Siamo giunti in dirittura d'arrivo per quanto riguarda la descrizione del controllo delle linee d'I/O di un computer nel collegamento con il mondo esterno. Oggi daremo i ragguagli necessari sui rimanenti registri del 6522 e come applicazione pratica forniremo la prossima volta gli schemi e i disegni dello stampato per la costruzione di un semplice e preciso combinatore telefonico interamente comandato dal computer. Prima di tutto ciò vogliamo però affrontare, seguendo le linee essenziali, un argomento tanto interessante quanto importante per problemi d'I/O: l'interrupt.

VIC da zero

di Tommaso Pantuso

Quinta Parte

Interrupt

Nel problema della comunicazione di una CPU (unità centrale di processo), per esempio un microprocessore nei microcomputer, con le varie periferiche, il problema dell'identificazione e del servizio ai vari dispositivi (scheduling) viene risolto con alcune tecniche di base (tra cui l'interrupt) che andiamo ad elencare.

La prima è quella del *polling* (registrazione). Con tale metodo la CPU interroga ciclicamente le varie periferiche collegate al BUS per verificare se una di esse ha richiesto servizio. Se la risposta è affermativa esso viene immediatamente concesso, cioè si rimanda in esecuzione il programma che permette alla periferica in oggetto di svolgere le proprie funzioni; se è negativa, essa viene ignorata e si passa all'interrogazione della seguente.

Tale tecnica è molto semplice e viene esplicata comodamente in quanto non è richiesto nessun supporto hardware potendo essere risolta completamente via software. Ha però notevoli svantaggi. La maggior parte del tempo della CPU è infatti sprecato per l'interrogazione di dispositivi che in quel momento possono anche non aver richiesto servizio ed inoltre, se una periferica richiede tale servizio subito dopo essere stata interrogata, lo riceve quando ritorna il proprio turno di colloquio. Si capisce come in questo caso essa venga servita in ritardo rispetto alla richiesta ed oltre allo spreco di tempo si può correre, in alcuni casi, il pericolo di perdere dei dati.

"Comunque — dice Rodnay Zaks — la registrazione viene usata estensivamente quando un processore non ha nient'altro di meglio da fare" e naturalmente quando si vuole mantenere semplice l'architettura del sistema.

Più efficace è un'altra tecnica che non spreca tempo a "dar conto" a dispositivi che non hanno effettuato richieste ma che vengono presi in considerazione solo sulla base di una eventuale *chiamata*: è questa la tecnica dell'*interrupt*. I dispositivi periferici sono ora connessi ad una *linea di interrupt* collegata con il processore (fig. 1a).

Quando il dispositivo richiede servizio, esso invia un impulso od un livello alla CPU la quale, dopo aver terminato di eseguire l'istruzione in corso, si incarica della periferica che ha effettuato la richiesta andando ad eseguire una routine ad essa asservita, detta *routine di servizio*, che è delegata all'elaborazione dell'interruzione.

Prima di far questo, cioè prima della diramazione , il processore deve conservare nello *stack* il program counter o contatore di programma il quale servirà ad identificare il punto in cui è stato interrotto il programma principale e da cui bisognerà ricominciare al ritorno dall'interruzione. Il 6502 preserverà in tale area anche il *registro di stato* (P) per evitare che questo venga alterato dalle routine di servizio. Per chi non lo sapesse, lo *stack* (pila) è un'area nella quale vengono immagazzinate delle informazioni fondamentali quando il programma principale dirama verso una subroutine od una routine di servizio per la gestione di un interrupt (fig. 2).

Tale operazione avviene, per alcune di tali informazioni, in modo automatico (contenuto del contatore di programma e registro di stato) e per altre in modo comandato dal programmatore tramite le apposite istruzioni del 6502. Lo *stack* è una struttura LIFO cioè, Last-In, First-Out, in quanto l'ultimo dato immagazzinato in esso è sempre il primo ad essere *ripescato*.

Ritornando al nostro *interrupt*, esso possiede lo svantaggio di richiedere che la CPU abbia almeno una linea, detta appunto di interrupt, ed inoltre richiede un'appo-

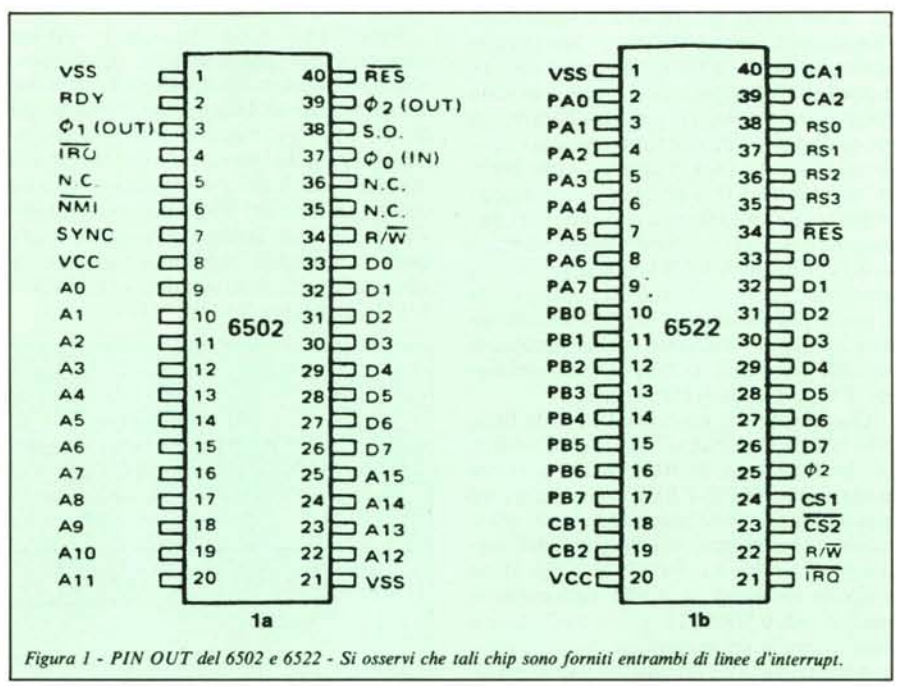

Figura 1 - PIN OUT del 6502 e 6522 - Si osservi che tali chip sono forniti entrambi di linee d'interrupt.

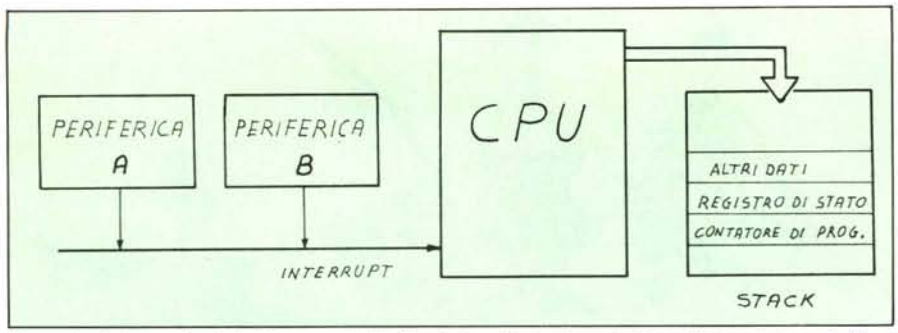

Figura 2 - Salvaguardia di dati importanti nello Stack dopo l'invio di una richiesta d'interrupt da parte di una periferica. Al termine della routine di manipolazione dell'interrupt, il primo elemento ad essere riletto è l'ultimo introdotto (struttura LIFO).

sita circuiteria la quale identifichi il dispositivo che ha trasmesso la richiesta e che asserva ad esso la CPU. A proposito di tale fatto, esistono due metodi fondamentali per l'indentificazione della periferica richiedente: uno software e uno hardware. Il primo, quello software, è quello precedentemente descritto di *registrazione* o *polling*: quando la CPU riceve richiesta di interruzione, esegue il *polling* dei vari dispositivi per intercettare quello interessato ed effettuare la relativa routine di *manipolazione dell'interrupt*. Il secondo è un metodo tutto hardware tramite il quale una adeguata circuiteria esterna fornisce *immediatamente* l'indirizzo della periferica che ha richiesto servizio. Esso è utilizzato quando si ha necessità di risposte più che rapide; in caso contrario (ed anche nella maggior parte dei casi) si agisce utilizzando il *polling*. Visto che siamo in argomento, accenniamo un attimo alle linee di *interrupt* del 6502. Esse sono di due tipi: a) NMI cioè *not maskable interrupt* (interruzione non mascherabile); b) IRQ cioè *interrupt request* (richiesta d'interruzione). Tali linee vengono attivate se mandate a livello basso per alcuni microsecondi (almeno). Esse sono sostanzialmente differenti. Infatti la richiesta d'interruzione sulla linea IRQ può (o non) venire ignorata dalla CPU: basta porre alto (o basso) un flag apposito del registro di stato detto *interrupt flag* (I) per mascherare, nel senso di nascondere, il fatto che si sia verificata richiesta d'interruzione su tale linea. In tal caso (I = 1) il programma prosegue come se IRQ non fosse avvenuto. Un'operazione del genere non può essere effettuata nei confronti di NMI la quale ha priorità assoluta ed i dispositivi ad essa collegati, in seguito alla richiesta, vengono serviti immediatamente. Vediamo brevemente cosa succede quando il 65502 riceve una richiesta d'*interrupt* su NMI ed IRQ.

Quando arriva una richiesta sulla linea NMI, la CPU effettua il salto ad un indirizzo che è registrato in ROM nelle locazioni esadecimali FFFA-FFFB e che punta ad alcune routine fondamentali le quali effettuano il trasferimento nello *stack* del contatore di programma (PC) ecc. Poi viene eseguito un salto in RAM agli indirizzi esadecimali 0318-0319 (per il VIC) dove è memorizzato un puntatore (byte basso e byte alto) che invia ad una routine di manipolazione. L'indirizzo in RAM può essere naturalmente modificato dal programmatore per i propri scopi (per esempio cedere il controllo ad un programma da lui ideato). La stessa cosa accade per una richiesta sulla linea IRQ solo che in tal caso la CPU salta agli indirizzi FFFE-FFFF e poi a 0314-0315 (gli ultimi sono sempre riferiti al VIC). Esiste inoltre tra le istruzioni del 6502, BRK (BReaK) che opera in modo identico ad IRQ ed è utilizzata per generare un *interrupt via software* programmato. Se si sia verificato un BRK od un IRQ può essere discriminato andando a controllare il flag B del registro di stato: esso sarà 1 se si è verificato un BRK o 0 se si è verificato un IRQ hardware. L'uso di BRK è molto delicato per via di alcune operazioni da compiere sul program counter e quindi non è consigliato ai neo-programmatori in assembler.

Aggiungiamo come ultima cosa, a titolo informativo, che lo *stack* ha importanza vitale nella gestione di interrupt multipli ma ulteriori spiegazioni esulano dai nostri scopi.

I registri del VIA

Avete ormai acquisito tutte le nozioni necessarie per poter seguire un discorso più compatto avendo le basi sufficienti sull'interpretazione delle terminologie che andremo ad usare, e quindi non ci soffermeremo oltre sulla spiegazione di concetti che riteniamo acquisiti. Sottolineiamo il fatto che, all'interno del VIC, sono presenti due VIA ma che noi faremo riferimento solo a quello accessibile tramite la *user port* ed allocato tra gli indirizzi decimali 37136 e 37151 (esadecimali 9110-911F).

Il 6522 possiede 16 registri e nel VIC 20 si può accedere ad essi direttamente trattandoli come semplici indirizzi di memoria. Su quelli riguardanti la sezione PIO non c'è più molto da dire, quindi passeremo subito alla descrizione degli altri. Essi sono: 6 registri per l'uso dei *timer* interni al VIA; uno shift register (SR); un registro di controllo ausiliario (ACR); un registro di controllo delle periferiche (PCR); un registro di stato o dei flag d'interrupt (IFR); un registro di abilitazione degli interrupt (IER). Premettiamo che i dati che stiamo per fornire potranno essere compresi meglio con la pratica: è questa che raccomandiamo al lettore aiutandosi anche con il VICLAB. Inoltre per motivi di spazio non potremo dilungarci molto su esempi pratici che lasciamo alla vostra pazienza... E chissà che non venga fuori qualche bella applicazione!

I registri IER ed IFR

Sono due registri che effettuano il controllo sugli interrupt. Sono posizionati nelle locazioni decimali 37149 (IFR) e 37150 (IER). Essi sono interdipendenti, più precisamente IER influisce sul comportamento di IFR così come avveniva per DDR e IOR.

Il registro dei flag d'interrupt è un registro d'ingresso e quello di abilitazione di tali flag è un registro d'uscita.

Per un esame più accurato si confronti la figura 3. Da essa si può notare che ad ogni bit di IER ne corrisponde uno di IFR. Ogni bit (da 0 a 6) di IER posto ad 1 farà sì che un determinato evento alzi il corrispondente flag in IFR; viceversa se uno di quei bit è a 0, il corrispondente flag non sarà alzato e l'interrupt verrà ignorato. Quindi, se le condizioni di IER lo permettono, ciascun flag di IFR, corrispondente ognuno alla posizione di un bit, sarà messo ad 1 dalle condizioni descritte in tabella 1.

IFR	Il flag è alzato da:
bit 0	opportuno segnale su CA2
bit 1	opportuno segnale su CA1
bit 2	operazione su SR (fine shift)
bit 3	opportuno segnale su CB2
bit 4	opportuno segnale su CB1
bit 5	scaricamento del TIMER 2
bit 6	scaricamento del TIMER 1

Tabella 1 - Operazioni che alzano i flag di IFR.

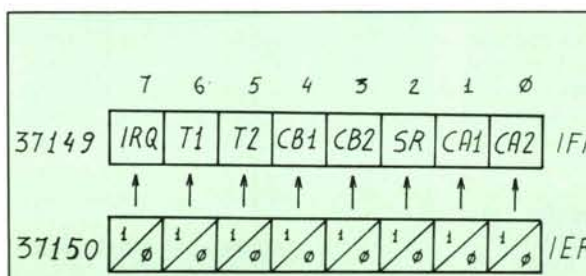

Figura 3 - INTERRUPT REGISTER ENABLE (IER) ed INTERRUPT FLAG REGISTER (IFR). IFR è controllato da IER nel senso che ogni bit posto ad 1 in quest'ultimo registro fa sì che un eventuale interrupt venga segnalato nel registro di stato IFR. Se lo stesso bit è posto a zero, l'interrupt viene ignorato dalla CPU. Per azzerare IFR si può procedere con le operazioni della tabella 2 (pag. 92) o come descritto in figura 4 (pag. 92).

made in italy
personal kid

I PIÙ DEL PERSONAL KID

- GARANZIA 1 ANNO
- PAD NUMERICO ESTESO
- CARATTERI MINUSCOLI
- CONTROLLO DIRETTO DEL CURSORE
- TASTI FUNZIONALI
- REPEAT AUTOMATICO
- EPROM UTENTE

CPU 6502 RAM 48Kb espandibile a 64 Kb
ROM 14 Kb - BASIC residente
Compatibile APPLE (marchio reg. APPLE Computer)

I PREZZI DEL PERSONAL KID

IVA esclusa garanzia 1 anno

KID 2010 (48 Kb. tastiera incorporata) L. 1.210.000
KID 2020S (48 Kb. monitor 12" incorp., tast. separata) L. 1.500.000
KID 2030S (48 Kb. monitor 12" e drive 5" incorp., tast. sep.) L. 2.300.000
KID 2040S (48 Kb. monitor 10" e due drive 5" incorp., tast. sep.) L. 3.000.000

PER PERIFERICHE E INTERFACCE
RICHIEDERE IL LISTINO COMPLETO

CERCASI CONCESSIONARI

SIPREL s.r.l.

Via Di Vittorio, 82 Tel. (071) 8046305
60020 CANDIA - ANCONA

Spett. SIPREL
gradirei ricevere:
☐ Ind. Concess. di Zona
☐ Documentazione ☐ Listino Prezzi

Nome _____
Via _____
Cap. _____ Città _____
MC

PERSONAL KID È PRODOTTO E GARANTITO DALLA SIPREL S.r.l.

Nelle indicazioni di tale tabella, per "opportuno segnale" intendiamo un fronte di salita, di discesa, un impulso od una variazione di livello. La natura del segnale abilitante è definita dal contenuto di un altro registro (PCR) che analizzeremo tra breve. Chiameremo d'ora in poi tale segnale di comando *transizione attiva*.

I flag precedenti possono essere riposti a 0 dalle condizioni elencate nella tabella 2.

IFR	Il flag è abbassato da:
bit 0	lettura o scrittura in IOR A
bit 1	lettura o scrittura in IOR A
bit 2	lettura o scrittura in SR
bit 3	lettura o scrittura in IOR B
bit 4	lettura o scrittura in IOR B
bit 5	lettura del byte basso o scrittura nel byte alto di T1
bit 6	lettura del byte basso o scrittura nel byte alto di T2

Tabella 2 - Operazioni per abbassare i flag di IFR.

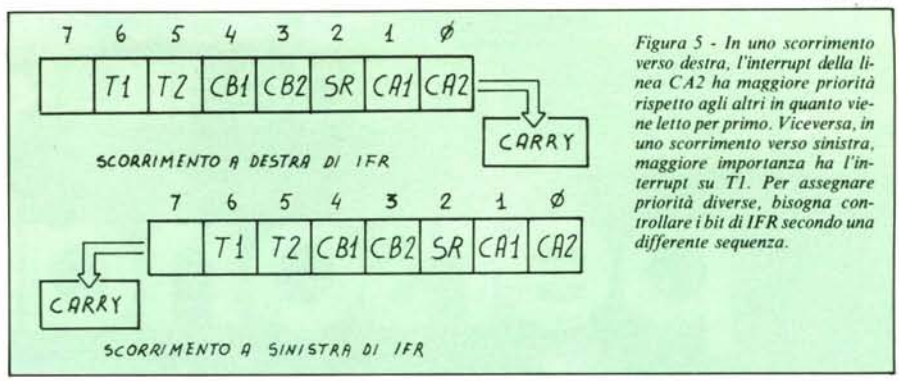

Figura 5 - In uno scorrimento verso destra, l'interrupt della linea CA2 ha maggiore priorità rispetto agli altri in quanto viene letto per primo. Viceversa, in uno scorrimento verso sinistra, maggiore importanza ha l'interrupt su T1. Per assegnare priorità diverse, bisogna controllare i bit di IFR secondo una differente sequenza.

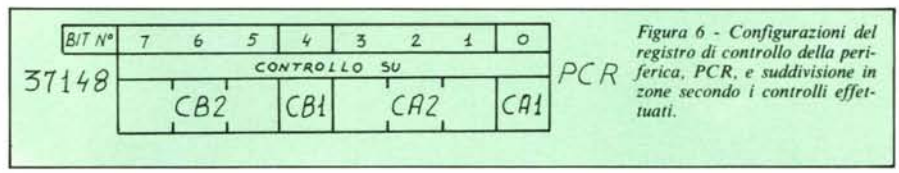

Figura 6 - Configurazioni del registro di controllo della periferica, PCR, e suddivisione in zone secondo i controlli effettuati.

Quindi, se per esempio una transizione attiva su CB2 pone ad 1 il bit 3 di IFR, per ricondurlo a 0, ripristinando così le condizioni iniziali, basterà leggere il contenuto di IOR B (o scrivere in esso). Questi *automatismi di ripristino* sono di estrema comodità nei problemi di interscambio dati.

Esistono però condizioni meno automatiche di azzeramento dei flag d'interrupt effettuati tramite la CPU potendo essa leggere o scrivere nei registri citati. Notevole importanza assume a questo punto il contenuto del bit 7 di IER. Se infatti tale bit è posto a 0, una parola scritta in IER e contenente degli *uno* in qualunque posizione da 0 a 6 azzererà automaticamente i bit corrispondenti di tale registro; eventuali *zero* nella parola che scriviamo in IER non avranno *nessuna* influenza sui flag.

Viceversa il millantato bit 7 è posto ad 1, ogni *uno* scritto in IER porrà alto un bit di abilitazione. Per fare un esempio pratico, se scriviamo in IER la parola 01111100 saranno azzerati i bit da 2 a 6; scrivendo poi in esso 10000011 saranno posti ad *uno* i bit 0 ed 1 (quindi abilitati gli interrupt di CA1 e CA2); per maggiori chiarimenti si consulti la figura 4.

Terminiamo dicendo che quando il registro dei flag rileva un *interrupt*, viene attivata la linea IRQ del VIA: il bit 7 di IFR segnala, se è ad 1, che si è verificato un interrupt su IRQ. Inoltre per rilevare ed identificare gli interrupt si fa scorrere il contenuto di IFR a destra od a sinistra e si controlla dopo ogni scorrimento il contenuto del flag C (carry) del registro di stato P del microprocessore. Questa tecnica assegna diversa priorità agli interrupt a seconda che lo scorrimento sia effettuato a destra od a sinistra: per esempio in uno scorrimento a sinistra un interrupt su T1 ha priorità più elevata rispetto agli altri perché lo stato del flag corrispondente in IFR viene rilevato per primo (cadendo per primo nel carry); si confronti la figura 5.

Il registro PCR

È situato nella locazione decimale 37148. Esso controlla, con le varie configurazioni dei suoi bit, le quattro linee di controllo, due per *porta*, del VIA denominate CA1, CA2, CB1, CB2. Esse sono fondamentali per effettuare procedure di *handshake* cioè *protocolli* ad uso e consumo dello scambio di informazioni tra unità centrale ed unità periferiche. Vi ricordiamo che sulla *user port* del VIC sono presenti solo CB1 e CB2 essendo CA1 e CA2 utilizzate rispettivamente dalla macchina per la gestione del RESTORE e per alcuni controlli sul motore del registratore a cassette. Tali linee possono inoltre essere usate come I/O seriale o come linee d'interrupt. Esaminiamo più in dettaglio tale registro facendo riferimento alla figura 6. Da quest'ultima si può vedere che i bit 7-6-5

N°	PCR7-PCR6-PCR5	Operazione
1)	000	Il flag d'interrupt di CB2 (F3) viene posto ad 1 quando tale linea passa da alta a bassa (fronte di discesa); F3 viene azzerato leggendo o scrivendo in IOR B o scrivendo 1 nella posizione F3
2)	001	Tutto come il punto 1) solo che F3 non viene azzerato da un'operazione di lettura o scrittura in IOR B
3)	010	Tutto come il punto 1) con l'unica differenza che la transizione attiva su CB2 deve essere un fronte di salita.
4)	011	Tutto come il punto 3) tranne che una operazione di lettura o scrittura in IOR B non azzera F3
5)	100	Scrivendo in IOR B, CB2 va bassa; essa ritorna a livello alto con una transizione attiva su CB1
6)	101	Scrivendo in IOR B, CB2 va bassa per un ciclo
7)	110	Abbassa CB2 e la mantiene in tale stato
8)	111	Mantiene alta CB2

Tabella 3 - Influenza dei bit 7-6-5 di PCR sulle linee di controllo CB2 e CB1.

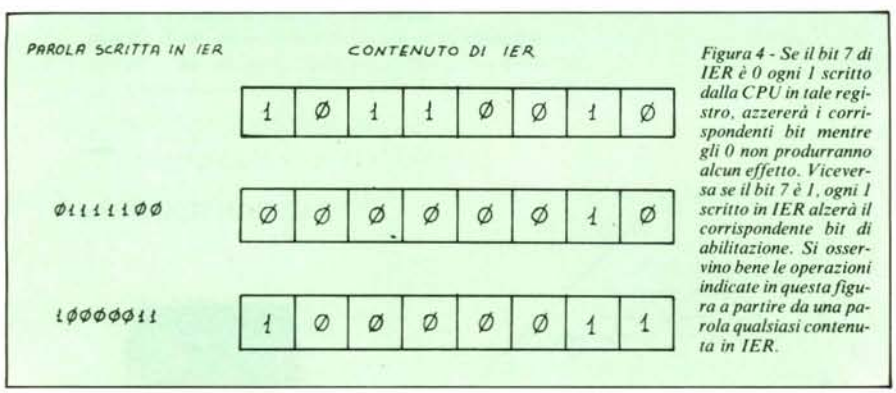

Figura 4 - Se il bit 7 di IER è 0 ogni 1 scritto dalla CPU in tale registro, azzererà i corrispondenti bit mentre gli 0 non produrranno alcun effetto. Viceversa se il bit 7 è 1, ogni 1 scritto in IER alzerà il corrispondente bit di abilitazione. Si osservino bene le operazioni indicate in questa figura a partire da una parola qualsiasi contenuta in IER.

controllano le operazioni su CB2; il bit 4 quelle su CB1; i bit 3-2-1 quelle su CA2; il bit 0 quelle su CA1.

Nelle tabella 3 sono descritte le operazioni a cui rispondono le linee CB2 e CB1 secondo il modo in cui sono configurati rispettivamente i bit 7-6-5 e 4 di PCR ritenendo simmetriche quelle a cui sono soggette CA2 e CA1 con la manipolazione dei bit 3-2-1 e 0. In seguito daremo qualche dimostrazione usando il VICLAB.

Per il *bit 4* valgono invece le seguenti considerazioni: *se esso è 0 il flag di CB1 (F4) è posto ad 1 da una transizione alto-basso su tale linea; se esso è ad 1, F4 è posto*

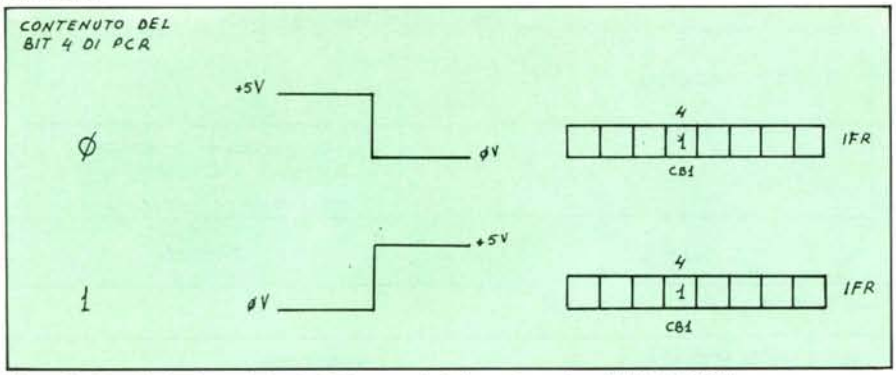

Figura 7 - Segnali che alzano il flag di CB1 a seconda della configurazione del bit 4 di PCR.

ad 1 da una transizione basso-alto (fig. 7). Verifichiamo ora qualcuna delle operazioni descritte con l'aiuto del VICLAB. Ponete in posizione operativa le due schede VL1 e VL2 e sistemate gli interruttori della VL2 in modo che tutti i LED del visualizzatore siano accesi. Eseguite:

PRINT PEEK (37148): RETURN

e vedrete comparire sul video il numero decimale 254. Questo significa che in PCR è contenuta la parola binaria 11111110. Per verificare il punto 1) della tabella 3 fate girare il seguente programma:

```
5 POKE 37148,30
10 PRINT PEEK (37148); PEEK (37149)
20 GET A$:IF A$= "*" THEN PRINT PEEK
   (37136)
30 GOTO 10
```

La prima istruzione azzera i bit 7-6-5 di PCR, lasciando invariati gli altri, memorizzando in esso la parola 00011110 (30 in decimale); la seconda farà apparire sullo schermo il contenuto di PCR ed IFR; la terza legge il contenuto di IOR B se viene premuto il tasto corrispondente all'asterisco.

Dopo il RUN appariranno sullo schermo due colonne di numeri affiancate. Una sarà composta solo dal numero 30 (contenuto di PCR) ed un'altra composta da tutti 0 (contenuto di IFR). Mentre il programma sta girando, ponete per un attimo a zero la linea CB2 agendo sul relativo interruttore. Vedrete il LED corrispondente che si spegnerà su un primo spostamento dell'interruttore e si riaccenderà quando quest'ultimo verrà riportato nella condizione di partenza: avete così provocato una transizione alto-basso su CB2. In base a tale operazione, come descritto al punto 1), si alzerà il bit 3 corrispondente ad F3 nel registro IFR ed allora la colonna di 0 si trasformerà in una colonna di 8. È infatti: $8_{10} = 00001000_2$. Il bit 3 è stato effettivamente posto ad 1. Sempre mentre il programma scorre, effettuiamo un'operazione di lettura del contenuto di IOR B, come programmato, premendo il tasto "*asterisco*": il bit 3 precedentemente alzato ritornerà a 0 (confronta tabella 2). In PCR abbiamo sempre la parola 00011110 quindi il bit 4 di tale registro è ad 1. Se è vero quanto abbiamo detto in precedenza, se noi provochiamo una transizione basso-alto questa volta su CB1, dovrebbe alzarsi il flag F4, corrispondente al bit 4 di IFR. Provocate quindi (sempre mentre il programma gira) tale transizione su CB1 spegnendo e riaccendendo il LED corrispondente: sullo schermo vedrete la colonna di 0 si trasformerà in una colonna di 16 che equivale ad aver posto ad 1 il bit 4 di IFR (essendo $16_{10} = 00010000$). Se effettuerete entrambe le operazioni descritte, la colonna citata sarà composta da 24 poiché risulteranno alzati i flag F3 ed F4 ($24_{10} = 00011000$).

Facciamo un ultimo esperimento verificando quanto descritto al punto 5) della tabella 3. Fate girare il seguente programma:

```
10 POKE 37148,130:POKE 37136,170
```

Esso porrà i bit 7-6-5 di PCR nella configurazione 100 e scriverà una parola (in questo caso 170) in IOR B. Partendo dalla configurazione in cui tutti i LED sono accesi, dopo il RUN quello corrispondente a CB2 si spegnerà. Riusciremo a riaccenderlo provocando, sempre tramite gli interruttori della VL2, una transizione alto-basso su CB1.

A questo punto dovreste aver cominciato a capire come avvengono gli scambi di dati asincroni tra dispositivi. Infatti l'ultimo esperimento è un primo rudimento di protocollo di handshake (eseguito naturalmente a mano).

Rimane ora da esaminare l'ultimo registro di controllo, ACR, che controlla i due timer, T1 e T2, lo shift register e l'operazione di latch in ingresso delle porte PA e PB. Le sue suddivisioni sono indicate in figura 8. ACR è posto alla locazione decimale 37147.

ACR ed il latch delle porte d'ingresso

Leggendo dei dati che si formano sulla porta d'ingresso e che variano nel tempo, può rendersi necessario bloccarli in determinati istanti e memorizzarli. Questo è possibile abilitando il latch degli ingressi tramite i bit 0 ed 1 di ACR che agiscono rispettivamente sulla porta A (PA) ee sulla porta B (PB). Quando tali bit sono a 0 non viene effettuato alcun latch. Quando sono ad 1 gli ingressi sono latch ed il valore in ingresso viene bloccato da una transizione attiva su CA1 o CB1 a seconda della porta usata. Chiariamo meglio il tutto con un esempio pratico. Montate i due pezzi del VICLAB sul computer.

Cominciamo con l'abilitare il latch sulla porta B scrivendo 1 nel bit *uno* (ACR1). In ACR all'accensione è contenuta la parola 01000000 (64 decimale) quindi dovendo alzare ACR1, dovremo scrivere in tale registro la parola 01000010 (66 decimale).

Eseguita allora

```
POKE 37147,66
```

il latch di PB è abilitato. Provate ora a comporre con gli interruttori la parola 10101010 che corrisponderà ad un LED acceso ed uno spento alternativamente (170 decimale). Provocate una transizione della linea CB1 (spegnendo e riaccendendo il relativo LED con l'interruttore) e sfilate la schedina d'ingresso VL2 dal connettore. Sul visualizzatore tutti i LED si riaccenderanno, quindi potremmo pensare che, dato che il registro IOR B rispecchia la condizione delle linee d'ingresso, in esso debba essere memorizzato il numero 255 (LED tutti accesi) mentre se andiamo a leggere il contenuto di tale registro ci accorgiamo che esso è 170. Questo era infatti il valore presente su PB quando abbiamo provocato la transizione attiva su CB1. Se inoltre, *prima* di andare a leggere IOR B (PRINT PEEK (37136)) andiamo a leggere il contenuto di IFR, PRINT PEEK (37148), esso sarà 16 che equivale alla condizione 1 per il flag d'interrupt di CB1 (F4 = 1).

ACR ed i due timer T1 e T2

Il 6522 possiede due timer interni. Essi possono trovare molteplici applicazioni tra cui: generazione di ritardi via hardware, generazione di singoli impulsi di durata programmabile (one-shot), generazione di treni d'impulsi (free running mode), conteggio della durata di un impulso, conteggio del numero d'impulsi contenuti in un certo intervallo di tempo ed altre cose interessanti. Le operazioni di conteggio ed invio di impulsi sono effettuate sulla linea PB6 per il timer 2 e su PB7 per il timer 1.

T1 è più potente di T2; infatti mentre il primo può manipolare impulsi singoli o treni, il secondo tratta solo singoli impulsi. Inoltre, come vedremo tra breve, T1 e T2 hanno conformazioni diverse. I bit 7-6 di ACR controllano T1 mentre il bit 5 controlla T2. Le modalità sono elencate nella tabella 4 (pagina 94).

Vediamo dove si trovano questi timer in

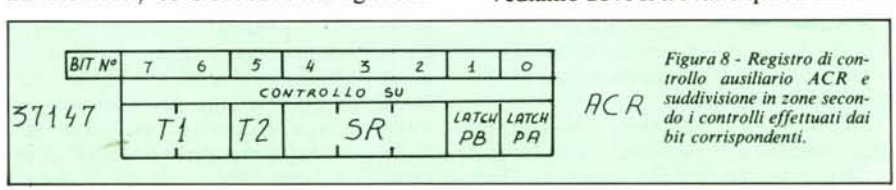

Figura 8 - Registro di controllo ausiliario ACR e suddivisione in zone secondo i controlli effettuati dai bit corrispondenti.

memoria e come funzionano (si faccia riferimento alla figura 9):

Timer 1

37140	contatore	: byte basso
37141		: byte alto
37142	latch	: byte basso
37143		: byte alto

Timer 2

37144	cont. + latch	: byte basso
37145	contatore	: byte alto

MODO	OPERAZIONE
bit 7 = 0	L'uscita su PB7 viene disabilitata
bit 7 = 1	L'uscita su PB7 viene abilitata
bit 6 = 0	T1 funziona in modo one-shot; viene generato un singolo interrupt alla fine di ogni conteggio
bit 6 = 1	T1 funziona in modo free-running e vengono generati interrupt continui
bit 5 = 0	T2 funziona in modo one-shot
bit 5 = 1	T2 conta gli impulsi che giungono sulla linea PB 6

Tabella 4 - Configurazione dei bit 7-6-5 di ACR per il controllo di T1 e T2.

Funzionamento del timer 1 - Possiede un contatore a 16 bit e quindi occupa due byte. In tali locazioni si deve porre un numero N che, occupando due byte, può essere compreso tra 0 e 65535. Quando il conteggio è avviato, tale numero viene decrementato alla frequenza di clock del sistema (per il VIC 1.1082 MHz circa). Si può allora supporre che sarà generato un ritardo di circa 1.1 microsecondi per ogni unità di N, quindi il ritardo massimo generabile con un solo caricamento del timer è 65535 × 1.1 = 72 millisecondi circa. Questo se sono rispettate le condizioni dette. In pratica, per ragioni che è inutile specificare, vengono generati N+2 cicli quindi, per essere precisi, nel contatore andrebbe caricato il numero N-2.

Caricamento del timer 1 - Vengono prima caricati il latch basso e quello alto rispettivamente con il byte basso e quello alto di N. Per far partire effettivamente il timer bisogna caricare il byte alto del contatore con il byte alto di N. Viene allora automaticamente abbassato il flag di T1 che blocca il timer, il contenuto dei latch viene trasferito nei due byte del contatore ed il conteggio inizia (viene cioè decrementato N con le modalità descritte). Quando esso termina viene alzato il flag d'interrupt. Il ciclo può ricominciare od arrestarsi secondo la configurazione dei bit 7-6 di ACR. Durante il funzionamento si può modificare il contenuto dei latch senza alterare il conteggio e ciò viene sfruttato per generare timing complessi. In corrispondenza ad ogni interrrupt vengono generati singoli impulsi su PB7 e rivelati in uscita se tale porta viene abilitata (contenuto del bit 7 di ACR). Se il contenuto dei latch non viene cambiato, gli impulsi inviati hanno frequenza costante.

Funzionamento del timer 2 - Possiede un unico registro a 16 bit di cui il byte basso è condiviso dal latch e dal contatore. Tale byte funziona da latch quando si scrive in esso e da contatore al momento in cui la CPU legge il valore in esso memorizzato. Il byte alto è contatore ed in esso si può sia leggere sia depositare un valore. Il modo di funzionamento come generatore one-shot è analogo a quello del timer 1. Come lettore di impulsi, con la debita configurazione del bit 5 di ACR, se noi introduciamo un valore nel byte alto, questo sarà decrementato di uno ogni volta che su PB6 giunge un impulso e quando il contenuto di tale byte sarà zero, si alzerà un flag d'interrupt.

Le applicazioni le vedremo la prossima volta con il programma VICTEL per la generazione dei segnali da inviare sulla linea telefonica per codificare i numeri telefonici.

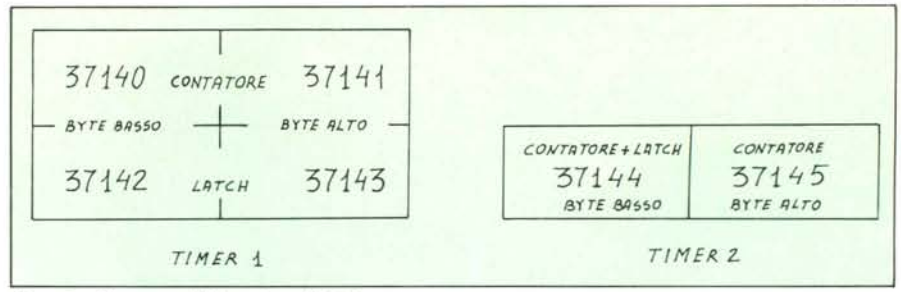

Figura 9 - Mappatura dei due timer del 6522.

N°	ACR4-ACR3-ACR2	OPERAZIONE
1)	000	SR è disabilitato
2)	001	Shift controllato da T1. Si carica SR e ciò avvia lo shift; quando sono stati inviati 8 bit si posiziona F2
3)	010	Shift controllato dal clock del VIA
4)	011	Shift controllato dagli impulsi applicati a CB1. Quando da CB2 sono entrati in SR 8 bit viene alzato il flag d'interrupt che potrà essere azzerato da un'operazione di lettura o scrittura in SR
5)	100	Il contenuto di SR è inviato in continuazione in uscita su CB2 sotto il controllo del TIMER 2
6)	101	Il contenuto di SR viene inviato in uscita e viene alzato il flag d'interrupt quindi lo shift si arresta. Tutto è controllato dal TIMER 2
7)	110	Shift sotto il controllo del clock del VIA
8)	111	Shift sotto il controllo degli impulsi applicati a CB1. Dopo l'ottavo bit trasmesso viene alzato il flag d'interrupt.

Tabella 5 - Configurazione dei bit 4-3-2 di ACR per il controllo dello Shift register.

ACR e lo shift register

Lo *shift register* posto alla locazione decimale 37146 può essere usato per trasmettere o ricevere una parola seriale sulla linea CB2; concettualmente ciò vuol dire che se noi carichiamo una certa parola in SR, per esempio 11000110, essa sarà trasmessa bit per bit (cioè si inizia con il primo 1, dopo un certo tempo viene trasmesso il secondo, poi uno 0 e così via). Le modalità di funzionamento di SR sono controllate dai bit 4-3-2 di ACR, la velocità di scorrimento della parola può essere controllata: dal timer 2 (valore contenuto nel byte alto del contatore); da un clock esterno applicato a CB1; dal segnale di clock generato sul pin 39 del 6502. La tabella 5 descrive le operazioni per cui è predisposto lo *shift register* a seconda della configurazione dei bit 4-3-2 di ACR. Illustriamo il punto 8) della tabella con un esperimento, certamente il più interessante, che dovrebbe escludere ogni dubbio sul funzionamento dello SR interno al VIA. Dopo aver sistemato al proprio posto il VICLAB introducete in macchina il seguente programma:

```
 5 POKE 37138,255
10 POKE 37146,1
15 POKE 37147,92
20 PRINT PEEK (37146)
25 POKE 37136, PEEK (37146)
30 GOTO 20
```

Spieghiamolo brevemente: 5 - pone tutte le linee come uscite; 10 - scrive in SR la parola 00000001 (1 decimale); 15 - predispone ACR come nel punto 8) della tabella; 20 - scrive sullo schermo il contenuto di SR; 25 - riporta sul visualizzatore il contenuto di SR; 30 - evidente.

Questo programma invia serialmente in uscita il contenuto di SR. Lo scorrimento avviene a partire dal bit più significativo; in pratica viene trasmesso il bit di ordine più elevato e gli altri 7 scorrono verso sinistra di una posizione e così via fino all'ultimo. Quando il bit che si trovava in posizione zero supera la settima posizione viene alzato il flag F2 di IFR (nel nostro caso esso viene subito riazzerato perché andiamo a leggere in IOR B ed il ciclo potrà ricominciare, *altrimenti no*). La serie di impulsi da applicare a CB1 la simuliamo aprendo e chiudendo l'interruttore ad essa relativo (dovremo in pratica accendere e spegnere il LED verde di destra). Lo scorrimento della parola in SR viene *simulato* sul visualizzatore. Vedrete infatti dopo il RUN accendersi il LED rosso in posizione zero cioè quello più a destra il quale si *sposterà* di una posizione verso sinistra dopo ogni impulso inviato su CB1; quando esso avrà superato la posizione *sette*, rientrerà in posizione *zero* e il ciclo potrà essere ricominciato. La trasmissione reale avviene su CB2 dove vedrete il LED corrispondente, spento per sette impulsi, accendersi all'ottavo: sono stati infatti inviati sette 0 ed un 1.

DRAGON

HOME
PROFESSIONAL
COMPUTER

32-64

Distributore: **ECO s.r.l.** - Verona - Tel. 045 - 913297

32 K

- Microprocessore 6809 E
- Almeno due volte più potente degli altri home computers
- Tastiera professionale
- Interfaccia parallela Centronics
- Floppy Disk 5'' 180 Kb - DOS avanzato
- Uscite indipendenti TV e monitor colore
- Basic Microsoft esteso
- Set di istruzioni grafiche
- Risoluzione 256 × 192 punti
- Doppio Joystick 64 direzioni
- Ampia disponibilità di software

64 K

- 100% compatibile con il DRAGON 32 ed in più:
- Interfaccia Seriale RS232C
- Sistema Operativo OS9 unix-like Real Time Multiuser Multitasking
- Linguaggi di programmazione: BASIC 09, C Compiler, PASCAL
- Programmi applicativi:
  Foglio Elettronico DYNACALC
  Trattamento Testi STILOGRAPH/MAILMERGE
  Banca Dati RECORD MANAGEMENT SYSTEM

BARI	NUOVA HALET	Via Capruzzi, 192
BERGAMO	BIT CENTER	Via Tito Livio, 4
BOLOGNA	TEKNOS	Via Zanardi, 23
BOLOGNA	ERRE TI.	Via Lombardi, 43
BOLZANO	COMPUTER MARKET	Via S. Maria del Conforto Merano
BRINDISI	DI BIASE	Viale P. Togliatti, 22/32
CAMPOBASSO	SISTEMA	Via Monsignor S. Bologna, 10
CATANIA	COMPUTER SHOP	Via V.E. Orlando, 164
CREMONA	ARCHIMEDE	Via Palestro, 11/B
FERRARA	PROGRAM	Via Pietro Gobetti, 13
FIRENZE	SUMUS	Via S. Gallo, 16/R
GENOVA	SOVECO	Tel. 010/594821
GORIZIA	TECNOPOWER	Via Marconi, 19 - Turriaco
LECCE	DI BIASE	Viale Marche, 21
MILANO	INTERSYSTEMS	Viale Certosa, 91
NAPOLI	C.F. ELETTRONICA	C.so Vittorio Emanuele, 64
NAPOLI	C.F. ELETTRONICA	Via Luca Giordano, 40/42
NAPOLI	2L COMPUTERS	Via Cintia - Parco S. Paolo Isolato 1 - Fuorigrotta
NAPOLI	MARIO DE MARCO	Via Kerbaker, 35
NAPOLI	ELETTRONICA MERID. di MICHELE TROMBONE	Via S. Tommaso D'Aquino, 53
PADOVA	GABRIELI	Piazza Erbe, 45/49
PADOVA	SIC ITALIA	Via S. Pietro, 82
PARMA	BIT SHOW	Via Imbriani, 41
PERUGIA	MICROCOGIT	Viale Indipendenza, 39
PESARO	CLOCK COMPUTER s.a.s.	Via Cherubini, 8
ROMA	BARBAGALLO	Via F.lli Bonnet, 5
ROMA	BIT COMPUTERS	Via Flavio Domiziano, 10
ROMA	BIT COMPUTERS	Via F. Satolli, 57
ROMA	COMPUTER CENTER	Via Nizza, 48/52
ROMA	COMPUTER MARKET	Piazza S. Donà di Piave, 14
ROMA	ECCELSA	G.R.A. Km. 42,800
ROMA	ELETTRONICA 2003	Via G. Gozzi, 13
ROMA	ERT 80	Via dei Georgofili, 67
ROMA	FOTO & COMPUTERS	Via Assisi, 78
ROMA	GEA	Via Taro, 3
ROMA	IL DISCOFILO	Via Tosatti, 19
ROMA	RINALDI	Via Corsinio, 13
RAVENNA	LEMON ITALIA	Via Rotta, 18/A
SIENA	ELETTRONICA	Via di Gracciano nel Corso, 111 Monte Pulciano
SIRACUSA	PASI ELETTRONICA	Via Dante Alighieri, 37 - Rosolini
TARANTO	FUTURA	Via Ovidio, 22
TERNI	EUREKA INFORMATICA	Via Beccaria, 20
TORINO	SOFTGAMES	Via Duchessa Iolanda, 9
TORINO	ZUCCA COMPUTERS	Via Tripoli, 179
TREVISO	M.C.E.	Via Dante, 9 - Vittorio Veneto
TRIESTE	ARIO DRIOLI	Viale XX Settembre
VARESE	SUPERGAMES	Via Carrobio, 13
VENEZIA	BIT COMPUTERS	Via Verdi, 9 - Mestre
VENEZIA	PERSONAL COMPUTER	Cannaregio 5898
VERONA	A.P.L.	Via Tombetta, 35/A
VERONA	COMPUTER SHOP	Piazza Garibaldi, 8 - Legnago
VERONA	MOS 80	Via del Pontiere, 2