Drives

di Maurizio Mangrella - Eboli (SA)

Una nota di colore prima di cominciare: nel momento in cui sto scrivendo è il 1° gennaio 1990 (come vedete la lista d'attesa per la rubrica software Amiga è piuttosto lunga, n.d.a.d.p.). Dopo aver festeggiato l'inizio del nuovo decennio con la scenografica (e «pornogastrica») esplosione di 800 petardi (diconsi ottocento!), sono qui abbastanza cosciente di me (nonostante gli assalti dello spumante traditore); anzi, come si suol dire dalle mie parti, «fresco e buono».

Nonostante spesso sia passata quasi inosservata, la capacità dell'Amiga di leggere dischetti in quasi tutti i formati di questo mondo è stata probabilmente una delle armi di successo di questo computer. In particolare i lettori di dischetti in formato MS-DOS e Atari ST (vedi Dos-2-Dos) e gli stessi emulatori di Intel 8088 (vedi Transformer 1.2) proliferano quasi come funghi. Vediamo come è possibile pilotare direttamente il DMA (Direct Memory Access, accesso diretto alla memoria) del disco.

Il disk DMA

Per quanto possa sembrare strano, la descrizione del funzionamento del disk DMA si riduce ad una asettica descrizione dei registri dei chip custom devoluti alla gestione dello stesso DMA: una volta conosciute le varie funzioni di cui questi registri si rendono capaci, la gestione del DMA praticamente vien fuori da sé.

Anzitutto ricordo che i chip custom di Amiga sono visti dal 68000 come un banco di registri, ognuno lungo una word (16 bit): questi registri iniziano a $DFF000 e terminano a $DFF1FF; pertanto, nel seguito, ometterò la specificazione delle prime tre cifre esadecimali: ad es., quando dirò «registro $01E» intenderò dire «registro $DFF01E».

Le informazioni che seguono sono state tratte in buona parte dall'«Hardware Reference Manual» (edizione corrispondente alla release 1.1 del KickStart, che sono riuscito a procurarmi, per giunta in maniera solo parziale, dopo molte peripezie), nel quale, oltre alle «ovvie» mancanze (e non è il solo manuale Amiga che ne è affetto), si registrano anche alcuni errori; quanto esporrò nel seguito, se nell'HRM non compare o è spiegato male, è frutto di mie personali sperimentazioni.

Il primo registro che ci interessa è ADKCON ($09E), che è una word conformata come rappresentato in tabella 1.

All'ADKCON (che è a sola scrittura) corrisponde il registro $010 (ADKCONR), che è a sola lettura e riporta fedelmente le impostazioni di ADKCON.

Per farvi alcuni esempi (spero) illuminanti, ricordo che i dischi MS-DOS e Amiga sono registrati in MFM Fast, mentre i dischi da 5'' 1/4 dell'AppleII dovrebbero (queste sono le voci NON confermate che corrono) essere registrati in GCR non-Fast. MFM sta per Modified Frequency Modulation, il che significa che i bit dati sono registrati con segnali di diverse frequenze; mentre GCR sta per Group Code Recording, una tecnica di registrazione che consiste nell'interpretare orientamenti magnetici uniformi del supporto (come SN SN o NS NS) come «0» e cambiamenti (come NS SN O SN NS) come «1». Entrambi i sistemi non riescono a riconoscere più di due o tre «0» in sequenza (pena de-sincronizzazione), pertanto i dati devono essere opportunamente convertiti. Ritorneremo sull'argomento per quanto riguarda la codifica MFM standard, in quanto, a proposito della conversione GCR, non posseggo dati sufficienti.

Altro registro: DMACON ($096), strutturato come in tabella 2.

Anche il DMACON è write-only, ed ha un corrispondente in lettura: DMACONR ($002); in particolare i bit 14 e 13 (BBUSY e BZERO) hanno un senso solo in lettura.

Prima di passare al sodo del disk DMA, diamo un'occhiata ai registri di interrupt. Essi sono INTENA ($09A, write-

Tabella 1

Bit	Funzione
15	Set-Clr (1 se i bit impostati a 1 devono settare i corrispondenti, 0 se devono resettarli)
14-13	PRECOMP 1-0 (%00: nessuna precompensazione, %01: 140 ns, %10: 280 ns, %11: 560 ns)
12	MFMPREC (1: lettura in MFM; 0: lettura in GCR)
11	UARTBRK (se impostato resetta i registri della posta seriale)
10	WORDSYNC (se attivo impone il sincronismo su una particolare word [vedi dopo])
9	MSBSYNC (se attivo impone il sync su un particolare bit; GCR standard Apple)
8	FAST (1: lettura con timing di 2 microsecondi per bit; 0: 4 microsec. per bit)
7-0	Impostazioni per le modulazioni dei canali audio (non ci interessano)

Tabella 2

Bit	Funzione
15	Set-Clr (vedi ADKCON)
14	BBUSY (Blitter Busy: settato se il Blitter è al lavoro)
13	BZERO (Blitter Zero: settato se l'ultima «blittata» ha dato un risultato completamente nullo; utile per il confronto di aree)
12-11	Don't Care
10	BLTPRI (Blitter Priority: se settato il Blitter ha priorità maggiore del 68000 negli accessi alla memoria)
9	MASTER (bisogna settarlo per attivare gli altri DMA impostati)
8	BPLEN (se settato attiva il DMA dei bit plane)
7	COPEN (se settato attiva il DMA del Copper)
6	BLTEN (se settato attiva il DMA del Blitter)
5	SPREN (se settato attiva il DMA degli sprite)
4	DSKEN (se settato attiva il DMA dei dischi [è lui!!])
3-0	AUD×EN (attivano i DMA dei canali audio)

only), INTENAR ($01C, read-only), INTREQ ($09C, write-only) e INTREQR ($01E, read-only). INTENA serve per abilitare gli interrupt, INTREQR serve (in caso di interrupt) a sapere quali sono attivi, cioè quali richieste hanno dato luogo all'interrupt (che, nell'Amiga, è quello di livello 1). La struttura è rappresentata in tabella 3.

Se uno dei bit settati in INTREQ corrisponde a un bit settato in INTENA (in altre parole, se l'AND tra INTREQ e INTENA è non nullo), viene generato un interrupt 1. Per evitare che i chip custom continuino a generare lo stesso Interrupt, bisogna azzerare «manualmente» INTREQ leggendo INTREQR e riscrivendo la stessa word (con il bit 15 resettato) in INTREQ.

E cominciamo con i registri dedicati al disk DMA. Il più «strano» è DSKBYTR ($01A), strutturato come in tabella 4.

Il disk DMA è attivo solo se l'AND logico tra il bit 9 di DMACON (DSKEN) e il bit 15 di DSKLEN è 1 (in pratica, se sono entrambi attivi). Notare come la lunghezza del DMA vada impostata in word (quindi il numero di byte da leggere deve essere pari).

Infine, il registro DSKDAT ($026) può essere utilizzato per scrivere direttamente delle word al dischetto (impostandole, appunto, in questo registro); analogamente DSKDATR ($008) può essere invece utilizzato per leggere direttamente da disco. Altri registri importanti sono DSKPT ($020, lungo una long-word) che contiene l'indirizzo a 19 bit (da 0 a 522487 nei 512K di CHIP memory) dei dati da leggere/scrivere dal/al disco, e DSKSYNC ($07E) che contiene la word di sincronismo. Il nuovo Fat Agnus (l'8372, che farà parte del bundle dell'Enhanced Chip Set) gestirà 1 mega di memoria, quindi tutti i puntatori saranno a 20 bit (per valori tra 0 e 1048575).

A proposito di sincronismo, il sincronismo tipico del formato standard MFM è $4489 (un pattern che in binario è $0100010010001001, facilmente riconoscibile dall'hardware predisposto alla gestione a basso livello del disco). L'Amiga, però, contempla anche un'altra forma di sincronismo (ben nota a chi fa uso frequente di copiatori...): l'Index Sync. L'Index è un sensore ottico (che avverte il passaggio di un apposito foro sui dischetti a 5" 1/4) o magnetico (che avverte il passaggio di un piccolo ma-

Tabella 3

Bit	Funzione
15	Set-Clr (al solito!)
14	INTEN (è un MASTER degli interrupt: deve essere impostato solo in scrittura)
13	EXTERN (settato se è occorso un interrupt esterno [???])
12	DSKSYNC (settato se la word di sincronismo è stata trovata tra i dati del dischetto)
11	RBF (Receive Buffer Full: per le comunicazioni seriali)
10-7	AUDx (fine della tavola dati per i canali audio 3-0)
6	BLIT (settato se il Blitter ha finito)
5	VERTB (settato se è avvenuto un Vertical Blank)
4	COPER (settato se il Copper ha voluto così...)
3	PORTS (settato se uno dei CIA genera un interrupt)
2	SOFT (riservato [???])
1	DSKBLK (settato se il DMA del disco è stato completato)
0	TBE (Transmit Buffer Empty: per le comunicazioni seriali)

Tabella 4

Bit	Funzione
15	DSKBYT (resettato se un nuovo byte è stato letto dal dischetto)
14	DMAON (AND tra il bit 9 di DMACON e il bit 15 di DSKLEN [vedi dopo])
13	DSKWRITE (immagine del bit 14 di DSKLEN [vedi dopo])
12	WORDEQUAL (settato se i byte provenienti dal dischetto corrispondono alla word di sincronismo)
11-8	Don't Care
7-0	Ultimo byte letto dal disco

Il registro DSKLEN ($024 contiene le seguenti informazioni:

Bit	Funzione
15	DMAEN (se attivo insieme al bit 9 di DMACON consente di effettuare il DMA da disco)
14	WRITE (1: DMA in scrittura sul disco; 0: DMA in lettura)
13-0	LENGTH (numero di word che devono essere lette da dischetto)

Tabella 5

Bit	Funzione
7	MTR (se attivo [resettato] accende il motore)
6	SEL3 (se attivo [resettato] seleziona il drive DF3:)
5	SEL2 (se attivo [resettato] seleziona il drive DF2:)
4	SEL1 (se attivo [resettato] seleziona il drive DF1:)
3	SEL0 (se attivo [resettato] seleziona il drive DF0:)
2	SIDE (faccia del disco; 0: faccia superiore; 1: faccia inferiore)
1	DIR (direzione di movimento delle testine; 0: verso la traccia 79; 1: verso la traccia 0)
0	STEP (se portato prima a 0, poi a 1 fa muovere le testine di una traccia)

Altra locazione importante è la $BFE001 (Porta A del CIA #1); ne considereremo solo i bit pertinenti allo stato del drive selezionato:

Bit	Funzione
5	RDY (Ready: resettato ogni qual volta il drive è pronto)
4	TK0 (resettato se le testine sono sulla traccia 0)
3	WPRO (resettato se il disco nel drive è protetto in scrittura)
2	CHNG (resettato se il disco è stato estratto dal drive; viene aggiornato ogni volta che le testine vengono mosse)

gnete posto sul volano di stabilizzazione dei motori di rotazione nei drive a 3" 1/2) che periodicamente avverte il computer della posizione del disco rispetto alle testine; i dischetti formattati secondo l'Index si dicono «hard sectored», quelli formattati secondo sincronismo in lettura si dicono «soft sectored». Nel caso dell'Amiga l'Index è adoperato raramente in quanto è poco preciso: in caso di passaggio dell'Index il CIA #2 può generare un interrupt di livello 6, che, per essere servito, richiede sempre una manciata abbondante di microsecondi. Invece di far generare un interrupt, è possibile testare in continuazione (polling) il bit 4 della locazione $BFDD00 (Interrupt Control Register del CIA #2), che vale 1 se in quel momento l'Index è attivo.

Prima di definire cosa dovremo fare per leggere questi benedetti dischetti (è passata la voglia anche a me!), diamo un'occhiata alla locazione $BFD100 (Porta B del CIA #2), che controlla alcuni segnali necessari per il funzionamento dei drive (vedi tabella 5).

Esistono delle temporizzazioni precise per quanto riguarda le operazioni che è possibile svolgere sui drive. Normalmente è buona norma attendere che la linea RDY (bit 5 della locazione $BFE001) vada a 0 (lo stato di questa linea è attendibile solo se il motore è acceso); in ogni caso valgono le seguenti regole ufficiali:

— dopo l'accensione del sistema, bisogna attendere 800 millisecondi per il setup dei drive;
— dopo l'accensione del motore, bisogna attendere almeno 500 ms;
— lo spostamento di una testina richiede almeno 3 ms; inoltre, tra i due impulsi inviati sulla linea STEP (vedi sopra) devono passare almeno 4 microsecondi;
— quando si cambia direzione, bisogna attendere almeno 18 ms (settling time delle testine);
— dopo la fine di un DMA bisogna attendere almeno 1.2 millisecondi prima di accedere al drive per qualunque altra operazione.

E adesso vediamo in sequenza cosa bisogna fare per accedere in DMA ai dischi. La prima cosa è accendere il motore; volendo, ad es., accendere il motore dei drive 1, potremo scrivere (in Assembler: è l'unico linguaggio che è consigliabile adoperare):

```
BSET    #4,$BFD100
BCLR    #7,$BFD100
BCLR    #4,$BFD100
```

In altre parole non abbiamo fatto altro che non-selezionare il drive 1 (portando la linea _SEL1 a 1), attivare il motore (linea _MTR a 0) e selezionare il drive 1 (_SEL1 a 0). Se vi state chiedendo perché bisogna fare così, sappiate che non sono in grado di darvi una risposta precisa...

Quindi, dobbiamo impostare i nostri bravi interrupt e i registri di controllo del DMA; non credo che sia il caso di dilungarsi troppo, in quanto un ulteriore tentativo di spiegazione probabilmente mi varrebbe il linciaggio... Preferisco rimandarvi al listato del programmetto che presento (v. dopo), che spero di aver commentato estensivamente.

A questo punto, dobbiamo attendere un opportuno sincronismo: avremmo

```
#ifndef RESOURCES_DISK_H
#define RESOURCES_DISK_H
/*
**  $Filename: resources/disk.h $
**  $Release: 1.3 $
**
**  external declarations for disc resources
**
**  (C) Copyright 1985,1986,1987,1988 Commodore-Amiga, Inc.
**      All Rights Reserved
*/

/* Resource structures */

struct DiscResourceUnit {
    struct Message  dru_Message;
    struct Interrupt dru_DiscBlock;
    struct Interrupt dru_DiscSync;
    struct Interrupt dru_Index;
};

struct DiscResource {
    struct Library      dr_Library;
    struct DiscResourceUnit  *dr_Current;
    UBYTE               dr_Flags;
    UBYTE               dr_pad;
    struct Library      *dr_SysLib;
    struct Library      *dr_CiaResource;
    ULONG               dr_UnitID[4];
    struct List         dr_Waiting;
    struct Interrupt    dr_DiscBlock;
    struct Interrupt    dr_DiscSync;
    struct Interrupt    dr_Index;
};

/* dr_Flags entries */
#define DRB_ALLOC0 0  /* unit zero is allocated */
#define DRB_ALLOC1 1  /* unit one is allocated */
#define DRB_ALLOC2 2  /* unit two is allocated */
#define DRB_ALLOC3 3  /* unit three is allocated */
#define DRB_ACTIVE 7  /* is the disc currently busy? */

#define DRF_ALLOC0 (1<<0)  /* unit zero is allocated */
#define DRF_ALLOC1 (1<<1)  /* unit one is allocated */
#define DRF_ALLOC2 (1<<2)  /* unit two is allocated */
#define DRF_ALLOC3 (1<<3)  /* unit three is allocated */
#define DRF_ACTIVE (1<<7)  /* is the disc currently busy? */

/* Hardware Magic */
#define DSKDMAOFF  0x4000  /* idle command for dsklen register */

#define DISKNAME  "disk.resource"

#define DR_ALLOCUNIT  (LIB_BASE - 0*LIB_VECTSIZE)
#define DR_FREEUNIT   (LIB_BASE - 1*LIB_VECTSIZE)
#define DR_GETUNIT    (LIB_BASE - 2*LIB_VECTSIZE)
#define DR_GIVEUNIT   (LIB_BASE - 3*LIB_VECTSIZE)
#define DR_GETUNITID  (LIB_BASE - 4*LIB_VECTSIZE)
#define DR_LASTCOMM   (DR_GIVEUNIT)

/* drive types */

#define DRT_AMIGA       (0x00000000)
#define DRT_37422D2S    (0x55555555)
#define DRT_EMPTY       (0xFFFFFFFF)

#endif  /* RESOURCES_DISK_H */

struct Interrupt {
    struct  Node is_Node;
    APTR    is_Data;            /* server data segment  */
    VOID    (*is_Code)();       /* server code entry    */
};
```

Figura 1 - L'header <resources/disk.h> (con una piccola appendice).

```
/* Tracker.c      by Maurizio Mangrella  1990
 *
 * SINTASSI: Tracker [drive [traccia [sync [mode [righe da mostrare]]]]]
 */

/* Includes e definizioni utili */

#include <exec/types.h>
#include <exec/memory.h>
#include <exec/io.h>
#include <devices/trackdisk.h>
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <proto/exec.h>

#define Printable(c) ((isprint(c)) ? c : '.')
#define BUFLENGTH 0x00003000L
/* lunghezza del buffer in bytes */

extern void ReadTrack(ULONG,UBYTE *,ULONG,ULONG,ULONG);

int myatoi(char *s) {
  int val;

  if(*s == '$') stch_i(&s[1],&val);  else val = atoi(s);
  return val;
}

UBYTE buf[9];

void main(int argc,char *argv[]) {
  UBYTE *ioBuf;
  UWORD *ioBufW;
  ULONG *ioBufL;
  int drive = 0,track = 0,sync = 0x00004489,mode = 0x00001500,rows = 30;
  struct MsgPort *myport;
  struct IOExtTD *myreq;
  short i,j,k;

  printf("Tracker          by Maurizio Mangrella  1990\n\n");

  /* Esame della command line ed eventuale sostituzione dei default */

  if(argc > 1) drive = myatoi(argv[1]);
  if(argc > 2) track = myatoi(argv[2]);
  if(argc > 3)  sync = myatoi(argv[3]);
  if(argc > 4)  mode = myatoi(argv[4]);
  if(argc > 5)  rows = myatoi(argv[5]);

  /* Alloca la memoria per il buffer di lettura */

  ioBuf = AllocMem(BUFLENGTH,MEMF_CHIP);
  ioBufW = (UWORD *)ioBuf; ioBufL = (ULONG *)ioBuf;

  /* Apre la trackdisk.device */

  myport = CreatePort("RTTestPort",PA_SIGNAL);
  myreq = (struct IOExtTD *)CreateExtIO(myport,sizeof(struct IOExtTD));
  OpenDevice(TD_NAME,drive,(struct IORequest *)myreq,0L);

  /* Si posiziona sulla traccia richiesta e la legge */

  myreq->iotd_Req.io_Command = TD_SEEK;
  myreq->iotd_Req.io_Offset = 11264*track;
  DoIO((struct IORequest *)myreq);
  ReadTrack(drive,ioBuf,sync,BUFLENGTH/2,mode);

  /* Stampa le righe con conversione MFM */

  for(i = 0;i < rows;i++) {
    buf[8] = '\0';
    for(j = 0;j < 8;j++) {
      buf[j] = '\0';
      for(k = 0;k < 16;k+=2) buf[j] |= (((*ioBufW)>>k)&0x0001)<<(k>>1);
      buf[j] = Printable(buf[j]);  ioBufW++;
    }
    printf("%06X: %08X %08X %08X %08X   %s\n",
      ioBufL,*ioBufL++,*ioBufL++,*ioBufL++,*ioBufL++,buf);
  }

  /* Ritorna ordinatamente al sistema */

  CloseDevice((struct IORequest *)myreq);
  DeleteExtIO((struct IORequest *)myreq,sizeof(struct IOExtTD));
  DeletePort(myport);
  FreeMem(ioBuf,BUFLENGTH);
}
```

Figura 2 - Tracker.c.

potuto far fare tutto a Paula (che è il chip custom che gestisce, tra le altre cose, il disk DMA), ma, dato che vogliamo la massima versatilità... dobbiamo anche sbrigarcela da soli! Infine, tempestivamente, setteremo la lunghezza del DMA (e la direzione: il tutto scrivendo una opportuna word in DMALEN), attenderemo un interrupt di livello 1 (che ne segnerà la fine) e rimetteremo tutto a posto.

Nello standard MFM i dati, prima della registrazione, sono sottoposti ad una particolare conversione: ogni bit originario viene convertito in due bit destinazione (per cui i byte diventano word), secondo le specifiche seguenti:

Bit sorgente	Bit precedente	Coppia risultante
0	0	10
0	1	00
1	Don't Care	01

Per cui, ad es., la sync word $4489, opportunamente tradotta, corrisponderà a $A1. Il programma che allego (figg. 2 e 3) si chiama Tracker; si compone di due sezioni: una in C (Tracker.c: impostazione, posizionamento della testina e visualizzazione dei dati con conversione MFM) e una in Assembler (Tracker Support.asm: la lettura vera e propria). Disponendo dell'ASSEM_DEVEL della Metacomco e del Lattice C V 5.0 possiamo formare Tracker con

```
assem TrackerSupport.asm -o Tracker
Support.o lc -v -L+TrackerSupport.à Tracker.c
```

Tracker si invoca da CLI con

```
Tracker [drive [traccia [sync word [mod [numero di righe da mostrare]]]]]
```

Tutti i parametri possono essere dati in esadecimale facendoli precedere dal simbolo '$'. <drive> è il drive cui accedere (da 0 a 3), <traccia> è la traccia da leggere (da 0 a 79 per i drive 3'' 1/2, da 0 a 39 per i 5'' 1/4), <sync word> è il sincronismo (solitamente $4489 o $AAAA), <mode> è una word così strutturata:

Bit	Funzione
15	Don't Care
14-13	PRECOMP 1- 0 (vedi ADKCON)
12	MFMPREC (vedi ADKCON)
11	Don't Care
10	WORDSYNC (vedi ADKCON)
9	MSBSYNC (vedi ADKCON)
8	FAST (vedi ADKCON)
7-2	Don't Care
1	Tipo di sincronismo (1: Index Sync, 0: sync impostato con i bit precedenti)
0	Faccia del drive (0: superiore, 1: inferiore)

e <numero di righe> è il numero di righe hex. (equivalenti a 8 byte effettivi ciascuna) da mostrare sul video. Il pro-

gramma sposterà le testine facendo uso della trackdisk.device (a proposito: lo sapevate che il SEEK della trackdisk.device riconosce 81 tracce? Io no!!!), quindi trasferirà il controllo alla funzione ReadTrack; questa cambierà stato al LED di Power-On (se è acceso lo spegnerà, e viceversa) una prima volta per segnalare il riconoscimento del sync, e una seconda per segnalare la fine del DMA (questo è servito a me per il debugging). Se avete sottomano un dischetto di sistema MS-DOS, provate a ficcarlo nel drive 0 e date:

Tracker 0 0 $4489 $1500 300

Attendendo l'uscita di tutti i dati sullo schermo, dovreste aver modo di notare la FAT (con i nomi di tutti i file sul disco), e, se siete fortunati (se avete «acchiappato» il settore giusto) anche le famose frasi «Non System Disk or Disk Error», «Replace and strike any key when ready». Altra curiosità: il drive 1581 (il drive 3" 1/2 per Commodore 64) gestisce i dischi in formato MFM standard (con tanto di separazione tra settori creata con byte $4E): l'unica differenza rispetto all'MS-DOS è che il numero di settori per traccia è 10, non 9.

Prestate attenzione nell'uso di Tracker perché il programma non è troppo educato, in quanto blocca il multitasking e si impadronisce dei drive senza troppi complimenti. In particolare, non settate sincronismi incompatibili: se impostate l'Index Sync (bit 1 della word <modes>) impostate la word <sync> a 0 (qualunque altro valore senza senso, come $FFFF, va bene ugualmente) e resettate i bit 9 e 10 nella word <modes> in modo da non confondere la routine di lettura. Ho preferito dotare Tracker del minimo indispensabile per porlo in condizione di funzionare (senza troppi orpelli), piuttosto che colmarlo di gadget e di funzioni, il che ne avrebbe diluito il valore esemplificativo in inutili dilungamenti rendendolo non pubblicabile per ragioni di spazio.

La disk.resource

Sembra facile, eh? Purtroppo dobbiamo anche fare i conti con la filosofia costruttiva di Amiga: il multitasking a più non posso. Le risorse hardware, come quelle software, devono sempre essere condivisibili tra i vari task appartenenti al sistema (o, perlomeno, devono essere disponibili a qualunque task ne faccia richiesta per primo). Allo scopo di inserire nel quadro organico dello «sharing» anche le risorse hardware, il sistema operativo di Amiga prevede le «resources», che, nella maggior parte dei casi altro non sono se non librerie gestite in maniera un po' cervellotica.

Nel caso specifico della gestione dei dischi, è disponibile la disk.resource. Aprirla, ovviamente, è sempre semplicissimo:

```
struct DiskResource *drlib;
.
.
drlib = (struct DiskResource *)OpenResource(DISKNAME,0L);
```

drlib è un puntatore a una struct Di-

```
* Tracker_Support.asm                    by Maurizio Mangrella  1990
*
* Funzione definita:
*  ReadTrack(drive,data buffer,sync,length,modes)
*  (<length> e' la lunghezza del buffer in words)

* Definizioni utili

_SysBase        EQU     4
CUSTOMREGS      EQU     $DFF000
CIAAPRA         EQU     $BFE001
CIABPRB         EQU     $BFD100
CIABDDRB        EQU     $BFD300
CIABICR         EQU     $BFDD00
INT1VECT        EQU     $0064

DMACONR         EQU     $002
ADKCONR         EQU     $010
DSKBYTR         EQU     $01A
INTENAR         EQU     $01C
INTREQR         EQU     $01E
DSKPT           EQU     $020
DSKLEN          EQU     $024
DSKSYNC         EQU     $07E
DMACON          EQU     $096
INTENA          EQU     $09A
INTREQ          EQU     $09C
ADKCON          EQU     $09E

                XREF    _LVOForbid
                XREF    _LVOPermit

                XDEF    _ReadTrack

* Il "valore di ozio" cui si fara'
* spesso riferimento per DSKLEN e'
* $4000,    corrispondente    alla
* scrittura  di 0 words senza abi-
* litazione del DMA.

* Talvolta alcune impostazioni nei
* registri  vengono  ripetute:  la
* sperimentazione pratica ha dimo-
* strato che i registri stessi so-
* no un po'  "sordi" ... quindi e'
* meglio gridare ...

_ReadTrack      MOVEM.L A2/A6,-(SP)             ;salva i registri
                MOVE.L  _SysBase,A6             ;una routine di Exec
                JSR     _LVOForbid(A6)          ;inibisce il multitasking
                MOVE.L  12(SP),D1               ;preleva <drive>
                ADDQ.W  #3,D1                   ;aggiunge 3 (ora va da 3 a 6)
                MOVE.B  #$FF,CIABDDRB           ;porta B in scrittura
                BSET    D1,CIABPRB              ;non-seleziona il drive
                BCLR    #7,CIABPRB              ;attiva il motore
                BCLR    D1,CIABPRB              ;seleziona il drive
                MOVE.L  D1,-(SP)                ;salva temporaneamente D1
                MOVE.W  30(SP),D0               ;preleva <mode>
                ANDI.B  #%00000001,D0           ;isola il bit 0 (faccia)
                ASL.B   #2,D0                   ;lo sposta di due a sinistra
                EORI.B  #%00000100,D0           ;inverte il bit
                MOVE.B  CIABPRB,D1              ;preleva il dato nella porta B
                ANDI.B  #%11111011,D1           ;tutti i bit fuorche' il 2
                OR.B    D0,D1                   ;vi imposta il bit manipolato
                MOVE.B  D1,CIABPRB              ;e il gioco e' fatto
                MOVE.L  (SP)+,D1                ;riprende D1
                MOVE.L  INT1VECT,OldIRQ1Vect    ;salva il vettore interrupt 1
                MOVE.L  #MyIRQ1Routine,INT1VECT ;imposta il nostro server
                CLR.W   EndFlag                 ;pulisce il flag di fine
                MOVE.B  #$1F,CIABICR            ;azzera tutti gli interrupt 6
                LEA     CUSTOMREGS,A2           ;forma l'indirizzo dei registri
                MOVE.W  INTENAR(A2),OldINTENA   ;salva INTENA
                MOVE.W  DMACONR(A2),OldDMACON   ;salva DMACON
                MOVE.W  ADKCONR(A2),OldADKCON   ;salva ADKCON
                MOVE.W  30(SP),D0               ;preleva <mode>
                ANDI.W  #$7700,D0               ;maschera i bit utili
                ORI.W   #$8000,D0               ;imposta il flag Set-Clr
                MOVE.W  #$0002,INTREQ(A2)       ;pulisce l'IRQ di DISKBLOCK
                MOVE.W  #$3FFF,INTENA(A2)       ;disabilita tutti gli int. 1
                MOVE.W  #$C002,INTENA(A2)       ;attiva solo MASTER e DSKBLK
                MOVE.W  #$C002,INTENA(A2)       ;meglio ripetere per sicurezza
                MOVE.W  #$4000,DSKLEN(A2)       ;"valore di ozio" per DSKLEN
```

skResource (vedi fig. 1), la quale comprende una struct Library (preceduta in memoria dalla jump table delle routine della resource), un puntatore a struct DiskResourceUnit, alcuni flag, i puntatori alla libreria Exec e alla libreria connessa alla ciab.resource, gli IDs dei drive, una lista (di che???) e alcune struct Interrupt (fig. 1).

Gli interrupt disponibili sono i soliti: trasferimento completato di un blocco del disco, risincronizzazione dello stesso, passaggio dell'indice sotto il rilevatore. Questi interrupt sono ripetuti nella struct DiskResourceUnit; se non ho sbagliato i calcoli, le struct DiskResourceUnit formano in realtà una lista bidirezionale attraverso il nodo mn_Node contenuti nella struct Message di ognuna: probabilmente l'item dr_Current punta alla DiskResourceUnit correlata all'unità a disco correntemente interessata da un DMA.

I flag ospitano nei bit da 0 a 3 altrettanti indicatori dell'avvenuta installazione delle rispettive unità nel sistema: un bit a 1 indica che l'unità correlata è montata a vista dal sistema; 0 viceversa. Il bit 7 indica se è in corso un DMA del disco lanciato da un task del sistema: si dovrebbe sempre attendere il reset di questo bit prima di intraprendere una qualunque operazione sul disco.

UnitID è un array che ospita le IDs delle rispettive unità: le IDs disponibili sono DRT_AMIGA (0×00000000, drive standard 3''1/2 per Amiga), DRT_37422D2S (0×55555555: qualche anima buona mi spiegherà il significato della misteriosissima sigla, che credo si riferisca alle unità Shugart 5''1/4) e DRT_EMPTY (0×FFFFFFFF, riservato alle unità non montate).

Le routine sono «raggiungibili» attraverso i seguenti LVOs (Library Value Offsets):

Entry	LVO	
DRALLOCUNIT	−6	(0×FFFA hex.)
DRFREEUNIT	−12	(0×FFF4 hex.)
DRGETUNIT	−18	(0×FFEE hex.)
DRGIVEUNIT	−24	(0×FFE8 hex.)
DRGETUNITID	−30	(0×FFE2 hex.)

L'unica routine che sono riuscito ad interpretare correttamente è, probabilmente, una delle più utili: è la DR_GETUNITID, che si può chiamare passando in A6 il puntatore alla struct DiskResource restituito dalla OpenResource() e in D0 il numero dell'unità (da 0 a 3): vi viene restituita in D0 l'ID dell'unità richiesta. Io l'ho chiamata da Assembler, ma credo che si possa farne uso dal Lattice C 5.0 costruendo un opportuno header «proto».

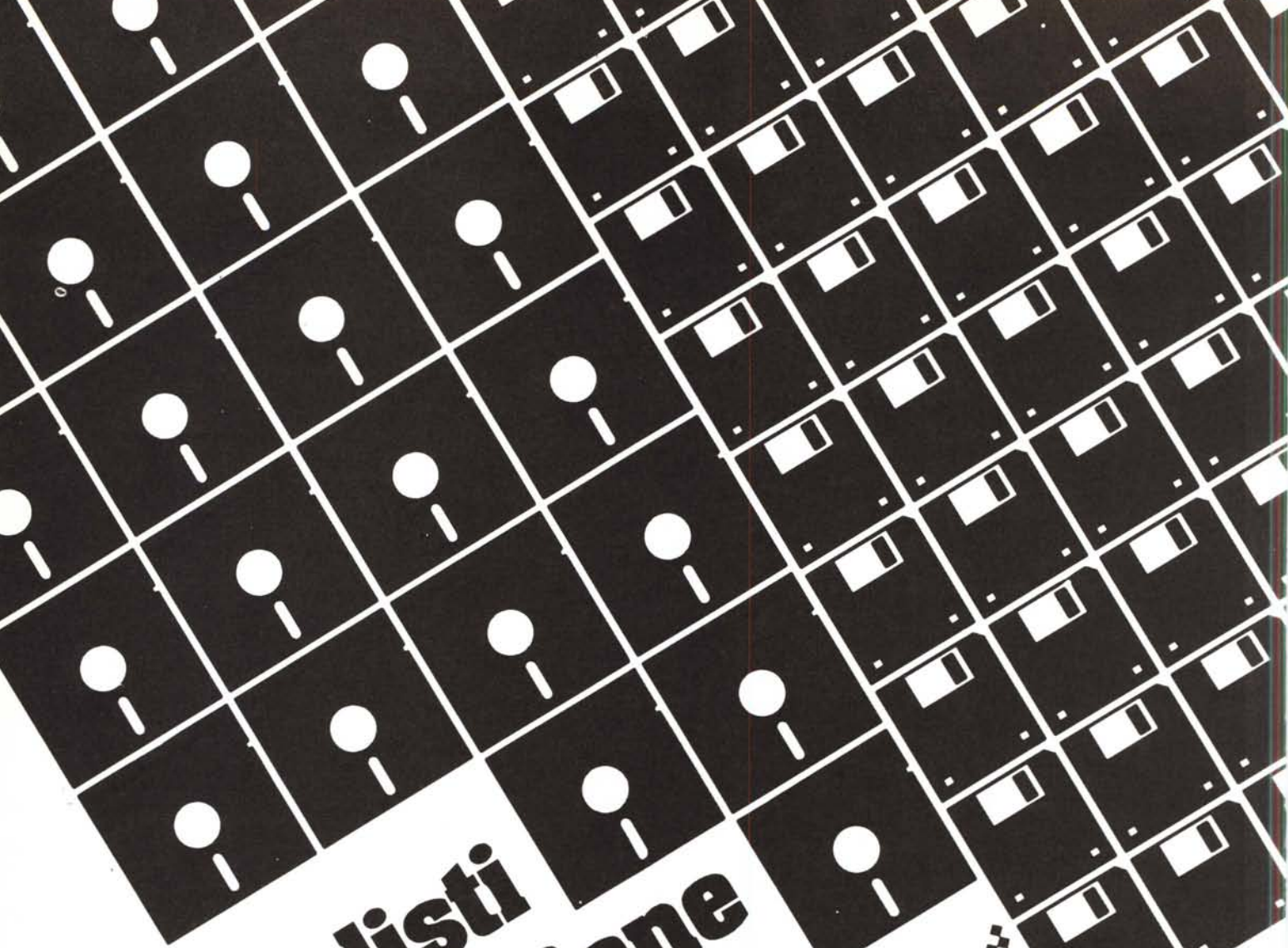
Conclusioni

Poche parole per le conclusioni di rito: vi invito solo a sperimentare Tracker (magari costringendolo a mostrare la corda...) e a usufruire della funzione ReadTrack per scrivere un editor di tracce o... un mega-copiatore (però evitate di sproteggere i programmi...).

MC

```
                MOVE.W  22(SP),DSKSYNC(A2)      ;imposta <sync> nel registro
                MOVE.W  #$7FFF,ADKCON(A2)       ;pulisce tutti i flag di ADKCON
                MOVE.W  D0,ADKCON(A2)           ;imposta i <mode> selezionati
                MOVE.L  16(SP),DSKPT(A2)        ;setta l'indirizzo del buffer
                MOVE.W  30(SP),D0               ;preleva <mode>
                BTST    #1,D0                   ;esamina il bit 1 (Index Sync)
                BEQ.S   WSync                   ;se 0 salta a WSync
                BSR     WaitIndex               ;attende l'Index Sync
                BRA.S   WaitSkip                ;piccolo salto
WSync           BSR     WaitSync                ;attende il sync impostato
WaitSkip        MOVE.W  26(SP),D0               ;preleva <length>
                ORI.W   #$8000,D0               ;abilita il flag DMAEN
                MOVE.W  D0,DSKLEN(A2)           ;imposta il registro DSKLEN
                MOVE.W  D0,DSKLEN(A2)           ;repetita iuvant
                MOVE.W  #$8210,DMACON(A2)       ;attiva MASTER e DSK in DMACON
1$              TST.W   EndFlag                 ;attende la fine del DMA
                BEQ.S   1$
                MOVE.W  #$4000,DSKLEN(A2)       ;"ozio" per DSKLEN
                MOVE.W  #$0002,INTREQ(A2)       ;azzera la richiesta di DSKBLK
                MOVE.W  OldDMACON(PC),D0        ;riprende il vecchio DMACON
                ORI.W   #$8200,D0               ;imposta Set-Clr e MASTER
                MOVE.W  #$7FFF,DMACON(A2)       ;pulisce DMACON ...
                MOVE.W  D0,DMACON(A2)           ;... e imposta il vecchio
                MOVE.W  OldADKCON(PC),D0        ;preleva il vecchio ADKCON
                ORI.W   #$8000,D0               ;setta il flag Set-Clr
                MOVE.W  #$7FFF,ADKCON(A2)       ;pulisce ADKCON ...
                MOVE.W  D0,ADKCON(A2)           ;... e imposta il vecchio
                MOVE.W  OldINTENA(PC),D0        ;riprende il vecchio INTENA
                ORI.W   #$C000,D0               ;setta i flag Set-Clr e MASTER
                MOVE.W  #$7FFF,INTENA(A2)       ;pulisce INTENA ...
                MOVE.W  D0,INTENA(A2)           ;... e imposta il vecchio
                MOVE.L  OldIRQ1Vect(PC),INT1VECT ;imposta il vecchio server
                MOVE.B  #$FF,CIABDDRB           ;porta B di CIA #2 in scrittura
                BSET    D1,CIABPRB              ;non-seleziona il drive
                BSET    #7,CIABPRB              ;spegne il motore
                BCLR    D1,CIABPRB              ;seleziona il drive richiesto
                BCHG    #1,CIAAPRA              ;cambia stato al LED di PowerOn
                MOVE.B  #$9F,CIABICR            ;attiva tutti gli interrupt 6
                JSR     _LVOPermit(A6)          ;riabilita il multitasking
                MOVEM.L (SP)+,A2/A6             ;riprende i registri
                RTS                             ;ritorna senza valore

MyIRQ1Routine   MOVEM.L D0/A0,-(SP)             ;salva i registri usati
                LEA     CUSTOMREGS,A0           ;forma l'indirizzo dei registri
                MOVE.W  INTREQR(A0),D0          ;preleva INTREQ
                ANDI.W  #$7FFF,D0               ;resetta il flag Set-Clr
                MOVE.W  D0,INTREQ(A0)           ;cancella tutte le richieste
                MOVE.W  #-1,EndFlag             ;setta il flag di fine DMA
                MOVEM.L (SP)+,D0/A0             ;riprende i registri
                RTE                             ;esce dall'Exception

WaitSync        MOVE.W  DSKBYTR(A2),D0          ;esamina DSKBYTR
                BTST    #12,D0                  ;testa il flag WORDEQUAL
                BEQ.S   WaitSync                ;attende il passaggio del sync
                BCHG    #1,CIAAPRA              ;cambia stato al LED di PowerOn
                RTS                             ;ritorna

WaitIndex       MOVE.B  CIABICR,D0              ;esamina gli interrupt 6
                BTST    #4,D0                   ;esamina la richiesta FLAG
                BEQ.S   WaitIndex               ;Index non ancora passato!
                BCHG    #1,CIAAPRA              ;cambia stato al LED di PowerOn
                RTS                             ;ritorna

* Area dati

OldIRQ1Vect     DC.L    0
OldINTENA       DC.W    0
OldDMACON       DC.W    0
OldADKCON       DC.W    0
EndFlag         DC.W    0

                END
```

Figura 3 - Tracker_Support.asm.



Specialisti in duplicazione

La Microforum di Toronto, Canada, produttrice dei famosi dischetti Mito, propone oggi al mercato italiano del software i suoi sofisticati impianti di duplicazione. Nel giro di pochi giorni, Microforum può assicurare la duplicazione dei vostri programmi, anche con protezione, con la massima accuratezza e a costi altamente competitivi. Se il vostro problema sono 1000 o 100.000 copie, scrivete o mandate un fax a

Microforum
MANUFACTURING INC.
TORONTO - CANADA

1 Woodborough Avenue, Toronto, Canada M6M 5A1
Tel. 001 416 656 6406 Fax 001 416 656 6368 Telex (06)23303

Ufficio di rappresentanza in Roma: Via Flaminia 215
Tel. 06/32 22 199