VIC da zero

Seconda parte **di Tommaso Pantuso**

Continua la nostra passeggiata dentro il VIC: questa volta posiamo lo sguardo sulla gestione degli ingressi e delle uscite.

Input-Output

Cominciamo oggi ad esaminare come, adoperando un computer basato sul 6502, si possa trasferire un'informazione da un registro della sezione *PIO* del relativo 6522 verso l'uscita. Come detto in precedenza, faremo riferimento al VIC 20 e per uscita intenderemo le otto linee della sua Porta Utente (User Port). Riportiamo in figura 1 lo schema di collegamento di tale Porta e la relativa descrizione.

La parte che ci interessa direttamente è quella che va dalla lettera C alla lettera L poiché racchiude le otto linee di uscita del *PIO* che noi utilizzeremo: l'informazione

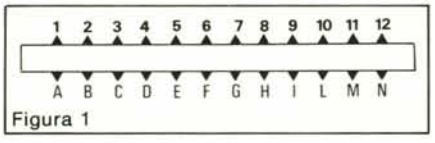


Figura 1

contenuta nel *registro IOR* verrà trasferita proprio su tali linee sotto forma di parola binaria. Vediamo in che modo si procede. Intanto si presti attenzione alle due sezioni del 6522 indicate in figura 2 ed i relativi indirizzi nella mappa di memoria del VIC 20.

Come abbiamo detto la volta precedente, le linee di uscita di un *PIO* sono bidirezionali, quindi se si vogliono trasmettere dei dati bisogna avvisare il sistema sulla configurazione da dare a ciascuna linea collegata al *registro di ingresso - uscita* (IOR) cioè se essa debba ricevere o trasmettere. Questo si fa ponendo in condizione logica 1 o 0 il bit che controlla la *direzione* della specifica linea il quale è contenuto nel *Registro Direzione Dati* (DDR): ad 1 corrisponde un'uscita ed a 0 un ingresso.

In altre parole, ogni linea di uscita *collegata* al registro IOR, può essere assimilata ad una pompa che può pompare acqua dall'esterno verso una cisterna o dalla cisterna verso l'esterno. Per stabilire la direzione del flusso sia presente un interruttore: quando l'interruttore è posto a zero, l'acqua entra; quando è posto a uno, esce. Ciò significa che DDR e IOR non sono indipendenti ma agiscono in concomitanza. Spieghiamo meglio questi concetti.

Osserviamo la figura 3: rappresenta la configurazione dei registri in oggetto al momento dell'accensione della macchina. Nel DDR tutti i bit sono 0 quindi, da quanto detto, le linee saranno tutte ingressi

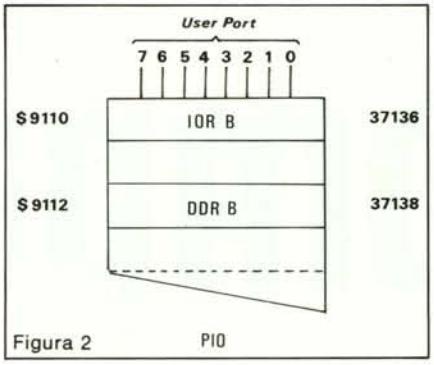


Figura 2

ed il sistema sarà pronto a ricevere qualunque *maschera* (parola di 8 bit) posta sul PB ed a trasferirla in IOR dove modificherà ciascun bit.

Infatti se in tale situazione colleghiamo la porta parallela ad un dispositivo che presenti in uscita un'informazione così codificata: 00110011, il registro IOR assumerà la stessa configurazione (fig. 4).

Proviamo ora a porre tutte le linee come uscite. Per far ciò bisogna accendere (porre

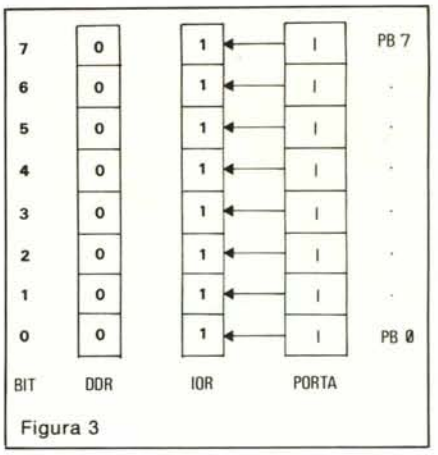


Figura 3

ad 1, *on*) tutti i bit del DDR, situato nella locazione decimale 37138 della memoria del VIC, cioè registrare in esso il codice binario 11111111: fatto questo, qualunque valore voi andrete a mettere in IOR, cioè in locazione 37136, sarà trasferito sulla porta d'uscita parallela. Apriamo una breve parentesi sul modo di scrivere o leggere in un determinato registro *RAM* per poter bene capire il funzionamento dei comandi POKE e PEEK che utilizzeremo in seguito per compiere tali operazioni sulla memoria del computer.

La *RAM è una memoria ad accesso casuale* (nel senso di: tempo necessario a raggiungere una locazione è lo stesso per tutte le locazioni) ed in essa si possono sia depositare che prelevare delle informazioni. Questo tipo di memoria trattiene il dato finché è connessa all'alimentazione, a differenza delle PROM, EPROM e ROM

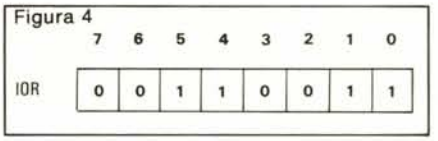


Figura 4

(*memorie permanenti*) e per questo motivo è detta anche *memoria volatile*.
Vediamo come si opera con tali elementi.

Prendiamo ad esempio una *RAM* 16 × 4: 16 è il numero di parole o registri che si possono depositare in essa e 4 è il numero di bit da cui è composta ogni parola. In essa si possono allora memorizzare 16 × 4 = 64 bit (fig. 5). Il problema è il seguente: come poter identificare il singolo registro per scrivere o leggere dentro di esso dei dati? Niente di più facile!

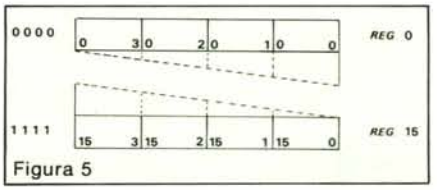
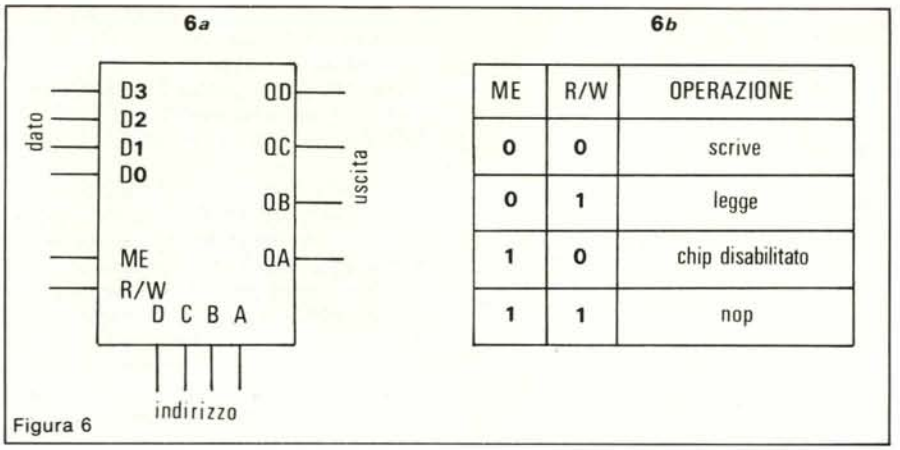


Figura 5

Supponiamo che la memoria in esame (fig. 6a) abbia come tabella operativa o *tabella della verità* quella indicata in figura 6b: essa definisce l'operazione per cui sarà



6b

ME	R/W	OPERAZIONE
0	0	scrive
0	1	legge
1	0	chip disabilitato
1	1	nop

Figura 6

abilitato il chip a seconda delle combinazioni presenti sui terminali ME (abilitazione memoria) ed R/W (lettura/scrittura).

In pratica proviamo a scrivere il dato 0101 nel registro 0011.

Per prima cosa dobbiamo selezionare il modo di operare e poiché vogliamo *scrivere* dobbiamo porre a *zero* entrambi i terminali ME ed R/W (vedi tabella).

Poniamo quindi il dato sugli ingressi D3 D2 D1 D0, l'indirizzo cui lo vogliamo inviare su DCBA ed il gioco è fatto: la parola è memorizzata! Per rileggerla basterà selezionare secondo la tabella operativa, il *modo di lettura*, ponendo ME = 0, R/W = 1, e porre sui terminali DCBA il valore binario del registro da verificare. Con tale operazione sulle uscite QD QC QB QA ci sarà restituito il contenuto della locazione selezionata. Per le sequenze indicate si confronti la figura 7. Ritornando al computer, do lettura ed il registro in cui vogliamo leggere. Quindi, ripetendoci: POKE = modo scrittura; PEEK = modo lettura. Bisogna però considerare che, con tali comandi, il valore del dato è ricevuto e restituito codificato in decimale. In altre parole se noi volessimo depositare all'indirizzo 37138 il valore binario 10000001, dovremmo scrivere

POKE 37138, 129

poiché 129 è l'equivalente decimale del binario 10000001. Viceversa il comando PEEK ci restituirà un valore decimale compreso tra 0 e 255 cioè tra 00000000 e 11111111.

Per verificare questo fate *girare* il seguente programma

10 PRINT PEEK (203): GOTO 10

che riporta sullo schermo (tramite la PRINT) il contenuto del registro 203, con-

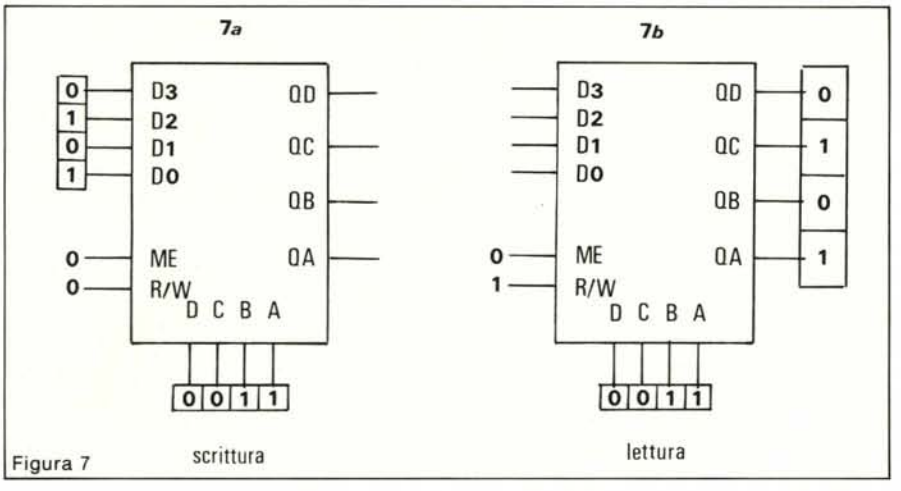


Figura 7

se non stiamo operando in LM (linguaggio macchina), per effettuare le operazioni di lettura o scrittura in un registro si possono adoperare i comandi POKE e PEEK. Infatti tutto va come se con il comando

POKE (indirizzo), valore

noi selezionassimo il modo di scrittura, il registro ed il valore che si vuole porre in esso. Viceversa, con

PEEK (indirizzo)

tutto va come se noi selezionassimo il motenente il codice di tastiera dell'ultimo elemento premuto.

Agendo sulla tastiera vedrete che il numero visualizzato sullo schermo (il decimale 64) verrà modificato ogni qual volta voi premerete un tasto e ritornerà al valore originario quando lo rilascerete.

Riguardo ai nostri problemi d'ingresso-uscita verifichiamo se è tutto chiaro con un esempio. Vogliamo configurare le linee da PB0 a PB3 come ingressi e quelle da PB4 a PB7 come uscite. Allora, sapendo che 1 nel DDR B corrisponde ad una uscita per la corrispondente linea sulla User Port e 0 ad un ingresso, i valori da porre in tale registro *bit per bit* sono:

1111 0000
uscite ingressi

Ma l'insieme di questi bit forma un numero binario e per scriverlo in DDR B (con il comando POKE) bisognerà codificarlo in decimale.

Si vede subito che:

$11110000_2 = 240_{10}$

quindi il nostro problema è risolto dall'istruzione

POKE 37138, 240

La stessa operazione, cioè la modifica di una locazione di memoria, può essere effettuata in LM. Con tale tecnica il valore che andrà ad inserirsi in memoria nella posizione voluta dovrà essere prima posto in un particolare registro del Microprocessore detto *accumulatore* (A) e poi da lì trasferito nella locazione desiderata. Nel nostro caso, dovremo caricare in A il numero 240 e poi trasferirlo in DDR B, che si trova all'indirizzo decimale 37138.

Ricordiamo che lavorando in LM i numeri andranno convertiti in codice esadecimale. La subroutine che realizza tali funzioni è la seguente:

LDA # $ F0:
carica in modo immediato l'accumulatore con 11110000

STA $ 9112:
memorizza il contenuto dell'Accumulatore in $ 9112

RTS:
ritorna dalla subroutine al programma principale.

Le istruzioni usate sono: LDA, STA, RTS.

LDA significa: carica (LoaD) il numero (#) esadecimale ($) F0 ($240_{10}$) nell'Accumulatore (A).

STA dice invece: memorizza (STore) il contenuto di A nella locazione 9112 (37138_{10}).

Con RTS si rientra dalla subroutine.

LDA, STA, RTS sono dette istruzioni *mnemoniche*, in quanto i loro nomi, essendo abbreviazioni delle funzioni che svolgono, aiutano a ricordarne l'uso.

Volendole introdurre in macchina, se si usa un assemblatore questo provvederà a trasformare automaticamente tali istruzioni in un codice operativo esadecimale (OPCODE) idoneo al microprocessore che si sta usando (cfr. listato 1) e ad inserirle nelle locazioni di memoria scelte. Il codice operativo di ogni istruzione è contenuto negli appositi manuali di programmazione in linguaggio macchina.

Se invece si vuole adoperare il comando POKE, esse dovranno essere codificate in notazione decimale prima di essere trasferite in memoria. In ogni caso, penserà poi la macchina (per fortuna!) a tradurre tutto nel linguaggio degli zero e degli uno in quanto l'unico conosciuto dai suoi chip interni.

La decodifica completa della precedente subroutine volendola memorizzare a partire dall'indirizzo $ 0334 (820 decimale), è:

LINEA	LOC $	MNEM	OPCODE	DEC
0001	0334	LDA	A9	169
0002	0335	F0	F0	240
0003	0336	STA	8D	141
0004	0337	91	12	18
0005	0338	12	91	145
0006	0339	RTS	60	96

Listato 1

```
100 PRINT"J":POKE36879,25 :C$="00000000":POKE650,128
101 PRINT"  IND     BIN   DEC  ":PRINT
102 FORI=37136TO37139:DC=PEEK(I)
103 A$="":Z=DC
104 V1=INT(Z/2):T=Z-2*V1:T$=RIGHT$(STR$(T),1):A$=T$+A$
105 G$=RIGHT$(C$+A$,8):O1$=RIGHT$(O$,8):Z=V1
106 IFV1<>0THEN104
107 PRINTI;O1$,DC:NEXT
108 PRINT:PRINT"INDIRIZZO  DA  MODIFIC CARE E NUOVO
    VALORE  "
109 PRINT:A=0:B1$=""
110 INPUTA,B1$:IFA<37136ORA>37139THEN109
111 IFLEN(B1$)<>8THEN109
112 FORI=1TO8:L$(I)=MID$(B1$,(9-I),1):NEXT
113 N=0
114 FORI=0TO7:R=VAL(L$(I+1))
115 N=N+R*(2^I):NEXTI
116 POKEA,N:GOTO100
```

Listato A

Si osservi che 9112 viene immagazzinato come 12 91 cioè in modo che venga conservato prima il byte meno significativo.

Da BASIC verrà immagazzinata nelle locazioni volute, come già detto, tramite il comando POKE (usando la codifica in decimale riportata nell'ultima colonna del listato) con il seguente segmento di programma:

10 FOR I = 0 TO 5 : READ A
20 POKE 820 + I,A : NEXT I
30 DATA 169,240, 141, 18, 145, 96

Caricate tale programma in macchina ed avviatelo. Ad esecuzione avvenuta eseguite SYS 820 ed andate a leggere il contenuto del registro 37138 con

PRINT PEEK (37138):

esso sarà 240 (vedi fig. 8).

Ricordiamo che il comando SYS manda alla locazione di memoria (per noi 820) da cui inizia il programma in linguaggio macchina e che RTS rimanda, in questo caso, al BASIC.

In questa pagina riportiamo il listato A di un programma che fornisce i contenuti dei registri IOR B, IOR A, DDR B, DDR A del 6522 in notazione binaria e decimale con la possibilità di intervento su di essi. Il contenuto del registro da modificare deve essere posto sotto forma di una stringa di 8 bit indicante il numero binario che si vuole immagazzinare in esso. Provate a farlo elaborare: sullo schermo dovrà apparire la tabellina di figura 9 e la richiesta di un eventuale cambiamento.

Potrete verificare che al momento dell'accensione in 37138 (DDR B) è contenuto il valore 0, quindi tutte le linee sono

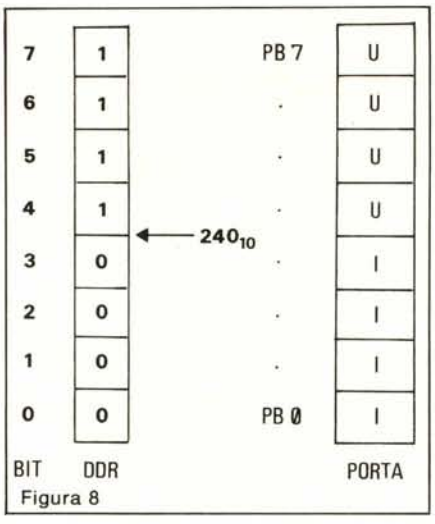


Figura 8

ingressi, e che rispondendo alla richiesta con

37138,11110000

il contenuto decimale di tale registro diverrà 240 configurando le porte come spiegato nell'esempio precedente. Se invece, quando in DDR B tutti i bit sono a zero, provate a scrivere un qualunque valore nel registro d'ingresso - uscita B, posto 37136, non noterete nessun cambiamento. Infatti

IND	BIN	DEC
37136	11111111	255
37137	01111110	126
37138	00000000	0
37139	10000000	128

Figura 9

essendo tutte le linee configurate come ingressi, esse potranno ricevere solo i dati posti sulla user port e non trasferire dati su di essa.

Diviene a questo punto necessaria la conoscenza dei dispositivi di base che permettono di prelevare o fornire l'informazione al computer e d'immagazzinarla in memoria.

Dispositivi

Quando una parola binaria si presenta in uscita sui pin (piedini) di un circuito integrato, essa è rappresentata da un insieme combinato di condizioni elettriche. D'ora in poi diremo che una linea è *a livello alto* o in *condizione 1* quando su di essa è presente una tensione di +5 volt rispetto a massa e viceversa che è a *livello basso* o in *condizione 0* quando su di essa non è presente alcuna tensione.

Ad esempio la parola 11110000 si presenterà in uscita sui terminali del 6522 relativo alla user port come indicato in figura 10.

Volendo prelevare l'informazione dovremo interporre tra l'utilizzatore (carico) ed il circuito integrato un componente di disaccoppiamento che venga *pilotato* dal chip ma che non gli assorba troppa corrente onde evitare di *caricarlo* (rischiando di danneggiarlo o di abbassare la tensione sulle uscite) e che eviti comunque un'interazione diretta tra il circuito pilotato e quello di comando. Tale elemento prende il nome di *buffer*.

Due esempi di realizzazione sono pre-

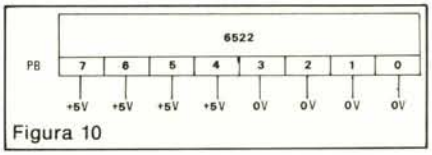


Figura 10

sentati in figura 11, dove è indicato come pilotare un Diodo Emettitore di Luce (LED) ed un relè. Si può vedere come tali dispositivi possano essere realizzati con transistori o con circuiti integrati (è stata qui utilizzata una sezione di un invertitore sestuplo, un particolare circuito integrato).

È bene far notare che quella ora descritta è la funzione elettrica di un circuito buffer. In altri casi per buffer si intende un dispositivo che accumula dei gruppi di dati in trasmissione fino a quando essi non vengano utilizzati dall'elemento ricevente compensando così le differenze di velocità fra trasmettitore e ricevitore.

Questo per quanto riguarda il prelievo di dati dall'esterno, cioè nella direzione macchina-mondo. Nella direzione opposta, cioè mondo-macchina, una volta poste le linee come ingressi basterà collegarle alla tensione di +5 volt o a massa a seconda che si voglia far leggere al computer un 1 oppure uno 0.

Due elementi circuitali che possono realizzare questa funzione sono rappresentati in figura 12.

Giunti a questo punto, abbiamo le basi sufficienti per affrontare il prossimo argomento che tratterà il collegamento di semplici dispositivi al 6502 e lo sviluppo di programmi applicativi per la loro realizzazione.

Figura 11

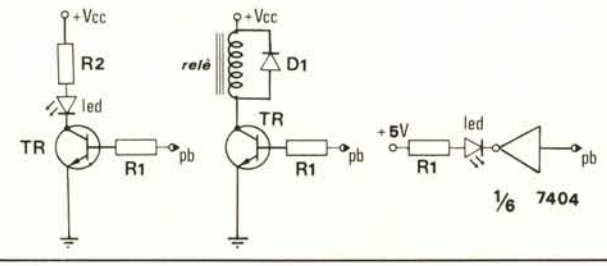


Figura 12