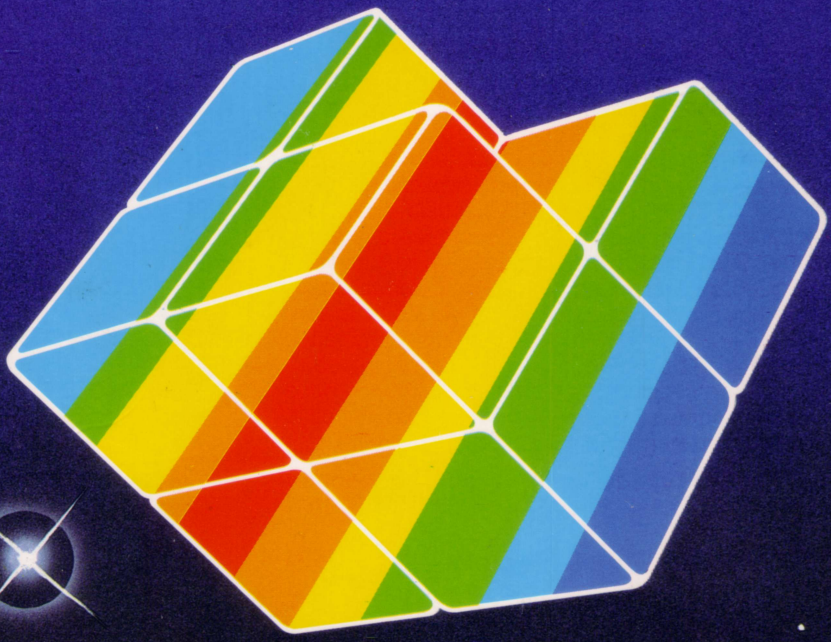
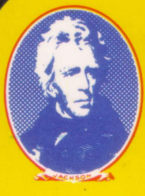
Jonathan A. Titus David G. Larsen
Christopher A. Titus

Tecniche di interfacciamento dell'APPLE

ESEMPI DI CIRCUITI INTERFACCIA
GIÀ COLLAUDATI, PER IL CONTROLLO
DEI DISPOSITIVI ESTERNI
E LA COMUNICAZIONE CON ALTRI CALCOLATORI



HOME E PERSONAL COMPUTER



GRUPPO EDITORIALE
JACKSON
DIVISIONE LIBRI

EDIZIONE ITALIANA

Jonathan A. Titus David G. Larsen
Christopher A. Titus

Tecniche di interfacciamento dell'APPLE

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

Seconda edizione 1986

Fotocomposizione: Corponove s.n.c. — Bergamo
Stampa: S.p.A. Alberto Matarelli - Milano -

SOMMARIO

PREFAZIONE

Scopo di questo libro è farvi conoscere i segnali presenti nel calcolatore Apple II ([1]) e mostrarvi come questi segnali possano essere utilizzati per controllare, mediante programmi scritti in linguaggio BASIC, dei dispositivi esterni. È stata sviluppata una scheda di prova (*breadboard*), interfacciabile con un qualunque calcolatore, allo scopo di rendere più rapida la vostra attività di progettazione e di verifica dei circuiti, in modo che possiate realizzare agevolmente i numerosi ed interessanti esperimenti che compaiono nel testo. Ricorrendo ad un metodo di progettazione come quello descritto in questo libro, potrete utilizzare il vostro tempo concentrandovi esclusivamente sui concetti esposti invece che ricercando gli errori dei vostri circuiti. Ciò non toglie che avrete ugualmente modo di costruire e provare molti circuiti digitali, come pure circuiti che utilizzano convertitori analogico-digitali e digitale-analogici.

Abbiamo optato per l'Apple II con una memoria RAM (cioè di lettura e scrittura) di 16 K e l'interprete BASIC Applesoft ([2]). Questo software garantisce un'elevata flessibilità, ed è notevolmente conveniente quando si usano circuiti esterni d'interfaccia. L'interprete BASIC Applesoft prevede due comandi d'uso generale con cui trasferire informazioni al e dal calcolatore. Queste istruzioni sono facilmente utilizzabili, anche senza una conoscenza particolareggiata del circuito integrato (CI) a microprocessore 6502, che costituisce il "cuore" dell'Apple.

Prima di tutto parleremo dei segnali di controllo che sono disponibili sul calcolatore Apple per l'interfacciamento, e ne illustreremo l'uso. Alcuni segnali non saranno descritti, in quanto generalmente non sono usati nei circuiti d'interfaccia, ma sono riservati a speciali schede d'interfaccia reperibili in commercio.

In secondo luogo illustreremo come l'Apple possa identificare o indirizzare dei dispositivi esterni mediante l'utilizzo di due istruzioni d'uso generale, PEEK e POKE. Questi comandi sono fondamentali per il controllo dei dispositivi esterni; pertanto ci soffermeremo abbastanza sulle loro modalità operative, e sull'uso di tutta una serie di circuiti che possono essere utilizzati per identificare specifici dispositivi d'ingresso/uscita (I/U). Si vedrà inoltre come l'Apple sia in grado di trasferire le informazioni a/da i dispositivi esterni attraverso il bus dati bidirezionale; i principali circuiti usati per le *porte d'ingresso* e le *porte di uscita* saranno descritti dettagliatamente. Presenteremo dei veri circuiti, in modo che il lettore possa in breve tempo utilizzare gli

---

(1) Apple e Apple II sono marchi di fabbrica registrati della Apple Computer, Inc.
(2) Applesoft è marchio di fabbrica della Apple Computer, Inc.

esempi, che sono numerosi, per progettare per conto proprio dei dispositivi d'interfaccia.

Si vedrà anche la potenza dei programmi scritti in linguaggio BASIC, e come i dati vengono elaborati all'interno del calcolatore al fine di fornire risultati significativi. Abbiamo introdotto dei semplici programmi di controllo, per mostrare come possano interagire i programmi in BASIC ed i dispositivi di I/U. Il lettore sarà messo in grado di scrivere semplici programmi di controllo e di elaborazione di dati, che potrà applicare alle porte e ai dispositivi di I/U di cui dispone.

Il calcolatore non è sempre sincronizzato con i dispositivi esterni; per questo motivo deve esistere una certa interazione fra il calcolatore stesso ed i dispositivi di I/U, affinché entrambi sappiano quando l'altro è pronto per una determinata operazione. E questo ci porta a parlare degl'indicatori (*flags*), e cioè di quei segnali utilizzati dal calcolatore e dai dispositivi esterni di I/U affinché le informazioni vengano trasferite secondo un certo ordine. Data l'importanza dei flags, dedicheremo loro un certo spazio, come pure ai circuiti corrispondenti usati in pratica nei dispositivi esterni. Tratteremo anche il software, perché i flags non hanno nessuna utilità se non sono rilevati da un programma di controllo.

Siamo partiti dal presupposto che chi legge conosca già sufficientemente i comandi del BASIC Applesoft. A chi prenda contatto solo adesso con il calcolatore Apple consigliamo di spendere un po' di tempo per rivedere i comandi elementari, come FOR, GOTO, IF... THEN, PRINT ed INPUT. Gli altri comandi saranno introdotti nel testo e negli esperimenti, con una spiegazione dettagliata della loro operatività. A lettura ultimata, l'uso di questi, e di altri comandi, dovrebbe essere del tutto automatico.

Nel Capitolo 6 troverete sedici esperimenti illustrati passo passo, che potrete eseguire allo scopo di suffragare i numerosi principi teorici sull'interfacciamento che sono stati sviluppati nel testo. I lettori vedranno altresì la potenza dei programmi scritti in linguaggio BASIC per il controllo dell'interfacciamento e per l'elaborazione delle informazioni trasferite a/da i dispositivi di I/U. Ci siamo sforzati d'illustrare una vasta gamma di applicazioni d'interfacciamento in qualche modo interessanti. Da questi esperimenti si vedrà che gli stessi principi di base valgono per tutti i circuiti d'interfaccia, dai più semplici ai più complessi.

Siamo coscienti della difficoltà di scrivere un libro che, come questo, sia destinato ad un pubblico estremamente differenziato sul piano delle conoscenze specifiche, un pubblico cioè che va dal principiante all'utente esperto; ed è per questo che abbiamo deciso di partire, diciamo così, a metà strada: quindi abbiamo saltato la numerazione binaria, la conversione decimale-binario, i principi fondamentali dell'elettronica digitale ed il montaggio della scheda di prova. Questi argomenti trovano ampia trattazione in altri testi, e sono probabilmente ben noti a quei lettori che si trovano a metà strada fra il principiante e l'esperto. In ogni caso abbiamo inserito qua e là dei brevi riepiloghi, proprio allo scopo di rinfrescare la memoria. Comunque in questa sede non siamo scesi eccessivamente nei particolari, ma solo quanto basta per permettere di proseguire la lettura.

Abbiamo presupposto poi una certa familiarità con i circuiti integrati, o chips, della famiglia SN7400, come le quattro porte NOR SN7402 ed il chip a quattro latches SN7475. Saranno poi illustrati altri chips, abbastanza dettagliatamente perché il lettore possa utilizzarli così come è spiegato nel testo e negli esperimenti. A chi volesse impiegare questi dispositivi in altre applicazioni suggeriamo di richiedere ai costruttori i cataloghi relativi, che forniranno le informazioni necessarie per moltissimi usi; i cataloghi riportano inoltre tutti i cambiamenti o le modifiche eventualmente introdotti in un dispositivo "aggiornato", o in un dispositivo "migliorato" grazie a qualche caratteristica particolare.

Il calcolatore Apple II ha otto connettori d'interfaccia a 50 conduttori, di uso generale. I segnali fondamentali di bus usati negli esperimenti sono derivati dai segnali relativi a tali connettori, in modo che, se qualcuno dei lettori vuol progettare e costruire per proprio conto dei circuiti d'interfaccia da inserire in uno di questi "slots", troverà che gli stessi segnali sono già pronti sui connettori. Comunque esistono anche dei segnali d'uso generale generati dall'Apple II allo scopo di facilitare il lavoro d'interfacciamento. Questi segnali, ed il modo di usarli, sono descritti dettagliatamente nel Capitolo 7. Ne parliamo per ultimi per il fatto che non sono di uso generale, ma specifici dell'Apple e, anzi, in molti casi di un particolare connettore. Per illustrarne l'uso abbiamo descritto un semplice circuito d'interfaccia per comunicazioni seriali asincrone, e abbiamo inoltre mostrato il listato del programma che lo controlla. Questo tipo d'interfaccia può servire per comunicare con altri calcolatori, stampanti seriali, modems ed altri dispositivi d'interfaccia che utilizzano per i dati il formato asincrono seriale.

Non descriviamo la programmazione fatta in linguaggio assemblatore, perché questo è un argomento specializzato che richiede un grosso bagaglio di conoscenze. Abbiamo comunque introdotto una semplice subroutine in linguaggio assemblatore, da utilizzare in più di un esperimento; l'abbiamo introdotta per un valido motivo: la funzione equivalente non è disponibile in Applesoft. Si tratta della funzione di AND logico fra bytes di 8 bits. In Applesoft l'AND logico non è altro che un'operazione che fornisce il valore "vero" o "falso", e come tale non è facilmente utilizzabile in un'operazione di AND fra bits. Oltre a ciò, la suddetta subroutine in linguaggio assemblatore fornisce un'idea del modo in cui tali routines sono accessibili da parte di un programma in BASIC. Abbiamo optato per il comando USR(X), più complicato, piuttosto che per il comando CALL, perché così imparerete qualcosa di più.

Abbiamo constatato che l'Apple ha alcuni limiti: ad esempio, manca un semplice comando di "arrotondamento" utilizzabile per arrotondare un numero ad un dato numero di cifre decimali, ad esempio 4.1986 a 4.20. Anche l'assenza di un comando di AND bit per bit è un limite, che viene superato, come si è già visto, con una routine in linguaggio assemblatore. Abbiamo poi constatato che il comando di WAIT, usato per verificare lo stato di singoli bits, pur essendo utile in potenza, "tiene in sospeso" il calcolatore se non è trovata la condizione richiesta: non incontrando la condizione, il calcolatore continua ad aspettare, e, per far riprendere l'esecuzione del programma, bisogna rimettere nello stato iniziale il calcolatore tramite il tasto RESET.

Sono disponibili una visualizzazione a colori ed una grafica gradevole; noi comunque, nel nostro sistema, ci siamo serviti di un monitor in bianco e nero.

Molti chips di uso specifico, come ad esempio i convertitori analogici, li abbiamo scelti per la loro semplicità, il costo contenuto e la facile disponibilità. Questo non vuol dire da parte nostra approvazione incondizionata per tali prodotti; aumentando il livello di sofisticazione dell'interfacciamento, troverete altri specifici dispositivi capaci di assolvere la stessa funzione, ma forse con delle caratteristiche aggiuntive, una più alta risoluzione, diverse alimentazioni, etc. Il nostro scopo è darvi degli strumenti per partire, non un repertorio di tutte le possibili interfacce al sistema Apple. Che sarebbe in ogni caso un'impresa impossibile.

A chi fosse interessato ad un ampliamento di quanto è detto in quest'opera consigliamo la lettura dei seguenti testi:

- *6502 Software Design (21656)*;
- *Programming & Interfacing the 6502, With Experiments (21651)*;
- *Microcomputer-Analog Converter Software and Hardware Interfacing (21540)*.

Si potrà leggere anche *TRS-80 Interfacing, Book 2*, che, pur avendo come argomento il calcolatore TRS-80, tratta dettagliatamente di alcuni aspetti più avanzati dell'interfacciamento, quali il pilotaggio di carichi ad alta corrente/alta tensione, le comunicazioni seriali, il controllo remoto, i convertitori analogici, il filtraggio, l'elaborazione dei dati ed altri interessanti argomenti. Noterete ben presto che le analogie fra il TRS-80 e l'Apple sono molto maggiori delle differenze: ad esempio, i segnali di controllo ed i comandi in BASIC sono pressoché identici.

Tutti i testi che abbiamo citato sono disponibili presso la Howard W. Sams & Co., Inc., 4300 West 62nd Street, Indianapolis, IN 46268.

Le configurazioni dei pins che compaiono nella maggior parte delle figure, a meno di specifiche segnalazioni, ci sono state gentilmente fornite dalla Texas Instruments, Incorporated. I nomi Apple e Applesoft sono marchi di fabbrica della Apple Computer, Inc., Cupertino, CA. Il nome TRS-80 è un marchio di fabbrica registrato della Radio Shack.

Ci auguriamo che questo libro incontri la vostra approvazione, e che possa insegnarvi a progettare e costruire per conto vostro dei circuiti d'interfaccia.

Jonathan A. Titus,
Christopher A. Titus,
David G. Larsen
"The Blacksburg Group"

CAPITOLO 1

IL PROCESSORE 6502

Il sistema Apple II® (Apple®), prodotto dalla Apple Computer, Inc., utilizza il circuito integrato a microprocessore 6502. Questo "chip" costituisce il cuore dell'unità centrale di elaborazione (CPU) del calcolatore, che è la sede dove effettivamente hanno luogo le operazioni matematiche, logiche, di decisione, e tutte le altre. Il microprocessore 6502 è prodotto dalla MOS Technology (Norristown, PA 19401), dalla Rockwell International (Anaheim, CA 92803) e dalla Synertek Corporation (Santa Clara, CA 95051).

Il 6502 è un processore ad 8 bits. Questo vuol dire che tutte le operazioni matematiche, logiche, di trasferimento di dati, d'ingresso e di uscita avvengono su otto bits binari alla volta. Chiaramente ciascun bit può essere o un uno logico o uno zero logico. Il 6502 utilizza un bus dati ad 8 bits per trasferire le informazioni fra lo stesso 6502 e le varie locazioni di memoria, e i dispositivi d'ingresso/uscita (I/U), quali tastiera, stampante, etc. Nei casi in cui il valore dell'informazione supera il limite degli otto bits, si utilizzano multipli delle parole-dato di 8 bits: ciascuna parola-dato di 8 bits prende il nome di *byte*.

Il massimo valore esprimibile con otto bits è 11111111_2, ovvero 255_{10}. Se, in un sistema ad 8 bits, si devono trattare valori superiori, si dovrà ricorrere ad operazioni multibyte. Generalmente, ciò significa operare sui bytes corrispondenti a due parole - dato; nel caso di parole-dato più lunghe, occorre gestire la corrispondente serie di bytes. In tal modo è possibile elaborare senza difficoltà valori elevati, superiori a 255. È comunque importante ricordare che la CPU dell'Apple è in grado di elaborare e trasferire non più di otto bits, ovvero un byte, per volta.

Il 6502 utilizza un'unica serie di otto pins per realizzare la connessione con il bus dati del calcolatore. Il bus dati serve per trasferire l'informazione al e dal calcolatore. Questo tipo di bus è detto *bidirezionale*, perché fa circolare l'informazione in due direzioni diverse. Possiamo paragonarlo ad un'autostrada, sulla quale le macchine possono viaggiare la mattina in una direzione, e la sera nella direzione opposta.

Il 6502 genera sul circuito integrato dei segnali di controllo, che vengono usati sia internamente che esternamente per controllare e gestire il flusso dell'informazione

sul bus, in una sola direzione per volta. Più avanti esamineremo il modo in cui questi segnali vengono generati ed utilizzati.

LA MEMORIA

Tutti i calcolatori sono dotati di memoria. La memoria generalmente serve per contenere tanto un programma destinato a controllare l'attività del calcolatore, quanto l'informazione da elaborare. In un calcolatore con il microprocessore 6502 in ciascuna locazione di memoria possono essere contenuti otto bits d'informazione, ovvero un byte di dati. La maggior parte delle memorie sono costituite da multipli di tali celle di memorizzazione da un byte, generalmente sotto forma di multipli di 1024, valore che viene detto in breve 1 K.

Le locazioni di memoria devono essere indirizzate in modo che il calcolatore sappia esattamente dove memorizzare i dati oppure dove ottenere le informazioni relative ad una data serie d'istruzioni del programma. Il microprocessore 6502 ha 16 uscite d'indirizzi, che gli consentono di specificare ciascuna delle 2^{16}, ovvero 65.536, locazioni di memoria, ciascuna delle quali può contenere un byte. Questo spesso si abbrevia con 64 K, il che vuol dire che possono venir indirizzati 64 K *bytes* d'informazione. In quasi tutti i sistemi di memoria dei microcalcolatori, ogni locazione di memoria è indirizzata unicamente con un indirizzo a 16 bits.

Le linee del bus indirizzi sono indicate con sigle che vanno da A0 ad A15, corrispondenti rispettivamente al bit meno significativo (LSB) ed a quello più significativo (MSB). Tanto l'LSB quanto l'MSB possono essere sia un uno logico che uno zero logico, ma in rapporto alla *posizione* l'LSB ha *valore* zero o uno, mentre l'MSB ha *valore* zero o 32.768. Essendo il 6502 un processore ad 8 bits, spesso le linee degl'indirizzi sono ripartite in due gruppi di otto linee ciascuno, A7-A0 ed A15-A8. Le linee A7-A0 sono riferite alla parte bassa dell'indirizzo (LO), le linee A15-A8 alla parte alta dell'indirizzo (HI). In molti calcolatori basati sul 6502 l'indirizzo HI è detto anche *indirizzo di pagina*: la memoria infatti può essere divisa arbitrariamente in 256 pagine, con 256 bytes per pagina. Esamineremo gli usi del bus indirizzi più avanti, quando parleremo delle istruzioni software e quando svilupperemo i circuiti d'interfaccia. Contrariamente al bus dati, il bus indirizzi è monodirezionale, cioè l'informazione di un indirizzo fluisce in una sola direzione, dalla CPU alla memoria ed ai dispositivi esterni.

In Figura 1.1 compare la configurazione dei pins del 6502. Anche se probabilmente molti dei segnali non hanno per il momento per voi alcun senso, dovreste comunque essere in grado di riconoscere gli 8 pins d'ingresso/uscita del bus dati ed i 16 pins di uscita degl'indirizzi.

Dal momento che in questo paragrafo parliamo della memoria, aggiungiamo che nei microcalcolatori sono impiegati fondamentalmente due tipi di memoria, e cioè:

1) *Memorie di lettura e scrittura* (RAM): questo tipo di memoria serve a memorizzare i dati destinati ad essere modificati o aggiornati. Il calcolatore dev'essere in

grado di collocare l'informazione in una locazione di memoria, e poi tornare a leggerla. Per lo stesso motivo nella memoria RAM sono memorizzati i programmi destinati ad essere cambiati. Il meno costoso dei calcolatori della linea Apple contiene 16.384, ovvero 16 K bytes, di memoria RAM.

2) *Memorie di sola lettura* (ROM): questo tipo di memoria si usa per i dati e le parti di programma destinate a restare invariate. Il programma interprete BASIC del vostro Apple risiede in chips di memoria di sola lettura, ed esattamente in 12 K di ROM.

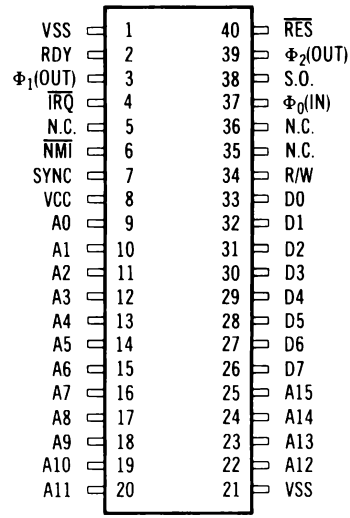


Figura 1.1 — Configurazione dei pins del chip del microprocessore 6502.

Questi due tipi di memoria si suddividono poi a loro volta in vari sottotipi: le memorie RAM possono essere *statiche* o *dinamiche*. I chips di una memoria statica conservano i valori in essi memorizzati finquando questi non vengono modificati. Le memorie dinamiche invece devono essere continuamente "rinfrescate", mediante un hardware esterno, a intervalli di qualche millisecondo, altrimenti "dimenticano", o, in altre parole, perdono i dati memorizzati in esse. Le memorie RAM presenti nell'Apple sono dinamiche e, corredate dell'opportuna circuiteria di rinfresco, trovano posto sulla scheda a circuito stampato del calcolatore.

Esistono diversi tipi di memorie di sola lettura, generalmente tutte statiche, che differiscono nel modo in cui i valori di 8 bits vengono memorizzati nelle locazioni di memoria. I due tipi più importanti sono quello *programmato a maschere* e quello *programmato a campi*. Nelle memorie programmate a maschere i valori dei dati, i programmi, etc. vengono memorizzati nel corso delle varie fasi della costruzione:

queste memorie in genere vengono dette ROM. Invece le memorie programmate a campi richiedono una circuiteria di programmazione particolare per la memorizzazione degli uni e degli zeri logici nelle diverse locazioni. Alcune ROM programmabili a campi, o PROM, come vengono generalmente chiamate, possono venir cancellate da una luce ultravioletta di elevata intensità, e sono poi riprogrammabili. Questo è molto utile con programmi in fase di sviluppo destinati ad essere memorizzati in memorie di sola lettura: infatti non occorre sviluppare maschere e chips (che è un'operazione costosa) tutte le volte che si trova un errore nel programma o si effettua una modifica.

Ancora qualche parola sulle memorie a semiconduttore. Le memorie di lettura/scrittura (RAM) sono *volatili*, perché i dati (cioè il programma ed i valori) "si volatilizzano", ovvero scompaiono, non appena si stacca l'alimentazione dal calcolatore. Le memorie di sola lettura, invece, sono dette non volatili, perché conservano i dati e le istruzioni di un programma (l'interprete BASIC ad esempio) anche dopo che è stata tolta l'alimentazione.

Nella maggior parte dei packages di memorie a circuito integrato, ovvero chips, non ci sono tutte le connessioni delle 16 linee d'indirizzi: i chips hanno solo le connessioni relative agl'indirizzi che bastano per indirizzare unicamente le celle di memoria contenute nel singolo chip. Quindi un chip a 64 bytes, piccolo rispetto agli standard attuali, avrà solo 6 ingressi per le linee d'indirizzo, mentre un chip di memoria di 1024 (1 K) bytes avrà 10 ingressi per le linee d'indirizzo. Chips di memoria di questo tipo dispongono di un ingresso aggiuntivo di controllo, ovvero di un ingresso di attivazione del chip (*chip-enable*), che consente di selezionare blocchi o gruppi di chips, uno per volta. Sono utilizzabili vari circuiti di decodifica e di selezione per costruire un blocco di memoria di 32 K partendo da chips di 64 bytes, o di 1 K byte, oppure dalla combinazione dei due tipi. Il punto più importante qui è che non è necessario *collegare direttamente* ai chips di memoria tutte le 16 linee d'indirizzo, anche se poi si userà una data combinazione di tutti i 16 bits d'indirizzo per selezionare univocamente un dato byte. Non ci si deve disorientare quando si prende in considerazione una memoria di 1 K 4 bits, che dispone soltanto di 10 ingressi per gl'indirizzi ed uno di attivazione del chip! Questo punto sarà comunque esaminato nei dettagli quando si parlerà dei trasferimenti di dati in ingresso/uscita.

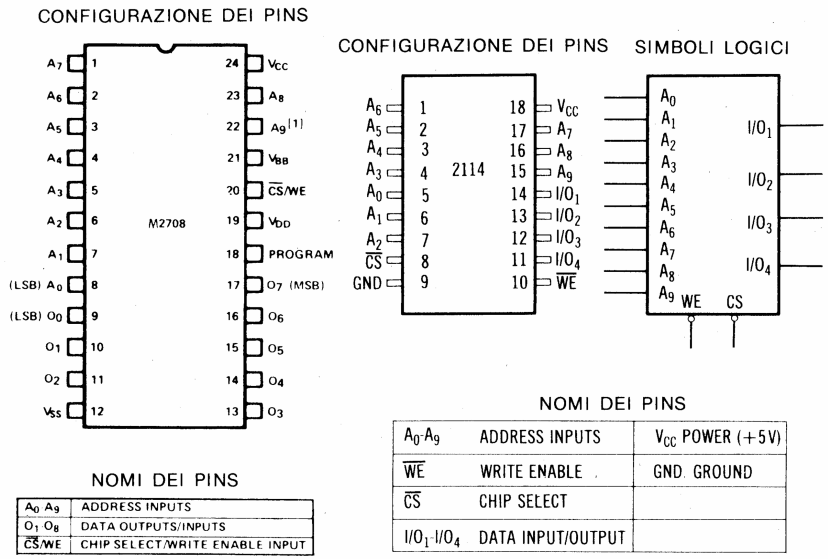
Il processore 6502 genera un segnale di controllo destinato a controllare il flusso dell'informazione lungo il bus dati. Questo segnale è indicato con READ/$\overline{\text{WRITE}}$, o, più semplicemente, con R/$\overline{\text{W}}$. Ogniqualvolta deve aver luogo un'operazione di lettura, o di scrittura, il 6502 deve specificare un indirizzo su 16 bits, che individui la "cella" di memoria che dev'essere interessata dal trasferimento. In questo caso la cella è costituita da una parola di 8 bits, vale a dire un byte.

La "sopralineatura" posta sopra una parte della notazione del segnale sta ad indicare che, quando il segnale è uno zero logico, è in corso un'operazione di scrittura, mentre, quando è nello stato logico uno, è in corso un'operazione di lettura. In tal modo un'unica linea controlla tutte le funzioni della memoria. In alcuni calcolatori e periferiche basati sul 6502, si può vedere una "separazione" del segnale, finalizzata

a fornire due segnali di controllo della memoria: lettura della memoria ($\overline{MEMR}$ oppure $\overline{MR}$) e scrittura della memoria ($\overline{MEMW}$ oppure $\overline{MW}$). In questo modo viene utilizzata qualche porta in più, e per questo in molti casi il segnale R/$\overline{W}$ viene usato direttamente. Nel microprocessore 6502 questo segnale è disponibile sul pin 34.

Troverete anche la notazione RAM, usata in modo abbastanza diffuso per designare le memorie di lettura/scrittura. Infatti la sigla RAM sta per *random-access memory* (memoria ad accesso casuale): tutti gli attuali dispositivi di memorizzazione sono ad accesso casuale, nel senso che si può indirizzare una locazione, e poi un'altra locazione, senza dover scorrere in sequenza tutte le locazioni poste fra i due indirizzi.

In figura 1.2 diamo le configurazioni dei pins relative a chips di memoria tipici.



NOMI DEI PINS

A_0 A_9	ADDRESS INPUTS
O_1 O_8	DATA OUTPUTS/INPUTS
$\overline{CS}$/$\overline{WE}$	CHIP SELECT/WRITE ENABLE INPUT

NOMI DEI PINS

A_0-A_9	ADDRESS INPUTS	V_{CC} POWER (+5V)
$\overline{WE}$	WRITE ENABLE	GND GROUND
$\overline{CS}$	CHIP SELECT	
I/O_1-I/O_4	DATA INPUT/OUTPUT	

Figura 1.2 — Configurazione dei pins della memoria PROM 2708 di 1 K x 8 e della memoria RAM 2114 di 1 K x 4.

Per ulteriori informazioni sulle memorie, rimandiamo a:

- *Intel Memory Design Handbook*, Intel Corporation, Santa Clara, CA 95051, 1975.
- *The 8080A / 9080A MOS Microprocessor Handbook*, Advanced Micro Devices, Inc., Sunnyvale, CA 94086, 1977.
- *Mostek Memory Products Catalog*, Mostek Corporation, Corollton, TX 75006, 1977.
- *Bipolar and CMOS Memory Data Book*, Harris Semiconductor Prod. Div., Melbourne, FL 32901, 1978.

I DISPOSITIVI D'INGRESSO/USCITA (I/U)

La maggior parte dei sistemi basati su microcalcolatori è inutilizzabile senza dei dispositivi d'ingresso/uscita. Questi dispositivi possono essere delle periferiche standard, come lettori di schede, stampanti, visualizzatori, oppure sensori, unità di controllo, ed altri dispositivi che non si associano normalmente ad un calcolatore. L'Apple non fa eccezione: già prevede infatti il collegamento con un certo numero di dispositivi, che sono uno schermo televisivo, un registratore a cassette ed una tastiera.

Si possono poi aggiungere al calcolatore degli altri dispositivi di I/U, o progettati da voi stessi o disponibili in commercio, purché siano compatibili con l'Apple. Questi dispositivi di I/U sono assimilabili alle singole locazioni di memoria di cui si è parlato nel paragrafo precedente: i dispositivi di I/U sono collegati al bus dati, dal momento che è da essi o ad essi che i dati vengono trasferiti, e sono collegati pure con il bus indirizzi, in modo da poter essere indirizzati unicamente dal microprocessore 6502.

Un segnale di controllo, cioè READ/$\overline{\text{WRITE}}$ (o R/$\overline{\text{W}}$), serve a sincronizzare il flusso dei dati verso e da i dispositivi di I/U. Nei calcolatori basati sul 6502 questo segnale serve anche a controllare il flusso delle informazioni verso e da i chips di memoria. Quindi nei calcolatori basati sul 6502 non c'è nessuna differenza fra gl'indirizzi di memoria e gl'indirizzi dei dispositivi di I/U. Invece nei calcolatori basati sui microprocessori 8085 o Z-80 esistono tecniche diverse per indirizzare in modo indipendente la memoria ed i dispositivi di I/U. Nel caso dell'Apple, visto che è usato un unico segnale di sincronizzazione per controllare tanto la memoria quanto i dispositivi di I/U, il processore 6502 sarà in qualunque momento o in lettura o in scrittura. Quando il segnale R/$\overline{\text{W}}$ è un uno logico, il 6502 legge informazioni *dal* bus dati; quando invece è uno zero logico, il 6502 scrive dei dati *su* un dispositivo esterno di I/U oppure *in* una locazione di memoria. La "sopralineatura" posta sulla W significa semplicemente che l'operazione di scrittura ha luogo quando il segnale R/$\overline{\text{W}}$ è uno zero logico. Vi capiterà certamente d'incontrare altri nomi di segnali con una sopralineatura: vorrà ancora dire semplicemente che quei segnali sono attivi nello stato logico zero.

Intendendo ora concentrare il discorso sull'uso dei dispositivi di I/U con l'Apple, rimandiamo gran parte della trattazione specifica ai prossimi paragrafi.

Riepilogo

A questo punto dovrebbe esservi ormai chiaro che il 6502 effettua i trasferimenti e le operazioni su otto bits di dato per volta. Calcoli ed operazioni complessi spesso richiedono più gruppi di 8 bits, cioè di bytes. I bytes vengono trasferiti verso la e dalla CPU del 6502 attraverso un bus di 8 bits.

Il 6502 utilizza un bus indirizzi a 16 bits per indirizzare le singole locazioni di memoria ed i dispositivi di I/U. Spesso il bus indirizzi è suddiviso in un bus d'indirizzi HI

ed uno d'indirizzi LO, di otto bits ciascuno. Il singolo segnale di controllo R/$\overline{W}$ controlla il flusso delle informazioni verso la e dalla CPU del 6502. In Tabella 1.1 compaiono i segnali e le loro denominazioni.

Bus dati	D7-D0	Serie di linee bidirezionali ad 8 bits per trasferire informazioni fra la CPU ed i dispositivi di I/U.
Bus indirizzi	A15-A0	Bus indirizzi monodirezionale a 16 bits per indirizzare sia la memoria, sia i dispositivi di I/U.
	A15-A8	Bus degl'indirizzi HI, cioè gli otto bits d'indirizzi più significativi.
	A7-A0	Bus degl'indirizzi LO, cioè gli otto bits d'indirizzi meno significativi.
Segnale di controllo	R/$\overline{W}$	Segnale di controllo della lettura/scrittura.

N.B.: La notazione con la "sopralineatura" (W) indica che lo zero logico è lo stato "attivo", lo stato cioè che fa sì che l'azione corrispondente si verifichi.
Nel caso dei segnali numerati, il numero cresce in funzione del peso dei bits: ad esempio, A15 è il bit d'indirizzo più significativo (MSB).

Tabella 1.1 — Segnali di controllo usati per l'interfacciamento.

LE ISTRUZIONI SOFTWARE PER IL CONTROLLO DELL'I/U

I comandi di I/U

Il calcolatore Apple prevede un certo numero d'istruzioni per il controllo dei dispositivi di I/U. Per lo più, comunque, queste istruzioni sono usate per controllare specifici dispositivi di I/U, o per svolgere funzioni specifiche. Per quanto probabilmente non abbiate realizzato ancora nulla, già conoscete abbastanza alcune, se non tutte, queste istruzioni. Comunque, per rinfrescarvi la memoria, abbiamo introdotto degli esempi particolari su queste istruzioni.

Probabilmente conoscete i comandi di INPUT e di PRINT. Il comando di INPUT fa sì che il programma in BASIC si arresti, e aspetti che vengano introdotti dei dati da tastiera. Il comando di PRINT fa "stampare" sullo schermo TV una risposta, o una stringa di caratteri.

ESEMPIO 1.1 — Semplice programma di I/U

```
10 INPUT "IL VALORE DI X ="; X
20 PRINT "IL VALORE INTRODOTTO ERA"; X
```

Se si esegue il programma dell'Esempio 1.1, il valore associato alla variabile X dovrà venir introdotto nel calcolatore prima che il programma passi il controllo all'istruzione (*statement*) 20. Questi due tipi d'istruzioni d'ingresso/uscita si usano spesso allo scopo di permettere all'operatore d'introdurre un valore, e vederlo visualizzato. Per i comandi sia di INPUT che di PRINT esistono diverse varianti, ma l'esempio visto serve ad illustrare l'essenziale: certamente avrete già usato senza difficoltà le operazioni di I/U in programmi in BASIC.

È anche possibile che siate già al corrente dell'esistenza, in BASIC, di comandi di I/U di *visualizzazione grafica*: e cioè, ad esempio, HOME, PLOT X,Y e SCRN(X,Y). Il comando di HOME pulisce lo schermo e posiziona il cursore lampeggiante alla posizione di riposo, nell'angolo in alto a sinistra dello schermo TV. I comandi di PLOT e di SCRN richiedono l'impiego di "coordinate", indicanti dove un'operazione deve avvenire.

L'Esempio 1.2 mostra come dei semplici comandi di visualizzazione grafica vengano utilizzati in un breve programma. Il programma visualizza sullo schermo TV dei punti colorati, cangianti in modo casuale. Se si utilizza uno schermo TV in bianco e nero (b/n), i punti saranno di varie tonalità di grigio.

ESEMPIO 1.2 — Generatore di una combinazione casuale di colori mediante i comandi di I/U

```
10 GR
20 X = INT(40*RND(1)) + 1
30 Y = INT(40*RND(1)) + 1
40 COLOR = INT(15*RND(1)) + 1
50 PLOT X,Y
60 GOTO 20
```

Esistono poi altri due comandi, che forse non avete considerato come comandi di I/U: si tratta dei comandi di LOAD e di SAVE, che si usano per leggere e memorizzare programmi su cassetta. Entrambi fanno sì che abbia luogo una serie prefissata di operazioni di controllo del registratore a cassette. L'uso di questi comandi è immediato, per cui non ne diamo nessun esempio.

I rimanenti comandi di I/U sono costituiti dalle operazioni di IN # X e di PR # X, che sono associate a dispositivi di I/U particolari, coi quali si possono sostituire la tastiera e lo schermo TV. È importante non dimenticare che queste istruzioni di I/U sono proprie dell'Apple e del suo programma interprete in BASIC: saranno infatti prive di senso per altri calcolatori basati sul 6502, a meno che utilizzino il programma BASIC dell'Apple. Per di più, ogni istruzione è specificamente riferita ad un particolare dispositivo di I/U: ad esempio, il comando di HOME non avrà nessun effetto sul registratore a cassette, e *su nessun altro dispositivo di I/U.* Analogamente, il comando di INPUT controlla solo l'ingresso dei valori introdotti mediante la tastiera della console.

I comandi di I/U d'uso generale

Sebbene nel programma interprete di BASIC Integer dell'Apple esistano diversi comandi di I/U d'uso generale, abbiamo deciso, per questo libro, di usare il programma interprete di BASIC Applesoft, a nostro parere più flessibile. Chi volesse adattare il proprio Apple a questo programma potrà richiedere l'assistenza di un rivenditore di calcolatori Apple.

I due comandi per i dispositivi di I/U sono PEEK e POKE: POKE trasferisce i dati dal calcolatore ad un dispositivo esterno, PEEK da un dispositivo esterno al calcolatore. Per queste istruzioni esiste un formato specifico, che bisogna rispettare se si vuole che le istruzioni operino nel modo esatto.

Per i dispositivi d'ingresso e di uscita useremo il termine *porte*; quindi un dispositivo di uscita sarà una *porta di uscita*, un dispositivo d'ingresso sarà una *porta d'ingresso.* Questa è la terminologia standard in uso nell'industria dei microcalcolatori.

L'istruzione di uscita, POKE, deve specificare l'*indirizzo* del dispositivo di I/U interessato dal trasferimento di dati, ed il valore che dev'essere trasferito al dispositivo indirizzato. Il formato effettivo dell'istruzione di POKE è POKE *x*, *y*, dove il valore *x* rappresenta l'indirizzo *decimale* del dispositivo di uscita che deve ricevere il valore del dato, che è *y*. Anche il dato *y* dev'essere espresso da un numero decimale. Dal momento che il microprocessore 6502 può indirizzare 65536 locazioni di memoria, l'indirizzo dev'essere compreso fra 0 e 65535 incluso. Il valore del dato dev'essere compreso fra 0 e 255 incluso, perché il calcolatore utilizza un bus dati di 8 bits per tutti i trasferimenti, ed il numero massimo che può venir trasferito attraverso tale bus è appunto 255.

Così, ad esempio, nell'istruzione POKE 12684,215 il valore 215 è inviato alla porta di uscita 12684.

L'istruzione d'ingresso, PEEK, è analoga all'istruzione di POKE, solo che nel comando non compaiono valori di dati. Dal momento che c'interessa determinare il valore presente in uno specifico dispositivo d'ingresso, viene specificato soltanto l'indirizzo decimale di quel dispositivo d'ingresso: scriveremo quindi PEEK (*x*), ove *x* è l'indirizzo decimale del dispositivo d'ingresso.

Serve a ben poco introdurre un valore senza usarlo per qualcosa; perciò il comando d'ingresso è sempre inserito in uno statement completo, e non costituisce uno statement a sé: ad esempio,

```
Q = PEEK(34579)
```

Qui alla variabile Q è assegnato il valore decimale che è stato introdotto dal dispositivo 34579.

Importante: non dimenticate di mettere fra parentesi l'indirizzo del dispositivo d'ingresso.

Con il comando di PEEK, il valore introdotto dev'essere compreso fra 0 e 255 incluso, sempre per le limitazioni imposte dal trasferimento su 8 bits.

POKE 45124,98 POKE N,120 POKE 45124,X POKE X,M	L = PEEK(23109) L = PEEK(Q)

Tabella 1.2 — Comandi d'ingresso (PEEK) e di uscita (POKE) corretti.

Nei comandi d'ingresso e di uscita possono essere specificate delle variabili, invece di valori particolari, per indicare gl'indirizzi delle porte, ed i valori dei dati relativamente al comando di POKE. Quindi tutti i comandi di PEEK e di POKE che compaiono nella Tabella 1.2 sono leciti. È chiaro che abbiamo presupposto che i valori delle variabili N, M, X e Q siano già stati specificati in un punto del programma precedente all'esecuzione delle istruzioni riportate nella Tabella 1.2.

I comandi d'ingresso e di uscita in cui i valori degl'indirizzi superino 65535 genereranno da parte dell'Apple il messaggio ILLEGAL QUANTITY ERROR (cioè ERRORE PER QUANTITA' NON LECITA). Il medesimo messaggio viene generato se si cerca d'inviare in uscita un valore relativo ad un dato numerico superiore a 255.

Abbiamo dato qualche esempio d'uso dei comandi di PEEK e di POKE. Ma, anche se i programmi dell'Esempio 1.3 sono eseguibili, non faranno nulla di utile, visto che per il momento al vostro calcolatore non è collegata nessuna porta esterna di I/U.

ESEMPIO 1.3 — Semplici programmi di I/U con comandi di PEEK e di POKE

```
10  INPUT "PORTA DI USCITA # ="; P
20  INPUT "VALORE USCITO"; V
30  POKE P,V
40  GOTO 10
```

```
10  INPUT "PORTA DI INGRESSO # ="; M
20  PRINT "VALORE INTRODOTTO ="; PEEK(M)
30  GOTO 10
```

Dal momento che i calcolatori basati sul 6502 non sono in grado di distinguere fra le locazioni di memoria utilizzate per la memorizzazione temporanea di programmi e dati, e quelle utilizzate per le porte di I/U, spesso le istruzioni di PEEK e di POKE vengono usate per esaminare e modificare il contenuto di svariate locazioni di memoria dell'Apple. Se, con un'operazione di POKE, introducete delle informazioni nella memoria di lettura/scrittura in maniera indiscriminata, vi può capitare di "scrivere sopra" parti importanti del vostro programma, oppure su informazioni che sono state temporaneamente memorizzate dall'interprete BASIC. Ne deriva una "catastrofe" per il calcolatore, per cui i vostri programmi e dati andranno persi o saranno alterati in modo significativo. Certo non è opportuno scrivere a caso informazioni a diversi indirizzi mediante il comando di POKE, senza specifiche direttive. È chiaro che si può usare il comando di PEEK per esaminare il contenuto di una locazione di me-

moria ogni volta che si vuole, perché questo comando non altera il contenuto della locazione di memoria esaminata.

Da quanto si è detto a proposito dei dispositivi di memoria, avrete certamente capito che l'operazione di POKE non ha nessun effetto sulle memorie di sola lettura (ROM) dell'Apple.

Le mappe di memoria

A questo punto è opportuno parlare brevemente delle "mappe" degl'indirizzi di memoria usati dall'Apple. In Figura 1.3 è rappresentata la mappa completa della memoria di 64 K. Per comodità gl'indirizzi di memoria sono riportati sia in notazione decimale (in base 10), sia in notazione esadecimale (in base 16). I numeri esadecimali hanno il suffisso H, per distinguerli dai numeri decimali.

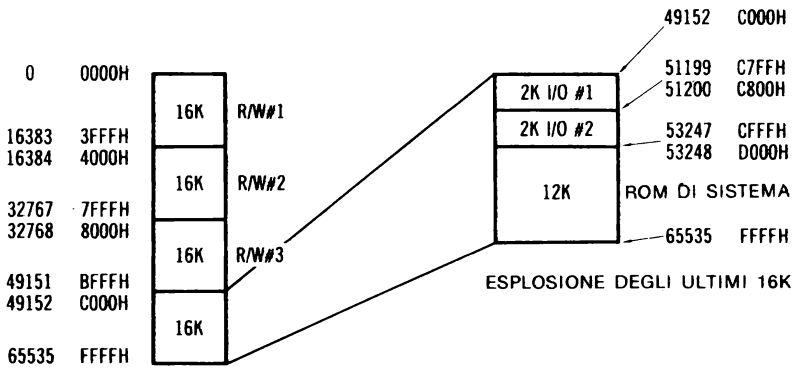


Figura 1.3 — Mappa della memoria di 64 K del calcolatore Apple.

Nell'Apple lo spazio di memoria è diviso in blocchi di 16 K. Tre di questi blocchi sono riservati alla memoria RAM, e nella maggior parte degli Apple il primo blocco di RAM è "occupato" dai chips di memoria di lettura/scrittura. Gli altri blocchi di RAM possono essere utilizzati per espansioni future della memoria RAM eventualmente richieste da applicazioni particolari. Abbiamo verificato che nella maggior parte dei casi bastano 16 K di memoria RAM. Molti fornitori offrono kits di chips di memoria supplementare, e certamente la maggior parte degli utenti di Apple non avrà eccessive difficoltà a corredare il proprio sistema di ulteriori chips di memoria.

L'ultimo blocco di memoria di 16 K è riservato all'indirizzamento tanto della ROM quanto delle porte di I/U. 12 dei 16 K del blocco sono occupati dalle ROM di sistema dell'Apple, e comprendono l'interprete BASIC ed i programmi di monitor. I restanti 4 K sono suddivisi a loro volta in due spazi di 2 K ciascuno, l'uno per l'indirizzamento dell'I/U, l'altro per espansioni future dell'Apple. Il blocco di I/U, con indirizzi che vanno da C000H a C7FFH incluso, è uno dei più importanti ai fini dell'interfacciamento, in quanto specificatamente riservato a questo scopo, e non utilizzabile, nell'Apple, per nessun altro scopo. Alcuni indirizzi di questo blocco di 2 K sono utilizzati

dall'Apple per controllare ad esempio l'altoparlante, la tastiera ed il registratore a cassette. Le effettive assegnazioni d'indirizzo sono riportate nella Tabella 1.3. Per un discorso particolareggiato sull'utilizzo pratico di questi indirizzi di I/U rimandiamo al *BASIC Programming Reference Manual* ed all'*Apple II Reference Manual*, che vengono forniti insieme con l'Apple II, ma possono anche venire richiesti direttamente alla Apple Computer, Inc., 10260 Bandley Dr., Cupertino, CA 95014.

L'ultimo blocco di memoria di 2 K, C800H-CFFFH, è riservato alle espansioni future. Potete usare questo spazio per un'ulteriore memoria di sola lettura, nel caso di programmi lunghi che si desidera avere subito disponibili.

	Indirizzo	
Funzione	**Decimale***	**Esadecimale**
Dato da tastiera	49152	**C000**
Strobe di azzeramento tastiera	49168	**C010**
Altoparlante	49200	**C030**
Uscita di cassetta	49184	**C020**
Ingresso di cassetta	49256	**C060**
Ingressi di flag	49249-49251	**C061-C063**
Ingressi analogici	49252-49255	**C064-C067**
Azzeramento analogico	49264	**C070**
Strobe di utilità	49216	**C040**

* Diamo soltanto indirizzi positivi. Per calcolare gl'indirizzi negativi, sommare −65536 agl'indirizzi decimali riportati.

Tabella 1.3 — Indirizzi di I/U dell'Apple e loro uso.

Nei prossimi capitoli descriveremo in dettaglio l'uso concreto degl'indirizzi di I/U. Per ora è sufficiente sapere che uno specifico gruppo d'indirizzi di memoria è riservato alle vostre applicazioni particolari. Un'altra cosa da tener presente è che la mappa di memoria rappresentata in Figura 1.3 è propria del calcolatore Apple. Gli altri calcolatori basati sul 6502 potranno avere mappe di memoria diverse, nelle quali la memoria RAM, la memoria di sola lettura e gl'indirizzi dei dispositivi di I/U saranno collocati in aree differenti della mappa.

I comandi software ed i circuiti d'interfaccia

A questo punto avrete certamente capito che le istruzioni di PEEK e di POKE determinano il verificarsi di certe azioni, relative ai dispositivi di I/U ed alle locazioni di memoria, come diretto risultato del loro utilizzo. Istruzioni come A = 1.359 faranno sì che determinati valori siano memorizzati, ma non sappiamo quali locazioni di memoria l'Apple ha assegnato alla variabile "A", né in che modo il valore 1.359 è stato memorizzato. Le istruzioni di PEEK e di POKE determinano il verificarsi di una se-

quenza nota e ben definita di operazioni: trasferimento di bytes di dati, generazione di segnali di controllo, e trasferimento d'informazioni d'indirizzi sulle linee del bus indirizzi. Queste azioni, ben definite e riproducibili, ci permettono di utilizzare tali comandi per controllare i dispositivi di I/U. Vedremo ora quali azioni sono determinate da ciascuno di questi comandi software.

Le istruzioni di PEEK e di POKE operano in modo molto simile. Entrambe specificano un indirizzo che richiede 16 bits d'informazione. Durante la loro esecuzione l'informazione *d'indirizzo* contenuta nel comando viene trasferita ai dispositivi esterni attraverso le linee del bus indirizzi A15-A0. Così l'indirizzo del dispositivo di I/U è disponibile per tutti i dispositivi ed i circuiti connessi con queste linee d'indirizzi, cioè sia per la memoria, sia per i dispositivi di I/U.

Quando in un programma si usa un'istruzione di POKE, anche il valore del dato viene fatto uscire dal chip del 6502, ma lungo le linee del bus dati D7-D0. Una volta che i bits dei dati ed i bits degl'indirizzi sono "stabili", ossia presenti sui loro rispettivi bus in una forma utilizzabile, il 6502 manda il segnale READ/$\overline{\text{WRITE}}$ sul bus di controllo: questo segnale sincronizza l'acquisizione del dato da parte del dispositivo di I/U che è stato indirizzato. Naturalmente è necessaria una circuiteria esterna che "catturi" il dato, identifichi il dispositivo di I/U selezionato e lo sincronizzi con il sistema basato sul 6502. In Figura 1.4 è rappresentato il diagramma di temporizzazione di questi segnali, così come si configurano in un sistema con 6502, in questo caso l'Apple. Ovviamente il comando di POKE implica molte istruzioni in linguaggio assemblatore, ed il diagramma di temporizzazione mostra soltanto quel che accade durante l'effettivo trasferimento dei dati. E fin qui abbiamo visto solo quel che il 6502 fa durante un'operazione di POKE.

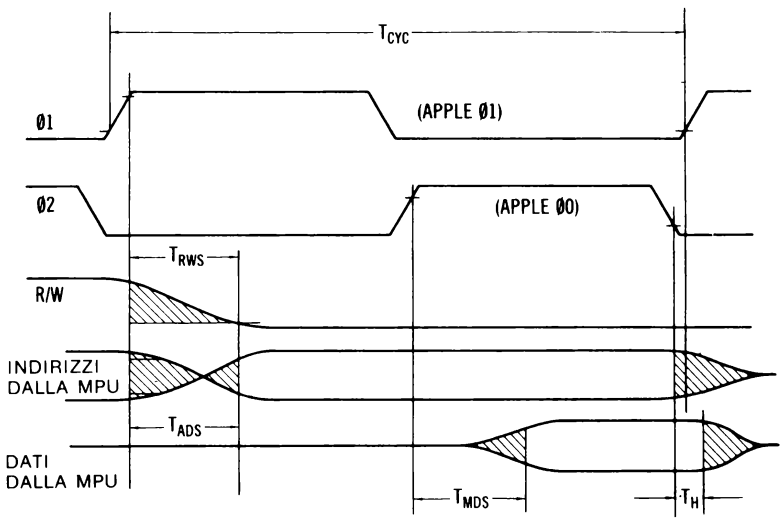


Figura 1.4 — Relazioni fra i segnali durante un'operazione di scrittura (v. Appendice C).

Quando viene eseguita un'istruzione di PEEK, i dati non compaiono in essa, ma vengono acquisiti da un dispositivo esterno di I/U: l'unico ad essere specificato è l'indirizzo. Al momento dell'esecuzione dell'istruzione di PEEK, l'indirizzo di 16 bits viene collocato sulle linee del bus indirizzi. Quando l'informazione d'indirizzo è presente, il dispositivo di I/U corrispondente collocherà il proprio dato sul bus dati, in modo che possa essere acquisito dal processore 6502. In un'operazione di lettura, il segnale R/$\overline{W}$ proveniente dal 6502 è un uno logico. In questo caso sono anche richiesti degli ulteriori circuiti per la selezione del dispositivo di I/U e per condizionare l'invio dei propri dati sul bus dati. In Figura 1.5 compare un tipico diagramma di temporizzazione relativo al comando di PEEK.

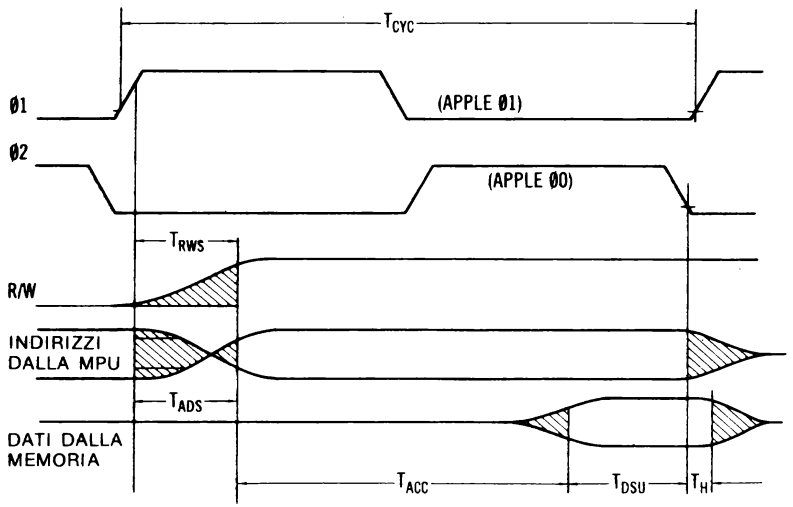


Figura 1.5 – Relazioni fra i segnali durante un'operazione di lettura (v. Appendice C).

Fra poco descriveremo brevemente alcuni dei circuiti usati per le porte d'ingresso e di uscita. Certamente avrete capito che, anche se abbiamo descritto le *porte di I/U* come qualcosa che può tanto ricevere dati uscenti dal microcalcolatore, quanto trasmettere dati che vengono poi acquisiti dal microcalcolatore, alcuni *dispositivi di I/U* possono di fatto contenere un certo numero di porte di I/U particolari. Le unità di controllo, i dispositivi per la memorizzazione di dati (dischi, cassette), i convertitori analogici ed altri dispositivi di I/U possono avere un certo numero di porte di I/U, perché possono richiedere più di 8 bits d'informazione da parte del calcolatore, e possono anche aver bisogno di trasferire al calcolatore più di 8 bits d'informazione. In ogni caso, i trasferimenti di dati costituiti da più di 8 bits implicano sempre il trasferimento di più bytes verso/da il calcolatore, e porte di I/U di 8 bits particolari. Non dimenticate mai che *l'informazione è sempre trasferita ad 8 bits per volta.*

I comandi software. Trasferimento e controllo dei dati

Per lo più i comandi di PEEK e di POKE si usano per trasferire valori di dati di 8 bits fra i dispositivi di I/U o le locazioni di memoria ed il calcolatore 6502. Come si è già detto, alcuni trasferimenti di dati richiederanno più di 8 bits d'informazione, e allora saranno trasferiti più bytes, uno alla volta.

Ci sono anche dei casi in cui il *valore* reale del dato trasmesso non è significativo. I bits possono essere usati per rappresentare particolari condizioni a due stati, che non sono in rapporto con i valori posizionali dei bits. Ad esempio, l'Apple può essere collegato ad un certo numero di sensori indicanti condizioni come serbatoio vuoto o pieno, riscaldamento acceso o spento, valvola aperta o chiusa, e così via. Si potrà usare un comando di PEEK per introdurre lo stato di questi bits indicatori mediante una porta d'ingresso ad 8 bits. Allora il valore letto da questa porta d'ingresso potrà essere 100_{10}, ma, dato che la porta *rileva* otto singoli stati di aperto o chiuso (cioè uno logico o zero logico), il valore 100_{10} non è significativo: ciascun bit binario rappresenta lo stato di un singolo sensore. In questo caso

$$100_{10} = 01100100_2$$

Ciò vuol dire che tre sensori si trovano nello stato logico 1, e cinque nello stato logico zero.

I comandi di POKE e di PEEK si possono usare anche, in modo analogo, per attivare o disattivare un dispositivo, sulla base dello stato dei singoli bits rilevati in un altro punto del programma di controllo. Infatti molti degl'indirizzi di I/U usati nell'Apple sono associati a semplici dispositivi caratterizzati da un funzionamento di "aperto/chiuso", come gli altoparlanti. Quindi il semplice comando

```
A = PEEK(49200)
```

genererà un "blip" da parte dell'altoparlante dell'Apple. Capite bene che la variabile A è una variabile "fittizia", e che il suo valore finale non ha importanza, perché l'effetto di questo semplice statement BASIC è quello d'inviare all'altoparlante un impulso. Il comando di controllo dell'altoparlante può essere inserito in un ciclo, ottenendo così un debole ronzio da parte dell'altoparlante: v. Esempio 1.4.

ESEMPIO 1.4 — Semplice programma di controllo dell'altoparlante

```
10  A = PEEK(49200)
20  GOTO 10
```

Qui la cosa più importante da ricordare è che le istruzioni di PEEK e di POKE non si limitano semplicemente a controllare il trasferimento d'informazioni sul bus dati, ma si possono usare anche per funzioni di controllo specifiche, come l'invio d'impulsi ad un contatore, l'attivazione di una pompa, o l'inclinazione di un collettore solare.

Linguaggio assemblatore e BASIC

I programmi in linguaggio BASIC che voi scrivete sul vostro Apple hanno ben poco a che vedere con le istruzioni che il microprocessore 6502 è effettivamente in grado di eseguire. Ciascuno statement e comando del BASIC viene *interpretato* dall'*interprete* BASIC residente nel calcolatore Apple. Anche il manuale di programmazione del microprocessore 6502 avrà ben poco a che vedere con il manuale software dell'Apple: i comandi infatti sono molti diversi.

Il 6502 non ha un comando di PRINT, per cui non è in grado di eseguire l'operazione seguente:

```
PRINT "QUESTO SEMBRA UNO SCHERZO"
```

L'interprete BASIC stabilisce che deve aver luogo un'operazione di PRINT, e poi esegue una serie d'istruzioni di un programma in linguaggio assemblatore, che provvedono a collocare nella memoria associata allo schermo i codici dei caratteri alfabetici che formano le parole "QUESTO SEMBRA UNO SCHERZO". Le istruzioni del linguaggio assemblatore sono costituite da uni e zeri logici, che determinano il verificarsi delle operazioni del 6502, interne ed esterne, necessarie per il trasferimento della parte messaggio del comando di PRINT alla memoria associata allo schermo.

Anche se in tutto il libro non programmeremo mai in linguaggio assemblatore, è necessario sapere che è il linguaggio "base" del calcolatore quello che fa funzionare l'Apple così come funziona.

I comandi di PEEK e di POKE determinano l'esecuzione di moltissimi comandi in linguaggio assemblatore, al fine di ottenere come risultato il trasferimento dei dati. Visto che queste istruzioni in linguaggio BASIC devono essere interpretate, anche quando vengono usate l'una subito dopo l'altra, o all'interno di un ciclo, il processo software d'*interpretazione* sarà necessariamente lento. Nell'Esempio 1.5 presentiamo due programmi, entrambi di controllo dell'altoparlante dell'Apple; le due sequenze fanno la stessa cosa, e cioè generano un suono dell'altoparlante. Solo ascoltando le differenze fra i due suoni prodotti potrete apprezzare la diversa velocità di esecuzione dei due programmi.

ESEMPIO 1.5 — Confronto fra due programmi di controllo dell'altoparlante, l'uno in linguaggio assemblatore, l'altro in BASIC

Programma in BASIC

```
10  A = PEEK(49200)
20  GOTO 10
```

Programma in linguaggio assemblatore

```
GO    LDY # $C0
LOOP  LDA # $0C
      JSR WAIT
      LDA SPKR
      DEY
      BNE LOOP
      JMP GO
```

Il programma in linguaggio assemblatore genera un suono gradevole e uniforme, quello in BASIC invece un debole brontolio. Il programma in linguaggio assemblatore è simile a quello impiegato dal programma Monitor dell'Apple, ed inoltre usa la subroutine interna WAIT per produrre un ritardo.

In certi casi i programmi in linguaggio assemblatore sono più convenienti dei programmi in BASIC in una proporzione di *cinquecento contro uno*; i programmi in BASIC d'altra parte sono indubbiamente più facili da scrivere e mettere a punto. Generalmente la programmazione in linguaggio assemblatore non è consigliabile per i principianti.

Noi ci occuperemo molto poco di programmazione in linguaggio assemblatore, essendo il nostro discorso incentrato sulla programmazione in linguaggio BASIC. Per ulteriori informazioni sulla programmazione in linguaggio assemblatore con il 6502, consigliamo la lettura di *6502 Software Design*, e di *Programming & Interfacing the 6502, With Experiments*, entrambi della Howard W. Sams & Co., Inc., Indianapolis, IN 46268.

Numerazione binaria e numerazione decimale

Il calcolatore Apple accetta, elabora e stampa numeri decimali (cioè in base 10). Questo fatto lo rende compatibile con il metodo di numerazione usato normalmente dalla gente, per la quale sarebbe complicato capire e convertire rapidamente valori di dati stampati in un formato diverso dal decimale. Le linee dei dati e degl'indirizzi sono collegate direttamente con il microprocessore 6502, per cui i dati relativi sono binari, avendo due soli stati: uno logico e zero logico. Quindi dobbiamo renderci conto che, quando specifichiamo l'indirizzo di una porta di I/U in un comando di PEEK o di POKE, l'indirizzo (0-65535) comparirà sul bus indirizzi nella sua forma binaria (0000000000000000-1111111111111111). Allora dovremmo essere capaci di eseguire nei due sensi la conversione decimale-binario!

Allo stesso modo, anche i valori dei dati trasferiti al e dal calcolatore tramite i comandi di PEEK e di POKE vengono indicati ed acquisiti sotto forma di valori binari di 8 bits, perché il bus dati ha una "larghezza" di soli 8 bits. Il bus dati da 8 bits dipende dalla capacità di elaborazione dei dati propria del 6502, *non* dall'Apple. Quindi dobbiamo limitarci a trasferire solo dati di 8 bits. È una grossa limitazione? In genere no. Infatti, malgrado ciò, l'Apple è in grado di elaborare grosse quantità d'informazioni, e, come si vedrà più avanti, questo fatto rende facile l'interfacciamento con i dispositivi di I/U.

Prima di chiudere il capitolo è necessaria un'ultima osservazione sugl'indirizzi. L'interprete BASIC del calcolatore Apple è stato concepito in modo che possa lavorare su indirizzi *sia negativi che positivi.* Questo non vuol dire che nel calcolatore ci siano realmente degl'indirizzi negativi! Riuscireste ad immaginare dei numeri civici negativi? I numeri negativi sono dovuti semplicemente al modo in cui nell'Apple sono memorizzati gli equivalenti *binari* degl'indirizzi. Così l'indirizzo 49200, relativo all'altoparlante, è equivalente a −16336. Per evitare confusioni, consigliamo decisamente di usare indirizzi positivi. Potrete attuare facilmente la conversione fra indiriz-

zi negativi e positivi semplicemente sommando 65536 ad un indirizzo negativo per ottenere il positivo equivalente, e sottraendo 65536 dall'indirizzo positivo per ottenere l'equivalente negativo. Entrambi gl'indirizzi, 49200 e −16336, generano lo stesso indirizzo di 16 bits, ma pensiamo che converrete sul fatto che gl'indirizzi negativi possono sembrare alquanto astratti e sconcertanti.

CAPITOLO 2

L'INTERFACCIAMENTO DELL'APPLE

A questo punto probabilmente vi porrete delle domande, e cioè:

- Come, concretamente, l'Apple trasferisce le informazioni ai dispositivi di I/U?
- Come, concretamente, i dispositivi di I/U si sincronizzano con le operazioni del calcolatore?
- Come vengono selezionati, e identificati, i singoli dispositivi di I/U?
- Come i dispositivi di I/U collocano i loro dati sul bus dati, e come li ricevono dal bus dati?

Si tratta di domande fondamentali, perché le risposte a questi quesiti vi forniranno le basi per capire in che cosa consista l'interfacciamento con un microcalcolatore. A queste domande risponderemo in questo e negli altri capitoli. Introdurremo inoltre degli esperimenti, che daranno corpo ai principi teorici in forma sperimentale.

In questo capitolo daremo qualche esempio di circuiti digitali. Abbiamo dato per scontato che sappiate "leggere" ed interpretare lo schema di un circuito logico, e che conosciate i più comuni circuiti in logica TTL (transistor-transistor logic) della serie SN7400.

DECODIFICA DEGL'INDIRIZZI DEI DISPOSITIVI DI I/U

Per poter parlare del trasferimento delle informazioni fra i dispositivi di I/U ed il calcolatore, dobbiamo prima conoscere la circuiteria ed i segnali usati per identificare e indirizzare i singoli dispositivi di I/U. Si possono usare molte tecniche, e ne esamineremo diverse; ma è impossibile illustrare tutti i possibili schemi d'interfacciamento dei dispositivi di I/U, per il fatto che molte modifiche si attuano per soddisfare necessità particolari.

Quando il calcolatore Apple viene programmato per eseguire un trasferimento di dati mediante uno dei due comandi di I/U di uso generale PEEK e POKE, il proces-

sore 6502 genera dei segnali destinati a sincronizzare il flusso dei dati. A questo punto quello che c'interessa di più è come si utilizzano le linee del bus indirizzi, che sono le 16 linee che indirizzano le singole locazioni di memoria ed i dispositivi di I/U. Ricorderete che le istruzioni di PEEK e di POKE contengono ciascuna l'informazione dell'indirizzo decimale, che serve ad identificare la locazione di memoria o il dispositivo di I/U indirizzato. Chiaramente, il calcolatore Apple non ha la facoltà di distinguere fra una locazione di memoria ed una porta di I/U.

INDIRIZZAMENTO DEI DISPOSITIVI

Tutti i dispositivi di I/U destinati ad essere utilizzati con il calcolatore devono essere in grado di riconoscere il proprio indirizzo. Dal momento che i comandi di PEEK e di POKE utilizzano indirizzi a 16 bits, i dispositivi di I/U devono "sorvegliare" queste 16 linee d'indirizzi, cioè le linee A15-A0, per l'evenienza dei propri indirizzi. I circuiti dei dispositivi di I/U possono rilevare la presenza di uno specifico indirizzo fondamentalmente secondo tre tecniche, che sono:

- *condizionamento logico*: rileva *una* particolare combinazione di segnali logici;
- *decodifica*: uno schema più flessibile del precedente, nel quale possono venir rilevati più indirizzi;
- *confronto*: un indirizzo assegnato o noto è paragonato con i segnali relativi al bus indirizzi, fino al verificarsi di un'uguaglianza.

Sono pure possibili delle combinazioni delle tre tecniche, ed anche diverse varianti. Noi daremo degli esempi per ciascuna delle tre tecniche di base.

Uso delle porte per la decodifica degl'indirizzi

Con la tecnica della decodifica degl'indirizzi dispositivo che prevede l'uso di porte specifiche, l'indirizzo dev'essere noto, affinché le porte possano essere configurate opportunamente. In quest'esempio supporremo che l'indirizzo del dispositivo sia 1010100011110111_2, ovvero 43255_{10}. La notazione binaria è lunga e abbastanza scomoda, e quindi vi sentirete certo più a vostro agio con l'equivalente esadecimale, A8F7H. Essendo le porte NAND/AND il tipo predominante attualmente disponibile per la realizzazione di condizioni logiche, nei nostri schemi logici utilizzeremo questi tipi di circuiti.

In Figura 2.1, al fine di rinfrescarvi la memoria, abbiamo evidenziato le configurazioni dei pins relative ad alcuni tipi di porte AND/NAND; nella Tabella 2.1 riportiamo la tabella della verità generalizzata relativa ad una porta di AND a due ingressi ed alla porta di NAND equivalente. Dal momento che spesso nei circuiti d'indirizzamento di un dispositivo si trovano inverters del tipo SN7404, in Figura 2.1 abbiamo inserito anche la configurazione dei pins relativa a questo chip. Anche le tabelle della verità riprodotte nella Tabella 2.1 mostrano la funzione di un inverter. *In tutti i casi lo stato*

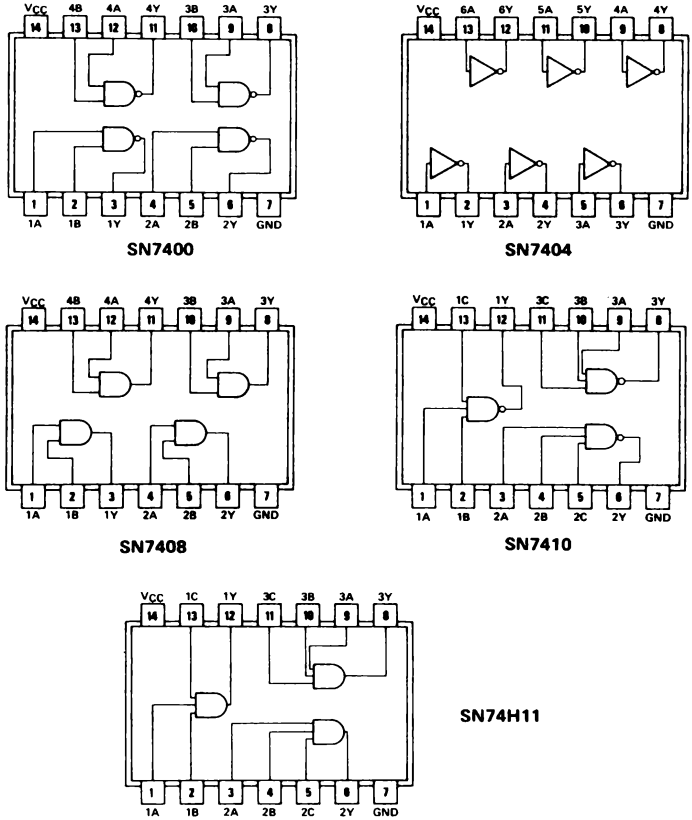


Figura 2.1 — Configurazione dei pins relativi ad un inverter ed a varie porte AND/NAND.

logico uno corrisponde alla tensione più alta (da +2.8 a +5 volts), mentre lo stato logico zero corrisponde alla tensione più bassa (da 0.0 a 0.8 volts). Le funzioni della porta NAND sono disponibili con gl'ingressi 2, 3, 4 ed 8, quelle della porta AND con gl'ingressi 2, 3 e 4.

Dal momento che l'unico stato di uscita, che è un uno logico per le porte AND ed uno zero logico per le porte NAND, si ha soltanto quando *tutti* gl'ingressi di una porta AND o di una porta NAND sono degli uni logici, dovremo configurare l'indirizzo binario 1010100011110111_2 in modo che generi 16 uni logici all'ingresso di una porta AND o NAND, quando è presente sul bus indirizzi a 16 bits. Avete probabilmente capito che non si trovano in commercio porte AND e NAND a 16 ingressi, per cui al loro posto si dovrà utilizzare qualche altra configurazione: ad esempio, non sarà un problema usare una distinta porta NAND ad 8 ingressi per rilevare una configurazione di bits binari d'indirizzo sul bus d'indirizzo alto (A15-A8), ed un'altra porta NAND

Porta AND		Porta NAND		Inverter	
Ingressi	Uscita	Ingressi	Uscita	Ingresso	Uscita
A B	Q	A B	Q	A	Q
0 0	0	0 0	1	0	1
0 1	0	0 1	1	1	0
1 0	0	1 0	1		
1 1	1	1 1	0		

Tabella 2.1 — Tabelle della verità relative ad una porta AND a due ingressi, ad una porta NAND e ad un inverter.

ad 8 ingressi per rilevare una configurazione di bits binari sul bus d'indirizzo basso (A7-A0). Semplici funzioni d'inverter sono usate per invertire i bits d'indirizzo posti a zero logico, affinché vengano applicati degli uni logici agl'ingressi della porta corrispondente, come si vede in Figura 2.2. In questo circuito sono stati usati due inverters ed una porta NAND, allo scopo di raggruppare le uscite di ciascuna porta ad 8 ingressi, in modo che l'uscita del circuito sarà uno zero logico soltanto quando *tutta* la configurazione di 16 bits 1010100011110111_2 viene rilevata sul bus indirizzi a 16 bits.

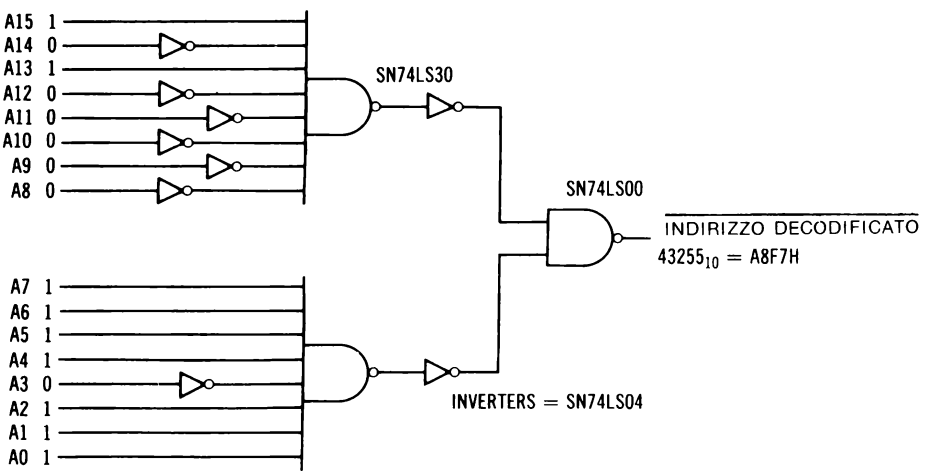


Figura 2.2 — Circuito logico di decodifica dell'indirizzo 43255, o A8F7H.

Uno dei principali inconvenienti di questo circuito è che alcuni segnali d'indirizzo devono passare attraverso quattro porte prima di uscire come indirizzo decodificato dalla porta NAND a 2 ingressi. E, dal momento che ciascuna porta carica il segnale di un leggero ritardo, questo fatto può provocare nel circuito qualche problema di temporizzazione. In realtà, i ritardi sono abbastanza trascurabili, e per il momento li ignoreremo. Comunque il ritardo può essere alquanto ridotto utilizzando nel circuito

una porta NOR o OR per raggruppare le uscite delle due porte NAND ad 8 ingressi: questa anzi è una buona regola di progettazione.

Le porte NOR ed OR sono facilmente procurabili, e sono largamente usate nell'interfacciamento dei calcolatori. Una tipica porta NOR ed una di OR sono rappresentate in Figura 2.3; nella Tabella 2.2 troverete le corrispondenti tabelle della verità.

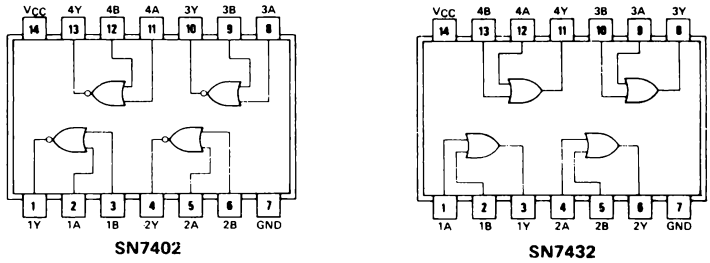


Figura 2.3 — Configurazione dei pins di un tipico circuito integrato di porte NOR ed OR.

Porta NOR		Porta OR	
Ingressi	**Uscita**	**Ingressi**	**Uscita**
A B	**Q**	**A B**	**Q**
0 0	1	0 0	0
0 1	0	0 1	1
1 0	0	1 0	1
1 1	0	1 1	1

Tabella 2.2 — Tabelle della verità relative ad una porta NOR e ad una porta OR a due ingressi.

La tecnica di condizionamento logico di Figura 2.2, pur essendo efficace nella decodifica di un singolo indirizzo e relativamente economica, è scarsamente flessibile. Un metodo più flessibile è rappresentato in Figura 2.4.

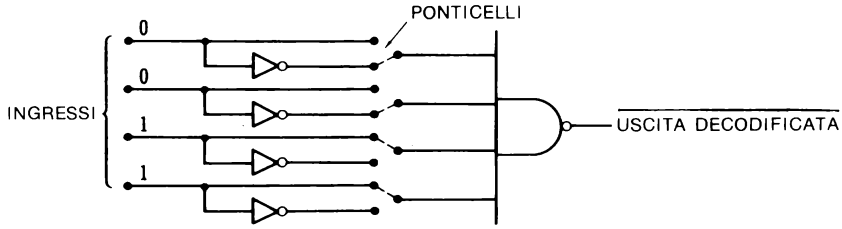


Figura 2.4 — Una semplice porta a quattro ingressi può venir programmata per ingressi logici 1 e 0.

In questo circuito è impiegata una tecnica di condizionamento logico nella quale gl'inverters possono essere usati per invertire, a seconda delle necessità, singoli bits d'indirizzo. Comunque i bits possono anche essere utilizzati non invertiti. I ponticelli permettono di assegnare l'indirizzo di dispositivo, come si vede in Figura 2.5, nella quale per chiarezza abbiamo rappresentato solo le condizioni logiche del bus relativamente alla parte bassa dell'indirizzo. Per le linee di bus relative alla parte alta dell'indirizzo è necessario un equivalente circuito logico. In questo tipo di circuito logico può essere selezionato uno qualunque dei 65536 possibili indirizzi, ma uno solo alla volta.

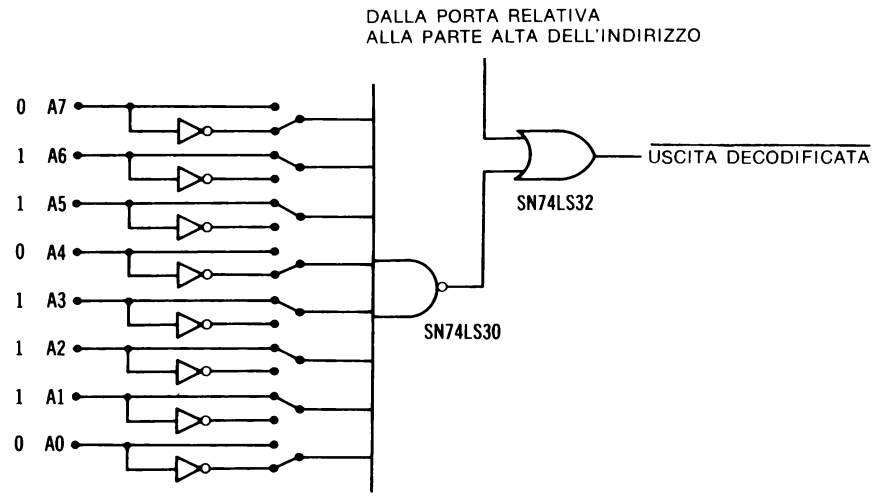


Figura 2.5 — Porta programmabile per la decodifica degl'indirizzi di dispositivo. (Il circuito relativo alla parte alta dell'indirizzo è equivalente.)

Il circuito con porte logiche programmabile assicura un'ampia flessibilità, in quanto gl'indirizzi possono venir facilmente cambiati al fine di soddisfare esigenze specifiche d'interfacciamento, ma d'altra parte un tale circuito può selezionare soltanto un unico indirizzo, e questa è una limitazione pesante. Quando si hanno più dispositivi di I/U posti sulla stessa scheda, ciascuno di essi richiede un proprio circuito per la decodifica dell'indirizzo. Ma questa limitazione è eliminabile con altre tecniche d'indirizzamento.

Purtroppo le tecniche di condizionamento logico che abbiamo illustrato non soddisfano tutte le necessità concernenti l'indirizzamento ed il controllo di un dispositivo di I/U. Ricorderete, da quanto si è detto nel Capitolo 1 sul segnale READ/$\overline{\text{WRITE}}$ (R/$\overline{\text{W}}$), che questo segnale serve per sincronizzare il flusso d'informazione diretto al e proveniente dal calcolatore. Anche i dispositivi di I/U devono ricorrere a questo segnale di controllo, se si vuole che utilizzino correttamente il bus dati. In molte interfacce progettate per calcolatori basati sul 6502, la linea R/$\overline{\text{W}}$ è usata per fornire l'impulso negativo di scrittura, mentre il segnale R/$\overline{\text{W}}$ è *invertito* per generare un di-

stinto impulso di lettura. I due segnali di controllo che ne risultano, $\overline{\text{WRITE}}$ ($\overline{\text{WR}}$) e $\overline{\text{READ}}$ ($\overline{\text{R}}$), sono di facile impiego nei circuiti d'interfaccia, perché sono attivi nello stato logico zero. L'uso di questi segnali è illustrato in Figura 2.6.

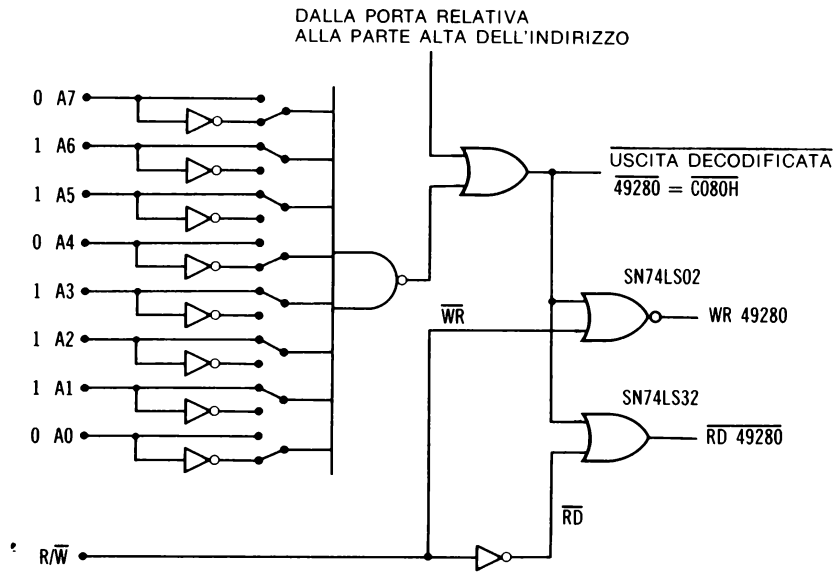


Figura 2.6 — Uso dei segnali $\overline{\text{RD}}$ e $\overline{\text{WR}}$ per generare impulsi di selezione dispositivo per la sincronizzazione delle relative operazioni.

In questo circuito l'uscita dal circuito che sente i 16 bits d'indirizzo è combinata con $\overline{\text{RD}}$ e $\overline{\text{WR}}$ allo scopo di fornire due segnali per il controllo delle porte di I/U. Questi due segnali di controllo sono una combinazione dell'indirizzo decodificato e dell'impulso $\overline{\text{WRITE}}$, ed una combinazione dell'indirizzo decodificato e dell'impulso $\overline{\text{READ}}$. L'impulso che ne risulta da ciascuna porta è detto *impulso di selezione indirizzo*, o *impulso di selezione dispositivo*. Più generalmente parlando, un *indirizzo decodificato* è condizionato da un *impulso di funzione* ($\overline{\text{RD}}$ o $\overline{\text{WR}}$), allo scopo di generare un *impulso di selezione dispositivo*. Nello schema del circuito di Figura 2.6, l'impulso $\overline{\text{RD 49280}}$ potrebbe essere usato per controllare una porta d'ingresso dati, mentre l'impulso WR 49280 potrebbe essere usato per controllare una porta di uscita dati. Si osservi che la notazione WR 49280 dell'impulso non ha la riga soprascritta: questo vuol dire che l'impulso è attivo nello stato logico uno, mentre l'impulso $\overline{\text{RD 49280}}$ è attivo nello stato logico zero. In quest'esempio è appropriato esprimere l'indirizzo relativo alle porte di I/U con un valore esadecimale, ad esempio $\overline{\text{RD C080H}}$.

Prima di proseguire, è necessario avere ben capito che un'*operazione di lettura* comporta l'*acquisizione dentro il calcolatore d'informazioni* provenienti da una porta d'ingresso, mentre un'*operazione di scrittura* implica il *trasferimento d'informazioni*

dal calcolatore ad un dispositivo esterno. È poi opportuno e utile usare un unico indirizzo per controllare una porta d'ingresso ed una porta di uscita: dato che gl'impulsi $\overline{RD}$ e $\overline{WR}$ non possono coincidere, non c'è conflitto fra una porta d'ingresso ed una porta di uscita a cui sia stato assegnato lo stesso indirizzo. Invece *non si può* assegnare lo stesso indirizzo a due porte d'ingresso, *né* a due porte di uscita. Di fatto potrete capire che, anche se ad una porta d'ingresso e ad una porta di uscita è stato assegnato lo stesso indirizzo, le due porte non possono essere in rapporto quanto alla funzione, e possono perciò venir utilizzate su distinti circuiti d'interfaccia.

I concetti svolti ed i circuiti base illustrati in questo paragrafo sono estremamente importanti, e saranno ulteriormente sviluppati negli altri paragrafi di questo stesso capitolo. È fondamentale che abbiate ben compreso l'uso dei segnali per la selezione dei dispositivi, di cui si è parlato. Finora non si è detto che cosa sono i dispositivi d'ingresso e uscita, e come operano; se ne parlerà nel prossimo capitolo.

Uso dei decodificatori

In molti casi è più agevole usare circuiti *di decodifica* al posto dei circuiti di rilevazione dell'indirizzo tramite porte logiche, e, a volte, anche al posto dei circuiti di selezione del dispositivo tramite porte NOR. Che cosa rende tanto convenienti i decodificatori? Probabilmente la cosa migliore da fare è esaminare rapidamente alcuni tipi di decodificatori, per vedere come si presentano e come funzionano. Esaminando i circuiti di decodifica, ricordate che non sono altro che dei raggruppamenti di porte, "integrate" in un circuito, detto decodificatore, di facile impiego.

I decodificatori sono generalmente indicati come circuiti che decodificano *x linee in y linee*, ove *x* rappresenta il numero d'ingressi binari, mettiamo quattro ingressi, ed *y* rappresenta il numero di uscite possibili, ovvero il numero dei vari stati binari presenti agl'ingressi *x*. Quindi per i quattro ingressi ci saranno 16 uscite possibili, e si avrà così un decodificatore da 4 linee a 16 linee, detto anche decodificatore da 4 a 16 linee. Come vedrete, questo è effettivamente un circuito di decodifica.

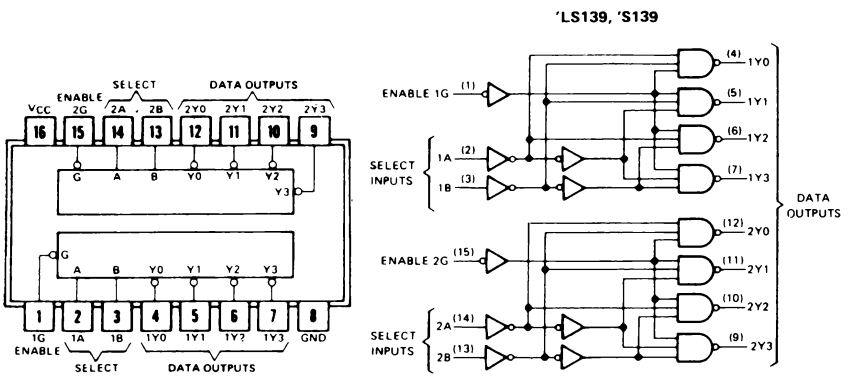


Figura 2.7 — Schema e configurazione dei pins del decodificatore SN74LS139.

Ogni ingresso binario ha due stati, uno logico e zero logico. Gl'ingressi sono indipendenti l'uno dall'altro. Anche le uscite sono binarie, nel senso che possono avere due stati, ma *non sono indipendenti*: ci sarà *un'unica* uscita dal decodificatore, che rappresenta il valore o "peso" presente agl'ingressi binari. Nella maggior parte dei circuiti l'unico stato di uscita è uno zero logico, mentre le altre uscite sono nello stato logico uno.

Un tipico circuito integrato di decodifica è l'SN74LS139, che contiene due decodificatori indipendenti, ciascuno dei quali è un decodificatore da due a quattro linee, come si può vedere in Figura 2.7.

La Tabella 2.3 riporta la tabella della verità relativa all'SN74LS139.

Ingressi			**Uscite**			
Abilitazione	**Selezione**					
G	**B**	**A**	**Y0**	**Y1**	**Y2**	**Y3**
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

H = livello alto, L = livello basso, X = non significativo

Tabella 2.3 — Tabella della verità relativa al decodificatore SN74LS139.

Chiaramente la tabella della verità si riferisce ad entrambi i decodificatori contenuti nel package del circuito integrato SN74LS139, o "chips". La maggior parte dei decodificatori contiene un ingresso di abilitazione, per cui il decodificatore può venir abilitato o disabilitato mediante un uno logico in ingresso. È questa la funzione dell'ingresso ENABLE, o "G", che si trova su ciascun decodificatore dell'SN74LS139. Si tenga presente che, quando l'ingresso "G" è un uno logico, tutte le uscite sono forzate nello stato logico uno, indipendentemente dagli stati degl'ingressi A e B. In tal modo il decodificatore può venir attivato o disattivato: nello stato di "disattivato" tutte le uscite sono forzate nello stato logico uno.

Vediamo ora un esempio semplice, e abbastanza banale, di uso di un decodificatore da due a quattro linee per la decodifica degl'indirizzi di un dispositivo. Supporremo di avere un numero limitato di dispositivi di I/U, in modo che i decodificatori presenti nel package SN74LS139 possano essere adeguati alle nostre necessità. Un tipico decodificatore compare in Figura 2.8: in questo circuito sono stati decodificati solo due bits d'indirizzi; il resto è stato ignorato. Si osservi che l'ingresso di abilitazione è stato messo a massa: così le uscite del decodificatore potranno operare correttamente. Le porte NOR ed OR aggiuntive generano gli effettivi impulsi di selezione dispositivo.

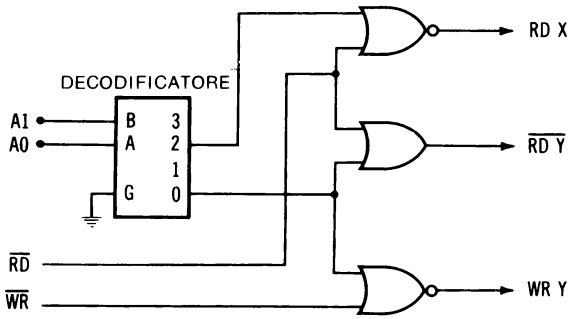


Figura 2.8 — Decodificatore da 2 a 4 linee per l'indirizzamento di un dispositivo.

I segnali di selezione dispositivo sono stati indicati con RD X, RD Y e WR Y, per il fatto che non c'è *un indirizzo specifico* che li renda attivi: gl'indirizzi 01010101 00000010, 00011101 11110110 e 00000000 11111110 genereranno tutti l'impulso di selezione dispositivo RD X, se usati in comandi di PEEK, come ad esempio A = PEEK(21762).

Quest'indirizzamento *non assoluto* del dispositivo deriva dal fatto che i bits d'indirizzo A15-A2 non sono stati usati nello schema di decodifica. Indirizzamento non assoluto vuol dire che più indirizzi attiveranno il dispositivo selezionato.

Il circuito di Figura 2.8 decodificherà quattro indirizzi, e di conseguenza potranno essere selezionati otto dispositivi, quattro d'ingresso e quattro di uscita; sono comunque richieste delle porte NOR e OR aggiuntive. Nei piccoli sistemi questo può andar bene, anche se questa tecnica di decodifica non offre molta flessibilità se si vogliono aggiungere dei nuovi dispositivi di I/U agli otto originari. Comunque, nonostante la sua scarsa flessibilità, esamineremo ora questa tecnica un po' più accuratamente, perché ci permette di sviluppare altri due concetti, applicabili anche ad altre tecniche di decodifica.

In figura 2.8, l'ingresso di abilitazione "G" del decodificatore è stato messo a massa in modo puro e semplice, in modo che l'attività di decodifica sia sempre abilitata. Quest'ingresso permette comunque di usare il decodificatore per una decodifica assoluta. Si può usare un circuito con porte logiche per dare il segnale di abilitazione al decodificatore solo in presenza di una configurazione predeterminata di bits d'indirizzo sulle linee d'indirizzo A15-A2. Abbiamo già esaminato l'uso dei circuiti di condizionamento logico a più ingressi; il circuito di Figura 2.5 ne è un esempio significativo. Questo circuito può essere facilmente adattato in modo che fornisca l'ingresso di abilitazione per un semplice decodificatore. Gl'ingressi A1 ed A0, essendo usati come ingressi del decodificatore, non sono usati come ingressi del circuito logico che fornisce il segnale di abilitazione del decodificatore. In Figura 2.9 potete vedere un semplice esempio di quanto si è appena detto; in questo circuito il segnale ADDRESS ENABLE è generato da un circuito con porte logiche (v. Figura 2.5). In questo caso i ponticelli relativi agl'ingressi d'indirizzo A1 ed A0 sono semplicemente sconnessi.

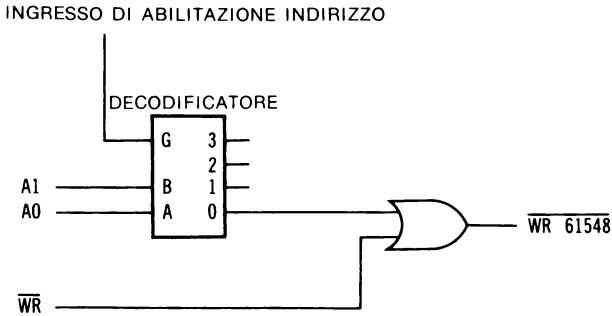


Figura 2.9 — Uso del decodificatore per la selezione assoluta di un indirizzo.

Supponendo che il circuito logico relativo alla parte alta dell'indirizzo sia stato programmato per una configurazione di bits d'indirizzo uguale ad 11110000, e che gl'ingressi A1 ed A0 al circuito siano stati sconnessi (v. Figura 2.5), il decodificatore che compare in Figura 2.9 sarà abilitato solo per gl'indirizzi da 11110000 01101100 a 11110000 01101111. Quindi, in questo circuito, le uscite 0, 1, 2 e 3 del decodificatore corrispondono agl'indirizzi di dispositivo che vanno da 61,548 a 61,551, ovvero da F06CH a F06FH. In quest'esempio è stato generato solo l'impulso di selezione dispositivo $\overline{\text{WR 61548}}$. Anche qui è necessaria una porta OR o NOR per ogni impulso di selezione dispositivo che si deve generare.

Un'altra soluzione è quella di utilizzare entrambi i decodificatori presenti nel chip SN74LS139, servendosi degl'impulsi di funzione $\overline{\text{RD}}$ e $\overline{\text{WR}}$ per abilitare i decodificatori. In tal modo la selezione dell'indirizzo è di nuovo non assoluta, ma il condizionamento per la selezione dispositivo è effettuato all'interno del chip (v. Figura 2.10). Non sono più necessarie le porte NOR ed OR per la generazione degl'impulsi di selezione dispositivo. Questo circuito, pur non essendo immediatamente utilizzabile, illustra l'uso dell'ingresso di abilitazione del decodificatore per generare l'impulso di selezione dispositivo. L'ingresso di condizionamento, ovvero di abilitazione del decodificatore, può essere usato per generare impulsi di selezione dispositivo, o per una decodifica assoluta; e, in qualche caso, per tutte e due le cose.

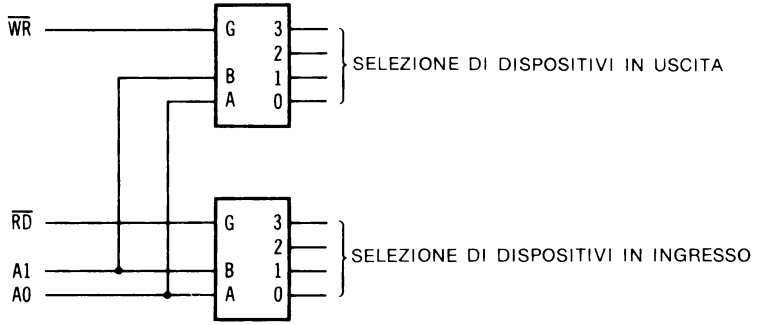
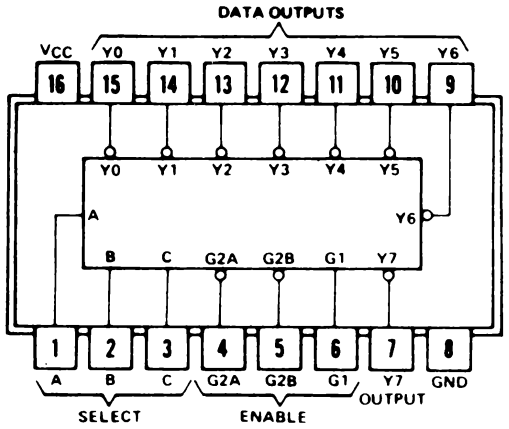


Figura 2.10 — Gl'ingressi di abilitazione del decodificatore sono usati con $\overline{\text{WR}}$ ed $\overline{\text{RD}}$ per generare segnali di selezione dispositivo.

I grossi decodificatori

Ci sono poi altri circuiti di decodifica che vi serviranno per interfacciare il vostro calcolatore Apple con i dispositivi esterni. Questi decodificatori avranno, a seconda del tipo prescelto, ingressi, linee di abilitazione e uscite aggiuntivi. In Figura 2.11 troverete un esempio relativo al decodificatore SN74LS138, in Figura 2.12 uno relativo al decodificatore SN74154.



'LS138, 'S138

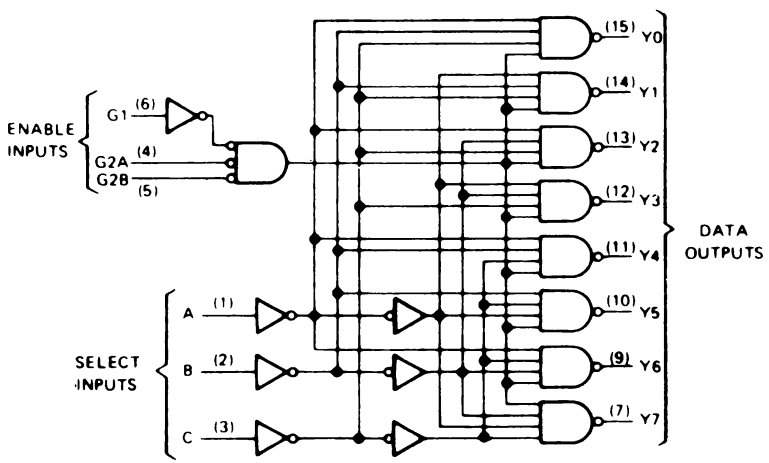
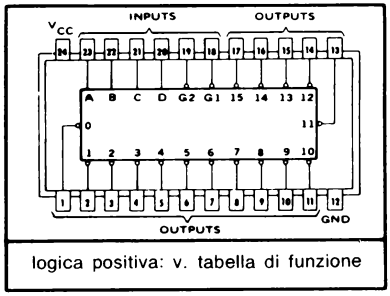


Figura 2.11 — Decodificatore SN74LS138.



logica positiva: v. tabella di funzione

diagramma a blocchi funzionale
e schema degl'ingressi e delle uscite

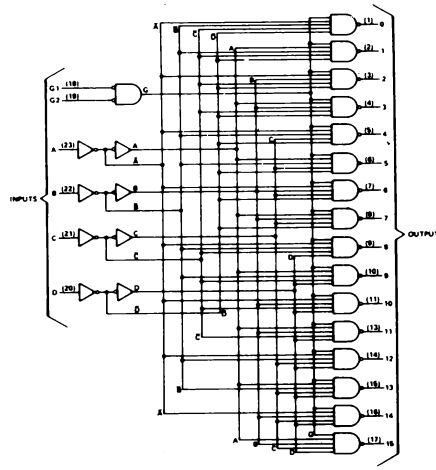


TABELLA DI FUNZIONE

INGRESSI						USCITE														
G1	G2	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H

H = livello alto, L = livello basso, X = non significativo

Figura 2.12 — Decodificatore SN74154.

Presentiamo in Figura 2.13 anche il decodificatore SN74155, perché comprende due sezioni distinte, ma con gl'ingressi degl'indirizzi, A e B, comuni ad entrambe: ciascuna sezione dell'SN74155 ha poi ingressi di controllo e di abilitazione distinti.

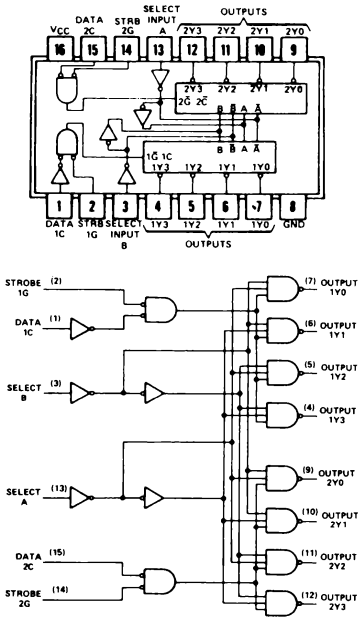


TABELLE DI FUNZIONE
DECODIFICATORE DA 2 A 4 LINEE
O DEMULTIPLEXER DA 1 A 4 LINEE

INGRESSI				USCITE			
SELECT		STROBE	DATA				
B	A	1G	1C	1Y0	1Y1	1Y2	1Y3
X	X	H	X	H	H	H	H
L	L	L	H	L	H	H	H
L	H	L	H	H	L	H	H
H	L	L	H	H	H	L	H
H	H	L	H	H	H	H	L
X	X	X	L	H	H	H	H

INGRESSI				USCITE			
SELECT		STROBE	DATA				
B	A	2G	2C	2Y0	2Y1	2Y2	2Y3
X	X	H	X	H	H	H	H
L	L	L	L	L	H	H	H
L	H	L	L	H	L	H	H
H	L	L	L	H	H	L	H
H	H	L	L	H	H	H	L
X	X	X	H	H	H	H	H

Figura 2.13 — Decodificatore SN74155.

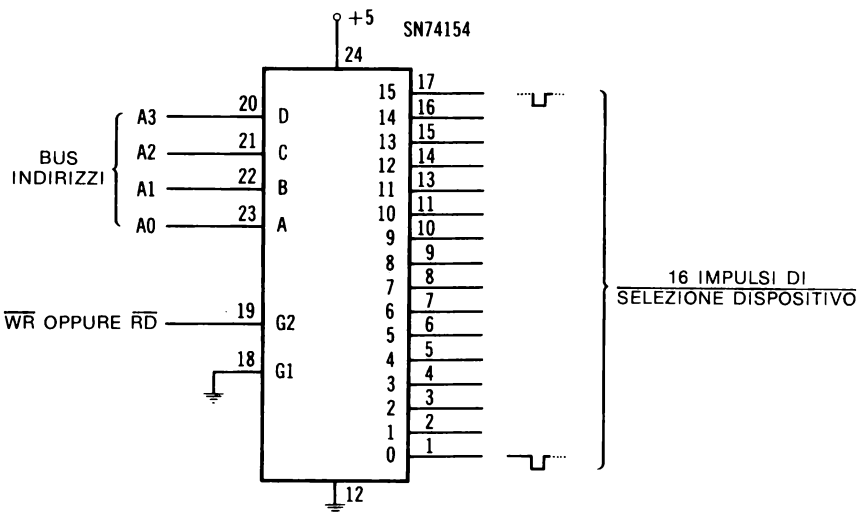


Figura 2.14 — Il decodificatore SN74154 è usato per generare 16 impulsi di selezione dispositivo. La decodifica non è assoluta.

Un decodificatore quale l'integrato SN74154, che è un decodificatore da 4 a 16 linee, offre un'elevata flessibilità nella decodifica degl'indirizzi. Un solo decodificatore SN74154 può essere usato per decodificare in modo non assoluto 16 indirizzi, e se si utilizza o $\overline{WR}$ o $\overline{RD}$ come uno degl'ingressi di abilitazione, l'SN74154 può generare direttamente 16 impulsi di selezione dispositivo, senza bisogno di altre porte (v. Figura 2.14).

Al circuito base si possono poi aggiungere altri decodificatori, o altre porte, per generare impulsi di selezione dispositivo con una decodifica assoluta. Un tipico esempio è in Figura 2.15. Si può usare o un segnale $\overline{RD}$ o un segnale $\overline{WR}$ come segnale di condizionamento logico ovvero di abilitazione del decodificatore posto in basso. Le porte NOR provvedono al condizionamento del *segnale di selezione indirizzo* (generato dalla parte alta del circuito) con la *selezione indirizzo più l'impulso di funzione*, generati dal decodificatore posto in basso. Quindi la parte alta del circuito "qualifica" le uscite provenienti dal decodificatore in basso in modo che la se-

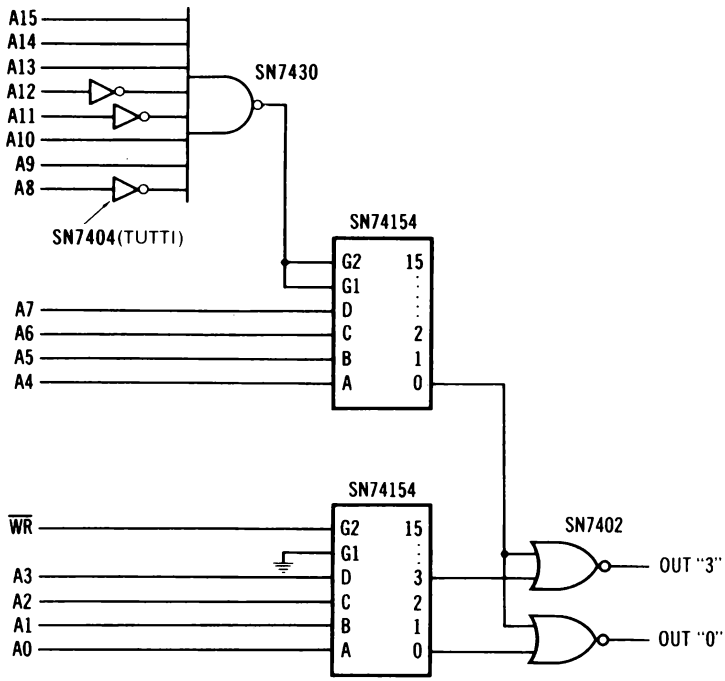


Figura 2.15 — Applicazione di decodificatori SN74154 e di porte per la selezione assoluta dell'indirizzo dispositivo. La decodifica è assoluta.

lezione dell'indirizzo sia assoluta. Nella figura sono evidenziati due impulsi di selezione dispositivo. Questo circuito, pur essendo funzionante, non è molto vantaggioso, perché passibile di semplificazioni.

Dal momento che i decodificatori SN74154 hanno due ingressi di abilitazione, G1 e G2, le porte NOR che compaiono in Figura 2.15 possono venir eliminate, usando il secondo ingresso di abilitazione come ingresso "di qualificazione" che abiliterà il decodificatore. L'uso di questo tipo di circuito è rappresentato in Figura 2.16. Qui il decodificatore in basso ha in ingresso due segnali di abilitazione, il segnale di controllo $\overline{RD}$ proveniente dal calcolatore, ed il segnale di abilitazione proveniente dalla parte alta del circuito. Osserverete che nel decodificatore in alto sono utilizzati entrambi gl'impulsi di abilitazione: in tal modo questo circuito è abilitato solo per una particolare configurazione di bits relativa alla parte alta HI del bus indirizzi. In questo caso le porte logiche generano il segnale di abilitazione per il decodificatore in alto.

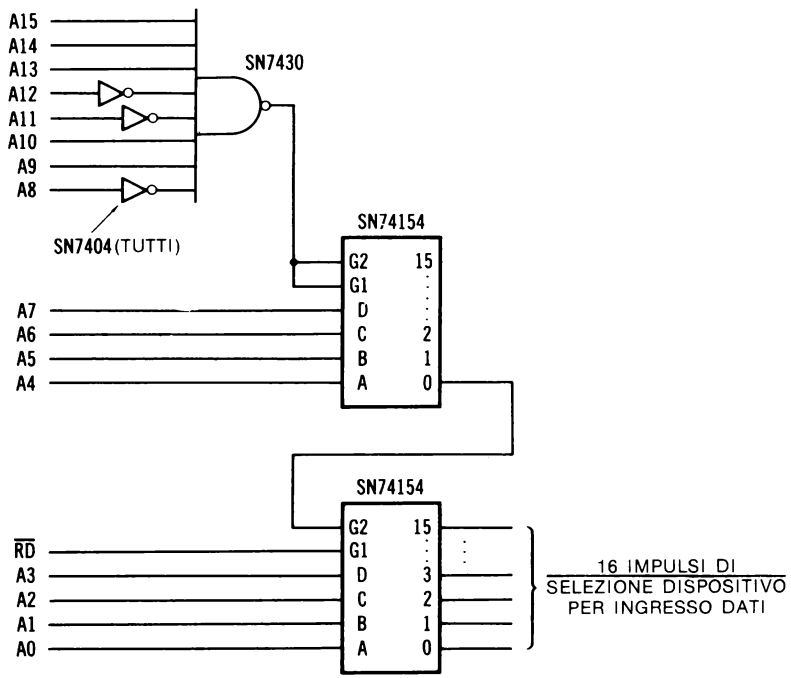


Figura 2.16 — Circuito di selezione dispositivo perfezionato.

Al circuito si può poi aggiungere un terzo decodificatore per generare gl'impulsi di selezione dispositivo per i dispositivi di uscita dati. Gl'ingressi a quest'altro decodifi-

catore saranno gli stessi del decodificatore in basso, solo che sarà usato il segnale $\overline{WR}$ invece del segnale $\overline{RD}$.

Sono possibili molti schemi di decodificatori, e negli esperimenti avrete modo di approfondirne le modalità d'uso. La cosa più importante è che l'impiego dei decodificatori semplifica il processo di selezione e di condizionamento del dispositivo. I decodificatori sono utilizzati in genere nelle situazioni che richiedono flessibilità e la generazione, sulle medesime schede, di più segnali di selezione dispositivo, o d'indirizzo dispositivo.

Uso dei comparatori

L'ultima tecnica che analizzeremo è l'uso dei comparatori digitali per la rilevazione dell'indirizzo dispositivo. Gli schemi basati sui comparatori sono relativamente semplici, e molto simili a quelli delle "porte programmabili" delle Figure 2.4 e 2.5. Ricordiamo che anche i comparatori non sono altro che dei raggruppamenti di porte, collegate o integrate, che svolgono una funzione di confronto. I comparatori ci permettono di presentare un indirizzo, che è quindi costantemente confrontato con i valori a 16 bits del bus indirizzi. Il confronto è operato da porte logiche situate all'interno del chip del comparatore. Un comparatore tipo è il comparatore (a 4 bits di grandezza) SN7485, rappresentato in Figura 2.17. Oltre alla condizione di uguale, l'SN7485 è in grado di rilevare anche le condizioni di maggiore e minore, che però non intervengono nei confronti d'indirizzi. Attenzione: *la versione SN74L85 del chip SN7485 non ha un'equivalenza uno a uno per quanto riguarda i pins*. Per ulteriori informazioni, consultate il catalogo del costruttore.

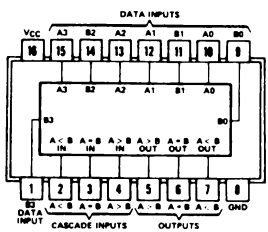


TABELLE DI FUNZIONE

COMPARING INPUTS				CASCADING INPUTS			OUTPUTS		
A3, B3	A2, B2	A1, B1	A0, B0	A > B	A < B	A = B	A > B	A < B	A = B
A3 > B3	X	X	X	X	X	X	H	L	L
A3 < B3	X	X	X	X	X	X	L	H	L
A3 = B3	A2 > B2	X	X	X	X	X	H	L	L
A3 = B3	A2 < B2	X	X	X	X	X	L	H	L
A3 = B2	A2 = B2	A1 > B1	X	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 < B1	X	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 > B0	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 < B0	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	L	L	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	H	L	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	H	L	L	H

'85, 'LS85, 'S85

A3 = B3	A2 = B2	A1 = B1	A0 = B0	X	X	H	L	L	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	H	L	L	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	L	H	H	L

Figura 2.17 — Chip comparatore a 4 bits di grandezza.

Un tipico schema di confronto indirizzi è in Figura 2.18, nella quale, per ragioni di chiarezza, sono disegnati solo 8 dei 16 bits d'indirizzo.

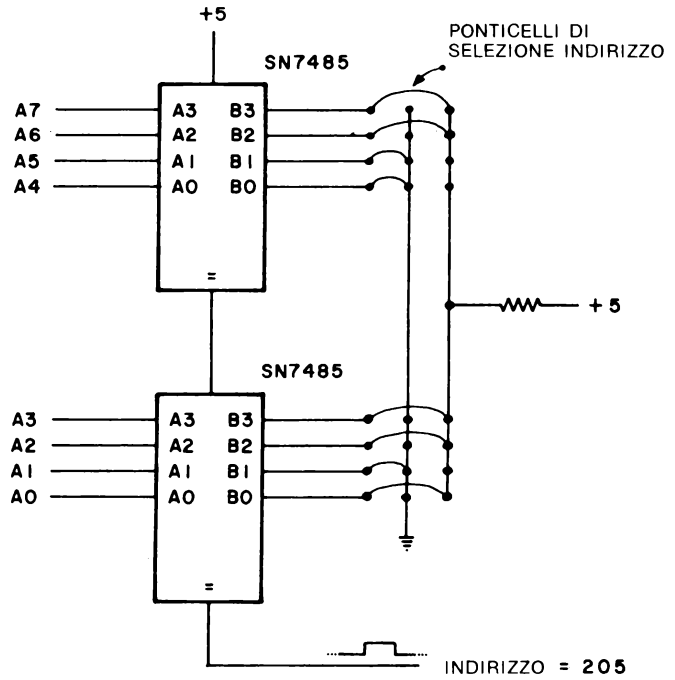


Figura 2.18 — Due comparatori SN7485 per la rilevazione dell'indirizzo 205..

I comparatori sono stati programmati per la rilevazione dell'indirizzo 205, ovvero 11001101_2. Analogamente alle porte logiche ad 8 ingressi, questo schema può rilevare soltanto un unico indirizzo, per cui per lo più i comparatori vengono utilizzati con decodificatori per ottenere uno schema di decodifica flessibile, come si vede in Figura 2.19. L'unica uscita di "condizione di uguale" dei comparatori SN7485 è un uno logico, per cui è stato invertito in modo da fornire il segnale di abilitazione ad un chip decodificatore. In questo circuito si sono introdotti altri due comparatori, per avere una decodifica assoluta degl'indirizzi di dispositivo. Ora, le uscite del decodificatore SN74154 diventano attive solo quando i bits d'indirizzo A15-A4 hanno gli stati logici corrispondenti a quelli che sono stati prefissati agl'ingressi dei tre comparatori. In questo caso i bits d'indirizzo devono essere 11101001 per A15-A8, e 0000 per A7-A4. Dal momento che dev'essere presente anche l'impulso di funzione $\overline{WR}$ per abilitare il decodificatore, potete facilmente verificare che i segnali di selezione indirizzo dispositivo per operazioni di uscita dati sono generati da questo circuito, relativamente agl'indirizzi che vanno da 59648 a 59655, ovvero da E900H ad E907H. Al circuito si potrà poi aggiungere un altro decodificatore SN74154, allo scopo di generare 16 segnali di selezione indirizzo dispositivo per operazioni d'ingresso dati: saranno necessari dei collegamenti in parallelo per gl'ingressi dei due decodificatori, solo che invece del segnale $\overline{WR}$ si dovrà usare il segnale $\overline{RD}$.

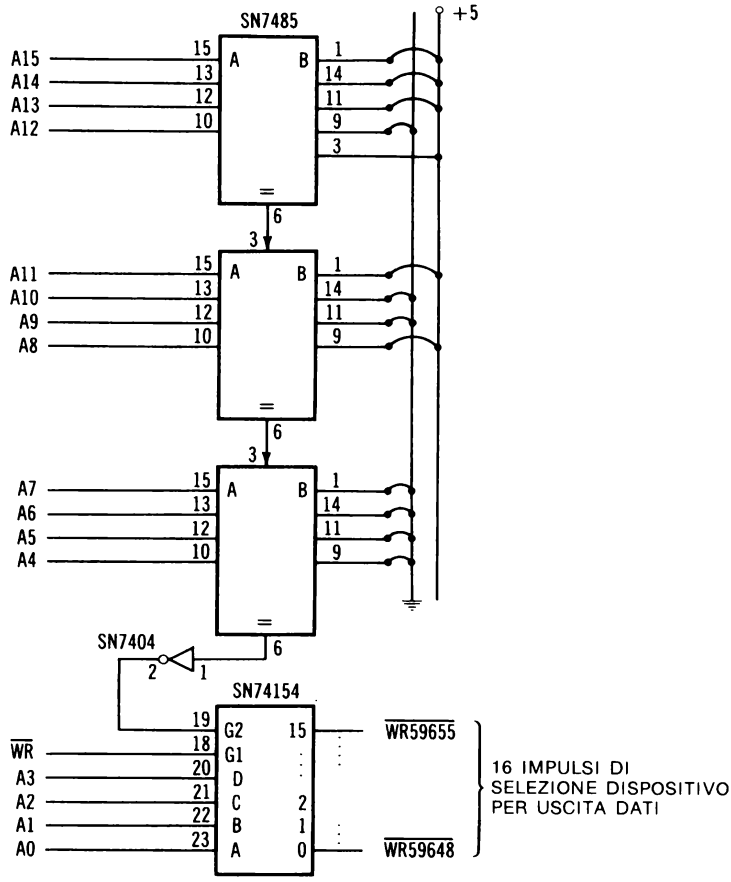


Figura 2.19 — Comparatori e decodificatori per la selezione indirizzo.

Con ciò abbiamo concluso la trattazione dei circuiti d'indirizzamento dispositivo, e delle combinazioni degl'indirizzi di dispositivo con gl'impulsi di funzione per ottenere impulsi di selezione dispositivo. Siamo certi che, negli esempi che seguiranno, sarete in grado di riconoscere nella notazione $\overline{\text{WR}}$ 54390 un impulso di selezione dispositivo attivo nello stato di zero logico, generato dall'opportuno condizionamento logico dell'impulso di funzione $\overline{\text{WR}}$ e dell'indirizzo 54390. In alcuni esempi saranno visibili le effettive porte logiche, ma per lo più daremo per scontato che sappiate qual è l'origine del segnale. E anche se, come è probabile, troverete in altri libri, o in articoli di riviste, circuiti d'indirizzamento e di selezione dispositivo diversi da quelli mostrati qui, vedrete subito che funzionano tutti press'a poco allo stesso modo: condizionamento logico di un segnale d'indirizzo con un impulso di funzione, allo scopo di selezionare uno specifico dispositivo.

In alcuni esperimenti avrete modo di studiare l'uso degl'impulsi di selezione dispositivo per controllare i dispositivi in questione. Nel prossimo capitolo imparerete come si usano quest'impulsi per controllare il flusso dei bytes di dati di 8 bits sul bus dati del 6502.

CAPITOLO 3

L'INTERFACCIAMENTO DELL'I/U

Dopo aver illustrato alcuni metodi di selezione ed identificazione di dispositivi di I/U, bisogna analizzare l'effettiva struttura e configurazione delle porte di I/U. In questo capitolo svilupperemo alcune tecniche effettive d'interfacciamento al bus che permettono ai dispositivi di I/U di trasferire bytes da 8 bits al calcolatore e di ricevere i bytes che il calcolatore trasferisce ad essi. Come si è visto per i circuiti di selezione del dispositivo, esistono molti circuiti adibiti a porte d'ingresso ed a porte di uscita. Ad illustrazione dei principi fondamentali dell'interfacciamento, presenteremo solo pochi circuiti campione.

LE PORTE DI USCITA

Le porte di uscita sono dispositivi che ricevono bytes di dati dal calcolatore, sotto il controllo di comandi di POKE contenuti in programmi in BASIC. Si è già visto che, quando è eseguito un comando di POKE, esiste un ben preciso rapporto temporale fra i dati sul bus, l'impulso $\overline{WR}$ e l'indirizzo di dispositivo (v. Figura 1.4). Nel calcolatore Apple, la durata di un impulso $\overline{WR}$ è di circa 500 nanosecondi. Se si usa l'impulso $\overline{WR}$ per condizionare l'invio dei dati dal bus ad un dispositivo di uscita (mediante l'impulso di selezione dispositivo), il dato è presentato al dispositivo di uscita solo per circa 500 nanosecondi. Questo tempo è appena sufficiente perché il dispositivo ricevente possa effettuare un'operazione significativa. Per eliminare questo problema, ciascuna porta di uscita dev'essere corredata da una qualche sorta di circuito capace di acquisire i dati dal bus e "trattenerli" per tutto il tempo che è necessario, o finché non sono "aggiornati" da un altro trasferimento di dati.

Il tipo di circuito capace di svolgere questa funzione è detto *latch* (lucchetto), perché è in grado di "chiudere" l'informazione e conservarla finché non viene aggiornata, o finché non si disinserisce l'alimentazione. Esistono molti tipi diversi di circuiti integrati di latch, che offrono diverse configurazioni di controllo e d'ingressi ed uscite dati. Piuttosto che descrivere tutti i vari tipi di latches, abbiamo preferito descrivere tre dispositivi d'uso generale, cioè l'SN7475, l'SN74175 e l'SN74LS373.

In Figura 3.1 troverete le configurazioni dei pins e le tabelle di funzione. Mentre l'SN7475 e l'SN74LS373 sono veri dispositivi di latch, l'SN74175 in realtà contiene dei flip-flops. Il chip di latch SN7475 contiene quattro circuiti di latch; l'SN74175 contiene quattro circuiti di flip-flop, e quindi per ogni porta di uscita ad 8 bits sono necessari due chips SN7475, o SN74175. L'SN74LS373 contiene otto circuiti di latch, e quindi ne basta uno solo per costruire una porta di uscita ad 8 bits.

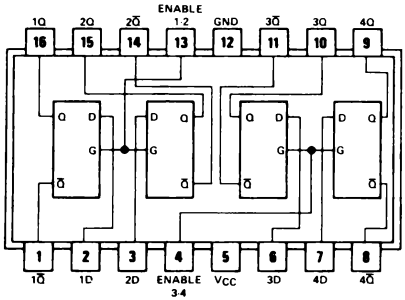


TABELLA DI FUNZIONE
(per ciascun latch)

INPUTS		OUTPUTS	
D	G	Q	$\bar{Q}$
L	H	L	H
H	H	H	L
X	L	Q_0	$\bar{Q}_0$

H = livello alto, L = livello basso, X = non significativo
Q_0 = livello di Q prima della transizione di G dal livello alto al livello basso

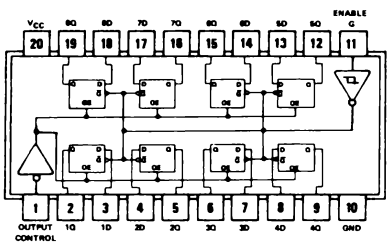


TABELLA DI FUNZIONE
PER LS373, S373

OUTPUT CONTROL	ENABLE G	D	OUTPUT
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

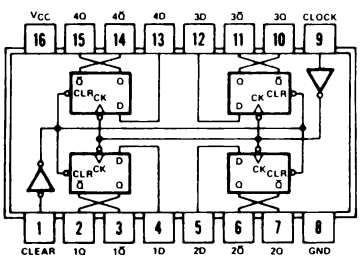


TABELLA DI FUNZIONE
(per ciascun flip-flop)

INPUTS			OUTPUTS	
CLEAR	CLOCK	D	Q	$\bar{Q}$↑
L	X	X	L	H
H	↑	H	H	L
H	↑	L	L	H
H	L	X	Q_0	$\bar{Q}_0$

Figura 3.1 — Configurazione dei pins e tabelle di funzione relative ai chips di latch SN7475 (in alto), SN74LS373 (al centro), e SN74175 (in basso).

Descriveremo ora brevemente il funzionamento di questi circuiti di latch, in modo da rendere evidente il loro uso. Come esempio ci riferiremo al chip di latch SN7475. I circuiti di latch SN7475 possono esser visti come "porte che ricordano": questo si può vedere nella tabella di funzione relativa al latch SN7475, in Figura 3.1. Esaminando questa tabella di funzione, noterete che, quando l'ingresso di abilitazione (G)

è un uno logico, il dato, o livello logico presente all'ingresso D, passa, attraverso il latch, all'uscita Q. L'uscita $\overline{Q}$ è l'inverso dell'uscita Q. Quando l'ingresso di abilitazione passa dall'uno logico allo zero logico, il livello presente in quel momento all'ingresso D è "trattenuto", ovvero ricordato, dagl'ingressi Q e $\overline{Q}$. Le temporizzazioni che compaiono in Figura 3.2 illustrano queste operazioni.

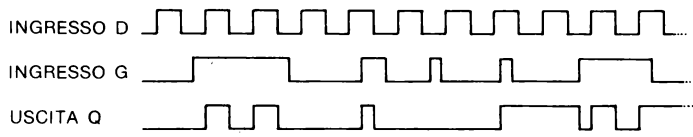


Figura 3.2 — Temporizzazione relativa al circuito di latch SN7475.

Non appena l'ingresso G raggiunge il livello logico uno, l'uscita Q assume lo stato dell'ingresso D, anche nel caso in cui i livelli all'ingresso D siano cambiati. I livelli logici passano dall'ingresso D all'uscita Q quando l'ingresso G è un uno logico; l'ingresso Q resta al livello dell'ingresso D quando l'ingresso G passa allo stato logico zero.

L'SN7475 è diviso in due parti, ciascuna delle quali può operare indipendentemente dall'altra. I due ingressi di abilitazione possono venir collegati, e in tal modo i quattro circuiti di latch lavoreranno in tandem. È chiaro comunque che gl'ingressi e le uscite dei latches restano indipendenti, per cui si possono trattare quattro segnali d'ingresso provenienti da punti diversi del circuito; tuttavia, se le funzioni sono svolte in tandem, tutti e quattro gl'ingressi saranno "trattenuti" contemporaneamente.

L'SN74LS373 opera allo stesso modo dell'SN7475, anche se in esso è usato un solo segnale di abilitazione. In questi chips ci sono solo le uscite Q, mentre non sono disponibili le uscite $\overline{Q}$. È previsto anche un segnale aggiuntivo di controllo uscita, ma questo segnale di controllo, Output Control (pin 1), è di solito messo a massa quando l'SN74LS373 è usato come porta di uscita.

Il chip SN74175 contiene quattro flip-flops, che acquisiscono e conservano l'informazione presente sul *fronte di salita* dell'impulso di clock. Le uscite vengono aggiornate solo in questo momento, per cui gl'ingressi non sono logicamente sentiti dall'SN74175 durante lo stato di zero logico o durante lo stato di uno logico del segnale di clock. Questo è quello che distingue un dispositivo flip-flop dai dispositivi latch, anche se poi nell'interfacciamento dei calcolatori i due tipi di chips hanno in pratica lo stesso effetto.

Nell'SN74175 è previsto anche un ingresso comune di azzeramento (CLEAR INPUT), per cui i flip-flops possono venir "azzerati" ($Q = 0$, $\overline{Q} = 1$) quando quest'ingresso è forzato nello stato logico zero. Nella maggior parte dei casi, l'ingresso di azzeramento sarà collegato a +5 volts (uno logico) e non verrà utilizzato.

Ciascun circuito integrato può essere usato per "bloccare" e conservare i dati emessi dal calcolatore Apple durante l'esecuzione di un comando di POKE: è sufficiente usare un impulso di selezione dispositivo in scrittura, per attivare il circuito di

latch, una volta che questo sia stato correttamente collegato al bus. In Figura 3.3 potete vedere una tipica porta di uscita ad 8 bits.

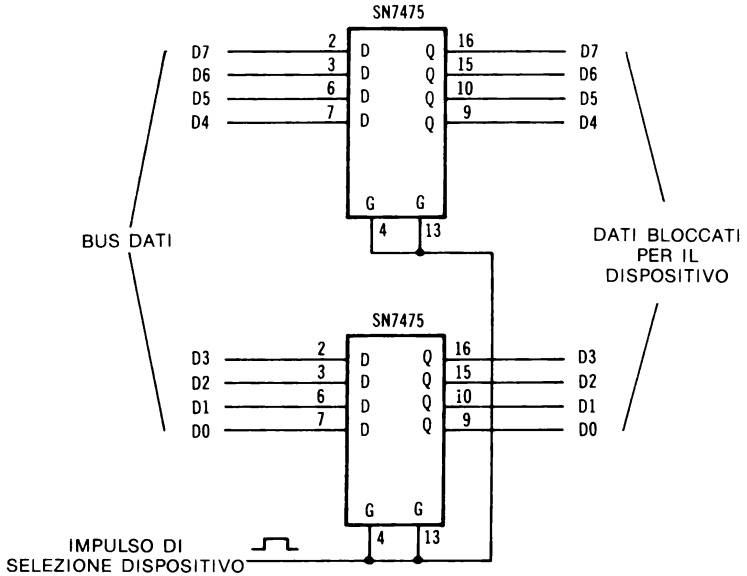


Figura 3.3 — Due chips di latch formano una porta di uscita.

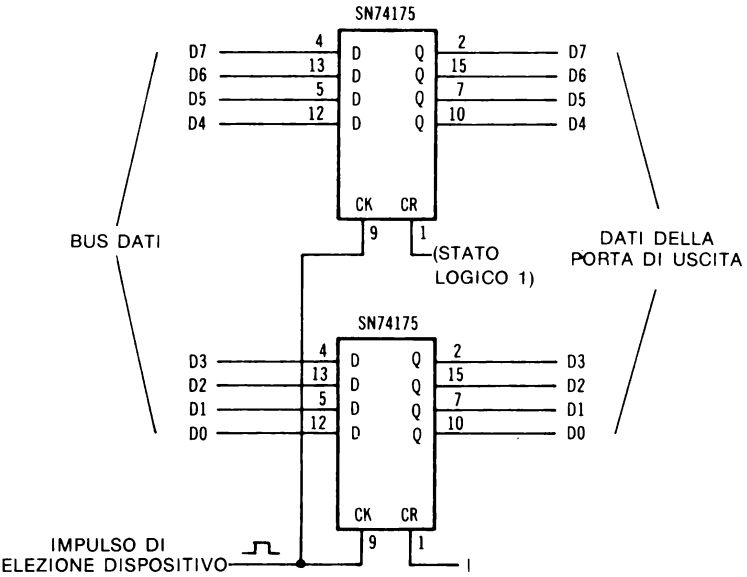


Figura 3.4 — Due chips di latch SN74175 formano una porta di uscita.

In questo circuito è necessario un impulso positivo di selezione dispositivo in scrittura, affinché i circuiti di latch acquisiscano e conservino l'informazione emessa dall'Apple.

In Figura 3.4 si vedono due chips SN74175 di latch usati come porta di uscita, quasi fossero dei monitors logici che forniscono delle indicazioni visive sulle informazioni "trattenute" dai circuiti.

L'indicazione "1" accanto ai collegamenti relativi agl'ingressi di azzeramento (CLEAR INPUTS) vuol dire che questi ingressi sono collegati a +5 volts, cioè al livello di uno logico. La notazione "1" si usa per distinguere un collegamento di livello logico da un collegamento di alimentazione, che è indicato con +5 volts, o +5V.

In Figura 3.5 abbiamo usato un latch SN74LS373 ad 8 bits, o *ottale*, come porta di uscita.

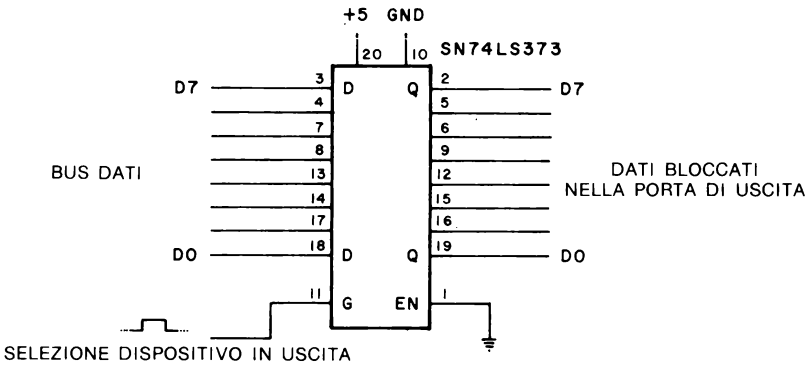


Figura 3.5 — Un solo chip di latch SN74LS373 forma una porta di uscita.

Per questa porta di uscita è necessario un solo circuito integrato. La linea di Output Control è stata messa a massa, in modo che le uscite siano abilitate in modo permanente. Anche qui la logica di selezione del dispositivo deve fornire un impulso di selezione dispositivo in scrittura. Una volta che una porta di uscita è stata correttamente collegata al bus dati ed alla sorgente che produce l'impulso di selezione dispositivo, allora ad essa si può accedere sotto il controllo di comandi software. Ad esempio, il comando POKE 49312,0 trasferirà il valore zero alla porta di uscita con indirizzo 49312. Se realmente c'è una porta di uscita collegata al bus dati che corrisponde a quest'indirizzo, il valore zero sarà trasferito ad essa.

Il programma dell'esempio 3.1 è in grado di generare un contatore binario crescente alla porta di uscita 49320. Il conteggio proseguirà in sequenza (in binario) nel modo seguente: 254, 255, 0, 1, 2 ... 254, 255, 0, 1, etc. Incontreremo ancora questo programma nel capitolo sugli esperimenti.

ESEMPIO 3.1 — Programma di conteggio binario su 8 bits relativo alla porta 49320

```
10  FOR N = 0 TO 255
20  POKE 49320,N
30  NEXT N
40  GOTO 10
```

Le porte di uscita sono abbastanza facili da costruire. Come latches si possono usare molto spesso dei dispositivi logici con parallelismo in ingresso e in uscita e con capacità interne di latch. Esempi di dispositivi che si possono usare come latches sono il contatore binario programmabile SN74193, il registro a scorrimento universale SN74LS194A, il registro a scorrimento SN74198, etc.

La maggior parte delle porte di uscita si possono facilmente creare con circuiti integrati standard, ma alcuni dei più recenti dispositivi a circuito integrato, intesi in modo specifico per l'uso con microcalcolatori, e che incorporano funzioni di latch, stanno ormai diventando disponibili; un esempio ne è il chip convertitore digitale-analogico ad 8 bits NE5018 della Signetics, che ha una parte costituita da un latch.

Applicazioni tipiche delle porte di uscita sono ad esempio:

- il trasferimento di dati ad una stampante;
- il trasferimento di dati ad uno schermo video;
- il controllo di un semaforo;
- il trasferimento di dati ad un floppy disk;
- il sistema di commutazione per un trenino elettrico;
- il controllo delle valvole e delle pompe nei processi chimici;
- il controllo di un plotter;
- il trasferimento di dati ad un display a sette segmenti;
- il controllo di un altro calcolatore.

In alcune applicazioni si utilizza il valore dell'informazione, in altre invece si utilizza lo stato logico (1 o 0) di ciascun bit. Alcune periferiche, come ad esempio le stampanti, possono servirsi di una combinazione di porte di uscita: porte per il trasferimento dei dati da stampare, e porte per il controllo delle funzioni della stampante. I displays costituiti da LED a sette segmenti spesso richiedono l'impiego di più porte di uscita, anche se il display è visto come una "periferica" unica.

LE PORTE D'INGRESSO

Le porte d'ingresso vengono usate con i dispositivi di I/U in modo da poter trasferire le informazioni al calcolatore sotto forma di bytes di 8 bits. Contrariamente alle porte di uscita, che devono essere in grado di ricevere e conservare le informazioni che si trovano sul bus in un determinato momento, e possono essere connesse permanentemente al bus dati, le porte d'ingresso devono essere in grado di "scollegarsi" dal bus quando non sono usate. Le porte d'ingresso devono trasmettere alla CPU gli uni e gli zeri logici, ma devono essere configurate in modo da non interferire con il funzionamento del bus nei momenti in cui non sono selezionate.

A seconda del tipo di porta usato, le porte semplici non possono provvedere all'invio condizionato di dati sulle linee del bus dati, per il fatto che il loro stato di uscita, quando "non" sono selezionate, sarà o un uno logico, o uno zero logico, come si vede in Figura 3.6

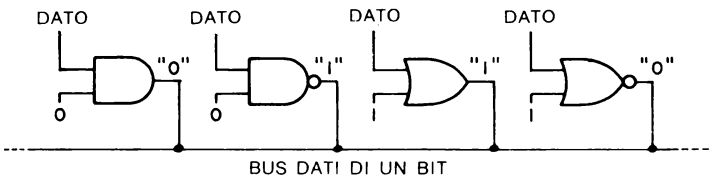


Figura 3.6 — Utilizzo errato di porte standard su un bus dati.

Si tenga presente che, anche quando nessuna porta è selezionata o abilitata, le uscite delle porte si trovano a livelli logici diversi, come è indicato dai livelli logici fra virgolette. Questi livelli "si disputano" l'utilizzo del bus, determinando verosimilmente il bruciarsi di uno o più chips. Questo fatto spiega chiaramente perché sui bus dati non vadano usate le porte da sole.

Sono disponibili speciali circuiti integrati con uscite *a tre stati*, che semplificano la progettazione delle porte d'ingresso. Un tipico dispositivo a tre stati è il buffer bus SN74125, rappresentato in Figura 3.7. Lo schema dei quattro dispositivi dovrebbe non esservi nuovo. Non è altro che un buffer (con uno logico in ingresso si ha uno logico in uscita, e così via), ma con una linea di controllo supplementare, che nella figura si vede collegata ad un lato del triangolo che rappresenta il simbolo del buffer. Il buffer passa gli uni e gli zeri logici dal suo ingresso alla sua uscita quando è abilitato, ma, a differenza di quanto accade con una porta semplice, quando viene disabilitato si presenta elettricamente disconnesso dal bus, o da un qualunque altro dispositivo di uscita al quale sia collegato. Nei dispositivi a tre stati, il terzo stato è spesso detto stato HI-Z, o di alta impedenza, per indicarne la condizione di non collegamento. Il passaggio dal collegamento al non collegamento e viceversa è rapido, richiedendo in genere meno di 20 nanosecondi.

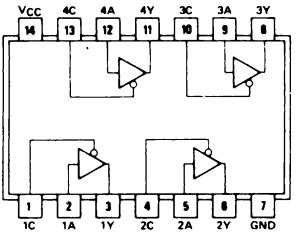


Figura 3.7 — Configurazione dei pins del chip di buffer bus SN74125.

Nel circuito SN74125 ciascun buffer a tre stati ha il suo proprio ingresso di abilitazione, che dev'essere uno zero logico affinché i dati siano trasferiti dall'ingresso all'uscita. Uno stato logico uno sull'ingresso di abilitazione forza l'uscita nello stato di alta impedenza. Un circuito integrato analogo, l'SN74126, è equivalente pin a pin all'SN74125, solo che è abilitato con uno logico e disabilitato con zero logico. Questi chips illustrano il funzionamento dei dispositivi a tre stati, ma di solito non si trovano nei circuiti d'interfaccia dei calcolatori, essendo disponibili altri dispositivi più convenienti.

A scopo illustrativo, presentiamo in Figura 3.8 un bus tipico.

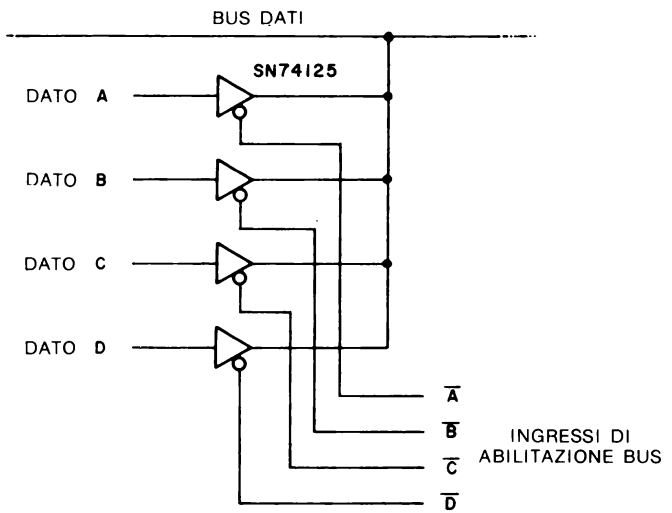


Figura 3.8 — Un tipico bus a tre stati per quattro dispositivi.

In questo circuito al bus sono collegati quattro dispositivi da un bit. Per chiarezza abbiamo disegnato soltanto un bus da un bit; comunque in un sistema di bus ad 8 bits ci saranno otto linee. Quando uno degl'INGRESSI DI ABILITAZIONE BUS è posto allo stato logico zero, il bit di dato corrispondente viene trasferito, attraverso il

buffer, al bus. Assumeremo che al bus non sia collegato nessun altro dispositivo; quindi la tabella della verità della Tabella 3.1 si riferisce unicamente a questo semplice circuito di bus.

Abilitazione				Contenuto del bus
D	C	B	A	
1	1	1	1	Indeterminato (tutti i dispositivi sono HI-Z)
1	1	1	0	Dato A
1	1	0	1	Dato B
1	0	1	1	Dato C
0	1	1	1	Dato D
0	0	0	0	Non consentito

Tabella 3.1 — Tabella della verità relativa ad un bus a tre stati per quattro dispositivi.

Quando nessun buffer è stato abilitato (ovvero collegato al bus), il bus non è collegato con nulla, tranne che con l'ingresso delle porte, delle memorie, etc., che sono i "ricevitori" del bit di dato: allora il valore logico del bus non è noto. Ogni volta che uno zero logico è applicato ad uno degl'indirizzi di abilitazione del buffer bus, *il buffer selezionato* trasferisce il suo dato sul bus. La condizione per cui più di un buffer è abilitato non è consentita, perché sorgerebbero dei conflitti di bus.

Tutti i dispositivi destinati ad essere utilizzati con il computer Apple per trasferire informazioni sulla CPU *devono* essere dotati di uscite a tre stati. Quindi tutti i chips di memoria devono avere uscite a tre stati, e di fatto così è nella pratica. Chi progetta un calcolatore dev'essere ben certo che il calcolatore sia progettato in modo che non vengano selezionati due dispositivi d'ingresso contemporaneamente. In caso di una selezione multipla di questo tipo, il calcolatore funzionerebbe in modo sbagliato.

Le porte d'ingresso utilizzabili per trasferire informazioni al calcolatore si possono facilmente costruire con circuiti integrati standard a tre stati. Per lo più si usano otto singoli buffers a tre stati, uno per ogni linea del bus; gl'ingressi di abilitazione sono tutti collegati in parallelo, cosicché tutti gli otto buffers trasferiscono la loro informazione al bus contemporaneamente. In alcuni casi l'ingresso di abilitazione comune si trova all'interno del chip, per cui un solo pin del chip è sufficiente per il controllo di tutti e otto i bits.

Molti sono i chips che possono venir utilizzati per costruire porte d'ingresso, ma solo pochi sono abbastanza generali da meritare di essere esaminati qui. I due più importanti circuiti integrati di cui ci serviremo negli esempi sono l'SN74365 e l'SN74LS244. Il primo può essere denominato anche DM8095 (National Semiconductor Corp.), che ne è un puntuale corrispettivo. In Figura 3.9 troverete la configurazione dei pins di questi due chips.

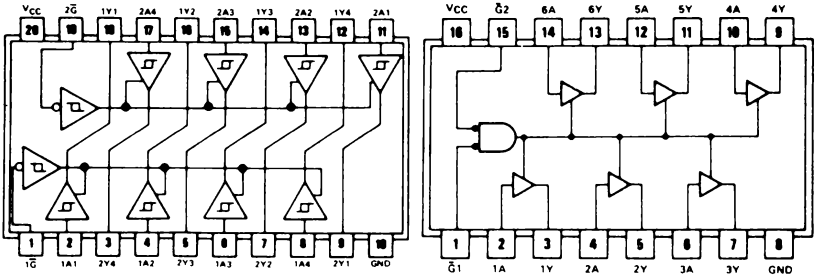


Figura 3.9 — Configurazione dei pins dei chips di driver bus a tre stati SN74LS24 ed SN74365 (DM8095).

Noterete subito che, mentre l'SN74LS244 ha otto buffers a tre stati su un unico chip, l'SN74365 ne ha soltanto sei. Quando con l'SN74365 si costruisce una porta d'ingresso, si devono usare due packages di circuito integrato. Una tipica porta d'ingresso ad 8 bits è illustrata in Figura 3.10.

Qui si sono usati solo due dei buffers a tre stati presenti nel chip SN74365 inferiore. L'SN74365 contiene delle porte NOR incorporate per il controllo dell'abilitazione

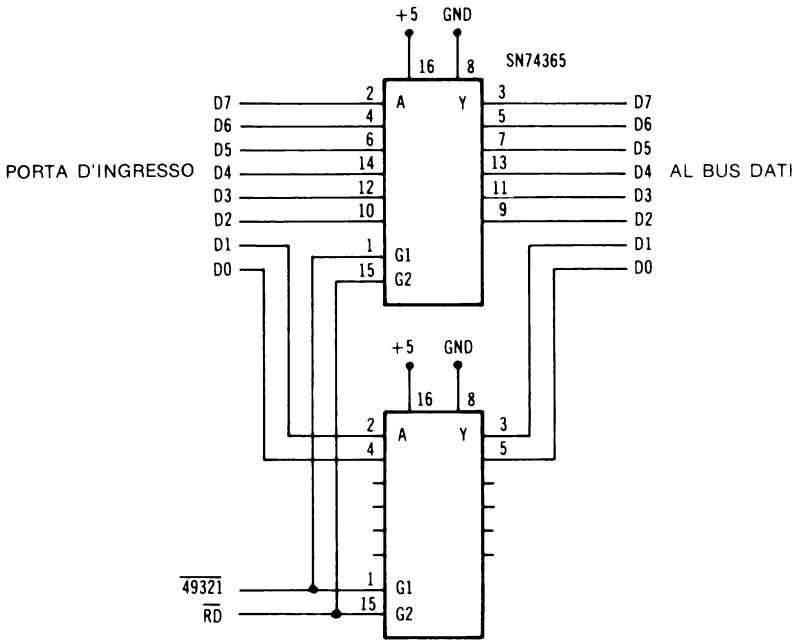


Figura 3.10 — Tipica porta d'ingresso costruita con chips SN74365.

dei buffers a tre stati: queste porte sono state usate per condizionare con l'impulso di funzione $\overline{RD}$ l'indirizzo di dispositivo $\overline{49321}$. Se il segnale di selezione dispositivo $\overline{RD\ 49321}$ è già stato generato in un altro punto del circuito d'interfaccia, questo potrebbe essere applicato ad uno degl'ingressi di abilitazione di ciascun chip, mentre gli altri ingressi di abilitazione saranno messi a massa, ovvero al valore logico zero. Questo schema di controllo è rappresentato in Figura 3.11.

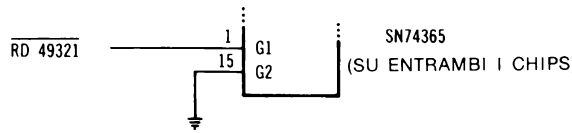


Figura 3.11 — Un altro schema di controllo con i chips a tre stati SN74365.

Con una porta d'ingresso di questo tipo i valori dei dati possono essere acquisiti dal calcolatore mediante il comando di PEEK, come è illustrato nell'Esempio 3.2.

ESEMPIO 3.2 — Programma d'ingresso dati dalla porta 49321

```
10 A = PEEK(49321)
20 PRINT A
30 GOTO 10
```

Qui il valore binario di 8 bits è convertito in un numero decimale compreso fra 0 e 255, quando viene acquisito dall'Apple per mezzo del comando di PEEK alla linea 10 del programma: questo valore è "stampato" sullo schermo del video. Sarebbe stato corretto anche il comando seguente:

```
10 PRINT PEEK(49321): GOTO 10
```

Una porta d'ingresso di questo tipo può essere costruita con un buffer ottale (cioè ad 8 bits) SN74LS244. Questo chip comprende due serie indipendenti di quattro buffers ciascuna, che vengono controllate in maniera indipendente per mezzo di due ingressi di abilitazione, $\overline{2G}$ ed $\overline{1G}$. Non essendoci nell'SN74LS244 porte NOR incorporate, sono necessarie delle porte esterne per la selezione del dispositivo. In Figura 3.12 è rappresentata una tipica porta d'ingresso nella quale si è utilizzato un chip SN74LS244. Per il controllo del flusso dell'informazione da questa porta al calcolatore saranno utilizzate linee di programma analoghe a quelle dell'Esempio 3.2.

Per entrambi i circuiti SN74365 ed SN74LS244 vi sono circuiti equivalenti pin a pin, che invertono i bits di dato nel momento in cui passano attraverso i chips, sul bus dati. Questi buffers sono, rispettivamente, l'SN74366 e l'SN74LS240. L'SN74366 è equivalente anche al chip DN8096. Nella maggior parte dei casi saranno i buffers non invertenti quelli usati nei circuiti d'interfaccia.

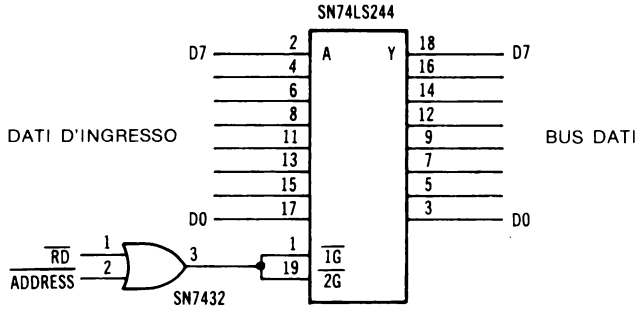


Figura 3.12 — Porta d'ingresso costruita con un chip SN74LS244.

In alcuni casi le periferiche possono generare più di otto bits d'informazione, che devono essere letti dal calcolatore: un esempio di dispositivo di questo tipo può essere un convertitore analogico-digitale a 12 bits. Quando devono essere introdotti più di otto bits d'informazione, i bits vengono suddivisi in gruppi di otto bits ciascuno. Nel caso del convertitore a 12 bits, ci saranno due gruppi, l'uno formato da 8 bits, l'altro dai rimanenti quattro bits. Analogamente, un valore di 16 bits richiederà due porte d'ingresso, esattamente come un valore di 9 bits. Quando non si utilizzano tutti gli otto bits di una porta d'ingresso, i bits non utilizzati vengono posti generalmente nello stato logico zero mettendoli a massa, o a zero logico. Se non è possibile determinare lo stato dei bits non utilizzati, vuol dire che probabilmente essi non sono stati effettivamente messi a zero logico nel circuito relativo alla porta d'ingresso. Questi bits possono essere "eliminati" con opportuni comandi software, che "mascherano" i bits non utilizzati: questi assumeranno così il valore zero.

Dal momento che un valore di 12 bits può rappresentare dei valori decimali compresi fra 0 e 4095, occorre trovare dei metodi per convertire i singoli bytes introdotti in un valore unico. Supporremo che gli otto bits meno significativi siano stati acquisiti come un unico byte dalla porta 49312, e che i quattro bits più significativi siano stati acquisiti dalla porta d'ingresso 49313, in corrispondenza delle posizioni dei bits D3-D0. Supporremo inoltre che i bits non utilizzati alla porta 49313 siano stati messi a massa, assumendo così il valore logico zero.

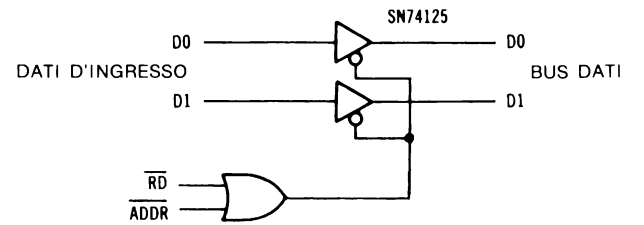


Figura 3.13 — Porta d'ingresso a due bits.

Una volta definita la configurazione delle porte d'ingresso, vediamo come viene manipolata l'informazione allo scopo di ricostruire il valore originario, partendo dai due distinti bytes di dato letti dalle due porte d'ingresso (v. Figura 3.13).

Dal momento che i bits meno significativi possono rappresentare valori, compresi fra 0 e 255, relativi al dispositivo d'interfaccia a 12 bits, questi bits non richiedono alcuna "conversione": infatti l'Apple si limiterà ad acquisire questi otto bits ed a convertirli in un valore compreso fra 0 e 255.

A questo punto, se i quattro bits più significativi vengono considerati separatamente dagli altri bits, convertendoli in decimale si avranno valori compresi fra 0 e 15, invece dei loro valori posizionali originari 0, 256, 512, e così via. Allora questi bits sono "sfasati" di un fattore 256, perché il valore del dato di 12 bits ha dovuto essere "spezzato" in due parti più piccole per poter essere acquisito dall'Apple. Non dimentichiamo che ogni valore di otto bits acquisito dal calcolatore Apple sarà convertito automaticamente in un numero decimale dal valore compreso fra 0 e 255.

Una volta che i due valori sono stati introdotti nell'Apple, è una cosa semplicissima "ricostruire" il dato: moltiplicando l'informazione relativa ai quattro bits più significativi per 256, e poi sommando il risultato ottenuto al valore relativo agli otto bits meno significativi, si otterrà un valore compreso fra 0 e 4095 incluso, che è il valore originariamente presente come valore binario a 12 bits sul dispositivo d'interfaccia. La routine software completa è riportata nell'Esempio 3.3.

ESEMPIO 3.3 — Programma di conversione di un ingresso a 12 bits

```
10 A = PEEK(49312)
20 B = PEEK(49313)
30 C = (B*256) + A
40 PRINT C
```

Il programma si può semplificare riunendo tutti i passaggi in un'unica linea:

```
10 PRINT (PEEK(49313)*256) + PEEK(49312)
```

Questo semplice programma stamperà il numero decimale equivalente al valore binario a 12 bits che si trovava sul dispositivo periferico o d'interfaccia al momento dell'esecuzione del programma. Il programma può servire per realizzare interfacciamenti con porte di uscita binarie da 9 a 16, ma badando a mettere a massa i bits non utilizzati. Negli esperimenti vedremo un altro metodo di "mascheratura", o azzeramento dei bits.

Le porte d'ingresso servono a trasferire informazioni dalle periferiche esterne al calcolatore. Queste informazioni possono rappresentare valori relativi a peso, temperatura, resistenza, etc., o possono essere interpretate come singoli bits binari corrispondenti allo stato (aperto o chiuso) di determinati dispositivi: ad esempio,

vuoto/pieno, libero/occupato, etc. Fra le applicazioni tipiche delle porte d'ingresso citiamo ad esempio:

- il trasferimento al calcolatore d'informazioni relative ad un rilevatore di traffico;
- il trasferimento al calcolatore di valori digitali provenienti da uno strumento;
- il trasferimento di bits di stato (aperto/chiuso) da una stampante al calcolatore.

Nelle applicazioni d'interfacciamento, il requisito principale per le porte d'ingresso è che le loro uscite abbiano tre stati, in modo da non provocare conflitti sul bus dati quando vengono utilizzate.

CAPITOLO 4

FLAGS E DECISIONI

In quasi tutti gli esempi visti finora abbiamo supposto che non vi siano problemi di sincronizzazione fra il calcolatore ed i dispositivi esterni di I/U. Abbiamo pertanto supposto che le porte di uscita siano sempre pronte a ricevere sequenze di dati destinate ad essere trasferite ad esse; quanto alle porte d'ingresso, abbiamo poi visto i valori dei dati come presenti e pronti per essere trasferiti al calcolatore, non appena questo trovi in un programma un comando di PEEK. In realtà le cose non vanno sempre così; spesso infatti si ha a che fare con dispositivi di I/U più lenti del calcolatore.

LA SINCRONIZZAZIONE DEI DISPOSITIVI DI I/U

Dato che non tutti i dispositivi di I/U possono essere a disposizione del calcolatore in qualunque momento, occorre trovare un metodo per sincronizzare il calcolatore ed i dispositivi di I/U. Generalmente la sincronizzazione implica l'uso di segnali che prendono il nome di *indicatori* (flags). Questi segnali indicano se determinati dispositivi sono nello stato di occupato o no, di pronto o di non pronto, in condizione di convertire o no, e così via. Quindi gl' "indicatori" segnalano lo stato di un dispositivo, e vengono perciò detti anche *indicatori di stato* o *flags di stato.*

A scopo illustrativo, supponiamo di voler interfacciare un dispositivo ad un calcolatore Apple: il dispositivo in questione fornirà al calcolatore valori di dati di 8 bits in modo non uniforme. Nella maggior parte dei casi dispositivi di questo tipo generano anche un segnale di flag che indica se il dispositivo è pronto a trasferire le proprie informazioni al calcolatore. Un dispositivo siffatto è rappresentato in Figura 4.1. Si osservi che per trasferire l'informazione al calcolatore si è usata una porta d'ingresso standard a tre stati. Il flag di READY (pronto) pone un problema interessante: come fa il calcolatore a controllare ed a verificare la condizione del flag di READY, in modo da poter determinare quando un nuovo valore di dato è disponibile?

Come si è detto prima, non esiste nessuna regola che limiti il trasferimento d'informazioni all'effettivo trasferimento di valori numerici. Il calcolatore non ha modo di sapere se il valore ad 8 bits 01100100_2 rappresenta 100, piuttosto che cinque dispo-

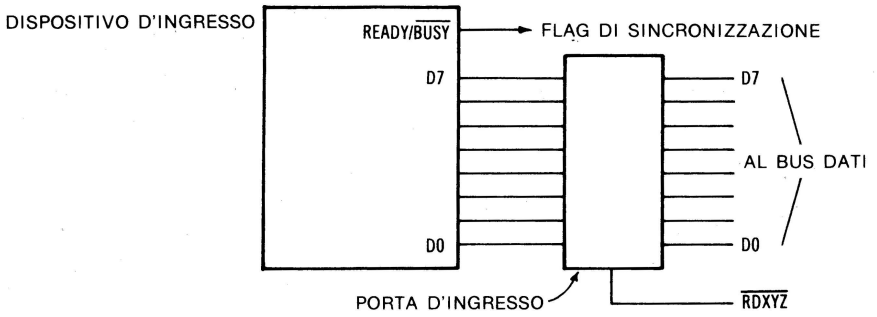


Figura 4.1 — Semplice dispositivo d'ingresso con flag di sincronizzazione in uscita.

sitivi nello stato di aperto (off) e tre dispositivi nello stato di chiuso (on). Quindi un'altra porta d'ingresso potrà andar benissimo come via per trasferire al calcolatore l'informazione del flag di stato relativa al dispositivo d'ingresso. Gli altri sette bits relativi a questa porta d'ingresso possono restare inutilizzati, o essere utilizzati per indicare lo stato di altri dispositivi esterni. In tal modo si può ricorrere ad un programma per verificare la condizione, ovvero lo stato, dei dispositivi esterni.

Quando si verifica con un programma lo stato di un flag, si può programmare il calcolatore o in modo che attenda che il flag passi ad un determinato stato prima di passare a svolgere le operazioni richieste, oppure in modo che verifichi periodicamente il flag, proseguendo nel frattempo con altre operazioni.

Sia il linguaggio assemblatore che il BASIC prevedono delle operazioni logiche con le quali si può verificare lo stato di singoli flags, cioè bits, di una parola-dato di 8 bits. In questo modo possiamo rilevare il reale stato logico (zero o uno) di un flag: il calcolatore quindi prende una decisione, sulla base dello stato del flag in questione.

OPERAZIONI LOGICHE E FLAGS

L'operazione logica più utile, quando si ha a che fare con la rilevazione dello stato di un flag, è probabilmente l'operazione logica di AND. Ricorderete che due bits A e B possono venir messi in AND (v. Figura 4.2).

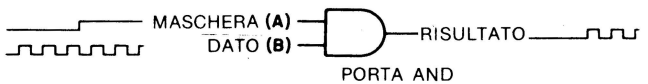


Figura 4.2 — Rappresentazione dell'operazione logica di AND, in cui da DATO e MASCHERA si ottiene RISULTATO.

Il risultato indica che solo quando tutti e due i bits sono degli uni logici il risultato sarà un uno logico. Un altro modo di considerare quest'operazione consiste nel vedere il bit "A" come una "maschera", ed il bit "B" come informazione, o dato: quando la maschera è zero, il risultato è zero; quando la maschera è uno, il dato vien fat-

to passare attraverso la porta. In questo modo i bits selezionati possono essere mascherati, mentre gli altri vengono "fatti passare attraverso" la maschera.

Se, ad esempio, volessimo verificare lo stato del bit D5 contenuto nella parola-dato 00111010, potremmo ricorrere alla maschera 00100000: la maschera viene messa in AND con la parola-dato, come si vede in Figura 4.3, per un certo numero di parole - dato.

VALORE	00111010	00011010	11110000	00011111
MASCHERA	00100000	00100000	00100000	00100000
RISULTATO	00100000	00000000	00100000	00000000

Figura 4.3 — Esempi di operazioni di AND nelle quali si lavora con otto bits d'informazione.

In tutti i casi lo stato logico di D5 passa nel bit D5 del risultato, mentre tutti gli altri bits sono mascherati, cioè posti a zero. Così procedendo il risultato globale è zero quando il bit D5 è zero, diverso da zero quando il bit D5 è uno.

Questo modo di operare può essere seguito, nei suoi principi, quando occorre scrivere un programma col quale prendere una decisione. Prima di usare le maschere in un programma BASIC, non dimenticate di convertirle nei loro equivalenti decimali: nel caso del bit D5, l'equivalente decimale è 32.

RILEVAZIONE DELLO STATO DEI FLAGS

Una volta che sia stata costruita un'interfaccia tale per cui si possano rilevare gli stati dei vari flags, così come si vede in Figura 4.4, si può ricorrere a programmi con cui prendere decisioni in base allo stato di questi flags.

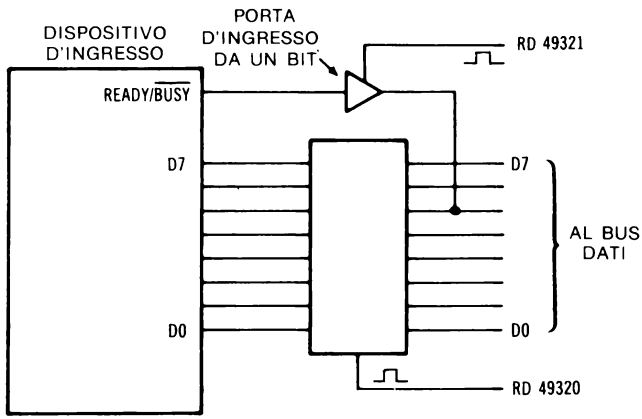


Figura 4.4 — Interfacciamento completo, nel quale lo stato del flag è rilevato mediante software.

Alcuni dialetti del BASIC prevedono delle operazioni logiche che eseguono operazioni di AND direttamente fra i singoli bits, come quelle di Figura 4.3: in questi casi si può ricorrere a semplici espressioni, all'interno di programmi BASIC, per eseguire le operazioni di AND fra due parole-dato costituite da valori compresi fra 0 e 255. (Non dimentichiamo che quelli che vengono di fatto sottoposti alle operazioni di AND sono gli *equivalenti binari*.) Gli Esempi 4.1 e 4.2 mostrano come si possano utilizzare le operazioni di AND per rilevare un flag acquisito come bit D5 da una porta d'ingresso, la porta 49321.

ESEMPIO 4.1 — Lo stato logico zero di un flag è usato a scopo di controllo

```
4010  A = PEEK(49321)
4020  IF (A AND 32) = 0 THEN 200
4030  ... Continua qui se stato flag = uno logico
```

ESEMPIO 4.2 — Lo stato logico uno di un flag è usato a scopo di controllo

```
4010  A = PEEK(49321)
4020  IF (A AND 32) > 0 THEN 200
4030  ... Continua qui se stato flag = zero logico
```

In entrambi i casi, quando s'incontra l'opportuna condizione, il programma acquisirà probabilmente dati da una porta d'ingresso, o svolgerà altre operazioni segnalate dalla presenza del flag.

Purtroppo il calcolatore Apple non utilizza in questo modo le sue istruzioni logiche: nell'Apple un'operazione di AND può aver luogo soltanto fra due distinte condizioni di vero o falso, per cui è molto difficile mascherare otto bits per determinare lo stato di uno solo di essi. A meno di essere disposti a perdere moltissimo tempo in una routine BASIC complessa, dobbiamo ricorrere ad una subroutine in linguaggio assemblatore, che eseguirà le operazioni logiche piuttosto rapidamente. Dal momento che è facile scrivere delle routines in linguaggio assemblatore per l'Apple, vale la pena di soffermarsi alquanto su questo punto, presentando in particolare qualche routine semplice e facile da usare.

LE OPERAZIONI LOGICHE IN LINGUAGGIO ASSEMBLATORE

L'insieme delle istruzioni in linguaggio assemblatore per il microprocessore 6502 comprende un'operazione di AND e una di OR. Ciascuna di queste istruzioni opera su due bytes da 8 bits, e dà come risultato un unico byte. Noi quindi dobbiamo scrivere una breve routine che esegua l'operazione.

L'Apple mette a disposizione un certo numero di locazioni di memoria di lettura/scrittura "di riserva" nella pagina di memoria 03H, ed è in questa pagina che abbiamo deciso di allocare le nostre routines, che saranno così indipendenti dalla grandezza totale di memoria del vostro calcolatore. La Tabella 4.1 fornisce il listato completo della routine, con gl'indirizzi esadecimali e decimali ed i valori di dato/istruzione. Per usare questa routine non c'è bisogno di essere esperti di programmazione in linguaggio assemblatore, ma in ogni caso abbiamo inserito dei commenti, per permettervi di seguire, se volete, l'operatività del programma.

Byte indirizzo		**Byte dato**		
Esadec.	**Dec.**	**Esadec.**	**Dec.**	
0300	768	—	—	Qui si trova il byte MASCHERA
0301	769	—	—	Qui si trova il byte DATO
0302	770	—	—	Qui è messo RISULTATO
0303	771	48	72	PHA salva il reg. A
0304	772	AD	173	LDA carica il reg. A con
0305	773	00	0	la locazione MASCHERA
0306	774	03	3	
0307	775	2D	45	AND fra il reg. A e DATO*
0308	776	01	1	
0309	777	03	3	
030A	778	8D	141	STA memorizza il risultato nella
030B	779	02	2	locazione RISULTATO
030C	780	03	3	
030D	781	68	104	PLA ripristina il reg. A
030E	782	60	96	RTS rientra nel BASIC

* Nel caso di un'operazione di OR, sostituire con 0DH, o 13 decimale.

Tabella 4.1 — Subroutine logica in linguaggio assemblatore.

Tre locazioni di memoria di lettura/scrittura sono adibite alla memorizzazione temporanea dei differenti bytes di dato, denominati MASCHERA, DATO e RISULTATO. Nella locazione MASCHERA si mette il byte di maschera, nella locazione DATO il byte sul quale si deve effettuare l'operazione. Eseguita l'operazione logica, la locazione RISULTATO conterrà il risultato.

Volendo usare questa routine, bisogna mettere l'informazione di MASCHERA nell'indirizzo 768, ed il byte di DATO nell'indirizzo 769: a tal fine si può ricorrere a delle operazioni di POKE. Fatto questo, non si deve far altro che richiamare la subroutine in linguaggio assemblatore, che eseguirà l'operazione. Come si procede in quest'ultima fase?

La chiamata di una subroutine in linguaggio assemblatore da un programma in

BASIC è un'operazione abbastanza semplice: lavorando con un calcolatore Apple, basta collocare un'istruzione di salto di tre bytes in tre locazioni successive, d'indirizzo 10, 11 e 12, o 0A, 0B e 0C in notazione esadecimale. Ora, la nostra routine inizia all'indirizzo 771 (0303H), per cui le informazioni che seguono vanno collocate in queste tre locazioni: un 76 nell'indirizzo 10, un 3 nell'indirizzo 11 ed un 3 nell'indirizzo 12. Una volta che queste informazioni d'indirizzo sono state collocate in queste tre locazioni, si può accedere alla subroutine in linguaggio assemblatore per mezzo di una funzione USR. Nel nostro esempio, prima si dovrà collocare l'informazione relativa a MASCHERA e a DATO, e *poi* usare la funzione USR (V. Esempio 4.3).

ESEMPIO 4.3 — Chiamata di una subroutine che esegue operazioni logiche

```
1590  POKE 768,32: POKE 769,129
1594  Q = USR(5)
```

Qui il valore 32 rappresenta il byte di maschera, e 129 è il valore che deve essere messo in AND con esso. Q è una variabile "fittizia" richiesta dalla funzione USR, ed il valore 5 è un valore "fittizio" che non ha nessun effetto sulla subroutine. Al posto di Q si può usare una qualunque variabile, purché non sia una variabile che compare anche altrove, ed al posto di 5 si può usare un valore qualunque, ad esempio 0.

Dopo aver richiamato la subroutine in linguaggio assemblatore, il risultato sarà posto nella locazione 770, e mediante un'operazione di PEEK sarà possibile accedere ad esso. Il programma dell'Esempio 4.4 mostra l'intero iter d'impiego della subroutine: abbiamo supposto che la subroutine sia stata caricata tramite il Monitor. In questo esempio, l'istruzione di salto di tre bytes è caricata per mezzo di operazioni di POKE.

ESEMPIO 4.4 — Utilizzo della subroutine che esegue operazioni logiche

```
2030  POKE 10,76: POKE 11,3: POKE 12,3
2040  POKE 768,32: POKE 769,PEEK(49321)
2050  Q = USR(7)
2060  IF PEEK(770) > 0 THEN 3460
2070  ... Continua qui se stato flag = 0
```

In questo esempio il dato da usare nell'operazione logica si ottiene da una porta d'ingresso per mezzo di un comando di PEEK in cui è specificato l'indirizzo relativo al dispositivo.

Con la stessa subroutine si può eseguire anche un'operazione di OR: basterà semplicemente sostituire il codice operativo (op-code) di AND, che è 2DH con 0DH. Anche in questo caso è possibile utilizzare un'operazione di POKE, immediatamente prima di dare il via alla subroutine.

Dunque la subroutine della Tabella 4.1 può eseguire entrambe le operazioni logiche.

Dovreste essere capaci di caricare la subroutine nella memoria di lettura/scrittura per mezzo del Monitor dell'Apple. Comunque, per delucidazioni circa il Monitor vi rimandiamo all'*Apple II Reference Manual*. Per caricare le varie istruzioni del programma, potreste usare anche dodici comandi di POKE, ma questo provocherebbe degli errori.

È un peccato dover ricorrere al linguaggio assemblatore per eseguire delle operazioni logiche, facilmente disponibili in altri dialetti del BASIC. D'altra parte il programma in linguaggio assemblatore è abbastanza semplice, ed offre un esempio elementare di utilizzo dei programmi di questo tipo, e di come vengano richiamati da un programma BASIC. E se non siete abituati a programmare in linguaggio assemblatore, può darsi che quest'esempio ve ne abbia fatto venire la voglia.

FLAGS COMPLESSI

A questo punto potrete chiedere: se il flag presente sul dispositivo d'ingresso illustrato in Figura 4.4 serve ad indicare che un valore di 8 bits è disponibile, come fa il dispositivo a sapere quando il calcolatore ha acquisito, o accettato, il valore che si è reso disponibile? A volte è usato un segnale dal calcolatore al dispositivo di I/U per indicare che il flag è stato rilevato, e che l'operazione richiesta ha avuto luogo: questo segnale "azzera" il flag. L'operazione di azzeramento del flag può essere effettuata da un segnale distinto, che può essere anche lo stesso che controlla la porta d'ingresso per l'acquisizione dei dati. Ciò è illustrato in Figura 4.5

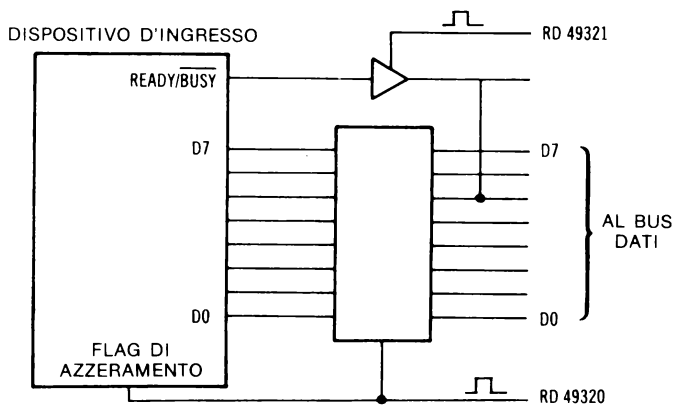


Figura 4.5 — Circuito nel quale il flag è azzerato da un impulso generato dal calcolatore.

In Figura 4.6 invece compare un semplice diagramma di temporizzazione.

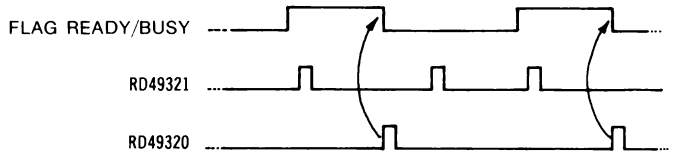


Figura 4.6 — Diagramma di temporizzazione del flag.

Quando il flag si trova nello stato logico uno, vuol dire che il dispositivo è pronto a trasferire un byte al calcolatore. L'impulso RD 49321 trasferisce l'informazione relativa allo stato del flag al calcolatore: quando il calcolatore interroga il flag, e lo trova allo stato logico uno, esegue quelle istruzioni che portano al trasferimento del dato dal dispositivo al calcolatore stesso. Qui l'impulso RD 49320 abilita i buffers a tre stati al momento giusto, e inoltre azzera il circuito interno di flag del dispositivo.

Il secondo impulso RD 49321 legge di nuovo lo stato del flag, ma, essendo ora questo allo stato di zero logico, il calcolatore non effettua altre operazioni. Quando il flag è interrogato per la terza volta, è trovato nello stato di uno logico: il dato è allora trasferito al calcolatore ed il flag azzerato. Nell'Esempio 4.5 riportiamo una semplice sezione di programma utilizzabile per controllare un'interfaccia. Abbiamo supposto che la subroutine che esegue l'operazione logica di AND sia stata caricata insieme con il puntatore di tre bytes.

ESEMPIO 4.5 — Semplice programma d'interrogazione di un flag

```
1050  POKE 768,32: POKE 769,PEEK(49321)
1060  Q = USR(0)
1070  IF PEEK(770) = 0 THEN 50
1080  D = PEEK(49320)
1090  ... Continua qui dopo l'acquisizione del dato
```

I flags sono utilizzati in questo modo tipicamente da dispositivi come tastiere, floppy disks, convertitori analogico-digitali, ed altri che forniscono bytes di dati in periodi di tempo non regolari.

LA CIRCUITERIA DI FLAG

In alcuni casi, i dispositivi non possono avere in sé i circuiti necessari per un facile controllo dei flags, oppure non generano livelli logici che si mantengano stabili per periodi relativamente "lunghi", abbastanza perché il calcolatore possa rilevarli correttamente. In questi casi, accade che il "flag" sia un impulso di durata estremamente breve. Infatti ci sono degl'impulsi di flag che durano troppo poco per poter es-

sere rilevati dal calcolatore, come è nel caso in cui vengono acquisiti semplicemente tramite una porta d'ingresso a tre stati.

In questi casi è necessario che il circuito venga progettato in modo da "catturare" l'impulso di flag, che potrà così essere rilevato dal calcolatore in momenti successivi. Anche se il calcolatore è in grado d'interrogare un bit di flag ad intervalli di qualche millisecondo, spesso "perderà" impulsi brevi, della durata di qualche microsecondo.

Per "ricordare" la presenza d'impulsi di flag si ricorre in genere a circuiti di flip-flop o di latch. Dispositivi a flip-flop tipici sono l'SN7474 di tipo D e l'SN7476 di tipo J-K. Se avete bisogno di rivedere il modo di funzionamento dei flip-flops, la maggior parte dei testi introduttivi di elettronica digitale trattano questo argomento.

In Figura 4.7 è illustrato un tipico circuito di flag basato su un flip-flop.

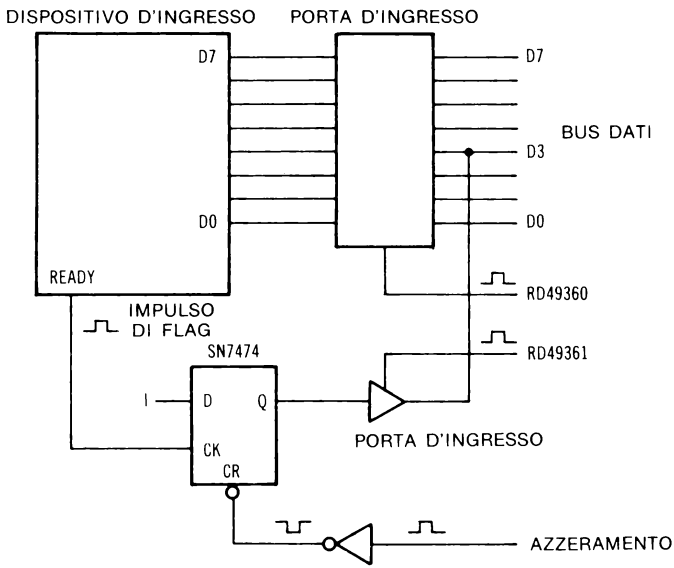


Figura 4.7 — Circuito a flip-flop per la rilevazione dell'impulso di flag.

In questo circuito il dispositivo d'ingresso genera un impulso di READY che, utilizzato come ingresso di clock del flip-flop, trasferisce il livello logico dell'ingresso D all'uscita Q. L'uscita Q è rilevata dal calcolatore per mezzo di una porta d'ingresso distinta dalla porta d'ingresso usata per il trasferimento degli otto bits di dato. Il calcolatore interroga facilmente lo stato del bit di flag, come si è visto. Una volta che le azioni necessarie sono state effettuate — in questo caso, l'acquisizione del dato dal dispositivo d'ingresso —, il flip-flop viene azzerato per mezzo di un impulso negativo di $\overline{\text{CLEAR}}$ applicato all'ingresso CR del flip-flop. Per quanto l'impulso RD 49360 usa-

to per controllare la porta d'ingresso ad 8 bits possa servire per azzerare il flip-flop, abbiamo evidenziato un segnale di azzeramento distinto, al fine di visualizzare le relazioni di tempo, in Figura 4.8.

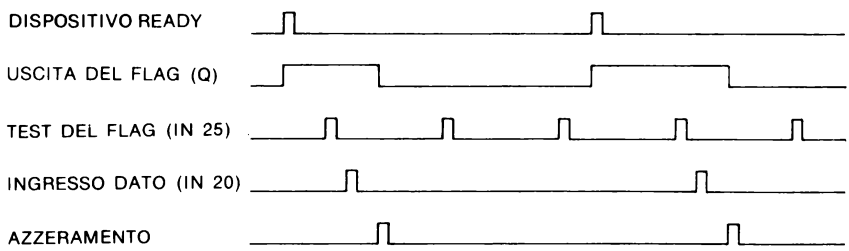


Figura 4.8 — Diagramma di temporizzazione relativo ad un flip-flop di flag.

Nel diagramma di temporizzazione l'impulso di READY pilota il flip-flop in modo che l'uscita Q di quest'ultimo sia un uno logico. L'uscita è rilevata quando l'informazione relativa al flag di stato è acquisita dalla porta 49361. Il fatto che il flag si trovi nello stato logico uno fa sì che vengano eseguite quelle istruzioni del programma che portano all'acquisizione del byte di dato, e poi all'azzeramento del flag.

Il segnale distinto di $\overline{\text{CLEAR}}$ potrà venir generato da un comando di POKE e da una circuiteria adatta, anche se è probabilmente più semplice ricorrere all'impulso RD 49360, subito disponibile.

In quest'esempio il flag è interrogato due volte mentre si trova nello stato logico zero. Poiché questo indica che non è pronto nessun dato, non viene dato il via ad operazioni né di acquisizione dato né di azzeramento flag.

Molti degli esperimenti del Capitolo 6 riguardano l'uso dei flags.

FLAGS MULTIPLI

Molti sistemi hanno un certo numero di flags, che vanno interrogati regolarmente. In alcuni casi occorre istituire una priorità, perché alcuni dispositivi sono più importanti, o richiedono maggiore attenzione di altri. La priorità è facilmente stabilita nel programma: infatti l'ordine nel quale i vari bits vengono interrogati determina quali dispositivi vadano "serviti" prima di altri. Le linee di programma dell'Esempio 4.6 verificano in sequenza una serie di bits di flag, che vanno dal bit D7 al bit D5, precisando la priorità con cui i dispositivi corrispondenti saranno serviti dal calcolatore.

ESEMPIO 4.6 — Linee di programma che determinano la priorità dei flags

```
300  POKE 769,PEEK(54098): POKE 768,128: Q = USR(0)
305  IF PEEK(770) > 0 THEN 1050
310  POKE 768,64: Q = USR(0)
315  IF PEEK(770) = 0 THEN 20
```

```
320  POKE 768,32: Q = USR(0)
325  IF PEEK(770) = 0 THEN 1010
330  ... E così via per gli altri bits
```

In quest'esempio, il flag relativo al bit D7 viene interrogato se è un uno logico, mentre gli altri due flags sono interrogati se sono degli zeri logici.

Potremmo aggiungere altre istruzioni per la rilevazione dello stato dei bits relativi ad altri flags, come pure modificare ad un dato momento l'ordine d'interrogazione dei bits, con una semplice modifica del programma che rifletta il nuovo ordine. Si tenga presente che i dati interessati dall'operazione di AND non cambiano, ma devono essere semplicemente acquisiti dalla porta d'ingresso all'inizio della sequenza d'istruzioni.

LE INTERRUZIONI

In alcuni casi è necessario che un dispositivo di I/U venga servito non appena è pronto: può darsi che esso non sia in grado di restare in attesa per diversi millisecondi (o anche più), che il calcolatore può impiegare per interrogare i flags e per prendere delle decisioni in base al loro stato. Quasi tutti i calcolatori hanno almeno un ingresso d'interruzione, che permette di "richiedere" al calcolatore il servizio immediato, senza badare all'operazione che sta svolgendo in quel momento. Il processore 6502 usato nel calcolatore Apple ha due ingressi d'interruzione: un ingresso di richiesta d'interruzione (IRQ), ed un ingresso d'interruzione non mascherabile (NMI). L'ingresso IRQ è sensibile allo zero logico, mentre l'ingresso NMI è *sensibile al fronte*, scattando quando c'è passaggio dall'uno logico allo zero logico. Quest'ingressi non sono utilizzati nella configurazione base del calcolatore Apple, ma si possono avere facilmente ai connettori d'interfaccia interni, per cui si possono usare con periferiche ed interfacce aggiuntive.

Se un dispositivo richiede un servizio particolarmente rapido, tanto da rendere necessario l'impiego di un'interruzione, bisognerà evidentemente programmare in linguaggio assemblatore. Dal momento che questo non rientra nel tema di questo libro, rimandiamo a: *Programming & Interfacing the 6502, With Experiments*, e *6502 Software Design*, entrambi della Howard W. Sams & Co., Inc., Indianapolis, IN 46268. I due testi trattano dettagliatamente dell'uso delle interruzioni, con esempi e programmi in linguaggio assemblatore per il controllo delle interruzioni.

Le interruzioni IRQ ed NMI dell'Apple utilizzano delle locazioni di memoria specifiche, dalle quali il processore 6502 "preleva" l'indirizzo della subroutine da usare come routine di servizio di ciascuna interruzione. L'IRQ utilizza le locazioni FFFEH ed FFFFH, l'NMI le locazioni FFFAH ed FFFBH. Essendo queste locazioni contenute nei chips della memoria di sola lettura che contengono l'interprete BASIC ed il Monitor, gl'indirizzi scritti in queste quattro locazioni sono fissi, e non è possibile cambiarli. Va detto comunque che quest'indirizzi fissi servono semplicemente a puntare ad altre locazioni della memoria di lettura/scrittura, per cui è effettivamente

possibile cambiare i puntatori relativi alle subroutines di servizio delle interruzioni.

Per informazioni dettagliate sull'uso di queste locazioni di "vettore", vi consigliamo di consultare l'*Apple II Reference Manual.*

ANCORA QUALCHE OSSERVAZIONE

Prima di concludere il capitolo, è necessario fare qualche ulteriore osservazione.

Abbiamo presentato una semplice subroutine in linguaggio assemblatore, che esegue le operazioni logiche di AND e di OR su bytes di 8 bits, ed abbiamo insieme illustrato l'uso dell'operazione di chiamata della subroutine in linguaggio assemblatore, cioè USR. In realtà, nell'insieme delle istruzioni proprie del calcolatore Apple esiste un comando di verifica dei flags: il comando di WAIT, utilizzabile per verificare flags singoli o gruppi di flags, e capace altresì di rilevarne lo stato logico (uno o zero). Comunque il loro uso è soggetto ad una limitazione: se non è rilevata l'appropriata configurazione di flag, non si può in alcun modo uscire dall'attività di test, ma si deve azzerare (tramite RESET) il calcolatore, al fine di riaverne il controllo. Allo stesso modo, non è possibile decidere di saltare ad una parte del programma se il flag o i flags non sono stati posizionati ad un dato valore, ed in un'altra direzione nel caso opposto. Utilizzando il comando di WAIT si continuerà semplicemente ad aspettare (wait) finché non viene raggiunta la condizione: è un meccanismo assolutamente inflessibile, ed è per questo che abbiamo deciso di evitare l'uso del comando di WAIT.

Abbiamo introdotto il comando di USR, per la chiamata di subroutines in linguaggio assemblatore; se, con ulteriori letture, approfondirete ed espanderete le nozioni sulla programmazione in linguaggio assemblatore, constaterete tutta l'utilità di quest'istruzione. Se invece v'interessa soltanto di sapere come accedere ad una subroutine in linguaggio assemblatore, del tipo della subroutine che esegue l'operazione di AND logico, potete ricorrere al comando di CALL, seguito dall'indirizzo decimale dell'inizio della subroutine stessa. Per chiamare la subroutine di AND logico si può ricorrere all'operazione CALL 771. Inutile dire che, prima di chiamare la subroutine, bisogna, con delle POKE, scrivere i bytes MASCHERA e DATO.

Abbiamo voluto illustrare qui un po' più diffusamente la potenza del calcolatore Apple, ed il modo in cui l'Apple riesce ad eseguire operazioni diverse, nella convinzione che la via più semplice non sempre è la più interessante e la più utile dal punto di vista pedagogico.

CAPITOLO 5

LAVORIAMO CON LA SCHEDA DI PROVA

Abbiamo sempre ritenuto che i calcolatori debbano essere facili da usare, sia per lo sviluppo di programmi, sia per lo sviluppo hardware, cioè per l'interfacciamento. Dal momento che i segnali necessari per interfacciare la maggior parte dei calcolatori sono disponibili in un punto o nell'altro del calcolatore stesso, abbiamo pensato di sviluppare dei circuiti d'interfaccia d'uso generale, che potrebbero essere utilizzati con svariati calcolatori. Questi circuiti sono semplicissimi, e si possono facilmente costruire e adattare a molti calcolatori, oltre all'Apple. Abbiamo sviluppato un circuito stampato contenente tutti i circuiti necessari per l'interfacciamento. La Figura 5.1 è una fotografia dell'interfaccia.

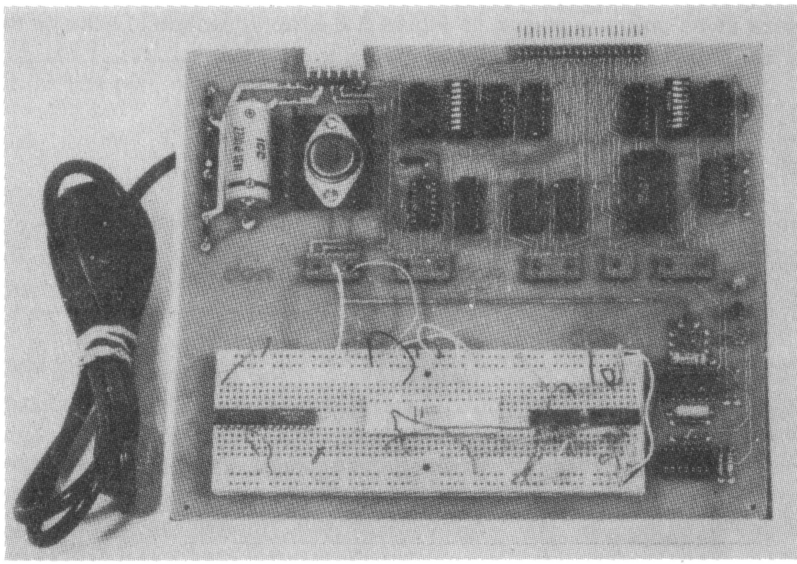


Figura 5.1 — La scheda di prova usata con l'Apple.

Un cavo piatto standard a 40 conduttori è usato per collegare la scheda di prova a vari calcolatori.

I circuiti per l'interfacciamento si sarebbero potuti mettere sulla scheda di prova, per poi usarli negli esperimenti, ma questo, a nostro parere, avrebbe fatto sorgere ulteriori problemi.

CONFIGURAZIONE BASE DELLA SCHEDA DI PROVA

La scheda di prova (breadboard in inglese) nella sua configurazione base ospita un certo numero di utili circuiti che permettono di allestire e verificare facilmente progetti d'interfacciamento. Si divide fondamentalmente in cinque parti: alimentazione, sonda logica, decodifica indirizzi, buffers bus e circuiteria di controllo.

L'alimentazione

La sezione alimentazione della scheda di prova può operare in due modi: si può ricorrere ad un'alimentazione esterna di +5 volts, e corrente massima erogabile di 1 ampère, oppure ad un trasformatore, sempre esterno, in grado di fornire 12.6 volts (alternati) ai circuiti di alimentazione presenti sulla scheda. In entrambi i casi l'alimentazione della scheda di prova è distinta da quella del calcolatore.

Spesso si ricorre ad un'alimentazione separata perché alcuni calcolatori non forniscono corrente sufficiente per i propri circuiti *e* per gli ulteriori circuiti d'interfaccia che si può desiderare di provare. Ogniqualvolta si utilizza l'alimentazione esterna, *ci si deve accertare che le due alimentazioni abbiano un buon collegamento a bassa resistenza verso un'unica massa.* In Figura 5.2 è rappresentato lo schema dell'alimentazione.

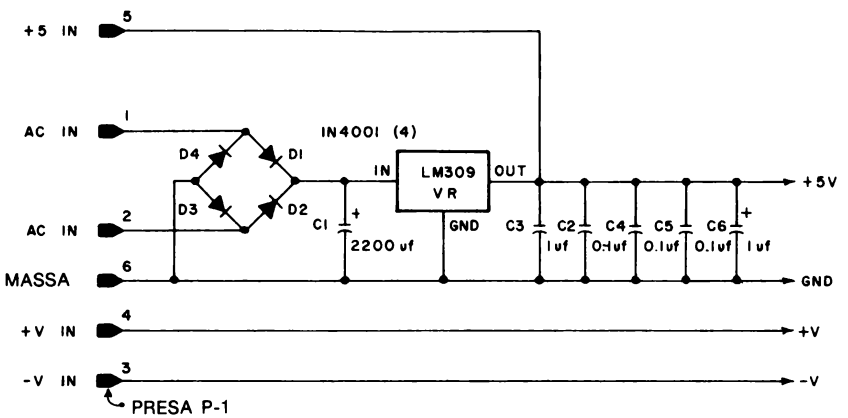


Figura 5.2 — Schema del circuito di alimentazione della scheda di prova.

Se si deve utilizzare l'alimentatore della scheda, il trasformatore di 12.6 V va collegato ai pins 1 e 2 presenti sulla presa numero 1 (P1); i diodi raddrizzatori D1-D4, il condensatore di filtro C1 ed il circuito di regolazione della tensione VR sono tutti installati sulla scheda. Consigliamo di usare, con il circuito di regolazione di +5 volts, un piccolo dissipatore termico. Quando la scheda di prova è utilizzata in questo modo, al pin 5 sono disponibili +5 volts, mentre la massa è disponibile al pin 6 di P1. Se è necessario, questi collegamenti possono essere utilizzati per dispositivi esterni.

Se invece si ricorre ad un'alimentazione di +5 volts distinta, le parti D1-D4, C1 e VR dell'alimentazione non sono necessarie, per cui dovrebbero essere tolte, oppure non installate. I collegamenti relativi a +5 volts ed a massa sono fatti rispettivamente ai pins 5 e 6 di P1.

Spesso sono necessarie tensioni diverse dai +5 volts, ad esempio ±12 oppure ±15 volts; a tal fine è previsto che su P1 si possano collegare delle ulteriori alimentazioni esterne. Le tensioni positiva (+V) e negativa (—V) sono collegate rispettivamente ai pins 4 e 3 di P1.

Tutte le tensioni sono disponibili anche sullo zoccolo che si trova nella posizione IC-16.

I collegamenti possibili sono elencati nella Tabella 5.1.

Pin*	Tensione disponibile
7,10	+5
5,12	GND
3,14	+V (esterna)
1,16	—V (esterna)

* Tutti gli altri pins non sono collegati.

Tabella 5.1 — Alimentazioni disponibili sullo zoccolo IC-16.

L'alimentazione per i circuiti integrati presenti sulla scheda a circuito stampato è derivata dall'alimentazione di +5 volts. I collegamenti relativi ad IC-16 permettono di avere facilmente l'alimentazione per gli esperimenti.

La sonda logica

Il circuito di sonda logica (vedi Figura 5.3) si usa quando si vuol determinare lo stato logico delle varie uscite, o anche per rilevare la presenza d'impulsi alle uscite. La parte sonda logica della scheda di prova ospita un rilevatore di livello ed un circuito rilevatore d'impulsi. Per rilevare i livelli uno logico e zero logico abbiamo usato un comparatore LM-319 (IC-15); per rilevare ed "allungare" gl'impulsi un SN74LS123 (IC-14). Come indicatore degli zeri logici abbiamo poi usato un LED (da light - emitting diode) verde (D-7), come indicatore degli uni logici un LED ros-

so (D-6), ed un LED giallo (D-5) come indicatore degli impulsi. L'ingresso alla sonda si ha ai pins 1-4 dello zoccolo IC-19: in figura questi ingressi sono indicati con "P". Tutti questi ingressi sono messi in parallelo, per cui ciascuno di essi può essere utilizzato, ma non provate a collegare la sonda logica con due segnali contemporaneamente. La sonda logica andrebbe vista come l'equivalente di due carichi low-power Schotty.

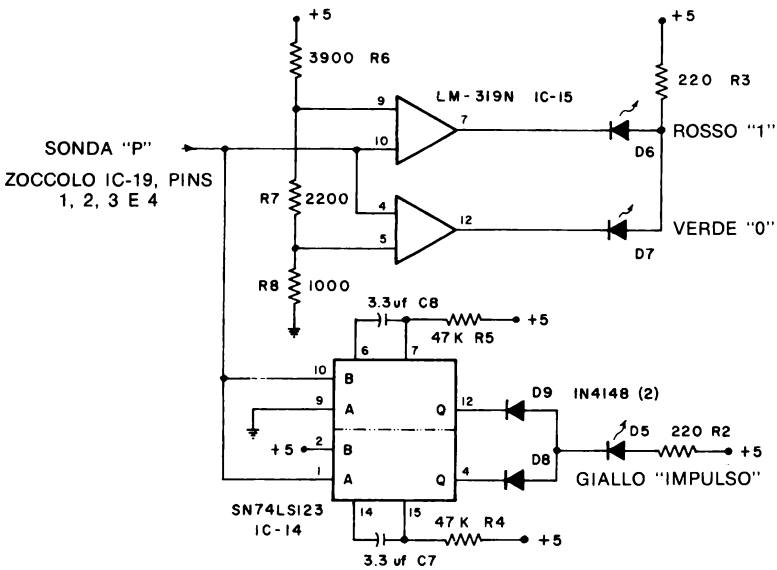


Figura 5.3 — Schema del circuito di sonda logica.

Se disponete di una sonda logica esterna, la circuiteria di cui si parla in questo paragrafo può non esservi necessaria, e quindi, volendo, potete fare a meno di costruire questa parte del circuito. In ogni caso però sarà bene essere capaci di rilevare gl'impulsi, lo stato degl'impulsi, e così via. Inoltre abbiamo verificato che la sonda logica è estremamente utile quando si devono scoprire dei guasti nei circuiti d'interfaccia posti sulla scheda di prova.

Decodifica degl'indirizzi

Una parte rilevante della circuiteria della scheda di prova è dedicata alla decodifica degl'indirizzi di I/U (vedi Figura 5.4).

I decodificatori possono funzionare o in modalità dispositivo o in modalità memoria, a seconda del tipo di calcolatore usato. Nell'indirizzamento in modalità dispositivo vengono decodificati soltanto i bits relativi alla parte bassa degl'indirizzi LO (A7-A0); nelll'indirizzamento in modalità memoria, invece, vengono decodificati tutti i

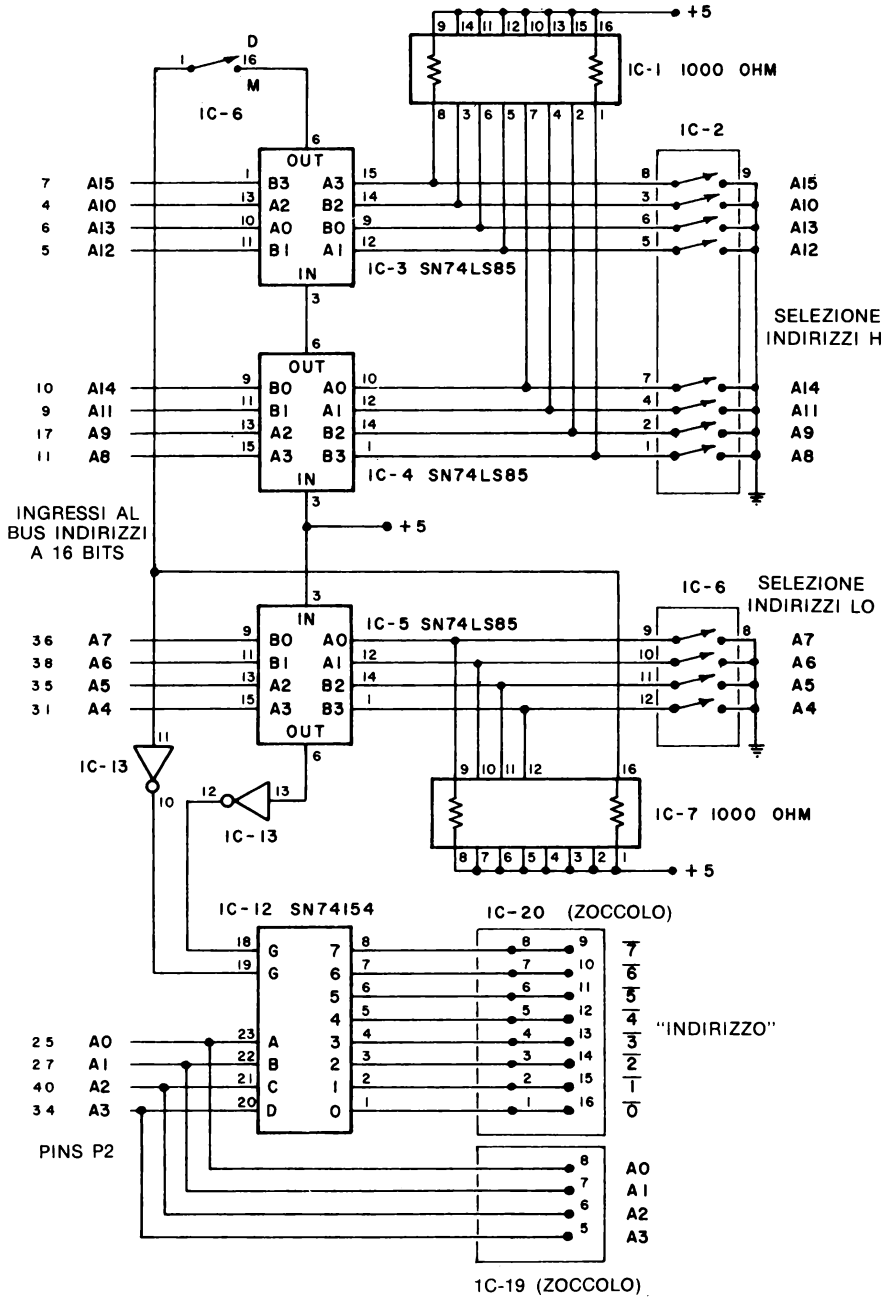


Figura 5.4 — Schema del circuito di decodifica degl'indirizzi.

bits d'indirizzo (A15-A0). Il calcolatore Apple si serve dell'indirizzamento memoria per identificare i dispositivi di I/U, perché si basa sul microprocessore 6502. Anche i calcolatori basati sul microprocessore 6800 ricorrono all'indirizzamento memoria. Invece i calcolatori costruiti attorno a chips appartenenti alle famiglie 8080, 8085 e Z-80 possono utilizzare entrambi i tipi d'indirizzamento. Osservando lo schema della Figura 5.4, dovreste notare che per la decodifica degl'indirizzi si ricorre ad un'associazione di comparatori e decodificatori digitali.

Nella modalità d'indirizzamento dispositivo si usa un comparatore a 4 bits SN74LS85 per confrontare i bits dell'indirizzo assegnato con i bits dell'indirizzo presente sulle linee A7-A4 del bus indirizzi LO. Gl'interruttori dell'IC-6 sono utilizzati per predisporre i livelli logici che dovranno venir confrontati con il bus indirizzi. Il package che si trova in IC-6 consiste in un insieme d'interruttori del tipo "dual-in-line": quindi la configurazione degl'interruttori è un'operazione che richiede attenzione. Le posizioni degl'interruttori sono chiaramente indicate con "7", "6", "5" e "4", con riferimento all'interruttore indicato con "LO". Installando l'interruttore, accertatevi che la posizione d'interruttore aperto (ovvero off) sia a destra (posizione di uno logico). I resistori di pull-up dell'IC-7 garantiscono che l'ingresso all'SN74LS85 sia costituito da un uno logico quando gl'interruttori sono aperti, vale a dire nella posizione di uno logico.

Quando si verifica una corrispondenza d'indirizzi fra i bits assegnati ed i bits d'indirizzo A7-A4, il decodificatore SN74154 (IC-12) è abilitato. Sebbene il decodificatore SN74154 abbia la capacità di decodificare i bits d'indirizzo A3-A0 in 16 distinte uscite d'indirizzo, abbiamo usato solo le prime 8, che sono più che sufficienti per le operazioni sulla scheda di prova e per le prove d'interfacciamento.

Quindi, se gl'interruttori d'indirizzo relativi ai bits 7-4 sono posti a 1011, il decodificatore decodificherà gl'indirizzi da 10110000_2 a 10110111_2, ovvero, in valore decimale, da 176 a 183. Nella modalità d'indirizzamento dispositivo, l'interruttore più basso in IC-6 dev'essere "aperto", vale a dire nella posizione "D": in questo modo il decodificatore opererà nel modo corretto.

Le uscite d'indirizzo decodificato si trovano sullo zoccolo IC-20, e sono indicate con "0", "1", e così via, fino a "7": l'intera sezione prende il nome d' "INDIRIZZO". I numeri d'indirizzo sono sopralineati, ad indicare che l'unico stato di uscita è un impulso negativo. La notazione d'indirizzo (cioè 0 ... 7) riflette un indirizzamento sequenziale che vi sarà di aiuto nel determinare quali pins sono collegati alle uscite d'indirizzo dispositivo. *Nella maggior parte dei casi i numeri non hanno nessun rapporto con gl'indirizzi effettivamente decodificati.* Nell'esempio d'indirizzamento che abbiamo appena portato, nel quale sono decodificati gl'indirizzi dal 176 al 183, l'uscita indicata con "0" corrisponderà all'indirizzo decodificato 176. La Tabella 5.2 specifica le uscite del decodificatore disponibili sullo zoccolo "INDIRIZZO" IC-20.

Anche gl'indirizzi di memoria sono decodificati facilmente dai circuiti d'interfaccia della scheda di prova. Altri due chips comparatori, l'IC-3 e l'IC-4, sono utilizzati per confrontare le linee A15-A8 del bus indirizzi con un indirizzo HI assegnato. I bits dell'indirizzo HI sono predisposti sul package di otto interruttori del tipo "dual-in-line":

Pin (IC-20)	Denominazione	Pin di uscita dell'SN74154
1,16	0	1
2,15	1	2
3,14	2	3
4,13	3	4
5,12	4	5
6,11	5	6
7,10	6	7
8,09	7	8

Tabella 5.2 — Collegamenti del decodificatore indirizzi con lo zoccolo "INDIRIZZO" IC-20.

questo package è indicato con HI ed è posizionato in IC-2. Utilizzando l'indirizzamento memoria, bisogna badare a non cercare e a non scegliere indirizzi che siano stati assegnati alla memoria interna dell'Apple (memoria ROM o RAM). E non bisogna dimenticare di convertire l'intero indirizzo a 16 bits nel valore decimale equivalente, al fine di utilizzarlo nelle istruzioni di PEEK e di POKE.

Nella modalità d'indirizzamento memoria, bisogna porre l'interruttore più basso in IC-6 nella posizione di "chiuso", ovvero "M": in tal modo il decodificatore SN74154 sarà attivato soltanto in caso di corrispondenza fra i bits d'indirizzo A15-A8 ed i bits predisposti sull'interruttore DIP HI *e* fra i bits d'indirizzo A7-A4 ed i bits predisposti sull'interruttore DIP LO. Quindi si potrà accedere agl'indirizzi compresi fra XXXXXXXX XXXX0000 e XXXXXXXX XXXX0111, dove X = 1 o 0. Questi indirizzi decodificati sono presenti come impulsi negativi sullo zoccolo "INDIRIZZO" (IC-20). Si tenga presente che di un blocco selezionato di 16 indirizzi, sono disponibili soltanto i primi otto indirizzi. Quindi, ponendo 10000001 per l'indirizzo HI e 1110 per l'indirizzo LO (bits A7-A4), gl'indirizzi da 33248 a 33256 genereranno degl'impulsi negativi, rispettivamente sui pins dall'1 all'8 dello zoccolo "INDIRIZZO". Non dimenticate che il decodificatore SN74154 decodifica tutti i 16 indirizzi, ma che si ha accesso soltanto agli otto "più bassi".

Sulla scheda di prova sono disponibili le connessioni alle linee A3-A0 del bus indirizzi (non bufferizzato), rispettivamente ai pins 8-5 dello zoccolo IC-19. Questi segnali sono utilizzabili in molti esperimenti, ma con cautela, perché non sono bufferizzati, e quindi presentano un collegamento diretto con il calcolatore Apple.

La sezione di decodifica indirizzo della scheda di prova permetterà di risparmiare molto tempo e lavoro, perché non occorrerà costruire i circuiti di decodifica dell'indirizzo dispositivo quando si vorrà realizzare delle porte di I/U o verificare dei semplici circuiti d'interfaccia.

I buffers bus

Due chips 8216 che rappresentano dei buffers bus non invertenti sono stati usati per bufferizzare il bus: in Figura 5.5 sono designati come IC-10 ed IC-11.

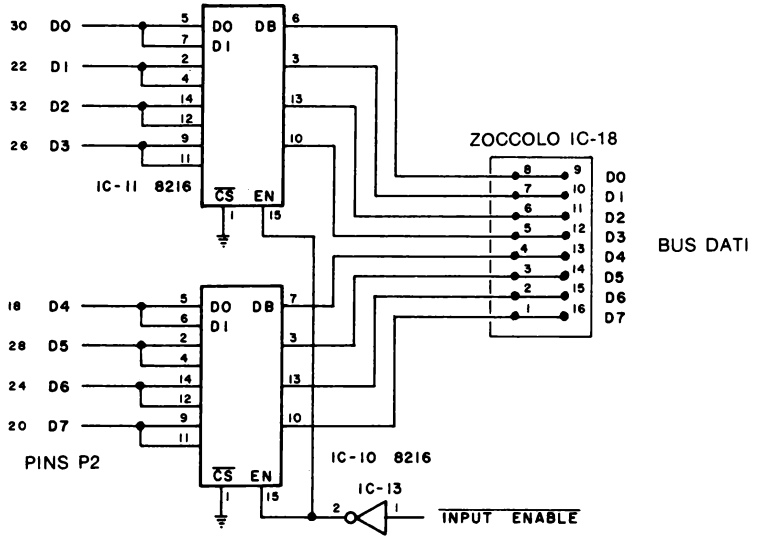


Figura 5.5 — Schema del circuito di buffer bus.

Questo vuol dire che il bus è disponibile per un fan-out totale di 30 (cioè può pilotare 30 ingressi standard tipo 7400), e che il bus stesso è separato dal bus dati dell'Apple. Gli otto bits del bus dati sono disponibili sullo zoccolo IC-18.

La Tabella 5.3 mostra i collegamenti con il bus dati.

Pin (IC-18)	Segnale del bus dati
1,16	D7
2,15	D6
3,14	D5
4,13	D4
5,12	D3
6,11	D2
7,10	D1
8,9	D0

Tabella 5.3 — Collegamenti del bus dati sull'IC-18.

I buffers bus sono sempre abilitati; la modalità operativa abituale è il trasferimento di dati dall'Apple alla scheda di prova. Ciò vuol dire che, senza bisogno di altri segnali, si può controllare l'"attività" del bus collegando delle sonde logiche, o altri dispositivi analoghi, alle uscite D7-D0 dei chips di buffer bus. Le porte di uscita si realizzano semplicemente ricorrendo agli opportuni segnali di controllo (dei quali si

parlerà nel prossimo paragrafo) per controllare un latch di 8 bits. Gli otto ingressi del latch sono collegati ai segnali D7-D0 dello zoccolo IC-18.

Le porte d'ingresso, invece, devono essere realizzate in modo da "orientare" i buffers bus nella direzione opposta, al fine di "pilotare" i dati nell'Apple. In realtà ci sono due buffers bus per ciascuna linea di bus, come lo schema logico di Figura 5.6 mostra per il bufffer 8216.

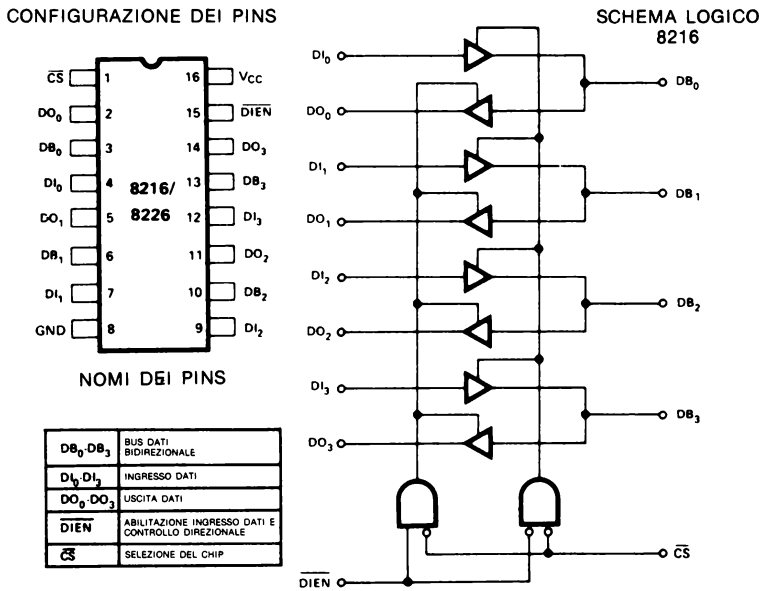


Figura 5.6 — Configurazione dei pins del chip di buffer bus 8216.

L'ingresso DIEN determina quale serie di buffers è abilitata, e quindi dirige i dati dentro o fuori l'Apple.

Tutte le operazioni d'ingresso devono attivare la giusta serie di buffers, in modo che l'Apple riceva i dati nel modo corretto. A tal fine è stata predisposta una speciale circuiteria di controllo relativamente alle operazioni d'ingresso dati.

La circuiteria di controllo

La circuiteria di controllo posta sulla scheda di prova è abbastanza semplice: è costituita principalmente da alcuni buffers d'uso generale, che provvedono a bufferizzare i segnali di controllo emessi dal calcolatore. Si hanno sei segnali: $\overline{IN}$, $\overline{RD}$, $\overline{OUT}$, $\overline{WR}$, $\overline{RESET}$ ed $\overline{INTAK}$. Nell'interfacciamento con l'Apple intervengono soltanto i segnali $\overline{WR}$, $\overline{RD}$ e $\overline{RESET}$, mentre gli altri si utilizzano quando si usa la scheda di prova con altri calcolatori. La circuiteria di controllo è illustrata in Figura 5.7.

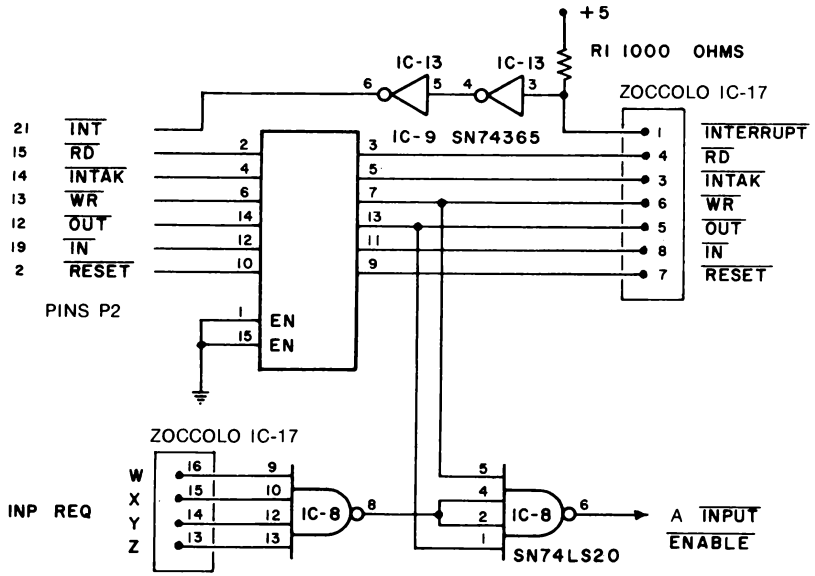


Figura 5.7 — Schema del circuito di controllo.

Anche il segnale d'uso generale destinato alla generazione d'interruzioni è bufferizzato: esso è un ingresso del calcolatore. I collegamenti con i segnali di controllo sono realizzati sullo zoccolo IC-17 (vedi Tabella 5.4).

Pin (IC-17)	Segnale di controllo	Direzione
1	$\overline{\text{INT}}$	Ingresso
2	Non utilizzato	—
3	$\overline{\text{INTAK}}$	Uscita
4	$\overline{\text{RD}}$	Uscita
5	$\overline{\text{OUT}}$	Uscita
6	$\overline{\text{WR}}$	Uscita
7	$\overline{\text{RESET}}$	Uscita
8	$\overline{\text{IN}}$	Uscita

Tabella 5.4 — Collegamenti dei segnali di controllo sull'IC-17.

La circuiteria di controllo genera poi anche un segnale che commuta i buffers bus 8216 nella modalità ingresso (*input mode*), in modo che i dati possano essere trasferiti nell'Apple. Si potrebbe pensare che si tratti semplicemente di "cambiare la direzione" del bus ogni volta che si ha un'operazione di lettura memoria: se così fosse, i buffers bus presenti sulla scheda di prova sarebbero posti nella modalità in-

gresso, anche quando venisse attivato un chip di memoria dell'Apple. Ciò provocherebbe un "conflitto" di bus, per cui il bus che si trova sulla scheda di prova dev'essere posto nella modalità ingresso solo quando è selezionato un dispositivo d'ingresso sulla scheda di prova in questione.

Per un opportuno trattamento delle porte d'ingresso, si userà il segnale di selezione dispositivo relativo alle porte d'ingresso per convogliare i dati sul bus dati e per controllare la modalità dei buffers bus 8216. Di fatto, si possono mettere insieme in OR fino a quattro impulsi di selezione dispositivo (relativi alle porte d'ingresso) con cui posizionare nella modalità ingresso i buffers bus della scheda di prova. È molto probabile che non si dovranno usare più di quattro porte d'ingresso sulla scheda di prova. Allora questi segnali cambiano la direzione del bus per l'ingresso dei dati solo quando sulla scheda di prova viene generato un segnale di selezione dispositivo riferito alla porta d'ingresso: questo è cablato dall'utilizzatore ad uno dei quattro ingressi di abilitazione del buffer bus.

È necessario che gl'impulsi di controllo "INPUT REQUEST" siano a logica negativa: questi sono applicati ai pins indicati con W, X, Y e Z, cioè ai pins dal 16 al 13 dello zoccolo IC-17.

L'OR effettivo di questi segnali di controllo viene fatto dalla porta SN74LS20 in IC-8. Il segnale "INPUT REQUEST" che si ha in uscita da questa porta NAND a 4 ingressi è ulteriormente condizionato dai segnali $\overline{OUT}$ e $\overline{WR}$. Quest'ultima porta logica dà luogo ad un circuito di sicurezza, in modo che, se i vostri circuiti sulla scheda di prova sono stati cablati in modo inesatto, i drivers di bus non possono venir posti nella modalità ingresso quando è in corso un'operazione del tipo uscita. Si ha allora il segnale risultante "INPUT REQUEST, BUT NOT $\overline{OUT}$ OR $\overline{WR}$", che controlla le modalità ingresso/uscita dei buffers bus 8216.

L'Apple genera soltanto il segnale di scrittura memoria, $\overline{WR}$, e questo vuol dire semplicemente che la vostra interfaccia non sarà in grado di cambiare la direzione del bus per un'operazione d'ingresso quando il calcolatore sta eseguendo un'operazione di scrittura. Il segnale $\overline{OUT}$ si usa per l'interfacciamento con i calcolatori basati sull'8080, l'8085 e lo Z-80.

In Figura 5.8 sono rappresentate due porte d'ingresso, ciascuna delle quali è controllata da un impulso di selezione dispositivo che abilita i buffers a tre stati. Questo stesso segnale ha anche la funzione di segnale di richiesta ingresso, INP REQ, e ciascuna porta d'ingresso deve generare il suo segnale di richiesta ingresso. In questo esempio i due segnali di richiesta ingresso sono stati collegati ai pins W e Z dello zoccolo IC-17 posto nel settore INP REQ. Sarebbe stato altrettanto semplice e corretto collegare le linee ai pins X ed Y.

Il segnale di blocco INPUT REQUEST, e la relativa circuiteria, si usano solo per i circuiti d'interfaccia sotto test che si trovano sulla scheda di prova. Invece, per costruire un'interfaccia che si colleghi direttamente all'Apple, e che non ricorra alla bufferizzazione del bus bidirezionale, un segnale di blocco di questo tipo non è necessario. Il ruolo fondamentale di questa circuiteria è quello di proteggere il calcolatore Apple dai danni che potrebbero essere provocati da un cablaggio poco accura-

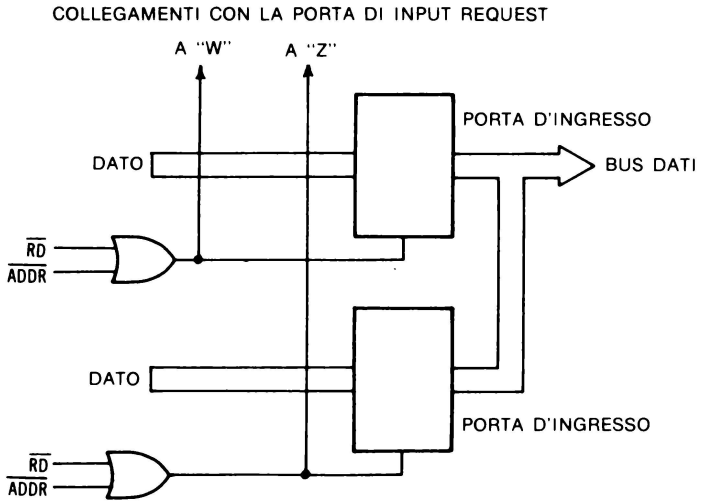


Figura 5.8 — Tipiche porte d'ingresso che illustrano l'uso del segnale INPUT REQUEST.

to, o sbagliato, di un circuito sotto test. Una volta che un circuito è stato compiutamente provato e messo a punto, solo allora lo si può collegare senza problemi con il bus dati dell'Apple.

Costruzione della scheda di prova

I circuiti della scheda di prova si possono costruire con la tecnica del wire-wrap (vedi Figura 5.9): in questo caso è possibile espanderli e modificarli attraverso delle

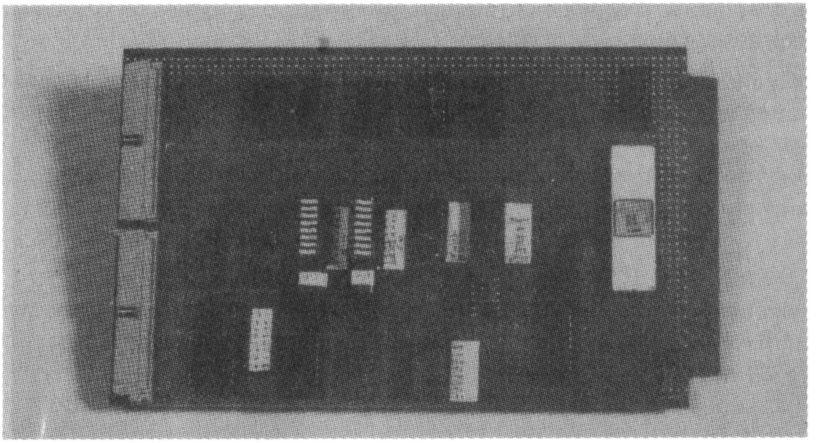


Figura 5.9 — Versione a wire-wrap del circuito d'interfaccia.

semplici modifiche di cablaggio, ma con lo svantaggio di una difficoltà nell'uso della scheda di prova.

Per rendere più agevole la costruzione e la verifica dell'interfaccia, abbiamo sviluppato un circuito stampato nel quale tutta la circuiteria necessaria trova posto su un'unica scheda. La sezione di alimentazione e la circuiteria relativa alla sonda logica sono state incorporate, per semplificare l'uso della scheda di prova: questa, che potete vedere in Figura 5.10, è distribuita, in kit oppure in versione assemblata, dalla Group Technology, P.O. Box 87B, Check, VA 24072.

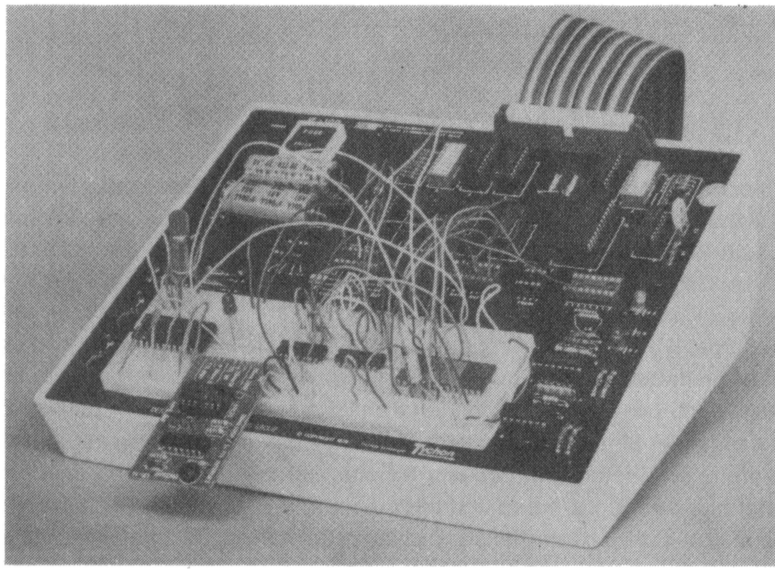


Figura 5.10 — Versione a circuito stampato dell'interfaccia. (Per gentile concessione della E & L Instruments, Inc.)

Una buona porzione della scheda di prova è stata lasciata libera; questo permette di montare direttamente sulla scheda a circuito stampato una piastra forata, per poter svolgere dei facili esperimenti. Tipiche piastre forate sono l'"SK-10", prodotta dalla E & L Instruments, Derby, CT 06418, e la "Super Strip", prodotta dalla AP Products, Inc., Mentor, 0H 44060. In appendice troverete l'elenco completo dei pezzi necessari per costruire la scheda di prova, ed il disegno della scheda a circuito stampato.

COLLEGAMENTO CON L'APPLE

La scheda di prova utilizza un cavo a 40 conduttori per collegarsi con i vari tipi di calcolatori; pertanto occorrerà collegare il cavo ad uno dei connettori (*slots*) del-

l'Apple riservati all'interfacciamento delle periferiche. A tal fine consigliamo un assemblaggio tramite cavo piatto, come si vede in Figura 5.11.

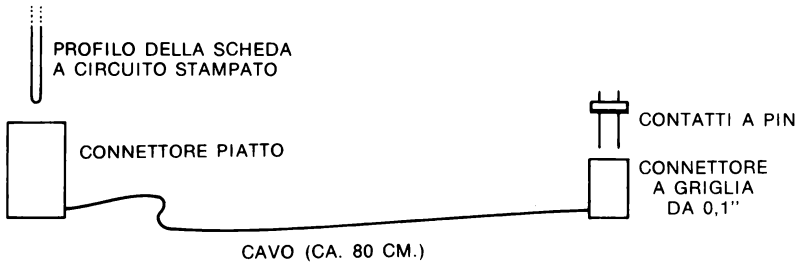


Figura 5.11 — Cavo per l'interfacciamento.

Ad un'estremità del cavo si trova assemblato un connettore piatto femmina per circuiti stampati, all'altra estremità un connettore a griglia per pins, femmina, da 0.1 × 0.1 pollici. Le aperture sui due connettori devono essere rivolte nella stessa direzione. La Group Technology fornisce un cavo già fatto, di sigla BG-100-Cable, consistente in un cavo piatto lungo 80 centimetri.

I collegamenti effettivi con i segnali di bus dell'Apple sono realizzati con una piccola scheda-adattatore. Questa scheda "devia" e "dispone" i vari segnali in modo che questi siano convogliati dal connettore piatto al connettore per periferiche che si trova nell'Apple. Si può facilmente mettere insieme un adattatore per mezzo della scheda prototipo Vector 4609: questa scheda, che s'innesta in uno dei connettori per periferiche dell'Apple, ha un connettore piatto a 40 conduttori che si collega direttamente con il cavo d'interfaccia. È chiaro che, volendo, si può collegare direttamente il cavo saldandolo, ma noi lo sconsigliamo. È possibile comunque realizzare dei collegamenti saldati diretti fra i conduttori dei segnali corrispondenti su ciascun connettore piatto, con dei pezzetti di filo di collegamento. Se si vogliono fare dei collegamenti saldati, si possono saldare i pins di wire-wrap nei fori che si trovano su tutti i connettori piatti, e realizzare il collegamento con il filo di wire-wrap.

I collegamenti sono rappresentati in Figura 5.12.

Se si vuole usare la scheda prototipo Vector, ci sono diverse cose importanti da fare prima di cominciare a fare i collegamenti fra i due connettori piatti, qualunque sia la tecnica che s'intende adottare. Fra il connettore a 40 conduttori ed i pins di massa e di +5 volts situati sul connettore a 50 conduttori dell'Apple potrebbero probabilmente esserci dei "percorsi metallici" di circuito stampato, ovvero percorsi conduttori. Tutti questi collegamenti devono essere interrotti, in modo che i contatti del connettore a 40 conduttori siano "liberi" e non impegnati da nessun segnale. Questi collegamenti si possono troncare con un taglierino, praticando due tagli per ogni conduttore, alla distanza di due o tre millimetri; poi, per "staccare" il pezzetto tagliato, lo si scalda con il saldatore. Quest'operazione va fatta solo per i collega-

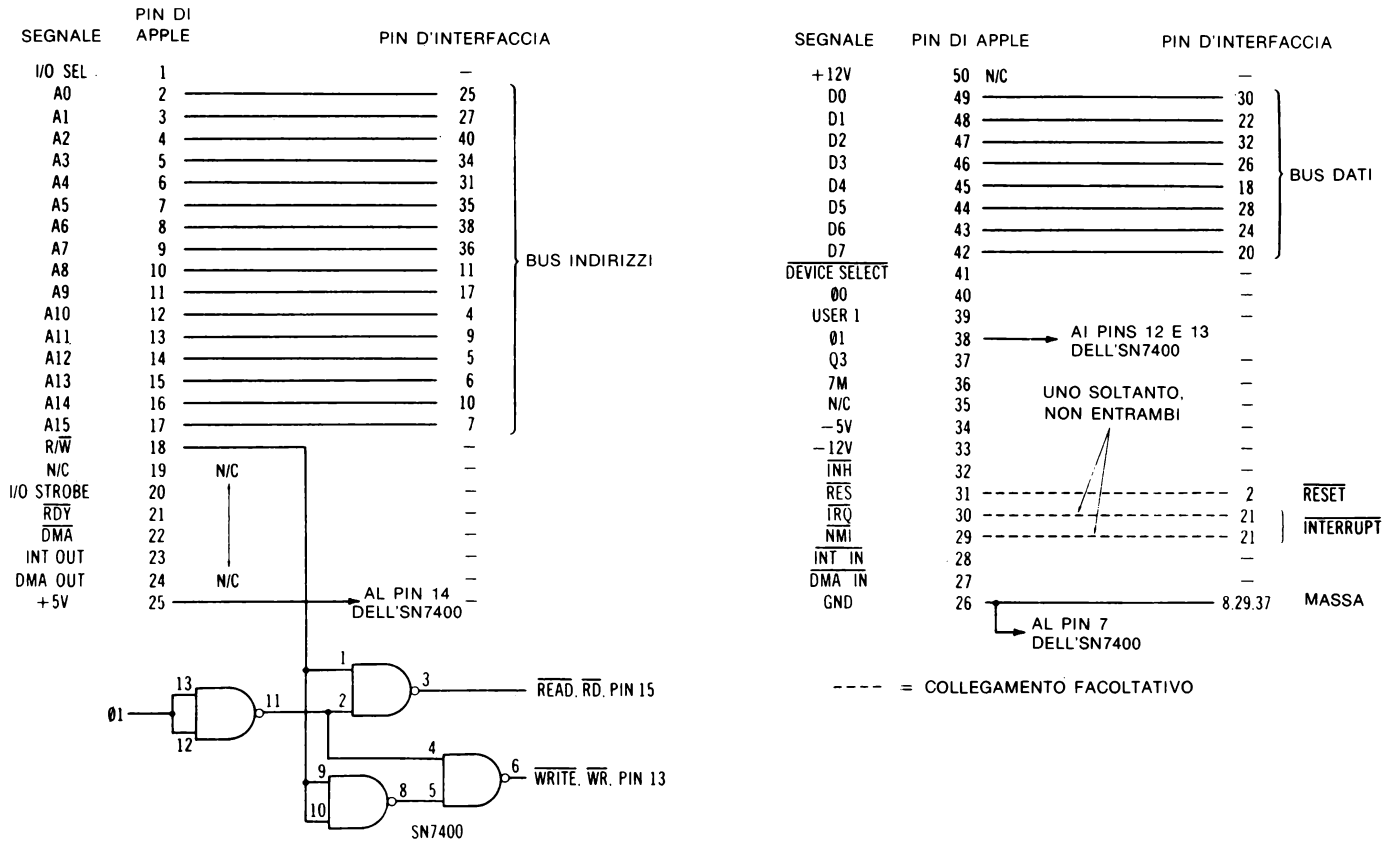


Figura 5.12 — Collegamenti fra il connettore dell'Apple ed il connettore d'interfaccia.

menti di alimentazione esistenti fra i due connettori; tutti gli altri pins sono "liberi".

La scheda prototipo Vector non ha fori metallizzati, per cui accertatevi di collegare la tensione di +5 volts e la massa ai rispettivi bus di alimentazione, e che siano realizzati i giusti collegamenti con il chip SN7400.

Il chip SN7400 condiziona il segnale di lettura/scrittura (R/$\overline{W}$) con il segnale di clock del processore 6502, cioè con $\Phi 1$. Questo condizionamento genera il segnale di lettura memoria, $\overline{RD}$, ed il segnale di scrittura memoria, $\overline{WR}$; se non viene realizzato, le periferiche del calcolatore poste sulla scheda di prova non lavorano in modo corretto. In alcuni calcolatori esistono segnali distinti per la lettura e la scrittura. Se nell'Apple ed in altri calcolatori basati sul microprocessore 6502 si vogliono usare segnali distinti di lettura e scrittura per il controllo della memoria, bisogna generarli mediante un appropriato sistema di porte logiche.

Le locazioni dei pins per i connettori piatti della scheda Vettore sono illustrati in Figura 5.13.

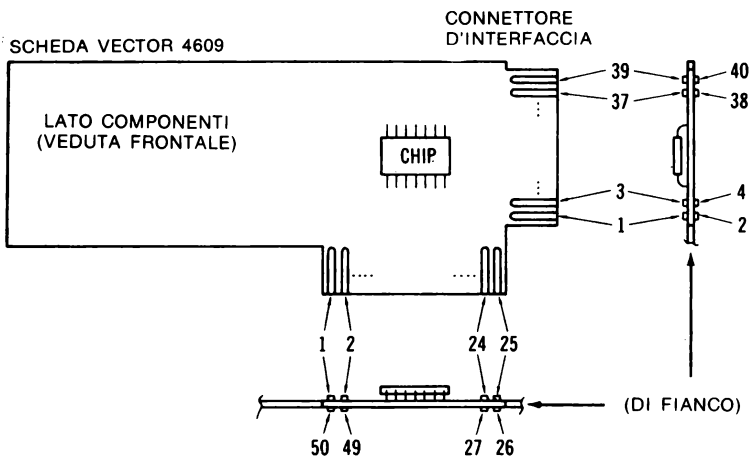


Figura 5.13 — Contatti e disposizione dei contatti d'interfaccia della scheda Vector 4609.

Noterete che la figura mostra *il lato componenti* della scheda.

Una volta realizzati gli opportuni collegamenti fra i due connettori piatti, e fra i connettori e l'SN7400, vi consigliamo di verificare, con un ohmetro o un altro strumento di controllo della continuità, che non ci siano corti circuiti fra pins adiacenti ed opposti, e che si siano effettuati gli esatti collegamenti. Queste verifiche vanno fatte con il chip SN7400 *non inserito nel suo zoccolo*: comunque, una volta verificati i collegamenti, non dimenticate di reinserirlo!

QUALCHE ALTRA OSSERVAZIONE

Se provate ed interfacciate un chip d'interfaccia della famiglia 6502, o anche uno

non appartenente a questa famiglia, constaterete che questi chips hanno tempi di accesso abbastanza lenti, al confronto dei chips d'ingresso a tre stati standard, come l'SN74365 e l'SN74LS244. Per questi grandi chips programmabili il tempo di accesso può essere di 200 ns.

Dato che il tempo di lettura/scrittura è decisamente critico per il chip 6502, non ci sarà tempo sufficiente per accedere ai dati provenienti dal chip stesso e collocarli sul bus dati, se si tiene conto dell'ulteriore ritardo causato dai chips di buffer bus 8216 e dai circuiti di bloccaggio. Di conseguenza, se si vuol usare la scheda di prova per verificare circuiti d'interfaccia che impieghino complessi chips d'interfaccia programmabili, si dovrà "eliminare" il circuito di bloccaggio. Questo si può fare abbastanza semplicemente togliendo i due chips di buffer bus 8216 e piazzando dei corti ponticelli in ciascuno zoccolo per collegare i segnali del bus dati dell'Apple con le linee del bus dati dell'interfaccia: ad esempio, vi occorrerà in ciascuno zoccolo un ponticello fra i pins 5 e 6, 2 e 3, 14 e 13, 9 e 10. Vi ricordiamo che il circuito che utilizza i chips di buffer bus 8216 è in Figura 5.5.

Ma attenzione: togliendo i chips di buffer bus, si collegano direttamente i vostri circuiti d'interfaccia con il bus dati dell'Apple. Quindi, nel fare quest'operazione, state bene attenti, in modo da non provocare corti circuiti o conflitti di bus nell'Apple.

Nel Capitolo 7 troverete un semplice esempio d'interfacciamento nel quale si ha l'interfacciamento diretto con il bus.

CAPITOLO 6

ESPERIMENTI D'INTERFACCIAMENTO CON L'APPLE

Scopo degli esperimenti contenuti in questo capitolo è quello di fornirvi una certa esperienza pratica nell'utilizzo dei circuiti costituiti da porte di uscita fornite di latch e da porte d'ingresso a tre stati, di cui si è parlato nei capitoli precedenti. Per il trasferimento dei dati al e dall'Apple abbiamo usato in questi esperimenti semplici dispositivi della serie SN7400.

OSSERVAZIONI INTRODUTTIVE AGLI ESPERIMENTI

In questo capitolo verificheremo i circuiti sulla scheda di prova: nell'Appendice B troverete l'elenco completo dei pezzi che si useranno.

Abbiamo presupposto da parte vostra una certa pratica nel verificare dei semplici circuiti logici, e la conoscenza degli accorgimenti di base nell'uso di una scheda di prova. Comunque questi esperimenti richiedono in più qualche altra funzione, per controllare e generare degli stati logici. In genere usiamo dei segnalatori luminosi, o LED, per indicare l'uno logico (on) e lo zero logico (off), interruttori logici per generare i livelli logici, e pulsanti antirimbalzo, ovvero semplicemente pulsanti, per generare livelli logici con transizioni pulite senza disturbi fra i livelli logici. In appendice troverete alcuni schemi di circuiti di questo tipo. Se non intendete costruirli, questi circuiti possono venir montati su una scheda di prova separatamente, oppure potete acquistarne di funzionalmente simili presso ditte come la E & L Instruments, Derby, CT 06418 e la PACCOM, Redmond, Wa 98052.

La maggior parte degli esperimenti che presentiamo sono realizzabili con pochi semplici circuiti.

Uno degli esperimenti proposti illustra l'uso di un circuito di decodifica per indirizzare un dispositivo. Per quanto un decodificatore possa essere fatto secondo molti schemi, riteniamo che un esperimento debba evidenziare i principi di base; comunque, se siete interessati a conoscere altri circuiti di decodifica, potrete trovarne di-

versi descritti in *8085A Cookbook*, ed in *Programming & Interfacing the 6502, With Experiments*, entrambi editi da Howard W. Sams & Co., Inc., Indianapolis, IN 46268. Di fatto l'indirizzamento della memoria e dei dispositivi di I/U è press'a poco lo stesso da un calcolatore all'altro. Nella maggior parte dei circuiti d'interfaccia, il circuito di decodifica presente sulla scheda di prova andrà abbastanza bene.

Sebbene questo libro tratti dell'interfacciamento dell'Apple ad un livello abbastanza basso, ci sono, sempre in tema d'interfacciamento, molti altri punti importanti che potrebbe interessarvi di studiare. Molti potete trovarli in *TRS-80® Interfacing, Book 2*, Howard W. Sams & Co., Inc., Indianapolis, IN 46268. Le informazioni presentate qui sono abbastanza generali, e si applicano senza problemi ai sistemi Apple. Gli argomenti trattati sono: circuiti di pilotaggio di carichi ad alta corrente ed alta tensione, convertitori digitale-analogici ed analogico-digitali, trattamento dati quali livellamento, filtraggio, calcolo della media, etc., comunicazioni seriali e controllo remoto.

La foto di Figura 6.1 mostra una tipica stazione di laboratorio con Apple e scheda di prova, uguale a quella che è servita per gli esperimenti del capitolo.

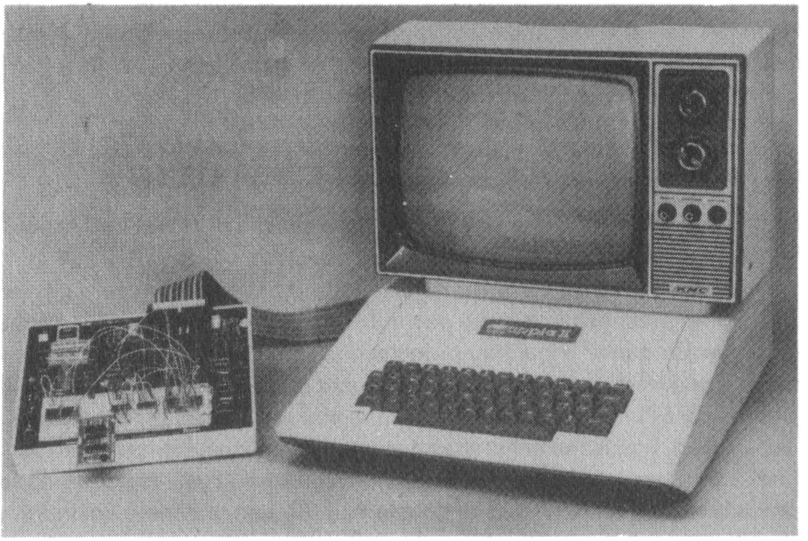


Figura 6.1 — Apple e scheda di prova usata negli esperimenti.

Per collegare la scheda di prova con l'Apple abbiamo usato un cavo a 40 conduttori (v. Figura 6.2).

Il cavo è stato descritto nel Capitolo 5.

Nel collegare la scheda di prova con l'Apple, accertatevi che il cavo sia orientato correttamente: *il cavo cioè dev'essere orientato dalla parte opposta al lato compo-*

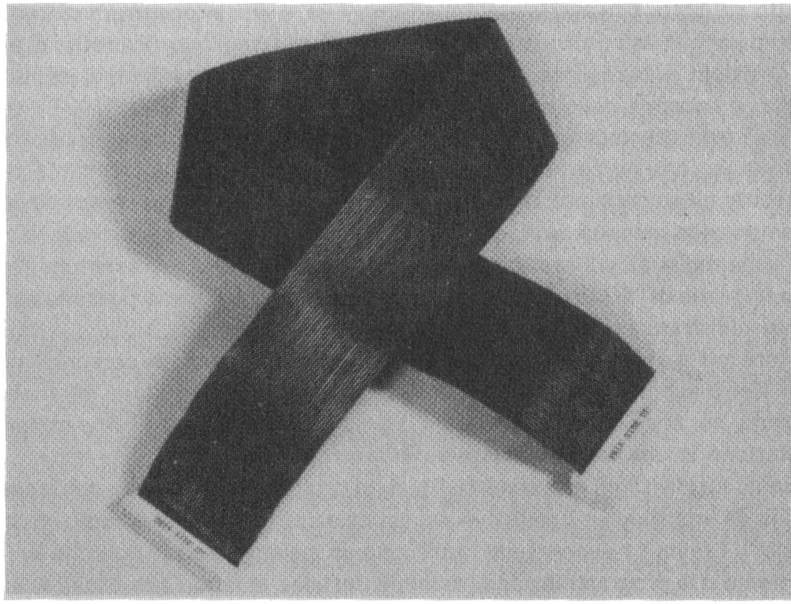


Figura 6.2 — Cavo d'interfaccia. (Si osservi che i connettori sono orientati sullo stesso lato del cavo piatto.)

nenti della scheda usata per collegare l'interfaccia all'Apple. All'estremità verso la scheda di prova, il cavo dev'essere inserito nei 40 pins, in modo da essere orientato in basso, cioè dalla parte opposta alla scheda a circuito stampato. Se il cavo è collegato in modo sbagliato, l'Apple, alla prima accensione, mostrerà lo schermo pieno di caratteri a caso, invece del messaggio "APPLE II". Questo non provoca danni permanenti all'Apple o all'interfaccia, a patto però che non siano collegati in questo modo per troppo tempo.

In alcuni esperimenti si costruiranno circuiti, o si utilizzeranno programmi, sviluppati in esperimenti precedenti.

Vi raccomandiamo di non togliere l'alimentazione al calcolatore e di non scollegare i circuiti finché non vi diciamo di farlo, sennò dovrete perdere una grande quantità di tempo per ricaricare i programmi e ricostruire i circuiti d'interfaccia. Per non dimenticarvene, troverete un promemoria al termine di ogni esperimento.

Riteniamo che la maggior parte dei lettori eseguirà gli esperimenti nell'ordine in cui sono presentati, in modo da potersi rifare senza problemi agli esperimenti precedenti per quanto riguarda i particolari dei circuiti d'interfaccia; se qualcuno invece non rispetterà la sequenza, e salterà qualche esperimento, si troverà un po' disorientato. In tutti i casi, a titolo di chiarimento per quanto riguarda i circuiti d'interfaccia, *in chiusura di capitolo*, in Figura 6.27, abbiamo riprodotto le porte d'ingresso, le porte di uscita ed i circuiti di controllo più importanti. Potete fotocopiare la figura, o

staccarla dal libro, per averla sotto mano ogni volta che vi occorra. I circuiti base che compaiono in essa sono utilizzati in quasi tutti gli esperimenti, a meno di esplicite segnalazioni, e si possono usare per costruire porte d'ingresso e di uscita d'uso generale, a seconda delle proprie necessità.

Gl'insegnanti che vogliano usare questo libro come testo di base per esperimenti di laboratorio con l'Apple troveranno che i programmi sono comodamente caricati su cassette; gli studenti potranno così disporne immediatamente, senza dover perdere tempo nella messa a punto. Se usate cassette, scegliete nastri di buona qualità e, una volta registrati sul nastro i programmi, staccate dal bordo esterno della cassetta la linguetta di "Write protect": questo eviterà che gli studenti possano accidentalmente registrare dei programmi su altri già registrati.

Ciò sarà molto utile agli studenti per conservare le loro cassette personali: potranno così avere sempre sotto mano le loro realizzazioni di laboratorio ed anche altri programmi, per scambi con altri studenti o gruppi di laboratorio, o come materiale di consultazione in una fase successiva del lavoro.

Abbiamo suddiviso gli esperimenti di questo capitolo in due gruppi, non evidenziati tuttavia da una suddivisione in due del capitolo stesso né da sottotitoli, né da altre indicazioni. I primi 11 esperimenti forniscono un insieme base di ricerche sull'interfacciamento e la programmazione, essendo destinati ai lettori interessati ai principi di base dell'interfacciamento: danno infatti le conoscenze di base per la sezione laboratorio del primo corso d'interfacciamento dei calcolatori ed elettronica dei calcolatori.

Gli ultimi esperimenti, pochi, rappresentano ulteriori ricerche di laboratorio relative ad argomenti più avanzati, e forniscono inoltre degli schemi che possono servire ad integrare il gruppo di esperimenti di base. Inutile dire che tutti gli esperimenti sono eseguibili praticamente.

Esperimento n. 1
USO DELLA SONDA LOGICA

Scopo

Scopo di quest'esperimento è mostrare come si usi il circuito di sonda logica presente sulla scheda di prova per rilevare livelli logici ed impulsi.

Presentazione dell'esperimento

Abbiamo supposto che usiate la sonda logica della scheda di prova, anche se altri circuiti di sonda logica possono rispondere allo scopo altrettanto bene. La suddivisione in passi vi aiuterà a prendere familiarità con la scheda di prova e con i segnali disponibili.

Passo n. 1

Il calcolatore Apple va collegato con il suo schermo video, e con la scheda di pro-

va per mezzo del cavo a 40 conduttori, collegamento che è stato descritto nel paragrafo d'introduzione agli esperimenti.

Alimentate l'Apple e la scheda di prova. Il calcolatore visualizzerà il messaggio "APPLE II", e quindi il cursore, costituito da un quadratino lampeggiante. Se ciò non avviene, disinserite l'alimentazione e verificate i collegamenti: controllate che il cavo a 40 conduttori sia inserito saldamente nei pins della scheda di prova e sul bordo della scheda di connessione dell'Apple. Verificate poi che l'orientazione del cavo sia esatta. Se non riuscite a localizzare da soli il problema, fatevi aiutare.

Passo n. 2

Con la scheda di prova alimentata, collegate con un ponticello uno dei pins d'ingresso P della sonda logica, posti sullo zoccolo "SONDA", ad uno dei pins di alimentazione a +5 volts posti sullo zoccolo "ALIMENTAZIONE". Quale effetto avrà quest'operazione sugl'indicatori della sonda logica?

Il LED rosso si accende, ad indicare la presenza dello stato logico uno.

Spostate ora il ponticello della sonda dal pin di alimentazione a +5 volts ad uno dei pins di massa posti sul medesimo zoccolo. Che cosa avviene quando si realizza questo collegamento?

Il LED verde si accende, ad indicare la presenza di uno stato logico zero all'ingresso del circuito di sonda.

Avete forse osservato che il LED di rilevazione degl'impulsi (il LED giallo) lampeggia non appena si realizza il collegamento a +5 volts o a massa. Questo sta ad indicare che la sonda ha rilevato un *cambiamento* di livello logico: tanto il passaggio dal livello logico uno al livello logico zero quanto il passaggio dal livello logico zero al livello logico uno causa un lampeggiamento del LED giallo. Per questo motivo il LED giallo è particolarmente utile per rilevare gl'impulsi e le transizioni logiche.

Collegate l'ingresso della sonda alla linea indirizzo A0 dell'IC-19. Che cosa notate in conseguenza di tale collegamento?

Tutti i LED si accendono, generalmente non tutti con la stessa intensità, per il fatto che il microprocessore 6502 sta eseguendo in questo momento moltissime istruzioni, in linguaggio assemblatore, poste nelle ROM che ospitano il BASIC ed il Monitor, e quindi utilizza il bus indirizzi per indirizzare le diverse locazioni di memoria. Spostate il filo di prova della sonda logica sulle altre linee del bus indirizzi, cioè sulle linee A1, A2 e A3. Anche per questi pins dovreste essere in grado di rilevare un analogo "funzionamento".

Passo n. 3

Supponiamo che desideriate esaminare degli altri punti della scheda di prova per mezzo della sonda logica. Esaminare le linee del bus dati ed i segnali di controllo sarà una cosa semplice. Ricordate che la sonda logica è sensibile solo ai livelli logici

presentati dalle uscite dei chips standard in logica TTL usati sulla scheda di prova e negli esperimenti. *Non* cercate di usare la sonda per misurare nient'altro oltre a questi livelli logici. Se collegate la sonda a tensioni che superino l'intervallo da 0 a + 5 volts, il circuito di sonda ne sarà danneggiato.

Passo n. 4

Usando la sonda, noterete che si possono avere svariate combinazioni di LED accesi e spenti: ad esempio, i LED rosso e giallo saranno accesi, mentre il LED verde sarà spento. Avete un'idea di che cosa significhi questo?

Significa che è stato rilevato un impulso, *e* che il livello logico *normale* del circuito sotto test è un uno logico. Il LED verde resta acceso per tempi molto brevi (tanto da non essere percepibile dall'occhio umano), ad indicare la presenza transitoria dell'impulso di zero logico. Il circuito per la rilevazione degl'impulsi allunga la durata dell'impulso e fa accendere il LED giallo, di modo che si può "vedere" che è stato "preso" un impulso.

Può anche capitare di vedere accesi i LED verde e giallo, e spento il LED rosso. Che cosa indica questo?

Indica la presenza di un livello logico zero, con brevi impulsi di uno logico.

È poi anche possibile che tutti i LED siano accesi. In questo caso l'ingresso alla sonda logica cambia continuamente e rapidamente fra lo stato logico uno e lo stato logico zero.

In alcuni dei prossimi esperimenti useremo la sonda logica per esaminare le uscite e rilevare stati logici ed impulsi. Ciò sarà indicato con le frasi "... usate la sonda per esaminare ...", oppure "... usate la sonda logica per misurare ...", che significheranno semplicemente che dovrete collegare la sonda logica al circuito sotto test, allo scopo di "vedere" quello che via via accade.

Ora spegnete il calcolatore.

Esperimento n. 2
USO DEL DECODIFICATORE D'INDIRIZZO DISPOSITIVO

Scopo

Quest'esperimento vi permette di conoscere l'utilizzo del circuito che decodifica l'indirizzo dispositivo, circuito presente sulla parte a circuito stampato della scheda di prova. E, dal momento che questo decodificatore sarà usato in tutti gli esperimenti, è indispensabile che ne comprendiate bene l'uso.

Presentazione dell'esperimento

In quest'esperimento si usano i bits indirizzo A15-A0 per identificare indirizzi specifici utilizzati da parte dei dispositivi di I/U. Gl'interruttori d'indirizzo saranno predi-

sposti per una determinata serie d'indirizzi, e la sonda logica servirà ad esaminare il funzionamento del circuito di decodifica. Userete poi anche il circuito integrato SN7402 di porte NOR.

Configurazione dei pins del circuito integrato (v. Figura 6.3)

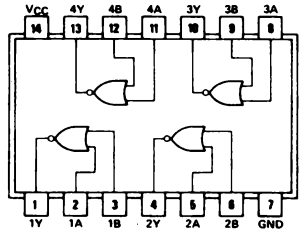


Figura 6.3 — Configurazione dei pins della porta NOR SN7402.

Passo n. 1

Per il momento nessun circuito dev'essere cablato sulla scheda di prova. Se vi si trovano dei circuiti, toglieteli dalla parte forata della scheda di prova. In quest'esperimento l'intero bus indirizzi a 16 bits sarà utilizzato dalla sezione di decodifica dell'interfaccia. Accertatevi che l'interruttore inferiore posto nel package d'interruttori DIP in IC-6 e relativo all'indirizzo LO si trovi nella posizione "M", ovvero "ON".

Passo n. 2

Ponete gl'interruttori DIP relativi a tutti i bits indirizzo da A15 ad A4 nella posizione di uno logico, e ricordate di non cambiare il posizionamento dell'interruttore "M". Siete in grado di determinare qual è la serie d'indirizzi che verrà decodificata dal decodificatore SN74154? E quali indirizzi di questo gruppo saranno disponibili sullo zoccolo di uscita "INDIRIZZO"? Potrà esservi utile riguardare lo schema della Figura 5.4.

Gl'indirizzi facenti parte del gruppo che va da 65520 a 65535 saranno decodificati dal decodificatore da 4 a 16 linee (l'SN74154). Dal momento che il decodificatore fornisce soltanto gli otto indirizzi "inferiori", saranno disponibili solo gl'indirizzi che vanno da 65520 a 65527.

Passo n. 3

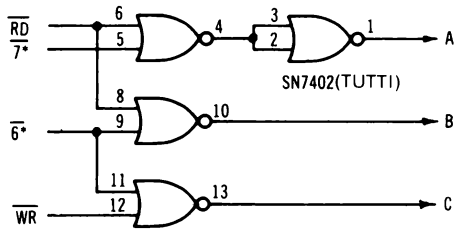
Accendete il calcolatore. Se state eseguendo un programma, premete il tasto RESET. Con la sonda logica esaminate le otto uscite indirizzo presenti sullo zoccolo "INDIRIZZO". Qualcuna delle uscite del decodificatore è attiva, cioè, in altre parole, emette impulsi? Se non state eseguendo un programma, è questo che vi aspettate?

Due uscite devono essere attive, la 0 e la 4, corrispondenti agl'indirizzi 65520 e

65524. Anche se il calcolatore non esegue un programma in BASIC, esegue tuttavia un gran numero d'istruzioni in linguaggio assemblatore relative all'esame della tastiera, etc. Ricordate che la circuiteria di decodifica indirizzo decodifica *sempre* degl'indirizzi.

Passo n. 4

Cablate il circuito illustrato in Figura 6.4



* POSIZIONI NELLO ZOCCOLO "DECODIFICATORE"

Figura 6.4 — Circuito di generazione d'impulsi-funzione.

State bene attenti a collegare il pin di alimentazione, cioè il 14, con il pin a +5 volts ed il pin di massa, cioè il 7, a massa. La configurazione dei pins dell'integrato SN7402 è in Figura 6.3. Questo chip può essere sostituito con un SN74LS02. Le uscite delle porte A, B e C in questa fase non sono collegate con nessun circuito.

Passo n. 5

Cambiate la posizione degl'interruttori relativi ai bits A15-A4 al fine di programmare l'indirizzo 49312. Quest'indirizzo è 11000000 10100000_2, del quale potete ignorare i quattro bits meno significativi. Quale gruppo d'indirizzi sarà disponibile quando gl'interruttori d'indirizzo sono posizionati in questo modo?

Saranno decodificati gl'indirizzi che vanno da 49312 a 49327, ma saranno disponibili unicamente quelli che vanno da 49312 a 49319.

Passo n. 6

Introducete nel calcolatore ed eseguite il programma seguente:

```
10  A = PEEK(49318)
20  GOTO 10
```

Controllate tramite la sonda logica le uscite del decodificatore, ed annotate qui di seguito le vostre osservazioni:

Dovreste vedere che è attiva l'uscita "6", come pure delle altre uscite.

Esaminate ora le uscite delle porte A, B e C, ed annotate nella tabella sottostante tutto quello che avviene a queste uscite, in base alle informazioni ottenute tramite la sonda logica.

	0 logico	1 logico	Impulso
A			
B			
C			

È questo quello che vi aspettavate? Siete capaci di spiegare quello che succede?

Si, è questo quello che ci si attendeva, perché il comando d'ingresso (PEEK) presente nel programma specificava il dispositivo 49318 come dispositivo d'ingresso e l'indirizzo decodificato è trovato all'uscita "6" del decodificatore. Quindi sarà attiva unicamente l'uscita "B". Nel programma non era specificato nessun altro dispositivo d'ingresso, e nessun altro dispositivo di uscita.

Passo n. 7

Modificate l'indirizzo dispositivo nella linea 10 del programma in modo che venga selezionato l'indirizzo 49325; la linea 10 sarà ora

```
10  A = PEEK(49325)
```

Eseguite il programma ed esaminate di nuovo le uscite A, B e C delle porte. Qualcuna di queste è attiva, ad indicare la presenza d'impulsi? E perché?

Non sarà attiva nessuna uscita, perché l'indirizzo dispositivo 49325 non è stato realizzato nel circuito, e, per di più, l'indirizzo 49325 non è immediatamente disponibile sulla scheda di prova. Di tutti gl'indirizzi che compongono il gruppo che va da 49312 a 49327, solo quelli che vanno da 49312 a 49319 sono disponibili sullo zoccolo "INDIRIZZO".

Passo n. 8

Modificate come segue la linea 10 del programma:

```
10 A = PEEK(49318): B = PEEK(49319)
```

In quale punto del circuito si avranno gl'impulsi, eseguendo il programma così modificato?

Vedrete che le uscite A e B sono attive; non lo è l'uscita C, perché è relativa agl'impulsi di controllo scrittura, e nel programma non ci sono comandi di uscita (POKE).

Passo n. 9

Apportate un'altra modifica al programma: questa volta modificate la linea 10 in

modo da poter controllare il dispositivo di uscita 49318. La vostra istruzione alla linea 10 sarà allora la seguente:

```
10  POKE 49318,0
```

Potete usare un qualsivoglia valore per il dato, purché compreso fra 0 e 255 incluso. Eseguite il programma ed esaminate le uscite A, B e C. Quale uscita prevedete che sia attiva? E le vostre previsioni sono realizzate?

È attiva l'uscita C, perché il comando di POKE è un comando di uscita, e l'indirizzo 49318 corrisponde al pin di uscita "6" del decodificatore. Sarete probabilmente sorpresi di vedere che è attiva anche l'uscita B. Quando un'istruzione di POKE è eseguita dall'interprete BASIC dell'Apple, il calcolatore effettua un'operazione di lettura prima della scrittura, per cui prima si legge dall'indirizzo selezionato, e poi si scrive nell'indirizzo selezionato. Questo è da tenere ben presente quando si progettano circuiti d'interfaccia.

Passo n. 10

È possibile configurare diversamente gl'interruttori presenti nel settore di decodifica indirizzo, in modo che siano generati dal decodificatore gl'indirizzi da 50944 a 50951? Come pensate di farlo? E questi indirizzi saranno realmente disponibili?

Sì, è possibile modificare la posizione degl'interruttori in modo che il decodificatore possa operare fra questi indirizzi. Innanzitutto convertite il primo indirizzo nel suo equivalente binario: 50944 = 11000111 00000000. Poi modificate le posizioni degl'interruttori per A15-A8 e per A7-A4. Ora, quali indirizzi corrisponderanno alle uscite "6" e "7" del decodificatore? Verificate le vostre risposte inserendo dei comandi di PEEK nel programmino con cui state lavorando in questo esperimento. Siete certamente capaci di vedere gl'impulsi alle uscite A e B provenienti dalle porte.

Fatte queste verifiche, badate di riportare gl'interruttori d'indirizzo nelle posizioni precedenti, corrispondenti al valore binario 11000000 10100000.

Non togliete il circuito dalla scheda di prova, perché vi servirà ancora. Il programma invece non verrà più utilizzato, per cui potete togliere corrente alla scheda di prova ed al calcolatore.

Esperimento n. 3
USO DEGL'IMPULSI DI SELEZIONE DISPOSITIVO

Scopo

In quest'esperimento si vedrà l'uso degl'impulsi di selezione dispositivo per controllare un dispositivo esterno. I comandi di PEEK e di POKE, anche se generalmente sono impiegati per controllare il flusso delle informazioni, possono anche servire per generare impulsi utili per controllare in modo semplice dispositivi esterni.

Presentazione dell'esperimento

In quest'esperimento un semplice dispositivo verrà attivato e disattivato per mezzo d'impulsi di selezione dispositivo. Come "dispositivo" useremo la sonda logica, ed un semplice flip-flop sarà controllato da due impulsi generati da software.

Configurazione dei pins del circuito integrato (v. Figura 6.5)

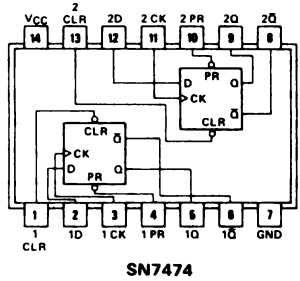


Figura 6.5 — Configurazione dei pins dei chips SN7402 ed SN7474.

Passo n. 1

In quest'esperimento usiamo lo stesso circuito dell'Esperimento n. 2. Se non lo avete ancora cablato, fatelo, rifacendovi alla Figura 6.4.

Passo n. 2

Cablate il flip-flop SN7474 come è mostrato in Figura 6.6. L'"1" che vedete scritto accanto all'ingresso "D" dell'SN7474 sta ad indicare che a quest'ingresso è applicato un uno logico (+5 volts). Uno "0" nella stessa posizione indicherebbe uno zero logico, o collegamento a massa. Le diciture 0 ed 1 servono a distinguere i collegamenti di livello logico dai collegamenti di alimentazione. L'uscita Q del flip-flop dev'essere l'unico dispositivo collegato alla sonda logica. Non dimenticate di alimentare il flip-flop SN7474: il pin 14 dev'essere messo a +5volts ed il pin 7 a massa.

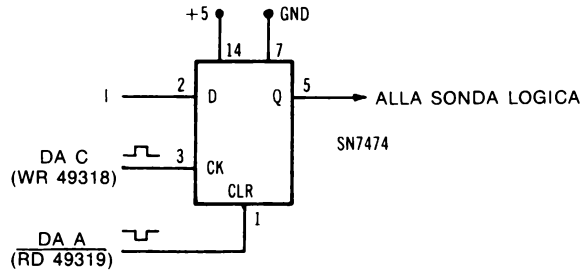


Figura 6.6 — Semplice circuito di controllo di un flip-flop.

Passo n. 3

In questo circuito l'impulso WR 49318 (segnale C) è il segnale di clock del flip-flop, e piloterà ad uno logico l'uscita, mentre l'impulso $\overline{\text{RD 49319}}$ (segnale A) la riporterà a zero logico. Dal momento che i flip-flops sono stabili in entrambi gli stati, una volta che il flip-flop ha ricevuto un impulso RD 49319, la sua uscita Q resta nello stato logico uno finquando non è tolta l'alimentazione, oppure finquando non è riportata a zero logico con un impulso $\overline{\text{WR 49318}}$.

Introducete nel calcolatore il programma che segue ed eseguitelo.

```
10  A = PEEK(49319)
20  POKE 49318,0
30  FOR T = 0 TO 300: NEXT T
40  A = PEEK(49319)
50  FOR T = 0 TO 300: NEXT T
60  GOTO 20
```

Non badate al lampeggio del LED, relativo agl'impulsi, della sonda logica. Quale effetto ha il programma sui LED relativi all'uno logico ed allo zero logico?

Lampeggiano ad indicare uno logico, zero logico, uno logico, e così via di seguito.

Passo n. 4

Modificate la routine di ritardo di tempo della linea 50 in questo modo:

```
50  FOR T = 0 TO 1000: NEXT T
```

Operata la modifica, eseguite il programma. Quale effetto avrà questa semplice modifica del programma?

Il LED relativo allo zero logico si accende per un tempo più lungo. Quindi è possibile generare impulsi di controllo di durata nota e prefissata, ad esempio di 1 secondo.

Passo n. 5

Siete in grado di determinare il ritardo software necessario, in uno statement di FOR ... : NEXT T, a generare un periodo di 1 secondo? Modificate il programma ed esaminate i vari contatori di ritardo fino ad arrivare a valori molto prossimi ad 1 secondo. Potete ad esempio provare con un periodo di 10 secondi, e poi dividere il contatore per 10 per ottenere quello relativo ad un periodo di 1 secondo. Con quale valore di contatore cominciate? Abbiamo constatato che uno statement di ritardo,

```
FOR T = 0 TO 780: NEXT T
```

richiede circa un secondo per essere eseguito.

Passo n. 6

A questo punto potete, servendovi del BASIC, dire al calcolatore per quanto tempo deve restare acceso ciascun LED. Potete quindi introdurre ed eseguire il programma che segue: questo prima di tutto vi chiede il periodo di accensione di ciascun LED, e poi esegue il resto del programma.

```
 10  A = PEEK(49319)
 20  INPUT "PERIODO DEL LED ROSSO"; Q
 30  INPUT "PERIODO DEL LED VERDE"; R
 40  PRINT "PERIODO GLOBALE DEL CICLO"; Q + R; "SECONDI"
 50  POKE 49318,0
 60  FOR S = 1 TO Q
 70  FOR T = 0 TO 780: NEXT T
 80  NEXT S
 90  A = PEEK(49319)
100  FOR S = 1 TO R
110  FOR T = 0 TO 780: NEXT T
120  NEXT S
130  GOTO 50
```

Durante l'esecuzione del programma i ritardi di tempo possono risultare alquanto più lunghi. Perché?

Le istruzioni in più del programma (FOR S = 1 TO Q, FOR S = 1 TO R e NEXT S) aumentano il tempo globale di esecuzione del programma, anche se di fatto non vi è un allungamento apprezzabile del programma.

Che cosa mostra questo programma?

Illustra diversi principi, riguardanti l'uso di programmi e circuiti semplici per controllare dispositivi esterni. Illustra poi la potenza del BASIC nel controllo di dispositivi esterni realizzati mediante istruzioni software relativamente semplici. Tenete presente tuttavia che il BASIC non è molto veloce.

Anche usando i comandi di PEEK e di POKE, la buona riuscita dell'interfaccia di flip-flop *non dipende dal fatto di trasferire effettivamente dati o informazioni*: il flip-flop è controllato, ovvero commutato, soltanto per mezzo d'impulsi di selezione dispositivo. Questo principio è spesso utilizzato quando è necessario un segnale di controllo o un impulso di controllo, senza alcun trasferimento di dati.

Non dimenticate, con riferimento all'interprete BASIC del calcolatore Apple, che, quando si ha un comando di POKE, *sono eseguite un'operazione di lettura ed un'operazione di scrittura*. Quindi, se decidete di ricorrere ad un comando di POKE per generare un impulso di selezione dispositivo a scopo di controllo, *ricordate che l'Apple eseguirà anche una lettura dallo stesso indirizzo*: Se usate due impulsi di controllo con lo stesso indirizzo, ad esempio WR XYZ ed RD XYZ, durante un'operazione di

scrittura determinata da un comando di POKE XYZ sarà attivato anche il segnale RD XYZ.

Potete togliere dalla scheda di prova il circuito di flip-flop SN7474, ma non il circuito SN7402. Il programma non sarà più usato, per cui togliete pure l'alimentazione al calcolatore.

Esperimento n. 4
COSTRUZIONE DI UNA PORTA D'INGRESSO

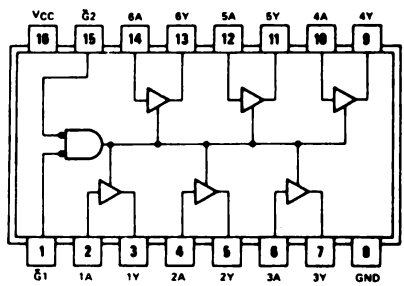
Scopo

Scopo di quest'esperimento è costruire una porta d'ingresso che utilizzi dei circuiti di buffer a tre stati.

Presentazione dell'esperimento

La semplice porta d'ingresso ad 8 bits la cui costruzione costituirà parte di quest'esperimento permetterà d'introdurre dati nel calcolatore. Questa porta d'ingresso interverrà anche in parecchi altri esperimenti. Qui useremo il circuito di selezione dispositivo dell'Esperimento n. 3, ed i chips di buffer a tre stati SN74365, o DM8095.

Configurazione dei pins del circuito integrato (v. Figura 6.7)



SN74365A
SN74LS365

Figura 6.7 — Configurazione dei pins del chip di buffer a tre stati SN74365, o DM8095.

Passo n. 1

Usiamo qui il circuito logico sviluppato nell'Esperimento n. 2. Se non c'è sulla vostra scheda di prova, rifatevi per i particolari alla Figura 6.4 e cablate il circuito ivi rappresentato. Il calcolatore e la scheda di prova devono essere spenti.

Passo n. 2

Cablate il circuito relativo alla porta d'ingresso ad 8 bits della Figura 6.8. Sono necessari due circuiti integrati a tre stati SN74365 (o DM8095).

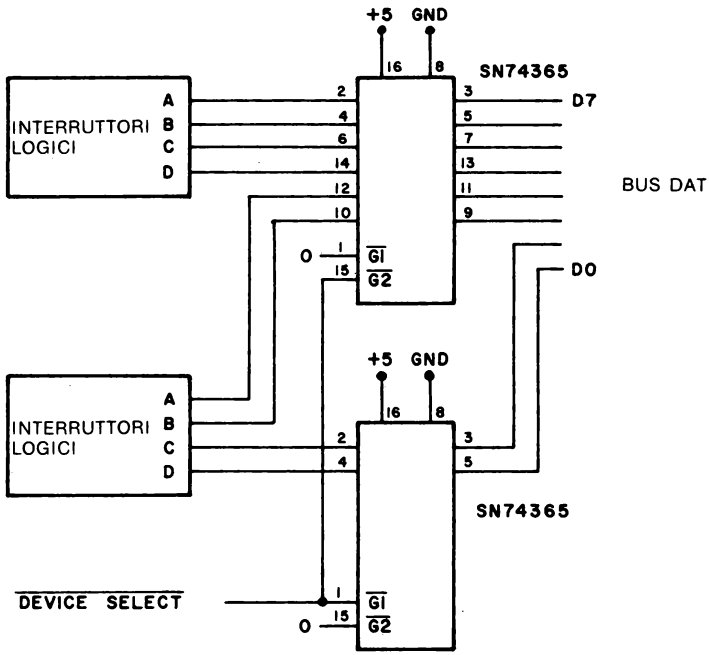


Figura 6.8 — Semplice porta d'ingresso ad 8 bits.

Passo n. 3

Notate che in questo circuito è utilizzato uno solo dei due ingressi di abilitazione dei chips di buffer a tre stati; l'ingresso non utilizzato è stato messo a massa, cioè collegato a zero logico. Quindi la porta interna non sarà usata per accoppiare un impulso di funzione ad un indirizzo di dispositivo. Il segnale di abilitazione sarà trasferito, attraverso la porta, direttamente ai circuiti di buffer a tre stati contenuti nel chip.

Collegate la linea di $\overline{\text{DEVICE SELECT}}$ al punto A (pin 1 dell'SN7402), come mostra la Figura 6.4: questo è il segnale per $\overline{\text{RD}}$ 49319.

La dicitura INTERRUTTORI LOGICI che compare in Figura 6.8 si riferisce ad interruttori capaci di generare segnali di uno logico o di zero logico agli otto singoli ingressi della porta d'ingresso. Si possono usare dei semplici ponticelli collegati alle linee di alimentazione a +5 volts ed a massa. In appendice troverete ulteriori chiarimenti su questo tipo di funzione logica.

Passo n. 4

Costruita la porta d'ingresso, e ottenuto l'invio dell'impulso di selezione dispositivo

della porta NOR SN7402, introducete ed eseguite il seguente programma di prova:

```
10  PRINT PEEK(49319): GOTO 10
```

Che cosa appare sullo schermo quando il programma è eseguito? Modificando gl'interruttori logici si avrà un qualche effetto sui valori visualizzati? Ed è questo quello che vi aspettavate di ottenere?

Viene visualizzato il valore 255, corrispondente a 11111111_2. La modifica degl'interruttori logici non incide sui valori già visualizzati. In un primo momento forse vi aspettavate che i valori cambiassero modificando la posizione degl'interruttori, ma questo non è avvenuto. Perché?

Il circuito d'interfaccia non ha un segnale di richiesta ingresso ($\overline{\text{INP REQ}}$), segnale che serve a porre i due buffers bus nella modalità ingresso.

Passo n. 5

Effettuate un collegamento fra il segnale A dell'SN7402, ovvero $\overline{\text{RD 49319}}$, e l'ingresso W posto nel settore INP REQ dello zoccolo "SEGNALI DI CONTROLLO". Questo segnale porrà i buffers bus 8216 nella modalità ingresso.

Effettuato il collegamento, fate ripartire il programma e cambiate la posizione degl'interruttori. Le modifiche delle posizioni degl'interruttori si traducono in una modifica dei numeri visualizzati? Provate parecchie posizioni diverse.

I valori degl'interruttori vengono ora trasferiti al calcolatore, convertiti in numeri decimali e visualizzati sullo schermo del monitor.

Se invece preferite vedere i numeri nella loro forma binaria, eseguite il programma che segue; questo programma visualizzerà, in modo continuo, i numeri binari.

```
 10  A = 128
 20  B = PEEK(49319)
 30  FOR Q = 1 TO 8
 40  IF B - A < 0 THEN GOTO 100
 50  PRINT "1";
 60  B = B - A
 65  A = A/2
 70  NEXT Q
 75  PRINT
 80  GOTO 10
100  PRINT "0";
110  GOTO 65
```

E ancora, se volete modificare la posizione di un interruttore, e *poi* averne l'equivalente in binario, modificate come segue la linea 10:

```
10  INPUT A$: HOME: A = 128
```

Ora, tutte le volte che desiderate visualizzare il valore binario relativo alla posizione dell'interruttore logico riferito alla porta d'ingresso, non dovrete far altro che premere il tasto RETURN sulla tastiera dell'Apple. È chiaro che le posizioni degl'interruttori sono già nel formato binario, per cui la correlazione fra il valore binario visualizzato ed i singoli bits della porta d'ingresso sarà una cosa semplice.

Non togliete il circuito dalla scheda di prova, e non staccate l'alimentazione: infatti tanto il programma quanto il circuito serviranno nel prossimo esperimento.

Esperimento n. 5
PORTE D'INGRESSO MULTIBYTE

Scopo

Scopo di quest'esperimento è mostrare come più bytes d'informazione possono essere acquisiti per essere elaborati da un programma in BASIC.

Presentazione dell'esperimento

Non tutti i dispositivi d'ingresso trasferiscono un solo byte d'informazione al calcolatore Apple: alcuni dispositivi infatti richiedono 9 o più bits. In quest'esperimento simulerete due porte d'ingresso servendovi della porta d'ingresso che abbiamo costruito nel corso dell'Esperimento n. 4, al quale dovrete riferirvi per i particolari costruttivi della porta d'ingresso medesima. Se non l'avete già fatto, vi raccomandiamo, prima di affrontare il presente esperimento, di realizzare l'Esperimento n. 4.

Passo n. 1

Se non avete una porta d'ingresso collegata con il vostro calcolatore Apple, vi rimandiamo all'Esperimento n. 4: infatti dovremo usare il circuito sviluppato in esso.

Passo n. 2

Quando si lavora su dati multibyte, bisogna programmare l'Apple in modo che i vari bytes vengano disposti ordinatamente, dal più significativo al meno significativo. In quest'esperimento assumeremo il byte "M" come byte più significativo (MSBY), ed il byte "L" come byte meno significativo (LSBY). Stante il fatto che l'Apple interpreta i valori ad 8 bits come numeri decimali compresi fra 0 e 255, sapete proporre un'equazione, oppure un insieme di operazioni con le quali ottenere l'equivalente decimale di un numero binario di due bytes?

L'MSBY è sfasato del fattore 256: di conseguenza potete ricorrere alla relazione seguente:

$$VALORE = (M*256) + L$$

ove VALORE è il valore decimale finale della parola di 16 bits.

Passo n. 3

Per verificare quest'equazione, introducete nel calcolatore questo programma:

```
200  INPUT "ASSEGNARE AGLI INTERRUTTORI IL VALORE DI MSBY"; A$
210  M = PEEK(49319)
220  INPUT "ASSEGNARE AGLI INTERRUTTORI IL VALORE DI LSBY"; A$
230  L = PEEK(49319)
240  V = (256*M) + L
250  PRINT V
260  GOTO 200
```

Ora eseguite il programma, facendolo partire introducendo GOTO 200 e premendo il tasto RETURN. Quando il calcolatore chiederà: "ASSEGNARE AGLI INTERRUTTORI IL VALORE DI MSBY?", ponete gli otto bits che danno il valore di MSBY sugli otto interruttori. Premete sulla tastiera il tasto RETURN. Alla domanda del calcolatore "ASSEGNARE AGLI INTERRUTTORI IL VALORE DI LSBY?" modificate gli otto interruttori in modo che rappresentino gli otto bits che volete introdurre per il valore di LSBY. Quando gl'interruttori saranno stati posizionati, premete il tasto RETURN: così il calcolatore saprà che siete pronti. A questo punto dovrà essere visualizzato il valore decimale sullo schermo del monitor. Qui di seguito riportiamo un elenco di valori a 16 bits tipici che potreste provare. Aggiungete voi i tre valori decimali corrispondenti, così come saranno generati dall'Apple. Questi dovreste verificarli abbastanza rapidamente con l'aiuto di una calcolatrice.

MSBY	**LSBY**	**VALORE**
11001010	11000001	
11000111	00011101	
00000001	10000001	

I valori che risulteranno sono 51905, 50973 e 385.

Passo n. 4

Il programma che segue è una combinazione del programma di uscita binaria e del programma relativo al calcolo decimale di due bytes: esso consente d'introdurre due bytes di otto bits, ottenendo un valore di 16 bits e la visualizzazione sia del valore decimale, sia di quello binario.

```
10  A = 32768
20  FOR S = 1 TO 2
30  FOR Q = 1 TO 8
40  IF B - A < 0 THEN GOTO 100
50  PRINT "1";
```

```
 60  B = B - A
 65  A = A/2
 70  NEXT Q
 75  PRINT "      ";: NEXT S
 80  PRINT: GOTO 200
100  PRINT "0";
110  GOTO 65
200  INPUT "ASSEGNARE AGLI INTERRUTTORI IL VALORE DI MSBY"; A$
210  M = PEEK(49319)
220  INPUT "ASSEGNARE AGLI INTERRUTTORI IL VALORE DI LSBY"; A$
230  L = PEEK(49319)
240  V = (256*M) + L
250  HOME: PRINT V
260  B = V: GOTO 10
```

Passo n. 5

Eseguite il programma introducendo il comando GOTO 200, e quindi premendo il tasto RETURN. Ponete i valori relativi a MSBY ed a LSBY sugl'interruttori. Ci sarà una relazione fra le posizioni degli interruttori fissate da voi ed i bits binari visualizzati sullo schermo. Siete certamente in grado di convertire senza problemi il valore binario in un valore decimale. Il valore binario di 16 bits è stato "scomposto" in due valori di 8 bits: in tal modo potete facilmente confrontare i bits con le posizioni degl'interruttori fissate da voi.

Ora che avete visto come l'Apple operi su bytes di 8 bits allo scopo di ricostituire un valore di 16 bits, capite certamente che è possibile eseguire anche altre operazioni. In quest'esperimento abbiamo usato una sola porta d'ingresso, ma si potrebbe facilmente costruirne un'altra con un nuovo indirizzo dispositivo, allo scopo di avere il byte di dato in più che occorre nell'applicazione su 16 bits simulata in quest'esperimento.

Avrete probabilmente osservato che in quest'esperimento, come pure nel precedente, abbiamo usato una nuova variabile, A$: si tratta di una variabile "fittizia", grazie alla quale il programma può venir fermato in un punto predeterminato, in modo da modificare le condizioni dell'esperimento prima di permettere al calcolatore di proseguire. La variabile A$ è una variabile-stringa, e, quando viene premuto il tasto RETURN, ad essa viene assegnata una stringa nulla di caratteri. Quindi è un "trucco" quello che ferma il calcolatore, finquando non premiamo il tasto RETURN.

Il circuito d'interfaccia che abbiamo usato in quest'esperimento servirà anche nel prossimo, quindi non toglietelo. Il software invece non servirà, quindi potete togliere l'alimentazione al calcolatore ed all'interfaccia.

Esperimento n. 6
APPLICAZIONI DELLE PORTE D'INGRESSO

Scopo

Scopo di quest'esperimento è mostrarvi come si possa usare una porta d'ingresso per applicazioni di controllo.

Presentazione dell'esperimento

In quest'esperimento la porta d'ingresso ad 8 bits sarà usata per trasferire informazioni all'Apple, che però elaborerà gli otto bits di dato in modo non numerico. In questo modo sarà controllato lo *stato* di otto dispositivi esterni.

Passo n. 1

In molti casi il calcolatore può essere utilizzato per elaborare informazioni non numeriche che lo ragguaglino sullo stato, o condizione, di dispositivi esterni. In tal modo è facile determinare quando un dispositivo è attivo o non attivo, una valvola è aperta o chiusa, un ascensore è su o giù, e così via.

Introducete nel calcolatore il programma che segue ed eseguitelo. Il programma mostra come un valore può far sì che il calcolatore prenda una direzione operativa preprogrammata.

```
10  INPUT A$: HOME
20  A = PEEK(49319)
30  IF A > 127 THEN GOTO 70
40  PRINT "INGRESSO < = 127"
50  GOTO 10
70  PRINT "INGRESSO > 127"
80  GOTO 10
```

Passo n. 3

Premete il tasto RETURN per far sì che il calcolatore effettui l'operazione di lettura ed il confronto. Ponete gl'interruttori logici relativi alla porta d'ingresso ad un valore minore di 127 (da 00000000 a 01111110), e premete il tasto RETURN. Che cosa accade? Provate con il valore 127 o uno maggiore (da 01111111 a 11111111). Che cosa accade? Che cosa accade quando il valore binario è uguale a 127 (01111111)?

Dovete vedere l'esatto messaggio per ciascun valore letto dal calcolatore. Questo programma illustra come con il calcolatore si possano prendere decisioni sulla base di un dato valore. In alcuni casi come base per una decisione può essere assunto il valore di un singolo bit. Il programma di conversione binaria dell'Esperimento n. 4 permetteva di vedere l'equivalente binario di un valore decimale; questo programma

invece prende delle decisioni sulla base del valore di singoli bits, per cui è possibile determinare se visualizzare un uno per ciascuna posizione del bit.

Passo n. 4

Utilizziamo ora la routine di visualizzazione in binario, ma questa volta, invece che degli uni e degli zeri, il calcolatore visualizzerà la parola "ON" per l'uno logico e la parola "OFF" per lo zero logico. Siete certamente in grado di modificare in tal senso il programma dell'Esperimento n. 4, modificando appunto gli statements di PRINT, ma in ogni caso vi proponiamo il programma già fatto. Notate che il programma dell'Esperimento n. 4 è stato "spostato", cioè i numeri di linea sono più alti. Prima d'introdurre questo programma, non dimenticate di cancellare quello vecchio, se non l'avete già fatto togliendo l'alimentazione. Per cancellare il vecchio programma potete usare il comando di NEW: basta battere la parola NEW, e subito dopo premere il tasto RETURN.

```
410  INPUT A$: HOME: A = 128
420  B = PEEK(49319)
430  FOR Q = 1 TO 8
440  IF B - A < 0 THEN GOTO 500
450  PRINT "ON    ";
460  B = B - A
470  A = A/2
480  NEXT Q
490  GOTO 410
500  PRINT "OFF   ";
510  GOTO 470
```

N.B.: dopo la parola ON ci sono due spazi, uno dopo la parola OFF: si fa così per avere una spaziatura uguale.

Eseguite il programma. Non dimenticate di posizionare gl'interruttori, e poi di premere il tasto RETURN, per avere la "conversione" e la visualizzazione. Verrà visualizzata una riga di messaggi di ON e di OFF: la scritta ON si riferisce ai bits ad uno logico, la scritta OFF ai bits a zero logico. Gli statements di PRINT possono venir modificati per visualizzare, in riferimento ai bits, le parole APERTO e CHIUSO, SOPRA e SOTTO, ed altri analoghi commenti per i bits.

Passo n. 5

Il programma presentato nel passo n. 4 trova un certo numero di applicazioni, ma a volte può essere più utile una visualizzazione in formato colonna dei messaggi ON ed OFF. Questo si può ottenere con i comandi BASIC di HTAB e VTAB. Utilizziamo ancora lo stesso programma di base del passo n. 4, facendo precedere gli statements modificati da un asterisco (*): in questa versione non è più necessario lasciare degli spazi dopo le parole ON ed OFF nelle linee 450 e 500 rispettivamente.

```
*400  H = 20: V = 8
 410  INPUT A$: HOME: A = 128
 420  B = PEEK(49319)
 430  FOR Q = 1 TO 8
 440  IF B - A < 0 THEN GOTO 500
*450  HTAB H: VTAB V: PRINT "ON";
 460  B = B - A
*470  A = A/2: V = V + 1
 480  NEXT Q
*490  GOTO 400
*500  HTAB H: VTAB V: PRINT "OFF";
 510  GOTO 470
```

Adesso le parole ON ed OFF vengono visualizzate in formato colonna, perché i comandi di HTAB e VTAB "spostano" il cursore lungo direttrici verticali.

Quindi le parole ON ed OFF possono essere visualizzate in diversi modi: infatti in alcuni calcolatori si possono associare rappresentazioni grafiche e caratteri alfanumerici, in modo che le parole ON/OFF possono essere visualizzate accanto ad una rappresentazione grafica del dispositivo o del procedimento che è sotto controllo.

Durante l'esecuzione del programma, apportate delle modifiche alle posizioni degl'interruttori allo scopo di verificare se la porta d'ingresso funziona correttamente.

Passo n. 6

Supponiamo che vogliate eseguire il programma in maniera continua, in modo da poter modificare gl'interruttori e controllare le condizioni di ON/OFF senza dover premere il tasto RETURN ogni volta che occorre una nuova visualizzazione. INPUT A$ è il comando d'ingresso "fittizio" grazie al quale il calcolatore si ferma e aspetta che si prema il tasto RETURN. Togliete questo statement dal programma: la linea 410 diventerà

```
410  HOME: A = 128
```

Eseguite di nuovo il programma. Si avrà una visualizzazione accettabile? E perché?

L'immagine visualizzata sarà tremolante, perché il comando di HOME cancella tutto lo schermo e posiziona il cursore nell'angolo in alto a sinistra dello schermo del monitor ogni volta che il calcolatore fa ripartire il programma. Questo fatto richiede tempo, e quindi rallenta la visualizzazione. Sapete proporre delle altre modifiche del programma per ridurre, o eliminare, lo sfarfallio?

Passo n. 7

Togliendo il comando di HOME, potete ridurre il tempo che l'Apple impiega per pulire tutto lo schermo e portare il cursore nella posizione di riposo nell'angolo in al-

to a sinistra dello schermo. Invece i comandi di HTAB e VTAB permettono di posizionare il cursore esattamente nel posto in cui visualizzare un ON o un OFF per linea e per ciascun bit. Se nella linea 450 non si lasciano due spazi dopo la parola ON, la stampa di questa parola non coprirà la seconda F di OFF, per cui si vedrà ONF invece di ON. Quindi gli spazi sono necessari per "cancellare" tutti i caratteri che rimangono sulla linea.

Vi consigliamo di formulare così la linea 410 nel vostro programma:

```
410  A = 128
```

Ora fate partire il programma battendo HOME: GOTO 400, e poi premendo ENTER. Se non usate il comando di HOME, il programma inizierà a scrivere *su* ciò che si trova sullo schermo. Il comando di HOME pulisce lo schermo subito prima che il programma parta.

Passo n. 8

I comandi di VTAB e di HTAB possono servire anche per generare titoli o intestazioni per ciascuna delle otto linee d'informazione visualizzate sullo schermo. Qui di seguito trovate una serie d'intestazioni: a voi modificarle, o aggiungerne delle altre.

```
 5  HOME
10  VTAB 8: HTAB 1
15  PRINT "POMPA PER ACIDI";
20  VTAB 9: HTAB 1
25  PRINT "POMPA PER BASI";
30  VTAB 10: HTAB 1
35  PRINT "FORNO";
40  VTAB 11: HTAB 1
45  PRINT "MISCELATORE";
50  VTAB 12: HTAB 1
55  PRINT "CICLO DI FLUSSAGGIO";
60  VTAB 13: HTAB 1
65  PRINT "IMPIANTO DI LAVAGGIO";
70  VTAB 14: HTAB 1
75  PRINT "VUOTO";
80  VTAB 15: HTAB 1
85  PRINT "ESSICCATORE";
```

Se avete l'intenzione di passare subito all'Esperimento n. 7, vi consigliamo di aggiungere queste linee al programma, e di provarlo dopo aver effettuato l'aggiunta.

Nel prossimo esperimento useremo tanto l'hardware quanto il software di questo appena terminato, per cui non smontate il circuito e non togliete l'alimentazione al calcolatore.

Esperimento n. 7
APPLICAZIONI DELLE PORTE D'INGRESSO (2)

Scopo

Scopo di quest'esperimento è illustrare come si possano eseguire operazioni logiche sui dati.

Presentazione dell'esperimento

In quest'esperimento si eseguiranno operazioni di AND su informazioni di tipo ON/OFF provenienti da otto "sensori" esterni. Le condizioni dei sensori avranno l'effetto di far scattare delle azioni ben precise da parte del calcolatore.

Passo n. 1

In quest'esperimento è utilizzato lo stesso programma dell'Esperimento n. 6. Nel caso che non lo abbiate introdotto tutto nel calcolatore, introducetelo e verificatelo. Se invece lo avete già introdotto e verificato nel corso dell'esperimento precedente, potete controllarlo facendo riferimento al seguente listato:

```
  5  HOME
 10  VTAB 8: HTAB 1
 15  PRINT "POMPA PER ACIDI";
 20  VTAB 9: HTAB 1
 25  PRINT "POMPA PER BASI";
 30  VTAB 10: HTAB 1
 35  PRINT "FORNO";
 40  VTAB 11: HTAB 1
 45  PRINT "MISCELATORE";
 50  VTAB 12: HTAB 1
 55  PRINT "CICLO DI FLUSSAGGIO";
 60  VTAB 13: HTAB 1
 65  PRINT "IMPIANTO DI LAVAGGIO";
 70  VTAB 14: HTAB 1
 75  PRINT "VUOTO";
 80  VTAB 15: HTAB 1
 85  PRINT "ESSICCATORE";
400  H = 20: V = 8
410  A = 128
420  B = PEEK(49319)
430  FOR Q = 1 TO 8
440  IF B - A < 0 THEN GOTO 500
450  HTAB H: VTAB V: PRINT "ON";
460  B = B - A
470  A = A/2: V = V + 1
```

```
480  NEXT Q
490  GOTO 400
500  HTAB H: VTAB V: PRINT "OFF";
510  GOTO 470
```

Una volta caricato e verificato con successo, il programma darà luogo ad una visualizzazione come quella riportata nella Tabella 6.1. Probabilmente le condizioni di ON e di OFF visualizzate dal vostro calcolatore saranno diverse da quelle in tabella, in conformità alle diverse posizioni degl'interruttori sulla vostra porta d'ingresso.

POMPA PER ACIDI	ON
POMPA PER BASI	OFF
FORNO	ON
MISCELATORE	ON
CICLO DI FLUSSAGGIO	ON
IMPIANTO DI LAVAGGIO	ON
VUOTO	OFF
ESSICCATORE	OFF

Tabella 6.1 — Uscita del programma di controllo.

Passo n. 2

Accanto alla Tabella 6.1 annotate i bits della porta d'ingresso corrispondenti alle varie etichette. Per fare questo, potete verificare i bits d'ingresso, oppure analizzare il programma. Troverete così ad esempio che il bit D7 corrisponde a "POMPA PER ACIDI", il bit D6 a "POMPA PER BASI", e così via fino al bit D0, che corrisponde ad "ESSICCATORE".

Passo n. 3

Riprendete l'Esempio 4.3 del Capitolo 4, ed introducete il relativo programma in linguaggio assemblatore nel calcolatore servendovi del Monitor dell'Apple: per richiamare il Monitor è sufficiente battere CALL — 151 e poi RETURN. Verificate che il programma sia stato introdotto correttamente, e non dimenticate che il programma Monitor lavora su numeri esadecimali. Se non sapete come si usa il Monitor, consultate l'*Apple II Reference Manual*, oppure procedete secondo i tre punti seguenti:

1) Premere il tasto RESET, battere CALL —151 e premere il tasto RETURN. L'Apple risponderà con un asterisco (*).
2) Battere 0300: 00 00 00 48 AD 00 03 2D 01 03 8D 02 03 68 60, lasciando uno spazio fra i gruppi di due cifre. Scrivere 00 per i primi tre valori del programma.

3) Premere il tasto RETURN, battere 02FF, premere il tasto RETURN, poi premere due volte il tasto RETURN e verificare i dati confrontandoli con quelli del listato dell'Esempio 4.3 e con quelli riportati prima.

Passo n. 4

Per verificare il programma in linguaggio assemblatore, introducete nel calcolatore il programma che segue ed eseguitelo. Fate a mano le opportune conversioni da decimale a binario e da binario a decimale per verificare i risultati. Premete il tasto RESET per tornare in BASIC.

```
1000  POKE 10,76: POKE 11,03: POKE 12,03
1010  INPUT "BYTE MASCHERA"; M: POKE 768,M
1020  INPUT "BYTE DATO"; D: POKE 769,D
1030  Q = USR(0): PRINT "RISULTATO"; PEEK(770)
1040  GOTO 1010
```

Se le risposte ottenute coincidono con quelle che avevate calcolato a mano, procedete con il passo successivo. In caso contrario controllate attentamente che tutte le istruzioni del programma in linguaggio assemblatore siano state introdotte correttamente, e verificate il programma ancora una volta. Tenete anche presente che gli errori potrebbero essere nei calcoli fatti a mano.

Passo n. 5

A questo punto vi chiediamo di modificare il vostro programma in modo che distingua quando è attiva l'una o l'altra delle apparecchiature IMPIANTO DI LAVAGGIO, ESSICCATORE e VUOTO, e quando sono attive entrambe le condizioni di POMPA PER ACIDI e di POMPA PER BASI. Si può ricorrere alla subroutine in linguaggio assemblatore per le operazioni di AND logico: ciò non toglie però che anche altre soluzioni sortirebbero il medesimo effetto.

D7	D6	D5	D4	D3	D2	D1	D0	
1	1	X	X	X	X	X	X	Pompe per acidi e per basi attive entrambe
X	X	X	X	X	0	0	1	
X	X	X	X	X	0	1	0	
X	X	X	X	X	0	1	1	
X	X	X	X	X	1	0	0	Tutte le apparecchiature attive
X	X	X	X	X	1	0	1	
X	X	X	X	X	1	1	0	
X	X	X	X	X	1	1	1	

X = non significativo (uno o zero logico).

Tabella 6.2 — Condizioni di controllo da rilevare.

Siete in grado di proporre un metodo per rilevare le condizioni richieste? Noi vi consigliamo di ripassarvi l'operazione di AND logico, di cui si è parlato nel Capitolo 4. Immaginate le operazioni così come compaiono nella Tabella 6.2.

Passo n. 6

L'operazione di AND logico può essere utilizzata per mascherare i bits che non servono, che sono i bits D5-D0 nella verifica delle pompe ed i bits D7-D3 nella verifica delle apparecchiature. Quindi bisogna predisporre due "maschere", una per le pompe ed una per le apparecchiature. Quali devono essere queste maschere, in valore decimale ed in valore binario?

La maschera per le pompe sarà 11000000_2, o, in decimale, 192, mentre la maschera per le apparecchiature sarà 00000111_2, o, in decimale, 7. Quando queste maschere vengono messe in AND con i valori d'ingresso provenienti dai sensori, o dagl'interruttori logici, i bits desiderati sono "filtrati" attraverso le maschere.

Passo n. 7

Ora che avete predisposto le due maschere, pensate a delle istruzioni software con cui si possa determinare lo stato dei bits "filtrati": dovete pensare ai singoli bits, come pure ai loro equivalenti decimali. Se ne avete bisogno, usate pure delle nuove variabili.

Noi abbiamo usato una nuova variabile, C, per rappresentare il valore introdotto dai sensori; ciò permette di usare la variabile B in modo indipendente nella parte del programma che tratta la visualizzazione delle condizioni ON/OFF. Se usate la variabile B, constaterete che è sempre uguale a zero. A voi provare e scoprire il perché. Noi abbiamo usato le istruzioni

```
POKE 768,7: POKE 769,C: Q = USR(0)
IF PEEK(770) = 0 THEN ...
```

e le istruzioni

```
POKE 768,7: POKE 769,C: Q = USR(0)
IF PEEK(770) > 0 THEN ...
```

per rilevare le condizioni delle apparecchiature, ed altre analoghe per rilevare le condizioni delle pompe. In tutti i casi comunque lo statement di THEN ... viene eseguito su una delle due condizioni, mentre se il programma prosegue viene considerata l'altra condizione.

Passo n. 8

Per verificare i vostri contributi personali al programma, aggiungete delle ulteriori istruzioni al programma di rilevazione degli stati dei flags, in modo che sia visualiz-

zato sullo schermo il messaggio "PERICOLO" se sono in funzione entrambe le pompe, ed il messaggio "APPARECCHIATURE" se è una delle apparecchiature ad essere in funzione. Scrivete queste nuove istruzioni nello spazio che lasciamo in bianco qui di seguito, e, prima di modificare il programma, riesaminatele accuratamente. Se intendete impiegare la subroutine in linguaggio assemblatore, non dimenticare d'inserire una linea come la linea 1000 del programma presentato nel passo n. 4. Questa linea di programma inizializza le tre locazioni utilizzate dal comando di USR, in modo che il calcolatore inizi ad eseguire l'opportuna subroutine.

La sezione di programma che avete scritto sarà probabilmente uguale a questa:

```
420  B = PEEK(49319): C = B
 .
 .
 .
490  GOTO 600
 .
 .
 .
600  POKE 768,7: POKE 769,C
605  Q = USR(0)
610  IF PEEK(770) = 0 THEN 700
615  HTAB 20: BTAB 17: PRINT "APPARECCHIATURE";
620  POKE 768,192
625  Q = USR(0)
630  IF PEEK(770) <> 192 THEN 800
635  HTAB 20: VTAB 18: PRINT "PERICOLO";
640  GOTO 400
700  HTAB 20: VTAB 17: PRINT "          ";
710  GOTO 620
800  HTAB 20: VTAB 18: PRINT "          ";
810  GOTO 400
```

Verificate il vostro programma: potreste aver dimenticato le istruzioni per cancellare dallo schermo le scritte APPARECCHIATURE e PERICOLO, oppure d'inserire i tre comandi di POKE con cui sono lette le informazioni richieste dal comando di USR. Per fare queste correzioni non è necessario aggiungere nessuna istruzione al programma: basta battere i comandi di POKE, e poi il tasto RETURN, dato che questi comandi devono essere eseguiti una sola volta.

I comandi delle linee 700 e 800 stampano degli spazi, per cui di fatto sono utilizzati per cancellare i segnali APPARECCHIATURE e PERICOLO visualizzati. Questo programma potrebbe essere molto più complesso, con istruzioni per la visualizzazione nero su bianco, o con scritte lampeggianti quando il programma rileva una condizione d'emergenza.

A questo punto avrete certamente capito che il software è in grado di gestire sia operazioni matematiche che operazioni logiche, e che l'impiego di subroutines in linguaggio assemblatore non presenta troppe difficoltà.

Spegnete pure il calcolatore, anche se il programma in linguaggio assemblatore che esegue le operazioni di AND servirà ancora in seguito. Dato che la porta d'ingresso sarà utilizzata di nuovo, non smontate il circuito.

Esperimento n. 8
COSTRUZIONE DI UNA PORTA DI USCITA

Scopo

Scopo di quest'esperimento è costruire una semplice porta di uscita ad 8 bits e studiarne l'uso.

Presentazione dell'esperimento

In quest'esperimento useremo un semplice circuito di latch ad 8 bits per costruire una porta di uscita, che utilizzeremo qui ed in alcuni degli esperimenti che seguiranno, nei quali si avrà bisogno di trasferire informazioni a dispositivi esterni. Ricorreremo a due circuiti integrati di quattro latches ciascuno, gli SN7475.

Configurazione dei pins del circuito integrato (v. Figura 6.9)

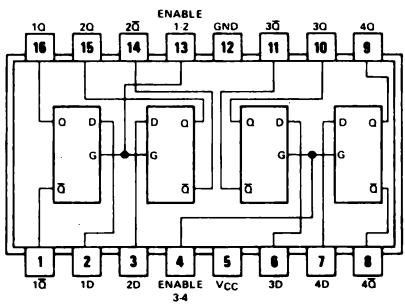


TABELLA DI FUNZIONE
(per ciascun latch)

INGRESSI		USCITE	
D	G	Q	$\bar{Q}$
L	H	L	H
H	H	H	L
X	L	Q_0	$\bar{Q}_0$

H = livello alto, L = livello basso,
X = non significativo
Q_0 = livello di Q prima della transizione di Q dal livello alto al livello basso

Figura 6.9 — Configurazione dei pins del chip di latch quadruplo SN7475.

Passo n. 1

Useremo il circuito logico introdotto nell'Esperimento n. 2. Se non lo avete già pronto sulla vostra scheda di prova nella parte forata, vi consigliamo di effettuare prima l'Esperimento n. 2, e poi questo. Comunque potete anche cablare ed usare direttamente il circuito logico dell'Esperimento n. 2. Per i particolari riferitevi alla Figura 6.4.

Passo n. 2

Cablate il circuito rappresentato in Figura 6.10. Sono necessari due circuiti integrati di latch SN7475 ed otto segnalatori luminosi specifici, oppure otto equivalenti circuiti di rilevazione di livello logico. Per il momento non collegate l'ingresso di selezione dispositivo DEV SEL.

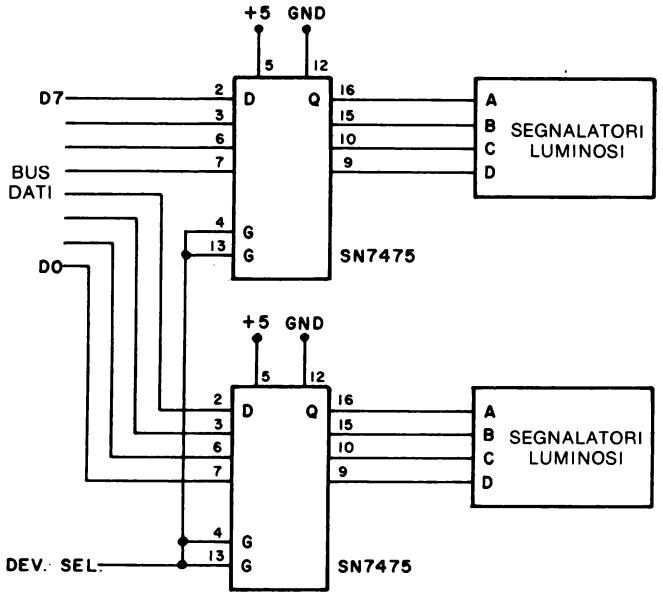


Figura 6.10 — Schema di un'elementare porta di uscita ad 8 bits.

Passo n. 3

Riferendovi al circuito illustrato in Figura 6.4, fate delle prove per determinare quale delle tre uscite di controllo A, B e C sarà usata per controllare gl'ingressi di abilitazione latch che sono collegati alla linea DEV SEL. Per quale delle tre optate? E perché?

L'uscita A, $\overline{\text{RD 49319}}$, è già stata usata, e la RD 49318 non va bene, perché è decodificata per una porta d'ingresso. La porta da usare è la WR 49318 (C), che fornisce un impulso positivo dello stesso tipo di quello richiesto dai chips di latch SN7475. Questa uscita è decodificata anche per un dispositivo di uscita. Certo ricorderete che i pins $\overline{7}$ e $\overline{6}$ di uscita dal decodificatore posto sulla scheda a circuito stampato corrispondono rispettivamente agl'indirizzi decodificati 49319 e 49318.

Fate un collegamento fra il pin 13 dell'SN7402 ed i pins 4 e 13 dei due chips di latch SN7475: questo è il collegamento DEV SEL illustrato in Figura 6.10.

Passo n. 4

Per verificare la porta di uscita, introducete questo programma nel calcolatore:

```
10  A = 0
20  POKE 49318,A
30  END
```

Inizializzate la variabile A a zero, come si vede nel programma, e fate partire l'esecuzione. Come si comportano i segnalatori luminosi?

Restano spenti, perché alla porta di uscita è stato trasferito uno zero. Ora ponete A a 255, ed eseguite ancora una volta il programma: i LED dovrebbero accendersi tutti; se ciò non accade, controllate di nuovo il circuito e verificate il programma.

Passo n. 5

Si può modificare il programma in modo da poter introdurre facilmente da tastiera i nuovi valori. Ed ecco il programma modificato:

```
10  INPUT A
20  POKE 49318,A
30  GOTO 10
```

Potete provare tutti i valori che volete, ma noi vi consigliamo di cominciare con potenze di due (0, 1, 2, 4, 8, etc.), perché questi valori verificano i singoli LED.

Dato che una porta di uscita ad 8 bits può visualizzare unicamente valori compresi fra 0 e 255, che cosa accade se provate ad inviare in uscita un valore che non rientra in quest'intervallo? Pensate che sarà visualizzata una "parte" del valore, ad esempio gli otto bits meno significativi? Provate ad eseguire il programma con il valore 256. Che cosa succederà?

L'Apple visualizzerà il messaggio

```
? ILLEGAL QUANTITY ERROR IN 20
```

che significa "ERRORE PER QUANTITÀ NON CONSENTITA ALLA LINEA 20", che segnala che il valore non rientra nell'intervallo giusto in rapporto alla funzione richiesta. Il messaggio riporta anche il numero di linea in cui si trova l'"errore". In questo modo vengono "localizzati" anche i numeri negativi.

Passo n. 6

Fate ripartire il programma ed introducete il valore 90. I segnalatori luminosi visualizzeranno il valore 01011010. Provate allora ad introdurre il valore −24. Una volta scoperto l'errore, e visualizzato il messaggio di errore, il valore visualizzato cambia?

No. Le condizioni di errore vengono rilevate prima che venga eseguita l'istruzione di POKE. Come credete che l'Apple lavori su numeri frazionari? Introducete una frazione decimale, ad esempio 6.01. Che cosa apparirà?

L'Apple "eliminerà" la parte decimale del numero. Provate a ripetere l'operazione con altri numeri.

Passo n. 7

Siete in grado di scrivere un breve programma utilizzabile per incrementare un valore compreso fra 0 e 255 e per visualizzare sui LED ogni nuovo valore? Scrivete il programma nello spazio che lasciamo in bianco qui di seguito, e verificatelo. Che cosa accade? E siete capaci di scrivere il programma in modo che esegua dei cicli su sé stesso, sicché l'incremento del contatore venga visualizzato continuamente?

Questo è il programma che noi proponiamo:

```
10  FOR A = 0 TO 255
20  POKE 49318,A
30  NEXT A
40  GOTO 10
```

Non dimenticate che non potete superare il valore 255, né restare al di sotto dello 0, sennò verrà generato un messaggio di errore. Potete anche introdurre nel programma un breve ritardo, per evitare che i LED lampeggino troppo in fretta; ad esempio con questo statement:

```
25  FOR T = 0 TO 500: NEXT T
```

Con questo avete visto che è abbastanza semplice costruire una porta di uscita, e controllarla con semplici comandi software.

La porta di uscita servirà ancora nei prossimi esperimenti, però l'alimentazione può essere tolta.

Esperimento n. 9
INTERAZIONE FRA PORTE D'INGRESSO E PORTE DI USCITA

Scopo

Scopo di quest'esperimento è mostrare come utilizzare in uno stesso programma comandi relativi alle porte d'ingresso ed alle porte di uscita.

Presentazione dell'esperimento

In alcuni casi nei circuiti d'interfaccia si usano insieme porte d'ingresso e porte di uscita, controllate mediante comandi di PEEK e di POKE contenuti nel medesimo

programma; fra le porte avvengono spesso dei trasferimenti d'informazioni. In quest'esperimento potrete vedere come queste porte si possano usare insieme in un circuito elementare.

Passo n. 1

In quest'esperimento usiamo la porta d'ingresso dell'Esperimento n. 7 e la porta di uscita dell'Esperimento n. 8, rifacendoci per i particolari dei circuiti agli Esperimenti n. 2, 3 e 8.

Passo n. 2

Costruite la porta d'ingresso e quella di uscita, introducete nel calcolatore il programma che segue ed eseguitelo. Questo programma serve a verificare i circuiti delle porte di I/U.

```
10  A = PEEK(49319)
20  POKE 49318,A
30  GOTO 10
```

Come voi posizionate gl'interruttori logici della porta d'ingresso, dovreste vedere i bits corrispondenti della porta di uscita cambiare secondo il posizionamento degl'interruttori in ingresso. Se questo non si verifica, controllate di nuovo il circuito ed il programma.

Passo n. 3

In questa fase desideriamo introdurre da tastiera due valori, che verranno quindi visualizzati sui LED. A questo punto dovreste essere in grado di scrivere un breve programma che realizzi quest'operazione.

Noi ci siamo serviti di questo programma, nel quale sono simulati il bit più significativo (MSBY) ed il byte meno significativo (LSBY):

```
10  INPUT "MSBY"; A$: M = PEEK(49319)
20  INPUT "LSBY"; A$: L = PEEK(49319)
30  POKE 49318,M
40  INPUT A$
50  POKE 49318,L
60  GOTO 10
```

In questo programma, la variabile-stringa A$ ha il ruolo di variabile "fittizia", che "ferma" il calcolatore, dandovi la possibilità di effettuare le operazioni necessarie prima che il programma possa proseguire.

Passo n. 4

Eseguite il programma. Siete certamente capaci d'introdurre nel calcolatore due valori. Quando voi battete RUN RETURN, il calcolatore è pronto, e perciò potete posizionare gl'interruttori con il valore dell'MSBY. Ora premete il tasto RETURN: il calcolatore potrà eseguire lo statement di acquisizione dei dati. Poi posizionate il valore dell'LSBY sugl'interruttori e premete nuovamente il tasto RETURN. Acquisito l'LSBY, sarà visualizzato l'MSBY. Premendo RETURN, il calcolatore visualizzerà l'LSBY.

Passo n. 5

Questo programma mostra come il calcolatore acquisisce e memorizza valori da visualizzare in seguito. Otto bits d'informazione si elaborano facilmente. Come può essere visualizzato su due porte di uscita un numero compreso fra 0 e 65535?

I numeri di questo tipo vanno "spezzati" in un MSBY di 8 bits ed un LSBY di 8 bits. Sapete dire come?

Bisogna dividere il numero per 256: l'MSBY sarà la parte *intera* del risultato. Ad esempio, se prendiamo come numero di partenza 10923, avremo

```
10923/256 = 42.668
```

La parte intera del risultato, 42, convertita in un numero binario di 8 bits, sarà l'MSBY del valore. Possiamo calcolare anche l'LSBY:

```
10923 – (42*256) = 171
```

Anche qui, l'equivalente binario su 8 bits di 171 sarà l'LSBY.
Scrivete un programma in BASIC con il quale l'Apple esegua queste funzioni.

Passo n. 6

Questo è il programma che abbiamo sviluppato per effettuare la "conversione":

```
10  INPUT "VALORE"; V
20  M = V/256
30  L = V – INT(M)*256
40  PRINT INT(M), L
50  INPUT A$
60  POKE 49318,M
70  INPUT A$
80  POKE 49318,L
90  GOTO 10
```

L'MSBY e l'LSBY saranno visualizzati sullo schermo video nella loro forma decimale. Per chiarezza abbiamo utilizzato il comando di INT per eliminare la parte de-

cimale dal valore relativo ad M. Per l'operazione di POKE questo non è necessario, perché in esso la parte decimale viene ignorata.

Passo n. 7

Introducete nel calcolatore il nostro o il vostro programma, e verificatelo. Premete il tasto RETURN per visualizzare l'MSBY sui LED, e tornate a premerlo per visualizzare l'LSBY.

Potete introdurre valori maggiori di 65535? Possono questi valori venir convertiti e visualizzati?

Sì, potete introdurli, e possono venir convertiti, ma non visualizzati, perché daranno luogo a risultati maggiori di 256 come MSBY, e questo genera una condizione di errore. Si può fare qualcosa per evitarlo?

Si possono aggiungere al programma degli statements che, prima di fare la conversione, controllino se il valore si colloca in un determinato intervallo. Si possono introdurre poi anche degli statements che eliminino la parte frazionaria del numero introdotto. Ad esempio si potranno introdurre gli statements seguenti:

```
12  IF V < = 65535 AND V > = 0 THEN 18
14  PRINT "VALORE FUORI INTERVALLO, RIPROVARE": GOTO 10
18  V = INT(V)
```

Provate ad inserire nel programma questi statements: statements come questi prevengono errori, e facilitano l'uso del programma da parte di un generico operatore. Tenete presente questo tipo di programmazione quando scrivete programmi complessi.

Esperimento n. 10
REGISTRAZIONE E VISUALIZZAZIONE DEI DATI

Scopo

Scopo di quest'esperimento è mostrare come usare la porta d'ingresso per acquisire informazioni, e come il calcolatore memorizzi queste informazioni per visualizzarle poi sui LED.

Presentazione dell'esperimento

In quest'esperimento un insieme di 10 valori verrà acquisito, come dati, dalla porta d'ingresso a tre stati, e successivamente visualizzato sui LED. Si possono immaginare anche dispositivi di visualizzazione più flessibili; così è pure possibile acquisire un maggior numero di dati.

Passo n. 1

Useremo la porta d'ingresso e la porta di uscita descritte precedentemente. Dovreste ormai conoscerle bene, ma in ogni caso vi rimandiamo per i particolari necessari agli Esperimenti 2, 3 e 8. Se non li avete ancora realizzati, vi consigliamo di farlo prima di passare all'esperimento in corso.

Passo n. 2

In quest'esperimento utilizzerete il calcolatore per acquisire e visualizzare un insieme di dati acquisiti dalla porta d'ingresso. Il programma di acquisizione dati è del tipo presentato qui di seguito:

```
50  INPUT A$
60  Q = PEEK(49319)
70  INPUT A$
80  R = PEEK(49319)
```

Come vedete, è necessario un certo numero di statements per acquisire piccole quantità d'informazione. Volete proporre una soluzione alternativa?

Una lista di valori può essere acquisita per mezzo di un ciclo, e per memorizzare l'informazione si può ricorrere ad un vettore: in tal modo non è necessario assegnare una nuova variabile a ciascun valore dell'insieme di dati. Siete in grado di scrivere un breve programma che acquisisca 10 valori?

Noi abbiamo scritto questo programma, al quale i vostri dovrebbero assomigliare. Osservate che per memorizzare l'informazione ci siamo serviti di un vettore.

```
 10  DIM A(10)
 20  PRINT "INIZIO"
 30  FOR P = 1 TO 10
 40  INPUT A$
 50  A(P) = PEEK(49319)
 60  NEXT P
 70  PRINT "INIZIO VISUALIZZAZIONE ..."
 80  FOR P = 1 TO 10
 90  GET A$
100  PRINT A(P): POKE 49318,A(P)
110  NEXT P
120  PRINT "FINE ESECUZIONE": END
```

In questo programma si deve premere il tasto RETURN per far sì che il calcolatore acquisisca un valore. Quando il calcolatore scrive sullo schermo "INIZIO VISUALIZZAZIONE ...", vuol dire che si visualizzerà un valore memorizzato in precedenza

ogni volta che premerete il tasto RETURN. Il valore sarà visualizzato anche sui LED, in forma binaria. Qui abbiamo usato il comando GET A$, invece di INPUT A$. C'è differenza fra le due soluzioni?

Sì: il comando GET A$ elimina il punto interrogativo (?), e permette di battere un qualsivoglia tasto (A, &, 1, etc.) invece del tasto RETURN. Il simbolo alfanumerico *non* viene visualizzato: questo permette di "tener pulito" lo schermo dai valori dei dati.

Passo n. 3

Eseguite il vostro o il nostro programma per acquisire 10 valori di dati e, dopo averli introdotti, fateli visualizzare dal calcolatore. Quali risultati avrete?

Dovrete aspettarvi che i valori siano stati memorizzati correttamente, e che quindi siano visualizzati e stampati sullo schermo del video. Se non si desidera che i valori siano presentati alla porta di uscita, si può modificare il programma in modo che si limiti a visualizzare i valori sul monitor?

Sì; basta modificare così la linea 100:

```
100  PRINT A(P)
```

e togliere la linea 90.

Passo n. 4

La modalità grafica a bassa risoluzione dell'Apple può anche servire per visualizzare i valori in forma grafica. Vi suggeriamo di provare ad usare il comando di HLIN per tracciare un insieme di linee orizzontali che rappresentino i corrispondenti valori introdotti dalla porta d'ingresso. Tenete presente che la zona dello schermo per il comando di HLIN è limitata a 39 punti tanto in altezza quanto in larghezza.

Nello spazio qui sotto scrivete il vostro programma di visualizzazione.

Noi abbiamo generato un grafo a righe orizzontali dell'informazione, con questa sequenza di programma:

```
 80  GR: COLOR = 5
 90  FOR P = 1 TO 10
100  D = A(P)/6.5
110  HLIN 0,D AT P
120  NEXT P
130  END
```

Questi statements vanno aggiunti al programma scritto nel passo n. 2. Provate il vostro programma, o quello proposto da noi.

In questa sequenza di programma il valore del dato viene diviso per 6.5; in tal modo, invece che un intervallo compreso fra 0 e 255, si ha un intervallo "condensato", che va da 0 a 39. Abbiamo poi usato l'indice del vettore per incrementare la posizione iniziale di ciascuna riga orizzontale: i dati partono dalla parte alta dello schermo in corrispondenza di A(1), e vengono via via rappresentati nelle righe inferiori. Potete anche usare il valore di P per avere un colore diverso per ciascuna riga orizzontale.

Passo n. 5

Si possono poi apportare al programma delle altre modifiche, che portino all'introduzione di una routine di ritardo di tempo al posto del comando INPUT A$. In tal modo i valori dei dati saranno ottenuti ad intervalli prestabiliti, sulla base di quanto si sarà programmato nella routine di ritardo, e quindi non ci sarà poi bisogno di premere il tasto RETURN per acquisire il valore di un nuovo dato.

Modificate il vostro programma in modo che utilizzi nella linea 40 una routine di ritardo di tempo invece del comando INPUT A$: definite un ritardo abbastanza lungo, di 2 o 3 secondi. Ecco un esempio di routine che può servire bene allo scopo:

```
40  FOR T = 0 TO 2000: NEXT T
```

Collegate la sonda logica all'uscita "A", pin 1, della porta NOR dell'SN7402. In seguito all'acquisizione del valore del dato dalla porta d'ingresso a tre stati lampeggerà il LED giallo del circuito di sonda logica; in tal modo saprete che un valore è stato acquisito.

Se poi desiderate modificare il programma in modo che acquisisca più di 10 punti, con la sola routine di visualizzazione potete acquisire fino a 39 valori.

Apportate al programma le modifiche necessarie per introdurre un ritardo di tempo con cui sincronizzare l'acquisizione dei dati dalla porta d'ingresso. Eseguite il programma. Volendo poi incrementare il ritardo per poter facilmente modificare gl'interruttori, otterrete un programma simile a questo:

```
 10  DIM A(10)
 20  PRINT "INIZIO"
 30  FOR P = 1 TO 10
 40  FOR T = 0 TO 2000: NEXT T
 50  A(P) = PEEK(49319)
 60  NEXT P
 70  PRINT "INIZIO VISUALIZZAZIONE ..."
 80  GR: COLOR = 5
 90  FOR P = 1 TO 10
100  D = A(P)/6.5
110  HLIN 0,D AT P
120  NEXT P
130  END
```

Avete notato che non tutti i valori provocano dei cambiamenti nella visualizzazione? Provate ad introdurre i valori 0, 1, 2, 3, e così via fino al 9. Può darsi che abbiate bisogno di aumentare il ritardo, o di reintrodurre il comando INPUT A$ nella linea 40, allo scopo di avere un tempo sufficiente per effettuare le modifiche degl'interruttori. Che cosa si verifica, a livello di visualizzazione, quando introducete questi numeri? E perché?

I valori dallo 0 al 6 fanno comparire lo stesso valore sullo schermo, e così pure i valori dal 7 al 9, ma con un "quadratino" più grande di quello relativo ai valori precedenti, dallo 0 al 6. Questo è dovuto al fatto che tutti i valori sono stati "compressi" nell'intervallo 0-39, per cui la risoluzione viene ridotta da un punto su 256 ad un punto su 40. Quindi, benché i dati abbiano 256 valori discontinui, l'immagine sullo schermo può contenere non più di 40 valori diversi. La divisione del valore per 6.5 lo "comprime" in modo da farlo rientrare nello spazio disponibile sullo schermo. Vedrete però anche che il valore zero farà "comparire" un quadratino sullo schermo del monitor. Purtroppo il programma in BASIC genererà un quadratino "luminoso" in corrispondenza del comando HLIN 0,0 riferito ad X, in qualunque punto dello schermo si trovi X.

Il punto principale di quest'esperimento è il fatto che il calcolatore può essere utilizzato per acquisire informazioni e visualizzarle o destinarle a diversi altri usi. Le porte d'ingresso e di uscita non sono altro che degli ulteriori mezzi per convogliare informazioni dentro il calcolatore, o fuori dal calcolatore.

Esperimento n. 11
UN ELEMENTARE CONVERTITORE ANALOGICO-DIGITALE

Scopo

Scopo di quest'esperimento è mostrare come un elementare convertitore analogico-digitale (DAC, oppure D/A) può essere interfacciato all'Apple.

Presentazione dell'esperimento

Interfacceremo all'Apple un semplice convertitore D/A, il convertitore ad 8 bits NE5018 della Signetics. Noi non abbiamo finora descritto i convertitori analogici, ma ne troverete una trattazione completa in *Microcomputer-Analog Converter Software and Hardware Interfacing*, edito dalla Howard W. Sams & Co., Inc., Indianapolis, IN 46268, al quale vi rimandiamo per ulteriori delucidazioni. Il libro tratta anche altri argomenti, come ad esempio gli amplificatori di sample and hold, i multiplatori (*multiplexers*) analogici e gli amplificatori da strumentazione.

Configurazione dei pins del circuito integrato (v. Figura 6.11)

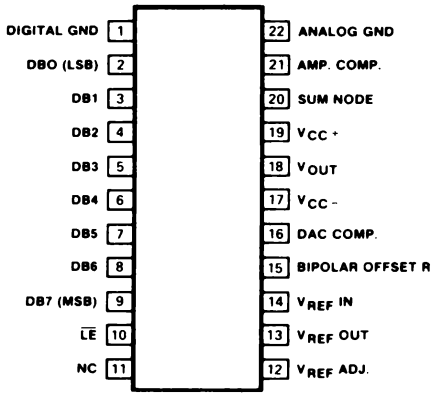


Figura 6.11 — Configurazione dei pins del chip convertitore D/A ad 8 bits NE5018 della Signetics.

Passo n. 1

Quest'esperimento richiede due ulteriori alimentazioni elettriche, +12 e −12 volts, destinate ad alimentare il circuito integrato del convertitore D/A. Prima di procedere accertatevi che tali alimentazioni siano disponibili, e che siano regolate alle tensioni esatte.

Cablate il circuito illustrato in Figura 6.12.

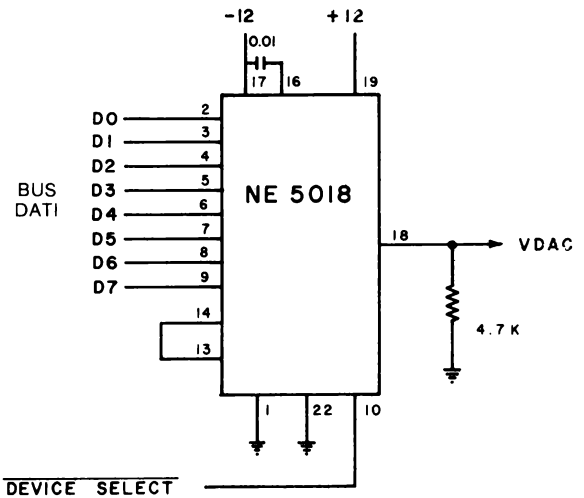


Figura 6.12 — Schema di un semplice interfacciamento con un convertitore D/A, utilizzando un chip convertitore D/A NE5018.

L'impulso di selezione dispositivo si ottiene dalla porta NOR dell'SN7402 del circuito che ci è servito negli esperimenti precedenti. Il segnale di selezione dispositivo è disponibile sul punto C (v. Figura 6.4), ma va invertito perché il chip NE5018 possa utilizzarlo. A questo scopo potrà servire un chip inverter SN7404 (v. Figura 6.13). Cablate anche questo circuito d'inversione, collegando l'ingresso dell'inverter SN7404 con il pin 13 dell'SN7402, e cablando l'uscita dell'inverter SN7404 all'ingresso $\overline{\text{DEVICE SELECT}}$ del convertitore NE5018.

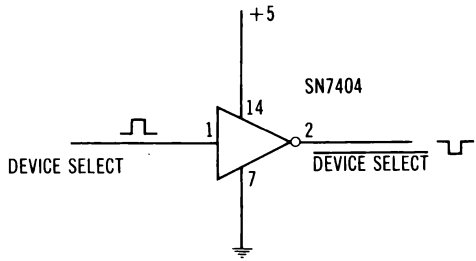


Figura 6.13 — Semplice circuito d'inversione dell'impulso di selezione dispositivo.

A questo punto controllate attentamente i collegamenti elettrici a +12 e a −12 volts per accertarvi che siano corretti. Se utilizzate alimentazioni elettriche distinte, accertatevi che il collegamento a massa sia a bassa resistenza e comune a tutte le alimentazioni ed alla scheda di prova.

Passo n. 2

Il convertitore D/A NE5018 converte valori compresi fra 0 e 255 in tensioni comprese fra 0 e +10 volts. Avendo diviso l'intervallo 0-10 volts in 256 valori, ovvero in 255 passi, il minimo incremento della tensione possibile è

10 volts/255 passi = 39 millivolts/passo

Scrivete un breve programma che incrementi un contatore di 8 bits, il cui valore sia quindi fornito in uscita al convertitore D/A. Non preoccupatevi del funzionamento interno del convertitore D/A, ma consideratelo come una porta di uscita. Il vostro programma darà luogo ad una rampa positiva di tensione, che cresce per piccoli incrementi. Scrivete il vostro programma.

Il nostro programma è il seguente:

```
10  FOR V = 0 TO 255
20  POKE 43918,V
30  NEXT V
40  GOTO 10
```

Per controllare l'andamento della tensione si può usare un voltmetro, o un volt-ohm-milliametro (vom). Collegate l'apparecchio fra la massa e l'uscita VDAC del chip NE5018 (VDAC è positiva). Provate il vostro programma. La tensione sale lentamente? Cosa accade quando la tensione raggiunge all'incirca +10 volts?

La tensione sale lentamente fino ad arrivare a +10 volts; quando arriva a questo valore, passa rapidamente a 0 volts, cioè a massa, e da qui riprende a salire lentamente.

Potete rallentare la rampa di tensione introducendo nel programma una breve routine di ritardo di tempo. Ecco quella che abbiamo usato noi:

```
25  FOR T = 0 TO 100: NEXT T
```

Passo n. 3

Scrivete un programma che generi una rampa negativa, ed uno che generi invece una rampa triangolare (la tensione prima sale, poi scende).

Ecco i nostri programmi:

Rampa negativa

```
10  FOR V = 255 TO 0 STEP -1
20  POKE 49318,V
30  NEXT V
40  GOTO 10
```

Uscita triangolare

```
10  FOR V = 0 TO 255
20  POKE 49318,V
30  NEXT V
40  FOR V = 254 TO 1 STEP -1
50  POKE 49318,V
60  NEXT V
70  GOTO 10
```

Provate entrambi questi programmi, o quelli scritti da voi. Perché l'intervallo relativo ad uno dei cicli contenuti nel programma della rampa triangolare è 254–1, e non 255–0?

Se l'intervallo fosse 255–0, questi due valori sarebbero emessi in uscita due volte, anche se voi probabilmente non potreste dire qual è la differenza sul voltmetro. Potrà essere utile inserire in questi programmi un ritardo, o dei ritardi di tempo.

Passo n. 4

Sapendo che la tensione da 0 a 10 volts corrisponde ad un certo numero di passi, e che precisamente si va da un minimo di 0 ad un massimo di 255, sapete scrivere un programma con cui sia possibile introdurre un valore di tensione da tastiera, e che quindi generi tale tensione sul voltmetro? Scrivete il programma.

Noi abbiamo scritto questo programma; provatelo.

```
10  INPUT "TENSIONE"; V
20  R = V*25.5
30  POKE 49318,R
40  GOTO 10
```

Passo n. 5

Provate ora il programma scritto da voi. La tensione generata dal convertitore D/A concorderà esattamente con quella che avete introdotto? Il nostro programma pare che vada bene, tenuto conto della scarsa precisione del voltmetro. Dal momento che il nostro programma non contiene statements di "rilevazione degli errori", si può provare a generare un segnale di +15 volts da parte del convertitore. Che cosa pensate che accadrà? Che il convertitore bruci?

Il convertitore non brucerà, perché può accettare soltanto un valore di 8 bits, che corrisponde ad un'uscita di +10 volts. L'ingresso "15" per 15 volts provocherà un messaggio di ILLEGAL QUANTITY ERROR (cioè ERRORE PER QUANTITÀ NON CONSENTITA), perché si sta tentando di trasferire il valore 382 ad un dispositivo ad 8 bits, e questo non si può fare con otto bits.

Passo n. 6

A questo punto dovreste saper scrivere un programma che vi permetta d'introdurre una tensione superiore ed una inferiore, in modo che l'Apple generi un'onda triangolare fra queste tensioni. Fate appello a tutte le vostre capacità di programmatori.

Questo è il programma che abbiamo usato noi:

```
10  INPUT "TENSIONE SUPERIORE"; H
20  IF H < = 10 AND H > = 0 THEN 40
30  PRINT "TENSIONE FUORI DEI LIMITI": GOTO 10
40  INPUT "TENSIONE INFERIORE"; L
50  IF L < = 10 AND L > = 0
60  PRINT "TENSIONE FUORI DEI LIMITI": GOTO 40
70  IF H > L THEN 90
```

```
 80  PRINT "LA V SUPERIORE DEVE ESSERE MAGGIORE"
 85  "DELLA V INFERIORE": GOTO 10
 90  H = H*25.5: L = L*25.5
100  FOR V = L TO H
110  POKE 49318,V
120  NEXT V
130  FOR V = H -1 TO L + 1 STEP -1
140  POKE 49318,V
150  NEXT V
160  GOTO 100
```

Eseguite il vostro programma e verificatelo. Fate in modo che l'ago del voltmetro "oscilli" fra la tensione superiore e quella inferiore. Se volete rallentare il movimento dell'ago del voltmetro per poterlo osservare agevolmente, ricorrete ad un ritardo o a ritardi di tempo.

Quest'esperimento mostra con chiarezza come interfacciare con il proprio calcolatore un semplice convertitore D/A. L'NE5018 ha dei latches interni, e gran parte della circuiteria analogica è stata collocata sul chip del convertitore. I convertitori D/A trovano impiego in applicazioni per le quali il calcolatore deve controllare dispositivi pilotati da tensione, come servomotori, amplificatori, etc.

Il convertitore D/A NE5018 non si userà più, per cui potete toglierlo dalla scheda di prova. Conservate il chip di porta NOR SN7402; togliete invece l'inverter SN7404. Potete staccare l'alimentazione. Togliete con attenzione i collegamenti con le alimentazioni a +12 ed a −12 volts, in modo che non vengano in contatto con nessuno dei circuiti.

Esperimento n. 12
PORTE DI USCITA, BCD E CODICI BINARI

Scopo

Scopo di quest'esperimento è studiare l'uso del chip SN74LS373 come porta di uscita.

Presentazione dell'esperimento

Disponiamo di circuiti integrati recenti, come il latch ottale SN74LS373, che semplificano la costruzione delle porte di uscita. In quest'esperimento costruirete una porta di uscita ad 8 bits servendovi di uno di questi chips, e studierete l'impiego dei numeri decimali codificati in binario.

Configurazione dei pins del circuito integrato (v. Figura 6.14)

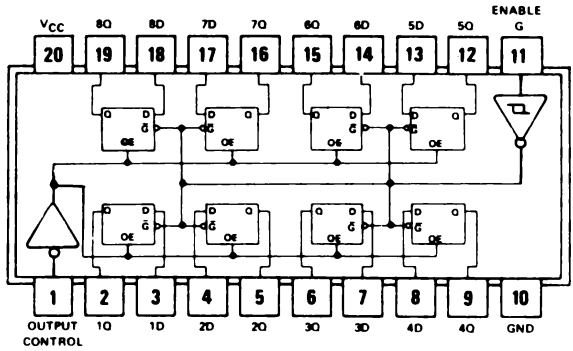


Figura 6.14 — Configurazione dei pins del chip di latch ottale SN74LS373.

Passo n. 1

Cablate il circuito rappresentato in Figura 6.15. Potete usare l'uscita "C", pin 13, della porta NOR del circuito rappresentato in Figura 6.4 come ingresso "G" del chip SN74LS373. Se la porta NOR di questo circuito non è stata cablata sulla vostra scheda di prova, per cablarla riferitevi all'Esperimento n. 2.

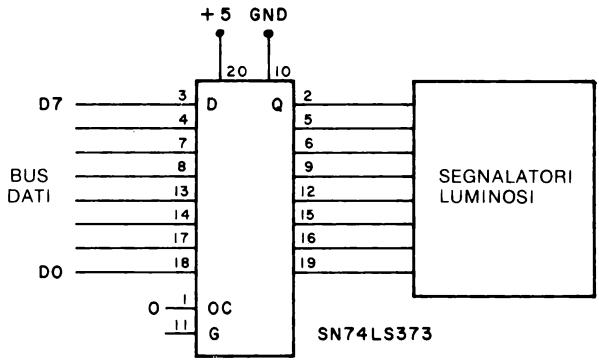


Figura 6.15 — Uso del chip di latch ottale SN74LS373 come porta di uscita.

Passo n. 2

Osservate che il chip SN74LS373 ha due ingressi di controllo, G ed OC. L'ingresso C controlla il latch, mentre l'ingresso OC controlla le uscite del latch, che sono a tre stati. Quindi il latch può essere usato non solo per avere informazioni da un bus, ma anche per passarle ad un altro bus. Le relazioni esistenti fra i segnali sono riportate in Tabella 6.3.

Controllo uscita	Abilitazione (G)	Dato	Uscita
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

Tabella 6.3 — Tabella della verità dei segnali di controllo relativa all'SN74LS373.

Quando il segnale di controllo uscite (OC) è un uno logico, le uscite sono abilitate, ovvero poste nello stato di alta impedenza (HI-Z). Quando l'ingresso di abilitazione, o di condizionamento (G), è un uno logico, l'informazione presente agl'ingressi D è trasferita, attraverso i circuiti di latch, alle uscite Q. Si tratta dello stesso tipo di procedimento che abbiamo visto per il chip di latch SN7475.

In quest'esperimento l'ingresso OC va posto a massa (zero logico) affinché le uscite siano costantemente abilitate.

Passo n. 3

Cablata la porta di uscita, per verificarla scrivete un breve programma che prenda i valori dalla tastiera e li visualizzi in binario sulla porta di uscita. La porta può essere verificata anche con un programma che realizzi un contatore binario che viene incrementato. Siete certamente capaci di scrivere questi programmi senza ulteriori suggerimenti.

Passo n. 4

Introducete nel calcolatore il programma che segue ed eseguitelo.

```
10  FOR C = 0 T0 255
20  POKE 49318,C
30  FOR T = 0 TO 500: NEXT T
40  NEXT C
50  GOTO 10
```

Che cosa accade sui LED?

Dovreste vedere un contatore binario che viene incrementato lentamente. Se volete, potete aumentare la durata del ritardo di tempo, riferendovi alla linea 30.

Mentre i LED visualizzano il contatore binario mentre è incrementato, togliete con attenzione il collegamento esistente fra il pin di OC, pin 1 del chip SN74LS373, e la massa. Che cosa accade sui LED? E che cosa, se ripristinate il collegamento?

Tutti i nostri LED si spengono quando si toglie il collegamento. Rimettendo a massa il pin di OC, il conteggio riprende. *Il segnale di controllo uscite (OC) non ha ripercussioni sul conteggio.* Anche se tutte le uscite fossero disabilitate e poste nello sta-

to di alta impedenza, il conteggio continuerebbe ed i latches sarebbero "aggiornati" con le nuove informazioni fornite dal calcolatore. Nel nostro calcolatore, lo stato di alta impedenza delle uscite fa sì che i LED si spengano. È possibile che con il vostro calcolatore si abbiano effetti diversi, ma comunque dovreste constatare un cambiamento evidente delle uscite del latch quando il pin d'ingresso OC non è a massa.

Il chip SN74LS373 è detto chip di otto latches a tre stati (*three state octal latch*) perché le otto funzioni di latch hanno uscite a tre stati. Questo chip è particolarmente utile nei circuiti d'interfaccia dei calcolatori, perché contiene tutti e otto i latches, e perché le sue uscite possono venir poste nello stato di alta impedenza. L'SN74LS373 può essere usato in interfacciamenti complessi, quali quelli in cui sono collegati i bus di differenti calcolatori. Infatti l'SN74LS373 potrebbe essere utilizzato come parte di un circuito di comunicazione collegante due o più calcolatori.

Passo n. 5

A questo punto avete un'altra porta d'ingresso cablata sulla scheda di prova; useremo questa porta per studiare qualche altra operazione che l'Apple può eseguire. Negli esempi precedenti il calcolatore ci è servito per controllare un contatore *binario* che veniva incrementato; ma questo non è l'unico codice utilizzato nelle apparecchiature elettroniche digitali: un altro codice molto usato è costituito dal formato decimale codificato in binario, nel quale a ciascuna delle cifre decimali è assegnato il relativo codice binario, indipendentemente dalle altre cifre. È chiaro che anche questo codice è "binario", nel senso che per ciascun bit sono possibili solamente due stati. Ad esempio, il numero decimale 9530 sarà, in decimale codificato binario, o BCD, 1001 0101 0011 0000. Notate le separazioni fra i gruppi di quattro bits, ciascuno dei quali rappresenta la cifra decimale relativa a ciascuna decade. Il codice BCD è usato in molti dispositivi elettronici, e per controllare visualizzatori a sette segmenti ed altri dispositivi che lavorano in decimale.

Provate ora a scrivere un programma che "spezzi" un numero nelle sue cifre equivalenti in BCD. Userete la porta di uscita per visualizzare i differenti gruppi, a due cifre BCD per volta. Sulla porta di uscita si dovranno visualizzare la prima volta le cifre BCD corrispondenti alle decine e alle unità, e poi quelle corrispondenti alle migliaia e alle centinaia. Per "fermare" il calcolatore fra una visualizzazione e l'altra, potete ricorrere al tasto RETURN, o ad un altro.

Questo è il nostro programma:

```
10  INPUT "VALORE"; A
20  IF A < 10000 THEN 30 ELSE 10
30  GOSUB 1000
40  POKE 49318,A + C
50  GET A$: A = B
60  GOSUB 1000
70  POKE 49318,A + C
80  GOTO 10
```

```
1000  B = 0: C = 0
1010  IF A > 99 THEN 1100
1020  IF A < 10 THEN RETURN
1030  C = C + 16: A = A - 10
1040  GOTO 1020
1100  A = A - 100: B = B + 1
1110  GOTO 1010
```

Nella subroutine, le variabili sono A, B e C, ove A rappresenta il valore decimale da convertire in BCD (cioè il valore iniziale), B le "centinaia" e C le "decine". Al termine della subroutine, A rappresenta le unità, cioè gli "uni". Volendo, potete però usare per le unità una variabile diversa.

Può capitare a volte che vi sia difficile ricordare che state forzando l'Apple a generare valori in BCD, perché quello che di fatto v'interessa sono i *codici binari* che devono essere emessi sulla porta di uscita. Quindi, anche se avete forzato l'Apple ad emettere la configurazione binaria 10011001, che rappresenta 99 in BCD, quello che effettivamente l'Apple crede di emettere sulla porta di uscita è il numero decimale 153, che è appunto il numero che fa comparire sui LED la configurazione binaria 10011001. Ci sono molti modi per "forzare" il calcolatore a lavorare con codici inaspettati, difformi cioè da quelli che usa normalmente.

Se avete l'intenzione di proseguire con i prossimi esperimenti, potete decidere di lasciare sulla scheda di prova la porta di uscita SN74LS373. Comunque, se sulla scheda di prova c'è già un'altra porta di uscita pronta da usare, il circuito SN74LS373 può essere rimosso. Togliete l'alimentazione.

Esperimento n. 13
CONTROLLO DI SEMAFORI CON LE PORTE DI USCITA

Scopo

Scopo di quest'esperimento è mostrare come si possa usare il calcolatore Apple come unità di controllo in un'applicazione concreta.

Presentazione dell'esperimento

Ci rendiamo conto che il controllo di un semaforo possa non sembrare un problema da affrontare con un calcolatore, ma è interessante perché mostra la capacità del calcolatore di prendere decisioni e controllare eventi esterni.

Passo n. 1

In quest'esperimento useremo una porta di uscita ad 8 bits. Se ne avete già una collegata con il calcolatore, potete usarla, a patto che controlli dei LED. Se avete portato a termine un esperimento sulle porte di uscita, usate un circuito con porta di uscita degli esperimenti precedenti. Se invece dovete costruire una porta di uscita, riferitevi all'Esperimento n. 8.

Userete dei segnalatori luminosi, o LED singoli, per simulare le luci del semaforo. Bastano sei LED, perché le luci nord e sud sono le medesime, e così pure le luci est ed ovest: per ciascuna direzione si avrà una luce rossa, una gialla ed una verde. Abbiamo utilizzato LED colorati, ricorrendo a questa convenzione:

Bit	LED		Bit	LED	
D0	Rosso	Via Bixio	D3	Rosso	Via Garibaldi
D1	Giallo		D4	Giallo	
D2	Verde		D5	Verde	

Passo n. 2

A questo punto determinate le configurazioni di uni e zeri necessarie per accendere e spegnere i singoli LED. Nel nostro circuito, abbiamo usato i chips di latch per pilotare i LED in maniera diretta; lo zero accende i LED, l'uno li spegne. Quali valori userete per accendere e spegnere i LED?

Noi abbiamo individuato a tal fine i valori binari che elenchiamo, con i valori decimali corrispondenti.

Via Bixio	Rosso	254	11111110	Via Garibaldi	Rosso	247	11110111
Via Bixio	Giallo	253	11111101	Via Garibaldi	Giallo	239	11101111
Via Bixio	Verde	251	11111011	Via Garibaldi	Verde	223	11011111

Passo n. 3

Come attività iniziale per il controllo del semaforo, scrivete un programma che faccia accendere il giallo verso Via Garibaldi, ed il rosso verso Via Bixio, lasciandoli rispettivamente acceso e spento per un secondo. Quale configurazione corrisponde alla condizione di "acceso", e quale alla condizione di "spento"?

La configurazione di "spento" è 255 (tutti uni logici), mentre nella configurazione di "acceso" i bits D4 e D0 sono zeri logici (238_{10}). Questo è il nostro programma:

```
10  POKE 49318,255
20  FOR T = 0 TO 770: NEXT T
30  POKE 49318,238
40  FOR T = 0 TO 770: NEXT T
50  GOTO 10
```

Passo n. 4

Determinate le configurazioni di luci necessarie per il funzionamento normale del semaforo. Quante ne servono? Quante sono? Come possono essere memorizzate nel calcolatore?

Esistono solo quattro configurazioni: a) rosso verso Via Bixio, verde verso Via Garibaldi (222); b) rosso verso Via Bixio, giallo verso Via Garibaldi (238); c) verde ver-

so Via Bixio, rosso verso Via Garibaldi (243); d) giallo verso Via Bixio, rosso verso Via Garibaldi (245). I valori si possono memorizzare mediante statements di DATA, variabili indicizzate, o anche semplici variabili, una per configurazione di luci.

Passo n. 5

Nel corso dell'esperimento assumeremo un "periodo di giallo" di due secondi. Quindi, se per Via Bixio è stato attivato un periodo di 10 secondi, il verde resterà acceso per 10 secondi, poi si avranno due secondi di giallo, e infine si accenderà il segnale rosso.

Scrivete un programma che vi permetta di vedere in sequenza le configurazioni di luci, con un periodo di 6 secondi verso Via Bixio ed uno di 10 secondi verso Via Garibaldi.

Questo è il nostro programma:

```
  10  M = 10: E = 6: P = 49318
  20  DATA 222, 238, 243, 245
  30  READ L
  40  POKE P,L
  50  FOR R = 1 TO M
  60  FOR T = 0 TO 770: NEXT T
  70  NEXT R
  80  READ L
  90  POKE P,L
 100  GOSUB 1000
 110  READ L
 120  POKE P,L
 130  FOR R = 1 TO E
 140  FOR T = 0 TO 770: NEXT T
 150  NEXT R
 160  READ L
 170  POKE P,L
 180  GOSUB 1000
 190  RESTORE
 200  GOTO 30

1000  FOR R = 1 TO 2
1010  FOR T = 0 TO 770: NEXT T
1020  NEXT R
1030  RETURN
```

Passo n. 6

Questo programma funziona bene, ma molti statements sono iterativi. Volete scrivere un nuovo programma semplificato? E come semplifichereste il programma?

Le uniche modifiche da apportare nelle quattro sezioni fondamentali del programma del passo n. 5 riguardano i ritardi di tempo e le configurazioni di luci. Ricorrendo ad un vettore di valori si può utilizzare un semplice ciclo. Il programma risponde bene allo scopo:

```
 10  A(1) = 222: A(2) = 238: A(3) = 243: A(4) = 245
 20  M(1) = 0: M(2) = 2: M(3) = 0: M(4) = 2
 30  INPUT "RITARDO SU VIA GARIBALDI"; M(1)
 40  INPUT "RITARDO SU VIA BIXIO"; M(3)
 50  FOR Q = 1 TO 4
 60  POKE 49318,A(Q)
 70  FOR R = 1 TO M(Q)
 80  FOR T = 0 TO 770: NEXT T
 90  NEXT R
100  NEXT Q
110  GOTO 50
```

In questo nuovo programma nel vettore A sono memorizzate le configurazioni di luci ed i ritardi di tempo. Ora ggiungeremo al programma di controllo del semaforo alcuni statements di controllo.

In Via Garibaldi il traffico è generalmente intenso, per cui il modo normale del semaforo sarà verde verso Via Garibaldi e rosso verso Via Bixio. Il programma dovrebbe essere in grado d'individuare un veicolo fermo al semaforo in Via Bixio, per potergli dare il verde. Comunque devono esserci almeno 30 secondi di "verde" verso Via Garibaldi prima che venga rilevato un veicolo in attesa in Via Bixio. In altre parole, un qualunque veicolo fermo al rosso in Via Bixio non farà scattare automaticamente il verde verso Via Bixio. Per avere funzionalità ancora più interessanti, mettiamo un sensore in Via Garibaldi: se cinque o più veicoli sono fermi al rosso in Via Garibaldi, scatterà il verde per loro, ed i veicoli transitanti in Via Bixio dovranno fermarsi.

Prima di scrivere il programma relativo, disegnate un semplice diagramma di flusso del problema. Per simulare i due sensori stradali, si può usare una porta d'ingresso, ma, se al suo posto userete la tastiera, imparerete qualcosa di più sul calcolatore Apple.

La tastiera utilizza per controllo due indirizzi di memoria: l'indirizzo 49152, che contiene i dati provenienti dalla tastiera, l'indirizzo 49168, con funzioni d'impulso di azzeramento flags.

Introducete nel calcolatore il programma che segue ed eseguitelo:

```
2000  PRINT PEEK(49152): GOTO 2000
```

Premete qualche tasto sulla tastiera e state attenti a quel che appare sul visualizzatore. Che cosa osservate?

Ogni volta che si preme un nuovo tasto, viene visualizzato un nuovo valore deci-

male, la cui visualizzazione continua finché non si preme un nuovo tasto. Pertanto l'informazione presente alla porta d'ingresso 49152 rappresenta il codice dell'*ultimo tasto* premuto.

Passo n. 8

Vogliamo ora che il calcolatore acquisisca un valore dalla porta d'ingresso corrispondente alla tastiera solo quando si sia premuto un tasto. A tal fine usate il bit di flag della tastiera, che è il bit D7 della porta d'ingresso 49152. Se questo bit è uno zero logico, tutti i valori provenienti da tale porta saranno minori di 128; se è un uno logico, i valori saranno uguali o maggiori di 128, fino a 255. Quindi, verificando il valore proveniente dalla porta d'ingresso, potete determinare se è stato premuto un tasto. Ovviamente, dopo aver "individuato" un tasto, dovete azzerare il bit di flag con un'operazione di lettura rivolta all'indirizzo 49168.

Introducete nel calcolatore ed eseguite il programma seguente:

```
2000  IF PEEK(49152) >= 128 THEN PRINT PEEK(49152)
2010  Z = PEEK(49168)
2020  GOTO 2000
```

Ora premete qualche tasto, uno per volta. Che cosa viene visualizzato? È visualizzato il codice decimale relativo a ciascun tasto via via che premete il tasto stesso?

Probabilmente avrete constatato che ogni tanto viene "saltato" un tasto: dal momento che il flag di tastiera viene azzerato in ciascun giro del ciclo, è possibile che l'Apple azzeri il flag di tastiera prima che venga rilevato. In realtà, voi desiderate che il flag venga azzerato solo *dopo* la rilevazione di un tasto! Vedremo come fare nel prossimo passaggio.

Passo n. 9

Scrivete un breve programma di controllo della tastiera che rilevi ogni tasto una volta soltanto, e ne stampi l'equivalente decimale.

Noi abbiamo utilizzato il programma che segue, che controlla in maniera continua la tastiera, ma stampa il carattere solo quando il flag è posizionato, e solo allora azzera il flag di tastiera.

```
2000  IF PEEK(49152) < 128 GOTO 2000
2010  PRINT PEEK(49152)
2020  Z = PEEK(49168)
2030  GOTO 2000
```

La variabile Z è una variabile fittizia, il cui unico ruolo è quello di permettere che il flag di tastiera venga azzerato con il comando PEEK(49168).

Se di un tasto volete utilizzare il valore decimale, senza il bit di flag, non dovete far altro che sottrarre 128.

Passo n. 10

Scrivete un programma di controllo del semaforo e verificatelo utilizzando il tasto "B" come sensore per Via Bixio ed il tasto "G" come sensore per via Garibaldi. Chiaramente dovrete determinare i codici corrispondenti ai tasti.

Noi abbiamo predisposto dei periodi di circa 10 secondi, a scopo di verifica, con periodi di giallo di 2 secondi. Questo è il nostro programma:

```
 10  A = 0: P = 49318
 20  POKE P,222
 30  FOR R = 0 TO 10
 40  FOR T = 0 TO 770: NEXT T
 50  NEXT R
 55  Z = PEEK(49168)
 60  IF PEEK(49152) = 194 GOTO 80
 70  GOTO 60
 80  Z = PEEK(49168): POKE P,238
 90  FOR R = 1 TO 2
100  FOR T = 0 TO 770: NEXT T
110  NEXT R
120  POKE P,243
130  FOR R = 0 TO 1000
150  IF PEEK(49152) = 199 THEN 190
170  NEXT R
180  GOTO 210
190  Z = PEEK(49168): A = A + 1
200  IF A < 5 THEN 170
210  POKE P,245
220  FOR R = 1 TO 2
230  FOR T = 0 TO 770: NEXT T
240  NEXT R
250  GOTO 10
```

Il flag di tastiera è stato azzerato prima di essere esaminato alla linea 60: quest'operazione annulla gli eventuali valori introdotti da tastiera durante il primo periodo di 10 secondi. Potete eliminare questo statement se desiderate che il sensore di Via Bixio "ricordi" tutti i veicoli che percorrono questa strada durante tale periodo.

Lo statement di rilevazione flag alla linea 150 è stato inserito nel ciclo di temporizzazione: questo vuol dire che il flag è verificato continuamente, e che questi statements di rilevazione flag vanno considerati all'interno dell'intero periodo di ritardo. Questo si può ottenere provando diversi valori della costante che esprime il ritardo nella linea 130.

Questo programma può fare molte altre cose; ad esempio, molti incroci hanno segnali di controllo per i pedoni, segnali di svolta a sinistra, luci lampeggianti, ed altri

dispositivi particolari. Potete rendere il programma complesso quanto volete. In queste condizioni la temporizzazione non è un punto particolarmente critico. Infatti non ha molta importanza se i veicoli devono aspettare qualche secondo in più, intanto che il flag viene verificato. Comunque è da tener presente che periodi di 10 o 20 secondi infastidirebbero gli automobilisti, e questo non va dimenticato nello scrivere il programma.

In alcuni casi le specifiche di tempo rientreranno in limiti tanto precisi, ed i periodi saranno tanto brevi, che sarà richiesta la programmazione in linguaggio assemblatore.

Togliete dalla scheda di prova i sei LED; conservate invece la porta di uscita, che servirà nel prossimo esperimento. Staccate pure l'alimentazione.

Esperimento n. 14
TESTER PER DISPOSITIVI LOGICI

Scopo

Scopo di quest'esperimento è mostrare come usare il calcolatore per provare un dispositivo elettronico.

Presentazione dell'esperimento

La maggior parte dei chips logici che contengono delle porte logiche può essere verificata applicando ai loro ingressi livelli logici noti, e confrontando poi le uscite con la tabella della verità relativa al dispositivo in prova. In quest'esperimento assegneremo questo compito al calcolatore. Sono necessarie una porta d'ingresso ed una di uscita. In questo modo possiamo verificare diversi dispositivi, come ad esempio l'SN7400, l'SN7402, l'SN7408. Si tratta qui di una verifica funzionale, non di una verifica delle proprietà dinamiche, come ad esempio il tempo di commutazione, il ritardo nella propagazione o altri parametri.

Passo n. 1

Per quest'esperimento dovete costruire una porta d'ingresso ed una di uscita, cosa che sapete certamente fare senza il nostro aiuto, dal momento che le indicazioni particolareggiate del caso vi sono già state fornite in molti degli esperimenti già fatti. Come porta d'ingresso potete usare un chip SN74LS373. Dopo aver costruito e verificato le porte, passate pure al passo seguente.

Passo n. 2

La Figura 6.16 riporta la configurazione di prova per un package di porte NAND, l'SN7400.

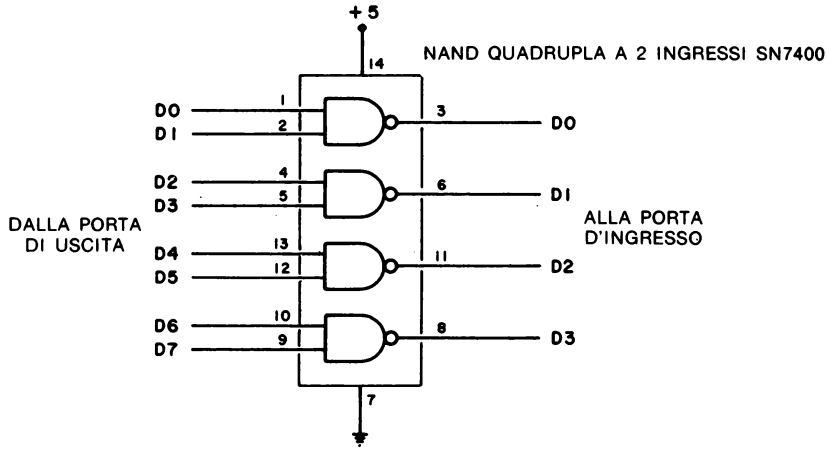


Figura 6.16 — Schema del circuito di prova delle porte NAND SN7400.

In Figura 6.17 invece troverete la configurazione dei pins di altri chips.

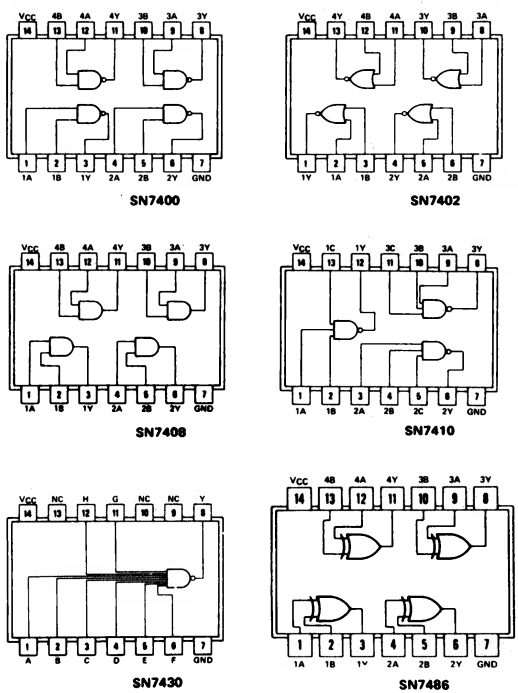


Figura 6.17 — Configurazione dei pins di alcune porte standard.

Cablate il circuito di prova secondo lo schema di Figura 6.16, non dimenticando di collegare gl'ingressi a +15 volts ed a massa sia ai chips d'interfaccia sia al circuito in prova. Mettete a massa gl'ingressi non utilizzati della porta d'ingresso.

Scrivete le tabelle della verità delle varie porte di Figura 6.17, a cominciare dalla porta NAND. Le porte a due ingressi ammettono solo quattro combinazioni di ingressi. Quante combinazioni ammetteranno quattro porte presenti in un unico package di circuito integrato?

Risponderete probabilmente che le combinazioni sono 16, cioè quattro per ciascuna delle quattro porte, oppure 256, che è il numero massimo che si può avere con otto ingressi binari. In realtà si hanno solo quattro combinazioni significative, perché tutte le porte vengono provate contemporaneamente. Il sapere che una porta non va bene per una particolare combinazione d'ingresso non serve a molto: se una porta non va bene, allora l'intero package viene scartato.

Passo n. 3

Quali sono le quattro combinazioni di otto bits che saranno utilizzate alla porta di uscita per provare la porta NAND? Scrivete questi numeri sia nella forma decimale che nella forma binaria.

Noi abbiamo ottenuto questi valori:

00	00	00	00	= 0
01	01	01	01	= 85
10	10	10	10	= 170
11	11	11	11	= 255

Avendo collegato le uscite con i bits d'ingresso D3-D0, ci aspetteremo valori di tutti uni o tutti zeri, cioè 0 o 15 a seconda della configurazione di prova. Per "eliminare" i bits che non servivano, li abbiamo collegati a massa. Quale valore leggeremo quando saranno acquisiti? E questo inciderà sul risultato? Volete proporre un altro sistema per "eliminare" questi bits dai dati di prova?

I bits entreranno come zeri logici, e non avranno conseguenze sui dati. Non collegandoli a massa, si possono mascherare con un'operazione di AND logico, e questo si può fare per mezzo della subroutine in linguaggio assemblatore.

Passo n. 4

Scrivete un breve programma che provi la porta NAND che avete interfacciato. Il programma sarà probabilmente molto simile al programma di controllo di un semaforo dell'Esperimento n. 13, passo n. 6. Il programma non dev'essere troppo complesso.

Il programma che segue è andato abbastanza bene per quest'applicazione:

```
10  T(1) = 0: T(2) = 85: T(3) = 170: T(4) = 255
20  R(1) = 15: R(2) = 15: R(3) = 15: R(4) = 0
30  FOR S = 1 TO 4
40  POKE 49318,T(S)
50  IF PEEK(49319) < > R(S) THEN 100
60  NEXT S
70  PRINT "PROVA OK": END
100 PRINT "GUASTO": END
```

Passo n. 5

Dal momento che le configurazioni dei pins dell'SN7400, dell'SN7408 e dell'SN7486 sono equivalenti, cioè, in altre parole, ingressi ed uscite occupano le stesse posizioni sul chip, è possibile scrivere un programma di prova generalizzato che vada bene per tutti e tre? E come?

Sì, possiamo scrivere un programma di prova generalizzato con il quale l'utilizzatore introdurrà il nome del dispositivo, mentre il calcolatore stabilirà le appropriate informazioni, relative alle tabelle della verità, da utilizzare nel corso della prova. Le tabelle della verità sono in Tabella 6.4.

SN7400			SN7408			SN7486		
A	**B**	**USCITA**	**A**	**B**	**USCITA**	**A**	**B**	**USCITA**
0	0	1	0	0	0	0	0	0
0	1	1	0	1	0	0	1	1
1	0	1	1	0	0	1	0	1
1	1	0	1	1	1	1	1	0

Tabella 6.4 — Tabelle della verità relative alle porte NAND, AND ed EXOR.

Osservate che le configurazioni di prova sono sempre le stesse; cambiano solo i risultati.

Questo è il programma di prova che noi abbiamo utilizzato:

```
10  INPUT "LE ULTIME DUE CIFRE"; A$
20  IF A$ = "00" THEN 200
30  IF A$ = "08" THEN 300
40  IF A$ = "86" THEN 400
50  PRINT "PROVA NON DISPONIBILE": GOTO 10
60  T(1) = 0: T(2) = 85: T(3) = 170: T(4) = 255
70  FOR S = 1 TO 4
80  POKE 49318,T(S)
```

```
 90  IF PEEK(49319) < > R(S) THEN 120
100  NEXT S
110  PRINT "PROVA SN74"; A$; "OK": END
120  PRINT "GUASTO": END
200  R(1) = 15: R(2) = 15: R(3) = 15: R(4) = 0
210  GOTO 60
300  R(1) = 0: R(2) = 0: R(3) = 0: R(4) = 15
310  GOTO 60
400  R(1) = 0: R(2) = 15: R(3) = 15: R(4) = 0
410  GOTO 60
```

Le ultime due cifre richieste dal programma sono le ultime due cifre del codice del dispositivo, cioé 00 per l'SN7400, 08 per l'SN7408, e cosi via. Se poi si hanno parecchi chips SN7400, SN7408 o SN7486, anche in questo caso è possibile verificare i singoli dispositivi. Se volete controllare l'interfacciamento ed il vostro programma, potete eliminare un collegamento d'ingresso o di uscita, simulando cosi un'anomalia.

Passo n. 6

Il calcolatore è anche in grado di provare dispositivi logici come flip-flops e contatori. Se conoscete il contatore binario a 4 bits SN7493, potete eseguire i passi che seguono; in caso contrario, varrà comunque la pena di leggerli.

In Figura 6.18 diamo la configurazione dei pins e lo schema del contatore SN7493. Per provare questo dispositivo, il calcolatore deve poter accedere alle uscite del contatore, nonché essere in grado di azzerarlo e d'inviare gl'impulsi di clock al chip contatore. Non cercheremo di provare il contatore in maniera completa, limitandoci a provare la capacità di azzeramento del contatore e la funzione di conteggio.

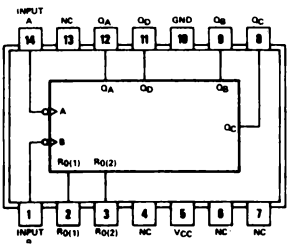


Figura 6.18 — Configurazione dei pins del contatore a 4 bits SN7493.

Passo n. 7

Cablate il contatore SN7493 riferendovi alla Figura 6.19: vi serviranno la porta d'ingresso e la porta di uscita definite nei precedenti passi dell'esperimento. Avrete pure bisogno di due porte NOR, come si desume sempre dalla Figura 6.19: queste

porte saranno fornite da un unico chip SN7402. Non usate un contatore SN74L93 al posto dell'SN7493, e ricordatevi di collegare a massa gl'ingressi non utilizzati della porta d'ingresso.

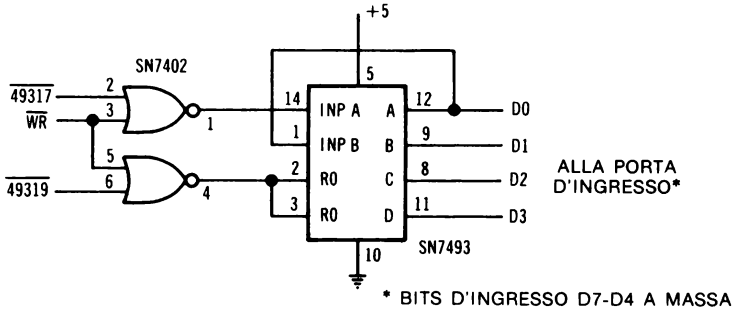


Figura 6.19 — Schema del circuito di prova usato per verificare il chip contatore SN7493.

Passo n. 8

Scrivete un breve programma di prova che attivi la funzione di azzeramento del contatore, ed uno che provi la capacità del calcolatore d'inviare impulsi di clock al contatore e d'incrementare di uno il suo conteggio.

Noi abbiamo utilizzato questo programma:

```
  10  POKE 49318,0
  20  IF PEEK(49319) > 0 THEN 1000
  30  PRINT "PROVA AZZERAMENTO OK"
  40  FOR C = 1 TO 15
  50  POKE 49317,0
  60  IF PEEK(49319) < > C THEN 1010
  70  NEXT C
  80  PRINT "PROVA CONTEGGIO OK": END
1000  PRINT "AZZERAMENTO GUASTO": END
1010  PRINT "CONTEGGIO GUASTO IN"; C: END
```

Il programma per prima cosa fa la prova di azzeramento, e poi dà il via alle prove necessarie per verificare la capacità del chip d'incrementare il suo conteggio di uno per ciascun impulso ricevuto sul pin INP A.

Passo n. 9

Questo programma non prova tutti i 16 stati del contatore; trascura cioè l'ultimo conteggio, che va da 1111 a 0000. Volete modificarlo in modo che verifichi anche questo?

Non dovrebbe esservi difficile aggiungere al programma la prova finale; potete seguire diverse strade; ad esempio,

```
 90  POKE 49317,0
100  IF PEEK(49319) <> 0 THEN 1010
110  PRINT "PROVA CONTEGGIO OK": END
```

Con questo statement il programma dà luogo al conteggio finale, e prova il conteggio da 1111 a 0000, col quale il contatore supera la propria capacità di conteggio.

La porta di uscita non servirà più, quindi potete toglierla dalla scheda di prova. La porta d'ingresso invece sarà ancora usata. Non così il programma, per cui potete togliere l'alimentazione.

Esperimento n. 15
CIRCUITI DI FLAG ELEMENTARI

Scopo

Scopo di quest'esperimento è illustrare la costruzione e l'impiego di circuiti di flag elementari.

Presentazione dell'esperimento

I flags sono segnali che il calcolatore ed i dispositivi di I/U usano per sincronizzare le loro operazioni. I flags servono solitamente ad indicare una condizione fra due possibili: libero/occupato, pieno/vuoto, caldo/freddo, e altre combinazioni inerenti alle condizioni di un'interfaccia nei confronti del calcolatore. L'Esperimento n. 6 illustrava l'uso delle porte d'ingresso nel trasferimento d'informazioni non numeriche al calcolatore; quest'esperimento approfondisce l'argomento. È necessaria una porta d'ingresso ad 8 bits.

Configurazione dei pins del circuito integrato (v. Figura 6.20)

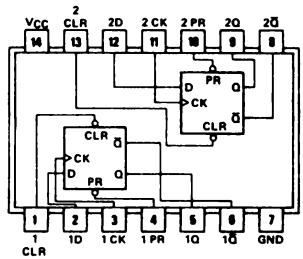


Figura 6.20 — Configurazione dei pins del doppio flip-flop di tipo D SN7474.

Passo n. 1

Quest'esperimento richiede una porta d'ingresso, e certamente sapete costruire un circuito con una porta d'ingresso senza il nostro aiuto: in molti degli esperimenti precedenti si spiega come costruire queste porte, e vi consigliamo pertanto di ricorrere ad uno di questi circuiti. Cablata e provata la porta d'ingresso, procedete con il prossimo passo.

Passo n. 2

Uno dei precedenti esperimenti illustrava l'uso d'interruttori semplici come sensori, ovvero ingressi di flag. In quest'esperimento useremo circuiti di flip-flop al posto degl'interruttori meccanici o di ponticelli. Cablate il circuito di Figura 6.21.

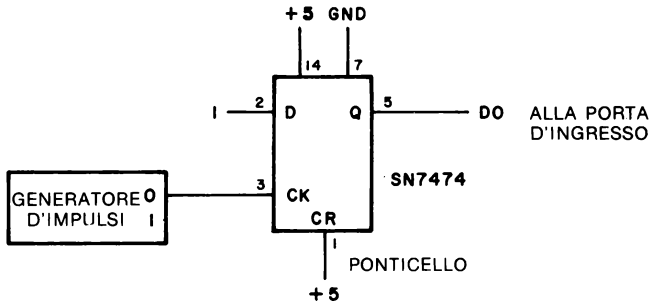


Figura 6.21 — Semplice circuito di flag basato su un flip-flop.

Utilizzate un ponticello per collegare fra loro +5 volts e l'ingresso di azzeramento, al pin 1; in tal modo potrete azzerare il flag spostando il ponticello da +5 volts a massa, e poi ancora a +5 volts. Il circuito generatore d'impulsi può essere costituito da una coppia di porte NAND accoppiate in modo incrociato, oppure da un circuito equivalente che generi delle transizioni logiche di "azzeramento" esenti da disturbi. Questo tipo di funzione è descritto in appendice.

Passo n. 3

Come pensate di programmare il calcolatore in modo da acquisire in maniera continua sul bit D0 della porta d'ingresso lo stato logico del flag? Supponete che le condizioni possibili siano due: a) gli altri bits sono a massa (zero logico); b) gli altri bits possono essere utilizzati per altri ingressi di flag.

Se gli altri bits sono a massa, il valore proveniente dalla porta d'ingresso sarà zero quando il flag è azzerato, diverso da zero quando è posizionato ad uno logico. Se invece gli altri bits sono utilizzati per degl'ingressi di flag, i bits che "non servono" devono essere mascherati. Alla mascheratura provvede un'operazione di AND logico, attuata da una subroutine in linguaggio assemblatore.

Passo n. 4

In quest'ultimo caso introdurrete il programma in linguaggio assemblatore che esegue l'operazione di AND su due bytes di dato. Nell'introdurre il programma attenetevi a questa sequenza:

1) Premere il tasto RESET, battere CALL – 151 e premere il tasto RETURN. L'Apple risponderà con un asterisco (*), quando si trova in modo Monitor.
2) Battere 0300: 00 00 00 48 AD 00 03 2D 01 03 8D 02 03 68 60, lasciando uno spazio fra i gruppi di due cifre, così come si vede qui. Utilizzare 00 per i primi tre valori del programma.
3) Premere il tasto RETURN, battere 02FF e premere per tre volte il tasto RETURN. A questo punto controllare i dati comparsi sul visualizzatore con le informazioni che si sono introdotte.

Per provare questa routine in linguaggio assemblatore, potete servirvi del programma che segue. Poiché l'operazione di AND impiega numeri binari, dovrete convertire i numeri usati nella prova in binario, per poter controllare i risultati.

```
1000  POKE 10,76: POKE 11,03: POKE 12,03
1010  INPUT "BYTE MASCHERA"; M: POKE 768,M
1020  INPUT "BYTE DATO"; D: POKE 769,D
1030  Q = USR(0): PRINT "RISULTATO"; PEEK(770)
1040  GOTO 1010
```

Se il programma non dà i risultati appropriati, riattivate il modo Monitor e controllate i bytes di dato che avete introdotto.

I comandi di POKE della linea 1000 hanno il compito di scrivere i bytes d'indirizzo del puntatore in modo che il comando di USR possa "localizzare" la subroutine in linguaggio assemblatore da voi introdotta. Per ulteriori chiarimenti sull'uso di questo tipo di subroutine in linguaggio assemblatore vi rimandiamo al Capitolo 4 ed all'Esperimento n. 7.

Passo n. 5

Introdotto il programma in linguaggio assemblatore che eseguirà l'operazione di AND fra due bytes, dando un risultato di 8 bits, utilizzatelo per provare il bit di flag. Che cosa userete come byte maschera?

Dal momento che il flag è stato introdotto nel calcolatore sul bit D0, solo il bit meno significativo (l'LSB) sarà "posizionato" ad uno, per cui la maschera sarà costituita da 00000001_2, ossia 1_{10}. Il byte maschera si troverà all'indirizzo 768, come potrete constatare dal programma di prova del prossimo passo.

Passo n. 6

Scrivete un breve programma utilizzabile per provare il circuito di flag a flip-flop. L'Apple dovrà stampare uno "0" se il flag è azzerato, un "1" se è posizionato ad uno. Potete azzerare manualmente il flag spostando il ponticello che collega il pin 1 del flip-flop a +5 volts in modo che il pin 1 sia momentaneamente collegato a massa.

Questo è il programma che noi abbiamo usato:

```
10  POKE 10,76: POKE 11,3: POKE 12,3
20  POKE 768,1
30  POKE 769,PEEK(49319)
40  Z = USR(0)
50  IF PEEK(770) = 0 THEN 80
60  PRINT "1"
70  GOTO 30
80  PRINT "0"
90  GOTO 30
```

Ci è parso che questo programma andasse ottimamente. È possibile "invertire" il programma in modo che venga rilevato uno 0 logico come condizione di "ON", ed un uno logico come condizione di "OFF"?

Si. Basta scambiare i comandi delle linee 60 ed 80. È molto facile "invertire" in un programma il senso di un flag.

Passo n. 7

Eseguiremo qui un breve programma che conti il numero delle volte in cui il flag è posizionato ad uno, servendoci ancora della subroutine in linguaggio assemblatore. Se lo desiderate, potete aggiungere un altro circuito generatore d'impulsi, che provveda all'operazione di azzeramento del flag al posto del ponticello che collega il pin 1 dell'SN7474 a +15 volts.

Introducete ed eseguite il programma seguente:

```
10  POKE 10,76: POKE 11,3: POKE 12,3
20  POKE 768,1
30  HOME: C = 0
40  POKE 769,PEEK(49319)
50  Z = USR(0)
60  IF PEEK(770) = 0 THEN 40
70  C = C + 1: VTAB 1; PRINT C
80  GOTO 40
```

Prima di provare il programma accertatevi che il flip-flop sia azzerato. In fase di esecuzione, attivate il generatore d'impulsi e posizionate il flip-flop ad uno. Che cosa accade? È quello che pensavate?

Nella prova fatta da noi il conteggio è cominciato non appena posizionato ad uno il flip-flop, ed è proseguito per tutto il tempo che il flag è rimasto in tale stato. Azzerando il flip-flop, il conteggio si è fermato. Quello che invece noi volevamo era che si contassero le volte in cui il flip-flop era posizionato ad uno. Perché non si hanno risultati rispondenti a quanto ci si attendeva?

Lo stato di posizionamento del flip-flop ha continuato ad essere verificato e rilevato dal programma: non potremmo azzerare manualmente il flip-flop abbastanza rapidamente da fermare il conteggio dopo ogni impulso di attivazione del conteggio.

Passo n. 8

Nella maggior parte dei sistemi, il calcolatore stesso, o il dispositivo che contiene il flag, azzera il flag dopo che questo è stato rilevato. Per far sì che la vostra interfaccia azzeri il flip-flop, aggiungete il circuito rappresentato in Figura 6.22. Vi servirà un chip di NOR SN7402: accertatevi di cablare il pin 14 all'alimentazione a +5 volts ed il pin 7 del chip SN7402 alla massa. Dal momento che il circuito di NOR fornirà il segnale di azzeramento del flip-flop, badate bene a togliere il filo che è servito per collegare +5 volts al pin 1 del flip-flop SN7474.

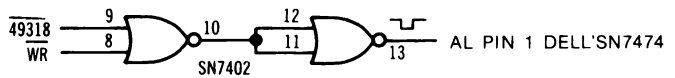


Figura 6.22 — Schema di un semplice circuito di azzeramento flag.

Il circuito di Figura 6.22 vi permetterà di azzerare il flip-flop con il comando POKE 49318.

Modificate il vostro programma aggiungendo la linea 65:

```
65  POKE 49318,0
```

Quando questo comando viene eseguito, il flag è azzerato. Dal momento che non potete conoscere lo stato del flag nel momento in cui il programma inizia, potreste allora, se volete, inserire un comando di azzeramento flag anche all'inizio del programma. Adesso eseguite il programma. Non appena rilevato, il flag è immediatamente azzerato; poi il contatore viene incrementato e visualizzato.

Uno dei vantaggi dell'uso di questo tipo di flag, e della routine in linguaggio as-

semblatore nel controllo del flag stesso, è che il calcolatore non si deve bloccare nell'attesa del flag, a meno che non lo vogliate voi. Allora potete scrivere un programma di controllo flag in questo modo: se il flag non è presente, il calcolatore passa ad un'altra operazione; se invece è attivo, viene servito il dispositivo ad esso associato, e poi il calcolatore prosegue.

L'interprete BASIC dell'Apple prevede un comando per il controllo flag che prende il nome di WAIT. Questo comando è utilizzabile per verificare lo stato di un flag, ma, se il flag non è trovato "vero", il programma continua ad aspettare, senza poter fare nient'altro. Se un programma si arresta nell'attesa che l'evento segnalato dal flag avvenga e questo non accade mai, premete il tasto RESET per riprendere il controllo dell'Apple. Per maggiori delucidazioni sul comando di WAIT vi consigliamo la lettura del *BASIC Programming Reference Manual* dell'Apple. Questo comando non comporta, quando è eseguito, l'azzeramento del flag.

Esperimento n. 16
CONVERTITORE ANALOGICO-DIGITALE ELEMENTARE

Scopo

In quest'esperimento interfaccerete al calcolatore un convertitore analogico-digitale ad 8 bits. Realizzeremo diversi tipi di misure.

Presentazione dell'esperimento

Con i calcolatori si possono realizzare molte applicazioni relative ai convertitori analogico-digitali, o convertitori A/D. I convertitori A/D permettono al calcolatore di misurare tensioni analogiche, come quelle che derivano da tutta una serie di sorgenti di segnali e di trasduttori. In quest'esperimento useremo un semplice convertitore A/D ad 8 bits, e precisamente un ADC0804 della National Semiconductor. Questo convertitore ha delle uscite a tre stati, per cui può essere interfacciato direttamente, e senza difficoltà, con il bus dati di un microcalcolatore. D'altra parte le uscite a tre stati hanno un tempo di accesso che può arrivare ai 200 nanosecondi, per cui, se provate ad usare il convertitore A/D ADC0804 sulla vostra scheda di prova, constaterete che il tempo in più necessario per attivare la circuiteria di accoppiamento del bus, con cui la direzione del bus dati è invertita in ingresso, sarà eccessivamente lungo. Il calcolatore non riuscirà a "prendere" i dati provenienti dal convertitore.

Per eseguire l'esperimento bisogna accedere al "puro e semplice" bus dati dell'Apple, come è spiegato nei prossimi passi.

Configurazione dei pins del circuito integrato (v. Figura 6.23)

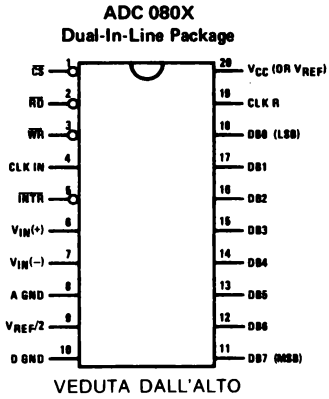


Figura 6.23 — Configurazione dei pins del convertitore A/D ADC0804.

Passo n. 1

In quest'esperimento interfacceremo il convertitore A/D ADC0804 direttamente al bus dati così come questo arriva dall'Apple. Togliete quindi accuratamente dalla scheda di prova nella sezione d'interfaccia i due chips di buffer bus 8216 posti in IC-10 ed IC-11.

Passo n. 2

Cablate il circuito integrato ADC0804 riferendovi alla Figura 6.24. Le linee del bus dati sono portate nei corrispondenti fori negli zoccoli posti in IC-10 ed IC-11. Se i fili

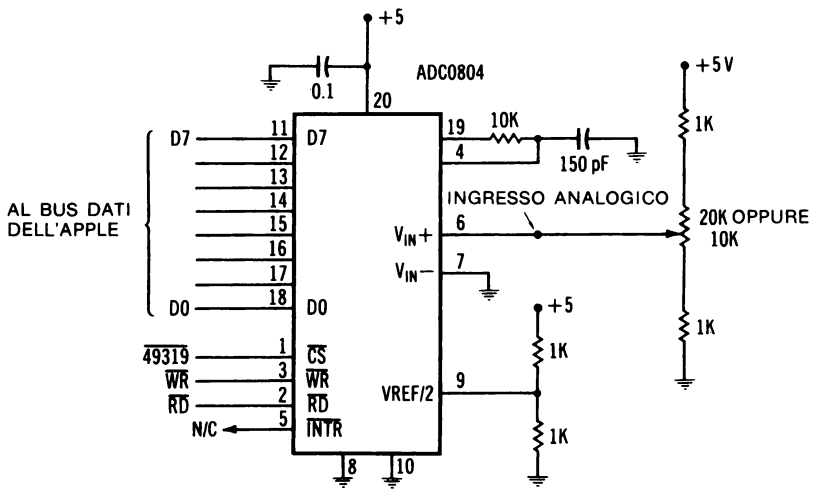


Figura 6.24 — Schema del circuito d'interfaccia ADC0804.

non entrano con facilità nei fori, vi suggeriamo di sistemare in IC-10 ed IC-11 degli zoccoli a 16 pins con fori più larghi; in tal modo potrete realizzare i collegamenti senza dover far entrare a forza i fili in fori troppo stretti. Infatti i fili devono entrare nei fori corrispondenti con uno sforzo minimo; spingendo troppo, si rischia di piegare i contatti dello zoccolo, che così non realizzeranno esattamente il contatto con il chip 8216 una volta che questi sono reinseriti nei rispettivi zoccoli.

Passo n. 3

Introducete nel calcolatore ed eseguite il programma seguente:

```
10  POKE 49319,0
20  FOR T = 0 TO 100: NEXT T
30  PRINT PEEK(49319)
40  GOTO 10
```

Che cosa fa il programma? Che cosa appare sullo schermo?

Il programma mette alla prova il convertitore A/D: avvia una conversione, esegue un ritardo di tempo durante il quale è realizzata la conversione, e infine legge e visualizza i dati. Per accertarvi che il convertitore sia in funzione regolate lentamente il potenziometro man mano che osservate i dati.

Variando la tensione fornita dal potenziometro, dovreste osservare una variazione corrispondente del valore visualizzato dall'Apple. Qual è il valore minimo? E il valore massimo? E corrispondono a quello che prevedevate?

Il valore minimo dovrebbe essere compreso fra 0 e 1; il valore massimo dovrebbe essere compreso fra 253 e 255. Ed è quello che ci si può aspettare da un dispositivo ad 8 bits, perché questi dispositivi possono dar luogo soltanto a valori compresi fra 0 e 255.

Passo n. 4

Il chip ADC0804 ha un'uscita di flag che può essere usata per conoscere lo stato del convertitore, cioè occupato o libero. Quest'uscita è a zero logico quando il dato è disponibile per il calcolatore, ad uno logico quando il convertitore sta effettuando una conversione. Si tratta veramente dell'uscita di un circuito di flag: il flag è azzerato quando gli otto bits che costituiscono il dato vengono letti dal calcolatore. Dal momento che il convertitore può eseguire molte migliaia di conversioni al secondo, è veramente necessario verificare lo stato del segnale di flag?

Generalmente no, perché il convertitore porta a termine il processo di conversione prima che al dato possa accedere un programma in BASIC.

Suggerite alcuni possibili usi per l'uscita di flag.

Il flag si può usare per controllare con un programma in linguaggio assemblatore un convertitore A/D. Nei programmi in linguaggio assemblatore il flag può essere testato come ingresso ad una porta d'ingresso, oppure usato come segnale d'interruzione del microprocessore 6502. Si tratta di applicazioni in cui la velocità è elevata, per cui potrà essere utile controllare il flag allo scopo di determinare il momento in cui il convertitore ha portato a termine una conversione.

Passo n. 5

Togliete la linea 20 ed eseguite il programma. Quali risultati si hanno?

I valori dei dati sono gli stessi di quelli che si sono avuti quando il programma comprendeva gli statements di ritardo di tempo. Quindi il convertitore "è più veloce" del programma di controllo in BASIC.

Passo n. 6

I valori visualizzati sullo schermo non rappresentano la tensione che è effettivamente misurata, ma costituiscono una rappresentazione binaria ad 8 bits. Scrivete un programma che realizzi la conversione alle tensioni reali, oppure, se volete, aggiungete gli statements opportuni al programma che state già utilizzando.

Noi abbiamo utilizzato gli statements che seguono, che si limitano ad eseguire una conversione matematica del valore decimale compreso fra 0 e 255 nella tensione corrispondente, compresa fra 0 e +5 volts.

```
10  POKE 49319,0
20  FOR T = 0 TO 100: NEXT T
30  PRINT (PEEK(49319)*5/255)
40  GOTO 10
```

Provate il vostro programma, o il nostro. Funziona?

Dovrebbe. Il calcolatore stamperà un gran numero di cifre decimali, forse troppe, perché il convertitore ha una precisione massima di uno su 256, cioè dello 0,25% circa. Purtroppo con l'Apple l'operazione di arrotondamento non è cosa da poco. Potete o eseguire un arrotondamento matematico, o, con un'operazione su stringhe, stampare dopo il punto decimale soltanto un numero prestabilito di cifre. Se volete, potete ricorrere a questi statements:

```
30  A$ = STR$(PEEK(49319)*5/255)
40  PRINT LEFT$(A$,4)
50  GOTO 10
```

Non dimenticate che questa routine non fa altro che *limitare* il valore visualizzato a quattro cifre decimali: questo è fatto senza ricorrere ad alcun arrotondamento.

Passo n. 7

Provate a scrivere una routine che sfrutti la capacità grafica ad alta risoluzione dell'Apple, in modo che il programma disegni i valori della tensione in funzione del tempo. Le misure dovrebbero essere effettuate ad intervalli regolari (routine di ritardo di tempo), e si dovrebbe disegnare una linea continua. Chi non conosca i formati ed i comandi della grafica ad alta risoluzione utilizzi il programma seguente:

```
100  HRG: COLOR = 3: Y1 = 0
110  FOR X = 0 TO 249
120  POKE 49319,0
130  Y2 = PEEK(49319)/1.594
140  HPLOT X,Y1 TO X + 1,Y2
150  Y1 = Y2
160  NEXT X
170  END
```

Provate il programma. Via via che l'esecuzione procede, variate il posizionamento del potenziometro. In corrispondenza delle variazioni dovrà apparire un tracciato; una variazione costante darà luogo ad una linea orizzontale sullo schermo.

Passo n. 8

Siete in grado di proporre un esperimento semplice che illustri l'uso del convertitore A/D e del programma grafico?

Ci sono parecchi semplici esperimenti che si potrebbero tentare, tutti implicanti la misura di una tensione proporzionale alla misura di una grandezza fisica, che viene così effettuata. Ad esempio, potreste misurare la tensione di una cellula fotoelettrica al variare delle condizioni d'illuminazione, la tensione di un condensatore in carica, o ancora una tensione proporzionale alla temperatura.

Cablate il circuito di Figura 6.25. Qui userete il convertitore A/D ed il calcolatore per misurare la tensione di carica di un grosso condensatore elettrolitico.

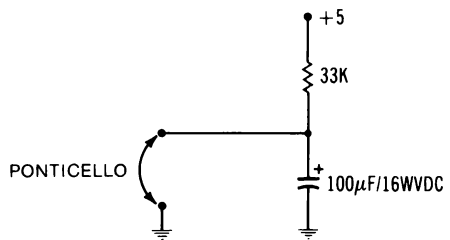


Figura 6.25 — Schema del circuito di carica di un condensatore.

Con il ponticello scaricate il condensatore, e lasciatelo fino al momento di far partire il programma. Una volta iniziata l'esecuzione, togliete il ponticello collegandolo

a massa. La tensione dovrebbe salire lentamente, man mano che il condensatore viene caricato. Per quale ragione nel grafico il punto corrispondente a tensione zero appare nella parte alta dello schermo, e quello corrispondente alla tensione massima appare nella parte bassa?

I grafi realizzati dal calcolatore procedono dall'alto verso il basso dello schermo man mano che i valori salgono, per cui, se volete invertire l'immagine visualizzata, dovrete "invertire" i valori che arrivano dal convertitore. Questo vuol dire che dovrete convertire lo zero in 159 ed il 159 in zero. A tal fine basterà modificare come segue la linea 130:

```
130  Y2 = 160 - (PEEK(49319)/1.594)
```

Passo n. 9

Il convertitore A/D può servire anche per misurare delle temperature. Userete un sensore di temperatura quale l'LM355 per generare una tensione proporzionale alla temperatura, essendo la costante di proporzionalità uguale a 10 mV/K. La scala Kelvin delle temperature utilizza la stessa suddivisione in gradi della scala Celsius, solo che 0°C = 273 K. Quindi la temperatura di una stanza di 20°C sarà equivalente a 293 K, e l'LM355 genererà come uscita 293 x 10 mV, cioè 2.93 volts.

Per misurare la temperatura, cablate il circuito rappresentato in Figura 6.26, accertandovi che il potenziometro, o il circuito di carica del condensatore, non sia collegato con l'ingresso del convertitore A/D contemporaneamente al sensore di temperatura.

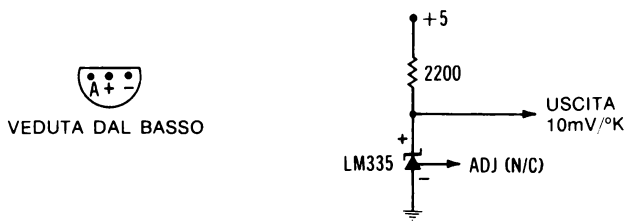


Figura 6.26 — Schema del circuito di misura della temperatura e configurazione dei pins del chip LM335.

Potete utilizzare lo stesso programma di visualizzazione grafica del passo precedente, inserendo però, se volete, la linea 155 per avere un ritardo di tempo:

```
155  FOR T = 0 TO 100: NEXT T
```

In tal modo la visualizzazione sarà ritardata, perché le variazioni di temperatura

saranno più lente delle variazioni di tensione che si verificano quando si carica il condensatore.

Eseguite il programma. Scaldate il sensore con le dita: si ha qualche variazione? Che cosa vi aspettate di vedere?

Probabilmente non osserverete una variazione apprezzabile, perché il visualizzatore è predisposto per un intervallo che va da 0 a 500 K, corrispondente, a livello di sensore, ad un divario da 0 a +5 volts. Se nel grafo visualizzato c'è un incremento di diversi "punti", vuol dire che avete scaldato il sensore in misura notevole. Per raffreddare più rapidamente il sensore usate un vapore umido, o uno spray refrigerante di quelli che servono per raffreddare i componenti elettronici. Se non lo avete, andrà bene anche un cubetto di ghiaccio.

È possibile "espandere" l'immagine visualizzata al fine di avere una rappresentazione più utile delle variazioni di temperatura? E come pensate di fare?

L'immagine visualizzata si può "espandere" in diversi modi. Se sapete che le temperature varieranno non oltre l'intervallo 200-300 K, potete modificare il programma in modo che l'immagine visualizzata sullo schermo rappresenti tensioni comprese fra +2 e +3 volts. Ma non dimenticate che, così facendo, non avete aumentato la risoluzione del convertitore, e che quindi l'intervallo ammesso dal convertitore è relativo ancora allo stesso numero di passi distinti di tensione: avete solo espanso l'immagine visualizzata di tali valori.

Potete anche usare altri circuiti, ad esempio degli amplificatori operazionali, per portare l'intervallo di tensione da +2/+3 volts a 0-5 volts in modo che l'intero intervallo da 200 a 300 K di temperatura generi tensioni da 0 a 5 volts. Questo si può misurare con il convertitore e visualizzare sullo schermo. Questa volta la risoluzione è stata aumentata, perché nell'intervallo di temperatura che interessa rientrano tutte le 256 tensioni possibili.

Ci sarebbe da dire molto di più riguardo all'interfacciamento di un convertitore analogico, ma comunque ci auguriamo che quest'esperimento abbia destato il vostro interesse sull'uso di questi importanti dispositivi. Per ulteriori principi teorici e tecniche d'interfacciamento, vi consigliamo la lettura di: *TRS-80 Interfacing, Book* 2, e *Microcomputer-Analog Converter Software and Hardware Interfacing*, entrambi editi da Howard W. Sams & Co., Inc., Indianapolis, IN 46268.

Tenete presente che in quest'esperimento abbiamo generato una tensione di riferimento di 2.5 volts utilizzando due resistenze da 1000 ohm per dividere a metà l'alimentazione di +5 volts. Nelle applicazioni in cui è richiesta la precisione della conversione analogica si ricorre, al posto delle resistenze, ad un riferimento di +2.500 volts. In quest'esperimento abbiamo optato per le resistenze perché costano poco e sono facili da montare. Va detto però che forniscono risultati non abbastanza accurati, come sarebbe necessario nelle misure di precisione. Disponiamo comunque di molti dispositivi e circuiti di riferimento, come è detto nei testi citati prima.

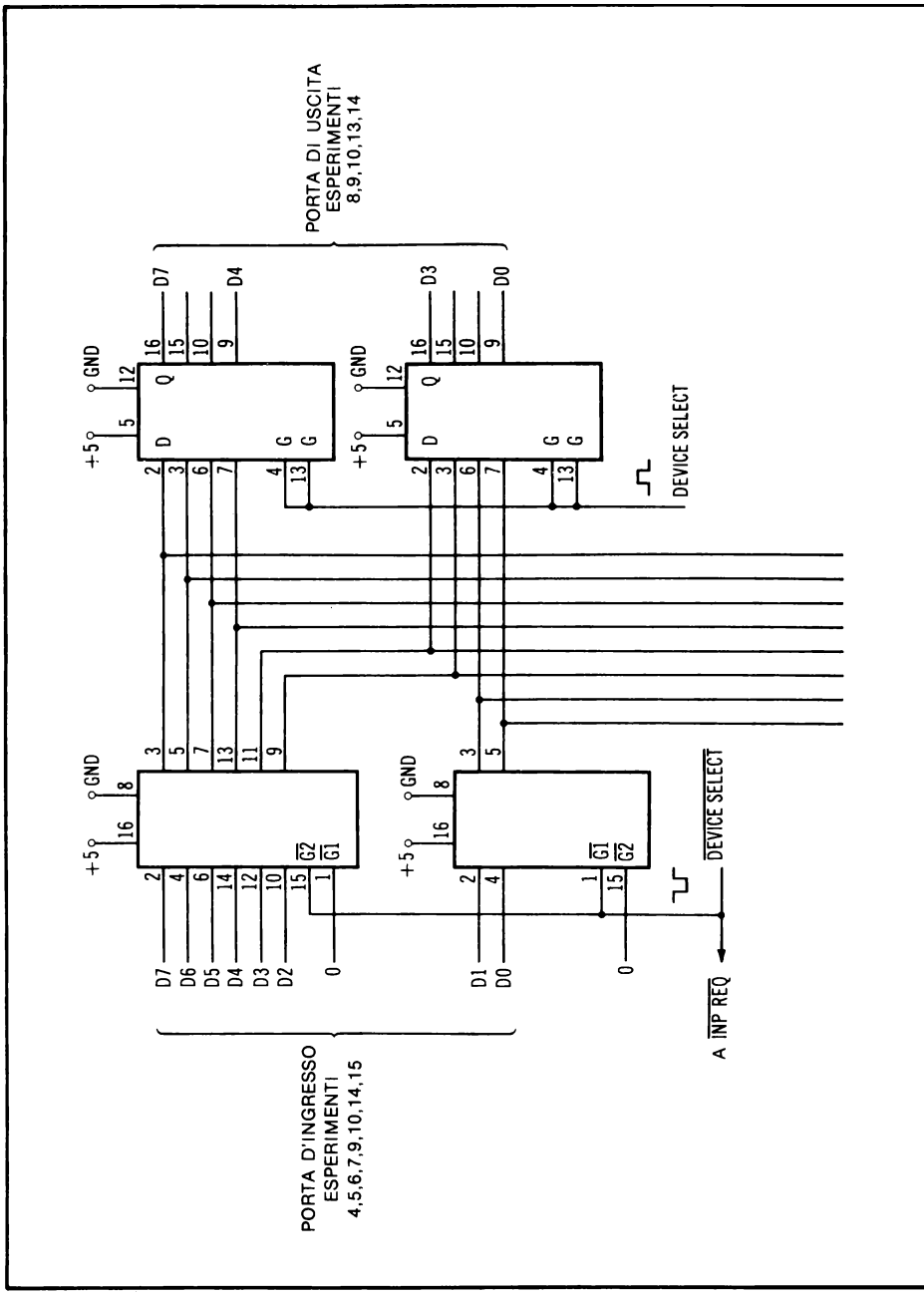
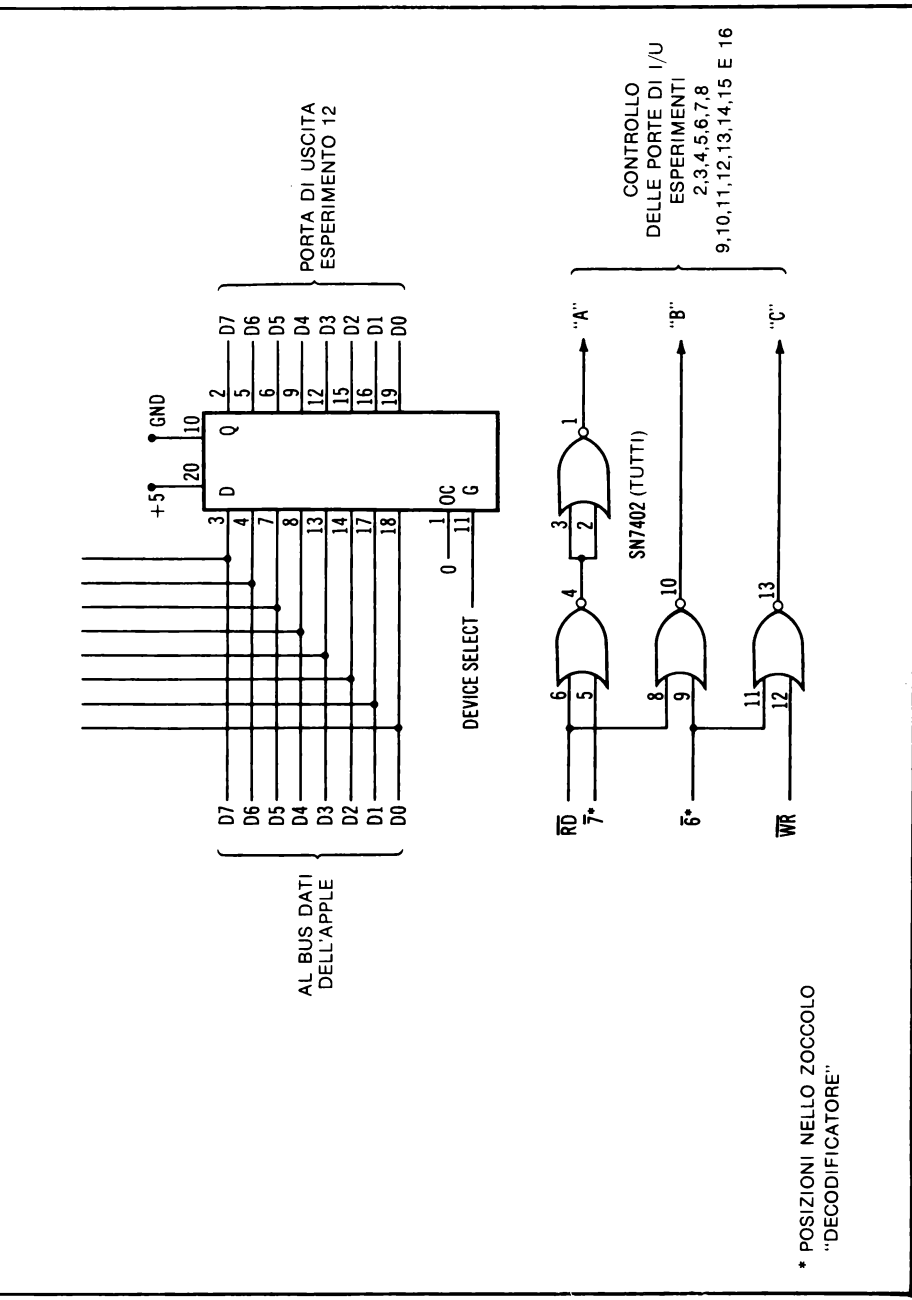


Figura 6.27 — Schema generale delle porte di I/U



e del circuito di controllo usati negli esperimenti.

CAPITOLO 7

IL BUS

Certamente a molti lettori basterà eseguire qualche esperimento del capitolo precedente, non essendo interessati al progetto ed allo sviluppo d'interfacce; ad altri invece interesserà sviluppare circuiti d'interfaccia specifici, destinati a diventare parte integrante del loro calcolatore. Ed è a questi ultimi che è dedicato questo capitolo. In esso spiegheremo come progettare speciali circuiti d'interfaccia che possano avvalersi di molte delle caratteristiche presenti nel calcolatore Apple.

Se avete l'esigenza di costruire un circuito d'interfaccia da usare ripetutamente, vorrete certamente costruirlo su qualcosa di diverso da una scheda di prova forata. I circuiti montati su scheda di prova occupano spazio, non sono ordinati, e spesso si staccano nel momento meno adatto. L'alternativa è costruire il circuito d'interfaccia in una forma permanente, in modo da montarlo in maniera sicura, all'interno della carrozzeria dell'Apple.

I progettisti che hanno concepito il calcolatore Apple sono partiti dal presupposto che gli utilizzatori sarebbero stati interessati ad espandere il sistema in modo da aggiungere al calcolatore periferiche standard e circuiti non standard. Pertanto sono stati predisposti otto connettori piatti femmina sulla parte posteriore della scheda a circuito stampato principale, in modo che i più importanti segnali del calcolatore fossero immediatamente disponibili per chi ne avesse bisogno. Alcuni di questi segnali li avete già utilizzati, perché l'interfaccia descritta precedentemente s'inserisce in uno degli "slots" disponibili.

Gli slots sono numerati da 0 a 7, e sono utilizzabili tutti tranne lo 0, riservato dal costruttore ad espansioni particolari del calcolatore. Gli slots dall'1 al 7 si possono invece usare come si vuole.

Molte ditte mettono in vendita interfacce compatibili con l'Apple che si possono inserire in questi slots senza dover fare nient'altro.

Nel Capitolo 5 abbiamo descritto alcuni segnali d'interfaccia più comunemente utilizzati: i segnali relativi al bus indirizzi, quelli relativi al bus dati, e alcuni segnali di controllo. Ma sono disponibili altri segnali, predisposti sui sette connettori piatti d'interfaccia; questi segnali sono descritti nella Tabella 7.1

Pin	Nome	Descrizione
1	$\overline{\text{I/O SELECT}}$	Segnale a logica negativa, attivo sullo slot *n*, quando il calcolatore indirizza le locazioni C*n*00H-C*n*FFH. Attivo durante Φ_0. Non disponibile sullo slot 0. (10)*
2-17	A15-A0	Linee del bus indirizzi bufferizzato. (5)
18	R/$\overline{\text{W}}$	Segnale di controllo di lettura/scrittura bufferizzato. (2)
19	SYNC	Segnale per la temporizzazione dei segnali video. Disponibile solo sullo slot 7. (?)
20	$\overline{\text{I/O STROBE}}$	Segnale a logica negativa, attivo su tutti gli slots quando il calcolatore indirizza le locazioni C800H-CFFFH. Attivo durante Φ_0. (4)
21	RDY	Segnale d'ingresso di controllo dello stato di "pronto" del microprocessore 6502.
22	DMA	Segnale d'ingresso di controllo dell'accesso diretto alla memoria (DMA).
23	INT OUT	Segnale d'interruzione con connessione a festone dello slot adiacente.
24	DMA OUT	Segnale di DMA con connessione a festone allo slot adiacente.
25	+5 volts	Collegamento di alimentazione a +5 volts: 500 mA massimo disponibili per tutte le schede.
26	GND	Massa elettrica del sistema.
27	DMA IN	Segnale di DMA con connessione a festone allo slot adiacente.
28	INT IN	Segnale d'interruzione con connessione a festone allo slot adiacente.
29	$\overline{\text{NMI}}$	Ingresso d'interruzione non mascherabile, destinato al chip 6502. Il vettore caricato punta alla subroutine in 03FBH.
30	$\overline{\text{IRQ}}$	Ingresso d'interruzione mascherabile, destinato al chip 6502. L'indirizzo della subroutine d'interruzione è in 03FE e 03FF.
31	$\overline{\text{RES}}$	Linea d'ingresso/uscita. Quando la linea è tenuta bassa, l'Apple è messo nello stato iniziale. L'interfaccia può controllarla o generare lo stato iniziale (reset).
32	$\overline{\text{INH}}$	Quando è portato a zero logico, tutte le ROM interne sono disabilitate.
33	−12 V	Collegamento di alimentazione a −12 volts: 200 mA complessivi disponibili per tutte le schede.
34	−5 V	Collegamento di alimentazione a −5 volts: 200 mA complessivi per tutte le schede.

Pin	Nome	Descrizione
35	COLOR REF	Questo segnale, di 3.580 MHz, di riferimento colore, è presente solo sullo slot 7. (?)
36	7M	Segnale standard di riferimento di 7.159 MHz. (2)
37	Q3	Segnale standard di riferimento di 2.046 MHz. (2)
38	Φ_1	Segnale standard di temporizzazione del microprocessore di 1.023 MHz. (2)
39	USER 1	Ingresso a logica negativa. Quando la linea è tenuta bassa, tutti i dispositivi interni di I/U sono disabilitati.
40	Φ_0	Segnale standard di temporizzazione del microprocessore di 1.023 MHz. Complemento di Φ_1. (2)
41	$\overline{\text{DEVICE SELECT}}$	Segnale a logica negativa, uno per slot. Attivo per 16 indirizzi per slot (v. Tabella 7.3). (10)
42-49	D7-D0	Linee del bus dati bufferizzato. (1)
50	+12 V	Collegamento di alimentazione a +12 volts: 250 mA complessivi disponibili per tutte le schede.

* I numeri fra parentesi indicano il numero d'ingressi della famiglia SN74LS00 che ciascun segnale può pilotare per ogni slot d'interfaccia.

Tabella 7.1 — Segnali relativi al bus dell'Apple, e loro descrizione.

Non parleremo più diffusamente dei segnali relativi al bus dati e al bus indirizzi, perché è un argomento che vi è già noto. Comunque, anche gli altri segnali sono importanti, e nel nostro caso semplificano notevolmente la costruzione dei circuiti d'interfaccia.

I SEGNALI DI CONTROLLO DELL'INTERFACCIA

I/O SELECT

Il segnale $\overline{\text{I/O SELECT}}$ (pin 1) è attivo quando è a zero logico, come indica la sopralineatura. Ciascuno dei sette slots d'interfaccia disponibili, dall'1 al 7, ha il suo specifico segnale $\overline{\text{I/O SELECT}}$: quindi questo segnale può essere utilizzato per selezionare una specifica scheda. Il segnale I/O SELECT relativo allo slot di scheda *n* è attivo quando le linee del bus indirizzi sono posizionate sugl'indirizzi che vanno da C*n*00 a C*n*FF compreso. Ad esempio, se l'Apple indirizza la locazione C5AB, il segnale $\overline{\text{I/O SELECT}}$ relativo allo slot 5 sarà uno zero logico: in questo momento non sarà attivo nessun segnale $\overline{\text{I/O SELECT}}$ relativo agli altri slots. A volte poi non è attivo *nessun* segnale di questo tipo. In Tabella 7.2 sono elencati gl'indirizzi che hanno effetto sui segnali $\overline{\text{I/O SELECT}}$.

Slot d'interfaccia	Intervallo degl'indirizzi	
1	C100-C1FF	49408-49663
2	C200-C2FF	49664-49919
3	C300-C3FF	49920-50175
4	C400-C4FF	50176-50431
5	C500-C5FF	50432-50687
6	C600-C6FF	50688-50943
7	C700-C7FF	50944-51199

Tabella 7.2 — Attribuzione degl'indirizzi di $\overline{\text{I/O SELECT}}$.

Questo segnale si presta a tutta una serie d'impieghi. Essendo attivo quando l'Apple indirizza un blocco continuo di 256 indirizzi, ossia una pagina, il segnale può servire per abilitare un chip di memoria con 256 indirizzi. Può anche essere utilizzato per abilitare un decodificatore d'indirizzo dispositivo capace d'indirizzare 256 dispositivi d'I/U. Queste applicazioni sono rappresentate sotto forma di diagramma a blocchi rispettivamente nelle Figure 7.1 e 7.2.

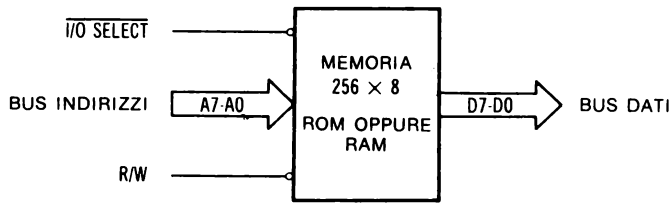


Figura 7.1 — Uso del segnale I/O SELECT per il controllo di una pagina di memoria.

Sarete forse stupiti del fatto che si possa aver bisogno di aggiungere un blocco di 256 bytes di memoria ad un calcolatore Apple, che può già di per sé contenere senza problemi 48 K di memoria. Ma in alcune applicazioni è necessario disporre di brevi routines in linguaggio assemblatore capaci di "gestire" un'interfaccia. I programmi in linguaggio assemblatore svolgono il loro compito con molta efficienza. Tali "routines di gestione" possono essere collocate in una memoria di sola lettura (ROM), ed il chip di ROM si può usare nei circuiti d'interfaccia. In tal modo le routines di gestione diventano una parte dell'interfaccia complessiva, e vengono "caricate" quando s'inserisce la scheda d'interfaccia. Non devono venir caricate da una cassetta o un disco, e non occupano ulteriore spazio di memoria.

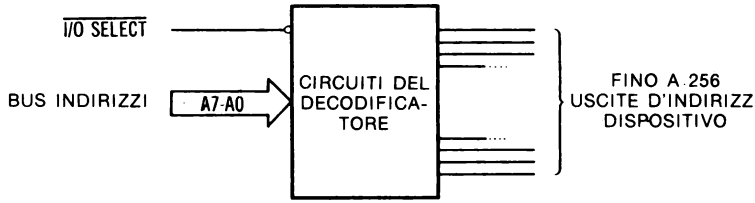


Figura 7.2 — Uso del segnale $\overline{\text{I/O SELECT}}$ per il controllo di un decodificatore d'indirizzi di memoria.

I/O STROBE

Il segnale $\overline{\text{I/O STROBE}}$ è un segnale a logica negativa che si ha su tutti gli slots d'interfaccia. È comune a tutti i connettori, non specifico di nessuno di essi. Sarà uno zero logico tutte le volte che l'Apple accederà ad una locazione compresa fra C800H e CFFFH incluso. Quindi ogni scheda riceverà questo segnale quando l'indirizzo presente sul bus indirizzi rientra in quest'intervallo, che comprende 2048 indirizzi, ossia 2 K di memoria.

Potete usare questo segnale per abilitare chips di memoria e dispositivi di I/U, ma, supponendo che desideriate "qualificarlo" ulteriormente, potrete inviarlo con alcune delle linee del bus indirizzi, cioè le linee A10-A0. La Figura 7.3 riporta un semplice diagramma a blocchi che illustra il modo d'uso di questo segnale.

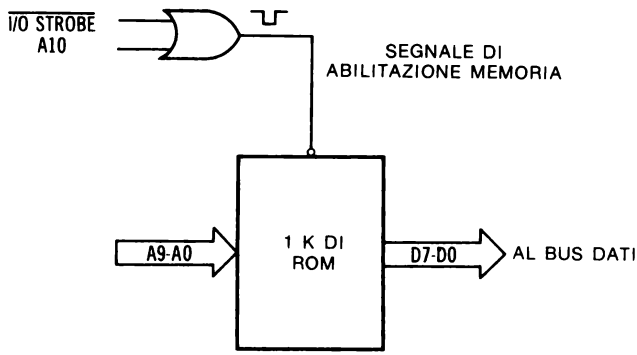


Figura 7.3 — Uso del segnale $\overline{\text{I/O STROBE}}$ per il controllo di un blocco di memoria da 1 K.

In questo circuito il segnale $\overline{\text{I/O STROBE}}$ è stato usato per selezionare un blocco di 1 K della memoria ROM su una scheda d'interfaccia. I restanti 1024 indirizzi possono essere suddivisi fra le altre interfacce secondo le proprie necessità. Comunque vi raccomandiamo cautela nell'uso di questo segnale, perché, come vi capiterà for-

se di constatare, alcuni costruttori hanno destinato questa linea alla decodifica della memoria e degl'indirizzi relativi ai dispositivi di I/U, fatta precisamente in questo modo. Ne consegue che potrete riscontrare conflitti d'indirizzamento fra un'interfaccia commerciale che avete l'intenzione di aggiungere al vostro sistema ed una che avete già progettato, costruito ed installato.

In alcuni casi l'interfaccia richiederà una quantità ridotta di memoria di lettura/scrittura (RAM) per una memorizzazione temporanea: la linea di $\overline{\text{I/O SELECT}}$ può servire anche a controllare un blocco di memoria RAM da 256 bytes.

Non dimenticate che ogni slot d'interfaccia ha il suo segnale $\overline{\text{I/O SELECT}}$, e che ciascun segnale è attivo quando l'Apple indirizza una particolare "pagina" di memoria.

$\overline{\text{DEVICE SELECT}}$

Si tratta di un segnale specifico di ciascuno degli slots d'interfaccia, per cui ha per ogni slot un intervallo di 16 indirizzi, come si può vedere in Tabella 7.3.

Slot d'interfaccia	Intervallo degl'indirizzi	
0	C080-C08F	49280-49295
1	C090-C09F	49296-49311
2	C0A0-C0AF	49312-49327
3	C0B0-C0BF	49328-49343
4	C0C0-C0CF	49344-49359
5	C0D0-C0DF	49360-49375
6	C0E0-C0EF	49376-49391
7	C0F0-C0FF	49392-49407

Tabella 7.3 — Attribuzione degl'indirizzi di $\overline{\text{DEVICE SELECT}}$.

Il segnale $\overline{\text{DEVICE SELECT}}$ è attivo nello stato logico zero, ed, essendo attivo unicamente per un blocco di 16 indirizzi, il suo uso sarà limitato all'indirizzamento di dispositivi di I/U (v. Figura 7.4).

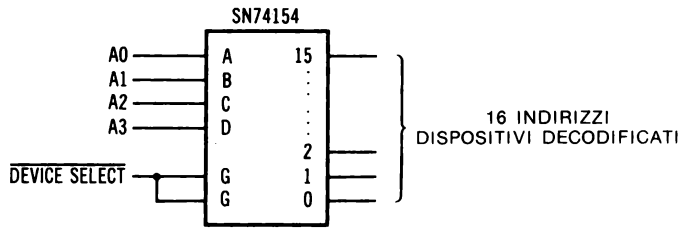


Figura 7.4 — Uso del segnale $\overline{\text{DEVICE SELECT}}$ per abilitare un decodificatore di 16 indirizzi.

In questo circuito il segnale $\overline{\text{DEVICE SELECT}}$ serve ad abilitare un decodificatore da 4 a 16 linee. Se un'interfaccia particolare ha soltanto un'unica funzione, e richiede soltanto un unico segnale di abilitazione, potete decidere di usare il segnale $\overline{\text{DEVICE SELECT}}$ da solo, senza un'ulteriore decodifica. Ciò è ammesso, a patto però di avere ben chiaro che il dispositivo così selezionato sarà attivo su 16 diversi indirizzi, dal C0*n*0H al C0*n*FH compreso. Inoltre questo modo d'uso del segnale limiterà la vostra capacità di aggiungere altre funzioni all'interfaccia, nel caso che vogliate espanderla successivamente.

$\overline{\text{IRQ}}$ ed $\overline{\text{NMI}}$

Sono due ingressi d'interruzione del microprocessore 6502. L'$\overline{\text{IRQ}}$ (richiesta d'interruzione) è mascherabile, e si può disabilitare per mezzo d'istruzioni software appropriate. L'$\overline{\text{NMI}}$ (interruzione non mascherabile) è sempre attivo.

Queste linee d'ingresso interruzione sono comuni a tutti e sette gli slots d'interfaccia: il segnale $\overline{\text{IRQ}}$ è collegato al pin 30, ed il segnale $\overline{\text{NMI}}$ è collegato al pin 29. Nella maggior parte dei circuiti d'interfaccia la linea $\overline{\text{NMI}}$ dovrebbe essere riservata ad una specifica periferica, non importando ciò che deve essere individuato; la linea $\overline{\text{IRQ}}$ va suddivisa fra più circuiti d'interfaccia. La subroutine di servizio dell'interruzione deve contenere delle istruzioni software opportune, in modo che il calcolatore possa individuare quale dispositivo ha realmente richiesto l'interruzione. Ogni dispositivo che genera un'interruzione deve avere una porta d'ingresso ad 1 bit, la cui lettura servirà a determinare lo stato relativo al flag d'interruzione. La Figura 7.5 riporta un tipico circuito relativo al flag d'interruzione. Tenete presente che l'azzeramento del flag avviene sotto il controllo software.

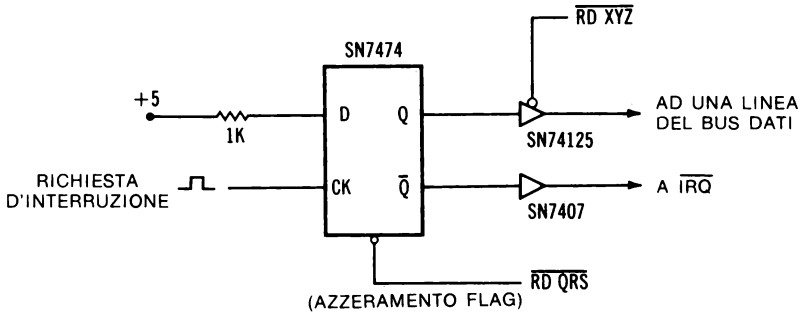


Figura 7.5 — Schema del circuito relativo al flag d'interruzione.

Ricorrendo a questo tipo d'interruzione, nel quale il calcolatore "interroga" uno per uno i dispositivi che possono aver generato un'interruzione, si può stabilire una priorità a livello software. Così, se il calcolatore verifica i dispositivi nell'ordine A, B, C, e così via, la priorità spetta al dispositivo A, che sarà infatti verificato per primo quando sarà rilevata un'interruzione.

Gli slots d'interfaccia hanno anche altre due linee d'interruzione che possono essere utili quando l'applicazione lo richieda. Questi segnali sono il segnale d'ingresso interruzione (INT IN), presente sul pin 28, e il segnale di uscita interruzione (INT OUT), presente sul pin 23. Questi segnali fanno sì che i segnali d'interruzione, connessi "a festone", colleghino una scheda a quella che vien dopo. I segnali vengono collegati solo fra i connettori d'interfaccia (v. Figura 7.6). Pertanto il segnale INT OUT presente sullo slot 1 è collegato al segnale INT IN presente sullo slot 2, il segnale INT OUT presente sullo slot 2 è collegato al segnale INT IN presente sullo slot 3, e così via. Le linee INT IN ed INT OUT sono collegate unicamente con lo slot d'interfaccia vicino, e non vanno oltre.

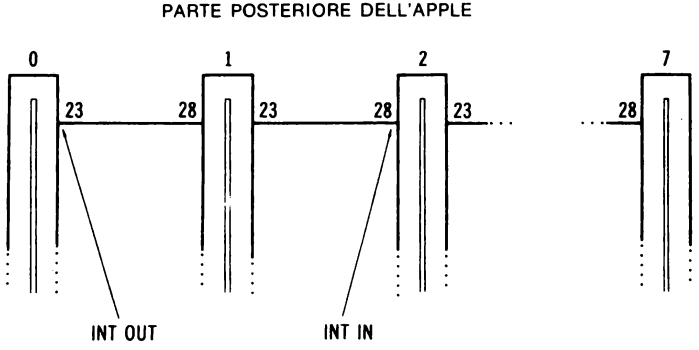


Figura 7.6 — I segnali di bus INT IN ed INT OUT.

La Figura 7.7 riporta un semplice schema d'interruzione collegato a festone. I dispositivi d'interruzione a più bassa priorità si trovano ancora più in basso nella catena, lontano dalla connessione del segnale $\overline{\text{INT}}$ al microprocessore 6502. In questo circuito un dispositivo a priorità più alta può far passare la sua richiesta d'interruzione lungo la catena, bloccando tutte le richieste d'interruzione provenienti dai dispositivi a priorità più bassa, che si trovano in posizioni più basse nella catena. Dopo che il dispositivo a priorità più alta è stato servito e che il relativo flag d'interruzione è stato azzerato, verrà "aperta" la porta logica per consentire alla richiesta d'interruzione a priorità più bassa di raggiungere il calcolatore.

Come potete vedere, il calcolatore deve sempre in qualche modo determinare quale dispositivo sta generando l'interruzione, in modo da poter selezionare la corrispondente subroutine di servizio dell'interruzione. Si tratta di uno schema d'interruzione abbastanza complesso, per cui vi consigliamo di usare il più semplice circuito con flag d'interruzione presentato in Figura 7.5: questo circuito sarà più che sufficiente per la maggior parte delle applicazioni. Nella configurazione a collegamento a festone non si possono avere slots "vuoti", cioè aperti, interposti fra le piastre con-

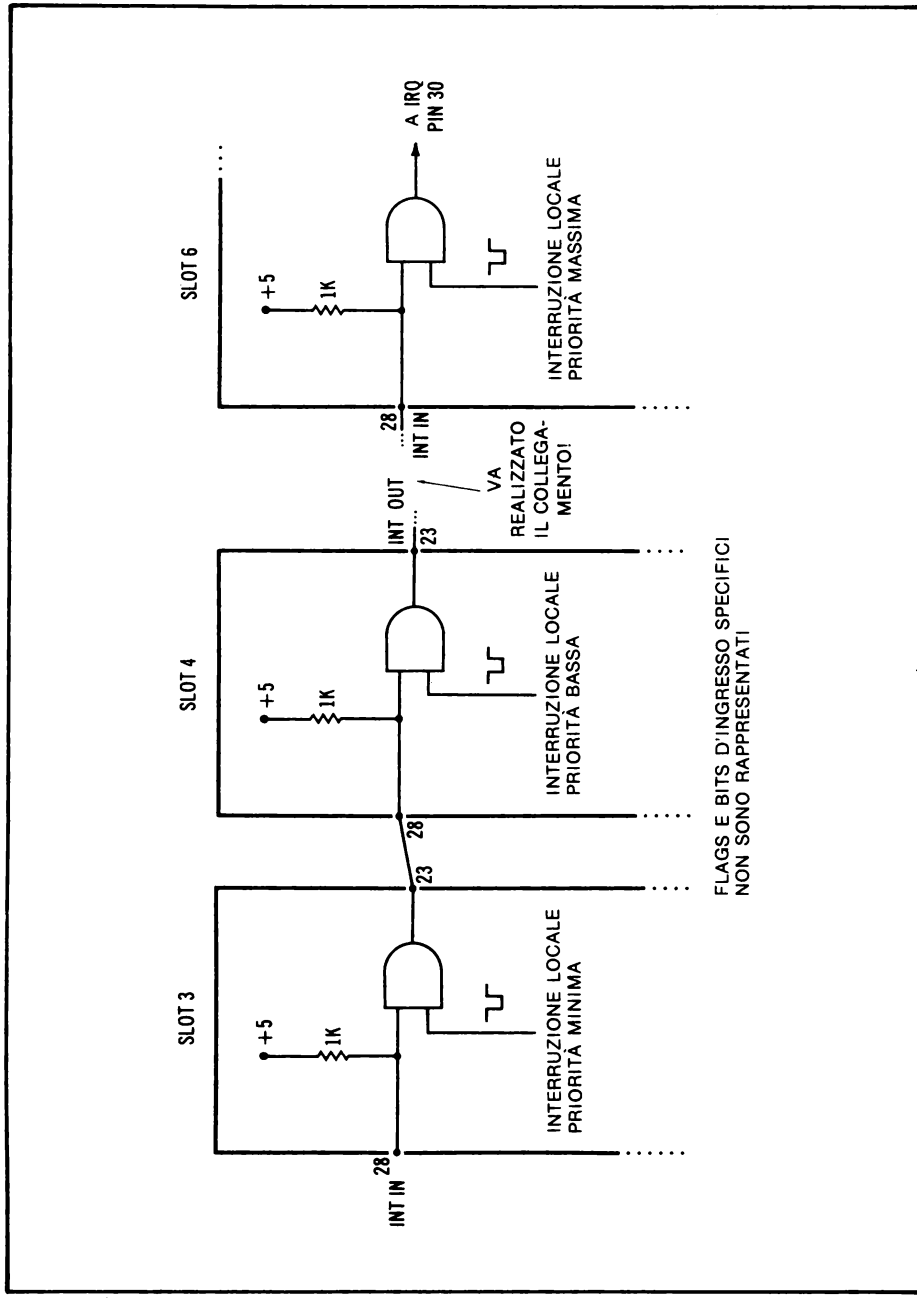


Figura 7.7 — Configurazione del segnale d'interruzione collegato a festone.

tenenti circuiti d'interfaccia, perché questo spezzerebbe la "catena" di circuito formata da INT IN/INT OUT.

Tanto basti per le interruzioni. Per ulteriori informazioni, vi consigliamo la lettura di *Programming & Interfacing the 6502, With Experiments*, Howard W. Sams & Co., Inc., Indianapolis, IN 46268.

$\overline{\text{DMA}}$

L'ingresso $\overline{\text{DMA}}$ consente ad un dispositivo esterno d'indirizzare locazioni di memoria senza dover prima passare per il microprocessore 6502. Ciò vuol dire che *il dispositivo esterno ha accesso diretto in memoria* (donde la sigla DMA, da *Direct Memory Access*). Dal momento che può capitare che più dispositivi richiedano un trasferimento d'informazioni in accesso diretto alla memoria, si può realizzare un collegamento a festone di un certo numero di periferiche: infatti gli slots d'interfaccia hanno i pins di DMA IN e di DMA OUT, che permettono il collegamento con i connettori d'interfaccia contigui. Le interfacce ad accesso diretto in memoria non sono una cosa semplice da progettare, per cui vi raccomandiamo di capire molto bene il funzionamento del microprocessore 6502 e della circuiteria relativa prima di provare ad usare questa funzionalità.

$\overline{\text{RES}}$

La linea di azzeramento (cioè di reset) $\overline{\text{RES}}$ è in realtà una linea di segnale bidirezionale, utilizzabile per riportare allo stato iniziale i circuiti d'interfaccia, perché sarà uno zero logico quando l'Apple, all'accensione, o quando si preme il tasto RESET, viene portato allo stato iniziale. Si può forzare l'Apple nella condizione di stato iniziale anche collegando questa linea a massa. Se decidete di usare questa linea per riportare l'Apple allo stato iniziale mediante la vostra interfaccia, dovete ricorrere ad una porta a collettore aperto e ad alto assorbimento di corrente, cioè ad un buffer, per portare la linea a massa. In un circuito di questo tipo si può usare il chip SN7407 relativo ad un buffer a collettore aperto. La linea del segnale $\overline{\text{RES}}$ è comune a tutti gli slots d'interfaccia.

$\overline{\text{INH}}$

Nel calcolatore Apple è possibile sostituire i propri programmi in linguaggio assemblatore ai programmi memorizzati nelle ROM riservate all'interprete BASIC. Collegando a massa la linea $\overline{\text{INH}}$ presente sul pin 32, verranno inibite tutte le ROM che contengono l'interprete BASIC ed il Monitor, in modo che siano i vostri programmi a controllare l'intero sistema. Dal momento che c'è già modo di effettuare tale tipo di operazione, è possibile che non intendiate utilizzare questa funzione, perché non avrete accesso a nessuna delle subroutines presenti nelle ROM standard fornite insieme con l'Apple. Ad esempio sarebbe difficile controllare la visualizzazione sen-

za le subroutines residenti nelle ROM riservate all'interprete BASIC. Se volete utilizzare questa funzione, vi servirà un chip di buffer a collettore aperto per collegare a massa questa linea.

USER 1

Quest'ingresso vi permetterà d'impedire la generazione di tutti i segnali $\overline{\text{I/O SELECT}}$ e $\overline{\text{DEVICE SELECT}}$ all'interno del calcolatore Apple, in modo da poter "disattivare" tutti i dispositivi di I/U. A tal fine questa linea va portata a zero logico. Per prevenirne l'uso accidentale, collegate con un ponticello le due parti di pista della scheda principale dell'Apple prima che il segnale USER 1 possa essere utilizzato. Per i particolari consultate l'*Apple II Reference Manual.*

Dal momento che il motivo più importante per cui si usano i segnali $\overline{\text{I/O SELECT}}$ e $\overline{\text{DEVICE SELECT}}$ è la semplificazione del progetto dell'interfaccia, è probabile che non ci sia bisogno di usare questa linea, a meno che si voglia realizzare una sorta di espansione del sistema per mezzo di dispositivi di I/U esterni alla configurazione di base del sistema, o che si utilizzino indirizzi di memoria assegnati ai segnali $\overline{\text{I/O SELECT}}$ e $\overline{\text{DEVICE SELECT}}$. Il segnale USER 1 è presente sul pin 39 dei connettori d'interfaccia.

RDY

A volte è necessario "ritardare" leggermente il microprocessore 6502 in modo che un dispositivo di I/U esterno o un chip di memoria abbia tempo sufficiente per accedere ai propri dati e presentarli sul bus dati. Il segnale d'ingresso RDY, che si trova sul pin 21 di tutti i connettori d'interfaccia, se collegato a massa pone il 6502 in una situazione di "attesa". Quest'ingresso dev'essere sincronizzato con il clock del microprocessore, ed i suoi cambiamenti di stato dovrebbero avvenire durante lo stato logico uno del clock Φ_1. L'ingresso RDY era usato nei vecchi calcolatori basati sul 6502, per il fatto che i relativi dispositivi di memoria non erano in grado di accedere ai dati alla velocità richiesta dal calcolatore, e quindi, quando erano indirizzati, dovevano porre il 6502 in una situazione di "attesa" che durava diversi cicli di clock, prima che i loro dati fossero disponibili. Pensiamo che farete un uso molto limitato di questo segnale, tranne che in interfacce specializzate.

I segnali di sincronizzazione

Esistono sei segnali di sincronizzazione disponibili per le interfacce: Φ_0, Φ_1, Q3, 7M, COLOR REF, SYNC. Φ_0 e Φ_1 sono i segnali di clock principale, che ha una frequenza di 1 MHz: questi segnali di clock sono l'uno l'inverso dell'altro, e servono a coordinare le operazioni di I/U esterne con il normale flusso dei dati lungo il bus. Come si può vedere in Figura 5.12, il segnale Φ_1 controlla la generazione dei segnali $\overline{\text{RD}}$ e $\overline{\text{WR}}$ destinati a dispositivi di I/U esterni. *I segnali $\overline{\text{I/O SELECT}}$ e $\overline{\text{DEVICE SE-}}$*

$\overline{LECT}$ presenti sui connettori di I/U sono già stati condizionati, cioè "qualificati", per mezzo del segnale di clock ϕ_1.

Il segnale Q3 è un segnale di clock a 2 MHz asimmetrico, cioè non è un'onda quadra.

Il segnale 7M è un segnale di clock a 7 MHz, ed è un'onda quadra.

I segnali di clock sono ottenuti dalla circuiteria del clock principale dell'Apple: i loro rapporti temporali sono rappresentati in Figura 7.8.

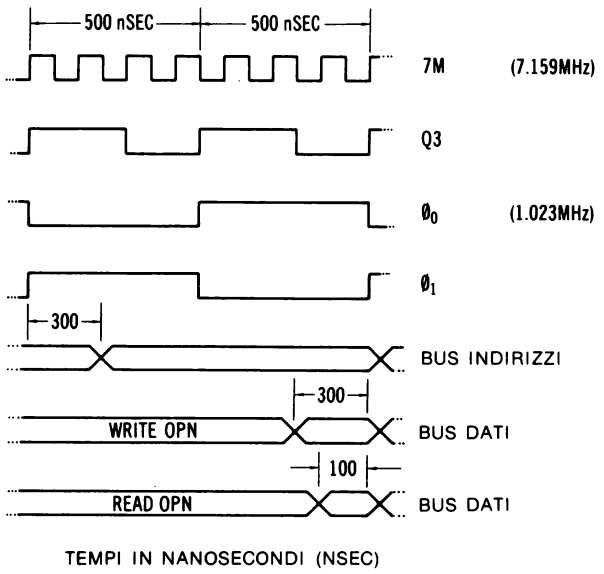


Figura 7.8 — Diagramma delle temporizzazioni dei vari segnali di clock dell'Apple.

Per ulteriori chiarimenti sulle temporizzazioni del 6502 vi rimandiamo ad un catalogo completo del microprocessore 6502.

I segnali COLOR REF e SYNC sono disponibili soltanto sullo slot d'interfaccia n. 7. Il segnale COLOR REF è il segnale di riferimento colore, ha una frequenza di 3,5 MHz ed è generato dal circuito di clock del video dell'Apple. Il segnale SYNC è il segnale di sincronizzazione delle temporizzazioni del video. È probabile che nei vostri progetti d'interfaccia non ricorrerete a questi segnali, a meno che utilizziate i circuiti di controllo video.

L'alimentazione

I connettori d'interfaccia danno accesso a quattro tensioni standard, cioè +12, −12, +5 e −5 volts, ed alla massa. La corrente per ciascuna di queste tensioni è li-

mitata a poche centinaia di milliampères, per cui dovreste prendere in considerazione chips d'interfaccia a bassa potenza (low power), come quelli della famiglia SN74LS00.

Altre osservazioni

La capacità di pilotaggio del bus da parte dei segnali d'interfaccia è abbastanza limitata, dal momento che la maggior parte di essi può pilotare soltanto un limitato numero d'ingressi del tipo SN74LS00. Nei vostri progetti state bene attenti a non sovraccaricare questi segnali, pretendendo che pilotino un numero d'ingressi del chip superiore a quello che possono pilotare. Se occorre che questi segnali forniscano una quantità maggiore di corrente, in modo da poter pilotare un maggior numero d'ingressi su una scheda d'interfaccia, allora dovete potenziare i segnali per mezzo degli opportuni chips di buffer. Tenete ben presente che i buffers avranno bisogno di corrente in più dalle alimentazioni, e che sui connettori d'interfaccia non si ha una grande quantità di corrente "extra". Quindi bisogna bilanciare le proprie esigenze di potenziamento dei segnali con la corrente di cui si può disporre. Si potrebbe sempre utilizzare un'alimentazione esterna per alimentare alcune schede d'interfaccia, ma questo vanificherebbe lo scopo dichiarato in precedenza, che è in primo luogo quello di sistemare i circuiti d'interfaccia nella carrozzeria dell'Apple.

UN ESEMPIO D'INTERFACCIAMENTO

Dopo aver descritto la maggior parte dei segnali d'interfaccia che servono, esaminiamo attentamente un tipico circuito d'interfaccia utilizzabile con il calcolatore Apple.

In molte applicazioni occorre che il calcolatore comunichi con altri dispositivi: stampanti, unità di controllo, stazioni remote di acquisizione di dati, o anche altri calcolatori. Per lo più si ricorre ad una forma di comunicazione seriale, che permette di evitare l'uso di cavi multiconduttore lunghi. Nella maggior parte degli schemi relativi alla comunicazione seriale si hanno tre o quattro fili, per cui l'informazione da scambiare viene trasmessa in modo seriale, un bit alla volta, lungo i fili. Un gruppo di fili è utilizzato dalla trasmissione, l'altro gruppo dalla ricezione. Questo tipo di comunicazione è detto generalmente comunicazione seriale-asincrona per il fatto che non prevede un segnale di clock, o un riferimento comune, che colleghi i due sistemi.

Quasi tutti i costruttori di chips di microprocessore hanno sviluppato un qualche tipo di chip di comunicazione per la famiglia di microprocessori che producono. In realtà, comunque, è possibile "incrociare" le famiglie, ad esempio utilizzando con un processore 6502 un chip di comunicazione originariamente sviluppato per la famiglia 8080A; ed è proprio quello che faremo in quest'esempio: interfacceremo cioè

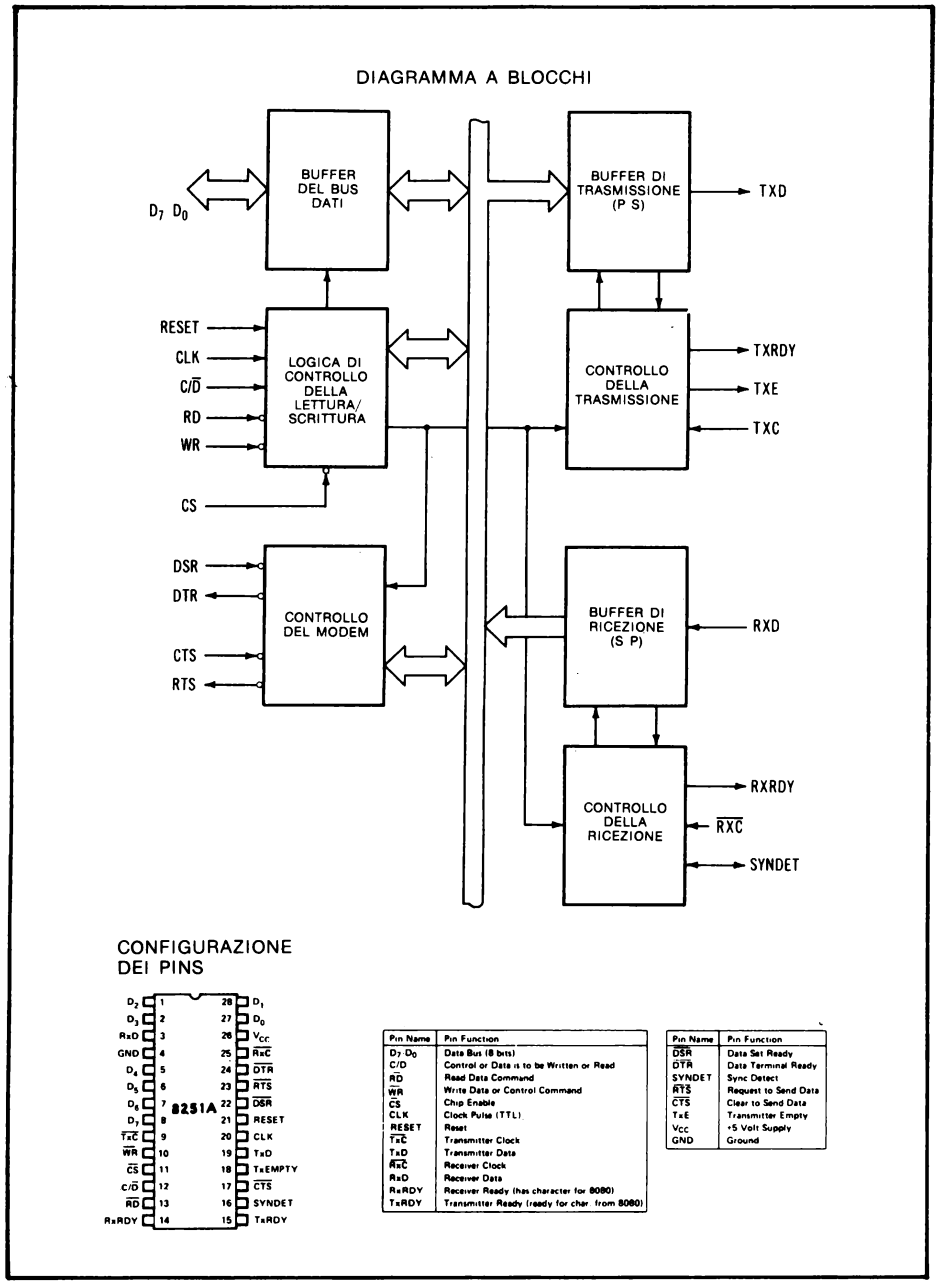


Figura 7.9 — Configurazione dei pins e diagramma a blocchi del chip USART 8251A.

il calcolatore Apple con l'8251, cioè un chip universale di ricezione-trasmissione sincrona/asincrona, utilizzando uno degli slots riservati alle interfacce. Non scenderemo molto in dettaglio riguardo al funzionamento del chip USART, perché potete trovarne una trattazione completa in *TRS-80 Interfacing, Book 2*, Howard W. Sams & Co., Inc., Indianapolis, IN 46268. Sull'argomento è disponibile anche un articolo: "Cross-Pollinating the Apple", in *Byte*, April 1979, pag. 24.

Il chip USART 8251 è un chip compatibile con una struttura a bus, per cui non dovrebbe essere troppo difficile interfacciarlo con l'Apple. La Figura 7.9 riporta la configurazione dei pins ed il diagramma a blocchi dell'USART.

Saprete certamente riconoscere gl'ingressi del bus dati, gl'ingressi di controllo $\overline{RD}$ e $\overline{WR}$ e l'ingresso di selezione chip, $\overline{CS}$. L'USART contiene due gruppi di registri, per cui dev'esserci un sistema per distinguerli: a ciò provvede l'ingresso CONTROL/$\overline{DATA}$, che si trova sul pin 12 (C/$\overline{D}$). L'uno logico poterà alla scelta della modalità controllo, o modalità comando, lo zero logico invece alla scelta della modalità dati. A quest'ingresso si può collegare uno dei bits d'indirizzo allo scopo di permettere al calcolatore di accedere ad ogni singolo registro interno utilizzando un indirizzo per il registro comando ed un altro indirizzo per il registro dati.

L'USART comunicherà con altri dispositivi asincrono-seriali, ragion per cui la trasmissione dei dati deve avvenire a velocità standard: di conseguenza la velocità dei dati dell'apparecchiatura di trasmissione e quella del calcolatore ricevente saranno abbastanza vicine. A questo scopo abbiamo optato per il chip MC14411 (bit rate generator) della Motorola, perché controllato da un cristallo. Comunque sono usati comunemente anche altri sistemi per la generazione della temporizzazione.

I livelli logici standard forniti dai dispositivi a logica transistor-transistor (TTL) della famiglia SN7400 non possono essere usati per pilotare linee di comunicazione lunghe; quindi dovete scegliere fra i segnali di current loop a 20 mA ed i livelli di controllo RS-232C standard. I necessari circuiti di conversione di livello sono facili da ottenere, e ne troverete una descrizione dettagliata nel testo e nell'articolo citati in precedenza.

Un qualunque tipo d'interfaccia di comunicazione non serve a niente senza il software che lo gestisca; quindi avrete bisogno di routines software che gestiscano il chip USART. Per lo più si tratta di routines semplici, e potrete senz'altro usare programmi in BASIC per il relativo controllo; se invece optate per programmi in linguaggio assemblatore, dovrete collocare i vostri programmi di controllo nella ROM, e quest'ultima sulla scheda d'interfaccia. Dal momento che per ogni slot d'interfaccia è disponibile uno spazio di memoria limitato a 256 bytes, è possibile utilizzare una ROM di piccole dimensioni. Lo spazio di 256 bytes è più che sufficiente per un numero non elevato di programmi di controllo dell'USART. Con il Monitor potete provare i vostri programmi in linguaggio assemblatore prima di passarli nella ROM.

In Figura 7.10 è rappresentato un interfacciamento completo dell'USART.

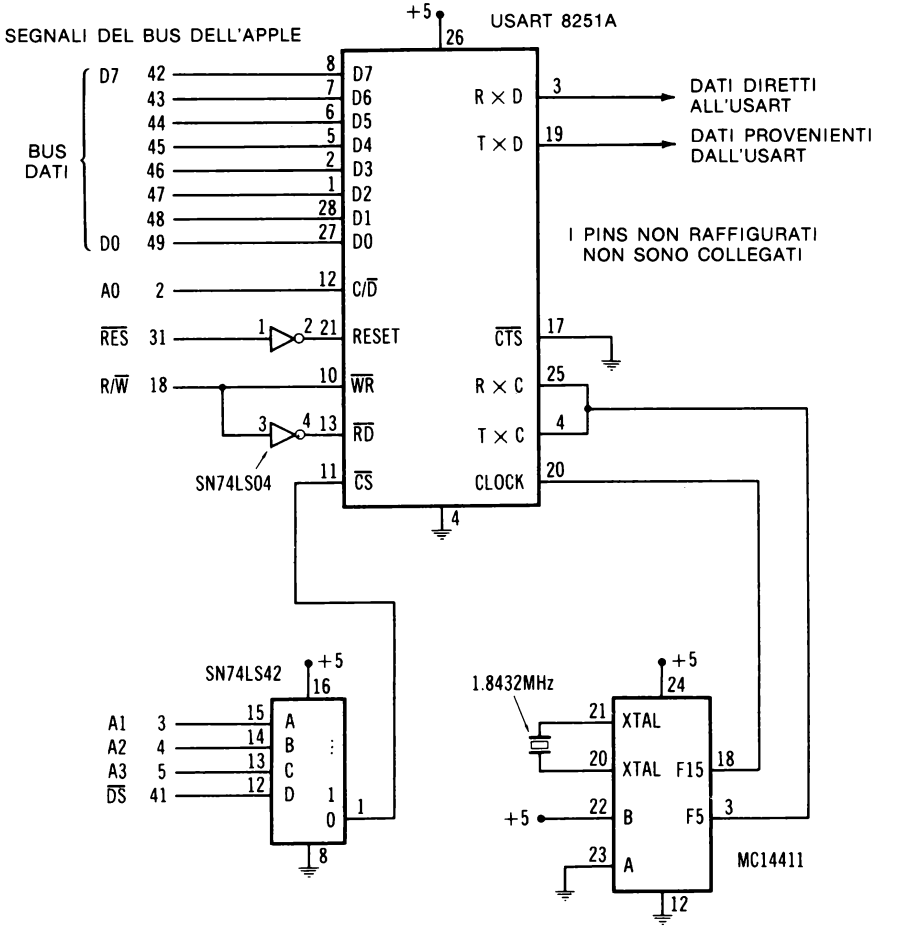


Figura 7.10 — Semplice schema del circuito d'interfacciamento dell'USART con l'Apple.

Questo circuito è stato cablato e provato sul nostro calcolatore Apple. Se intendete usarlo nel vostro calcolatore, vi suggeriamo di procurarvi i cataloghi relativi al chip USART 8251 o 8251A ed al chip MC14411 (bit rate generator) della Motorola, che vi permetteranno di capirne il funzionamento. La Figura 7.11 riporta un circuito d'indirizzamento generale per un blocco di ROM di 256 bytes, utilizzabile per memorizzare le routines di controllo dell'USART in linguaggio assemblatore. Il circuito effettivo dipenderà dal particolare chip o chips di ROM che deciderete di adottare. Nel circuito riportato in figura abbiamo utilizzato delle ROM 93427 della Fairchild, che sono ROM veloci, bipolari, e del tipo "fusible link". Ciascun chip contiene 1024 bits, organizzati in 256 parole di 4 bits, per cui, per formare una parola di 8 bits completa,

sono necessari due chips. Non sono consigliabili i chips PROM lenti e cancellabili, perché hanno tempi di accesso abbastanza lenti, e possono provocare dei problemi. La maggior parte di questi dispositivi contiene un numero di locazioni molto maggiore di quelle che vi capiterà di usare.

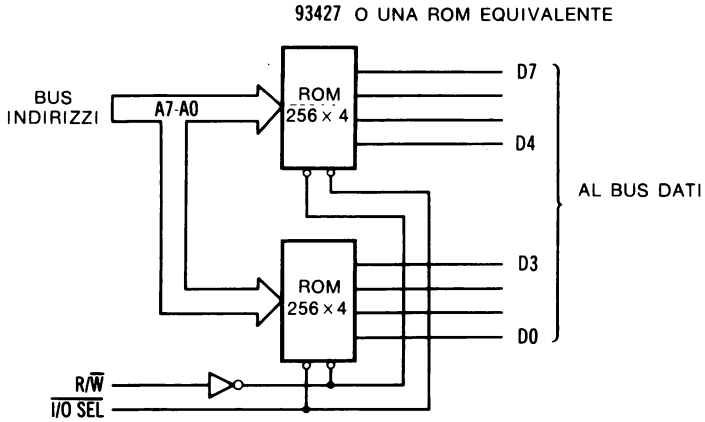


Figura 7.11 — Schema del circuito di espansione memoria da 256 bytes.

Potete costruire questo circuito su una scheda d'interfaccia standard ricorrendo a connessioni a filo (wire-wrap), o su un altro tipo di scheda prototipo adatto allo scopo, che si possa inserire in uno degli slots d'interfaccia disponibili. Se utilizzate la tecnica del prototipo con connessioni a filo (wire-wrap), noterete che i pins attorno ai quali è avvolto il filo ed i chips sporgono da entrambi i lati della scheda, cosa che rende difficile l'uso degli slots d'interfaccia adiacenti.

Nel nostro calcolatore abbiamo posto l'interfaccia con l'USART nello slot 3, in modo che l'USART venisse indirizzato come dispositivo 49328 e 49329. I registri corrispondenti all'indirizzo 49328 sono i registri di ricezione e di trasmissione, mentre i registri corrispondenti all'indirizzo 49329 sono i registri di controllo e di flag. Ricordate che potete disporre di due registri per ciascun indirizzo, perché l'uno è un registro di scrittura in, l'altro un registro di lettura da. Spostando la scheda in un altro slot d'interfaccia, gl'indirizzi relativi all'USART cambiano (v. Tabella 7.3).

Per usare l'interfaccia USART, dovete prima di tutto inizializzare il chip con alcune informazioni di controllo che sono inviate sotto forma di due bytes consecutivi al registro di controllo. Non preoccupatevi per il fatto che vengono inviati due bytes allo stesso registro: l'USART "sa" che cosa farne. Inizializzato l'USART, potete usarlo per trasmettere e ricevere flussi asincrono-seriali d'informazione. Con il programma riportato nell'Esempio 7.1 si può trasmettere un byte di dati di 8 bits; invece con il programma dell'Esempio 7.2 si può ricevere un byte di 8 bits.

ESEMPIO 7.1 — Subroutine di controllo della sezione trasmissione dell'USART

```
1010  POKE 49328,TX
1020  WAIT 49329,1
1030  RETURN
```

ESEMPIO 7.2 — Subroutine di controllo della sezione ricezione dell'USART

```
1050  WAIT 49329,2
1060  RX = PEEK(49328)
1070  RETURN
```

Il software verifica i flags necessari in modo che il trasmettitore trasmetta i suoi dati solo quando è pronto, ed il ricevitore fornisca i dati solo quando ne ha effettivamente ricevuti.

Qui il problema più importante è stato sviluppare una semplice interfaccia che utilizzasse molti dei segnali di controllo del bus d'interfaccia dell'Apple, per darvi la possibilità di vederli concretamente all'opera. È bene sapere anche che questi esempi d'interfaccia funzionano davvero, e si possono usare in molte applicazioni concrete.

Ci auguriamo che vi siate resi conto di come sia semplice sviluppare un'interfaccia per l'Apple, partendo dai concetti d'indirizzamento delle porte, di controllo delle porte e dei flags, etc., tutti concetti che sono stati trattati nel corso del libro.

APPENDICE A

LE FUNZIONI LOGICHE

Gli esperimenti svolti nel corso del libro hanno richiesto un certo numero di funzioni logiche, indicate come segnalatori luminosi, interruttori logici e generatori d'impulsi. In ogni caso i circuiti equivalenti sono semplici, ma, invece di ripeterli in ciascuno schema, abbiamo preferito ricorrere a diagrammi a blocchi. I paragrafi seguenti descrivono ciascuno una funzione.

I SEGNALATORI LUMINOSI

I segnalatori luminosi non sono altro che diodi che emettono luce (LED), o comunque dispositivi passibili di essere in ON o in OFF, in modo da indicare lo stato logico di un'uscita. Abbiamo adottato la convenzione secondo la quale uno logico vuol dire acceso, ovvero stato di ON, e zero logico vuol dire spento, ovvero stato di OFF. I due circuiti di Figura A.1 si possono usare per costruire segnalatori luminosi.

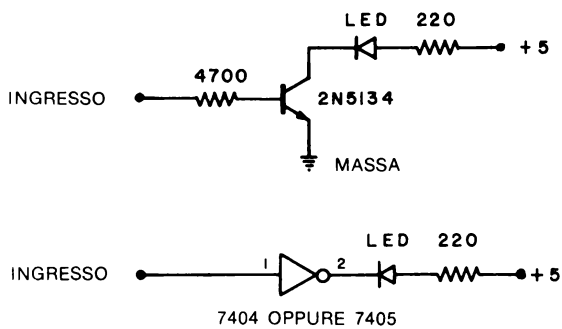


Figura A.1 — Schemi di due semplici circuiti per segnalatori luminosi utilizzabili negli esperimenti.

Consigliamo di usare LED rossi, perché costano poco e sono molto visibili. Per eseguire gli esperimenti presentati nel libro vi serviranno almeno otto segnalatori luminosi.

GL'INTERRUTTORI LOGICI

Gl'interruttori logici sono semplicemente degl'interruttori configurati in modo tale da fornire la tensione di uno logico, o quella di zero logico, agl'integrati, TTL compatibili, usati negli esperimenti. Un interruttore logico tipico è quello che compare in Figura A.2. Si può usare un interruttore a singola asticciola, oppure un interruttore a scorrimento. Nell'esperimento serviranno almeno otto circuiti d'interruttore logico.

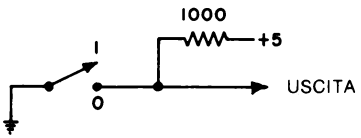


Figura A.2 — Schema di un semplice circuito ad interruttore logico utilizzabile per generare uscite di uno logico o di zero logico.

I GENERATORI D'IMPULSI

Negli esperimenti il circuito generatore d'impulsi provvede uscite "pulite", libere da "rimbalzi", che si hanno normalmente con gl'interruttori meccanici. Poiché nella maggior parte degl'interruttori si hanno contatti metallici tipo molla, i contatti spesso si aprono e si chiudono diverse volte dopo l'apertura o la chiusura dell'interruttore. Se si utilizza un siffatto interruttore meccanico per fornire impulsi ad un contatore, saranno contati fino a 30 o 40 impulsi, a seconda del tipo d'interruttore impiegato. In molti casi è necessario avere una transizione pulita dall'uno logico allo zero logico e dallo zero logico all'uno logico, quindi molto spesso sarà opportuno ricorrere ad un interruttore antirimbalzo. Si può facilmente eliminare il rimbalzo degl'interruttori meccanici se sono del tipo singola asticciola, doppio contatto. Un tipico circuito antirimbalzo è rappresentato in Figura A.3.

Qui due porte NAND formano un flip-flop che può venir portato ad uno o a zero logico tramite l'interruttore. Come si vede in Figura A.3, sono disponibili due uscite. Con l'interruttore nella posizione rappresentata nella figura, gli stati logici normali compaiono alle uscite delle porte; quando l'interruttore è spostato nell'altra posizione, le uscite delle porte NAND vengono commutate. Nei circuiti generatori d'impulsi vi suggeriamo di utilizzare un interruttore provvisorio.

Segnalatori luminosi, interruttori logici e generatori d'impulsi sono tutti dispositivi che si usano nei circuiti costruiti sulla scheda di prova. Benché i circuiti rappresen-

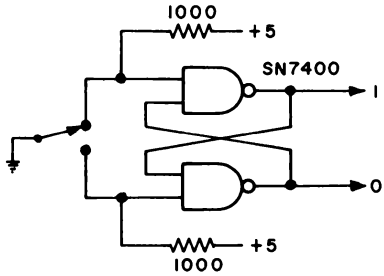


Figura A.3. — Schema di un generatore d'impulsi antirimbalzo nel quale si è usata una porta NAND "in collegamento incrociato" per eliminare i rimbalzi del contatto.

tati nelle Figure A.1, A.2 ed A.3 siano molto semplici, è possibile che non vogliate costruirveli da voi. Molte società producono dispositivi digitali per schede di prova nei quali compaiono segnalatori luminosi, interruttori logici e generatori d'impulsi, oltre ad altre funzioni digitali. Vi consigliamo pertanto di richiedere alle ditte che seguono informazioni sui loro sistemi elettronici digitali per schede di prova:

E & L Instruments, Inc.
61 First Street
Derby, CT 06418

AP Products, Inc.
Mentor, OH 44060

PACCOM
14825 NE 40th, Suite 340
Redmond, WA 98025

APPENDICE B

COMPONENTI UTILIZZATI NEGLI ESPERIMENTI

4 integrati SN7402: NOR quadruplo
2 SN7474: doppio flip-flop di tipo D
2 DM8095 o SN74365: buffer con ingresso a tre stati
2 SN7475: latch quadruplo
1 NE5018 (Signetics Corporation): convertitore D/A ad otto bits
1 SN7404: invertitore sestuplo
2 SN74LS373: latch ottale a tre stati
1 condensatore ceramico a disco da 0,01 μF
1 resistore da 4700 ohm ed 1/4 watt
6 resistori da 220 ohm ed 1/4 watt
6 LED (2 rossi, 2 verdi e 2 gialli)
1 potenziometro del tipo trimmer da 10 K
1 resistore da 10 K ed 1/4 watt
1 condensatore elettrolitico da 100 μF e 16 V dcw
1 resistore da 33 K ed 1/4 watt
1 condensatore a disco da 150 pF
1 resistore da 2200 ohm ed 1/4 watt
1 ADC0804 (National Semiconductor Corp.): convertitore analogico-digitale
1 LM335: sensore di temperatura
4 resistori da 1000 ohm ed 1/4 watt

Oltre a questi componenti, vi servirà un certo numero di circuiti integrati SN7400, SN7408, SN7402, SN7410, SN7486, SN7430 ed SN7493, da utilizzare nell'Esperimento n. 14 di prova logica dei componenti. Vi suggeriamo di leggere tutto quest'esperimento per decidere esattamente quali circuiti volete provare.

Potrà servire anche il seguente materiale: alimentatore a ±12 volts (da usare con il circuito convertitore D/A), filo per i collegamenti, una scheda forata di prova extra, generatori d'impulsi, interruttori logici, segnalatori luminosi, ed un voltmetro (vom).

Potrete richiedere delucidazioni sui convertitori analogici presso:

National Semiconductor Corp.
2900 Semiconductor Drive
Santa Clara, CA 95051

per il convertitore A/D ADC0804;

Signetics Corporation
811 East Arques Avenue
Sunnyvale, CA 94086

per il convertitore D/A NE5018.

I circuiti integrati ed i componenti sono prodotti da molti costruttori, per cui vi consigliamo di leggere le inserzioni pubblicitarie che compaiono in gran numero nelle ultime pagine di *Radio-Electronics*, *Popular Electronics*, *Kilobaud Microcomputing* ed altre riviste di elettronica. Noi, per parte nostra, ci siamo sforzati di utilizzare, ogni volta che era possibile, componenti standard.

APPENDICE C

DATI TECNICI SUL MICROPROCESSORE 6502

Le pagine seguenti contengono informazioni di carattere tecnico sul microprocessore 6502. Queste informazioni sono tratte dal *1980 Component Data Catalog*, edito dalla MOS Technology, Inc., 950 Rittenhouse Rd., Norristown, PA 19403. Per informazioni più esaurienti sul processore 6502 e la famiglia di componenti associata, vi consigliamo di richiedere direttamente al costruttore un catalogo completo.

Il chip 6502 è prodotto anche dalle seguenti ditte:

Rockwell International
3310 Miraloma Avenue
Anaheim, CA 92803

Synertek, Inc.
3001 Stender Way
Santa Clara, CA 95051

Questi costruttori potranno fornirvi anche informazioni sul chip 6502 e sui dispositivi relativi.

MCS6500 Microprocessors

- **Single +5V Supply**
- **N-Channel, Silicon-Gate, Depletion-Load Technology**
- **8-Bit Parallel Processing**
- **56 Instructions**
- **Decimal and Binary Arithmetic**
- **13 Addressing Modes**
- **Programmable Stack Pointer and Variable-Length Stack**
- **Usable With Any Type or Speed Memory**
- **1 or 2 MHz Operation**
- **Pipelined Architecture**

DESCRIPTION

The MCS6500 Series microprocessors represent the first totally software-compatible microprocessor family. This family of products includes a range of software-compatible microprocessors which provide a selection of addressable memory range, interrupt input options and on-chip clock oscillators and drivers. All of the microprocessors in the MCS6500 group are software-compatible within the group and are bus compatible with the M6800 product offering.

The family includes five microprocessors with on-board clock oscillators and drivers and four microprocessors driven by external clocks. The on-chip clock versions are aimed at high-performance, low-cost applications where single-phase inputs, crystal or RC inputs provide the time base. The external clock versions are geared for multi-processor system applications where maximum timing control is mandatory. All versions of the microprocessor are available in 1 MHz and 2 MHz ("A" suffix on product numbers) maximum operating frequencies.

MEMBERS OF THE FAMILY

Part Numbers							
Plastic	Ceramic	Clocks	Pins	$\overline{IRQ}$	$\overline{NMI}$	RDY	Addressing
MCS6502	MCS6502	On-Chip	40	√	√	√	16 (64 K)
MCS6503	MCS6503	"	28	√	√		12 (4 K)
MCS6504	MCS6504	"	28	√			13 (8 K)
MCS6505	MCS6505	"	28	√		√	12 (4 K)
MCS6506	MCS6506	"	28	√			12 (4 K)
MCS6507	MCS6507	"	28			√	13 (8 K)
MCS6512	MCS6512	External	40	√	√	√	16 (64 K)
MCS6513	MCS6513	"	28	√	√		12 (4 K)
MCS6514	MCS6514	"	28	√			13 (8 K)
MCS6515	MCS6515	"	28	√		√	12 (4 K)

PIN FUNCTIONS

Clocks (Φ1 and Φ2)

The MCS651X requires a two-phase, non-overlapping clock that runs at the V_{CC} voltage level.

The MCS650X clocks are supplied with an internal clock generator. The frequency of these clocks is externally controlled. Details of this feature are discussed in the MCS6502 portion of this data sheet.

Address Bus (A0-A15)

(See sections on each processor for respective address lines on those devices).

These outputs are TTL-compatible, capable of driving one standard TTL load and 130pF.

Data Bus (D0-D7)

Eight pins are used for the data bus. This is a bi-directional bus, transferring data to and from the device and peripherals. The outputs are three-state buffers capable of driving one standard TTL load and 130pF.

Data Bus Enable (DBE)

This TTL-compatible input allows external control of the three-state data output buffers and will enable the microprocessor bus driver when in the high state. In normal opera-

tion, DBE would be driven by the phase two ($\Phi 2$) clock, thus allowing data input from microprocessor only during $\Phi 2$. During the read cycle, the data bus drivers are internally disabled, becoming essentially an open circuit. To disable data bus drivers externally, DBE should be held low.

Ready (RDY)
This input signal allows the user to single-cycle the microprocessor on all cycles except write cycles. A negative transition to the low state during or coincident with phase one ($\Phi 1$) will halt the microprocessor with the output address lines reflecting the current address being fetched. This condition will remain through a subsequent phase two ($\Phi 2$) in which the Ready signal is low. This feature allows microprocessor interfacing with low-speed PROMS as well as fast (max. 2 cycle) Direct Memory Access (DMA). If Ready is low during a write cycle, it is ignored until the following read operation.

Interrupt Request ($\overline{IRQ}$)
This TTL-compatible signal requests that an interrupt sequence begin within the microprocessor. The microprocessor will complete the current instruction being executed before recognizing the request. At that time, the interrupt mask bit in the Status Code Register will be examined. If the interrupt mask flag is not set, the microprocessor will begin an interrupt sequence. The Program Counter and Processor Status Register are stored in the stack. The microprocessor will then set the interrupt mask flag high so that no further interrupts may occur. At the end of this cycle, the program counter low will be loaded from address FFFE, and program counter high from location FFFF, transferring program control to the memory vector located at these addresses. The RDY signal must be in the high state for any interrupt to be recognized. A 3KΩ external resistor should be used for proper wire-OR operation.

Non-Maskable Interrupt ($\overline{NMI}$)
A negative-going edge on this input requests that a nonmaskable interrupt sequence be generated within the microprocessor.

$\overline{NMI}$ is an unconditional interrupt. Following completion of the current instruction, the sequence of operations defined for $\overline{IRQ}$ will be performed, regardless of the state of the interrupt mask flag. The vector address loaded into the program counter, low and high, are locations FFFA and FFFB respectively, transferring program control to the memory vector located at these addresses. The instructions loaded at these locations cause the microprocessor to branch to a non-maskable interrupt routine in memory.
$\overline{NMI}$ also requires an external 3KΩ register to V_{CC} for proper wire-OR operations.

Inputs $\overline{IRQ}$ and $\overline{NMI}$ are hardware interrupts lines that are sampled during $\Phi 2$ and will begin the appropriate interrupt routine on the $\Phi 1$ following the completion of the current instruction.

Set Overflow Flag (S.O.)
A NEGATIVE-going edge on this input sets the overflow bit in the Status Code Register. This signal is sampled on the trailing edge of $\Phi 1$.

SYNC
This output line is provided to identify those cycles during which the microprocessor is doing an OP CODE fetch. The SYNC line goes high during $\Phi 1$ of an OP CODE fetch and stays high for the remainder of that cycle. If the RDY line is pulled low during the $\Phi 1$ clock pulse in which SYNC went high, the processor will stop in its current state and will remain in the state until the RDY line goes high. In this manner, the SYNC signal can be used to control RDY to cause single instruction execution.

Reset
This input is used to reset or start the microprocessor from a power down condition. During the time that this line is held low, writing to or from the microprocessor is inhibited. When a positive edge is detected on the input, the microprocessor will immediately begin the reset sequence.

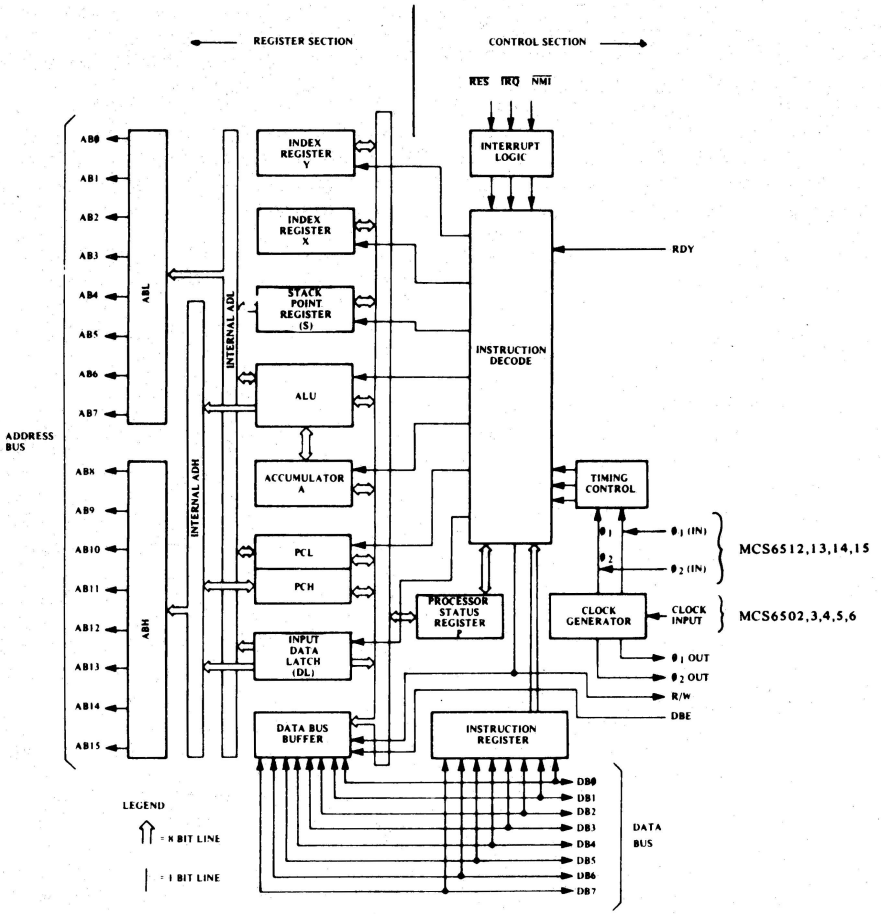
After a system initialization time of six clock cycles, the mask interrupt flag will be set and the microprocessor will load the program counter from memory vector locations

FFFC and FFFD. This is the start location for program control.

After V_{CC} reaches 4.75 volts in a power up routine, reset must be held low for at least two clock cycles. At this thime the R/W and (SYNC) signal will become valid.

When the reset signal goes high following these two clock cycles, the microprocessor will proceed with the normal reset procedure detailed above.

INTERNAL ARCHITECTURE



NOTES

1. Clock Generator is not included on MCS6512,13,14,15
2. Addressing Capability and control options vary with each of the MCS6500 Products.

INSTRUCTION SET
ALPHABETICAL SEQUENCE

ADC Add Memory to Accumulator with Carry
AND "AND" Memory with Accumulator
ASL Shift left One Bit (Memory or Accumulator)

BCC Branch on Carry Clear
BCS Branch on Carry Set
BEQ Branch on Result Zero
BIT Test Bits in Memory with Accumulator
BMI Branch on Result Minus
BNE Branch on Result not Zero
BPL Branch on Result Plus
BRK Force Break
BVC Branch on Overflow Clear
BVS Branch on Overflow Set

CLC Clear Carry Flag
CLD Clear Decimal Mode
CLI Clear Interrupt Disable Bit
CLV Clear Overflow Flag
CMP Compare Memory and Accumulator
CPX Compare Memory and Index X
CPY Compare Memory and Index Y

DEC Decrement Memory by One
DEX Decrement Index X by One
DEY Decrement Index Y by One

EOR "Exclusive-or" Memory with Accumulator

INC Increment Memory by One
INX Increment Index X by One
INY Increment index Y by One

JMP Jump to New Location
JSR Jump to New Location Saving Return Address

LDA Load Accumulator with Memory
LDX Load Index X with Memory
LDY Load index Y with Memory
LSR Shift One Bit Right (Memory or Accumulator)

NOP No Operation

ORA "OR" Memory with Accumulator

PHA Push Accumulator on Stack
PHP Push Processor Status on Stack
PLA Pull Accumulator from Stack
PLP Pull Processor Status from Stack

ROL Rotate One Bit Left (Memory or Accumulator)
ROR Rotate One Bit Right (Memory or Accumulator)
RTI Return from Interrupt
RTS Return from Subroutine

SBC Subtract Memory from Accumulator with Borrow
SEC Set Carry Flag
SED Set Decimal Mode
SEI Set Interrupt Disable Status
STA Store Accumulator in Memory
STX Store Index X in Memory
STY Store Index Y in Memory
TAX Transfer Accumulator to Index X
TAY Transfer Accumulator to Index Y
TSX Transfer Stack Pointer to Index X
TXA Transfer Index X to Accumulator
TXS Transfer Index X to Stack Pointer
TYA Transfer Index Y to Accumulator

ADDRESSING MODES

Accumulator Addressing. This form of addressing is represented with a one-byte instruction, implying an operation on the accumulator.

Immediate Addressing. In immediate addressing the operand is contained in the second byte of the instruction, with no further memory addressing required.

Absolute Addressing. In absolute addressing, the second byte of the instruction specifies the eight low-order bits of the effective address while the third byte specifies the eight high-order bits. Thus, the absolute addressing mode allows access to the entire 65K bytes of addressable memory.

Zero Page Addressing. The zero page instructions allow for shorter code and execution times by only fetching the second byte of the instruction and assuming a zero high-address byte. Careful use of the zero page can result in significant increase in code efficiency.

Indexed Page Addressing. (X, Y indexing) — This form of addressing is used in conjunction with the index register and is referred to as "Zero Page, X" or "Zero Page, Y". The effective address is calculated by adding

the second byte to the contents of the index register. Since this is a form of "Zero Page" addressing, the content of the second byte references a location in page zero. Additionally due to the "Zero Page" addressing nature of this mode, no carry is added to the high order 8 bits of memory and crossing of page boundaries does not occur.

Indexed Absolute Addressing. (X, Y indexing) — This form of addressing is used in conjunction with X and Y index register and is referred to as "Absolute, X", and "Absolute, Y". The effective address is formed by adding the contents of X or Y to the address contained in the second and third bytes of the instruction. This mode allows the index register to contain the index or count value and the instruction to contain the base address. This type of indexing allows any location referencing and the index to modify multiple fields resulting in reduced coding and execution time.

Implied Addressing. In the implied addressing mode, the address containing the operand is implicitly stated in the operation code of the instruction.

Relative Addressing. Relative addressing is used only with branch instructions and establishes a destination for the conditional branch. The second byte of the instruction becomes the operand which is an offset added to the contents of the lower eight bits of the program counter when the counter is set at the next instruction. The range of the offset is −128 to +127 bytes from the next instruction.

Indexed Indirect Addressing. In indexed indirect addressing (referred to as Indirect, X), the second byte of the instruction is added to the contents of the X index register, discarding the carry. The result of this addition points to a memory location on page zero whose contents is the low-order eight bits of the effective address. The next memory location in page zero contains the high-order eight bits of the effective address. Both memory locations specifying the high and low-order bytes of the effective address must be in page zero.

Indirect Indexed Addressing. In indirect indexed addressing (referred to as Indirect, Y), the second byte of the instruction points to a memory location in page zero. The contents on this memory location is added to the contents of the Y index register, the result being the low-order eight bits of the effective address. The carry from this addition is added to the contents of the next page zero memory location, the result being the high-order eight bits of the effective address.

Absolute Indirect. The second byte of the instruction contains the low-order eight bits of a memory location. The high-order eight bits of that memory location is contained in the third byte of the instruction. The contents of the fully specified memory location is the low-order byte of the effective address. The next memory location contains the high-order byte of the effective address which is loaded into the 16-bit program counter.

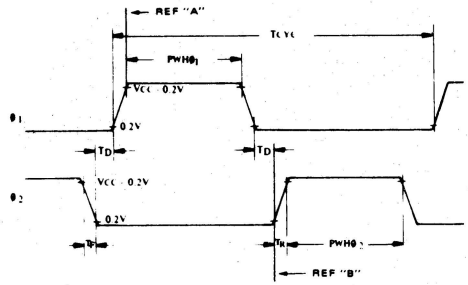
ABSOLUTE MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 to +7.0	Vdc
Input Voltage	V_{IN}	−0.3 to +7.0	Vdc
Operating Temperature	T_A	0 to +70	°C
Storage Temperature	T_{STG}	−55 to +150	°C

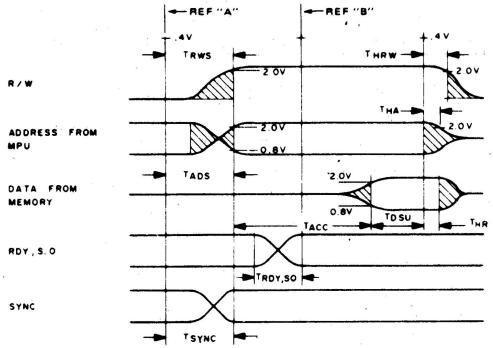
CAUTION

This device contains input protection against damage due to high static voltages or electric field; however, precautions should be taken to avoid application of voltages higher than the maximun rating.

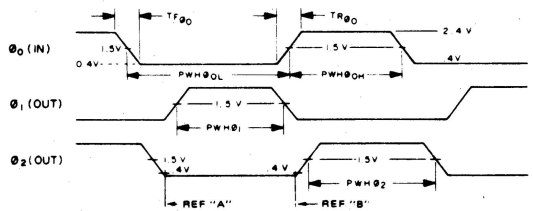
CLOCK TIMING—MCS6512, 13, 14, 15



TIMING FOR READING DATA FROM MEMORY OR PERIPHERALS



TIMING FOR WRITING DATA TO MEMORY OR PERIPHERALS



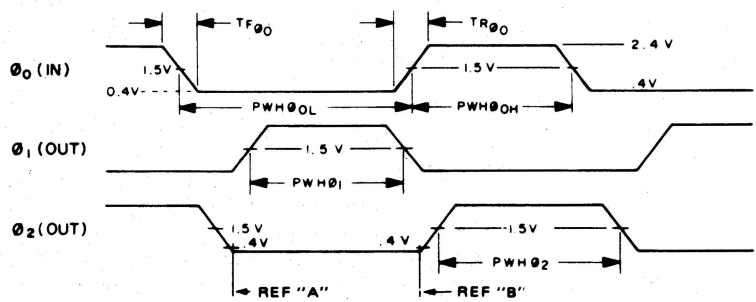
NOTE
"REF" means Reference Points on clocks.

ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0V ± 5%, V_{SS} = 0, T_A = 25°C)
$\emptyset_1$, $\emptyset_2$ applies to MCS6512, 13, 14, 15, $\emptyset_{o\,(in)}$ applies to MCS6502, 03, 04, 05 and 06

Symbol	Parameter	Min	Typ	Max	Unit	Test Condition
V_{IH}	Input High Voltage	V_{SS} + 2.4 V_{CC} − 0.2		V_{CC} V_{CC} + 0.25	Vdc	Logic, $\emptyset_{o\,(in)}$ $\emptyset_1$, $\emptyset_2$
V_{IL}	Input Low Voltage	V_{SS} − 0.3 V_{SS} − 0.3		V_{SS} + 0.4 V_{SS} + 0.2	Vdc	Logic, $\emptyset_{o\,(in)}$ $\emptyset_1$, $\emptyset_2$
V_{IHT}	Input High Threshold Voltage	V_{SS} + 2.0			Vdc	$\overline{RES}$, $\overline{NMI}$, RDY, $\overline{IRQ}$, Data, S.O.
V_{ILT}	Input Low Threshold Voltage			V_{SS} + 0.8	Vdc	$\overline{RES}$, $\overline{NMI}$, RDY, $\overline{IRQ}$, Data, S.O.
I_{IN}	Input Leakage Current			 2.5 100 10.0	 μA μA μA	(V_{IN} = 0 to 5.25V, V_{CC} = 0) Logic (Excl. RDY, S.O.) $\emptyset_1$, $\emptyset_2$ $\emptyset_{o\,(in)}$
I_{TSI}	Three-State (Off State) Input Current			10	μA	(V_{IN} = 0.4 to 2.4V, V_{CC} = 5.25V) Data Lines
V_{OH}	Output High Voltage	V_{SS} + 2.4			Vdc	(I_{LOAD} = −100μAdc, V_{CC} = 4.75V) SYNC, Data, A0-A15, R/W
V_{OL}	Output Low Voltage			V_{SS} + 0.4	Vdc	(I_{LOAD} = 1.6mAdc, V_{CC} = 4.75V) SYNC, Data, A0-A15, R/W
P_D	Power Dissipation		.25	.70	W	
C C_{IN} C_{OUT} $C_{\emptyset o\,(in)}$ $C_{\emptyset 1}$ $C_{\emptyset 2}$	Capacitance		 30 50	 10 15 12 50 50 80	pF	(V_{IN} = 0, T_A = 25°C, f = 1MHz) Logic Data A0-A15, R/W, SYNC $\emptyset_{o\,(in)}$ $\emptyset_1$ $\emptyset_2$

NOTE
$\overline{IRQ}$ and $\overline{NMI}$ require 3K pull-up resistors

CLOCK TIMING—MCS6502, 03, 04, 05, 06



1 MHz TIMING

CLOCK TIMING—MCS6512, 13, 14, 15

Symbol	Characteristic		Min	Typ	Max	Unit
T_{CYC}	Cycle Time		1000			nsec
PWH ϕ1 PWH ϕ2	Clock Pulse Width (Measured at V_{CC} − 0.2 V)	ϕ1 ϕ2	430 470			nsec
T_F	Fall Time (Measured from 0.2 V to V_{CC} − 0.2 V)				25	nsec
T_D	Delay Time Between Clocks (Measured at 0.2 V)		0			nsec

CLOCK TIMING—MCS6502, 03, 04, 05, 06

Symbol	Characteristic	Min	Typ	Max	Unit
T_{CYC}	Cycle Time	1000			ns
PWHϕ_o	$\phi_{o\,(IN)}$ Pulse Width (measured at 1.5 V)	460		520	ns
TRϕ_o, TFϕ_o	$\phi_{o\,(IN)}$ Rise, Fall Time			10	ns
T_D	Delay Time Between Clocks (measured at 1.5 V)	5			ns
PWHϕ_1	$\phi_{1\,(OUT)}$ Pulse Width (measured at 1.5 V)	PWHϕ_{oL}−20		PWHϕ_{oL}	ns
PWHϕ_2	$\phi_{1\,(OUT)}$ Pulse Width (measured at 1.5 V)	PWHϕ_{oH}−40		PWHϕ_{oH}−10	ns
T_R, T_F	$\phi_{1\,(OUT)}$, $\phi_{2\,(OUT)}$ Rise, Fall Time (measured .8 V to 2.0 V) (Load = 30pF + 1 TTL)			25	ns

READ/WRITE TIMING

Symbol	Characteristic	Min	Typ	Max	Unit
T_{RWS}	Read/Write Setup Time From MCS6500		100	300	ns
T_{ADS}	Address Setup Time From MCS6500		100	300	ns
T_{ACC}	Memory Read Access Time			575	ns
T_{DSU}	Data Stability Time Period	100			ns
T_{HR}	Data Hold Time—Read	10			ns
T_{HW}	Data Hold Time—Write	30	60		ns
T_{MDS}	Data Setup Time From MCS6500		150	200	ns
T_{RDY}	RDY, S.O. Setup Time	100			ns
T_{SYNC}	SYNC Setup Time From MCS6500			350	ns
T_{HA}	Address Hold Time	30	60		ns
T_{HRW}	R/W Hold Time	30	60		ns

2 MHz TIMING

CLOCK TIMING—MCS6512, 13, 14, 15, 16

Symbol	Characteristic		Min	Typ	Max	Unit
T_{CYC}	Cycle Time		500			nsec
PWH ϕ1 PWH ϕ2	Clock Pulse Width (Measured at V_{CC} − 0.2 V)	ϕ1 ϕ2	215 235			nsec
T_F	Fall Time (Measured from 0.2 V to V_{CC} − 0.2 V)				12	nsec
T_D	Delay Time Between Clocks (Measured at 0.2 V)		0			nsec

CLOCK TIMING—MCS6502, 03, 04, 05, 06

Symbol	Characteristic	Min	Typ	Max	Unit
T_{CYC}	Cycle Time	500			ns
$PWH\phi_o$	$\phi_{o\,(IN)}$ Pulse Width (measured at 1.5 V)	240		260	ns
$TR\phi_o$, $TF\phi_o$	$\phi_{o\,(IN)}$ Rise, Fall Time			10	ns
T_D	Delay Time Between Clocks (measured at 1.5 V)	5			ns
$PWH\phi_1$	$\phi_{1\,(OUT)}$ Pulse Width (measured at 1.5 V)	$PWH\phi_{oL}-20$		$PWH\phi_{oL}$	ns
$PWH\phi_2$	$\phi_{2\,(OUT)}$ Pulse Width (measured at 1.5 V)	$PWH\phi_{oH}-40$		$PWH\phi_{oH}-10$	ns
T_R, T_F	$\phi_{1\,(OUT)}$, $\phi_{2\,(OUT)}$ Rise, Fall Time (measured .8 V to 2.0 V) (Load = 30pF + 1 TTL)			25	ns

READ/WRITE TIMING

Symbol	Characteristic	Min	Typ	Max	Unit
T_{RWS}	Read/Write Setup Time From MCS6500A		100	150	ns
T_{ADS}	Address Setup Time From MCS6500A		100	150	ns
T_{ACC}	Memory Read Access Time			300	ns
T_{DSU}	Data Stability Time Period	50			ns
T_{HR}	Data Hold Time – Read	10			ns
T_{HW}	Data Hold Time – Write	30	60		ns
T_{MDS}	Data Setup Time From MCS6500A		75	100	ns
T_{RDY}	RDY, S.O. Setup Time	50			ns
T_{SYNC}	SYNC Setup Time From MCS6500A			175	ns
T_{HA}	Address Hold Time	30	60		ns
T_{HRW}	R/W Hold Time	30	60		ns

APPENDICE D

PARTI DELLA SCHEDA DI PROVA PER L'INTERFACCIAMENTO CON L'APPLE

Parti necessarie per costruire la scheda di prova per l'interfacciamento con l'Apple:

IC 1 e 7	rete di resistori a 16 pins, otto resistori indipendenti da 1000 ohm
IC 2 e 6	interruttore DIP ad otto posizioni (ON-OFF)
IC 3, 4 e 5	integrato SN74LS85: comparatore quadruplo. *Non si deve sostituire con l'*SN74L85!
IC 8	integrato SN74LS20: NAND doppio a quattro ingressi
IC 9	SN74365 o DM8095: buffer a tre stati
IC 10 e 11	8216 (Intel o equivalente): buffer bus non invertente
IC 12	integrato SN74154: decodificatore
IC 13	integrato SN7404: inverter
IC 14	integrato SN74123 o SN74LS123: monostabile doppio
IC 15	LM319N (package di 14 pins): comparatore doppio
IC 16, 17, 18 e 20	zoccoli a 16 pins di buona qualità (516-AG-10D della Augat, o equivalenti)
IC 19	zoccolo ad 8 pins di buona qualità (508-AG-10D della Augat, o equivalente)
D1-D4	1N4001*: diodi da 50 piv ed 1 ampère
D5	LED giallo
D6	LED rosso
D7	LED verde
D8 e D9	1N4148 o 1N4154: diodi per segnali piccoli
R1 ed R8	resistore da 1000 ohm ed 1/4 watt

R2 ed R3	resistori da 220 ohm ed 1/4 watt
R4 ed R5	resistori da 47 K ed 1/4 watt
R6	resistore da 3900 ohm ed 1/4 watt
R7	resistore da 2200 ohm ed 1/4 watt
C1	condensatore elettrolitico da 2200 µF e 16 V dcw*
C2, C4 e C5	condensatori ceramici a disco da 0,1 µF e 50 volts
C3 e C6	condensatori elettrolitici al tantalio da 1 µF e 35 V dcw
C7 e C8	condensatori elettrolitici da 3,3 µF e 50 V dcw
VR	LM309K*: regolatore di tensione da 5 volts ed 1 amp
P1	connettore Molex a 6 pins ad angolo retto (PN 09-75-1061); facoltativo Richiede 1 alloggiamento femmina (PN 09-50-7061) e 6 pins per connettori (PN 08-50-0106 o 08-50-0108)
P2	connettore di testata a 40 pins ad angolo retto (923875R della AP Products, o equivalente)
T1	trasformatore da 12,6 V ed 1 amp
Varie	11 zoccoli per IC da 16 pins
	3 zoccoli per IC da 14 pins
	1 zoccolo per IC da 24 pins
	Cavo d'assemblaggio: testata con 40 pins ad un'estremità ed un connettore piatto a 40 pins all'altra estremità, rivolti nella stessa direzione
	Zoccolo per la scheda di prova a circuito forato SK-10 della Superstrip, o equivalente, 4 viti a testa piatta 4-40 x 5/8, 4 rondelle di bloccaggio dentate internamente del n. 4, 4 dadi esagonali del n. 4
	Dissipatore di calore per VR, 2 viti 4-40 x 1/2, 2 rondelle di bloccaggio dentate internamente del n. 4, 2 dadi esagonali del n. 4, isolante di mica, grasso termico (facoltativo)
	Filo elettrico con spina

I pezzi contraddistinti dal segno * non sono necessari quando per alimentare il sistema si usa un'alimentazione a +5 volts.

APPENDICE E

DISEGNO DELLA SCHEDA A CIRCUITO STAMPATO

In quest'appendice riproduciamo il disegno del quale ci si può servire per costruire il circuito stampato della scheda di prova per l'interfacciamento con l'Apple. Il disegno è ridotto rispetto alle dimensioni reali, per cui, per usarlo, occorre ingrandirlo. Fatevi fare in un laboratorio fotografico uno sviluppo in negativo fortemente contrastato, oppure in positivo, a seconda del processo che adopererete. La riga nera grossa e lunga che compare in tutti e tre gli schemi va ingrandita fino a raggiungere la lunghezza di quattro (4) pollici (10,16 cm.). Il fotografo dovrà curare l'ingrandimento in modo che la pellicola raggiunga esattamente le dimensioni giuste per la scheda a circuito stampato. Potete a questo punto decidere di non fare la sovraimpressione dei componenti; noi comunque l'abbiamo introdotta come guida per la sistemazione dei vari pezzi.

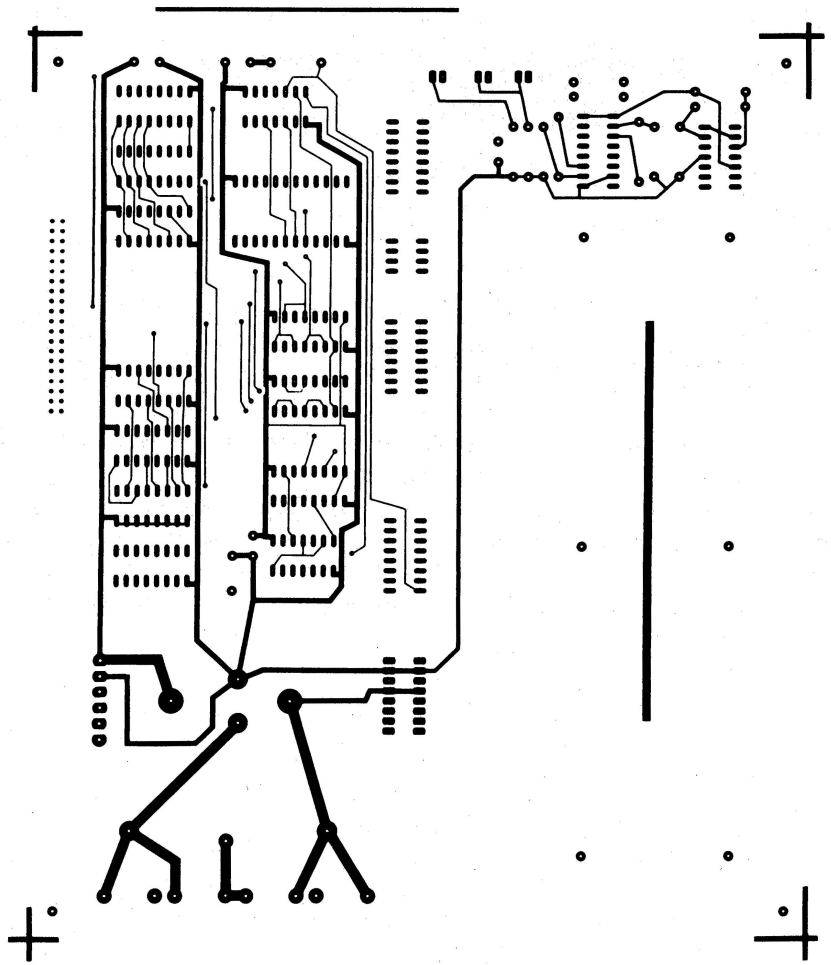


Figura E.1 — Disegno del circuito stampato relativo al lato componenti della scheda d'interfaccia (dritto).

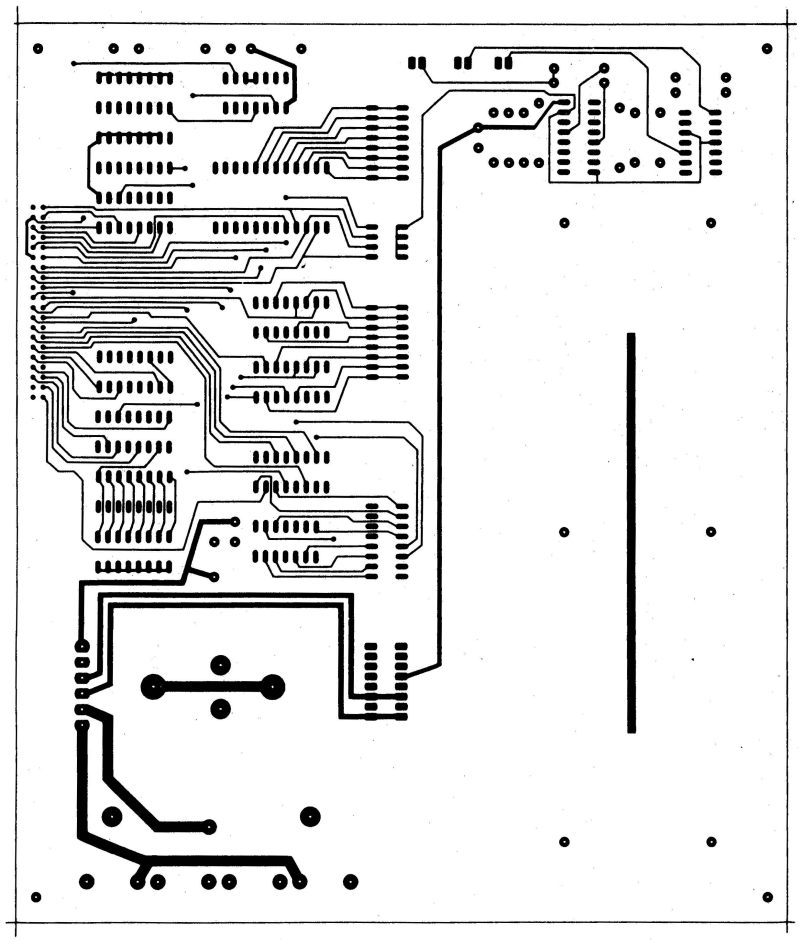


Figura E.2 — Disegno del circuito stampato relativo al lato saldature della scheda d'interfaccia (rovesciato).

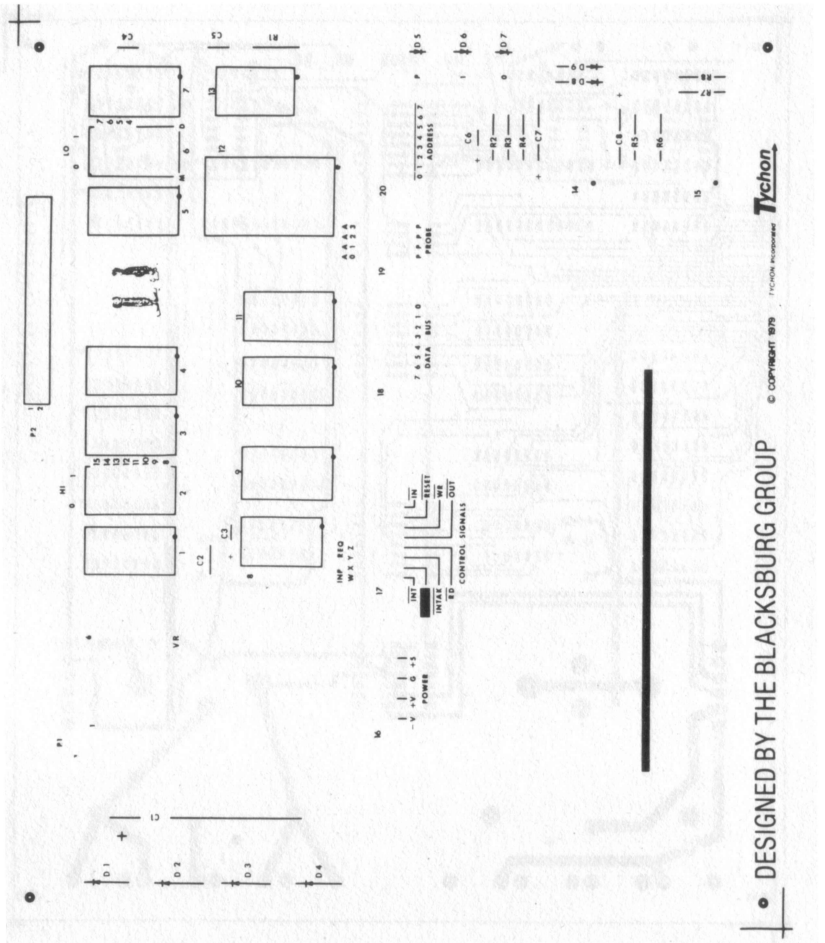


Figura E.3 — Sovrapposizione della nomenclatura relativa alla scheda d'interfaccia (dritta).

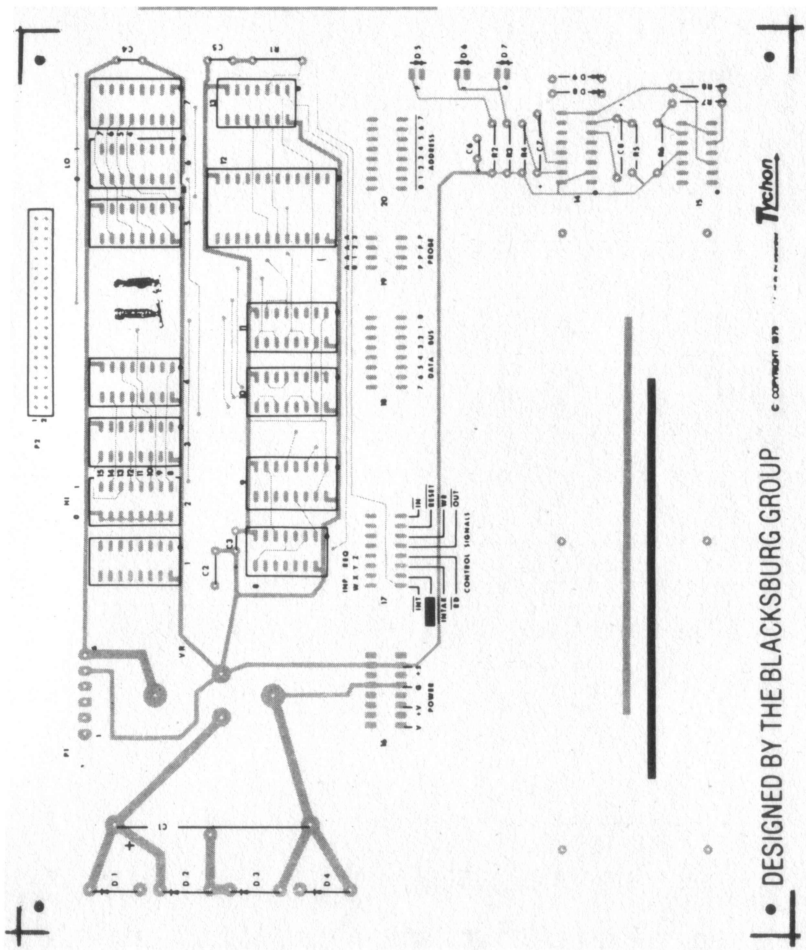


Figura E.4 — Sovrapposizione della nomenclatura dei componenti sul lato componenti della scheda. Utilizzabile come guida nella sistemazione dei componenti.

Scopo di questo libro è svelare i segreti dell'interfacciamento dell'Apple. Si parla dei segnali di controllo, illustrandone l'uso. Viene spiegato come l'Apple identifica o indirizza dei dispositivi esterni mediante due istruzioni d'uso generale: PEEK e POKE. Ci si sofferma sulle loro modalità operative e sull'utilizzo di tutta una serie di circuiti che possono essere utilizzati per identificare specifici dispositivi di ingresso/uscita (I/O). Si analizzano i flag come pure i circuiti corrispondenti usati in pratica nei dispositivi esterni.

Si affronta come l'Apple è in grado di trasferire le informazioni a/da i dispositivi esterni attraverso il bus dati bidirezionali: i principali circuiti usati per le porte di ingresso e le porte di uscita sono descritti dettagliatamente. Vengono presentati, quindi, dei reali e sperimentati circuiti in modo che il lettore possa in breve tempo utilizzare i numerosi esempi per progettare autonomamente dei dispositivi di interfacciamento.

Questo, presentando contemporaneamente il software (in Basic) relativo. Per rendere più rapida la vostra attività di progettazione e analisi dei circuiti è stata realizzata, e descritta minutamente, una scheda di prova (breadboard). Tutti questi elementi vi consentono di:

1) controllare dispositivi elettronici ed elettromeccanici; 2) tenere sotto controllo eventi esterni, come temperatura, pressione, livelli di liquidi, soglie luminose, ecc.; 3) comunicare con altri calcolatori, modem, stampanti seriali ed altri dispositivi d'interfaccia.

Seconda edizione del libro INTERFACCIAMENTO DELL'APPLE.

L. 20.000

Cod. CC 420 ISBN 88-7056-647-1

ISBN 88-7056-647-1

9 788870 566475

GRUPPO EDITORIALE JACKSON 420

Jonathan A. Titus David G. Larsen
Christopher A. Titus

Tecniche di interfacciamento dell'APPLE