Riprendiamo in esame le funzioni AND, OR (inclusivo) ed OR ESCLUSIVO per esaminarne l'uso pratico e cioè come controllare, con semplici algoritmi basati sul loro uso, i singoli bit di un registro di memoria e verificare determinate condizioni.

VIC da zero

Quarta parte **di Tommaso Pantuso**

Tecniche fondamentali di controllo

La volta scorsa abbiamo presentato schemi elettrici e circuiti stampati relativi a due semplici schede didattiche utili per effettuare alcune operazioni d'ingresso dati e per verificarne altre in uscita tramite degli opportuni visualizzatori a diodi LED.

Tutto ciò è stato fatto perchè siamo convinti che, coloro i quali realizzeranno tali circuiti, verificando *praticamente* le nozioni ed i problemi posti alla loro attenzione, ne trarranno un notevole beneficio relativamente all'apprendimento e potranno perciò più facilmente elaborare soluzioni per i loro specifici problemi. Non dimentichiamo inoltre che tramite un visualizzatore si può verificare in maniera diretta l'esattezza di un qualunque programma di controllo esterno.

Non vogliamo perdere altro tempo in chiacchiere, quindi veniamo al dunque esaminando qualche elemento fondamentale.

Riassumiamo in figura 1, per comodità del lettore, le tabelle della verità delle operazioni AND, OR, EOR, NOT.

```
0 AND 0 = 0   0 OR 0 = 0   0 EOR 0 = 0   NOT 1 = 0
0 AND 1 = 0   0 OR 1 = 1   0 EOR 1 = 1   NOT 0 = 1
1 AND 0 = 0   1 OR 0 = 1   1 EOR0 = 1
1 AND 1 = 1   1 OR 1 = 1   1 EOR 1 = 0
```

Figura 1

L'operazione di OR ESCLUSIVO è stata qui indicata con EOR per linearità di esposizione in quanto EOR è il codice mnemonico dell'istruzione del 6502 che effettua appunto tale operazione in linguaggio macchina tra il contenuto dell'accumulatore ed un certo dato. Se avete montato la scheda VL1 vi consigliamo di seguire questo articolo verificando, passo-passo, quanto andremo a dire con il circuito inserito nella *porta utente* (del VIC o del 64). Riportiamo ancora una volta in figura 2 gli indirizzi del REGISTRO DIREZIONE DATI e del REGISTRO D'INGRESSO-USCITA di tali macchine. Nel seguito faremo riferimento per tutte le operazioni al VIC essendo del tutto ovvia la conversione al Commodore 64.

	VIC 20	
	DEC	ESA
DDR	37138	9112
IOR	37136	9110

	COMMODORE 64	
	DEC	ESA
DDR	56579	DD03
IOR	56577	DD01

Figura 2

Supponiamo di aver letto il contenuto di un registro di memoria e che esso sia $10101010_2 = 170_{10}$. Se volessimo porre a zero *solo* il bit 7 senza curarci dello stato degli altri, basterebbe scrivere nel registro in questione una qualunque parola contenente zero nel settimo bit, per esempio $00101010_2 = 42_{10}$. In questo modo è semplice, ma se abbiamo la necessità di azzerare la settima posizione (o qualunque altra) non conoscendo il contenuto del registro da modificare e non vogliamo alterare lo stato degli altri bit, ecco che le cose sembrano complicarsi poichè non è possibile utilizzare la procedura precedente. Infatti, come detto, per il puro azzeramento del bit 7 basterebbe scrivere in memoria un qualunque numero che contenga uno zero in settima posizione, ma in questo caso perderemmo il controllo dei rimanenti bit correndo il rischio di modificarli.

Se per esempio in IOR fosse contenuta la parola 10101010 (POKE 37136,170) con tutte le linee predisposte come uscite (POKE 37138,255), vedremmo accesi i LED 7,5,3,1 sul visualizzatore. Se al posto di ogni LED acceso ci fosse un relè che stesse controllando degli utilizzatori e volessimo disattivare *solo* il settimo, automaticamente da programma, non potremmo assolutamente inserire nel REGISTRO D'INGRESSO-USCITA (IOR) una parola casuale, anche se contenente uno zero in settima posizione, perchè rischieremmo di scollegare anche i relè interessati al controllo.

Ecco che ci viene in aiuto l'operazione AND che risolve tutti i nostri problemi. Essa è appunto utilizzata per azzerare un bit in una determinata posizione di una parola in memoria effettuando un procedimento detto di *mascheratura*.

N.B. — Questo modo di definire l'AND è esatto se si opera in LM ma non lo è del tutto se si lavora in BASIC. Per il momento non si toglie generalità al discorso se la si suppone corretta anche in quest'ultimo caso.

Consultando la tabella della verità dell'AND vediamo che il risultato è 1 solo se entrambi gli operandi sono 1: quindi, volendo azzerare il famigerato bit 7 della parola 10101010, basterà *mascherarla* con 01111111. Infatti

```
  10101010   AND
  01111111   =
-----------
  00101010
```

avendo eseguito l'AND bit per bit. Se volessimo azzerare i bit 7 e 5 basterebbe eseguire

```
  10101010   AND
  01011111   =
-----------
  00001010
```

cioè porre nella *maschera* uno zero in posizione 7 ed uno in posizione 5.

Concretizziamo come al solito tutto con un esempio pratico. Utilizziamo per ora il BASIC, quindi le parole binarie andranno codificate in decimale (vedi MC n° 25 pag. 99).

L'esempio consiste nel riportare in uscita sul visualizzatore la parola $11111111_2 = 255_{10}$ (LED tutti accesi) e modificare di volta in volta solo i bit desiderati.

Scrivete e fate girare il programma listato n°1: sul visualizzatore tutti i LED saranno accesi.

```
10  POKE 37138,255 :POKE 37136,255
20  INPUT M
30  A=37136 : B=PEEK(37136)
40  POKE A,B AND M
50  GOTO 20
```

Listato 1 - Mascherature con l'AND

Azzeriamo ora il bit 0: basterà, come detto, mascherare il contenuto di IOR, PE-

IOR	MASCHERA	DEC	LED
11111111	11111110	254	○○○○○○○●
11111110	11110001	241	○○○○●●●●
11110000	01111111	112	●○○○●●●●
01110000	10001111	143	●●●●●●●●

Figura 3 - La modifica di alcuni bit con l'operazione di mascheratura. I pallini neri indicano i LED spenti corrispondenti ad un bit posto a zero.

EK (37136), con 11111110, quindi rispondiamo alla richiesta di input con la codifica decimale di tale *maschera* che è 254. Le operazioni per azzerare successivamente i LED posizionati in 3-2-1, 7, 6-5-4 sono descritte in figura 3.

La subroutine che effettua la *mascheratura* del contenuto del REGISTRO D'INGRESSO-USCITA in LM è la seguente:

LDA #$ FF
carica il numero FF_{16} = 11111111_2 nell'accumulatore (A);

STA $ 9112
memorizza il contenuto di A nel registro 9112_{16} = 37138_{10} (DDR) ponendo tutte le linee come uscite;

LDA $ 9110
carica in A il contenuto del registro 9110_{16} = 37136_{10} (IOR);

AND "MASCHERA"
esegue l'AND di A con la maschera voluta e memorizza il risultato in A;

STA $ 9110
memorizza il contenuto di A nel registro 9110.

La funzione di LDA e STA è già stata descritta (MC n° 25 pag. 99). L'istruzione AND "MASCHERA" effettua in questo caso in modo immediato l'AND logico dell'accumulatore e della *maschera* ponendo il risultato in A. Le due istruzioni LDA che compaiono nel segmento indicato hanno entrambe la funzione di caricare l'accumulatore ma mentre la prima carica in esso un numero prestabilito, FF, la seconda vi carica il contenuto del registro di memoria 9110. I codici operativi di tali istruzioni non saranno quindi gli stessi poichè il primo è un *caricamento in modo immediato* ed il secondo è un *caricamento in modo assoluto*. La codifica del precedente segmento è indicata nel listato 2. Il programma è stato

LINEA	LOC$	MNEM	OPCODE	DEC
0001	0334	LDA	A9	169
0002	0335	FF	FF	255
0003	0336	STA	8D	141
0004	0337	91	12	18
0005	0338	12	91	145
0006	0339	LDA	AD	173
0007	033A	91	10	16
0008	033B	10	91	145
0009	033C	AND	29	41
000A	033D	MASCH.	MASCH.	MASCH.
000B	033E	STA	8D	141
000C	033F	91	10	16
000D	0340	10	91	145
000E	0341	RTS	60	96

Listato 2 - Mascheratura in linguaggio macchina con l'AND.

memorizzato a partire dalla locazione decimale 820 (0334_{16}) con la seguente routine:

```
10 FOR I = 0 TO 13: READ A
20 POKE 820+I,A: NEXT
30 DATA 169, 255,141,18,145,173,
        16,145,41, MASCHERA,141
        16,145,96
40 POKE 37136,255: NEW
```

Al posto di "MASCHERA" potete cominciare ad inserire 254 e poi modificarlo con 241, 112, 143 per ottenere lo stesso risultato del programma in BASIC precedentemente descritto. Le sequenze sono le seguenti:

1) POKE 829, 241 : SYS 820
2) POKE 829, 112 : SYS 820
3) POKE 829, 143 : SYS 820

Naturalmente lavorando in LM si effettuano controlli in tempo reale quindi questo è l'unico modo per ottenere in uscita dei *timing* sofisticati.

Se a qualcuno può interessare diremo che il programma fornito in LM esegue la sola modifica dei byte in 10 cicli macchina mentre la routine completa gira in 22. Ogni ciclo può essere assunto pari ad un microsecondo circa.

Esaminiamo ora l'operazione opposta a quella descritta fin'ora, e cioè come porre ad 1 un singolo bit. Ciò può essere effettuato utilizzando l'OR logico.

Dalla relativa tabella della verità si vede che si ha un 1 come risultato se *almeno uno* degli operandi è 1, quindi nella maschera che si utilizza dovranno essere posti degli 1 nella posizione in cui si desidera *alzare* il bit e degli 0 nelle altre posizioni. Per esempio

```
10000111  OR
01001000  =
--------
11001111
```

alza i bit 6 e 3 lasciando invariati gli altri. Il listato 3 esegue le mascherature indicate in figura 4 operando in maniera opposta rispetto al precedente definito per l'operazione AND.

IOR	M	DEC	LED
00000000	00000001	1	●●●●●●●○
00000001	00001110	14	●●●●○○○○
00001111	10000000	128	○●●●○○○○
10001111	01110000	112	○○○○○○○○

Figura 4 - Mascheratura con l'operazione OR. Anche qui i pallini neri indicano i led spenti sul visualizzatore.

Analogamente, il segmento in LM che svolge la funzione descritta è il seguente:

```
LDA  $ 9110
ORA #$ MASCHERA
STA  $ 9110
```

Lasciamo al lettore il compito di ampliarlo in analogia al caso precedente.

Un passo avanti

Abbiamo visto che, per modificare dei bit in determinate posizioni in un registro di memoria, si *maschera* opportunamente il suo contenuto, usando *maschere* diverse a seconda che si voglia porre *alto* o *basso* lo stato dei bit suddetti. Può capitare a volte di dover controllare il REGISTRO D'USCITA e quindi lo stato della user port per abilitare o disabilitare dei relé ad essa collegati usando una sola *maschera*. In tal modo si avrebbe per l'uscita un funzionamento tipo TOGGLE e cioè: *mascherando* una prima volta l'uscita va *alta, mascherando* una seconda volta va *bassa* e così via.

Un tale modo di operare è consentito dall'operazione logica OR ESCLUSIVO (EOR). Dalla tabella della verità si può facilmente vedere che l'uscita è 1 *se e solo se uno solo* degli operandi è 1.

Supponiamo infatti che in IOR sia contenuta la parola 00000000, cioè che le uscite siano tutte *basse*.

Mascheriamo il contenuto di tale registro con 10000000:

```
0000000  EOR
10000000   =
--------
10000000   ;
```

abbiamo quindi ottenuto come risultato quello di alzare il bit 7 di IOR che equivale ad accendere il settimo LED del visualizzatore. Effettuiam oncora la *mascheratura* del REGISTRO D'INGRESSO-USCITA (che ora contiene 10000000) con la *maschera* già utilizzata:

```
10000000  EOR
10000000   =
--------
00000000
```

il risultato è dunque quello di resettare il settimo bit (e LED). Per implementare tale algoritmo in LM non ci sono problemi in quanto esiste, tra le istruzioni del 6502, quella (EOR) che effettua l'OR ESCLUSIVO del contenuto dell'accumulatore con un dato specifico e memorizza il risultato in A. La routine che realizza tale funzione è la seguente:

```
LDA    $ 9110
EOR #  $ MASCHERA
STA    $ 9110
```

```
10 POKE 37138,255
20 INPUT M
30 A=37136 : B=PEEK(37136)
40 POKE A, B OR M
50 GOTO 20
```

Listato 3 - Mascherature con l'OR.

In BASIC non esiste un'operazione diretta di OR ESCLUSIVO, ma possiamo provare a ricavarla come combinazione di AND, OR,NOT. Se ben ricordate, nel n° 26 di MC avevamo detto che l'operazione EOR risultava dalla seguente combinazione:

$$A \oplus B = \overline{A}B + \overline{B}A$$

Ora, l'accostamento di due termini indica un'operazione di prodotto quindi AND, il + indica una somma quindi OR, il trattino indica una negazione quindi NOT. Possiamo allora riscrivere:

A EOR B = A AND NOT B OR B AND NOT A

Se A è il contenuto del registro d'uscita che di volta in volta viene modificato e B rappresenta la maschera relativa ai LED da

accendere o spegnere, scrivete e fate girare il seguente programma:

```
10 POKE 37138,255 : POKE 37136,0
20 INPUT B
30 A=PEEK(37136) : C=37136
40 POKE C,A AND NOT B OR B AND NOT A
50 GOTO 20
```

Introducete poi un numero da 0 a 255 e premete RETURN: vedrete accendere alcuni LED che si spegneranno introducendo lo stesso numero usato per accenderli (o premendo ancora RETURN).
L'algoritmo trovato quindi funziona.

Usiamo gli ingressi

Fin'ora abbiamo parlato delle operazioni relative al controllo delle uscite del VIA 6522 sulla user port. Essendo però spesso necessario rilevare dei dati dal mondo esterno per immagazzinarli nella macchina, esaminiamo brevemente il modo di procedere effettuando alcune semplici esperienze con la scheda VL2 del VicLab. Inserite allora tale scheda nell'apposito connettore della VL1. All'accensione, come già sapete, tutte le linee della porta utente sono configurate come ingressi quindi, per il momento, volendo verificare operazioni d'ingresso ad otto bit, non dovremo operare nessuna modifica sul REGISTRO DIREZIONE DATI del VIA. Come già sappiamo, in tale condizione ogni informazione sulle linee della *user port* viene trasferita nel REGISTRO D'INGRESSO - USCITA. Se leggiamo il contenuto di tale registro da BASIC, avremo sullo schermo un numero che è la codifica decimale della parola binaria che di volta in volta si forma sulla porta d'ingresso.

OPC	PAROLA	LED
A9	10101001	○●○●○○●○
FF	11111111	●●●●●●●●
8D	10001101	●○○○●○●●
12	00010010	○○○●○○●○
91	10010001	●○○●○○○●
60	01100000	○●●○○○○○

Figura 5 - Dopo aver avviato il programma indicato nell'articolo, introducete le parole di questa tabella sotto forma di un LED acceso in corrispondenza ad un 1 ed un LED spento in corrispondenza ad uno zero premendo RETURN dopo aver composto ogni parola.

Per verificare ciò, scrivete ed avviate il seguente programma:

```
10 PRINT PEEK(37136) : GOTO 10
```

e ponete gli interruttori in modo da accendere i LED corrispondenti ai bit 7,6,5,4,: in IOR sarà allora trasferita la parola 11110000 e sullo schermo sarà visualizzato il numero 240 che è appunto la codifica decimale di tale parola. Provate anche con altri numeri configurando in modo diverso gli interruttori: vi renderete facilmente conto che abbiamo realizzato un traduttore binario - decimale.

Prendiamo come ulteriore esempio la seguente subroutine che predispone le linee come uscite e, se viene abilitata dopo l'accensione, spegne gli otto LED di parola:

```
LDA # $ FF
STA   $ 9112
RTS
```

Noi potremo inserirla in macchina tramite gli interruttori in ingresso codificando i codici operativi del programma in binario.

Riportiamo in figura 5 lo svolgimento in esadecimale e binario di tale programma.

Inserite in macchina e fate girare la seguente procedura:

```
10 FOR I = 0 TO 5
20 GET A$ : IF A$ = "" THEN 20
30 POKE 820+I,PEEK(37136): PRINT I
40 NEXT
50 PRINT "FINE"
```

Poi introducete i codici operativi codificati in binario tramite gli interruttori accendendo un LED in corrispondenza ad un 1 e spegnendolo in corrispondenza ad uno 0; premete poi RETURN ogni volta che una parola sarà composta fino a che non viene visualizzata la scritta FINE.

Estraete quindi la VL2 e premete insieme i tasti STOP/RESTORE. Scrivete POKE 37136,0:SYS 820, digitate RETURN e gli otto LED rossi si spegneranno.

Pensate che fatica se dovessimo colloquiare direttamente con la macchina tramite 0 ed 1!

Una breve panoramica: il Flip-Flop

Terminiamo l'articolo con alcuni elementi di elettronica digitale per aumentare il nostro bagaglio di conoscenza in tal senso. Come al solito cercheremo di essere meno rigorosi possibile essendo il nostro scopo quello di fornire dei concetti fondamentali che, a conti fatti, stuzzichino la curiosità di chi legge o chiariscano qualche dubbio riguardo alle terminologie usate, spesso, quando si entra più in dettaglio nelle descrizioni di macchine complesse quali un computer. Per le trattazioni più ampie rimandiamo ai testi specializzati e per i suggerimenti attendiamo le vostre lettere.

L'argomento di questo mese è costituito dai FLIP-FLOP.

Un FLIP-FLOP è un dispositivo sulla cui uscita possono essere presenti due stati stabili (esclusivi) ed è capace di passare da uno all'altro sotto l'azione di un *segnale di comando*, rimanendo nella condizione in cui si è portato anche quando tale segnale viene rimosso. Ad esempio un interruttore della luce può essere considerato un semplicissimo FLIP-FLOP meccanico potendo assumere due condizioni (aperto o chiuso) se stimolato dalla pressione esercitata su di esso dal dito, che rappresenta il segnale di comando. Lo *stato* (condizione dell'uscita) rimane quando la causa modificante (in questo caso la pressione esercitata dal dito) viene rimossa: ciò differenzia un F.F. da una normale *porta logica* la cui uscita assume la configurazione indicata dalla tabella della verità e la mantiene fintanto che vengono mantenute le condizioni degli ingressi. Per segnale s'intende generalmente l'andamento nel tempo di un fenomeno fisico; per noi un *segnale di comando* è per esempio il passaggio di una tensione da 0 a +5 volt, *fronte di salita*, o da +5 a 0 volt, *fronte di discesa*. Il cambiamento dello stato delle uscite di un F.F. può avvenire in corrispondenza di uno di tali *fronti* o nel tempo in cui il segnale rimane ad un certo livello logico. Nel primo caso avremo dei dispositivi *edge triggered* e nel secondo dei dispositivi *level triggered*.

Forse tutte queste definizioni date a raffica hanno lasciato molti un po' perplessi,

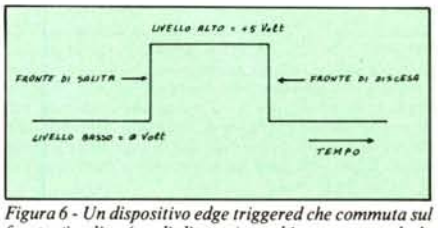


Figura 6 - Un dispositivo edge triggered che commuta sul fronte di salita (o di discesa) cambia stato quando la tensione relativa al segnale ha superato una certa soglia partendo da 0 volt (o da +5 volt). Con questo tipo di commutazione si ha una elevata velocità di trasferimento in quanto i dati vengono abilitati sull'ingresso e riportati in uscita quasi contemporaneamente.

ma se fate mente locale alla figura 6 vi accorgerete che in fondo si tratta di cose semplicissime.

Esaminiamo ora, per concretizzare il preludio, qualche tipo di F.F. e qualche applicazione. Quello più elementare è il F.F. SR (set-reset) che è indicato in figura 7a. Come potete osservare, esso può essere realizzato utilizzando due NAND a due ingressi che ormai conoscete bene (vedi MC n. 25). Le sue funzioni sono indicate dalla relativa tabella della verità. È bene osservare che le due uscite di un F.F. sono sempre in condizioni complementari l'una rispetto all'altra, cioè se una è in condizione logica 1, l'altra sarà in condizione 0. Pertanto basterà indicare lo stato di una di esse essendo ovvio quello dell'altra.

Sempre in riferimento alla figura 7a, con i terminali d'ingresso scollegati, l'uscita si trovi nella condizione $\overline{Q}=1$ (quindi $\overline{Q}=0$): se portiamo ad esempio l'ingresso S a livello *zero* (cioè lo colleghiamo a massa), lo stato dell'uscita si invertirà e *nessun'altra azione su S* provocherà dei cambiamenti. Per riportare il dispositivo nelle condizioni di partenza, dovremo agire sul terminale di *reset* R collegandolo a massa. Senza dilungarci sull'enorme importanza di tale comportamento, teniamo a farvi notare che in tal modo si riesce a mantenere uno stato, cioè a memorizzarlo, per il tempo desiderato ed è questo il punto su cui vogliamo farvi riflettere.

Prima di continuare diamo un'altra definizione. Chiameremo *segnale di clock* un segnale periodico che serve per sincronizzare degli eventi, cioè farli avvenire ad istanti prestabiliti. Esso può essere rappresentato da una tensione che varia nel tempo da un livello basso ad uno alto alternativamente, rimanendo a ciascun livello per un tempo dipendente dalla frequenza di

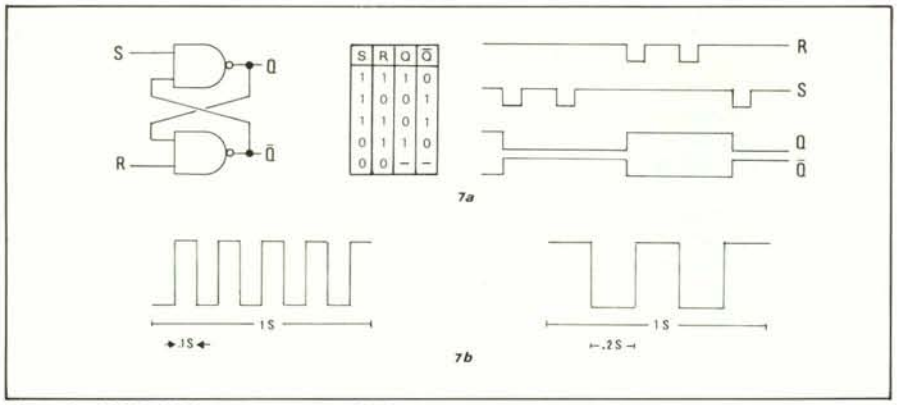


Figura 7 - Il F.F. SR è un esempio tipico di dispositivo asincrono in quanto il cambiamento dello stato segue immediatamente il comando di ingresso che lo ha provocato. Esaminando i diagrammi temporali si vede che tale F.F. cambia stato agendo alternativamente sull'uno e sull'altro terminale. Dalla tabella della verità si può notare che non è ammessa la condizione S=0 = R poichè in tal caso entrambe le uscite sarebbero allo stato 1. Si presti attenzione al fatto che per alcuni autori un latch è un semplice elemento di memoria come in questo caso mentre per altri è un dispositivo che memorizza dei dati quando riceve un impulso di clock. Nella parte b è riportato un esempio di segnale di clock. Il tempo in cui il segnale rimane a livello alto (o basso) dipende da quante volte esso varia in un secondo, cioè dalla sua frequenza.

tali cambiamenti (fig. 7b). Modificando opportunamente un F.F. SR, cioè aggiungendogli un ingresso di clock C, come è indicato in figura 8, si ottiene una versione *sincrona* di tale dispositivo, in cui cioè gli ingressi S ed R possono influenzare l'uscita con i loro cambiamenti solo durante il tempo in cui il segnale di clock è a livello alto e l'informazione in ingresso viene memorizzata sul fronte di discesa di tale segnale.

Anche in questo caso ci interessa farvi afferrare un concetto: che il dato viene *prima* preparato in ingresso e memorizzato *dopo*, quando il segnale di sincronizzazione lo permette.

Con altre modifiche che non stiamo qui a indicare si possono ottenere nuovi tipi di F.F. per esempio di tipo JK, di tipo D o T (toggle). Tramite questi componenti è possibile realizzare contatori, divisori, sommatori o shift-register. Non vogliamo, per ovvie ragioni, soffermarci oltre sull'argomento ma daremo qualche esempio per ribadire l'importanza di tali dispositivi.

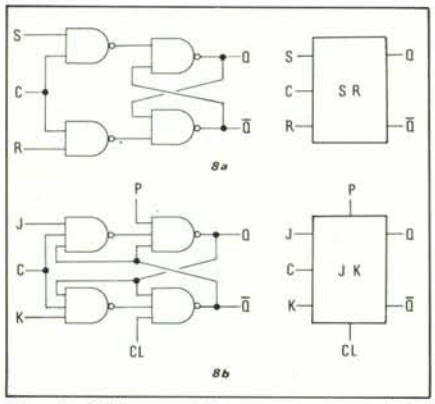


Figura 8 - Nella parte a) è riportato un esempio di F.F. sincronizzato. Gli ingressi S e R influenzano lo stato solo quando il segnale di clock è a livello alto. Il dato viene memorizzato sul fronte di discesa di tale segnale. In ingresso non è ammessa la combinazione S = 1 = R. Nella parte b) è rappresentato un F.F. JK ed il suo simbolo. Il funzionamento è il seguente: 1) se J = 0 = K il F.F. non commuta mai; 2) se J = 1 = K il F.F. commuta su ogni fronte del segnale di clock (toggle); 3) se, p.es., J = 1 il F.F. commuta sul primo fronte di salita e poi non cambia più stato. Per far avvenire il cambiamento bisogna porre K = 1 (e viceversa). Preset (P) e clear (CL) sono ingressi asincroni cioè provocano un cambiamento di stato ogni qual volta si agisce su di essi indipendentemente dal segnale di clock. Se P = 0 e CL = 1 allora Q si porta a livello basso; se P = 1 e CL = 0, Q si porta a livello alto.

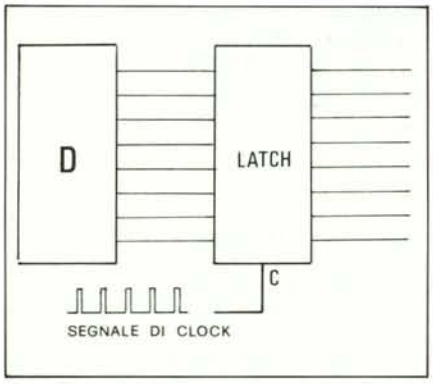


Figura 9 - In uno schema del genere si ha il trasferimento dell'informazione dal dispositivo all'uscita in corrispondenza ad ogni impulso di clock.

Come ulteriore esempio consideriamo, senza entrare molto in dettaglio, una CPU che divida il BUS dei dati con la parte bassa del byte degli indirizzi mentre il rimanente byte alto è indipendente.

Questo è un classico esempio di BUS in multiplex (*vedi figura 10*). Ad un certo istante il byte di ordine basso è posto sul BUS comune dei dati-indirizzi. Viene poi generato dalla CPU un impulso sul terminale di clock del latch ed in virtù di quest'ultimo il byte basso dell'indirizzo viene memorizzato. Una volta memorizzato nel latch, non è più necessario che esso venga trattenuto sulle linee del microprocessore e può quindi essere inizializzato il ciclo di trasferimento del dato vero e proprio che non influenzerà più l'indirizzo non essendo generato durante il tempo di trasferimento nessun segnale di sincronizzazione sul terminale di clock del latch.

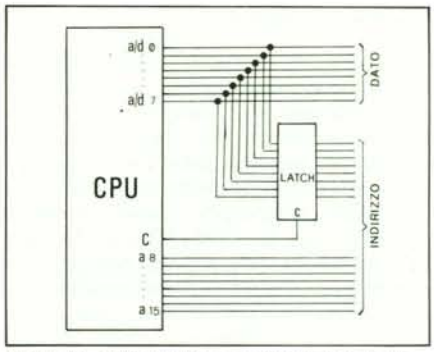


Figura 10 - BUS in multiplex realizzato con un latch.

Per finire

Ritorniamo per un attimo al VIC 20 e al suo VIA.

Abbiamo in linea di massima terminato la trattazione delle operazioni di base per il controllo delle linee direttamente collegate al REGISTRO D'INGRESSO-USCITA del 6522. Vogliamo comunque accennare ad un fatto molto importante e cioè che se in ingresso l'informazione è fluttuante, cioè variabile nel tempo (ad esempio in uscita da un convertitore analogico-digitale), essa può essere bloccata (latched) ad intervalli regolari e trasferita in memoria senza aggiunta di circuiteria ausiliaria grazie alla grande versatilità del 6522 il quale permette agli ingressi di operare in modo *latch*. In altre parole, come avrete potuto notare dalla mappa di memoria del VIA pubblicata su MC n. 25 pag. 90, esiste un registro detto REGISTRO DI CONTROLLO AUSILIARIO (ACR) e configurando in modo opportuno alcuni suoi bit, se un programma sta leggendo dei dati posti sulla *user port*, ogni qual volta noi provochiamo una transizione di livello su CB1 facendo passare tale linea da *bassa* ad *alta* (fronte di salita), viene *catturato* il dato presente in ingresso e trasferito nel REGISTRO D'INGRESSO-USCITA. Vari tipi di segnale possono inoltre essere generati sulla linea CB2 che agisce, insieme alla CB1, sotto il controllo diretto di un altro importante registro detto REGISTRO DI CONTROLLO DI PERIFERICA (PCR) per permettere la comunicazione con le varie unità esterne che devono lavorare controllate dal computer.

Di queste cose avremo comunque modo di parlare in seguito.

Alla prossima volta. MC

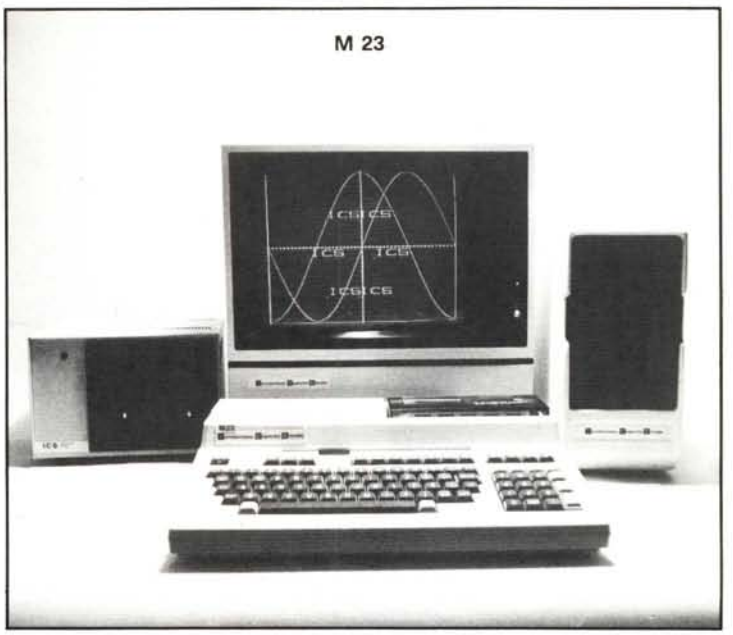
INTERNATIONAL COMPUTER SYSTEMS

Uffici di Roma: Via della Balduina, 85-89 – Tel. 34.81.85 - 34.92.760-660 – Telex 611091 CRMC Stabilimento: Via Nettunense, 49 – 00042 Anzio – Tel. 98.46.206

In Italia come in tutto il mondo la gamma dei nostri elaboratori sta ricevendo l'adesione degli esperti di informatica e degli utilizzatori. Per ragioni che sono le più valide: rigore tecnologico, fabbricazione professionale e sforzo costante di creare degli autentici sistemi di informatica al costo più basso. La International Computer Systems garantisce la distribuzione dei prodotti migliori direttamente dagli stabilimenti produttivi situati in Giappone, Irlanda, Italia.

M23 mark III – M23 mark V

Piccolo. Leggero. Potente.
Si impara a programmarlo in tre giorni!



Configurazioni a scelta con floppy da 5 o da 8 pollici monitor a fosfori verdi o a colori (RGB) da 14 pollici.
Scheda grafica a colori optionale.

Unità centrale
Un microprocessore ZILOG Z 80A con un clock a 4 MHz gestisce le risorse del sistema.
Un 2° micro APU effettua tutti i calcoli matematici.
Una memoria RAM da 128 Kbytes è a disposizione utente.
Due interfacce seriali RS232 programmabili e un'interfaccia parallela permettono il collegamento con l'esterno.
Questo insieme dà all'unità centrale la potenza richiesta per una larga gamma di applicazioni.

Unità minifloppy
Due minifloppy da 5'' (328 Kbytes ciascuno), semplice faccia, doppia densità, gestiti da un'interfaccia interna DMA (accesso diretto memoria).

Unità floppy 8''
Due Driver doppia faccia , doppia densità (1,1 MB ciascuno), con possibilità di formattazione in tutti i formati IBM.

Tastiera
Un blocco alfanumerico standard con maiuscole e minuscole.
Un blocco numerico separato con i comandi del cursore.
Un blocco di 14 funzioni programmabili.
Le sue numerose funzioni permettono una grande flessibilità di utilizzo.

Schermo
25 righe per 80 colonne maiuscole e minuscole in visione normale o ''negativa''.
32 caratteri semigrafici permettono la costruzione di tabelle o di grafici.

SYSTEM SOFTWARE

● Relocatable assembler ● Editor ● Debugger ● Relocatable loader ● Library file editor

● Subroutines in Assembler possono essere richiamate all'interno di programmi in BASIC o in Fortran ● EBASIC – Interprete esteso occupa circa 32 Kbytes ● CBASIC – Compilatore compatibile con Ebasic consente di aumentare di 5/6 volte la velocità di esecuzione ● MBASIC – A doppia precisione (13 cifre) per calcoli tecnici e matriciali ● TBASIC – Per trasmissione dati e collegamento con altri computers ● FORTRAN IV – Per calcoli tecnico-scientifici ● COBOL – Corrispondente a livello ANSI 74 ● UCSD PASCAL ● L'SGL è un linguaggio grafico che permette, eventualmente anche con monitor a colori, di eseguire disegni estremamente complessi utilizzando la libreria BASIC con delle subroutines per le funzioni più comuni.

Vasta scelta di software applicativo gestionale-scientifico

PIPS, un linguaggio facile da imparare, sfrutta al massimo le capacità della macchina

Il PIPS, software unico, sviluppato per uso gestionale, è molto più vicino alla mente umana dell'Assembler, del Fortran, del Basic. Il PIPS permette a tutti di usare un potente computer con facilità. Il PIPS lavora utilizzando oltre 100 comandi. La gestione dei dati avviene tramite la semplice selezione di questi comandi. Per ricercare dei dati si imposta il comando CS. Per sortare si imposta SORT. Per funzioni grafiche si imposta GR. E così via. Vari programmi e funzioni possono essere ottenute a seconda dell'ordine con cui si selezionano i comandi. Il PIPS elimina la necessità di programmi specialistici. Alcuni tipi di lavoro richiedono soltanto di digitare i comandi nel loro ordine, per ottenere i risultati richiesti!

M 243 – M 343 Una famiglia di micro da 8 e da 16 bit multiutente con multiprogrammazione


M 343 mark V

L'**M 243** e l'**M 343** sono il culmine di anni di esperienza combinati con la più sofisticata tecnologia. Sono microcomputers completamente nuovi che si adattano perfettamente ai più disparati tipi di applicazioni, Offrono possibilità di ampliamento in memoria centrale con schede; in memoria di massa con dischi floppy da 5'' e da 8'' e dischi rigidi Winchester. Oltre ad avere inserite interfacce di qualsiasi tipo e a poter essere utilizzati come terminali intelligenti di computers più potenti, sono dotati di uno schermo completamente grafico ad altissima definizione anche a colori e permettono la gestione di più posti dilavoro in multi-programmazione.

Unità Centrale
Un microprocessore a 8 bit Z80A gestisce le risorse del sistema nel M 243.
Un microprocessore a 16 bit 8086 è invece utilizzato nel modello M 343.
Un 2° processore logico effettua tutte le operazioni logiche sui numeri fino a 32 bit in virgole flottanti.
Un counter/timer programmabile da software controlla la successione delle operazioni.
Un orologio in tempo reale, con batteria tampone, fornisce la data e l'ora e permette di avviare, tra l'altro, dei programmatori ad ore prestabilire.
Una memoria RAM da 192 Kbytes a 1 Mbytes è a disposizione utente. Tale memoria consente la presenza di più posti lavorocompleti in multiprogrammazione.
Quattro canali seriali RS232 programmabili da 50 a 19.200 Baud e un canale parallelo permettono il collegamento con l'esterno.

M5 – Home Computer
Il micro più piccolo della nostra famiglia

Si collega al televisore a colori di casa e ad un registratore a cassette
Unità centrale
Z 80A - RAM 20K RAM + 16 ROM espandibile con cassetta fino ad altri 32 K.
Uscita per stampante parallela.
Uscita per TV color.
Uscita per monitor e altoparlante.
Sintetizzatore musicale, generatore di rumori bianchi, vera grafica 16 colori in configurazione standard.
Optionali n. 2 Joypads per video game.
Tastiera con 52 tasti a 4 funzioni (maiuscoli, minuscoli, istruzioni basic e semigrafica).
N° 1 cassetta elettronica con basic, 20 video games su nastro nelle forniture standard

INSTALLAZIONE IN TUTTA ITALIA CON LE SEGUENTI PROCEDURE

- Contabilità generale magazzino fatturazione.
- Contabilità generale e semplificata per commercialisti.
- Contabilità generale a booking per Agenzie di Viaggi.
- Trattamento testi e mailing list merge universale.
- Contabilità finanziaria per scuole ed enti pubblici.
- Paghe e stipendi per scuole.
- Gestione magazzini componenti o ricambi.
- Gestione biblioteche.
- Gestione iscritti ordini professionali.
- Calcolo strutture per zone sismiche.
- Gestione laboratori di analisi cliniche.

STAMPANTI 80-132-220 COLONNE ANCHE GRAFICHE A MATRICE DI 9 AGHI ED A MARGHERITA.

PLOTTER A 8 COLORI.
CONVERTITORI ANALOGICI/DIGITALI E D/A.

Cercansi distributori per zone libere

Qual è il Personal computer a 16 bit più venduto in Europa?

Ormai abituati alla risonanza dei grossi nomi, forse non ci viene subito in mente. Eppure, il Personal computer a 16 bit più venduto in Europa è Victor, di Harden Italia.

Saranno le sue incredibili capacità grafiche e di elaborazione, o le sue eccellenti possibilità di comunicazione e dialogo con altri computers, o la sua biblioteca di programmi.
Saranno l'eccellente Harden-Text per la videoscrittura o il versatilissimo Harden-Azienda per la gestione, entrambi interamente in italiano.

O sarà magari la capillarità del suo servizio assistenza e vendita (a tutt'oggi sul solo territorio italiano conta ben 150 dealers).

Resta il fatto che il Personal computer a 16 bit più venduto in Europa è ancora Victor.

Di Harden Italia.

HARDEN ITALIA

HARDEN ITALIA S.p.A.
Centro Direzionale Milano Fiori
Strada 7 - Palazzo T 3
20088 ROZZANO - Tel. (02) 8243741 r.a.