VIC da zero

Con questo articolo inizia una serie dedicata al VIC-20, sicuramente ai primissimi posti per economia e diffusione. Ci occuperemo di argomenti legati sia all'hardware sia al software, partendo in ogni caso dagli inizi in modo da rendere sempre la trattazione accessibile a tutti.

La possibilità di comunicare con l'esterno tramite lo scambio di informazioni è una delle più rilevanti qualità di un computer.

Noi introdurremo il lettore alle tecniche d'ingresso-uscita, mettendolo in grado di sfruttare meglio le potenzialità del proprio microcomputer oltre che per calcoli, giochi e gestioni, anche per il controllo di processi tramite l'uso e le applicazioni delle porte I/O e dei timer interni e per il colloquio con un altro computer o con una qualunque periferica con i protocolli handshake.

Per quello che riguarda la parte teorica tratteremo l'argomento in modo del tutto generale, quindi le conclusioni saranno concettualmente estendibili; in particolare concretizzeremo tali conclusioni con esperimenti e circuiti relativi al VIC 20, contenente nel suo interno un microprocessore 6502 e due VIA 6522 (questi ultimi ci interessano direttamente). In ogni caso, appreso l'uso delle tecniche essenziali, chiunque sia dotato di buona volontà potrà facilmente trasferire da solo molti problemi e risultati alla macchina in suo possesso, una volta soddisfatta la conoscenza degli indirizzi necessari nella memoria del proprio computer. Inoltre, chiunque sia interessato a farlo, grazie ai circuiti pubblicati potrà montare un microlaboratorio da inserire nella user port del VIC 20 per poter verificare i risultati raggiunti. Gli stessi circuiti potranno essere usati come interfaccia per la raccolta di dati esterni eventualmente da elaborare (come quelli provenienti da sistemi di misura o di sorveglianza) o per l'invio di dati verso l'esterno sotto forma di parole binarie o di forme d'onda, come la generazione e la trasmissione sulla linea telefonica delle previste sequenze che codificano i numeri, per la realizzazione di un combinatore telefonico gestito interamente dal computer.

Prima di entrare nel vivo della discussione, cioè prima di parlare di PIO, TIMER, VIA, PRIOT, ecc... bisogna rinfrescarsi le idee su alcuni concetti fondamentali quali quelli di numerazione binaria, esadecimale e quello di stato logico, indispensabili per poter digerire completamente la minestra che sarà servita in seguito.

Rappresentazione di numeri

Un sistema di simboli e delle regole che assegnano ad ogni loro combinazione uno ed un solo significato si chiama *sistema di numerazione*.

I calcoli di amplificatori, di filtri od i conti della spesa vengono normalmente eseguiti seguendo il sistema di numerazione decimale (arabico) o, come si suol dire, in base 10. L'espressione "in base 10" indica che la rappresentazione dei numeri è basata su un alfabeto di dieci simboli e la loro associazione forma un numero; in questo caso, cioè di numerazione in base 10, essi sono: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

I sistemi di numerazione odierna sono detti *posizionali* in quanto ogni cifra assume una determinata importanza a seconda della posizione che occupa nel contesto della stringa che compone il numero. Senza dilungarci in noiose argomentazioni, diamo qualche esempio riferendoci a numeri interi positivi.

Consideriamo il decimale 3041; esso può essere scritto come

$$3000 + 40 + 1 = 3041$$

o, in maniera più significativa come

$$3 \times 10^3 + 0 \times 10^2 + 4 \times 10^1 + 1 \times 10^0 = 3041_{10};$$

si osservi come, una volta specificato il numero, le cifre si associno in uno ed un solo modo. Le potenze di 10 per cui vengono moltiplicate le cifre dell'alfabeto che compone la base vengono dette *pesi* quindi, in questo caso, i pesi sono:

$10^0 = 1$, $10^1 = 10$, $10^2 = 100$, $10^3 = 1000$.

Il numero a pedice dell'insieme di cifre, indica la base in cui si sta lavorando.

Proviamo ora a rappresentare il numero 217_{10} in base 2.

Come già detto, "in base 2" significa che

N	W=INT(N/B)	RESTO=N-B*W
47	47/2=23	47-2*23=1
23	23/2=11	23-2*11=1
11	11/2= 5	11-2* 5=1
5	5/2= 2	5-2* 2=1
2	2/2= 1	2-2* 1=0
1	1/2= 0	1-2* 0=1

$47_{10} = 101111_2$

Figura 1 - Conversione decimale-binario.

N	W=INT(N/B)	RESTO=N-B*W	ESA
9515	9515/16=594	9515-16*594=11	B
594	594/16= 37	594-16* 37= 2	2
37	37/16= 2	37-16* 2= 5	5
2	2/16= 0	2-16* 0= 2	2

$9515_{10} = 252B_{16}$

Figura 2 - Conversione decimale-esadecimale.

l'alfabeto è formato da due soli simboli, 0 e 1, che in questo caso prendono il nome di *bit*, abbreviazione del termine inglese *binary digit* (cifra binaria) e che i pesi sono rappresentati da potenze di 2, cioè:

$2^0=1$, $2^1=2$, $2^2=4$, $2^3=8$, ecc... quindi 217_{10} si rappresenta in base 2 come

$1\times2^7 + 1\times2^6 + 0\times2^5 + 1\times2^4 + 1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0 = 11011001$.

Viceversa, se il numero è dato in base 2, per ottenere il corrispondente in base 10 basta moltiplicare la prima cifra più a destra per 2^0, la seconda per 2^1 e così via fino all'ultima, sommando poi i risultati.

Osservate ancora una volta l'importanza delle cifre in relazione alla posizione occupata: esse diventano sempre più significative, cioè influiscono sempre più sulla grandezza del numero man mano che ci si sposta verso sinistra. Se dividiamo il numero precedente in due parti ciascuna di 4 bit, dette nibble, l'insieme delle 4 cifre più a sinistra si dirà parte più significativa o parte alta mentre quello più a destra parte meno significativa o parte bassa.

Un analogo discorso si può fare per la rappresentazione di un numero in esadecimale, cioè in base 16.

In questo caso i sedici simboli del sistema sono

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, dove le lettere A, B, C, D, E, F sono state introdotte per rappresentare con un solo carattere i numeri da 10 a 15. Il significato posizionale di ciascuna cifra esadecimale è legato alle potenze di 16, in altre parole i pesi sono:

$16^0=1$, $16^1=16$, $16^2=256$, $16^3=4096$,... quindi ad esempio 4128_{10} si può esprimere in base 16 come

$1\times16^3 + 0\times16^2 + 2\times16^1 + 0\times16^0 = 1020$

ed il numero 35840_{10} come

$8\times16^3 + 12\times16^2 + 0\times16^1 + 0\times16^0 = 8C00$;

il processo di conversione inverso è analogo a quello spiegato per i numeri binari.

Un'altra cosa utile da conoscere è la codifica di un numero binario in esadecimale.

Prendiamo in proposito il numero 11110010, dividiamolo in gruppi di 4 bit ed associamo ad ogni gruppo il proprio valore esadecimale:

$1111_2 = 15_{10} = F_{16}$ e $0010_2 = 2_{10} = 2_{16}$;

affianchiamo quindi i due numeri esadecimali ottenuti ricavando il valore $F2_{16}$ che è la rappresentazione esadecimale del numero binario 11110010.

Aggiungiamo qualche cenno sulla conversione da base 10 a base 16 e 2.

Consideriamo il numero 47_{10} e ricaviamo la sua rappresentazione posizionale in base 2. Il procedimento è il seguente: si divide 47 per 2, si conserva il resto e si ripete l'operazione con la parte intera della divisione; poi si affiancano i resti mettendo più a sinistra l'ultimo valore trovato ottenendo la rappresentazione cercata.

Riportiamo in figura 1 una tabella che riproduce tale algoritmo passo-passo con b = 2.

Per la conversione in base 16, la logica è la stessa: cambia solo b che diventa 16. In figura 2 riportiamo anche per questo caso il calcolo dettagliato riferendoci questa volta al numero 9515_{10}.

Conversioni con il computer

Nello stesso articolo presentiamo un programma, approntato per evitarvi noiosi calcoli manuali, che effettua le conversioni più utili: DECIMALE/ESADECIMALE e viceversa, DECIMALE/BINARIO e viceversa su due byte (un byte è qui un insieme di 8 bit). Una volta dato il RUN, apparirà sullo schermo il menu indicante le operazioni da svolgere per abilitare le varie opzioni. L'unica cosa da aggiungere in proposito è che quando introdurrete in macchina un numero di notazione esadecimale esso dovrà essere introdotto come una stringa di quattro elementi: ad es. il numero F2 andrà scritto come 00F2, perché in caso contrario verrà inviato sullo schermo un messaggio di errore. Lo stesso dicasi per un numero binario; lavorando su 16 bit, il programma richiederà prima l'introduzione di una stringa di 8 bit per la parte più significativa del numero stesso e poi un'altra, sempre di 8 bit, per quella meno significativa. Quindi, per introdurre 0100010011100010 dovrete scrivere 01000100, premere il tasto RETURN, poi comporre 11100010 e ripremere RETURN. Se nella stringa c'è qualche elemento che non sia 0 od 1, il programma invierà un messaggio di errore.

Stati logici ed informazioni binarie

Uno stato logico in elettronica digitale è sostanzialmente uno stato elettrico nel senso che ora spieghiamo.

Consideriamo una lampadina L, un interruttore S ed una batteria B assemblati come indicato nelle figure 3 e 4.

Diremo che quando l'interruttore è aperto, (fig. 3) ai capi della lampadina non sarà presente alcuna tensione e chiameremo tale condizione *stato logico zero*; viceversa, quando l'interruttore è chiuso (fig. 4) ai capi della lampadina sarà presente una tensione ed in tale situazione diremo

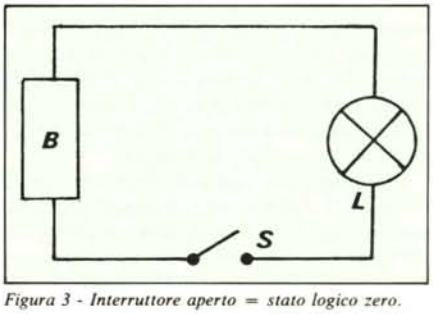


Figura 3 - Interruttore aperto = stato logico zero.

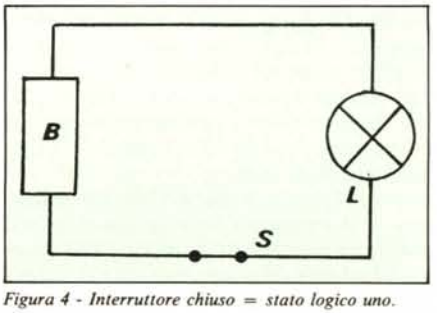


Figura 4 - Interruttore chiuso = stato logico uno.

che è presente ai suoi capi uno *stato logico uno*.

L'interruttore aperto corrisponde quindi ad uno stato 0 o L (Low) e l'interruttore

Figura 5

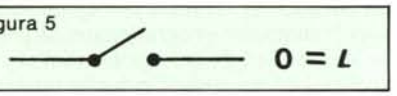
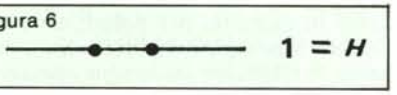


Figura 6



chiuso ad uno stato 1 o H (High).

Alla possibile condizione, cioè 0 od 1, si dà il nome di *variabile booleana*: una tale variabile è quella che può assumere solo due condizioni mutuamente esclusive.

Nel caso di circuiti elettronici, a tali condizioni (0 ed 1) si fa corrispondere un diverso valore di potenziale elettrico; per i nostri scopi è sufficiente supporre che quando un dispositivo si trova nello stato 1, cioè a livello alto, su di esso sia presente una tensione positiva rispetto a massa e che quando esso si trova nello stato 0 o a livello basso, su di esso non vi sia alcuna tensione.

Quando si considera un insieme di linee di un qualunque sistema digitale (ad esempio l'uscita parallela di una user port o i piedini di una RAM) e su ciascuna di esse è presente uno stato logico (fig. 7) il loro insieme formerà una *parola*.

In seguito sarà ampiamente spiegato come memorizzare delle informazioni e cosa succede quando vengono abilitati i singoli bit di un registro di memoria. Per il momento quanto detto abilita alla comprensione degli argomenti che seguiranno.

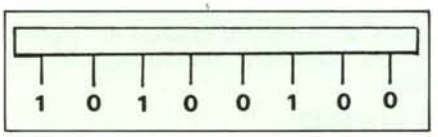


Figura 7 - Parola logica.

I chip di ingresso uscita

Supponiamo di dover realizzare un sistema d'allarme per proteggere un locale dalla presenza indesiderata di qualche intruso. Tanto per cominciare potremmo sistemare sulla porta d'ingresso un interruttore che avrebbe la funzione di *sensore* per la rilevazione del segnale di apertura e chiusura della porta stessa. Poi dovremmo realizzare una centralina rivelante la situazione in cui si trova l'interruttore e che, analizzando tale situazione, dia l'allarme in caso si verifichino le condizioni richieste. Se poi volessimo inserire e disinserire l'allarme dall'interno del locale dovremmo introdurre nel circuito dei timer che dessero all'utente il tempo di uscire dopo aver innescato il circuito ed il tempo di entrare e disinnescarlo prima che entri in azione la sirena. Per realizzare ciò si potrebbe far uso della logica digitale standard: è questo l'approccio della *logica sparsa* (random logic) o *logica cablata* (hard-wired logic). Le soluzioni progettuali di questo tipo richiedono l'assemblaggio di un certo numero di circuiti integrati rappresentanti i vari elementi logici.

Si vede come l'approccio al problema non sia del tutto immediato, richiedendo un certo tempo di studio per la progettazione ed un altro per la realizzazione, senza contare che, a lavoro fatto, è impossibile cambiare le caratteristiche del circuito. Con l'approccio a logica sparsa sostanzial-

```
10 REM*****************************************************
20 REM**********************   CONVERSIONI   **********************
30 REM**********************   (C) TP 1983   **********************
40 REM*****************************************************
50 DIML$(16)
60 POKE36879,25  :  POKE 650,128  :  PRINT"□"
70 PRINT "■  MENU'       ■ 0 ■      "
80 PRINT "■  DEC - ESA   ■ 1 ■      "
90 PRINT "■  ESA - DEC   ■ 2 ■      "
100 PRINT"■  DEC - BIN   ■ 3 ■      "
110 PRINT"■  BIN - DEC   ■ 4 ■      "
120 PRINT"■  STOP        ■ 5 ■      "
130 PRINT"                  -    "  :  INPUT "QUALE OPZIONE  "; A$
140 ON VAL (A$) GOSUB 160 , 240 , 360 , 490 , 620      :   GOTO60
150 GOTO 60
160 REM ******** D/E ********
170 HX$="0123456789ABCDEF":B$="0000":PRINT"□":PRINT"    DEC   ESA
180 PRINT:INPUT" ";B:IFB < 0 OR B > 65535 THEN GOSUB 610:GOTO 180
190 IFB=0THEN230
200 B$ = "" : W = B : FORI = 1TO4 : W1=INT(W/16) : RS = W - 16*W1
210 B$=MID$(HX$,(RS+1),1)+B$:W=W1:NEXTI:PRINT"□□":PRINTTAB(10) B$
220 GOTO 180
230 RETURN
240 REM ******** E/D ********
250 PRINT"□"  :  PRINT"  ESA    DEC" :  PRINT
260 PRINT  :  INPUT Z$
270 IF LEN(Z$) <> 4 AND ASC (Z$) = 48 THEN 60
280 IF LEN (Z$) <> 4 THEN GOSUB 600 : GOTO260
290 M=1 : N=0:FORI=1TO4 :CH=ASC(RIGHT$(Z$,I))
300 IF CH <48 OR CH>70THEN GOSUB 600 :GOTO260
310 IF CH>57 AND CH<65 THEN GOSUB 600:GOTO260
320 IF CH > 57 THEN CH = CH - 7
330 N = N + M * (CH - 48): M = M * 16 : NEXTI
340 PRINT "□□"  :  PRINT TAB(8)N  :  GOTO 260
350 RETURN
360 REM ******** D/B ********
370 PRINT "□" : R$ = "0000000000000000" : PRINT"   DEC      BIN "
380 PRINT : INPUT DC
390 IF DC < 0 OR DC > 65535  THEN   GOSUB  610     :      GOTO380
400 IF DC = 0 THEN 480
410 N$ = "" : VV = DC
420 V1=INT (VV/2) : T=VV-2*V1 : T$=RIGHT$(STR$(T),1)  :  N$=T$+N$
430 G$=RIGHT$((R$+N$),16):G2$=RIGHT$(G$,8):G1$=LEFT$(G$,8) :VV=V1
440 IF V1 <> 0 THEN 420
450 PRINT  "□□"     :     PRINT TAB(9) G1$;     :     PRINT" MSB"
460 PRINT  TAB(9) G2$;     :     PRINT" LSB"       :      GOTO380
470 GOTO 420
480 RETURN
490 REM ******** B/D ********
500 PRINT "□"                :                  PRINT"     BIN    ┐  DEC
510 PRINT : INPUT"MSB";G2$ : IF LEN(G2$)<>8 AND ASC(G2$)=48THEN60
520 IF LEN (G2$)  <>  8  THEN  GOSUB  600      :       GOTO   510
530 INPUT "LSB";G1$ : IF LEN (G1$) <> 8 THEN GOSUB 600 :  GOTO510
540 F$ = G2$ + G1$
550 FOR I = 1 TO 16  :   L$(I) = MID$(F$ ,(17-I),1)    :    NEXTI
560 N=0
570 FORI=0TO15 : R=VAL(L$(I+1)) : IFR<0ORR>1THENGOSUB600 :GOTO510
580 N=N+R*(2↑I) : NEXT I : PRINT "□□" : PRINT TAB(15)N : GOTO 510
590 RETURN
600 PRINT " FORMA NON CORRETTA"    :   RETURN
610 PRINT " VALORE NON PREVISTO"   :   RETURN
620 PRINT"□"  :  STOP
```

Figura 8

			Indirizzi memoria VIC 20 DEC	ESA
Porta dati B I/O (uscita user port)	OR B - PBØ - PB7	00	37136	9110
Porta dati A I/O	OR A - PAØ - PA7	01	37137	9111
Registro direzioni dati della porta B	DDR - B	02	37138	9112
Registro direzione dati della porta A	DDR - A	03	37139	9113
Contatore basso	T1L - L/T1C - L	04	37140	9114
Contatore alto	T1C - H	05	37141	9115
Latch basso	T1L - L	06	37142	9116
Latch alto	T1L - H	07	37143	9117
Latch basso - Contatore basso	T2L - L/T2C - L	08	37144	9118
Contatore alto	T2 C - H	09	37145	9119
Shift Register	S R	0A	37146	911A
Registro controllo ausiliario	A C R	0B	37147	911B
Registro controllo periferica	PCR (CA1,CA2,CB2,CB1)	0C	37148	911C
Registro Flag Interrupt	I F R	0D	37149	911D
Registro abilit. Flag	I E R	0E	37150	911E
	O R A	0F	37151	911F

Figura 9

mente le funzioni del sistema sono stabilite da blocchi funzionali e dai loro collegamenti e non da un programma.

Alternativamente, per svolgere lo stesso compito, si può ricorrere ad un microcomputer che realizzerebbe le funzioni opportune seguendo un preciso programma. Si capisce come per mettere a punto il sistema di controllo richiesto non sia necessario saper progettare dei circuiti elettronici ma solo saper approntare un semplice programma, che controlli le porte d'ingresso-uscita.

Inoltre è evidente come il programma possa essere adattato a qualunque situazione modificando solo qualche istruzione. Con quest'ultima soluzione sarebbe allora necessario solo un elemento che rivelasse l'apertura della porta, cioè l'informazione che il computer dovrà elaborare (p. es. un interruttore), un relé ed una sirena; quest'ultimo elemento può anche essere simulato da software tramite una linea di I/O ed in questo caso servirebbe solo un piccolo stadio amplificatore di potenza (un paio di transistori, per capirci!).

Si vede quindi come questo secondo approccio, per chi già possiede un microcomputer, sia più economico, meno laborioso e non richieda conoscenze superiori: l'unica cosa che serve è un'informazione da introdurre in macchina ed elaborare.

Iniziamo a questo proposito (finalmente!) la trattazione dei chip d'ingresso-uscita che sono normalmente connessi con il microprocessore e che servono per l'acquisizione e l'invio dei dati. Per le applicazioni e la progrmmazione di tali chip faremo riferimento al microprocessore 6502 ed in particolare al VIC 20.

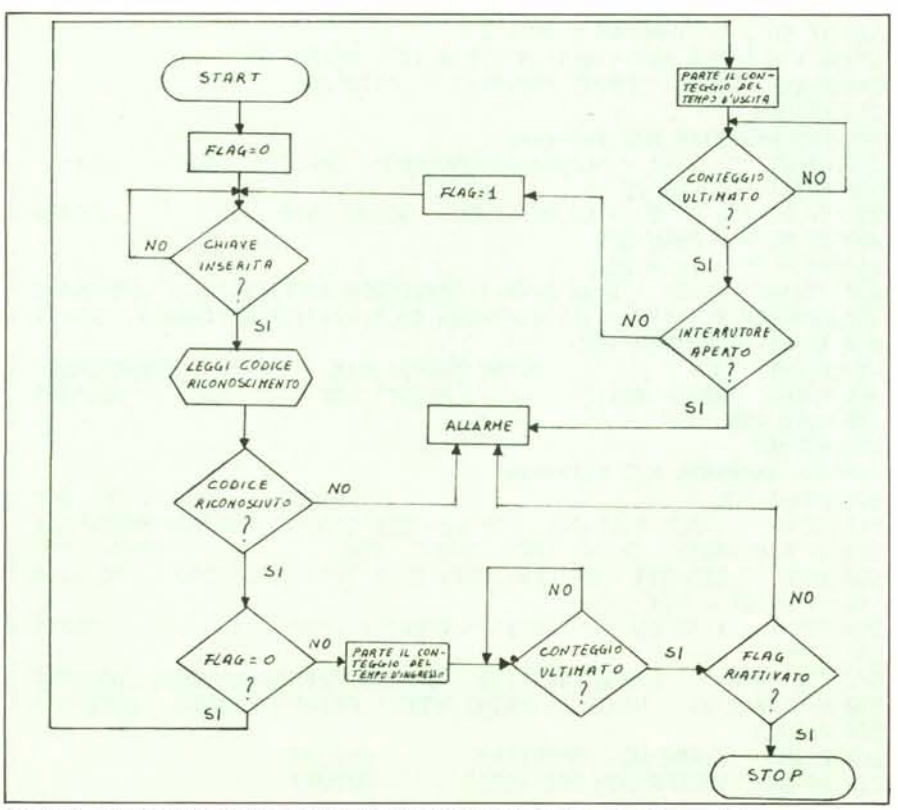


Figura 10 - Diagramma di flusso di un possibile programma per la gestione di un sistema d'allarme

Il PIO

Il più generale dispositivo d'I/O è il PIO o *porta d'ingresso-uscita programmabile*; esso fornisce almeno due porte I/O bidirezionali ad 8 bit. Tramite tale componente, opportunamente programmato, si possono leggere dati da un registro di memoria qualsiasi e trasmetterli in uscita oppure si possono acquisire dati dall'esterno e depositarli in memoria per poterli elaborare. Essendo le porte I/O del PIO bidirezionali, prima di effettuare qualunque operazione bisogna specificare quale linea della porta rappresenta un ingresso e quale un'uscita, e questo si fa ponendo un opportuno valore nel *registro direzione dati* (DDR) del PIO: una linea è un ingresso se il corrispondente bit nel DDR è posto a zero ed è un'uscita se è posto ad uno. Questa convenzione rende più sicuro il sistema: infatti all'accensione della macchina (o quando si esegue un *reset*) il contenuto di tutti i registri è zero, quindi lo è anche quello del DDR, ed essendo in tale circostanza tutte le linee configurate come ingressi, non sarà mai generato alcun impulso esterno prima dell'esecuzione del programma.

Altri elementi importanti di un PIO sono le *linee di controllo*, due per porta, che permettono di realizzare la funzione di handshaking per scambiare dati con le periferiche. In pratica una di queste due linee invia il segnale di "pronto a ricevere" e l'altra quello di "pronto a trasmettere". Altro elemento di fondamentale importanza del PIO è il *registro di controllo* (CR), che permette di configurare il chip secondo le proprie esigenze: per esempio programmando opportunamente tale registro si può decidere se ad abilitare il flag d'interrupt interno debba essere una transizione alto-basso (fronte di discesa) del segnale od una transizione basso-alto (fronte di salita).

Il VIA 6522

VIA significa *Adattatore Versatile d'Interfaccia*. Racchiude nel suo interno PIO, Timer, Shift Register. In figura 9 ne riportiamo la mappa di memoria. Si vede da tale figura che esso contiene 16 registri così suddivisi:

5 registri PIO (da 9Ø a 93 più 9F)
6 registri TIMER (da 94 a 99)
1 registro di shift (9A)
4 registri di controllo (da 9B ad 9E)

La sezione PIO è già stata descritta sufficientemente.

I TIMER, usati in uscita, serviranno per generare ritardi variabili da un microsecondo a 65 millisecondi o treni d'impulsi; usati in ingresso serviranno per misurare la durata di un impulso applicato dall'esterno o l'intervallo intercorso tra due impulsi successivi.

Lo shift register effettua la conversione seriale-parallelo e parallelo-seriale con velocità di scorrimento controllabile con tre tipi di temporizzazioni.

Tommaso Pantuso - Leo Sorge

TECNAD

METRO IMPORT

DIVISIONE INFORMATICA

Rivenditori Autorizzati:
SINCLAIR – COMMODORE – TEXAS – EPSON – SEIKOSHA – SAICO – JACKSON ED.

La **METRO IMPORT** nell'ambito della sua organizzazione, sempre all'avanguardia e in continua progressiva evoluzione sia qualitativa che tecnica, è in grado di fornire ai propri clienti, per corrispondenza o direttamente presso i punti vendita di Roma e Milano:

- **Una serie di home computers fra i più qualificati con i relativi accessori, software applicativi su cartridge, su nastro o su disco.**
- **Personal computers e periferiche con assistenza hardware da parte di personale specializzato.**
- **Assistenza software sia su pacchetti applicativi standard** (contabilità, fatturazione, magazzino, paghe e stipendi) **che per procedure personalizzate** (scientifiche e gestionali).
- **Leasing finanziario.**

Ogni realizzazione, dopo un accurato studio e sopralluogo, verrà consegnata "CHIAVI IN MANO".

Omaggio il catalogo di Informatica

Per ricevere il catalogo in omaggio, ritagliare e spedire il coupon allegando L. 500 in francobolli.

Ritagliare e spedire in busta chiusa a: **METRO IMPORT s.n.c. - VIA DONATELLO, 37 - 00196 ROMA**

Nome e Cognome .....

Indirizzo ..... **C.A.P.** ..... **Città** .....

QUOTAZIONI

Materiale nuovo imballato

CENTRO ASSISTENZA SPECTRUM

sumus

SUMUS s.r.l.
Via S. Gallo 16/r
50129 Firenze
tel. 055/29.53.61
tlx. 57.10.34

16K L. 325.000 IVA INC.

AL PARADISO DEI

SINCLAIR ZX SPECTRUM

IL PIÙ GRANDE ASSORTIMENTO ITALIANO DI ACCESSORI!

Il NEGOZIO DI SUPER SUMUS!

Tutto per SPECTRUM:

Amplificatore Audio	18.300 IVA inc.
Generatore di suoni programmabile	52.500 IVA inc.
"Orator" Sintet Vocale	105.000 IVA inc.
Master Unit (contiene Sound Synth., Ampl. Audio, Orator, Interf. X Joystick)	144.000 IVA inc.
Interfaccia RS 232	91.500 IVA inc.
Interfaccia Centronics	91.500 IVA inc.

Tastiera/contenitore per SPECTRUM o ZX-81.
Finalmente potrete usare comodamente il vostro microcomputer!
L. 79.000 IVA inc.

Mille altre novità, altri computers, video giochi, programmi ecc. ecc. Visitateci!

Grandioso assortimento di libri per SPECTRUM
novità del mese (in inglese)
L'hardware dello SPECTRUM.
Come conoscere ogni dettaglio.
Come costruire facilmente una tastiera ausiliaria - il Joystick - l'Interf. stampante - le Interfacce AD ecc. ecc.
Tutta la ROM SPECTRUM disassemblata istruzione per istruzione con spiegazioni.
20 giochi per lo SPECTRUM
Disponibili: Editor/Assembler - Debug - Forth.