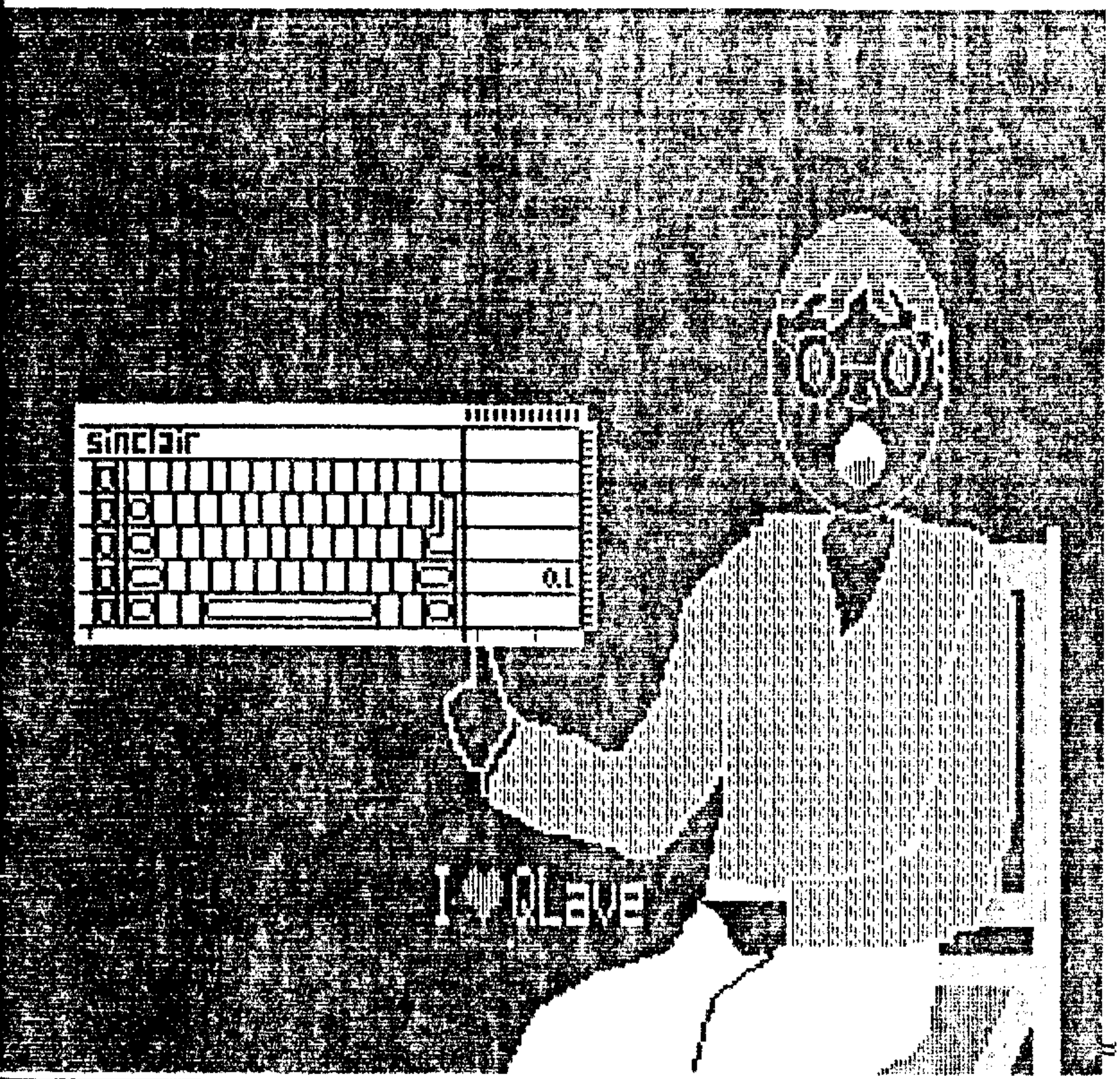
QLAVE

VOLUMEN II NO.2 AGOSTO 1986



INFORMACION SOBRE EL CLUB

La integración en la asociación QLave se hace por suscripción anual o semestral. QLave publica mensualmente el boletín de los socios.

Más información sobre la asociación puede obtenerse desde la secretaría del Club.

Para ser miembro de QLave se requiere estar interesado por el ordenador personal Sinclair QL.

El Club mantiene una librería de software. Una lista de los programas existentes en la librería se publicará de vez en cuando para así ir actualizándola. Los programas que se quieran aportar o sacar de la librería se deben notificar al encargado de la misma.

Presidente	Secretario	Librero
Serafín Olcoz	Juan Palacio	Angel Asín.

La correspondencia debe enviarse a: QLave / Apartado de correos 483 / 50080 - Zaragoza

Contribuciones a QLave

Las contribuciones a QLave deben ser archivis de QUILL en cartuchos de microdrive preferiblemente. Los cartuchos se devuelven a vuelta de correo.

Los programas cuya extensión no sea muy grande se incluirán en la revista, para aquellos de gran extensión pasarán a formar parte de la librería.

Los programas que se envíen deben adjuntar una descripción de los mismos y de su funcionamiento.

==========

Portada de Enrique J. Sarabia

EDITORIAL

Por fin y como esperàbamos desde hace tanto tiempo, ya somos ASOCIACION. No se si la fruta ha caido madura del àrbol o si ha tenido en cuenta la carta que enviè al Ministerio del Interior a madiados del mès pasado interesàndome por la aprobaciòn de los estatutos del C.E.I.U.B.L..

Ya se acabò la "historia interminable" por capitulos sobre la "salud" de los estatutos del club, que mès a mès y capitulo a capitulo os he ido relatando, este ha sido el final del "culebròn":

"La Direcciòn General de Politica Interior, con fecha del 30 de Julio pasado, comunica la Inscripciòn de la Entidad denominada Club Español Independiente de Usuarios del BL".

La asociaciòn ha quedado inscrita en el Registro Nacional de Asociaciones con el nùmero 65.210 y en el Registro Provincial de Zaragoza con el nùmero 1.742.

Ahora hay que convocar y celebrar una Asamblea General Extraordinaria de Asociados al objeto de proceder a la designaciòn de los miembros de la Junta Directiva.

En la ùltima reuniòn de la Junta Gestora celebrada el 12 de Agosto se decidiò publicar los estatutos aprobados por el Gobierno Civil y enviar una copia de ellos a cada uno de los socios asi como convocar para el mès de Septiembre no antes del dia 15 , seguramente el dia 18 en Zaragoza en algùn lugar por determinar todavia, la Asamblea General Extraordinaria.

Ahora se van a presentar en el Gobierno Civil de Zaragoza para su diligenciamiento un Libro de Actas o de acuerdos, un Libro de Caja o de Ingresos y gastos y un Libro de Registro de Socios. Otra cosa que tambièn se va a hacer es abrir una cuenta en un Banco o Caja de Ahorros para asi facilitar la labor de nuestro tesorero.

Sin nada màs que esperar que os alegre la noticia os dejo en la esperanza de poder conocernos en la Asamblea General Extraordinaria.

Serafin Olcoz

COMENTARIO DE PROGRAMAS

Programa: QL Toolkit II 2.06
Editorial: QJUMP
Dirección: 24 King Street Rampton
Cabridge CB4 49D
ENGLAND

Este programa es una versión mejorada y ampliada del conocido Toolkit I. Y se encontrará dentro de poco tiempo en forma de microdrive en el mercado en lugar de su presentación habitual EPROM.

La versión microdrive que es la que poseemos viene con un programa de configuración que es preciso ejecutar de forma que nos aparece en pantalla unas ventanas (ver figura 1) que nos ayudan a seleccionar aquellos de los comandos del toolkit que deseemos implementar en nuestro ordenador. Una vez efectuada la creación de nuestro toolkit particular tenemos a nuestra disposición más de 100 comandos de los que 31 son nuevos respecto al Toolkit I.

Parte de estos comandos permiten la corrección de algunos de los errores introducidos en las ROM de las distintas versiones del QL, así se corrige en la AH y JM el error de los CALL ,los WHEN ERROR de la MG y JS y el POINT de la MG.

En esta versión se hace extensivo el sistema de directorio jerarquizado a todos los comandos que implican manejo de ficheros. Se mantienen los viejos DATA_USE, PROG_USE, SPL_USE y se incluye DEST_USE de forma que nos referimos al directorio en el que nos encontremos para programas, datos o destino en los comandos que lo requieran.

El concepto de directorio jerarquizado se concreta por el uso de las 'extensiones' así 'mdv1_' es un directorio, también lo es 'mdv1_usr_pedro_', un fichero lo será 'mdv1_usr_pedro_renum' los comandos DDOWN, DUP, DNEXT, DLIST nos permiten movernos por esta estructura así si estoy en 'mdv1_' puedo decir 'DDOWN usr_pedro_' de modo que he bajado por la estructura de árbol desde mdv1_, por usr_, hasta pedro_ y ahora todos los demás comandos de manejo de ficheros se centrarán únicamente en los programas que estén en este directorio. DATAD$, PROGD$, DESTD$ nos informan dónde estamos.

A los conocidos WDIR(obtención de directorio), WSTAT(obtención de longitud de prog...),WDEL(borrado de grupos de ficheros) se les une WCOPY que permite copiar todos los ficheros del directorio en curso, COPY_H que copia el fichero y su cabecera de forma separada y COPY_O que sobregraba. A SPL_ se le añade SPLF_ que manda un (FF) (form feed a la impresora) al final de la copia del fichero. WREN que renombra todos los ficheros cambiando solo el directorio donde se encuentren, de esta forma se pueden trasladar fácilmente todos los ficheros de un directorio a otro.

Se ha reescrito para que acepten directorios por defecto LOAD, LRUN, MERGE, MRUN, SAVE, SAVE_O (salva sobregrabando si el fichero ya existe), LRESPR (reserva memoria y

carga en ella), CALL , SBYTES, SBYTES_(sobregraba), SEXEC, SEXEC_(sobregraba), EXEC y EXEC_W (corresponden a las EX, EW del antiguo Toolkit. Se ha incluido el comando DO que permite ejecutar un fichero de comandos; asi si hemos salvado un fichero llamado ventana y contiene: open #4,scr_:paper #4,0:ink #4,2:cls #4. Toda esta serie de comandos serán ejecutados al teclear o incluir en un programa DO ventana.

El control de Jobs se sigue realizando con JOBS(lista los jobs en curso), RJOB(para liberar un job), SPJOB(colocar prioridad), AJOB(activar un job), PJOB(localiza la prioridad de un job), OJOB(localiza el propietario del job), JOB$(da el nombre del job), NXJOB(encuentra el siguiente job en el àrbol de jobs, recordar que los jobs se organizan internamente como padres e hijos, de modo que un padre puede tener varios hijos y estos también pueden tenerlos, organizàndose todo en una estructura de àrbol).

A los comandos OPEN se une el OPEN_OVER (sobregrabar) y a las funciones FTEST que nos devuelve el valor del mensaje de error consiguiente al intentar abrir un fichero, funciones que informan de ficheros longitud, tipo, fecha de grabación. A los comandos de acceso aleatorio de ficheros, BGET, BPUT, GET, PUT, FPOS, se les alade FLUSH limpia los buffers de manejo de ficheros.

Se ha incluido en esta versión el comando PRINT_USING que es una versión formateada del comando PRINT, por ejemplo si asigno f='-#.#### !!!!\' y tecleo PRINT_USING f,1234567, el resultado será:- 1.2356 E 06. Además tenemos todas las demàs funciones de conversión a decimal, exponencial, binario o hexadecimal.

En lo que se refiere a la salida por pantalla CURSEN activa el cursor y CURDIS lo desactiva. CHAR_USE permite seleccionar un conjunto de caracteres definidos por el usuario para asociarlos a un determinado canal. CHAR_INC nos permite dar un incremento en pixels de los caracteres sacados por un determinado canal, el incremento que se consigue es màs controlado que el de CSIZE.

En lo que se refiere a la asignación de bloques de memoria del QL es posible hacerlo de una manera eficiente mediante el 'common heap', asi ALCHP nos reserva espacio en esta área de memoria devolviendo la base del área. RECHP devuelve el espacio reservado al área común (common heap). CLCHP libera no sólo un área sino todas la reservas pertenecientes a ésta que hayamos realizado. El hacer unas grandes reservas de espacio en la zona de área común y posteriormente acceder al microdrive en primer lugar puede provocar lo que se llama 'gran escala de fragmentación' en éste área haciendo imposible la carga de programas una solución aunque peligrosa viene dada por el comando DEL_DEF que borra los bloques de definición de la 'common heap'.

Respecto a los procedimientos de Superbasic se han incluido PARTYP que nos da el tipo del parámetro usado (nulo, cadena, coma flotante, entero) y PARUSE que indica el uso que

no le da (ninguno, variable, matrices).

Se ha incluido un reloj como procedimiento residente y una alarma es posible incluir texto en la ventana del reloj. Asi CLOCK #6, 'QL time %h:%m'.

Es de destacar en el aspecto de las extensiones al teclado que pulsando ALT & ENTER es posible recuperar el último comando que se tecleó siempre y cuando la longitud de la linea a introducir no sea excesiva. Además es posible asignar a la combinación ALT & OTRA TECLA una serie de comandos, de esta forma ALTKEY 'r','RJOB''SPL''',''' hará que el comando 'RJOB''SPL''' se ejecute al pulsar ALT & R.

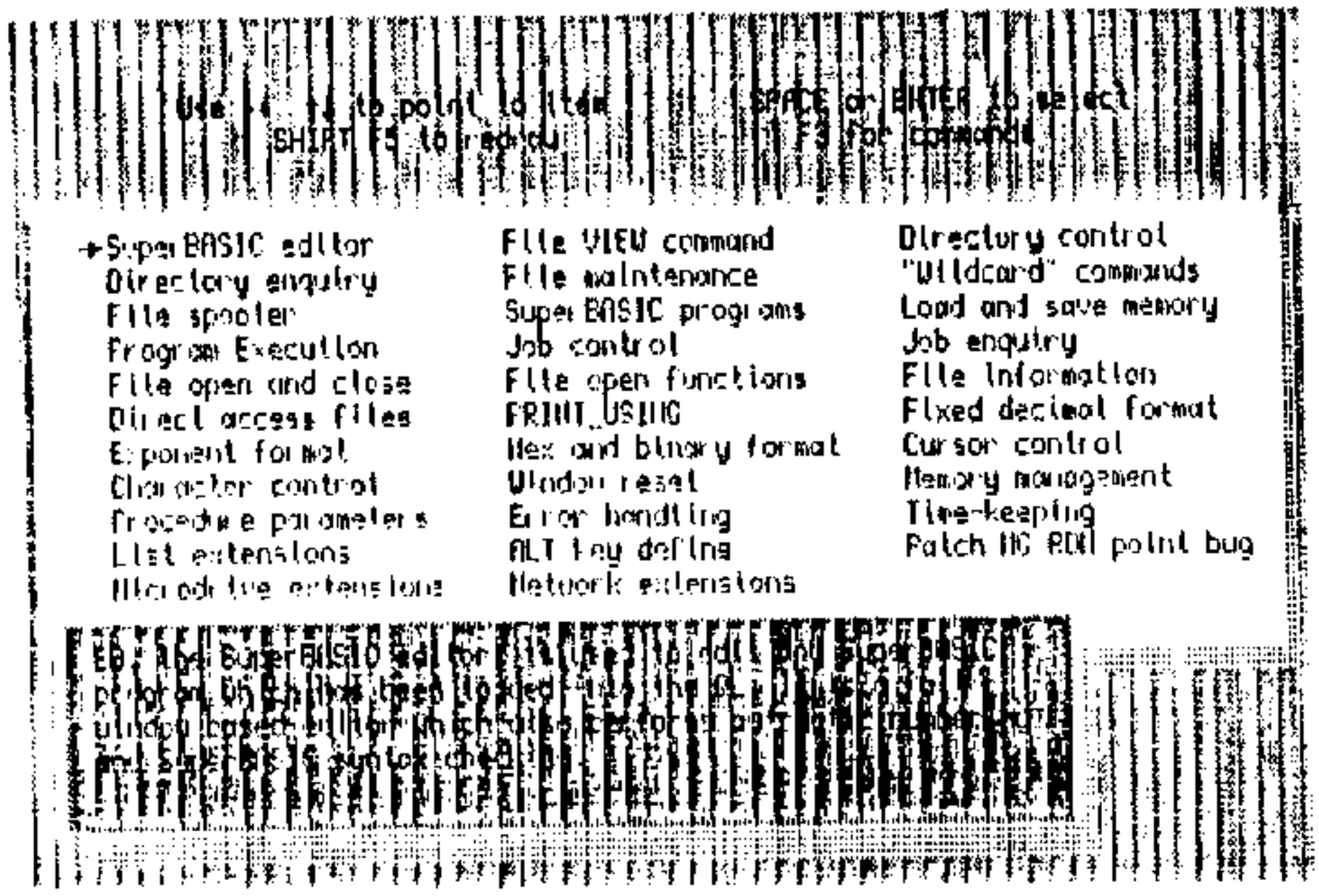
Se han escrito nuevos protocolos más completos para el manejo de la red local del QL. Asi el comando FSERVE permite a varios QL acceder a una misma unidad de discos. NFS_USE permite redefinir los periféricos a los que nos referimos desde la net asi NFS_USE mdv,n2_flp1,n2_flp2_ , hace que las referencias al mdv1_ en este QL se refieran al flp1_ del QL num. 2 de net y de igual forma con el mdv2_ y flp2_. De igual forma el mandar mensajes es enormemente sencillo.

En toolkitII de igual forma que en el primero es posible escribir programas en ensamblador que el comando EX o EW acepta como especiales y que permiten la conexión de pipes entre ellos en su ejecución. Para ello es necesario incluir cabeceras especiales en los programas que deseemos se ejecuten asi.

Junto con el programa se entrega la documentación necesaria para ejecutar correctamente todos los comandos descritos.

En conjunto pues un programa que amplia de forma definitiva el QDOS a una situación que es la que debiera haber tenido desde el momento de su lanzamiento.

QLave



PREGUNTAS

1.- Aunque sigo vuestro curso de código máquina quisiera que me aconsejáseis una publicación en castellano sobre el 68000 y otra sobr el 8DOS. Si no hay en castellano informadme de alguna en inglès o francès.

2.- Tengo el ensamblador de MetaComCo con los manuales en inglès. Quisiera saber si es el mejor del mercado y si existe un desensamblador para el QL.

3.- Si teneis los manuales en castellano de este ensamblador me gustaria que me hicièseis fotocopia que os pagaria a vuelta de correo.

4.- Quisiera saber exactamente qué es un monitor y que diferencia hay con un ensamblador.

Juan Santiago Vila
Marbella

==========

Respecto a una publicación en castellano sobre el 68000 podemos recomendarte MICROPROCESADORES DE 16 BITS de José Maria Angulo Usategui editado en Paraninfo. Respecto al 8DOS PROGRAMACION AVANZADA de Adrian Dickens editado por RA-MA. En lengua inglesa el Manual de Referencia editado por Motorola y 68000 Assembly Language Programing de MacGraw Hill.

El ensamblador de MetaComCo es posiblemente de los más potentes del mercado si bien tiene el inconveniente de ocupar demasiado espacio tanto el editor como el ensamblador y es preciso poseer una ampliación de memoria para obtener programas de más de 2K. En este mismo número se ofrece una descripción exhaustiva de su funcionamiento, que esperamos despeje la mayoria de las dudas sobre su funcionamiento.

Existen muchos desensambladores para el QL, entre los mejores se encuentran el CIMON de Computer One y el QMON de Tony Tebby cuyo comentario es posible encontrar en otro número anterior.

Un monitor está intimamente ligado a los desensambladores, ya que una virtud común de los monitores es desensamblar(obtener una traducción a mnemónicos del programa en código máquina), asi como permitir la ejecución en modo Trace(paso a paso) de la instrucciones en ensamblador, de igual forma se obtienen listados en ASCII y Hexadecimal de la memoria del ordenador. Todo esto es habitual y normalmente incluyen mucho más. La finalidad del ensamblador es exactamente la contraria es decir escribimos en mnemónicos y el programa se encarga de pasarlo a código máquina.

QLave

Yo tengo la memoria ampliada y los discos de MicroPeripherals. Algunos programas, por ejemplo ABACUS no funcionan con la ampliación, si alguno de vosotros lo sabe me gustaría que me dijerais como solucionarlo.

En el último boletín (Vol II n- 1) viene un programa de backup yo diría que imprescindible para cualquier usuario, que lee el número aleatorio de un cartucho y hace una copia de seguridad de los programas que vienen protegidos con ese sistema. Yo tengo varios programas en cartuchos que tienen esa protección y quisiera pasarlos a disco. He probado vuestro programa sin discos ni ampliación y funciona muy bien, pero cuando intento hacer una copia en disco, con o sin la ampliación no me funciona. Quizá podais aclararme el porqué.

Por último he observado que con algunas impresoras (SEIKOSHA 1000) se producen fallos de impresión esporádicos cuando se corre un programa en código máquina, bien sea QUILL, ARCHIVE o bien se trate de un programa compilado. No sé si la solución es blindar el cable pero como bien sabeis el conector de la impresora es imposible de encontrar en España como no sea comprando un adaptador de joystick (2700 pts.). Si creeis que hay otra solución por favor publicadla.

José Morón
Sevilla

==========

Lamentamos tener pocas respuestas a tus preguntas y esperamos que si algún otro socio lo sabe nos escriba al respecto.

En lo que se refiere a tu primera pregunta efectivamente ABACUS funciona incorrectamente con ampliación de memoria ahora bien nosotros lo hemos hecho con una ampliación comentada en este mismo número y con la EPROM que adjuntan y el comando MULTI es posible hacerlo funcionar. No obstante hemos escrito a PSION al respecto así como respecto a las últimas versiones de sus programas esperamos poder decirte algo al respecto pronto.

No hemos tenido oportunidad de probar el programa copiador en disco ya que mucha de la gente de aquí está de vacaciones no obstante esperamos tener una respuesta el próximo mes.

Respecto a los fallos eporádicos con la SEIKOSHA no te podemos decir mucho ya que las que hemos usado exahustivamente es la ADMATE y ésta no ofrece problemas. Nos parecen extraños tus problemas con ella.

QLave.

ENSAMBLADOR DE METACOMCO

Numerosos socios nos han escrito indicando su interès en conocer màs a fondo el ensamblador de MetaComCo bien sea por problemas de familiaridad con este tipo de programas o por la inexistencia de manuales en castellano de los mismos. Aqui vamos a intentar tocar todos los puntos necesarios para trabajar con èste programa.

Consta de tres partes fundamentales: editor, ensamblador y lincador.

El editor es el programa que nos permite introducir el texto del programa que deseamos realizar y que posteriormente serà compilado por el ensamblador. De forma que comenzaremos ejecutando èste programa: 'exec mdv1_ed' al haber usado este modo de ejecuciòn podemos tener ejecutàndose otros programas al mismo tiempo y para pasar desde el basic al programa 'ed' tendremos que pulsar 'CTRL + C'. Si hubiesemos deseado no tener nada màs ejecutàndose en tal caso habriamos tecleado 'exec_w mdv1_ed'.

Nada màs comenzar el programa se nos hacen unas preguntas; la primera hace referencia al nombre del fichero que pretendemos editar de forma que responderemos con el nombre que deseemos dar al fichero, por ejemplo 'mdv2_mifichero'. En caso de que èste fichero no exista en el microdrive aparece un mensaje en la parte inferior de la pantalla que indica que se comienza un nuevo fichero. En el caso de que hayamos tecleado erròneamente el nombre del fichero debemos responder las demàs preguntas y esperar a que se nos devuelva el control, teclear 'F3' y despuès 'R/mdv2_mifichero/' dònde mifichero es el correcto.

La segunda pregunta se refiere al espacio de trabajo que vayamos a usar,en caso de que no respondamos nada y pulsemos 'ENTER' se asignan 12K. La tercera pregunta indica si alteramos la ventana de trabajo o no. Una vez dentro del editor podemos acceder a dos tipos de comandos, inmediatos y extendidos. Los inmediatos se obtienen pulsando directamente las teclas, los extendidos a tavès de 'F3'.

COMANDOS INMEDIATOS

F2	Repite el ùltimo comando extendido pulsado.
F3	Entra en modo extendido.
F4	Redibuja la pantalla.
IZDA	Mueve el cursor a la izda.
SHIFT+IZDA	Mueve el cursor a la palabra anterior.
ALT+IZDA	Mueve el cursor al comienzo de la linea.
CTRL+IZDA	Borra el caracter a la izda.
CTRL+ALT+IZDA	Borra linea
DCHA	Mueve el cursor a la derecha.
SHIFT+DCHA	Mueve el cursor al comienzo de la siguiente palabra.
ALT+DCHA	Mueve el cursor al final de la linea.
CTRL+DCHA	Borra el caracter de la derecha.
CTRL+ALT+DCHA	Borra hasta el final de la linea.

SHIFT+CTRL+DCHA	Borra palabra a la derecha.
ARRIBA	Mueve el cursor hacia arriba.
SHIFT+ARRIBA	Mueve el cursor a la parte superior de la pantalla.
ALT+ARRIBA	Scroll hacia arriba.
ABAJO	Mueve el cursor hacia abajo.
SHIFT+ABAJO	Mueve el cursor a la parte inferior de la pantalla.
ALT+ABAJO	Scroll hacia abajo.
CTRL+ABAJO	Inserta una linea en blanco.

COMANDOS EXTENDIDOS

BS	Marca el comienzo de un bloque en el cursor.
BE	Marca el fin de un bloque en el cursor.
A/@Lave/	Inserta una linea que contiene '@Lave' tras la actual.
I/@Lave/	Inserta una linea que contiene '@Lave' antes de la actual.
IF/mdv1_@Lave/	Inserta un fichero llamado 'mdv1_@Lave'.
IB	Inserta una copia de un bloque.
B	Nos lleva hasta la parte inferior del fichero en curso.
T	Nos lleva a la parte superior del fichero en curso.
CE	Mueve el cursor al final de la linea.
CL	Mueve el cursor una posición a la izda.
CR	Mueve el cursor una posición a la dcha.
CS	Mueve el cursor al comienzo de la linea.
M n	Mueve el cursor a la linea n.
N	Mueve el cursor al comienzo de la linea siguiente.
P	Mueve el cursor al comienzo de la linea anterior.
D	Borra la linea en curso.
DC	Borra el caracter sobre el que está el cursor.
DB	Borra un bloque.
BF	Va a la linea anterior.
E/yo/tu/	Cambia yo por tu.
E@/yo/tu/	Cambia yo por tu pero pregunta primero.
F/yo/	Busca la cadena yo.
LC	Distinguirá entre mayúsculas y minúsculas al buscar.
UC	No distingue entre mayúsculas y minúsculas al buscar.

S	Rompe la linea en la posición del cursor.
J	Junta la linea actual con la siguiente.
SB	Muestra el bloque actual en pantalla.
SH	Muestra página de información.
SL 1	Coloca a 1 el margen izquierdo.
SR 80	Coloca a 80 el margen derecho.
ST 5	Coloca la distancia entre tabuladores a 5.
EX	Extiende margen derecho.
WB/mdvi_f/	Escribe el bloque marcado en el fichero 'mdvi_f'.
SA/mdvi_f/	Salva el texto en el fichero 'mdvi_f'.
Q	Salir del editor sin salvar el texto.
X	Salir del editor salvando el texto con el nombre con el que se entró.
R/mdvi_g/	Reentrar en el editor con el fichero 'mdvi_g'.
RP	Repetir hasta que se produzca un error.
U	No hacer cambios en la linea actual.

Se puede observar que con los comandos extendidos es posible realizar todo lo que con los inmediatos y algo más, esto es asi ya que podemos agrupar los comandos extendidos en una sola linea y evitar asi el tedioso teclear de forma repetitiva algunas opciones.

Por ejemplo supongamos que deseamos rellenar todo el fichero de la palabra clave EQU, en lugar de escribirla constantemente podemos hacer en la linea de comandos que aparece tras pulsar F3:

```
RP A/EQU/
```

De igual forma es posible repetir un comando un número concreto de veces, asi si deseo sólo 10 EQU:

```
10 A/EQU/
```

Y esto puede aplicarse a todos los comandos arriba expuestos.

Una vez tecleado el programa en ensamblador y tras haberlo salvado en un fichero, podemos comenzar la fase de compilación que pasará los mnemónicos a código máquina propiamente dicho. Ejemplos de ficheros fuente para teclear con el editor se han publicado

en numerosos números de QLave y pueden teclearse directamente, ya que todos ellos se realizaron con este editor y compilador.

Para ejecutar el compilador es preciso teclear 'exec mdv1_asm' o 'exec_w mdv1_asm'.

Al iniciarse el programa se nos hacen las siguientes preguntas:

En primer lugar debemos indicar cual es el nombre del fichero que anteriormente salvamos y que contiene el programa. Después debemos indicar si deseamos listado del programa ensamblado, si pulsamos ENTER se asume que no; si hemos respondido 'Y' entonces se nos pregunta nombre del fichero al que se dirige el listado, si se pulsa ENTER se dirige a la pantalla, podemos pues decir ser1, mdv2_... En este listado aparecen las lineas que contienen algún error con una E por delante y con signos + las partes que se refieran a macros.

Posteriormente se nos pregunta por el nombre del fichero que contendrá el código objeto. Si tecleamos ENTER no se genera código y el compilado sólo nos servirá para comprobar si hemos cometido errores en la redacción del programa.

Se nos pregunta también por el espacio de trabajo del programa si se pulsa ENTER se toma el reservado por defecto. De igual forma se nos interroga sobre la alteración del tamaño de la ventana de forma que un ENTER o N impliquen dejarla igual. La alteración de tamaño es por ALT+cursores y su situación se controla por los cursores sólo.

Es preciso mantener siempre dos microdrives introducidos, uno con el programa y otro con el fichero a compilar ya que el programa carga constantemente extensiones de si mismo para realizar las distintas fases del compilado. De entrada los programas vienen instalados para trabajar en el mdv1_ y los ficheros a compilar en el mdv2_ esto es alterable por el fichero install de forma que aquellos que dispongan de una ampliación de memoria puedan cargar los programas en disco ram y dirigir los ficheros compilados ahí luego bastará un copy a un mdv_ o disco del compilado definitivo.

Dependiendo del fichero que hayamos deseado compilar se nos generará un código independiente de la dirección de carga o no así si no hemos incluido ORG por lo general será independiente y ejecutable vía EXEC que suele ser lo deseable, en caso contrario el programa se ejecuta con CALL tras cargarlo con LBYTES.

Al final se nos pregunta si deseamos compilar más programas debemos responder Y o N.

Los mnemónicos a usar son los que se nos han mostrado en el curso de introducción al código máquina pero existen algunas características de cada compilador que es preciso tener en cuenta:

Así a la hora de teclear el programa en el editor es preciso hacerlo en el siguiente orden:

(Etiqueta) Mnemónico (Operandos) (Comentarios)

En ensamblador no existen números de linea por lo que no podemos decir algo similar a GOTO ni tampoco tenemos las estructuras REPEAT o DO WHILE, de forma que los ciclos se Etiquetan, así si quiero ir a una subrutina tecleare BSR DELTA donde DELTA es un nombre

que le hemos asignado a la rutina y que está escrita antes de que ésta comience.

Las etiquetas sólo se colocan pues donde sea necesario, al comienzo de una rutina o de un ciclo del que se sale con alguna condición. Por este motivo aparece en la línea superior entre paréntesis ya que es opcional, pero si no la tecleamos el Mnemónico debe teclearse separado de la columna 0 por lo menos un espacio, ya que todo lo que esté en la 0 se considera etiqueta.

En las columnas siguientes se colocarán los operandos necesarios dejando un espacio con la instrucción y de igual forma un espacio con el comentario que se desee introducir.

Las instrucciones pueden llevar los siguientes especificadores cuando sean necesarios:

.B	indica dato de un byte de longitud(8 bits).
.W	indica dato de una palabra de longitud(16 bits).
.L	indica dato de una palabra larga de longitud(32 bits) o especifica un salto largo.
.S	indica un salto corto.

Así por ejemplo podemos encontrarnos BRA.S o BRA.L. En las expresiones en que sea necesario el compilador tomará .W por defecto.

Existen los siguientes operadores en expresiones que pueden colocarse entre paréntesis.

"-"	Signo negativo.
"((" y "))"	Desplazamiento de bits en un byte o palabra a la izda o a la decha.
"&" y "!"	And y Or.
"*" y "/"	Multiplicación y División.
"+" y "-"	Adición y sustracción.

Los números se introducirán en decimal de forma directa. Ej: 1234.
En Hexadecimal con el símbolo $ precediéndolos. Ej: $89AB.
En Binario con el símbolo % precediéndolos. Ej: %10110111.
Cadenas ASCII se colocan entre comillas. Ej: 'CLave'.

Los modos de direccionamiento tienen el mismo formato que el indicado en el curso de código máquina. Indicar que las referencias relativas al contador del programa no precisan la referencia explícita con (PC).

Ej: LEA SCR,A1

Donde SCR es una etiqueta colocada en el programa en curso.

Así mismo se aceptan como palabras reservadas: USP, CCR, SR. Ej: MOVE A0,USP.

Existen una serie de comandos aparte de los usuales y descritos en el curso y se

colocan en la misma posición que éstos en las líneas de programa es decir tras las etiquetas en caso de que las haya.

ORG $30000 Indica origen absoluto del programa o los datos en $30000. Será preciso cargar lo etiquetado así con LBYTES.

RORG 0 Indica origen relativo del programa, de forma que el contador de programa es colocado a este valor y las instrucciones que le siguen se sitúan de forma relativa a éste. Si no indicamos nada es como si hubiesemos colocado este RORG 0 con lo que el PC tiene el valor 0 inicialmente. Los programas que contienen éste son ejecutables vía EXEC. Habitualmente no se suele colocar ORG ni RORG con lo que el programa es ejecutable tanto vía EXEC como por CALL(previo LBYTES).

SIZE 1000 Indica el espacio de datos a resevar como interno al programa, recordar que todo programa ejecutable por EXEC debe de contener espacio necesario para sus datos y el stack. Si no se incluye se supone un espacio de 500 octetos.

END Se coloca para indicar el fin del programa, si no se coloca genera un mensaje de tipo Warning al finalizar el compilado.

EQU Asigna el valor de la expresión del campo de perandos a la etiqueta colocada a la instrucción, en el ejemplo 12000 a CASA la asignación es permanente y la expresión no debe contener referencias a otras zonas del programa. Ej: CASA EQU 12000.

EQUR Permite asignar a un nombre creado por nosotros uno de los registros del procesador, no es posible renombrar SR, CCR, USP. La asignación es permanente y no es redefinible en ninguna otra zona del programa. En el ejemplo se asigna el nombre DATO a D0. Ej: DATO EQUR D0.

SET Permite asignar a una etiqueta una expresión de igual forma que lo hace EQU con la diferencia de que el valor puede ser alterado a lo largo del programa. La expresión no deberá contener referencias a otras zonas del programa. Ej: CASA SET 12000. Asigna 12000 a CASA.

DC Define valores constantes en un programa, pueden tener un número indefinido de operandos separados por comas. Son semejantes a los DATAS de un programa BASIC. Es preciso asignarles la especificación de Byte, Palabra o Palabra larga. Se les suele colocar una etiqueta de forma que una referencia a este espacio de datos desde el programa se hace por medio de la etiqueta.
Ej: CASA DC.B 6,7,8,9,4

```
MESA    DC.W   240,300
ASA     DC.L   70000
```

De esta forma si sitúo el registro A0 en CASA, con LEA CASA,A0 , la celdilla de memoria a la que apunta A0 contendrá 6, la celdilla a la que apunta A0+1 7, A0+2 8...De igual modo si hago LEA MESA,A0 , la celdilla a la que apunta A0 contendrá 240, A0+2 contendrá 300.

DS — Permite reservar zonas de memoria vacías que posteriormente serán rellenadas con datos. Igual que las anteriores las referencias a las mismas se realizan vía etiquetas.

```
Ej: CASA  DS.B  5  Reserva 5 octetos de memoria.
    MESA  DS.W  4  Reserva 4 palabras( 8 octetos ).
    ASA   DS.L  2  Reserva 2 palabras largas ( 8 octetos ).
```

PAGE — Avanza el listado en ensamblador hasta la siguiente página. Este comando no aparece en los listados generados.

LIST — Hace que a partir de donde esté los comandos sean listados. Por defecto se genera siempre listado.

NOLIST — Desactiva el listado de forma que los comandos que le siguen no son listados hasta que aparezca LIST.

SPC — Hace que aparezcan las líneas en blanco que se deseen en un listado. Ej: SPC 10 . Si deseamos 10 líneas en blanco.

NOPAGE — Desactiva el avance de página y la colocación de cabeceras en los listados.

LLEN — Coloca la longitud de una línea en listado de ensamblador, la longitud debe estar entre 60 y 132. El defecto es 132.

Ej: LLEN 90

PLEN — Coloca la longitud de la página de listado. El valor debe estar entre 24 y 100. El defecto es 60.

Ej: PLEN 72

TTL — Asigna una cadena como título de un programa. Será reproducida cada avance de página. La cadena no deberá tener más de 40 caracteres.

Ej: TTL Programa en ensamblador.

NOOBJ Desactiva la generaciòn de còdigo objeto hasta el fin del programa aun cuando se haya indicado un fichero cuando se nos interroga al comenzar la ejecuciòn del compilador.

FAIL Genera un error de usuario.

CNOP Permite que una seccion de còdigo sea alineada a un limite determinado. En particular permitirla que una estructura de datos se alinee a un limite de palabra larga.

Ej: CNOP 0,4 alinea el còdigo al limite de la prôxima palabra larga.

CNOP 2,4 alinea el còdigo a un limite de palabra de 2 bytes antes de la màs cercana alineaciòn de palabra larga.

IFEQ expresiòn

IFNE expresiòn Permiten activar o desactivar el ensamblado dependiendo del valor de la expresiòn. Asi si el operando es cero(EQUAL), IFEQ activa el ensamblado y si no lo fuese lo desactivarla. IFNE se comporta de modo contrario.

ENDC Marca el fin de un ensamblado condicional de las caraceristicas mencionadas arriba.

MACRO Marca el comienzo de una macro-definiciòn, ejemplos de ellas se pueden ver en cualquier nùmero anterior de QLave. Contienen un conjunto de instrucciones que se ejecutan cada vez que se la llama tecleando la etiqueta que se coloca a su comienzo. Son semejantes a las funciones y procedimientos del SuperBasic. De igual forma admiten parànetros. Para indicar un parànetro se coloca el simbolo 'Ñ' seguido por el nùmero de parànetros que llevemos usados; 1 o 2 o...Dentro de una macro-definiciòn no se puede comenzar la definiciòn de otra macro, es preciso definirlas separadamente y es posible que una macro use otras macros ahora bien hay un limite en el nivel de anidaciòn que se puede usar, èste limite es de diez.

ENDM Indica el fin de la macro-definiciòn.

MEXIT Permite forzar la salida de una macro durante su ejecuciòn. Se suele usar conjuntamente con IFEQ o IFNE.

XDEF Nos ayuda a crear ficheros que puedan ser lincados entre si por el lincador incluido en el paquete. Asi podemos encontrarnos en un fichero:

XDEF DECIMAL,SIST,EV

Esto nos indica que este fichero contiene las definiciones de tres subrutinas que podràn ser usadas por otros programas. DECIMAL,SIST,EV son las etiquetas que habremos

colocado al comienzo de las subrutinas.

XREF Es el comando que debemos incluir en el caso de que en un fichero vayamos a usar subrutinas que no estén escritas en el programa y que se incluirán más tarde al linkar éste fichero con el contenga estas rutinas. Asi pues si en un fichero fuésemos a usar las rutinas antes definidas en la linea anterior que están en otro fichero tecleariа: XREF DECIMAL,SIST,EV

GET Permite insertar ficheros externos en el fichero fuente actual. Ver que esto es completamente distinto a las referencias anteriores ya que aqui se inserta código fuente mientras que antes linkábamos código objeto. Este comando es útil para insertar ficheros con EQU o MACROS que sean muy usuales. De hecho hemos incluido dos ficheros en librería que contienen los EQU de los mnemonicos de los TRAP y de las variables del GDOS tal y como aparecen referenciados en el libro de Adrian Dickens. De modo que en sucesivos listados de ensamblador aparecerán rederencias a estos ficheros en lugar de incluir largas listas de EQU en los ficheros fuente.

Ej: GET 'MDV1_TRAP_EQU'

Decir también que son admitidos de igual modo los mnemónicos estándar de Motorola que se refieren a saltos condicionales..

CC	Guito acarreo	CS	Pongo acarreo	LS	Menor o igual	LT	Menor que
EQ	Igual	F	Falso	MI	Menos	NE	No igual
GE	Mayor o igual	GT	Mayor que	PL	Más	T	Cierto
HI	Mayor	LE	Menor o igual	VC	Sin desbordam.	VS	Desbordamiento

Ej: BNE VUELTA Donde VUELTA es una etiqueta.

El paquete suministrado por MetaComCo se completa con un programa INSTALL que permite alterar el tamaño de la ventana en la que se comenzará a trabajar. También permite alterar el nombre del programa con vistas a su identificación como job independiente, este nombre es posible leerlo con aplicaciones como el Toolkit de Tony Tebby. También es posible alterar el medio de almacenamiento que vaya a usar el programa, asi si usamos Floppys entonces al indicar en la alteración del fichero ASM que FLP1_ es el medio, los overlays se cargarán de éste. De igual modo podriamos indicar un disco Ram.

El último fichero que se incluye es el linkador al que hemos hecho referencia al describir los comandos XREF y XDEF. Para ejecutarlo haremos como siempre EXEC MDV1_LINKER o EXEC_W MDV1_LINKER. Se nos comenzará interrogando por el espacio de trabajo, el minimo es 1K si tecleo ENTER se asignan por defecto 10K. Posteriormente se nos va interrogando por los nombres de los ficheros a linkar si respondemos ENTER indicamos que se puede comenzar el proceso ya que no introduciremos más nombres. Al finalizar se nos pregunta por el nombre del fichero que se va a generar y su stack.

TARJETA DE EXPANSION RAM

Se trata de una expansiòn de memoria interna de 512 K bytes fabricada por S.P.E.M. de Guido Masoero (Torino, Italia).

Dicha expansiòn deja libre la puerta izquierda del QL , de modo que no crea ningùn tipo de problema a la hora de instalar el interface del floppy disk. Tambièn tiene la ventaja de no presentar un aspecto antiestètico debido a que se trata de un ampliaciòn interna y por tanto no es visible desde el exterior.

La memoria que queda libre una vez instalada es de 577 K bytes y las ventajas que el tener tal capacidad de RAM representan no creo que sea necesario mencionarlas por ser obvias e incluso necesarias y esto ùltimo lo puede comprobar cualquiera que trabaje con matrices de grandes dimensiones o con el editor y el ensamblador de metacomco o con el compilador de digital o cargando documentos largos con Quill etc, etc.

Por el contrario frente a los 128 K bytes originales presenta la desventaja de que tarda algùn segundo màs en hacer un RESET y en aparecer el conocido menù : F1 Monitor, F2 Televisiòn.

La presentaciòn de la expansiòn es una tarjeta de buen acabado y fàcil montaje sobre todo si se trata de un QL cuya ISSUE es 5 ya que en ese caso el montaje se limita a destornillar el teclado, levantar con cuidado la ULA que està junto a la CPU y en su lugar instalar la tarjeta y sobre ella nuevamente la ULA asi como levantar de su zòcalo el 68008 e instalar ahi el zòcalo que cuelga de la tarjeta de expansiòn y sobre èl instalar nuevamente el 68008. Como se puede ver es cosa muy simple y de unos cinco minutos màs o menos.

Si por el contrario se trata de un QL cuya PCB es ISSUE 6 o 7 ademàs de lo anterior hay que cortar una pista y realizar un empalme , todo ello muy bièn explicado en las instrucciones y de fàcil realizaciòn. Por tanto se puede decir que en cualquier caso el montaje es muy sencillo y ràpido y que no es necesario en absoluto tener ningùn conciniento ni llevar el QL a ningùn especialista ya que lo puede hacer cualquiera siguiendo las instrucciones al pie de la letra.

Una vez realizado el montaje ya no hay nada màs que hacer para que funcione perfectamente el QL con la expansiòn.

Junto con la expansiòn propiamente dicha, S.P.E.M. presenta una EPROM para aprovechar ampliamente las posibilidades del QL expandido.

Vamos a dar un breve repaso a los comandos y funciones que se amplian:

Como era de esperar se pude acceder por medio de una fuciòn a la cantidad de memoria libre en cada momento, pero esta no es la mejor ventaja ni la màs esperada quizà si lo sea la psibilidad de usar la ampliaciòn como disco RAM con un formato a elegir por el usuario ùnicamente limitado en el nùmero de sectores por la cantidad de memoria libre (cada K byte de memoria son dos sectores), obviamente.

Otra de las mejores ventajas es el poder usar por medio del comando MULTI cuatro jobs a la vez, por ejemplo los cuatro programas del paquete de Psion (Quill, Abacus, Easel y Archive), pudiendo acceder a cada uno de ellos simplemente seleccionàndolo en un menù y sin la tediosa tarea de abandonar y cargar cada uno de ellos cada vez; sòlo recordar que antes de abandonar los cuatro al dejar MULTI hay que grabar los datos ya que si no el trabajo seria en balde.

Por supuesto que en el software hay un comando para la copia de pantalla a impresora, en escala de grises, y tambièn la posibilidad de que se trabaje por defecto con el floppy o el microdrive seleccionado.

Tambièn se puede modificar la velocidad de ejecuciòn hasta en tres niveles diferentes. Asi como tambièn el que un programa BASIC muy largo se almacene y cargue en un, o desde un, microdrive de un sòlo tiròn aunque el nuevo programa resulta ilegible al estar codificado y tambièn ocupa màs espacio, evita la tediosa espera de carga en programas largos (supongo que esta espera os serà màs o menos conocida).

Ah!, no hay que preocuparse si es indispensable el uso de otra extendidisimas EPROMS: I.C.E. y TOOLKIT 2, ya que al desarrollar el software se ha pensado en su compatibilidad y se prentan dos programas para poder parsar ambos, I.C.E. y TOOLKIT 2 de la EPROM a microdrive y asi poder emplearlos conjuntamente.

Tambièn presenta la " Super Expansiòn " un set de caracteres cursivo que funciona perfectamente aunque sòlo en modo 4 y con Csize 1,1.

Otro de los comandos màs òtiles es el que permite definir hasta 20 teclas de funciòn, por medio de F1,F2,F3,F4 y F5 solas y junto con SHIFT o ALT o ambas a la vez. Tambièn es de utilidad el comando que refleja en pantalla el contenido de todas esas teclas de funciòn en cualquier momento y por el canal deseado.

Por òltimo se presenta un problema , se trata de la imposibilidad de emplear RESPR para reservar memoria en presencia de un job en memoria. Pero tal problema està superado ya que la reserva de memoria se puede hacer por medio de la funciòn ALCHP y es màs por medio de la funciòn RECHP se puede liberar el àrea de memoria reservada .

Este ha sido un repaso general de las posibilidades del software de la EPROM junto con el uso de la tarjeta de expansiòn.

Sobre todo destacaria la sencillez del montaje de la tarjeta de expansiòn, la utilidad de todos los comandos y funciones en general pero sobre todo del Disco RAM y de la posibilidad de poder trabajar con Quill, Abacus, Easel y Archive simultàneamente, en particular.

Por supuesto que todos los demàs comandos y funciones son de gran utilidad pero concretamente me llaman la atenciòn los destacados arriba. Asi como tambièn es cierto que no digo nada especial sobre tener libres 577 K bytes porque supongo que todos sabréis apreciar la diferencia y las posibilidades que ello supone.

Para finalizar si tuviese que puntuar la expansiòn de 0 a 10 le daria un 9 quizà ya que siempre se pueden establecer mejoras, por ejemplo la presentaciòn de la EPROM.

Glave

TRUCOS

Los que han trabajado con programas como el GL-PAINT encuentran una opción que permite invertir un bloque definido. Despues de invertir por la mitad zonas de la pantalla se consiguen curiosos efectos parecidos al caleidoscopio.

El siguiente progrma permite invertir una pantalla almacenada en una zona reservada con:

```
a=resspr(32768):lbytes mdvl_dibujo,a
```

Listado1

```
10 FOR g=127 TO 0 STEP -1
20 FOR f=128 TO 256 STEP 4
22 POKE_L 130944+f+(127-g)*256,PEEK_L (a+f+g*256)
24 POKE_L 131072+f+(127-g)*256,PEEK_L (a+f-128+g*256)
26 END FOR f
30 END FOR g
```

Si deseamos invertir zonas determinadas en vez de toda la pantalla,bastará con sustituir en la linea 20 el 0 por la coordenada Y superior y el 127 por la coor denada Y inferior; en la linea 30 el 128 por la X de la izquierda y el 256 por la X de la derecha. Si se desea mas resolución cambiar los pokes_L y los peeks_L por POKE y PEEK eliminando el Step 4 de la linea 30 (esto hace perder algo de velocidad). Si se superan los màrgenes se puede quedar colgado.

Este programa funciona en mode 4 y en mode 8.

CHIS

Un problema que se presenta al utilizar distintas impresoras es la compatibilidad de còdigos de control entre las mismas. Los programas a la venta intentan remediar este problema añadiendo un programa Install a sus programas, esto es suficiente para las letras pero se queda corto en gràficos.

Un claro ejemplo de esto es el programa Tasprint que distribuye Abaco, el programa es muy bueno y ofrece distintos tipos de letras,desde la clásica a la palaciega, pasando por la futurista en sus distintas versiones inversa, recuadrada, subrayada etc.

Pese a ofrecer en su menu diez impresoras distintas, hay una impresora que necesita de este programa porque solo tiene un tipo de letra, y resulta imposible de utilizar.

Para ello se ofrece el siguiente programa que adapta el código suministrado por Tasprint para ser usado en una Seikosha GP 700 ,o cualquier otro tipo de impresora que use un código decimal para definir la longitud de un bloque gráfico (Tasprint envia un número de dos bits en forma de MSB-LSB).

Listado 2

```
10 CLS:INPUT "Fichero "|fi$
20 OPEN_IN #4,fi$
30 OPEN #3,ser:
40 INPUT #4,a$
50 adapta a$:bi$=resul$
60 REPeat bucle
70 IF EOF (#4) THEN EXIT bucle
80 INPUT #4,a$
90 adapta a$:b2$=resul$
100 IF pos THEN PRINT #3,bi$|:ELSE bi$=bi$&CHR$(0):PRINT #3,bi$|
110 bi$=b2$
120 END REPeat bucle
130 CLOSE#3:CLOSE#4
140 DEFine PROCedure adapta (f$)
150 f=LEN(f$):pos=0
160 FOR g=1 TO f
170 IF CODE(f$(g))=27 THEN
180  IF CODE(f$(g+1))=75 THEN pos=g:EXIT g
190 END IF
200 END FOR g
210 IF pos()0 THEN
220  cifra$=CODE(f$(pos+2))+256&CODE(f$(pos+3)):can=LEN(cifra$)
230  IF can=3 THEN cen=cifra$(1):dec=cifra$(2):uni=cifra$(3)
240  IF can=2 THEN cen=0:dec=cifra$(1):uni=cifra$(2)
250  IF can=1 THEN cen=0:dec=0:uni=cifra$
260  f$=f$(1 TO pos+1)&CHR$(cen)&CHR$(dec)&CHR$(uni)&f$(pos+3 TO)
270 END IF
280 resul$=f$
290 END DEFine adapta
```

Para usar este programa debemos obligar al Tasprint a que nos salve en el microdrive el código que va a enviar a la impresora, según el manual esto es posible contestando a la confirmación de dispositivo de salida con Mdv2_archivo, una vez hecho esto abandonamos Tasprint, cargamos el programa adaptador y a disfrutar de letras grandes y hermosas.

El programa consta de un bucle que va cargando bloques del documento a imprimir , y un procedure . Las lineas 160 a 200 examinan el bloque y buscan los códigos de impresión gràfica (27 75) si los encuentra deja en pos la posición de estos carácteres y las lineas 220 a 270 convierten los dos números que les siguen en tres números devolviendo el bloque arreglado en resul$.

El intercambio de las lineas 90 a 110 controlan la falta del caracter 10 que es enviado como salto de linea dentro de los bloques gràficos,sustituyèndolo por el caracter 0 cuando hace falta.

Los códigos a introducir en el programa install para la GP 700 son:

Especiado linea normal: 27,66,13
Especiado gràfico: 27,69,27,84,1,1
Bit Image: 13,27,75
Carac por linea: 64

Byte mas sig: I
Cabezal: 8
Bit Image: 2

CNIB

La rutina que viene a continuación permite utilizar una zona de memoria como archivo de pantalla intercambiando la pantalla normal (SCR1) con la reservada (SCR2) y viceversa.

Para utilizarla reservaremos una zona con a=respr(32820) y cargaremos la rutina con lbytes mdv1_cambia_52,a ,si deseamos cargar inicialmente algún dibujo lo haremos con lbytes mdv1_dibujo,a+52.

Ahora solo queda llamar con call a,a para copiar de SCR2 a SCR1, o llamar con call a,a+26 para copiar de SCR1 a SCR2. Funciona en mode 4 y mode 8

El siguiente programa nos grabará la rutina Cambia_52 en caso de no tener ensamblador.

Listado3

```
10 CLS:RESTORE :x=0
20 a=RESPR(52)
30 FOR g=0 TO 51 STEP 2
40 READ n
50 POKE_W a+g,n
60 x=x+n
70 END FOR g
80 IF x<>187517 THEN PRINT "Hay un data erroneo":STOP
90 SBYTES mdvl_cambia_52,a,52
100 DATA -11652,52,8769,8316,2,0,13372,8192,8409,20938,-4,17024,20085
110 DATA -11652,52,8769,8316,2,0,13372,8192,8920,20938,-4,17024,20085
```

Si por el contario disponeis de ensamblador,podreis comprobar con el programa fuente que en realidad es una rutina repetida.

Listado4

```
        ADD.W   #52,D1          Incrementa D1 dando el inicio de SCR2
        MOVE.L  D1,A1           Apunta a A1 al inicio de SCR2
        MOVE.L  #131072,A0      Apunta a A0 al inicio de SCR1
        MOVE.W  #8192,D2        Inicia el contador
GRA     MOVE.L  (A1)+,(A0)+     Carga en SCR1 una palabra larga de SCR2
        DBRA    D2,GRA          Controla el contador
        CLR.L   D0
        RTS                     Vuelve al Superbasic
        ADD.W   #52,D1
        MOVE.L  D1,A1
        MOVE.L  #131072,A0
        MOVE.W  #8192,D2
REP     MOVE.L  (A0)+,(A1)+     Carga en SCR2 una palabra larga de SCR1
        DBRA    D2,REP
        CLR.L   D0
        RTS
        END
```

RECUPERACION DE LA ULTIMA LINEA DE COMANDO

Con esta rutina se pretende recuperar el contenido de la última linea de comando que fue introducida en el ordenador. La recuperación se produce al pulsar TAB.

El programa debe ser ejecutado con EXEC MDV1_RECUP y el ensamblado se supone realizado con el ensamblador de MetaComCo. Existe una copia en librería para aquellos que no deseen introducir el texto en ensamblador.

La rutina comienza en la parte etiquetada como inicio donde se realiza la conexión de dos rutinas una de planificador y otra de polling. La conexión de estas rutinas se realiza de la forma habitual, A1 apunta al comienzo de la rutina, A0 se refiere a la zona de conexión con la lista de interrupciones.

La rutina de polling es usada como conexión con la linea de comandos del SuperBasic es por esto por lo que se usa la variable del QDOS que apunta al comienzo del Superbasic, es la llamada en el libro de Dickens SV_BASIC, $28010 que aqui se ha llamado BASICSP. En la rutina se hace un desplazamiento de $168 con lo que nos situamos en la zona de memoria intermedia ya que hemos saltado la zona de definición del job 0 y las variables del Superbasic que ocupan $100 octetos. Ahora estamos en condiciones de recoger aquellas lineas de comando que se tecleen desde el Superbasic y llevarlas a la zona etiquetada como MEMINT.

La tecla TAB es fácilmente alterable si os interesa otra, la definición de fila y columna de la tecla escogida está en la zona etiquetada como COMANDO, y se testea en la sexta linea tras comenzar RUTINA1. En caso de haber pulsado, se traslada el contenido que se almacenó en MEMINT a la zona de la memoria reservada para la cola de teclado donde está aquello que se muestra en la pantalla en cada momento. La variable del QDOS que apunta a esta zona es SV_KEYQ, su dirección $2804C y en el programa se le ha llamado TECLADO.

```
*
*
*
MT_LSCHD        EQU             $1E
MT_SUSJB        EQU             $08
IO_OPEN         EQU             $01
IO_CLOSE        EQU             $02
IO_SSTRG        EQU             $07
MT_IPCOM        EQU             $11
MT_LPOLL        EQU             $1C
*
BASICSP         EQU             $28010
TECLADO         EQU             $2804C
*
* MACROS
*
QDOS            MACRO
                MOVEQ           #\1,D0
                TRAP            #\2
                ENDM
*
* INICIO
*
INICIO
                LEA.L           RUTINA1,A1
                LEA.L           MEMORIA1,A0
                MOVE.L          A1,4(A0)
                QDOS            MT_Lschd,1
                LEA.L           MEMINT,A6
                MOVE.B          #$A,(A6)
                LEA.L           RUTINA2,A1
                LEA.L           MEMORIA2,A0
                MOVE.L          A1,4(A0)
                QDOS            MT_Lpoll,1
EXIT
                moveq           #-1,d1
                moveq           #-1,d3
                qdos            mt_susjb,1
                RTS
```

```
MEMORIA1        DS.L            4
MEMORIA2        DS.L            4
*
RUTINA1
                MOVEM.L         A0-A6,-(A7)
                MOVEM.L         D0-D7,-(A7)

                CLR.I           D1
                LEA.L           COMANDO,A3
                @DOS            MT_IPCON,1
                BTST            #3,D1
                BNE             COMIENZO
                LEA.L           MEM1,A4
                MOVE.L          #-1,(A4)
                BRA             FIN
COMIENZO
                LEA.L           MEM1,A0
                MOVE.L          (A0),D2
                CMP.L           #-1,D2
                BNE             FIN
                MOVE.L          #0,(A0)
*
                MOVE.L          TECLADO,A5
*
*
                ADDA.L          #$4,A5
                MOVE.L          (A5),A2
                ADDA.L          #$4,A5
                MOVE.L          (A5),A3
                LEA.L           MEMINT,A4
BUCLE4
                MOVE.L          A2,D0
                CMP.L           A3,D0
                BEQ             AJUSTAR
```

```
RETURN
                MOVE.B          (A4),D7
                CMP.B           #$A,D7
                BEQ             FIN
                MOVE.B          (A4)+,(A3)+
                MOVE.L          A3,(A5)
                BRA             BUCLE4


AJUSTAR
                MOVE.L          A5,A3
                ADDA.L          #$8,A3
                BRA.S           RETURN
FIN
                MOVEM.L         (A7)+,D0-D7
                MOVEM.L         (A7)+,A0-A6
                RTS
*
* RUTINA 2
*


RUTINA2
                MOVEM.L         A0-A6,-(A7)
                MOVEM.L         D0-D7,-(A7)
                MOVE.L          #$2808B,A1
                MOVE.B          (A1),D2
                CMP.B           #$A,D2
                BEQ.S           ENTER
                LEA.L           MEM2,A0
                MOVE.B          #$FF,(A0)
                BRA.S           FIN
```

```
*
ENTER
                        LEA.L           MEM2,A0
                        MOVE.B          (A0),D1
                        CMP.B           #$FF,D1
                        BNE             FIN
                        MOVE.B          #$0,(A0)
                        MOVEA.L         $28010,A3
                        ADDA.L          #$168,A3
                        MOVE.L          #500,D2
                        LEA.L           MEMINT,A6
                        MOVE.L          A6,A4
BUCLE9
                        MOVE.B          (A3),(A6)+
                        MOVE.B          (A3)+,D3
                        CMP.B           #$A,D3
                        BEQ             FIN
                        DBRA            D2,BUCLE9
                        MOVE.B          #$A,(A4)
                        BRA             FIN
*
*
*
*
*
COMANDO                 DC.B            9,1
                        DC.L            0
                        DC.B            5,2
MEM1                    DS.L            4
MEM2                    DS.L            4
MEMINT
                        DS.B            510
                        END
```

Javier Boira.

Esta rutina permite realizar un espejo vertical de la pantalla. En primer lugar la tenemos en ensamblador y en segundo lugar en un programa basic en el que el código máquina viene en datas.

```
        ADD.W   #32640,D1     Incrementa D1 dando el final de SCR2
        MOVE.L  D1,A1         Apunta a A1 al final de SCR2
        MOVE.L  #131072,A0    Apunta a A0 al inicio de SCR1
        MOVE.W  #127,D2       Pepara el contador de filas
HOR     MOVE.W  #63,D3
MSB     MOVE.W  (A1)+,(A0)+   Copia fila impar
        DBRA    D3,MSB
        MOVE.W  #63,D3
        SUBA.L  #256,A1
LSB     MOVE.W  (A1)+,(A0)+   Copia fila par
        DBRA    D3,LSB
        SUBA.L  #256,A1
        DBRA    D2,HOR
        CLR.L   D0
        RTS                   Vuelve al Superbasic
        END
```

```
100 CLS:RESTORE :a=RESPR(54):prueba=0
110 FOR g=0 TO 54 STEP 2
120 READ x
130 prueba=prueba+x
140 POKE_W a+g,x
150 END FOR g
160 IF prueba<>149569 THEN PRINT "Hay un error en los datas":STOP
170 SBYTES mdvl_vertical_cde,a,56
180 DATA -11652,32640,8769,8316,2,0,13372,127,13884,63,12505,20939

    ,-4,13884,63,-27652,0,256,12505,20939,-4,-27652,0,256,20938,-34,17024,20085
```

INTRODUCCION AL CODIGO MAQUINA

-ANDI AL CCR

Realiza la operaciòn lògica AND entre un dato inmediato y los còdigos de condiciòn del registro de estado. Almacena el resultado en el byte de menos peso del registro de estado.

Unicamente admite operandos de 1 byte.

Ej:

```
ANDI #$AA,CCR
```

-- Alteraciòn de los flags:

'N' Pasa a 0 si el bit 3º del dato inmediato es 0.
'Z' Pasa a 0 si el bit 2º del dato inmediato es 0.
'V' Pasa a 1 si el bit 1º del dato inmediato es 0.
'C' Pasa a 0 si el bit 0 del dato inmediato es 0.
'X' Pasa a 0 si el bit 4 del dato inmediato es 0.

En el caso contrario al indicado para cada bit, èste no varia.

-ANDI AL SR

Al igual que en el caso anterior, realiza una operaciòn AND entre un dato inmediato (sòlo admite tamaño de 16 bits) y el resgistro de estado (los 16 bits de dicho registro).

Almacena el resultado en el registro de estado.

Obviamente esta instrucciòn es privilegiada y sòlo puede hacerse en modo supervisor.

Los flags varian de igual forma que con "ANDI AL CCR".

-OR

Resulta ya fácil predecir la función de esta instrucción: Realiza una operación lógica OR entre el operando fuente y el operando destino, almacenando el resultado en el destino.

Los datos pueden direccionarse de dos formas distintas:

- Si el operando destino es un registro de datos, los modos de direccionamiento posibles son:

INSTRUCCION OR

Modos Posibles	op. fuente	op destino
Reg. datos directo	X	X
Reg. direcc. directo		
Reg. direcc. indirecto	X	
Reg. indirecto con postincr.	X	
Reg. indirecto con predecr.	X	
Reg. indirecto con desplaz.	X	
Reg. indirecto indexado.	X	
Absoluto largo	X	
Absoluto corto	X	
Relat. cont. programa con despl.	X	
Relat. cont. programa indexado.	X	
Inmediato.	X	

- También puede direccionarse esta instrucción, siendo el operando fuente el contenido en un registro de datos:

INSTRUCCION OR

Modos Posibles	op. fuente	op destino
Reg. datos directo	X	X
Reg. direcc. directo		
Reg. direcc. indirecto	X	
Reg. indirecto con postincr.	X	
Reg. indirecto con predecr.	X	
Reg. indirecto con desplaz.	X	
Reg. indirecto indexado.	X	
Absoluto largo	X	
Absoluto corto	X	
Relat. cont. programa con despl.		
Relat. cont. programa indexado.		
Inmediato.		

Los flags se alteral de igual modo que con la instrucción AND.

-ORI

Es una variante de OR, al igual que ocurría con la instrucción ANDI. LLeva a cabo una instrucción OR entre un dato inmediato (que funciona como operando fuente) y un operando destino que puede direccionarse de los modos que se figuran en la siguiente tabla.

ORI admite tamaños de operando de 8,16 y 21 bits.

Los flags varían igual que con la instrucción OR.

Esta instrucción, al igual que ANDI admite una variante de "ORI AL CCR" y "ORI AL SR", haciendo una operación or entre un dato inmediato y el Registro de estado, con un tamaño de operando de 8 bits en el primer caso, y de 16 en el segundo. (En en primer caso sólo afectaría a los códigos de condición). Lógicamente cuando se hace la operación ORI a la totalidad del Registro de Estado, hay que hacerlo en modo supervisor, ya que es una instrucción privilegiada.

-- Alteración de los flags:

'N','Z' Se alteran de acuerdo con el resultado obtenido.
'V','C' Se ponen a 0 con la presente instrucción.
'X' No se altera.

INSTRUCCION EOR

Modos Posibles	op. fuente	op destino
Reg. datos directo	X	X
Reg. direcc. directo		
Reg. direcc. indirecto		X
Reg. indirecto con postincr.		X
Reg. indirecto con predecr.		X
Reg. indirecto con desplaz.		X
Reg. indirecto indexado.		X
Absoluto largo		X
Absoluto corto		X
Relat. cont. programa con despl.		
Relat. cont. programa indexado.		
Inmediato.		

INSTRUCCION ORI

Modos Posibles	op. fuente	op destino
Reg. datos directo		X
Reg. direcc. directo		
Reg. direcc. indirecto		X
Reg. indirecto con postincr.		X
Reg. indirecto con predecr.		X
Reg. indirecto con desplaz.		X
Reg. indirecto indexado.		X
Absoluto largo		X
Absoluto corto		X
Relat. cont. programa con despl.		
Relat. cont. programa indexado.		
Inmediato.	X	
Registro de estado.		X

-EORI

Realiza una operación OR EXCLUSIVA "entre el operando fuente y el destino, almacenando el resultado en el destino" (este el el punto comùn de casi todas las instrucciones -¿fàcil no?-).

Admite operandos de 8, 16 y 32 bits.

El operando fuente ha de ser siempre un registro de datos, y el operando destino puede direccionarse de cualquier modo de los que aparecen en la tabla siguiente.

NOTIFICACIONES

LIBRERIA DE PROGRAMAS

Este mes incorporamos en libreria:

19.- rep_en long. 3741
rep_com long. 864

Se trata de las versiones fuente y objeto del listado incluido en ensamblador en este mismo número.

20.- Trap_equ long. 3767
Bdos_equ long. 4281

Se trata de los dos ficheros a los que nos hemos referido en el articulo del ensamblador de MetaComCo. Se deben incluir en los ficheros fuente con el comando GET 'MSYS_TRAP_EQU' por ejemplo.

21.- Copiador_bas

Se refiere al programa copiador aparecido el mes pasado en nuestras pàginas incorporado a la libreria para aquellos que no deseen teclearlo.

QLave

SUMARIO