System Extension Modules

di Bruno Rosati

Ultima puntata "monstre" quella con la quale si chiude il primo ciclo del corso di RISC-OS. Il Filing System visto attraverso le funzionalità di servizio del FileXSwitch e del FileCore; le caratteristiche di un altro gruppo di "Extension modules"; ed infine il completamento della trattazione dei livelli d'interattività dell'unità di visualizzazione

C'eravamo lasciati due mesi orsono, affrontando, attraverso le funzionalità d'interazione con la VDU, le argomentazioni relative al Font Manager; il primo modulo di estensione che si andava ad analizzare nella sua struttura generale. Il modo di procedere alla ricognizione di tale modulo, proprio perché visto attraverso la "lente" dell'interattività, ci aveva fatto assumere un certo filo logico che anche in questa ultima puntata non abbandoneremo.

Dovendo tra l'altro completare il discorso sui livelli d'interazione e capitalizzare l'importanza dei Filing System, il sistema di estensione vedrà gli stessi moduli distribuirsi in tre diversi blocchi di argomentazione. Uno generale: System Extension Modules; gli altri particolarmente dedicati al Filing Systems e agli stessi livelli di interattività VDU-moduli di estensione, già intravisti la volta scorsa. A parte la necessità di continuare il discorso, tale modo di procedere ci rende una visione sinceramente più dinamica dei moduli stessi e lo preferiamo alla solita mera elencazione di nomi e caratteristiche.

Entrando nel merito, tutto questo tra l'altro trova conferma anche da una rilettura del Programmer's Reference Manual al riguardo della suddivisione fra Kernel e moduli di estensione. Riprendendo difatti la generica definizione data dal manuale (il Kernel quale atomo ed i moduli simili a degli "elettroni") ed anche in base a ciò che si è cominciato già a vedere attraverso i livelli d'interattività della VDU, questi "elettroni" non sono tutti di uguali specifiche, valenza ed importanza.

Sia chiaro, la definizione resta comunque valida, ma dovendo entrare nel particolare è inevitabile operare un'ulteriore distinzione fra ... elettrone ed elettrone.

In parole povere possiamo anche dire che i vari *Filing Systems*, la gestione delle finestre nella moderna concezione del WIMP, la gestione-font, l'emulatore del coprocessore matematico, il Sistema Sonoro, i Printer Driver, la gestione del colore e così via, possono essere tutti intesi come "elettroni" del microcosmo Archimedes, ma ad esempio: i *Filing Systems* ed il *Window Manager* non possono non esser considerati come due sistemi distinti da tutto il resto.

Non tanto perché non appartengano all'area dei moduli di estensione quanto per le rispettive valenze che li distinguono enormemente dal resto.

Invero, lo stesso PRM, se all'inizio dà la generica definizione, se uno osserva poi il modo con il quale sono ripartiti gli argomenti, si accorge immediatamente del fatto che, dal grosso capitolo *System Extensions*, sono tenuti fuori sia i *Filing Systems* che il *Window Manager*.

Procederemo perciò allo stesso modo. Dando una generica informazione dei moduli di estensione ed entreremo un po' più nel particolare per quanto riguarda gli altri due sistemi.

SOUND SYSTEM: CAPACITA' DEL BUFFER

1 channel	208 bytes	224 bytes	48microsec.
2 channels	416 bytes	448 bytes	24microsec.
4 channels	832 bytes	896 bytes	12microsec.
8 channels	1664 bytes	1792 bytes	6microsec.

Buffer period	0.9984cs	1.0752cs
Interrupt rate	100.16Hz	93,01Hz
Bytes x channel	&D0	&E0

SINGLE CHANNEL FORMAT

0	byte0	byte1	byte2	byte3	byte4	byte5	byte6	byte7
	chan1	chan1	chan1	chan1	chan1	chan1	chan1	chan1
+8	byte8	byte9	byte10	byte11	byte12	byte13	.....	
	chan1	chan1	chan 1	chan 1	chan 1	chan 1		

EIGHT CHANNEL FORMAT

0	byte0	byte0	byte0	byte0	byte0	byte0	byte0	byte0
	chan1	chan2	chan3	chan4	chan5	chan6	chan7	chan8
+8	byte1	byte1	byte1	byte1	byte1	byte1	.....	
	chan1	chan2	chan3	chan4	chan5	chan6		

Figura 1

System Extension Modules

Il quarto ed ultimo volume del Programmer's Reference Manual si apre con il modulo Econet usato dal RISC-OS

per il controllo di un eventuale sistema a rete. La delega alla gestione è imposta ai moduli NetFS e NetPrint che, guarda caso, sono argomenti propri da "Filing System". Li tratteremo più avanti nel capitolo dedicato. In questo è sufficiente procedere alla presentazione del modulo Econet che è (meglio chiarire subito) la sola parte software del sistema di rete vero e proprio: l'hardware della Econet-card.

In pratica una sorta di *amministratore* della memoria necessaria per il trasporto in trasmissione e ricezione delle informazioni inviate via rete. Il sistema Econet non usa, (differentemente dagli altri sistemi di Input/Output sotto RISC-OS) bufferizzare i propri dati.

La sua specificità è quella di muovere le informazioni direttamente dal modulo di rete alla memoria. Il modulo software omonimo allora s'incarica di provvedere a ciò, controllando e rendendo disponibili blocchi di memoria utile per il trasporto delle informazioni. Il procedimento, "anomalo" rispetto a tutto il sistema archimediano, è ovviamente indispensabile al fine di conservare i massimi requisiti di velocità di trasferimento.

L'intero meccanismo che ruota intorno al sistema-Econet parte dal concetto di *packet*, ovvero la procedura a "singola trasmissione", che porterà le informazioni contenute dalla stazione trasmittente a quella ricevente, formate in una modalità usualmente denominata a "quattro modi" di handshake. Questi modi (frame) dovranno contenere nei primi quattro byte l'informazione relativa al numero d'identificazione della stazione ricevente, il numero del network della stessa, il numero della stazione trasmittente e del suo network. In pratica le informazioni che necessitano al software Econet che poi gestirà l'informazione sui blocchi di memoria disponibili. L'intera gestione software avviene attraverso 24 chiamate ad interrupt, dedicate alla creazione di blocchi RCB (Receive Control Block) destinati al controllo della ricezione del packet (le informazioni di cui sopra e l'eventuale, pseudo-bufferizzazione che è in grado di effettuare); quindi uscite dalla trasmissione, setting di stati di protezione, allocazione e de-allocazione di porte, start primari, etc.

Sempre in tema di trasmissione dei dati, ma uscendo dalla rete... e trasmettendo dalla tastiera (e dal monitor) ad una "semplice" stampante, è particolarmente interessante vedere come il RISC-OS, attraverso il modulo dei Printer Drivers risolve il problema della stampa "universale". Ovvero come è possibile gestire direttamente da sistema qualsiasi tipo di stampante, senza distinzioni fra matrice di punti, stampa PostScript e le modalità specifiche dei plotter. Semplice e geniale (come tutto il RISC-OS d'altronde!) il Printer Driver si "ricostruisce" attraverso l'uso delle stesse chiamate utilizzate per scrivere allo schermo. SWI verso i VDU-driver al Draw module, il ColourTrans (per entrambi vedere la scheda relativa all'interattività con la VDU) e il Font Manager che verranno intrappolate dal printer driver (trapping) ed interpretate nel modo più appropriato a seconda della user-printer di cui è stato comunicato l'uso. Quindi tutto quanto riguarda il size e lo stile ricavabili attraverso le caratteristiche del Font Manager (*Internal Measurement System* compreso) e la conversione a video (le unità OS usate dai VDU-driver) verranno. interpretate dal printer driver con il metodo del Draw-module, con scale, rotazioni e traslazioni finali su carta.

Dalla scrittura alla musica: il Sound System. Il DMA e il Channel Handler, lo Scheduler ed i Generatori di Voci, sono la base di quest'altro complesso quanto raffinato sistema di estensione del RISC-OS sfruttabile pienamente su otto canali di uscita indipendenti. Insiemi di *facilities* per sintetizzare e quindi eseguire in notazione digitale campionature di suoni, voci "umane" ed effetti sonori in genere, sono predisposte dal sistema per la più completa gestione che, a livello di sistema operativo sia mai stata vista.

La bufferizzazione delle zone DMA è regolata dal sistema sonoro che usa due buffer per campionare ed immagazzinare segnali digitali come funzioni logaritmiche. Il dato prelevabile dal primo dei due buffer viene quindi letto e con-

ECId-Share: Identificatore di scheda (semplice)-1byte-

7	6	5	4	3	2	1	0
A	ID[3]	ID[2]	ID[4]	ID[0]	FIQ	0	IRQ

Bit	Value	Meaning
A	0	Acorn conformant expansion card
	1	Non conformant expansion card
ID[3:0]	not 0	ID field
	(0	extended ECId used)
FIQ	0	Not requesting FIQ
	1	Requisting FIQ
IRQ	0	Not Requesting IRQ
	1	Requesting IRQ

Figura 2

CHIAMATA AD INTERRUPT per l'identificazione di una scheda

Sintassi:	Podule_ReadID
IN:	R3= numero della slot della scheda
OUT:	R0= byte-identify (ECId) della scheda
USO:	Tale chiamata legge nel registro R0 un identificatore (della versione semplice) o il byte meno significativo di un'Extended Expansion Card Identify.
VETTORE RELAZIONALE:	Podule_ReadHeader

Figura 3

vertito in un segnale analogico; nello stesso tempo verrà anche scritto nel secondo buffer per essere infine passato al Generatore di voce che provvederà alla sua campionatura. Il Channel Handler a sua volta provvede al controllo del suono prodotto da ciascun canale e a mantenere le tavole interne necessarie al sistema per continuare a produrre gli stessi suoni.

Volume e tonalità, ampiezza e durata dei suoni vengono infine gestiti dallo stesso Channel Handler affinché sia possibile abbinare i diversi canali ed altrettanti Generatori di voce. I quali a loro volta, sono predisposti per generare un suono campionato da indirizzare al buffer del DMA.

Lo scheletro infine è una sorta di sequencer di bordo, usato per eseguire sequenze di note attraverso un contatore di bit auto-resettabile.

È ovvio che tutto ciò diviene estremamente potente nell'uso congiunto con la classica MIDI. Una scheda di espansione che, a parte le proprie caratteristiche e le informazioni sulle quali già in passato si è "articolato", ci serve da ideale anello di congiunzione con l'argomento che segue: il modulo di estensione Expansion Cards.

Ovvero quali requisiti debbono soddisfare e come verranno poi gestite a livello di RISC-OS, le eventuali schede di espansione inseribili nel backplane di Archie.

Anzitutto una scheda dev'essere in grado d'informare il modulo software delle Expansion Cards delle sue caratteristiche. Ovvero: fornire un'identificazione (ECId) così che il RISC-OS possa riconoscerla e quindi gestirla. Il segnale di identificazione dovrà apparire dopo ogni reset al momento della reinizializzazione del sistema.

In seconda istanza si rende necessario un controllo (interrupt) sullo stato dei puntatori con il quale la scheda stessa obbliga al sistema operativo dove indirizzarsi quando si genera uno stato d'interrupt. Terzo requisito: la presenza di un *Loader*, con il quale è possibile accedere alla memoria paginata presente sulla scheda e nella quale è contenuto del software (un modulo rilocabile) per la gestione o semplici informazioni aggiuntive da inviare al sistema.

L'ultima informazione necessaria è quella legata al concetto di *Chunk Directory* con il quale s'informa il sistema sul modo di usare l'eventuale ROM operativa presente sulla scheda stessa.

In un RISC-OS che pensa a tutto non può certo mancare il raffinato controllo dell'International Module; il settaggio del tipo di tastiera e l'alfabeto relativi alla nazionalità e quindi il tipo di linguaggio dell'utilizzatore. In pratica, a seconda della nazionalità e delle conseguenti differenze fra i tipi di tastiera e di caratteri usati, tale modulo comunica al RISC-OS l'eventuale selezione della mappa sia della *keyboard* che dei *characters*. A tal riguardo ogni "paese" viene archimedianamente rappresentato con un numero di riferimento come descritto in figura 4, osservando la quale è tra l'altro possibile notare della presenza di 6 differenti tipi di alfabeti adattabili in mappa sulla tastiera a seconda delle diverse combinazioni di tasti.

A prescindere dalle quattro chiamate ad interrupt (per leggere e scrivere il numero della nazione prescelta, l'alfabeto e la tastiera) sono a disposizione del programmatore i seguenti comandi:

* **Alphabet** (per selezionare un alfabeto)
* **Alphabets** (per listare i vari alfabeti installati)
* **Configure Country** (per settare a default alfabeto e tastiera)
* **Country** (per selezionare alfabeto e tastiera di una nazione imponendo il layout della tastiera posseduta)
* **Countries** (per listare le "nazioni" installate)
* **Keyboard** (per selezionare il driver-tastiera del numero-nazione desiderato).

Utile è anche il modulo Debugger, predisposto dal RISC-OS per il break ai programmi in esecuzione e quindi effettuare delle complete ricognizioni sugli stessi così come la lista dei comandi disponibili mostrati in figura 5 ci chiarisce.

Assai più complesso ed importante a livello di applicazione è certamente lo stra-famoso Floating Point Emulator. Il modulo di estensione che il RISC-OS rende all'utente come emulatore software al coprocessore matematico. Un modello equivalente questo in standard IEEE 754 in grado di gestire la propria operatività su otto registri di calcolo ad alta precisione fra i quali detiene un FP-Status Register.

I suoi valori sono quindi immagazzinati nella memoria dell'ARM in uno dei quattro formati descritti in figura 6.

TASTIERE ed ALFABETI INTERNAZIONALI

Codice	Keyboard	Alphabet
0	Default	Select the configured country
1	UK	Latin1
2	Master	BFont
3	Compact	BFont
4	Italy	Latin1
5	Spain	Latin1
6	France	Latin1
7	Germany	Latin1
8	Portugal	Latin1
9	Esperanto	Latin3
10	Greece	Greek
11	Sweden	Latin1
12	Finland	Latin1
13	Non used	
14	Denmark	Latin1
15	Norway	Latin1
16	Iceland	Latin1
17	Canada1	Latin1
18	Canada2	Latin1
19	Canada	Latin1
20	Turkey	Latin3
21	Arabic	Special-ISO 8859/6
22	Ireland	Latin1
23	Hong Kong	Non definito
80	ISO1	Latin1
81	ISO2	Latin2
82	ISO3	Latin3
83	ISO4	Latin4

CODICE RELATIVO ALLA SELEZIONE DEL SOLO ALFABETO

Code	Alphabet
100	BFont
101	Latin1
102	Latin2
103	Latin3
104	Latin4
107	Greek

Figura 4

COMANDI per il DEBUGGER

Command	Description
*BreakClr	Remove breakpoint
*BreakList	List currently set breakpoints
*BreakSet	Set a breakpoint at a given address
*Continue	Start execution from a breakpoint saved state
*Debug	Enter the debugger
*InitStore	Fill memory with given data
*Memory	Display memory between two address/register
*memoryA	Disassemble ARM instructions
*ShowRegs	Display registers caught by traps

Figura 5

Filing Systems

L'Advanced Disc Filing Systems (ADFS) usato nella gestione di floppy ed hard disk drive; il Network Filing System (NetFS) predisposto per l'accesso ai server Econet; il Ram Filing System (RamFS) quale management del disco virtuale realizzabile sulla memoria volatile ed infine il NetPrint, predisposto quale filing system per il controllo del server di stampa sempre in rete condivisa.

Sono questi i sistemi disponibili per "default" per l'organizzazione e l'accesso ai dati immagazzinati. Filing System "di serie" ai quali, come moduli, sono da aggiungere in lista: il Desktop Filing System (DeskFS) che fornisce supporto per la gestione del windowing e il controllo delle varie opzioni disponibili a livello di scrivania; e poi il modulo del System Devices. Un contenitore di routine per la gestione dell'Input/Output (tastiera, porta seriale, VDU, printer, etc.).

A tali moduli è poi possibile aggiungere, a discrezione del programmatore, altri moduli personali programmabili in base alle proprie esigenze. Un sistema estremamente elastico.

Orbene, fra Filing System di sistema, moduli aggiuntivi e programmabili, a livello di programmazione è ovvio che si presupponga la possibilità di poter gestire la cosa nel modo più pratico possibile. La disponibilità a livello di RISC-OS di un "qualcosa" simile ad una specie di "interfaccia-software" capace di gestire i sistemi e di renderci eventualmente la possibilità di commutare a richiesta da un filling all'altro.

Il "qualcosa" esiste ed ha nome e compiti ben definiti: FileSwitch. Il modulo è in grado di svolgere una serie di servizi sui e per i Filing System (controllo del loro stato attivo e quindi della presenza nel sistema) e la importante funzione di commutazione fra un FS e l'altro. Un limite, ovvio ma che è comunque il caso di non sottintendere, riguarda la modalità con la quale il FileSwitch controlla l'eventuale stato attivo del sistema: cosa questa che è svolta solo a livello di software. Nel senso che il FileSwitch è in grado di riconoscere solo la presenza del FS.

Tutto ciò che riguarda eventuali parti aggiuntive di hardware non sono verificabili dal FileSwitch, ma dal relativo filing system che le controlla e le introduce in configurazione. In definitiva il FileSwitch svolge tutte le funzioni tranne che quelle di accesso all'hardware. Quando ciò diventa necessario, il modulo stesso si rivolgerà al Filing System che governa l'hardware attraverso specifiche chiamate.

In linea teorica è possibile definire il FileSwitch come una sorta di "Main Filing System" e i vari ADFS, NetFS e via discorrendo, moduli aggiuntivi predisposti al controllo diretto di drive (hard, floppy e virtuali che siano) stampanti e network. Una simile definizione non è affatto azzardata, se consideriamo il fatto che il FileSwitch svolge in effetti gran parte della mole di lavoro in luogo del Filing System. Se osserviamo poi la figura 7 possiamo renderci ancor meglio conto di quale tipo di struttura è organizzata sotto RISC-OS a livello di gestione-filing system. Il File-Switch posto a diretto contatto con il Kernel rimane in relazione diretta solo con il Filing System di rete, eventualmente FS sviluppati personalmente dal programmer ed un certo FileCore. Un secondo modulo questo — in effetti vedremo che si tratta di un altro Filing System — posto a sua volta come interfaccia fra gli altri FS di sistema.

Come prima definizione per introdurre al concetto di FileCore possiamo pensare ad un modulo in grado di convertire le informazioni provenienti dal FileSwitch ed indirizzarle al Filing System che controlla l'hardware. Ma la sua vera definizione è quella di autentico Filing System. Ovvero la base comune a cui tutti i Filing System Modules aggiunti al sistema si agganciano con le loro (poche) specifiche. In pratica quelli che noi definiamo "Filing System" altro non sono che semplici moduli aggiuntivi che agganciano al FileCore le sole routine di accesso all'hardware a cui sono legati. Ogni volta che si necessita di accedere alle caratteristiche dell'hardware controllato, tali routine si uniscono al FileCore e realizzano il filing.

Prendiamo il caso dell'Advanced Disk Filing System. Questo si tratta del classico modulo archimediano per il controllo del read/write su disco. L'ADFS provvede un comando di selezione ed uno per il formato dei dischi, una piccola serie (quattro) di SWI per accedere alle corrispondenti chiamate del FileCore ed infine entry-point e routine specifiche affinché il FileCore possa accedere, nel caso specifico, a controller del disco e al relativo hardware.

Quello che è quindi contenuto sotto la pur importante denominazione di ADFS non è molto in realtà. Lo diventa

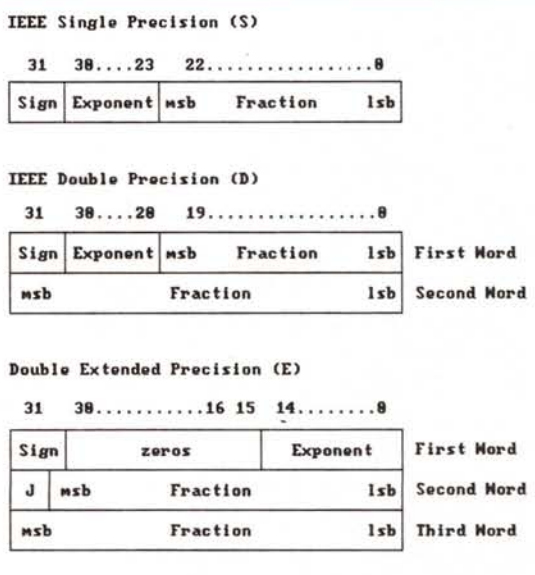

Figura 6

eventualmente, una volta invocato il FileCore, con la richiesta del tipo di formato-dischi prescelto — vedi figura 8 — unitamente ad altri setting, come ad esempio l'informazione relativa all'identificazione dei dischi, presenti negli standard del FileCore.

Insomma tanti Filing System di sistema, tantissimi modulabili esternamente a cura del programmer, ma in definitiva un solo sistema. Estremamente elastico e potente a cui, di volta in volta è possibile agganciarsi formando una lunga catena: dall'hardware al FileSwitch.

In riferimento alla creazione di un proprio modulo di Filing System spero che la schematizzazione riportata in figura 9 sia sufficientemente chiara.

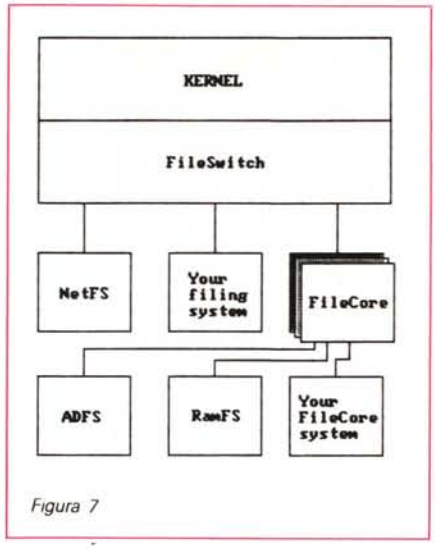

Figura 7

Window Manager

Ovvero il Desktop. Una fra le parti principali del RISC-OS che nasce sulla scorta dell'evoluzione del concetto di WIMP (Window Icons Menus e Pointer) comprendendo fra le sue maggiori caratteristiche sia la facilità all'uso (richiamo delle directory, scelta di file, inizializzazione e copie di dischi, etc.) che la potenza applicativa (il sistema multitask).

Quella che ne faremo è la più classica delle *overview* per correre poi al suo rapporto d'interazione con la VDU.

Dice il PRM che la caratteristica immediatamente riconoscibile di un programma WIMP è nella finestra grafica eventualmente posta sullo schermo. Ed è in tale finestra la dimostrazione stessa di tutte (o quasi) le potenzialità offerte dal Window Manager.

L'altra ovvia caratteristica del WIMP sono i menu che, nel caso del sistema grafico sotto RISC-OS, risultano di tipo "pop-up". Ciò significa che tali menu appariranno in video solo all'effettivo bisogno dell'utilizzatore che li richiamerà in qualsiasi punto dalla pagina grafica premendo il "menu-button" del mouse. Rispetto alla tradizionale gestione dei menu fatta dai primi WIMP-computer che delimitava una zona dello schermo — solitamente la parte alta di questo — ed in pratica per limiti fisici, imponeva un limitato numero di "tendine" apribili, sotto RISC-OS, con l'adottata tecnica del pop-up, oltre alla praticità di aprire sui menu nel punto dove si trova attualmente il puntatore del mouse c'è la massima libertà d'inserimento. In linea teorica non esiste un limite né numerico, né fisico ai menu richiamabili.

La caratteristica più affascinante dei moderni WIMP-computer come Archie, è senz'altro la possibilità offerta dell'operare in multitask. Più programmi contemporaneamente in screen tutti attivi e pronti all'esecuzione.

In definitiva però, come è reso possibile, "lavorare" sugli applicativi tradotti graficamente in video? Cosa accade e perché si determinano certe azioni a secondo dello spostamento del pointer del mouse e ad ogni "click" sui bottoni del mouse?

Indubbiamente, pointer, menu, icone e più in generale l'intera pagina grafica, sono continuamente tenuti sotto controllo dal WIMP, pronto in tal senso a produrre "event" ad ogni richiesta dell'utilizzatore. Questi "event" sono veri e propri messaggi che il Window Manager spedisce al programma a secondo del tipo di azione prodotta dall'utente. Il premere uno dei tre tasti del mouse innesca un procedimento prima di ricerca e quindi d'interpretazione ed esecuzione a comando dell'ordine ricevuto via mouse direttamente dall'utente al WIMP.

I programmi (o task) sotto detti "event driven"; ovvero sono guidati dai messaggi spediti dal WIMP e solo a questo rispondono. In pratica l'user "parla" con il WIMP e si ritrova nella condizione di svolgere l'azione desiderata sull'applicativo (ed impartita come ordine) solo perché, quest'ultimo, ha ricevuto l'event dal WIMP. La tecnica di tale controllo: dall'utente al WIMP, dal WIMP al programma è quella conosciuta come *Polling* (dall'inglese: scrutinare). L'applicativo sarà "interfacciato" col WIMP tramite la routine Wimp_Poll chiamata continuamente al controllo di eventuali event accaduti. Ad ogni evento (la richiesta dell'utente ad eseguire qualsiasi tipo di operazioni lecita) verrà trasmesso un codice al programma.

Ciò comporta ovviamente che il programma venga confezionato in modo consono all'ambiente. Con una struttura che, inizializzandosi al WIMP, sia comprensiva di routine di controllo e chiamate a Wimp_Poll.

FILECORE: formato dei dischi (Parametri standard)

Format	Tracks	Density	Sectors Track	Bytes Sector	Storage
L	80	Double	16	256	640Kbytes
D	80	Double	5	1024	800Kbytes
E	80	Double	5	1024	800Kbytes

Figura 8

VDU-drivers: livelli d'interattività

Visto come i VDU-drivers pilotano genericamente a video tutte le varie informazioni inviate — dal Character Output — e come è possibile "giocare" sui vari * command disponibili, il discorso sull'interazione, iniziato con lo Sprite Manager e il modulo estensivo del Font Manager, si completa finalmente con l'introduzione del concetto di ColourTrans e Draw (ultimi moduli di estensione ancora da trattare) ed il livello interattivo VDU-Window Manager.

Partendo col presentare il ColourTrans, possiamo considerare tale modulo come una sorta di controller disposto alla selezione dei colori fisici (quali componenti RGB) richiesti all'uso. (Per la cronaca tale modulo è nella maggior parte dei casi disponibile solo dopo caricamento in RAM prelevandolo dal disco di sistema nella directory *System: Modules. Colours*. Così è sul mio RISC-OS 2.0, anche se l'Acorn si è riservata la possibilità di inserirlo direttamente in ROM nelle versioni successive).

Alla base del modulo ColourTrans stanno i due comandi per la selezione del colore chiamati GCOL e colour number, due forme di selezione-colore che lavorano in 256-mode. In molti fra i programmatori archimediani ne avranno probabilmente già fatto uso anche nei loro lavori in BBC-Basic.

Altro parametro necessario per rendere il controllo totale della *translation* dei colori è quello della *Palette Entry*; una word che contiene una descrizione completa dei colori fisici nei rispettivi livelli RGB.

Altro parametro è quello del *Palette Pointer*, un puntatore ad un lista di entrate disponibili in numero pari a quello dei colori logici possibili nel modo selezionato. Nel caso del modo a 256 colori, ovviamente, ci saranno 16 entrate (quindi 16 registri).

In tutto ciò sta il meccanismo che s'innesca

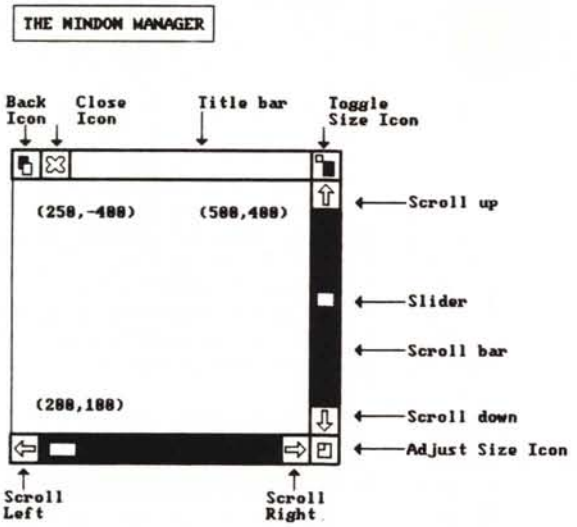

Figura 9

Window Manager

Wimp_Poll:

```
SINTASSI:       Wimp_Poll

IN:             R0= mask
                R1= pointer to 256 byte block
                    (usato per avere il ritorno del dato)

OUT:            R0= reason code
                R1= pointer to block
                    (dato dipendente dal codice ritornato)

USO:            Tale chiamata serve per controllare il
                verificarsi di determinati eventi quali
                selezioni via mouse, tastiera e menu.
```

CODICI di RITORNO a CHECK avvenuto

Code	Reason
0	Null_Reason_Code
1	Redraw_Window_Request
2	Open_Window_Request
3	Close_Window_Request
4	Pointer_Leaving_Window
5	Pointer_Entering_Window
6	Mouse_Click
7	User_Drag_Box
8	Key_Pressed
9	Menu_Selection
10	Scroll_Request
11	Lose_Caret
12	Gain_Caret
13-16	Reserved
17	User_Message
18	User_Message_Recorded
19	User_Message_Acknowledge

Figura 10

con la traduzione del colore logico nella composizione più prossima dei valori relativi alle componenti RGB.

Il massimo uso delle caratteristiche del ColourTrans si ha nel controllo dei colori relativi agli Sprite e ai caratteri del Font Manager, con particolare valenza (vedi qualità finale) nell'output su stampante, dove i colori logici vengono sostituiti dai più «veri» componenti RGB.

Spostandoci verso il modulo Draw, andiamo a prendere in considerazione tutto quello che precede l'eventuale aggiunta di colore: la rappresentazione di forme grafiche in screen. Il manuale afferma che il Draw può essere inteso come un modulo implementabile le tipologie grafiche del PostScript. Una collezione di movimenti, linee e curve che, raggruppate insieme, possono infine formare un oggetto unico e manipolabile nella sua interezza, denominato path.

È in pratica la caratteristica principale dell'applicativo presente sulla *Suite* di sistema, chiamato *Draw*.

Curve di Bezier, matrici di trasformazione, punti di giunzione e caratteristica di riempimento e perimetrazione che abbiamo conosciuto usando l'applicativo Draw, altro non sono che "primitive grafiche" già presenti nel modulo di estensione.

La valenza del *postscriptante* modulo oltre che in video è possibile quantificarla in massima misura anche in fase di stampa (nel senso di printer). La tecnica è potenzialmente tutta nel procedimento di creazione del *path* che una votla pronto verrà direttamente inviato al VDU per essere rappresentato.

Direte voi: ColourTrans e Draw sono OK, ma dov'è il livello d'interazione?

In pratica è possibile considerare i due moduli di estensione come integrativi e/o sostitutivi delle più complesse e ripetitive funzioni presenti nei vari comandi VDU.

Tendendo all'eliminazione dei passaggi intermedi per fornire direttamente al VDU le informazioni da stampare in video o direttamente su carta, nella miglior forma possibile.

Quantomeno è in simile considerazione che vengono usati dagli applicativi di più recente commercializzazione.

Tali applicativi, rifacendosi ai dettami del RISC-OS, come all'imposizione dello standard grafico, usano in modo assai più moderno le feature messe a disposizione dallo stesso sistema operativo. Nel caso specifico, la VDU si tende sempre più a renderla operativa con una serie d'informazioni già definite.

E lo stesso discorso, o se vogliamo: l'impostazione (giusta ed inevitabile) dell'uso interattivo di moduli estensivi in relazione alle peculiarità della VDU, può essere valido anche osservando le caratteristiche del Window Manager nella sua forma WIMP estesa a tutti i livelli.

Partendo comunque dall'abilità dei VDU-driver a "disegnare" nello schermo due differenti tipi di oggetti: testo e grafica (ovvero: i caratteri e ogni primitiva grafica) è possibile introdurre il concetto di WIMP-interattivo solo nel momento in cui, testo e grafica, vengono uniti in un'unica gestione: con il testo stampato in riferimento al cursore grafico e in una screen window.

Tutto ciò s'innesta nel discorso WIMP attraverso il comando VDU 5 ed a partire da questa semplice operazione le moderne implementazioni come il RISC-OS, permettono l'introduzione e l'uso esteso della finestra grafica.

Dalla "primitiva" finestrellazione operabile tutt'ora dal VDU 5, l'ambiente congiunto testo/grafica trova nel modello del desktop archimediano una soluzione più potente dove tutto è tanto semplificato quanto potenziato: multiwindow, multitask, scrolling e ridimensionamento e ordinamento delle pagine di lavoro.