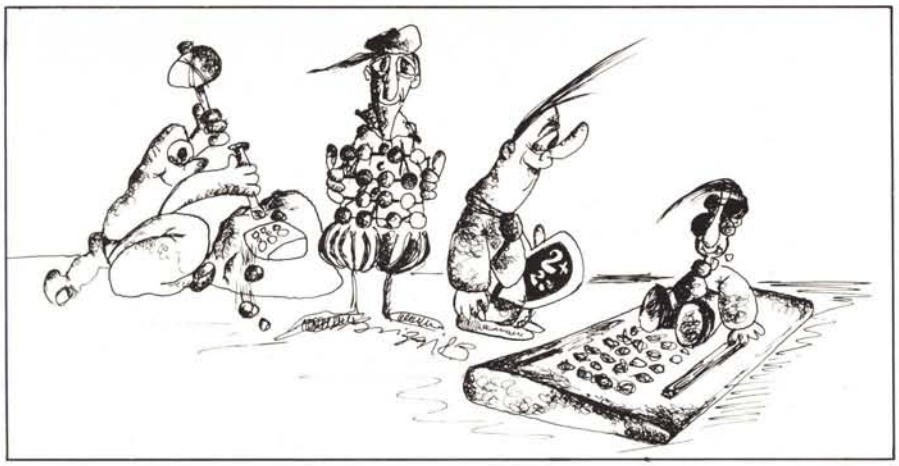

VIC da zero +64

di Tommaso Pantuso

Trasmissione: perdite di dati

Eccoci giunti all'ultimo degli articoli che illustrano il funzionamento dell'RS 232 del VIC e del C 64. Continuiamo gli esperimenti esaminando come evitare le perdite di dati.

Trasmissione sincronizzata

La volta scorsa ci eravamo fermati dopo aver effettuato il trasferimento di un programma dalla memoria di un 64 a quella di un VIC 20 servendoci della porta RS 232 di questi due computer e di un cavo di collegamento di circa un metro messo a punto per l'occasione. Se ben ricordate, il modo di trasmissione usato è stato quello a tre linee e con esso continueremo i nostri esperimenti apportando ai segmenti di programma della volta scorsa delle modifiche tali da eliminare alcuni inconvenienti che si possono verificare in trasmissione.

Riprendiamo il discorso partendo dall'ultimo fatto di rilievo su cui ci siamo soffermati nello scorso articolo.

Per maggiore chiarezza riportiamo nella figura 1 i due programmini che permettevano di usare un computer come trasmettitore ed un altro come ricevitore.

Vi abbiamo fatto osservare che il segmento usato per la ricezione contiene (linea 200) un'istruzione, IF A$ = "" THEN 200, grazie alla quale chi riceve non risente di eventuali rallentamenti da parte del trasmettitore. In pratica, quando non viene trasmesso nessun dato è come se il ricevitore stesse ricevendo un carattere nullo ("") ed in queste condizioni esso rimane in attesa fino a quando non viene inviato dal trasmettitore il prossimo dato. Se infatti si aumenta l'intervallo tra due successivi dati trasmessi tenendo ad esempio premuto il tasto CTRL sul trasmettitore, si otterrà un rallentamento delle operazioni su entrambe le macchine. Questo fatto è molto importante perché permette una certa elasticità nella trasmissione ma non è sufficiente ad evitare una perdita di dati se è il ricevitore a non essere pronto ad un certo istante, per qualsiasi ragione, a ricevere un dato. Per verificare questo fatto provate a tenere premuto il tasto CTRL sul computer che sta ricevendo: potrete notare che il ritmo di ricezione diminuisce mentre quello di trasmissione rimane invariato incurante delle necessità del ricevitore.

Per ovviare ad inconvenienti di questo tipo esistono modi di trasmissione asincroni che si avvalgono, oltre che delle linee dati, anche di alcune linee di controllo le quali permettono di realizzare una funzione di handshake. Grazie ai segnali scambiati su queste linee, il trasmettitore invia un dato solo quando è sicuro che il ricevitore è pronto a riceverlo ed a sua volta il ricevitore comunica subito l'avvenuta ricezione di un dato, sempre tramite un segnale sull'apposita linea di controllo.

Un'interfaccia RS 232 offre la possibilità di comunicare in questo modo, essendo dotata di linee di controllo che naturalmente vanno ad aggiungersi a quelle dei dati, realizzando un tipo di interfacciamento a più linee. Ci chiediamo ora se sia possibile ottenere un certo controllo sui dati anche utilizzando un modo di comunicazione a tre linee: a prima vista sembrerebbe di no, non essendo presenti linee dedicate allo scambio di messaggi di controllo in una interfaccia *three line*, quindi l'impossibilità esiste da un punto di vista prettamente hardware. Esaminando però il problema da un altro punto di vista, quello software, riusciamo a trovare un sistema che ci permette di effettuare delle verifiche sui dati in maniera abbastanza efficace avvalendoci, come linee di controllo, delle stesse linee dati. Il modo in cui realizzare quanto vogliamo è molto semplice e avviene concettualmente in maniera analoga a quanto effettuato per mezzo di linee dedicate, con la differenza che mentre in quest'ultimo caso i segnali di controllo sono rappresentati da impulsi inviati su di esse, nel nostro caso adotteremo come flag di ricetrasmissione un carattere su cui le macchine baseranno le loro verifiche.

```
1 REM * RX SINCRONO *
10 OPEN2,2,0,CHR$(136)+CHR$(16)
24 PRINT"OK"
30 FORI=4096TO4396
40 GOSUB200
50 IF A$="*"THEN80
60 POKEI,VAL(A$):PRINTA$,I
70 NEXT
80 END
200 INPUT#2,A$:IFA$=""THEN200
210 RETURN
```

```
1 REM * TX SINCRONO *
10 OPEN2,2,0,CHR$(136)+CHR$(16)
20 GETA$:IFA$=""THEN20
27 FORI=4096TO4396
30 A=PEEK(I):PRINTA,I
40 GOSUB100
50 NEXT
60 END
100 PRINT#2,STR$(A):RETURN
```

Figura 1 - Programmi di ricezione e trasmissione usati nello scorso numero.

La ricezione

Cominciamo ad esaminare la costruzione del segmento di programma utile per la ricezione illustrando il diagramma a blocchi riportato nella figura 2 e la sua traduzione in Basic riportata in figura 3.

La prima operazione che viene effettuata dal programma è quella di aprire il canale RS 232 in modo da essere pronto alla ricetrasmissione. A questo primo blocco corrisponde la linea 10 del programma che abbiamo denominato "RX SINCRONO".

```
10 OPEN 2,2,0 CHR$(136) + CHR$(16)
```

Mediante tale linea viene aperto un canale RS 232 per scambio alla velocità di 1200 baud di una parola di otto bit con due bit di stop, senza controlli di parità in modo half duplex a three line. Sul significato di queste specifiche abbiamo già insistito per cui, per eventuali dubbi, vi rimandiamo ai due precedenti articoli sull'argomento che stiamo ancora trattando nei quali sono

state illustrate in dettaglio le funzioni svolte dagli pseudo registri di comando e di controllo che adattano i vari modi di trasmissione alle esigenze specifiche di ognuno.

Ritornando al nostro diagramma di flusso con il secondo e terzo blocco il ricevitore viene posto in attesa del primo carattere inviato dal trasmettitore: solo quando tale carattere sarà ricevuto il ricevitore comincerà ad incorporare i dati trasmessi: in caso contrario rimarrà in attesa. Le funzioni svolte da questi due blocchi sono racchiuse nella linea 20 del programma che stiamo illustrando:

```
20 GET#2,A$:PRINT A$; : IF A$="" THEN 20
```

dove PRINT A$ serve solo per imprimere il dato ricevuto sullo schermo.

A questo punto entra in ballo una linea che discosta concettualmente questo segmento da quelli precedenti. Nel quarto blocco leggiamo infatti: "invia un segnale di ricevuto" (se naturalmente è stato catturato il primo carattere), espressione tradotta dalla linea 25 in:

```
25 B$="•": PRINT#2,B$
```

Come è facile osservare, il "segnale" di ricevuto è rappresentato nel nostro caso dall'asterisco e quindi più propriamente dovremo parlare di "carattere di ricevuto". Il fatto di aver utilizzato l'asterisco al posto di un altro carattere non è una scelta del tutto limitativa. Infatti se esso dovesse creare dei problemi (dovuti alla presenza di asterischi in altri punti di un blocco di dati da trasmettere, ad esempio operazioni di moltiplicazione sparse in un programma) potrebbe essere sostituito tranquillamente da qualunque altro carattere meno usato.

Superata questa prima fase inizia il ciclo di ricezione vero e proprio, mediante il quale vengono letti sequenzialmente i dati in arrivo sulla user port attendendo di volta in volta il primo carattere non nullo ed inviando, conseguentemente alla sua lettura, un carattere che avverte il trasmettitore che il dato inviato è stato ricevuto e che di conseguenza si può procedere all'invio del successivo. Questo processo viene effettuato nelle linee da 30 a 70 che a questo punto non necessitano di ulteriori commenti.

La trasmissione ed una prova

Veniamo all'esame del programma di trasmissione che chiameremo "TX ASINCRONO" ed il relativo diagramma a blocchi riportati nelle figure 4 e 5.

La prima operazione svolta è quella di apertura del canale RS 232 con le medesime modalità descritte in precedenza che rendono così compatibile il ricevitore con il trasmettitore (linea 10).

Si prosegue attendendo la pressione di un tasto qualunque sulla tastiera del trasmettitore per avviare la ricezione (linea 15): quando questa condizione si verifica, parte il ciclo di trasmissione.

Anche qui è facile osservare che la tecnica adoperata questa volta per trasmettere è diversa da quelle discusse la volta scorsa. Qui vediamo la macchina che, dopo aver inviato un primo dato, attende la comunicazione di ricevuto da parte del trasmettitore (linea 40) e solo in seguito al verificarsi di questo evento essa procede all'invio del dato successivo: in caso contrario resta in attesa.

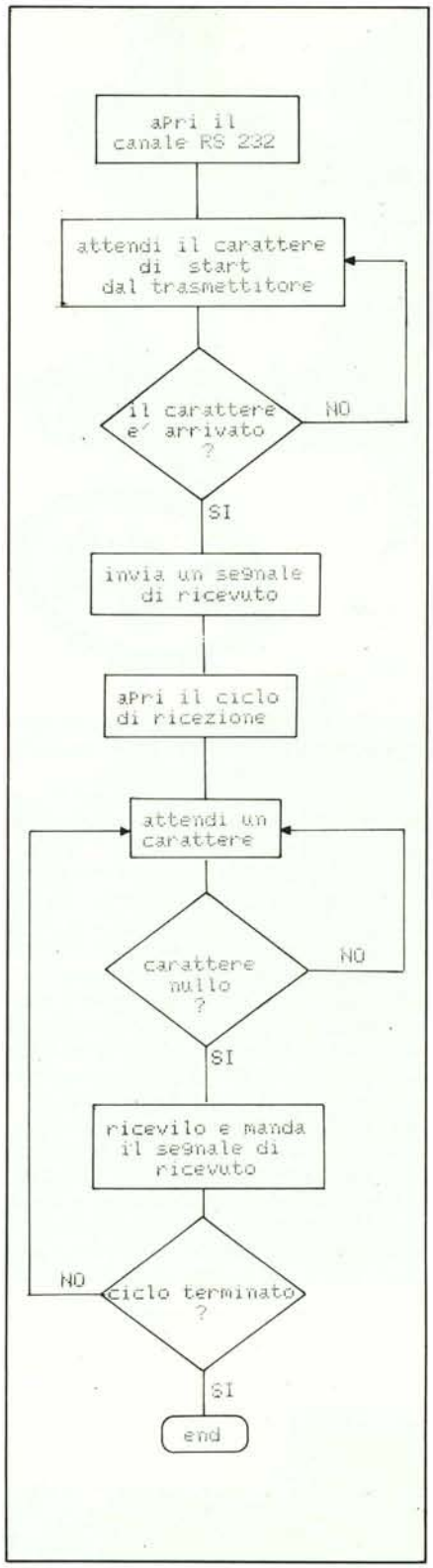

Figura 2 - Diagramma di flusso di una trasmissione sincronizzata con un carattere di controllo.

A questo punto possiamo andare ad osservare le implicazioni di quanto discusso finora sul modo di trasmissione adottato. Come al solito effettueremo una verifica mediante un semplicissimo esperimento in cui utilizzeremo il C64 come trasmettitore ed il VIC 20 come ricevitore.

```
1 REM * RX ASINCRONO *
10 OPEN2,2,0,CHR$(136)+CHR$(16)
15 PRINT"■■■ATTENDO LO START"
16 PRINT"■  DAL TRASMETTITORE"
20 GET#2,A$:PRINTA$;:IFA$=""THEN20
25 B$="*":PRINT#2,B$
30 FORI=A1 TO A2
40 INPUT#2,A$:PRINTA$,:IFA$=""THEN40
50 POKEI,VAL(A$)
60 B$="*":PRINT#2,B$
70 NEXT
```

Figura 3 - Traduzione in Basic del diagramma di figura 2.

L'esperimento consiste nel sostituire la prima linea del programma "RX ASINCRONO" contenuto nel VIC

```
1 REM • RX ASINCRONO •
```

con la prima del programma "TX SINCRONO" presente sul C64

```
1 REM • TX ASINCRONO •
```

Per le ragioni che ormai conoscete, la prima cosa da fare è far sì che i due segmenti inizino dallo stesso punto di memoria sia nel VIC (che consideriamo in configurazione base) che nell'altro computer. L'inizio del programma nel VIC nelle condizioni supposte viene portato dal sistema all'indirizzo 4097 (puntato dal contenuto delle locazioni 43 e 44 che contengono rispettivamente 1 e 16) ed uno "0" viene posto nella locazione 4097.

Viceversa nel C64 il programma inizia dalla locazione 2049 ed uno "0" viene depositato in 2048 come flag di inizio programma. Il puntatore di start BASIC è rappresentato anche in questo caso dal contenuto delle locazioni 43 e 44 che contengono rispettivamente 1 e 8.

Per prima cosa porteremo quindi l'inizio del programma nel C64 a 4097 (non potendo portare quello del VIC a 2048 per ragioni di configurazione del sistema) digitando e facendo eseguire su di esso in modo diretto:

```
POKE 44,16:POKE 4096,0:CLR
```

dopo di che potremo caricare in macchina il programma TX. Senza nessuna precauzione potremo invece digitare il programma RX per il VIC.

Il passo successivo consiste nel calcolare il numero effettivo di byte da trasmettere dal 64 al VIC. Contiamoli basandoci sulla linea da trasferire:

```
1 REM • TX ASINCRONO •
```

byte	note
2	link
2	numero di linea
1	token REM
2	spazio + asterisco
3	spazio + TX
10	spazio + ASINCRONO
2	spazio + asterisco
22	totale

Il totale ottenuto è 22, quindi l'area da trasmettere è quella che va da 4096 a 4118 per cui nella linea 30 di RX e nella 20 di TX dovremo sostituire 4096 al posto di A1 e 4118 al posto di A2.

A questo punto possiamo dare il <run>. Sullo schermo del VIC comparirà la scritta

ATTENDO LO START DAL TRASMETTITORE

mentre su quello del C64:

PREMI UN TASTO PER PARTIRE.

Premuto quindi questo famigerato tasto, il processo di trasmissione avrà inizio ed il contenuto delle varie locazioni trasferite si susseguirà sugli schermi. Se, quando i due programmi si arresteranno, andremo a listare la prima linea di RX, troveremo: 1 REM TX ... ecc.

Ma questo era abbastanza chiaro sin dalla volta scorsa; vediamo invece come si manifesta la presenza delle nuove istruzioni nel programma.

Riponete il segmento RX nelle condizioni originali modificando la scritta sostituita ed avviate i due programmi. Se, mentre i contenuti delle locazioni interessate scorrono sullo schermo, tenete premuto il tasto CTRL sul trasmettitore, come sapete i dati vengono inviati ad intervalli di tempo più lunghi e lo scorrere dei numeri rallenterà su entrambi i teleschermi. Il fatto nuovo è che ciò succede anche se tenete premuto il tasto CTRL sul ricevitore, cosa che non succedeva precedentemente. Di fatto in questo caso non si ha perdita di dati perché se rallentiamo la "velocità" di ricezione, automaticamente rallenterà anche quella di trasmissione perché il trasmettitore riceverà ad intervalli più lunghi il benestare per l'invio del prossimo dato. La parola velocità è stata posta pocanzi tra virgolette perché venga richiamata l'attenzione sul fatto che la velocità vera e propria con cui viene inviato ciascun dato è sempre quella definita nei parametri di apertura del canale RS 232, mentre quella cui ci siamo riferiti scrivendo "velocità" tra virgolette è, più propriamente, l'intervallo che intercorre tra un dato ed il successivo. Analizzando più a fondo questo fatto appaiono evidenti due cose: la prima è che la tecnica da noi introdotta non contribuisce a diminuire la probabilità di errori di trasmissione; la seconda è che non è detto che non si abbiano lo stesso perdite di dati (sparisce forse una parte delle condizioni negative che li determinano) dovute a motivi intrinseci quale ad esempio il mancato aggancio di un dato inviato ad una certa velocità.

In altre parole se vogliamo ad esempio effettuare la trasmissione di una certa mole di dati alla velocità di 2400 baud, il sistema da noi utilizzato ci permette di inviare ciascun dato in perfetto sincronismo con il momento in cui il ricevitore è pronto a riceverlo ma non impedisce errori dovuti ad altre cause quali il rumore su una linea di trasmissione troppo lunga o altro.

Vogliamo farvi riflettere inoltre sul fatto che anche il carattere di controllo che sincronizza la trasmissione potrebbe essere

```
1 REM * TX ASINCRONO *
10 OPEN2,2,0,CHR$(136)+CHR$(16)
12 PRINT"□PREMI UN TASTO PER □ PARTIRE"
15 GET#:IFA$=""THEN15
20 FORI=A1 TO A2
30 A=PEEK(I):PRINT#2,STR$(A):PRINTA,I
40 GET#2,B$:IFB$<>"*"THEN40
50 NEXT
```

Figura 5
Traduzione in Basic del diagramma di figura 4.

Figura 4 - Diagramma di flusso di una ricezione sincronizzata da un carattere di controllo.

ricevuto affetto da errore e quindi non riconosciuto. Le cose non sono quindi sempre così semplici ed immediate come sembrano. Certo potremmo perfezionare ulteriormente il programma aggiungendo istruzioni che provochino la richiesta del carattere di controllo da parte del trasmettitore o del ricevitore se questo non viene ricevuto entro un certo tempo dall'invio o dalla ricezione di un dato. Per quanto riguarda gli errori potremmo ad esempio aggiungere una sezione di programma per cui il ricevitore rispedisca indietro il dato appena ricevuto affinché il trasmettitore possa a sua volta verificare che esso è giunto senza modifiche e procedere all'invio del successivo. Naturalmente tutto è possibile ma costa un rallentamento delle operazioni (che tutto sommato in molti casi non rappresenta un fatto di rilievo) e comunque non annulla mai totalmente la probabilità di errore.

Per semplificare le cose, fermo restando il fatto che gli errori sui dati possono sempre farsi beffe di noi, sono stati introdotti codici di trasmissione molto potenti ad autocorrezione e trasmissioni a più linee di cui molte utilizzate per controlli e lo standard RS 232 ne è un esempio.

Per concludere

In conclusione dell'articolo vi presentiamo due programmini dimostrativi (i quali non utilizzano la tecnica descritta) che hanno sempre lo scopo di trasferire il contenuto di una certa zona di memoria da un computer all'altro ma con qualche particolarità in più rispetto a quanto precedentemente illustrato. Non dovendo trasferire programmi, non sarà necessario spostare l'inizio del Basic sul C64.

I due segmenti, cui abbiamo dato il nome di TX MEMORY (trasmettitore) ed RX MEMORY (ricevitore) vengono riportati nelle figure 6 e 7: esaminiamone molto brevemente il funzionamento.

Dato il <run>, per prima cosa compariranno sullo schermo dei due computer gli estremi delle zone occupate dai due programmi. Questa informazione è utile soprattutto sul ricevitore in quanto conoscendo tali estremi, conosceremo anche la zona che non dovremo invadere con il nostro trasferimento. Se ora diamo un'occhiata allo schermo del trasmettitore vedremo che esso ci chiede gli estremi della

```
1 REM * TX MEMORY *
5 P=PEEK(45)+PEEK(46)*256
6 PP=PEEK(43)+PEEK(44)*256
7 PRINT"■AREA  OCCUPATA  DAL   PROGRAMMA:■"
8 PRINT"DA ";PP;"A ";P
10 OPEN2,2,0,CHR$(136)+CHR$(16)
19 PRINT"■AREA DI PARTENZA■"
20 INPUT H,K
22 Z=K-H
29 PRINT"■INIZIO AREA DI DESTINAZIONE■"
30 INPUT C
32 D=C+Z:IFD<0THEN7
33 PRINT"AREA DI DESTINAZIONE:■  DA";C;"A ";D
35 GOSUB 200
40 PRINT"■PREMI UN TASTO PER     TRASMETTERE "
50 GETA$:IFA$=""THEN50
60 FORI=HTOK
70 A=PEEK(I):PRINTA,I
80 GOSUB 100:NEXT
90 END
100 PRINT#2,STR$(A):RETURN
110 END
200 PRINT#2,STR$(C)
210 PRINT#2,STR$(D)
220 RETURN
```

Figura 6 - Questo programma trasmette in più l'indicazione del pezzo di memoria in cui vogliamo depositare il programma nel ricevitore.

```
1 REM * RX MEMORY *
5 P=PEEK(45)+PEEK(46)*256
6 PP=PEEK(43)+PEEK(44)*256
7 PRINT"■AREA  OCCUPATA  DAL   PROGRAMMA:■"
8 PRINT"DA ";PP;"A ";P
10 OPEN2,2,0,CHR$(136)+CHR$(16)
20 PRINT"■RICEVITORE PRONTO"
30 INPUT#2,C$:IFC$=""THEN30
40 INPUT#2,D$
50 C=VAL(C$):D=VAL(D$)
51 REM PRINTC$,D$
60 PRINT"■AREA DI DESTINAZIONE:■"
61 PRINT"DA ";C;" A ";D
70 FORI=CTOD
80 GOSUB 100
90 POKEI,VAL(A$):PRINTA$,I
95 NEXT:END
100 INPUT#2,A$:IFA$=""THEN100
110 RETURN
```

Figura 7 - Programma che riceve anche l'indicazione della zona di memoria in cui deve andare a depositare i dati ricevuti.

zona di cui vogliamo trasferire il contenuto e, inseriti i dati richiesti, verrà richiesto un ulteriore dato, cioè l'inizio della zona, relativa al ricevitore, interessata al trasferimento. Inserito quest'ultimo dato e premuto <return> il trasmettitore calcolerà l'ampiezza della zona all'interno del ricevitore in cui saranno posizionati i dati e la comunicherà a quest'ultimo. A questo punto potrà avere inizio il trasferimento. Appare evidente come per mezzo di TX sia quindi possibile interagire su RX comunicandogli gli estremi della zona interessata al trasferimento che saranno utilizzati come estremi di un ciclo di FOR ... NEXT.

Come aggiunta presentiamo in figura 8 un'ulteriore versione del programma RX con l'aggiunta di una sezione che provvede ad imprimere sullo schermo il dump esadecimale del pezzo di memoria ricevuto.

Tenete presente che quest'ultimo programma, eseguendo un numero di passaggi superiore a quelli compiuti dal trasmettitore, non è in grado di agganciare tutti i dati se operiamo alle velocità usate finora; se quindi sorgeranno dei problemi, sarà il caso o di diminuire le velocità di trasmissione-ricezione o di aggiungere dei caratteri di controllo come abbiamo fatto in precedenza. Noi abbiamo preferito lasciarlo in questa forma perché chi vorrà cimentarsi con gli esperimenti da noi proposti possa rendersi conto degli effettivi problemi che possono sorgere in trasmissione.

Di sfuggita, vi informiamo della presenza di un apposito registro, il REGISTRO DI STATO dell'RS 232 posto nella locazione 663. Il contenuto di tale registro è normalmente 0 e varia a seconda dell'errore commesso. In pratica ogni bit è legato ad un errore specifico e per l'identificazione del tipo si faccia riferimento alla figura 10. Come ultima cosa vi informiamo della presenza di due buffer complessivamente da 0.5K, uno per la trasmissione ed uno per la ricezione, posti nella parte più alta della RAM i quali conservano temporaneamente i dati che verranno in seguito processati. L'indirizzo di partenza di queste due zone è puntato da 4 locazioni di memoria (2 per ognuno).

Quanto abbiamo illustrato negli ultimi tre articoli dedicati all'RS 232 del VIC e del C64 non è certo tutto quello che si può dire sull'argomento, forse gli esempi e gli esperimenti sono stati affrontati in maniera troppo elementare, però riteniamo che sia sufficiente per prendere con l'argomento un dimestichezza tale da permettere in seguito di procedere su questa strada in maniera autonoma. MC

BIT	DESCRIPTION
7	break detected bit
6	DSR signal missing bit
5	unused bit
4	CTS signal missing bit
3	receiver buffer - empty
2	receiver buffer overrun bit
1	framing error bit
0	Parity error bit

Figura 9 - Registro di stato tramite il quale è possibile tenere sotto controllo la trasmissione rilevando prontamente la causa dell'errore.

```
10 REM RX MEMORY DUMP
15 R=0
20 P=PEEK(45)+PEEK(46)*256
30 PP=PEEK(43)+PEEK(44)*256
40 PRINT"■AREA  OCCUPATA  DAL   PROGRAMMA:■"
50 PRINT"DA ";PP;"A ";P
60 OPEN2,2,0,CHR$(136)+CHR$(16)
70 PRINT"■RICEVITORE PRONTO"
80 INPUT#2,C$:IFC$=""THEN80
90 INPUT#2,D$
100 C=VAL(C$):D=VAL(D$)
120 PRINT"■AREA DI DESTINAZIONE:■"
130 PRINT"DA ";C;" A ";D
140 FORI=CTOD
150 GOSUB 180
160 POKEI,VAL(A$):GOSUB210
170 NEXT:END
180 INPUT#2,A$:IFA$=""THEN180
190 RETURN
210 IFR=0THENB=I:GOSUB290:PRINTB$+" ";
220 IFI/5-INT(I/5)=0ANDR=1THENB=I:GOSUB290:PRINTB$+" ";
225 R=0
240 B=VAL(A$):GOSUB290
250 PRINTRIGHT$(B$,3);
265 R=1
280 RETURN
290 HX$="0123456789ABCDEF":B$=" "
300 W=B:FORS=1TO4:W1=INT(W/16):RS=W-16*W1
310 B$=MID$(HX$,(RS+1),1)+B$:W=W1:NEXTS
315 IFI/5-INT(I/5)=0ANDR=1THENPRINTCHR$(13);
320 RETURN
330 PRINTB$+" ";:RETURN
```

Figura 8 Programma ricevitore che imprime sullo schermo il Dump esadecimale del "pezzo" di memoria ricevuto.

DEDICHIAMO IL NUOVO POCKET COMPUTER CASIO® PB-410 A DIEGO MARADONA.

Ci sarebbe proprio piaciuto ingaggiarlo per la nostra campagna, ma il Napoli ci ha battuti sul filo di lana dei 12 miliardi. Peccato. Il nuovo pocket personal CASIO PB-410, con memoria RAM a schede da 2 a 4 KB intercambiabili e utilizzabili anche come sistema di registrazione dati, è il personal ideale per tutti coloro che non vogliono rinunciare a portare costantemente con sè uno strumento per la gestione d'affari personale.
Versatile e capace, con linguaggio di programmazione BASIC, e una Banca Dati per appunti e agenda personale predisposta per l'uso senza necessità di programmazione, lo consigliamo caldamente a Diego Maradona che a soli 23 anni riesce a muovere attorno a sè tanti miliardi e a risvegliare l'interesse di una città dinamica come Napoli.
Siamo anzi ben disposti a regalargliene uno e a illustrargliene i pregi di persona quando e come vorrà: lo aspettiamo a Milano nostro ospite. Da parte nostra speriamo che l'effetto Maradona continui e ci auguriamo infatti di vendere tanti PB-410 proprio a Napoli. In tutti i casi il PB-410 è ovviamente dotato di una facile guida alla programmazione, di una interfaccia per cassette e, per gli affari che possono lasciare traccia di sè, di una utilissima stampante.

CASIO®

Gioielli della microinformatica.

DITRON
DISTRIBUTORE NAZIONALE
SpA

Viale Certosa, 138 Milano - Tel. 02/3085645 (5 linee ric. aut.)

IL BITTEGONE

00123 ROMA VIA U. COMANDINI 49 TEL 06/6133025-2920559 TX.621166 FEPAG II

di Felice Pagnani

SISTEMI SUSY 2

48K pad numerico	744.000
64K pad num. e t.funz.	843.000

SCHEDE MADRI SUSY 2

48K zoccoli	408.480
64K " "	501.700

INTERFACCE PER SUSY 2

DISK DRIVE CARD	80.500
DISK DRIVE DOUBLE/FACE	126.000
PRINT INT. EPSON CARD	75.700
PARALLEL PRINTER CARD	72.200
UNIVERSAL PRINTER CARD	134.400
PRINT CABLE	37.800
LANGUAGE CARD	105.000
16K RAM CARD	105.000
INTEGER CARD	105.000
Z80 CP/M CARD	78.100
80X24 VIDEO CARD	134.300
80X24 VIDEO W/SWITCH	175.800
RS232 CARD	112.300
COMMUNICATION CARD	112.300
7710 ASYNCHRONOUS CARD	231.900
FORTH CARD	92.700
GRAPPLE CARD W/BUFFER	466.500
GRAPPLE CARD W/CABLE	186.900
BUFFER CARD 32KRAM	312.500
BUFFER CABLE 2/PCS	58.800
6522 CONTROL CARD	87.900
IEE488 CARD	268.600
SPEECH CARD W/SW	87.900
128K RAM SATURN W/SW	537.200
6809 CARD W/FLEX MANU.	308.900
MUSIC SYSTEM W/SW	131.800
WILD CARD	90.300
PAL CARD W/MODULATORE	113.300
AD/DA CARD W/SW	356.500
EPROM WRT (2716-32-64)	131.800
CLOCK CARD W/SW	126.900
OLIVETTI PRAXIS CARD	243.600
IBM CARD 8088 W/SW	659.300
RGB CARD W/CABLE	136.700
APPLI C. Z80,64K W/SW	693.000
IC TEST CARD W/SW	369.300
INTER.DRIVE 48/96TPI	407.000

ACCESSORI PER SUSY 2

RF MODULATOR	12.800
RF MOD. W/VOICE	15.000
JOYSTICK	22.200
DESK TOP JOYSTICK	30.100
JOYSTICK AUTOCENTERING	44.400
JOYSTICK AUTO QUIK FIRE	53.200
"MOUSE" 91 FUNCTIONS	133.200
FAN	22.200
COOLING FAN W/CABLE	66.600
LIGHT PEN HI RES.	417.300
TAVOLETTA GRAFICA	128.700
DRIVE 5" SINGLE HEAD	421.800
DRIVE 5" SINGLE HEAD	532.800
DRIVE 5" DOUBLE HEAD	748.400
DOUB.DRIVE MULTITECH	1.043.400
DRIVE 5" D/H 96 TPI	592.600
TASTIERA MULTITECH	222.000
MONITOR COLORE RGB 14"	560.000
MONITOR B/N,VERDE,OCRA	205.000

PRO-DOS (COMPATIBILE)

I1 PRO-DOS compatibile con tutti i compatibili. Riconosce l'ambiente in cui si trova e ci si adatta automaticamente:
Il disketto: 35.000
Un altro modo per rendere compatibile con il PRO-DOS il proprio compatibile: EPROM PRO DOS UNIVERSALE si inserisce al posto della ROM F8 e non e' piu' necessario modificare i dischetti. 38.500

SISTEMI SUSY 5

128K COLORE/GRAFICA, 2 DRIVES DF/DD 2 PORTE SER. 1 PAR. MONITOR B/N 25 MHZ 4.000.000
SUSY 5/HD Come sopra ma con Hard Disk 10 Mbytes, 1 floppy
PREZZO 6.000.000

INTERFACCE PER SUSY 5

MOTHER BOARD	1.100.000
FLOPPY CONTR. DRC-1	325.000
MULTIFUNCTIONS CARD	650.000
WINCHESTER CONTROLLER	690.000
GRAF. COL.1024X1024	4.500.000

ACCESSORI PER SUSY 5

DRIVE DOUBLE/H 48TPI	535.000
DRIVE S/H 48TPI	400.000
MONITOR 12" B/N,VERDE,OCRA 25 MHZ BANDA ERGONOMICO	245.000

CABINETS PER SISTEMI

Monitor ergonomico con spazio per scheda e alim. 100.000
Computer in due pezzi 75.000
Per SUSY 2 FP-4403 con tastiera tipo IBM alloggiamento per minifloppies tutta la meccanica di fissaggio. Elegante, facile da montare 465.800

PRINTER AD AGHI

STAMPANTI A MARGHERITA

SP100 100 CPS GRAFICA	720.000
SP120 120 CPS GRAFICA	820.000
SCP400 COLORE 40/80 CL	540.000
SP560LG GRAF.NERO/ROS.	378.000
STAMPANTE MARGHERITA	900.000

XY PLOTTER

* Compact DIN A3 size.
* Numerous intelligent funct.
* 4 Colour graphics.
* OHP Film Drawings.
* Usable as printer.
* Support graphics and special graphic symbols 1.800.000

MATERIALI DI CONSUMO

DISCHETTI 5" ACCUTRACK	37.000
DATA LIFE SF/DD BOX	40.000
NASHUA SF/DD	37.000
DISCO DIAGNOSI	65.000
RHONE POULENK SF/DD	40.000
RHONE POULENK DF/DD	60.000
CARTA 2000 FOGLI 80 C	30.000
CONTENITORE DISCHI	44.000

STAZIONE SUSY SUPER-GRAPHIC

512X512 4 piani di colore, 4096 colori in pallet da 16, tavoletta digitale, 2 floppy, tastiera separata, una porta parallela, monitor colore HR BARCO 512x512 e uno B/N 25 MHz adatta per sviluppo disegni ad alta risoluzione, grafica pubblicitaria, artistica, scientifica. lit. 8.500.000
MONITOR COL.512X512 2.040.000

UN SISTEMA PER CHI INIZIA

SUSY 2 48K, FLOPPY CONTROLLER, DRIVE 5",MONITOR 12", User's Manual, dischetto con il sistema operativo velocizzato.

TOTALE 1.300.000

NOSTRA PRODUZIONE

SUSY SUPER-GRAPHIC

Trasforma un SUSY][o compatibile, un APPLE][E in un potentissimo sistema grafico. I piu' alti livelli della grafica per impieghi professionali prima irraggiungibili per l'alto costo ora sono alla portata dei piu'.
Risoluzione 1Mega pixels (1024 x 1024 b/n o 512x512 4 piani colore). Generazione di disegni da hardware: vettori, cerchi, archi e rettangoli. possibilita' di PAN, SCROLL, ZOOM (fino a 16 volte). Uscita RGB. 128 KRAM a bordo, processore NEC7220 (16bit). Software fornito: interprete e PAINT consente l'uso di una tavoletta grafica digitale o del joystick. 1.750.000
Schedino PIG-BACK SSG per avere un uscita RGB lineare, videocomposito e una tavolozza di 4096 colori 250.000
Software opzionale:
PRIMITIVE consente l'uscita dal PAINT su basic con possibilita' di aggancio di set di caratteri e figure, generare delle funzioni. 175.000
RAM DISK consente di usare la scheda anche come disco virtuale 100.000

SCHEDE IN STD-BUS Z80

ADATTE PER CONTROLLI INDUSTRIALI IN AMBIENTI AD ELEVATO STRESS - FUNZIONAMENTO 24 ORE SU 24 - ELEVATISSIMA AFFIDABILITA' - ADATTE ANCHE PER GESTIONALI CON IMPIEGO GRAVOSO.

CPU-I/O 64KRAM 2 seriali 1 parallela zocc.EPROM	750.000
CPU-I/O cs.senza RAM	390.000
FC2 Floppy Contr.DD	515.000
DR1 RAM 64K	470.000
DR2 RAM 256K	880.000
SPP1 4 p.seriali	407.000
BW1 8 zoccoli Byte W.	242.000
PPP1 4 p.parallele	319.000
AD1 adapter Winchest.	96.000
accessori:	
BOX 4 posti scheda	180.000
BOX 6 con terminazioni	220.000
BOX 8 con terminazioni	270.000
PAL/20 PAL PROGRAMMER	900.000
PPG/128 EPROM PROG	700.000
VDB 033 scheda video 80X24	
RS232	320.000

SOFTWARE DI SUPPORTO ALLE SCHEDE: ADATTAMENTO CP/M, MP/M MULTIUSERS, BASIC RESIDENTE.

UN TERMINALE IN UFFICIO UNO A CASA E I DATI.....IN TASCA

Un nuovo concetto di portatilita':
SISTEMA 10 POCKET Z80 4MHZ, 2 porte seriali, 1 parallela, 64kRam, 10MBytes Winchester, 700KBytes minifloppy. CP/M. Piu' piccolo di un beauty-case a lire 5.000.000

GESTIONALE FP10M MODULARE ESPANDIBILE

10MBytes + 1 Floppy 8", 256K RAM, 2 PORTE SERIALI, 1 PARALLELA, 1 TERMINALE VIDEO,
PREZZO 7.900.000
CON DUE TERMINALI 8.800.000
CON 5 TERMINALI 11.800.000

GESTIONALE FP10 SBC, 10MBYTES + 700 KBYTES minifloppy. 64k RAM. Interfacce 2 seriali, 1 parallela. 1 video 6.800.000

FP VIDEO TERMINAL 80X24
HAZELTINE 1500 COMPATIBLE
ERGONOMICO basculante e orientabile, tastiera staccata super piatta 92 tasti con user keys, monitor verde 25 MHz, estetica gradevole 900.000

SUSY 2 E' APPLE 2 COMPATIBILE

SUSY 5 E' IBM PC COMPATIBILE

Tutti i prezzi sono IVA esclusa, pagamento in contanti, spedizioni in tutta Italia contrassegno. GARANZIA 3 MESI.

MODULO D'ORDINE

nome *ind.* *cap* *citta*

p.iva o c.fiscale *mezzo spedizione*
paghero' in contrassegno al ricevimento il seguente materiale: (o il materiale elencato nel foglio allegato)

n. ..... *n.* .....
n. ..... *n.* .....
n. ..... *n.* .....
n. ..... *n.* .....
n. ..... *n.* .....
n. ..... *n.* .....
707.....+IVA.....707.gen.....
Data *firma*

COMUNICAZIONE

Nonostante i massimi storici che il dollaro conquista noi abbiamo aumentato del minimo. I prezzi che pagate sono quelli indicati, quindi niente sorprese all'atto della fatturazione per il mese di DICEMBR

LA PROPOSTA DEL MESE
Una scatola con 10 dischetti con tutti i migliori giochi del mondo piu' un joystick analogico autocen. 100.000
Con un joystick a interruttore adatto anche per Commodore 64 solo lire 75.000

ULTIMISSIME TELEX: PER SUSY 2: TOUCH PAD WITH JOYSTICK Lit. 180.700. * DISKETTE STORAGE (CAPACITY:100PCS) LIT. 22.800. * JOHN BELL A/D CARD 166.000 * REPLAY][CARD 146.500 * PROM WRITER 513.000 * 8748/49 MPU PROGRAMER 415.000 * NEW EPROM WRITER 136.700 * IMAGE PROCESS CARD 605.600 * SUPER PARALLEL CARD 327.000 * TELEFAX CARD 537.000 * PORTABLE PC(W/2 DRIVES,1MONITOR 3.700.000 *