIX), y es, de hecho, su "padre".

La implementación realizada en el QL se puede resumir en tres adjetivos: compacta, completa y elegante.

Se han incluido todos los comandos básicos del lenguaje tal y como se definen en su manual de referencia, incluyendo algunos que quedan obsoletos frente a comandos más completos creados específicamente para el QL, gracias a la gran flexibilidad del QDOS. Entre ellos cabe nombrar aquellos que se refieren al manejo de archivos y de pantalla, que se han reunido en uno solo con un parámetro que es el código de la operación a realizar, lo que evita la necesidad de realizar múltiples bifurcaciones tipo IF para acceder a estos recursos.

Uno de los detalles más interesantes que tiene esta realización del lenguaje es la posibilidad de unir rutinas creadas en código de máquina por medio de un ensamblador con rutinas creadas en BCPL en los puntos donde hiciera falta, y de permitir la creación de OVERLAYS, es decir, de fraccionar el programa de tal modo que se deje más memoria libre para datos, tomándose automáticamente de disco o mocrodrive las partes de programa que se necesiten y no se hallen en memoria en ese momento.

Sin embargo, se echa en falta algún programa de demostración complejo o alguna librería de rutinas interesantes ya grabados en el cartucho, aparte de la librería estándar de entrada/salida del BCPL.

MONITOR PARA EL QL

El monitor de código de máquina para el QL desarrollado por la casa Q-JUMP para Sinclair es de un tipo absolutamente clásico y, tal vez, excesivamente sencillo.

Al invocar al monitor por medio de QMON, aparece una línea encabezada por "Qmon>" en la que podemos escribir diversos comandos. Dichos comandos se pueden englobar en cuatro categorías: de ejecución, de visualización, de modificación y varios.

Los comandos de ejecución son los que tienen la función de supervisar el modo en que se ejecuta un programa o tarea. Tales son la ejecución paso a paso y la inserción de puntos de ruptura (BREAKPOINTS).

Los comandos de visualización muestran en pantalla los contenidos de la memoria o de los registros, o el listado desensamblado de las tareas.

Los comandos de modificación permiten variar a voluntad los contenidos de los registros o de la memoria para ejercer un control extra sobre el funcionamiento de las tareas, o corregirlas sobre la marcha.

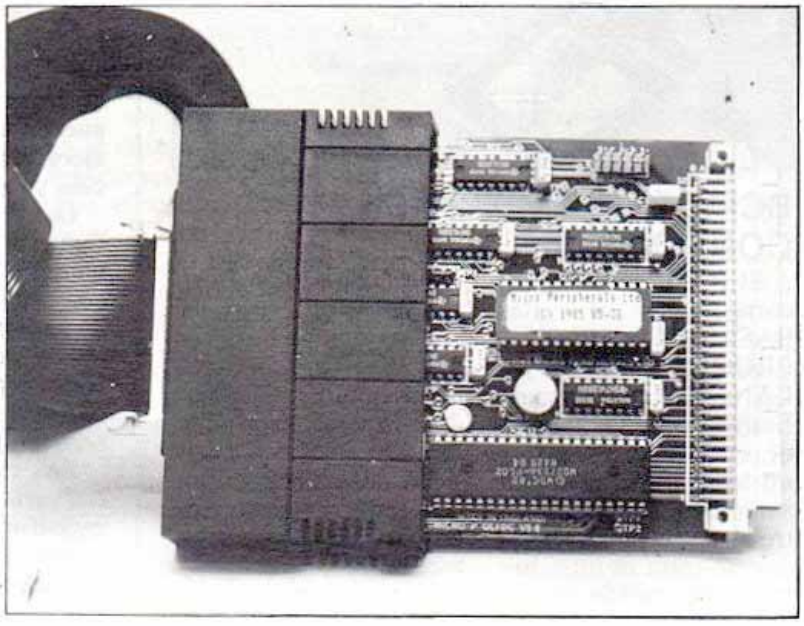
En el capítulo de varios se engloban dos comandos. Uno permite abrir canales para entrada/salida, para evitar tener que volver al BASIC para realizar estas tediosas labores. El otro, permite calcular los valores numéricos de direcciones como composición de los valores contenidos en diversos registros más unas cantidades fijas. Esto es muy útil en una máquina como el QL, en la que todas las tareas en código de máquina han de ser relocalizables y, por tanto, se basan en el direccionamiento relativo.

Como conclusión, este monitor es una herramienta básica en la depuración de programas de código de máquina, pero no pasa de eso.

Discos para el QL

En un ordenador con un procesador tan potente como el 68008, 128 K de memoria y unos programas de gestión excelentes, sólo queda el problema del almacenamiento masivo para que el QL sea una verdadera alternativa a los diversos PCs del mercado. La unidad de MicroPeripherals, comercializada en España por Investrónica, es la primera alternativa en nuestro país.

El sistema operativo del QL dedica buena parte de su código a conseguir un funcionamiento eficiente del *microdrive,* con excelentes resultados. La carga y grabación de programas son bastante rápidas, así como la copia de ficheros. Donde comienza a hacer agua el planteamiento es en el manejo de ficheros. Si bien el procesamiento secuencial de información no plantea grandes problemas, acceder a zonas separadas del fichero continuamente hace que los tiempos de espera aumenten, mientras la cinta se fatiga por el giro constante, planteando problemas de fiabilidad. La solución: añadirle una unidad de disco si queremos utilizar esta máquina en aplicaciones profesionales con acceso a muchos datos.



Un disco muy compacto

Para su capacidad de 720 K, la unidad tiene unas dimensiones verdaderamente pequeñas. El *interface* se introduce en el bus de expansión a la izquierda del teclado, y sobresale apenas 7 centímetros del aparato. Una larga cinta comunica al controlador con la unidad propiamente dicha, de 3" 1/2, 80 pistas y doble cara. A nueve sectores por pista, hace un total de 1.440 sectores (720 K). El controlador maneja hasta cuatro unidades de disco que funcionen con el estándar Shuggart, y tengan alimentación independiente. Por tanto, se pueden utilizar, si disponemos de ellas, viejas unidades de simple o doble cara, en cualquier formato. En cualquier caso, es necesario que sean de doble densidad.

El funcionamiento del *disquette* desde SuperBasic se realiza igual que todos los dispositivos de entrada/salida, referenciándose la nueva unidd como "fdk1_". El *interface* incluye, sin embargo, algunos comandos en ROM que facilitan el uso del nuevo periférico.

Por ejemplo, algunos programas pueden plantear problemas para adaptar su funcionamiento a disco. Además, al estar activos los microdrives y esta unidad, la reserva de memoria para tablas y directorios es mayor. Si algún programa no funciona correctamente en disco, siempre nos queda la opción de incluir el comando MSET, que hace que los *floppies* emulen al *microdrive:* cualquier referencia de nuestro programa a *microdrive* será llevada a cabo por el disco. Esta opción se puede manejar también por *hardware,* mediante un puente del *interface.* En este caso, al no inicializarse los microdrives, la cantidad de memoria ocupada es menor, y desaparecen los problemas que pueden plantear programas excepcionalmente grandes.

El comando MSET tiene su contrapartida en FSET y VSET. El primero sirve para restaurar el nombre del dispositivo a "fdkX_". VSET nos permite elegir el nombre, por ejemplo, "dskX_" o "flpX_". Así se pueden hacer compatibles programas diseñados para otros sistemas de disco.

Otros comandos incluidos son SAVEO, SBYTESO y SEXECO. Los tres realizan la misma función que sus homónimos del SuperBasic, pero no dan

mensaje de error si el fichero existe. Permiten sobreescribir programas sin tener que usar el tedioso DELETE cada vez que deseamos actualizar un programa o fichero.

Para finalizar la lista de extensiones, los comandos DGET y DPUT proporcionan acceso directo a los sectores del disco. Con ellos se puede leer un sector cualquiera del disco a un vector entero de 512 elementos. El uso más obvio es para obtener réplicas de los discos sector a sector, opción conveniente siempre, y absolutamente necesaria para copias de seguridad en sistemas con un solo disco. También sirve para implementar un sistema de acceso aleatorio; para ello hay que saber cómo se organiza el directorio y los ficheros. Las herramientas para ello están, pero la implementación queda a cargo del usuario, lo que hace muy peligroso intentarlo, ya que una escritura en un sector incorrecto puede destruir todo el disco.

Programas de utilidad, para facilitar el manejo

La unidad se completa por un disco de utilidades, que incluyen un módulo ejecutable que acelera las copias entre disco y *microdrive,* ya que, aunque parezca paradójico, la copia de disco a microdrive es más lenta que entre dos cartuchos. Esto se debe a la gestión de bloques esclavos. El operativo utiliza toda la memoria sobrante para almacenar copias de sectores de *microdrives.* En caso de ser necesaria esta memoria, el operativo le indica al controlador que vuelva a escribir el bloque al cartucho, si es que había sido modificado.

Así, cuando copiamos un fichero de cartucho a cartucho, el operativo le busca un hueco en la memoria a la unidad de origen, y le da determinado tiempo para copiar todo lo que pueda de él. A continuación repite la operación en escritura, y es aquí donde se gana el tiempo, ya que cada sector que se escribe debe ser verificado. Para ello, se debe esperar a que la cinta dé una vuelta entera (unos 7 segundos). Al escribir varios sectores de una vez, el operativo no espera para cada sector, sino que hace las operaciones según pasa la cinta por la cabeza. Con el disco, éste devuelve sólo un sector cada vez, por lo que la espera se hace eterna si el fichero es largo. El programa que se proporciona hace las copias en bloques de 32 K, con una velocidd muy grande.

Otros programas permiten posicionar hasta nueve ventanas en cualquier lugar de la pantalla, cambiar una cadena de carácteres por otra en todo un fichero (muy útil para convertir programas de *microdrive* a disco, si se cambia mdv por fdk). DELETE_X sirve para borrados múltiples, y puede borrar hasta un disco (o *microdrive*) entero de una sola instrucción.

Otra de las utilidades permite redireccionar ficheros hacia otro periférico, sin interrumpir la ejecución de nuestro programa. Entre otras cosas, proporciona la posibilidad de un *spooler,* que manda los ficheros a impresora mientras hacemos otro trabajo.

Se completa la serie de programas con un editor de memoria o disco, que sirve para parchear *bytes* en los sectores de un disco o echarle una ojeada a la memoria.

Rápido, fiable y de gran capacidad

El nuevo sistema abre el camino a un uso profesional sin restricciones del QL: con una unidad simple se puede almacenar un libro entero, si trabajamos con QUILL; una gran cantidad de registros, si nuestro objetivo es ARCHIVE, y se mejoran los tiempos de acceso a la información. Para poder utilizar en disco los programas de Psion, es necesario ejecutar el programa config_bas, incluido en el cartucho de ABACUS, y que permite cambiar las unidades por defecto de los programas, las definiciones de impresora y la unidad donde se buscará el fichero de Ayuda.

Por lo demás, cualquiera que haya sufrido grandes esperar mientras se escribía el resultado de una compilación, o se ordenaba la base de datos, notará la diferencia de tiempos. Si el problema eran los malabarismos para conseguir meter un paquete en las 100 K de un cartucho, aquí hay 720. Si nos fatigaba el mensaje "medio incorrecto", los discos no suelen dar este tipo de problemas.

SOBRE EL SUPERBASIC

Cuando Sir Clive Sinclair denominó **SuperBasic** al lenguaje de programación que instaló en su QL, lo hizo por dos razones. La primera de ellas, de tipo comercial, surge de que cualquier producto que ponga la palabra SUPER en su propaganda suele vender más que uno que no lo haga. La segunda razón es más presuntuosa, ya que da a entener que dicho dialecto del **BASIC** es muy superior al lenguaje original y a todos los dialectos aparecidos hasta la fecha.

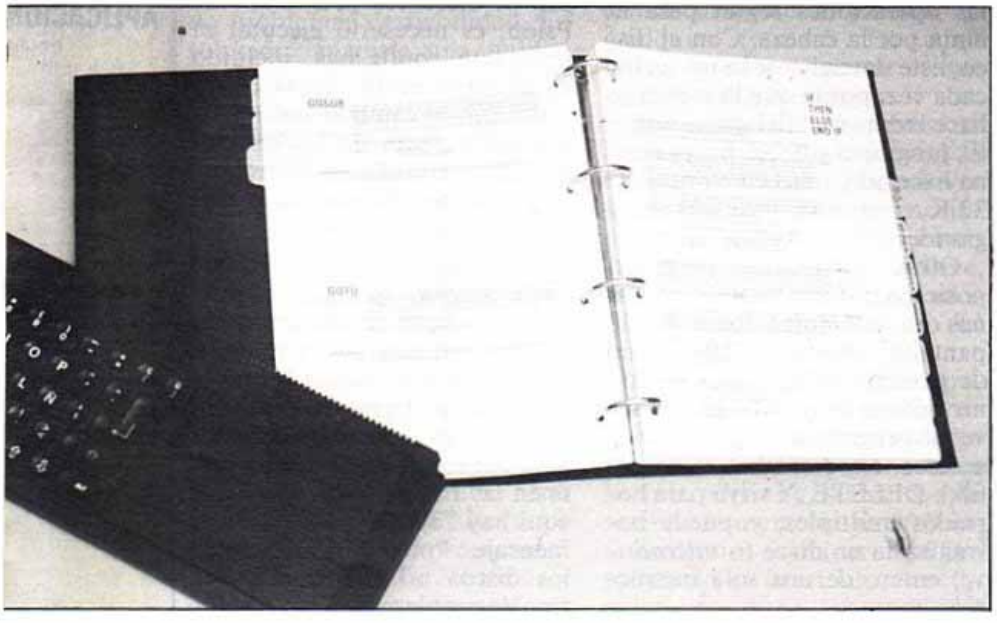
Una traición para arreglar una herejía

En la gran maraña de los lenguajes de programación de alto nivel, se distinguen dos tipos importantes: los lenguajes estructurados y los no estructurados. El representante más conocido del segundo tipo es el **BASIC.** En él existe una gran dependencia del programa con los números de línea en que van asignadas las sentencias. Esto es así por culpa de una carencia de estructuras de control de flujo, que se reducen a **GOTO, GOSUB, FOR...NEXT** e **IF...THEN,** lo que limita las posibilidades de programación.

Por contra, los lenguajes estructurados, de los que el representante más conocido es el **PASCAL,** disponen de más formas de controlar el flujo del programa. Cabe destacar dentro de estas formas los *bucles* con salida condicional **(WHILE...DO, BEGIN...UNTIL),** la definición de secuencias que se pueden ejecutar llamándolas por su nombre **(PROCEDURE, FUNCTION)** y las decisiones múltiples **(CASE...OF).**

Los programadores profesionales suelen renegar de los lenguajes inestructurados, pues no permiten la creación de sistemas flexibles y fácilmente corregibles. De hecho, los consideran una herejía.

El manual del SuperBasic nos abre las puertas de un nuevo lenguaje.



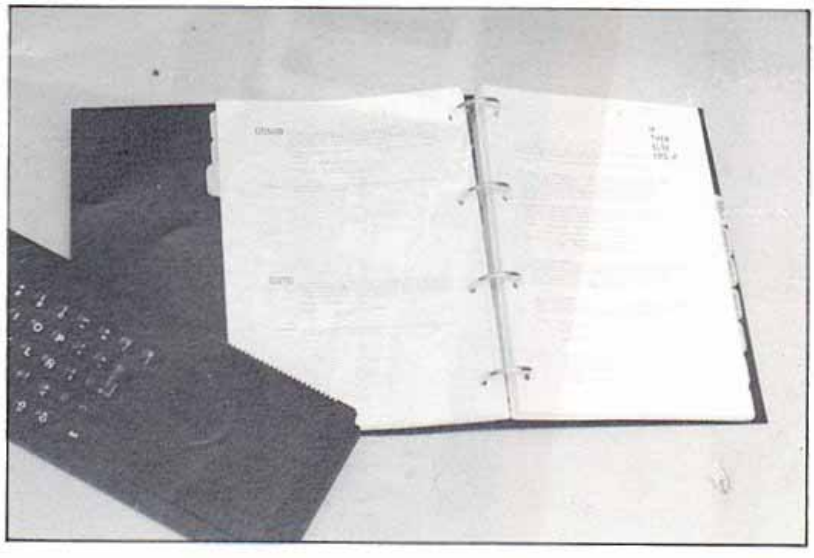
Pero los diseñadores del **BASIC** sabían perfectamente que es muy difícil para un novicio en los ordenadores aprender el manejo de las estructuras, y por eso hicieron un lenguaje más intuitivo, pero menos potente, puesto que los pensaban dedicar a la enseñanza.

Cuando Sinclair Reserch Ltd. presenta su dialecto, en el que se han flexibilizado las instrucciones **FOR...NEXT** y **IF...THEN,** se han añadido las definiciones de proceso **(PROCEDURE)** y función **(FUNCTION),** y se ha incluido la estructura **REPEAT...EXIST...END REPEAT,** lo que ha hecho es traicionar el espíritu de sencillez del **BASIC,** manteniendo y ampliando sus fines didácticos y deshaciendo a medias la herejía tradicional de este lenguaje. Porque, sin perder las estructuras clásicas del lenguaje, añade unas versiones sencillas de las estructuras propias de otros lenguajes que permiten dar el hasta ahora difícil salto entre ambos tipos de programación.

Una bella labor de selección

Uno de los fallos que se puede encontrar en los viejos ordenadores de Sinclair, la familia ZX, es su completa desnudez en caso de producirse un error, que automáticamente devuelve el control al usuario.

En el QL, recogiendo la experiencia de otras marcas, por fin se ha incluido la posibilidad de manejar los errores desde dentro de los programas, sin que se haya de interrumpir necesariamente su funcionamiento. Para ello ha incluido dos estructuras y varias variables asociadas que,



como nota curiosa, hay que destacar que no están documentadas en ningún punto del manual.

Las estructuras son **WHEN EOF...END WHEN** y **WHEN ERR...END WHEN,** a las que se salta en caso de producirse un error por fin de fichero y un error cualquiera, respectivamente. Para poder identificar correctamente la causa del error, se han incluido también dos variables, **err_num** y **err_lin,** cuya función es conocer el código del error y la línea donde se ha producido, respectivamente.

Con esto, podemos programar más tranquilos porque podemos protegernos fácilmente de las típicas multiplicaciones de errores que se producían, por ejemplo, al darse un error y olvidársenos cerrar todos los canales que hubiera abierto el programa.

Un BASIC para manitas

Uno de los más importantes avances realizados con el **SuperBasic** es el simple modo en que se puede ampliar el conjunto de comandos a nuestra disposición. Aparte de la ampliación que puede representar la definición apropiada de procesos y funciones en **BASIC,** disponemos de la posibilidad de añadir procesos y funciones en código de máquina a través de una sola llamada al sistema operativo. A su vez, estos procesos y funciones de ampliación pueden hacer uso de los parámetros que se les pasen y del calculador de punto flotante con gran sencillez, de nuevo a través de unas cuantas llamadas al sistema operativo, todo lo cual está convenientemente documentado en varios libros al respecto. Esta propiedad es la que utilizan los dispositivos como discos y similares para incluir comandos específicos de dichos aparatos que permiten aprovecharse de sus especiales características. También se aprovechan de esto los llamados **"TOOLKITS",** conjuntos de instrucciones destinadas a hacer más fácil (o menos árida) la vida del programador. Una típica aplicación desarrollada en éstos, es el editor de pantalla completa, que permite moverse todo a lo largo de un programa EN SU LISTADO.

Los problemas del SuperBasic

Todas estas ventajas, siguiendo el viejo refrán "no hay mal que por bien no venga", habían de traer consigo una serie de inconvenientes nacidos de la misma naturaleza de dichas ventajas.

Tal es el caso de la velocidad del lenguaje. Cuando uno maneja un lenguaje estructurado y recursivo, se encuentra con el terrible problema de tener que mover arriba y abajo grandes pilas de datos y direcciones de retorno. Cuando se compila el lenguaje, estos movimientos no parecen lentos, pero al tratarse el **SuperBasic** de un lenguaje interpretado, la lentitud producida por ellos es agobiante. Esto no quiere decir que el lenguaje sea lento comparado con otros dialectos, pero sí que se podía haber conseguido que fuera más rápido. En cualquier caso, es mucho más rápido cuando maneja las estructuras que cuando interpreta los **GOTO** y **GOSUB.**

Se trata, pues, de un lenguaje que resulta lento frente a lo que pudiera haberse esperado del procesador que lleva, pero resulta un gran avance en velocidad frente a otros ordenadores de precio, que no características, comparable. Y, especialmente, es un "slato cuántico" **(QL, QUATUM LEAP)** frente a los anteriores aparatos de la casa Sinclair.