QL

Programación Avanzada



ADRIAN DICKENS

ta-ma

NUESTRO AGRADECIMIENTO A **INVESTRONICA** POR SU COLABORACION EN LA REVISION DE ESTE MANUAL; Y POR LA APORTACION DE LAS NOTAS ACLARATORIAS SOBRE LA VERSION ESPAÑOLA.

QL

Programación Avanzada

ADRIAN DICKENS

ta-ma

Titulo de la obra original:
QL ADVANCED USER GUIDE
por Adrian Dickens

Copyright del QDOS Operating System
(c) 1983 Sinclair Research Limited

Copyright (c) 1984 Adder Publishing

Traducido del inglés por:
M.C.Dopazo

Primera edición en español:
Copyright (c) 1985
RA-MA
c/ Chiquinquirá, 28 (Cocuy)
28033 MADRID

Primera impresión Noviembre 1985

RESERVADOS TODOS LOS DERECHOS

No está permitida la reproducción parcial o total de este libro, ni el almacenamiento en un sistema informático ni transmisión en cualquier forma o por cualquier medio, electrónico, mecánico, fotocopia, registro u otros métodos, sin el permiso previo y por escrito de los propietarios del Copyright.

Aúnque se han tomado todas las precauciones en la preparación de este libro, el editor no asume ninguna responsabilidad por posibles errores u omisiones.

El autor agradece a Motorola Semiconductors el permiso para la reproducción de los diagramas del la hoja de datos del MC68008. Los diagramas siguen con el Copyright de Motorola.

Este libro está basado en la documentación actual producida por SINCLAIR Research Limited, y puede estar sujeta a cambios. El contenido de este libro no está garantizado por Sinclair Research Limited.

Cualquier consulta referida a este libro o a su contenido deberá dirigirse a Editorial RA-MA.

Imprime: Signo Impresores, S. A.
Albasanz, 27. 28037 MADRID
Depósito Legal: M. 40353 - 1985
I. S. B. N.: 84 - 86381 - 08 - 8

Contenido

1 Introducción 9

1.1 Introducción general 9

2 El microprocesador 68008 11

2.1 Introducción para los principiantes en c/máquina 11
2.2 Estructura interna del 68008 14
2.1.1 Registros de datos 14
2.2.2 Registros de dirección 15
2.2.3 Apuntadores de pila 15
2.2.4 El contador de programa 16
2.2.5 El registro de estado 16
2.2.6 Interrupciones 17
2.3 Organización de datos en la memoria 18
2.4 'TRAPS' y procesos de excepción 18
2.5 Direcciones efectivas 18
2.5.1 Modos de registro directo 19
2.5.2 Modos de direccionamiento de memoria 19
2.5.3 Modos especiales de direccionamiento 20
2.6 Sumario del juego de instrucciones 21
2.6.1 Operaciones de movimiento de datos 22
2.6.2 Operaciones aritméticas con enteros 23
2.6.3 Operaciones lógicas 24
2.6.4 Operaciones de desplazamiento y rotación 24
2.6.5 Operaciones de manipulación de bits 25
2.6.6 Operaciones decimales codificadas en bin. 25
2.6.7 Operaciones de control de programa 26
2.6.8 Operaciones de control del sistema 28

3 QDOS - Visión general 29

3.1 Introducción 29
3.2 QDOS y control del sistema 30
3.2.1 Gestión de memoria 30
3.2.2 Asignación de recursos 34
3.2.3 Planificación de tareas 34
3.3 Entrada y salida de Jobs 35
3.4 Soporte para programas de usuario 36

4 Experimentando con QDOS 37

4.1 Introducción 37
4.2 BASIC a código máquina 37
4.3 El programa EXPERIMENTADOR 41
4.3.1 Introducción 41
4.3.2 Listado del programa 41
4.3.3 Como trabaja el EXPERIMETADOR 44
4.3.4 Experimetos 46
4.4 Programando en ensamblador 52
4.4.1 Un reloj en la pantalla 53
4.4.2 El programa autoduplicable 55

5 Los TRAPS del Gestor 61

5.1 Introducción a los TRAPs del Gestor 61
5.2 Gestión de memoria en el QL 61
5.2.1 Procedimientos residentes 62
5.2.2 Jobs como programas transitorios 62
5.2.3 Asignación de memoria para programas BASIC 64
5.2.4 Asignación en el área común 64
5.2.5 Asignación del área de usuario 66
5.3 Gestión del 'hardware' 68
5.3.1 'Harware' estándar del QL 68
5.3.2 Extensiones al QDOS 68
5.4 TRAPs del gestor - Sección de referencia 69
5.4.1 Creación y borrado de Jobs 71
5.4.2 Control de Job 78
5.4.3 Gestión del área de usuario 84
5.4.4 Comunicación con el IPC 90
5.4.5 Extendiendo el sistema operativo 102

6 Asignación de Entrada/Salida 111

6.1 Introducción a E/S 111
6.2 Especificación de nombres de dispositivo 112
6.3 Rellenando especificaciones de nombres del sis. 113
6.3.1 Requerimientos para rellenar nombres sis. 113
6.3.2 Aproximación al rellenado de nombres sis. 114
6.3.3 Componentes de nombres de ficheros 115
6.4 Abriendo ficheros 115
6.4.1 Introducción general 115
6.4.2 Orden de búsqueda de ficheros 116
6.5 TRAPs de asignación de E/S 119

7 Utilización de Entrada/Salida 125

7.1 Introducción 125
7.1.1 Llamadas de E/S serie 125
7.1.2 Control de salida a pantalla 126
7.1.3 Manejo de ficheros 127
7.2 Llamadas de E/S serie 128
7.2.1 Opciones de retorno de E/S serie 129
7.2.2 TRAPs de E/S serie 130
7.3 Control de salida a pantalla 137
7.3.1 Controladores de pantalla del usuario 137
7.3.2 Buscando el tamaño de la ventana actual 138
7.3.3 Control de ventanas 140
7.3.4 Control del cursor 142
7.3.5 'Scroll' de las ventanas 152
7.3.6 Movimiento lateral de ventanas 155
7.3.7 Limpiando ventanas 158
7.3.8 Cambio del tipo de caracteres 163
7.3.9 Poniendo los colores 164
7.3.10 Cambio de los atributos de caracteres 170
7.3.11 Relleno de bloques 174
7.3.12 TRAPs de gráficos 175
7.4 TRAPs del sistema para manejo de ficheros 182

8 Utilidades por vector 192

8.1 Introducción 192
8.2 Rutinas en modo supervisor 195
8.3 Rutinas de TRAP simplificadas 202
8.4 Rutinas de interés general 209
8.4.1 Manejo de listas encadenadas 209
8.4.2 Manejo del área de usuario 211
8.4.3 Rutinas de manejo de colas 213
8.4.4 Utilidad de nombre de dispositivo 218
8.5 Rutinas de utilidad del SuperBASIC 219
8.5.1 Comparación de cadenas 219
8.5.2 Rutinas de conversión 222
8.5.3 Rutinas de soporte de procedimientos 240
8.5.4 Rutinas aritméticas 247
8.5.5 SINH como función residente 252
8.6 Rutinas de soporte de microdrives 255
8.6.1 Seleccionar un microdrive 256
8.6.2 Buscar un sector 256
8.6.3 Deseleccionar un microdrive 257

9 Controladores de dispositivo 263

9.1 Introducción 263
9.2 Bolques de definición de controlador de dispos. 264
9.3 Mediador de acceso 266
9.3.1 Abrir un canal 266
9.3.2 Entrada y salida de un canal 267
9.3.3 Cerrar un canal 267
9.4 Mediador físico 268
9.5 Decodificación del nombre de dispositivo 269
9.5.1 Descripción del nombre del dispositivo 270
9.6 Asignación de memoria 271
9.7 Colas de E/S 272
9.8 E/S serie simple 272
9.9 Controladores de dispositivo de directorio 273
9.9.1 Bloque de definición de cont. de direct. 273
9.9.2 Mediador de acceso de cont. de directorio 274
9.10 Bloques dependientes 276
9.11 Ejemplo de controlador de impresora paralela 277
9.11.1 Introducción 277
9.11.2 Programa controlador del dispositivo 278
9.11.3 Funcionamiento del programa 280

10 SuperBASIC 283

10.1 Introducción 283
10.2 La tabla de nombres 283
10.3 La lista de nombres 286
10.4 El área de valores de variables 286
10.5 Ejemplos de almacenamiento de variables 287
10.6 Pasando parámetros a procedimientos/funciones 288
10.7 Escribiendo procedimientos en código máquina 290
10.7.1 Introducción 290
10.7.2 Creando entradas en la tabla de nombres 290
10.7.3 La pila aritmética 292
10.7.4 Tomando argumentos de los procedimientos 292
10.7.5 Retorno de valores de variables 293
10.7.6 Retorno de valores de parámetros 293
10.8 Cambiando entre las pantallas 0 y 1 293
10.8.1 Programa de soporte de dos pantallas 294
10.8.2 Cómo funciona el programa 297
10.9 Implementación de BPUT# como procedimiento 298
10.9.1 Cómo funciona el programa 300

Apéndices

A	Mapa de la memoria	301
B	Sumario del juego de instrucciones del 68008	302
C	Especificaciones de la ROM enchufable	303
D	Formato de los Microdrives	306
E	Descripción de la pantalla	316
F	Especificaciones de expansiones 'hardware'	318
G	Códigos de error	323
H	Sumario de TRAPs	324
I	Sumario de utilidades por vector	328
J	Apuntadores a variables de SuperBASIC	330
K	Tipos de 'Token'	333
L	Códigos de operación del paquete aritmético	334
M	Definición de variables del sistema Operativo	335
N	Bloque de definición de canal	338
P	Bloque definición de canal de fichero del sist.	339
Q	Bloque de control de Job	340
R	Cabecera del área común	341
S	Bolque de definición de ventana	342
T	Programa para copias de seguridad microdrives	343
U	Circuito para impresora paralela	344
V	Juego de caracteres ASCII - Versión inglesa	347
W	Juego de caracteres ASCII - Versión española	353
X	Notas aclaratorias para la versión española	359

Bibliografía 363

Glosario 365

Indice 371

1 Introducción

La guia del usuario que viene con el QL contiene una simple explicación de la programación en BASIC e instrucciones para el paquete de cuatro programas Quill, Archive, Abacus y Easel. Apenas se ofrece alguna información de los aspectos más interesantes del QL como ordenador de nueva generación. Este libro se introduce dentro del QL, donde reside la potencia real de esta máquina. El BASIC es interesante para cierto número de aplicaciones, pero es bastante limitado. A diferencia de otras generaciones de microprocesadores, el QL contiene un procesador muy potente (el 68008) y un potente sistema operativo llamado QDOS.

Inicialmente debemos introducirnos en el BASIC. Una vez que lo hemos hecho, podremos controlar la máquina fácilmente. La transición entre la programación en BASIC y en Ensamblador se realiza por medio de un intermediario, el programa Experimentador de QDOS. Esto le permite acceder a la mayoría de las facilidades del código máquina mientras el entorno BASIC permanece relativamente protegido.

Una vez que hemos sobrepasado los confines del BASIC, la programación en lenguaje Ensamblador le abre un mundo completamente nuevo. Se consigue la programación en multitarea, que permite ejecutar varios trabajos al mismo tiempo. Se explican en profundidad las reglas fundamentales para preparar los trabajos, como lo requiere la programación en multitarea. Se ilustra con ejemplos como el programa que se auto-duplica, que ayuda a comprender los principios que se utilizan.

La producción de sus propios programas se expande después con los temas sobre manejo de E/S, pilas, listas encadenadas procedimientos residentes, programas transitorios y muchas más cosas.

El conocimiento del sistema tiene muchas ventajas. Esto se ilustra con un programa completo de utilidad para sacar copias de seguridad de los microdrives (escrito completamente en BASIC). Se consiguen muchas otras ventajas mediante la comprensión del sistema 'hardware'. Tambien se proporciona un esquema completo de un circuito para añadir un adaptador paralelo de tipo Centronics para impresora, así como el 'software' necesario. Una vez instalado, le permitirá listar los programas y datos en cualquiera de las impresoras que se encuentran normalmente en el comercio.

Este libro le proporcionará toda la información requerida

instrucciones que se pueden construir con 16 bits (hasta 65536 códigos diferentes), existe un lenguaje especial de alto nivel (escrito en código 68008) que convierte los códigos fácilmente comprensibles para los humanos (como 'ADD') en los que entiende el 68008. El programa que lo hace se llama *ensamblador*.

La programación en código máquina con un ensamblador es sustancialmente diferente del BASIC. Para empezar, solamente se se pueden utilizar diez y seis variables. Estas se llaman registros y tienen 32 bits de longitud. Ocho de ellos se usan para almacenar direcciones de memoria y se llaman registros de dirección. Los otros ocho se usan generalmente para almacenar datos, y se llaman registros de datos. Imagínese lo dificil que sería programar en BASIC con solo 16 variables! Para obviar esta limitación de almacenamiento, los registros se pueden almacenar en memoria y recuperarlos posteriormente cuando se necesiten. El microprocesador puede, también, ejecutar operaciones aritméticas, pero no de coma flotante como en BASIC. Tenemos suma, resta, multiplicación y división de enteros simples, además de las operaciones lógicas como AND, OR, etc. También disponemos de instrucciones para saltar de una parte a otra del programa en código máquina. En definitiva, se ha creado un potente rango de instrucciones en el 68008, para permitir la creación de programas bastante sofisticados. El intérprete BASIC es también un programa en código máquina.

El código máquina difiere considerablemente del BASIC, principalmente en el manejo de errores. Si a una instrucción 'SAVE', se le da una dirección incorrecta, probablemente la ejecutará en medio de un programa. Si se ejecuta este programa modificado, normalmente hará fallar al QL. La única posibilidad de recuperarlo sería el botón de RESET, en cuyo caso perderíamos el programa y los datos.

En este punto, podría pensar que no es muy interesante programar en código máquina si se nos pueden presentar este tipo de problemas. Esto no es cierto, si tenemos en cuenta la enorme diferencia de velocidad con que podemos contar, aparte que muchas de las facilidades del QDOS son únicamente accesibles desde el código máquina. Una de las más notables es la multitarea que nos permite ejecutar varios programas diferentes al mismo tiempo. Hay muchas otras ventajas que podrá apreciar después de haber leido este libro.

Para resumir, el microprocesador 68008 ejecuta programas escritos en código máquina. Estos se ejecutan mucho más rápidamente que en BASIC, pero son más complicados de escribir. Solamente se pueden utilizar todas las ventajas

del QL con programas en código máquina.

El QDOS es una colección de subrutinas que pueden ser utilizadas por el usuario en código máquina. Estas rutinas proporcionan una amplia selección de funciones estándar, lo que significa que el programador no tiene por que escribir subrutinas una y otra vez. Una de las cosas que se pueden hacer a través de las subrutinas del QDOS es escribir letras en la pantalla. Si un programador lo quisiera hacer por si mismo, necesitaría almacenar en memoria la estructura por puntos de cada carácter, copiarlo a su lugar de la pantalla y ejecutar algunas comprobaciones. Esto necesitará de varias operaciones como comprobar si el carácter es el último de la línea (si lo es, el siguiente carácter se debe escribir en la línea siguiente) etc. Usted puede apreciar por si mismo lo tedioso que es programar de esta manera. Afortunadamente, el QDOS tiene una rutina que lo hace por usted. Solo hay que pasarle el código del carácter, el QDOS hace el resto. Además de estas rutinas simples, el QDOS soporta otras funciones más complejas. Cubre operaciones como salvar datos en microdrive, soportar la multitarea y reservar memoria. Todas estas posibilidades se exploran en profundidad en los siguientes capítulos.

Para comprender como se accede a las diferentes rutinas del QDOS, necesitamos conocer como opera el 68008. El resto del capítulo está dedicado a este punto.

2.2 Estructura interna del 68000

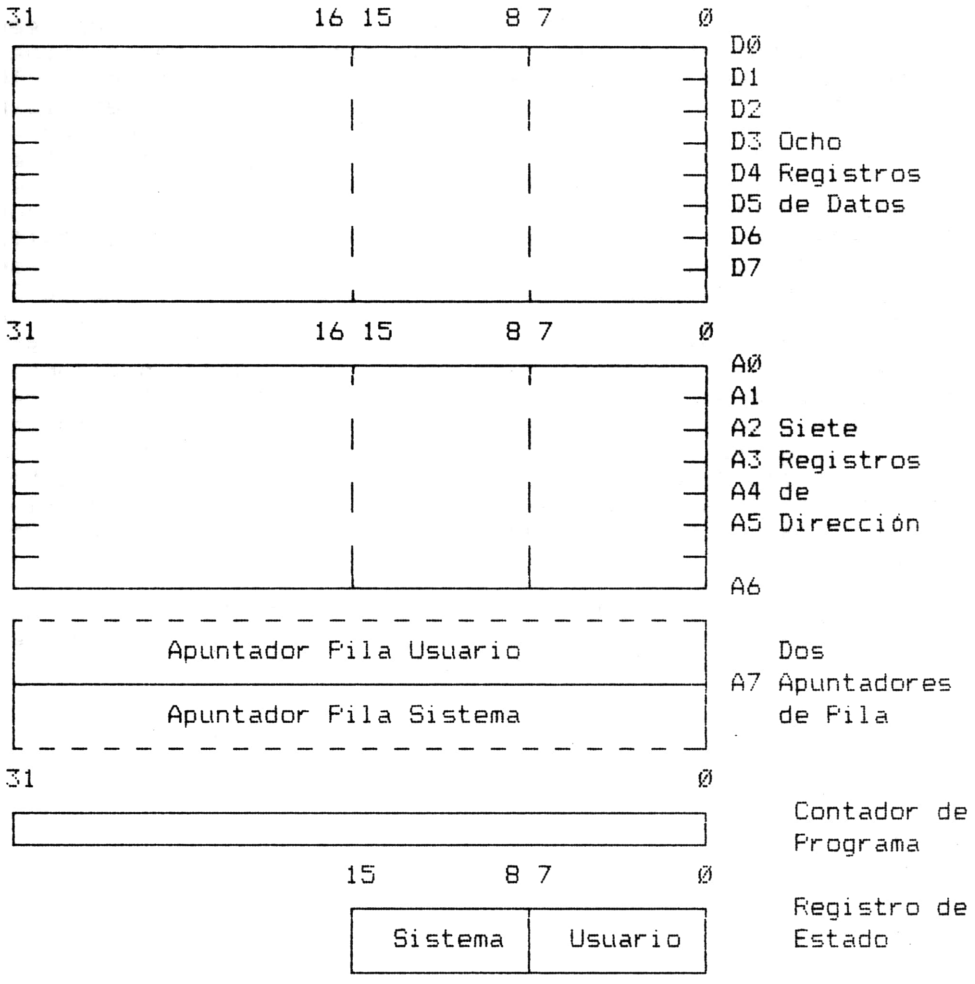


Figura 2.1 Estructura interna de 68000

Internamente, el microprocesador consta de varios registros. Estos se ilustran en la figura 2.1. Los registros se pueden dividir en cinco grupos funcionales:

2.2.1 Registros de datos

Hay ocho registros de datos, cada uno de 32 bits de longitud (una 'palabra larga'). La mayoría de las operaciones de almacenamiento y transferencia de datos se realizan

con este juego de registros. Se pueden cargar datos dentro de ellos desde otro registro, memoria o dispositivos periféricos como los microdrives. Una vez cargados los datos, se pueden manipular de diferentes formas, como sumarles algún valor. Los datos pueden volver a ser almacenados en memoria o mandados a un periférico.

2.2.2 Registros de dirección

Hay ocho registros generales de dirección (el octavo es especial,porque se usa normalmente como apuntador de la pila - ya lo veremos con más detalle en la siguiente sección). Estos registros de dirección se usan para ayudar en el movimiento de datos. Contienen las direcciones desde las que se toman los datos y en las que se almacenan. Los datos tomados de memoria se pueden cargar tanto en los registros de datos como en los de dirección. También se pueden usar brevemente en algunas operaciones (para sumarlo a alguno de los registros de datos). Los datos pueden ser salvados desde cualquiera de ls registros a la posición de memoria apuntada por el registro de dirección adecuado. Hay varias formas de apuntar los registros de dirección a los datos en memoria. De todas ellas de habla en la sección 2.5 donde se explican los *modos de direccionamiento*.

2.2.3 Apuntadores de pila

El octavo registro de dirección (A7) tiene una función especial. Actúa como el *apuntador de pila*. Una pila es un área de memoria en la que se apilan los datos y se toman posteriormente en orden inverso. El primer item que se almacena en la pila es el último que se toma - igual que una pila de cajas. A7 mantiene el apuntador a la dirección del último item almacenado, y se actualiza automáticamente a medida que se almacenan o se toman datos.

Como habrá observado, hay dos apuntadores de pila en el sistema. Sólamente puede estar activo uno de ellos a un mismo tiempo. El otro es invisible. El que está seleccionado viene determinado por el estado *privilegiado* del procesador. Hay dos niveles de privilegio. El más bajo está presente siempre que el 68008 está en modo *USUARIO*. La mayoría de los programas escritos por el usuario se ejecutan en este estado. Por lo tanto al meter o sacar datos de la pila, lo hacemos en la pila del usuario. El estado de privilegio más alto es el de *SUPERVISOR*. La mayoría de las rutinas del QDOS se ejecutan en este estado. En este nivel se puede conseguir mayor potencia. Por ejemplo, la intrucción de RESET del sistema solo se puede ejecutar en modo supervisor. El tener dos niveles de seguridad garantiza que el sistema no se pueda

interrumpir por un simple programa de usuario (aúnque no hay garantía de que no lo pueda hacer). Se pueden usar cualquiera de los otros registros de dirección para usarlos como apuntadores de pila.

2.2.4 El contador de programa

Este registro de 32 bits no es accesible directamente como los de datos o de direcciones. Mantiene la dirección de la instrucción del programa que se está ejecutando en cada momento. Es algo así como el recordar el número de línea BASIC que se está ejecutando. Si se ejecuta un salto a otra dirección del programa, este registro se cambia de acuerdo con la dirección de la instrucción que se debe ejecutar. Se puede usar el contador de programa (o PC) en cálculo de direcciones, como 'carga los datos almacenados 20 octetos después de esta instrucción'. Esto es bastante útil para escribir programas *reubicables* (los que se pueden ejecutar en cualquier dirección de la memoria).

2.2.5 El registro de estado

Este registro de 16 bits se divide en dos mitades, el octeto del supervisor y el de usuario. Solo se pueden modificar *directamente* en estado supervisor.

En el octeto de usuario, solamente se usan los cinco bits inferiores. Estos marcan los estados que ocurren normalmente después de ciertas instrucciones aritméticas. Están al nivel de 'aritmética de escuela primaria'. Si usted suma dos números, cuyo resultado sea demasiado largo para caber en un registro de datos, se pondrá el bit de acarreo. Si se resta un número de sí mismo, se pondrá el indicador de cero, y así sucesivamente. Las otras instruciones que se registran son rebasamiento (V), negativo (N) y extendido (X).

En estado supervisor se usan solo cinco bits. Un bit indica el estado del procesador (usuario o supervisor). Otro bit indica si el modo 'trace' está activo o no ('trace' se usa para ejecutar las instrucciones de un programa, de una en una, cuando se está desarrollando). Los tres bits que quedan contienen la prioridad de *interrupción*. La prioridad se puede poner a uno de ocho niveles. El nivel 0 es el más bajo y el 7 el más alto. Todas la interrupciones que tengan una prioridad mayor que la especificada en este registro, no serán *enmascaradas*. Las interrupciones de nivel 7 no se pueden enmascarar.

2.2.6 Interrupciones

Las interrupciones son generadas por dispositivos externos de 'hardware'. El circuito de vídeo del QL, genera interupciones 50 veces por segundo. La llegada de la señal al 68008 hace que sucedan varias cosas. Se suspende la operación que se está ejecutando en ese momento (asumiendo que la prioridad de la interrupción sea suficientemente alta), y el 68008 salta a ejecutar la rutina de servicio de interrupciones. Antes de entrar en esta rutina, debe salvar el contenido de todos los registros, realizar las funciones necesarias para dar servicio a la interrupción, restaurar los registros y devolver control. El efecto aparente de la interrupción es que se han realizado algunas funciones, pero sin afectar seriamente a la operación que estaba ejecutando el programa, solamente retrasándola unos pocos microsegundos. Las interrupciones se usan generalmente para dar servicio a dispositivos que requieren una atención inmediata. Si no son atendidos inmediatamente, se pueden perder algunos datos (ej. la cinta del microdrive puede pasar de largo los datos esperados).

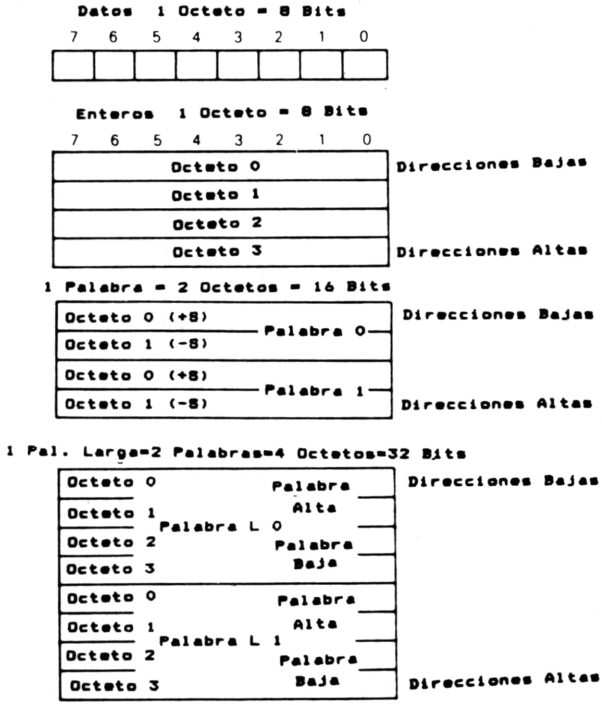


Figura 2.2 Organización de datos en memoria

2.3 Organización de datos en la memoria

El 68008 puede realizar operaciones en memoria con cuatro tipos de datos diferentes. Estos son bit, octetos, palabras y palabras largas, que consisten en 1, 8, 16 y 32 bits respectivamente. En la figura 2.2 se pueden ver los diferentes modos en que se organizan los datos en la memoria.

NOTA: Las instrucciones están formadas siempre por palabras y estas palabras deben empezar en una dirección par. Algunas veces, al almacenar un octeto de datos, es necesario desperdiciar el octeto que hay después de los datos y antes de la siguiente instrucción.

2.4 'TRAPS' y procesos de excepción

En el 68008 pueden ocurrir varios tipos de *excepciones*. Estas pueden dividirse en dos categorias según sean generadas interna o externamente. Como ya hemos visto, las interrupciones pertenecen a las excepciones generadas externamente. Las excepciones internas se producen mediante la ejecución de una instrucción TRAP, TRAPV, CHK o DIV.

Cuando ocurre cualquiera de estas excepciones, se ejecuta una llamada a la rutina de servicio adecuada. La dirección de esta rutina de servicio (una por cada excepción) está contenida en el primer kilobyte de memoria (ver figura 2.3).

Las excepciones más comunmente usadas son las interrupciones (una cada 20 ms para invocar al planificador) y los TRAP para invocar a las subrutinas del sistema. Los parámetros que se han de pasar a estas subrutinas deben cargarse en los registros. En D0 cargamos el número de la subrutina requerida, y después generamos un TRAP #N. Las partes finales de este libro explican lo que hace cada una de estas subrutinas. Hay 16 TRAP pero solo se pueden usar cinco para el sistema operativo. Estos realizan las siguientes funciones:

TRAP #0 pasa de modo usuario a modo supervisor
TRAP #1 se encarga del manejo de recursos del sistema
TRAP #2 se encarga de la asignación de entrada/salida
TRAP #3 se encarga de la utilización de entrada/salida
TRAP #4 se usan en conjunción con el BASIC, TRAP 2 y 3

2.5 Direcciones efectivas

Las instrucciones del 68008 operan con datos en los registros o en memoria. Por lo tanto es necesario tener bien

definida la forma de direccionar los datos. Para ello se ha incluido un potente grupo de modos de *direccionamiento* que se acoplan a la mayoría de las aplicaciones. Nos solemos referir a estas direcciones como *direcciones efectivas*. Algunos modos son aplicables a ciertos *tipos* de argumentos solamente. Por ejemplo, puede ser inadecuado usar un registro de datos como apuntador de pila. En la descripción de los modos, que viene a continuación, se mencionan estas restricciones. Disponemos de los siguientes modos:

2.5.1 MODOS DE REGISTRO DIRECTO

En estos modos de direccionamiento efectivo, el operando (el dato con el que vamos a operar) debe estar en uno de los diez y seis registros multifunción.

2.5.1.1 Registro de datos directo

El operando está en el registro de datos especificado por la dirección efectiva.

2.5.1.2 Registro de dirección directo

El operando está en el registro de dirección especificado por la dirección efectiva.

2.5.2 MODOS DE DIRECCIONAMIENTO DE MEMORIA

En estos modos de direccionamiento efectivo, el operando está en memoria y disponemos de su dirección.

2.5.2.1 Registro de dirección indirecto

La dirección del operando está en el registro de dirección especificado. Está clasificada como referencia de datos, excepcto en las instrucciones JMP y JSR.

2.5.2.2 Registro indirecto con post-incremento

La dirección del operando está cargada en el registro (como en direcionamiento por registro indirecto). Después de usar la dirección del operando, se incrementa en uno, dos o cuatro, dependiendo de que la longitud del operando sea un octeto, una palabra o una palabra larga de longitud. Si el registro de direción es el apuntador de la pila y el tamaño del operan- do es un octeto, la dirección se incrementa en dos en vez de uno, para mantener el apuntador de la pila en frontera de palabra. Está clasificada como referencia de datos.

2.5.2.3 Registro indirecto con post-decremento

La dirección del operando está cargada en el registro (como en direcionamiento por registro indirecto). Después de usar la dirección del operando, se decrementa en uno, dos o cuatro, dependiendo de la longitud del operando sea un octeto, una palabra o una palabra larga de longitud. Si el registro de dirección es el apuntador de la pila y el tamaño del operando es un octeto, la dirección se decrementa en dos en vez de uno, para mantener el apuntador de la pila en frontera de palabra. Está clasificada como referencia de datos.

2.5.2.4 Registro de dirección indirecto con desplazamiento

Requiere una palabra de extensión. La dirección del operando se calcula añadiendo al registro de dirección, el entero de desplazamiento de 16 bits, con signo extendido, contenido en la palabra de extensión. Está clasificada como referencia de datos, excepto en las instrucciones JMP y JSR.

2.5.2.5 Registro de direción indirecto con índice

Requiere una palabra de extensión. La dirección del operando se calcula sumando al registro de dirección, el entero de desplazamiento con signo extendido, contenido en los 8 bits inferiores de la palabra de extensión, más el contenido del registro de índice. Está clasificada como referencia de datos, excepto en las instrucciones JMP y JSR.

2.5.3 MODOS ESPECIALES DE DIRECCIONAMIENTO

Para especificar el modo especial de direccionamiento, se usa la direción efectiva en vez del número de registro.

2.5.3.1 Dirección absoluta corta

Requiere una palabra de extensión. La dirección del operando es la palabra de extensión. La dirección de 16 bits se extiende con signo antes de usarse. Está clasificada como referencia de datos, excepto en las instrucciones JMP y JSR.

2.5.3.2 Dirección absoluta larga

Requiere dos palabras de extensión. La dirección del operando se encuentra contenida íntegramente en las palabras de extensión. Está clasificada como referencia de datos, excepto en las instrucciones JMP y JSR.

2.5.3.3 Contador de programa con desplazamiento

Requiere una palabra de extensión. La dirección del operando se obtiene añadiendo el contenido de esta palabra de desplazamiento de 16 bits con signo extendido, al valor del contador de programa. Está clasificada como referencia de programa.

2.5.3.4 Contador de programa con índice

Requiere una palabra de extensión. La direción del operando se obtiene añadiendo el desplazamiento de signo extendido contenido en los ocho bits inferiores de la palabra de extensión y el contenido del registro de índice, al contador de programa. Está clasificada como una referencia de programa.

2.5.3.5 Datos inmediatos

Requiere uno o dos octetos de extensión, dependiendo del tamaño de la operación. Hay tres alternativas posibles:

operación de octeto - el operando está en el octeto inferior de la palabra de extensión.

operación de palabra - el operando es la palabra de extensión

Palabra larga - el operando está en las dos palabras de extensión que van a continuación.

2.5.3.6 Referencia implícita

Algunas instrucciones hacen referencia implícita al contador de programa (PC), al apuntador de la pila del sistema (SP), al apuntador de la pila del supervisor (SSP), al apuntador de la pila de usuario (USP), o al registro de estado. Un grupo seleccionado de instrucciones pueden referirse al registro de estado mediante su dirección efectiva, estas son instrucciones privilegiadas.

2.6 Sumario del juego de instrucciones

Esta sección contiene una visión genereal del juego de instrucciones del 68008. Las instrucciones se pueden agrupar bajo ocho enunciados principales:

1. Movimiento de datos
2. Aritmética con enteros
3. Operaciones lógicas

4. Desplazamiento y rotación
5. Manipulación de bit
6. Decimal codificado en binario
7. Control de programa
8. Control del sistema

A continuación vamos a explicarlos más detalladamente. Hemos usado los siguientes códigos para indicar los tipos particulares de operandos:

An- registro de dirección
Dn- registro de datos
Rn- registro de datos o de dirección
PC- contador de programa
SR- registro de estado
CCR- codigos de condición(octeto inferior del registro de estado)
SSP- apuntador de pila del supervisor
USP- apuntador de pila del usuario
SP- apuntador de pila activo(equivalente a A7)
X- operación extendida(de los códigos de condición)
Z- código de condición cero
V- código de condición de rebasamiento
Inmediate data- datos inmediatos de la instrucción
d- desplazamiento de dirección
Source- dirección efectiva de origen
Destination- dirección efectiva de destino
Vector- situación del vector de excepción

2.6.1 Operaciones de movimiento de datos

La instrucción MOVE permite trasferir o almacenar datos. Los datos se pueden mover en unidades de octeto, palabra o palabra larga. Se pueden trasferir datos de memoria a memoria, de memoria a un registro, de un registro a memoria, o de un registro a otro. Las operaciones con registros de dirección solamente permiten operaciones de trasferencia de palabra o de palabra larga. Además de las instrucciones generales de transferencia, hay varias instrucciones especiales de movimiento de datos. MOVEM mueve registro múltiples, MOVEP mueve datos de los periféricos, EXG intercambia el contenido de registros, LEA carga una dirección efectiva, PEA introduce una dirección efectiva en la pila, LINK conecta la pila, UNLK desconecta la pila y MOVEQ mueve un octeto a la dirección especificada. Las operaciones de movimiento de datos se ilustran en la figura 2.4.

Instruction	Operand Size	Operation
EXG	32	Rx↔Ry
LEA	32	EA → An
LINK	-	AN → -(SP) SP → An SP + displacement → SP
MOVE	8, 26, 32	(EA)s → (EA)d
MOVEM	16, 32	(EA) → An, Dn An, Dn → (EA)
MOVEP	16, 32	(EA) → Dn Dn → (EA)
MOVEQ	8	#xxx → Dn
PEA	32	EA → -(SP)
SWAP	32	Dn[31:16]↔Dn[15:0]
UNLK	-	An → SP (SP)+ → An

NOTAS:

s=fuente

d=destino

[]=nuemro de bit

-=indir. con predecrem.

+=indir. con postdecrem.

#=datos inmediatos

Figura 2.4 - Operaciones de movimiento de datos

2.6.2 Operaciones aritméticas con enteros

En el 68008 se soportan las cuatro operaciones aritméticas básicas, ADD (suma), SUB (resta), MUL (multiplicación) y DIV (división). Hay también una comparación aritmética CMP, limpiar CLR y negar NEG. La suma y la resta se pueden utilizar en operaciones de datos y de direcciones (pero deben tener una palabra o una palabra larga de longitud en el caso de direcciones).

Se pueden multiplicar y dividir operandos con signo y sin signo. La multiplicación de palabras producen una palabra larga en el resultado. Dividir una palabra larga por una palabra produce como resultado una palabra para el cociente y otra para el resto.

También tenemos una serie de instrucciones que permiten realizar operaciones de precisión múltiple y aritmética de tamaños mezclados. Estas instrucciones son suma con extensión ADDX, resta con extensión SUBX, signo extendido EXT y negación binaria con extensión NEGX.

Se puede comparar el operando con cero mediante la instrucción TST (test). Esta hace que se pongan los códigos de condición como resultado de la operación. En la figura 2.5 tenemos un resumen de las operaciones de aritmética con enteros.

Instruction	Operand Size	Operation
ADD	8, 16, 32	Dn + (EA) → Dn (EA) + Dn → (EA) (EA) + #xxx → (EA)
	16, 32	An + (EA) → An
ADDX	8, 16, 32	Dx + Dy + X → Dx
	16, 32	-(Ax) + -(Ay) + X → -(Ax)
CLR	8, 16, 32	0 → (EA)
CMP	8, 16, 32	Dn - (EA) (EA) - #xxxx
	16, 32	(Ax)+ - (Ay)+ An - (EA)
DIVS	32 ÷ 16	Dn ÷ (EA) → Dn
DIVU	32 ÷ 16	Dn ÷ (EA) → Dn

Instruction	Operand Size	Operation
EXT	8 → 16 16 → 32	$(Dn)_8 \rightarrow Dn_{16}$ $(Dn)_{16} \rightarrow Dn_{32}$
MULS	16 x 16 → 32	Dn x (EA) → Dn
MULU	16 x 16 → 32	Dn x (EA) → Dn
NEG	8, 16, 32	0 - (EA) → (EA)
NEGX	8, 16, 32	0 - (EA) - X → (EA)
SUB	8, 16, 32	Dn - (EA) → Dn (EA) - Dn → (EA) (EA) - #xxx → (EA)
		An - (EA) → An
SUBX	8, 16, 32	Dx - Dy - X → Dx -(Ax) - -(Ay) - X → -(Ax)
TAS	8	(EA) - 0, 1 → EA(7)
TST	8, 16, 32	(EA) - 0

NOTAS:

[]=numero de bit **-=indir. con predecrem.**

#=datos inmediatos **+=indir. con postdecrem.**

Figura 2.5 - Operaciones aritméticas con enteros

2.6.3 Operaciones lógicas

Con las operaciones lógicas estándar AND, OR, EOR y NOT, se pueden manipular operandos de datos enteros de un octeto, una palabra o una palabra larga de longitud. Hay también un juego de instrucciones inmediatas equivalentes, ANDI, ORI y EORI, que permiten estas operaciones lógicas con cualquier tamaño de datos inmediatos. En la figura 2.6 tenemos un resumen de las operaciones lógicas.

Instruction	Operand Size	Operation
AND	8, 16, 32	Dn Λ (EA) → Dn (EA) Λ Dn → (EA) (EA) Λ #xxx → (EA)
OR	8, 16, 32	Dn V (EA) → Dn (EA) V Dn → (EA) (EA) V #xxx → (EA)
EOR	8, 16, 32	(EA) ● Dy → (EA) (EA) ● #xxx → (EA)
NOT	8, 16, 32	~(EA) → EA

NOTAS:

#=datos inmediatos **V=O logico**

~=invertido **●=O logico exclusivo**

V=Y logico

Figura 2.6 - Operaciones lógicas

2.6.4 Operaciones de desplazamiento y rotación

Este grupo de operaciones permite el desplazamiento y giro de los bits dentro de un octeto, una palabra o una palabra larga (para registros-en memoria solo se puede hacer con

palabras).

El contador del desplazamiento se puede especificar en el registro de datos siempre que la operación se realize en un registro. Las operaciones en memoria solo soportan desplazamientos y rotaciones de un bit.

Hay dos tipos de desplazamientos. Las instrucciones de desplazamiento aritmético son ASR y ASL. Las instrucciones lógicas son LSR y LSL, para desplazamiento a la derecha y a la izquierda respectivamente.

Las rotaciones se pueden hacer con o sin el bit de extensión en el bucle. Las instrucciones son ROXR, ROXL, ROR y ROL. En la figura 2.7 se puede ver como funcionan.

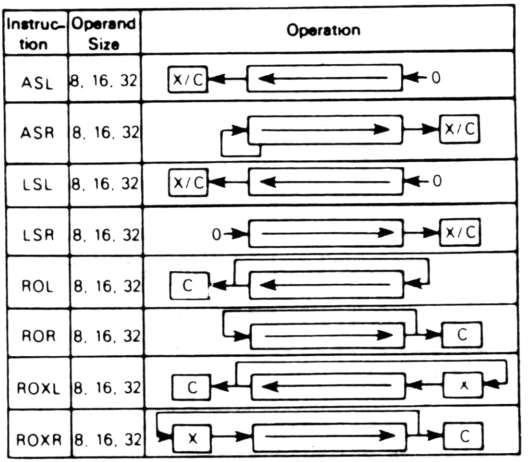


Figura 2.7 - Instrucciones de desplazamiento y rotación

2.6.5 Operaciones de manipulación de bits

Hay cuatro instrucciones de manipulación de bits; test de bit BTST, test y puesta de bits BSET, test y limpieza de bits BCLR y test y cambio de bits BCHG. Se pueden ver en la figura 2.8. Tenga en cuenta que la 'Z' se refiere al bit de 'cero', que es el bit 2 del registro de estado.

2.6.6 Operaciones decimales codificadas en binario

Con los números decimales codificados en binario, se pueden

Instruction	Operand Size	Operation
BTST	8, 32	~ bit of (EA) → Z
BSET	8, 32	~ bit of (EA) → Z 1 → bit of EA
BCLR	8, 32	~ bit of (EA) → Z 0 → bit of EA
BCHG	8, 32	~ bit of (EA) → Z ~ bit of (EA) → bit of EA

NOTE ~ = invert

Figura 2.8 - Operaciones de manipulación de bits

realizar tres operaciones de precisión múltiple. Estas son; suma decimal con extensión ABCD, resta decimal con extensión SBCD y negación decimal con extensión NBCD. Estas instrucciones se pueden ver en la figura 2.9.

Instruction	Operand Size	Operation
ABCD	8	$Dx_{10} + Dy_{10} + X \rightarrow Dx$ $-(Ax)_{10} + -(Ay)_{10} + X \rightarrow (Ax)$
SBCD	8	$Dx_{10} - Dy_{10} - X \rightarrow Dx$ $-(Ax)_{10} - -(Ay)_{10} - X \rightarrow (Ax)$
NBCD	8	$0 - (EA)_{10} - X \rightarrow (EA)$

Figura 2.9 - Operaciones BCD

2.6.7 Operaciones de control de programa

Las operaciones de control de programa usan una serie de instrucciones de bifurcación condicionales e incondicionales, de salto y de retorno.

Las instrucciones condicionales pueden ser llamadas para bifurcar en cualquiera de las siguientes condiciones:

CC(HS)	acarreo quitado	LS	menor o igual
CS(LO)	acarreo puesto	LT	menor que
EQ	igual	MI	menos
F	falso	NE	no igual
GE	mayor o igual	PL	mas
GT	mayor que	T	cierto
HI	mayor	VC	sin revasamiento
LE	menor o igual	VS	revasamiento

Instruction	Operation
Conditional	
B_{CC}	Branch Conditionally (14 conditions) 8- and 16-Bit Displacement
DB_{CC}	Test Condition, Decrement, and Branch 16-Bit Displacement
S_{CC}	Set Byte Conditionally (16 Conditions)
Unconditional	
BRA	Branch Always 8- and 16-Bit Displacement
BSR	Branch to Subroutine 8- and 16-Bit Displacement
JMP	Jump
JSR	Jump to Subroutine
Returns	
RTR	Return and Restore Condition Codes
RTS	Return from Subroutine

Figura 2.10 - Operaciones de control de programa

Instruction	Operation
Privileged	
ANDI to SR	Logical AND to Status Register
EORI to SR	Logical EOR to Status Register
MOVE EA to SR	Load New Status Register
MOVE USP	Move User Stack Pointer
ORI to SR	Logical OR to Status Register
RESET	Reset External Devices
RTE	Return from Exception
STOP	Stop Program Execution
Trap Generating	
CHK	Check Data Register Against Upper Bounds
TRAP	Trap
TRAPV	Trap on Overflow
Status Register	
ANDI to CCR	Logical AND to Condition Codes
EORI to CCR	Logical EOR to Condition Codes
MOVE EA to CCR	Load New Condition Codes
MOVE SR to EA	Store Status Register
ORI to CCR	Logical OR to Condition Codes

Figura 2.11 - Operaciones de control del sistema

2.6.8 Operaciones de control del sistema

Bajo este título se agrupan una serie de instrucciones privilegiadas, de generacion de TRAPs y otras que usan o modifican el registro de estado. Las diferentes opciones disponibles se pueden ver en la figura 2.11.

Vector Number(s)	Address			Assignment
	Dec	Hex	Space	
0	0	000	SP	Reset Initial SSP
–	4	004	SP	Reset Initial PC
2	8	008	SD	Bus Error
3	12	00C	SD	Address Error
4	16	010	SD	Illegal Instruction
5	20	014	SD	Zero Divide
6	24	018	SD	CHK Instruction
7	28	01C	SD	TRAPV Instruction
8	32	020	SD	Privilege Violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 Emulator
11	44	02C	SD	Line 1111 Emulator
12	48	030	SD	(Unassigned, Reserved)
13	52	034	SD	(Unassigned, Reserved)
14	56	038	SD	(Unassigned, Reserved)
15	60	03C	SD	Uninitialized Interrupt Vector
16-23	64	04C	SD	(Unassigned, Reserved)
	95	05F		–
24	96	060	SD	Spurious Interrupt
25	100	064	SD	Level 1 Interrupt Autovector
26	104	068	SD	Level 2 Interrupt Autovector
27	108	06C	SD	Level 3 Interrupt Autovector
28	112	070	SD	Level 4 Interrupt Autovector
29	116	074	SD	Level 5 Interrupt Autovector
30	120	078	SD	Level 6 Interrupt Autovector
31	124	07C	SD	Level 7 Interrupt Autovector
32-47	128	080	SD	TRAP Instruction Vectors
	191	0BF		–
48-63	192	0C0	SD	(Unassigned, Reserved)
	255	0FF		–
64-255	256	100	SD	User Interrupt Vectors
	1023	3FF		–

Figura 2.12 - Tabla de Vectores

3 QDOS - VISION GENERAL

3.1 Introducción

La intención de este capítulo es presentarle los conceptos fundamentales del QDOS. La explicaciones detalladas de aspectos particulares se han dejado para los capítulos de referencia en la última mitad del libro.

El QL se diferencia de otros ordenadores en el hecho de que soporta trabajos en *multitarea*. Esto significa que pueden trabajar al mismo tiempo varios programas completamente independientes.

Si se piensa un poco, se ve que este método de operación no es tan complejo como parece en un principio. Por ejemplo, usted puede querer jugar al ajedrez contra su QL. Los movimientos se pueden hacer en base a un movimiento diario. Esto permitiría al QL ejecutar gran cantidad de analisis complejos de los movimientos, pero no podría usarlo para ninguna otra cosa durante este tiempo. Aquí es donde se puede apreciar el uso real de los trabajos en *multitarea*. Si el programa de ajedrez se prepara como un trabajo de baja prioridad, puede operar todo el dia como un trabajo *subordinado*. Otros programas, como una aplicación de negocios, juegos u otras tareas generales, pueden ejecutarse con prioridades mayores. Entonces, el QL, usa el procesador para jugar ajedrez solamente cuando no lo requiere otra tarea preferente (ej. cuando la tarea preferente está esperando a que se pulse una tecla).

Otro ejemplo de multitarea es un reloj. La fecha y la hora se pueden ajustar a tiempo de encendido del QL. Después se puede cargar un trabajo que muestre un reloj en la pantalla. Este trabajo puede ejecutarse en prioridad muy baja, de forma que la hora del reloj se actualiza a intervalos regulares.

Resumiendo, un Job es una unidad de programación que opera aislado de las otras unidades de programación en el QL. El programa de la sección 4.4.2 nos muestra la cantidad de Jobs que pueden coexistir.

Ahora pensará que debe ser terriblemente complejo mantener todos estos trabajos ejecutandose al mismo tiempo. En otros sistemas menos potentes, esto podría ser un problema, y nos conduciría a una verdadera confusión del sistema interceptando interrupciones. Sin embargo, en el QL no es demasiado complejo - el QDOS se encarga de las partes má

difíciles. Para asegurarse que todos los Jobs coexisten pacíficamente, el QDOS debe comprobar que no usan las mismas áreas de memoria para propsitos diferentes, y que se controla cuidadosamente la asignación de los recursos de la máquina. Por ejemplo, si dos Jobs quieren mandar datos a la impresora al mismo tiempo, se producirían datos ininteligibles, si se mezclaran estas dos salidas. Para evitarlo, el último Job que haya requerido la impresora, es forzado a esperar a que termine el que la está usando. El QDOS evita este tipo de trabajo al programador, y se asegura de que el segundo Job se encola para imprimir, dejando al primero que termine con su cometido.

En este momento se puede enunciar la regla número uno. **USE EL QDOS EN TODO MOMENTO Y NO ESCRIBA PROGRAMAS QUE NO ESTEN DEBIDAMENTE CONECTADOS DENTRO DEL SISTEMA.** Esto significa que el QDOS debe ser siempre el encargado de asignar memoria, canales de impresión etc., Si el usuario hace, por ejemplo, un POKE de un programa de 68008 directamente en la memoria, podría ser desastroso si el QDOS no tiene noticias de ello. Por qué? Debido a que el mapa de memoria del QDOS está cambiando continuamente, si el QDOS no sabe lo que hay en un área particular, el programa introducido podría ser asignado como área libre de memoria!. Toda la memoria libre es susceptible de ser usada como memoria intermedia al leer o escribir en los microdrives. Por lo tanto el QDOS debe ser avisado cuando se prepara un trabajo o cuando se usa la memoria directamente.

Mientras el QDOS tenga total control de la memoria y de los otros recursos de la máquina, los trabajos operaran en el entorno de multitarea. Las tres secciones del QDOS que realizan el control de la memoria son, gestión de memoria (se asegurv que unos trabajos no modifican los datos de otros en la memoria), asignación de recursos (se asegura de que los recursos como la impresora, son usados por un solo trabajo a la vez) y planificación (decide que trabajos deben usar la unidad central de proceso 68008 en cada momento).

3.2 QDOS y control del Sistema

3.2.1 Gestión de memoria

En la seción de introducción decíamos que es necesaria cierta forma de control de la memoria. Si un buen control, al tener varios trabajos funcionando al mismo tiempo, escribiría unos encima de la memoria de los otros. Una de las misiones principales del QDOS es la de cuidar de la memoria del QL y asegurarse que dos trabajos no pueden usar

accidentalmente la misma memoria. Por supuesto, esto no significa que pueda usted ejecutar un programa con errores y esperar que el QDOS proteja el resto de la máquina!. Hacer POKEs aleatorios en la memoria del QL conducirá inevitablemente a un fallo del sistema. El QDOS solo es capaz de prevenir programas operacionales que sigan sus reglas. La memoria disponible está dividida en varias secciones (ver figura 3.1).

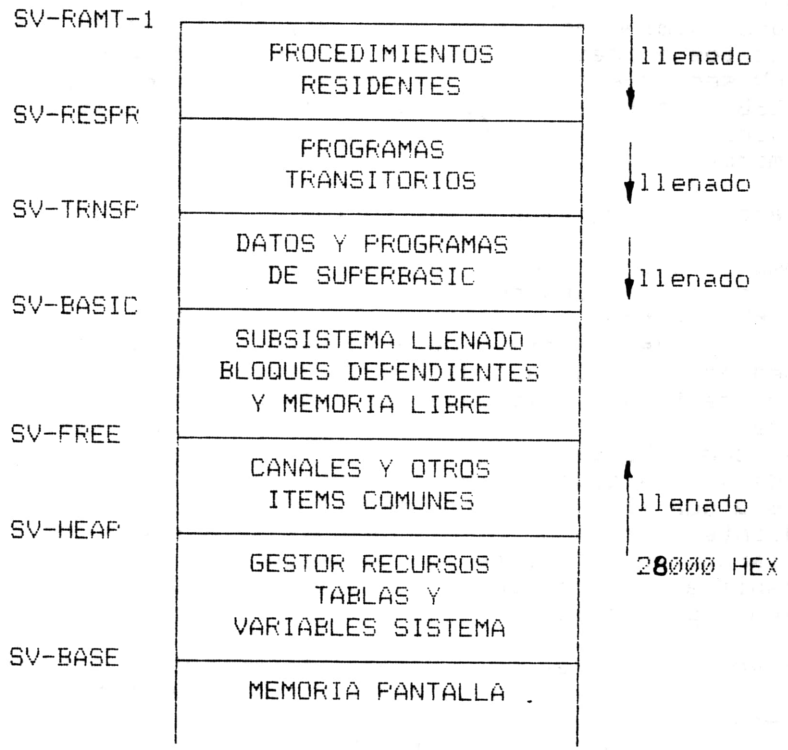


Figura 3.1 - Mapa de memoria del QDOS

Usos de la memoria - de arriba hacia abajo

El área de Procedimientos Residentes

Este bloque de memoria es asignado al encender la máquina, y una vez asignado permanece fijo. Toda la memoria entre la parte alta de la RAM disponible (SV.RAMT) y la parte inferior del área (SV.RESPR) está dedicada a procedimientos

residentes. En esta parte de memoria van los programas que son requeridos durante el uso de la máquina. Tenga en cuenta que SV.RAMT varia de una máquina a otra, dependiendo de la cantidad de memoria de extensión de que disponga.

Los procedimientos residentes añadidos al SuperBASIC deben ser colocados en este área. Los nombres de los procedimientos (o funciones) pueden ser conectados en la lista de nombres del BASIC, de forma que sean accesibles desde el BASIC. En la sección 10.8 puede encontrar un par de ejemplos de procedimientos y funciones. Hay que tener en cuenta que todos los procedimientos que se definan por este método DEBEN ser *re-entrantes e independientes de su posición* y NO DEBEN contener ningún código *automodificable* ni *variables locales* (en el glosario se explica el significado de estos términos).

El área de Programas Transitorios

Como su propio nombre indica, este área contiene programas que no son residentes, y que pueden ser eliminados en cualquier momento. La mayoría de los trabajos son colocados en este área de memoria. Los programas transitorios se cargan inmediatamente debajo de los procedimientos residentes. Cada programa transitorio debe sel auto contenido (código, área de datos y pila). Cualquier programa que no sea reubicable (ej. que debe ser cargado en un área de memoria específica) ha de ser cargado por un *cargador-conector* especial. Esto se debe a que los programas transitorios se cargan normalmente mediante el comando EXEC y después se activan como trabajos. La carga mediante este método significa que no se puede garantizar una posición determinada de memoria para un trabajo particular.

Si se usa el área de programas transitorios para otra cosa que no sea contener programas, como almacenar datos, debe tenerse mucho cuidado de que los datos no sean activados. Su activación provocaría un fallo de la máquina.

El área de SuperBASIC

En este área se almacenan todos los datos relaccionados con una operación correcta de SuperBASIC., incluido el programa BASIC tecleado y todas las variables del sistema de BASIC.

Es importante tener en cuenta que este área de memoria se está moviendo continuamente. Se mueve hacia abajo en la memoria cada vez que se asigna más área de programas transitorios. Se mueve hacia arriba cuando se desasigna parte de este área. Parte de este área se mueve cuando se teclean

nuevas líneas y cambia de tamaño a medida que se van creando nuevos datos durante la ejecución de un programa BASIC.

El mapa completo de memoria del QL está cambiando constantemente. Este sistema es diferente del de la mayoría de los ordenadores pequeños, en donde las diferentes partes se almacenan en posiciones fijas de memoria. Aunque a primera vista pueda parecer una forma no muy buena de hacer las cosas, en la práctica es bastante limpio. El 68008 puede usar simplemente uno de sus ocho registros de dirección (A6 para el BASIC) para apuntar a la parte inferior de cada área. Todas las operaciones dentro de ese área se realizan con relacción a este registro de dirección, creando la ilusión de que todos los datos están en posiciones fijas.

Espacio libre (usado como bloque dependiente del subsistema de relleno)

Este área se extiende desde la parte inferior del BASIC hasta la parte superior del área de canales y de almacenamiento. Normalmente, cuando no se está accediendo a ficheros, este área está vacía. Sin embargo, cuando se le pide al QDOS que lea o escriba un fichero, se pone en uso.

El acceso a los microdrives es realmente lento. Los ficheros suelen ser bastante largos y están contenidos en más de un sector del cartucho. Si todos los datos requeridos se pueden cargar en memoria de una vez, en vez de hacer lentos accesos a los sectores individuales, nos supondrá una clara ventaja. Esto es lo que hace el QDOS, aunque pase totalmente desapercibido para el usuario, se hace el máximo uso de la memoria disponible, por lo que la copia de ficheros con la máquina vacía es mucho más eficiente. Una vez que la máquina se llena con otros programas (como un programa largo de BASIC), el espacio libre en este área se hace mucho menor. En este caso, solo se pueden almacenar pequeñas cantidades de datos en memoria, y la copia de microdrives consiste en la transferencia de muchos paquetes pequeños de datos desde una unidad a otra - poco eficiente.

El almacenamiento del sistema

Este área de memoria se usa para almacenar definicones de canales, datos, memoria de trabajo para el subsistema de E/S etc. El área puede ser expandida o reducida de tamaño por medio de los TRAPS adecuados del gestor correspondiente. Los trabajos pueden reservar memoria en este área para su propio uso. El sistema se encarga de borrar todas las áreas reservadas por un trabajo cuando éste ha terminado su ejecución. El área de almacenamiento del sistema se explica

con más detalle en la sección 5.2.4.

Variables y tablas del sistema y pila del Supervisor

Este área fija de memoria reside justo sobre la memoria de pantalla. La pila del supervisor es usada por las rutinas que se ejecutan en modo supervisor en el 68008. Las variables del sistemas están listadas en el Apéndice M.

3.2.2 Asignación de recursos

El proceso de asignación de recursos se ocupa de todos los dispositivos que pueden ser direccionados por un trabajo. La pantalla, los microdrives, el teclado, el generador de sonido y la impresora serie son estos recursos. Si hay más de un trabajo que necesita usar un dispositivo al mismo tiempo, solamente uno de ellos lo podrá hacer. El resultado de una asignación correcta de recursos es el asegurarse de que el primer trabajo tiene prioridad y el segundo es colocado en modo de espera hasta qu el primero haya terminado.

El proceso de asignación de recursos se controla mediante los TRAPS del gestor (capítulo 5) y los TRAPS de asignación de E/S (capítulo 6).

3.2.3 Planificación de tareas

Hay varios aspectos de la planificación de tareas en una máquina multitarea. La CPU 68008 solo puede realizar un Job a la vez. Si se permitiera a un Job ejecutarse sin la intervención del planificador, no habría manera de evitar que usara todo el tiempo de CPU para sí mismo. Aquí es donde hace falta la intervención del planificador.

Siempre que se crea un Job, se la asigna una *prioridad*. Esta prioridad determina la forma de compartir el tiempo de la CPU de un Job concreto. Un Job con baja prioridad (1), solo podrá usar la CPU una décima parte del tiempo dado a otro Job con prioridad 10. La prioridad puede ser entre 0 (en cuyo caso el trabajo no coje nunca tiempo de la CPU) hasta 127. El intérprete de comandos BASIC se ejecuta como Job 0 y suele tener prioridad 32.

Los Jobs pueden estar en uno de los tres estados siguientes:

Activo Tienen una prioridad entre 1 y 127, y comparten los recursos de la CPU, de acuerdo con la prioridad.

Suspendido Son los capaces de ejecutarse (priorid entread 1 y 127), pero que no están funcionando por esperar a otro Job o E/S. El planificador rearrancará automáticamente un Job suspendido tan pronto como termine aquel por el que estaba esperando, o el dispositivo de E/S esté disponible. Los Jobs se pueden suspender indefinidamente, o por un periodo específico de tiempo (ver sección 5.4.2). Esta es un facilidad sumamente útil del QDOS. En los sistemas antiguos, un programa esperando por E/S puede acaparar todo el tiempo de CPU para intentar transferir pequeñas cantidades de datos.

Inactivo Son a los que se le ha asignado una prioridad de cero. Los trabajos inactivos no pueden disponer de los recursos de la CPU.

El planificador es invocado generalmente por una interrupción, 50 veces cada segundo. También se le invoca cuando se activa un trabajo (ver MT.ACTIV). A cada trabajo se le asigna una *prioridad temporal*. Esta prioridad es incrementada con la prioridad asignada del trabajo en cada llamada del planificador. La prioridad temporal del trabajo que se estaba ejecutando, es puesta a cero antes de incrementar de nuevo todas las prioridades temporales.

Algunos programas que contienen partes críticas, dependientes del tiempo (como leer un sector de microdrive), **no deben** ser interrumpidas por el planificador. A la rutina que debe terminar en un periodo de tiempo, se le llama *atómica*. La mayoría de las llamadas del sistema son atómicas. Es posible asegurar que una rutina de usuario sea *atómica* ejecutándola en modo *supervisor* en el 68008 (pero tenga cuidado de no llamar a una rutina del sistema que sea 'parcialmente atómica desde una rutina de usuario atómica - El planificador puede entrar en funcionamiento en las rutinas parcialmente atómicas).

3.3 Entrada y salida a Jobs

El QDOS contiene extensas facilidades de comunicación entre trabajos y dispositivos, ya sean estos estándar o expansiones. Teniendo en cuenta que todas las comunicaciones las lleva a cabo el QDOS, no puede haber problemas entre dispositivos. La sección 3.2.1 explica cómo están diseñados los trabajos para una interacción correcta en la asignación de recursos.

Las Entradas y Salidas se manejan con los TRAPS de utilización de E/S (capítulo 7) y con algunas de las utilidades especiales (capítulo 8). Las facilidades de que disponemos

son suficientes para la mayoría de las aplicaciones. La pantalla, por ejemplo, se maneja com las rutinas de definición de ventanas, definición de canales de salida en estas ventanas, control de color, tamaño de texto, tipo de cursor, dibujo y rellenado de gráficos, producción de bordes etc. En caso de que estas rutinas no se acoplen a los requerimientos de un programa particular, es posible incorporar rutinas especiales de usuario que trabajen en conjunción con las del QDOS. El principal criterio a seguir en todas ellas es que el QDOS debe saber en todo momento lo que se está usando, aún en el caso de que no se esté ejecutando una rutina residente del QDOS.

En este punto, los canales pueden ser preparados en una forma estándar que entienda el QDOS. Si se abre o cierra un canal usando un formato estándar (llamado bloque de definición de canal), el QDOS podrá saber lo que se está haciendo.

3.4 Soporte para programas de usuario

El QDOS proporciona vectores a todas sus rutinas, que sirven para supervisar las operaciones del QL. Esta es la principal tarea del QDOS.

La ROM que hay dentro del QL no solamente tiene que ver con las operaciones del sistema. Hay un considerable número de rutinas de utilidad a las que se puede acceder desde el código del usuario. Todas estas rutinas se explican en el capítulo 8.

Probablemente, el que mejor entiende estas utilidades es el intérprete de comandos de BASIC. Este es un Job por derecho propio (y muy especial), pero no es parte esencial del QL. Se pueden ejecutar otros trabajos en ausencia del BASIC. Muchas de las rutinas usadas por el BASIC, se pueden usar igualmente por el código del usuario. El paquete aritmético es un buen ejemplo de ello. El ejemplo de la sección 8.5.5 nos lo ilustra, produciendo una función BASIC extra que evalúa SINH de una variable.

Otras rutinas para entrada y salida serie, conversión de números y manejo de microdrives son algunas entre la amplia selección que están documentadas en las últimas partes de este libro.

4 EXPERIMENTANDO CON QDOS

4.1 Introducción

La mayoría de los capítulos de este libro se dedican a describir como opera el QDOS. Este capítulo es diferente porque está dedicado a experimentar desde el BASIC y el Ensamblador. Si no ha leido los capítulos 2 y 3, este es un buen momento para hacerlo, ya que este capítulo asume ciertos conocimientos del sistema y conceptos como 'TRAPS'.

Tanto el programa Experimentador del QL como los ejemplos en esta sección, han sido pensados para dar a los usuarios un *sentido intuitivo* del sistema QL. El experimentador le permite acceder a muchas de la facilidades del QDOS. Inicialmente, el programa Experimentador le permitirá generar TRAPS desde el BASIC. Es esencial entender los conceptos que hay detrás de cada TRAP en el QL si se quieren explotar otras interesantes facilidades del QDOS. A medida que nos introduzcamos en las profundidades del ordenador, quedará mas patente que muchas de las facilidades avanzadas solo se pueden usar adecuadamente desde el Ensamblador.

En esta fase se introducen programas prácticos escritos en Ensamblador, para ilustrar los aspectos más importantes del QDOS, vistos desde un programa en código máquina. Como introducción, tenemos un programa que nos pone un reloj en la pantalla. Este simple programa es muy útil para analizar, ya que nos muestra la iniciación de un Job, preparando una ventana, y convirtiendo la hora para poderla sacar a la pantalla. El programa *auto-duplicable* que ilustra muchos de los aspectos más complicados de los Jobs, incluida la asignación de memoria y el control de Jobs.

Hay otros ejemplos más especializados, repartidos a lo largo del libro. El capítulo 10 contiene ejemplos completos para producir una FUNCION BASIC y un PROCEDIMIENTO BASIC en Ensamblador 68008.

4.2 BASIC a Código Máquina

Antes de comenzar con el programa Experimentador, debemos clarificar algunos puntos acerca del uso del código máquina.

Las siguiente palabras clave de SuperBASIC tiene que ver con los programas en código máquina, pero no están suficientemente explicados en la 'Sinclair QL Guia del Usuario':

CALL

Este comando permite acceder al código máquina directamente desde el BASIC. Se pueden pasar a la rutina siete registros de datos y seis de dirección, usando el formato:

```
CALL dirección,D1,D2,D3,D4,D5,D6,D7,A0,A1,A2,A3,A4,A5
```

Esto hará que se ejecute un JSR a la 'dirección'. Se le pueden pasar como parámetros los regidtros de datos D1 a D7 y los de dirección A0 a A5. No es necesario que se pasen todos:

```
CALL 262000,23,4
```

pondrá solo D1=23 y D2=4. Tenga en cuenta que D0 no se pasa mediante el comando CALL, ya que se suele utilizar como registro de error. Sin embargo, esto no significa que no sea posible poner D0 en una rutina particular de TRAP desde el BASIC. El programa experimentador resuelve este problema pasando D0 en D4, trasfiriendolo después de D4 a D0 en código máquina. En las primeras versiones de la ROM, el comando CALL puede fallar si se usa desde un programa BASIC que ocupe más de 32K de memoria.

EXEC

EXEC carga y ejecuta un programa. Al programa se le asigna una parte de tiempo del procesador, pero el BASIC permanece operacional. Esto es debido a una de las principales características del QDOS - el hecho de soportar multitarea. Esencialmente, es una rutina dentro del sistema operativo, llamada planificador. Se invoca normalmente 50 veces cada segundo por medio de interrupciones. Cuando toma control, mira todos los Jobs que hay en la máquina y decide cual de ellos debe ejecutarse (de acuerdo con la prioridad). El Job seleccionado se ejecutará durante el siguiente cincuentaavo de segundo.

Por ejemplo:

```
EXEC MDV1-prog1
```

cargará el programa 'prog1' y comenzará a ejecutarlo, compartiendo su tiempo con el BASIC y los otros trabajos en el QL.

Tenga en cuenta que el programa que se inicializa usando EXEC debe haber sido salvado usando el comando SEXEC.

EXEC_W

Es casi idéntico al comando EXEC. La única diferencia es que completa el proceso antes de volver al BASIC. Si el proceso no está diseñado para retornar, el BASIC no volverá a tomar control.

LBYTES

LBYTES carga un fichero en memoria, en la dirección de memoria especificada. Por ejemplo:

```
LBYTES mdv1_programa,comienzo
```

cargará el fichero 'programa' desde el microdrive 1, poniendo el primer octeto en la dirección 'comienzo'.

PEEK

PEEK es una función que retorna el octeto contenido en una posición de memoria especificada. Por ejemplo:

```
PRINT PEEK(100)
```

nos devolvería el contenido del octeto 100(en la ROM).

PEEK_W

PEEK_W es una función similar a PEEK, pero devuelve la palabra contenida en una posición especificada (2 octetos consecutivos). Tenga en cuenta que la dirección debe ser par.

PEEK_L

PEEK_L es similar a PEEK, pero devuelve la palabra larga contenida en la posición de memoria especificada (4 octetos consecutivos). La dirección debe ser par. Esto se debe a que todas las palabras y palabras largas están alineadas en frontera de palabra. Los octetos pueden ocupar algunas veces media palabra. En esos casos, el resto de la palabra se suele rellenar a ceros y olvidarlo.

POKE

El comando POKE permite cambiar el contenido de un octeto de la memoria. Por ejemplo:

```
POKE 140000,52
```

pondrá 52 en la posición 140000 de memoria. Si hacemos POKE dentro de la ROM, no cambiaremos el contenido de esta posición de memoria, ya que la ROM no se puede alterar.

POKE_W

POKE_W es similar a POKE, pero cambia una palabra entera (2 octetos). La dirección especificada debe ser par.

POKE_L

POKE_L es similar a POKE, pero cambia una palabra larga en memoria (4 octetos). La dirección especificada debe ser par.

RESPR

RESPR es una función que asigna memoria en el área de procedimientos residentes. Se usa, después de que se ha inicializado la máquina, para asignar memoria para código maquina que debe estar residente. No le es posible asignar memoria una vez que se ha introducido algo en el área de programas transitorios. Por ejemplo:

```
PRINT RESPR(100)
```

asignará 100 octetos de memoria en el área de procedimientos residentes. El valor retornado es la dirección de comienzo del bloque de memoria asignado.

SBYTES

SBYTES permite salvar áreas de memoria en un dispositivo del QL. Por ejemplo:

```
SBYTES smdv1_panta,131072,32768
```

almacenará el contenido de la pantalla en el fichero 'panta' en el microdrive 1.

SEXEC

Este comando permite salvar un área de memoria de forma que se pueda ejecutar posteriormente con los comandos EXEC o EXEC_W. Tiene este formato:

```
SEXEC mdv1_prog,pricipio,longitud,espacio de datos
```

donde 'prog' es el nombre del fichero almacenado en el microdrive 1, 'principio' es la dirección desde donde se salva el programa (y donde será cargado después), 'longitud'

es la longitud total del área y 'espacio de datos' es la cantidad de memoria requerida por el programa salvado.

VER$

Esta variable contiene el número de versión del intérprete de BASIC contenido en la ROM, de modo que:

```
PRINT VER$
```

nos dará la versión de la ROM contenida en el QL.

4.3 El programa EXPERIMENTADOR

4.3.1 Introducción

Esta sección contiene el programa QL Experimentador. Debido a que el QL no contiene un Ensamblador residente, es necesario acceder al QDOS desde el BASIC (a menos que tenga usted la suerte de poseer un Ensamblador). Este programa le permite manejarse por dentro del sistema. No puede enseñarle todo, pero le proporciona un buen punto de partida. La parte más importante de esta sección es la de ayudarle a experimenter por usted mismo. La información contenida en las secciones de referencia de los capítulos 5, 6 y 7, le serán útiles para este propósito. Diviertase!

4.3.2 Listado del programa Experimentador

```
100 REMark Programa QL Experimentador
110 REMark
120 REMark por A C Dickens 1984
130 REMark
140 REMark Asignar memoria primer
150 ad=RESPR(100)
160 screen
170 REPeat main
180 valin
190 trpgen
200 peekout
210 END REPeat main
220 STOP
230 REMark ********** VALIN ****************
240 DEFine PROCedure valin
250 REMark VALIN toma los valores de entrada
260 valout
270 AT 2,7:INPUT a$
280 IF a$<>"" THEN trap=a$
```

```
290 AT 2,13:INPUT a$
300 IF a$<>"" THEN D0=hexdec(a$)
310 AT 8,4:INPUT a$
320 IF a$<>"" THEN D1=hexdec(a$)
330 AT 9,4:INPUT a$
340 IF a$<>"" THEN D2=hexdec(a$)
350 AT 10,4:INPUT a$
360 IF a$<>"" THEN D3=hexdec(a$)
370 AT 12,4:INPUT a$
380 IF a$<>"" THEN A0=hexdec(a$)
390 AT 13,4:INPUT a$
400 IF a$<>"" THEN A1=hexdec(a$)
410 AT 14,4:INPUT a$
420 IF a$<>"" THEN A2=hexdec(a$)
430 AT 15,4:INPUT a$
440 IF a$<>"" THEN A3=hexdec(a$)
450 END DEFine
460 REMark ********** VALOUT ***************
470 DEFine PROCedure valout
480 REMark Imprime los parámetros
490 REMark de la CALL
500 AT 2,7:PRINT trap
510 AT 2,13:Dechex(D0)
520 AT 8,4:Dechex(D1)
530 AT 9,4:Dechex(D2)
540 AT 10,4:Dechex(D3
550 AT 12,4:Dechex(A0)
560 AT 13,4:Dechex(A1)
570 AT 14,4:Dechex(A2)
580 AT 15,4:Dechex(A3)
590 END DEFine
600 REMark ********** PEEKOUT **************
610 DEFine PROCedure peekout
620 REMark PEEK desde memoria de
630 REMark los valores devueltos
640 RESTORE
650 DATA 24,8,24,9,24,10
660 DATA 24,12,24,13,24,14,24,15
670 REMark Imprime en decimal el error retornado
680 AT 17,18:PRINT PEEK_L(ad+16);" "
690 FOR p=1 TO 7
700 READ xp,yp
710 AT yp,xp:Dechex (PEEK_L(ad+16+4*p))
720 NEXT p
730 END DEFine
740 REMark ********** TRPGEN ***************
750 DEFine PROCedure trpgen
760 REMark Genera el TRAP requerido
770 init
780 CALL ad,D1,D2,D3,D0,0,0,0,A0,A1,A2,A3
```

```
790 END DEFine
800 REMark ********** INIT ******************
810 DEFine PROCedure init
820 REMark POKE el siguiente código
830 REMark en la primera parte de
840 REMark la memoria de la pantalla
850 REMark MOVE D4,D0
860 REMark TRAP #n
870 REMark MOVEML D0D1D2D3A0A1A2A3,abs.L
880 REMark MOVEQ #0,D0
890 REMark RTS
900 REMark Los datos se introducen aqui
910 POKE_W ad,8196
920 POKE ad+2,78
930 POKE ad+3,(64+trap)
940 POKE_W ad+4,18681
950 POKE_W ad+6,3855
960 POKE_L ad+8,ad+16
970 POKE_W ad+12,28672
980 POKE_W ad+14,20085
990 END DEFine
1000 REMark ********* HEXDEC ***************
1010 DEFine FuNction hexdec(hex$)
1020 REMark Convierte hex a decimal
1030 dec=0
1040 FOR n=1 TO LEN(hex$)
1050 LET v=CODE(hex$(n))
1060 IF v>64 THEN v=v-55
1070 IF v>47 THEN v=v-48
1080 dec=dec+2^((8-n)*4)*v
1090 NEXT n
1100 RETurn dec
1110 END DEFine
1120 REMark ********* DECHEX ***************
1130 DEFine PROCedure Dechex(dec)
1140 REMark Convierte decimal a hex
1150 LOCal x,y
1160 x=dec
1170 c$="00000000"
1180 FOR y=1 TO 8
1180 c=INT(x/(16^(8-y)))
1200 x=x-16^(8-y)*c
1210 IF c<10 THEN
1220 c$(y)=CHR$(48+c)
1230 ELSE
1240 c$(y)=CHR$(55+c)
1250 END IF
1260 NEXT y
1270 hex$=c$
1280 PRINT hex$
```

```
1290 END DEFine
1300 REMark ********* SCREEN ***************
1310 DEFine PROCedure screen
1320 CLS
1330 D0=0
1340 D1=0
1350 D2=0
1360 D3=0
1370 A0=0
1380 A1=0
1390 A2=0
1400 A3=0
1410 trap=1
1420 PAPER 4
1430 CLS
1440 PAPER 7:INK 4
1450 AT 0,0:PRINT " QDOS EXPERIMENTADOR v1.0  por
ACD 1984"
1460 PRINT
1470 INK 2
1480 AT 4,1:PRINT "Introduzca los valores en HEX
con mayúsculas"
1490 PAPER 1:INK 7
1500 AT 6,1:PRINT "CALL, parámetros"
1510 AT 6,21:PRINT "Valores Retornados"
1520 AT 2,1:PRINT "Trap #";trap
1530 AT 2,9:PRINT " D0"
1540 FOR pos = 0 TO 20 STEP 20
1550 AT 8,pos:PRINT " D1"
1560 AT 9,pos:PRINT " D2"
1570 AT 10,pos:PRINT " D3"
1580 AT 12,pos:PRINT " A0"
1590 AT 13,pos:PRINT " A1"
1600 AT 14,pos:PRINT " A2"
1610 AT 15,pos:PRINT " A3"
1620 NEXT pos
1630 AT 17,1:PRINT "Error retornado: "
1640 AT 18,1:PRINT " (valor en do)"
1650 END DEFine
```

4.3.3 Como trabaja el Experimentador

El programa funciona en un bucle contínuo. Usted introduce los valores que deben ser pasados a un TRAP, el Experimentador llama al TRAP, usted introduce más valores y así sucesivamente.

La operación del programa se realiza en dos títulos generales; inicialización y experimento.

La inicialización consiste en la asignación de memoria para la subrutina en código máquina. Esta subrutina es un pequeño programa en código 68008 para generar el TRAP y devolver al BASIC los valores de los registros. Para reservar 100 octetos de memoria se ha usado RESPR(100). A continuación se llama al procedimiento *screen*. Se inicializan todas las variables a sus valores por defecto y se imprimen los rótulos fijos de la pantalla.

El experimento se realiza mediante el bucle *principal*, por medio de una sentencia REPEAT. Este bucle se ejecuta continuamenten repitiendo los ciclos de tomar los datos del TRAP, generar el TRAP e imprimir los resultados en la pantalla.

Se usan los siguientes procedimientos:

valin se usa para tomar todos los parámetros desde el teclado. Pulsando la tecla ENTER se toma el valor por defecto. Si no, se deben introducir los ocho dígitos de longitud de la palabra larga del parámetro. Tenga en cuenta que deben ser introducidos en hexadecimal (usando solamente letras mayúsculas).

valout se usa para imprimir los valores actuales de todos los registros de dirección y de datos en sus lugares correspondientes en la pantalla.

peekout se usa para recoger los valores de los registros de dirección y datos, retornados por la subrutina especial en código máquina (ver más adelante), e imprimirlos en la pantalla.

trpgen es el que lanza el TRAP mediante la llamada a una pequeña subrutina en código máquina. Esta subrutina se explica en **init.**

hexdec es una función que convierte un número hexadecimal dentro de una cadena ASCII, en un número decimal.

dechex es un procedimiento que convierte el parámetro decimal en una cadena hexadecimal ASCII. Esta cadena (de 8 caracteres de longitud) se imprime en la pantalla en la posición del cursor.

screen inicializa las variables y prepara el formato de la pantalla para que esta pueda aceptar los valores de los parámetros.

init inicializa la subrutina en código máquina en la memoria y la llama. El espacio para init se reserva al principio del programa usando RESPR. La rutina es:

```
        MOVE D4,D0
        TRAP #N
        MOVEM.L D0-D3/A0-A3,DATA
        MOVEQ #0,D0
        RTS
*
DATA    aquí van los valores retornados
```

El MOVE D4,D0 transfiere el contenido del registro D4 dentro del registro D0. Esto es necesario porque el comando CALL del BASIC no puede pasar D0.

EL TRAP#N genera el TRAP número N. Todos los valores de los registros, que se introdujeron por la pantalla, se pasan en este TRAP.

La instrucción MOVE.L es un movimiento múltiple. Almacena todos los registro retornados, D0,D1,D2,D3,A0,A1,A2,A3 en la memoria, empezando en DATA. Esto permite al programa BASIC hacer PEEK de todas las variables de retorno para que se puedan imprimir en la pantalla.

El MOVEQ #0,D0 se usa para poner el registro D0 a ceros. Si no se hiciera esto, cualquier error devuelto por el TRAP se imprimiría en la pantalla al retorno al BASIC, y cesaría la ejecución.

El RTS devuelve control al BASIC.

4.3.4 Experimentos

Ahora que ya conocemos el funcionamiento del programa Experimentador, empieza lo divertido. Antes de nada, el programa debe teclearse en el QL y salvarse en microdrive. Después ya se puede ejecutar.

Asumiendo que se ha introducido correctamente el programa, al ejecutarlo debe mostrarnos una pantalla como la que tenemos a continuación:

```
QDOS EXPERIMENTADOR v1.0  por ACD 1984

Trap #1    D0 00000000

Introduzca los valores en HEX con mayúsculas

CALL, parámetros     Valores retornados

D1 00000000          D1
D2 00000000          D2
D3 00000000          D3

A0 00000000          A0
A1 00000000          A1
A2 00000000          A2
A3 00000000          A3

Error retornado:
 (valor en D0)
```

El Experimentador está dispuesto para usarse. Se puede teclear el número del TRAP y el contenido de los registros que se van a pasar al TRAP. Tenga en cuenta que todos los valores se deben teclear en hexadecimal y con mayúsculas. Es conveniente usar el CAPS LOCK para seleccionrlas. Cuando se haya tecleado una entrada completa, pulse ENTER. Si se pulsa ENTER sin cambiar los valores, asume los anteriores.

Se deben usar las capítulos 5, 6 y 7 junto con esta sección para ver exactamente lo que se está haciendo.

Obtener información del estado del sistema

Como primer ejemplo, intente usar el TRAP #1 con D0=0. Este es un *TRAP del Gestor* llamado MT.INF, que devuelve información del sistema. No requiere que se le pase ningún parámetro, así que pulsaremos simplemente la tecla ENTER hasta que llegue a A3. Pulsando ENTER de nuevo hará que se genere el TRAP. Los valores devueltos que aparecerán en la pantalla, serán similares a los que se ven más adelante.

Si nos referimos a la sección 5.4, veremos que el identificador del trabajo actual es devuelto en D1, el número ASCII de la versión en D2 y el apuntador a las variables del sistema en A0. El BASIC es el Job 0 y este programa está ejecutándose desde el BASIC, por lo tanto D1 está a ceros. El número de versión está en D2. Convirtiendo los valores de HEX a ASCII veremos que la versión del QDOS de este QL es 1.01 (una versión un poco antigua). Puede ser diferente dependiendo del QL donde se ejecute. El valor en A0 es

$28000, un octeto por encima de la parte superior de la pantalla en el mapa de memoria del QL. Las variables del sistema comienzan en este punto.

```
QDOS EXPERIMENTADOR v1.0  por ACD 1984

Trap #1    D0 00000000

Introduzca los valores en HEX con mayúsculas

CALL, parámetros     Valores retornados

D1 00000000          D1 00000000
D2 00000000          D2 312E3032
D3 00000000          D3 00000000

A0 00000000          A0 00028000
A1 00000000          A1 00000000
A2 00000000          A2 00000000
A3 00000000          A3 00000000

Error retornado: 0
 (valor en D0)
```

Finalmente, el error retornado es cero. Esto significa que el TRAP no ha generado ningún error. Cuando se generan errores, aparece un número negativo en este punto (ver Apéndice G para los códigos de error).

Suspender el BASIC

```
QDOS EXPERIMENTADOR v1.0  por ACD 1984

Trap #1    D0 00000008

Introduzca los valores en HEX con mayúsculas

CALL, parámetros     Valores retornados

D1 00000000          D1 00000000
D2 00000000          D2 00000000
D3 00000100          D3 00000100

A0 00000000          A0 00003DA0
A1 00000000          A1 00000000
A2 00000000          A2 00000000
A3 00000000          A3 00000000

Error retornado: 0
 (valor en D0)
```

El TRAP del Gestor MT.SUSJB, se usa para suspender un trabajo por un periodo de tiempo determinado. En esta rutina ponemos D0 con $08. El periodo de tiempo se pone en unidades de cuadros (cincuentaavos de segundo), por lo que $100 (256 en decimal) son 5 segundos). Este valor se pasa en el registro D3. Intente generar este TRAP. Tenga en cuenta que el cursor no reaparecerá hasta pasados los cinco segundos. Esto es debido a que el Job 0 (BASIC) ha sido suspendido durante 5 segundos. Se puede suspender por periodos más largos (hasta 10 minutos) o indefinidamente si D3 es $FFFF.

Sacar un carácter a la pantalla

El canal de comandos (las cuatro líneas inferiores de la pantalla) están definidas como canal 0 en BASIC. Hay un TRAP, llamado IO.SBYTE, que nos permite mandar caracteres a un canal de salida. IO.SBYTE es uno de los TRAPs de utilización de E/S (3). El octeto inferior de D1 contiene el código Hex del carácter a imprimir. Si ponemos $41 en D1, se imprimirá el carácter 'A' en la pantalla (suponiendo que se haya puesto un 0 en A0 para el canal de la parte inferior de la pantalla).

```
QDOS EXPERIMENTADOR v1.0  por ACD 1984

Trap #3   D0 00000005

Introduzca los valores en HEX con mayúsculas

CALL, parámetros     Valores retornados

D1 00000041          D1 00000041
D2 00000000          D2 00000000
D3 0000FFFF          D3 0000FFFF

A0 00000000          A0 00000002
A1 00000000          A1 00000000
A2 00000000          A2 00000000
A3 00000000          A3 00000000

Error retornado: 0
 (valor en D0)
```

Mover una ventana en la pantalla

Se puede mover lateralmente todo el contenido de una ventana un número de puntos determinado. Para ello se usa el TRAP SD.PAN . Como en el últimp TRAP de E/S, D3 contiene el *tiempo* y A0 la *identificación del canal*. D1 contiene el número de puntos requeridos para el movimiento,un valor positivo la

mueve a la derecha el número de puntos. El ejemplo que vemos a continuación mueve a la derecha 64 puntos ($40). Los valores negativos harán que se mueva hacia la izquierda.

```
QDOS EXPERIMENTADOR v1.0  por ACD 1984

Trap #3   D0 0000001B

Introduzca los valores en HEX con mayúsculas

CALL, parámetros     Valores retornados

D1 00000040          D1 000000D8
D2 00000000          D2 00000000
D3 0000FFFF          D3 0000FFFF

A0 00000000          A0 00000000
A1 00000000          A1 00028E36
A2 00000000          A2 00000000
A3 00000000          A3 00000000

Error retornado: 0
 (valor en D0)
```

Poner el color de la tinta

El TRAP SD.SETIN pone el color de la tinta que se usa para imprimir caracteres.

```
QDOS EXPERIMENTADOR v1.0  por ACD 1984

Trap #3   D0 00000029

Introduzca los valores en HEX con mayúsculas

CALL, parámetros     Valores retornados

D1 00000005          D1 000000D5
D2 00000000          D2 00000000
D3 0000FFFF          D3 0000FFFF

A0 00000000          A0 00000000
A1 00000000          A1 00028E3E
A2 00000000          A2 00000000
A3 00000000          A3 00000000

Error retornado: 0
 (valor en D0)
```

Si le llamamos con los parámetros que vemos en nuestro

ejemplo, todos los caracteres que se impriman en adelante tendrán el color cyan. Para verlo, intente llamar a IO.SBYTE de nuevo. Los caracteres se imprimirán en cyan.

```
QDOS EXPERIMENTADOR v1.0  por ACD 1984

Trap #3   D0 00000029

Introduzca los valores en HEX con mayúsculas

CALL, parámetros     Valores retornados

D1 00000005          D1 00000005
D2 00000000          D2 00000000
D3 0000FFFF          D3 0000FFFF

A0 00000000          A0 00000000
A1 00000000          A1 00028E3E
A2 00000000          A2 00000000
A3 00000000          A3 00000000

Error retornado: 0
 (valor en D0)
```

Operaciones más complicadas

Los cinco ejemplos anteriores son bastantes sencillos de realizar. Si los ha seguido junto con los capítulos 5 y 7, probablemente habrá visto que hay bastantes más TRAPs. Gran parte de ellos se pueden llamar desde el programa Experimentador, pasando en los registros todos los parámetros necesarios.

Sin embargo hay un grupo de TRAPs que requieren más información. Esta suele almacenarse en algún lugar de la memoria al que apunta un (o varios) registro de dirección. Un ejemplo típico de este tipo de TRAPs es SD.RECOL (TRAP #3 con D0=$26). Este TRAP cambia el color de una ventana. A cada color de la ventana se le puede asignar otro distinto. La lista que define el nuevo color de cada uno de los existentes es apuntado por el registro A1. Detrás del código máquina de nuestra rutina residente en el area de procedimientos, se pueden almacenar estos datos.

Ya que la variable de BASIC 'ad' contiene la dirección de la subrutina, y el espacio que hay 60 octetos más arriba no se usan, podemos teclear:

```
POKE (ad+60),7
POKE (ad+61),6
POKE (ad+62),5
POKE (ad+63),4
POKE (ad+64),3
POKE (ad+65),2
POKE (ad+66),1
POKE (ad+67),0
dechex(ad+60)
```

introducirá una tabla de valores de conversión, en la memoria. El procedimiento dechex devolverá el valor hexadecimal del principio de la tabla. Asumiendo que esta fuera $FFCA, el ejemplo que vemos abajo cambiará los colores del dispositivo conectado al canal 0 (parte inferior de la pantalla). Tenga en cuenta que no necesita ejecutar de nuevo el Experimentador, simplemente teclee GO TO 160 para evitar que se asigne nueva memoria con el comando RESPR(100).

```
QDOS EXPERIMENTADOR v1.0  por ACD 1984

Trap #3   D0 00000026

Introduzca los valores en HEX con mayúsculas

CALL, parámetros    Valores retornados

D1 00000000         D1 000000D8
D2 00000000         D2 00000000
D3 00000000         D3 00000000

A0 00000000         A0 00000000
A1 0003FFCA         A1 0003FFCA
A2 00000000         A2 00000000
A3 00000000         A3 00000000

Error retornado: 0
 (valor en D0)
```

Las ideas que se presentan en adelante en este capítulo, no son fáciles de realizar con el BASIC. Para conseguir todas las ventajas del QDOS necesitaremos un Ensamblador. Ahora vamos a dejar el BASIC y nos introduciremos en el lenguaje Ensamblador.

4.4 Programando en Ensamblador

Los ejemplos de esta seción le ayudarán a clarificar algunos de los problemas y errores de concepto que le puedan surgir. Cada ejemplo es un programa completo, y se podrá

ejecutar si se introduce con un Ensamblador adecuado.

4.4.1 Un reloj en la pantalla

Este programa produce un reloj en la pantalla que se va actualizando continuamente. Para comprender este programa, le sugiero que lea las explicaciones junto con los comentarios del programa. Se hacen continuas referencias a los capítulos de Utilidades y TRAPs.

```
*
* reloj - un reloj en lo alto de la ventana 0 (modo monitor)
*
ME       EQU     -1
MT.SUSJB EQU     $08
MT.PRIOR EQU     $0B
MT.RCLCK EQU     $13
IO.SSTRG EQU     $07
SD.TAB   EQU     $11
*
UT.SCR   EQU     $C8
CN.DATE  EQU     $EC
*
         BRA.S   RELOJ           bifurca a código reloj
         DS.B    4               y se salta
         DC.W    $4AFB           ind. de Job en octetos 6 y 7
         DC.W    5               caracteres en nombre de Job
         DC.W    'Clock'         seguido de caracteres ASCII
*
CLOCK
         SUBA.L  A6,A6           limpia A6 para CN_DATE y
         MOVEQ   #MT.PRIOR,D0    pone prioridad
         MOVEQ   #ME,D1          ... de este Job
         MOVEQ   #1,D2           ... a 1 (la menor)
         TRAP    #1
*
         MOVE.W  UT.SCR,A2       abrir ventana / sacar reloj
         LEA     SCR(PC),A1      dirección de definición
         JSR     (A2)
         MOVE.L  A0,A4           salvar identificación canal
*
CLOCK_LOOP
         MOVEQ   #MT.RCLCK,D0    leer hora en D1
         TRAP    #1
*
         LEA     BUF_TOP(PC),A1 usar buffer de arriba abajo
         MOVE.W  CN.DATE,A2      para convertir fecha
         JSR     (A2)
*
```

```
        MOVEQ   #IO.SSTRG,D0    mandar la cadena resultante
        MOVE.W  (A1)+,D2        de 20 caracteres
        MOVEQ   #-1,D3          ... sin perder tiempo
        MOVE.L  A4,A0           a nuestra ventana
        TRAP    #3
*
        MOVEQ   #SD.TAB,D0      ahora, mandar el cursor al
        MOVEQ   #0,D1           ... principio de la línea
        TRAP    #3
*
        MOVEQ   #MT.SUSJB,D0    susp. para reducir carga ES
        MOVEQ   #ME,D1
        MOVEQ   #5,D3           por una décima de segundo
        SUBA.L  A1,A1           limpiar indicadores
        TRAP    #1

        BRA.S   CLOCK_LOOP
*
SCR
        DC.B    0               no borde
        DC.B    0               no color borde
        DC.B    4               fondo verde
        DC.B    0               caracteres negros
*
* definición de la ventana para modo monitor
*
        DC.W    120     puntos ancho (20 car./6 de ancho)
        DC.W    10      10 puntos de alto
        DC.W    512-120 120 puntos del margen derecho
        DC.W    206     y en lo alto de la ventana #0
*
* definición alternativa del modo TV
*
*       DC.W    240     puntos ancho (20 car./12 ancho)
*       DC.W    10      10 puntos alto
*       DC.W    480-240 240 puntos del margen derecho
*       DC.W    216     y en lo alto de la ventana #0
BUFFER
        DS.B    22
BUF_TOP
        END
```

Antes de nada, el programa pone la prioridad del Job (trabajo) a la más baja posible (1) usando MT.PRIOR. Tenga en cuenta que la identificación del Job (Job ID) se pasa a MT.PRIOR en D1 como '-1'. Esto permite que MT.PRIOR cambie la prioridad actual del Job.

La utilidad UT.SCR (accedida mediante vector), se usa después para abrir una ventana en la pantalla. La ventana se

puede definir para usarla en un monitor o en un televisor, dependiendo del bloque de definición que se ponga en SCR.

La parte principal de la rutina está situada dentro de CLOCK_LOOP. Se lee el valor binario del reloj, se convierte en ASCII y se manda a la pantalla dentro del bucle. MT.RCLK toma la hora de RTC dentro de D1. Se deja un espacio en memoria de 22 octetos para convertirlo en una cadena ASCII. CN.DATE se usa para convertir la fecha de binario a ASCII. IO.SSTRG saca después la cadena a la pantalla.

Observe que se usa MT.SUSJB al final del bucle. Este TRAP del gestor suspende el Job del reloj durante una décima de segundo (ya que no tiene caso actualizar continuamente el reloj, que cambia cada segundo). Todos los trabajos que puedan ser suspendidos se deben suspender para que otros trabajos que necesitan más atención puedan disponer del tiempo del procesador.

4.4.2 El programa autoduplicable

Este programa ilustra la enorme potencia de multitarea en el QL. El propósito de este programa es dibujar un grupo de burbujas en una línea desde el centro de la pantalla hasta el borde. Sin embargo, tiene una característica especial. Después de que se hayan dibujado 25 burbujas, el programa se duplica por sí mismo. La única diferencia entre la versión original y la nueva es que producen burbujas de diferentes colores que se mueven en línea recta. Cada duplicado se divide hasta que hay gran cantidad de burbujas separadas, moviendose hacia los bordes de la pantalla.

```
*
* Programa para preparar el programa autoduplicable
*
* Los duplicados dibujan una línea de puntos en una direc-
* ción y de un color determinados por un número aleatorio en
* el OS. Este número es actualizado cada vez que se usa.
*
ME       EQU     -1
MT.CJOB  EQU     $01
MT.RJOB  EQU     $04
MT.FRJOB EQU     $05
MT.SUSJB EQU     $08
MT.ACTIV EQU     $0A
MT.PRIOR EQU     $0B
SD.FILL  EQU     $2E
UT.SCR   EQU     $C8
SV_RAND  EQU     $2802E
JB_A7    EQU     $5C
```

```
*
        BRA.S   MCLONE          comienzo copia maestra
        DS.B    4
        DC.W    $4AFB
        DC.W    6,'Mclone'
MCLONE
        LEA     SCR(PC),A1      prepara ventana toda la
        MOVE.W  UT.SCR,A2       pantalla
        JSR     (A2)
        MOVE.W  #254,D4         punto comienzo en el centro
        MOVEQ   #127,D5         ... X = 254, Y = 127
        MOVE.L  A0,A4           pone A4 para crear
        BSR.W   CREATE_JOB
        MOVE.L  D1,D6           salvar job ID nuevo
*
KILL
        MOVEQ   #MT.SUSJB,D0    suspendeme
        MOVEQ   #ME,D1
        MOVEQ   #50,D3          por un segundo
        SUBA.L  A1,A1           borrar indicadores
        TRAP    #1
*
        MOVEQ   #MT.RJOB,D0     ahora intento eliminar todos
        MOVE.L  D6,D1           mis jobs
        TRAP    #1
        TST.L   D0              están todos inactivos?
        BNE.S   KILL
*
        MOVEQ   #MT.FRJOB,D0    ahora me elimino a mi mismo
        MOVEQ   #ME,D1
        MOVEQ   #0,D3           no error
        TRAP    #1              (no puede volver)
*
SCR     DC.B    0,0,208,7       no borde, blanco en rojo
        DC.W    512,256,0,0     oscuro toda la pantalla
*
* Programa auto duplicable
*
SCLONE
        ADDQ    #2,A7           sacar parámetros de la pila
        MOVE.L  (A7)+,A4        primero id del canal
        ADDQ    #2,A7
        MOVEM.W (A7),D4/D5      después coordenadas X e Y
        MOVE.L  #$00040003,-(A7) pone bloque 4X3 puntos
        MOVE.L  A7,A5           parámetros bloque en pila
*
        MOVE.W  SV_RAND,D6      toma num. aleatorio sistema
        ROR.W   #2,D6           hace cambiar byte +signific.
        MOVE.W  D6,D7           ... incrementa Y
        ROR.W   #4,D6           rota e incrementa X
```

```
        MOVE.W  D6,SV_RAND      actualiza numero aleatorio
*
        MOVEQ   #7,D1           pone color de bloque
        AND.B   D7,D1           a 0 y a 7
*
        EXT.L   D6              incremeto en palabras largas
        EXT.L   D7
        ASL.L   #2,D6           hasta 4 en la palabra más
        ASL.L   #2,D7           ... significativa
        MOVEM.L D6/D7,-(A7)     pone incrementos en la pila
*
        MOVE.L  D1,D6           y salva el color
SET_LOOP
        MOVEQ   #24,D7          dibuja 25 burb. entre copias
LOOP
        SWAP    D4              mueve X,Y a palab. + signif.
        SWAP    D5
        ADD.L   (A7),D4         añade los incrementos
        ADD.L    4(A7),D5
        SWAP    D4              y muevelos atrás de nuevo
        SWAP    D5
        MOVEM.W D4/D5,4(A5)     pone X,Y actuales
*
        BSR.S   WBLOB           dibuja burbuja
        TST.L   D0              lo ha hecho bien?
        BNE.S   DONE            ... no, parar
*
        MOVEQ   #MT.SUSJB,D0    suspendeme
        MOVEQ   #ME,D1
        MOVEQ   #10,D3          durante 10 cuadros
        SUBA.L  A1,A1           limpia indicadores
        TRAP    #1
*
        DBRA    D7,LOOP         y repitelo de nuevo
*
        BSR.S   CREATE_JOB      crea una copia
        BRA.S   SET_LOOP        y la ejecuta
*
DONE
        MOVEQ   #MT.PRIOR,D0    desactivame
        MOVEQ   #ME,D1
        MOVEQ   #0,D2
        TRAP    #1
*
WBLOB
        MOVEQ   #SD.FILL,D0     llena las burbujas
        MOVE.L  D6,D1           con un color predefinido
        MOVEQ   #-1,D3          espera
        MOVE.L  A4,A0           ID canal
        MOVE.L  A5,A1           dirección del bloque lleno
```

```
        TRAP    #3
        RTS
*
* Produce un duplicado
*
CREATE_JOB
        MOVEQ   #MT.CJOB,D0     crea un nuevo job
        MOVEQ   #ME,D1          de mi propiedad
        MOVEQ   #16,D2          espacio para nombre del job
        MOVEQ   #20,D3          espacio para la pila
        SUB.L   A1,A1           dirección inicio 0
        TRAP    #1
        TST.L   D0              tuvo exito?
        BNE.S   CR_EXIT         ... no darselo de momento
*
        MOVE.L  A0,A6           dirección base para el job
        MOVE.W  #$4EF9,(A6)+    JMP.L
        LEA     SCLONE(PC),A1
        MOVE.L  A1,(A6)+        a sclone
        MOVE.W  #$4AFB,(A6)+    poner id de job estándar
        MOVE.L  #$00065343,(A6)+ y nombre de job
        MOVE.L  #$6C6F6E65,(A6)
*
        LEA     36(A0),A6
        MOVEM.W D4/D5,-(A6)     pone posición actual en pila
        MOVE.W  #4,-(A6)        (4 octetos de longitud)
        MOVE.L  A4,-(A6)        e ID canal
        MOVE.W  #1,-(A6)        (uno)
*
        MOVEQ   #MT.PRIOR,D0    usa la CALL poner prioridad
        MOVEQ   #0,D2           para encontrar el apuntador
        TRAP    #1              ... de la pila salvada
        MOVE.L  A6,JB_A7(A0)    y restaurarlo
*
        MOVEQ   #MT.ACTIV,D0    activa el job
        MOVEQ   #1,D2           en prioridad 1
        MOVEQ   #0,D3
        TRAP    #1
CR_EXIT
        RTS
        END
```

El programa se puede dividir en dos partes principales. La primera se llama MCLONE (la copia maestra) y prepara la pantalla completa como una ventana, después comienza una copia en el centro. Después de esperar un segundo, la maestra comprueba si se pueden eliminar todas las copias. Continua haciendo esto cada segundo desde este punto hacia fuera hasta que todas las copias han llegado al borde de la

pantalla. En este punto, se destruye a sí mismo.

En la segunda parte del programa, los duplicados están ocupados en producir más copias. El color de las burbujas que se van a dibujar, viene dado por un número aleatorio seleccionado y el número aleatorio que se mantiene en el espacio de variables. Las burbujas se dibujan 25 al mismo tiempo dentro de LOOP. Cuando se ha terminado, se crea un nuevo duplicado. El nacimiento del job duplicado tiene lugar en CREATE_JOB. Para ello se usa el TRAP del gestor MT.CJOB.

Los jobs se desactivan cuando llegan al borde de la pantalla. El maestro termina el job. A lo largo de todo el proceso, solo hay una copia(reentrante) del código. Cada copia subsidiaria son solo datos.

5 Los TRAPs del Gestor

4.1 Introdución a los TRAPs del gestor

Este capítulo trata de la asignación de los recursos disponibles en el QL. Si no ha leido y comprendido el capítulo de introducción al QDOS (capítulo 3), este es un buen momento para hacerlo.

Los recursos disponibles en el QL se dividen en dos áreas. La primera es la memoria, y la segunda está constituida por otra piezas de 'hardware' que está disponible para el QL, incluyendo la CPU 68008. Todos los programas que se ejecuten en el QL necesitan memoria en la que almacenar los datos asociados con la tarea que se ejecuta. Para esto hay una serie de rutinas que asignan memoria a los programas para el uso del programa y/o almacenamiento de datos. Estas rutinas se tratan en la sección 5.2. Hay otro grupo de rutinas para la gestión de los dispositivos 'hardware' que no sean la memoria. Estos dispositivos pueden ser los componentes internos estándar, como el reloj, IPC (controlador inteligente de la periferia), o dispositivos externos como disco duro o puertas paralelas para impresora. El 'hardware' interno se trata con una serie de '*controladores de dispositivos*' que se encuentran dentro del QDOS. El 'harware' externo puede ser conectado fácilmente al sistema por un grupo de vias bien definidas. Este aspecto del gestor del sistema se trata en la sección 5.3.

Generalmente, la mayoría de la gestión de los recursos del sistema se realiza mediante los TRAPs del gestor, a los que se accede mediante el TRAP #1. Se puede realizar, también, una cierta cantidad de manejo de recursos, mediante las rutinas de utilidad por vector que se describen en el capítulo 8. Estas las pueden usar los 'controladores de dispositivos' llamados con el TRAP #2 o el TRAP #3, pero NO se deben usar directamente desde el código del usuario.

5.2 Gestión de memoria en el QL

La gestión de memoria se ocupa generalmente de la protección y asignación de la memoria. En el QL, solo se encarga de la asignación. En la practica, esto significa que se puede hacer fallar al sistema entero mediante la introducción de datos en el área de memoria usado como espacio de trabajo del QDOS (en un sistema de 'hardware' más sofisticado no se podría hacer). Ya que pueden existir bloques

importantes de programa y datos en cualquier parte de la memoria del QL, es *esencial* que se use la memoria solamente de la forma requerida por el QDOS. Tenemos gran cantidad de potentes TRAPs para realizar la asignación de memoria. El tipo particular de asignación depende de las demandas exactas del programa y/o datos almacenados en memoria. Las siguientes secciones nos explican los diferentes tipos de asignación de memoria que se pueden usar.

5.2.1 Procedimientos residentes y su memoria

Ya hablamos, en el capítulo 3, de los procedimientos residentes como una forma muy potente de programar. Se situan en la parte alta del mapa de memoria (ver figura 3.1). La razón de su potencia reside en que la memoria en la que se sitúan se asigna solamente en el momento de puesta en marcha del QL. Una vez que se ha asignado el área de procedimientos, no se puede borrar ni extender. La razón para esto es que inmediatamente debajo está el area de programas transitorios. Este área no puede ser reasignada, y por lo tanto no puede moverse, como se requeriría para cambiar el tamaño del área de procedimientos residentes. Solamente en la circunstancia excepcional de que no hubiera procedimientos transitorios, se podría cambiar el área de procedimientos residentes.

5.2.2 Jobs como programas transitorios

Como se explicó en el capítulo 3, el área de memoria de programas transitorios se puede alterar dinámicamente. Se pueden añadir nuevos programas (en cuyo caso se expande el área) o borrar programas viejos (contrayendose el área). Los programas que se pueden introducir en este área se suelen llamar *Jobs*.

Durante la vida de un Job pueden suceder una gran cantidad de eventos. Algunos de estos son esenciales e inherentes al hecho de la existencia del Job (ej. creación del Job), otras son opcionales (como cambiar el Job de prioridad o borrarlo).

El área de memoria asignada a un Job se puede considerar en dos partes. El código del Job ocupa la primera parte de la memoria asignada a él. Los datos usados por el Job ocupan el resto, desde la parte superior del código hasta la parte superior del área. Estas áreas se especifican cuando se crea el Job incialmente. A tiempo de creación del Job se puede definir éste como una unidad independiente por sí mismo o

como una subunidad de otro Job. Los Jobs que son *propiedad* de otros Jobs serán borrados cuando lo sea su propietario.

Para permitir al usuario seguir el desarrollo de todos estos Jobs, a cada uno se le asigna un marcador único llamado *Job ID*. El primer Job que se crea es el Job 0, y siempre es el intérprete de BASIC. A los Jobs posteriores se les asigna su propio número. La numeración de los Jobs es totalmente automática, y el número particular de un Job depende exclusivamente del número de Jobs que haya en el QL cuando se crea. La palabra inferior del 'Job ID' contiene el número de Job y se usa como un índice dentro de la tabla de Jobs. La palabra superior del 'Job ID' es una *etiqueta*. Cada nuevo Job se distingue con su propia etiqueta que es uno mayor que la última asignada.

Una vez que se ha creado el Job, puede estar en uno de tres estados claramente diferenciados. Puede estar *activo*, *suspendido* o *inactivo*.

Un Job *activo* comparte los recursos de la CPU con otros Jobs. Para determinar la cantidad de tiempo de CPU que se debe asignar a un Job particular, a cada uno se le asigna una prioridad entre 1 y 127. La mayor es 127 y la menor (para un Job activo) es 1. De esta forma, el planificador, decide cual de los Jobs activos debe disponer de la CPU por un tiempo determinado.

Un Job *suspendido* es uno que está activo todavia, pero está suspendido mientras espera por entrada, salida u otro Job que no está aún preaparado. Por lo tanto, el planificador no asigna tiempo de CPU a un Job suspendido hasta que el proceso por el que está esperando, se ha completado. Entonces se reactiva el Job.

Un Job *inactivo*, es aquel cuya prioridad ha sido puesta a 0. A este tipo de Job no se le asigna tiempo de CPU. Un Job inactivo está presente en el área de programas transitorios, y puede ser reactivado en cualquier momento por otro Job.

Finalmente, un Job puede ser borrado del área de programas transitorios. Hay dos tipos básicos de borrado. El primer tipo solo permite borrar un Job si está inactivo. El otro permite forzar el borrado del Job (aunque esté activo). Cuando se borra un Job, se borran también todos aquellos de los que es propietario.

5.2.3 Asignación de memoria para programas BASIC

El intérprete de BASIC se inicia como Job 0. Como en los demás Jobs, se requiere memoria para almacenar el programa (en este caso un programa BASIC) y los datos. A diferencia de otros Jobs, a los que se les asigna una cantidad fija de memoria, al Job 0 se le permite asignar más espacio de memoria por sí mismo. NINGUN otro Job puede expandirse dinamicamente como éste. Existen dos TRAPs del gestor (D0=16 y 17) para expandir y contraer la memoria disponible para el BASIC.

5.2.4 Asignación en el Area Común

Si un Job requiere más memoria extra (sobre la que se le asignó cuando fué creado), se le puede asignar en el área común. El espacio en este área pertenece a este Job hasta que es *liberada*, o el Job es borrado. El área común se usa también para almacenar bloques de definición de canal, datos, y almacenamiento de trabajo para el IOSS (sussistema de E/S).

La secuencia de la figura 5.1 ilustra una progresión típica del almacenamiento en el área común. Inicialmente, este área se encuentra vacía (fig 5.1a). El Job 'A' decide entonces que le gustaría usar parte de este área, así que le dice al QDOS que le asigne una parte usando MT.ALCHP (TRAP #1 con D0=$18). El área asignada aparece sombreada en la figura 5.1b. Ahora el Job 'B' pide otra parte de este área. De nuevo, el QDOS aloca algo de memoria a este Job, justo encima del área perteneciente al Job 'A' (ver fig 5.1c). Otro Job (Job 'C') requiere ahora algo de memoria. Esta es alocada justo encima de la del Job 'B' (ver fig 5.1d).

Algún tiempo después, el Job 'B' es borrado del sistema. El QDOS conoce exáctamente a quien pertenece cada bit del área común, así que sabe que el área asignada al Job 'B' no se va a necesitar más. El área de este job es borrada para que pueda ser usada por otro Job (ver fig 5.1c). Otro Job (Job 'D') hace una petición de memoria en el área común. El QDOS encuentra la parte que le perteneció al Job 'B', disponible, de modo que se la asigna al Job 'D' (ver fig 5.1f). El Job 'E'hace una petición de un área más grande. El QDOS se da cuenta que el único espacio suficientemente grande está sobre la del Job 'C' por lo que se la asigna al Job 'E'. El Job 'A' ha llegado a un punto en el que necesita más memoria. Hace una nueva petición usando MT.ALCHP. El QDOS busca por el área común y encuentra que puede caber entre la parte alta de 'D' y la baja de 'C'. Consecuentemente el nuevo

espacio se asigna ahí.

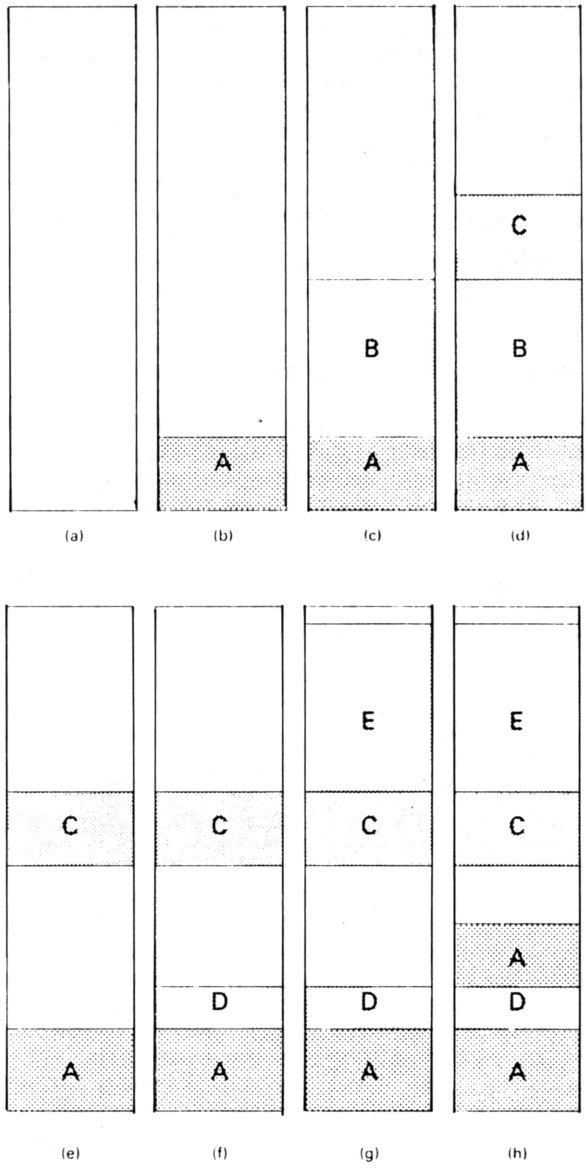


Figura 5.1 - Uso del área común

Este proceso continúa mientras la máquina permanece encendida. Se puede apreciar que el área común se fragmenta terriblemente si los Jobs la asignan continuamente, devolviendo y reasignando espacio. Hay una considerable sobrecarga por cada espacio asignado, ya que el sistema operativo mantiene 16 octetos de información de cada sección (para poder disponer de las áreas de los Jobs que son borrados). Un modo más eficiente de asignar área común es tomar la mayor cantidad de memoria que se va a necesitar. El Job puede preparar su propia área común en este espacio, sin recargar al sistema operativo.

Hay un gran número de TRAPs y utilidades que ayudan a los programas del usuario a gestionar sus propias áreas.

5.2.5 Asignación del área de usuario

El área común se ha estudiado en la sección 5.2.4. Esta sección nos muestra como el sistema operativo puede hacer una gran cantidad de trabajo extra de preparación si se asignan áreas pequeñas de memoria en el área común. El concepto de área de usuario es muy útil. Básicamente involucra al Job en la asignación de un área grande de área común o de su propia memoria asignada. Entonces es el Job el que se cuida de su propia área común (con alguna ayuda de las utilidades).

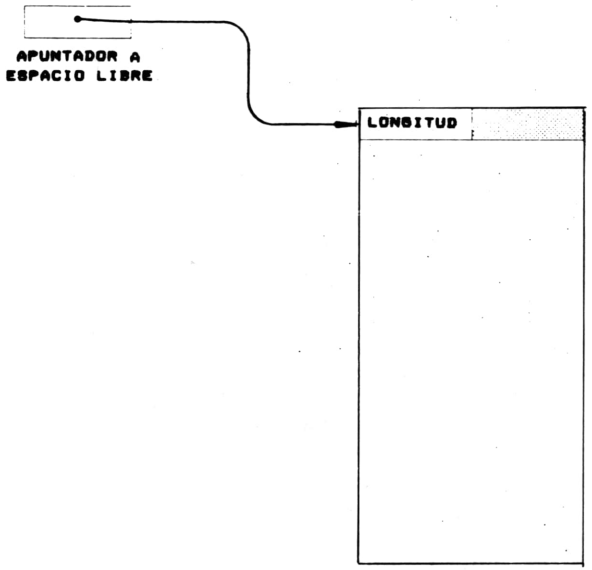


Figura 5.2a - Area de usuario nueva (vacía)

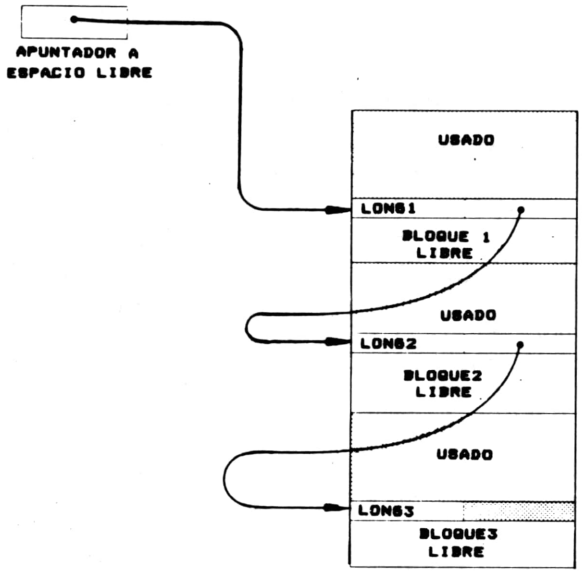


Figura 5.2b - Area de usuario (parcialmente usada)

La figura 5.2 muestra como se maneja el área de usuario, con la ayuda de MT.ALLOC (TRAP #1, D0=$0C) y MT.LNKFR (TRAP #1, D0=$0D). El área de usuario se prepara inicialmente conectando un área de RAM a un área inexistente (con el apuntador al espacio libre = 0). Esto se ve en la figura 5.2a.

El Job asigna espacio por sí mismo dentro del área de usuario; todos los espacios se encadenan juntos en un formato de lista encadenada. A medida que se va asignando y liberando, el área de usuario se va fragmentando. En un cierto momento, habrá muchos espacios libres dentro del área. Para encontrar el que se va a usar a continuación, las rutinas de utilidad recorren la lista para encontrar un espacio de tammaño adecuado, que es asignado. Esto se ve en la figura 5.2b. Tenga en cuenta que el área de usuario se puede extender dentro de otra área de memoria (área de trabajo del Job del área común) simplemente enganchandola al final de la lista de espacio libre.

5.3 Gestión del 'hardware'

Así como gestiona el 'hardware' estándar del QL, el QDOS ha sido diseñado para permitir una fácil expansión del sistema.

5.3.1 'Hardware' estándar del QL

Hay cuatro bloques de 'hardware' que tienen relacción con TRAPs del gestor. Estos son la pantalla, el IPC (incluye el teclado y el sonido), las puertas serie y el reloj. El control de pantalla selecciona alta resolución 512X512 modo de cuatro colores, o baja resolución 256X512 modo de ocho colores más modo parpadeo. El control del IPC es más complejo, ya que el 68008 tiene que comunicarse con el segundo procesador 8049 (el IPC). El IPC puede leer información del teclado y hacer sonar el altavoz. El IPC se encarga de todos los parámetros del sonido programable, de forma que el 68008 no tiene que perder tiempo controlándolo. Se puede cambiar la velocidad de trasmisión de las puertas serie mediante un TRAP del gestor. Se puede leer el reloj, ponerlo en hora o ajustarlo, usando los TRAPS $13, $14 y $15, del gestor.

5.3.2 Extensiones al QDOS

El QDOS se puede extender añadiéndole rutinas para servir interrupciones y controladores de dispositivos. Ests tipo de rutinas adicionales se suelen asociar (no necesariamente) con dispositivos 'hardware' que se han añadido al QL estándar. Todas las interrupciones se desarrollan mediante listas encadenadas. Una lista encadenada consiste en una serie de apuntadores. Entra la primera rutina de la lista. A su retorno, pasa el control a la siguiente de la lista, y así sucesivamente hasta que todas las rutinas se han ejecutado. Claramente, cuanto más cerca esté una rutina del principio de la lista, mayor prioridad tiene, ya que puede decidir si pasa o no a la siguiente rutina. Ya que las llamadas a rutinas del sistema están realaccionadas con las últimas, es posible reemplazar rutinas del sistema con las nuevas creadas para aplicaciones específicas, según se necesiten.

Las cinco listas encadenadas sirven para:

rutina de servicio de interrupciones externas
rutina de servicio de 'polling' 50/60 Hz
tarea de bucle del planificador
controloador de dispositivos E/S
controladore de dispositivos con directorio

5.4 TRAPs del gestor - Sección de referencia

En esta sección se proporciona toda la información sobre los TRAPs del gestor. Se accede a ellos mediante el TRAP #1, la función concreta se selecciona mediante el contenido del registro D0. La siguiente tabal es una lista de todas las funciones en orden numérico. Para obtener una descripción más completa de cada una, así como una explicación del código de llamada, se debe referir a las secciones posteriores del capítulo.

Sumario de los TRAPs del gestor

D0	Nombre	Descripción
00	MT.INF	Da información del Job actual y sistema
01	MT.CJO	Crea Job en área de programas transitorios
02	MT.JINF	Proporciona información de un Job
04	MT.RJOB	Borra Job de área de programas transitorios
05	MT.FRJOB	Fuerza borrado de Job del área de programas transitorios
06	MT.FREE	Busca el área libre mayor en el espacio de programas transitorios
07	MT.TRAPV	Pone un vector de TRAP para un Job
08	MT.SUSJB	Suspende un Job
09	MT.RELIB	Libera un Job
0A	MT.ACTIV	Activa un Job
0B	MT.PRIOR	Cambia la prioridad de un Job
0C	MT.ALLOC	Asigna memoria en el área común
0D	MT.LNKFR	Encadena un espacio libre en el área común
0E	MT.ALRES	Asigna área de procedimientos residentes
0F	MT.RERES	Libera área de procedimientos residentes
10	MT.DMODE	Pone y lee el modo de pantalla
11	MT.IPCOM	Manda un comando al IPC
12	MT.BAUD	Pone la velocidad de trasmisión
13	MT.RCLCK	Lee el reloj
14	MT.SCLCK	Pone el reloj
15	MT.ACLCK	Ajusta el reloj
16	MT.ALBAS	Asigna área de programa BASIC
17	MT.REBAS	Libera área de programa BASIC
18	MT.ALCHP	Asigna área común
19	MT.RECHP	Libera área común
1A	MT.LXINT	Encadena rutina de interrupción externa al QDOS
1B	MT.RXINT	Elimina una rutina de interrupción externa
1C	MT.LPOLL	Encadena una rutina de 'polling' 50/60 Hz al QDOS
1D	MT.RPOLL	Elimina rutina de 'polling' 50/60 Hz

1E	MT.LSCHD	Encadena tarea de bucle del planificador al QDOS
1F	MT.RSCHD	Elimina tarea de bucle del planificador
20	MT.LIOD	Encadena al QDOS un comtrolador de dispositivos de E/S
21	MT.RIOD	Elimina un controlador de dispositivo E/S
22	MT.LDD	Encadena un controlador de dispositivos con directorio, al QDOS
23	MT.RDD	Elimina un controlador de dispositivos con directorio

MT.INF TRAP#1 D0=0

Obtiene información del sistema

Parámetros de llamada	**Parámetros retornados**
D1	D1.L Job ID actual
D2	D2.L versión ASCII (n.nn)
D3	D3 preservado
A0	A0 apuntador a variables del sistema
A1	A1 preservado
A2	A2 preservado
A3	A3 preservado

Error devuelto:

ninguno

Descripción:

Este TRAP nos proporciona información sobre el estado actual del sistema, el Job ID del Job actual y su número de versión. También nos proporciona el apuntador a la base de las variables del sistema.

5.4.1 Creación y borrado de Jobs

Los Jobs se crean en el área de programas transitorios de la memoria del QL. Es esencial que la memoria asignada al Job sea suficiente para contener el código ejecutable, una pila y espacio de trabajo. Si se activa un Job, la ejecución comenzará en la dirección base, a menos que se le especifique una dirección distinta de comienzo. La dirección de principio se da como dirección *absoluta*. No es necesario que el Job tenga código ejecutable, se puede haber creado para contener un área de trabajo de otro Job. Sin embargo, si el Job no tiene código ejecutable, no debe ser *activado*. Si lo activaramos, haríamos fallar al sistema completo.

El área del Job se divide en dos partes. El código comienza en la base del Job, el resto del espacio sobre el código, hasta la alto del área del Job, es espacio de datos. Cuando se activa un Job por primera vez, (A6) apunta a la base de su área, (A6,A4) a la parte inferior del espacio de datos, y (A6,A5) a lo alto de su área. Al activar un Job de formato estándar, se le puede pasar información en la pila. En lo alto de la pila se coloca una palabra que contiene el número de canales abiertos para el Job. Esta va seguida del número de identificación de canal (palabras largas), para entrada básica, salida y reporte de canales, respectivamente, y finalmente una cadena de comandos del Job en formato estándar.

MT.CJOB TRAP#1 DO=1

Crea un Job en el área de programas transitorios

Parámetros de llamada		Parámetros retornados	
D1.L	Job ID propietario	D1.L	Job ID
D2.L	longitud del código	D2	preservado
D3.L	longitud espacio datos	D3	preservado
A0		A0	preservado
A1	dirección principio o 0	A1	preservado
A2		A2	preservado
A3		A3	preservado

Error devuelto:

OM no hay memoria
NJ no hay espacio en tabla de Jobs o D1 no es un Job

Descripción:

Este TRAP de creación de Job tiene dos efectos. Asigna espacio para el Job en el área de programas transitorios, y prepara una entrada de Job en las tablas del planificador. Sin embargo, esto no hace que el Job se incie totalmente. La única iniciación que ocurre es que pone dos palabras a ceros en la pila. El Job 'padre' carga normalmente el nuevo Job en el área de memoria asignada, después de esta llamada al sistema. El apuntador de la pila (en el área de control del Job) se pone inicialmente apuntando a las dos palabras a ceros de la pila. Estas se colocan en la dirección más alta del área de datos del Job.

Si el Job va a tener algún canal abierto, o se le va a pasar una cadena de comandos, se puede hacer antes de que sea activado. El Job ID propietario, pasado en D1, debe ser cero si el Job va a ser independiente. Si el Job actual va a ser el propietario del nuevo, D1 debe pasar un valor negativo. En el sistema operativo versión 1.03 y anteriores, A1 debe ser puesto siempre a cero a la entrada.

MT.JINF TRAP#1 D0=2

Obtiene información de un Job

Parámetros de llamada	Parámetros retornados
D1.L Job ID	D1.L siguiente Job en cadena
D2.L Job más alto	D2.L propietari del Job
D3	D3 Primer octeto negativo si está suspendido Segundo octeto, prioridad
A0	A0 dirección base del Job
A1	A1 indefinido
A2	A2 preservado
A3	A3 preservado

Error devuelto:

NJ no existe el Job

Descripción:

Este TRAP devuelve el estado actual del Job. Se puede usar para comprobar el estado de una cadena completa de Jobs. Si se va a usar de esta forma, D2 debe contener siempre el ID del Job en lo alto de la cadena. Para poder analizar la cadena completa, se prepara el TRAP con D1 conteniendo el valor devuelto por el TRAP anterior. Cuando se ha alcanzado al último Job de la cadena, D1 devuelve ceros.

MT.RJOB TRAP#1 DO=4

Borra un Job del área de programas transitorios

Parámetros de llamada	Parámetros retornados
D1.L Job ID	D1 indefinido
D2	D2 indefinido
D3.L Código de error	D3 indefinido
A0	A0 indefinido
A1	A1 indefinido
A2	A2 indefinido
A3	A3 indefinido

Error devuelto:

NJ no existe el Job
NC Job no está inactivo

Descripción:

Este TRAP borra un Job y todos sus subsidiarios del área de programas transitorios. Este TRAP borrará solamente Jobs inactivos. Los Jobs activos pueden ser borrados, forzándolos mediante MT.FRJOB D0=5.

Tenga en cuenta que este TRAP no está garantizado como atómico y no puede borrar el Job 0.

MT.FRJOB TRAP#1 D0=5

Fuerza borrado de un Job del área de programas transitorios

Parámetros de llamada	Parámetros retornados
D1.L Job ID	D1 indefinido
D2	D2 indefinido
D3.L Código error	D3 indefinido
A0	A0 indefinido
A1	A1 indefinido
A2	A2 indefinido
A3	A3 indefinido

Error devuelto:

NJ el Job no existe

Descripción:

Este TRAP inactiva una cadena completa de Jobs, y borra todos los Jobs de la cadena. Si D1 es una palabra negativa, se borra el Job actual. Si hay un Job esperando que se complete el Job que se va a borrar, ese Job se libera con D0 conteniendo un código de error (ver MT.ACTIV D0=A).

Tenga en cuenta que este TRAP no está garantizado como atómico y por lo tanto no puede borrar el Job 0.

MT.FREE TRAP#1 D0=6

Busca el mayor espacio libre en el área de transitoria

Parámetros de llamada	Parámetros retornados
D1	D1.L longitud del espacio
D2	D2 indefinido
D3	D3 indefinido
A0	A0 indefinido
A1	A1 indefinido
A2	A2 indefinido
A3	A3 indefinido

Error devuelto:

ninguno

Descripción:

Este TRAP busca el mayor espacio libre contiguo que puede ser asignado en el área de programas transitorios. La longitud del espacio es devuelta en el registro D1.

MT.TRAPV TRAP#1 D0=7

Pone un vector de TRAP para un Job

Parámetros de llamada		Parámetros retornados	
D1.L	Job ID	D1.L	Job ID
D2		D2	preservado
D3		D3	preservado
A0		A0	base del Job
A1	apuntador a la tabla	A1	preservado
A2		A2	preservado
A3		A3	preservado

Error devuelto:

ninguno

Descripción:

A un Job le es posible redireccionar TRAPs y vectores de excepción que no usa el QDOS. Esta redirección se define en una tabla (ver abajo). Si el Job prepara una tabla de vectores de TRAP para el mismo, se usará automáticamente siempre que se ejecute el Job. Un Job usará inicialmente la tabla del que lo ha creado, aunque no le pertenezca. Este estado de de cosas persiste hasta que otro Job prepara su propia tabla. Las tablas de vectores usadas por otros Jobs no son afectadas. Si el Job ID es una palabra negativa, la tabla la preparará el Job llamado.

La tabla consiste en una dirección de una palabra larga de longitud por cada TRAP o excepción, en el siguiente orden:

```
principio  $00  error de dirección
           $04  instrucción ilegal
           $08  división por cero
           $0C  CHK (comprobar registro con fronteras)
           $10  TRAPV (TRAP por rebasamiento)
           $14  violación de privilegio
           $18  trace
           $1C  nivel 7 de interrupción
           $20  trap #5
           $24  trap #6
           $28  trap #7
           $2C  trap #8
```

```
$30  trap #9
$34  trap #10
$38  trap #11
$3C  trap #12
$40  trap #13
$44  trap #14
$48  trap #15
$4C  fin de la tabla
```

5.4.2 Control de Job

Los Jobs están siempre en uno de tres estados definidos. Pueden estar *activos* (compartiendo los recursos de la CPU con otros Jobs), *suspendidos* (esperando por E/S o por otros Jobs) o *inactivos* (ocupando memoria pero sin poder hacer uso de los recursos de la CPU). La única diferencia práctica entre un Job inactivo y uno que ha sido suspendido indefinidamente es que el último no puede borrarse con MT.RJOB D0=4.

La información sobre un Job, su estado actual, prioridad, propietario, tamaño, etc. está contenida en el área de control del Job. Este área está en la región de memoria de programas transitorios y precede al área asignada para el Job. En QDOS v1.03, la base del área de control está $68 octetos antes que el Job. Esto es susceptible de cambio en versiones futuras. Para el QDOS V1.03, la organización de memoria es:

dir.	long.	nombre	descripción
$00	4	LEN*	longitud total control Job + área Job
$04	4	START	dirección de principio en activación
$08	4	OWNER?	Job ID del propietario
$0C	4	HOLD?	apuntador a un octeto que se limpiará cuando el planificador libere al Job (ver MT.SUSJB D0=8)
$10	2	TAG*	etiqueta del Job asignada por MT.CJOB
$12	1	PRIOR?	prioridad acumulada actual. Se incrementa cuando el Job está activo pero no ejecutándose. El planificador permite ejecutarse al Job con mayor prioridad acumulada.
$13	1	PRINC?	esta es la prioridad inicial del Job. El Basic activa Jobs con prioridad $20

$14	2	STAT*	Estado del Job: 0 no suspendido >0 número de cuadros antes de liberarlo -1 suspendido (ES o MT.SUSJB) -2 esperando que termine otro Job
$16	1	RELA6?	Bit superior puesto si el siguiente TRAP #2 o #3 tiene direccionamiento relativo (como lo puso el TRAP #4).
$17	1	WFLA6?	puesto si hay otro Job esperando por este
$18	4	WJOB?	ID del Job que espera por este.
$1C	4	TRAPV?	apuntador a los vectores de TRAP.
$20	32		valores de D0 a D7 salvados.
$40	32		valores de A0 a A7 salvados.
$60	2		valor del registro de estado salvado.
$62	4		valor del contador de programa salvado.

* significa valor que no debe ser cambiado
? significa valor que puede ser cambiado por un TRAP, o directamente (pero con cuidado!)
long longitud en octetos:
1 un octeto
2 una palabra
una palabra larga
3 8 palabras largas

MT.SUSJB TRAP#1 D0=8

Suspende un Job

Parámetros de llamada		Parámetros retornados	
D1.L	Job ID	D1.L	Job ID
D2		D2	preservado
D3.W	Periodo de espera	D3	preservado
A0		A0	base área control Job
A1	dirección indicadores	A1	preservado
A2		A2	preservado
A3		A3	preservado

Error devuelto:

NJ Job ID inválido

Descripción:

Este TRAP suspende un Job. El periodo de suspensión puede ser indefinido, o por un periodo de tiempo especificado. El periodo de espera se define como el número de cuadros, po lo que la suspensión puede ser hasta un periodo de 32K cuadros (unos 10 minutos). Si se le especifica el periodo de espera como -1, la suspensión será indefinida. No se puede usar otro valor negativo. Si el Job ID es una palabra negativa, se suspende el Job actual. El octeto de indicadores se limpiará cuando se libere el Job. Si no hay octeto de indicadores, A1 debe ponerse a cero. Si el Job ya estaba suspendido, Se quita la suspensión. Todos los Jobs son replanificados.

Tenga en cuenta que este TRAP no es totalmente atómico, ya que invoca al planificador!.

MT.RELJB TRAP#1 DO=9

Libera un Job

Parámetros de llamada	Parámetros retornados
D1.L Job ID	D1.L Job ID
D2	D2 preservado
D3	D3 preservado
A0	A0 base área control Job
A1	A1 preservado
A2	A2 preservado
A3	A3 preservado

Error devuelto:

NJ Job inválido

Descripción:

Este TRAP hace que se libere un Job que se encuentra suspendido. Todos los Jobs pueden ser replanificados después de esta llamada.

Tenga en cuenta que este TRAP no es totalmente atómico, ya que invoca al planificador!.

MT.ACTIV TRAP#1 D0=A

Activa un Job

Parámetros de llamada

D1.L Job ID
D2.B prioridad (0 a 127)
D3.W espera (-1 o 0)

A0
A1
A2
A3

Parámetros retornados

D1.L Job ID
D2 preservado
D3 preservado

A0 base área control Job
A1 preservado
A2 preservado
A3 preservado

Error devuelto:

NJ no existe el Job
NC Job activo

Descripción:

Este TRAP hace que un Job determinado en el área de programas transitorios, sea activado. La ejecución comenzará desde la dirección de comienzo que fué definida cunado se creó. La actividad de un Job se puede controlar también ajustando su nivel de prioridad (MT.PRIOR D0=B). Si la prioridad se pone a cero, el Job se pone inactivo.

Si el periodo de espera se pone a cero, la ejecución del Job actual continúa, si se pone otro valor, el Job será suspendido hasta que el Job activo terminé. Este TRAP volverá entonces con un código de error de ese Job.

Este TRAP no es totalmente atómico ya que invoca al planificador.

MT.PRIOR TRAP#1 D0=B

Cambia la prioridad de un Job

Parámetros de llamada	Parámetros retornados
D1.L Job ID	D1.L Job ID
D2.B prioridad (0 a 127)	D2 preservado
D3	D3 preservado
A0	A0 base área control Job
A1	A1 preservado
A2	A2 preservado
A3	A3 preservado

Error devuelto:

NJ no existe Job

Descripción:

Este TRAP cambia la prioridad de un Job. Si D1 es una palabra negativa, cambiará la prioridad del Job actual. Si se pone prioridad 0, el Job será inactivado. Este Job reentra el planificador, de modo que si el Job pone su prioridad a 0, se inactivará instantaneamente. Esta llamada no es totalmente atómica, ya que invoca al planificador.

5.4.3 Gestión del área de usuario (ver capítulo 8)

Bajo ciertas circunstancias, la gestión del área de usuario puede ser atómica. Hay dos TRAPs que lo hacen.

El área de usuario es asignada en múltiplos de 8 octetos. El espacio libre se encadena usando dos palabras largas por espacio. La primera palabra larga contiene la longitud del espacio y la segunda el apuntador relativo al siguiente espacio libre. Este área de usuario es totalmente reubicable, ya que los apuntadores son relativos. Se puede usar toda el área asignada por el código de usuario, asumiendo que en el programa esté previsto mantener control de la longitud de los items dentro del área. Cuando se asigna el área, la primera palabra larga contiene la longitud del área, de modo que pueda ser retenida por el código del usuario si lo desea.

El código del usuario debe mantener un apuntador al espacio libre. En la sección 5.2.5 se explica claramente este punto. El apuntador al espacio libre debe ser relativo (una palabra larga). Si el área no tiene espacio libre, porque se ha llenado o no existe, este apuntador será cero.

Estas áreas se preparan encadenando un área de RAM dentro de un área no existente (con el apuntador al espacio libre = 0). Se pueden expandir encadenandose a otra área de RAM. Para que su uso sea óptimo deberían ser contíguas.

MT.ALLOC TRAP#1 DO=C

Asigna espacio en el área común

Parámetros de llamada		**Parámetros retornados**	
D1.L	longitud requerida	D1.L	Longitud asignada
D2		D2	indefinido
D3		D3	indefinido
A0	apuntador al apuntador al espacio libre	A0	base del área asignada
A1		A1	indefinido
A2		A2	indefinido
A3		A3	indefinido
A6	direción base	A6	preservado

Error devuelto:

OM no hay suficiente espacio libre

Descripción:

Este TRAP asigna memoria en el área común para usarla con el código de usuario. El TRAP es totalmente atómico. A6 se usa como dirección base para este TRAP, de forma que A0 debe ser relativo a A6.

MT.LNKFR TRAP#1 D0=D

Encadena espacio libre (devuelto) en el área común

Parámetros de llamada		Parámetros retornados	
D1.L	longitud a encadenar	D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
A0	base del nuevo espacio	A0	indefinido
A1	apuntador al apuntador al espacio libre	A1	indefinido
A2		A2	indefiinio
A3		A3	indefinido
A6	dirección base	A6	preservado

Error devuelto:

ninguno

Descripción:

Este TRAP encadena un espacio libre de memoria (devuelto) en el área común. A6 se usa como direción base para esta llamada, así que A0 y A1 deben ser relativas a A6.

MT.ALRES TRAP#1 D0=E

Asigna área de procedimientos residentes

Parámetros de llamada	Parámetros retornados
D1.L # octetos requeridos	D1 indefinido
D2	D2 indefinido
D3	D3 indefinido
A0	A0 dirección base del área
A1	A1 indefinido
A2	A2 indefinido
A3	A3 indefinido

Error devuelto:

OM no hay memoria
NC imposible asignar (TRNSP área no está vacía)

Descripción:

El área de procedimientos residentes está situada en lo alto de la memoria, y se asigna normalmente a tiempo de encendido. La razón para ello es que todos los procedimientos transitorios se sitúan debajo de los procedimientos residentes en la memoria. Si se expande el espacio de procedimientos residentes cuando hay procedimientos transitorios, estos deberían ser reubicables, como normalmente no lo son, esto resulta imposible. Sin embargo, en el caso excepcional de que no hay procedimientos transitorios en la máquina, sería posible extenderla o reducirla. Los TRAPs que se encargan de ello so MT.ALRES D0=E y MT.RERES D0=F. La asignación se hace por un número de octetos. La función RESPR(n) del BASIC puede usarse para asignar octetos a la memoria de procedimientos residentes.

MT.RERES TRAP#1 DO=F

Libera área de procedimientos residentes

Parámetros de llamada	Parámetros retornados
D1	D1 indefinido
D2	D2 indefinido
D3	D3 indefinido
A0	A0 indefinido
A1	A1 indefinido
A2	A2 indefinido
A3	A3 indefinido

Error devuelto:

NC imposible liberar (TRNSP no está vacía)

Descripción:

Este TRAP libera memoria de procedimientos residentes para su uso por el sistema. Solamente se puede usar cuando el área de programas tansitorios está completamente vacía. Ver también MT.ALRES D0=E para la asignación de memoria de procedimientos residentes.

MT.DMODE TRAP#1 D0=10

Pone y lee el modo de pantalla

Parámetros de llamada		Parámetros retornados	
D1.B	-1 lee modo 0 modo 4 colores 8 modo 8 colores	D1.B	modo pantalla
D2.B	-1 leer pantalla 0 monitor 1 TV 625 líneas	D2.B	typo pantalla
D3		D3	preservado
A0		A0	preservado
A1		A1	preservado
A2		A2	preservado
A3		A3	preservado
A4		A4	indefinido

Error devuelto:

ninguno

Descripción:

Este TRAP pone o lee el modo actual de pantalla. Se suele seleccionar a tiempo de iniciación del sistema, para decirle al QL si está usando un televisor o un monitor. Este TRAP afecta a todas las ventanas, por lo que no se debe usar a menos que la máquina esté completamente vacia, ya que los programas pueden requerir una pantalla diferente a la que le estamos forzando a usar. Si se usa para leer el modo de pantalla, no hay ningún peligro.

5.4.4 Comunicación con el IPC

El IPC (controlador inteligente de periféricos) es un microprocesador de un solo chip 8049. Este *segundo procesador* se encarga de los datos de entrada del teclado y de los de salida al canal de sonido. El 68008 se comunica con el 8049 mediante medios octetos (nibble).

Los comandos se mandan al IPC mediante los 'nibbles' de comando. Estos van seguidos por los parámetros del comando, que son una cadena de 'nibbles' y/u octetos. Toda la información devuelta por el IPC (como la lectura de una fila de teclas) viene en formato de octeto.

El comando que se manda al IPC (usando MT.IPCOM) debe almacenarse en memoria en el siguiente formato. Consiste en una cabecera que describe el comando, seguida por el número de parámetros requeridos por el comando, terminando en un octeto que indica si se espera respuesta.

```
1 oct  el comando va en los cuatro bits inferiores
1 oct  contiene el número de octetos de parámetros
2 oct  número de bits a mandar de cada octeto de parámetro

       bits 1,0 = cantidad a mandar del primer octeto
       bits 3,2 = cantidad 2 mandar del segundo octeto
       etc.

n oct  octetos de los parámetros
1 oct  longitud de la respuesta codificada en bits 1,0
```

Se pueden transferir 0, 4 u 8 bits de cada octeto de parámetro. El número de bits que se van a transferir están codificados en una palabra larga. El código de dos bits es como sigue:

```
00  mandar los 4 bits inferiores
01  no mandar nada
10  mandar los 8 bits del octeto
11  no mandar nada
```

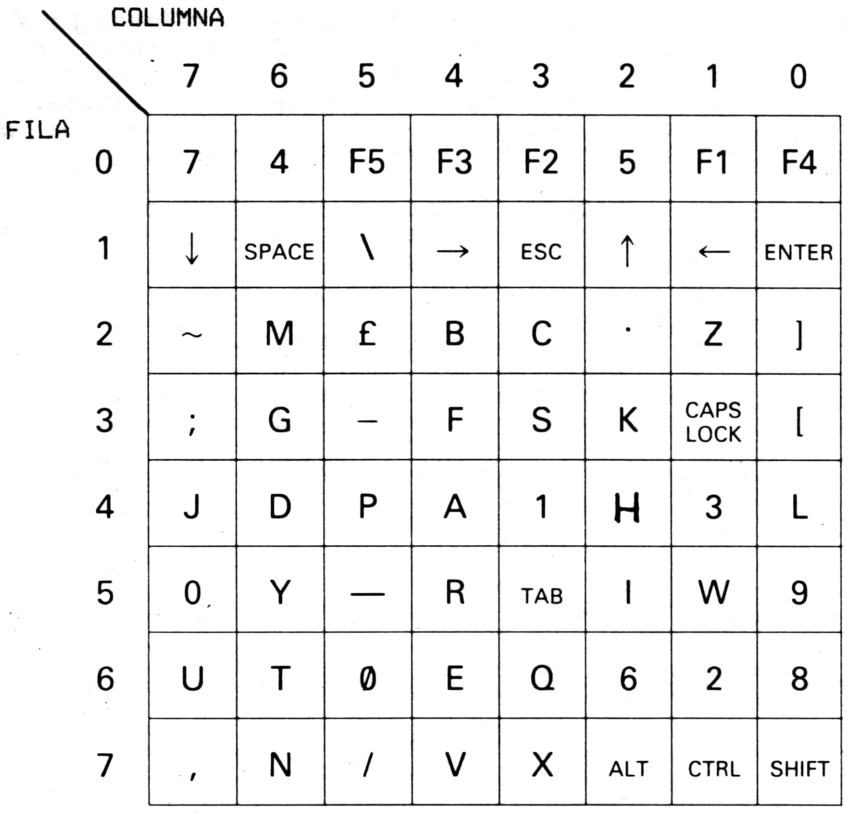
La comunicación con el IPC está **completamente desprotegida**, por lo que los comandos **no** deben contener errores. Si hay erores, el QL entero fallará. La comunicación con el IPC es un proceso muy lento, por lo tanto es esencial no hacer excesivo uso de él. Por ejemplo, es más eficiente leer solamente las teclas del cursor (están todas en una misma fila del teclado) que leer el teclado entero. Si se lee todo el teclado regularmente, resultará excesivamente caro en tiempo de proceso.

La mayoría de los comandos del IPC están diseñados para ser usados por el sistema operativo, y no se deben intentar usar desde los programas de usuario. Este tipo de intento puede resultar en pérdida de datos o fallos del sistema. Sin embargo, hay tres comandos que han sido diseñados para usarse en programas de aplicación. Estos son:

comando 9 lee el teclado (requiere un parámetro)
4 bits para el número de fila
8 bits para la respuesta
(ver fig. 5.3 con el esquema del teclado)

comando A inicia la generación de sonido (8 parámetros)
8 bits para tono 1 (1 mayor que en BASIC)
8 bits para tono 2 (1 mayor que en BASIC)
16 bits intérvalo entre pasos
16 bits duración sonido
4 bits paso en tonos
4 bits repetición
4 bits paso aleatorio
4 bits ruido
no necesita respuesta
(ver QL User Guide para la descripción de los parámetros de sonido)

comando B cortar sonido - sin parámetros - sin respuesta



FILA \ COLUMNA	7	6	5	4	3	2	1	0
0	7	4	F5	F3	F2	5	F1	F4
1	↓	SPACE	\	→	ESC	↑	←	ENTER
2	~	M	£	B	C	·	Z	]
3	;	G	–	F	S	K	CAPS LOCK	[
4	J	D	P	A	1	H	3	L
5	0	Y	—	R	TAB	I	W	9
6	U	T	Ø	E	Q	6	2	8
7	,	N	/	V	X	ALT	CTRL	SHIFT

Figura 5.3 - Esquema del teclado de la versión inglesa (para la versión española, ver el Apéndice X)

MT.IPCOM TRAP#1 DO=11

Manda comandos al IPC

Parámetros de llamada		Parámetros retornados	
D1		D1.B	parametro devuelto
D2		D2	preservado
D3		D3	indefinido
A0		A0	preservado
A1		A1	preservado
A2		A2	preservado
A3	apuntador al comando	A3	preservado

Error devuelto:

ninguno

Descripción:

Este TRAP manda un comando al IPC. El formato del comando debe ajustarse a lo descrito en las páginas anteriores, ya que no se realiza comprobación de los parámetros. Cualquier fallo en la definición de los comandos puede ser desastroso, desembocando en un fallo de todo el sistema.

MT.BAUD TRAP#1 D0=12

Pone la velocidad de transmisión

Parámetros de llamada	**Parámetros retornados**
D1.W velocidad trasmisión	D1 indefinido
D2	D2 preservado
D3	D3 preservado
A0	A0 preservado
A1	A1 preservado
A2	A2 preservado
A3	A3 preservado

Error devuelto:

ninguno

Descripción:

Las dos puertas serie del QL operan a la misma velocidad, tanto en trasmisión como en recepción. MT.BAUD se usa para selecionar la velocidad requerida. Las siguientes velocidades son válidas:

75	trasmisión y recepción
300	trasmisión y recepción
600	trasmisión y recepción
1200	trasmisión y recepción
2400	trasmisión y recepción
4800	trasmisión y recepción
9600	trasmisión; recepción (solo con más de un bit de parada)
19200	solo trasmisión

Tenga en cuenta que la paridad y el número de bits de parada se ponen independientemente de la velocidad. Para más detalles, vea el capítulo 6 (IO.OPEN y IO.CLOSE).

MT.RCLCK TRAP#1 D0=13

Lee el reloj

Parámetros de llamada	Parámetros retornados
D1	D1.L tiempo en segundos
D2	D2 indefinido
D3	D3 preservado
A0	A0 indefinido
A1	A1 preservado
A2	A2 preservado
A3	A3 preservado

Error devuelto:

ninguno

Descripción:

Con este TRAP se puede leer el reloj de tiempo real. El tiempo viene en una palabra larga conteniendo el número de segundos desde 00:00 1 Enero 1961. Esto significa que el QL puede dar la fecha hasta el año 2029!. Para convertir el tiempo en segundos a una cadena con la fecha, la hora y/o el dia de la semana, se pueden usar las utilidades por vector CN.DATE y CN.DAY.

MT.SCLCK TRAP#1 D0=14

Pone en hora el reloj

Parámetros de llamada	Parámetros retornados
D1.L tiempo en segundos	D1.L tiempo en segundos
D2	D2 indefinido
D3	D3 indefinido
A0	A0 indefinido
A1	A1 preservado
A2	A2 preservado
A3	A3 preservado

Error devuelto:

ninguno

Descripción:

Este TRAP paermite poner la hora y dia del reloj de tiempo real.

MT.ACLCK TRAP#1 D0=15

Ajusta el reloj

Parámetros de llamada	Parámetros retornados
D1.L ajuste en segundos	D1.L tiempo en segundos
D2	D2 indefinido
D3	D3 indefinido
A0	A0 indefinido
A1	A1 preservado
A2	A2 preservado
A3	A3 preservado

Error devuelto:

ninguno

Descripción:

Con este TRAP Se puede ajustar el reloj. Tenga en cuenta que el poner el reloj en hora lleva su tiempo, por lo tanto si llama a MT.ACLCK con D1=0, el QDOS no intentará ajustar el tiempo.

MT.ALBAS TRAP#1 DO=16

Asigna área para el programa BASIC

Parámetros de llamada		Parámetros retornados	
D1.L	# octetos requeridos	D1.L	# octetos asignados
D2		D2	indefinido
D3		D3	indefinido
A0		A0	indefinido
A1		A1	indefinido
A2		A2	indefinido
A3		A3	indefinido
A6	dirección base	A6	nueva dirección base
A7	apuntador pila usuario	A7	nuevo apuntador de pila

Error devuelto:

OM no hay memoria

Descripción:

MT.ALBAS asigna más memoria para que la use el intérprete de comandos BASIC. Esta memoria puede usarse para líneas de programa adicionales o para almacenamiento de datos. El intérprete de comandos es un Job muy especial, ya que es el único que se le permite expandir su memoria de esta forma. El intérprete de comandos se ejecuta en *modo usuario* como Job 0.

MT.REBAS TRAP#1 DO=17

Libera área del programa BASIC

Parámetros de llamada		**Parámetros retornados**	
D1.L	# octetos a liberar	D1.L	# octetos liberados
D2		D2	indefinido
D3		D3	indefinido
A0		A0	indefinido
A1		A1	indefinido
A2		A2	indefinido
A3		A3	indefinido
A6	dirección base	A6	nueva dirección base
A7	apuntador pila usuario	A7	nuevo apuntador pila

Error devuelto:

ninguno

Descripción:

MT.REBAS libera área de programa de usuario para que la usen otros Jobs. El espacio solo se libera cuando se ejecuta un comando **NEW** o **CLEAR.**

MT.ALCHP TRAP#1 D0=18

Asigna área común

Parámetros de llamada	Parámetros retornados
D1.L # octetos requeridos	D1.L # octetos asignados
D2.L Job ID propietario	D2 indefinido
D3	D3 indefinido
A0	A0 indefinido
A1	A1 indefinido
A2	A2 indefinido
A3	A3 indefinido

Error devuelto:

OM no hay memoria
NJ no existe el Job

Descripción:

MT.ALCHP permite a un Job asignar por sí mismo espacio en el área común. El espacio asignado no debe ser propiedad del Job creado. Cuando se borra el Job que posee el espacio, todo el espacio que le ha sido asignado es limpiado y puesto a disposición del sistema para que lo pueda asignar a otra aplicación.

MT.RECHP TRAP#1 D0=19

Libera área común

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
A0	base del área a liberar	A0	indefinido
A1		A1	indefinido
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

ninguno

Descripción:

MT.RECHP libera espacio del área común que se asignó con el TRAP MT.ALCHP.

5.4.5 Extendiendo el sistema operativo

Es posible extender el QDOS añadiendole rutinas para servicio de interrupciones o controlar dispositivos. Esto se realiza uniendo las nuevas rutinas en una lista encadenada que mantiene el QDOS. El diagrama y la explicación que le sigue nos muestra como funciona.

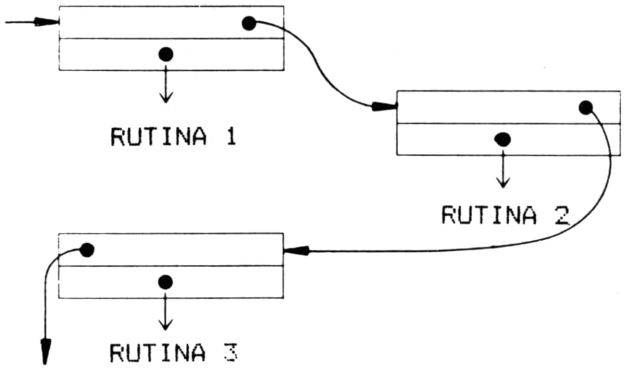


Figura 5.4a - Lista encadenada original

Un item en una lista encadenada consiste en un *apuntador de unión* que contiene la dirección del siguiente elemento en la lista. Con cada apuntador de unión hay asociada una (o varias) direcciones, que apuntan a las rutinas de servicio. La lista se examina por medio del apuntador de unión, que tiene asociada la dirección de la primera rutina de servicio que se ha de llamar (rutina 1 en el diagrama). Cuando devuelve control esta rutina, el QDOS usa el viejo apuntador de unión para encontrar el nuevo, desechándolo después. El nuevo apuntador tiene otra rutina asociada (rutina 2) que es llamada. A su retorno, se repite el proceso completo hasta que se llega al final de la lista.

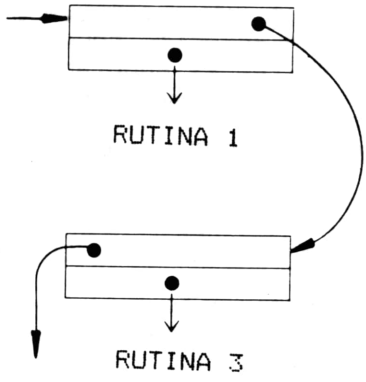


Figura 5.4b - Desconexión de un item de la lista

En la figura 5.4b se muestra como se borra un item de la lista. El apuntador de unión del que se va a borrar (el que estaba asociado a la rutina 2) se pone en el apuntador del anterior. Al hacer esto, forzamos a que la lista vaya de la rutina 1 a la 3, perdiendo la rutina 2 como si no hubiera existido nunca. A este proceso se le llama *desconectar* un item de la lista.

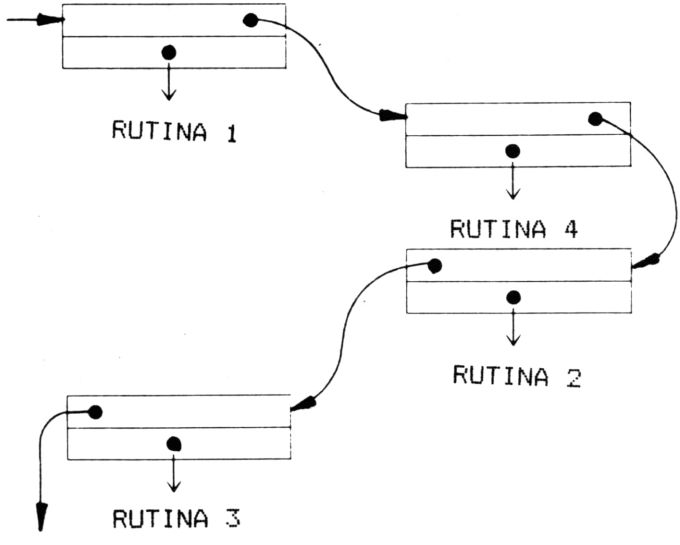


Figura 5.4c - Conectando un item a la lista

Cuando se va a conectar un nuevo item en la lista, el apuntador de unión del item anterior debe ponerse apuntando al nuevo. En nuestro ejemplo, el apuntador viejo del item 1 se ha puesto apuntando al nuevo (item 4), y en el nuevo se pne el apuntador que había antes en el anterior. Esto nos asegura que la lista permanece completa y se puede extender a cualquier tamaño (dentro de los límites de memoria de la máquina).

Hay cinco listas encadenadas en el QL:

1. servidores de interrupciones externas
2. servidores de interrupciones 50/60 Hz
3. tareas de bucle del planificador
4. controladores de dispositivos
5. controladores de directorio de dispositivos

Para cada una de estas, hay un TRAP que conecta una rutina a su lista, y otro que borra una rutina.

Para la lista encadenada de interrupciones, se deben asignar 8 octetos de RAM. Los primeros 4 octetos forma el apuntador de unión (puesto por MT.LXINT, MT.LPOLL y MT.LSCHD). Los otros 4 octetos forman una palabra larga que contiene la dirección de la rutina conectada.

Para los controladores de dispositivos, se han de asignar 16 octetos de RAM. La primera palabra larga es el apuntador de unión, la segunda, tercera y cuarta palabras largas apuntan respectivamente a las rutinas de entrada/salida, abrir y cerrar. Para los controladores de directorio se requieren 40 octetos de RAM como mínimo.

Antes de añadir un elemento a una lista encadenada,es esencial que la RAM en la que se va a conectar esté asignada al Job. La memoria asignada debe estar en el área de procedimientos residentes, si hace falta (código controlador de dispositivos en RAM), o alternativamente en el área común (código controlador de dispositivos en ROM). Si se usa el área común, el espacio debe ser propiedad del Job 0. Si es propiedad de otro Job y éste es forzado a borrarse antes de que se hayan desconectado las rutinas correspondientes de la lista, se puede garantizar que fallará el sistema operativo.

MT.LXINT TRAP#1 DO=1A

Conecta una rutina de servicio de interrupciones al QDOS

Parámetros de llamada		Parámetros retornados	
D1		D1	preservado
D2		D2	preservado
D3		D3	preservado
A0	dirección de la conexión	A0	preservado
A1		A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

ninguno

Descripción:

ver sección 5.4.5 para la descripción de lista encadenada

MT.RXINT TRAP#1 DO=1B

Borra una rutina de servicio de interrupciones al QDOS

Parámetros de llamada		Parámetros retornados	
D1		D1	preservado
D2		D2	preservado
D3		D3	preservado
A0	dirección de la conexión	A0	preservado
A1		A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

ninguno

Descripción:

ver sección 5.4.5 para la descripción de lista encadenada

MT.LPOLL TRAP#1 D0=1C

Conecta al QDOS una rutina de servicio de 'polling' 50/60 Hz

Parámetros de llamada		**Parámetros retornados**	
D1		D1	preservado
D2		D2	preservado
D3		D3	preservado
A0	dirección de la conexión	A0	preservado
A1		A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

ninguno

Descripción:

ver sección 5.4.5 para la descripción de lista encadenada

MT.RPOLL TRAP#1 D0=1D

Borra del QDOS una rutina de servicio de 'polling' 50/69 Hz

Parámetros de llamada		**Parámetros retornados**	
D1		D1	preservado
D2		D2	preservado
D3		D3	preservado
A0	dirección de la conexión	A0	preservado
A1		A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

ninguno

Descripción:

ver sección 5.4.5 para la descripción de lista encadenada

MT.LSCHD TRAP#1 DO=1E

Conecta al QDOS una tarea de bucle del planificador

Parámetros de llamada		Parámetros retornados	
D1		D1	preservado
D2		D2	preservado
D3		D3	preservado
A0	dirección de la conexión	A0	preservado
A1		A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

ninguno

Descripción:

ver sección 5.4.5 para la descripción de lista encadenada

MT.RSCHD TRAP#1 DO=1F

Borra del QDOS una tarea de bucle del planificador

Parámetros de llamada		Parámetros retornados	
D1		D1	preservado
D2		D2	preservado
D3		D3	preservado
A0	dirección de la conexión	A0	preservado
A1		A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

ninguno

Descripción:

ver sección 5.4.5 para la descripción de lista encadenada

MT.LIOD TRAP#1 DO=20

Conecta al QDOS un controlador de dispositivos ES

Parámetros de llamada		**Parámetros retornados**	
D1		D1	preservado
D2		D2	preservado
D3		D3	preservado
A0	dirección de la conexión	A0	preservado
A1		A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

ninguno

Descripción:

ver sección 5.4.5 para la descripción de lista encadenada

MT.RIOD TRAP#1 DO=21

Borra del QDOS un controlador de dispositivos ES

Parámetros de llamada		**Parámetros retornados**	
D1		D1	preservado
D2		D2	preservado
D3		D3	preservado
A0	dirección de la conexión	A0	preservado
A1		A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

ninguno

Descripción:

ver sección 5.4.5 para la descripción de lista encadenada

MT.LDD TRAP#1 DO=22

Conecta al QDOS un controlador de dispositivos de directorio

Parámetros de llamada		**Parámetros retornados**	
D1		D1	preservado
D2		D2	preservado
D3		D3	preservado
A0	dirección de la conexión	A0	preservado
A1		A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

ninguno

Descripción:

ver sección 5.4.5 para la descripción de lista encadenada

MT.LXINT TRAP#1 DO=1A

Borra del QDOS un controlador de dispositivos de directorio

Parámetros de llamada		**Parámetros retornados**	
D1		D1	preservado
D2		D2	preservado
D3		D3	preservado
A0	dirección de la conexión	A0	preservado
A1		A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

ninguno

Descripción:

ver sección 5.4.5 para la descripción de lista encadenada

6 Asignación de Entrada/Salida

6.1 Introducción a E/S

Este capítulo trata de la asignación de los recursos de entrada y salida en el QL. Estos recursos cubren todos los aspectos de comunicación del ordenador; teclado, puertas serie, pantalla, microdrives, etc. Sin embargo, no es necesario aprender diferentes grupos de reglas para cada dispositivo, ya que el QL ha sido diseñado de forma que la comunicación con todos estos dispositivos se realiza de una forma muy bien estructurada.

Las entradas y salidas en el QL se pueden considerar como escribir y leer datos desde un *fichero lógico*. Este *fichero lógico* está en formato de dispositivo estándar. Los datos que se mandan a la puerta serie para imprimir, o al microdrive para almacenar son tratados de forma similar. Los detalles específicos de interconexión para cualquier dispositivo en particular los realizan los *controladores de dispositivo*. Estos han sido escritos específicamente para cada uno de los dispositivos del QL. El sistema encuentra automáticamente el controlador correspondiente dentro del sistema. Todos ellos están en un formato especial (ver capítulo 9.2). En el capítulo 7 se explica con más detalle la entrada y salida.

La asignación de E/S se puede dividir en dos categorías generales. La de abrir un fichero para que se puedan recoger o mandar datos a él, y la de cerrar ficheros. Esta sección explica como abrir y cerrar ficheros. Es necesario conocer ahora como especificar un canal y toda la información sobre él, antes de poder abrirlo. La sección 6.2 explica posteriormente, la forma en que se deben especificar los dispositivos.

Tenemos disponibles los siguientes TRAPs de asignación de E/S:

D0	Nombre	Descripción
01	IO.OPEN	Abre un canal
02	IO.CLOSE	Cierra un canal
03	IO.FORMT	Formatea un medio en sectores
04	IO.DELET	Borra un fichero

6.2 Especificación de nombres de dispositivo

En el QL, toda la entrada y salida se manda o recoge desde un dispositivo o fichero lógico. Los dispositivos lógicos y físicos están unidos a través de un controlador de dispositivo. Para activar un dispositivo, se debe abrir el canal al que está unido. Para abrir correctamente un canal, suele ser necesaria alguna información básica extra para el comtrolador del dispositivo (ej. paridad en una puerta serie o tamaño de la ventana para una consola). Esta información extra se añade al nombre del dispositivo. A la combinación de un nombre de dispositivo y la información añadida se le llama nombre de fichero.

Los nombres de los dispositivos lógicos siguen las mismas convenciones que los nombre reales de fichero, por lo que hay ciertos nombre de fichero reservados que no se pueden usar (como mdv1 o ser1). En los nombres de fichero, no hay diferencia entre mayúsculas y minúsculas. Los siguientes dispositivos son los estándar del QL.

Dispositivos estándar del QL

CON_WxHaXxY_K esta es la consola con un área de ventana de 'W' por 'H' puntos con la esquina superior derecha en la posición 'X','Y'. 'K' es el número de caracteres en la memoria intermedia del teclado. El tamaño y la posición se definen en puntos en un mapa de pantalla de 512x256 (la posición 256x128 es el centro de la pantalla en ambos modos). El ancho y la posición X se especifican en múltiplos de 2 puntos. El valor por defecto es:

CON_448x180a32x16_128

SCR_WxHaXxY esta es la salida a pantalla. El tamaño y posición se definen en puntos en un mapa de pantalla de 512x256 puntos (la posición 256x128 es el centro de la pantalla en ambos modos). El ancho y la posición X se especifican en múltiplos de 16 puntos. El valor por defecto es:

SCR_448x180a32x16

SERnphz E/S serie, con puerta 'n'. 'p' indica la paridad: E,O,M o S para par, impar, marca o espacio respectivamente. 'h' es el parámetro

de intercambio de iniciación ('handshaking'),I para ignorarlo y H para usarlo. 'z' indica el protocolo: R,Z o C para datos sin procesar sin EOF, control Z como EOF o control Z como EOF con <CR> como separador de datos en vez de <LF>. El valor por defecto es:
SER1R (8 bits sin paridad)

NETI_nn unión a red de datos serie desde el nodo 'nn'

NETO_nn unión a red de datos serie hacia el nodo 'nn'

PIPE_n si se le pone 'n', es un canal de salida de n octetos de longitud. Si no se le pone, es un canal de entrada conectado al identificador de canal pasado en D3

MDVn_nombre acceso a ficheros en microdrive, 'n' es el número de microdrive, nombre es el *nombre* del fichero

6.3 Rellenando las especificaciones de nombres del sistema

Los nombres que se dan en esta sección **no** están incluidos en el QL actual. Es posible que estas convenciones de nombres se usen alguna vez en el futuro, y se han incluido aquí para completar.

6.3.1 Requerimientos para rellenar nombres del sistema

1. Los nombres de los ficheros deben ser compatibles con las convenciones de nombres del SuperBASIC del QL.

Esto nos permite una construcción como esta:

```
OPEN #1,mitexto
```

donde 'mitexto' es el nombre del fichero, así como

```
OPEN #canal,nombref$
```

donde el nombre del fichero está en la cadena 'nombref$'

2. Los nombres de los ficheros deben permitir el uso de estructuras de directorio, ya sean reales o simiuladas.

3. Los nombres de los ficheros deben permitir que la unidad o el medio sea

explícito (incluido en el nombre)
por defecto (valor por defecto del sistema o especificado anteriormente) o
indefinido (se busca en todas las unidades)

4. Los nombres de ficheros deben permitir la creación de nombres de ficheros relaccionados.

6.3.2 Aproximación al rellenado de nombres del sistema

1. Los nombres del sistema deben consistir en letras, dígitos y subrayados.

2. El nombre debe ser una serie o grupo de caracteres alfa-numéricos separados por subrayados. Cada grupo está relaccionado con un directorio.

3. Un Job puede proporcionar una cadena por defecto al sistema operativo. Esta cadena por defecto se añadirá al principio de cualquier nombre de fichero dado en una llamada para abrir un fichero. El sistema reconocerá ciertas cadenas (como MDV1) como nombre de dispositivos físicos. A estos nombres no se le añadirá la cadena por defecto.

La opción para buscar un fichero en todas las unidades (especificando el nombre del medio, por ejemplo) puede resultar muy caro en terminos de carga del procesador. Esto no se ha considerado en el 'software' inicial del QL.

4. La creación automática de nombres de fichero relaccionados con otro dado, se realiza normalmente añadiendo una extensión al final del nombre. Al QDOS le es imposible distinguir entre el nombre de un fichero de extensión (como JIM_BAS) y un nombre de fichero directorio (como ESTEFI_JIM), Por lo que las extensiones aparecen como ficheros dentro del directorio.

Tanto el nombre de fichero como el de extensión deben ser suministrados en la llamada OPEN. Así podrá tomarlo el sistema de la mejor manera posible.

6.3.3 Componentes de nombres de fichero

Los nombres de ficheros tienen el formato general de grupos de cadenas alfa-numéricas conectadas por signos de subrayado.

En las siguientes definiciones '**aaa**' es una cadena alfa-numérica que empieza con una letra.

nombre.unidad es	**aaan**	nombre definido por una unidad seguida por un dígito, ej. mdv1.
nombre.medio es	**aaa**	nombre con que se definió cuando se formateó el medio.
defecto es	**nombre.unidad{_aaa}**	
o	**nombre.medio{_aaa}**	
nombre.fichero es	**aaa{_aaa}**	
nombre.compl es	**unidad.nombre_fichero.nombre**	
extensión es	**aaa**	

Nota. En QDOS v1.03, nombre.coml es la única forma permitida de nombre de fichero. No se asumen defectos. No es posible distinguir entre **nombre.fichero** y **nombre.medio_nombre.fichero.**

6.4 Abriendo ficheros

6.4.1 Introducción general

Hay muchos métodos disponibles para abrir ficheros. Es posible abrir un fichero explicitamente mediante una operación OPEN. Alternativamente, se pueden abrir ficheros implicitamente con las operaciones COPY o DELETE.

Cada operación de OPEN se caracteriza por el estado del almacenamiento del fichero antes de abrirlo y por los derechos de acceso después de hacerlo. El estado antes de abrirlo es si el fichero existe o no. Después de abrirlo, los derechos de acceso al fichero pueden ser exclusivos o compartidos.

La siguiente tabla nos muestra los estados correspondientes a cada operación con el fichero:

Operación	**estado**		**acceso**		**tipo**
	nuevo	existe	exclusivo	compartido	
OPEN nuevo	X		X		
OPEN reescribir	X	X	X		
OPEN exclusivo		X	X		
OPEN compartido		X		X	
COPY origen		X		X	compartido
COPY destino	X		X		nuevo
DELETE		X	X		exclusivo

Los ficheros *compartidos* son a los que pueden acceder varios Jobs al mismo tiempo. Un ejemplo puede ser una base de datos maestra con varios Jobs refiriendose a ella. A los ficheros de acceso *exclusivo* solo puede acceder un Job cada vez.

6.4.2 Orden de búsqueda de ficheros

Esta subsección cubre el orden de búsqueda de ficheros. Deben tener en cuenta los lectores que esto no ha sido implementado en las versiones actuales del sistema operativo. Como existe una posibilidad de que se implemente en poco tiempo, se ha incluido para completar.

Hay tres partes posibles en un nombre de fichero que se va a abrir. Cada Job define un nombre por defecto. Como defecto adicional, también se puede proporcionar una extensión (por llamada OPEN). Finalmente, se proporciona un nombre completo o un nombre de fichero.

Cuando se abre un fichero nuevo, el nombre usado por el OPEN se define completo. En caso de que se abra un fichero

que ya existe, el sistema intenta encontrar el nombre de fichero que corresponde a una de las combinaciones del nombre que le hemos dado y los defectos.

El orden de búsqueda es:

nombre.completo dado	**nombre.fichero** dado

para nuevo o reescribir:

1er nombre.compl	1er defecto_nombre.fich

para exclusivo y compartido:

1o nombre.compl_extens	1o defecto_nombre.fich_extens
2o nombre.compl	2o defecto_nombre.fich
	3o nombre.fich_extens
	4o nombre.fich

Ejemplos de OPEN de ficheros existentes

Defecto	**fichero**	**extensión**	**Orden de búsqueda**
mdv2	jim	bas	mdv2_jim_bas mdv2_jim *y si jim es un medio* jim_bas jim
mdv2	fred_data	data	mdv2_fred_data_data mdv2_fred_data *y si fred es un medio* fred_data_data fred_data
mdv2	mdv1_temp	data	mdv1_temp_data mdv1_temp
mdv2	mdv1_temp	nulo	mdv1_temp
ser1_utils	backup	exec	ser1_utils_backup_exec ser1_utils_backup *si backup es un medio* backup_exec backup

```
ser1_utils  mdv2        nulo        mdv2(solo leer directorio)
```

Tenga en cuenta que en QDOS V1.03 solo está permitido usar nombre.completo.

6.5 Sección de referencia de TRAPS de asignación E/S

IO.OPEN TRAP#2 DO=1

Abre un canal

Parámetros de llamada	**Parámetros retornados**
D1.L Job ID	D1 Job ID
D2	D2 preservado
D3.L código en bits:	D3 preservado

0=viejo(exclusivo)fichero o dispositivo
1=viejo(compartido)fichero
2=nuevo(exclusivo)fichero
3=nuevo(reescribir)fichero(no soportado para microdrives en QDOS V1.03)
4=abrir directorio

A0	direcc. nombre canal	A0	ID del canal
A1		A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

NO no abierto porque hay demasiados canales abiertos
NJ el Job no existe
OM no hay memoria
NF no se encuentra fichero o dispositivo
EX el fichero ya existe
IU fichero o dispositivo en uso
BN nombre inválido de fichero o dispositivo

Cualquier código de retorno que no sea cero significa que el canal no ha sido abierto.

Descripción:

Este TRAP abre un canal. El nombre de fichero o dispositivo se usa para determinar el tipo de dispositivo requerido. Cada Job mantiene su propia lista de canales, por lo que se debe especificar el número de Job. Si el Job ID es una palabra negativa entonces el canal se asociará con el Job actual.

El nombre de fichero o dispositivo se debe almacenar en una cadena de caracteres ASCII. La cadena es precedida por una palabra que contiene el número de caracteres de la cadena, seguida por los los caracteres. El apuntador pasado en A0 debe apuntar a esta palabra (en frontera de palabra).

Un error **BN** significa que el nombre del dispositivo ha sido reconocido, pero los parámetros eran incorrectos. Por ejemplo CON_20y30.

En circunstancias normales, cuando se accede a un dispositivo no estándar, la información del código se ignora. En la práctica, esto significa que el código solo es relevante si se accede a un fichero almacenado.

IO.CLOSE TRAP#2 DO=2

Cierra un canal

Parámetros de llamada		Parámetros retornados	
D1		D1	preservado
D2		D2	preservado
D3		D3	preservado
A0	ID del canal	A0	indefinido
A1		A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

NO el canal no estaba abierto

Descripción:

Este TRAP cierra un canal que ha sido abierto previamente.

IO.FORMT TRAP#2 D0=3

Formatea un medio por sectores

Parámetros de llamada		Parámetros retornados	
D1		D1.W	número de sectores buenos
D2		D2.W	número total de sectores
D3		D3	preservado
A0	apuntador nombre medio	A0	indefinido
A1		A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

OM no hay memoria
FF ha fallado el formateo
NF no se encuentra
IU en uso

Descripción:

IO.FORMT hace que se formatee un medio por sectores, como los cartuchos de microdrive. El nombre del medio se almacena como una palabra con el número de caracteres seguida de una cadena de caracteres ASCII, el número de la unidad,un subrayado y hasta diez caracteres del nombre del medio. Ej. mdv2_Utilidad.

Este TRAP nos devuelve el *número total de sectores* y el *número de sectores buenos*. El total de secotres es el máximo número que puede caber en el medio. Ya que algunas partes del medio pueden estar en error, l número real de sectores útiles puede ser menor que el total.

IO.DELET TRAP#2 D0=4

Borra un fichero

Parámetros de llamada		Parámetros retornados	
D1.L	Job ID(el que lo abrió)	D1	indefinido
D2		D2	preservado
D3		D3	indefinido
A0	dirección nombre canal	A0	indefinido
A1		A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

NO no abierto (hay demasiados canales abiertos)
OM no hay memoria
NF fichero o dispositivo no encontrado
BN nombre de fichero o dispositivo inválido

Descripción:

Este TRAP hace que se borre el fichero especificado. A0 apunta a una palabra que contiene el número de caracteres ASCII que vienen a continuación. Estos caracteres ASCII definen el nombre.

7 Utilización de Entrada/Salida

7.1 Introducción

En el capítulo 6 hemos tratado los conceptos y procedimientos para asignar canales E/S. Para usar los canales asignados es encesario analizar los datos que se mandan y reciben desde el canal.

Los TRAP's quese encargan de la utilización de los canales se pueden dividir en tres grupos bien definidos. Estos grupos se resumen en las siguientes subsecciones.

7.1.1 Llamadas de E/S serie

Aquí se trata la E/S totalmente redireccionable del QL. Las operaciones de E/S serie incluyen el teclado, líneas serie RS232, pantalla, impresora, microdrives, etc. (sujetos como es lógico a las restricciones de lectura/escritura propias de cada dispositivo). Solo se necesita especificar el número de canal al que se van a mandar o del que se van a recibir los datos. Toda la información sobre las características particulares de cada fichero, se ha especificado ya en el proceso de OPEN (ver capítulo 6).

Cuando tratamos con el sistema de E/S totalmente redirecionable, hay tres posibles estados de retorno. El retorno se puede inhibir hasta que el proceso especificado se haya completado, forzar a que ocurra inmediatamente (ej sin esperar), o programarlo para que ocurra después de un periodo de tiempo predefinido (o si se completa antes el proceso). El periodo de espera se especifica como cero, un tiempo entre 0 y 10 minutos o indefinidamente. El incremento de tiempo es un número múltiplo de 50 o 60 Hz.

Sumario de llamadas de E/S serie

D0	Nombre	Descripción
00	IO.PEND	Comprueba una entrada pendiente
01	IO.FBYTE	Recoje un octeto
02	IO.FLINE	Recoje una línea caracteres terminada en <LF>
03	IO.FSTRG	Recoje una cadena de octetos
04	IO.EDLIN	Edita una línea de caracteres(solo en la consola)
05	IO.SBYTE	Manda un octeto
07	IO.SSTRG	Manda una cadena de octetos

7.1.2 Control de salida a pantalla

El control de la pantalla no es por sí mismo una parte del sistema redireccionable de E/S. Sin embargo, la pantalla es una parte integral del QL y como tal debe ser incluida en la utilización de E/S. Por medio del control de salida a pantalla se llevan a cabo procesos tales como modificar una ventana, controlar el cursor, hacer 'scroll' y poner los colores. La operación de sacar caracteres a la pantalla se puede realizar haciendo OPEN a CON_ o SCR_canal. Esto es parte del sistema redireccionable de E/S.

Sumario de las llamadas de salida a pantalla

D0	Nombre	Descripción
09	SD.EXPTOP	Invoca a una rutina adicional como parte del controlador de pantalla
0A	SD.PXENQ	Devuelve el tamaño de ventana y posición del cursor (en puntos)
0B	SD.CHENQ	Devuelve el tamaño de ventana y posición del cursor (en caracteres)
0C	SD.BORDR	Pone el borde de un color
0D	SD.WDEF	Redefine una ventana
0E	SD.CURE	Habilita el cursor
0F	SD.CURS	Suprime el cursor
10	SD.POS	Posiciona el cursor en la fila, columna en intervalos de carácter
11	SD.TAB	Posiciona el cursor en cualquier columna de caracteres definida
12	SD.NL	Posiciona el cursor en una nueva línea de caracteres
13	SD.PCOL	Posiciona el cursor en la columna anterior
14	SD.NCOL	Posiciona el cursor en la siguiente columna

15	SD.PROW	Posiciona el cursor en la línea anterior
16	SD.NROW	Posiciona el cursor en la siguiente línea
17	SD.PIXP	Posiciona el cursor en coordenadas de puntos
18	SD.SCROL	'Scroll' de toda una ventana
19	SD.SCRTP	'Scroll' de la parte superior de una ventana
1A	SD.SCRBT	'Scroll' de la parte inferior de una ventana
1B	SD.PAN	Mueve lateralmente una ventana
1E	SD.PANLN	Mueve lateralmente la línea del cursor
1F	SD.PANRT	Mueve lateralmente la parte derecha de la línea del cursor
20	SD.CLEAR	Limpia toda una ventana
21	SD.CLRTP	Limpia la parte alta de una ventana
22	SD.CLRBT	Limpia la parte baja de una ventana
23	SD.CLRLN	Limpia la línea del cursor
24	SD.CLRRT	Limpia la parte derecha de la línea del cursor
25	SD.FOUNT	Pone y quita un juego de caracteres
26	SD.RECOL	Recolorea una ventana
27	SD.SETPA	Pone el color del papel
28	SD.SETST	Pone el color del punteado
29	SD.SETIN	Pone el color de la tinta
2A	SD.SETFL	Pone el parpadeo
2B	SD.SETUL	Pone el subrayado
2C	SD.SETMD	Pone el modo caracteres o gráficos
2D	SD.SETSZ	Pone tamaño de los caracteres y espaciado
2E	SD.FILL	Rellena bloques rectangulares en las ventanas
30	SD.POINT	Dibuja un punto
31	SD.LINE	Dibuja una línea
32	SD.ARC	Dibuja un arco
33	SD.ELLIPS	Dibuja una elipse
34	SD.SCALE	Pone la escala de una ventana
35	SD.FLOOD	Pone o quita el relleno de un área gráfica
36	SD.GCUR	Pone la posición del cursor gráfico

7.1.3 Manejo de ficheros

El manejo de ficheros se puede realizar ampliamente con el sistema redireccionable de E/S. Por ejemplo, se pueden transferir ficheros enteros entre microdrives y las puertas RS232 sin problemas. En este capítulo se tratan dos áreas básicas de manejo de ficheros. Estas son la carga y savado de ficheros completos, y tratamiento de ficheros de acceso directo y de registros estructurados.

Sumario del manejo de ficheros

DO	Nombre	Descripción
40	FS.CHECK	Comprueba todas las operaciones pendientes de un fichero
41	FS.FLUSH	Borra la memoria intermedia de un fichero
42	FS.POSAB	Posiciona el apuntador de fichero a una posición absoluta en el fichero
43	FS.POSRE	Posiciona el apuntador de fichero a una posición relativa en el fichero
45	FS.MDINF	Obtine información sobre el medio
46	FS.HEADS	Pone una cabecera de fichero
47	FS.HEADR	Lee una cabecera de fichero
48	FS.LOAD	Carga un fichero en memoria
49	FS.SAVE	salva un fichero desde memoria

7.2 Llamadas de E/S serie

La E/S serie se refiere a la transferencia de octetos o caracteres. Esta transferencia es totalmente redireccionable, por lo tanto la salida se puede cambiar de la pantalla a la impresora (por ejemplo), redefiniendo simplemente el canal de transferencia. El proceso detallado de los datos a un periférico determinado es ejecutado por un *controlador de dispositivo*. Esta pieza de 'software' asegura que el QDOS pueda hablar con cualquier periférico de una forma estándar, a pesar de que los detalles de operación de estos periféricos pueden variar ampliamente.

La transferencia serie por octetos se suele usar cuando el Job que trata los datos debe controlar la acción del controlador. En estos casos, los octetos representan códigos de control que deben ser procesados por el Job y no por el controlador.

La alternativa a mandar octetos individuales es mandar cadenas completas. La trasferencia de estas cadenas es *mucho* más eficiente que transferir los ficheros de octeto en octeto. Cuando se lee una cadena con un tiempo de espera de cero, solamente se pueden leer los octetos que están disponibles en un tiempo determinado. La trasferencia de cadenas debe usar tanta memoria intermedia como sea posible. Si esta es muy pequeña, el proceso de trasferencia entre ficheros será ineficiente (a menos que el fichero contenga solamente un pequeño número de octetos).

Es importante usar la facilidad del tiempo de espera con cuidado,cuando está accediendo más de un Job al mismo canal.

Esto es debido a un posible problema de acceso de clases. Por ejemplo, si un Job está esperando por una ES y otro Job requiere otra ES en el mismo canal, entonces, suponiendo que el periodo de espera no sea cero, el segundo Job será replanificado y no se ejecutará hasta que el primer Job haya terminado su espera (poque haya terminado su tarea o vencido el tiempo de espera). El periodo de espera del segundo Job no empezará hasta que haga *realmente* el acceso al canal y no desde cuando lo intentó la primera vez.

7.2.1 Opciones de retorno de E/S serie

Las llamadas al sistema de E/S serie se pueden dividir en tres tipos de retorno diferentes, dependiendo de la aplicación. Las opciones son:

Esperando complexión

Esta opción de retorno no volverá hasta que se haya completado el proceso requerido. Este tipo de llamada se debe usar con mucho cuidado, ya que, si el evento del proceso permanece incompleto, el Job que lo llamó puede permanecer inactivo eternamente.

Retorno inmediato

Esta opción es la opuesta a la anterior. Cuando se realiza esta llamada, se realizarán todos los procesos que vayan a ser afectados inmediatamente. Todo lo que no se pueda realizar instantaneamente permanecerá sin terminar. Tenga en cuenta que si una llamada de salida retorna incompleta (ej, la memoria intermedia de canal está temporalmente llena), permanecerá incompleta. Los datos originales no habrán sido mandados, y deben ser reenviados posteriormente para que no se pierdan.

En espera del periodo o hasta la complexión

Esta opción retornará después de un tiempo predefinido o cuando se haya completado la tarea (si ocurre antes de que el periodo de espera se haya terminado). Un ejemplo del uso de este tipo de periodo puede ser un RS232 conectado a un teclado. Si no se pulsa una tecla en 10 minutos, por ejemplo, la llamada de lectura del teclado terminará su periodo de espera y puede desconectar el teclado del QL. Todos los tiempos se calculan en periodos de refresco de la pantalla, 50/60 Hz. El periodo de espera para las llamadas de E/S serie se pasa en D3 y no se modifica por los TRAPS.

General

Todas las opciones de retorno anteriores están disponibles a través de una llamada. Cada una de ellas se caracteriza por la longitud máxima de espera permitida. El periodo de espera puede ser cero, definido o ilimitado. Para definir el número de ciclos de cuadro antes del retorno, se usa una palabra, que puede tomar un valor máximo de 32767 (once minutos). Un periodo de espera de -1 indica que se debe esperar indefinidamente. No se debe usar ningún otro valor negativo.

7.2.2 Sección de referencia de E/S serie

IO.PEND TRAP#3 D0=0

Comprueba una entrada pendiente

Parámetros de llamada		**Parámetros retornados**	
D1		D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

NC no completado
NO el canal no está abierto
EF fin del fichero

Descripción:

Este TRAP comprueba una entrada pendiente en un canal. La información se devuelve al programa que lo lanzó, por medio del código de error. No lee ningún dato ni modifica el canal de entrada de ninguna forma. En las primeras versiones del QDOS, este TRAP no se puede realizar cuando se usa con el controlador de la consola. Se debe usar IO.FBYTE en su lugar.

IO.FBYTE TRAP#3 DO=1

Recoje un octeto

Parámetros de llamada	**Parámetros retornados**
D1	D1.B octeto recibido
D2	D2.L preservado
D3.W periodo de espera	D3.L preservado
A0 ID del canal	A0 preservado
A1	A1 indefinido
A2	A2 preservado
A3	A3 preservado

Error devuelto:

NC no completado
NO el canal no está abierto
EF fin del fichero

Descripción:

Este TRAP recoge un octeto del canal definido con un periodo de espera programado. El octeto recogido (si hay alguno) lo devuelve en el registro D1.

IO.FLINE TRAP#3 DO=2

Recoge una línea de caracteres terminados por <LF>

Parámetros de llamada		**Parámetros retornados**	
D1		D1.W	número octetos recogidos
D2.W	longitud memoria int.	D2.W	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1	base de memoria int.	A1	apuntador nuevo a mem. int
A2		A2	preservado
A3		A3	preservado

Error devuelto:

NC	no completado
NO	el canal no está abierto
EF	fin del fichero
BO	rebasamiento de la memoria intermedia

Descripción:

Con este TRAP se puede recoger una línea de caracteres. El número de caracteres en la línea (incluido el <LF>) se devuelve
en el registro D1. Para todas las lecturas múltiples de un octeto, el periodo de espera es el máximo periodo de espera desde el lanzamiento del TRAP, y no el tiempo entre octetos consecutivos. Por lo tanto no se puede usar el periodo de espera para detectar si el operador está tecleando muy lentamente. El máximo número de octetos que se pueden recoger dentro del límite de tiempo, son los que se obtienen.

El recoger una línea de caracteres de la consola tiene un significado especial. Todos los caracteres se van escribiendo en la ventana de correspondiente de la pantalla, a medida que se introducen desde el teclado. Cuando se usa este TRAP, el cursor se habilita en la ventana, y se inhabilita una vez que se ha leido la línea. Para modificar la línea se pueden usar las teclas de control del cursor, del modo normal:

←	mueve el cursor hacia la izquierda
→	mueve el cursor hacia la derecha
CTRL+←	borra un carácter a la izquierda
CTRL+ →	borra un carácter a la derecha

IO.FSTRG TRAP#3 D0=3

Recoge una cadena de octetos

Parámetros de llamada	Parámetros retornados
D1	D1.W número octetos recogidos
D2.W longitud memoria int.	D2.W preservado
D3.W periodo de espera	D3.L preservado
A0 ID del canal	A0 preservado
A1 base de memoria int.	A1 apuntador nuevo a mem. int
A2	A2 preservado
A3	A3 preservado

Error devuelto:

NC no completado
NO el canal no está abierto
EF fin del fichero

Descripción:

Este TRAP recoge una cadena de octetos del canal asignado. Si estos vienen del teclado, no serán impresos en la pantalla y no se podrá manejar el cursor desde la aplicación. Todo esto se puede conseguir mediante el TRAP IO.FLINE (TRAP #3, D0=2). Si se pasa en A1 un apuntador a una matríz de caracteres, a la vuelta apuntará al siguiente octeto.

IO.EDLIN TRAP#3 DO=4

Edita una línea de caracteres (solo en consola)

Parámetros de llamada		Parámetros retornados	
D1.L	longitud cursor/linea	D1.L	longitud cursor/linea
D2.W	longitud memoria int.	D2	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1	base final línea	A1	apuntador final de línea
A2		A2	preservado
A3		A3	preservado

Error devuelto:

NC no completado
NO el canal no está abierto
BO rebasamiento memoria intermedia

Descripción:

Este TRAP es similar a IO.FLINE, pero con ciertas diferencias. Se puede usar para editar una línea. A1 debe apuntar siempre al final de la línea. En D1 se pasa la posición del cursor en la palabra alta, y la longitud de la línea en la inferior. De este modo se puede editar la línea facilmente, se escribe (desde la posición actual del cursor) a la consola cada vez que se lanza el TRAP. Una vez lanzado, la línea no debe tener carácter de terminación. Sin embargo, a la vuelta, el caracter de terminación se incluirá en la cuenta de caracteres. Se pueden usar como caracteres válidos de terminación <CTRL J>, <ENTER>, cursor arriba o cursor abajo.

IO.SBYTE TRAP#3 DO=5

Manda un octeto

Parámetros de llamada	Parámetros retornados
D1.B octeto a mandar	D1 indefinido
D2	D2.L preservado
D3.W periodo de espera	D3.L preservado
A0 ID del canal	A0 preservado
A1	A1 indefinido
A2	A2 preservado
A3	A3 preservado

Error devuelto:

NC no completado
NO el canal no está abierto
DF unidad llena
OR fuera de rango

Descripción:

Con este trap se pueden mandar octetos separados a un canal. Para ciertos dispositivos, la espera en salida puede no operar de la forma esperada. El comportamiento esperado ocurre cuando el dispositivo de salida contiene algún tipo de memoria intermedia (ej. la memoria de una impresora). Todo esto es totalmente invisible para el sistema de E/S. Un retorno con la indicación de que se ha completado la transacción, solamente significa que los octetos se han transferido al controlador del dispositivo, no implica necesariamente que hayan llegado al otro extremo del dispositivo (ej. que hayan sido impresos). Ver también IO.SSTRG.

IO.SSTRG TRAP#3 D0=7

Manda una cadena de octetos

Parámetros de llamada		Parámetros retornados	
D1		D1	# octetos mandados
D2.W	# octetos a mandar	D2	preservado
D3.W	periodo de espera	D3	preservado
A0	ID del canal	A0	preservado
A1	base memoria interm.	A1	nuevo apuntador mem. int.
A2		A2	preservado
A3		A3	preservado

Error devuelto:

NC	no completado
NO	el canal no está abierto
DF	unidad llena

Descripción:

Con este TRAP se pueden mandar cadenas de octetos. Se han hecho algunas previsiones para generar nuevas líneas durante la salida a consola o pantalla. Cuando se manda un <LF>, ocurre la inserción automática de nuevas líneas, o si el cursor llega a la parte derecha de la pantalla. A diferencia del TRAP para mandar un octeto, este no genera el error OR. En caso de que el cursor se haya suprimido, la nueva línea no queda afectada inmediatamente, y permanece pendiente. Se puede activar de las siguientes formas:

mandando otro octeto o cadena,
cambiando el tamaño de carácter,
activando el cursor,
o pidiendo la posición del cursor.

Las nuevas líneas pendientes se cancelan si se posiciona el cursor. Ver también IO.SBYTE.

7.3 Control de salida a pantalla

El Control de la salida a pantalla cubre las operaciones de modificar ventanas, controlar el cursor, limpiar parte o toda la ventana, hacer 'scroll', y poner los colores. Ya que estas operaciones son específicas de la pantalla, estos TRAPs no forman parte del sistema redireccionable de E/S. Esto significa que puede ser posible escribir emuladores de controlador de pantalla que se pueden acceder con un TRAP#3.

7.3.1 Controladores de pantalla del usuario

SD.EXTOP TRAP#3 D0=9

Llama a una operación extendida

Parámetros de llamada		Parámetros retornados	
D1	pasa a rutina usuario	D1	devuelto por la rutina del usuario
D2	pasa a rutina usuario	D2	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1	pasa a rutina usuario	A1	devuelto por la rutina del usuario
A2	dir. comienzo rutina	A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Con este comando se puede llamar a una rutina proporcionada por el usuario, como si formara parte del controlador estándar de pantalla. El periodo de espera y el ID del canal se pasan en D3 y A0 como es usual. A2 contiene la dirección de comienzo de la rutina de usuario. Los demás registros D1, D2 y A1 pueden ser asignados por el programa de aplicación si se requiere.

7.3.2 Buscando el tamaño de la ventana actual

SD.PXENQ TRAP#3 DO=A

Obtiene tamaño ventana y posición del cursor (en puntos)

Parámetros de llamada		Parámetros retornados	
D1		D1	preservado
D2		D2	preservado
D3.W	periodo de espera	D3	preservado
A0	ID del canal	A0	preservado
A1	base del bloque pedido	A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Con este TRAP se puede averiguar el tamaño de la ventana asignada al canal definido. También se obtiene la posición actual del cursor dentro de esa ventana. Todas las posiciones se obtienen en coordenadas de puntos. El tamaño y la posición vienen en un bloque de petición de cuatro palabras. A1 apunta a la base de este bloque. Su formato es:

base	dimensión X de la ventana
base+2	dimensión Y de la ventana
base+4	posición X del cursor
base+6	posición Y del cursor

La posición superior izquierda es 0,0.

SD.PXENQ TRAP#3 DO=A

Obtiene tamaño ventana y posición del cursor (en caracteres)

Parámetros de llamada		**Parámetros retornados**	
D1		D1	preservado
D2		D2	preservado
D3.W	periodo de espera	D3	preservado
A0	ID del canal	A0	preservado
A1	base del bloque pedido	A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Con este TRAP se puede averiguar el tamaño de la ventana asignada al canal definido. También se obtiene la posición actual del cursor dentro de esa ventana. Todas las posiciones se obtienen en coordenadas de caracteres. El tamaño y la posición vienen en un bloque de petición de cuatro palabras. A1 apunta a la base de este bloque. Su formato es:

base	dimensión X de la ventana
base+2	dimensión Y de la ventana
base+4	posición X del cursor
base+6	posición Y del cursor

La posición superior izquierda es 0,0.

7.3.3 Control de ventanas

SD.BORDR TRAP#3 D0=C

pone el ancho y color del borde

Parámetros de llamada		Parámetros retornados	
D1.B	color	D1	indefinido
D2.W	ancho	D2	preservado
D3.W	periodo de espera	D3	preservado
A0	ID del canal	A0	preservado
A1		A1	preservado
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Esta llamada se usa para redefinir el borde de una ventana. El valor por defecto es 0. El ancho del borde es doblado en los bordes verticales y cae **dentro** de los límites originales de la ventana. Cuando se cambia el ancho del borde, el cursor se pone en posición de inicio(0,0).

Todos los TRAPs que se llamen a continuación para definir la posición del cursor y los límites de la pantalla, usan el tamaño nuevo de la ventana.

Las definiciones de color se tratan en la sección 7.3.9. En este TRAP disponemos de una definición nueva de color. Si definimos el color como $80 produciremos un borde *transparente*, lo que significa que no se altera el contenido original del borde.

SD.Wdef TRAP#3 DO=D

Redefine una ventana

Parámetros de llamada		Parámetros retornados	
D1.B	color del borde	D1	indefinido
D2.W	ancho del borde	D2	preservado
D3.W	periodo de espera	D3	preservado
A0	ID del canal	A0	preservado
A1	base bloque ventana	A1	indefinido
A2		A2	preservado

Error devuelto:

NC	no completado
NO	el canal no está abierto
OR	la ventana no cabe en la página

Descripción:

Esta llamada permite redefinir la posición y el diseño de una ventana. El contenido de la ventana anterior no se mueve ni se modifica, pero el cursor se reposiciona en la esquina superior izquierda de la nueva ventana. El bloque de ventana contiene 4 palabras como siguen:

base	ancho ventana
base+2	alto ventana
base+4	origen X
base+6	origen Y

7.3.4 Control del cursor

SD.CURE TRAP#3 D0=E

Habilita el cursor

Parámetros de llamada	Parámetros retornados
D1	D1 indefinido
D2	D2.L preservado
D3.W periodo de espera	D3.L preservado
A0 ID del canal	A0 preservado
A1	A1 indefinido
A2	A2 preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Este TRAP hace que se habilite el cursor en la ventana definida por el ID del canal. Siempre que se lance un TRAP para leer o editar una línea en una ventana, el cursor se habilitará automaticamente. No devolverá ningún error si el cursor ya está habilitado, en cuyo caso, este trap solo se asegura de que lo esté.

Todos los TRAPs que se lancen a partir de éste, para definir la posición del cursor y los límites de la ventana usarán el nuevo tamaño reducido de la ventana.

Las definiciones de color se tratan en la seción 7.3.9.

SD.CURS TRAP#3 D0=F

Suprime el cursor

Parámetros de llamada	Parámetros retornados
D1	D1 indefinido
D2	D2.L preservado
D3.W periodo de espera	D3.L preservado
A0 ID del canal	A0 preservado
A1	A1 preservado
A2	A2 preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Esta llamada hace que desaparezca el cursor en el canal especificado. Si la ventana no tuviera ya el cursor, esta llamada no devolverá error.

SD.POS TRAP#3 D0=10

Posiciona el cursor (coordenadas de caracteres)

Parámetros de llamada		**Parámetros retornados**	
D1.W	número de columna	D1	indefinido
D2.W	número de línea	D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto
OR la posición está fuera de la ventana

Descripción:

Este TRAP coloca el cursor en una posición absoluta de la pantalla, usando coordenadas de caracteres. La posición del cursor es la esquina superior izquierda del siguiente rectángulo de carácter. La posición de los rectángulos de carácter es relativa a la esquina superior izquierda de la ventana. Si el TRAP devuelve un error, el cursor no cambia de posición. Este TRAP cancela todas las nuevas líneas pendientes.

SD.TAB TRAP#3 DO=11

Posiciona el cursor en una columna de caracteres

Parámetros de llamada		Parámetros retornados	
D1.W	número de columna	D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto
OR la posición cae fuera de la ventana

Descripción:

Este TRAP hace que se tabule el cursor a la columna especificada en el registro D1. El cursor se sitúa en la esquina superior izquierda del siguiente rectángulo de carácter. La posición de los rectángulos de carácter es relativa a la esquina superior izquierda de la ventana. Si el TRAP devuelve un error, el cursor no cambia de posición. Este TRAP cancela todas las nuevas líneas pendientes.

SD.NL TRAP#3 D0=12

Posiciona el cursor en una nueva línea de caracteres

Parámetros de llamada	Parámetros retornados
D1	D1 indefinido
D2	D2.L preservado
D3.W periodo de espera	D3.L preservado
A0 ID del canal	A0 preservado
A1	A1 indefinido
A2	A2 preservado

Error devuelto:

NC no completado
NO el canal no está abierto
OR la posición está fuera de la ventana

Descripción:

Este TRAP posiciona el cursor al principio de la siguiente línea de caracteres. El cursor se posiciona en la parte superior izquierda del siguiente rectángulo de carácter. La posición de los rectángulos de carácter es relativa a la esquina superior izquierda de la ventana. Si el TRAP devuelve un error, el cursor no cambia de posición. Este TRAP cancela todas las nuevas líneas pendientes.

SD.PCOL TRAP#3 D0=13

Posiciona el cursor en el carácter anterior

Parámetros de llamada	Parámetros retornados
D1	D1 indefinido
D2	D2.L preservado
D3.W periodo de espera	D3.L preservado
A0 ID del canal	A0 preservado
A1	A1 indefinido
A2	A2 preservado

Error devuelto:

NC no completado
NO el canal no está abierto
OR la posición está fuera de la ventana

Descripción:

Este TRAP posiciona el cursor en el carácter anterior. El cursor se posiciona en la parte superior izquierda del siguiente rectángulo de carácter. La posición de los rectángulos de carácter es relativa a la esquina superior izquierda de la ventana. Si el TRAP devuelve un error, el cursor no cambia de posición. Este TRAP cancela todas las nuevas líneas pendientes.

SD.NCOL TRAP#3 DO=14

Posiciona el cursor en el siguiente carácter

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto
OR la posición está fuera de la ventana

Descripción:

Este TRAP posiciona el cursor en el siguiente carácter de la derecha. El cursor se posiciona en la parte superior izquierda del siguiente rectángulo de carácter. La posición de los rectángulos de carácter es relativa a la esquina superior izquierda de la ventana. Si el trap devuelve un error, el cursor no cambia de posición. Este TRAP cancela todas las nuevas líneas pendien- tes.

SD.PROW TRAP#3 DO=15

Posiciona el cursor en la línea anterior

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto
OR la posición está fuera de la ventana

Descripción:

Mueve el cursor a la línea anterior. El cursor se posiciona en la parte superior izquierda del siguiente rectángulo de carácter. La posición de los rectángulos de carácter es relativa a la esquina superior izquierda de la ventana. Si el trap devuelve un error, el cursor no cambia de posición. Este TRAP cancela todas las nuevas líneas pendientes.

SD.NROW TRAP#3 DO=16

Posiciona el cursor en la línea siguiente

Parámetros de llamada	Parámetros retornados
D1	D1 indefinido
D2	D2.L preservado
D3.W periodo de espera	D3.L preservado
A0 ID del canal	A0 preservado
A1	A1 indefinido
A2	A2 preservado

Error devuelto:

NC no completado
NO el canal no está abierto
OR la posición está fuera de la ventana

Descripción:

Este TRAP posiciona el cursor en la línea siguiente. El cursor se posiciona en la parte superior izquierda del siguiente rectángulo de carácter. La posición de los rectángulos de carácter es relativa a la esquina superior izquierda de la ventana. Si el TRAP devuelve un error, el cursor no cambia de posición. Este TRAP cancela todas las nuevas líneas pendientes.

SD.PIXP TRAP#3 DO=17

Posiciona el cursor usando coordenadas de puntos

Parámetros de llamada	**Parámetros retornados**
D1.W coordenada X	D1 indefinido
D2.W coordenada Y	D2.L preservado
D3.W periodo de espera	D3.L preservado
A0 ID del canal	A0 preservado
A1	A1 indefinido
A2	A2 preservado

Error devuelto:

NC no completado
NO el canal no está abierto
OR la posición está fuera de la ventana

Descripción:

Con este TRAP se puede posicionar el cursor dentro de la pantalla. Todas las coordenadas son relativas a la esquina superior izquierda de la pantalla, que es la posición 0,0. La nueva línea pendiente dentro de la pantalla, será cancelada. Si devuelve un error, el cursor no se mueve.

7.3.5 'Scroll' de las ventanas

SD.SCROL TRAP#3 DØ=18

Hace 'scroll de una ventana

Parámetros de llamada		Parámetros retornados	
D1.W	distancia del 'scroll'	D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Este TRAP hace que se mueva toda la ventana, hacia arriba o hacia abajo, el número de puntos que indica el registro D1. Las líneas que aparecen nuevas al hacer el 'scroll', se rellenan con el color del papel. Un valor positivo hace que se mueva la ventana hacia abajo. Una ventana de texto se suele mover hacia arriba, y requiere un valor negativo.

Ver también SD.SCRTP y SC.SCRBT para el 'scroll' parcial.

SD.SCRTP TRAP#3 DØ=19

Hace 'scroll de la parte superior de una ventana

Parámetros de llamada		Parámetros retornados	
D1.W	distancia del 'scroll'	D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
AØ	ID del canal	AØ	preservado
A1		A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Este TRAP hace 'scroll' del contenido de la parte superior de la ventana. La división entre la parte superior e inferior de la ventana viene dada por la línea del cursor, que no se mueve. Las líneas que aparecen nuevas al hacer el 'scroll', se rellenan con el color del papel. Un valor positivo hace que se mueva la ventana hacia abajo. Una ventana de texto se suele mover hacia arriba, y requiere un valor negativo.

SD.SCRBT TRAP#3 DO=1A

Hace 'scroll de la parte inferior de una ventana

Parámetros de llamada		Parámetros retornados	
D1.W	distancia del 'scroll'	D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Este TRAP hace 'scroll' del contenido de la parte inferior de la ventana. La división entre la parte superior e inferior de la ventana viene dada por la línea del cursor, que no se mueve. Las líneas que aparecen nuevas al hacer el 'scroll', se rellenan con el color del papel. Un valor positivo hace que se mueva la ventana hacia abajo. Una ventana de texto se suele mover hacia arriba, y requiere un valor negativo.

7.3.6 Movimiento lateral de ventanas

SD.PAN TRAP#3 DO=1B

Mueve una ventanas lateralmente

Parámetros de llamada		**Parámetros retornados**	
D1.W	distancia a mover	D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Este TRAP hace que toda la ventana se mueva lateralmehte el número de puntos deseado hacia la izquierda (argumento negativo) o hacia la derecha (argumento positivo). El espacio que se deja se rellena con el color del papel.

SD.PANLN TRAP#3 DO=1E

Mueve lateralmente la línea del cursor

Parámetros de llamada		Parámetros retornados	
D1.W	distancia a mover	D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Este TRAP hace que se mueva lateralmente la línea del cursor el número de puntos deseado hacia la izquierda (argumento negativo) o hacia la derecha (argumento positivo). El espacio que se deja se rellena con el color del papel. La línea del cursor puede tener 10 o 20 puntos de altura, dependiendo del tamaño de caracteres seleccionado.

SD.PANRT TRAP#3 D0=1F

Mueve lateralmente el final de la línea del cursor

Parámetros de llamada		Parámetros retornados	
D1.W	distancia a mover	D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Este TRAP hace que se mueva lateralmente la parte derecha, desde la línea del cursor el número de puntos deseado hacia la izquierda (argumento negativo) o hacia la derecha (argumento positivo). El espacio que se deja se rellena con el color del papel. La parte derecha incluye el carácter sobre el que está situado el cursor.

7.3.7 Limpiando las ventanas

SD.CLEAR TRAP#3 D0=20

Limpia toda una ventana

Parámetros de llamada	Parámetros retornados
D1	D1 indefinido
D2	D2.L preservado
D3.W periodo de espera	D3.L preservado
A0 ID del canal	A0 preservado
A1	A1 indefinido
A2	A2 preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Con este TRAP se puede limpiar una ventana completa. Todos los puntos de la ventana se ponen del color del papel. Se pueden limpiar otras partes de la ventana mediante los TRAPS SD.CLRTP, SD.CLRBT, SD.CLRLN y SD.CLRRT que limpian la parte superior, inferior, línea del cursor y final de la línea del cursor. La línea del cursor no se considera en la parte superior ni en la inferior de la ventana.

SD.CLRTP TRAP#3 DO=21

Limpia la parte superior de una ventana

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Con este TRAP se puede limpiar la parte superior de una ventana. Todos los puntos del área designada se ponen del color del papel. La línea del cursor hace de división entre la parte superior e inferior de la ventana (pero no está incluido en ninguna de ellas). También se pueden limpiar otras partes de a ventana.

SD.CLRBT TRAP#3 DO=22

Limpia la parte inferior de una ventana

Parámetros de llamada	**Parámetros retornados**
D1	D1 indefinido
D2	D2.L preservado
D3.W periodo de espera	D3.L preservado
A0 ID del canal	A0 preservado
A1	A1 indefinido
A2	A2 preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Con este TRAP se puede limpiar la parte inferior de una ventana. Todos los puntos del área designada se ponen del color dl papel. La línea del cursor hace de división entre la arte superior e inferior de la ventana (pero no está incluido en ninguna de ellas). Tambien se pueden limpiar otras partes de a ventana.

SD.CLRLN TRAP#3 DO=23

Limpia la línea del cursor

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Con este TRAP se puede limpiar la línea del cursor. Todos los puntos del área designada se ponen del color del papel. Se limpia la altura completa del tamaño de carácter (10 o 20 puntos).

SD.CLRRT TRAP#3 DO=24

Limpia la parte derecha de la línea del cursor

Parámetros de llamada		**Parámetros retornados**	
D1		D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Con este TRAP se puede limpiar la parte derecha de la línea del cursor. Todos los puntos de la parte derecha de la línea se ponen del color del papel. Se limpia la altura completa del tamaño del carácter (10 o 20 puntos).

7.3.8 Cambio del tipo de caracteres

SD.FOUNT TRAP#3 DO=25

Pone o quita el tipo de carácteres

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1	base de los caracteres	A1	indefinido
A2	base 2o tipo caracter.	A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto
DF dispositivo lleno

Descripción:

Los caracteres que van a salir a la pantalla se almacenan en una matríz de 5x9 puntos dentro de un rectángulo de 6x10. Esto asegura que los caracteres no se tocarán cuando se sitúen en celdas adyacentes. Hay un tipo por defecto y otro alternativo, dentro del QL, pero tambien se pueden seleccionar juegos de caracteres definidos por el usuario. El juego de caracteres por defecto incluye los cídigos ASCII desde $20 a $7F, y el segundo se extiende desde $80 a $BF.

Cuando se cambia el juego, se seleciona el defecto, si la dirección base que se le proporciona es 0. El juego debe almacenarse en memoria en un formato especial. Por debajo de cierto valor, los caracteres serán inválidos ($1F en el defecto). Desde el siguiente valor ($1F en el defecto) son válidos un número determinado de caracteres ($61 en el defecto). El formato es como sigue:

$00	carácter válido más pequeño (octeto)
$01	número de caracteres válidos-1(octeto)
$02 a $0A	9 octetos de puntos del 1er carácter válido
$0B a $13	9 octetos de puntos del 2o carácter válido
$14 a $1B	etc.

Los puntos válidos se almacenan en los bits 6 al 2 (inclusive) de cada octeto de definición de carácter. La fila superior de cada carácter se deja en blanco, y por lo tanto no se debe almacenar con las otras filas.

Si el carácter no es válido en el primer juego, se usa el segundo. Si tampoco es válido en el segundo, se usa el carácter más bajo del segundo juego.

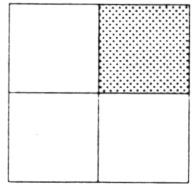
7.3.9 Poniendo los colores

En el QL disponemos de dos modos de pantalla. El modo 8 es el de más baja resolución, 512x256 puntos con 8 colores más parpadeo. El modo 4 es el de más alta resolución, 512x512 puntos con 4 colores sin parpadeo. Los códigos de los colores son:

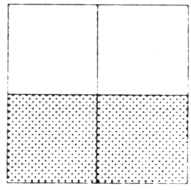
color	modo 256	modo 512
0	negro	negro
1	azul	negro
2	rojo	rojo
3	magenta	rojo
4	verde	verde
5	cyan	verde
6	amarillo	blanco
7	blanco	blanco

Además de los colores anteriores, se pueden crear áreas punteadas con dos colores, dentro de un cuadro de 2x2 puntos. Cuando se crea un área grande de este tipo, el punteado produce un efecto muy interesante.

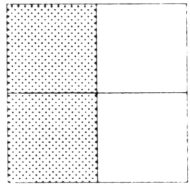
El color base se coloca siempre en los bits 2 a 0 del octeto de color. Los bits 5 a 3 contiene el XOR del color y la plantilla del punteado. Si el segundo color es cero, el color aparece sólido. Los bits 7 y 6 se usan para indicar la plantilla de punteado, como se ve a continuación:



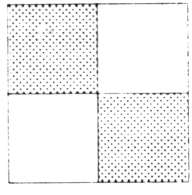
00sssbbb un punto de (sss XOR bbb) por cada cuatro puntos



01sssbbb barras horizontales de (sss XOR bbb)



10sssbbb barras verticales de (sss XOR bbb)



11sssbbb plantilla ajedrezada

SD.RECOL TRAP#3 DO=26

Cambia los colores de una ventana

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2.L	indefinido
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1	apuntador lista colores	A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Se pueden cambiar los colores de una ventana sin alterar la información que contiene. Este TRAP cambia cada uno de los ocho colores por otros nuevos. Algunos ordenadores cambian los atributos de la pantalla por 'hardware', lo que permite que cambien los colores rápidamente, sin embargo, en el QL es mucho más lento ya que tiene que cambiar el valor de cada posición de memoria.

La lista para el cambio de color es de ocho octetos de longitud. Contiene los colores requeridos para cada uno de los existentes en la ventana. Cada uno de los colores nuevos debe estar en el rango de 0 a 7. En modo de 4 colores solo se necesita rellenar los octetos 0, 2, 4 y 6.

SD.SETPA TRAP#3 DO=27

Pone el color del papel

Parámetros de llamada		Parámetros retornados	
D1.B	color	D1	preservado
D2		D2	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

El controlador de pantalla usa tres colores. Este TRAP se usa para poner el color del *papel*, que es el color de fondo de la ventana, y es con el que se rellena el espacio que aparece cuando se hace 'scroll', se mueve lateralmente o se limpia la ventana. Ver también SD.SETST y SD.SETIN para conocer la descripción de cada uno de los colores y su uso.

SD.SETST TRAP#3 DO=28

Pone el color del punteado

Parámetros de llamada		**Parámetros retornados**	
D1.B	color	D1	preservado
D2		D2	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

El controlador de pantalla usa tres colores. Este TRAP se usa para poner el color del *punteado*, que es el color que usa el generador de caracteres para resaltar el fondo de palabras o caracteres individuales. Por lo tanto, este color aparecerá siempre que se impriman caracteres en la pantalla. Ver también SD.SETPA y SD.SETIN para conocer la descripción de cada uno de los colores y su uso.

SD.SETIN TRAP#3 DO=29

Pone el color de la tinta

Parámetros de llamada		Parámetros retornados	
D1.B	color	D1	preservado
D2		D2	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

El controlador de pantalla usa tres colores. Este TRAP se usa para poner el color de la *tinta*, que es el color que se usa el para escribir caracteres y dibujar gráficos. Ver también SD.SETST y SD.SETPA para conocer la descripción de cada uno de los colores y su uso.

7.3.10 Cambio de los atributos de caracteres

SD.SETFL TRAP#3 D0=2A

Pone el modo parpadeo

Parámetros de llamada		**Parámetros retornados**	
D1.B	atributo de parpadeo	D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Este TRAP pone el modo parpadeo en los caracteres impresos. Si el atributo que se pasa en el registro D1 es 0, los caracteres se imprimen sin parpadeo, si no es 0, los caracteres que se impriman a continuación serán parpadeantes.

SD.SETUL TRAP#3 DO=2B

Pone el modo subrayado

Parámetros de llamada		Parámetros retornados	
D1.B	atributo de subrayado	D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Este TRAP pone el modo subrayado en los caracteres impresos. Si el atributo que se pasa en el registro D1 es 0, los caracteres se imprimen sin subrayado, si no es 0, los caracteres que se impriman a continuación serán subrayados.

SD.SETMD TRAP#3 DO=2C

Pone el modo en carácter o dibujo

Parámetros de llamada		**Parámetros retornados**	
D1.W	modo	D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Este TRAP permite definir la manera en que se producen los caracteres. Los siguientes modos son válidos:

D1=-1 se hace XOR de la tinta con el fondo

D1=0 el fondo del carácter está punteado, dibuja con el color de la tinta

D1=1 El fondo del carácter es transparente, por lo que no se puede escribir encima, se dibuja en el color de la tinta

SD.SETSZ TRAP#3 D0=2D

Pone el tamaño y espacio del carácter

Parámetros de llamada		Parámetros retornados	
D1.W	ancho carac./espaciado	D1	indefinido
D2.W	alto carac./espaciado	D2	preservado
D3.W	periodo de espera	D3	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Este TRAP se usa para seleccionar el alto, ancho y espacio de los caracteres dentro de su rectángulo. El generador de caracteres soporta directamente dos anchos y altos de estándar. El ancho simple es de 5 puntos en un espacio de 6 u 8 puntos. El doble ancho es de 10 puntos en un espacio de 12 o 16 puntos. El usuario puede producir otros espacios con controladores propios de pantalla, pero no son soportados directamente por este TRAP. El valor por defecto en modo simple es de 5 puntos en un cuadrado de 6. El valor por defecto en modo doble es de 10 puntos en un espacio de 12. En el modo de 8 colores, solamente se pueden usar caracteres de doble ancho. Las llamadas con D1=0 o 1 en modo de ocho colores aparecerán como si la llamada se hubiera hecho con D1 igual a 2 o 3, pero no se genera error.

Disponemos de los siguientes anchos/espacio de caracteres:

0 ancho simple (5 puntos) en espacio de 6
1 ancho simple (5 puntos) en espacio de 8
2 ancho doble (10 puntos) en espacio de 12
3 ancho doble (10 puntos) en espacio de 16

Los siguientes valores son los de altura/espaciado:

0 altura simple (9 puntos) en espacio de 10
1 altura doble (18 puntos) en espacio de 20

7.3.11 Relleno de bloques

SD.FILL TRAP#3 D0=2E

Rellena un bloque rectangular en una ventana

Parámetros de llamada		Parámetros retornados	
D1.B	color	D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1	base definición bloque	A1	indefinido
A2		A2	preservado

Error devuelto:

NC	no completado
NO	el canal no está abierto
OR	el bloque está fuera de la ventana

Descripción:

Este TRAP modifica un bloque rectangular dentro de una ventana. La definición del bloque consiste en 4 palabras en el siguiente formato:

base	ancho del bloque en puntos
base+2	alto del bloque en puntos
base+4	origen X del bloque (relativo al origen de ventana)
base+6	origen Y del bloque (relativo al origen de ventana)

El color elegido puede escribirse directamente a la pantalla (de forma que cubra todo lo que hay en ella), o hacer XOR con el contenido de la misma. La manera particular de rellenarla se controla mediante el modo impresión/dibujo que se haya definido con SD.SETMD (TRAP#3 con D0=$2C). Tambien se pueden usar colores con punteado. Se puede conseguir una gran rapidez en el cambio de colores, dentro de un área particular de la ventana, poniendo el modo impresión/dibujo en modo -1(XOR).

Truco útil - Resulta más rápido dibujar líneas *horizontales* y *verticales* con este TRAP que con el de dibujar una línea.

7.3.12 Traps de gráficos

SD.POINT TRAP#3 D0=30

Dibuja un punto

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1	apuntador pila aritmét.	A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Este TRAP dibuja una punto dentro de una ventana. Opera del mismo modo que el comando POINT de SuperBASIC. Si el punto cae fuera de la ventana, no se dibuja. Los parámetros que necesita se le deben dar en la pila aritmética en formato de coma flotante (apuntado por A1). Se deben asignar por lo menos 240 octetos en la pila, antes de llamarlo. Las coordenadas se refieren a un origen arbitrario (0,0 por defecto) con una escala arbitraria (la altura de la ventana por defecto es 100) Todos estos datos se pueden alterar mediante SD.SCALE.

Los parámetros requeridos son:

0(A1) coordenada Y
6(A1) coordenada X

Tenga en cuenta que 6(A1) significa 6 octetos después del punto que direcciona A1.

SD.LINE TRAP#3 D0=31

Dibuja una línea

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1	apuntador pila aritmét.	A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Este TRAP dibuja una línea dentro de una ventana. Opera del mismo modo que el comando LINE de SuperBASIC.Si alguna parte de la línea cae fuera de la ventana, esta no se dibuja. Los parámetros que necesita se le deben dar en la pila aritmética, en formato de coma flotante (apuntado por A1). Se deben asignar por lo menos 240 octetos en la pila antes de llamarlo. Las coordenadas se refieren a un origen arbitrario (0,0 por defecto) con una escala arbitraria (la altura de la ventana por defecto es 100) Todos estos datos se pueden alterar mediante SD.SCALE.

Los parámetros requeridos son:

0(A1)	coordenada Y del final de la línea
6(A1)	coordenada X del final de la línea
C(A1)	coordenada Y del principio de la línea
12(A1)	coordenada X del principio de la línea

Tenga en cuenta que 6(A1) significa 6 octetos después del punto que direcciona A1.

SD.ARC TRAP#3 D0=32

Dibuja un arco

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1	apuntador pila aritmét.	A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Este TRAP dibuja un arco dentro de una ventana. Opera del mismo modo que el comando ARC de SuperBASIC. Si alguna parte del arco cae fuera de la ventana, esta no se dibuja. Los parámetros que necesita se le deben dar en la pila aritmética, en formato de coma flotante (apuntado por A1). Se deben asignar por lo menos 240 octetos en la pila antes de llamarlo. Las coordenadas se refieren a un origen arbitrario (0,0 por defecto) con una escala arbitraria (la altura de la ventana por defecto es 100) Todos estos datos se pueden alterar mediante SD.SCALE.

Los parámetros requeridos son:

0(A1)	ángulo del arco
6(A1)	coordenada Y del final de la línea
C(A1)	coordenada X del final de la línea
12(A1)	coordenada Y del principio de la línea
18(A1)	coordenada X del principio de la línea

Tenga en cuenta que 6(A1) significa 6 octetos después del punto que direcciona A1.

SD.ELIPS TRAP#3 DO=33

Dibuja una elipse

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1	apuntador pila aritmét.	A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Este TRAP dibuja una elipse dentro de una ventana. Opera del mismo modo que el comando CIRCLE de SuperBASIC. Si alguna parte de la misma cae fuera de la ventana, esta no se dibuja. Los parámetros que necesita se le deben dar en la pila aritmética, en formato de coma flotante (apuntado por A1). Se deben asignar por lo menos 240 octetos en la pila antes de llamarlo. Las coordenadas se refieren a un origen arbitrario (0,0 por defecto) con una escala arbitraria (la altura de la ventana por defecto es 100) Todos estos datos se pueden alterar mediante SD.SCALE.

Los parámetros requeridos son:

0(A1)	ángulo de rotación
6(A1)	radio de la elipse
C(A1)	excentricidad de la elipse
12(A1)	coordenada Y del centro
18(A1)	coordenada X del centro

Tenga en cuenta que 6(A1) significa 6 octetos después del punto que direcciona A1.

SD.SCALE TRAP#3 DO=34

Pone la escala para dibujar en la pantalla

Parámetros de llamada		**Parámetros retornados**	
D1		D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1	apuntador pila aritmét.	A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Este TRAP es el mismo que el SCALE de SuperBASIC. Permite alterar el factor de escala que usan todos los TRAPs gráficos. Se cambian dos facetas - el origen de los gráficos y la longitud máxima del eje de las X. El TRAP se puede usar para mover la ventana que se está usando, alrededor de un espacio de gráficos mucho más amplio. Los parámetros que requiere este TRAP deben estar en ela pila aritmética, en formato de coma flotante (apuntados por A1). Se deben asignar 240 octetos de la pila como mínimo antes de llamar a este TRAP.

La altura por defecto es 100 y el origen 0,0.

Los parámetros requeridos son:

0(A1) coordenada Y de la línea inferior de la ventana
6(A1) coordenada X del punto más a la izquierda
C(A1) longitud del eje X (altura de la ventana)

Tenga en cuenta que 6(A1) significa 6 octetos después del punto que direcciona A1.

SD.FLOOD TRAP#3 DO=35

Pone o quita el formato de relleno

Parámetros de llamada		Parámetros retornados	
D1.L	octeto de estado	D1	indefinido
D2		D2	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1	apuntador pila aritmét.	A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Este TRAP pone o quita el relleno de gráficos de forma similar a como lo hace el comando FILL de SuperBASIC. Si se ha puesto el relleno, las formas no reentrables se rellenarán con el color de la tinta a medida que se dibujan. Una vez que la figura ha sido completamente cerrada, se debe volver a llamar a este TRAP para comenzar a rellenar la siguiente figura. El único parámetro especial que se pasa a la rutina es el octeto de estado. Se pone a 0 cuando se quita el relleno de áreas, y a 1 para que funcione.

SD.GCUR TRAP#3 DO=36

Define la posición del cursor gráfico

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2.L	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1	apuntador pila aritmét.	A1	indefinido
A2		A2	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Este TRAP es el mismo que el CURSOR de SuperBASIC, con cuatro parámetros. Permite posicionar el cursor gráfico en cualquier parte de la ventana del canal definido. Los parámetros que requiere este TRAP deben estar en la pila aritmética, en formato de coma flotante (apuntados por A1). Se deben asignar 240 octetos de la pila como mínimo antes de llamar a este TRAP. Las coordenadas se refieren a la esquina superior izquierda de la ventana. Los dos primeros parámetros fijan el origen del cursor gráfico, las otras dos colocan el cursor en una posición relativa al primer punto.

Los parámetros requeridos son:

0(A1)	coordenada gráfica X
6(A1)	coordenada gráfica Y
C(A1)	puntos hacia la derecha
12(A1)	puntos hacia abajo

Tenga en cuenta que 6(A1) significa 6 octetos después del punto que direcciona A1.

7.4 TRAPs del sistema de manejo de ficheros

La selección de TRAPs de manejo de ficheros que se incluyen en esta sección de la Utilización de E/S, cubren el manejo general de ficheros. La apertura y cierre de ficheros se trató en el capítulo 6, mientras que la lectura y escritura se hizo en la sección 7.1. Esta sección trata de las estructuras particulares de ficheros y de las utilidades.

Entre las utilidades se han incluido las operaciones necesarias para obtener información sobre el medio de almacenamiento, leer y escribir una cabecera de fichero, y cargar y salvar ficheros completos a y desde memoria.

El nivel de soporte para ficheros estructurados, lo proporciona el TRAP de posicionamiento del apuntador. Este apuntador se puede dirigir a cualquier octeto de un fichero, desde donde se puede leer una cadena de octetos de longitud arbitraria (ver 7.1). Con esta facilidad, se puede manejar cualquier estructura de fichero que se requiera. Las bases de datos requieren algún método de asegurarse de que se han escrito todos los datos en el fichero deseado, y de que ya no hacen falta en memoria. Se puede usar este TRAP para liberar la memoria intermedia del fichero y asegurarse de que todos los datos han llegado a su destino.

FS.CHECK TRAP#3 D0=40

Comprueba todas las operaciones pendientes en un fichero

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Este TRAP comprueba si se han completado todas las operaciones pendientes en un fichero. Si se han completado, la llamada retorna sin errores; si hay alguna operación pendiente, devuelve el código de error NC (no completado). Esta comprobación es necesaria, ya que cualquier intento de una operación de lectura o escritura a un fichero, o cambio de un apuntador de fichero, hace uso de la memoria intermedia. Esto significa que los bloques físicos requerdos del fichero en cuestión, son trasferidos dentro del área de bloques del QL. Este manejo de la memoria intermedia continúa aúnque las llamadas que causan la operación retornen completas.

FS.FLUSH TRAP#3 DO=41

Libera la memoria intermedia de un fichero

Parámetros de llamada		**Parámetros retornados**	
D1		D1	indefinido
D2		D2	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Esta llamada asegura que todos los datos que hay en el área de bloques de memoria, se escriben al fichero correspondiente. Es muy importante que se haga esta llamada en los casos en los que se debe comprobar que se han salvado todos los ficheros, antes de realizar otra operación. No es suficiente asumir que el TRAP se ha completado correctamente, ya que este tipo de retorno no implica que se hayan trasferido los datos al fichero correspondiente.

FS.POSAB TRAP#3 DO=42

Posiciona el apuntador absoluto de un fichero

Parámetros de llamada		Parámetros retornados	
D1.L	posición en el fichero	D1.L	nueva posición
D2		D2	preservado
D3.W	periodo de espera	D3	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

NC no completado
NO el canal no está abierto
EF fin del fichero

Descripción:

Este TRAP nos permite posicionar el apuntador del fichero en un punto determinado del mismo. Si el TRAP retorna con un error de fin de fichero, el apuntador se pone al límite más cercano (que es 0 o el final del fichero).

FS.POSRE TRAP#3 DO=43

Posiciona el apuntador del fichero, relativo a la posición anterior

Parámetros de llamada		Parámetros retornados	
D1.L	desp. apuntador fich.	D1.L	nueva posición
D2		D2	preservado
D3.W	periodo de espera	D3	preservado
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

NC no completado
NO el canal no está abierto
EF fin del fichero

Descripción:

Este TRAP nos permite posicionar el apuntador del fichero en un punto determinado, relativo a la posición anterior. Si el TRAP retorna con un error de fin de fichero, el apuntador se pone al límite más cercano (que es 0 o el final del fichero). Si se pone este desplazamiento a 0, el TRAP nos devolverá la posición actual en el registro D1. Aunque la llamada retorne como no completada, el apuntador se pondrá, y la tarea continuará cargando el bloque que contiene el siguiente octeto dentro de la memoria.

FS.MDINF TRAP#3 DO=45

Obtiene información sobre el medio

Parámetros de llamada		Parámetros retornados	
D1		D1.L	sectores vacios/buenos
D2		D2	preservado
D3.W	periodo de espera	D3	preservado
A0	ID del canal	A0	preservado
A1	apuntador a 10 octetos de memoria intermedia	A1	fin del nombre del medio
A2		A2	preservado
A3		A3	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Este TRAP nos proporciona el nombre del medio, el espacio disponible y la capacidad. Para hacer todo esto, se debe abrir el directorio del fichero. El nombre del medio es de 10 octetos de longitud, con espacios a la derecha. A1 debe apuntar a 10 octetos de memoria disponible, antes de hacer la llamada. La palabra más significativa de D1 contiene el número de sectores vacios, y la menos significativa el número de sectores buenos que contiene el emdio. Los sectores son de 512 octetos de longitud.

FS.HEADS TRAP#3 DO=46

Pone cabecera de fichero

Parámetros de llamada		**Parámetros retornados**	
D1		D1.W	longitud cabecera
D2		D2	preservado
D3.W	periodo de espera	D3	preservado
A0	ID del canal	A0	preservado
A1	base bloque cabecera	A1	fin definición cabecera
A2		A2	preservado
A3		A3	preservado

Error devuelto:

NC no completado
NO el canal no está abierto

Descripción:

Todos los ficheros tienen una cabecera que contiene información sobre el fichero. Este TRAP prepara los 14 primeros octetos de esta cabecera. El formato estándar para los ficheros con directorio es de 64 octetos de longitud y es el siguiente:

$00 longitud del fichero (palabra larga)
$04 aceso al fichero (octeto) - es 0 en QDOS V1.03
$05 tipo de fichero (octeto). Para ficheros ordinarios y programas BASIC, el tipo de fichero es cero. Para programas ejecutables es 1.
$06 8 octetos de información dependiente del tipo. Para programas ejecutables (tipo 1), los 4 primeros octetos contienen el tamaño asignado por defecto al espacio de datos del programa.
$0E longitud del nombre del fichero
$10 nombre del fichero (hasta 36 caracteres)
$34 reservado para información de fecha (no implementado en las primeras versiones del QL)

Cuando el TRAP trata con dispositivos serie puros, la longitud de la cabecera en D1 será espúreo (=15). Esto se debe a que, cuando se mande la cabecera a un dispositivo serie, los 14 octetos de la cabecera van precedidos por un octeto a $FF

FS.HEADR TRAP#3 DO=47

Lee la cabecera de un fichero

Parámetros de llamada		Parámetros retornados	
D1		D1.W	longitud cabecera leida
D2.W	long. memoria interm.	D2	preservado
D3.W	periodo de espera	D3.	preservado
A0	ID del canal	A0	preservado
A1	base de la memoria intermedia de lectura	A1	alto de la memoria intermedia de lectura
A2		A2	preservado
A3		A3	preservado

Error devuelto:

NC no completado
NO el canal no está abierto
BO rebasamiento memoria intermedia

Descripción:

La llamada de lectura de cabecera de fichero se usa para que un Job pueda conocer las características más relevantes de un fichero, incluida su longitud. Esta información nos permite asignar espacio suficiente (usando uno de los TRAPs #1) para cargar el fichero.

A1 apunta a la memoria intermedia de lectura que debe tener 14 octetos como mínimo. En el caso de que se use para un dispositivo serie, la longitud de la cabecera retornada en D1 es epúrea (=15). A la vuelta del TRAP, la memoria intermedia contendrá la información de la cabecera del fichero.

El apuntador del fichero está diseñado de forma que apunta a la posición cero que es el primer octeto de datos del fichero. Las fronteras de los bloques en un controlador de directorio estándar están en las posiciones 512*n-64.

FS.LOAD TRAP#3 DO=48

Carga un fichero en memoria

Parámetros de llamada		**Parámetros retornados**	
D1		D1	indefinido
D2.L	longitud del fichero	D2	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1	direc. base de carga	A1	direc. alta después carga
A2		A2	preservado
A3		A3	preservado

Error devuelto:

NO el canal no está abierto

Descripción:

Este TRAP permite cargar todo un fichero en memoria. Antes de intentarlo se debe reservar suficiente espacio para él. Si este espacio va a estar en el área de programas transitorios, debe ser asignado mediante un TRAP#1 antes de intentar cargarlo. El periodo de espera debe ser infinito (D0=-1). Tambien es importante asegurarse de que A1 contiene una dirección par, ya que los programas solo se pueden cargar en frontera de palabra.

FS.SAVE TRAP#3 D0=49

Salva un fichero desde memoria

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2.L	longitud del fichero	D2	preservado
D3.W	periodo de espera	D3.L	preservado
A0	ID del canal	A0	preservado
A1	direc. base fichero	A1	direc. alta dichero
A2		A2	preservado
A3		A3	preservado

Error devuelto:

NO el canal no está abierto
DF unidad llena

Descripción:

Este TRAP permite salvar un fichero desde memoria. D3 debe ser puesto a -1 antes de llamarlo, y A1 debe ser par para que se sitúe en frontera de palabra cuando se cargue.

8 Utilidades por Vector

8.1 Introducción

En esta sección tenemos una gran variedad de rutinas de la ROM que pueden ser usadas en aplicaciones muy diferentes. Las Utilidades por vector se llaman así porque se acceden tomando sus direcciones de una tabla de vectores (apuntadores al principio de cada rutina). Esto significa que lasrutinas funcionarán en futuras versiones del QDOS a pesar de que se puedan haber cambiado sus direcciones.

Acceso a las rutinas por medio de la tabla de vectores

La tabla de vectores se extiende desde la dirección $C0 de la ROM hasta la $12B. Las entradas son de una *palabra* y apuntan a la dirección real de inicio de las rutinas correspondientes. Ya que las entradas están formadas por una palabra en vez de dobles palabras, todas las rutinas deben estar en las primeras 32K de memoria. Estas rutinas no comprueban el espacio en la pila, por lo que debe haber por lo menos 64 octetos disponibles para las rutinas ordinarias y 96 para las aritméticas. Para acceder a las rutinas se deben usar las siguientes líneas de código máquina:

```
MOVE.W  aa.aaa,An   aa.aaaa es el vector
JSR     (An)        salta a la subrutina en aa.aaaa
```

Si la utilidad pone un código de estado en D0, el registro de estado del 68008 se pondrá de acuerdo con él a su regraso de la rutina.

Estas rutinas se pueden dividir en cinco grandes grupos, definidos por los requerimientos del código que las llama.

Rutinas en Modo Supervisor

En ciertas situaciones, el código que no está dentro del QDOS llama a las rutinas en modo supervisor. Esto puede ocurrir, por ejemplo, en parte de los controladores de dispositivo usados por los TRAPs #2 o #3. Hay varias reglas a las que se debe atener el código que las llama:

1. El código que las llama debe estar en modo supervisor antes de hacer la llamada

2. El código que las llama no debe haber sido iniciado por una interrupción

3. A6 debe apuntar a la base de las variables del sistema

Hay disponibles las siguientes rutinas en Modo Supervisor:

Vec.	Nombre	Tipo	Descripción
C0	MM.ALCHP	SM	Asigna área común
C2	MM.RECHP	SM	Libera área común
E8	IO.SERQ	SM	Dirige la manejo de las colas
EA	IO.SERIO	SM	Manejo general de ES

Rutinas de TRAP simplificadas

Hay un grupo de TRAPs que se usan una y otra vez. Para reducir el código requerido para estos TRAPs, donde los parámetros se ajustan a un formato común, se pueden usar las rutinas por vector simplificadas. Estas rutinas aparecen listadas debajo. Todas se deben llamar en modo usuario.

Vec.	Nombre	Tipo	Descripción
C4	UT.WINDW	ST	Prepara una ventana con el nombre dado
C6	UT.CON	ST	Prepara una ventana de consola
C8	UT.SCR	ST	Prepara una ventana de pantalla
CA	UT.ERRO	ST	Escribe un mensaje de error en el canal 0
CC	UT.ERR	ST	Escribe un mensaje de error en otro canal
CE	UT.MINT	ST	Convierte un entero en ASCII
D0	UT.MTEXT	ST	Manda un mensaje a un canal

Rutinas de utilidad general

Estas rutinas se pueden usar desde cualquier código. No hay requisitos especiales.

Vec.	Nombre	Tipo	Descripción
D2	UT.LINK	GU	Conecta un item a una lista
D4	UT.UNLNK	GU	Desconecta un item de una lista
D8	MM.ALLOC	GU	Asigna un espacio en el área común
DA	MM.LNKFR	GU	Conecta espacio libre en el área común
DC	IO.QSET	GU	Prepara una cola
DE	IO.QTEST	GU	Mira el estado de una cola

E0	IO.QIN	GU	Pone un octeto en una cola
E2	IO.QOUT	GU	Extra un octeto de una cola
E4	IO.QEOF	GU	Pone la marca de EOF en una cola
122	IO.NAME	GU	Decodifica un nombre de dispositivo

Rutinas de utilidad del BASIC

Las rutinas de utilidad del BASIC están compuestas por rutinas de soporte de procedimientos BASIC y el paquete aritmético. Estas rutinas pueden ser usadas por cualquier código, pero debe asegurarse de que las direcciones pasadas a las rutinas son relativas a A6.

Vec.	Nombre	Tipo	Descripción
E6	UT.CSTR	BU	Compara dos cadenas
EC	CN.DATE	BU	Toma fecha y hora
EE	CN.DAY	BU	Toma dia de la semana
F0	CN.FTOD	BU	Convierte coma flotante a ASCII
F2	CN.ITOD	BU	Convierte un entero a ASCII
F4	CN.ITOBB	BU	Convierte un octeto a ASCII binario
F6	CN.ITOBW	BU	Convierte una palabra a ASCII binario
F8	CN.ITOBL	BU	Convierte una palabra larga a ASCII binario
FA	CN.ITOHB	BU	Convierte un octeto hex a ASCII
FC	CN.ITOHW	BU	Convierte una palabra hex a ASCII
FE	CN.ITOHL	BU	Convierte una palabra larga hex a ASCII
100	CN.DTOF	BU	Convierte ASCII en coma flotante
102	CN.DTOI	BU	Convierte ASCII en un entero
104	CN.BTOIB	BU	Convierte ASCII en un octeto binario
106	CN.BTOIW	BU	Convierte ASCII en una palabra binaria
108	CN.BTOIL	BU	Convierte ASCII a una palabra larga binaria
10A	CN.HTOIB	BU	Convierte ASCII en un octeto hex
10C	CN.HTOIW	BU	Convierte ASCII en una palabra hex
10E	CN.HTOIL	BU	Convierte ASCII en una palabra larga hex
110	BP.INIT	BP	Procedimiento básico de iniciación
112	CA.GTINT	BP	Obtiene enteros (palabra)
114	CA.GTFP	BP	Obtiene coma flotante (6 octetos)
116	CA.GTSTR	BP	Obtiene cadenas
118	CA.GTLIN	BP	Obtiene enteros largos (palabra larga)
11A	BV.CHRIX	BP	Reserva espacio en la pila aritmética
11C	RI.EXEC	AR	Ejecuta una operación
11E	RI.EXECB	AR	Ejecuta una lista de operaciones
120	BP.LET	BP	Retorna valores de parámetros

Rutinas de soporte de microdrives

Las rutinas de soporte de microdrives se han puesto en vectores para simplificar la escritura a las utilidades que necesitan acceder a determinados sectores (ej. programas de recuperación de ficheros).

Vec.	Nombre	Tipo	Descripción
124	MD.READ	MD	Lee un sector
126	MD.WRITE	MD	Escribe un sector
128	MD.VERIN	MD	Verifica un sector
12A	MD.SECTR	MD	Lee una cabecera de sector

8.2 Rutinas en Modo Supervisor

Las rutinas en modo supervisor se acceden normalmente mediante el TRAP#1. Sin embargo, bajo ciertas circunstancias resulta útil poder llamar a estas rutinas desde código que no está residente en el QDOS. Por ejemplo, controladores de dispositivos que se acceden mediante un TRAP#2, pueden necesitar usar rutinas en modo supervisor. Las que describimos aquí están relaccionadas con el manejo del área común y con ES simple.

El código que las llama debe cumplir las siguientes especificaciones:

1. El código que las llama debe estar en modo supervisor antes de hacer la llamada

2. El código que las llama no debe haber sido iniciado por una interrupción

3. A6 debe apuntar a la base de las variables del sistema

MM.ALCHP VECTOR CO

Asigna espacio en el área común

Parámetros de llamada	Parámetros retornados
D1.L espacio requerido	D1.L espacio asignado
D2	D2 indefinido
D3	D3 indefinido
A0	A0 base del área asignada
A1	A1 indefinido
A2	A2 indefinido
A3	A3 indefinido

Error devuelto:

OM no hay memoria

Descripción:

Esta rutina permite asignar área común. El espacio debe ser suficiente pará contener una entrada de cabecera (ver Apéndice R). Este espacio extra es de 16 octetos para entradas simples, o de 24 octetos para los canales de IOSS. En A0 le vuelve la dirección base del área común y no la base del área que puede usar (ver TRAP#1 con D0=18 o 19). El espacio asignado en el área común se rellena con ceros. La descripción del área común se da en las secciones 5.2.4 y 5.2.5.

MM.RECHP VECTOR C2

Libera espacio en el área común

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
A0	Base área a liberar	A0	indefinido
A1		A1	indefinido
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

ninguno

Descripción:

Esta rutina libera espacio que ha sido asignado previamente en el área común.

IO.SERQ VECTOR E8

Maneja directamente la cola de ES serie

Parámetros de llamada		**Parámetros retornados**	
D0	número de rutina	D0	error devuelto
D1	valor IOSS estándar	D1	valor IOSS estándar
D2	valor IOSS estándar	D2	valor IOSS estándar
D.W	tiempo de espera	D3	preservado
A0	ID del canal	A0	preservado
A1	valor IOSS estándar	A1	valor IOSS estándar
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

BP acción indefinida

Descripción:

Esta es una rutina de manejo directo de colas. Para usarla, es necesario asegurarse que las palabras largas 7 y 8 del bloque de definición de canal (ver página siguiente) apuntan a las colas de entrada y salida respectivamente. Se puede inhibir la entrada, poniendo la palabra larga 7 a ceros, y la salida poniendo la palabra larga 8 a ceros.

El tipo particular de operación que se realiza se define con el contenido del registro D0, como en el TRAP#3. Los valores permitidos para D0 son $00, $01, $02, $03, $05, $07, $46, $47 y $49. Los valores estándar del IOSS (subsistema de entrada/salida) son los que se pasarían a las rutinas asociadas del TRAP#3.

Para acciones indefinidas, IO.SERQ devolverá el ERR.BP (si se pide un TRAP#3 inválido).

Bloque de definición de canal

Cada vez que se abre un canal nuevo, se construye un bloque de definición de canal, donde se mantiene toda la información concerniente al mismo, como el controlador de disposique se ha de usar, el Job propietario y el número y estado del canal.

```
CH.LEN    $00 4  longitud del bloque
CH.DRIVR  $04 4  dirección del controlador
CH.OWNER  $08 4  Job propietario
CH.RFLAG  $0C 4  dirección que se manda cuando se libera
                 espacio
CH.TAG    $10 2  ID del canal
CH.STAT   $12 1  estado:

                 0= OK
                 -1= A1 absoluto
                 $80= A1 relativo a A6
                 negativo= esperando

CH.ACTN   $13 1  acción almacenada para el Job en espera
CH.JOBWT  $14 4  ID del Job esperando por E/S
*
* definición extendida de canal para manejo de colas serie
*
CH.QIN    $18 4  apuntador a cola de entrada (o cero)
CH.Qout   $1C 4  apuntador a cola de salida (o cero)
```

IO.SERIO VECTOR EA

Manejo general de ES serie

Parámetros de llamada		**Parámetros retornados**	
D0	número de rutina	D0	error devuelto
D1	valor IOSS estándar	D1	valor IOSS estándar
D2	valor IOSS estándar	D2	valor IOSS estándar
D.W	tiempo de espera	D3	indefinido
A0	ID del canal	A0	preservado
A1	valor IOSS estándar	A1	valor IOSS estándar
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

BP acción indefinida

Descripción:

Esta rutina se usa cuando la rutina IO.SERQ no es suficiente.

Como IO.SERQ, el tipo particular de la operación a realizar se define con el registro D0, como en el TRAP#3. Los valores permitidos para D0 son $00, $01, $02, $03, $05, $07, $46, $47, $48 y $49. Cualquier otro valor hará que devuelva D0=ERR.BP. Los valores estándar de IOSS (sussistema de entrada/salida) se refieren a los valores que se deben pasar a las rutinas del TRAP#3.

IO.SERIO requiere que se le proporcione la dirección de tres rutinas para comprobar, la entrada pendiente, ,recoger octetos y mandar octetos. Las direcciones de entrada de estas rutinas se deben colocar en tres palabras largas a continuación de la instrucción CALL, en el siguiente orden:

palabra1	comprueba entrada pendiente	(siguiente octeto devuelto en D1)
palabra2	recojer un octeto	(octeto en D1)
palabra3	mandar un octeto	(octeto en D1)

Puede ser dificil de implementar si se ponen direcciones absolutas, por lo que es mejor poner las direcciones de las rutinas en el bloque de definición física para el controlador (apuntado por el registro A3) cuando se inicializa el controlador de dispositivo. Hay varia maneras de hacerlo. Aquí le doy un par de sugerencias:

Ejemplo 1

código en $28(A3) o posición similar

```
Hex       Nemotécnicos de Ensamblador

387800E8  MOVE.W  $E8,A4
4E94      JSR     (A4)
          DC.L    TEST   dirección rutina de comprobar
          DC.L    FETCH  dirección rutina de recoger
          DC.L    SEND   dirección rutina mandar
4E75      RTS
```

Este código se llama con:

```
          JSR     $28(A3)
```

Ejemplo 2

```
Hex       Nemotécnicos de Ensamblador

          DC.L    TEST   dirección rutina de comprobar
          DC.L    FETCH  dirección rutina de recoger
          DC.L    SEND   dirección rutina mandar
4E75      RTS
```

Este código se llama con:

```
          PEA     $28(A3)
          MOVE.W  $E8,A4
          JMP     (A4)
```

Las tres rutinas de servicio deben devolver los códigos de error en D0. Los otros registros D1 a D3 y A1 a A3 pueden ser cambiados.

8.3 Rutinas de TRAP simplificadas

Estas rutinas se usan para operaciones muy comunes. Todas se deben llamar en modo usuario.

UT.WINDW VECTOR C4

Prepara una ventana con el nombre dado

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
A0	apuntador al nombre	A0	ID del canal
A1	apunt. bloque parámet.	A1	indefinido
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

BN	nombre de dispositivo inválido
OM	no hay memoria
NO	no hay canales
OR	ventana fuera de la pantalla

Descripción:

Esta rutina abre una ventana usando el nombre suministrado. La sentencia CALL va seguida por un bloque de parámetros de 4 octetos que define el color del borde, el punteado, el papel y la tinta. También se define el ancho del borde.

El formato del bloque de parámetros es:

00	octeto	color del borde
01	octeto	ancho del borde
02	octeto	color papel/punteado
03	octeto	color de la tinta

UT.CON VECTOR C6

Prepara una ventana de consola

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
A0		A0	ID del canal
A1	apunt. bloque parámet.	A1	indefinido
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

OM no hay memoria
NO no hay canales
OR ventana fuera de la pantalla

Descripción:

Esta rutina abre una ventana de consola. La sentencia CALL va seguida por un bloque de parámetros de 4 octetos y 4 palabras que define el color del borde, el punteado, el papel y la tinta. También se define el ancho del borde, el tamaño de la ventana (se da como origen la esquina superior izquierda),el ancho y la altura de la ventana.

El formato del bloque de parámetros es:

00	octeto	color del borde
01	octeto	ancho del borde
02	octeto	color papel/punteado
03	octeto	color de la tinta
04	palabra	ancho de la ventana
06	palabra	alto de la ventana
08	palabra	origen X
0A	palabra	origen Y

UT.SCR VECTOR C8

Prepara una ventana de pantalla

Parámetros de llamada		**Parámetros retornados**	
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
A0		A0	ID del canal
A1	apunt. bloque parámet.	A1	indefinido.
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

OM no hay memoria
NO no hay canales
OR ventana fuera de la pantalla

Descripción:

Esta rutina abre una ventana de pantalla. La sentencia CALL va seguida por un bloque de parámetros de 4 octetos y 4 palabras que define el color del borde, el punteado, el papel y la tinta. También se define el ancho del borde, el tamaño de le ventana (se da como origen la esquina superior izquierda),el ancho y la altura de la ventana.

El formato del bloque de parámetros es:

00	octeto	color del borde
01	octeto	ancho del borde
02	octeto	color papel/punteado
03	octeto	color de la tinta
04	palabra	ancho de la ventana
06	palabra	alto de la ventana
08	palabra	origen X
0A	palabra	origen Y

UT.ERRO VECTOR CA

Escribe un mensaje de error al canal 0

Parámetros de llamada	Parámetros retornados
D0.L código de error	D0.L preservado
D1	D1 preservado
D2	D2 preservado
D3	D3 preservado
A0	A0 preservado
A1	A1 preservado
A2	A2 preservado
A3	A3 preservado

Error devuelto:

ninguno

Descripción:

Esta rutina se debe llamar con el número de error, del mensaje que se va a imprimir, en el registro D0. El mensaje de error se imprimirá en el canal 0.

UT.ERR VECTOR CC

Escribe un mensaje de error en un canal dado

Parámetros de llamada		**Parámetros retornados**	
D0.L	código de error	D0.L	preservado
D1		D1	preservado
D2		D2	preservado
D3		D3	preservado
A0	ID del canal	A0	preservado
A1		A1	preservado
A2		A2	preservado
A3		A3	preservado

Error devuelto:

ninguno

Descripción:

Esta rutina se debe llamar con el número de error, del mensaje que se va a imprimir, en el registro D0. El mensaje de error se imprimirá en el canal especificado en A0.

UT.MINT VECTOR CE

Convierte un entero a ASCII y lo manda a un canal

Parámetros de llamada		**Parámetros retornados**	
D1.W	valor entero	D1	indefinido
D2		D2	indefinido
D3		D3	-1 (si A0=0, D3=0)
A0	ID del canal	A0	preservado
A1		A1	indefinido
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

NC no completado (solo para canal 0)
NO el canal no está abierto
DF unidad llena

Descripción:

Esta rutina convierte el entero, pasado en D1, en ASCII. La cadena ASCII resultante se manda al canal especificado en A0. Al final de la cadena ASCII se le añade un espacio.

Las rutinas en modo supervisor de servicio de interrupcionesn, y UT.ERRO pueden llamar a esta rutina con A0=0. Si se está usando el canal de comandos, la rutina intentará usar el canal 1. Esta operación no está recomendada por Sinclair, pero parece que funciona.

UT.MTEXT VECTOR DO

Manda un mensaje a un canal

Parámetros de llamada		**Parámetros retornados**	
D1		D1	indefinido
D2		D2	indefinido
D3		D3	-1 (si A0=0, D3=0)
A0	ID del canal	A0	preservado
A1	base del mensaje	A1	indefinido
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

NC no completado (solo para canal 0)
NO el canal no está abierto
DF unidad llena

Descripción:

Esta rutina manda un mensaje a un canal. El mensaje debe estar en una cadena de caracteres apuntada por A1, y debe ir precedido por una palabra que contiene el número de caracteres. Si se requiere, el usuario debe incluir un <LF> al final del mensaje.

Las rutinas en modo supervisor de servicio de interrupciones, llaman a esta rutina con A0=0. Si el canal de comandos está en uso, la rutina intentará usar el canal 1. Esta operación ne está recomendada por Sinclair, pero parece que funciona.

8.4 Rutinas de interés general

8.4.1 Manejo de listas encadenadas

UT.LINK VECTOR D2

Conecta un item a una lista

Parámetros de llamada		**Parámetros retornados**	
D1		D1	preservado
D2		D2	preservado
D3		D3	preservado
A0	base del item	A0	preservado
A1	apuntador item anterior	A1	actualizado
A2		A2	preservado
A3		A3	preservado

Error devuelto:

ninguno

Descripción:

Esta rutina conecta un item a una lista encadenada. Se le deben pasar dos parámetros a esta rutina, el primero es la dirección del item que se va a conectar (en A0), el segundo es el apuntador a un item de la lista (en A1). Si este apuntador es al primero de la lista,el nuevo item se conectará al principio. Si no, se conectará después del item al que apunta.

Cuando se prepara una nueva lista, el apuntador al primer item debe ser cero.

Tenga en cuenta que se deben reservar cuatro octetos al principio de cada item, para el apuntador.

Ver en la sección 5.4.5 la descripción de las listas encadenadas.

UT.UNLK VECTOR D4

Desconecta un item de una lista

Parámetros de llamada		**Parámetros retornados**	
D1		D1	preservado
D2		D2	preservado
D3		D3	preservado
A0	base del item	A0	preservado
A1	apuntador item anterior	A1	actualizado
A2		A2	preservado
A3		A3	preservado

Error devuelto:

ninguno

Descripción:

Esta rutina desconecta un item de una lista encadenada. Se le deben pasar dos parámetros a esta rutina, el primero es la direción del item que se va a desconectar (en A0), el segundo es el apuntador a un item de la lista (en A1). El único requerimiento de este item es que sea el anterior al que va a ser desconectado de la lista. Puede ser también un apuntador al apuntador a la lista encadenada.

Ver en la sección 5.4.5 la descripción de las listas encadenandas.

8.4.2 Manejo del área de usuario

MM.ALOC VECTOR D8

Asigna espacio en el área de usuario

Parámetros de llamada		**Parámetros retornados**	
D1.L	longitud requerida	D1.L	longitud asignada
D2		D2	indefinido
D3		D3	indefinido
A0	apuntador al apuntador al espacio libre	A0	base del área asignada
A1		A1	indefinido
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

OM no hay memoria

Descripción:

La asignación del área del usuario, se lleva a cabo en unidades de ocho octetos. Se asignan dos palabras largas por cada espacio en el área. La primera contiene la longitud del espacio y la segunda es un apuntador relativo al siguiente espacio libre. Se usan apuntadores relativos para asegurarse que el área de usuario es reubicable. Se puede usar toda el área asignada, siempre que se recuerde en alguna parte la longitud de los items. Cuando se asigna el área, la primera palabra larga del espacio contiene la longitud del área, de forma que puede ser retenida por el código del usuario si se necesita.

El código de usuario debe mantener un apuntador a la siguiente área de espacio libre. A0 apunta a este apuntador, antes de llamar a esta rutina. Si no queda espacio libre en el área de usuario, este apuntador será cero.

Vea la sección 5.2.5 que trata de las áreas de usuario. El TRAP del gestor MT.ALLOC es la versión atómica de esta rutina de utilidad.

MM.LNKFR VECTOR DA

Conecta un espacio libre (devuelto) al área de usuario

Parámetros de llamada		**Parámetros retornados**	
D1.L	longitud a conectar	D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
A0	base nuevo espacio	A0	indefinido
A1	apuntador al apuntador al espacio libre	A1	indefinido
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

ninguno

Descripción:

Esta rutina permite conectar en el área de usuario, espacios libres devueltos. Estos espacios pueden tener dentro de ellos áreas que se asignaron mediante MM.ALLOC.

Para conectar un área de RAM, se debe conectar un bloque inexistente.

Para expandir un bloque, el área de RAM contígua a la parte alta, se debe conectar a ella. Tambien se puede usar memoria de trabajo propiedad del Job, aunque esto es menos eficiente que usar memoria contígua.

El código de usuario debe mantener un apuntador a la siguiente área de espacio libre. A1 apunta a este apuntador, antes de llamar a esta rutina. Si no hay espacio libre, el apuntador debe ser cero.

Vea la sección 5.2.5 donde se habla del área de usuario. El TRAP del gestor MT.LNKFR es la versión atómica de esta rutina.

8.4.3 Rutinas de manejo de colas

IO.QSET VECTOR DC

Prepara una cola

Parámetros de llamada		**Parámetros retornados**	
D1	longitud de la cola	D1	indefinido
D2		D2	preservado
D3		D3	preservado
A0		A0	preservado
A1		A1	preservado
A2	apuntador a la cola	A2	preservado
A3		A3	indefinido

Error devuelto:

ninguno

Descripción:

Esta rutina prepara una cola. La longitud requerida se pasa en D1 y un apuntador a la cola en A2.

En la sección 9.7 puede encontrar una descripción general de las colas de E/S.

Nota: D1 debe ser una palabra larga hasta 32767 como máximo.

IO.QTEST VECTOR DE

Comprueba el estado de una cola

Parámetros de llamada		**Parámetros retornados**	
D1		D1	siguiente octeto
D2		D2	espacio libre
D3		D3	preservado
A0		A0	preservado
A1		A1	preservado
A2	apuntador a la cola	A2	preservado
A3		A3	indefinido

Error devuelto:

NC cola vacía
EF se ha llegado al final del fichero

Descripción:

Esta rutina comprueba el estado de una cola. En el registro D2 nos devuelve el espacio libre, y en D1 el siguiente octeto que se va a recojer (el octeto no es recogido por esta rutina). Si la cola está vacía, devuelve ERR.NC.

En la sección 9.7 puede encontrar una descripción general de las colas de E/S.

IO.QIN VECTOR EO

Pone un octeto en la cola

Parámetros de llamada		Parámetros retornados	
D1	octeto a poner	D1	preservado
D2		D2	preservado
D3		D3	preservado
A0		A0	preservado
A1		A1	preservado
A2	apuntador a la cola	A2	preservado
A3		A3	indefinido

Error devuelto:

NC la cola está llena

Descripción:

Esta rutina pone un octeto en una cola. El octeto se le pasa en D1.

En la sección 9.7 hay una descripción general de la colas de E/S.

IO.QOUT VECTOR E2

Extrae un octeto de una cola

Parámetros de llamada		Parámetros retornados	
D1		D1	siguiente octeto
D2		D2	preservado
D3		D3	preservado
A0		A0	preservado
A1		A1	preservado
A2	apuntador a la cola	A2	preservado
A3		A3	indefinido

Error devuelto:

NC la cola está vacía
EF se ha llegado al final del fichero

Descripción:

Esta rutina saca un octeto de una cola. El octeto lo devuelve en D1.

En la sección 9.7 hay una descripción general de las colas de E/S.

IO.QEOF VECTOR E4

Pone un marcador de fin de fichero en una cola

Parámetros de llamada		Parámetros retornados	
D1		D1	preservado
D2		D2	preservado
D3		D3	preservado
A0		A0	preservado
A1		A1	preservado
A2	apuntador a la cola	A2	preservado
A3		A3	preservado

Error devuelto:

NC la cola está llena

Descripción:

Esta rutina pone un marcador de fin de fichero en una cola.

En la sección 9.7 hay una descripción general de las colas de E/S.

8.4.4 Utilidad de nombre de dispositivo

IO.NAME VECTOR 122

Decodifica un nombre de dispositivo

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
A0	apuntador al nombre	A0	preservado
A1		A1	indefinido
A2		A2	indefinido
A3	apuntador a parámetros	A3	preservado

Error devuelto:

NF no reconocido
BN nombre reconocido, pero parámetros inválidos

Descripción:

Esta rutina es una ayuda muy útil para decodificar los nombres de dispositivos. Comprueba el nombre de dispositivo y evalúa los parámetros adicionales.

ERR.NF retorna directamente a la instrucción siguiente a la llamada, ERR.BN retorna 2 octetos después de la llamada, y so no hubo errores retorna 4 octetos después de la llamada. La descripción del nombre del dispositivo (en ASCII) comienza 6 octetos después de la instrucción de llamada.

8.5 Rutinas de utilidad del SuperBASIC

Las rutinas de utilidad del BASIC pueden ser llamadas desde cualquier código, pero todas las direcciones que se le pasan deben ser relativas a A6.

8.5.1 Comparación de cadenas

UT.CSTR VECTOR E6

Compara dos cadenas

Parámetros de llamada		**Parámetros retornados**	
D0.B	tipo de comparación	D0.L	-1, 0 o +1
D1		D1	preservado
D2		D2	preservado
D3		D3	preservado
A0	base cadena 0	A0	preservado
A1	base cadena 1	A1	preservado
A2		A2	preservado
A3		A3	preservado
A6	dirección base	A6	preservado

Error devuelto:

ninguno

Descripción:

Esta rutina compara dos cadenas. La primera cadena está apuntada por (A6,A0) y la segunda por (A6,A1). Después de la llamada, D0 contiene uno de tres posibles valores, -1 si la primera cadena es menor, 0 si son iguales, y +1 si es mayor.

Para hacerse una idea de como trabaja esta rutina, es necesario tener una idea clara de como se comparan las cadenas. Esto es bastante complicado en el QL, ya que existen diferentes opciones de comparación.

Comparaciones

Dos cadenas que se comparan darán uno de los siguientes resultados:

Menor que las cadenas se comparan hasta que se encuentra una diferencia de caracteres (o un segmento de la cadena en caso de números). La primera cadena es menor que la segunda, de acuerdo con un orden de comparación definido (ver más adelante).

Igual a todos los caracteres y/o números de las dos cadenas son iguales y están en el mismo orden.

Mayor que las cadenas se comparan hasta que se encuentra una diferencia. El valor de la primera cadena es mayor que el de la segunda de acuerdo con un orden de comparación definido (ver más adelante).

Orden de comparación

Desde el punto de vista del orden de comparación de cadenas, es necesario que podamos clasificarlas en un tipo de orden de directorio. Este orden está bien definido y puede ser modificado selecionando uno de los cuatro modos.

El espacio está definido como el primer carácter.

Los signos de puntuación van en el orden estándar ASCII, excepto el '.' que es el último.

Todos los signos de puntuación están delante de las letras o dígitos (ej. **A?** va antes que **AA?**.

Como una opción especial, los números entre caracteres se comparan por orden numérico(ej. **Size6B** va antes que **Size12B** y también **Size.12** va antes que **Size6.4**).

Todos los dígitos o números, van antes que las letras (ej. **car1** va antes que **cart1**).

Una letra mayúscula va delante de la correspondiente minúscula, pero después que la letra minúscula anterior (ej. **Car** va antes que **car** pero después de **aid**).

Como selección especial, se pueden tratar las mayúsculas como equivalentes de las minúsculas.

Resúmen de los órdenes

```
ESPACIO
!"#$%&'()*+,-./:;<=>?@[\]^_£{|}~".
Digitos o números
AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz
```

Tenga en cuenta que este orden es para la versión inglesa, para otras versiones puede cambiar ligeramente.

Tipos de comparación

Se pueden seleccionar varios tipos de comparación. dependiendo del tipo de aplicación en la que se use la comparación de cadenas.

Tipo Ø se hace directamente carácter a carácter. Se distinguen mayúsculas de minúsculas.

Tipo 1 es el mismo que el tipo Ø, pero no se diferencian las mayúsculas de las minúsculas.

Tipo 2 en los números embebidos, se usa el orden numérico en la comparación, en vez del código del carácter. Se distingue entre mayúsculas y minúsculas.

Tipo 3 es como el tipo 2, pero no se dietingue entre mayúsculas y minúsculas.

La comparación de ficheros y nombres de variables usa el tipo 1.

Los operadores BASIC <,<=,=,>=,> y <> usan el tipo 2.

El operador BASIC == (equivalencia) usa el tipo 3.

8.5.2 Rutinas de conversión

CN.DATE VECTOR EC

Devuelve una cadena ASCII con el dia y la hora

Parámetros de llamada		**Parámetros retornados**	
D1.L	fecha (valor interno)	D1	preservado
D2		D2	preservado
D3		D3	preservado
A0		A0	preservado
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	preservado
A3		A3	preservado

Error devuelto:

ninguno

Descripción:

Esta rutina convierte el tiempo en una cadena ASCII en formato de palabra larga. La cadena toma la forma:

aaaa mmm dd hh:mm:ss

donde aaaa es el año, mmm es el mes, dd es el dia, hh la hora, mm los minutos y ss los segundos. La cadena resultante se pone en la pila a la que apunta A1. Se requieren 22 octetos en la pila.

Ver también CN.DAY que nos da el dia de forma similar.

CN.DAY VECTOR EE

Devuelve una cadena ASCII con el dia de la semana

Parámetros de llamada		**Parámetros retornados**	
D1.L	fecha (valor interno)	D1	preservado
D2		D2	preservado
D3		D3	preservado
A0		A0	preservado
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	preservado
A3		A3	preservado

Error devuelto:

ninguno

Descripción:

Esta rutina convierte el tiempo en una cadena ASCII en formato de palabra larga con el dia de la semana. La cadena toma la forma de las tres priemras letras del dia de la semana. La cadena resultante se pone en la pila a la que apunta A1. Se requieren 6 octetos en la pila.

Ver también CN.DAY que nos da el dia de forma similar.

CN.FTOD VECTOR FO

Convierte un número de coma flotanta en una cadena ASCII

Parámetros de llamada		**Parámetros retornados**	
D0		D0	indefinido
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
A0	apuntador a mem. int.	A0	apuntador a mem. int.
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

indefinido

Descripción:

Esta rutina convierte un número de coma flotante en la pila (apuntado por A1) en una cadena de caracteres ASCII en una memoria intermedia (apuntada por A0).

CN.ITOD VECTOR F2

Convierte un entero en una cadena ASCII decimal

Parámetros de llamada		**Parámetros retornados**	
D0		D0	indefinido
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
A0	apuntador a mem. int.	A0	apuntador a mem. int.
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

indefinido

Descripción:

Esta rutina convierte un número entero en la pila (apuntado por A1) en una cadena de caracteres ASCII en una memoria intermedia (apuntada por A0), formando un número decimal.

CN.ITOBB VECTOR F4

Convierte un octeto en una cadena ASCII en binario

Parámetros de llamada		**Parámetros retornados**	
D0		D0	indefinido
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
A0	apuntador a mem. int.	A0	apuntador a mem. int.
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

indefinido

Descripción:

Esta rutina convierte un octeto en la pila (apuntado por A1) en una cadena de caracteres ASCII en una memoria intermedia (apuntada por A0) formando un número binario de 8 caracteres. Por ejemplo, si se convierte el octeto $C5, producirá los caracteres '11000101'.

CN.ITOBW VECTOR F6

Convierte una palabra a una cadena ASCII en binario

Parámetros de llamada		**Parámetros retornados**	
D0		D0	indefinido
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
A0	apuntador a mem. int.	A0	apuntador a mem. int.
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

indefinido

Descripción:

Esta rutina convierte una palabra en la pila (apuntado por A1) en una cadena de caracteres ASCII en una memoria intermedia (apuntada por A0), formando un número binario de 16 caracteres.

CN.ITOBL VECTOR F8

Convierte una palabra larga en una cadena ASCII en binario

Parámetros de llamada		Parámetros retornados	
D0		D0	indefinido
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
A0	apuntador a mem. int.	A0	apuntador a mem. int.
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

indefinido

Descripción:

Esta rutina convierte una palabra larga en la pila (apuntada por A1) en una cadena de caracteres ASCII en una memoria intermedia (apuntada por A0), formando un número binario de 32 caracteres.

CN.ITOHB VECTOR FA

Convierte un octeto en una cadena ASCII hexadecimal

Parámetros de llamada		**Parámetros retornados**	
D0		D0	indefinido
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
A0	apuntador a mem. int.	A0	apuntador a mem. int.
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

indefinido

Descripción:

Esta rutina convierte un octeto en la pila (apuntado por A1) en una cadena de caracteres ASCII en una memoria intermedia (apuntada por A0), formando un número hexadecimal de dos dígitos.

CN.ITOHW VECTOR FC

Convierte una palabra en una cadena ASCII en hexadecimal

Parámetros de llamada		**Parámetros retornados**	
D0		D0	indefinido
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
A0	apuntador a mem. int.	A0	apuntador a mem. int.
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

indefinido

Descripción:

Esta rutina convierte una palabra en la pila (apuntada por A1) en una cadena de caracteres ASCII en una memoria intermedia (apuntada por A0), formando un número hexadecimal de cuatro dígitos.

CN.ITOHL VECTOR FE

Convierte una palabra larga en una cadena ASCII hexadecimal

Parámetros de llamada		**Parámetros retornados**	
D0		D0	indefinido
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
A0	apuntador a mem. int.	A0	apuntador a mem. int.
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

indefinido

Descripción:

Esta rutina convierte una palabra larga en la pila (apuntada por A1) en una cadena de caracteres ASCII en una memoria intermedia (apuntada por A0), formando un número hexadecimal de ocho dígitos.

CN.DTOF VECTOR 100

Convierte una cadena ASCII de coma flotante en un número de coma flotante

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
D7	0 o apuntador al final de memoria intermedia	D7	preservado
A0	apuntador a mem. int.	A0	apuntador a mem. int.
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

XP error en conversión (ej 1..0)

Descripción:

Esta rutina convierte una cadena de caracteres ASCII en una memoria intermedia (apuntada por A0) en un número de coma flotante en la pila (apuntado por A1). Si hay un error, A0 y A1 permanecen sin cambio. Si termina correctamente, A0 apunta al siguiente carácter en la memoria intermedia.

CN.DTOI VECTOR 102

Convierte una cadena ASCII decimal en un entero

Parámetros de llamada		**Parámetros retornados**	
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
D7	0 o apuntador al final de memoria intermedia	D7	preservado
A0	apuntador a mem. int.	A0	apuntador a mem. int.
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

XP error en conversión (ej 1A)

Descripción:

Esta rutina convierte una cadena de caracteres ASCII en una memoria intermedia (apuntada por A0) en un número entero de una palabra larga de longitud en la pila (apuntado por A1). Si hay un error, A0 y A1 permanecen sin cambio. Si termina correctamente, A0 apunta al siguiente carácter en la memoria intermedia.

CN.BTOIB VECTOR 104

Convierte una cadena ASCII binaria de 8 bits en un octeto

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
D7	0 o apuntador al final de memoria intermedia	D7	preservado
A0	apuntador a mem. int.	A0	apuntador a mem. int.
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

XP error en conversión (ej 10110201)

Descripción:

Esta rutina convierte una cadena de caracteres ASCII en una memoria intermedia (apuntada por A0) en un octeto en la pila (apuntado por A1). Si hay un error, A0 y A1 permanecen sin cambio. Si termina correctamente, A0 apunta al siguiente carácter en la memoria intermedia.

Nota: esta rutina no funciona en QDOS V1.03 y anteriores.

CN.BTOIW VECTOR 106

Convierte cadena ASCII de 16 bits binarios en una palabra

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
D7	0 o apuntador al final de memoria intermedia	D7	preservado
A0	apuntador a mem. int.	A0	apuntador a mem. int.
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

XP error en conversión

Descripción:

Esta rutina convierte una cadena de caracteres ASCII en una memoria intermedia (apuntada por A0) en una palabra en la pila (apuntado por A1). Si hay un error, A0 y A1 permanecen sin cambio. Si termina correctamente, A0 apunta al siguiente carácter en la memoria intermedia.

NOTA: esta rutina no funciona en QDOS V1.03 y anteriores.

CN.BTOIL VECTOR 108

Convierte una cadena ASCII binaria de 32 bits en una palabra larga

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
D7	0 o apuntador al final de memoria intermedia	D7	preservado
A0	apuntador a mem. int.	A0	apuntador a mem. int.
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

XP error en conversión

Descripción:

Esta rutina convierte una cadena de caracteres ASCII en una memoria intermedia (apuntada por A0) en una palabra larga en la pila (apuntado por A1). Si hay un error, A0 y A1 permanecen sin cambio. Si termina correctamente, A0 apunta al siguiente carácter en la memoria intermedia.

NOTA: esta rutina no funciona en QDOS V1.03 y anteriores

CN.HTOIB VECTOR 10A

Convierte una cadena ASCII de 2 caracteres hex en un octeto

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
D7	0 o apuntador al final de memoria intermedia	D7	preservado
A0	apuntador a mem. int.	A0	apuntador a mem. int.
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

XP error en conversión (ej 1E4R)

Descripción:

Esta rutina convierte una cadena de caracteres ASCII en una memoria intermedia (apuntada por A0) en un octeto en la pila (apuntado por A1). Si hay un error, A0 y A1 permanecen sin cambio. Si termina correctamente, A0 apunta al siguiente carácter en la memoria intermedia.

Nota: esta rutina no funciona en QDOS V1.03 y anteriores.

CN.HTOIW VECTOR 10C

Convierte una cadena ASCII de 4 caracteres en una palabra

Parámetros de llamada		**Parámetros retornados**	
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
D7	0 o apuntador al final de memoria intermedia	D7	preservado
A0	apuntador a mem. int.	A0	apuntador a mem. int.
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

XP error en conversión

Descripción:

Esta rutina convierte una cadena de caracteres ASCII en una memoria intermedia (apuntada por A0) en una palabra en la pila (apuntado por A1). Si hay un error, A0 y A1 permanecen sin cambio. Si termina correctamente, A0 apunta al siguiente carácter en la memoria intermedia.

NOTA: esta rutina no funciona en QDOS V1.03 y anteriores.

CN.HTOIL VECTOR 10E

Convierte una cadena ASCII de 8 caracteres en una palabra larga

Parámetros de llamada		**Parámetros retornados**	
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
D7	0 o apuntador al final de memoria intermedia	D7	preservado
A0	apuntador a mem. int.	A0	apuntador a mem. int.
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	indefinido
A3		A3	indefinido

Error devuelto:

XP error en conversión

Descripción:

Esta rutina convierte una cadena de caracteres ASCII en una memoria intermedia (apuntada por A0) en una palabra larga en la pila (apuntado por A1). Si hay un error, A0 y A1 permanecen sin cambio. Si termina correctamente, A0 apunta al siguiente carácter en la memoria intermedia.

NOTA: esta rutina no funciona en QDOS V1.03 y anteriores

8.5.3 Rutinas de soporte de procedimientos BASIC

Estas rutinas se listan solamente para asegurarse de que la sección de utilidades por vector está completa. Solamente se da una explicación somera de su modo de operación. Para una explicación más detallada de los términos usados y la comprensión de su manejo y aplicaciones, refierase al capítulo 10 de Intercomunicación con el SuperBASIC.

BP.INIT VECTOR 110

Inicializa un procedimiento o función en la tabla de nombres del BASIC

Parámetros de llamada		**Parámetros retornados**	
D1		D1	preservado
D2		D2	preservado
D3		D3	preservado
A0		A0	preservado
A1	apunt. lista definición	A1	indefinido
A2		A2	preservado
A3		A3	preservado

Error devuelto:

ninguno

Descripción:

Esta rutina de soporte permite conectar procedimientos o funciones BASIC en la lista de tablas de nombres del BASIC. A1 apunta al principio de la lista de definición de procedimientos antes de llamar a la rutina. Ver sección 10.7.2 para más detalles.

CA.GINIT VECTOR 112

Obtiene parámetros enteros del BASIC

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2	indefinido
D3		D3.W	número de argumentos obtenidos
D4		D4	indefinido
D6		D6	indefinido
A0		A0	indefinido
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	indefinido
A3	apunt. primer param.	A3	preservado
A5	apunt. ultimo param.	A5	preservado

Error devuelto:

BP parámetros inválidos
XP error en la expresión

Descripción:

Esta rutina obtiene un número indeterminado de argumentos de procedimiento enteros (palabra), y los pone en la pila aritmética. A3 apunta al principio de los parámetros y A5 al final de los parámetros. Todas las referencias se deben hacer relativas a A6. Para más detalles ver la sección 10.7.4.

CA.GTFP VECTOR 114

Obtiene parámetros de coma flotante del BASIC

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2	indefinido
D3		D3.W	número de argumentos obtenidos
D4		D4	indefinido
D6		D6	indefinido
A0		A0	indefinido
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	indefinido
A3	apunt. primer param.s	A3	preservado
A5	apunt. ultimo param.	A5	preservado

Error devuelto:

BP parámetros inválidos
XP error en la expresión

Descripción:

Esta rutina obtiene un número indeterminado de argumentos de procedimiento, de coma flotante (6 octetos), y los pone en la pila aritmética. A3 apunta al principio de los parámetros y A5 al final de los parámetros. Todas las referencias se deben hacer relativas a A6. Para más detalles ver la sección 10.7.4.

CA.GTSTR VECTOR 116

Obtiene parámetros de cadena del BASIC

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2	indefinido
D3		D3.W	número de argumentos obtenidos
D4		D4	indefinido
D6		D6	indefinido
A0		A0	indefinido
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	indefinido
A3	apunt. primer param.	A3	preservado
A5	apunt. ultimo param.	A5	preservado

Error devuelto:

BP parámetros inválidos
XP error en la expresión

Descripción:

Esta rutina obtiene un número indeterminado de argumentos de procedimiento en forma de cadenas (2+n(impar) o 3+n(par) octetos) y los pone en la pila aritmética. A3 apunta al principio de los parámetros y A5 al final de los parámetros. Todas las referencias se deben hacer relativas a A6. Para más detalles ver la sección 10.7.4.

CA.GTLIN VECTOR 118

Obtiene parámetros enteros largos, del BASIC

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2	indefinido
D3		D3.W	número de argumentos obtenidos
D4		D4	indefinido
D6		D6	indefinido
A0		A0	indefinido
A1	apuntador a la pila	A1	apuntador a la pila
A2		A2	indefinido
A3	apunt. primer param.s	A3	preservado
A5	apunt. ultimo param.	A5	preservado

Error devuelto:

BP parámetros inválidos
XP error en la expresión

Descripción:

Esta rutina obtiene un número indeterminado de argumentos de procedimiento, en forma de enteros (palabras largas) y los pone en la pila aritmética. A3 apunta al principio de los parámetros y A5 al final de los parámetros. Todas las referencias se deben hacer relativas a A6. Para más detalles ver la sección 10.7.4.

BV.CHRIX VECTOR 11A

Asigna espacio en la pila aritmética

Parámetros de llamada	**Parámetros retornados**
D1.L # octetos requeridos	D1 indefinido
D2	D2 indefinido
D3	D3 indefinido
A0	A0 preservado
A1	A1 preservado
A2	A2 preservado
A3	A3 preservado

Error devuelto:

indefinido

Descripción:

Esta rutina permite asignar espacio en la pila aritmética, para las rutinas de usuario. Las rutinas de obtención de argumentos, reservan por sí mismas suficiente espacio en la pila. En el registro D1 se le pasa el número de octetos que debe asignar. Ver seción 10.7.3 para más detalles.

BP.LET VECTOR 120

Retorna un valor de parámetro al BASIC

Parámetros de llamada		**Parámetros retornados**	
D1		D1	indefinido
D2		D2	indefinido
D3		D3	indefinido
A0		A0	indefinido
A1		A1	indefinido
A2		A2	indefinido
A3	apunt. a entr. de tabla	A3	apunt. a entr. de tabla

Error devuelto:

BP parámetros inválidos

Descripción:

Esta rutina permite devolver valores a un procedimiento o función a través de una lista de parámetros. Los parámetros deben estar en la pila aritmética, antes de llamarla. Deben estar en la forma requerida por los parámetros de llamada. A3 apuntará a la entrada correspondiente en la tabla de nombres. D0 dará el código de error. Ver seción 10.7.6 para más detalles.

8.5.4 Las rutinas aritméticas

El paquete aritmético del QL se encuentra disponible para ser usado en dos formas. La primera es RI.EXEC que permite realizar una operación simple, como la suma de dos números. La otra forma es RI.EXECB que permite realizar una lista completa de operaciones.

El 68008 es capaz de realizar operaciones aritméticas en octetos, palabras y palabras largas, de forma que no necesita que lo soporten las rutinas aritméticas. Todas las operaciones se realizan en formato de coma flotante. Antes de entrar en particularidades, es necesario comprender unos pocos conceptos básicos.

A diferencia del 68008, no hay *registros* para almacenar datos, en el *paquete arimético*. Todas las operaciones se realizan desde y hacia la pila aritmética. Por ejemplo, si tenemos dos variables BASIC VAL1 y VAL2 almacenadas en cualquier lugar de la memoria, para que se puedan sumar, cada una debe ser copiada desde la memoria a la pila aritmética. Se ejecuta una operación ADD con RI.EXEC y se pone en la pila, desde donde se puede volcar de nuevo a la memoria.

La pila aritmética opera hacia abajo desde (A6,A1.L). Los valores que se toman son los oue hay en lo alto de la pila (llamado TOS) y el valor de debajo (llamado NOS). Algunas operaciones se ralizan solamente con el TOS y el resultado se coloca en el lugar que ocupaba esta. Hay otras operaciones que operan con ambos, TOS y NOS.

(A6,A1.l) apunta al TOS mientras que al NOS lo apuntan 6(A6,A1.L). La figura 8.1 nos muestra el formato de la pila.

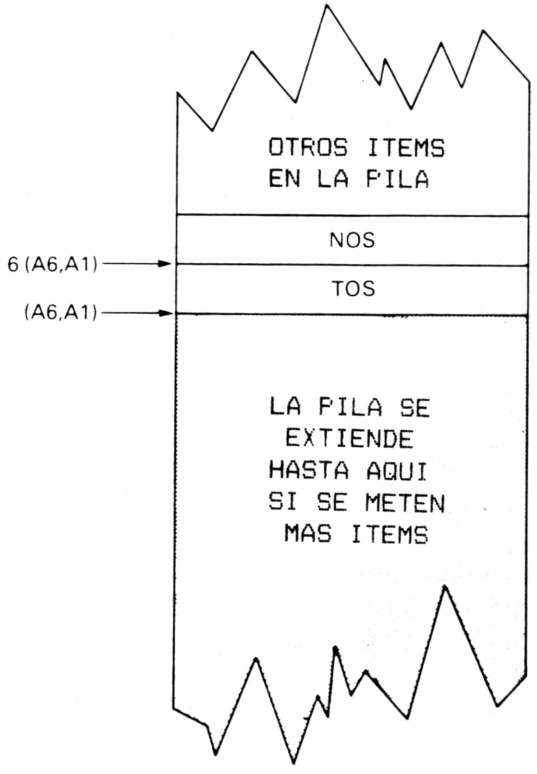


Figura 8.1 - La pila de coma flotante

Los números de coma flotante en el QL, se almacenan en forma de una palabra larga para la mantisa y dos octetos de exponente (los cuatro bits más significativos van a ceros). Aquí se puede ver:

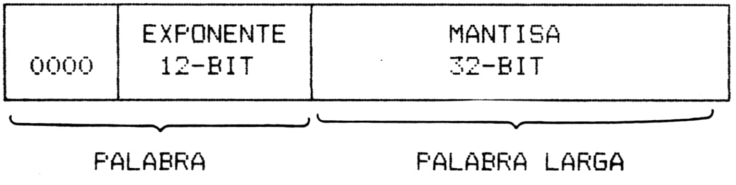


Figura 8.2 - Formato de almacenamiento de los números de coma flotante

El valor de coma flotante es:

mantisa * (2^(exponente - $800 - $1F))

El paquete de coma flotante acepta dos tipos de código de operación. Los códigos entre $01 y $30 son operaciones de aritmética real. Los otros códigos se exteinden a una palabra negativa entre $FFFF y $FF31 inclusive y son instrucciones de carga y almacenamiento de números de coma flotante en las direcciones dadas por (A6.L+A4.L+((cod.oper OR $FF00) AND $FFEE)). Si el bit cero del código de operación es 0, es de carga. Si es 1, l la operación es de almacenamiento. La carga hace que se decremente A6 en 6 octetos y el almacenamiento hace que A1 se incremente en 6. El NOS viejo se convierte en el TOS nuevo.

Códigos de operación del paquete aritmético

Cód.	Nombre	Cambio de A1	Operación
00	RI.TERM		
02	RI.NINT	+4	busca el entero más cercano a TOS
04	RI.INT	+4	trunca TOS en un entero
06	RI.NLINT	+2	entero largo más cercano a TOS
08	RI.FLOAT	-4	convierte TOS de entero a coma flotante
0A	RI.ADD	+6	suma TOS a NOS
0C	RI.SUB	+6	resta TOS de NOS
0E	RI.MULT	+6	multiplica TOS por NOS
10	RI.DIV	+6	divide TOS entre NOS
12	RI.ABS	0	pone TOS en positivo
14	RI.NEG	0	niega TOS
16	RI.DUP	-6	duplica TOS
18	RI.COS	0	saca el coseno de TOS
1A	RI.SIN	0	saca el seno de TOS
1C	RI.TAN	0	saca la tangente de TOS
1E	RI.COT	0	saca la cotangente de TOS
20	RI.ASIN	0	saca el arco seno de TOS
22	RI.ACOS	0	saca el arco coseno de TOS
24	RI.ATAN	0	saca el arco tangente de TOS
26	RI.ACOT	0	saca el arco cotangente de TOS
28	RI.SQRT	0	extrae la raíz cuadrada de TOS
2A	RI.LN	0	saca el logaritmo natural de TOS
2C	RI.LOG10	0	saca el logaritmo en base 10 de TOS
2E	RI.EXP	0	saca el exponencial de TOS
30	RI.POWFP	+6	eleva NOS a la potencia TOS
00	RI.LOAD		clave operando, si bit=0 es cargar
01	RI.STORE		clave operando, si bit=1 es salvar

RI.EXEC VECTOR 11C

Ejecuta una operación aritmética

Parámetros de llamada		**Parámetros retornados**	
D0.W	código de operación	D0	código de error
D1		D1	preservado
D2		D2	preservado
D3		D3	preservado
D7	debe ser puesto a cero	D7	preservado
A0		A0	preservado
A1	apunt. pila aritmética	A1	actualizado
A2		A2	preservado
A3		A3	preservado
A4	apunt. área variables	A4	preservado

Error devuelto:

OV rebasamiento aritmético

Descripción:

Esta rutina permite realizar una operación con el paquete aritmético, como se ha descrito en la primera parte de esta sección. El código de operación se pasa como una palabra en D0. A1 apunta a la pila aritmética. A4 apunta a la base del área de variables. Tenga en cuenta que, en el QDOS V1.03 y anteriores, se debe poner el registro D7 a ceros.

RI.EXECB VECTOR 11E

Ejecuta una lista de operaciones aritméticas

Parámetros de llamada		Parámetros retornados	
D1		D1	preservado
D2		D2	preservado
D3		D3	preservado
D7	debe ser puesto a cero	D7	preservado
A0		A0	preservado
A1	apunt. pila aritmética	A1	actualizado
A2		A2	preservado
A3	apunt. lista operacio.	A3	preservado
A4	apunt. área variables	A4	preservado

Error devuelto:

OV rebasamiento aritmético

Descripción:

Esta rutina permite realizar una lista de operaciones con el paquete aritmético, como se ha descrito en la primera parte de esta sección. Los códigos de operación se pasan como una tabla de octetos, apuntada por A3. A1 apunta a la pila aritmética. A4 apunta a la base del área de variables. Tenga en cuenta que, en el QDOS V1.03 y anteriores, se debe poner el registro D7 a ceros.

8.5.5 SINH como función residente

Este programa de ejemplo nos muestra cómo usar las utilidades que se han descrito al principio de esta sección. Vamos a construir una función residente para calcular el valor de de SINH(x) donde 'x' es un valor de coma flotante.

Ahora nos vamos a referir al listado del programa SINH al final de esta subsección.

Se llama a la rutina de inicialización de un procedimiento BASIC. Esto hace que se encadene a la lista de procedimiento /función del BASIC, el bloque de definición de procedimiento en PROC_TAB. Una vez inciado de esta forma, podemos acceder a la rutina SINH como si fuera otra función de BASIC.

Una vez que ha entrado la rutina SINH, asigna espacioen la pila aritmética, para 12 números de coma flotante (usando BV.CHRIX). Los parámetros que se pasan a la función se pueden obtener usando CA.GTFP. En este paso, dará un error si le pasamos más, o menos, de un parámetro.

El algoritmo estándar para calcular la función SINH es:

$$\sinh x = (e^{x} - e^{-x})/2$$

Ahora debemos preparar la pila aritmética para poner en ella todos los valores requeridos. Si al parámetro le llamamos 'X', ponemos 'X','1.8' y '0.5' en la pila. Las dos constantes se usarán más adelante.

La secuencia de operaciones que realiza el paquete aritmético es:

```
carga X en TOS
toma el exponencial de TOS
carga 1.0 en TOS
carga exp(X) en TOS
divide NOS por TOS para generar exp(-X)
resta NOS de TOS
carga 0.5 en la pila en TOS
multiplica TOS por NOS
pone TOS de vuelta en X
y restaura A1
```

En este punto tenemos el resultado de SINH(X) en lo alto de la pila, de modo que devolvemos este valor como el valor de la función, A1 (apuntador a la pila) se almacena en BV_RIP.

```
*
* SINH
*
*
BP.INIT  EQU    $110
CA.6TFP  EQU    $114
BV.CHRIX EQU    $11A
RI.EXECB EQU    $11E
RI.SUB   EQU    $0C
RI.MULT  EQU    $0E
RI.DIV   EQU    $10
RI.EXP   EQU    $2E
RI.STORE EQU    $1
*
BV_RIP   EQU    $58
*
ERR.BP   EQU    -15
*
        LEA     PROC_TAB(PC),A1  carga dirección procedimiento
        MOVE.W  BP.INIT,A2       tabla definición y añadelo
        JSR     (A2)             a la tabla de BASIC
        MOVEQ   #0,D0            retorno sin error
        RTS
*
PROC_TAB
        DC.W    0                NO procedimientos
        DC.W    0                fin de procedimientos
        DC.W    1                una función
        DC.W    SINH-*
        DC.B    4,'SINH',0       5 octetos + 0 para alinear
        DC.W    0                fin de funciones
*
* Función SINH
*
SINH
        MOVEQ   #72,D1           reserva espacio para 12
        MOVE.W  BV.CHRIX,A2      números de coma flotante
        JSR     (A2)
*
        MOVE.W  CA.6TFP,A2       toma uno de los números
        JSR     (A2)             de coma flotante
        BNE.S   SINH_RTS         ... hooop
        SUBQ.W  #1,D3            pero solo queremos uno
        BNE.S   ERR_BP
```

```
*
        LEA     6(A1),A4        alto del apuntador de pila
        CLR.W   -2(A6,A1.L)             pon 1.0
        MOVE.L  #$08014000,-6(A6,A1.L)
        CLR.W   -8(A6,A1.L)             pon 0.5
        MOVE.L  #$08004000,-12(A6,A1.L)
        SUB.W   #12,A1
        LEA     OP_TAP(PC),A3   prepara tabla de operaciones
        MOVE.W  RI.EXECB,A2     usa paquete aritmético
        JSR     (A2)
*
        ADD.W   #12,A1          limpia pila aritmética
        MOVE.L  A1,BV_RIP(A6)
        MOVEQ   #2,D4           y pone tipo
SINH_RTS
        RTS
ERR_BP
        MOVEQ   #ERR.BP,D0
        RTS
*
* Contenido de la pila RI
*
RIS_VAL EQU     -6
RIS_1.0 EQU     -12
RIS_0.5 EQU     -18
RIS_EXP EQU     -24
*
OP_TAB
        DC.B    RIS_VAL         Pila    x
        DC.B    RI.EXP                  exp(x)
        DC.B    RIS_1.0                 exp(x),1.0
        DC.B    RIS_EXP                 exp(x),1.0,exp(x)
        DC.B    RI.DIV                  exp(x),exp(-)
        DC.B    RI.SUB                  exp(x)-exp(-x)
        DC.B    RIS_0.5                 exp(x)-exp(-x),0.5
        DC.B    RI.MULT                 (exp(x)-exp(-x))/2
        DC.B    RIS_VAL+RI.STORE
        DC.B    0
```

8.6 Rutinas de soporte de microdrives

Normalmente se accede a los datos almacenados en los microdrives a través del sistema redireccionable de ES. El manejo general de ficheros y el acceso de datos, se deben realizar a través del IOSS. Sin embargo, en ciertas aplicaciones es necesario tener un mayor control de la lectura y escritura de datos en los microdrives. El QDOS nos proporciona una selección de rutinas para este tipo de aplicaciones.

Probablemente encontrará útil referirse al apéndice D antes de leer el resto de esta sección.

Ya que no es posible posicionar la cabeza del microdrive en un sector particular de la cinta, es necesario encontrar la posición correcta leyendo los datos de la cinta. La secuencia que sigue para encontrar un sector es:

1. selecciona la unidad requerida
2. para cada sector requerido;
   lee las cabeceras de sector hasta que encuentra el que está buscando
   realiza la operación requerida en 100us (escribir) o 2ms (leer/verificar)
3. libera la unidad

Las siguientes secciones describen cada una de estas operaciones en detalle. Las variables del sistema y los registros 'hardware' usados en las rutinas son:

```
SV_TMODE  $A0  copia y trasmite el registro de modo

SV_MDRUN  $EE  = 0 si mdv está en marcha, si no, = # unidad

PC_TCTRL  $18002 trasmite registro de control
    bits 0-2   velocidad de trasmisión
    bits 3,4   modo de trasmisión (10=microdrive selecc.)
    bit  7     salida directa

PC_MCTRL  $18020 registro de control de microdrive
    bit 0      selecciona bit de microdrive
    bit 1      selecciona reloj microdrive
    bit 2      0=leer,  1=escribir
    bit 3      1=modo borrado
```

Una rutina para seleccionar y deseleccionar puede ser esta:

```
SELECT    MOVEQ  #3,D2         reloj alto/bit selcc. puesto
          SUBQ.W #1,D1
          BRA.S  SEL_LOOP
*
DESELECT  MOVEQ  #2,D2         reloj alto/bit selecc. quitado
          MOVEQ  #7,D1
*
SEL_LOOP  MOVE.B D2,(A3)       reloj alto
          MOVEQ  #60,D0        espera
          ROR.L  D0,D0         ... 150 ciclos (20us)
          AND.B  #1,D2         relcj bajo
          MOVE.B D2,(A3)
          MOVEQ  #60,D0        espera
          ROR.L  D0,D0         ... 150 ciclos (20us)
          MOVEQ  #2,D2         reloj alto de nuevo
          DBRA   D1,SEL_LOOP
          RTS
```

8.6.1 Seleccionar un microdrive

Para seleccionar un microdrive particular, se debe usar la siguiente secuencia:

entrar en modo supervisor
poner en A6 la base de las variables del sistema (MT.INF)
esperar a que pare el mdv (SV_MDRUN(A6)=0)
inhabilitar interrupciones
esperar al menos 250ms para limpiar trasmisión serie
cargar $18020 en A3
ponerse en modo microdrive, ej.:

```
MOVEQ  #$E7,D0         desenmascarar modo trasmisión
AND.B  SV_TMODE(A6),D0 ... desde modo compuesto
OR.B   #$10,D0         poner modo microdrive
MOVE.B D0,-$1E(A3)     ... en chip de periféricos
MOVE.B D0,SV_TMODE(A6) ... y en variables sistema
```

cargar en D1.W el número del microdrive requerido
llamar a la rutina SELECT

8.6.2 Buscar un sector

En D1.w ponemos el número del sector que buscamos, y en A3 cargamos $18020 (registro de control de microdrive)

```
SEARCH      MOVE.W  #1000,D0         hacer 1000 intentos
:
SEARCH_LOOP MOVEM.W D0/D1,-(SP)
:
SEARCH_LI   MOVE.W  $12A,A2          llama a MD.SECTR
            JSR     $4000(A2)
            BRA.S   BAD_MEDIUM       devol. error medio erróneo
            BRA.S   SEARCH_LI        devol. no cabecera sector
            MOVEM.W (SP)+,D0/D1
            CMP.B   D1,D7            sector correcto?
            DBEQ    D0,SEARCH_LOOP
            RTS
```

La rutina SEARCH devuelve EQ si se ha encontrado el sector, y NE si no se ha encontrado. En este punto es en donde se debe usar una de las rutinas de leer, escribir o verificar.

8.6.3 Deseleccionar un microdrive

Se debe usar la siguiente secuencia para deseleccionar un microdrive determinado:

llamar a DESELECT
quitar el modo; ej.

```
MOVEQ  #$E7,D0          desenmascarar modo trasmisión
AND.B  SV_TMODE(A6),D0  ... desde modo compuesto
MOVE.B D0,-$1E(A3)      ... en chip de periféricos
MOVE.B D0,SV_TMODE(A6)  ... y en variables sistema
```

poner modo usuario y habilitar interrupciones

MD.READ VECTOR 124

Lee un sector de un microdrive

Parámetros de llamada		Parámetros retornados	
D1		D1	número de fichero
D2		D2	número de bloque
D7		D7	indefinido
A0		A0	indefinido
A1	apuntador al principio de memoria intermedia	A1	apuntador al final de la memoria intermedia
A2		A2	indefinido
A3	$18020	A3	$18020

Error devuelto:

normal = ha fallado
retorno+2 = OK

Descripción:

Esta rutina lee un sector de un microdrive. Todos los registros, excepto A3 y A6, son volátiles. A3 debe apuntar al registro de control del microdrive, a la entrada , y se deben inhabilitar las interrupciones. A la vuelta, no se pone D0, pero tiene dos posibles retornos; el normal que implica que ha fallado la lectura, y el normal más dos que indica que ha leido correctamente.

El vector de esta rutina apunta $4000 octetos antes del código real. Se puede usar el siguiente código para invocarla:

```
MOVE.W  aa.aaaa,An
JSR     $4000(An)
RET1.W  si retorna en este punto, ha fallado
RET2.W  si retorna aquí, ha leído correctamente
```

MD.WRITE VECTOR 126

Escribe un sector en un microdrive

Parámetros de llamada		Parámetros retornados	
D1		D1	indefinido
D2		D2	indefinido
D7		D7	indefinido
A0		A0	indefinido
A1	apuntador al principio de memoria intermedia	A1	apuntador al final de la memoria intermedia
A2		A2	indefinido
A3	$18020	A3	$18020

Error devuelto:

ninguno

Descripción:

Esta rutina lee un sector de un microdrive. Todos los registros, excepto A3 y A6, son volátiles. A3 debe apuntar al registro de control del microdrive, a la entrada , y se deben inhabilitar las interrupciones. Antaes de llamar a MD. WRITE, el apuntador de la pila debe apuntar a dos palabras; la primera es el número del fichero, y la segunda el número de bloque del sector que se está escribiendo.

El vector de esta rutina apunta $4000 octetos antes del código real. Se puede usar el siguiente código para invocarla:

```
MOVE.W  aa.aaaa,An
JSR     $4000(An)
```

MD.VERIN VECTOR 128

Verfifica un sector de un microdrive

Parámetros de llamada		Parámetros retornados	
D1		D1	número de fichero
D2		D2	número de bloque
D7		D7	indefinido
A0		A0	indefinido
A1	apuntador al principio de memoria intermedia	A1	apuntador al final de la memoria intermedia
A2		A2	indefinido
A3	$18020	A3	$18020

Error devuelto:

normal = ha fallado
retorno+2 = OK

Descripción:

Esta rutina verifica un sector de un microdrive. Todos los registros, excepto A3 y A6, son volátiles. A3 debe apuntar al registro de control del microdrive, a la entrada , y se deben inhabilitar las interrupciones. A la vuelta, no se pone D0, pero tiene dos posibles retornos; el normal que implica que ha fallado la lectura, y el normal más dos que indica que ha leido correctamente.

El vector de esta rutina apunta $4000 octetos antes del código real. Se puede usar el siguiente código para invocarla:

```
MOVE.W  aa.aaaa,An
JSR     $4000(An)
RET1.W  si retorna en este punto, ha fallado
RET2.W  si retorna aqui, ha leido correctamente
```

MD.SECTR VECTOR 12A

Lee una cabecera de sector de un microdrive

Parámetros de llamada		Parámetros retornados	
D1		D1	
D2		D2	
D7		D7	número de secotr
A0		A0	indefinido
A1	apuntador al principio de memoria intermedia	A1	apuntador al final de la memoria intermedia
A2		A2	indefinido
A3	$18020	A3	$18020

Error devuelto:

normal = medio inválido
retorno+2 = cabecera de sector inválida
retorno+4 = OK

Descripción:

Esta rutina lee una cabecera de sector de un microdrive. Todos los registros, excepto A3 y A6, son volátiles. A3 debe apuntar al registro de control del microdrive, a la entrada, y se deben inhabilitar las interrupciones. A la vuelta, no se pone D0, pero tiene dos posibles retornos; el normal que indica que el medio está mal, el normal más 2 que implica que la cabecera del sector está mal, y el normal más 4 que indica que ha leido correctamente.

El vector de esta rutina apunta $4000 octetos antes del código real. Se puede usar el siguiente código para invocarla:

```
MOVE.W  aa.aaaa,An
JSR     $4000(An)
RET1.W  si retorna en este punto, medio erróneo
RET2.W  si retorna aquí, cabecera inválida
RET3.W  si retorna aquí, ha leido correctamente
```

9 Controladores de dispositivo

9.1 Introducción

En este capítulo se tratan las formas en las que los dispositivos típicos se conectan al sistema operativo del QL. Como hemos podido apreciar en los primeros capítulos, el QL tiene un sistema de E/S redireccionable, bastante potente. Para que este sistema de E/S funcione igualmente bien con una gran variedad de dispositivos físicos, es necesario tener un mecanismo de interconexión eficaz. Este mecanismo de interconexión lo proporcionan los *controladores de dispositivo*.

El controlador de dispositivo se sitúa en medio de dos procesos. En una parte está el QL IOSS (subsistema de E/S). Este espera que el controlador de dispositivo abra y cierre los canales y mande o reciba datos por éllos. En la otra parte, el controlador de dispositivo debe ser capaz de controlar ciertas piezas de 'hardware'. El control se pasa normalmente siguiendo la interrupción de un dispositivo, o la interrupción de los 50/60Hz del QL, o cuando lo invoca el planificador.

El controlador de dispositivo se puede considerar como un sistema de comunicación entre dos medios separados. El *mediador de acceso*, comunica con el IOSS. Se deben proporcionar tres puntos de entrada al IOSS. Uno para abrir un canal, uno para cerrarlo, y uno bidireccional que permite transferir datos entre el dispositivo y el IOSS. El otro mediador se llama el *mediador físico*, y su función esencial es el control del dispositivo físico y la comunicación de datos con él.

El *mediador físico* debe ser un dispositivo altamente especializado, ya que la mayoría de los dispositivos físicos son bastante similares. Sin embargo el procedimiento de interconexión del mediador de acceso es muy espcífico y debe seguir unas normas muy rígidas, independientemente del dispositivo que esté controlando.

El controlador de dispositivo debe mantener espacio en la RAM para almacenar cierto tipo de información. A Cada controlador de dispositivo se le asigna un *bloque de definición de controlador de dispositivo*, que puede estar en el bloque de variables para los controladores de dispositivos integrados, o en otra parte de la memoria (con algunas restricciones en su localización) para los controladores adi-

cionales. Este bloque de RAM está disponible como área de trabajo *estática* del controlador de dispositivo, y contiene información,como el número de canales abiertos a ese controlador, la dirección de estos canales (o un apuntador a una lista encadenada de canales-si el número puede ser infinito), el estado actual del dispositivo de E/S etc. Este área es *propiedad* del controlador de dipositivo, y por lo tanto del sistema operativo del que forma parte.

Es posible abrir un mismo dispositivo por varios canales a la vez. En este caso, el controlador del dispositivo debe ser re-entrante. Lógicamente, si el controlador del dispositivo intenta mantener el estado de cada canal por sí mismo, se hará un verdadero lío. Por esta razón, cada canal asigna un *bloque de definición de canal*, que se crea al abrirlo con IO.OPEN (TRAP#2 D0=1), y que es propiedad del Job que lo ha creado.

Ahora se tratarán más en profundidad los conceptos básicos que acabamos de ver.

9.2 Bloques de definición de controlador de dispositivo

Los controladores de dispositivo se suelen conectar al sistema operativo cuando se enciende la máquina. Disponemos de una serie de TRAPs del gestor (TRAP#1 con D0=1A al 23), para poder hacerlo. No es imprescindible conectar los controladores de dispositivo al sistema operativo a tiempo de encendido - se pueden añadir después si se requieren. El gestor mantiene varias listas encadenadas, para distintas operaciones, que siguen el formato estándar que se vió en detalle en la sección 5.4.5.

Existen cinco listas encadenadas. Tres de ellas son para el mediador físico que se invoca por:

1. interrupciones externas
2. interrupciones de los 50/60Hz
3. bucle del planificador

Como se describió en la sección 5.4.5, una lista encadenada de produce reservando dos palabras largas. La primera forma la conexión a la lista, la segunda debe ser la dirección del código del controlador del dispositivo físico. Estas dos palabras largas se encuentran normalmente al principio de un bloque de memoria. El resto de los bloques están disponibles para que las use por el controlador del dispositivo. Alguno de estos usos son, almacenamiento de indicdores, apuntadores, memoria intermedia, etc.

El sistema operativo pasa, del dispositivo físico al controlador, varias piezas de información útiles. Estas van en los registros del 68008 como sigue:

D3 número de interrupciones de 50/60Hz desde la última llamada del palnificador (solo bucle planificador).
A3 apuntador bloque de definición del controlador del dispositivo.
A6 apuntador a las variables del sistema.
A7 pila del supervisor (se pueden usar 64 octetos).

A6 y A7 deben ser preservadas por el controlador de dispositivo.

Para las llamadas del mediador de acceso, A3, A6 y A7 tienen el mismo significado, pero los otros registros son diferentes (ver 9.4).

A3 apunta a la dirección base del bloque de definición. Este se refiere a una definición de un controlador de dispositivo estándar (ver debajo). Sin embargo, es posible tener una definición no estándar diferente (por ejemplo, algunas direcciónes de entrada y apuntadores de conexión se pueden eliminar). Si se usa una definición no estándar, A3 no tendrá un valor adecuado.

$00(A3) conecta la siguiente rutina de int. externa
$04(A3) dirección de la rutina de int. externa
$08(A3) conecta la siguiente rutina de int. 50/60Hz
$0C(A3) dirección de la siguiente rutina de int. 50/60Hz
$10(A3) conecta la siguiente rutina de bucle del planif.
$14(A3) dirección de la rutina de bucle del planif.
$18(A3) conecta el mediador de acceso del siguiente controlador de dispositivo
$1C(A3) dirección de la rutina de entrada/salida
$20(A3) dirección de la rutina de abrir canal
$24(A3) dirección de la rutina de cerrar canal
$28(A3) espacio de trabajo del controlador de dispositivo físico

9.3 Mediador de acceso

Para incializar el mediador de acceso, se debe conectar un bloque de memoria en el IOSS, mediante MT.LIOD (TRAP#1 con D0=$20). Este bloque de memoria debe estar en el área de procedimientos residentes o en el área común. Debe tener cuatro palabras de longitud, desde el principio del bloque. El gestor usa la primera, para formar la lista encadenada, la segunda, tercera y cuarta palabras largas son las direcciones de entrada para entrada/salida, abrir y cerrar el canal. El resto del bloque se puede usar para almacenar apuntadores, memoria intermedia, etc.

El formato del bloque es:

palabra larga 1	apuntador de conexión a la siguiente rutina encadenada
palabra larga 2	dirección de entrada para entrada/salida
palabra larga 3	dirección de entrada para abrir un canal
palabra larga 4	dirección de entrada para cerrar un canal

Todas las llamadas del mediador de acceso se hacen en modo supervisor.

9.3.1 Abrir un canal

Cuando se hace una llamada para abrir un canal, A0 contiene un apuntador al nombre del dispositivo y D3 contiene una clave. En A3 está la base del bloque de definición del controlador del dispositivo. A6 apunta al área de variables del sistema y A7 contiene el apuntador a la pila del supervisor. Los registros D0 a D7 y A1 a A6 se pueden tratar como volátiles.

La secuencia a seguir para una operación de OPEN es:

Decodificar el nombre del dispositivo
Asignar un bloque de definición de canal y memoria intermedia en el área común
Inicializar el bloque de definición de canal (excepto las 6 primeras palabras largas)
Devolver la dirección del bloque de definición del canal en el registro A0.

Todos los canales son bidireccionales. Es responsabilidad del controlador del dispositivo, el detectar las operaciones ilegales. Cuando se hace OPEN a un canal, pueden ocurrir cuatro tipos comunes de error, que se pasan en el registro D0. **ERR.NF** significa que el nombre del dispositivo no ha

sido reconocido. **ERR.IU** significa que el dispositivo de E/S está siendo usado por otro Job y no se puede compartir. **ERR.BN** significa que se ha reconocido el nombre del dispositivo pero es incorrecta parte de la información que se ha pasado. **ERR.OM** significa que todo es correcto, pero el bloque de definición de canal o la memoria intermedia no se pueden asignar, debido a falta da memoria.

9.3.2 Entrada y salida de un canal

La llamada de entrada/salida se realiza primero durante el TRAP de E/S. Si el TRAP ha sido especificado como de *esperar hasta completarlo*, la llamada se hace una vez cada bucle del planificador. Esta operación continúa hasta que se ha completado, o hasta que se cumpla el tiempo de espera.

A0 debe apuntar al bloque de definición de canal, antes de hacer la llamada de E/S. D0 define la operación requerida (es una clave de un octeto y el IOSS limpia los tres octetos superiores). En D1, D2, A1 y A2 se pasa información adicional. El registro D1 lo pone el IOSS a ceros para las operaciones de cadena (IO.FSTRG, IO.FLINE, IO.SSTRG, FS.HEADS, FS.HEADR, FS.LOAD y FS.SAVE). Si una operación no se completa, se deben poner a la salida los valores de D1 y A1 de modo que se puedan usar en los siguientes reintentos. Para ayudar en este proceso, se pone D3 a ceros en la primera entrada, pero se va modificando a -1 en la segunda y entrada y siguientes. D0, D2 y A2 mantienen sus valores originales a través del proceso.

D0 define siempre la operación, A3 apunta a la base del bloque de definición de controlador de dispoditivo, A6 apunta al área de variables del sistema y A7 es el apuntador a la pila del supervisor.

Dentro del controlador de dispositivo, se pueden modificar los registros D2 a D7 y A2 a A6.

9.3.3 Cerrar un canal

Una operación de CLOSE de un canal comprueba que todo está limpio y después libera el espacio que fué asignado para el bloque de definición del canal y la memoria intermedia. Si quedan todavía algunos octetos que trasmitir, no es posible liberar el espacio inmediatamente. En estos casos, el bloque será borrado en una llamada posterior del planificador, cuando los datos hayan sido transmitidos.

A la rutina de CLOSE se le pasa la dirección del bloque de definición de canal, en el registro A0. La base del bloque de definición de controlador de dispositivo se pasa en el registro A3. A6 apunta al área de variables del sistema y A7 es el apuntador a la pila del supervisor. Los registros D1 a D3 y A0 a A3 pueden ser modificados. En D0 se debe pasar el código de error. Normalmente este debe ser cero (ej. sin errores) ya que la mayor parte del sistema asume que el CLOSE nunca falla.

9.4 Mediador físico

Como ya mencionamos en la sección 9.1, el mediador físico tiene como función el interconectar el controlador del dispositivo y el 'hardware' externo. Esta interconexión se puede realizar en cualquiera de los siguientes eventos:

1. Interrupciones externas

Las interrupciones externas son generadas por dispositivos externos particulares. Si un controlador de dispositivo espera interrupciones externas mientras se está ejecutando, primero debe comprobar el estado del 'hardware' para ver si la interrupción la ha mandado él. Si la interrupción no viene del dispositivo correcto, se debe hacer un retorno mediante la instrucción RTS. Si la interrupción es para él, se debe servir al 'hardware' y limpiar la fuente de la interrupción. Se debe retornar a la fuente de la interrupción mediante una instrucción RTS. Las operaciones atómicas (como las llamadas del mediador de acceso al mismo controlador) pueden ser interrumpidas por esta interrupción.

2. Interrupciones de 50/60Hz

Esta interrupción solo debe ser usada para operaciones de temporización crítica, u operaciones que no se deban mantener por periodos largos. Un ejemplo de operación de temporización crítica puede ser el incremento de un registro para controlar el tiempo , o la generación de envolventes de sonido. Si el 'hardware' externo no puede interrumpir al sistema cuando necesita servicio, se puede usar la interrupción de 50/60Hz para comprobar si un dispositivo necesita servicio. Las operaciones atómicas (como las llamadas del mediador de acceso al mismo controlador) pueden ser interrumpidas por esta interrupción.

3. Bucle del planificador

Las llamades del bucle del planificador llegan a intervalos irregulares (dependiendo de la carga de trabajo del sistema), por lo que no pueden usar este bucle las operaciones que que deban ser procesadas inmediatamente. Las operaciones atómicas no pueden ser interrumpidas por el planificador, por lo que las operaciones de asignación y liberación de memoria se pueden realizar, para mayor seguridad, con rutinas que se ejecuten fuera del planificador.

Notas generales sobre el mediador físico.

El mediador físico permite trasferir datos dentro o fuera de las colas o ejecutar funciones de control asíncronas con un Job que requiera E/S. Todas las rutinas del mediador físico son llamadas en modo supervisor, de forma que todo está en condiciones para que puedan inhabilitar las interrupciones. Cualquier error debe ser tratado mediante un indicador que entienda el mediador de acceso. Sin embargo, es posible usar las rutinas de utilidad UT.ERR y UT.MTEXT con A0=0. Después se hará un intento de escribir un mensaje al canal 0, o, si esto falla, al canal 1. El mensaje aparecerá en uno de estos canales si no están esperando por entrada. Si ambos están esperando por entrada, no aparecerá ningún mensaje. Las rutinas de manejo de colas han sido diseñadas para que puedan trasferir datos desde/hacia una cola asincronamente.

9.5 Decodificación del nombre de dispositivo

La rutina de utilidad IO.NAME (vector $122) es muy útil para ayudarnos en la decodificación de nombres de dispositivos. Realiza dos operaciones, comprobar el nombre del dispositivo y evaluar cualquier parámetro opcional.

El nombre completo del dispositivo contiene cuatro componentes:

1. Nombre — Caracteres ASCII, normalmente letras independientemente de si son mayúsculas o minúsculas.

2. Separador — Caracteres ASCII, si es una letra no importa si es mayúscula o minúscula.

3. Número — Número decimal en el rango 0 a 2^15-1

4. Código — Uno de una lista de caracteres ASCII

Cuando se llama a IO.NAME, A0 debe contener un apuntador al nombre real del dispositivo. A3 debe contener un apuntador a un bloque de memoria suficientemente grande para almacenar los valores de los parámetros. Si la rutina se ejecuta satisfactoriamente, el bloque se llenará con los parámetros dados o los asignados or defecto.

IO.NAME tiene tres posibles retornos:

retorno	D0(+SR)
estándar	ERR.NF nombre no reconocido
estándar+2	ERR.BN nombre reconocido, parámetro malo
estándar+4	0 retorno satisfactorio

La descripción del nombre de dispositivo comienza 6 octetos después de la llamada.

9.5.1 Descripción del nombre del dispositivo

La descripción está en la siguiente forma:

Número de caracteres en el nombre, caracteres del nombre
Número de parámetros
Por cada parámetro uno de:

Espacio + separadores, valor por defecto (numérico)
Número negativo, valor por defecto (número, no separ)
Número de códigos, lista de códigos ASCII.

Todos los items se definen como **palabras** y todas las letras deben ir en mayúsculas. Ya que los códigos ASCII de los caracteres caben realmente en octetos, cualquier octeto impar al final, se rellena con cero. Por ejemplo:

```
DC.W  'ABC'
```

aparece en memoria así:

palabra 1	$00 03
palabra 2	$41 42
palabra 3	$43 00

Por cada parámetro numérico de la descripción, la rutina devolverá el valor dado o el asignado por defecto. Por cada lista de códigos en la descripción, la rutina devolverá la posición del código en la lista o cero.

Ejemplos

La descripción CON es:

```
DC.W  3,'CON'           consola
DC.W  5                 cinco parámetros
DC.W  '_',448,'X',200   tamaño ventana
DC.W  'A',32,'X',16     posición ventana
DC.W  '_',128           longitud memoria intermedia teclado
```

Nombre de dispositivo	Parámetros devueltos
CON	448,200,32,16,128
CON_256	**256**,200,32,16,128
con_60	448,200,32,16,**60**
cona0x12	440,200,**0**,**12**,128
con_256x64a64x128_20	**226**,**64**,**64**,**128**,**20**

La descripción de SER es:

```
DC.W  3,'SER'           Dispositivo serie RS232
DC.W  3                 tres parámetros
DC.W  -1,1              número de puerta (1 por defecto)
DC.W  4,'EOMS'          paridad impar,par,marca o espacio
DC.W  2,'1H'            ignorar/usar 'handshaking'
```

Nombre de dispositivo	Parámetros devueltos
SER	1,0,0
sere	1,1,0
ser2mi	**2**,**3**,**1**

9.6 Asignación de memoria

Es esencial que los controladores del mediador de acceso sean totalmente reentrantes. Esto significa que la información específica del canal debe almacenarse en el bloque de definición de canal y no en el bloque de definición del controlador del dispositivo. Los bloques de definición de canal requieren dejar 6 palabras largas al principio para que las use el IOSS. El resto del espacio en el bloque lo puede usar el controlador de dispositivo.

El controlador del mediador físico nunca debe asignar ni liberar recursos de la máquina, a menos que lo haga en una llamada al planificador.

El espacio se debe asignar siempre en el área común. Se puede usar para ello la rutina MM.ALCHP. Este espacio se puede liberar con la rutina MM.RECHP.

9.7 Colas de E/S

Hay cinco rutinas residentes para el manejo de colas. Estas controlan la entrada y salida de datos de la cola, y comprueban el contenido de los bloques de definición, y las condiciones de fin de fichero. IO.QSET pone los apuntadores de las colas. IO.QIN pone un octeto en una cola. IO.QOUT saca un octeto de una cola. IO.QEOF pone el indicador de fin de fichero IO.TEST comprueba si hay algo en la cola, devuelve el espacio libre y el valor del siguiente octeto que va a sacar sin quitarlo de la cola.

Las colas se definen por bloques de 4 palabras largas al principio. El octeto más significativo de la primera palabra larga lo usan las rutinas de las colas para identificar el fin del fichero. IO.QSET limpia la palabra larga comple- ta. el resto (ej. sin bit más significativo) se puede usar para el contrlador de dispositivos para conectar colas, etc.

La longitud usable de una cola es un octeto menos de la longitud real. El mínimo espacio ocupado por una cola y su cabecera es por lo tanto de 18 octetos.

9.8 E/S serie simple

La E/S serie simple se puede manejar con IO.SERQ. Para esta rutina, la séptima y octava palabra larga en el bloque de definición de canal deben apuntar a las colas de entrada y salida. Si una de ellas está prohibida, el apuntador correspondiente se pondrá a cero. Nos devolverá el error ERR.BP para las acciones indefinidas. Ver la descripción de IO.SERQ en el capítulo 8 para más detalles.

En los casos en que las operaciones de entrada y salida no sean tan simples, se puede llamar a la rutina IO.SERIO. A la instrucción de llamada le deben seguir tres palabras largas, que son las direcciones de entrada para:

1. Comprobación de entrada pendiente(D1=siguiente octeto)
2. Tomar un octeto (octetto en D1)
3. mandar un octeto (octeto en D1)

9.9 Controladores de dispositivo de directorio

Los controladores de dispositivos de directorio son más potentes en sus aplicaciones que los controladores de dispositivo puros. Los normales son responsables de las comunicaciones con otro dispositivo. Los de dispositivo de directorio son responsables de las comunicaciones con ficheros particulares dentro del dispositivo, como un disco duro winchester. El IOSS tiene un rango de operaciones extendido para estos controladores.

9.9.1 Bloque de definición de controlador de directorio

Los bloques de definición para controladores de dispositivo de directorio son similares a los de los controladores de dispositivos. Sin embargo, hay tres partes; el bloque de conexión del controlador de directorio (uno por controlador de directorio), el bloque de definición física (uno por controlador) y el bloque de definición de canal. El IOSS asigna los dos últimos.

El bloque de conexión

La forma estándar es una versión extendida de la que usan los controladores de dispositivo. La conexión del mediador de acceso es por lo menos de 10 palabras de longitud, y tiene el siguiente formato estándar:

```
$00(A3) conecta la siguiente rutina de int. externa
$04(A3) dirección de la rutina de int. externa
$08(A3) conecta la siguiente rutina de int. 50/60Hz
$0C(A3) dirección de la rutina de int. 50/60Hz
$10(A3) conecta la siguente rutina de bucle del planifi.
$14(A3) dirección de la rutina de bucle de planificador
$18(A3) conecta el mediador de acceso del siguiente
        controlador de directorio
$1C(A3) dirección la rutina de entrada/salida
$20(A3) dirección de la rutina de OPEN
$24(A3) dirección de la rutina de CLOSE
$28(A3) dirección de entrada de dependencia forzada
$2C(A3) reservada
$30(A3) reservada
$34(A3) dirección de entrada de formateo de medio
$38(A3) longitud del bloque de definición física
$3C(A3) palabra - longitud del nombre del controlador
$3E(A3) caracteres del nombre del controlador (ej. mdv)
```

9.9.2 Mediador de acceso de controlador de directorio

Como en los controladores de dispositivo, el mediador de acceso debe abrir, cerrar y sacar y recibir datos de los ficheros. Las rutinas de CLOSE y entrada/salida son las mismas que las de los controladores de dispositivo. Refierase a las secciones 9.3.2 y 9.3.3 para más detalles. Sin embargo, abrir canales en un controlador de dispositivo de directorio es un poco más complejo.

Abriendo un canal

Antes de llamar a la rutina de abrir un canal, se deben asignar el bloque de definición de canal y el bloque de definición física. En este punto, el IOSS debe haber llenado parcialmente estos bloques. A la entrada de la rutina de OPEN, A0 apuntará al bloque de definición de canal y A1 al bloque de definición física. A3 apuntará a la base del bloque de conexión. El bloque de definición física puede haber sido asignado ya. Los ficheros se componen de bloques de 512 octetos de longitud cada uno. La posición real de un octeto es por lo tanto 512*número de bloque+número de octeto

Bloque de definición de canal

Cabecera de definición de canal para todos los canales

$00	CH.LEN	4	longitud del bloque
$04	CH.DRIVR	4	dirección del controlador
$08	CH.OWNER	4	Job propietario
$0C	CH.RFLAG	4	dirección cuando se libera el espacio
$10	CH.TAG	2	etiqueta de canal
$12	CH.STAT	1	estado:
			0 = OK
			-1 = A1 absoluto
			$80 = A1 relativo a A6
			negativo = esperando
$13	CH.ACTN	1	acción almacenada para el Job en espera
$14	CH.JOBWT	4	ID del Job esperando por ES

Bloque de definición de canal del sistema de ficheros

$18	FS.NEXT*	4	conecta al siguiente canal de fichero
$1C	FS.ACCES*	1	modo de acceso (D3 en llamada OPEN)
$1D	FS.DRIVE*	1	ID de la unidad
$1E	FS.FILNR+	2	# de fichero en la unidad
$20	FS.NBLOK	2	# de bloque que contiene siguiente octeto
$22	FS.NBYTE+	2	siguiente octeto en el bloque
$24	FS.EBLOK+	2	# bloque que contiene el octeto después del EOF
$26	FS.EBYTE+	2	octeto después del EOF
$28	FS.CBLOK	4	apuntador a la tabla de bloques dependiente para el bloque dependiente actual que puede contener el octeto actual/siguiente
$2C	FS.UPDT		puesto si se ha actualizado el fichero
$32	FS.NAME*	2+36	nombre del fichero
$58	FS.SPARE	72	libres

Bloque de definición física

$10	FS.DRIVR*	4	apuntador a la conexión del mediador de acceso para el controlador
$14	FS.DRIVN*	1	número de unidad
$15		1	reservado
$16	FS.MNAME	2+10	nombre del medio
$22	FS.SPARE	1	número de ficheros abiertos

La longitud viene determinada por la conexión del mediador de acceso.

Claves

* Inicialmente los bloques están llenos de ceros excepto los marcados con * que los llena el IOSS.

+ La rutina de OPEN no es llamada para abrir un fichero compartido a más de un canal. En este caso, FS.NBYTE se pone a $40 y la información fija se copia desde el primer canal abierto al fichero.

El ID de la unidad es un índice a la tabla de bloques de definición física, empezando en $100 desde la base de las variables del sistema.

El borrado de ficheros es parte de la operación de apertura de ficheros. Los dos procesos se distinguen por la clave de acceso que es negativa para el borrado. Los bloques de definición se asignan de la forma usual para apertura de ficheros, pero el IOSS libera el bloque de definición del

canal a su retorno. El bloque de definición de canal es liberado también por el IOSS si devuelve algún código de error en el rgistro D0.

9.10 Bloques dependientes

Toda la interacción con los dispositivos de almacenamiento se lleva a cabo a través de la memoria del QL, usándola como memoria intermedia. Este proceso pasa totalmente inadvertido al usuario. Si se accede a un fichero en microdrive, algunos sectores se leerán en memoria. Si se reescribe un sector al microdrive, no es necesario que se haga justo después del salvado, los datos pueden permanecer en memoria algún tiempo hasta que se vuelcan al dispositivo. A medida que el espacio usado por los programas transitorios, procedimientos residentes y el área común, aumenta, el área de bloques dependientes disminuye de tamaño.

El espacio en los bloques dependientes está definido en la tabla de bloques dependientes. Esta tabla es de tamaño variable, y depende de la cantidad de memoria disponible. Hay tres apuntadores a la tabla de bloques dependientes.

$54(A6)	SV.BTPNT	4	apuntador a la entrada asignada más recientemente en la tabla
$58(A6)	SV.BTBAS	4	apuntador a la base de la tabla
$5C(A6)	SV.BTTOP	4	apuntador a lo alto de la tabla

Cada entrada de la tabla tiene 8 octetos de longitud. El sistema operativo usa solamente el primer octeto como octeto de estado. Los otros octetos no son usados por el sistema operativo.

$00 BT.STAT oct. octeto de estado

00	imposible llenar el sistema
01	bloque vacío
X3	bloque es representación del fichero
X7	bloque modificado (esperando escribir)
X9	bloque esperando lectura
XB	bloque esperando verificación
X	el ID de la unidad del fichero

$01	BT.PRIOR	1	disponible para algoritmos dependientes
$02	BT.SECTR	2	sector físico en el medio
$04	BT.FILNR	2	número de fichero
$06	BT.BLOCK	2	número del bloque contenido en este bloque dependiente

Ya que la asignación de bloques dependientes, es una forma muy simple de asignar memoria, solo se debe realizar con llamadas al mediador de acceso o al bucle del planificador.

Será necesario buscar un bloque dependiente para usarlo como memoria intermedia de un bloque de fichero. Solamente se pueden usar los bloques vacíos o las representaciones auténticas. Cuando se asigna un bloque nuevo a un fichero, SV.BTPNT debe ser modificado.

No asuma nunca que los bloques que fueron asignados en una llamada están asignados aún en la siguiente, porque pueden no estarlo!. FS.CBLOK es una variable de aviso solamente. Compriueba **siempre** la entrada de la tabla de bloques dependientes, antes de proceder.

Algunas veces el gestor de memoria requiere el espacio asignado a un bloque actualizado o esperando por lectura o verificación. Si ocurre esto, la rutina dependiente de forzado es llamada para convertirla en una copia auténtica. La rutina no comprueba si el bloque es una copia auténtica, pero inicia el proceso que lo hace. A la entrada de la rutina dependiente de forzado, A1 apunta al bloque dependiente, A2 apunta al de definición física y A3 a la base del bloque de conexión (definición de controlador). Los registros D0 a D3 y A0 a A4 pueden ser alterados si se desea.

9.11 Un ejemplo con un controlador de impresora paralela

9.11.1 Introducción

Esta sección ilustra muchos de los puntos que se han explicado en las partes anteriores de este capítulo. Un ejemplo completo de un controlador de dispositivo para construir un interface Centronics estándar para impresora. El diagrama completo del circuito está en el Apéndice U.

Este controlador de dispositivo ha sido diseñado para contenerlo en una ROM en una tarjeta periférica. Esta tarjeta (con una ROM) se conecta al QL. En el momento de encender el ordenador, aparecerá en la pantalla el mensaje 'Interface Centronics Simple'. Desde este momento, existirá un nuevo canal llamado 'PAR'. Para listar un fichero de microdrive en la impreosra, solo debe teclear:

```
copy mdv1_textfile to par
```

Ya que es un simple ejemplo de controlador de dispositivo, no se ha previsto una memoria intermedia de impresora, por lo que el controlador intentará sacar un octeto y devolverá un 'ERR.NC' si no se puede mandar (debido a que la impresora está ocupada). Un controlador más sofisticado puede operar una memoria intermedia y mandar interrupciones generadas por el circuito de interface.

9.11.2 El programa controlador de dispositivo

```
*
* Interface Paralelo Centronics (sin interrupciones)
*
ERR.NC    EQU     -1
ERR.BP    EQU     -15
MT.ALCHP  EQU     $18
MT.LIOD   EQU     $20
IO.SSTRG  EQU     $7
MM.ALCHP  EQU     $C0
MM.RECHP  EQU     $C2
IO.SERIO  EQU     $EA
IO.NAME   EQU     $122
*
DATA      EQU     $0800   dirección separador de datos
STROBE    EQU     $1000   dirección 'strobe'
BUSY      EQU     $1800   dirección memoria de BUSY
*
BASE
          DC.L    $4AFB0001
          DC.W    0
          DC.W    INIT-BASE
          DC.B    0,28,'Interface Centronics Simple',$A
*
INIT
          MOVEM.L A0/A3,-(A7)     A0 & A3 usado para inic. cód.
          MOVEQ   #MT.ALCHP,D0    asignación bloque conexión
          MOVEQ   #$36,D1         $28+3 palabras largas+1 pal.l
          MOVEQ   #0,D2           propiedad del Job 0
          TRAP    #1
          TST.L   D0              ver si OK
          BNE.S   INIT_EXIT
*
          LEA     $1C(A0),A3      empieza a llenar bloque
          LEA     IO(PC),A2       entrada/salida
          MOVE.L  A2,(A3)+        ... en $1C
          LEA     OPEN(PC),A2     abrir
          MOVE.L  A2,(A3)+        ... en $20
          LEA     COLSE(PC),A2    cerrar
          MOVE.L  A2,(A3)+        ... en $24
          LEA     ERR_BP(PC),A2   comprov. pend. + tomar octet.
```

```
        MOVE.L  A2,(A3)+        ... en $28
        MOVE.L  A2,(A3)+        ... y en $2C
        LEA     OUT_BYTE(PC),A2 manda un octeto
        MOVE.L  A2,(A3)+        ... en $30
        MOVE.W  #$4E75,(A3)+    RTS en $34
*
        LEA     $18(A0),A0      conectar en
        MOVEQ   #MT.LIOD,D0     lista controlador ES
        TRAP    #1
INIT_EXIT
        MOVEM.L (A7)+,A0/A3
        RTS
*
*
OPEN
        MOVE.W  IO.NAME,A4      decodifica nombre dispositi.
        JSR     (A4)
        BRA.S   EXIT            mal
        BRA.S   EXIT            mal
        BRA.S   ALCHP           nombre es 'PAR'
        DC.W    3,'PAR'         definición  nombre PAR
        DC.W    0                           sin parámetros
ALCHP
        MOVEQ   #$18,D1         reserva el canal mínimo
        MOVE.W  MM.ALCHP,A4
        JMP     (A4)
CLOSE
        MOVE.W  MM.RECHP,A4     elimina el canal
        JMP     (A4)
IO
        CMP.B   #IO.SSTRG,D0    trap operación fichero (ej.
        BHI.S   ERR_BP          ... poner cabecera), porque
*                               ... no tiene sentido en una
*                               ... impresora
        PEA     $28(A3)         debemos tener una llamada
        MOVE.W  IO.SERIO,A4     ... en $24(A3) a SERIO
        JMP     (A4)
*
*
OUT_BYTE
        LEA     BASE(PC),A3     toma direc. base periférico
        TST.B   BUSY(A3)        impresora BUSY?
        BMI.S   ERR_NC          señalar ocupada
        MOVE.B  D1,DATA(A3)     mandar octeto
        SF      STROBE(A3)      y STROBE (>500ns)
        MOVEQ   #0,D0
EXIT
        RTS
ERR_NC
        MOVEQ   #ERR.NC,D0      octeto mandado
```

```
        RTS
ERR_BP
        MOVEQ   #ERR.BP,D0      no puede conseguir octeto y
        RTS                     ... no puede almacenar oper.
```

9.11.3 Funcionamiento del programa

Para entender cómo funciona el controlador de dispositivo, es importante comprender cómo aparece el dispositivo físico en el mapa de memoria del QL. La puerta de salida de 8 bits a la impresora está situada en la posición $0800, desde la dirección base del periférico. la salida de 'strobe' está en el desplazamiento $1000, y la entrada 'BUSY' en el $1800.

Para los que no están familiarizados con el mecanismo de decodificación de las direcciones de periféricoa (Apéndice F),aquí le damos una pequeña explicación. Se pueden conectar hasta 16 tarjetas periféricas en el QL. A cada una de estas se le asignan 16K octetos de memoria en la parte superior del mapa de memroria. La dirección base particular de cada tarjeta viene determinada por su posición. Por lo tanto, todas las direcciones de periféricos vienen especificadas en relacción con la dirección base de la tarjeta. Cuando se enciende el QL, se comprueba si existe la identificación $4AFB0001 al principio de la dirección de cada periférico. Si se encuentra el código, asume que hay una ROM presente, conteniendo el controlador del dispositivo, funciones y/o procedimientos.

Para indicar al QDOS que la ROM contiene un controlador de dispositivo, debe tener unos datos de reconocimiento al principio (ver Apéndice C). Los datos que aparecen desde la etiqueta 'BASE' en adelante se deben colocar en los primeros octetos. La primera palabra larga identifica que es una ROM y no 'espacio libre'. La palabra que sigue indica que no hay procedimientos o funciones en esta ROM. La siguiente palabra apunta a la rutina de iniciación de la ROM, que debe ser ejecutada una vez, a tiempo de encendido. Finalmente, la longitud del nombre del controlador del dispositivo, suponiendo que lo tenga.

La rutina de iniciación 'INIT' asigna espacio en el área común para los bloques de conexión, rellena las direcciones y los datos importantes y encadena el bloque en la lista de controladores de E/S. A0 y A3 se salvan a la entrada de INIT, y son restaurados a la salida, ya que se modifican en la rutina.

Como en los controladores de dispositivo estándar, se proporcionan las tres rutinas de OPEN, CLOSE y entrada/salida.

OPEN

Abrir un canal consta de dos operaciones principales. Lo primero que hay que hacer es decodificar el nombre del canal. En este caso se ha usado IO.NAME para simplificarlo. Después, una vez que se ha reconocido el nombre, se asignan $18 octetos en el área común. Será necesario asignar más para ciertas aplicaciones, pero aquí usamos el mínimo. El canal está ahora abierto.

CLOSE

Cerrar un canal es muy sencillo. La operación consiste en eliminar el espacio asignado en el área común. En este caso se ha usado MM.ALCHP para hacerlo.

Entrada/Salida

Esta rutina usa las direcciones almacenadas desde $28(A3) en adelante, que apuntan a las rutinas de comprobación de entrada pendiente, recoger un octeto y mandar un octeto, respectivamente. La rutina de manejo general de cadenas, IO.SERIO, se usa para manejar la ES. La mayoría de las operaciones de E/S son válidas, excepto IO.SSTRG (ésta trata las operaciones con ficheros, pero no es importante para la impresora).

En nuestro controlador, las rutinas de entrada pendiente y de recoger un octeto devuelven el error 'Parámetros inválidos'. Solo es válida la rutina de mandar un octeto a la impresora (OUT_BYTE).

OUT_BYTE comprueba, primero, si la impresora está ocupada (BUSY); si lo está, devuelve un error ERR.NC. Si la impresora no está ocupada, se escribe un octeto de datos. Después lo manda a la impresora y pone el STROBE durante >500ns. Des- pués de esta secuencia de operaciones, los datos han sido transferidos a la impresora.

10 SuperBASIC

10.1 Introducción

Este capítulo es ligermente diferente del resto del libro, ya que está dedicado exclusivamente al SuperBASIC. Puede haber considerables ventajas al usar el BASIC, ya que toda la estructura del lenguaje, el editor, etc. están ya ahí. Por lo tanto, crear funciones especializadas de BASIC, así como procedimientos para aplicaciones particulares, puede ser una tarea muy interesante.

Antes de comenzar a ver cómo añadir utilidades al SuperBASIC, es necesario tener una idea general de las estructuras para almacenar variables y otros nombres.

Cuando se introduce desde el teclado, un nombre dentro del programa BASIC, se crea una entrada para él en la memoria. Todo lo que se almacena en el propio programa es un apuntador a la entrada corespondiente. Este método de asignación de variables es muy eficiente en el manejo de espacio cuando se usa una variable particular una y otra vez. De esta forma no tenemos una carga extra de alamcenamiento para nombre largos de variables, que hace a los programas más fáciles de entender.

Hay cuatro áreas distintas en las que se almacena la información. Son, la *tabla de nombres*, la *lista de nombres*, la *lista de valores variables* y la *pila aritmética*. A diferencia de los ordenadores menos sofisticados, toda el área de trabajo es susceptible de moverse sin peligro. Por esta razón, todas las referencias en BASIC se realizan relativas al registro A6 que apunta siempre a la base del área.

10.2 La tabla de nombres

El SuperBASIC maneja las variables y las cadenas en modo indirecto. Como se mencionó en la sección 10.1, cada referencia a una variable dentro del fichero de programas BASIC, es un simple apuntador a una entrada en la *tabla de nombres*. Esta tabla no contiene en sí misma los valores de las variables, sino un apuntador al nombre en la *lista de nombres* y un valor en el *área de valores de variables*. Cada entrada en la tabla de nombres es de ocho octetos de longitud y tiene el siguiente formato:

palabra 1 código describiendo el uso del nombre
palabra 2 apuntador al nombre en la lista de nombres
palabra 3 apuntador al valor

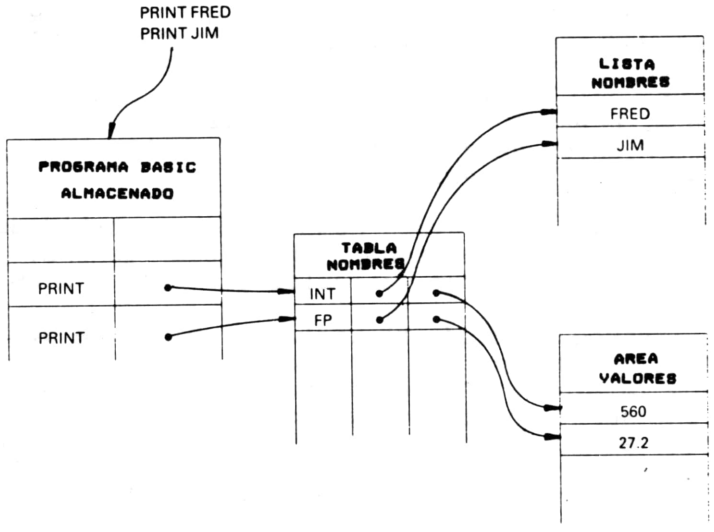


Figura 10.1 - Almacenamiento de variables en BASIC

Código de tipo

Es la primera palabra del bloque en la tabla de nombres. Hay varios tipos diferentes:.

Variable	Código
cadena no usada	0001
número de coma flotante no usado	0002
entero no usado	0003
variable de cadena	0201
número de coma flotante	0202
entero	0203
índice de bucle REPEAT	0602
índice de bucle FOR	0702

Matrices	Código
matriz de subcadena	0300 (interno solamente)
matriz de cadena	0301
matriz de coma flotante	0302
matriz de enteros	0303

Los procedimientos y las funciones no tienen tipo. Por otro lado, las funciones BASIC tienen un tipo que se define por el último carácter del nombre de la función ($ o % o ninguno).

Tipo de módulo	Código
procedimiento BASIC	0400
función de cadena BASIC	0501
función de coma flotante BASIC	0502
función entera BASIC	0503
procedimiento en código máquina	0800
función en código máquina	0900

También hay algunos códigos que se refierena valores en la pila aritmética, dentro del evaluador de expresiones.

Expresiones	Código
cadena	0101
coma flotante	0102
entero	0103

El apuntador de nombre

Es la segunda palabra de la entrada en la tabla de nombres. Contiene el desplazamiento del nombre dentro de la lista, a menos que la entrada sea el valor de una expresión, en cuyo caso contiene -1. Alternativamente, si la entrada es una copia de otra, se incluye un apuntador a esa entrada.

Apuntador al valor

Es una palabra larga y es el tercer argumento en la tabla de nombres. Si es negativo, el valor es indefinido. Si es positivo, es el desplazamiento desde la base del área VV (valor de variable) en el caso de variables, o el desplazamiento del descriptor de matriz en caso de matrices. La palabra más significativa es el número de línea del DEF PROC o del DEF FN de un procedimiento o función del BASIC. El apuntador es la dirección absoluta de un procedimiento o una función en código máquina. El apuntador al valor no se define para las entradas de nombre internas que se refieren

a un valor en la pila aritmética.

10.3 La lista de nombres

Los nombres de variables y matrices se almacenan en una lista de nombres. Cada nombre se almacena como un octeto seguido por los caracteres ASCII del nombre. Las entradas en la lista son apuntadas por la tabla de nombres descrita en la sección 10.2.

10.4 El área de valores de variables

Esta es el área donde se almacenan los valores reales de las variables. El área está construida como una pila de entradas, asignadas en múltiplos de 8 octetos.

Las variables de cadena se almacenan como una palabra (que contiene el número de caracteres de la cadena) seguida por los caracteres de la cadena. El espacio ocupado por una cadena se redondea al número par más cercano, de forma que termine en frontera de palabra.

Los números de coma flotante se almacenan con dos octetos de exponente y cuatro de mantisa. Ver la sección 8.5.4 para más detalles sobre el almacenamiento de números de coma flotante.

Los enteros se almacenan como una palabra.

Almacenamiento de matrices

El descriptor de matriz, que es apuntado desde la tabla de nombres, tiene el siguiente formato (ver figura 10.2); comienza con una palabra larga que contiene el desplazamiento de los valores de la matriz desde la base del área de valores de variables. Va seguida por una palabra que contiene el número de dimensiones. Después de cada dimensión va un par de palabras; la primera es el índice máximo, y la segunda es el índice multiplicador para esa dimensión.

Las matrices de coma flotante y de enteros se almacenan en una forma regular. La primera toma hasta 6 octetos por elemento y la de entero ocupa 2 octetos por elemento.

Las matrices de cadena no son tan regulares, debido a la longitud variable de sus entradas. Estas matrices se almacenan como una matriz de caracteres. El elemento cero de la última dimensiónm es una palabra que contiene la longitud máxima de la cadena. Esta dimensión se redondea siempre al valor par más cercano (de forma que cada cadena empiece en

frontera de palabra).

Las operaciones de troceado de cadenas producen subcadenas, que son matrices normales de caracteres. La base a que se refiere el índice es distinta de cero. Se debe tener en cuenta que el descriptor de la subcadena de una subcadena de una matriz es incorrecta en la versión actual del BASIC.

10.5 Ejemplos de almacenamiento de variables

Variables de coma flotante(en hex)

almacenadas como:	representan:
0000 0000 0000	0.0
0801 4000 0000	1.0
0800 8000 0000	-1.0
0804 5000 0000	10.0

ver tambien sección 8.5.4

Matrices de coma flotante

base,2,3,3,2,1	DIM A(3,2)

Almacenamiento de cadenas(números en decimal)

4;65,66,67,68	'ABCD'

Matrices de cadena

base,2,3,12,10,1	DIM A$(3,10)
4;65,66,67,68, x, x, x, x, x, x	"ABCD"
9;49,50,51,52,53,54,55,56,57, x	"123456789"
0; x, x, x, x, x, x, x, x, x, x	""
1;32, x, x, x, x, x, x, x, x, x	" "

Matrices de subcadena

base,1,3,1	A$(0,1 TO 3)
65,66,67	"ABC"

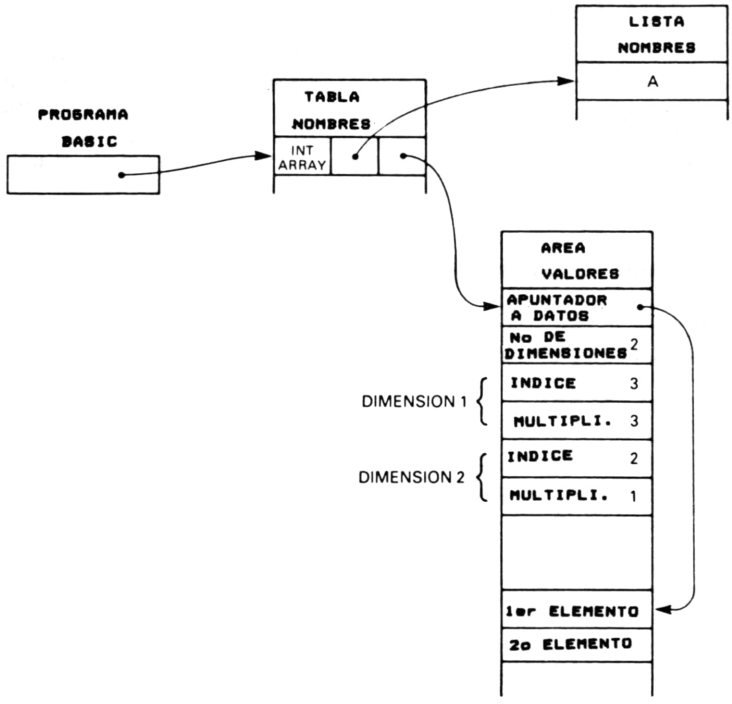


Figura 10.2 - Almacenamiento de matrices en BASIC

10.6 Pasando parámetros a procedimientos/funciones

Antes de entrar en más detalles sobre como escribir procèdimientos, es necesario entender como se transfieren paráme-tros entre los diferentes tipos de operaciónes.

Variables locales

Las variables locales dentro de los procedimientos y funciones, son las más fáciles de entender, porque no sucede practicamente nada. Cuando se encuentra una sentencia LOCAL, la entrada de la variable dada se copia en lo alto de la tabla de nombres. El contenido de la entrada original se reemplaza por el nuevo valor de la variable vacía. Esta nueva entrada vacía es la que se usa dentro del procedimiento. Al final, no se requiere más la variable LOCAL, por lo que la vieja entrada se copia de vuelta desde lo alto de la pila, borrando la entrada LOCAL en proceso.

Parámetros de llamada a procedimientos y funciones BASIC

El método que se sigue para pasar parámetros a un procedimiento es ligeramente más complejo que con las variables LOCAL. Cuando se llama al procedimiento, se crea una entrada completa por cada parámetro, en lo alto de la tabla de nombres. Para las funciones BASIC y procedimientos, se crean entradas nulas para los parámetros omitidos. El contenido de cada nueva entrada debe ser intercambiado con el contenido de cada entrada apuntada en la lista de parámetros. Todas las referencias dentro de la rutina, a los parámetros omitidos, usarán las nuevas definiciones.

Cuando se llega al final de un procedimiento, se vuelven a copiar los valores antiguos. Después de esto, se borran de la tabla de variables todas las entradas de parámetros y las entradas temporales. El mismo proceso ocurre con las rutinas en código máquina.

Los parámetros de llamada tienen el mismo formato que las variables ordinarias en la tabla de nombres, pero contienen información extra enmascarada dentro del octeto de código (el segundo) de la entrada de la tabla de nombres. La forma de este segundo octeto es:

hsssvvvv

h=1 si el parámetro va precedido por #

sss es el siguiente separador

000	sin separador
001	,
010	;
011	ñ
100	!
101	TO

vvv es el *tipo*

0000	nulo
0001	cadena
0010	coma flotante
0011	entero

10.7 Escribiendo procedimientos en código máquina

10.7.1 Introducción

Esta sección reune todas las ideas que se han ido presentando en este capítulo. Una vez qeu lo haya leido, será capaz de producir procedimientos simples y funciones para el SuperBASIC. La sección 10.8 incluye un ejemplo comprensible para poner facilidades para las pantallas 0 y 1.

Las siguiente reglas pueden serle útiles para escribir procedimientos en código máquina:

1. Recuerde en todo momento que el área del programa SuperBASIC es susceptible de moverse mientras se ejecuta en modo USUARIO. Por esta razón, es muy importante asegurarse que todas las referencias a este área se hacen usando A6 o A7 como índices. Ya que A6 y A7 son susceptibles de cambiar en cualquier momento, nunca se deben salvar, alterar (excepto metiendo o sacando A7 de la pila) o usar en cálculos aritméticos o de direcciones. Una vez dicho esto, es posible entrar en modo SUPERVISOR (usando el TRAP#0) para que la acción sea atómica. Si se hace así, asegurese que ni A6 ni USP(A7) son alterados antes de entrar en modo supervisor, y que son restaurados completamente antes de volver al modo usuario y retornar.

2. No use más de 128 occetos de la pila de usuario

3. D0 debe devolverse como un código de error de palabra larga.

4. A la entrada de la rutina, A3 apunta a la primera entrada en la tabla de nombres y A5 a la última. El número de parámetros en la tabla de nombres se puede calcular como (A5-A3)/8.

5. D1 a D7 y A0 a A5 se pueden cambiar.

10.7.2 Creando entradas en la tabla de nombres

Antes de que un procedimiento pueda ser totalmente acesible desde BASIC, debe estar debidamente conectado en la tabla de nombres. Los procedimientos BASIC en la ROM se conectan automáticamente (ver apéndice C). Es bastante simple conectar los procedimientos que están en la RAM. Generalmente, estos procedimientos están en el área de procedimientos residentes. Esto es debido a que no se pueden borrar, excepto

rearrancando el QL. Si se quita un procedimiento,una vez que se ha puesto su nombre en la tabla de nombres, cualquier intento de acceso haría que fallara el sistema entero.

Para conectar un procedimiento en RAM, A1 debe ponerse apuntando al principio de la lista de definición del procedimiento, y se debe hacer una llamada a BP.INIT (utilidad por vector $110). Se puede usar un código como este:

```
LEA    PROCDEF(PC),A1
MOVE.W $110,A2
JSR    (A2)
```

PROCDEF es una lista de definición de procedimiento en formato estándar:

1 palabra número de procedimientos

... por cada procedimiento:

1 palabra apuntador a la rutina
1 octeto longitud del nombre del procedimiento
x caracteres del nombre del procedimiento

... por cada función:

1 palabra apuntador a la rutina
1 octeto longitud del nombre de la función
x caracteres del nombre de la función

1 palabra 0 fin de la lista de definición

Los apunatdores a las rutinas son todos relativos a la dirección donde está asignado el apuntador.

El espacio interno de la tabla se reserva por el número de funciones o procedimientos. Si la longitud del nombre del procedimiento o función excede de 7, esta debe ser (número total de caracteres + número de funciones o procedimientos + 7)/8

BP.INIT preserva todos los registros excepto A1, y usa un máximo de 46 octetos en la pila del usuario.

10.7.3 La pila aritmética

El paquete de rutinas aritméticas requiere un poco de área de trabajo, para contener los argumentos y resultados de la evaluación de expresiones. La pila aritmética proporciona este área. Esta pila está normalmente referenciada por A1. Se puede usar en código máquina para evaluar la llamada o retornar parámetros, o simplemete como espacio general de trabajo.

Las rutinas que toman argumentos, reservan espacio suficiente en la pila aritmética, para su propio uso. Si un procedimiento diseñado por el usuario quiere utilizar espacio en la pila aritmética, lo debe reservar con una llamada a BV.CHRIX (utilidad por vector $11A). El número de octetos requeridos se pasa en D1.L. La pila aritmética se puede mover cuando se efectúa esta llamada. Por lo tanto, el apuntador a la pila aritmética, se debe salvar (en BV.RIP(A6)),si hubiera algo en la ella. BV.RIP debe volver a su valor original, al terminar.

La pila aritmética se limpia automáticamente por las rutinas de interconexión de los procedimientos y funciones, después de los procedimientos y funciones en error.

10.7.4 Tomando argumentos de los procedimientos

Se puede obtener un número indeterminado de argumentos de procedimiento, usando una de las rutinas CA.GTINT, CA.GTFP, CA.GTSTR y CA.GTLIN que obtienen palabras enteras, números de coma flotante, cadena y palabras largas enteras, respectivamente

A3 apunta a los argumentos (el primer parámetro en la tabla de nombres) y A5 (el último parámetro en la lista). Las rutinas convierten los parámetros entre estos límites a la forma requerida, poniendo los resultados dentro de la pila aritmética. El primer argumento está en la dirección más baja apuntada por (A6,A1.L). D3 contiene el número de argumentos que son recogidos al retorno. El # y los indicadores del separador en la entrada de la tabla de nombres, son destruidos. Los argumentos se pueden procesar de uno en uno extrayendo el # y el indicador de separador. A4 debe cargarse con 8 octeos más que A3 y se debe llamar a la rutina adecuada para obtener un argumento.

Las rutinas se pueden encontrar en la sección de las utilidades por vector:

$112	CA.GTINT	obtine palabras de enteros
$114	CA.GTFP	obtiene números de coma flotante
$116	CA.GTSTR	obtiene cadenas
$118	CA.GTLIN	obtiene palabras largas de enteros

10.7.5 Retorno de valores de variables

Se pueden devolver valores de variables desde las funciones, poniendo el valor en la pila aritmética con (A6,A1.L) apuntandolo. El valor de A1 se debe almacenar en el área de variables BASIC en BV.RIP(A6) que está situada en ($58(A6)). D4 debe contener el tipo del argumento retornado. Los argumentos posibles son:

D4=1	argumento de cadena
D4=2	argumento de coma flotante
D4=3	argumento entero (palabra)

Los enteros de palabra larga se deben convertir en formato de coma flotante. Solo se debe hacer un retorno bueno desde una función, si la pila está limpia. Esto significa que el argumento de retorno debe estar en lo alto de la pila, y la función no debe dejar nada debajo.

10.7.6 Retorno de valores de parámetros

Hay una rutina especial de utilidad, llamada BP.LET, que permite retornar valores a través de la lista de parámetros de un procedimiento o función. Para hacerlo, se dbe poner el valor a retornar en la pila aritmética (BV.RIP(A6) debe contenerlo). El argumento debe estar en la forma requerida por el parámetro de llamada, ej. cadena, entero o coma flotante. La entrada apropiada de parámetro en la tabla de nombres, debe ser apuntada por A3. Después se debe llamar a la rutina BP.LET (vector $120). Ver la sección 8.5.3 para más detalles sobre esta llamada.

Si el valor de retorno es una expresión y no una variable con nombre, el valor retornado se perderá. Si se devuelven cadenas, se debe tener cuidado en asegurarse que el contador de octetos de la cadena está en frontera de palabra. Esto se puede hacer rellenando la longitud impar con un octeto a ceros al final.

10.8 Cambiando entre las pantallas 0 y 1

En el QL, hay dos diferentes áreas de memoria que se pueden

usar para generar una imagen en la pantalla. La pantalla *normal* se llama pantalla 0 y está situada en los 32K inferiores de la RAM, debajo de las variables del sistema. La segunda pantalla está situada en los 32K de RAM por encima de la primera. Desgraciadamente, Las variables del sistema está situadas por encima de la pantalla 0. Esto significa que está realmente dentro de la pantalla 1.

Sin embargo, no todo está perdido. Hay todavía una posibilidad de usar la mayoría de la pantalla 1 (quitando los 5K asignados a las variables del sistema). Para hacerlo, la pantalla 0 se debe usar solamente para mostrar la quinta parte superior de la pantalla. La pantalla 1 se vuelca a la pantalla, bajo control de las interrupciones, cada cuadro. La parte inferior de la pantalla se puede usar entonces para gráficos rápidos. El dibujo ocurre en una pantalla mientras que se muestra en la otra, y viceversa.

Ahora puede usted pensar que, el cambio de pantallas se debe hacer mediante procedimientos y funciones añadidas al BASIC. En esta sección se presenta un paquete completo de utilidad para el manejo de pantallas. Se añaden tres nuevos procedimientos al BASIC. SCR0 pone la pantalla 0, SCR1 pone la pantalla 1 (con la parte superior enmascarada) y SCRA pone el modo de *auto-cambio* de forma que al pulsar CTRL-F5, se cambiará de una pantalla a otra. Para que el programa nos indique que pantalla está usandose en cada momento, se proporciona una función llamada SCRNUM.

10.8.1 Programa de soporte de dos pantallas

```
*
* Utilidad de soporte de dos pantallas
*
MT.ALCHP  EQU     $18
MT.LPOLL  EQU     $1C
BP.INIT   EQU     $110
SV_SCRST  EQU     $33
SV_MCSTA  EQU     $34
SV_SER1C  EQU     $98
SV_SER2C  EQU     $9C
SV_MDRUN  EQU     $EE
MC_STAT   EQU     $18063
MC..SCRN  EQU     $7
BV_RIP    EQU     $58
*
* Primero inicia los procedimientos extra de BASIC
*
          MOVE.W  BP.INIT,A2
```

```
        LEA     PROC_DEF(PC),A1
        JSR     (A2)
*
* Ahora reserva la segunda pantalla
*
        MOVEQ   #MT.ALCHP,D0
        MOVE.L  #$6BE0,D1
        TRAP    #1
        TST.L   D0
        BNE.S   EXIT
*
* Ahora conéctalo al gestor de interrupciones
*
        LEA     INT_SERVE(PC),A1  dirección de entrada
        MOVEQ   #MT.LPOLL,D0
        LEA     INT_LINK(PC),A0   dirección de conexión
        MOVE.L  A1,4(A0)          poner entrada en conexión
        TRAP    #1                conectarla
EXIT
        RTS
*
* Definición de procedimientos BASIC
*
PROC_DEF
        DC.W    3
        DC.W    SCRA-*
        DC.B    4,'SCRA'
        DC.W    SCR0-*
        DC.B    4,'SCR0'
        DC.W    SCR1-*
        DC.B    4,'SCR1'
        DC.W    0
        DC.W    1
        DC.W    SCRNUM-*
        DC.B    6,'SCRNUM'
        DC.W    0
*
* Poner auto-cambio de pantallas
*
SCRA
        LEA     SCR_AUTO(PC),A1
        ST      (A1)              poner auto-cambio
        BRA.S   EXIT_OK
*
* Poner pantalla 0
*
SCR0
        LEA     SCR_AUTO(PC),A1
        CLR.W   (A1)              pantalla 0, no auto-cambio
        BRA.S   EXIT_OK
```

```
*
* Poner pantalla 1
*
SCR1
          LEA     SCR_AUTO(PC),A1
          MOVE.W  #$00FF,(A1)       pantalla 1, no auto-cambio
          BRA.S   EXIT_OK
*
* Ver que pantalla está mostrandose
*
SCRNUM
          MOVE.L  BV_RIP(A6),A1     debe haber sitio para dos
*                                   octetos
          SUBQ    #2,A1
          MOVE.L  A1,BV_RIP(A6)
          MOVEQ   #1,D1             retorno 1
          AND.B   SCR_1(PC),D1      ... o   0
          MOVE.W  D1,(A6,A1.L)
          MOVEQ   #3,D4             es tipo entero
EXIT_OK
          MOVEQ   #0,D0
          RTS
*
*
* Servicio de interrupciones
*
SCR_AUTO  DS.B    1
SCR_1     DS.B    1
INT_LINK  DS.L    2               dirección conex. y entrada
INT_BASE  EQU     INT_LINK-8      tabla conex. (A3 a entrada)
INT_SERVE
          MOVE.B  SV_MCSTA(A6),D1 devolver pantalla 0
          MOVE.B  D1,MC_STAT
*
          TST.B   SCR_1-INT_BASE(A3) requiere pantalla 1?
          BEQ.S   INT_TOGGLE      ... no,
          TST.L   SV_SER1C(A6)    está abierta serie 1
          BNE.S   EXIT_INT        ... si, error temporiz.
          TST.L   SV_SER2C(A6)    está abierta serie 2
          BNE.S   EXIT_INT        ... si, error temporiz.
          TST.B   SV_MDRUN(A6)    está funcionando microdriv.
          BNE.S   EXIT_INT        ... si, error temporiz.
*
          MOVE.W  #1360,D0        esperar a pasar las
          DBRA    D0,*            ... variables del sistema
*
          BSET    #MC..SCRN,D1
          MOVE.B  D1,MC_STAT
*
```

```
INT_TOGGLE
        TST.B   SCR_AUTO-INT_BASE(A3) CTRL F5 cambiando?
        BEQ.S   EXIT_INT        ... no
        TST.B   SV_SCRST(A6)    se ha pulsado CTRL F5?
        BEQ.S   EXIT_INT        ... no
        SF      SV_SCRST(A6)
        NOT.B   SCR_1-INT_BASE(A3) cambiar pantalla
EXIT.INT
        RTS
```

10.8.2 Cómo funciona el programa

La primera parte del programa se encarga de la iniciación. BP.INIT se usa para preparar las entradas de la tabla de nombres de los procedimientos y las funciones (ver sección 10.7.2 para más información). Después se llama al TRAP del gestor MT.ALCHP para reservar memoria en el área común ($6BE0 octetos). Esto se hace para que no pueda escribir sobre la pantalla 1 ninguna otra asignación del área común. Finalmente, se conecta el director de interrupciones a la lista encadenada de interrupciones 50/60Hz. Esta es la rutina que cambia de la pantalla 0 a la 1, controlando cuidadosamente la posición en que ocurre el cambio.

El código de cada procedimiento es bastante simple. Cada uno contiene un octeto en SCR_AUTO. El gestor de interupciones lee esta posición para determinar cuando debe cambiar de pantalla o poner la 0 o la 1.

SCRNUM es una función que devuelve el número de la pantalla actual como 0 o 1. retorna el valor al BASIC usando el mecanismo que se describe en la sección 10.7.5. El valor devuelto se convierte en tipo entero.

La rutina de servicio de interrupciones comprueba el contenido de SCR_AUTO para determinar que acción (si hay alguna) se debe tomar. La pantalla 1 solo se puede seleccionar después de las variables del sistema (de otro modo aparecería la imagen de las variables del sistema). La temporización para este periodo se hace usando DBRA D0,* que pone el 68008 en un bucle interno temporizado. Esto solo puede funcionar si no se están usando las interrupciones de alta prioridad de las puertas serie o los microdrives. Si va a ocurrir alguna de estas interrupciones, la pantalla 1 es retenida.

10.9 Implementación de BPUT# como procedimiento

Este ejemplo prepara un procedimiento BASIC que permite mandar octetos individuales a un canal (como un microdrive).

```
*
* Utilidad de sacar un octeto simple
*
MT.LPOLL  EQU   $1C
CA.GTINT  EQU   $112
IO.SBYTE  EQU   $05
ERR.BP    EQU   -15
ERR.NO    EQU   -6
BP.INIT   EQU   $110
BV_CHBAS  EQU   $30
BV_CHP    EQU   $34
*
* Primero inicia el procedimiento BASIC BPUT#n,dato
*
          MOVE.W  BP.INIT,A2
          LEA     PROC_DEF(PC),A1
          JSR     (A2)
          MOVEQ   #0,D0
          RTS
*
* Definición del procedimiento BASIC
*
PROC_DEF
          DC.W    1
          DC.W    BPUT-*
          DC.B    4,'BPUT'
          DC.W    0
          DC.W    0
          DC.W    0
*
* Rutina BPUT
*
BPUT
          BSR.S   CHANNEL
          MOVE.W  CA.GTINT,A2
          JSR     (A2)
          SUBQ.W  #1,D3           solo un argumento?
          BNE.S   ERR_BP
          MOVE.B  1(A6,A1.L),D1
          MOVEQ   #IO.SBYTE,D0    mandar el octeto al canal
          MOVEQ   #-1,D3
          TRAP    #3
          RTS
```

```
*
ERR_BP
        MOVEQ   #ERR.BP,D0      parámetros inválidos
        RTS
*
* Pone el canal dado o el defecto
* Parámetros de llamada   :
*
* A3 y A5 apuntadores estándar a la tabla de nombres para los
* parámetros
*
* Parámetros retornados   :
*
* D6 apuntador a la tabla de canales
* A0 ID del canal
*
CHANNEL
        MOVEQ   #1,D6           canal #1 por defecto
        CMPA.L  A3,A5           algún parámetro?
        BEQ.S   CHAN_LOOK       ... no
*
        BTST    #7,1(A6,A3.L)   es # el 1er parámetro?
        BEQ.S   CHAN_LOOK       ... no
*
        MOVE.L  A5,-(A7)        salvar apuntador a 1er param
        MOVE.L  A3,A5           poner nuevo tope
        ADDQ    #8,A5           a 8 octetos desde abajo
        MOVE.L  A5,-(A7)        (ahora es la nueva base)
        MOVE.W  CA.GTINT,A2     tomar un entero
        JSR     (A2)
        MOVE.L  (A7)+,A3        restaurar apunt. parámetros
        MOVE.L  (A7)+,A5        (no altera codigos condición
        BNE.S   CHAN_EXIT       está bien?
        MOVE.W  0(A6,A1.L),D6   reemplazar defecto con D6
*
CHAN_LOOK
        MULU    #$28,D6         D6 apunta a tabla canales
        ADD.L   BV_CHBAS(A6),D6
        CMP.L   BV_CHP(A6),D6   está dentro de la tabla?
        BHI.S   ERR_NO          ... no
        MOVE.L  0(A6,D6.L),A0   poner ID del canal
        MOVEQ   #0,D0           sin error
CHAN_EXIT
        RTS
ERR_NO
        MOVEQ   #ERR.NO,D0      canal no abierto
        RTS
```

10.9.1 Cómo funciona el programa

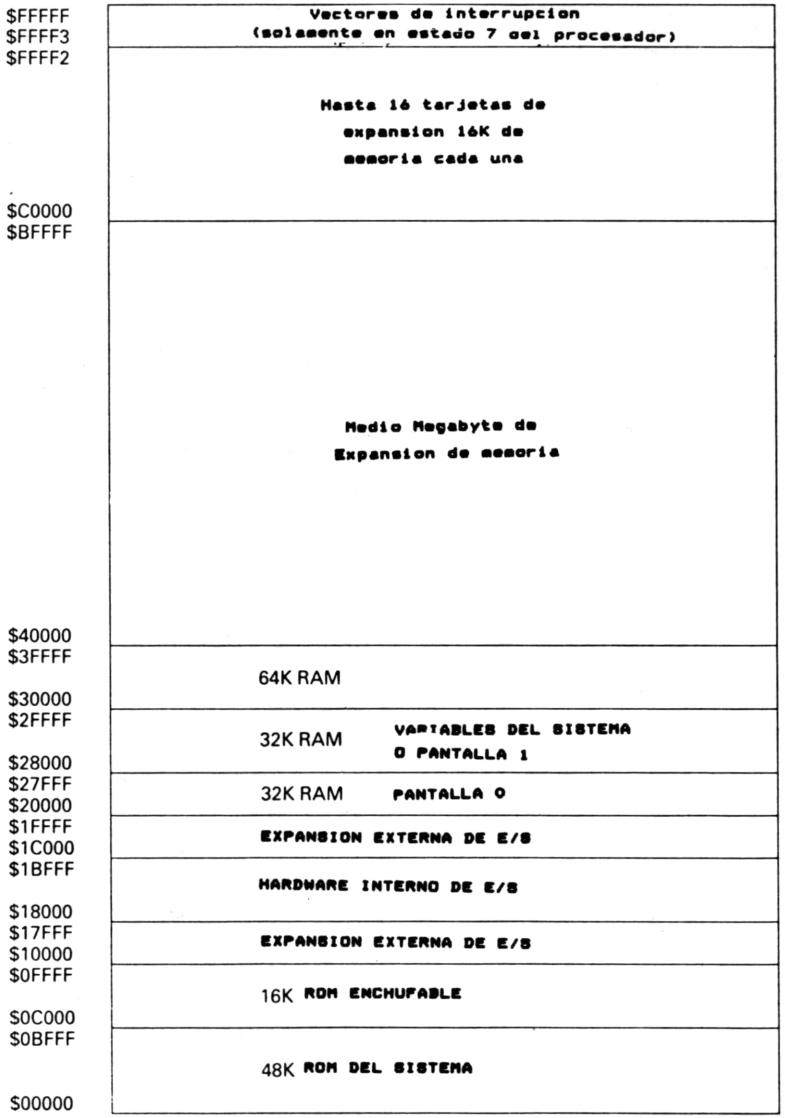
La primera parte del programa se encarga de la iniciación. BP.INIT se usa para preparar la entrada del procedimiento BPUT, en la tabla de nombres.

Después, la rutina CHANNEL, busca el número del canal precedido por '#'. Si lo encuentra, el registro A0 nos devuelve el ID del canal con un apuntador a la tabla de canales en D6. El octeto que se va a mandar a este canal se obtiene usando CA.GTINT. Finalmente, se manda el octeto con IO.SBYTE.

En la sección 8.8.5 podemos ver otro ejemplo de conexión de una función al BASIC.

Apéndice A - Mapa de la memoria

Este apéndice muestra el mapa de la memoria del QL desde el punto de vista del 'hardware'. El punto de vista de la mamoria, desde el QDOS, se trató en la sección 3.2.



Apéndice B - Sumario de juego de instrucciones del 68008

En el capítulo 2 puede encontrar un análisis más detallado del microprocesador 68008 y su juego de instrucciones.

Mnemonic	Description
ADBC	Add Decimal With Extend
ADD	Add
AND	Logical And
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
B_{CC}	Branch Conditionally
BCHG	Bit Test and Change
BCLR	Bit Test and Clear
BRA	Branch Always
BSET	Bit Test and Set
BSR	Branch to Subroutine
BTST	Bit Test
CHK	Check Register Against Bounds
CLR	Clear Operand
CMP	Compare
DB_{CC}	Test Condition, Decrement and Branch
DIVS	Signed Divide
DIVU	Unsigned Divide
EOR	Exclusive Or
EXG	Exchange Registers
EXT	Sign Extend
JMP	Jump
JSR	Jump to Subroutine
LEA	Load Effective Address
LINK	Link Stack
LSL	Logical Shift Left
LSR	Logical Shift Right

Mnemonic	Description
MOVE	Move
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral Data
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NOP	No Operation
NOT	One's Complement
OR	Logical Or
PEA	Push Effective Address
RESET	Reset External Devices
ROL	Rotate Left without Extend
ROR	Rotate Right without Extend
ROXL	Rotate Left with Extend
ROXR	Rotate Right with Extend
RTE	Return from Exception
RTR	Return and Restore
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
S_{CC}	Set Conditional
STOP	Stop
SUB	Subtract
SWAP	Swap Data Register Halves
TAS	Test and Set Operand
TRAP	Trap
TRAPV	Trap on Overflow
TST	Test
UNLK	Unlink

Variaciones de los tipos de instrucciones

Instruction Type	Variation	Description
ADD	ADD	Add
	ADDA	Add Address
	ADDQ	Add Quick
	ADDI	Add Immediate
	ADDX	Add with Extend
AND	AND	Logical And
	ANDI	And Immediate
	ANDI to CCR	And Immediate to Condition Codes
	ANDI to SR	And Immediate to Status Register
CMP	CMP	Compare
	CMPA	Compare Address
	CMPM	Compare Memory
	CMPI	Compare Immediate
EOR	EOR	Exclusive Or
	EORI	Exclusive Or Immediate
	EORI to CCR	Exclusive Or Immediate to Condition Codes
	EORI to SR	Exclusive Or Immediate to Status Register

Instruction Type	Variation	Description
MOVE	MOVE	Move
	MOVEA	Move Address
	MOVEQ	Move Quick
	MOVE from SR	Move from Status Register
	MOVE to SR	Move to Status Register
	MOVE to CCR	Move to Condition Codes
	MOVE USP	Move User Stack Pointer
NEG	NEG	Negate
	NEGX	Negate with Extend
OR	OR	Logical Or
	ORI	Or Immediate
	ORI to CCR	Or Immediate to Condition Codes
	ORI to SR	Or Immediate to Status Register
SUB	SUB	Subtract
	SUBA	Subtract Address
	SUBI	Subtract Immediate
	SUBQ	Subtract Quick
	SUBX	Subtract with Extend

Apéndice C - Especificaciones de la ROM enchufable

Introducción

Este apéndice puede ser de interés para aquellos que quieran añadir sus propios programas de utilidad al QL, en formato de ROM. También puede ser de interés para los que quieran saber como conecta el QDOS las rutinas suministradas en ROM.

La ranura de cartuchos ROM se encuentra en la parte trasera del QL. Esta ROM se conecta directamente en el circuito impreso del QL, mediante un conector para circuito impreso de 30 vias a doble cara. Esto permite conectar ROMs de hasta 16K de tamaño. El cartucho puede contener, por ejemplo:

1. Programas que pueden reemplazar al SuperBASIC tales como nuevos lenguajes como LISP, PASCAL, BCPL, ensambladores, utilidades para depuración de programas, aplicaciones especializadas, etc.

2. Programas que se pueden conectar al SuperBASIC para añadir procedimientos nuevos.

—	a	1	b	+5v
A12	a	2	b	A14
A7	a	3	b	A13
A6	a	4	b	A8
A5	a	5	b	A9
SLOT		6		SLOT
A4	a	7	b	A11
A3	a	8	b	ROMOEH
A2	a	9	b	A10
A1	a	10	b	A15
A0	a	11	b	D7
D0	a	12	b	D6
D1	a	13	b	D5
D2	a	14	b	D4
GND	a	15	b	D3

Figura C.1 - Señales en la ranura de la ROM

Apéndice B - Sumario de juego de instrucciones del 68008

En el capítulo 2 puede encontrar un análisis más detallado del microprocesador 68008 y su juego de instrucciones.

Mnemonic	Description
ADBC	Add Decimal With Extend
ADD	Add
AND	Logical And
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
B_{CC}	Branch Conditionally
BCHG	Bit Test and Change
BCLR	Bit Test and Clear
BRA	Branch Always
BSET	Bit Test and Set
BSR	Branch to Subroutine
BTST	Bit Test
CHK	Check Register Against Bounds
CLR	Clear Operand
CMP	Compare
DB_{CC}	Test Condition, Decrement and Branch
DIVS	Signed Divide
DIVU	Unsigned Divide
EOR	Exclusive Or
EXG	Exchange Registers
EXT	Sign Extend
JMP	Jump
JSR	Jump to Subroutine
LEA	Load Effective Address
LINK	Link Stack
LSL	Logical Shift Left
LSR	Logical Shift Right

Mnemonic	Description
MOVE	Move
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral Data
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NOP	No Operation
NOT	One's Complement
OR	Logical Or
PEA	Push Effective Address
RESET	Reset External Devices
ROL	Rotate Left without Extend
ROR	Rotate Right without Extend
ROXL	Rotate Left with Extend
ROXR	Rotate Right with Extend
RTE	Return from Exception
RTR	Return and Restore
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
S_{CC}	Set Conditional
STOP	Stop
SUB	Subtract
SWAP	Swap Data Register Halves
TAS	Test and Set Operand
TRAP	Trap
TRAPV	Trap on Overflow
TST	Test
UNLK	Unlink

Variaciones de los tipos de instrucciones

Instruction Type	Variation	Description
ADD	ADD	Add
	ADDA	Add Address
	ADDQ	Add Quick
	ADDI	Add Immediate
	ADDX	Add with Extend
AND	AND	Logical And
	ANDI	And Immediate
	ANDI to CCR	And Immediate to Condition Codes
	ANDI to SR	And Immediate to Status Register
CMP	CMP	Compare
	CMPA	Compare Address
	CMPM	Compare Memory
	CMPI	Compare Immediate
EOR	EOR	Exclusive Or
	EORI	Exclusive Or Immediate
	EORI to CCR	Exclusive Or Immediate to Condition Codes
	EORI to SR	Exclusive Or Immediate to Status Register

Instruction Type	Variation	Description
MOVE	MOVE	Move
	MOVEA	Move Address
	MOVEQ	Move Quick
	MOVE from SR	Move from Status Register
	MOVE to SR	Move to Status Register
	MOVE to CCR	Move to Condition Codes
	MOVE USP	Move User Stack Pointer
NEG	NEG	Negate
	NEGX	Negate with Extend
OR	OR	Logical Or
	ORI	Or Immediate
	ORI to CCR	Or Immediate to Condition Codes
	ORI to SR	Or Immediate to Status Register
SUB	SUB	Subtract
	SUBA	Subtract Address
	SUBI	Subtract Immediate
	SUBQ	Subtract Quick
	SUBX	Subtract with Extend

Apéndice C - Especificaciones de la ROM enchufable

Introducción

Este apéndice puede ser de interés para aquellos que quieran añadir sus propios programas de utilidad al QL, en formato de ROM. También puede ser de interés para los que quieran saber como conecta el QDOS las rutinas suministradas en ROM.

La ranura de cartuchos ROM se encuentra en la parte trasera del QL. Esta ROM se conecta directamente en el circuito impreso del QL, mediante un conector para circuito impreso de 30 vias a doble cara. Esto permite conectar ROMs de hasta 16K de tamaño. El cartucho puede contener, por ejemplo:

1. Programas que pueden reemplazar al SuperBASIC tales como nuevos lenguajes como LISP, PASCAL, BCPL, ensambladores, utilidades para depuración de programas, aplicaciones especializadas, etc.

2. Programas que se pueden conectar al SuperBASIC para añadir procedimientos nuevos.

—	a	1	b	+5v
A12	a	2	b	A14
A7	a	3	b	A13
A6	a	4	b	A8
A5	a	5	b	A9
SLOT		6		SLOT
A4	a	7	b	A11
A3	a	8	b	ROMOEH
A2	a	9	b	A10
A1	a	10	b	A15
A0	a	11	b	D7
D0	a	12	b	D6
D1	a	13	b	D5
D2	a	14	b	D4
GND	a	15	b	D3

Figura C.1 - Señales en la ranura de la ROM

Conexiones de 'hardware'

En la ranura de la ROM se proporcionan una gran variedad de señales útiles. La 16 líneas de dirección permiten direccionar hasta 64K octetos de ROM, pero en la práctica solo hay 16K disponibles en el mapa de memoria (ver apéndice A). Las líneas de datos permiten transferir octetos desde la ROM al microprocesador 68008. ROMOE es una señal, activa cuando está baja, que habilita a la ROM para salida cuando el 68008 intenta leerla. Finalmente, hay dos líneas de alimentación con 0 y +5 voltios, respectivamente. En la figura C.1 se puede ver la dispodición de las diferentes señales.

Especificaciones de los programas en ROM

Los programas de la ROM enchufable tinen varios aspectos. Primero, debe haber una secuencia especial para el reconocimineto de la ROM, que comprueba el QDOS. Si existe esta secuencia, reconoce que hay una ROM conectada en la parte trasera del QL. Segundo, una vez que se ha cerciorado de que hay una ROM ahí, el QDOS debe ser capaz de encontrar las rutinas que quiere usar, y conectarlas dentro del sistema.

El código especial de reconocimiento se debe colocarla al principio de la ROM y es 4AFB0001 Hex seguido de la cabecera del controlador. Esta cabecera apunta a una lista de procedimientos y funciones BASIC dentro de la ROM, junto con las direcciones de iniciación de las rutinas. Hay también una cadena que identifica la ROM. El formato de la cabecera es:

```
00  palabra larga  indicador de reconocimiento($4AFE0001)
04  apuntador a la lista de procedimientos y funciones
06  apuntador a la rutina de iniciación
08  cadena que identifica la ROM
```

Todos los apuntadores son relativos a la dirección base de la ROM. La cadena que identifica la ROM debe estar en forma de contador de caracteres (palabra) seguido por los caracteres ASCII de la descripción del dispositivo. Esta cadena debe terminar con un <LF>. Se recomienda no pasar de 36 caracteres.

La lista de los procedimientos y funciones debe estar en el siguiente formato:

```
palabra   número de procedimientos
```

por cada procedimiento:

```
palabra     apuntador a la rutina de la ROM
octeto      longitud del nombre del procedimiento
caracteres  nombre
```

seguido por:

```
palabra     cero al final de un procedimiento
palabra     número de funciones
```

y por cada función:

```
palabra     apuntador a la rutina en la ROM
octeto      longitud del nombre de la función
caracteres  nombre
```

y finalmente:

```
palabra     igual a cero al final de las funciones
```

el código de las rutinas puede empezar en este punto hasta rellenar el resto de la ROM.

La secuencia de los eventos, cuando se enciende la máquina, es la siguiente. Primero, se comprueba el indicador de reconocimiento de la ROM. Si hay una ROM presente, la máquina conecta los procedimientos BASIC adicionales que incluye la ROM. El número de procedimientos y funciones se usa para reservar espacio en las tablas internas en RAM. Si la longitud del nombre del procedimiento o función, excede de 7, el número efectivo de procedimientos y funciones se calcula como (número total de caracteres + número de funciones o procedimientos + 7)/8. Después de conectarse al BASIC, se llama a la rutina de iniciación (en modo usuario). Esta rutina no debe modificar A6, y debe restaurar A0 (la ID de la ventana inicial) y A3 (el apuntador a la ROM), antes de retornar. Se pueden usar hasta 128 octetos en la pila de usuario.

Finalmente, para más información sobre la forma de escribir controladores de dispositivo, refierase al capítulo 9.

Apéndice D - Formato de los Microdrives

Este apéndice es una colección de datos sobre los microdrives del QL. Nos da detalles sobre la forma en que se disponen los ficheros sobre el bucle contínuo de cinta y por que. La lectura de este apéndice le dará unos conocimientos más profundos sobre la operación de los microdrives según los ve la máquina.

D.1 Formato de los microdrives

Los microdrives están formados por un largo bucle de cinta magnética de alta calidad. Cuando está funcionando el motor, cualquier punto concreto de la cinta pasará sobre la cabeza de lectura/escritura una vez cada 7 segundos. Se requiere un sistema de encontrar el sitio de la cinta donde están situados unos datos concretos. Idealmente, los datos se escriben sobre toda la longitud de la cinta. En la práctica, esto no se hace por varias razones. Primera de todas, un sistema no es eficiente cuando no es posible localizar fácilmente el dato que estamos buscando. Segundo, si alguien quiere modificar un octeto, la cabeza de borrado se debe activar instantaneamente antes de que halla pasado el octeto, vuelta a desactivar al final del mismo y asegurarse de que no se han borrado otros octetos.

Para producir un sistema práctico y eficiente, el bucle se ha dividido en sectores. Cada uno de estos sectores tiene 512 octetos de datos. Cuando se formatea por primera vez un cartucho de microrive, se graban pequeñas marcas de sector (cabeceras de sector) a lo largo de toda la cinta. El espacio entre las cabeceras de sector se rellena con datos inservibles, para comprobar que se pueden grabar datos correctamente en esa parte del bucle. Al principio y al final de cada sector hay un espacio libre para asegurarse de que el sector, se puede modificar sin afectar a los sectores adyacentes, aúnque se use en otra unidad donde la velocidad del motor puede variar ligeramente. Para leer o modificar un solo octeto, se debe leer el sector completo. Una vez que se ha procesado el octeto apropiado, y por supuesto modificado, es necesario escribir de nuevo al microdrive los 512 octetos del sector.

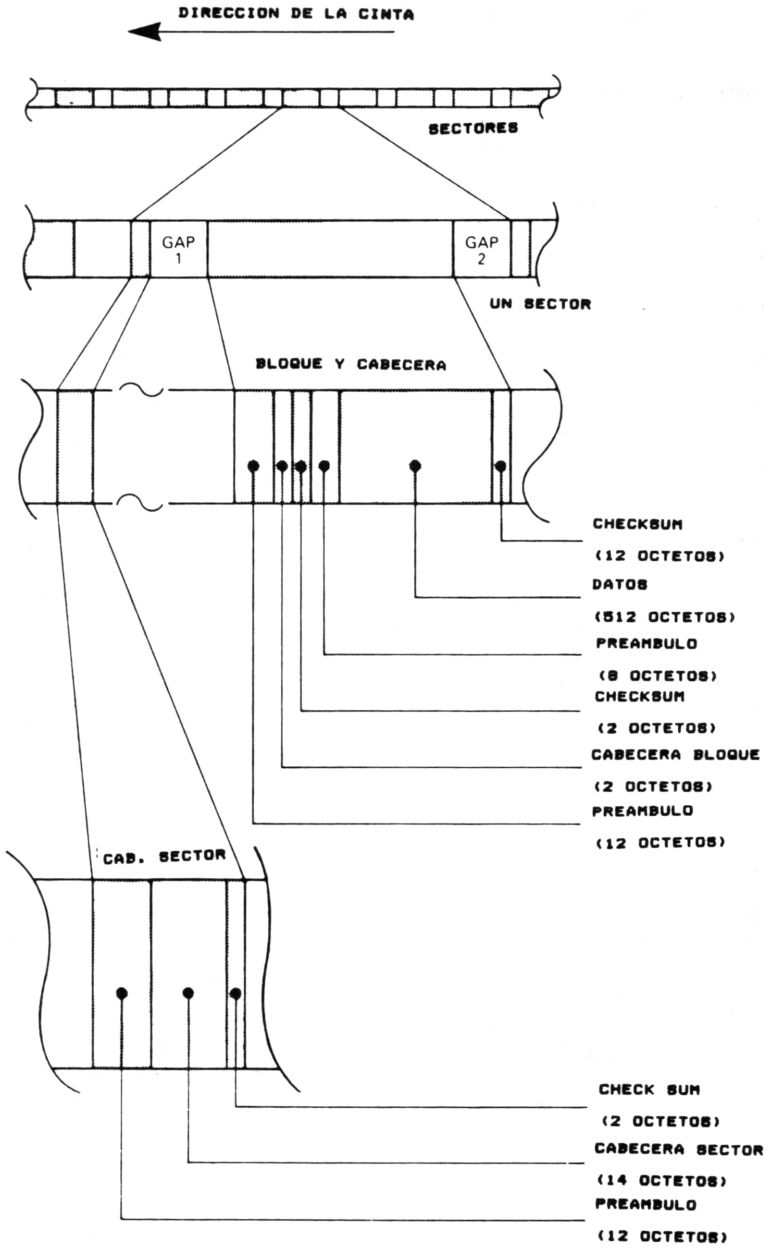


Figura D.1 - Formato de los sectores de microdrive

Hay cuatro elementos reconocibles grabados en un microdrive

1. Preámbulo/Sinc

El 'preámbulo/sinc' consiste en un mínimo de 5 octetos a ceros. Los dos primeros permiten reconocer el preámbulo. Una vez reconocido, se usan los siguientes tres octetos para enganchar el PLL('phase locked loop'). El preámbulo va seguido por un octeto a unos ($FF) en cada pista (hay dos pistas) para sincronizar los registros de lectura en frontera de octeto.

El preámbulo estándar al principio de un secotr tiene 10+2 octetos de longitud. Dentro del bloque se usa un preámbulo especial de 6+2 octetos. Esto anula el PLL de forma que se pueda leer la cabecera del bloque separadamente de la parte principal del mismo.

2. Datos

El propósito final del microdrive es almacenar datos útiles, de modo que este es el más importante de todos los elementos (pero que no puede existir por sísolo). Los datos se graban en múltiplos de 2 octetos. Hay tres tipos de datos en un sector de formato estándar. Son, la cabecera del sector($FF, número de sector y 10 octetos del nombre), la cabecera de bloque (número del fichero y número del bloque), y el bloque en sí (512 octetos).

3. Checksum

Cada bloque de datos tiene una palabra especial detrás. Esta palabra es el 'checksum', que permite la verificación de los datos. Si el 'checksum' que se genera al leer los datos no coincide con el que hay grabado al final, normalmente significa que ha habido un error de lectura. El 'checksum' es rápido de calcular, es trasparente a la pérdida de bits (rotación de octetos aparentes de datos) y es efectivo ante los grupos de bits a ceros o a unos. El 'checksum' se calcula de la siguiente forma:

1. se pone al principio a $0F0F
2. por cada octeto, se añade este al 'checksum'
3. grabar o comprobar el 'checksum', primero el octeto inferior y después el superior.

4. Hueco

Los huecos se usan para evitar que se escriban, inadvertidamente, unos datos encima de otros. Hay un hueco para separar la cabecera de sector (que solo se escribe cuando se cormatea la cinta) de los datos que contiene (que es reescrito cada vez que se necesita). Hay otro hueco entre el final de un sector y el principio del siguiente.

La longitud mínima requerida para el hueco, se calcula de la siguiente forma. El peor caso de tiempo de borrado (200us) más el tiempo requerido para eliminar el borrado (20us), más un tiempo incierto de los procedimientos de escritura y formateo (siempre menor de 160us), más (solo para los huecos entre sectores - la cabecera del sector no se reescribe nunca) la máxima variación del bloque escrito, incluido el hueco anterior al bloque (10% de la longitud total).

Con esta longitud de hueco, es posible que permanezcan partes del formato del bloque, después que se ha reescrito un sector. Este problema se combate controlando cuidadosamente el contenido del bloque de formato, que nos asegura que no sa va a tomar parte del final del bloque de formato, como si fuera una cabecera de sector.

La cabecera de sector

La cabecera de sector tiene 14 octetos de longitud. El primer octeto es el indicador de cabecera de sector, y está puesto a $FF. El segundo octeto contiene el número del sector. Los sectores están numerados por orden descendente desde 255 (esto depende de la longitud real de la cinta en el cartucho) a 0.

Los 10 octetos siguientes contienen el nombre del medio, que puede tener hasta 10 caracteres de longitud. Si el nombre es menor, los restantes octetos se rellenan con espacios. Los octetos 13 y 14 contienen un número aleatorio de 16 bit.

La cabecera de bloque

La cabecera de bloque tiene dos octetos. El primero es el número del fichero o un indicador. El número del fichero está en el rango de $00 a $F0 para los ficheros comunes. Hay dos números de fichero especiales, $F8 está reservado para el mapa del microdrive y el valor de indicador $FD que marca

el bloque como vacante o malo. El segundo octeto contiene el número de bloque ($00 a $FE). Los números de bloque empiezan por el cero, por lo que los octetos 0 a 511 del fichero están en el bloque 0, los octetos 512 a 1023 en el bloque 1, y así sucesivamente.

El bloque

Los bloque tienen 512 octetos de longitud y contienen los datos del fichero. Los datos no comienzan inmediatamente después del final de la cabecera de bloque, sino que tienen un pequeño preámbulo que permite a los programas de alto nivel hacer algún pequeño proceso antes de leer el bloque.

Bloques especiales

El sector 0 contiene el único bloque especial del medio. Este bloque contiene el mapa de sectores, y se le ha designado como fichero número $F8, bloque 0.

El bloque contiene 255 pares de octetos. Cada par contiene el número del fichero que ocupa ese sector más el número del bloque dentro del fichero. Un número de fichero de $FD indica que el sector está vacante, mientras que un número de fichero de $FF, indica que el sector está dañado y no debe ser usado. El último octeto del bloque del mapa del sector, contiene el número del sector más recientemente asignado en el medio.

Estructura de ficheros

Hay una cabecera de 64 octetos al principio de cada fichero. Normalmente no concierne al usuario, ya que los programas de alto nivel lo esconden a las aplicaciones ordinarias.

El formato es:

$00	longitud del fichero
$04	clave de acceso del fichero
$05	tipo de fichero
$06	8 octetos de información dependiente del tipo
$0E	nombre del fichero (2 octetos de contador + caracteres ASCII)
$34	fecha actualizada-no implementada en QDOS V1.03
$38	fecha de referencia-no implementada en QDOS V1.03
$3C	fecha copia-no implementado en QDOS V1.03

El directorio

El directorio es el fichero 0, y es un fichero muy especial ya que mantiene copias de las cabeceras de todos los ficheros del medio. La cabecera del fichero 0 (él mismo) empieza en el octeto 0 del fichero, la copia de la cabecera del fichero 1, empieza en el octeto 64 y asísucesivamente. ParA borrar un fichero, se borran de la copia en el directorio su longitud y la de su nombre. Esto significa que el fichero puede ser recuperado con un programa de utilidad que reemplace estas dos piezas de información.

Asignación de sectores

Los microdrives tiene un tiempo de comienzo y de fin, finito. Por lo tanto, es necesario que el motor consiga la velocidad adecuada, antes de que pueda leer o escribir. Hay dos factores que se deben acomodar lo mejor posible (esto depende de lo lleno que esté el microdrive). Antes de nada, el primer bloque del fichero se debe posicionar, por lo que hay 20 sectores entre el sector asignado más recientemente y él mismo (esto se hace para permitir las operaciones que comprueban que el medio no se ha alterado). El segundo factor es para un fichero que tenga más de un bloque de longitud. Se saltan doce sectores entre bloques sucesivos de un fichero. Esto permite al motor pararse y rearrancar entre los bloques.

Estructura de un sector en un microdrive

Descripción	# de octetos	tiempo	tiempo total
Preámbulo	12 octetos	480 us	
Cabecera sector	14 octetos	560 us	1040 us
Checksum	2 octetos	80 us	1120 us
Hueco 1		3600 us	4720 us
Preámbulo	12 octetos	480 us	
Cabecera bloque	2 octetos	80 us	560 us
Checksum	2 octetos	80 us	640 us
preámbulo	8 octetos	320 us	960 us
Registro	512 octetos	20480 us	21440 us
Checksum	2 octetos	80 us	21520 us
Hueco 2		5520 us	27040 us
	Total	**31760 us**	

Estos tiempos dan 225 + 5% sectores por medio

Estructura de sectores especiales

La utilidad de formateo usa una estructura especial para asegurarse de que la región entera puede escribirse. Esta estructura especial requiere que la región de la cinta verificada solape todas las regiones que puedan ser usadas para escribir datos.

La cabecera de sector se deja en su formato estándar ya que no se reescribe. Sin embargo, el Hueco 1 es acortado de forma que el bloque que se escribe solapa la región que ocupa normalmente el preámbulo. El siguiente bloque se extiende añadiendo 86 octetos al final. El bloque se rellena con $AA55 (binario 1010 1010 0101 0101) excepto la palabra que está a 512 octetos del principio, ej. el 'checksum' estándar. La estructura es:

Descripción	# de octetos	tiempo	tiempo total
Preámbulo	12 octetos	480 us	
Cabecera sector	14 octetos	560 us	1040 us
Checksum	2 octetos	80 us	1120 us
Hueco 1	>2840 us (3600-10%-400(longitud preámbulo))	2840 us	3960 us
Preámbulo	12 octetos	480 us	
Oct. form.(EE)	610 octetos	24400 us	24880 us
Checksum	2 octetos	80 us	24960 us
Hueco 2	>2840 us	2840 us	27800 us
	Total	**31760 us**	

La extensión está diseñada para cubrir un 10% más del tiempo nominal desde el principio del hueco 1 al final del sector estándar (2620 us).
A esto se le debe añadir el tiempo que se le ha acortado al Hueco 1 (760 us) más el tiempo incierto (160 us) que totaliza 3440 us, el equivalente a 86 octetos.

Tipos de error

Estos son los errores más comunes cuando se usan los microdrives:

1. errores temporales de lectura

Este tipo de errores de lectura pueden ser causados por el ruido o las interferencias causadas por transiciones falsas durante la lectura de un bloque. Los ruidos pueden ser cau-

sados por partícula de suciedad que distorsiona temporalmente la señal cuando pasa la cinta por la cabeza de lectura. Cuando la cinta vuelve a pasar, la suciedad se habrá ido(si tenemos suerte) y se podrán leer los datos correctamente. Estos errores de lectura pueden ser causados también cuando los datos se han grabado mal y la señal de '1' se puede confundir con la de '0'. Estos errores se pueden recuperar normalmente - si no se pueden recuperar, será un error permanente.

2. Errores de temporales de escritura

Los errores temporales de escritura ocurren cuando no se puede volver a leer un bloque que se ha grabado, pero reescribiendolo, el bloque se hace legible.

3. Errores no detectados

Es posible que ocurran errores de lectura, pero que el 'checksum' esté correcto. La condición de error no puede ser detectada y se leen datos erróneos en memoria.

4. Errores permanentes

Los errores permanentes ocurren cuando hay defectos en la cinta (empalmes, suciedad o la capa magnética dañada). El procedimiento de formateo está diseñado para evitar el uso de las partes de la cinta sospechosas.

5. Entre unidades

Si un microdrive se ha formateado o escrito en una unidad, puede darse el caso de que no se pueda leer en otra.

6. Degradación

Después de un tiempo de uso de un cartucho, comienzan a incrementarse el número de errores temporales y permanentes. Esto es análogo a la degradación de la calidad del sonido reproducido en un cinta de casette que ha sido usada muchas veces. En este caso, los datos que queden deben ser copiados a un cartucho nuevo y desechar el viejo.

Precauciones con los errores

Se deben tener las siguientes precauciones para asegurarse que los datos se han almacenado correctamente en los microdrives:

1. Mapa de sectores en error

Con el procedimiento de formateo se crea un mapa de los sectores buenos. Cuando se formatea un cartucho, se escribe un bloque inválido dentro de cada secotr. Después se verifican los sectores. Se han tomado muchas precauciones para que este proceso sea lo más fiable posible.

a) Cuando se escriben los bloques de prueba, se escribe un bloque que es más largo que la longitud del bloque estándar. Estos bloques aseguran que si se reescribe un bloque en una unidad con velocidad de motor diferente, la longitud total ocupada por el nuevo bloque, ha sido comprobada.

b) Cada sector es comprobado dos veces. Si ocurre un error en cualquiera de las comprobaciones, el sector en el que ha ocurrido será eliminado.

c) El proceso de formateo escribe 255 sectores en orden decreciente. Los números bajos se escriben sobre los altos. Finalmente, cuando se escribe el sector 0, se encuentra cerca de él el sector más alto que no ha sido reescrito. Si estos dos sectores están muy cerca, puede causar problemas, por lo tanto el sector más alto disponible es eliminado también para asegurarse una buena fiabilidad.

2. Verificación de bloques

Cada bloque que se escribe, es verificado cuando pasa de nuevo por la cabeza (aproximadamente después de 7 segundos). Esto significa que los datos escritos permanecen en la memoria hasta que hayan sido verificados. Si falla la verificación, el bloque se vuelve a escribir en la misma posición.

3. 'Checksum'

Hay un 'checksum' de 16 octetos por cada cabecera y bloque de datos. Por lo tanto hay una posibilidad entre 65535 de que pase inadvertido un error de lectura.

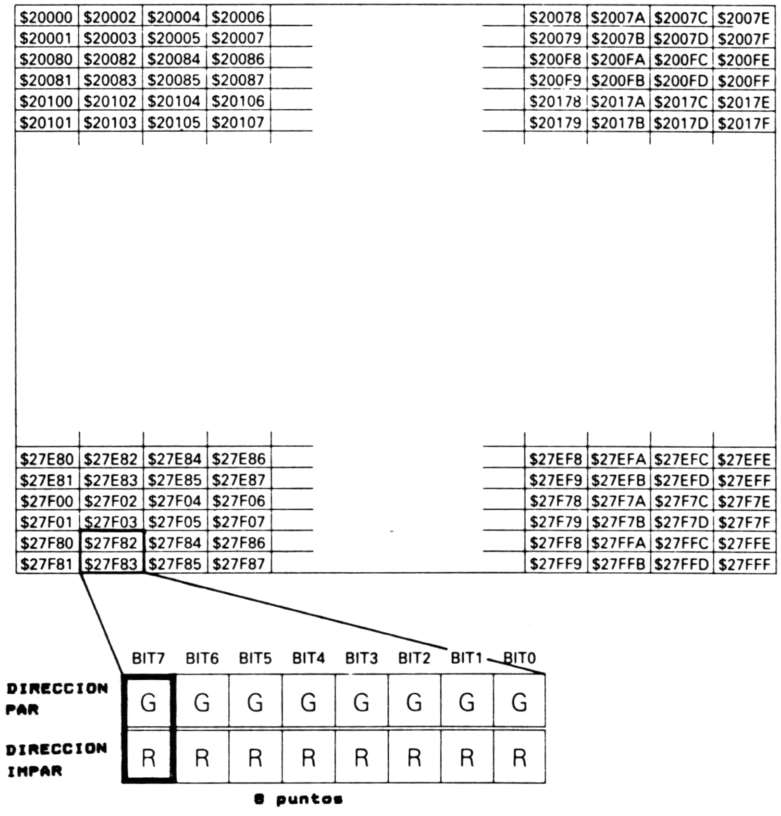
4. Reintentos

El sistema operativo abandona la transacción, solamente, después de 8 pasadas (detectadas en el sector 0) sin que se ejecute la operación satisfactoriamente. La operación de lectura se reintenta 8 veces, y la de escritura 4.

Apéndice E - Descripción de la pantalla

MODE 4

Gráficos 512x256
Colores 4



R= **ROJO**
G= **VERDE**

MODE 8

Gráficos 256x256
Colores 8 + parpadeo

$20000	$20002	$20004	$20006							$20078	$2007A	$2007C	$2007E
$20001	$20003	$20005	$20007							$20079	$2007B	$2007D	$2007F
$20080	$20082	$20084	$20086							$200F8	$200FA	$200FC	$200FE
$20081	$20083	$20085	$20087							$200F9	$200FB	$200FD	$200FF
$20100	$20102	$20104	$20106							$20178	$2017A	$2017C	$2017E
$20101	$20103	$20105	$20107							$20179	$2017B	$2017D	$2017F
$27E80	$27E82	$27E84	$27E86							$27EF8	$27EFA	$27EFC	$27EFE
$27E81	$27E83	$27E85	$27E87							$27EF9	$27EFB	$27EFD	$27EFF
$27F00	$27F02	$27F04	$27F06							$27F78	$27F7A	$27F7C	$27F7E
$27F01	$27F03	$27F05	$27F07							$27F79	$27F7B	$27F7D	$27F7F
$27F80	$27F82	$27F84	$27F86							$27FF8	$27FFA	$27FFC	$27FFE
$27F81	$27F83	$27F85	$27F87							$27FF9	$27FFB	$27FFD	$27FFF

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
DIRECCION PAR	G	F	G	F	G	F	G	F
DIRECCION IMPAR	R	B	R	B	R	B	R	B

4 puntos

R=ROJO
G=VERDE
B=AZUL
F=PARPADEO

Apéndice F - Especificaciones las expansiones 'hardware'

Este apéndice explica la filosofía básica de la expansión 'hardware' del QL. Esta información se puede ampliar con un buen libro sobre interconexiones con el microprocesador 68008. Los detalles sobre la expansión de la ROM se incluyen en el apéndice C.

Aspectos generales de la expansión

El QL ha sido diseñado con la facilidad de añadir una cantidad considerable de 'hardware'. Este se puede concetar al QL por medio de el conector de expansión que hay en el lateral izquierdo. El acceso está tapado normalmente por una pieza de plastico negro. Si se quita esta tapa, es posible ver un conector DIN-41612 macho de 64 vias. La figura F1 nos muestra un esquema del mismo.

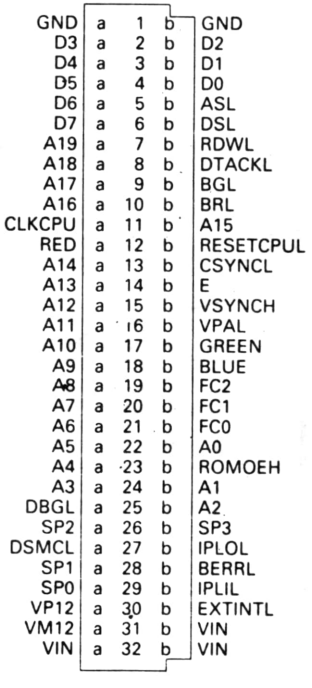


Figura F1 - Señales disponibles en el conector lateral

Las señales que aparecen seguidas de una L está activas cuando están bajas.

Para conocer la descripción de cada una de estas señales y su forma de operación, se debe consultar un libro de 68008.

Todas las señales del 68008 salen por el conector de expansión, por lo que es posible conectar cualquier tipo de periférico para el QL. Estos periféricos se colocarán en tarjetas enchufables. Solamente se puede añadir una de estas tarjetas al QL estándar. Para permitir una mayor flexibilidad, se dispondrá pronto de un módulo de expansión, que proporcione varias ranuras para poder conectar distintos tipos de periféricos a medida que estén disponibles.

Descripción de las señales

Señales de salida

A0 a A19	Líneas de direccion del 68008
ASL	Dirección de 'strobe' (indica direcciones válidas)
RDWL	Lectura/Escritura
DSL	'Strobe' de los datos
BGL	Permiso para las líneas de trasmisión
DSMCL	'Strobe' de datos *Sostenido* (chip maestro)
CLCKCPU	Reloj de la CPU
E	Señal de habilitación de periféricos 6800
RED	Salida vídeo rojo
BLUE	Salida vídeo azul
GREEN	Salida vídeo verde
CSYNCL	Señal de sincronismo del vídeo compuesto
VSYNCH	Señal de sincronismo vertical
ROMOEH	Salida de ROM habilitada
FC0	Estado del procesador
FC1	Estado del procesador
FC2	Estado del procesador
RESETCPUL	Iniciación de la CPU

Señales bidireccionales

D0 a D7	Líneas de transmisión de datos

Señales de entrada

DTACKL	Arbitro de líneas de transmisión
BRL	Petición a las líneas de transmisión
VPAL	Dirección periférica válida
IPL0L	Prioridad de interrupción de nivel 0
IPL1L	Prioridad de interrupción de nivel 1
BERRL	Error de líneas de transmisión
EXTINTL	Interrupción externa
DBGL	Recogida de datos de las líneas de transmisión

Misceláneas

SP0 a SP3	Líneas de selección de periféricos (hasta 16)
VIN	9 V DC nominales (7 V mínimo) a 500mA máximo
VM12	-12 V
VP12	+12 V
GND	0 V (tierra de señales y alimentación)

Tipos de tarjetas de expansión

Hay dos tipos de tarjetas de expansión, de memoria y de periféricos.

El mapa de memoria entre las direcciones $40000 y BFFFF ha sido asignada como memoria adicional (por lo tanto se puede añadir hasta medio 'megabyte' (512K octetos)). Esta memoria es contigua desde $40000 hacia arriba. Solamente se puede conectara un tiempo una de estas tarjetas de memoria adicional. Cuando se enciende el QL, el QDOS comprueba si está disponible esta memoria.

Las tarjetas periféricas son muy interesantes. Cada una puede contener una ROM con controladores (ver sección 9) para los diferentes dispositivos que contiene la tarjeta. Todos los controladores tienen posiciones independientes, debido a la forma en que se asignan las posiciones de memoria a las diferentes tarjetas.

Asignando memoria a las tarjetas de expansión

El conector del QL tiene cuatro líneas de señal, SP0, SP1, SP2 y SP3. Todas ellas tiene su masa conectada a la tarjeta principal. Cada tarjeta de expansión contiene un circuito que produce un número diferente. Los circuitos de cada tarjeta deben comparar la dirección en SP0-SP3 con la que hay en las líneas A14-A17. Si son iguales, y está seleccionada el área de memoria de la tarjeta, la tarjeta es seleccionada. Cada tarjeta ocupa un hueco de 16K octetos en la memoria comprendida entre las direcciones $C000+4000*n, donde n es el número introducido en cada tarjeta particular. De esta forma se puede comprender porqué cada tarjeta tiene su posición independiente. Cuando se conectan varias tarjetas, se pueden colocar en cualquier lugar de la parte superior de la memoria.

Interconexión con las líneas de datos

El 68008 no tiene un control completo de todas las líneas de transmisión. Cuando se transfiere información desde la memoria al circuito de vídeo, el ULA del retiene las líneas usando la señal DTACKL. El acceso a memoria se controla solamente con DSL, y no se usa ASL.

El reparto del mapa de memoria se realiza por los estados lógicos de las líneas altas de dirección. Para ayu- darle a comprender la decodificación necesaria, vea la tabla que le damos a continuación:

A19	A18	A17	A16	A15	A14	Rango direc.	
0	0	0	0	0	X	00000-07FFF	ROM sistema (32K prim.)
0	0	0	0	1	0	08000-9BFFF	ROM sistema (16K exp.)
0	0	0	0	1	1	0C000-0FFFF	ROM enchuf. hasta 16K
0	0	0	1	0	X	10000-17FFF	Reservadas
0	0	0	1	1	0	18000-1BFFF	Registro internos y ES
0	0	0	1	1	1	1C000-1FFFF	128K RAM estándar
0	0	1	X	X	X	20000-3FFFF	Reservadas
0	1	X	X	X	X	40000-7FFFF	256K RAM de expansión
1	0	X	X	X	X	80000-BFFFF	256K RAM de expansión
1	1	X	X	X	X	C0000-FFFFF	Expansión 'hardware'

En un QL no expandido las líneas A18 y A19 no se usan nunca. Cuando se añaden periféricos, deben inhabilitar los circuitos principales del QL mientras reconoce las direccio- nes válidas. Esto se realiza poniendo *alta* la señal DSMCL. Se recomienda que se use para esto un transistor NPN de conmutación rápida.

A todas las señales de las líneas se pueden conectar dos TTL de baja potencia, como máximo. El periférico debe sumi- nistrar las señales DTACKL (reconocimiento de transferencia de datos) o VPAL (dirección periférica válida) segú se requiera, y debe inhabilitar DSMCL ('strobe' de datos del chip maestro).

El módulo de expansión debe proporcionar memoria intermedia extra de líneas. La memoria intermedia de datos se habilita cuando A18 y A19 están a 1, o si DGBL (señal de recogida de datos) es puesta a 1 por cualquier tarjeta periférica. Si se usa DBGL, se debe reforzar por un circuito de colector abierto. Cuando se conecta el módulo de expansión, se pone a cero RESETCPUL mientras se enciende el módulo de expansión. Si se apaga el módulo de expansión, RESETCPUL se pone a 0.

Apéndice G - Códigos de error

Nombre		
ERR.NC	-1	No completado
ERR.NJ	-2	Job inválido
ERR.OM	-3	No hay memoria
ERR.OR	-4	Fuera de rango
ERR.BO	-5	Memoria intermedia llena
ERR.NO	-6	No se encuentra el canal
ERR.NF	-7	No se encuentra
ERR.EX	-8	Ya existe
ERR.IU	-9	En uso
ERR.EF	-10	Fin de fichero
ERR.DF	-11	Unidad llena
ERR.BN	-12	Nombre inválido
ERR.TE	-13	Error de transmisión
ERR.FF	-14	Falló el formateo
ERR.BP	-15	Parámetro inválido
ERR.FE	-16	Medio erróneo
ERR.XP	-17	Error en la expresión
ERR.OV	-18	Rebasamiento
ERR.NI	-19	No implementado (aún)
ERR.RO	-20	Solo lectura
ERR.BL	-21	Línea mala

Apéndice H - Sumario de. TRAPS

De los 16 TRAPs disponibles, del 0 al 4 han sido reservados para usarlos con el QDOS, los otros, del 5 al 15, se pueden definir para aplicaciones de usuario. Para una información más detallada acerca de los TRAPs, vea los capítulos 5 a 7.

TRAP#0

Este TRAP se usa para cambiar de modo usuario a modo supervisor. Ningún registro se ve afectado por este TRAP.

TRAP#1 - TRAPs del Gestor

D0	Nombre	Descripción.
00	MT.INF	Obtiene información del sistema y job actual
01	MT.CJOB	Crea job en el área de programas transitorios
02	MT.JINF	Proporciona información de un job
04	MT.RJOB	Borra job del área de programas transitorios
05	MT.FRJOB	Fuerza el borrado de un job del área de programas transitorios
06	MT.FREE	Busca el mayor espacio libre en el área de programas transitorios
07	MT.TRAPV	Pone un vector de TRAP para un job
08	MT.SUSJB	Suspende un job
09	MT.RELJB	Libera un job
0A	MT.ACTIV	Activa un job
0B	MT.PRIOR	Cambia la prioridad de un job
0C	MT.ALLOC	Asigna espacio en el área común
0D	MT.LNKFR	Encadena espacio libre(devuelto) en el área común
0E	MT.ALRES	Asigna área de procedimientos residentes
0F	MT.RERES	Libera área de procedimientos residentes
10	MT.DMODE	Pone y lee el modo de pantalla
11	MT.IPCOM	Manda comandos al IPC
12	MT.BAUDK	Pone la velocidad de transmisión
13	MT.RCLCK	Lee el reloj
14	MT.SCLCK	Pone en hora el reloj
15	MT.ACLCK	Ajusta el reloj
16	MT.ALBAS	Asigna área para el programa BASIC
17	MT.REBAS	Libera área del programa BASIC

18	MT.ALCHP	Asigna área común
19	MT.RECHP	Libera área común
1A	MT.LXINT	Conecta una rutina de servicio de interrupciones al QDOS
1B	MT.RXINT	Borra una rutina de servicio de interrupciones del QDOS
1C	MT.LPOLL	Conecta al QDOS una rutina de servicio de 'polling' de 50/60Hz
1D	MT.RPOLL	Borra del QDOS una rutina de servicio de 'polling' de 50/60Hz
1E	MT.LSCHD	Conecta al QDOS una tarea de bucle del planificador
1F	MT.RSCHD	Borra del QDOS una tarea de bucle del planificador
20	MT.LIOD	Conecta al QDOS un controlador de dispositivos
21	MT.RIOD	Borra del QDOS un controlador de dispositivos
22	MT.LDD	Conecta al QDOS un controlador de dispositivos de directorio
23	MT.RDD	Borra del QDOS un controlador de dispositivos de directorio

TRAP#2 - Asignación de E/S

D0	Nombre	Descripción
01	IO.OPEN	Abre un canal
02	IO.CLOSE	Cierra un canal
03	IO.FORMT	Formatea un medio por sectores
04	IO.DELET	Borra un fichero

TRAP#3 - Utilización de E/S

D0	Nombre	Descripción
00	IO.PEND	Comprueba una entrada pendiente
01	IO.FBYTE	Recoje un octeto
02	IO.FLINE	Recoje una línea de caracteres terminados con un <LF>
03	IO.FSTRG	Recoje una cadena de octetos
04	IO.EDLIN	Edita una línea de caracteres (solo consola)
05	IO.SBYTE	Manda un octeto
07	IO.SSTRG	Manda una cadena de octetos
09	SD.EXTOP	Llama a una operación extendida
0A	SD.PXENQ	Obtiene el tamaño de una ventana y la posición de su cursor (en puntos)

0B	SD.CHENQ	Obtiene el tamaño de una ventana y la posición del cursor (en caracteres)
0C	SD.BORDR	Pone el ancho y color del borde
0D	SD.WDEF	Redefine una ventana
0E	SD.CURE	Habilita el cursor
0F	SD.CURS	Suprime el cursor
10	SD.POS	Posiciona el cursor (coordenadas de caracteres)
11	SD.TAB	Posiciona el cursor en una columna de caracteres
12	SD.NL	Posiciona el cursor en una línea nueva de carcateres
13	SD.PCOL	Posiciona el cursor en el carácter anterior
14	SD.NCOL	Posiciona el cursor en el siguiente carácter
15	SD.PROW	Posiciona el cursor en la línea anterior
16	SD.NROW	Posiciona el cursor en la línea siguiente
17	SD.PIXP	Posiciona el cursor por coordenadas de puntos
18	SD.SCROL	Hace 'scroll' de una ventana
19	SD.SCRTP	Hace 'scroll' de la parte superior de una ventana
1A	SD.SCRBT	Hace 'scroll' de la parte inferior de una ventana
1B	SD.PAN	Mueve una ventana lateralmente
1E	SD.PANLN	Mueve lateralmente la línea del cursor
1F	SD.PANRT	Mueve lateralmente el final de la línea del cursor
20	SD.CLEAR	Limpia toda una ventana
21	SD.CLRTP	Limpia la parte superior de una ventana
22	SD.CLRBT	Limpia la parte inferior de una ventana
23	SD.CLRLN	Limpia la línea del cursor
24	SD.CLRRT	Limpia la parte derecha de la línea del cursor
25	SD.FOUNT	Pone o quita el tipo de caracteres
26	SD.RECOL	Cambia los colores de una ventana
27	SD.SETPA	Pone el color del papel
28	SD.SETST	Pone el color del punteado
29	SD.SETIN	Pone el color de la tinta
2A	SD.SETFL	Pone el modo parpadeo
2B	SD.SETUL	Pone el modo subrayado
2C	SD.SETMD	Pone el modo en carácter o dibujo
2D	SD.SETSZ	Pone el tamaño y espacio del carácter
2E	SD.FILL	Rellena un bloque rectangular en una ventana
30	SD.POINT	Dibuja un punto
31	SD.LINE	Dibuja una línea
32	SD.ARC	Dibuja un arco
33	SD.ELLIPS	Dibuja una elipse
34	SD.SCALE	Pone la escala de la ventana
35	SD.FLOOD	Pone o quita el formato de relleno
36	SD.GCUR	Define la posición del cursor gráfico

40 FS.CHECK Comprueba todas las operaciones pendientes en un fichero .
41 FS.FLUSH Libera la memoria intermedia de un fichero
42 FS.POSAB Posiciona el apuntador absoluto de un fichero
43 FS.POSRE Posiciona el apuntador del fichero, relativo a la posición anterior
45 FS.MDINF Obtiene información sobre el medio
46 FS.HEADS Pone cabecera de fichero
47 FS.HEADR Lee la cabecera de un fichero
48 FS.LOAD Carga un fichero en memoria
49 FS.SAVE Salva un fichero desde memoria

TRAP#4 - Especial para BASIC

Este TRAP es especial para el intérprete de comandos del BASIC. Hace que las direcciones que se pasan a los TRAPs de E/S, sean relativos a A6. Se usa para preceder a los TRAPs #2 y #3. Su efecto se cancela por los TRAPs #2 y #3. Para el TRAP#2, a la entrada se añade A6 a A0. Para el TRAP#3, se añade A6 a A1 y se quita a la salida. El TRAP#4 no tiene parámetros, no devuelve errores y preserva todos los registros. El TRAP#4 no se cancela con el TRAP#3 que devuelve el error NO (no abierto).

TRAP#5 - #15

Estos TRAPs no los usa el QDOS y se pueden asignar a aplicaciones del usuario

Apéndice I - Sumario de Utilidades por Vector

Vec.	Nombre	Tipo	Descripción
C0	MM.ALCHP	SM	Asigna espacio en el área común
C2	MM.RECHP	SM	Libera espacio en el área común
C4	UT.WINDW	ST	Prepara una ventana con un nombre dado
C6	UT.CON	ST	Prepara una véntana de consola
C8	UT.SCR	ST	Prepara una ventana de pantalla
CA	UT.ERRO	ST	Escribe un mensaje de error en el canal 0
CC	UT.ERR	ST	Escribe un mensaje de error en un canal
CE	UT.MINT	ST	Convierte un entero a ASCII
D0	UT.MTEXT	ST	Manda un mensaje a un canal
D2	UT.LINK	GU	Conecta un item a una lista
D4	UT.UNLNK	GU	Desconecta un item de una lista
D8	MM.ALLOC	GU	Asigna espacio en el área de usuario
DA	MM.LNKFR	GU	Conecta espacio libre (devuelto) al área de usuario
DC	IO.QSET	GU	Prepara una cola
DE	IO.QTEST	GU	Comprueba el estado de una cola
E0	IO.QIN	GU	Pone un octeto en una cola
E2	IO.QOUT	GU	Extrae un octeto de una cola
E4	IO.QEOF	GU	Pone el fin de fichero a una cola
E6	UT.CSTR	BU	Comapara dos cadenas
E8	IO.SERQ	SM	Manejo directo de la cola de E/S
EA	IO.SERIO	SM	Manejo general de E/S serie
EC	CN.DATE	BU	Obtiene el dia y la hora
EE	CN.DAY	BU	Obtiene el dia de la semana
F0	CN.FTOD	BU	Convierte coma flotante en ASCII
F2	CN.ITOD	BU	Convierte un entero a ASCII
F4	CN.ITOBB	BU	Convierte un octeto binario a ASCII
F6	CN.ITOBW	BU	Convierte una palabra binaria a ASCII
F8	CN.ITOBL	BU	Convierte una palabra binaria larga a ASCII
FA	CN.ITOHB	BU	Convierte un octeto hexadecimal a ASCII
FC	CN.ITOHW	BU	Convierte una palabra hexadecimal a ASCII
FE	CN.ITOHL	BU	Convierte una palabra larga hex a ASCII
100	CN.DTOF	BU	Convierte ASCII en coma flotante
102	CN.DTOI	BU	Convierte ASCII en un entero
104	CN.BTOIB	BU	Convierte ASCII en un octeto binario
106	CN.BTOIW	BU	Convierte ASCII en una palabra binaria
108	CN.BTOIL	BU	Convierte ASCII en palabra binaria larga
10A	CN.BTOIB	BU	Convierte ASCII en un octeto hexadecimal
10C	CN.HTOIW	BU	Convierte ASCII en una palabra hexadecimal
10E	CN.HTOIL	BU	Iniciación de procedimientos BASIC
110	BP.INIT	BP	Obtiene enteros (palabras)
112	CA.GTINT	BP	Obtiene coma flotante (6 octetos)
114	CA.GTFP	BP	Obtiene cadenas
116	CA.GTSTR	BP	Obtiene enteros largos (palabras largas)

```
118 CA.GTLIN BP  Obtiene enteros largos (palabra larga)
11A BV.CHRIX BP  Reserva espacio en la pila aritmética
11C RI.EXEC  AR  Ejecuta una operación aritmética
11E RI.EXECB AR  Ejecuta lista de operaciones aritméticas
120 BP.LET   BP  Retorna valores de parámetros
122 IO.NAME  GU  Decodifica un nombre de dispositivo
124 MD.READ  MD  Lee un sector de un microdrive
126 MD.WRITE MD  Escribe un sector a un microdrive
128 MD.VERIN MD  Verifica un sector de un microdrive
12A MD.SECTR MD  Lee una cabecera de sector de un microdrive
```

Claves del código de tipo

SM = Gestión del sistema
ST = TRAPs simplificados
GU = Utilidad general
BU = Utilidad BASIC
BP = Soporte de procedimientos BASIC
AR = Operaciones del paquete aritmético
MD = Utilidades de los microdrives

Apéndice J - Apuntadores a las variables del SuperBASIC

```
BV.START  EQU $00 Principio de los apuntadores
BV.BFBAS  EQU $00 Base de la memoria intermedia
BV.BFP    EQU $04 Apuntador a la memoria intermedia en uso
BV.TKBAS  EQU $08 Lista de 'token'
BV.TKP    EQU $0C
BV.PFBAS  EQU $10 Fichero de programa
BV.PFP    EQU $14
BV.NBAS   EQU $18 Tabla de nombres
BV.NTP    EQU $1C
BV.NLBAS  EQU $20 Lista de nombres
BV.NLP    EQU $24
BV.VVBAS  EQU $28 Valores de variables
BV.VVP    EQU $2C
BV.CHBAS  EQU $30 Nombres de canales
BV.CHP    EQU $34
BV.RTBAS  EQU $38 Tabla de retorno
BV.RTP    EQU $3C
BV.LNBAS  EQU $40 Tabla de números de líneas
BV.LNP    EQU $44

BV.CHANGE EQU $48 Marcador de cambio de dirección

BV.BTP    EQU $48 Pila de retroceso durante análisis
BV.BTBAS  EQU $4C
BV.TGP    EQU $50 Pila temporal de gráficos durante análisis
BV.TGBAS  EQU $54
BV.RIP    EQU $58 Pila aritmética
BV.RIBAS  EQU $5C
BV.SSP    EQU $60 Pila del sistema
BV.SSBAS  EQU $64

BV.ENDPT  EQU $64 Fin de apuntadores

BV.LINUM  EQU $68 Número de la línea actual (palabra)
BV.LENGTH EQU $6A Longitud actual (palabra)
BV.STMNT  EQU $6C Sentencia dentro de línea actual (octeto)
BV.CONT   EQU $6D CONTINUE ($80) o STOP (0) del proceso
                  (octeto)
BV.INLIN  EQU $6E Procesando una clausula en línea o no
                  (octeto)

BV.SING   EQU $6F Ejecución de línea sencilla SI($FF) o
                  NO (0)
BV.INDEX  EQU $70 Columna en la tabla de nombres del último
                  índice 'inlin lp' leido (palabra)
```

```
BV.VVFREE EQU $72 Primer espacio libre en VVTABLE (larga)
BV.SSSAV  EQU $76 Sp salvado del último 'memoria no
                  disponible' para volver (larga)

BV.RAND   EQU $80 Número aleatorio (larga)
BV.COMCH  EQU $84 Canal de comando (larga)

BV.NXLIN  EQU $88 Número de línea donde continuar (palabra)
BV.NXSTM  EQU $8A Sentencia donde continuar (octeto)
BV.COMLN  EQU $8B Línea de comando salvada ($FF) o no (0)
                  (octeto)
BV.STOPN  EQU $8C Número de parada puesto (palabra)
BV.EDIT   EQU $8E El programa ha sido editado ($FF) o no (0)
                  (octeto)
BV.BRK    EQU $8F Ha habido un 'BREAK' (0) o no ($80)
                  (octeto)
BV.UNRVL  EQU $90 Se necesita descifrar ($FF) o no (0)
                  (octeto)
BV.CNSTM  EQU $91 Sentencia desde donde CONTinuar (palabra)

BV.CNLND  EQU $92 Línea desde donde CONTinuar (palabra)

BV.DALNO  EQU $94 Número actual de línea DATA (palabra)
BV.DASTM  EQU $96 Número actual de sentencia DATA (octeto)
BV.DAITM  EQU $97 Siguiente item de DATA a leer (octeto)

BV.CNIND  EQU $98 Indice de bucle en línea con el que
                  CONTinuar (palabra)
BV.CNINL  EQU $9A Indicador de bucle en línea para
                  CONTINUE (octeto)
BV.LSANY  EQU $9B Comprobar lista ($FF) o no (0) (octeto)
BV.LSBEF  EQU $9C Línea superior invisible (palabra)
BV.LSBAS  EQU $9E Ultima línea en la ventana (palabra)
BV.LSAFT  EQU $A0 Línea inferior invisible (palabra)
BV.LENLN  EQU $A2 Longitud de la línea de ventana (palabra)
BV.MAXLN  EQU $A4 Número máximo de líneas en la ventana
                  (palabra)
BV.TOTLN  EQU $A6 Número de líneas que quedan en la ventana
                  (palabra)

BV.AUTO   EQU $AA AUTO/EDIT si ($FF) o no (0) (octeto)
BV.PRINT  EQU $AB Imprimir desde PRTOK ($FF) o dejarlo en la
                  memoria intermedia (0) (octeto)
BV.EDLIN  EQU $AC Número de la línea a editar (palabra)
BV.EDINC  EQU $AE Rango de incremento en EDIT (palabra)

BV.TKPOS  EQU $B0 Posición de A4 en lista TK a la entrada
                  del procedimiento (larga)
```

```
BV.PTEMP  EQU $B4 Apuntador temporal para GO.PROC (larga)
BV.UNDO   EQU $B8 deshacer pila RT y rehacer después el
                  procedimiento (octeto)

BV.ARROW  EQU $B9 flecha abajo ($FF), arriba (01) o ninguna
                  (0) (octeto)

BV.LSFIL  EQU $BA Rellenar la ventana hasta aquí por lo
                  menos, cuando se liste

BV.END    EQU $100 Límite superior del área de BV
```

Apéndice K - Tipos de 'Token'

Estos son todos los *tipos de 'token'* que se pueden encontrar en un programa SuperBASIC resumido.

Token	Valor	Descripción
SPC.B	$80	espacio
KEY.B	$81	palabra clave
BIP.B	$82	procedimiento interno
BIF.B	$83	función interna
SYM.B	$84	símbolo
OPS.B	$85	símbolo de operación
MON.B	$86	símbolo de operación unitaria
SYV.B	$87	variable del sistema
NAM.B	$88	nombre
SHI.B	$89	entero corto
LGI.B	$8A	entero largo
STR.B	$8B	cadena
TXT.B	$8C	texto
LNO.B	$8D	número de línea
SEP.B	$8E	separador
FP.B	$F0	número de coma flotante

Apéndice L - Códigos de operación del paquete aritmético8

Las operaciones aritméticas se pueden realizar con el *paquete aritmético*. Para acceder a él, se usan las rutinas de utilidad RI.EXEC (vector $11C) y RI.EXECB (vector $11E). El paquete de coma flotante acepta dos tipos de códigos de operación. Los códigos entre $02 y $30 son operaciones aritméticas reales. Los otros códigos se extienden a una palabra negative entre $FFFF y $FF31 enclusive, y son instrucciones de carga y almacenamiento de números de coma flotante con las direcciones dadas por (A6.L+A4.L+((codigo operación OR $FF00) AND $FFFE). Si el bit 0 del código de operación está a cero, la operación es de carga, si lo tiene a uno, la operación es de almacenamiento. La carga hace que A1 se decremente en 6 octetos y los de salvar hacen que se incremente en 6. El NOS viejo se convierte en el TOS nuevo.

Códigos de operación del paquete aritmético

Cód.	Nombre	Cambio de A1	Operación
00	RI.TERM		
02	RI.NINT	+4	Busca el entero más cercano a TOS
04	RI.INT	+4	Trunca TOS en un entero
06	RI.NLINT	+2	Entero largo más cercano a TOS
08	RI.FLOAT	-4	Convierte TOS de entero a coma flotante
0A	RI.ADD	+6	Suma TOS a NOS
0C	RI.SUB	+6	Resta TOS de NOS
0E	RI.MULT	+6	Multiplica TOS por NOS
10	RI.DIV	+6	Divide TOS entre NOS
12	RI.ABS	0	Pone TOS en positivo
14	RI.NEG	0	niega TOS
16	RI.DUP	-6	duplica TOS
18	RI.COS	0	Saca el coseno de TOS
1A	RI.SIN	0	Saca el seno de TOS
1C	RI.TAN	0	Saca la tangente de TOS
1E	RI.COT	0	Saca la cotangente de TOS
20	RI.ASIN	0	Saca el arco seno de TOS
22	RI.ACOS	0	Saca el arco coseno de TOS
24	RI.ATAN	0	Saca el arco tangente de TOS
26	RI.ACOT	0	Saca el arco cotangente de TOS
28	RI.SQRT	0	Extrae la raíz cuadrada de TOS
2A	RI.LN	0	Saca el logaritmo natural de TOS
2C	RI.LOG10	0	Saca el logaritmo en base 10 de TOS
2E	RI.EXP	0	Saca el exponencial de TOS
30	RI.POWFP	+6	Eleve NOS a la potencia TOS

00	RI.LOAD	Clave operando, bit 0=0 si es cargar
01	RI.STORE	Clave operando, bit 0=1 si es salvar

Apéndice M - Definición de Variables del Sistema Operativo

```
SV.BASE  EQU $28000 Base de las variables del sistema

SV.IDENT EQU $00    identificación (palabra)
SV.CHEAP EQU $04    base del área común (larga)
SV.CHPFR EQU $08    primer espacio libre en el área común
                    (larga)
SV.FREE  EQU $0C    base del área libre (larga)
SV.BASIC EQU $10    base de la pila de BASIC (larga)
SV.TRNSP EQU $14    base del área de programas transitorios
                    (larga)
SV.TRNFR EQU $18    primer espacio libre en el área de
                    programas transitorios (larga)
SV.RESPR EQU $1C    base del área de programas residentes
                    (larga)
SV.RAMT  EQU $20    tope de la RAM (+1)(larga)

SV.RAND  EQU $2E    número aleatorio (palabra)
SV.POLLM EQU $30    contador de interrupciones de 'poll'
                    perdidas (palabra)
SV.TVMOD EQU $32    0 si no tiene TV (octeto)
SV.SCRST EQU $33    estado de la pantalla  0 = activa)
SV.MCSTA EQU $34    valor actual del registro de estado MC
                    (octeto)
SV.PCINT EQU $35    valor actual del registro de interrupción
                    PC (octeto)

SV.NETNR EQU $37    número de estación en la red de datos
                    (octeto)

SV.I2LST EQU $38    apuntador a lista de interrupciones 2
                    controladores
SV.PLIST EQU $3C    apuntador a lista de tareas conectadas
SV.SHLST EQU $40    apuntador a tareas del planificador
                    (larga)
SV.DRLST EQU $44    apuntador a lista de controladores de
                    dispositivo (larga)
SV.DDLST EQU $48    apuntador a lista de controladores de
                    dispositivos de directorio (larga)
SV.KEYQ  EQU $4C    apuntador a la cola del teclado (larga)
SV.TRAPV EQU $50    apuntador a la tabla de redirecciona-
                    miento de TRAPs (larga)

 Apuntadores a las tablas de gestión de recursos

SV.BTPNT EQU $54    apuntador a la entrada de bloque
                    dependiente más reciente (larga)
```

```
SV.BTBAS EQU $58    apuntador a la base de la tabla de
                    bloques dependientes (larga)
SV.BTTOP EQU $5C    apuntador a lo alto de la tabla de
                    bloques dependientes

SV.JBTAG EQU $60    valor actual de la etiqueta de Job
                    (palabra)
SV.JBMAX EQU $62    número mayor de Job actual (palabra)
SV.JBPNT EQU $64    apuntador a la entrada actual de la tabla
                    de Job (larga)
SV.JBBAS EQU $68    apuntador a la base de la tabla de Job
                    (larga)
SV.JBTOP EQU $6C    apuntador a lo alto de la tabla de Job
                    (larga)

SV.CHTAG EQU $70    valor actual de la etiqueta de canal
                    (palabra)
SV.CHMAX EQU $72    número mayor de canal actual (palabra)
SV.CHPNT EQU $74    apuntador al último canal comprobado
                    (larga)
SV.CHBAS EQU $78    apuntador a la base de la tabla de
                    canales (larga)
SV.CHTOP EQU $7C    apuntador a lo alto de la tabla de
                    canales (larga)

SV.CAPS  EQU $88    mayúsculas (palabra)
SV.ASBUF EQU $8A    memoria de autorrepetición (palabra)
SV.ARDEL EQU $8C    tiempo de autorrepetición (palabra)
SV.ARFRQ EQU $8E    autorrepetición 1/frecuencia (palabra)
SV.ARCNT EQU $90    contador de autorrepetición (palabra)
SV.CQCH  EQU $92    código de caracter para cola de cambio
                    del teçlado (palabra)

SV.WP    EQU $94    protección de escritura (palabra)
SV.SOUND EQU $96    estado del sonido (palabra)
SV.SER1C EQU $98    dirección de la cola de recepción para el
                    canal 1 (larga)
SV.SER2C EQU $9C    dirección de la cola de recepción para el
                    canal 2 (larga)
SV.TMODE EQU $A0    modo de transmisión ZX8032 (incluye
                    velocidad de transmisión) (octeto)

SV.CSUB  EQU $A2    subrutina donde saltar en caso de
                    CAPSLOCK (larga)
SV.TIMO  EQU $A6    tiempo de espera para modo de transmisión
                    conmutada (palabra)
SV.TIMOV EQU $A8    valor de tiempo de espera de conmutación
                    (2 caracteres) (palabra)
```

```
SV.FSTAT EQU $AA    estado del cursor parpadeante (palabra)

SV.MDRUN EQU $EE    que unidad está funcionando? (octeto)
SV.MDCNT EQU $EF    contador de funcionamiento parada de
                    microdrive (octeto)
SV.MDDID EQU $F0    ID de la unidad * 4 por cada microdrive
                    (8 octetos)
SV.MDSTA EQU $F8    estado 0=no hay operaciones pendientes (8
                    octetos)
SV.FSDEF EQU $100   apuntadores a las definiciones físicas de
                    ficheros del sistema (16 largas)
SV.FSLST EQU $140   apuntadora la lista de definiciones de
                    canales de fichero (larga)

SV.STACB EQU $180   parte inferior de la pila (192 largas)
SV.STACT EQU $480   pila grande - NO se comprueba (hasta
                    aquí)
SV.TRAPO EQU $2*($28+2) desplazamiento de la tabla de
                    vectores de TRAP desde (SV.TRAPV)

SV.IDENT EQU $D254  bits de verde, rojo, azul y negro con
                    parpadeo
```

Definición de controladores de dispositivo (para las variables del pseudo sistema propiedad de los controladores)

```
SV.LXINT EQU $00    conexión al siguiente servicio de
                    interrupción externa (larga)
SV.AXINT EQU $04    dirección del servicio de interrupción
                    externa (larga)
SV.LPOLL EQU $08    conexión al siguiente servicio de
                    interrupción de 'polling' (larga)
SV.APOLL EQU $0C    dirección del servicio de interrupción de
                    'polling' (larga)
SV.LSCHD EQU $10    conector a la siguiente tarea del
                    planificador (larga)
SV.ASCHD EQU $14    dirección de la siguiente tarea del
                    planificador (larga(
SV.LIO   EQU $18    conector al siguiente controlador de ES
                    (larga)
SV.AIO   EQU $1C    dirección de la rutina de ES (larga)
SV.AOPEN EQU $20    dirección de la rutina de abrir canal
                    (larga)
SV.ACLOS EQU $24    dirección de la rutina de cerrar canal
                    (larga)
```

Apéndice N - Bloque de definición de canal

Cada vez que se abre un canal nuevo, se crea un bloque de definición de canal. En este bloque se mantiene toda la información importante sobre el canal, tal como el controlador de dispositivo que se está usando, el Job propietario, el número del canal y su estado.

```
CH.LEN    $00 larga     longitud del bloque de definición
CH.DRIVR  $04 larga     dirección del controlador
CH.OWNER  $08 larga     Job propietario
CH.RFLAG  $0C larga     dirección que se manda cuando se
                        libera espacio
CH.TAG    $10 palabra   ID del canal
CH.STAT   $12 octeto    estado:

                        0 = OK
                        -1 = A1 absoluto
                        $80 = A1 relativo a A6
                        negativo = esperando

CH.ACTN   $13 octeto    acción almacenada para el Job que
                        está esperando
CH.JOBWT  $14 larga     ID del Job esperando por ES
*
* extensión de la definición del canal, para los  directores
* de colas serie
*
CH.QIN    $18 larga     apuntador a la cola de entrada(o cero)
CH.QOUT   $1C larga     apuntador a la cola de salida(o cero)
```

Apéndice P - Bloque de definición de canal de fichero del sistema

Bloque de definición de canal de fichero del sistema

$18	FS.NEXT*	4	conexión al siguiente canal de fichero del sistema
$1C	FS.ACCES*	1	modo de acceso(D3 en llamada OPEN)
$1D	FS.DRIVE*	1	ID de la unidad
$1E	FS.FILNR+	2	número de ficheros en la unidad
$20	FS.NBLOK	2	número del bloque que contiene el siguiente octeto
$22	FS.NBYTE+	2	siguiente octeto del bloque
$24	FS.EBLOK+	2	número del bloque que contiene el octeto después del EOF
$26	FS.EBYTE+	2	octeto después del EOF
$28	FS.CBLOK	4	apuntador a la tabla de bloque dependiente para el bloque actual, que puede contener el octeto actual/siguiente
$2C	FS.UPDT		puesto si se ha actualizado el fichero
$32	FS.FNAME*	2+36 octetos	nombre del fichero
$58	FS.SPARE	72	libres

Bloque de definición física

$10	FS.DRIVR*	4	apuntador a la conexión del mediador de acceso para el controlador
$14	FS.DRIVN*	1	número de unidad
$15		1	reservado
$16	FS.MNAMR*	2+10	nombre del medio
$22	FS.FILES	1	número de ficheros abiertos

La longitud viene determinada por el conector del mediador de acceso

Clave

* Inicialmente los bloques están a ceros, menos los marcados con * que los llena el IOSS

\+ La rutina de OPEN no se llama si intenta abrir un fichero compartido, abierto a más de un canal. En ese caso, FN.BYTE se pone a $40 y la información fija se copia desde el primer canal abierto al fichero.

Apéndice Q - Bloque de control de Job

Dir.	Lon.	Nombre	Descripción
$00	4	LEN	longitud total de control + área del Job
$04	4	START	dirección de comienzo en activación
$08	4	OWNER?	Job propietario de este
$0C	4	HOLD?	apuntador a un octeto que se limpiará cuando el planificador lo libere (ver MT.SUSJB D0=8)
$10	2	TAG*	etiqueta del Job, asignada por MT.CJOB
$12	1	PRIOR?	prioridad acumulada actual. Se incrementa cuando el JOB está activo, pero no ejecutándose. Se pone a cero cuando se está ejecutando. El planificador permite ejecutarse al Job con mayor prioridad acumulada
$13	1	PRINC?	Esta es la prioridad inicial del Job. El BASIC activa los Jobs con prioridad $20.
$14	2	STAT*	Estado del Job: 0 no suspendido >0 número de cuadros antes de liberarse 1 suspendido (ES o MT.SUSJB) 2 esperando a que termine otro Job
$16	1	RELA6?	Octeto más significativo puesto, si el siguiente TRAP #2 o #3 tiene direccionamiento relativo (puesto por el TRAP#4)
$17	1	WFLAG?	puesto si hay un Job esperando por éste
$18	4	WJOB?	ID del Job que espera a que termine éste
$1C	4	TRAPV	apuntador a los vectores de redireccionamiento de TRAP
$20	32		valores de D0 a D7, salvados
$40	32		valores de A0 a A7, salvados
$60	2		valor del registro de estado, salvado
$62	4		valor del contador de programa, salvado

* valor que no debe ser cambiado
? valor que puede ser cambiado por un TRAP, o directamente (pero con cuidado)

Apéndice R - Cabecera del área común

Nombre	Valor	Descripción
LEN	$00	longitud del bloque de definición
DRIVR	$04	dirección del controlador que libera el espacio cuando se borra al propietario
*NEXT	$04	apuntador al siguiente espacio libre
OWNER	$08	Job propietario (si es espacio libre)
RFLAG	$0C	dirección a poner cuando se libera el espacio
*COLSE	$0C	desplazamiento del punto de entrada de CLOSE del controlador de escritura

Apéndice S - Bloque de.definición de ventana

Nombre	Valor		
*SD.XMIN	$18	2	esquina superior izquierda de la ventana
SD.YMIN	$1A	2	
SD.XSIZE	$1C	2	tamaño de la ventana
SD.YSIZE	$1E	2	
SD.BORWD	$20	2	ancho del borde
*SD.XPOS	$22	2	posición del cursor
SD.YPOS	$24	2	
SD.XINC	$26	2	incremento del cursor
SD.YINC	$28	2	
*SD.FONT	$2A	8	dirección del juego de caracteres
*SD.SCRB	$32	4	base de la pantalla
*SD.PMASK	$36	4	máscara de color de papel
SD.SMASK	$3A	4	máscara de color del punteado
SD.IMASK	$3E	4	máscara de color de la tinta
*SD.CATTR	$42	1	atributos de los caracteres
	bit 0		subrayado
	bit 1		parpadeo
	bit 2		fondo transparente
	bit 3		XOR caracteres/gráficos
	bit 4		doble altura
	bit 5		ancho extendido
	bit 6		doble ancho
	bit 7		caracteres posicionados por gráficos
*SD.CURF	$43	1	indicador del cursor, 0=suprimido >0=visible
SD.PCOLR	$44	1	octeto de color del papel
SD.SCOLR	$45	1	octeto de color de punteado
SD.ICOLR	$46	1	octeto de color de la tinta
SD.BCOLR	$47	1	octeto de color del borde
*SD.NLSTA	$48	1	estado de nueva línea (>0= implícita, <0 explícita)
*SD.FMOD	$49	1	modo relleno (0=no, 1=si)
SD.YORG	$4A	flot	origen de la ventana gráfica
SD.XORG	$50	flot	origen de la ventana gráfica
SD.SCAL	$56	flot	factor de escala gráfica
SD.FBUF	$5C	4	apuntador a rellenar memoria intermedia
SD.FUSE	$60	4	apuntador a vectores del usuario
SD.END	$64		

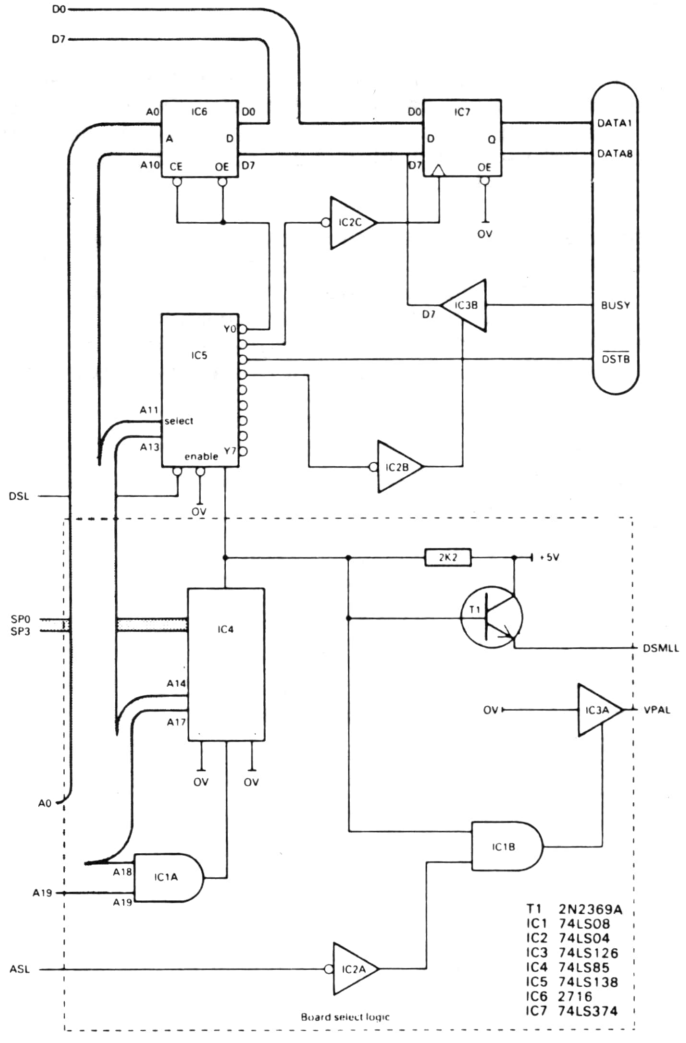
Apéndice T - Utilidad para copias de seguridad de microdrives

Este pequeño programa en BASIC, permite hacer copias de seguridad de cartuchos de microdrive. A diferencia de la mayoría de los programas de copia, no requiere que se especifiquen los nombres de todos los ficheros del cartucho, en las sentencias de datos.

El catálogo de entradas de mdv2_ se copia dentro del fichero llamado mdv1_dir. Después se leen la entradas desde el nuevo fichero de una en una, y se copian los ficheros correspondientes. Para mantener informado al operador del estado actual de la copia, el programa imprimirá el nombre de cada nuevo fichero que comience la transferencia.

```
100 rem Utilidad de copia de microdrives
110 OPEN_NEW #3,mdv1_dir
120 DIR #3,mdv2_
130 OPEN_IN #3,mdv1_dir
140 INPUT #3,a$;a$
150 REPeat cop
160     IF EOT (#3):EXIT cop
170     INPUT #3,a$
180     PRINT "Copiando mdv1_";a$
190     COPY 'mdv2_'&a$ TO 'mdv1_'&a$
200 END REPeat cop
210 CLOSE #3
220 DELETE mdv1_dir
```

Apéndice U - Circuito para impresora paralela



El circuito que presentamos aquí es para un 'interface' Centronics para impresora paralela. El 'software' que le permite funcionar desde el QDOS, se ha descrito en la sección 9.11.

La mitad inferior del circuito (o su equivalente) es el requerido en cualquier tarjeta de expansión del QL. Refierase al Apéndice F para una descripción detallada de los requerimientos de un interface estándar.

Descripción del circuito

IC1a se usa para detectar cuando es válida la dirección generada por el periférico (ej $CXXXX a $FXXXX). La salida de IC1a proporciona una entrada '=' para el comparador de cuatro bits, IC4, que compara las líneas de dirección A14 -A17 con las de selcción SP0-S. Si la dirección seleccionada es igual a la dirección real, la salida '=' del comparador se pone alta. Esta salida está conectada al transistor NPN de conmutación rápida que inhabilita el espacio interno de direcciones del QL, usando la señal DSMCL. La señal '=' se combina también con la dirección de 'strobe' para proporcionar una señal VPAL de reconocimiento de direcciones válidas de periféricos. Esta señal asegura que el 68008 opera en modo de direccionamiento 6800, de modo que se puede acceder a los periféricos lentos y a las memorias. En este circuito, IC3a tiene su entrada conectada a 0V, de forma que está operando como un inversor de colector abierto.

La parte del circuito que hemos descrito (o su equivalente(se requiere para todas las tarjetas periféricas. El circuito que vamos a describir ahora es específico del controlador de impresora paralela.

La señal '=' del comparador, se combina con la señal DSL para habilitar el selector de direcciones IC5, que se usa para decodificar direcciones dentro del espacio de direcciones de 16K octetos que se asigna a cada tarjeta periférica. Se decodifican ocho grupos de 2K. El grupo inferior debe ser siempre la ROM (IC6) ya que la primera palabra larga en esta dirección es el indicador de identificación. El grupo siguiente ($0800) proporciona un umbral al monoestable en DLS. Este monoestable se usa para pasar los datos desde las líneas al circuito octal IC7, que proporciona una salida estable para la impresora. El siguiente grupo ($1000) proporciona la señal de 'strobe' de los datos, y el último grupo ($1800) se usa para leer la línea BUSY de la impresora, en el bit 7 de las líneas de datos.

La señal VPAL debe ser usada por la mayoría de los periféricos. Esta señal pone al 68008 en modo periférico 6800, de forma que que todas las temporizaciones de las líneas son similares a las que generaría un 6800. Solo se puede acceder a las memorias lentas de los periféricos, seleccionando este modo. También asegura que ciertas señales (como el 'strobe' Centronics) duran más de 500ns.

Apéndice V-Juego de caracteres ASCII

Versión Inglesa

Decimal	Hex	Teclas	Pantalla/Función
0	00	CTRL £	NULO
1	01	CTRL A	
2	02	CTRL B	
3	03	CTRL C	cambia canal de entrada
4	04	CTRL D	
5	05	CTRL E	
6	06	CTRL F	
7	07	CTRL G	
8	08	CTRL H	
9	09	TAB(CTRL I)	Siguiente campo
10	0A	ENTER(CTRL J)	Nueva Línea/Entr. comandos
11	0B	CTRL K	
12	0C	CTRL L	
13	0D	CTRL M	Entrada
14	0E	CTRL N	
15	0F	CTRL O	
16	10	CTRL P	
17	11	CTRL Q	
18	12	CTRL R	
19	13	CTRL S	
20	14	CTRL T	
21	15	CTRL U	
22	16	CTRL V	
23	17	CTRL W	
24	18	CTRL X	
25	19	CTRL Y	
26	1A	CTRL Z	
27	1B	ESC(CTRL SHIFT [)	Aborta nivel act. comando
28	1C	CTRL SHIFT \	
29	1D	CTRL SHIFT]	
30	1E	CTRL SHIFT £	
31	1F	CTRL SHIFT ESC	
32	20	Espacio	Espacio
33	21	SHIFT 1	!
34	22	SHIFT '	"
35	23	SHIFT 3	#
36	24	SHIFT 4	$
37	25	SHIFT 5	%
38	26	SHIFT 6	&
39	27		'
40	28	SHIFT 9	(
41	29	SHIFT 0	)
42	2A	SHIFT 8	*

Decimal	Hex	Teclas	Pantalla/Función
43	2B	SHIFT =	+
44	2C	,	,
45	2D	-	-
46	2E	.	.
47	2F	/	/
48	30	0	0
49	31	1	1
50	32	2	2
51	33	3	3
52	34	4	4
53	35	5	5
54	36	6	6
55	37	7	7
56	38	8	8
57	39	9	9
58	3A	SHIFT ;	:
59	3B	:	;
60	3C	SHIFT ,	<
61	3D	=	=
62	3E	SHIFT .	>
63	3F	SHIFT /	?
64	40	SHIFT 2	@
65	41	SHIFT A	A
66	42	SHIFT B	B
67	43	SHIFT C	C
68	43	SHIFT D	D
69	45	SHIFT E	E
70	46	SHIFT F	F
71	47	SHIFT G	G
72	48	SHIFT H	H
73	49	SHIFT I	I
74	4A	SHIFT J	J
75	4B	SHIFT K	K
76	4C	SHIFT L	L
77	4D	SHIFT M	M
78	4E	SHIFT N	N
79	4F	SHIFT O	O
80	50	SHIFT P	P
81	51	SHIFT Q	Q
82	52	SHIFT R	R
83	53	SHIFT S	S
84	54	SHIFT T	T
85	55	SHIFT U	U
86	56	SHIFT V	V
87	57	SHIFT W	W
88	58	SHIFT X	X
89	59	SHIFT Y	Y

Decimal	Hex	Teclas	Pantalla/Función
90	5A	SHIFT Z	Z
91	5B	[	[
92	5C	\	\
93	5D	]	]
94	5E	SHIFT 6	^
95	5F	SHIFT -	_
96	60	£	£
97	61	A	a
98	62	B	b
99	63	C	c
100	64	D	d
101	65	E	e
102	66	F	f
103	67	G	g
104	68	H	h
105	69	I	i
106	6A	J	j
107	6B	K	k
108	6C	L	l
109	6D	M	m
110	6E	N	n
111	6F	O	o
112	70	P	p
113	71	Q	q
114	72	R	r
115	73	S	s
116	74	T	t
117	75	U	u
118	76	V	v
119	77	W	w
120	78	X	x
121	79	Y	y
122	7A	Z	z
123	7B	SHIFT [	{
124	7C	SHIFT \	¦
125	7D	SHIFT]	}
126	7E	SHIFT £	~
127	7F	SHIFT ESC	(c) copyright
128	80	CTRL ESC	ä
129	81	CTRL SHIFT 1	ã
130	82	CTRL SHIFT '	å
131	83	CTRL SHIFT 3	ö
132	84	CTRL SHIFT 4	õ
133	85	CTRL SHIFT 5	ø
134	86	CTRL SHIFT 7	ü
135	87	CTRL '	ç
136	88	CTRL SHIFT 9	ñ

Decimal	Hex	Teclas	Pantalla/Función
137	89	CTRL SHIFT 0	æ
138	8A	CTRL SHIFT 8	œ
139	8B	CTRL SHIFT =	á
140	8C	CTRL ,	à
141	8D	CTRL -	â
142	8E	CTRL .	ë
143	8F	CTRL /	é
144	90	CTRL 0	è
145	91	CTRL 1	ê
146	92	CTRL 2	ï
147	93	CTRL 3	í
148	94	CTRL 4	ì
149	95	CTRL 5	î
150	96	CTRL 6	ó
151	97	CTRL 7	ò
152	98	CTRL 8	ô
153	99	CTRL 9	ú
154	9A	CTRL SHIFT ;	ù
155	9B	CTRL ;	û
156	9C	CTRL SHIFT ,	beta
157	9D	CTRL =	¢
158	9E	CTRL SHIFT .	¥
159	9F	CTRL SHIFT /	'
160	A0	CTRL SHIFT 2	Ä
161	A1	CTRL SHIFT A	Ã
162	A2	CTRL SHIFT B	Å
163	A3	CTRL SHIFT C	Ö
164	A4	CTRL SHIFT D	Õ
165	A5	CTRL SHIFT E	Ø
166	A6	CTRL SHIFT F	Ü
167	A7	CTRL SHIFT G	Ç
168	A8	CTRL SHIFT H	Ñ
169	A9	CTRL SHIFT I	Æ
170	AA	CTRL SHIFT J	Œ
171	AB	CTRL SHIFT K	alfa
172	AC	CTRL SHIFT L	delta
173	AD	CTRL SHIFT M	theta
174	AE	CTRL SHIFT N	lambda
175	AF	CTRL SHIFT O	mu
176	B0	CTRL SHIFT P	pi
177	B1	CTRL SHIFT Q	phi
178	B2	CTRL SHIFT R	psi
179	B3	CTRL SHIFT S	abrir interjección
180	B4	CTRL SHIFT T	abrir interrogación
181	B5	CTRL SHIFT U	
182	B6	CTRL SHIFT V	§
183	B7	CTRL SHIFT W	

Decimal	Hex	Teclas	Pantalla/Función
184	B8	CTRL SHIFT X	<<
185	B9	CTRL SHIFT Y	>>
186	BA	CTRL SHIFT Z	°
187	BB	CTRL [	÷
188	BC	CTRL \	←
189	BD	CTRL]	→
190	BE	CTRL SHIFT 6	↑
191	BF	CTRL SHIFT -	↓
192	C0	Izquierda	Cursor izquierda un carácter
193	C1	ALT Izquierda	Cursor a principio de línea
194	C2	CTRL Izquierda	Borrar un carácter a la izq.
195	C3	CTRL ALT Izquier.	Borrar línea
196	C4	SHIFT Izquierda	Cursor izquierda una palabra
197	C5	SHIFT ALT Izq.	Mover hacia la izquierda
198	C6	SHIFT CTRL Izq.	Borrar palabra a izquierda
199	C7	SHIFT CTRL ALT Izq.	
200	C8	Derecha	Cursor derecha un carácter
201	C9	ALT Derecha	Cursor a fin de línea
202	CA	CTRL Derecha	Borrar carácter en cursor
203	CB	CTRL ALT Derecha	Borrar hasta fin de línea
204	CC	SHIFT Derecha	Cursor derecha una palabra
205	CD	SHIFT ALT Der.	Mover hacia la derecha
206	CE	SHIFT CTRL Der.	Borrar pal. cursor hasta fin
207	CF	SHIFT CTRL ALT Der.	
208	D0	Arriba	Cursor hacia arriba
209	D1	ALT Arriba	Scroll hacia arriba
210	D2	CTRL Arriba	Búsqueda hacia atrás
211	D3	ALT CTRL Arriba	
212	D4	SHIFT Arriba	Parte alta de pantalla
213	D5	SHIFT ALT Arriba	
214	D6	SHIFT CTRL Arriba	
215	D7	SHIFT CTRL ALT Arr.	
216	D8	Abajo	Cursor hacia abajo
217	D9	ALT Abajo	Scroll hacia abajo
218	DA	CTRL Abajo	Buscar hacia adelante
219	DB	ALT CTRL Abajo	
220	DC	SHIFT Abajo	Parte inferior de pantalla
221	DD	SHIFT ALT Abajo	
222	DE	SHIFT CTRL Abajo	
223	DF	SHIFT CTRL ALT Abajo	
224	E0	CAPSLOCK	Bloquear mayúsc./minúsc.
225	E1	ALT CAPSLOCK	
226	E2	CTRL CAPSLOCK	
227	E3	ALT CTRL CAPSLOCK	
228	E4	SHIFT CAPSLOCK	
229	E5	SHIFT ALT CAPSLOCK	
230	E6	SHIFT CTRL CAPSLOCK	

Decimal	Hex	Teclas	Pantalla/Función
231	E7	SHIFT CTRL ALT CAPSLOCK	
232	E8	F1	
233	E9	CTRL F1	
234	EA	SHIFT F1	
235	EB	CTRL SHIFT F1	
236	EC	F2	
237	ED	CTRL F2	
238	EE	SHIFT F2	
239	EF	CTRL SHIFT F2	
240	F0	F3	
241	F1	CTRL F3	
242	F2	SHIFT F3	
243	F3	CTRL SHIFT F3	
244	F4	F4	
245	F5	CTRL F4	
246	F6	SHIFT F4	
247	F7	CTRL SHIFT F4	
248	F8	F5	
249	F9	CTRL F5	
240	FA	SHIFT F5	
251	FB	CTRL SHIFT F5	
252	FC	SHIFT espacio	Espacio "especial"
253	FD	SHIFT TAB	Tabulador hacia atrás
254	FE	SHIFT ENTER	Nueva línea "Especial"
255	FF	Ver abajo	

Los códigos hasta el 20 hex. son o caracteres de control o caracteres no imprimibles.

Hay que tener en cuenta que los códigos C0-Df son comandos de control del cursor

Si se pulsa la tecla ALT junto con otra letra, que no sean als de contol del cursor o CAPSLOCK, genera el código FF seguido del código de la otra tecla pulsada.

Apéndice W-Juego de caracteres ASCII

Versión Española

Decimal	Hex		Teclas	Pantalla/Función
0	00	*	CTRL ESC	NULO
1	01		CTRL A	
2	02		CTRL B	
3	03		CTRL C	cambia canal de entrada
4	04		CTRL D	
5	05		CTRL E	
6	06		CTRL F	
7	07		CTRL G	
8	08		CTRL H	
9	09		TAB(CTRL I)	Siguiente campo
10	0A		↵ CTRL J)	Nueva Línea/Entr. comandos
11	0B		CTRL K	
12	0C		CTRL L	
13	0D		CTRL M	Entrada
14	0E		CTRL N	
15	0F		CTRL O	
16	10		CTRL P	
17	11		CTRL Q	
18	12		CTRL R	
19	13		CTRL S	
20	14		CTRL T	
21	15		CTRL U	
22	16		CTRL V	
23	17		CTRL W	
24	18		CTRL X	
25	19		CTRL Y	
26	1A		CTRL Z	
27	1B		ESC(CTRL ↑ [)	Aborta nivel act. comando
28	1C	*	CTRL ↑ -	
29	1D		CTRL ↑]	
30	1E	*	CTRL ↑ =	
31	1F		CTRL ↑ ESC	
32	20		Espacio	Espacio
33	21	*	↑ .	!
34	22		↑ '	"
35	23		↑ 3	#
36	24		↑ 4	$
37	25		↑ 5	%
38	26	*	↑ 7	&
39	27	*	CTRL '	'
40	28		↑ 9	(
41	29		↑ 0	)
42	2A		↑ 8	*

Decimal	Hex		Teclas		Pantalla/Función
43	2B		↑	=	+
44	2C		,		,
45	2D		-		-
46	2E				.
47	2F	*	↑	6	/
48	30		0		0
49	31		1		1
50	32		2		2
51	33		3		3
52	34		4		4
53	35		5		5
54	36		6		6
55	37		7		7
56	38		8		8
57	39		9		9
58	3A		↑	;	:
59	3B		:		;
60	3C	*	<		<
61	3D		=		=
62	3E	*	↑	<	>
63	3F	*	↑	.	?
64	40	*	CTRL	8	@
65	41		↑	A	A
66	42		↑	B	B
67	43		↑	C	C
68	43		↑	D	D
69	45		↑	E	E
70	46		↑	F	F
71	47		↑	G	G
72	48		↑	H	H
73	49		↑	I	I
74	4A		↑	J	J
75	4B		↑	K	K
76	4C		↑	L	L
77	4D		↑	M	M
78	4E		↑	N	N
79	4F		↑	O	O
80	50		↑	P	P
81	51		↑	Q	Q
82	52		↑	R	R
83	53		↑	S	S
84	54		↑	T	T
85	55		↑	U	U
86	56		↑	V	V
87	57		↑	W	W
88	58		↑	X	X
89	59		↑	Y	Y

Decimal	Hex		Teclas		Pantalla/Función
90	5A		⇑ Z		Z
91	5B		[		[
92	5C	*	CTRL 9		\
93	5D		]		]
94	5E		⇑ ↑		^
95	5F		⇑ -		_
96	60	*	CTRL -		£
97	61		A		a
98	62		B		b
99	63		C		c
100	64		D		d
101	65		E		e
102	66		F		f
103	67		G		g
104	68		H		h
105	69		I		i
106	6A		J		j
107	6B		K		k
108	6C		L		l
109	6D		M		m
110	6E		N		n
111	6F		O		o
112	70		P		p
113	71		Q		q
114	72		R		r
115	73		S		s
116	74		T		t
117	75		U		u
118	76		V		v
119	77		W		w
120	78		X		x
121	79		Y		y
122	7A		Z		z
123	7B	*	CTRL =		{
124	7C	*	CTRL O		¦
125	7D	*	CTRL [		}
126	7E	*	CTRL]		~
127	7F		⇑ ESC		(c) copyright
128	80	*	CTRL ⇑	R	ä
129	81		CTRL ⇑	1	Ä
130	82	*	CTRL ⇑	2	à
131	83	*	CTRL ⇑	3	é
132	84	*	CTRL ⇑	4	ö
133	85	*	CTRL ⇑	5	Ö
134	86	*	CTRL ⇑	6	ø
135	87	*	CTRL ⇑	]	ü
136	88	*	CTRL ⇑	[	ç

Decimal	Hex		Teclas	Pantalla/Función
137	89	*	ñ	ñ
138	8A	*	CTRL ↑ 7	æ
139	8B	*	CTRL ↑ 8	œ
140	8C	*	CTRL ,	á
141	8D	*	CTRL ↑ 9	à
142	8E	*	CTRL .	ä
143	8F	*	CTRL ↑ S	ë
144	90	*	CTRL ↑ Q	è
145	91	*	CTRL ↑ L	ê
146	92	*	CTRL ↑ M	ï
147	93	*	CTRL ↑ N	í
148	94	*	CTRL ↑ O	ì
149	95	*	CTRL ↑ P	î
150	96	*	CTRL ↑ T	ó
151	97	*	CTRL ↑ I	ò
152	98	*	CTRL ↑ ñ	ô
153	99	*	CTRL ´ y U	ú
154	9A	*	CTRL ` y U	ù
155	9B	*	CTRL ñ	û
156	9C		CTRL ↑ ,	ß
157	9D	*	CTRL <	¢
158	9E		CTRL ↑ .	¥
159	9F	*	CTRL ´	'
160	A0	*	CTRL ↑ <	Ä
161	A1		CTRL ↑ A	Á
162	A2		CTRL ↑ B	Â
163	A3	*	CTRL ↑ E	É
164	A4	*	CTRL ↑ D	Ö
165	A5	*	CTRL ↑ E	Õ
166	A6	*	CTRL ↑ F	Ø
167	A7	*	CTRL ↑ G	Ü
168	A8	*	CTRL ↑ H	Ç
169	A9	*	↑ ñ I	Ñ
170	AA	*	CTRL ↑ J	Æ
171	AB	*	CTRL ↑ K	Œ
172	AC	*	CTRL 1	α
173	AD	*	CTRL 2	δ
174	AE	*	CTRL 3	θ
175	AF	*	CTRL 4	λ
176	B0	*	CTRL 5	μ
177	B1	*	CTRL 6	π
178	B2	*	CTRL 7	Ω
179	B3	*	↑ 1	¡
180	B4	*	↑ 2	¿
181	B5		CTRL ↑ U	§
182	B6	*	CTRL ↑ V	
183	B7	*	CTRL ↑ W	

Decimal	Hex	Teclas	Pantalla/Función
184	B8	CTRL ⇧ X	<<
185	B9	CTRL ⇧ Y	>>
186	BA	CTRL ⇧ Z	°
187	BB *	CTRL ⇧ O	÷
188	BC *	CTRL ⇧ ;	←
189	BD *	CTRL :	→
190	BE *	CTRL ⇧ '	↑
191	BF *	CTRL ⇧	↓
192	C0	Izquierda	Cursor izquierda un carácter
193	C1	ALT Izquierda	Cursor a principio de línea
194	C2	CTRL Izquierda	Borrar un carácter a la izq.
195	C3	CTRL ALT Izquier.	Borrar la línea
196	C4	⇧ Izquierda	Cursor izquierda una palabra
197	C5	⇧ ALT Izq.	Mover hacia la izquierda
198	C6	⇧ CTRL Izq.	Borrar palabra a izquierda
199	C7	⇧ CTRL ALT Izq.	
200	C8	Derecha	Cursor derecha un carácter
201	C9	ALT Derecha	Cursor a fin de línea
202	CA	CTRL Derecha	Borrar carácter en cursor
203	CB	CTRL ALT Derecha	Borrar hasta fin de línea
204	CC	⇧ Derecha	Cursor derecha una palabra
205	CD	⇧ ALT Der.	Mover hacia la derecha
206	CE	⇧ CTRL Der.	Borrar pal. cursor hasta fin
207	CF	⇧ CTRL Der.	
208	D0	Arriba	Cursor hacia arriba
209	D1	ALT Arriba	Scroll hacia arriba
210	D2	CTRL Arriba	Búsqueda hacia atrás
211	D3	ALT CTRL Arriba	
212	D4	⇧ Arriba	Parte alta de pantalla
213	D5	⇧ ALT Arriba	
214	D6	⇧ CTRL Arriba	
215	D7	⇧ CTRL ALT Arr.	
216	D8	Abajo	Cursor hacia abajo
217	D9	ALT Abajo	Scroll hacia abajo
218	DA	CTRL Abajo	Buscar hacia adelante
219	DB	ALT CTRL Abajo	
220	DC	⇧ Abajo	Parte inferior de pantalla
221	DD	⇧ ALT Abajo	
222	DE	⇧ CTRL Abajo	
223	DF	⇧ CTRL ALT Abajo	
224	E0	⇳	Bloquear mayúsc./minúsc.
225	E1	ALT ⇳	
226	E2	CTRL ⇳	
227	E3	ALT CTRL ⇳	
228	E4	⇧ ⇳	
229	E5	⇧ ALT ⇳	
230	E6	⇧ CTRL ⇳	

Decimal	Hex	Teclas	Pantalla/Función
231	E7	⇑ CTRL ALT ⇕	
232	E8	F1	
233	E9	CTRL F1	
234	EA	⇑ F1	
235	EB	CTRL ⇑ F1	
236	EC	F2	
237	ED	CTRL F2	
238	EE	⇑ F2	
239	EF	CTRL ⇑ F2	
240	F0	F3	
241	F1	CTRL F3	
242	F2	⇑ F3	
243	F3	CTRL ⇑ F3	
244	F4	F4	
245	F5	CTRL F4	
246	F6	⇑ F4	
247	F7	CTRL ⇑ F4	
248	F8	F5	
249	F9	CTRL F5	
240	FA	⇑ F5	
251	FB	CTRL ⇑ F5	
252	FC	⇑ Espacio	Espacio "especial"
253	FD	⇑ TAB	Tabulador hacia atrás
254	FE	⇑ ENTER	Nueva línea "Especial"
255	FF	Ver abajo	

El '*' marca los códigos que se teclean de diferente forma o que producen diferente símbolo, en la versión española con respecto a la inglesa.

Los códigos hasta el 20 hex. son o caracteres de control o caracteres no imprimibles.

Hay que tener en cuenta que los códigos C0-Df son comandos de control del cursor

Si se pulsa la tecla ALT junto con otra letra, que no sean als de contol del cursor o CAPSLOCK, genera el código FF seguido del código de la otra tecla pulsada.

Apéndice X - Notas aclaratorias para la versión Versión Española

Las diferencias entre las versiones inglesa y española del QL, se limitan al juego de caracteres y a la forma de acceder a ellos desde el teclado. Todos los caracteres usados en español vienen ya incluidos en el QL estándar, por lo que solamente se ha tenido que cambiar la forma de acceso desde el teclado. Así, las letras acentuadas se forman pulsando primero el acento y después la letra correspondiente. También se ha incluido como tecla directa la correspondiente a la letra Ñ. Así como ciertos signos propios de nuestro idioma.

Debido a estos cambios, el esquema del teclado que se da en la figura 5.3 del capítulo 5, queda modificado así

FILA \ COLUMNA	7	6	5	4	3	2	1	0
0	7	4	F5	F3	F2	5	F1	F4
1	↓	SPACE	]	→	ESC	↑	←	↵
2	;	M	[	B	C	·	Z	`
3	Ñ	G	=	F	S	K	⇕	´
4	J	D	P	A	1	M	3	L
5	0	Y	—	R	TAB	I	W	9
6	U	T	Ø	E	Q	6	2	8
7	,	N	<	V	X	ALT	CTRL	⇑

Figura X.1 Esquema del teclado de la versión española

Modificaciones a las rutinas de impresión

Existen también algunas modificaciones en las rutinas de manejo de la impresora. En éstas se ha incluido un nuevo comando SuperBASIC:

TRA 0 los caracteres se envían a la impresora tal y como están almacenados en la memoria.

TRA 1 antes de mandarlos a la impresora, los caracteres se modifican de acuerdo con la siguiente tabla.

Carácter en el QL	Secuencia de impresión	
á	' BS a	
é	' BS e	
í	' BS i	
ó	' BS o	
ú	' BS u	
à	` BS a	
è	` BS e	
ì	` BS i	
ò	` BS o	
ù	` BS u	
¿	]	nota 1
¡	[	nota 1
Ñ	\	nota 1
ñ	¦	nota 1
ç	, BS c	
ü	{ BS u	nota 1 { escribe ¨
[	(	nota 2
]	)	nota 2
{	(	nota 2
}	)	nota 2

NOTA 1 - Estos caracteres se han modificado en la ROM de la impresora que comercializa Investrónica para el QL, para producir los caracteres correspondientes.

NOTA 2 - Las llaves y los corchetes se imprimen como paréntesis, ya que sus códigos se han usado para los caracteres de la nota 1. Se han elegido los paréntesis por ser los más parecidos y porque en la mayoría de los lenguajes para ordenadores pueden ser fácilmente sustituidos.

BS - es la abreviatura de 'back space' y es un código que hace volver un espacio hacia atrás a la impresora para escribir sobre el carácter anterior.

Nuevas variables de BASIC

La ROM que incluye el actual QL en español, contiene la versión MGE (MG es la versión estándar y la E es la de Español). Está en preparación una nueva versión que modifica la sintáxis del comando TRA, para poder suministrar una dirección como argumento que corresponderá a una tabla de conversión de caracteres, con lo que se podrá elegir entre la propia de la ROM o una definida por el usuario y que se ajuste a sus propias necesidades.

En esta versión se incluyen también una serie de variables BASIC que se crean automáticamente a tiempo de encendido del QL, y que se usan para la detección y posible recuperación de errores. Estas son:

ERROR = 0 última instrucción,que se ejecutó correctamente
1 última instrucción que se ejecutó con un error

ERNUM = número del último error producido (ver apéndice G)
ERLIN = número de la línea en que se produjo el error

ERR_NC = nota 3
ERR_NJ = nota 3
ERR_OM = nota 3
ERR_OR = nota 3
ERR_BO = nota 3
ERR_NO = nota 3
ERR_NF = nota 3
ERR_EX = nota 3
ERR_IU = nota 3
ERR_EF = nota 3
ERR_DF = nota 3
ERR_BN = nota 3
ERR_TE = nota 3
ERR_FF = nota 3
ERR_BP = nota 3
ERR_FE = nota 3
ERR_XP = nota 3
ERR_OV = nota 3
ERR_NI = nota 3
ERR_RO = nota 3
ERR_BL = nota 3

NOTA 3 - Todas estas variables pueden tener un valo 0 o 1 y corresponden a los errores listados en el Apéndice G. El 1 significa que el último error producido ha sido el descrito por el nombre, sustituyendo el '_' por un '.'. El valor 0 significa que no se ha producido éste tipo de error.

Por ejemplo, si después de un error ejecutamos:

```
PRINT ERR_EF
```

y nos da como resultado un 1, debemos ir al apéndice G y localizar el error ERR.EF que significa 'Fin del fichero'.

Nuevos instrucciones SuperBASIC

Para sacar mayor partido a estas variables, se incluyen también en esta versión dos nuevas instrucciones SuperBASIC:

```
WHEN <expresión booleana>
     <cuerpo de la instrucción>
END WHEN
```

<expresión booleana> corresponde a una de las variables ántes descritas.

<cuerpo de la instrucción> son sentencias BASIC que se ejecutarán en el caso de que sea 1 la expresión booleana.

Se debe terminar siempre con END WHEN.

Esta sentencia se debe poner al principio del programa y se ejecutará cada vez que el SuperBASIC encuentre un error en la ejecución. De esta forma podemos establecer un sistema de análisis y recuperación, si procede, de los errores que se produzcan durante la ejecución de nuestros programas.

La otra instrución nueva es:

```
REPORT [# canal]
```

Envía el mensaje de error, asociado con el error producido, por el canal indicado. Si no se le especifica canal, lo mandará a la pantalla principal.

Se puede usar para lanzar los mensajes de error a ventanas distintas de la que se está usando o a otros dispositivos conectados a nuestro QL.

Todos los datos suministrados en éste apéndice, no aparecen en el manual que se suministra con el QL comercializado en España, y se incluyen por cortesía, y con permiso, de Investrónica.

Bibliografía

M68000 Cross Macro Assembler Reference Manual, Motorola Inc.,1979

M68000 16/32-bit Microprocessor Programmer's Reference Manual, Prentice-Hall INC., Englewood Clifs, N.J.07632,U.S.A

MC68008 16-bit Microprocesor with 8-bit Data Bus, Advance Information, Motorola Semiconductors, Scotland, 1983

Sinclair QL User Guide, Sinclair Research Limited, Cambridge, 1984

Sinclair QDOS Documentation, Sinclair Research Limited, Cambridge, 1984

Activo bajo - se conocen como señales *activas bajas* aquellas que son válidas cuando están a nivel lógico 0.

Alto - se usa algunas veces para designar el nivel lógico 1

Area común - es un área de memoria que mantiene el sistema operativo. Los programas de usuario pueden pedir al sistema operativo que les ceda parte de este área para distintos propósitos.

Asíncrono - dos dispositivos que operan independientemente un del otro, se dicen que operan asincronamente.

Atómico - todas las acciones que se tienen que ejecutar en un periodo de tiempo fijo, se dice que son atómicas. Estas acciones se realizan, normalmente, en modo supervisor.

Bajo - se usa algunas veces para designar el nivel lógico 0

Bidireccional - una línea de comunicación es bidireccional si los datos se pueden mandar y recibir a través de ella. Las líneas de del 68008 datos son bidireccionales.

Bit de memoria - es la unidad fundamental de la memoria de los ordenadores. Solo puede estar en uno de dos estados, representados normalmente por 0 o 1.

Canales - son los cauces a través de los que se dirige la entrada y salida. Cada canal está definido por un *bloque de definición de canal*.

Chip - es una pequeña pieza de oblea de silicona encerrada en una envoltura de cerámica o plástico, que contiene una gran cantidad de circuitos lógicos dentro de ella. De esta envoltura de plástico o cerámica salen unas patillas metálicas que lo conectan con el exterior.

Código auto-modificable - es una técnica de programación, no muy segura, que consiste en un programa que se modifica a sí mismo.

Código máquina - son programas producidos por el Ensamblador del 68000. Un programa en código máquina consiste en series de palabras en memoria que el 68008 puede ejecutar directamente

Código reubicable - código máquina que funciona sin problemas, independientemente del lugar de la memoria en que se enceentre ubicado.

Colector abierto - es una característica de la línea de salida de un transistor. Significa que la patilla del colector del transistor no tiene ninguna carga conectada, es decir está *abierto*.

Controlador de dispositivo - todos los dispositivos físicos conectads al QL requieren algo para poder interconectarse con el sistema operativo. Los controladores de dispositivo son programas en código máquina que realizan esta labor.

Extendido - son operaciones en código 68008 que permiten la conversión entre octetos, palabras y palabras largas.

Handshaking - este protocolo de comunicación se usa para transferir datos entre dos dispositivos asíncronos (como puertas serie RS232). Se usan normalmente dos líneas. Una de estas es para la señal de *datos preparados* desde el dispositivo que manda al que recibe. Cuando el que recibe ha aceptado los datos, manda una señal de *datos recibidos*, al dispositivo original, que sabe así que puede mandar otro grupo de datos, y así sucesivamente.

Interrupción - las interrupciones son genereadas por piezas de 'hardware' que quieren obtener un servicio inmediato del 68008. Las interrupciones externas vienen de los dispositivos de expansión y las internas vienen de los dispositivos contenidos dentro del QL. Una de las más importantes es la interrupción de vídeo 50/60 Hz, que pasa control al planificador de Jobs.

IOSS(Input/Output Sub-System) - es el subsistema de entrada/salida, y está compuesto por las rutinas asociadas con las entradas y salidas básicas de los dispositivos, como las puertas serie o los microdrives.

IPC(Intelligent Peripherial Controler) - es el *segundo procesador* del QL, que controla el teclado, las puertas serie y el sonido.

Job - es un programa que ha sido preparado para funcionar independientemente. Por ejemplo, se puede preparar un Job para producir una imágen de un reloj de tiempo real en la pantalla. Este Job puede funcionar independientemente del BASIC.

LED(Light Emitting Diode) - es como un diodo que permite pasar la corriente en una sola dirección, pero que emite luz mientras pasa la corriente. Hay tres en el QL, uno indica que está conectado, los otros dos indican cual de los microdrives está en uso.

Líneas de datos - es un grupo de ocho conexiones por las que se realizan las transacciones entre los dispositivos y el QL.

Líneas de dirección - es un grupo de 20 conexiones, cada una de las cualse puede estar a nivel lígico 0 o 1. Esto permite a la CPU direccionar $FFFFF(1048576) posiciones diferentes.

Lista encadenada - es una estructura de datos que tiene una gran variedad de aplicaciones. Por ejemplo, un grupo de rutinas que se deben llamar en secuencia, pueden conectarse en una lista encadenada.

LSW - es la abreviatura inglesa de 'palabra menos significativa'.

Modo supervisor - el 68008 puede operar en dos modos, usuario o supervisor. En modo supervisor se pueden realizar varias instrucciones que no se pueden ejecutar en modo usuario.

Modo usuario - el 68008 puede operar en dos modos, usuario o supervisor. El modo usuario es el normal, pero no puede ejecutar ciertas instrucciones (como RESET).

MSW - es la abreviatura inglesa de 'palabra más significativa'.

Multitarea - es el hecho de poder ejecutar varios programas al mismo tiempo. Cada programa puede funcionar por un corto periodo de tiempo, después pasa a ejecutarse otro. Esto asegura que todos los programas operan durante el tiempo disponible.

Octeto de memoria - está formado por 8 bits. Los datos se transfieren normalmente entre dispositivos un octeto de cada vez, por las líneas de datos.

OS - abreviatura inglesa de 'sistema operativo'.

Palabra de memoria - está formada por dos octetos de memoria consecutivos, el primero es siempre par.

Palabra larga de memoria - está formada por dos palabras o cuatro octetos consecutivos, el primero es siempre par.

Paralelo - la transferencia paralela de datos ocurre cuando se manda datos por dos o más líneas al mismo tiempo. Las líneas de datos del sistema operan 8 en paralelo.

Periférico - es cualquier dispositivo conectado al procesador central 68008, tales como las puertas serie, la red de datos, microdrives etc., pero sin incluir la memoria.

Pila - es un bloque de memoria de *almacenamiento temporal*. Los datos se pueden añadir con *PUSH* y sacar con *POP*. Se usa como almacenamiento temporal de datos cuando no se pueden usar posiciones específicas de memoria. En el QL hay dos pilas del sistema usadas por el 68008. Una se usa exclusivamente en *modo supervisor* y la otra se usa exclusivamente en *modo usuario*. Se llaman pilas de 'supervisor' y de 'usuario' respectivamente.

Planificador - el planificador es el que decide, cada 20ms, cual de los Jobs debe ejecutarse en el QL.

Procedimientos residentes - son programas que se cargan en el área de procedimientos residentes en el momento de encender el QL. En este area se pueden almacenar las utilidades y comandos adicionales de BASIC.

QDOS - es el paquete de programas del QL que ejecutan todas las tareas del sistema necesarias, tales como recoger datos del teclado, o mandar datos a la pantalla. Es el *Sistema operativo del QL*.

RAM(Random Access Memory) - la memoria principal del QL es memoria RAM. Este tipo de memoria almacena datos y programas mientra está encendido. Estos datos se pueden modificar en cualquier momento. Cuando se apaga la máquina, se pierden todos los datos almacenados en RAM.

Re-entrante - son rutinas que se pueden ejecutar, incluso desde el interior de sí mismas, sin que fallen. Las rutinas de servicio de interrupciones deben ser, normalmente, re-entrantes. Para que una rutina sea re-entrante, no se deben usar posiciones fijas de memoria. Por lo tanto, todas las variables que se usan en la rutina deben almacenarse en la pila.

Renovación - todas las RAM del QL son memorias dinámicas. Esto significa que se debe renovar su contenido cada pocos milisegundos, para que no se pierdan los datos. La función de renovar se realiza accediendo regularmente a la memoria al tiempo de la salida a pantalla.

ROM(Read Only Memory) - es la memoria donde está almacenado el BASIC y el QDOS. En esta memoria se almacenan los datos cuando se fabrica, y no se pueden borrar ni grabar de nuevo. Esta memoria retiene los datos aunque se apague la máquina.

Serie - los datos que se transmiten a lo largo de una única línea lo hacen en forma serializada. La transmisión de datos en serie es normalmente más lenta que en paralelo, ya que se transmiten los bits de uno en uno.

Tipo de letra - es el grupo de caracteres alfanuméricos que aparecen en la pantalla del QL.

TRAP - es una instrucción del 68008 que permite acceder a rutinas específicas. Hay 16 TRAPs disponibles, de los que el QDOS usa solamente 5.

ULA(Uncommitted Logic Array) - son chips especiales que contiene una gran cantidad de puertas lógicas. La conexión entre las puertas se define cuando se fabrica el chip. Sinclair ha incluido dos ULAs en el QL. Una controla la salida de vídeo y la memoria, y la otra los microdrives y la red de datos.

Variables locales - son variables específicas de un procedimiento BASIC, pero que no existen fuera de él.

Velocidad de transmisión - se usa para definir la velocidad a la que trasmite la conexión serie del QL. Un baudio es igual a 1 bit de datos por segundo.

Vectores - son unas posiciones de memoria que contienen las direcciones de rutinas específicas. Se les suele llamar *vectores* porque la dirección se puede usar para redirigir el programa a alguna parte específica de la memoria.

Indice

68008 11
  juego de instrucciones 21
  modelo de programación 14
  registros 14
  sumario del juego de instrucciones 302
8049 IPC 68,90

A

Abriendo ficheros 115
Abrir un canal 119,266
  ejemplo 281
  para controlador de directorio 274
Activar un Job 84
Ajuste del reloj 97
Almacenamiento de variables 283
Apuntador a variables 330
Apuntador de fichero 184,185
Apuntador de nombre 274
Apuntadores de pila (68008) 13
Area común 30
  asignación 64,196
  asignación por el usuario 66
  asignar 64,100,196
  cabecera 341
  gestión por el usuario 84
  liberando espacio 86
  liberar 101,197
Area de usuario
  asignación 66,85
  gestión 84,211
Argumentos para procedimientos 292
Asignación de recursos 34
Asignar área común 196
Asignar área del usuario 85
Atómico 337
  interrupción del planificador 269
  rutinas 35
Atributos de caracteres
  modo dibujo 172
  modo parpadeo 170
  modo subrayado 171
  tamaño y espaciado 173

B

BASIC Ver SuperBASIC
BASIC, rutinas de utilidad
  comparación de cadenas 219
  rutinas de conversión 222
  variables/apuntadores 320
Bloque de control de Job 340
Bloque de definición de canal 199,274,338
Bloque definición controlador directorio 273
Bloque definición controlador dispositivo 267
Bloques dependientes 275
Borrando ficheros 123,274
Borrar Jobs 74
BPUT#, ejemplo de procedimiento 298

C

Cabecera de fichero 188,189
Cabecera definición canal 274
CALL 37
Cambio de pantallas 293
Canales 111
  abrir 119
  cerrar 121
  comprobar entradas pendientes 130
  ejemplo 280
  entrada/salida 267
  llamadas E/S serie 130
  mandar un octeto 135
  mandar una cadena de octetos 136
  obtener un octeto 131
  obtener una cadena de octetos 133
  obtener una linea 132
Cargar un fichero 190
Cerrando un canal 121,267
  ejemplo 280
CHECKSUM 308
Circuito ejemplo de interface 344
Códigos de operación aritméticos 334
Códigos de error 323
Color de tinta 169
Color de papel 167
Colores
  poner barras 168
  poner papel 167
  poner tinta 169
  poner ventana 166
  ponerlos 164
  punteado 164
Comparación de cadenas 219
Comprobar operaciones con ficheros 183
Conectar a una lista 209
Conector de expansión 317
Conexiones de la ROM 303
Controlador de directorio
  bloque de definición 273
  encadenado 110
  mediador de acceso 274
Controlador de dispositivo 111,263
  asignación de memoria 271
  bloques de definición 264
  decodificación de nombres 269
  ejemplo 277
  en ROM 280
  encadenado 109
  interrupciones 268
  mediador de acceso 266
  mediador físico 268
Contador de programa (68008) 16
Control del cursor 142
Control de salida a pantalla 137
Control del sistema
  asignación de recursos 34
  gestión de memoria 30
  planificación de tareas 34
Controladores de pantalla, de usuario 137
Conversión de dia 223
Creando Jobs 72

D

Definición canal de ficheros del sistema 339
Definición de bloque de ventana 342
Desconectar un item de una lista 210
Devolución de parámetros 246
Dibujar un arco 177

Dibujar una elipse 178
Dibujar una línea 175
Direcciones efectivas 18
Duplicado de Jobs 55

E

Ejemplo de impresora paralela 277
Ejemplos de almacenamiento de variables 287
Eliminar Jobs 74
Eliminar sonido 91
END WHEN, instrucción BASIC Ver WHEN
Entrada/Salida 111
  controladores de directorio 110
  controladores de dispositivo 109
  ejemplo 281
  manejo de colas 272
  serie 125
  serie simple 272
Entrada de tabla de nombres 283
Error
  códigos de 323
  escribir mensaje de 205,362
  variables para la detección de 361
Escala 179
Escala del dibujo 179
Escribir mensaje de error 205
Escribir sector de microdrive 259
Espacio libre en el área común 86
Español
  notas aclaratorias 359
  teclado 359
  impresora 360
  variables BASIC, nuevas 361
  instrucciones BASIC, nuevas 362
Especificaciones de expansión hardware 317
Especificaciones de la ROM enchufable 303
Estado de las colas 214
Estado de los Jobs 73
Estructura de un sector 311
Excepciones
  proceso 18
  redireccionamiento 77
EXEC 37
EXEC_W 38
Experimentador, programa 41
Experimentando con el QDOS 37
Extendiendo el QDOS 102

F

Fichero lógico 111
Ficheros 111
  abrir 119
  borrado 123
  compartidos 116
  exclusivos 116
  orden de búsqueda 116
FILL 180
Formateo de un medio 123
Función residente SINH 252

G

Generación de sonido 91
Gestión de memoria 30,61
  área común 64
  área de usuario 66
  áreas comunes 64
  asignación a controlador de dispositivo 271
  asignación a SuperBASIC 64
  Jobs 62
  procedimientos residentes 62
  programas transitorios 62
Gestor de colas 198

H

Habilitar el cursor 142
Hardware 68
Hueco 308

I

Identificador del Job actual 70
Información del medio 187
Instrucciones BASIC, nuevas 362
Interface impresora, Centronics 277
Interface impresora, circuito 344
Interrupciones 16
  para controlador de dispositivo 268
  rutinas de servicio 106
IOSS (I/O sub-system) 263
IPC 68,90
  comunicación con 90
  mandar un comando 93

J

Jobs 34,62
  activación 82
  activo 35,62
  borrar 74
  como programas transitorios 62
  control 78
  creando y borrando 71
  duplicación 55
  empezar 37
  forzar el borrado 75
  inactivo 34,62
  liberar 81
  manejo de la memoria 63
  obtener el estado 73
  prioridad 83
  suspendido 34,62
  suspendiendo 48,80
Juego de caracteres ASCII
  versión inglesa 347
  versión española 353
Juego de instrucciones (68008) 21

L

LBYTES 38
Leer cabecera de sector 262
Leer modo pantalla 89
Leer reloj 95
Leer sector microdrive 255
Leer teclado 91
Liberación memoria intermedia de ficheros 184
Liberar Jobs 81
Limpiar ventanas
  a la derecha de la línea del cursor 161

línea del cursor 161
parte inferior 160
parte superior 159
toda la ventana 158
Línea de edición 134
Lista de nombres 285
Listas encadenadas 102
Llamadas E/S serie 125
comprobar entrada pendiente 130
editar una línea 134
mandar una cadena de octetos 136
mandar un octeto 135
obtener una cadena de octetos 133
obtener una línea 132
obtener un octeto 131

M

Mandar un mensaje a un canal 208
Mandar un entero ASCII 207
Mandar un octeto 135
Mandar una cadena de octetos 136
Manejo de las colas 272
Manejo de ficheros 127
Mapa de la memoria 31,301
Marca de fin de fichero 217
Matrices en SuperBASIC 284
Mediador de acceso 266
Mediador físico 268
Memoria libre 31
usada por bloques dependientes 275
Microdrive
errores 313
escribir un sector 259
estructura de un sector 311
formateo de ficheros 310
formato 306
leer cabecera de sector 261
leer un sector 258
rutinas de soporte 195
utilidad de copias de seguridad 343
verificar un sector 260
Modo parpadeo
Modo subrayado 171
Modo Supervisor 15
Modo usuario 15
Modos de direccionamiento 19
Movimiento de ventanas 49
derecha de la línea del cursor 157
línea del cursor 156
toda la ventana 155

N

Nombres
de dispositivo 112
ejemplos de nombres de dispositivo 271
sistema de ficheros 113
utilidad 210
Número aleatorio 59

O

Obtención de parámetros
de cadena 243
de coma flotante 242
enteros 241
enteros largos 244
Obtener un octeto 131
Obtener una línea 132

P

Pantalla
configuración 316
control de salida 126
mover ventanas 49
poner/leer modo pantalla 89
salida 48
Parámetros 279
PEEK 39
PEEK_L 39
PEEK_W 39
Pendiente comprobación entrada 130
Periodo de espera 125
Pila aritmética 292
Planificador de tareas 34
Planificador 33,62
bucle de tareas 107
mediador físico controlador dispositivo 269
POKE 39
POKE_L 40
POKE_W 40
Poner modo pantalla 89
Poner reloj en hora 96
Poner un octeto en una cola 215
posicionar el cursor
en la columna anterior 147
en columna de carácter 144
en coordenadas de carácter 144
en coordenadas de punto 151
en la fila anterior 149
en nueva línea 146
en la siguiente columna 148
en la siguiente fila 150
gráficos 181
Preámbulo 308
Prioridad 34
cambio 83
Procedimientos residentes
área de memoria 30
asignación de memoria 62
asignar memoria 87
liberar memoria 88
RESPR asignación de memoria 40
Procedimientos
devolver valores 293
ejemplo BPUT# 298
ejemplo de cambio de pantalla 293
en código máquina 290
entradas en tabla de nombres 290
obtener argumentos 292
Programas transitorios
área de memoria 30
buscar espacio libre 76
Jobs 62
Programa autoduplicable 55
Programación en ensamblador 52
Puertas serie
poner velocidad de transmisión 94
Punteado 164

Q

QDOS 29

control del sistema 30
Entrada/Salida 35
experimentando con el 37
extendiendo el 102
número de versión 47,70

R

Recolorear una ventana 50
Redireccionar TRAP/Excepción 77
Registros (68008) 14
de datos 14
de dirección 15
de estado 16
Relleno de áreas 180
Relleno de bloques 174
Reloj
ajuste 97
lectura 95
poner en hora 96
programa para mostrarlo 53
REPORT, instrucción BASIC 362
RESPR 40,87
Rutinas aritméticas
asignar espacio en la pila 245
ejecutar lista de operaciones 251
ejecutar operaciones 250
Rutinas de conversión
binario a octeto 234
binario a palabra 235
binario a palabra larga 236
cadena a entero 233
cadena a coma flotante 232
coma flotante a cadena 224
entero a cadena 225
hexadecimal a octeto 237
hexadecimal a palabra 238
hexadecimal a palabra larga 239
obtiene cadena con fecha/hora 222
obtiene dia de la semana 223
octeto a binario 226
octeto a hexadecimal 229
palabra a binario 227
palabra a hexadecimal 230
palabra larga a binario 228
palabra larga a hexadecimal 231
Rutinas en modo supervisor
asignación de área común 196
liberar área común 197
manejo de la cola serie 198
manejo de E/S serie 200
Rutinas de TRAP simplificadas 193
escribir mensaje de error 205
mandar entero ASCII 207
mandar mensaje 208
preparar consola 203
preparar ventana 202
preparar ventana de pantalla 204
Rutinas de utilidad general
desconectar un item 210
encadenar a una lista 209
gestión del área de usuario 211
rutinas de manejo de colas 213
utilidad de nombre de dispositivo 218
Rutina de conversión de fecha 222
Rutina de conversión de hora 222
Rutina de servicio de 'polling' 106
Rutinas de servicio 50/60 Hz 106
Rutinas de manejo de colas
Suspender Jobs 47,80

T

Tamaño de los caracteres 173
Tarjetas periféricas 280
Teclado
lectura 91
versión inglesa 92
versión española 359
Tipo de letra 163
Tipos de TOKEN 323
Tipos de variables en SuperBASIC 284
Tomar un octeto de una cola 216
TRA, instrucción BASIC 360
TRAPS de asignación de E/S
abrir un canal 119
borrar un fichero 123
cerar un canal 121
formatear un medio 122
sección de referencia 119
TRAPS, generación desde el BASIC 44
TRAPS del Gestor 61
activar Jobs 82
ajustar el reloj 97
asignar área BASIC 98
asignar área común 100
asignar procedimientos residentes 87
asignar área del usuario 85
borrar un Job 74
buscar memoria libre 76
cambiar la prioridad a los Jobs 83
conectar controlador de directorio 109
conectar controlador de dispositivo 109
conectar rutina de 'polling' 106
conectar rutina de servicio 105
conectar una tarea al planificador 107
crear un Job 72
forzar la terminación de un Job 75
gestión de memoria 61
información del sistema 70
leer el reloj 95
liberar área de BASIC 99
liberar área común 101
liberar área procedimiento residentes 88
liberar espacio común 86
liberar un Job 81
mandar un comando al IPC 93
obtener información de un Job 73
poner el reloj en hora 96
poner velocidad de transmisión 94
poner/leer modo de pantalla 89
sección de referencia 69
sumario 69
suspender un Job 80
TRAPS de gráficos
dibujar un arco 177
dibujar una elipse 178
dibujar una linea 175
poner la posición del cursor 181
rellenado si/no 181
TRAPS, proceso de 18
TRAPS, redireccionamiento 77
TRAPS del sistema de manejo de ficheros 182
cargar un fichero 190
comprobación de operaciones 183
leer la cabecera de un fichero 189
liberación de memoria intermedia 184
obtener información de un medio 187

comprobar estado 214
poner marcador fin de fichero 217
poner un octeto 215
preparar cola 213
tomar un octeto 216

S

Salida a pantalla 137
control del cursor 142
control de ventanas 140
controlador de pantalla del usuario 137
gráficos 175
limpiar ventanas 158
llenado de bloques 174
mover ventanas 155
poner atributos de carácter 170
poner borde ventana 140
poner colores 164
poner tipo de letra 163
redefinir ventanas 141
scroll de ventanas 152
obtener tamaño de pantalla 138,139
SBYTES 40
Scroll de ventanas
parte inferior 154
parte superior 153
toda la ventana 152
sector
escribir 259
leer 258
leer cabecera 261
verificar 260
Serie
manejo de colas 198
manejo de E/S 200
SEXEC 40
SINH, ejemplo de función residente 252
Sistema de ficheros
bloque de definición de canal 274,339
nombres 113
Soporte de procedimientos
asignar espacio en la pila 245
inicializar 240
parámetro de retorno 246
tomar parámetro de coma flotante 242
tomar parámetro de cadena 243
tomar parámetro entero 241
tomar parámetro entero largo 244
Sumario de TRAPS 324
de asignación de E/S 325
del gestor 324
Sumario de utilidades por vector 328
Sumario de utilización de TRAPS de E/S 325
SuperBASIC
almacenamiento de matrices 286
almacenamiento de variables 283
apuntador a los valores 285
apuntador a los nombres 285
área de valores de variables 286
área de memoria 31
asignación de memoria 64,98
liberar memoria 99
lista de nombres 286
suspender el Job 0 48
tabla de nombres 283
versión 41
Suprimir el cursor 143
poner el apuntador de un fichero 185,186
preparar cabecera de fichero 188
salvar un fichero 191

U

Utilidades por vector 192
modo supervisor 193,195
rutinas aritméticas 247
soporte de microdrives 195,255
soporte de procedimientos 240
TRAPS simplificados 193,202
Utilidades BASIC 194
Utilidad de copias de seguridad 194,209
Utilidades generales 343
Utilidad de nombre de dispositivo 218
Utilidades del SuperBASIC 194

V

Variables BASIC, nuevas 361
Variables del sistema 335
buscar la base de las 70
número aleatorio 59
Variables locales 288
Velocidad de transmisión 94
Ventanas
limpiar 158
mover 155
movimiento 49
obteniendo el tamaño 138,139
poner color al borde 140
preparar 202
preparar consola 203
preparar pantalla 204
recoloreado 50,166
redefinir 141
scroll 152
VER$ 41
Verificar sector de microdrive 260

W

WHEN, instrucción BASIC 362

QL Programación Avanzada.

Esta es la guía más avanzada del Sistema Sinclair QL. Proporciona una completa introducción al QDOS asumiendo un desconocimiento previo tanto del sistema operativo como del Ensamblador 68000.

El autor es miembro de la Universidad de Cambridge y tiene una considerable experiencia en escribir literatura de ordenadores avanzados. Esta guía de referencia ha sido cuidadosamente desarrollada por lo que resulta fácil de seguir y entender.

Se explican detalladamente la multitarea y otras potentes facilidades, ilustrados con numerosos ejemplos. Se incluye el listado completo de un programa experimentador en BASIC, que le permitirá acceder a muchas de las facilidades que proporciona el QDOS, directamente desde el Super BASIC.

Algunas de las áreas cubiertas son:

- "Jobs" y multitarea.
- Manejo de Entrada/Salida.
- Almacenamiento de ficheros.
- Procedimientos residentes.
- Programas transitorios.
- Pilas y áreas comunes de memoria.
- "TRAPs" del sistema y utilidades.
- Sección completa de referencia del 68008.

Ningún programador del QL debe quedarse sin una copia de este valioso manual.

Otros libros de RA-MA para QL:
- Manual de Referencia para Sinclair QL. *T. Hartnell.*
- QL, Libro de Juegos. *Hurley y Virgo.*

ra-ma
CHIQUINQUIRA, 28 (COCUY)
28033 MADRID.

ISBN 84-86381-08-8