Se il software dei lettori del numero scorso non è per nulla bastato a calmare i vostri bollori smanetterecci che da bravi utenti Amiga certamente non vi mancheranno, questo mese vi «massacreremo» con una seconda lezione, sempre opera dello stesso autore, sul tema dello «scrolling & affini». Nella lettera che accompagnava l'articolo in questione, il prode Maurizio Mangrella ci consolava dicendo: «... quando, seguendo il filone della ricerca, ci si avvicina al mondo nascosto dei chip, scoprire nuove cose diviene sempre più difficile: nemmeno la grazia (!) di un Guru Meditation, ma solo blocchi inesorabili o allucinanti visioni che è meglio non descrivere...».
Come lo capisco!

Scrolling hardware, Copper et similia

di Maurizio Mangrella - Eboli (SA)

Domenica 12 giugno, ore 15: mentre il sole mi rosicchia lentamente il cervello, mi appresto ad uno dei più temibili viaggi nella memoria di un computer: vago nei meandri della grafica dell'Amiga. «Di nuovo?», si chiederà qualcuno. Sì, di nuovo, perché so che non ho visto tutto. Almeno non tutto quello che si poteva vedere.

Al caldo si aggiungono le cifre e le Guru Meditation, ed ho l'impressione di uscire fuori di senno, se non fosse per qualche risultato concreto che rinfranca lo spirito...

È inutile negarlo, Paolo Russo, l'autore di «B... come Blitter», mi ha lasciato perplesso: è possibile che in Basic non sia possibile «desiderare» i modi grafici speciali o lo scrolling hardware dello schermo? Al primo dilemma ho già dato risposta, ma il secondo continuava a non farmi dormire la notte.

Però qualcosa l'ho fatta, e ve la presento: anche da Basic, signori miei, è possibile scrollare — quant'è brutto questo! — la bitmap ed il video a livello hardware. La cosa è un tantino complessa, perciò vi invito a seguirmi attentamente.

Come già sappiamo, lo Screen Record — quella struttura dati che definisce le caratteristiche di uno schermo — contiene altre due strutture, la ViewPort e la RastPort: la prima per i modi di visualizzazione, la seconda per l'allocazione dei bitplane in memoria ed altre cosette.

Qualche altra routine

In aggiunta a quelle che vedremo dopo, riportiamo nella tabella sottostante la descrizione di alcune utili routine della intuition.library.

Per «puntare» qui si intende puntare al relativo record descrittore: così screen è il puntatore allo Screen Record e window quello al Window Record. Una routine che non chiede parametri in input non può essere chiamata da Basic come si farebbe da C (senza scrivere nulla tra le parentesi), ma bisogna fornire loro un «dummy argument», perfettamente inutile: ad esempio

```
vpos& = VBeamPos& (0)
```

La ViewPort

La ViewPort è più importante di quanto si possa pensare — e di quanto io abbia pensato —: il suo schema è in figura 1. Prima di parlarne, ricordo che l'indirizzo della ViewPort contenente la Current Output Window è dato da

```
VPort& = 44+PEEKL (WINDOW 7 + 46)
```

o dalla routine ViewPortAddress della intuition.library, nella forma

Alcune routine della intuition.library

ScreenToBack (screen)
Manda lo schermo puntato da screen dietro tutti gli altri
ScreenToFront (screen)
Porta lo schermo puntato da screen davanti a tutti gli altri
WindowToBack (window)
Come ScreenToBack, ma per la finestra puntata da window
WindowToFront (window)
Come ScreenToFront, ma per la finestra puntata da window
MoveScreen (screen, dx, dy)
Muove lo schermo puntato da screen di dx pixel in orizzontale (non illudetevi! non ha effetto) e di dy in verticale
MoveWindow (window, dx, dy)
Come MoveScreen, ma per la finestra puntata da window
DisplayBeep (screen)
Attua un «beep» visivo (inverte per un attimo il colore 0) sullo schermo puntato da screen
CloseWorkBench ()
Chiude tutte le finestre del WorkBench, conservandone in memoria solo i dati; se non ci sono altre finestre aperte chiude anche lo schermo
OpenWorkBench ()
Riapre il WorkBench
Aggiungo anche tre routine della graphics.library:
GetRGB4 (colormap, i)
Ritorna una word nel formato XXXXRRRRGGGGBBBB rappresentante le componenti cromatiche dell'i-esimo colore nella ColorMap puntata da colormap. Va usata nella forma
DECLARE FUNCTION GetRGB4% LIBRARY
LIBRARY "graphics.library"

.
.
.

c% = GetRGB4% (cm&,i&)
pena un "Overflow Error"
ScrollRaster (rastport, dx, dy, x1, y1, x2, y2)
Scrolla la BitMap della rastport puntata da rastport di dx pixels in orizzontale e dy in verticale, nell'ambito del rettangolo (x1, y1)-(x2, y2).
VBeamPos ()
Ritorna la posizione verticale del raster nel momento in cui viene chiamata.

```
struct ViewPort
{
    struct    ViewPort *Next;
    struct    ColorMap *ColorMap;
    struct    CopList *DspIns;
    struct    CopList *SprIns;
    struct    CopList *ClrIns;
    struct    UCopList *UCopIns;
    SHORT     DWidth,DHeight;
    SHORT     DxOffset,DyOffset;
    UWORD     Modes;
    UBYTE     SpritePriorities;
    UBYTE     reserved;
    struct    RasInfo *RasInfo;
}
```

Figura 1 - La ViewPort Structure.

VPort& = ViewPortAddress& (WINDOW 7)

Delle CopLists parleremo dopo, per il momento accontentiamoci — si fa per dire — del resto.

DWidth e DHeight sono, rispettivamente, larghezza ed altezza dello schermo: fin qui nulla di nuovo. La novità sta in DxOffset e DyOffset: queste due word definiscono la posizione del relativo schermo nell'ambito della videata. I loro indirizzi sono, rispettivamente, vport&+28 e vport&+30, dove port& è l'indirizzo della ViewPort. Già vi vedo tutti intenti a pokare in queste locazioni nella speranza di vedere l'agognato effetto; peccato, però, non succede niente... Aspettate il prossimo paragrafo e vedrete...

Poi c'è un byte dedicato alla priorità degli sprite: 8 bit per 8 sprite, dunque un bit per ogni sprite — sia lo schema che il funzionamento sono simili a quelli del 64 —; se un bit è settato, il relativo sprite sarà completamente sovrapposto alla BitMap, altrimenti sarà sovrapposto solo al colore 0 (e passerà sotto tutti gli altri). Per chi non lo sapesse, gli sprite hanno due bitplane e si condividono i registri colore nel seguente modo: ▼

Colore mappato	Sprites: 0 - 1	2 - 3	4 - 5	6 - 7
0		trasparente		
1	registri 17	21	25	29
2	registri 18	22	26	30
3	registri 19	23	27	31

Per qualche oscuro motivo, allo sprite 0 — che è il puntatore del mouse, riservato al sistema operativo — corrisponde il bit 5...

Segue un byte riservato — che è meglio non modificare — ed il puntatore alla struct RasInfo, che «informa» la ViewPort sull'allocazione della BitMap in memoria e sul video. La conformazione di RasInfo è in figura 2.

RasInfo punta al prossimo RasInfo — questo schema «a rilancio» è tipico dell'Amiga — e l'ultimo non punta a un bel niente... Segue il puntatore alla BitMap — identico a quello contenuto nella RastPort — e i valori RxOffset e RyOffset, che servono a scrollare la BitMap all'interno dello schermo, lasciando inalterata la posizione di quest'ultimo rispetto al video. In pratica, questi due valori non sono altro che le coordinate del punto della BitMap che apparirà nell'angolo superiore sinistro dello schermo. Chiaro il giochetto?

```
struct RasInfo
{
    struct    RasInfo *Next;
    struct    BitMap *BitMap;
    SHORT     RxOffset,RyOffset;
}
```

Figura 2 - La RasInfo Structure.

```
struct View
{
    struct    ViewPort *ViewPort;
    struct    cprlist *LOFCprList;
    struct    cprlist *SHFCprList;
    SHORT     DyOffset,DxOffset;
    UWORD     Modes;
}
```

Figura 3 - La View Structure.

View

Per chi mastica un po' di C amighese — è un dialetto particolare — il titolo sibillino di questo paragrafo dovrebbe dire tanto. Chiariamo per i nuovi arrivati.

Ciò che noi vediamo sul monitor del nostro Amiga è una View, la quale è divisa in «fette» orizzontali che si chiamano ViewPorts — ma guarda un po'...

È chiaro che il computer deve sapere come è organizzata la View: a questo scopo provvede una struttura dati, organizzata come recita la figura 3.

Il primo puntatore punta — lo dice il nome stesso... — alla prima ViewPort, che punta alla seconda, che punta alla terza... e così via. Poi vengono due voci di cui poche persone al mondo — a mio parere — conoscono il significato, dunque posizione e modo grafico del primo schermo.

Per cambiare i parametri di schermo non basta pokare nella ViewPort, ma bisogna anche ricalcolare la schermata; dunque, dopo aver settato i valori desiderati, dovremo dare (da Basic):

```
CALL MakeVPort (view&,vport&)
CALL MrgCop (view&)
CALL LoadView (view&)
```

Mamma li Turchi! Calma, calma, sono tutte routine della graphics.library: la prima ricostruisce la schermata puntata da view& — indirizzo di struct View — aggiungendo, nel caso ve ne fosse bisogno, la ViewPort puntata da vport&; la seconda (Merge CopLists) fonde le CopLists degli schermi di cui si compone la View e la terza spedisce il tutto a Denise.

L'indirizzo di View

Ma dove sta View? Per capirlo ci ho messo parecchio, ma ora lo so — suspense cattiva atta a destare l'invidia dei lettori... Scherzo...

La View attualmente visualizzata sullo schermo — da ora in poi diremo «attiva» — è puntata da una locazione della parte bassa della RAM, a testimonianza che questi dati il S.O. li ha a sua disposizione — e se li tiene per sé.

Per sapere qual è il puntatore a View, dobbiamo aprire la graphics.library tra-

mite Exec — e non attraverso l'interprete Basic — dando:

```
DECLARE FUNCTION OpenLibrary& LIBRARY
LIBRARY "exec.library"

   .
   .
   .

GfxBase& = OpenLibrary& (SADD("graphics.library"+CHR$(0)),0)
```

In C daremmo

```
#include <graphics/gfxbase.h>

   .
   .
   .

GfxBase = (struct GfxBase *)OpenLibrary ("graphics.library",0)
```

Gf xBase& — lo ricordate? In Assembler bisogna porlo nel registro A6, pena una colorita Guru Meditation — punta ad una complessa struttura dati (figura 4) che contiene informazioni necessarie per il regolare svolgimento delle operazioni grafiche: per farla breve, l'indirizzo della View attiva è:

View& = PEEKL (Gf xBase&+34)

Lo 0, secondo parametro della OpenLibrary, indica al S.O. che non abbiamo particolari predilezioni circa la versione della relativa libreria: per noi vanno tutte bene. Per garantire la massima compatibilità, è bene specificare sempre 0.

Come al solito, esiste un metodo più semplice, basta dare:

View& = ViewAddress& (0)

```
struct GfxBase
{
    struct   Library LibNode;
    struct   View *ActiView;
    struct   copinit *copinit;
    LONG     *cia;
    LONG     *blitter;
    UWORD    *LOFlist;
    UWORD    *SHFlist;
    struct   bltnode *blthd,*bltt1;
    struct   bltnode *bsblthd,*bsbltt1;
    struct   Interrupt vbsrv,timsrv,bltsrv;
    struct   List TextFonts;
    struct   TextFont *DefaultFont;
    UWORD    Modes;
    BYTE     VBlank;
    BYTE     Debug;
    SHORT    BeamSync;
    SHORT    system_bplcon0;
    UBYTE    SpriteReserved;
    UBYTE    bytereserved;
    USHORT   Flags;
    SHORT    BlitLock;
    SHORT    BlitNest;
    struct   List BlitWaitQ;
    struct   Task *BlitOwner;
    struct   List TOF_WaitQ;
    UWORD    DisplayFlags;
    struct   SimpleSprite **SimpleSprites;
    UWORD    MaxDisplayRow;
    UWORD    MaxDisplayColumn;
    UWORD    NormalDisplayRows;
    UWORD    NormalDisplayColumns;
    UWORD    NormalDPMX;
    UWORD    NormalDPMY;
    struct   SignalSemaphore *LastChanceMemory;
    UWORD    *LCMptr;
    UWORD    MicrosPerLine;
    ULONG    reserved[2];
}
```

Figura 4 - La Gf xBase Structure.

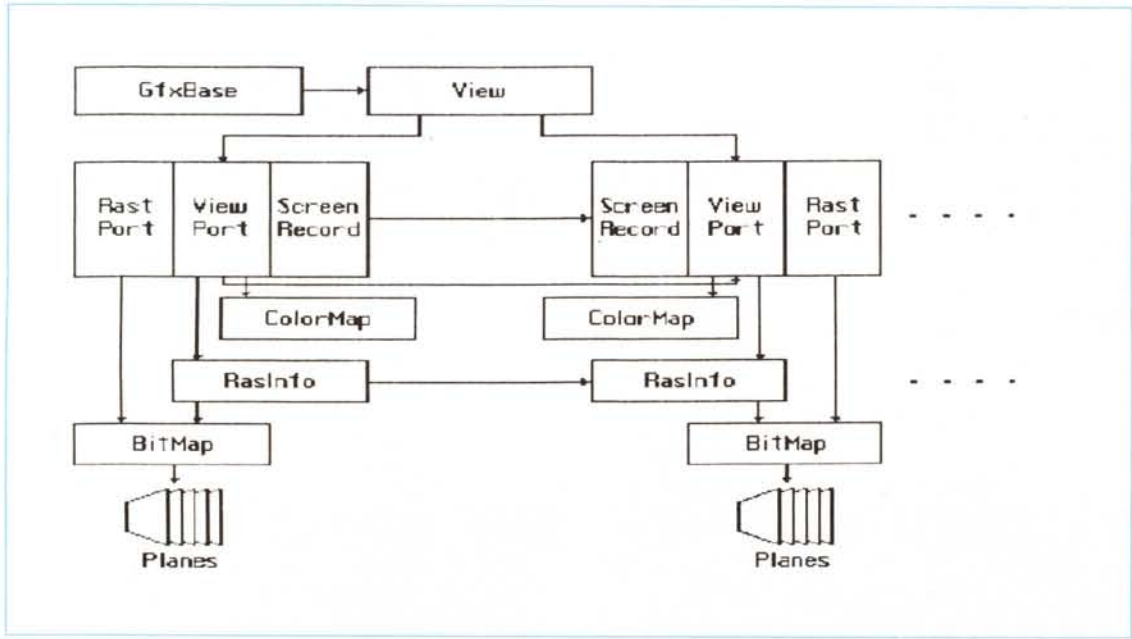

Figura 5 L'albero genealogico delle strutture grafiche.

(la ViewAddress () è una routine della intuition.library). In Basic usare l'uno o l'altro metodo è quasi indifferente, mentre da C, paradossalmente, conviene il primo, in quanto ci evita di dover aprire anche la intuition.library.

Il Copper

Bene bene, è venuta ora di parlare un po' del nostro beneamato Copper.

Il Copper — abbreviazione di coprocessor — è una parte del chip Agnus che si preoccupa di risistemare i puntatori alle zone grafiche utilizzati da Denise: in pratica, i registri del chip grafico vengono gestiti quasi esclusivamente dal Copper. Ad esempio, i registri DFF0E0 - DFF0F7, che contengono gli indirizzi dei bitplane, sono manipolabili solo dal Copper, ed un qualunque tentativo di scrittura attuato dal microprocessore non ha effetto.

Il Copper è, in realtà, una sorta di piccolo microprocessore RISC dedicato, che riesce ad eseguire solo tre istruzioni diverse — a quanto so — basandosi prevalentemente sulla posizione del pennello elettronico.

Ad ogni cambio di schermo — quando termina la parte visibile di una ViewPort e comincia quella successiva — il nostro coprocessore sistema tutti i puntatori: lavoro ingrato, questo, che gli costa circa due o tre righe vuote, ovvero due o tre Horizontal Blanks (tra 128 e 192 microsecondi, tempo in cui nulla, nemmeno gli sprite, è visualizzabile).

Le CopLists — e così chiariamo ogni dubbio — sono i programmi del Copper, intesi come elenchi di istruzioni. Una istruzione presenta tipicamente il seguente formato:

```
struct CopIns
{
  SHORT    OpCode;
  LONG     *NextList;
  SHORT    VWaitPos;
  SHORT    DestAddr;
  SHORT    HWaitPos;
  SHORT    DestData;
}
```

(ho cercato di semplificare un po': il codice originale presenta delle union).

Nulla so degli OpCodes, so solo che sono tre — i nomi dovrebbero essere MOVE, SEARCH e SKIP, come lessi tempo fa su una rivista, ma non so se e quanto ciò sia vero —; una cosa è degna di nota: il Copper è talmente veloce da riuscire ad intercettare anche il percorso orizzontale del pennello elettronico: questo è indubbiamente vantaggioso, se è vero che la SEARCH può generare un interrupt per il 68000 e

```
'  Viaggio all'interno dell'Amiga   c by Maurizio Mangrella  1988
DECLARE FUNCTION OpenLibrary& LIBRARY       ' Apriamo la graphics.library
LIBRARY "exec.library"                      ' tramite Exec
LIBRARY "graphics.library"
                                            ' Si parte !
GfxBase& = OpenLibrary&(SADD("graphics.library"+CHR$(0)),0)
View& = PEEKL(GfxBase&+34)                  ' Indirizzo di ActiView
VPort& = 44+PEEKL(WINDOW(7)+46)             ' La ViewPort del Wb Screen
RasInfo& = PEEKL(VPort&+36)                 ' Puntatore a RasInfo
RxOffset& = RasInfo&+8                      ' Indirizzo di RxOffset
RyOffset& = RasInfo&+10                     ' e di RyOffset
x = PEEKW(RxOffset&)                        ' Posizioni iniziali
y = PEEKW(RyOffset&)
WHILE 1                                     ' Comincia il viaggio (eterno!)
     a$ = ""
     WHILE a$ = "" : a$ = INKEY$ : WEND     ' Un tasto cursore, please ...
     a = ASC(a$)
     IF a = 28 THEN y = y+1                 ' Sale
     IF a = 29 THEN y = y-1                 ' Scende
     IF a = 30 THEN x = x-1                 ' A destra
     IF a = 31 THEN x = x+1                 ' E a sinistra
     POKEW RxOffset&,x                      ' Settiamo i nuovi valori ...
     POKEW RyOffset&,y
     CALL MakeVPort(View&,VPort&)           ' ... e li mettiamo in pratica
     CALL MrgCop(View&)
     CALL LoadView(View&)
WEND                                        ' Continua
```

Figura 6 - Con i tasti cursore viaggerete nella memoria dell'Amiga...

```
'  Viaggio sul monitor dell'Amiga   c by Maurizio Mangrella  1988
'  (In pratica e' lo stesso di prima ...)
DECLARE FUNCTION OpenLibrary& LIBRARY       ' Apriamo la graphics.library
LIBRARY "exec.library"                      ' tramite Exec
LIBRARY "graphics.library"
                                            ' Si parte !
GfxBase& = OpenLibrary&(SADD("graphics.library"+CHR$(0)),0)
View& = PEEKL(GfxBase&+34)                  ' Indirizzo di ActiView
VPort& = 44+PEEKL(WINDOW(7)+46)             ' La ViewPort del Wb Screen
DxOffset& = VPort&+28                       ' Indirizzo di DxOffset
DyOffset& = VPort&+30                       ' e di DyOffset
sx = PEEKW(DxOffset&)                       ' Posizioni iniziali
sy = PEEKW(DyOffset&)
x = sx : y = sy                             ' Si parte dall'inizio
dx = -2 : dy = -2                           ' Passo dello spostamento
WHILE 1                                     ' Comincia il rimbalzo
     x = x+dx
     y = y+dy                               ' Il solito algoritmo ...
     IF x < (sx-30) OR x >= (sx+30) THEN dx = -dx
     IF y < (sx-20) OR y >= (sx+20) THEN dy = -dy
     POKEW DxOffset&,x                      ' Settiamo i nuovi valori ...
     POKEW DyOffset&,y
     CALL MakeVPort(View&,VPort&)           ' ... e li mettiamo in pratica
     CALL MrgCop(View&)
     CALL LoadView(View&)
WEND                                        ' Continua
```

Figura 7 - Lo schermo che rimbalza sul monitor.

```
'  WorkBench 320x200 e sprite diafani !   c by Maurizio Mangrella  1988
'  (Ho inserito solo i commenti necessari)
DECLARE FUNCTION OpenLibrary& LIBRARY
LIBRARY "exec.library"
LIBRARY "graphics.library"
GfxBase& = OpenLibrary&(SADD("graphics.library"+CHR$(0)),0)
View& = PEEKL(GfxBase&+34)
VPort& = 44+PEEKL(WINDOW(7)+46)
POKEW VPort&+32,0                           ' ViewPort Mode (0 = LORES)
POKE VPort&+34,0                            ' SpritePriorities
CALL MakeVPort(View&,VPort&)
CALL MrgCop(View&)
CALL LoadView(View&)
CALL CloseLibrary(GfxBase&)                 ' Chiudiamo la graphics.library
LIBRARY CLOSE                               ' Chiudiamo le librerie
```

Figura 8 - Workbench 320x200 e sprite a priorità più bassa.

```
'  OverScan : una realta' anche da BASIC !       c Maurizio Mangrella  1988
DECLARE FUNCTION OpenLibrary& LIBRARY
LIBRARY "exec.library"
LIBRARY "graphics.library"
GfxBase& = OpenLibrary&(SADD("graphics.library"+CHR$(0)),0)
View& = PEEKL(GfxBase&+34)              ' Risparmiamo la intuition.library
SCREEN 1,352,240,5,1                    ' OverScan 352x240
WINDOW 2,"Prova OverScan",,0,1          ' Aprire una finestra solo per i pa -
WINDOW OUTPUT 2                         ' rametri di schermo, in fondo, e' pur
VPort& = 44+PEEKL(WINDOW(7)+46)         ' sempre il metodo piu' sbrigativo ...
POKEW VPort&+28,-16                     ' 16 pixels a sinistra
CALL MakeVPort(View&,VPort&)            ' Il solito Trio Lescano
CALL MrgCop(View&)
CALL LoadView(View&)
RANDOMIZE TIMER
FOR k = 1 TO 200                        ' Linee casuali
   c = FIX(32*RND)
   x1 = FIX(352*RND)
   x2 = FIX(352*RND)
   y1 = FIX(240*RND)
   y2 = FIX(240*RND)
   LINE (x1,y1)-(x2,y2),c
NEXT k
WHILE INKEY$ = "" : WEND                ' Premere un tasto ...
CLS
FOR k = 1 TO 100                        ' Rettangoli a caso
   c = FIX(32*RND)
   x1 = FIX(352*RND)
   x2 = FIX(352*RND)
   y1 = FIX(240*RND)
   y2 = FIX(240*RND)
   LINE (x1,y1)-(x2,y2),c,bf
NEXT k
WHILE INKEY$ = "" : WEND                ' Premere un tasto ...
WINDOW CLOSE 2                          ' Scusate se e' poco, ma non mi e'
SCREEN CLOSE 1                          ' venuto in mente niente di meglio ...
CALL CloseLibrary(GfxBase&)
LIBRARY CLOSE
```

Figura 9 - OverScan da Basic: 352x240 in 32 colori.

che, dunque, sarebbe possibile cambiare parametri di schermo molte volte in una sola riga.

Come da figura 1, una ViewPort contiene tre CopLists — non so cosa sia UCop — Ins —: DspIns (Display Instructions, programma per il display), SprIns (Sprite Instructions, programma per gli sprite) e ClrIns (???).

Quando si uniscono più ViewPorts in una View, le CopLists devono essere fuse insieme: di ciò si occupa MrgCop.

Ad ogni Vertical Blank il Copper invia un interrupt al 68000, che viene intercettato tramite la WaitTOF () (Wait for Top Of Frame).

IntuiMessage

Lo so, lo so, non c'entra (quasi) niente, ma, dato che l'ho scoperto proprio adesso, mentre sto scrivendo quest'articolo (meraviglie del multitasking), voglio inserirlo lo stesso: ecco come si fa ad ottenere un messaggio dal port di Intuition.

L'indirizzo del port legato alla Current Output Window è dato da: ▼

```
PEEKL (WINDOW(7)+86)

  Per ottenere un messaggio (da BASIC), daremo

DECLARE FUNCTION GetMsg& LIBRARY
LIBRARY "exec.library"

   .
   .
   .

Port& = PEEKL (WINDOW(7)+86)
Msg& = GetMsg& (Port&)
```

Fatte le debite semplificazioni, la struct IntuiMessage (il cui puntatore è, nell'esempio precedente, Msg&) è quella mostrata in figura 10.

C'è un Node (informazioni per i tecnici: niente nome, priorità 0, tipo 5, corrispondente a MESSAGE), il puntatore al ReplyPort (il port cui ripetere il messaggio), la lunghezza del messaggio e gli «items» che abbiamo visto l'altra volta: Class e Code, insieme ad un non meglio specificato Qualifier, poi (tante grazie!) ascissa e ordinata del mouse e secondi e microsecondi trascorsi dall'accensione... Dunque avremo:

```
struct IntuiMessage
{
    struct   Message ExecMessage;
    ULONG    Class;
    USHORT   Code;
    USHORT   Qualifier;
    APTR     IAddress;
    SHORT    MouseX, MouseY;
    ULONG    Seconds, Micros;
    struct   Window *IDCMPWindow;
    struct   IntuiMessage *SpecialLink;
};

struct Message
{
    struct   Node mn_Node;
    struct   MsgPort *mn_ReplyPort;
    UWORD    mn_Length;
};

struct Node
{
    struct   Node *ln_Succ;
    struct   Node *ln_Pred;
    UBYTE    ln_Type;
    BYTE     ln_Pri;
    char     *ln_Name;
}
```

Figura 10 - La IntuiMessage Structure.

```
Class& = PEEKL (Msg&+20)
Code% = PEEKW (Msg&+24)
Qualifier% = PEEKW (Msg&+26)
MouseX% = PEEKW (Msg&+32)
MouseY% = PEEKW (Msg&+34)
Seconds& = PEEKL (Msg&+36)
Micros& = PEEKL (Msg&+40)
```

Bel colpo!

Conclusioni

Qui termina il mio lavoro, come al solito aperto ad ampliamenti da parte di chi sia riuscito a scoprire qualcosa di più. Così fughiamo ogni dubbio anche per quanto riguarda il Basic: con questo linguaggio è possibile quasi tutto; non si può certo pretendere di realizzare un'animazione tipo Juggler o di riuscire ad ottenere effetti che solo il Linguaggio Macchina consente (e solo in casi particolari). Qualcosa di buono si può comunque fare...

A presto! MC

Il Computer e' servito!

postalbit

VENDITA PER CORRISPONDENZA
Terminal Centro e Sud d'Italia
Tel. 06/5424303

PERSONAL COMPUTER 386
Personal Computer MsDos, OS/2 e Xenix compatibile.
Microprocessore 80386 a 16 Mhz. Velocita' 20 Mhz.
Coprocessore matematico opzionale. Tastiera 101 tasti.
Memoria RAM 2 Mb on board. Alimentatore. Cabinet Tower con spie di funzionamento, tasti di selezione e chiave.
Controller per Floppy disk drive e Hard Disk. Scheda video.
Manuale operativo. Garanzia 12 mesi.
Configurazioni disponibili. Un Floppy Disk Drive 1,2 Mb.
Un Floppy disk drive 1,2 Mb e hard disk 20/40/70 Mb.

Configurazioni e prezzi

386/1	Un Fd 1,2Mb	lire	4.100.000
386/2	Un Fd 1,2Mb Hard Disk 20Mb	lire	4.450.000
386/3	Un Fd 1,2Mb Hard Disk 40Mb	lire	4.750.000
386/4	Un Fd 1,2Mb Hard Disk 70Mb	lire	5.560.000

Parti per Computer Kit

Cabinet XT/AT completo di alimentatore	lire	115.000
Case Tower completo di alimentatore	lire	520.000
Mother Board Turbo Xt OK	lire	130.000
Mother Board AT 13 Mhz OK	lire	480.000
Mother Board 386 20Mhz 2Mb	lire	3.350.000
Floppy Disk Controller Xt	lire	25.000
Hard Disk Controller Xt	lire	82.000
Hard/Floppy Disk Controller At	lire	195.000
Alimentatore 150W	lire	70.000
Alimentatore 180W	lire	115.000
Scheda video grafica colore con printer	lire	80.000
Scheda video monocromatica con printer	lire	80.000
Scheda video monocrom 16Mhz con printer	lire	100.000
Scheda video EGA	lire	230.000
Scheda video PGA	lire	320.000
Scheda video VGA	lire	385.000
Scheda seriale	lire	35.000
Chip abilitazione seconda porta seriale	lire	25.000
Scheda seriale,giochi,mouse,clock,printer	lire	75.000
Scheda joystick	lire	30.000
Scheda Clock real time	lire	55.000
Scheda modem 300/1200 baud	lire	150.000

PERSONAL COMPUTER 8088
Personal Computer Ms Dos. Microprocessore Intel 8088 Turbo a 4,7/8 Mhz. Coprocessore matematico opzionale.Tastiera 101 tasti. Memoria Ram 256K espandibile a 1024 K su scheda madre. Alimentatore 150W. Cabinet flip top metallico.
Pannello di controllo esterno con tasto reset e tasto turbo.
Chiave di sicurezza. Spie di funzionamento. Controller per due floppy disk drive. Scheda video.
Manuale operativo . Garanzia 12 mesi
Opzioni : Coprocessore matematico. Espansioni di memoria.
Scheda video EGA o PGA. Monitor vari. Stampanti.

Configurazioni e prezzi

8088/1	Un FDD 360K	lire	630.000
8088/2	Due FDD 360K	lire	740.000
8088/3	Un FDD 360K e Hard Disk 20Mb	lire	1.070.000
8088/4	Un FDD 720K	lire	670.000
8088/5	Due FDD 720K	lire	820.000
8088/6	Un FDD 720K e Hard Disk 20Mb	lire	1.120.000
8088/7	Un FDD 360K e un FDD 720K	lire	780.000

PERSONAL COMPUTER 80286
Personal Computer MsDos/OS2/Xenix compatibile.
Microprocessore Intel 80286 a 6/8/10 Mhz. Velocita' 13 Mhz.
Coprocessore matematico opzionale. Tastiera 101 tasti.
Memoria RAM 512K espandibile 1 Mb su scheda madre. Cabinet Flip Top metallico. Pannello di controllo esterno con tasto di reset e turbo. Spie di funzionamento. Chiave di sicurezza
Controller per Floppy Disk Drive e Hard Disk. Scheda video
Manuale operativo. Garanzia 12 mesi.
Configurazioni disponibili: Un Floppy Disk Drive 1,2 MB.
1 Floppy Disk Drive da 1,2 Mb e Hard disk da 20/40/70 Mb.
Opzioni: Coprocessore matematico. Espansioni di memoria.
Schede video EGA e PGA. Monitor vari. Stampanti.

Configurazioni e prezzi

286/1	Un FDD 1,2 Mb	lire	1.350.000
286/2	Un FDD 1,2 Mb HD 20 Mb	lire	1.710.000
286/3	Un FDD 1,2 Mb HD 40 Mb	lire	2.010.000
286/4	Un FDD 1,2 Mb HD 70 Mb	lire	2.830.000

Personal Computer Travel
Personal Computer IBM Compatibile formato valigetta.
Microprocessori 8088 o 80286. Schermo LCD con scheda video grafico colore o alta risoluzione grafica EGA. Floppy disk drive da 360K o 720K e 1,2Mb. Dischi rigidi da 20 e 40 Mb.
Tastiera. Controllo di contrasto e tasto reset.
Uscita per monitor e tastiera esterni e stampante.

Configurazioni e prezzi

TRV1	XT Turbo 256K 2 FDD 360K	lire	1.620.000
TRV2	XT Turbo 256K 1 FDD 360K HD20Mb	lire	1.830.000
TRV3	XT Turbo 256K 2 FDD 720K	lire	1.620.000
TRV4	XT Turbo 256K 1 FDD 720K HD20Mb	lire	1.830.000
TRV5	AT 13Mhz 512K 1 FDD 1,2M HD20Mb	lire	2.440.000
TRV6	AT 13Mhz 512K 1 FDD 1,2M HD40Mb	lire	2.730.000

Memorie di massa

Floppy disk drive 360K	lire	110.000
Floppy disk drive 720K completo di adattatore	lire	158.000
Floppy disk drive 1,2Mb	lire	175.000
Hard Disk Drive 20Mb 3"1/2	lire	360.000
Hard Disk Drive 40Mb 40Ms	lire	680.000
Hard Disk Drive 70Mb 28Ms	lire	1.550.000
Back Up 20Mb con cassetta	lire	900.000
Back Up 60Mb con cassetta	lire	1.050.000
Cassetta per Back Up 20Mb	lire	40.000
Cassetta per Back Up 60Mb	lire	55.000

Add In / Add On

Deviatore 2 Personal 1 Stampante	lire	65.000
Mouse Meccanico Microsoft Comp.le	lire	65.000
Mouse Logitech Compatibile	lire	90.000
Joystick per PC	lire	35.000
Gruppo di continuita' 250W	lire	350.000
Gruppo di continuita' 500W	lire	600.000
Coprocessore matematico 8087/5	lire	220.000
Coprocessore matematico 80287/6	lire	360.000
Coprocessore matematico 80287/8	lire	530.000
Coprocessore matematico 80287/10	lire	820.000
Cavo Centronics per stampante	lire	10.000
Cavo Centronics per stampante 5 mt	lire	40.000
Cavo prolunga tastiera	lire	25.000
Cavo prolunga monitor	lire	25.000
Cavo collegamento PC TV Scart	lire	130.000

Personal Computer Philips
PP0
Personal Computer IBM Compatibile
Processore Intel 8088 a 4,7/8 Mhz
RAM 768K On board. Scheda video Monocromatica e Grafica Colori. Scheda Seriale e Parallela.
Real Time clock. Controller HD su mother board
Tastiera. Manuale d'uso. MS DOS e GW BASIC.

PP1 Con 2 FDD 720K lire 1.165.000
PP2 Con 1 FDD 720K e HD 20 Mb lire 1.750.000

Monitor

M1	Goldstar Monocromatico Fosfori verdi	lire	140.000
M2	Goldstar Colori Media Risoluzione	lire	470.000
M3	Goldstar Colori Alta Risoluzione EGA	lire	670.000
M4	Data Engineering TTL/CGA Bianco Carta	lire	190.000
M5	Data Engineering Colori Media Ris.ne	lire	460.000
M6	Data Engineering Colori E.G.A.	lire	680.000
M7	Data Engineering Multisync Colori	lire	950.000
M8	Philips Basculante FOsfori Verdi/Ambra	lire	165.000
M9	Philips Colori E.G.A.	lire	630.000
M10	Samsung Basculante Monocromatico	lire	140.000

Stampanti OKI Microline 9/18 aghi

ML182P	Stampante ad aghi 80 colonne 120 cps bidirezionale grafica NLQ int.parall	lire	470.000
ML182S	Come sopra con interfaccia seriale	lire	540.000
ML192P	Stampante ad aghi 80 colonne 200 cps bidirezionale grafica NLQ int.parell con inseritore frontale fogli singoli	lire	660.000
ML192S	Come sopra con interfaccia seriale	lire	790.000
ML193P	Stampante ad aghi 136 colonne 200 cps bidirezionale grafica NLQ int.parall con inseritore frontale fogli singoli	lire	810.000
ML193S	Come sopra con interfaccia seriale	lire	940.000
ASM001	Inseritore automatico fogli singoli 80 colonne per 192/193	lire	430.000
ASM002	Inseritore automatico fogli singoli 136 colonne per 193	lire	530.000
ASM003	Kit trattore/schermo acustico per modelli 182/192	lire	125.000
ASM004	Cartuccia nastro nero 182/192/193	lire	25.000
ML292	Stampante 18 aghi 80 col. 240 cps bidirezionale grafica NLQ stampa in nero o a colori interfaccia parallela o seriale	lire	820.000
ML293	Come sopra 136 colonne	lire	1.100.000
ML294	Come sopra 400 cps	lire	1.400.000
ASM005	Inseritore automatico fogli singoli 80 colonne per 292/293	lire	450.000
ASM006	Inseritore automatico fogli singoli 136 colonne per 293	lire	550.000
ASM007	Kit trattore/schermo acustico 292	lire	125.000
ASM008	Cartuccia nastro nero 292	lire	30.000
ASM009	Cartuccia nastro colore 292	lire	35.000
ASM010	Cartuccia nastro nero 293/294	lire	40.000
ASM011	Cartuccia nastro colore 293/294	lire	45.000

Stampanti Oki Microline 24 Aghi

ML393	Stampante 24 aghi 136 col 360 cps bidirezionale grafica NLQ interfaccia parallela e seriale	lire	1.750.000
ML393C	Come sopra ma a colori	lire	2.000.000
ML390P	Stampante 24 aghi 80 col 270 cps bidirezionale grafica NLQ int.par.	lire	940.000
ML390S	Come sopra ma interfaccia seriale	lire	1.060.000
ML391P	Come 390P ma a 136 colonne	lire	1.220.000
ML391S	Come sopra con interfaccia seriale	lire	1.350.000
ASM012	Inseritore singolo automatico 393	lire	620.000
ASM013	Inseritore doppio automatico 393	lire	1.000.000
ASM014	Inseritore singolo automatico 391	lire	620.000
ASM015	Inseritore doppio automatico 391	lire	1.000.000
ASM016	Inseritore singolo automatico 80col per 390/391	lire	620.000
ASM017	Come sopra doppio	lire	1.000.000
ASM018	Cartuccia Font 393/390/391	lire	120.000
ASM019	Kit trattore opzionale 393	lire	190.000
ASM020	Kit trattore opzionale 390	lire	120.000
ASM021	Kit trattore opzionale 391	lire	140.000
ASM022	Cartuccia nastro nero 393	lire	55.000
ASM023	Cartuccia nastro color 393	lire	65.000
ASM024	Cartuccia nastro nero 390/391	lire	22.000

Stampanti OKI Microline Laser

LS61	Stampante laser 6 pagine minuto RAM 512K Stampa alta qualita' Risoluzione 300x300 Interfaccia parallela o seriale Emulazioni HPLjet II, Epson, IBM, Diablo Qume, NEC	lire	3.600.000
LS62	Come sopra con RAM 1,5 Mb	lire	3.980.000
LS63	Come sopra con RAM 2,5 Mb	lire	4.500.000
LS121	Come LS61 12 pagine minuto	lire	5.100.000
LS121	Come sopra con RAM 2Mb	lire	6.000.000
ASM25	Memoria RAM 1,5 Mb per LS6	lire	750.000
ASM26	Memoria RAM 2,5 Mb per LS6	lire	1.500.000
ASM27	Vassoio alimentazione carta 500 fogli per LS6	lire	1.425.000
ASM28	Cartuccia font per LS6	lire	290.000
ASM29	Cartuccia font per LS12	lire	450.000
ASM30	Cartuccia generazione barcode LS12	lire	1.000.000
ASM31	Cartuccia emulazione plotter LS12	lire	1.000.000
ASM32	Espansione memoria RAM 1,5 Mb LS12	lire	1.500.000
ASM33	Gruppo pulizia con tamburo e filtro ottico per LS6	lire	350.000
ASM34	Come sopra senza tamburo e filtro	lire	180.000
ASM35	Toner per LS6	lire	45.000
ASM36	Gruppo pulizia, stampa, olio fusione per LS12	lire	650.000
ASM37	Gruppo fusione per LS12	lire	350.000
ASM38	Tonar per LS12	lire	55.000

Diskettes e Moduli Continui

DM1	10 Diskettes Postalbit 5"1/4 DSDD	lire	9.000
DM2	10 Diskettes Postalbit 3"1/2 DSDD	lire	20.000
DM3	10 Diskettes Postalbit 5"1/5 HD	lire	40.000
DM90	1000 Fogli 80 colonne bianco	lire	16.500
DM91	1000 Fogli 136 colonne bianco	lire	23.000
DM92	1000 Fogli 80 colonne lettura fac.	lire	16.500
DM93	1000 Fogli 136 colonne lettura fac.	lire	23.000
DM94	500 Fogli Uso Bollo	lire	13.000
DM95	200 Buste in modulo continuo	lire	26.500
DM96	500 Ricevute bancarie e norme ABI	lire	19.000

Accessori EDP

A1	Sottostampante plexiglass 80 col	lire	40.000
A2	Sottostampante plexiglass 136 col	lire	60.000
A3	Sottostampante tubolare 80 col	lire	25.000
A4	Sottostampante tubolare 136 col	lire	30.000
A5	Portadischetti 3" 1/2 10 posizioni	lire	2.500
A6	Portadischetti 3" 1/2 40 posizioni	lire	12.500
A7	Portadischetti 3" 1/2 80 posizioni	lire	21.000
A8	Portadischetti 5" 1/4 10 posizioni	lire	2.500
A9	Portadischetti 5" 1/4 50 posizioni	lire	12.500
A10	Portadischetti 5" 1/4 90 posizioni	lire	15.000
A11	Base in gomma antistatica per mouse	lire	28.000
A12	Base in gomma antistatica tastiera	lire	30.000
A13	Base in gomma antistatica computer	lire	33.000
A14	Piastra girevole monitor e computer	lire	35.000
A15	Schermo filtrante per monitor 12"	lire	75.000
A16	Schermo filtrante per monitor 14"	lire	90.000
A17	Kit di Pulizia per Computer	lire	10.500
A18	Kit di Pulizia per Drive 5" 1/4	lire	13.500
A19	Kit di pulizia per Drive 3" 1/2	lire	22.000
A20	Gas purissimo spray	lire	6.500
A21	Copritastiera PC	lire	16.500
A22	Copritastiera PC 101 tasti	lire	16.500
A23	Copritastiera PC AT	lire	16.500
A24	Copritastiera Olivetti M19/M28	lire	16.500
A25	Copritastiera Olivetti M24	lire	16.500
A26	Copricomputer PVC per PCXT	lire	15.000
A27	Copricomputer PVC per PCAT	lire	15.000
A28	Copricomputer PVC per Olivetti M19	lire	15.000
A29	Copricomputer PVC per Olivetti M24	lire	15.000
A30	Copricomputer PVC per Olivetti M28	lire	15.000

TUTTI I PREZZI SONO IVA ESCLUSA

Conservate questo listino - Resterà valido fino alla prossima uscita