

KIM-1

MODULO MICROCOMPUTER

Manuale d'uso

MARIO STAVOLTA EDITORE

Pubblicazione Numero 6500-15B

KIM - 1

MODULO MICROCOMPUTER

Manuale d'uso

**SKYLAB SRL
Via M. Gioia, 66
20125-MILANO**

Mario Stavolta Editore - Pordenone

TUTTI I DIRITTI RISERVATI

I N D I C E

PREMESSA	Il Vostro modulo microcalcolatore KIM-1	1
CAPITOLO 1	AVVIAMENTO	3
	1.1 Sommario delle parti	3
	1.2 Precauzioni	3
	1.3 Primi passi	4
	1.4 Proviamo un semplice programma	7
	1.5 Aggiunta di un registratore a nastro	12
	1.6 Collegamento con la telescrivente	17
CAPITOLO 2	IL SISTEMA KIM-1	21
	2.1 Descrizione del sistema	21
	2.2 Distribuzione delle locazioni di memoria	34
	2.3 Programmi operativi del KIM-1	40
CAPITOLO 3	I COMANDI OPERATIVI DEL KIM-1	43
	3.1 Uso della tastiera e del display	43
	3.2 Uso del registratore	46
	3.3 Uso della telescrivente	49
CAPITOLO 4	UN'APPLICAZIONE PRATICA	54
	4.1 Definizione dell'interfaccia	54
	4.2 Scrittura del programma	57
	4.3 Introduzione del programma	65
	4.4 Esecuzione del programma	66
	4.5 Controllo e modifica del programma	66

CAPITOLO 5	ESPANSIONE DEL SISTEMA	70
	5.1 Espansione delle memorie e degli I/O	70
	5.2 Conduzione del Vettore di Interrupt	74
CAPITOLO 6	ASSISTENZA E GARANZIA	78
	6.1 Servizio di garanzia	78
	6.2 Servizio fuori garanzia	79
	6.3 Riserva di cambiamenti	79
	6.4 Istruzioni per il trasporto	79
APPENDICE A	Elenco dei componenti	A-1
APPENDICE B	Configurazione circuitale del KIM-1	B-1
APPENDICE C	In caso di difficoltà	C-1
APPENDICE D	Schema di alimentatore	D-1
APPENDICE E	Formato dei dati sul nastro audio	E-1
APPENDICE F	Formato dei dati per il nastro di carta	F-1
APPENDICE G	Caratteristiche del 6502	G-1
APPENDICE H	Caratteristiche del 6530	H-1
APPENDICE I	Programma contenuto come "Monitor" nel KIM-1	I-1

INDICE DELLE FIGURE

CAPITOLO 1	1-1	Modulo KIM-1	5
	1-2	Collegamenti di alimentazione	6
	1-3	Collegamenti per registratore audio	13
	1-4	Connessioni TTY	18
CAPITOLO 2	2-1	Diagramma a blocchi	24
	2-2	Dettaglio del diagramma a blocchi	25
	2-3	Controllo e Temporizzazione	26
	2-4	Memoria RAM 1K x 8	27
	2-5	Tastiera e Display	28
	2-6	Dettaglio della tastiera	29
	2-7	Interfaccia TTY	30
	2-8	Interfaccia per registratore	31
	2-9	Connettore di applicazione	32
	2-10	Connettore di espansione	33
	2-11	Diagrammi dei blocchi di memoria	37
	2-12	Mappa di memoria	38
	2-13	Indirizzi di memoria specializzati	39
	2-14	Carta di flusso	41
CAPITOLO 4	4-1	Collegamento dell'altoparlante	56
	4-2	Lista del programma in linguaggio assemblatore	59
	4-3	Uscita onda quadra	61
	4-4	Tabella dei codici di linguaggio macchina	62
	4-5	Sequenza dei tasti per l'introduzione del programma	65
	4-6	Analisi del programma a passo singolo	69
CAPITOLO 5	5-1	Collegamento con memoria di espansione 4K	72
	5-2	Espansione fino a 65K	73
	5-3	Selezione di vettore	77

P R E M E S S A

IL VOSTRO MODULO MICROCALCOLATORE KIM-1

Lo studio dei microprocessori, allo stato attuale della tecnologia, non può prescindere dall'uso e dalle caratteristiche d'un particolare prodotto.

Questo volume intende riferirsi al sistema KIM-1, prodotto originariamente dalla MOS TECHNOLOGY INC..

Si tratta di un computer digitale completamente operativo, collaudato e molto potente. Tale modulo consente di eliminare tutti i problemi di costruzione e controllo di un sistema a microprocessore, consentendo allo studente di utilizzare il tempo disponibile esclusivamente per studiare come opera il sistema e di cominciare subito ad applicarlo alle sue specifiche aree d'interesse. Infatti, seguendo alcune semplici modalità descritte nel manuale, è possibile cominciare ad adoperare il modulo KIM-1 fin dal primo momento.

Il KIM-1 è stato progettato in modo da fornire una vasta scelta di linee operative: potete azionare il sistema usando solo la tastiera ed il visualizzatore inclusi nel modulo; aggiungere un normale registratore a cassette per la registrazione ed il reinserimento dei programmi; collegare una telescrivente con interfaccia seriale per fornire i comandi dalla tastiera, la stampa dei programmi, la perforazione di un nastro di carta e la sua lettura.

Il cuore del sistema KIM-1 è un microprocessore MCS 6502 che lavora congiunto con due circuiti MCS 6530. Ciascuno di questi fornisce un totale di 1024 bytes di ROM (Read Only Memory, memoria di sola lettura), 64 bytes di RAM (Random Access Memory, memoria di lettura e scrittura), 15 piedini di entrata ed uscita (I/O) ed un temporizzatore degli intervalli (Interval Timer). Permanentemente memorizzati nelle ROM dell'MCS 6530 stanno il programma monitor ed il programma esecutivo sviluppati dalla Mos Technology Inc. per controllare i vari modi di operazione del sistema.

Il KIM-1 costituisce un potente microcomputer da usare nelle applicazioni pratiche. In aggiunta mette a disposizione 15 piedini I/O bidirezionali per il controllo dell'interfaccia nelle specifiche applicazioni. Infine uno dei temporizzatori d'intervallo inseriti nel sistema è disponibile per la generazione dei segnali secondo la cadenza richiesta dalle varie applicazioni.

Il sistema KIM-1 è un sistema completo di tutti i componenti già montati e collaudati. Non dovrete preoccuparvi della generazione dei segnali di temporizzazione (comprende un generatore a cristallo da 1 MHz), della logica di interfaccia o di livello tra i componenti del sistema, o dei circuiti d'interfaccia verso i dispositivi periferici.

Lo studente deve limitarsi a connettere le prescritte tensioni di alimentazione alle connessioni indicate.

CAPITOLO 1

A V V I A M E N T O

Questo capitolo è inteso come guida ai primi importanti passi nel meraviglioso mondo dei microprocessori. Inizialmente vi chiederemo di effettuare determinate operazioni senza spiegarvene la ragione. Le spiegazioni complete per ciascun procedimento operativo, saranno fornite nei capitoli successivi.

1.1 SOMMARIO DELLE PARTI

Dopo avere disimballato il vostro microcomputer, avrete notato i seguenti oggetti:

3 libri: KIM-1 Manuale d'uso

Hardware Manual

Programming Manual

1 prontuario delle istruzioni

1 poster del sistema

1 modulo KIM-1

1 connettore

1 imballo-custodia del modulo

6 piedini di gomma

1.2 PRECAUZIONI

Attenzione: il KIM-1 comprende un certo numero di circuiti integrati MOS, provvisti di circuiti di protezione atti a prevenire danneggiamenti dovuti all'accidentale applicazione di alte tensioni ai piedini. Devono essere però prese le normali precauzioni per evitare l'applicazion

ne di scariche statiche ad alta tensione ai piedini dei dispositivi MOS.

Subito dopo aver disimballato il vostro modulo, dovete prendere la abitudine di usare le seguenti precauzioni:

1. Scaricare ogni carica elettrostatica presente nel vostro corpo toccando una connessione di massa prima di toccare qualsiasi parte del modulo. Questa precauzione è importante specialmente se lavorate in ambiente col pavimento a mouquette.
2. Assicuratevi che i saldatori e gli equipaggiamenti di controllo siano appropriatamente messi a massa e non diano origine ad alte tensioni di livello pericoloso.

Sul KIM-1 noterete un potenziometro. Questo è stato regolato in fabbrica per assicurare la corretta operazione dell'interfaccia della audiocassetta. Non dovrebbe essere mai necessario cambiare la posizione di questo potenziometro.

1.3 PRIMI PASSI

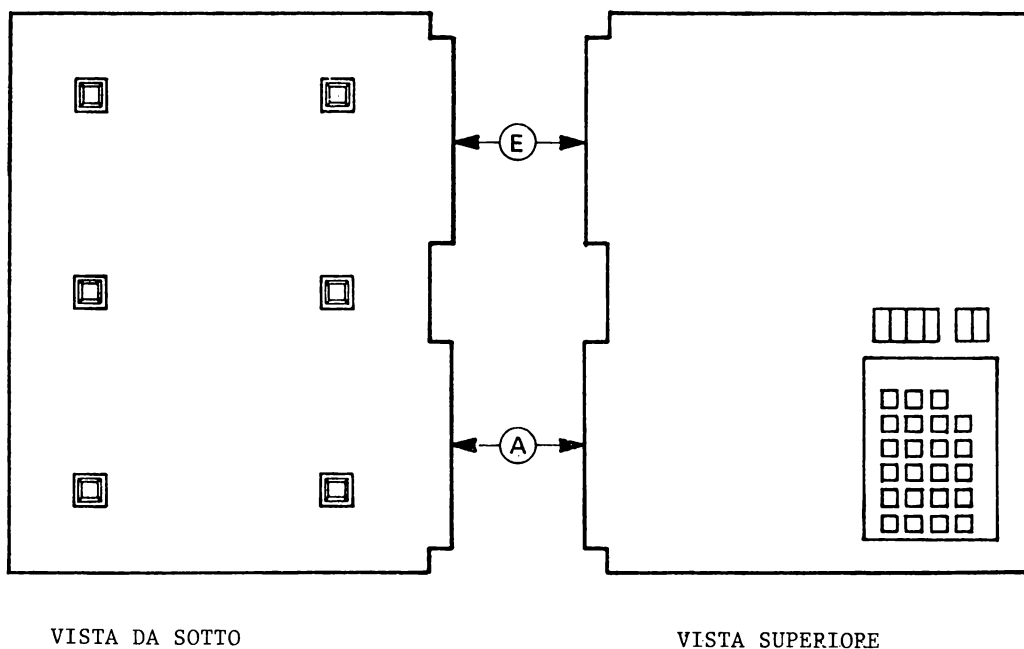
Applicate i piedini di gomma che vanno fatti aderire sul fondo del modulo come illustrato.

Essi servono sia per sollevare il modulo dal piano di lavoro che per dare un supporto meccanico al modulo quando si premono i tasti.

Sistamate il modulo in modo che la tastiera sia alla vostra destra in basso ed osservate che le piste dei connettori siano alla vostra sinistra.

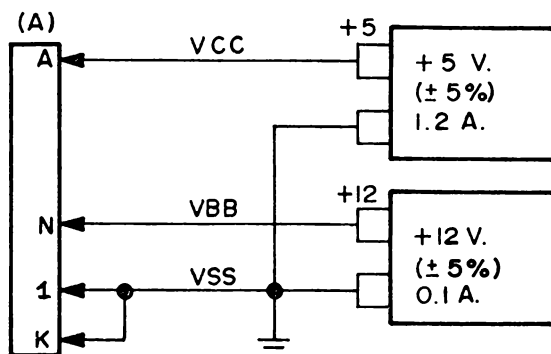
Il connettore a sinistra in basso si chiama connettore di applicazione (A).

Il connettore a sinistra in alto dovrà essere usato per future espansioni e si chiama connettore d'espansione (E).



MODULO KIM - 1

Figura 1.1



COLLEGAMENTI DI ALIMENTAZIONE

Figura 1.2

Eseguite sul connettore fornito i collegamenti illustrati nella figura 1.2 e inseritelo nella espansione di connessione (A), controllando la corretta orientazione.

L'alimentazione a +12 V è necessaria solo se userete un registratore a cassette.

Il cavallotto tra i piedini A-K e Vss (piedino A-1) è essenziale per il funzionamento del sistema. In seguito, se vorrete espandere il sistema, questo cavallotto andrà rimosso e vi diremo cosa dovrete fare del piedino A-K.

Se non avete già disponibili gli adatti alimentatori, potrete costruirveli con poca spesa secondo le indicazioni dell'appendice D. In ogni caso l'alimentatore deve essere stabilizzato per assicurare un corretto funzionamento del sistema e deve essere capace di fornire i livelli di corrente indicati nello schema.

Ora controllate nuovamente le connessioni, accendete l'alimentatore e premete il tasto RS (Reset).

Se le cifre del visualizzatore si illuminano, significa che il sistema è operativo. Altrimenti controllate i collegamenti. Se non si accendono ancora, consultate l'appendice C.

1.4 PROVIAMO UN SEMPLICE PROGRAMMA

Qualora abbiate completato con successo i passi finora indicati, si può provare un semplice programma per mostrare come funziona il sistema ed assicurarvi ulteriormente che tutto funzioni correttamente.

Useremo per questo esempio solo la tastiera ed il visualizzatore montati sul modulo. Nelle successive due sezioni ci occuperemo della telescrivente e delle audiocassette.

Per il primo esempio sommeremo due numeri binari da 8 bit e visualizzeremo il risultato, presupponendo che abbiate familiarità con la rappresentazione esadecimale dei numeri e con le regole generali dell'aritmetica binaria.

Controllate per prima cosa che il deviatore a slitta posto nell'angolo della tastiera sia spinto verso l'interno della tastiera stessa (modo SST in OFF). Ora procedete premendo i tasti nel seguente ordine:

<u>Tasto da premere</u>	<u>Lettura sul display</u>	<u>Passi \neq</u> (numero di operazioni svolte)
AD	xxxx xx	1
0 0 0 2	0002 xx	2
DA	0002 xx	3
1 8	0002 18	4
+ A 5	0003 A5	5
+ 0 0	0004 00	6
+ 6 5	0005 65	7
+ 0 1	0006 01	8
+ 8 5	0007 85	9
+ F A	0008 FA	10
+ A 9	0009 A9	11
+ 0 0	000A 00	12
+ 8 5	000B 85	13
+ F B	000C FB	14
+ 4 C	000D 4C	15
+ 4 F	000E 4F	16
+ 1 C	000F 1C	17

Con le operazioni svolte avete inserito un programma e lo avete memorizzato in RAM, nelle locazioni che vanno da 0002 a 000F.

Prima di procedere oltre, ecco la funzione dei vari tasti speciali della vostra tastiera:

- AD - seleziona l'ingresso di un indirizzo
- DA - seleziona l'ingresso di un dato
- + - incrementa l'indirizzo senza cambiare il modo
 selezionato
- da 0 a F - 16 tasti per la definizione in codice esadecimale
 dell'indirizzo o del dato

Il display è di 6 cifre: le quattro a sinistra servono per mostrare il codice esadecimale di un indirizzo; le due a destra mostrano il codice esadecimale del dato immagazzinato nell'indirizzo indicato. Quindi premendo il tasto AD (passo 1) e 0002 (passo 2) avete scelto il modo per la definizione dell'indirizzo (indirizzamento) e scelto l'indiirizzo 0002 visualizzando le cifre corrispondenti nella sezione sinistra del display (quattro cifre). Se nella tabella abbiamo delle x come indicazione del display, vuol dire che non sappiamo cosa apparirà e che non ce ne curiamo.

Premendo quindi il tasto DA (passo 3) seguito da 18 (passo 4) avete scelto il modo per l'introduzione dei dati ed introdotto il valore 18 che sarà memorizzato nell'indirizzo 0002 già scelto, e visualizzato sulle due cifre a destra del display.

Rimanendo ora nel modo di introduzione dei dati, premete il tasto + seguito da un numero di 2 cifre (passi da 5 a 17): noterete che ogni pressione del tasto + incrementa di 1 l'indirizzo visualizzato.

I tasti esadecimali che seguono la pressione di + continuano ad interessare il campo dati del display.

Questa procedura è di pura convenienza quando si devono riempire un certo numero di successive locazioni di indirizzo.

Se fate qualche errore nel premere i tasti, la correzione è possibile semplicemente reintroducendo il dato fino a che appaiono sul display i numeri corretti.

Il programma introdotto è un semplice anello per addizionare due numeri binari da 8 bit e presentare il risultato sul display. Per un programmatore, la sequenza del programma introdotto potrebbe apparire come segue:

POINTL		= \$FA	
POINTH		= \$FB	
START		= \$1C4F	
0000		VAL1	
0001		VAL2	
0002	18	PROG	CLC
0003	A5 00		LDA VAL1
0005	65 01		ADC VAL2
0007	85 FA		STA POINTL
0009	A9 00		LDA ≠00
000B	85 FB		STA POINTH
000D	4C 4F 1C		JMP START

In altri termini, il programma azzererà il flag di riporto (carry flag) (CLC), caricherà il primo valore VAL1 nell'accumulatore (LDA VAL1) l'addiziona con riporto al secondo valore VAL2 all'accumulatore (ADC VAL2) e memorizzerà il risultato nella locazione POINTL (STA POINTL). Un valore zero è memorizzato in locazione POINTH (LDA ~~≠00~~ e STA POINTH) ed il programma salterà ad un punto definito START (JMP START). Questo programma preimmagazzinato attiverà il display e farà sì che il campo degli indirizzi del display mostri i numeri memorizzati in locazione POINTL e POINTH.

Da notare che il risultato dell'addizione è già stato memorizzato in locazione POINTL.

Nota:

- FLAG = Segnale (bit) che testimonia che un certo evento è successo oppure che un modo di funzionamento è stato selezionato.
- POINTL = E' l'indirizzo della locazione in cui vengono depositati i dati da visualizzare nelle 2 cifre meno significative del display degli indirizzi.
- POINTH = E' lo stesso di POINTL, ma per la visualizzazione nelle 2 cifre più significative.

I codici esadecimali che appaiono vicino al campo degli indirizzi nella sequenza di programma sono esattamente i numeri che voi avete introdotto per memorizzare il programma. Questi ultimi rappresentano i codici in linguaggio macchina.
Per esempio 4C è il codice esadecimale per l'istruzione JMP del micro-processore. I successivi due bytes del programma definiscono 1C4F (START) come indirizzo del salto.

Ora come adesso, non potete ancora avviare il programma in quanto non avete ancora introdotto le due variabili (VAL1 e VAL2).

Provate ora un esempio pratico:

<u>Tasto da premere</u>	<u>Lettura sul display</u>	<u>Passi ≠</u> (numero di operazioni svolte)
AD	000F 1C	17A
O O F 1	00F1 xx	17B
DA O O	00F1 00	18
AD	00F1 00	19
O O O O	0000 xx	20
DA O 2	0000 02	21
+ O 3	0001 03	22
+ GO	0002 18	23

I passi 17A, 17B e 18 assicurano che è stato scelto il modo aritmetico binario.

I passi da 19 a 21 memorizzano il valore esadecimale 02 nella loca_zione 0000 (VAL1).

Il passo 22 memorizza il valore esadecimale 03 in locazione 0001 (VAL2).

Ora siamo pronti ad avviare il programma.

Nel passo 23 il tasto GO comanda l'esecuzione del programma ed il risultato, 05 appare sulle due cifre di destra del display degli indirizzi.

Per quanto il programma possa apparire banale, esso illustra i principi basilari dell'introduzione e dell'esecuzione di un qualsiasi programma, oltre a verificare ancora una volta che il Vostro KIM-1 lavora in modo corretto.

Potete provare un altro esempio usando il programma da voi memorizzato. Ripetete i passi da 17A a 23 ma sostituite il valore FF per VAL1 e VAL2.

Ora premete il tasto GO. Sul display leggerete:

OOFE xx

La risposta è corretta in quanto:

```
      FF = 1111 1111
+     FF = 1111 1111
=====
      FE  1111 1110
```

Provate tutti gli esempi che volete e quindi completate il sistema seguendo le indicazioni di questo manuale.

Se spegnete l'alimentatore, il programma perde, in quanto la memoria entro il quale era immagazzinato non è permanente.

Se avete nuovamente bisogno dello stesso programma dovrete ricollegare l'alimentazione e reintrodurre il programma come nell'esempio precedente.

1.5 AGGIUNTA DI UN REGISTRATORE A NASTRO

Il sistema KIM-1 è progettato per lavorare in connessione con un registratore audio a cassette che vi fornisce un mezzo di memorizzazione permanente dei programmi e dei dati.

Infatti la cassetta con i dati registrati potrà essere riletta dal sistema infinite volte.

In questa sezione illustreremo come si collega il registratore al sistema e come se ne collauda il funzionamento.

La tecnica di registrazione usata dal KIM-1 ed i circuiti di interfaccia sono stati scelti per assicurare un lavoro privo di inconvenienti con tutti i tipi e qualità di registratori in commercio, dal più sofisticato a quello più economico.

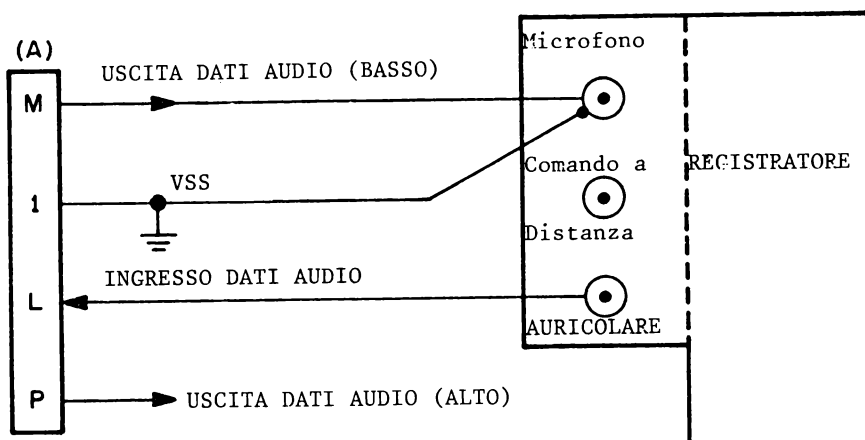
I nastri registrati possono essere introdotti in macchina da un qualsiasi altro apparecchio. Raccomandiamo naturalmente di usare i migliori registratori e le migliori qualità di nastro.

Nella scelta del registratore controllate che sia equipaggiato con i seguenti dispositivi:

1. Una presa auricolare da cui prelevare i dati registrati da introdurre nel microelaboratore.
2. Una presa microfono per registrare i dati provenienti dal KIM-1.
3. I comandi standard per riproduzione, registrazione, riavvolgimento e fermata.

Evitate i registratori miniaturizzati (dittafoni) con microfono e altoparlante incorporati e privi di connessioni esterne. Se volete usare tali apparecchi, dovrete modificarli internamente per portare all'esterno le connessioni desiderate.

Per collegare il vostro registratore al micro, spegnete l'alimentazione e togliete il connettore (A) dal modulo. Effettuate gli allacciamenti come nello schema:



COLLEGAMENTI PER REGISTRATORE AUDIO

Figura 1.3

Mantenete i fili più corti possibile ed evitate di farli passare vicino a sorgenti di disturbi elettrici. Le connessioni illustrate si riferiscono a un tipico apparecchio portatile.

Il segnale di uscita dei dati audio (LO) ha un livello di circa 15 mV di picco sul piedino M. Qualora desideriate usare registratori più costosi e complessi potreste collegarne l'ingresso "line" col segnale audio di alto livello (1 V di picco) che si trova sul piedino P.

Rimettete il connettore (A) nella giusta posizione sul modulo ed accendete l'alimentatore. Per controllare il funzionamento del registratore, provate le seguenti procedure:

1. Reintroducete il programma di prova secondo le procedure indicate nella precedente sezione (1.4). Provate nuovamente il programma di prova per essere sicuri che il sistema lavori correttamente.
2. Inserite una cassetta nel registratore e riavvolgetela fino all'inizio corsa.

3. Definite l'indirizzo di partenza e di arrivo del programma da memorizzare ed assegnate al medesimo un numero di identificazione (ID).

<u>Tasto da premere</u>	<u>Lettura sul display</u>	<u>Passi \neq</u> (numero di operazioni svolte)
AD	xxxx xx	1
0 0 F 1	00F1 xx	2
DA 0 0	00F1 00	3
AD	00F1 00	4
1 7 F 5	17F5 xx	5
DA 0 0	17F5 00	6
+ 0 0	17F6 00	7
+ 1 0	17F7 10	8
+ 0 0	17F8 00	9
+ 0 1	17F9 01	10
AD	17F9 01	11
1 8 0 0	1800 xx	12

Ricorderete che il programma che vogliamo memorizzare sul nastro era caricato nelle locazioni da 0000 a 000F della memoria. Quindi noi definiamo un indirizzo di partenza per la registrazione come 0000 e lo memorizziamo in locazione 17F5 e 17F6 (passi da 4 a 7).

Definiamo l'indirizzo di fine registrazione come uno più dell'ultimo passo del nostro programma e memorizziamo il valore 0010 (= 000F + 1) nelle locazioni 17F7 e 17F8 (passi 8 e 9). Infine assegnamo un numero d'identificazione arbitrario come 01 e memorizziamo questo valore nella locazione 17F9 (passo 10).

L'indirizzo di partenza del programma di registrazione sul nastro è 1800. Nei passi 11 e 12 inseriamo il valore di questo indirizzo nel sistema. Se ora premiamo GO il sistema procederà al caricamento dei da ti sul nastro magnetico. Naturalmente bisognerà avviare il nastro.

In particolare:

4. Disponete il registratore nella condizione di registrazione, aspettate alcuni secondi che il nastro cominci a muoversi e quindi premete GO.
5. Il display si spegnerà per un breve tempo e quindi si illuminerà nuovamente mostrando:

0000 xx

6. Non appena il display si riaccende, la registrazione è finita e potete fermare il registratore.

Per verificare che la registrazione sia avvenuta correttamente, provate a leggere il nastro appena registrato, procedendo come segue:

1. Riavvolgete il nastro portandolo in posizione di partenza.
2. Spegnete e riaccendete l'alimentazione. In tal modo cancellerete il programma introdotto in precedenza ma già registrato sul nastro.
3. Preparate il sistema per la lettura del nastro come segue:

<u>Tasto da premere</u>	<u>Lettura sul display</u>	<u>Passi \neq</u> (numero di operazioni svolte)
RS		
AD	xxxx xx	1
0 0 F 1	00F1 xx	2
DA 0 0	00F1 00	3
AD	00F1 00	4
1 7 F 9	17F9 xx	5
DA	17F9 xx	6
0 1	17F9 01	7
AD	17F9 01	8
1 8 7 3	1873 xx	9
GO	(Spento)	10

Il sistema KIM-1 esplorerà i dati provenienti dal registratore, selezionando quelli con numero di identificazione 01. Ricordate che si tratta della stessa etichetta ID applicata al momento della registrazione.

4. Se il registratore ha un controllo di volume, regolatelo circa a mezza corsa.
5. Se esiste un controllo di tono regolatelo al massimo acuto.
6. Ora disponete il registratore nella condizione di ascolto. Il nastro si muoverà in avanti mentre il sistema accetterà i dati registrati. Quando sarà terminata la lettura della registrazione con il numero ID = 01, il display si illuminerà nuovamente mostrando:

0000 xx

A questo punto potete fermare il nastro. Se il display si illumina e mostra:

FFFF xx

significa che la registrazione è stata memorizzata e letta, ma che è av-

venuto un errore durante la lettura dei dati. In tale caso premete il tasto RS e ripetete il procedimento dall'inizio.

Se appare ancora FFFF sul display, ripetere l'intera registrazione e riproduzione controllando con cura ogni passo.

Se l'inconveniente permane, consultare l'appendice C.

Se il nastro continua a girare ed il display non si illumina, significa che il sistema non ha potuto leggere efficacemente qualche dato sul programma.

Pertanto occorrerà ripetere l'intero procedimento di registrazione e lettura. Se il display continua a rimanere spento, consultate l'appendice C.

Dopo che il nastro è stato letto correttamente, potete verificare che anche il programma è stato reinserito correttamente in memoria, realizzando una prova. ($02 + 03 = 05$).

Nota:

Il fatto che sul display appaiano i valori 0000 oppure FFFF ha puramente un significato convenzionale e non si riferisce in alcun modo ai nostri indirizzi di programma.

I circuiti di interfaccia audio del sistema KIM-1 sono stati progettati in modo che non occorrono speciali apparecchiature di prova per stabilire i corretti livelli operativi.

Se avete seguito le procedure indicate, il registratore lavorerà senza che abbiate bisogno di regolarlo.

1.6 COLLEGAMENTO CON LA TELESKRIVENTE

Se disponete di una telescrivente seriale, potete facilmente collegarla al KIM-1.

Una delle macchine più comunemente disponibili è la teletype modello ASR 33 che noi usiamo come esempio di riferimento.

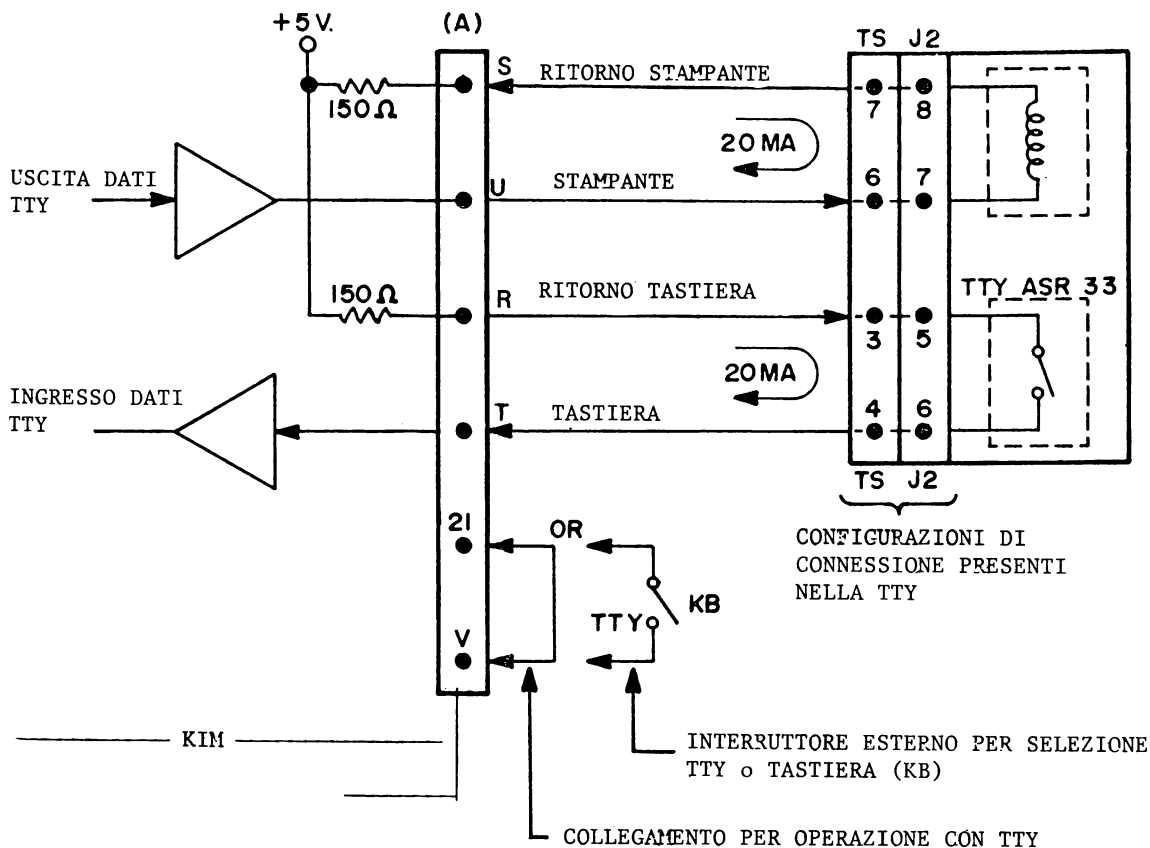
Ma, se disponete di apparecchiature diverse, potete usare le informazioni qui fornite come traccia per il collegamento della vostra macchina specifica.

In ogni caso per effettuare i collegamenti necessari è raccomandabile seguire le indicazioni fornite dal produttore.

Il KIM-1 prevede un'interfaccia a 4 fili per la telescrivente. In particolare si deve usare la configurazione ad "anello da 20 mA" e controllare che la vostra teletype sia predisposta per questa configurazione. In caso contrario, potete facilmente cambiare la configurazione da 60 mA a 20 mA seguendo le istruzioni del fabbricante. Non dovete preoccuparvi delle configurazioni duplex o semiduplex in quanto il KIM-1 opera bene con entrambe. Inoltre, non siete costretti a scegliere macchine con determinata velocità di trasmissione (10 CPS per la TTY) dal momento che il sistema KIM-1 si adatta automaticamente ad una grande varietà di cadenze di uscita dati: 10 CPS, 15 CPS, 30 CPS, ecc.

Per connettere la TTY al sistema, procedete come segue:

1. Spegnete l'alimentazione del sistema e rimuovete il connettore (A) dal modulo.
2. Aggiungete i collegamenti indicati nello schema al connettore (A) ed all'appropriato connettore dell'unità TTY.



CONNESSIONI TTY

Figura 1.4

- Il cavallotto da A-21 ad A-V serve a stabilire che per il KIM-1 viene usata una telescrivente come unico dispositivo di ingresso e visualizzazione per il sistema. Se vi interessa usare sia la TTY, sia la tastiera ed il visualizzatore, installate l'interruttore indicato al posto del cavallotto. L'interruttore, se aperto, permetterà l'uso della tastiera e del display del KIM-1 e, se chiuso, sceglierà la telescrivente come apparecchiatura di ingresso e visualizzazione. Naturalmente, se volete, potete usare un filo con coccodrillo.

4. Accertatevi che i piedini A-21 ed A-V siano collegati, reinstallare il connettore (A), riaccendete l'alimentatore e accendete la TTY.
5. Premete il tasto RS sul modulo KIM-1 e successivamente il tasto RUBOUT della TTY. Questa sequenza è molto importante dal momento che il KIM-1 si adatta automaticamente alla frequenza di emissione dei bit della telescrivente seriale e per ottenere ciò è necessario premere il suddetto tasto.
Se tutto funziona appropriatamente, la telescrivente stamperà un messaggio di macchina pronta, "KIM", indicante che la TTY è in linea e il sistema KIM-1 è pronto ad accettare comandi dalla tastiera TTY. Se il messaggio KIM non viene stampato, ricontrollate le connessioni e la stessa TTY e riprovate.
Se l'inconveniente persiste, controllate l'appendice C.
6. Se la TTY è operativa, potete provare un semplice gruppo di operazioni per verificare la corretta operazione del sistema.

<u>Tasto da premere</u>	<u>Stampa</u>	<u>Passi \neq</u> (numero di operazioni svolte)
	KIM	
	xxxx xx	1
0 0 0 2	0002	2
spazio	0002 xx	3
1 8 .	18.	4
	0003 xx	5
A 5 .	A5.	6
	0004 xx	7
LF	0003 A5	8
RUBOUT	KIM	
	xxxx xx	9

Il passo 1 mostra il messaggio di macchina pronta "KIM".
Nel passo 2 viene selezionato un indirizzo (0002) seguito da uno spazio nel passo 3. La cella di indirizzo 0002 insieme al dato ivi memorizzato (xx) appaiono stampati. Il passo 4 mostra l'operazione di "modifica della cella" usando il tasto (.) preceduto dai dati esadecimali.
Il passo 5 mostra l'incrementazione alla successiva cella di memoria (0003) dopo il tasto (.). Notare che avviene anche la modifica della cella 0002. I passi 6 e 7 mostrano la modifica del dato nella cella 0003 e

l'incremento alla cella 0004.

Il passo 8 mostra l'azione del tasto LF nel tornare ad una cella 0003 dove possiamo vedere stampato che un dato corretto è stato memorizzato a questa locazione (A5).

Il passo 9 mostra la reazione al tasto RUBOUT che resetta il sistema e produce un nuovo messaggio KIM di macchina pronta.

Notare, tra l'altro, che in questo esempio avete ripetuto una porzione dell'introduzione del programma esattamente come avete fatto nella sez. 1.4, ma questa volta usando la TTY.

E basta per adesso! Se tutte le operazioni sono avvenute correttamente, potete essere certi che la vostra TTY ed il modulo KIM-1 lavorano bene insieme.

Descriveremo in dettaglio tutte le altre operazioni possibili con la TTY in un capitolo successivo.

Se siete arrivati fino a questo punto senza inconvenienti, avete completato tutti i controlli del sistema e potete confidare che il modulo KIM-1 e le periferiche lavorano correttamente.

Il nostro prossimo compito è di studiare per apprendere qualcosa di più sul KIM-1 ed i suoi programmi operativi.

CAPITOLO 2

IL SISTEMA KIM-1

Fino a questo punto siete stati occupati nella messa in esercizio del vostro KIM-1 e nel verificare il suo buon funzionamento. Ora è venuto il momento di studiare qualcosa di più sulle varie parti del microprocessore, come queste lavorano insieme formando un sistema, e come i programmi operativi controllano le varie attività del sistema.

Gli schemi inclusi in questo capitolo vi saranno d'aiuto.

2.1 DESCRIZIONE DEL SISTEMA

La figura 2.1 mostra uno schema a blocchi completo del sistema KIM-1. Notate per prima cosa la presenza del circuito microprocessore MCS 6502, che agisce come elemento centrale di controllo del sistema. L'unità è un microprocessore ad 8 bit che comunica con gli altri elementi del sistema su tre linee separate. Per prima cosa una linea degli indirizzi a 16 bit permette al 6502 di indirizzare direttamente fino a 65.536 locazioni di memoria nel sistema. Successivamente una linea dei dati bidirezionale ad 8 bit trasporta i dati dal 6502 a qualsiasi locazione di memoria o da qualsiasi locazione di memoria al 6502. Per ultimo una linea (o bus) di controllo, porta i vari segnali di temporizzazione tra il 6502 e gli altri elementi del sistema.

Connesso con il 6502 c'è un cristallo di quarzo da 1 MHz che opera con un circuito oscillatore contenuto nello stesso 6502. Tale oscillatore è la sorgente base dei segnali di temporizzazione dal quale sono derivati tutti gli altri segnali di temporizzazione del sistema. In particolare il segnale \emptyset_2 generato dal 6502 che è usato sia da solo che con altri segnali di controllo, ed inoltre come base dei tempi da tutti gli altri elementi del sistema.

Il microprocessore 6502 è strutturato per lavorare in congiunzione con vari tipi di memoria. Nel sistema KIM-1 tutta la memoria è del tipo ROM (memoria di sola lettura) e RAM (memoria di lettura e scrittura). La porzione ROM della memoria provvede ad una conservazione permanente dei programmi operativi essenziali al controllo del sistema KIM-1. Noterete l'inclusione di due dispositivi chiamati 6530-002 e 6530-003, ciascuno dei quali comprende 1024 byte da 8 bit cadauno di ROM in ciascuna delle quali sono memorizzate porzioni diverse del programma operativo, in modo permanente.

Le memorie di tipo RAM sono disponibili in tre gruppi di locazione del sistema. Inoltre ciascun circuito 6530 comprende 64 bytes di RAM usati principalmente per la memorizzazione temporanea di programmi e dati definiti dall'utente.

Anche i controlli di ingresso/uscita del sistema sono inclusi nei circuiti 6530. Ciascuno di questi ultimi ha 15 piedini ingresso/uscita mentre il microprocessore ed il programma operativo stabiliscono se un piedino è di entrata oppure di uscita, quale dato deve apparire sui piedini di uscita, e legge quali dati appaiono su quelli d'ingresso. I piedini ingresso/uscita del circuito 6530-002 sono dedicati alla interfaccia con particolari elementi del sistema KIM-1, compresa la tastiera, il display, il circuito di interfaccia TTY, ed il circuito di interfaccia del registratore. I 15 piedini ingresso/uscita del 6530-003 sono collegati ad un connettore e sono disponibili all'utente per controllare le sue particolari applicazioni.

Infine, ciascun circuito 6530 comprende un temporizzatore degli intervalli capace di contare un dato numero di impulsi di clock e provocare comandi cadenziati esattamente nel tempo. La durata dell'intervallo è stabilita sotto il controllo del programma. Il temporizzatore degli intervalli del circuito 6530-003 è disponibile per un programma di applicazione definito dall'utente e non è impegnato dai programmi operativi del sistema KIM-1.

La fig. 2.1 mostra un blocco chiamato logica di controllo. In questa categoria rientrano un decodificatore degli indirizzi per la generazione dei segnali di selezione del chip per i circuiti 6530 e la RAM statica, la logica richiesta per evitare il rimbalzo dei tasti, per il resettaggio del sistema (Tasto RS) e la fermata del programma (Tasto ST). Infine è inclusa una logica speciale che permette al sistema di lavorare nel modo "a singola istruzione", in modo da facilitare la ricerca degli errori nel programma (debugging).

La fig. 2.1 mostra la logica d'interfaccia della tastiera e del display con i piedini ingresso/uscita del 6530-002. Sono inoltre evidenti i circuiti d'interfaccia per la trasmissione/ricezione dei dati verso e dalla TTY ed il registratore.

La fig. 2.2 mostra in dettaglio le interconnessioni tra l'MCS 6502 ed i due dispositivi MCS 6530.

La fig. 2.3 mostra in dettaglio lo schema della logica di controllo.

La fig. 2.4 mostra uno schema dettagliato della RAM statica.

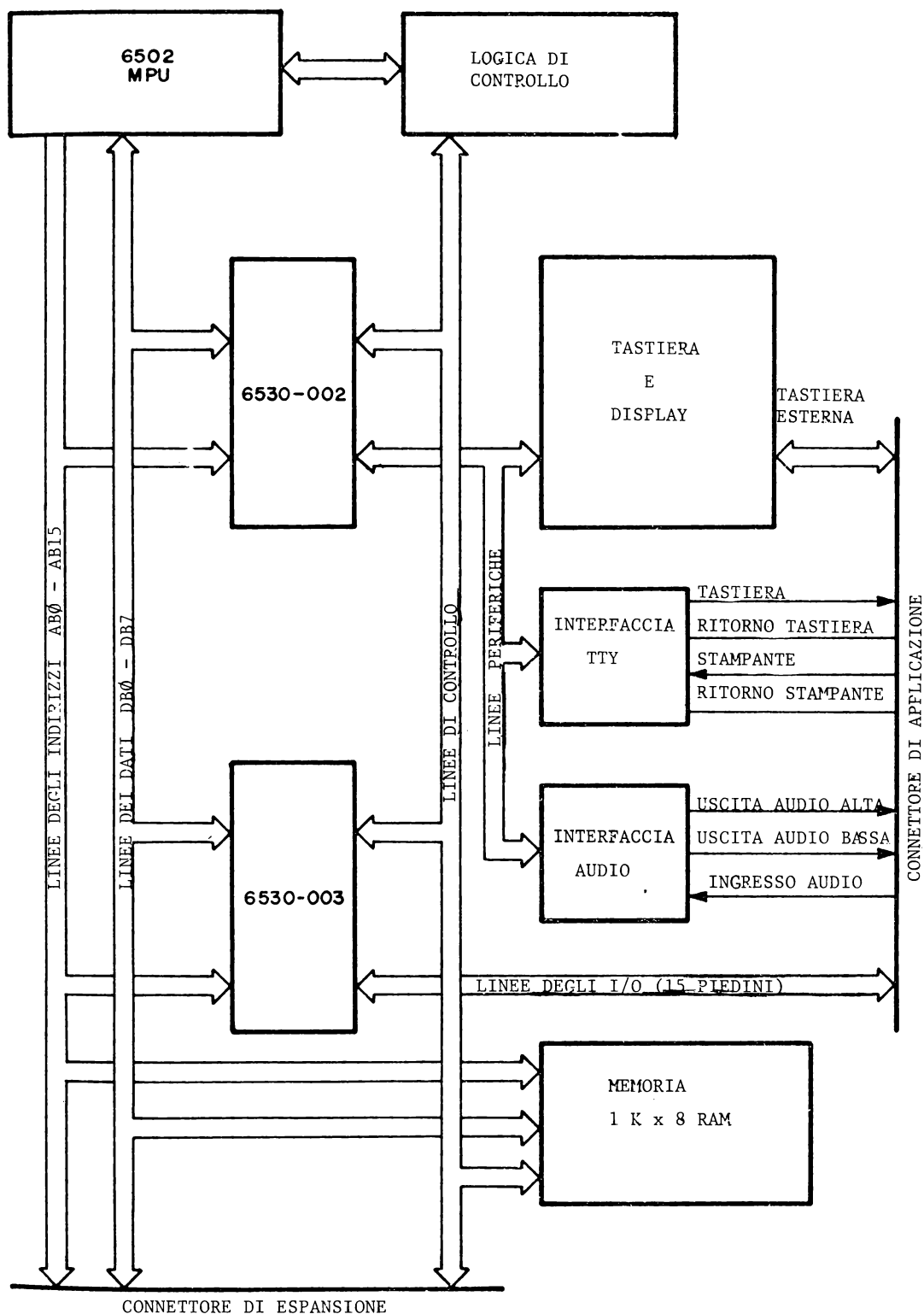
Le fig. 2.5 e 2.6 mostrano lo schema dettagliato delle logiche e dei circuiti della tastiera e del display.

La fig. 2.7 dettaglia lo schema del circuito di interfaccia TTY.

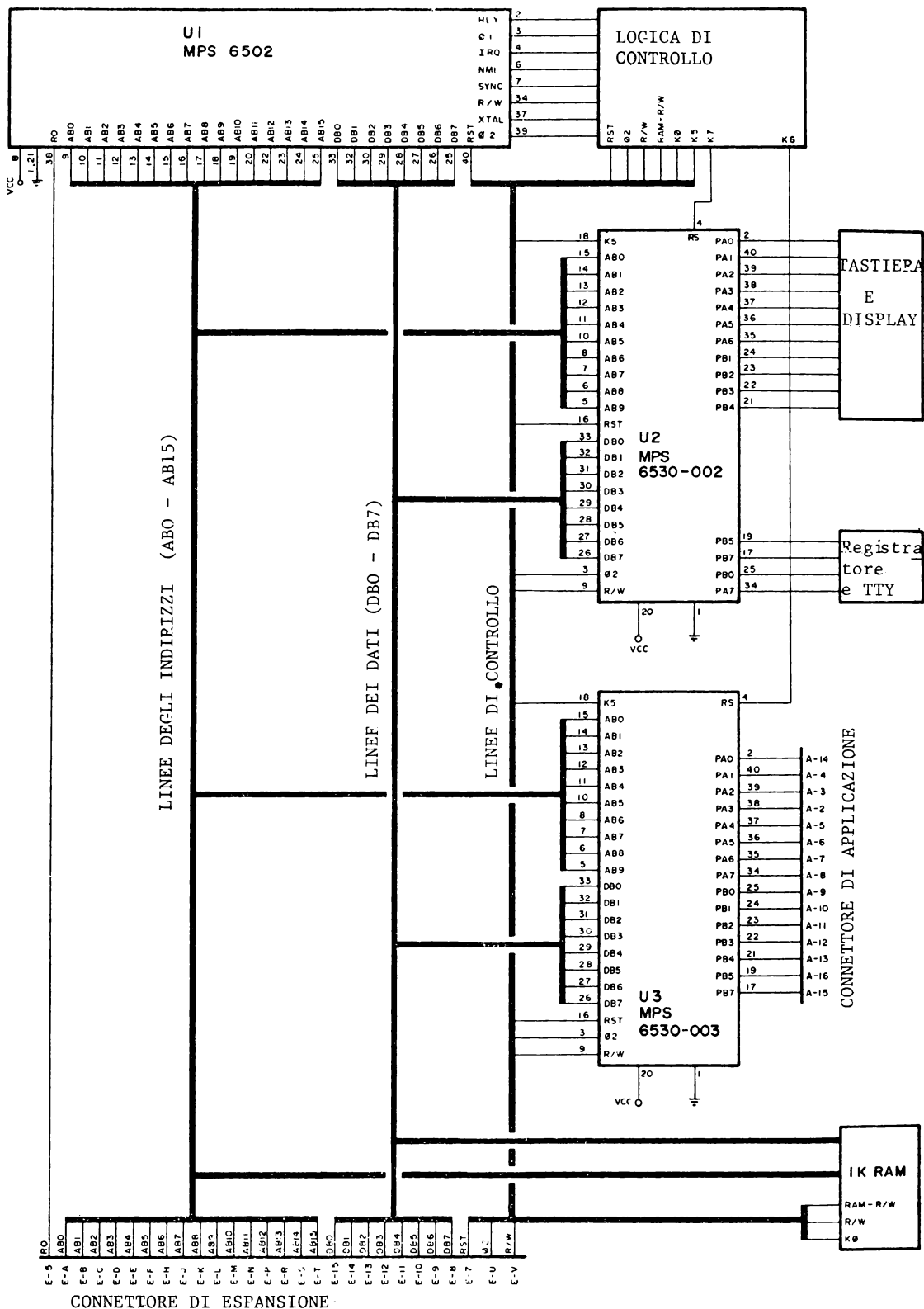
La fig. 2.8 dettaglia lo schema dell'interfaccia audiocassette.

Le fig. 2.9 e 2.10 forniscono un sommario di tutti i segnali che si trovano sul connettore di applicazione e su quello di espansione.

Il manuale Hardware vi darà particolari aggiuntivi sulle caratteristiche operative del 6502 e del 6530 come pure dettagliate informazioni sulla temporizzazione del sistema.

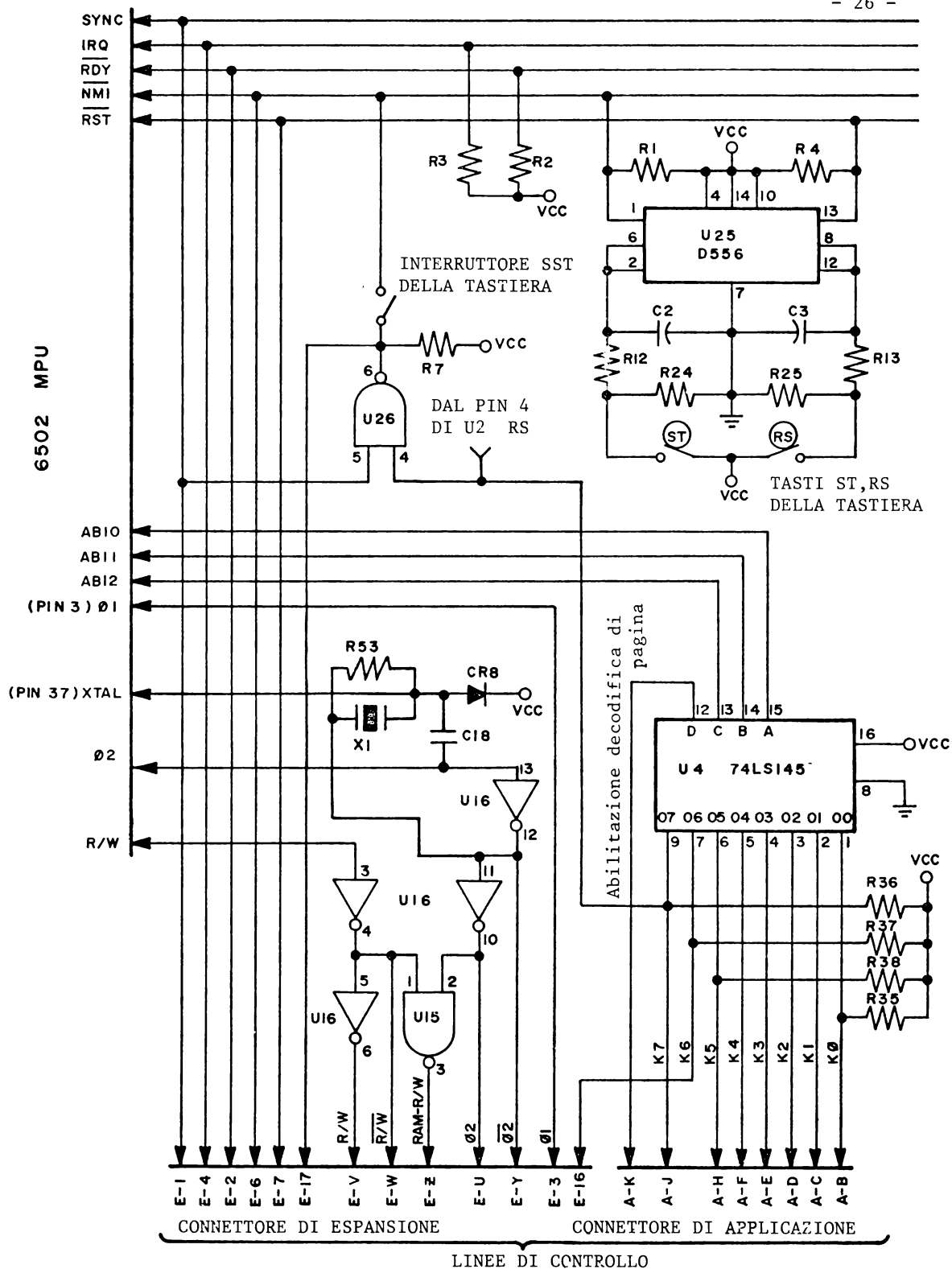


KIM-1 DIAGRAMMA A BLOCCHI
Figura 2.1



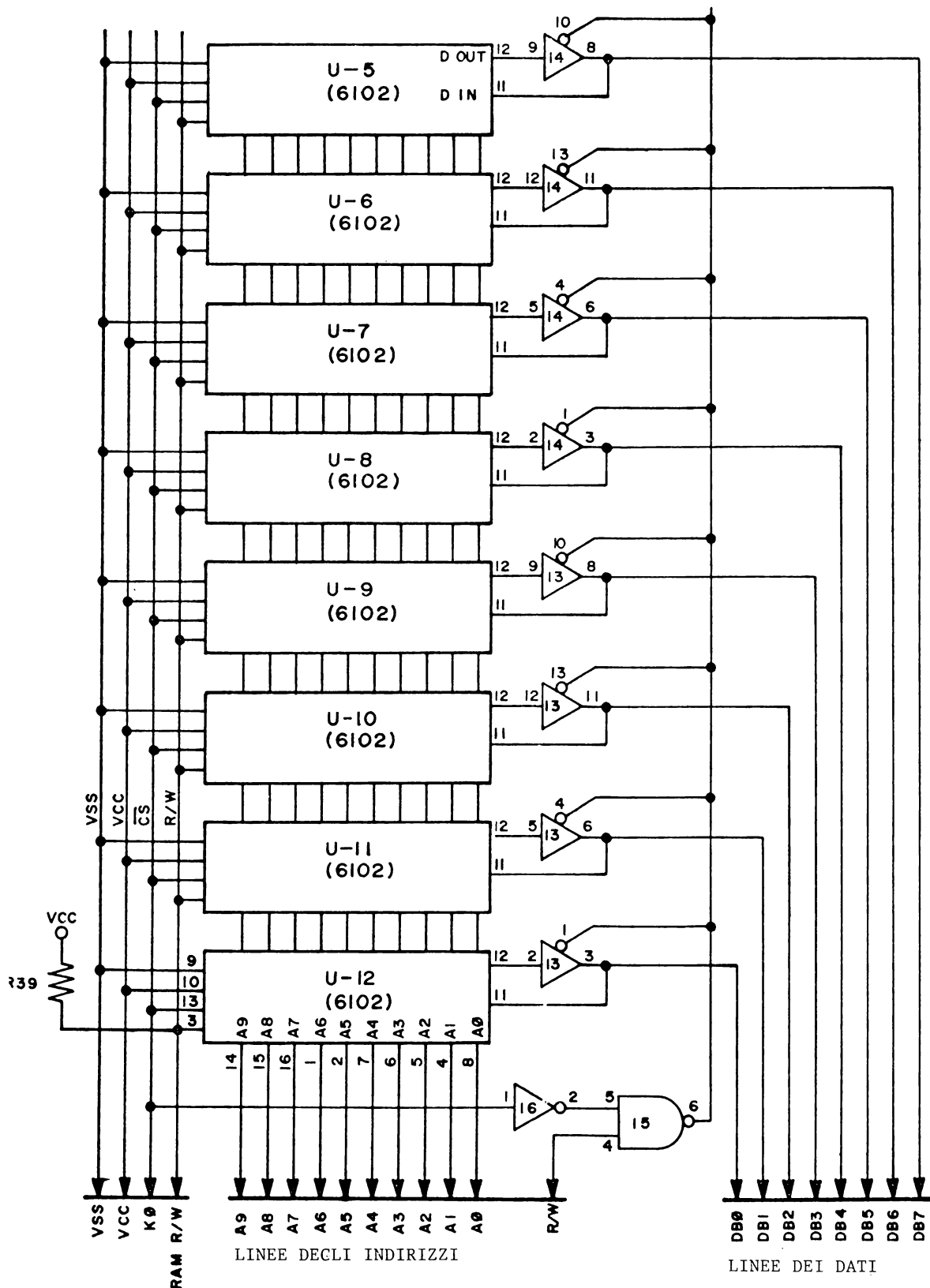
DETTAGLIO DEL DIAGRAMMA A BLOCCHI

Figura 2.2



CONTROLLO E TEMPORIZZAZIONE

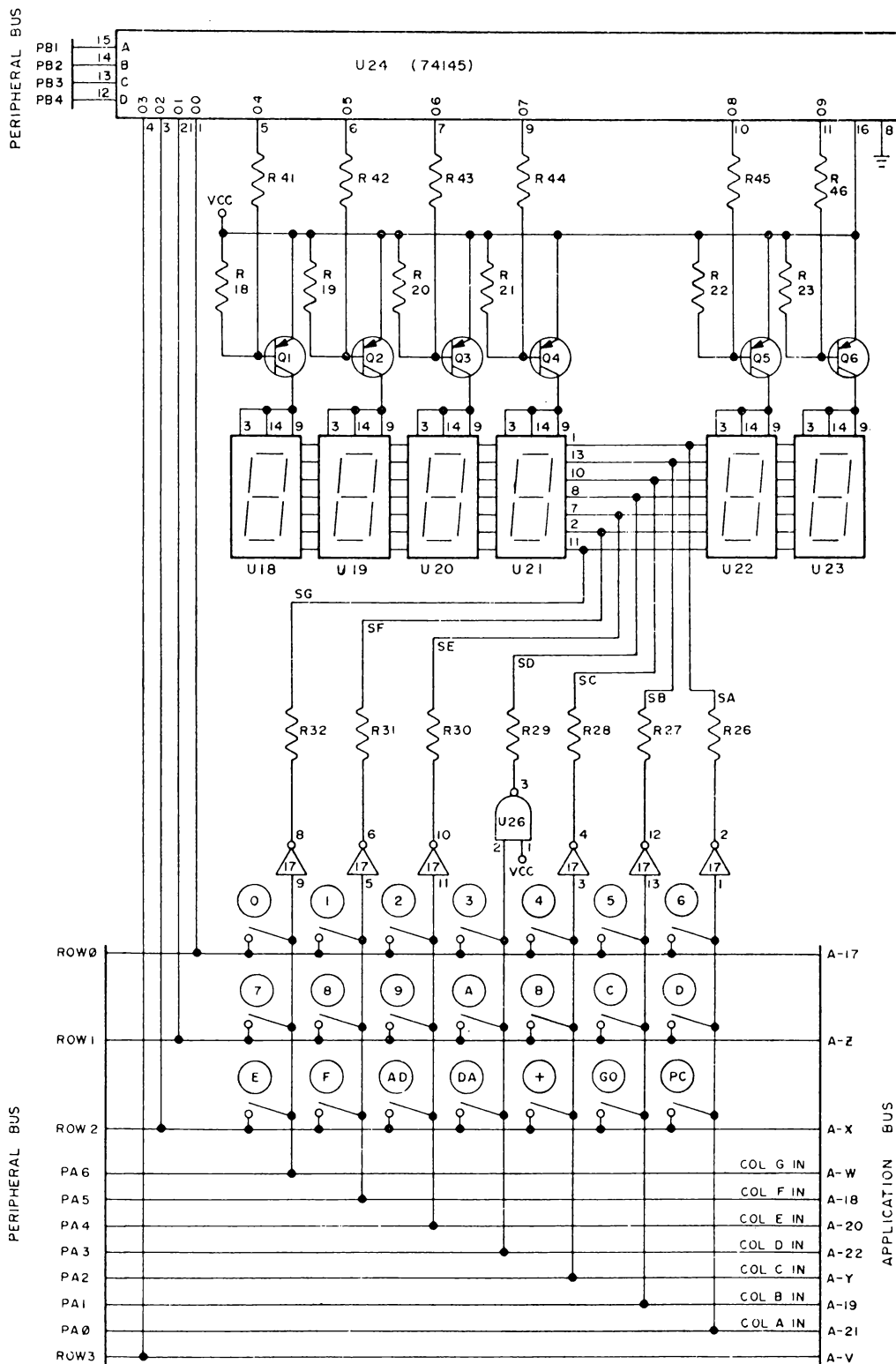
Figura 2.3



LINEE DI CONTROLLO

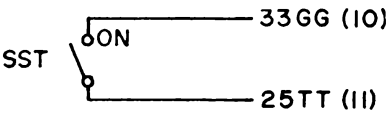
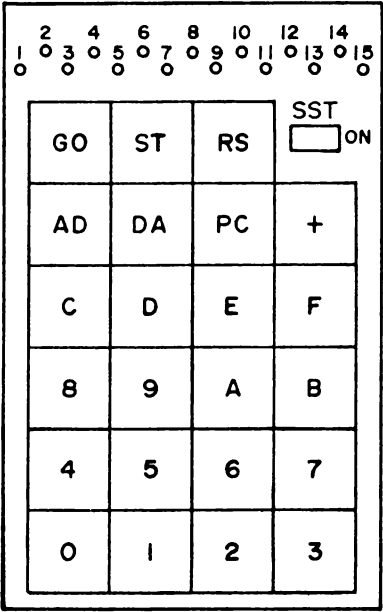
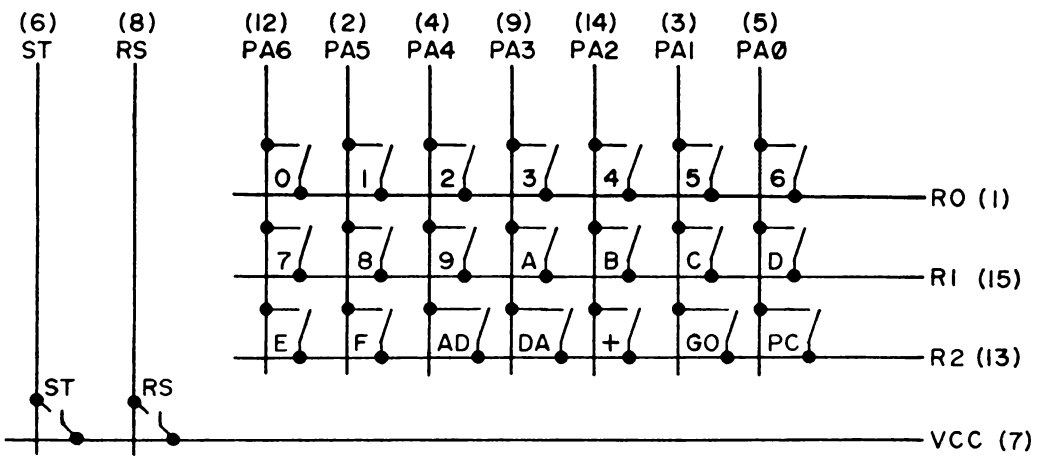
MEMORIA RAM 1K x 8

Figura 2.4



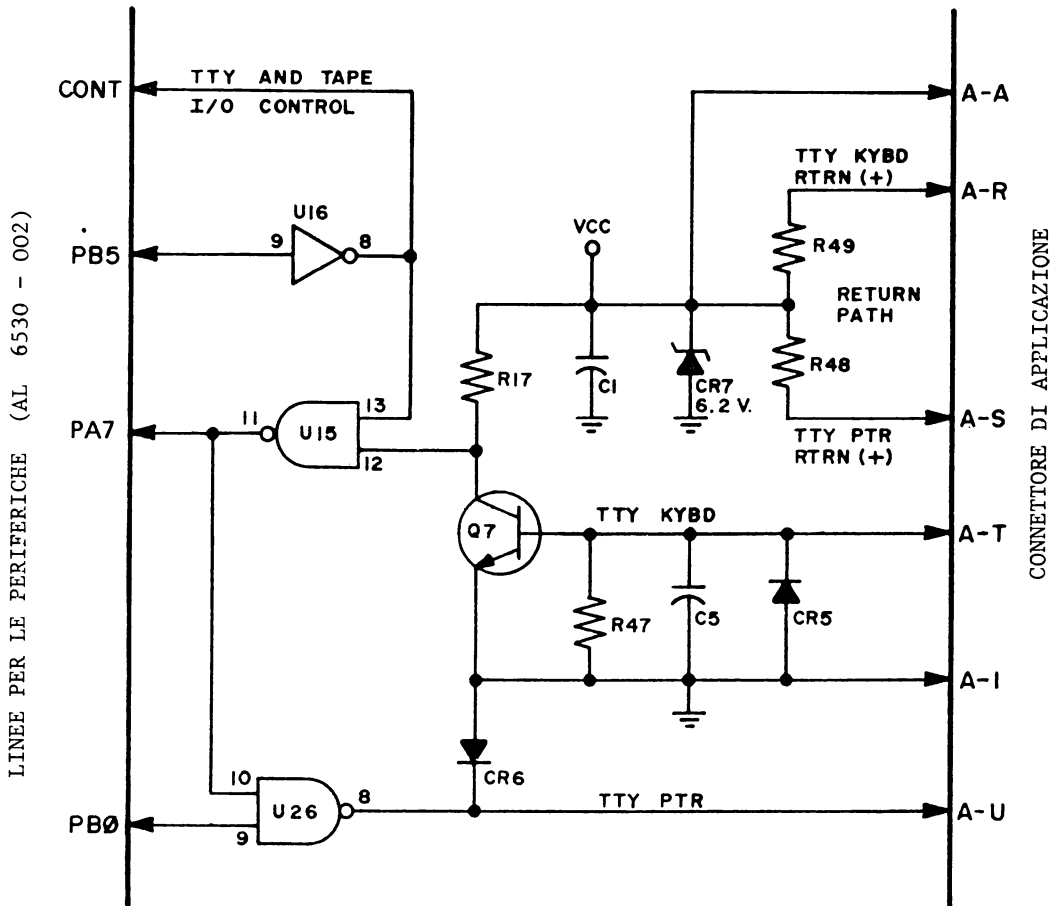
TASTIERA E DISPLAY

Figura 2.5



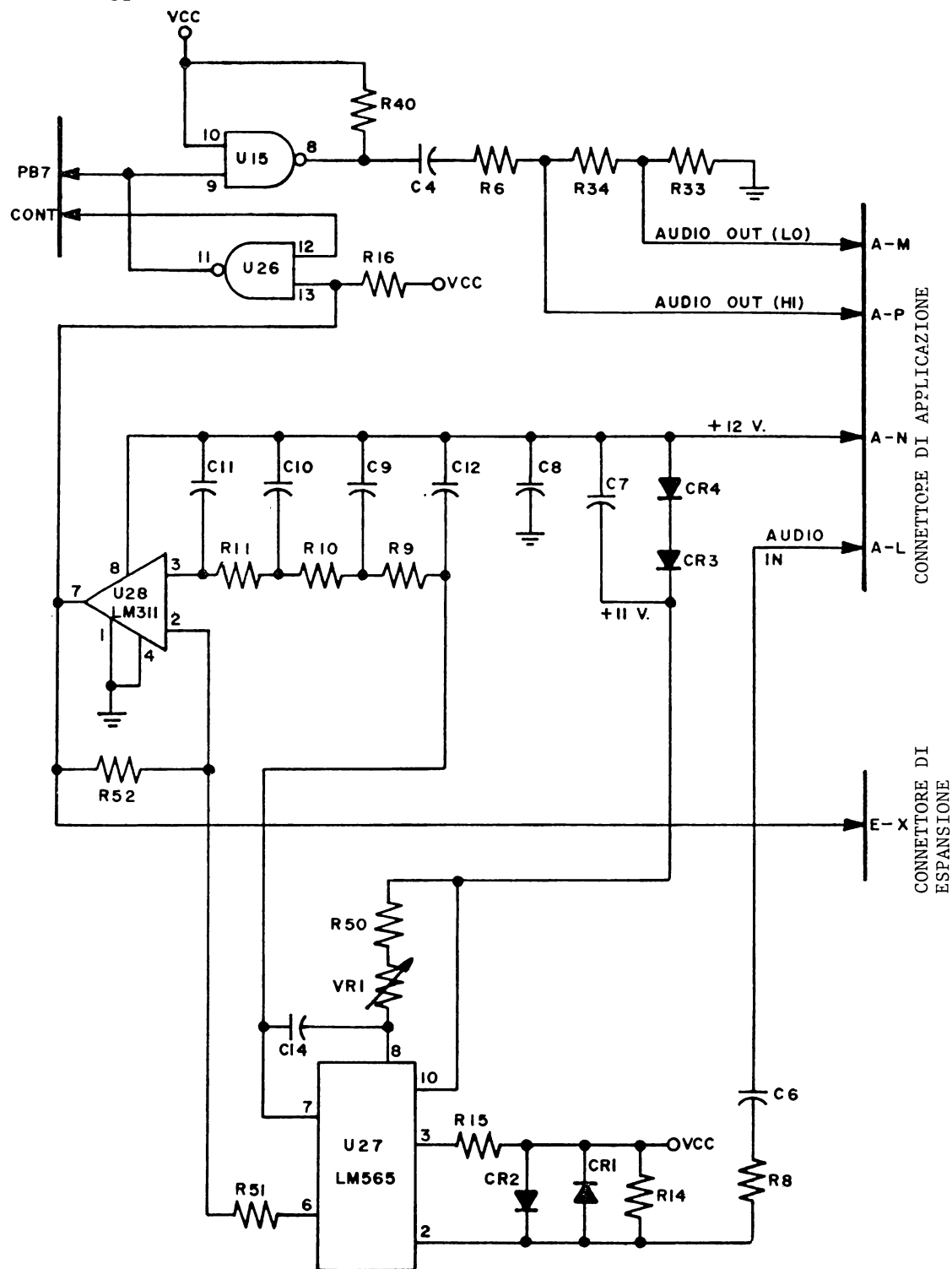
DETTAGLIO DELLA TASTIERA

Figura 2.6



INTERFACCIA TTY

Figura 2.7



INTERFACCIA PER REGISTRATORE

Figura 2.8

22	KB Colonna D	Z	KB Riga 1
21	KB Colonna A	Y	KB Colonna C
20	KB Colonna E	X	KB Riga 2
19	KB Colonna B	W	KB Colonna G
18	KB Colonna F	V	KB Riga 3
17	KB Riga Ø	U	STAMPANTE
16	PB5	T	TASTIERA
15	PB7	S	RITORNO STAMPANTE (+)
14	PAØ	R	RITORNO TASTIERA (+)
13	PB4	P	USCITA AUDIO ALTA
12	PB3	N	+12v
11	PB2	M	USCITA AUDIO BASSA
10	PB1	L	INGRESSO AUDIO
9	PBØ	K	ABILITAZIONE DELLA DECODIFICA
8	PA7	J	K7
7	PA6	H	K5
6	PA5	F	K4
5	PA4	E	K3
4	PA1	D	K2
3	PA2	C	K1
2	PA3	B	KØ
1	VSS GND	A	VCC +5v

CONNETTORE DI APPLICAZIONE

Figura 2.9

22	VSS GND	Z	RAM/R/W
21	VCC +5	Y	$\overline{\emptyset 2}$
20		X	PLL TEST
19		W	$\overline{R/W}$
18		V	R/W
17	SST OUT	U	$\emptyset 2$
16	K6	T	AB15
15	DB \emptyset	S	AB14
14	DB1	R	AB13
13	DB2	P	AB12
12	DB3	N	AB11
11	DB4	M	AB10
10	DB5	L	AB9
9	DB6	K	AB8
8	DB7	J	AB7
7	RST	H	AB6
6	NMI	F	AB5
5	RO	E	AB4
4	IRQ	D	AB3
3	$\emptyset 1$	C	AB2
2	RDY	B	AB1
1	SYNC	A	AB \emptyset

CONNETTORE DI ESPANSIONE

Figura 2.10

2.2 DISTRIBUZIONE DELLE LOCAZIONI DI MEMORIA

Si è detto che il microprocessore 6502 incluso nel sistema KIM-1 è capace di indirizzare ciascuna delle 65.536 locazioni di memoria. Il KIM-1 tuttavia non include tutte queste memorie. In questo capitolo dettaglieremo esattamente le locazioni di memoria incluse nel sistema ed il loro posizionamento (indirizzi esatti).

Ciascun byte di memoria del sistema contiene 8 bit, e ogni locazione indirizzabile può svolgere indifferentemente una delle seguenti quattro funzioni:

1. Un byte di ROM - Memoria di sola lettura, nel quale abbiamo immagazzinato il programma operativo.
2. Un byte di RAM - Memoria di lettura/scrittura per stoccaggio di dati variabili.
3. Una posizione di I/O, che comprende sia il registro di direzione che definisce quali piedini di I/O siano ingressi e quali uscita, sia il registro buffer che contiene i dati che devono essere trasmessi ai piedini di uscita oppure i dati letti dai piedini di entrata. Ciascuna posizione I/O è vista come una locazione di memoria RAM (read/write) con uno specifico indirizzo.
4. Una posizione per temporizzatore di intervalli. Una serie di indirizzi è riservata ai temporizzatori d'intervallo del sistema. Voi potete scrivere al temporizzatore per definire il suo periodo di conteggio o leggere dal temporizzatore quale sia il suo stato.

La fig. 2.11 mostra un diagramma a blocchi che dettaglia tutti i blocchi di memoria nel KIM - 1.

La fig. 2.12 mostra una mappa globale con tutte le posizioni indirizzabili incluse nel sistema e le loro interrelazioni. Da notare le aree nelle locazioni di memoria destinate all'espansione, di cui parleremo nel cap. 5.

Infine la fig. 2.13 dà una lista completa delle più importanti locazioni di memoria e ad essa si deve fare frequente riferimento quando si progettano i programmi.

Nota: I/O sta per: I = Input = Ingresso
O = Output = Uscita

Riferendoci alla fig. 2.12, va notato che la mappa delle memorie mostra un blocco di 8192 indirizzi tutti esistenti nelle locazioni più basse delle 65.536 posizioni di indirizzamento possibili. Questo spazio di indirizzi è ulteriormente diviso in 8 blocchi di 1024 locazioni ciascuno. Ogni blocco da 1024 è ulteriormente diviso in quattro pagine da 256 locazioni ciascuna. Il riferimento K definisce uno specifico blocco di 1024 locazioni e si riferisce al numero K del decodificatore di indirizzi incluso entro la logica di controllo del sistema. Il riferimento "pagina" definisce uno specifico gruppo di 256 indirizzi. Un totale di 32 pagine (da 0 a 31) è incluso nelle 8192 posizioni di indirizzo. I codici esadecimali riguardanti certi indirizzi sono situati in posizioni strategiche nella mappa delle memorie.

Cominciando dalla più alta locazione di indirizzo (8192) noterete che il primo blocco da 1024 (K7) è assegnato alla ROM del 6530-002 ed il secondo da 1024 (K6) è assegnato alla ROM del 6530-003. L'intero programma operativo del KIM-1 è racchiuso in questi due blocchi.

Una porzione del blocco K5 è dedicata alla RAM, all'I/O ed alle posizioni del timer dei due sistemi 6530 ed illustrata in fig. 2.12, in modo particolareggiato. Gli indirizzi RAM per il 6530-002 (da 17E7 a 17FF esadecimale) sono riservati per l'uso da parte del programma operativo e non devono apparire in un programma di applicazione creato dall'utente. Lo stesso vale per le locazioni I/O e Timer che nel 6530-002 sono pure riservate all'uso da parte dei programmi operativi interni.

I successivi quattro blocchi (K4, K3, K2, K1) sono riservati a memorie aggiuntive in un sistema espanso.

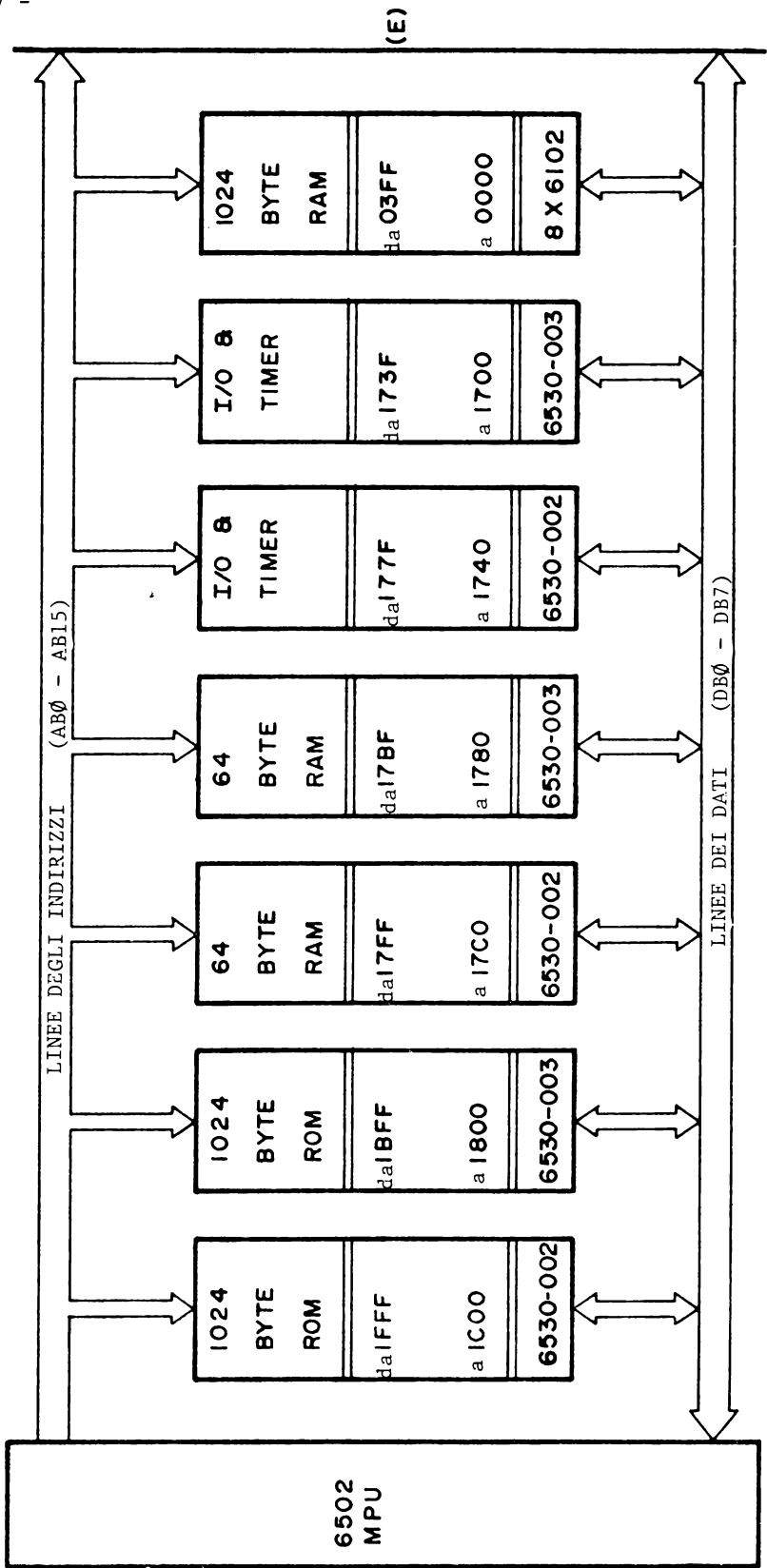
Infine, le 1024 locazioni più basse sono assegnate alla RAM statica inclusa nel sistema KIM-1. Entro questo blocco, la pagina 0 e la pagina 1 hanno una speciale importanza. La pagina 1, infatti, è usata come stack (catasta) del sistema nella quale sono parcheggiati gli indirizzi da conservare e le parole che definiscono lo stato di macchina, quando il sistema risponde a comandi di interrupt (interruzione) o di subroutine. La pagina 0 ha valore in quanto favorisce certi speciali modi di indirizzamento disponibili nel la programmazione del microprocessore 6502.

La figura 2.12 mostra in forma espansa le pagine 0 ed 1.

Da notare che 17 indirizzi (da 00EF a 00FF) sono riservati per l'uso nel programma operativo e non devono mai apparire nel programma d'applicazione dell'utente. Va notato inoltre che un massimo di 8 locazioni possono essere richieste dallo stack (pagina 1) per servire l'operazione di interrupt del programma.

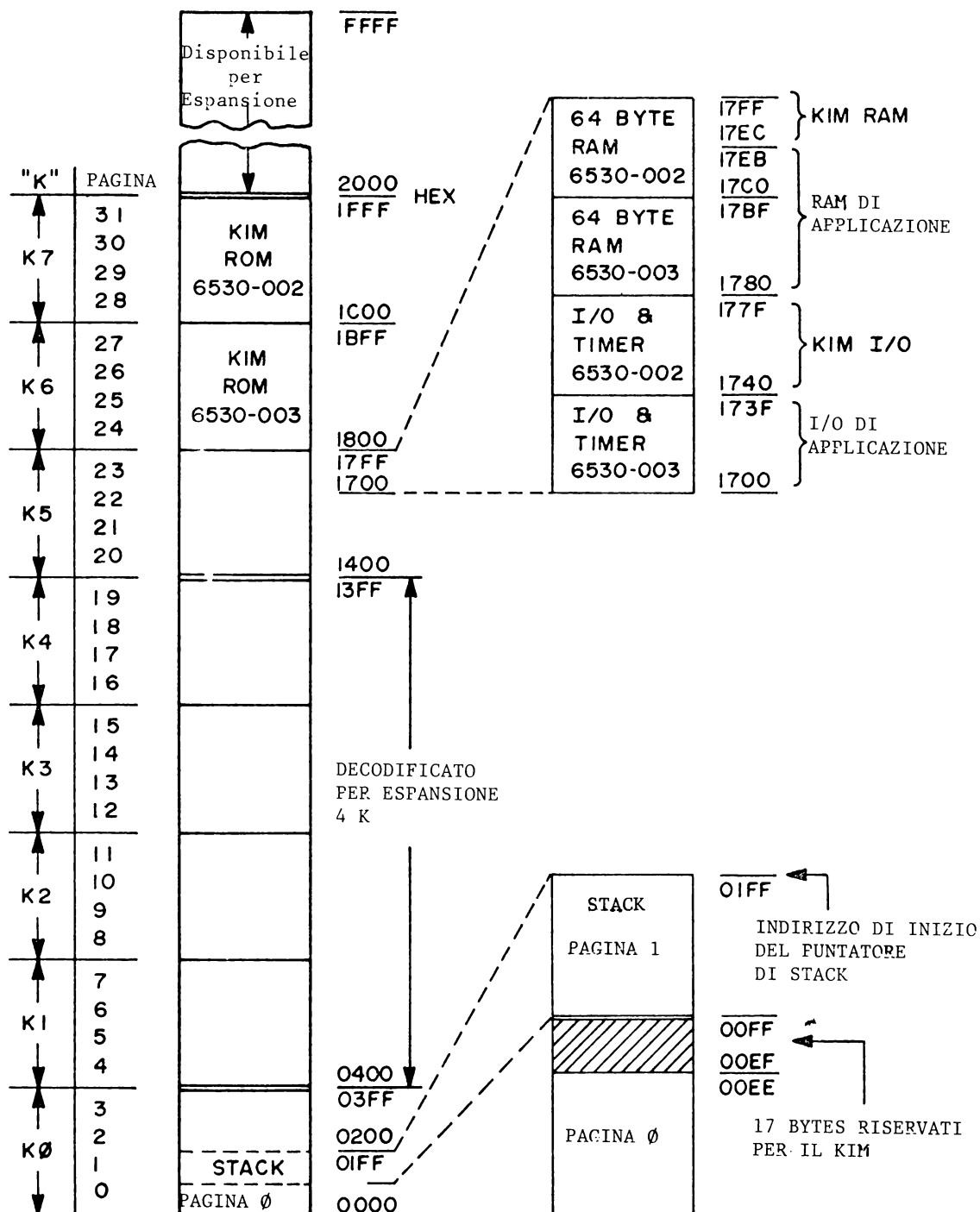
Concludendo, l'utilizzatore dispone per i suoi programmi dei seguenti campi di memoria:

1. tutti quelli della pagina 0, esclusi da 00EF a 00FF
2. quelli della pagina 1 (ricordare che lo stack necessiterà di un extra di 8 bytes che servono al programma operativo)
3. la pagina 2 e la pagina 3
4. nella pagina 23:
 - le posizioni di I/O da 1700 a 173F
 - i 64 bytes di RAM da 1780 a 17BF
 - 44 bytes di RAM addizionali da 17C0 a 17E6



DIAGRAMMI DEI BLOCCHI DI MEMORIA

Figura 2.11



MAPPA DI MEMORIA

Figura 2.12

INDIRIZZI	DESCRIZIONE	ETICHETTA	FUNZIONE
00EF	Indirizzi di salvataggio dei registri del processore	PCL	Contatore di programma - Byte meno significativo
00F0		PCH	Contatore di programma - Byte più significativo
00F1		P	Registro degli stati
00F2		SP	Puntatore dello stack
00F3		A	Accumulatore
00F4		Y	Registro indice Y
00F5		X	Registro indice X
1700	I/O di applicazione	PAD	6530-003 Registro dei dati lato A.
1701		PADD	6530-003 Registro direzione lato A
1702		PBD	6530-003 Registro dei dati lato B
1703		PBDD	6530-003 Registro direzione lato B
1704	Temporizzatore		6530-003 Temporizzatore (vedi sezione 1.6 del Manuale Hardware)
170F			
17F5	Registratore Caricamento da nastro e registrazione su nastro	SAL	Indirizzo di partenza - Byte meno significativo
17F6		SAH	Indirizzo di partenza - Byte più significativo
17F7		EAL	Indirizzo di fine - Byte meno significativo
17F8		EAH	Indirizzo di fine - Byte più significativo
17F9		ID	Numero di identificazione della registrazione
17FA	Vettori di interruzione	NMIL	Vettore di NMI - Byte meno significativo
17FB		NMIH	Vettore di NMI - Byte più significativo
17FC		RSTL	Vettore di RST - Byte meno significativo
17FD		RSTH	Vettore di RST - Byte più significativo
17FE		IRQL	Vettore di IRQ - Byte meno significativo
17FF		IRQH	Vettore di IRQ - Byte più significativo
1800	Registratore	DUMPT	Indirizzo di partenza - Registrazione
1873		LOADT	Indirizzo di partenza - Caricamento da nastro
1C00	Tasto di STOP, +, SST		Indirizzo di partenza per NMI che utilizza la routine che inizia da 1C00. Questo indirizzo deve essere caricato in 17FA & 17FB
17F7	Trasferimento su nastro perforato (Q)	EAL	Indirizzo di fine-Byte meno significativo
17F8		EAH	Indirizzo di fine-Byte più significativo

2.3 PROGRAMMI OPERATIVI DEL KIM - 1

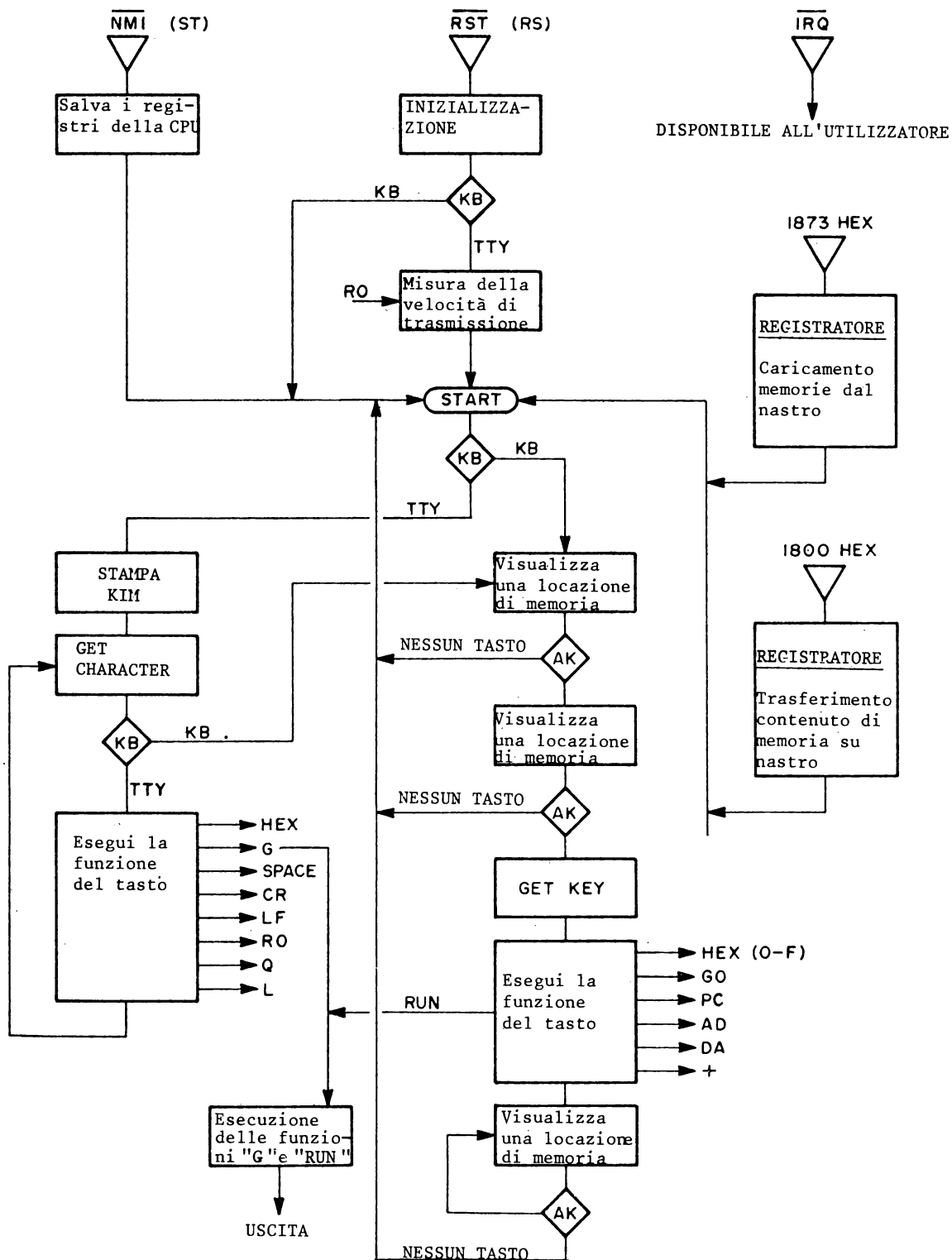
La fig. 2.14 mostra una carta di flusso semplificata dei programmi operativi del KIM-1. Questo capitolo tende a spiegare brevemente questi programmi per aiutarvi nella comprensione dei vari modi di operare del sistema.

Per prima cosa va notato che quando l'alimentazione è applicata al KIM-1 e si preme il tasto RS (reset), il controllo del sistema è automaticamente preso dal programma operativo. Ciò si verifica altresì per ogni successiva pressione del tasto reset.

Ad ogni pressione del tasto di reset, il sistema è inizializzato. A questo momento è stabilito il valore dello stack pointer, si definisce la configurazione I/O, e si condizionano i flags di stato. In seguito il programma determina se il sistema deve rispondere ad ingressi TTY o deve operare con la tastiera ed il display contenuti nel KIM-1.

Se è stato scelto il modo TTY, il programma si ferma ed attende la prima pressione di un tasto sulla TTY (il tasto RUBOUT). Alla ricezione di questa battuta, il programma misura automaticamente la cadenza dei bit e ne memorizza il corretto valore per usarlo nella ricezione e decodifica dei dati successivamente scambiati con la TTY. La misura della cadenza dei bit è eseguita dopo ciascuna pressione del tasto di reset.

Il programma passerà immediatamente ad una routine che produce il messaggio di macchina pronta ("KIM") stampato dalla TTY. Ora, il programma si ferma all'anello denominato "get character" (**prendere carattere**). Ogni qualvolta un tasto è premuto nella TTY, il dato codificato è accettato ed analizzato nella routine chiamata "execute key" (esegui il tasto). I vari tasti premuti causeranno il salto (Branch) del programma alle appropriate subroutines necessarie per eseguire l'operazione desiderata. Dopo il completamento degli ordini dati dai singoli tasti, il programma rientra al "Get Key" ed aspetta la successiva pressione di un tasto.



CARTA DI FLUSSO

Figura 2.14

L'uscita dal programma TTY avverrà in seguito a:

1. pressione del tasto di reset
2. pressione del tasto G che avvia l'esecuzione del programma di applicazione, oppure
3. cambiamento del modo da TTY a tastiera/display.

Se, dopo un reset od una inizializzazione del sistema, si decide di dare esecuzione al modo tastiera/display (KB) il programma passerà direttamente ad eseguire le routines di scansione del display e della tastiera. Il programma continuerà a fare la scansione del display ("display cell") sinchè non sarà premuto uno dei tasti della tastiera (AK). La convalida dei tasti è eseguita durante un ciclo addizionale di scansione. Se il tasto è veramente premuto (e non si tratta di un disturbo), il programma svolge una routine chiamata riconoscimento tasto ("get key") che definisce l'esatto tasto premuto. Successivamente il programma si sposta alla routine "esecuzione tasto" ("execute key"), nella quale saranno svolte le diramazioni alle appropriate routines esecutive. Finalmente, dopo l'esecuzione del comando del tasto, il programma ritorna alla normale routine "display cell" ed aspetta che il tasto sia rilasciato.

Se non si preme alcun tasto il programma ritorna alla normale routine "display cell" ed aspetta il successivo comando della tastiera.

Sia nel modo TTY che nel KB, le routines di registrazione o riproduzione da registratore possono essere eseguite usando adatti comandi dalla tastiera scelta. In ambedue i casi, l'esecuzione della routine di trasferimento dal/sul nastro (Dump - Load) permette al programma di tornare alla posizione START che, come al solito, attiverà il display del KIM-1 o provocherà l'apparizione del messaggio di macchina pronta "KIM" sulla TTY.

Va notato l'uso del tasto di Stop per attivare l'ingresso di interruzione non mascherabile ($\overline{\text{NMI}}$) del microprocessore 6502. La pressione di questo tasto provocherà un arresto incondizionato dell'esecuzione del programma, una conservazione del registro di stato della macchina sullo stack ed un ritorno al controllo del programma operativo.

E' indispensabile un secondo ingresso d'interruzione chiamato $\overline{\text{IRQ}}$. Questo interrupt può essere stabilito dall'utente e provocherà un salto del programma alla locazione che l'utilizzatore definirà nel proprio programma.

CAPITOLO 3

I COMANDI OPERATIVI DEL KIM-1

Ora che avete una migliore idea di cosa contenga il KIM-1 e di come esso funzioni, possiamo ad esaminare le varie operazioni eseguibili.

Separeremo le possibili procedure operative in tre gruppi che forniranno specifiche indicazioni sull'uso della tastiera e del display compresi nel KIM-1, dell'operazione con il registratore e della telescrivente seriali.

3.1 USO DELLA TASTIERA E DEL DISPLAY

La tastiera comprende un totale di 23 tasti ed un deviatore a slitta. Vediamo per prima cosa la funzione di ciascun tasto:

da 0 a F - sedici tasti usati per definire il codice esadecimale degli indirizzi e dei dati.

AD - seleziona il modo di ingresso degli indirizzi.

DA - seleziona il modo di ingresso dei dati

+ - incrementa l'indirizzo di 1 senza cambiare il modo di ingresso

PC - richiama gli indirizzi immagazzinati nelle locazioni del contatore di programma.

RS - provoca un reset totale del sistema ed un ritorno al controllo del programma operativo.

GO - avvia l'esecuzione del programma a cominciare dall'indirizzo indicato sul display

ST - interrompe l'esecuzione di un programma e provoca il ritorno al programma operativo.

In un precedente capitolo si è visto che il display a sei cifre include una sezione di 4 cifre per gli indirizzi (a sinistra) ed una sezione a due cifre per i dati (a destra).

Usando solo la tastiera ed il display del KIM-1 potete eseguire una qualsiasi delle seguenti operazioni:

1. Scelta di indirizzo

Premete il tasto AD seguito da un qualsiasi gruppo di quattro cifre esadecimali della tastiera. L'indirizzo scelto appare sul display. Se si è commesso un errore nell'introdurre l'indirizzo, è sufficiente continuare a premere i tasti finchè l'indirizzo desiderato appare sul display. I due display di destra mostreranno i dati presenti in quel momento nell'indirizzo scelto.

2. Modifica dei dati

Dopo aver scelto l'indirizzo desiderato, premete il tasto DA seguito da due tasti esadecimali che definiscano esattamente i dati da immagazzinare nell'indirizzo selezionato. I dati introdotti verranno visualizzati nei due display di destra per indicare che il codice desiderato è stato inserito.

E' possibile selezionare un indirizzo di una cella ROM oppure un indirizzo di una cella di memoria che non esiste nel sistema, nel qual caso non è possibile cambiare il display dei dati dal momento che il sistema chiaramente non può inscrivere dati in una ROM od in una locazione di memoria inesistente.

3. Incremento degli indirizzi

Premendo il tasto + l'indirizzo visualizzato viene automaticamente incrementato di 1. Naturalmente i dati memorizzati nel nuovo indirizzo appariranno sul display. Questa operazione è utile quando si voglia leggere o modificare il contenuto di una serie di locazioni successive di memoria. Da notare che l'uso del tasto + non cambia il modo di ingresso. Se in precedenza avete premuto il tasto AD rimanete nel modo di ingresso degli indirizzi e se avete premuto il tasto DA rimanete nel modo di ingresso dati.

4. Richiamo del contatore di programma (PC)

Ogni volta che sia attivato il piedino di interruzione NMI del microprocessore 6502, l'esecuzione del programma in corso sarà fermata ed i registri interni del 6502 saranno conservati in speciali locazioni di memoria finchè il controllo del sistema non sarà ripreso dal programma operativo.

Nel sistema KIM-1 l'interruzione NMI può avvenire in risposta alla pres

sione del tasto ST (Stop) oppure, se si lavora nel modo a passo singolo (Single Step = SST), dopo che ogni istruzione del programma è eseguita in seguito alla pressione del tasto GO. Il tasto PC vi permette di richiamare automaticamente il valore che il contatore di programma aveva al momento in cui è avvenuta l'interruzione, anche se dopo l'interruzione avete eseguito numero se altre operazioni quali ispezionare il contenuto dei vari registri di macchina immagazzinati in specifiche locazioni di memoria. Se premete il tasto PC, i contenuti che il contatore di programma aveva al momento dell'interruzione sono richiamati nel campo degli indirizzi del display, e potrete proseguire l'esecuzione del vostro programma premendo il tasto GO.

5. Esecuzione di un programma

Scelto l'indirizzo di partenza del programma desiderato, premete il tasto GO e l'esecuzione del programma comincerà a partire dall'indirizzo apparso sul display.

6. Termine di un programma

Il tasto ST permette di arrestare l'esecuzione di un programma, attivando l'interruzione NMI del microprocessore 6502.

L'ST funzionerà correttamente solo se avrete memorizzato l'esatto vettore d'interruzione nella locazione 17FA e 17FB. Per la maggior parte del vostro lavoro con il sistema KIM-1 potrete memorizzare l'indirizzo 1C00 nelle posizioni indicate premendo i tasti:

AD			
1	7	F	A
DA		0	0
+		1	C

Quando avviene l'interruzione NMI il programma ritornerà alla posizione 1C00 e procederà alla conservazione di tutti i registri di macchina prima di restituire il controllo al programma operativo.

Dovrete ricordarvi di definire il vettore NMI tutte le volte che togliete corrente al sistema: una difettosa reazione al tasto ST può significare che avete dimenticato di definire il vettore NMI.

7. Esecuzione del programma a passo singolo (SST)

Nel processo di correzione di un nuovo programma potrete trovare molto utile eseguire il programma per singola istruzione. Per operare in questo modo spostate il deviatore a slitta SST nella posizione ON. In tal modo occorre premere il tasto GO ogni volta che si voglia far eseguire un passo di programma. Il display mostrerà l'indirizzo ed il dato della successiva istruzione da eseguire.

Va notato, però, che nel corso dell'esplorazione di un programma certi indirizzi appariranno saltati; un'istruzione di programma occuperà uno, due o tre bytes di memoria a seconda del tipo d'istruzione.

Nel modo a singola istruzione tutti i bytes coinvolti nell'esecuzione dell'istruzione sono toccati ed il programma si fermerà solo al primo byte dell'istruzione successiva.

Il metodo SST fa anche uso dell'interruzione NMI del microprocessore 6502. Inoltre il vettore NMI deve essere definito come descritto al punto (6) affinché il modo SST possa operare correttamente.

Quanto detto finora tratta tutte le operazioni standard che potrete eseguire dalla tastiera del KIM-1. Usando combinazioni delle suddescritte operazioni potrete raggiungere alcuni obiettivi specializzati come:

a. Definire il vettore IRQ

Ricorderete che nel microprocessore 6502 esiste un piedino di interruzione denominato IRQ. Volendolo usare, dovrete introdurre l'indirizzo al quale il vostro programma dovrà saltare. Tale indirizzo sarà memorizzato nelle posizioni 17FE e 17FF che corrispondono al "vettore" IRQ.

b. Interrogare lo stato di macchina

Abbiamo detto che dopo un'interruzione NMI in risposta alla pressione del tasto ST, oppure durante il modo SST, i contenuti dei vari registri di macchina sono memorizzati in particolari locazioni di memoria. Se desiderate ispezionare queste locazioni, i loro indirizzi sono:

OOEF = PCL Byte meno significativo)del contatore
OOF0 = PCH Byte più significativo (di programma
OOF1 = Status Register o registro di stato (P)
OOF2 = Stack pointer o puntatore dello Stack
(SP)
OOF3 = Accumulatore (A)
OOF4 = Registro indice Y
OOF5 = Registro indice X

3.2 USO DEL REGISTRATORE

Ci sono due utilizzazioni possibili del registratore: potete trasferire dati dalla memoria del KIM-1 e registrarli sul nastro, oppure potete leggere un nastro registrato in precedenza trasferendo i dati ivi contenuti nella memoria del KIM - 1.

Registrazione

La procedura per la registrazione su nastro audio richiede l'esecuzione dei seguenti passi:

1. Definizione di un numero di identificazione (ID) per il blocco di dati da registrare. Questo numero di due cifre è caricato nell'indirizzo 17F9. Non usate ID = 00 oppure FF
2. Definizione dell'indirizzo di partenza del blocco dei dati da trasferire. Questo indirizzo deve essere caricato nelle locazioni:

17F5 = 2 cifre meno significative dell'indirizzo
di partenza (SAL)
17F6 = 2 cifre più significative dell'indirizzo
di partenza (SAH)

3. Definizione dell'indirizzo di fine blocco, aggiungendo 1 all'ultimo indirizzo del blocco di dati da registrare.

Tale indirizzo deve essere caricato nelle locazioni:

17F7 = 2 cifre meno significative dell'indirizzo
di fine (EAL)
17F8 = 2 cifre più significative dell'indirizzo
di fine (EAH)

Tanto per fare un esempio, supponiamo di voler registrare il blocco di dati che va dall'indirizzo 0200 all'indirizzo 03FF incluso (le pagine 2 e 3). A tale blocco vorrete assegnare un numero di identificazione (ID = 06). Usando la tastiera del KIM-1, caricate i dati negli indirizzi indicati come segue:

Indirizzi		Dati
00F1	=	00 (esclude il modo di funzionamento decimale)
17F5	=	00 (SAL)
17F6	=	02 (SAH)
17F7	=	00 (EAL) (= 03FF + 1 = 0400
17F8	=	04 (EAH) (
17F9	=	06 (ID)

Da notare che per una corretta operazione l'indirizzo finale deve essere maggiore di quello di partenza.

4. Se state usando una cassetta nuova sulla quale non sono stati registrati dati in precedenza, inserite la cassetta nel registratore e riavvolgetela fino alla sua posizione di partenza.
5. Posizionatevi sull'indirizzo di avviamento del programma di registrazione su nastro. Questo indirizzo è 1800 (DUMPT).
6. Mettete il registratore sulla posizione di incisione ed attendete alcuni secondi che il nastro cominci a muoversi.
7. Premete il tasto GO ed il processo di incisione comincerà. Il display si spegnerà per un certo periodo dopo di che si illuminerà nuovamente mostrando 0000 xx. Ciò significa che il blocco dei dati scelto è stato registrato.

8. Ora potete fermare il nastro oppure lasciar passare alcuni secondi in modo da lasciare una sezione di nastro vuota.

Caricamento dei dati dal registratore

Per caricare i dati dal registratore nella memoria del KIM-1 si richiede l'esecuzione dei seguenti passi:

1. Escludete il modo di funzionamento decimale memorizzando 00 nella locazione 00F1.
Definite il numero ID del programma da caricare dal nastro e memorizzatelo nella locazione di indirizzo 17F9.
2. Scegliete l'indirizzo di partenza del programma di carico da nastro, ossia 1873 esadec. (LOADT)
3. Premete il tasto GO: il KIM-1 è in attesa di vedere apparire i dati dal nastro..
4. Introducete la cassetta nel registratore e, supponendo che non conosciate in quale posizione del nastro sia caricato il programma, riavvolgete il nastro sino all'inizio. Controllate il volume dell'apparecchio.
5. Avviate il nastro in posizione "ascolto" e osservate se il nastro si muove.
6. Attendete che il display del KIM - 1 si riaccenda mostrando 0000 xx. Ciò significa che il blocco dei dati è stato memorizzato con successo nella memoria del KIM-1.
Se il display si riaccende mostrando FFFF xx, è stato riconosciuto il blocco corretto dei dati ma c'è stato un errore rilevato durante l'operazione di lettura. Se il nastro continua a girare ed il display non si riaccende vuol dire che il sistema non è riuscito a trovare il programma con il numero ID assegnato.
7. Se nel passo (1) avete definito un ID = 00, il numero ID registrato sul nastro verrà ignorato ed il sistema leggerà il primo blocco valido di dati che incontrerà sul nastro.
I dati letti dal nastro saranno caricati negli indirizzi di memoria specificati sul nastro.
8. Se nel passo (1) avete scelto un ID = FF, il numero di identificazione inciso sul nastro verrà ignorato ed il sistema leggerà il primo blocco valido di dati che troverà sul nastro. Inoltre il blocco dei dati sarà caricato a partire dalla locazione di memoria il cui indirizzo è specificato nelle locazioni 17F5 e 17F6 (SAL, SAH), invece che nelle locazioni specificate nel nastro.

Operazioni speciali

Il sistema KIM-1 effettua la registrazione dei dati sul nastro audio grazie ad uno specifico formato illustrato nell'appendice E. Ciascun blocco dati registrato è preceduto da un gruppo di caratteri di sincronizzazione e da un codice di identificazione che caratterizza lo specifico blocco. I blocchi di dati possono essere di lunghezza arbitraria.

Con un po' di precauzione, non ci sono controindicazioni perchè si carichino parecchi blocchi di dati sullo stesso nastro. Se state caricando dei blocchi in sequenza e non avete riavvolto il nastro nel passaggio da un blocco all'altro, dovreste solo specificare i parametri di ciascun nuovo blocco (ID, SAL, SAH, EAH, EAL) e procedere con la registrazione del nuovo blocco.

Se il nastro è stato riavvolto, sarà indispensabile conoscere il numero di identificazione ID dell'ultimo blocco di dati registrato. Riavvolgete il nastro sino al suo inizio ed inserite i parametri richiesti per la lettura dell'ultimo blocco di dati registrato. Dopo la lettura di questo blocco arrestate il nastro, al quale potrete ora aggiungere uno o più nuovi blocchi.

Se lo desiderate, potete aggiungere messaggi a voce tra i blocchi di dati registrati sul nastro. Il sistema KIM-1 ignorerà questi messaggi audio mentre sta leggendo il nastro, ma voi aggiungendo un'auricolare od un altoparlante in parallelo con il KIM-1, potete sentire i messaggi incisi.

Raccomandiamo di NON tentare di registrare blocchi di dati in sezioni di nastro già usate per precedenti registrazioni, in quanto la variabilità della velocità del nastro e della lunghezza dei blocchi, potrebbe provocare sovrapposizioni con i blocchi adiacenti, che di conseguenza verrebbero letti in maniera non corretta.

3.3 USO DELLA TELESKRIVENTE

L'aggiunta di una telescrivente seriale (come la teletype modello ASR 33) permette di eseguire una varietà di operazioni speciali. In ogni caso voi definite le operazioni desiderate premendo gli opportuni tasti

mentre contemporaneamente si produce una documentazione stampata di ciascuna operazione. Se la teletype è equipaggiata con un lettore-perforatore di nastro, potrete generare e leggere nastri perforati usando il sistema KIM-1.

Utilizzando la telescrivente seriale potrete eseguire le seguenti operazioni:

1. Scelta di un indirizzo

Battete quattro tasti esadecimale (da 0 ad F) per definire l'indirizzo desiderato, quindi premete la barra di spaziatura. La stampante risponderà mostrando il codice dell'indirizzo scelto seguito da due cifre di codice esadecimale per il dato memorizzato nella locazione di indirizzo scelta.

Battere: 1234 SPAZIO
la stampante risponde: 1234 AF

mostrando che il dato AF è immagazzinato nella locazione 1234.

2. Modifica dei dati

Scegliete un indirizzo come nella sezione precedente e battete due caratteri esadecimale per definire il dato da memorizzare a questo indirizzo. Battete quindi il tasto (.) per autorizzare la modifica del dato all'indirizzo scelto:

[illegible]

Notate che l'indirizzo scelto (1234) è stato modificato e che il sistema si incrementa automaticamente all'indirizzo successivo (1235).

Gli zeri di testa non devono essere introdotti per ciascun indirizzo o campo di dati; per esempio:

EF	SPAZIO	sceglie l'indirizzo 00EF
E	SPAZIO	sceglie l'indirizzo 000E
A	(.)	introduce il dato 0A
	(.)	introduce il dato 00 eccetera

3. Passaggio al successivo indirizzo

Battete CR per passare al successivo indirizzo senza modificare quello corrente:

Vedi stampato:	1234 AF	
Battere:		CR
La stampante risponde:	1235 B7	
Battere:		CR
La stampante risponde:	1236 C8	eccetera

4. Passaggio all'indirizzo precedente

Battete LF per indietreggiare al precedente indirizzo:

Vedi stampato:	1234 AF	
Battere:		LF
La stampante risponde:	1233 9D	
Battere:		LF
La stampante risponde:	1232 8E	eccetera

5. Interruzione dell'operazione corrente

Battete RUBOUT per terminare l'operazione corrente: verrà stampato il messaggio di macchina pronta ("KIM") che indica che si può procedere con una nuova operazione:

Battere:	1264	RUBOUT
Stampante:	KIM	
	xxxx xx	
Battere:	1234	SPAZIO
Stampante:	1234 AF	

Nell'esempio il tasto RUBOUT è usato per correggere una scelta errata dell'indirizzo.

Il tasto RUBOUT deve essere premuto dopo ciascuna pressione del reset del KIM-1 per permettere al programma operativo di definire la cadenza dei bit seriali per la telescrivente.

6. Caricamento da nastro perforato

Nastri adatti per l'uso nel sistema KIM-1 sono generati usando il formato descritto in appendice F. Per leggere tale nastro si procede come segue:

- 1) caricate il nastro perforato sul meccanismo di lettura,
- 2) battete L,
- 3) attivate il lettore di nastro.

Il nastro di carta avanzerà ed i dati saranno caricati agli indirizzi specificati sul nastro. Una copia stampata dei da ti letti sarà generata simultaneamente.

Cifre di controllo sono generate durante la lettura del nastro di carta e sono confrontate con le somme di controllo (check-sums) contenute nel nastro. Un errore nella somma di controllo causerà un messaggio di errore che apparirà sulla copia stampata.

7. Punzonatura del nastro di carta

Il sistema può essere usato per punzonare nastri di carta secondo il formato descritto in appendice F. La procedura per generare tali nastri è la seguente:

- 1) Definite l'indirizzo di partenza e l'indirizzo di fine del blocco di dati da punzonare sul nastro di carta.
- 2) Caricate un nastro di carta vergine su l'unità perforatrice ed attivate la medesima.

Battete:	17F7	SPAZIO
Stampante:	17F7 xx	
Battete:	FF(.)	
Stampante:	17F8 xx	
Battete:	03(.)	
Stampante:	17F9 xx	
Battete:	200	SPAZIO
Stampante:	0200 xx	

Avete così caricato l'indirizzo finale (03FF) nelle locazioni di indirizzo 17F7 (EAL) e 17F8 (EAH). L'indirizzo di avviamento (0200) è scelto come indicato.

- 3) Ora battete Q.

Il nastro di carta avanzerà e procederà la punzonatura. Contemporaneamente sarà stampata su carta una copia scritta dei dati.

8. Lista del programma

Per realizzare una copia stampata su carta del contenuto della memoria, la procedura è la stessa di quella usata per la punzonatura del nastro, salvo il fatto che non viene attivato il mecc canismo di punzonatura.

9. Esecuzione del programma

Per iniziare l'esecuzione di un programma usando la tastiera TTY

si deve seguire la seguente procedura:

- 1) Introdurre l'indirizzo di inizio del programma
- 2) Battere G

Per esempio per eseguire il programma che inizia dalla loca
zione di indirizzo 0200:

Battete: 200 SPAZIO

Stampante: 0200 xx

Battete: G

L'esecuzione del programma comincia dalla locazione 0200 e continuerà sinchè non saranno premuti i tasti ST o RS del KIM-1. La possibilità dell'esecuzione a passo singolo può essere impiegata anche con la TTY.

CAPITOLO 4

UN'APPLICAZIONE PRATICA

Non sarebbe possibile descrivere in questo manuale tutte le applicazioni possibili o tutte le tecniche di programmazione. D'altronde, ora che vi sono diventati familiari gli elementi basilari e le procedure operative del sistema, questa sezione vi mostrerà come applicare quanto avete appreso con un semplice ma realistico esempio.

Questo comprende la generazione di un'onda quadra di frequenza variabile che sarà connessa ad un altoparlante in modo da produrre una nota udibile. La frequenza della nota sarà scelta usando un gruppo di sette pulsanti. Procederemo con l'esempio definendo l'interfaccia, scrivendo ed inserendo il programma, ed eseguendolo. Infine studieremo una serie di tecniche di correzione (debugging) del programma utilizzabili per ciascun nuovo programma che voi potrete scrivere.

4.1 DEFINIZIONE DELL'INTERFACCIA

Ricorderete che 15 piedini I/O sono collegati al connettore di applicazione, provenienti dal circuito 6530-003. La logica ed i dettagli circuitali dei piedini I/O sono qui descritti nell'appendice H ed inoltre nella sezione 1.6 del manuale Hardware (Interfaccia delle periferiche/dispositivi di memoria - MCS 6530).

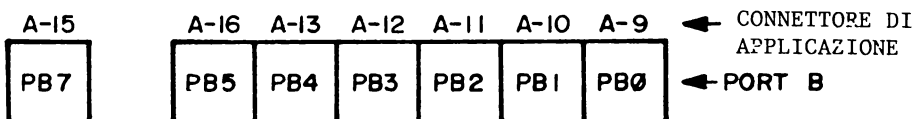
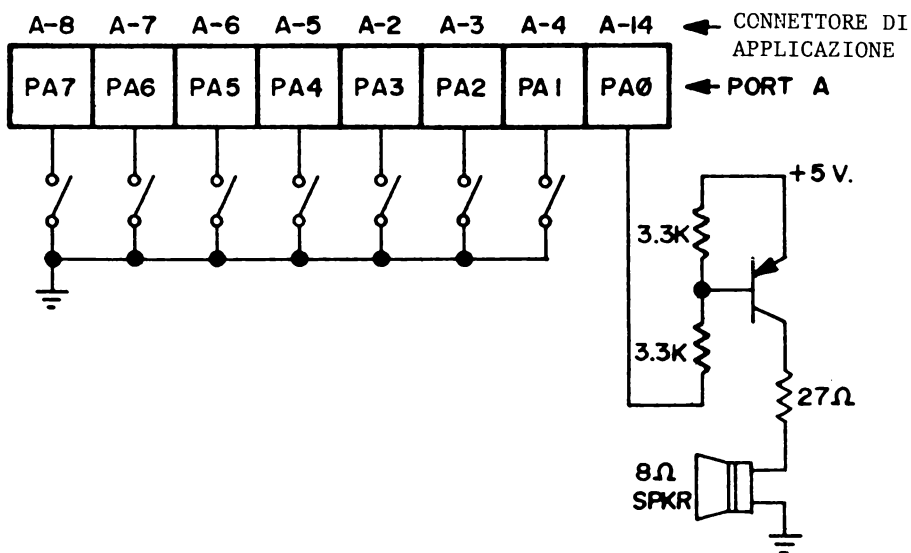
Per il nostro esempio di applicazione useremo otto di questi piedini I/O. Un piedino (PA0) sarà usato come uscita per fornire l'onda quadra ad un circuito amplificatore con altoparlante. Gli altri sette piedini I/O (da PA1 a PA7) sono ingressi ed a ciascuno di loro viene connesso un pulsante.

La fig. 4.1 mostra la configurazione circuitale dell'esempio. Da notare che i restanti piedini I/O (il port PB) non sono usati in questo esempio.

Nota:

PORT: è un insieme di 8 piedini di ingresso e uscita.

Per i pulsanti connessi ai piedini di ingresso, stabiliamo che l'interruttore sia definito come "0" logico quando è aperto, ed "1" logico quando è chiuso. Connettendo gli interruttori a massa produrremo però l'effetto contrario, per cui dovremo ricordarci di fare il complemento dello stato dei pulsanti con il software quando scriveremo il nostro programma. Inoltre dobbiamo stabilire che l'interruttore connesso a PA1 deve essere l'LSB (Least significant bit = bit meno significativo) e l'interruttore connesso a PA7 deve essere l'MSB (Most significant bit = bit più significativo) della parola binaria di sette bit formata da tutti e sette gli interruttori. In questo modo lo stato degli interruttori può definire un numero binario che può variare da 0 (tutti gli interruttori aperti) a 127 decimale (tutti gli interruttori chiusi).



(IL PORT B NON E' USATO IN QUESTO ESEMPIO DI APPLICAZIONE)

Esempio di applicazione per generatore di note

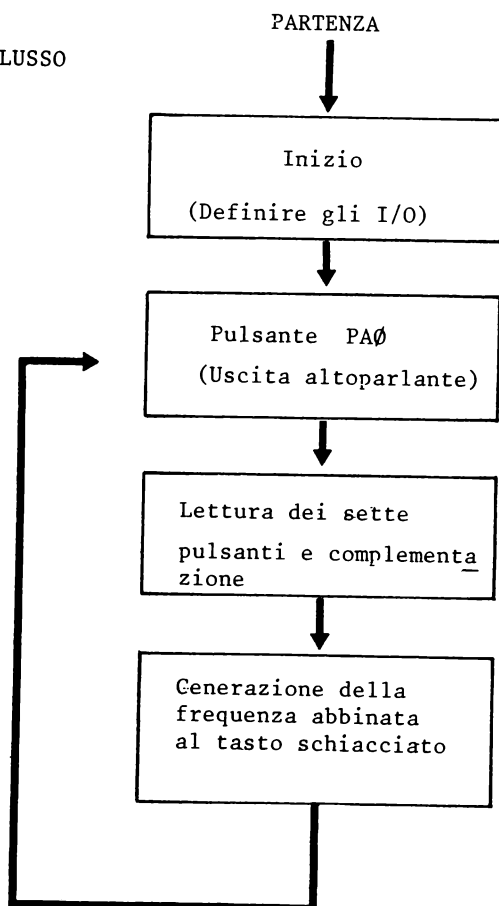
Figura 4.1

4.2 SCRITTURA DEL PROGRAMMA

Definita l'interfaccia per la nostra applicazione, passiamo alla scrittura del programma. Il lavoro si sviluppa in quattro fasi:

- 1) Generazione di un diagramma di flusso
- 2) Generazione di un codice in linguaggio assembler
- 3) Analisi del programma
- 4) Generazione del codice in linguaggio macchina

DIAGRAMMA DI FLUSSO



In breve, il nostro diagramma di flusso mostra un primo passo dell'inizializzazione del sistema. Durante questo passo dobbiamo definire la configurazione I/O del sistema così che il piedino PA0 divenga l'uscita all'altoparlante ed i piedini da PA1 a PA7 divengano ingressi dei sette interruttori.

Dopo l'inizializzazione, è sistemato un anello (loop) che comincia invertendo lo stato di PA0. Successivamente viene letto lo stato degli interruttori ed il dato viene complementato in modo da produrre il corretto significato dello stato degli interruttori. Il valore così letto è usato per definire un ritardo prima di tornare all'inizio dell'anello e commutare nuovamente lo stato di PA0. Pensandoci un momento si vedrà che questo anello produrrà un'onda quadra con una frequenza determinata dalla posizione "aperto" e "chiuso" dei sette interruttori.

Programma in linguaggio assembler

Il nostro successivo obiettivo è di convertire un semplice diagramma di flusso in un programma. Il programma dapprima è scritto in linguaggio assembler. (Consultate il manuale di programmazione per diventare familiari con tutte le istruzioni possibili per il 6502: in particolare leggete l'appendice B e il prontuario delle istruzioni).

La fig. 4.2 mostra l'esempio applicativo programmato in linguaggio assembler.

ETICHETTA	CODICE OPERATIVO	CODICE OPERANDO	N° CICLI MACCHINA	COMMENTI
INIT	LDA	$\neq \$01$	2	Definisci I/O O=Ingresso I=Uscita
	STA	PADD	4	PADD=Registro direzione dati PORT A
START	INC	PAD	6	Commuta PAØ. (PA1 - PA7 immutati)
READ	LDA	PAD	4	Acquisisci i tasti nell'accumulatore
	EOR	$\neq \$FF$	2	Complementa l'accumulatore
	LSR	A	2	SPOSTA (Shift) l'accumulatore di un bit a destra
	TAX		2	Trasferisci il conteggio finale nel registro X
DELAY	DEX		2	Genera un tempo determina <u>to</u> to.....
	BPL	DELAY	3,2dal contenuto del registro X
	BMI	START	3	Vai a START
PADD	$=\$1701$			Valore dell'etichetta PADD
PAD	$=\$1700$			Valore dell'etichetta PAD

LISTA DEL PROGRAMMA IN LINGUAGGIO ASSEMBLATORE

Figura 4.2

Ciascuna riga del programma è divisa in diversi campi:

- Un campo delle etichette che permette di assegnare un "nome" ad una specifica locazione del programma.
- Un campo dei codici di operazione (codice operativo) nel quale viene definita l'esatta istruzione da eseguire.
- Un campo degli operandi dove l'esatto dato richiesto dalla istruzione è definito insieme con certi simboli che definiscono modi di indirizzamento o il formato del dato.
I simboli incontrati generalmente nei manuali della Mos Technology Inc. sono:

\neq	Indirizzamento immediato
\$	Codice esadecimale
@	Codice ottale
%	Codice binario
'	Letterale ASCII
=	Uguaglia una etichetta ad un valore

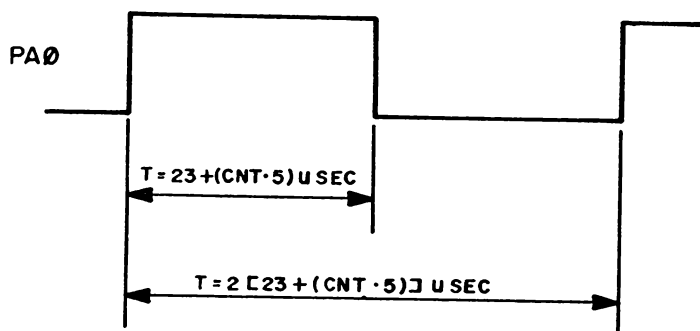
- Un campo del ciclo di macchina che definisce il numero totale di cicli macchina occorrenti per eseguire un'istruzione. (Questa informazione si ricava dall'appendice B del manuale di programmazione).
- Un campo dei commenti dove il programmatore può definire lo scopo degli specifici passi di programma.

Analisi del programma

L'inclusione dell'informazione sui cicli di macchina nel diagramma di programmazione (Fig.4.2) permette di analizzare le esatte relazioni di temporizzazione attinenti al nostro esempio di programma. Notate che il sistema KIM-1 lavora con una frequenza fissa generata da un oscillatore da 1 MHz quindi ogni ciclo macchina è di 1 μ S.

Quindi un'istruzione come "INC PAD", che richiede 6 cicli macchina, sarà eseguita in un periodo di 6 μ S.

Contando il totale dei cicli macchina intercorrenti tra ciascuna commutazione di PAØ, si può impostare un'equazione per la frequenza dell'onda quadra. La frequenza effettiva è determinata dalla posizione dei sette interruttori, dal numero dei cicli macchina che passano tra ciascuna commutazione di PAØ, dalla frequenza base di 1 MHz del sistema KIM-1. La fig. 4.3 mostra la forma dell'onda quadra su PAØ e l'equazione per calcolare l'esatta frequenza.



$$FREQ = \frac{1}{T} = \frac{10^6}{46 + 10 \cdot CNT} \text{ CPS}$$

NOTA: CNT (COUNT) corrisponde al valore nel registro indice X che è stato calcolato dai sette interruttori $0 \leq CNT \leq 127$

USCITA ONDA QUADRA

Figura 4.3

Codifica in linguaggio macchina

Il successivo passo è di convertire il programma in linguaggio assemblatore in un programma scritto in 'linguaggio macchina'.

Il metodo più rapido ed a miglior prova di errore per eseguire questa conversione è di usare il Programma Assemblatore.

Se decidete di non usare questo metodo, dovrete convertire il vostro programma originario in codice macchina usando tecniche di "carta e matita". In particolare dovrete costruirvi una tavola simile a quella mostrata in fig. 4.4.

INDIRIZZI	ISTRUZIONI			CODICE SORGENTE	
	BYTE 1	BYTE 2	BYTE 3	ETICHETTA OPERATIVO	OPERANDO
0200	A9	01		INIT	LDA ≠ \$01
0202	8D	01	17		STA PADD
0205	EE	00	17	START	INC PAD
0208	AD	00	17	READ	LDA PAD
020B	49	FF			EOR ≠ \$FF
020D	4A				LSR A
020E	AA				TAX
020F	CA			DELAY	DEX
0210	10	FD			BPL DELAY
0212	30	F1			BMI START
0214					

TABELLA DEI CODICI DI LINGUAGGIO MACCHINA

Figura 4.4

Il codice sorgente contenuto nel vostro programma in linguaggio assemblatore (fig. 4.2) è inserito per primo nella tabella. Una colonna è prevista per permettervi di definire gli specifici indirizzi ai quali un'istruzione è locata. La colonna delle istruzioni prevede spazio per definire istruzioni ad uno, due o tre bytes.

(Consultate l'appendice B del manuale di programmazione oppure la vostra scheda di programmazione per gli specifici Codici Operativi (Op Code)).

La prima istruzione del programma sorgente è LDA \neq \$01 che, una volta tradotta, significa carica l'accumulatore con il byte memorizzato nella successiva locazione di programma (esadecimale 01). Questo è il modo di indirizzamento immediato definito dal simbolo \neq . Il codice operativo per LDA \neq è A9. Tale valore è introdotto nella prima delle colonne "Istruzioni". La seconda colonna contiene il valore esadecimale 01 definito dall'istruzione sorgente. L'indirizzo iniziale del programma è inserito nella colonna "Indirizzi" come 0200 (una scelta arbitraria). L'istruzione totale LDA \neq \$01 ora occupa le locazioni di indirizzo 0200 e 0201.

Il successivo indirizzo disponibile è 0202 che è inserito nella colonna "Indirizzi" per la successiva istruzione sorgente. In questo modo procederete attraverso tutte le istruzioni sorgenti decodificando le ed inserendo uno, due o tre bytes di codice macchina, secondo la necessità, nelle colonne "Istruzioni". La colonna "Indirizzi" conterrà l'indirizzo del primo byte del codice macchina (Op Code) per ciascuna istruzione.

Se l'operando dell'istruzione sorgente è un simbolo, l'indirizzo al quale il simbolo è stato uguagliato verrà introdotto come 2° e 3° byte delle colonne "Istruzioni". Per esempio l'istruzione sorgente "INC PAD" richiede l'incremento del valore memorizzato nella locazione PAD che nel nostro programma assembler è : PAD = 1700. Quindi l'indirizzo 1700 è introdotto come secondo e terzo byte della istruzione sorgente INC PAD. (vedi fig. 4.4). Da notare inoltre che quando si introduce un indirizzo, come 1700, il byte meno significativo (00) è introdotto per primo ed immediatamente dopo il codice operativo, ed il byte più significativo (17) è introdotto successivamente come terzo byte dell'istruzione.

Quando si lavora con istruzioni di salto (Branch: BPL, BMI, ecc.) avrete bisogno di calcolare l'esatto valore dello spostamento. Questo può essere sia positivo (branch forward = salto in avanti) che negativo

(branch backward = salto indietro). Per l'approfondimento del concetto di branch potete riferirvi alla sezione 4.1.1 del manuale di programmazione al paragrafo intitolato "concetto base del salto (branch) relativo".

Come esempio, l'istruzione sorgente "BMI START" (vedi figg.4.2 e 4.4) richiede un salto all'indietro di (-15) locazioni sino all'indirizzo denominato START (Dall'indirizzo 0213 all'indietro sino alla locazione 0205 compresa. Il complemento a 2 di -15 è F1 esadecimale, che dovrete inserire allà locazione 0212). Se il salto dovesse avvenire in avanti, bisognerà inserire il valore positivo dello spostamento anzichè il complemento a 2.

4.3 INTRODUZIONE DEL PROGRAMMA

Una volta tradotto il programma in codici di linguaggio macchina, potrete inserire gli indirizzi del programma ed i codici dei dati elencati in fig. 4.4, procedendo nel modo seguente:

<u>Tasto da premere</u>	<u>Lettura sul display</u>
AD 0 2 0 0	0200 xx
DA A 9	0200 A9
+ 0 1	0201 01
+ 8 D	0202 8D
+ 0 1	0203 01
+ 1 7	0204 17
+ 'E E	0205 EE
+ 0 0	0206 00
+ 1 7	0207 17
+ A D	0208 AD
+ 0 0	0209 00
+ 1 7	020A 17
+ 4 9	020B 49
+ F F	020C FF
+ 4 A	020D 4A
+ A A	020E AA
+ C A	020F CA
+ 1 0	0210 10
+ F D	0211 FD
+ 3 0	0212 30
+ F 1	0213 F1

SEQUENZA DEI TASTI PER L'INTRODUZIONE DEL PROGRAMMA

Figura 4.5

4.4 ESECUZIONE DEL PROGRAMMA

Quando il programma è introdotto potrete procedere alla sua esecuzione. Per prima cosa, se non è stato fatto in precedenza, bisogna definire il vettore NMI. Il vettore si introduce come segue:

<u>Tasto da premere</u>	<u>Lettura sul display</u>
AD 1 7 F A	17FA xx
DA 0 0	17FA 00
+ 1 C	17FB 1C

Questa procedura assicura che il tasto ST è operante all'arresto del programma. Scegliete ora l'indirizzo di partenza del programma (0200) e cominciate l'esecuzione come segue:

<u>Tasto da premere</u>	<u>Lettura sul display</u>
AD 0 2 0 0	0200 A9
GO	(Dark)

Il programma verrà ora eseguito. Se i sette interruttori di selezione saranno tutti aperti, non udrete probabilmente alcun suono dall'altoparlante in quanto la frequenza dell'onda quadra è troppo alta. Se tutti gli interruttori sono chiusi, udrete nell'altoparlante la minima frequenza generata con il programma così come è scritto. Potrete provare con altre combinazioni di commutazione degli interruttori per udire una varietà di note dall'altoparlante.

La pressione del tasto ST causerà la fermata dell'esecuzione del programma (la nota finirà) ed il display si illuminerà di nuovo, mostrando l'indirizzo ed il dato della successiva istruzione da eseguire (probabilmente 020F oppure 0210 dal momento che stanno nell'anello di ritardo dove il programma passa la maggior parte del suo tempo).

4.5 CONTROLLO E MODIFICA DEL PROGRAMMA

Se il programma non opera correttamente, la procedura di controllo

(debugging) necessaria comprende i seguenti passi:

Passo 1: lista del programma

Per prima cosa assicuratevi di aver introdotto correttamente i passi del programma. Selezionate l'indirizzo di partenza (AD 0200) ed osservate che sia visualizzato il dato corretto (A9). Ora, usando il tasto + visualizzate le restanti locazioni di programma controllando che in ogni locazione siano memorizzati i dati corretti.

Passo 2: analisi del programma a passo singolo

Seguite le procedure elencate nel paragrafo 4.4 per l'esecuzione del programma, ma, prima di premere il tasto GO, piazzate il deviatore a slitta SST nella posizione ON. Premete il tasto GO e la prima istruzione verrà eseguita. Il display si riaccenderà indicando che il programma operativo è di nuovo sotto il controllo del sistema. L'indirizzo visualizzato sarà l'indirizzo del primo byte della successiva istruzione da eseguire. Potrete premere nuovamente il tasto GO ed eseguire la successiva istruzione oppure potrete decidere di indagare i cambiamenti nei contenuti dei registri di macchina memorizzati nelle locazioni di memoria specificate (vedi fig.2.13). La procedura illustrata nella fig. 4.6 dà una buona indicazione delle varie operazioni che potrete eseguire nel modo SST.

Passo 3: controllo delle operazioni di I/O

Dopo avere verificato il programma introdotto e dopo che l'esecuzione del medesimo nel modo SST si rivela corretto, potete verificare il buon funzionamento della vostra specifica configurazione di I/O. Ricordate però, che scrivere o leggere da ciascun port I/O è la stessa cosa che leggere o scrivere su qualsiasi altra locazione di memoria del sistema. Quindi, se scegliete l'indirizzo di un port I/O, il display vi mostrerà il codice esadecimale per il dato letto a tale indirizzo, e quindi indicherà direttamente lo stato di ciascun piedino I/O del port. Per esempio l'indirizzo del port I/O usato nel nostro esempio è 1700. Se premete AD 1 7 0 0 il display mostrerà il codice esadecimale corrispondente alla sistemazione dei vostri interruttori di selezione.

Se cambiate la posizione degli interruttori, vedrete il cambiamento del codice esadecimale nel display.

Ora, lasciando lo stesso indirizzo (1700) premete il tasto DA.

Premendo qualcuno dei tasti esadecimali da 0 ad F, iscriverete il dato al port I/O. Siccome sette dei piedini del port I/O sono definiti come ingressi, soltanto uno (PAØ) agirà come uscita e risponderà ai dati introdotti dalla tastiera. Provate a premere alternativamente e con rapidità i tasti 0 ed 1 e potrete udire un crepitio nell'altoparlante indicante che state commutando effettivamente il piedino PAØ.

Questo concetto dell'uso della tastiera e del display per comandare e verificare l'operazione del port I/O è una tecnica ricorrente per il controllo delle sezioni hardware di molte applicazioni particolari.

<u>Tasto da premere</u>	<u>Lettura sul display</u>	<u>Commenti</u>
AD 0 2 0 0	0200 A9	Seleziona il 1° indirizzo di istruzioni
SST)(⁰ _N	0200 A9	Posiziona SST in ON: tutti gli interruttori aperti
GO	0202 8D	L'accumulatore è caricato con \$01
GO	0205 EE	PADD caricato
GO	0208 AD	PAØ commutato
GO	020B 49	Valore degli interruttori (PA1-PA7) caricato
GO	020D 4A	Accumulatore complementato
GO	020E AA	Accumulatore spostato a destra di 1 bit
AD 0 0 F 3	00F3 xx	Visualizzazione dell'accumulatore
+	00F4 xx	Visualizzazione dell'indice Y
+	00F5 00	Visualizzazione dell'indice X
PC	020E AA	ricarica PC (TAX sarà eseguito successivamente)
GO	020F CA	Accumulatore caricato in indice X
AD 0 0 F 3	00F3 00	Visualizzazione dell'accumulatore
+	00F4 xx	Visualizzazione dell'indice Y
+	00F5 00	Visualizzazione dell'indice X (A=0 → X)
PC	020F CA	Ricarica PC
GO	0210 10	DEX completato
AD 0 0 F 5	00F5 FF	Visualizzazione dell'indice X (X < 0)
PC	0210 10	Ricarica PC
GO	0212 30	Nessun salto(per risultato di DEX <u>non</u> positivo)
GO	0205 EE	Salto (per risultato di DEX negativo)

ANALISI DEL PROGRAMMA A PASSO SINGOLO

Figura 4.6

CAPITOLO 5

ESPANSIONE DEL SISTEMA

Nelle precedenti sezioni avete appreso che il microprocessore MCS 6502 è capace di indirizzare direttamente fino a 65.536 locazioni (bytes) di memoria (usualmente abbreviato a 65 K dove K significa 1.024 locazioni di memoria). In questa sezione discuteremo per prima cosa le tecniche per aggiungere locazioni di memoria o di I/O al sistema, e successivamente, l'opportuno trattamento dei vettori di interruzione in un sistema espanso.

5.1 ESPANSIONE DELLE MEMORIE E DEGLI I/O

Nel sistema KIM-1 la conduzione dei dati in ingresso/uscita è trattata esattamente nello stesso modo dei trasferimenti da e verso qualunque altra locazione di memoria del sistema. Non ci sono istruzioni che si riferiscano specificamente ai trasferimenti degli I/O. Invece, il trasferimento dei dati è ottenuto leggendo da o scrivendo sui registri connessi alla linea dei dati ed ai piedini I/O in particolari dispositivi di interfaccia I/O (come il circuito integrato 6530).

Questi registri hanno uno specifico indirizzo nel sistema proprio come avviene per ogni altra locazione di memoria. Quindi, quando parliamo di espansione della memoria del sistema, noi definiamo i metodi per espandere sia la memoria reale (RAM, ROM, PROM, ecc.), sia i port I/O dal momento che ambedue sono trattate esattamente allo stesso modo una volta che siano stabilite le assegnazioni degli indirizzi.

La prima e più facile espansione di memoria è l'aggiunta di spazio di memoria fino a 4 K. Ricorderete che gli 8 K più bassi sono definiti da un decodificatore di indirizzi incluso nel modulo KIM-1 (dispositivo U4 nello schema). Le otto uscite di questo decodificatore (da K0 a K7) definiscono ciascuna un blocco di indirizzi da 1K negli 8K inferiori della mappa della memoria. Tre delle uscite (K5, K6 e K7) sono usate per selezionare locazioni di ROM, RAM, I/O e temporizzatore nei due circuiti 6530, mentre una quarta (K0) è usata per selezionare le 1.024 locazioni della memoria statica RAM. Le restanti quattro uscite (K1, K2, K3, K4) non sono usate nel modulo KIM-1 ma invece sono portate al connettore di espansione per potere essere usate come selettori di chip per aggiunte di memorie oppure I/O.

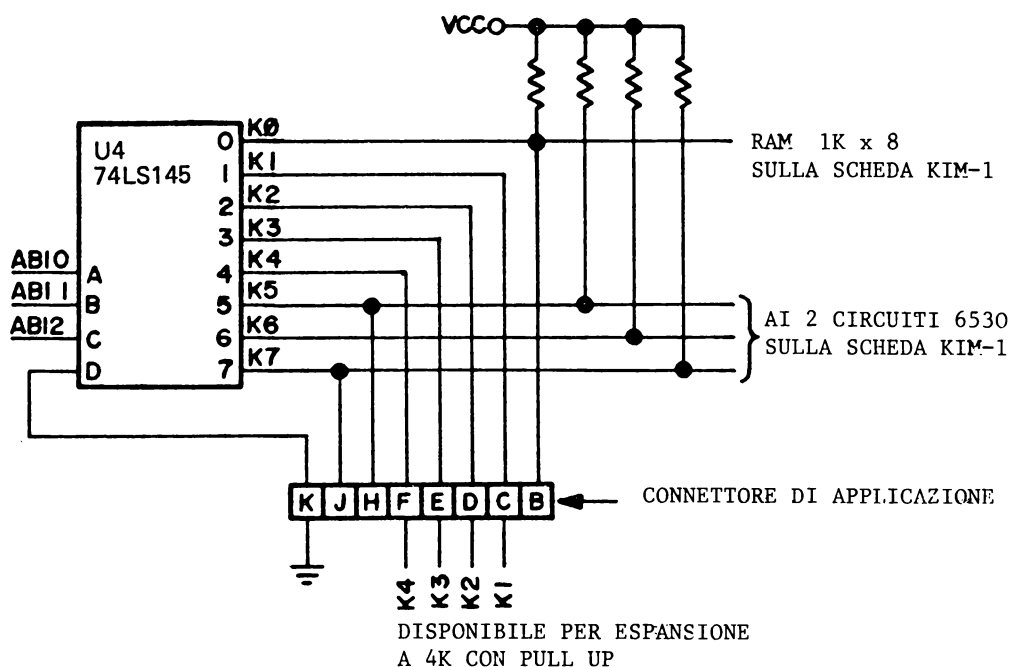
La fig.5.1 mostra il metodo adatto per derivare i segnali di selezione del chip per 4K aggiuntivi di memoria. Uno dei piedini di ingresso del decodificatore (D) è stato collegato al connettore di applicazione. E' stato questo piedino che vi abbiamo chiesto di collegare a massa nel capitolo 1 di questo manuale. Fintanto che questo punto resta connesso a massa, il decodificatore sceglierà sempre gli indirizzi degli 8 K inferiori del campo di memoria, senza tener conto dello stato di AB13, AB14 ed AB15.

Se volete espandere la memoria e gli indirizzi I/O oltre gli 8K inferiori, dovete fare in modo di rendere selezionabili gli altri 7 blocchi da 8K superiori. Un metodo suggerito per l'espansione al di là del primo blocco di 8K è quello mostrato nella figura 5.2.

I tre bit di ordine superiore degli indirizzi (AB13, AB14, AB15) sono collegati ad un decodificatore. Le otto uscite del decodificatore agiscono nel senso di dividere lo spazio totale di memoria di 65K in otto blocchi da 8K ciascuno (8K0, 8K1, ecc.). Ora, l'uscita 8K0 può essere collegata come quarto ingresso (D) al decodificatore U4 sul KIM-1 provocando l'esatta selezione e deselection di questo blocco entro il completo spazio degli indirizzi. Le rimanenti sette uscite (da 8K1 ad 8K7) possono essere usate per selezionare o no i decodificatori aggiunti mostrati in figura 5.2.

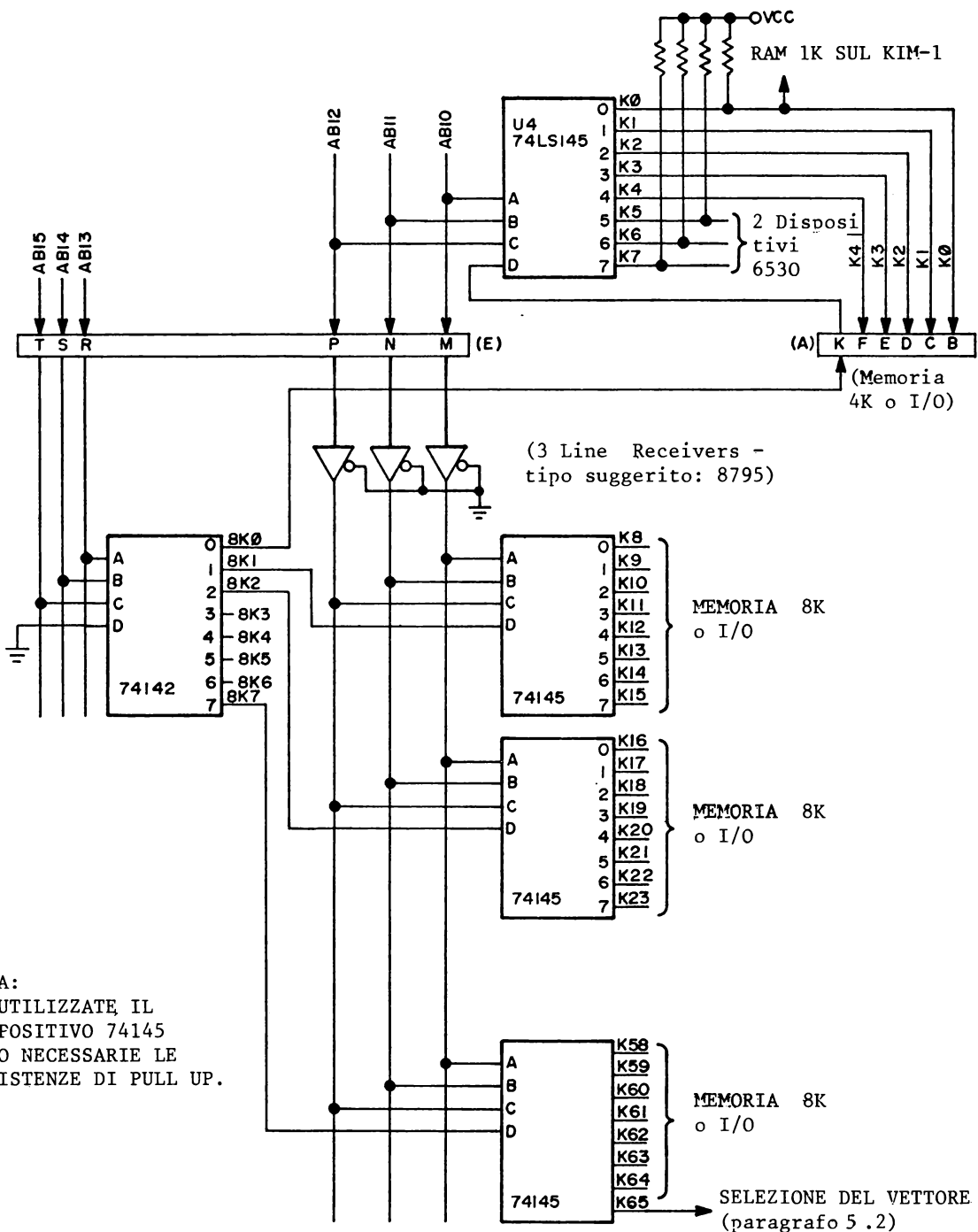
Basterà aggiungere tanti decodificatori (uno per ciascun blocco di memoria da 8K) quanti ne avete bisogno per l'espansione della memoria desiderata.

Quando decidete di aggiungere memoria al vostro sistema, dovrete usare qualche cautela. Infatti avrete notato l'inserzione dei ricevitori di linea per i segnali di AB10, AB11 ed AB12 (vedi fig.5.2). Questi dispositivi sono previsti a causa delle limitazioni di carico piazzate sulle linee degli indirizzi del 6502 (ciascuna di queste linee è capace di pilotare un solo carico standard TTL e 130 pF di capacità. Vedi appendice G.).



ESPANSIONE PER MEMORIA FINO A 4K

Figura 5.1



Prima di decidere come espandere il sistema, si raccomanda un accurato studio di tutte le limitazioni di carico dei segnali del KIM-1 in quanto quasi certamente avrete bisogno di circuiti addizionali d'amplificazione per ottenere una corretta operazione.

5.2 CONDUZIONE DEL VETTORE DI INTERRUPT

Abbiamo parlato varie volte nelle precedenti sezioni, dei dispositivi di interruzione del microprocessore 6502. Sugeriamo ora un'accurata lettura del capitolo 9 del manuale di programmazione riguardante le "considerazioni sul reset e l'interrupt".

In breve, sono possibili tre tipi di interruzione: Reset, NMI ed IRQ. Ciascuna avviene in risposta all'attivazione di uno dei tre piedini del circuito 6502 ($\overline{\text{RST}}$, $\overline{\text{NMI}}$, $\overline{\text{IRQ}}$). In risposta a questi ingressi, il circuito 6502 preleverà i dati memorizzati ad una specifica coppia di indirizzi e caricherà i dati prelevati nel contatore di programma. Gli indirizzi sono determinati dall'hardware (ossia corrispondono ad un circuito elettrico fisso) e non sono sotto il controllo del programmatore. Gli specifici indirizzi per ciascun tipo di interruzione sono:

FFFA, FFFB	Vettore	$\overline{\text{NMI}}$
FFFC, FFFD	Vettore	$\overline{\text{RST}}$
FFFE, FFFF	Vettore	$\overline{\text{IRQ}}$

Questi indirizzi definiscono le sei locazioni più alte della mappa di memoria da 65K.

Nel sistema KIM-1 tre bits di indirizzo (AB13, AB14, AB15) non sono decodificati per niente. Quindi, se il circuito 6502 genera un prelievo da FFFC ed FFFD in risposta ad un ingresso $\overline{\text{RST}}$, questi indirizzi saranno letti come 1FFC ed 1FFD ed il vettore di reset verrà prelevato da queste locazioni. Ora voi vedete che tutti i vettori di interruzione saranno prelevati dalle sei locazioni di testa del blocco inferiore di memoria da 8K che è l'unico blocco decodificato dal sistema KIM-1 non espanso.

E' tipico di ciascun sistema la memorizzazione dei vettori di interruzione in una ROM perchè possano essere immediatamente disponibili al

momento dell'alimentazione. D'altronde, è utile che per le interruzioni $\overline{\text{NMI}}$ ed $\overline{\text{IRQ}}$, il programmatore abbia la possibilità di definire come variabile il vettore esatto al quale queste interruzioni dirigeranno il sistema. Perciò le locazioni dei vettori $\overline{\text{NMI}}$ ed $\overline{\text{IRQ}}$ contengono una istruzione di salto indiretto che si riferisce ad una locazione di RAM entro la quale il programmatore memorizzerà lo specifico vettore per i due tipi di interruzione. Nel sistema KIM-1 le locazioni 17FA e 17FB contengono l'effettivo vettore $\overline{\text{NMI}}$ e le locazioni 17FE con 17FF contengono l'effettivo vettore $\overline{\text{IRQ}}$. Il vettore $\overline{\text{RST}}$ non è trattato in questo modo e, alla connessione dell'alimentazione, dirige sempre il sistema al primo passo della routine di inizializzazione.

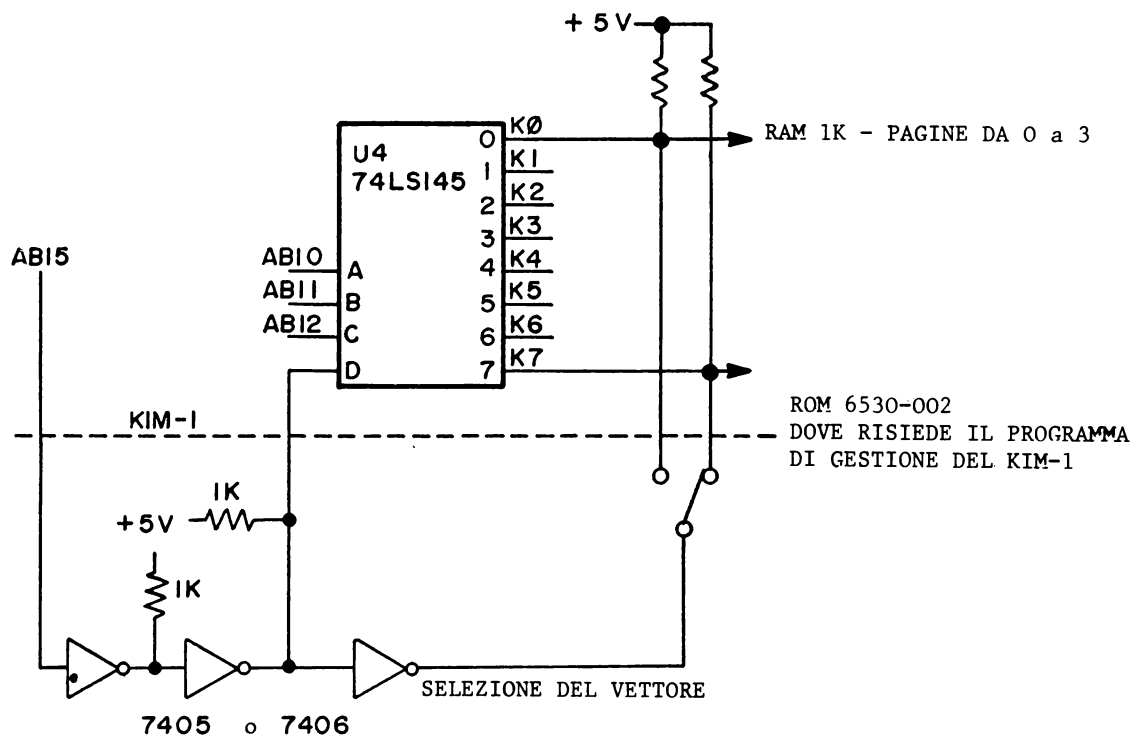
Ma cosa succederà se espandiamo la nostra memoria al di sopra del blocco inferiore da 8K compreso nel sistema KIM-1? Ricordate che ora dobbiamo usare AB13, AB14 ed AB15 per decodificare gli indirizzi addizionali della memoria. Facendo così le locazioni del vettore di interruzione non saranno più collocate nel blocco di memoria K7 dal momento che il decodificatore (U4) non è selezionato in risposta agli indirizzi generati dal 6502 nella ricerca dei vettori di interruzione (FFFA, per esempio). Avremo il medesimo problema anche in un sistema non espanso se vorremo usare un vettore $\overline{\text{RST}}$ ed una routine di inizializzazione diversa da quella fornita dal sistema KIM-1 e se il vettore $\overline{\text{RST}}$ dovesse essere collocato in un blocco da 1K più basso di K7 (per esempio K0).

La soluzione consiste nel generare logicamente uno speciale segnale per la scelta dell'interruzione. Con riferimento alla Fig. 5.2, si crea uno speciale segnale chiamato "scelta del vettore" (vector select) per definire il più alto blocco di memoria da 1K (K65). Il prelievo di ciascun vettore di interruzione manderà basso (0 logico) il "select". Supponendo che lo stato K65 non sia usato per selezionare una RAM, questo segnale può essere trattato con un OR cablato insieme ad uno qualsiasi degli altri segnali "K" (da K0 a K64) per definire esattamente quale blocco di 1K dovrà contenere i vettori di interruzione.

Tanto per fare un esempio, supponiamo che abbiate connesso la linea di selezione di vettore K65 alla linea K0. Quando avviene un RST, il microprocessore 6502 genererà un prelievo dalle locazioni FFFC ed FFFD. Questi indirizzi provocheranno una scelta di K65 che, a sua volta, accederà al campo di memoria K0 e causerà l'effettivo prelievo del vettore RST dalle locazioni 03FC e 03FD (Qualora abbiate scelto di connettere K65 a K7 il prelievo del vettore di reset av verrà dalle locazioni 1FFC ed 1FFD).

In questo modo i sei indirizzi più alti di un blocco di memoria da 1K possono essere usati per fornire i vettori di interruzione per il sistema. Se desiderato, può essere montato un interruttore atto a permettervi di selezionare differenti aree di memoria come locazioni di memoria dei vettori di interruzione (a tale proposito abbiamo scelto in fig. 5.2 i decodificatori tipo 75145 proprio per permettere di combinare in un OR cablato il K65 con un qualsiasi altro K. Questo è possibile in quanto il 75145 è provvisto di uscita a collettore aperto che permettono l'OR cablato di parecchi stadi usando un resistore di carico esterno).

Un arrangiamento ancora più semplice che usa il sistema "vector select" ossia scelta del vettore, è mostrato in fig. 5.3, che suppone che il sistema KIM-1 abbia disponibile solo il blocco da 8K inferiore. Il decodificatore degli indirizzi (U4) non è selezionato usando il segnale di AB15 che diviene "vero" ogni qualvolta la ricerca di un vettore di interruzione sia iniziato dal sistema. Il medesimo segnale (AB15) è invertito e collegato con configurazione OR per mezzo di un interruttore alle linee di selezione del chip di K0 o K7. Ora, in dipendenza della posizione dell'interruttore, i vettori di interruzione verranno prelevati dai sei indirizzi superiori di K0 oppure K7. K0 nel sistema KIM-1 è la RAM e K7 è la ROM del circuito 6530-002 (il programma operativo). In questo modo potrete avere due diversi gruppi di vettori di interruzione nel vostro sistema e potrete scegliere quale gruppo usare con un semplice commutatore.



SELEZIONE DI VETTORE

Figura 5.3

CAPITOLO 6

ASSISTENZA E GARANZIA

Se avete delle difficoltà con il vostro modulo KIM-1 e non riuscite a diagnosticare o correggere il guasto, potete restituire l'apparecchio alla Mos Technology Inc. per la riparazione.

6.1 SERVIZIO DI GARANZIA

Tutti i moduli microcalcolatori della serie KIM sono garantiti dalla Mos Technology contro difetti di costruzione e dei materiali, per un periodo di novanta giorni dalla data di consegna. Durante il periodo di garanzia la Mos Technology riparerà oppure, a sua scelta, sostituirà gratuitamente i componenti difettosi restituendo il modulo in porto franco a:

SKYLAB SRL
Via M. Gioia 66
20125 MILANO

Questa garanzia non si applica a moduli danneggiati da incidenti o da uso non appropriato o come risultato di riparazioni e modifiche fatte da personale non autorizzato secondo le norme di servizio dette sopra. Nessun'altra garanzia è espressa od implicita. La Mos Technology non è responsabile per danni a terzi.

6.2 SERVIZIO FUORI GARANZIA

Al di là del periodo di garanzia di 90 giorni, i moduli KIM vengono riparati ad un ragionevole costo di servizio. Tutti i lavori eseguiti fuori garanzia dalla Mos Technology sono garantiti per un ulteriore periodo di 90 giorni dalla spedizione del modulo riparato.

6.3 RISERVA DI CAMBIAMENTI

Tutti i moduli KIM-1 sono venduti sulla base di specifiche descrittive in vigore al momento della vendita. La Mos Technology non avrà alcun obbligo di modificare od aggiornare prodotti già venduti. La Mos Technology si riserva i diritti di apportare cambiamenti o miglioramenti periodici a ciascun modulo della serie KIM.

6.4 ISTRUZIONI PER IL TRASPORTO

E' responsabilità del cliente la restituzione del modulo della serie KIM in porto franco per fruire dei servizi elencati.

Per il servizio in garanzia il modulo KIM verrà restituito al cliente in porto franco con il vettore più rapido ed economico.

Per i servizi fuori garanzia il cliente pagherà il trasporto in ambedue le direzioni.

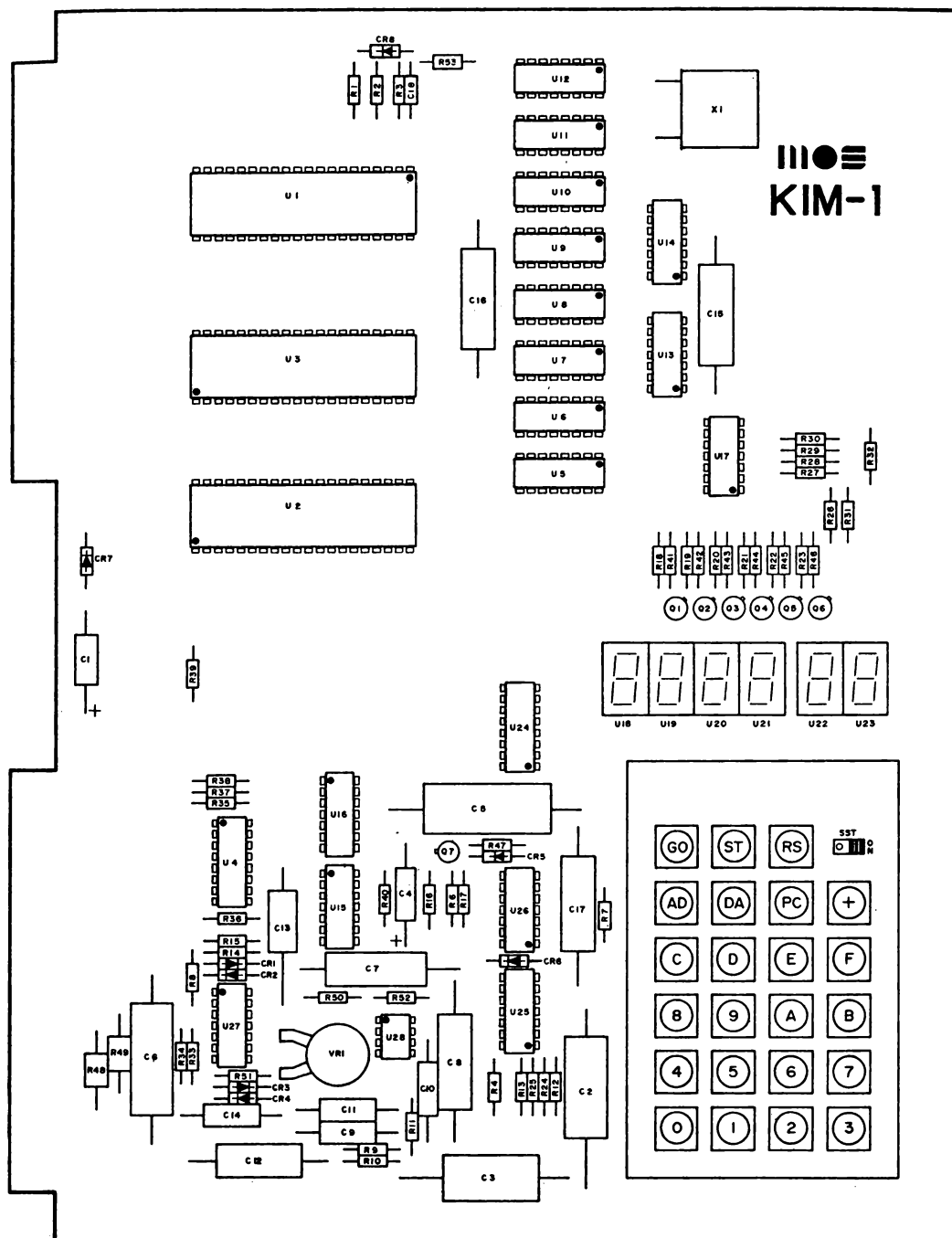
Qualora doveste restituire il modulo KIM-1 per riparazioni, assicuratevi che sia protetto da un sicuro imballo.

APPENDICE A

ITEM	PARTE	QUANT.	DESCRIZIONE
1.	U1	1	6502 Microprocessor
2.	U2	1	6530 ROM RAM I/O,TIMER, Chip-02
3.	U3	1	6530 ROM RAM I/O,TIMER, Chip-03
4.	U5 through U12	8	6102 RAM 500ns Acc,0ns
5.	U18 through U23	6	7 SEG .3" Red Display
6.	U25	1	556 Timer IC
7.	U27	1	565 Phase Lock Loop
8.	U28	1	311 Comparator
9.	U24	1	74145 BCD Decoder IC
10.	U13 & U14	2	74125 TRI STATE Buffer
11.	U15	1	7400 Quad Nand IC
12.	U16	1	7404 Hex Inverter IC
13.	U17	1	7406 Hex Inv. O/C IC
14.	U26	1	7438 Quad Nand O/C IC
15.	CR1,2,3,4,&8	5	20 MA. 50v Diode - IN914
16.	CR5, CR6	2	1A 50v Diode - IN4001
17.	CR7	1	6.2v $\frac{1}{2}$ W Z. Diode - IN4735
18.	Q7	1	NPN Transistor B>20, VCE>12 - 2N5371
19.	Q1 through Q6	6	PNP Transistor B>20, VCE>6 - 2N5375
20.	R24 & R25	2	47K Ω $\pm 10\%$ $\frac{1}{4}$ W Resistor
21.	R1,2,3,4, & 6	5	3.3K Ω $\pm 10\%$ $\frac{1}{4}$ W Resistor
22.	R34 & R50	2	2.2K Ω $\pm 10\%$ $\frac{1}{4}$ W Resistor
23.	R12-R17, R41-R46	12	1.0K Ω $\pm 10\%$ $\frac{1}{4}$ W Resistor
24.	R35 through R40	6	560 Ω $\pm 10\%$ $\frac{1}{4}$ W Resistor
25.	R18-R23, R47	7	220 Ω $\pm 10\%$ $\frac{1}{4}$ W Resistor
26.	R33	1	47 Ω $\pm 10\%$ $\frac{1}{4}$ W Resistor
27.	R52	1	5 Meg. $\pm 10\%$ $\frac{1}{4}$ W Resistor
28.	R51	1	30K Ω $\pm 5\%$ $\frac{1}{4}$ W Resistor
29.	R7,R8,R9,R10&R11	5	10K Ω $\pm 5\%$ $\frac{1}{4}$ W Resistor
30.	R48, R49	2	150 Ω $\pm 5\%$ $\frac{1}{2}$ W
31.	R26 through R32	7	82 Ω $\pm 5\%$ $\frac{1}{4}$ W
32.	VR1	1	5K Ω Potentiometer
33.	C2, C3, C6	3	.22 $\pm 10\%$ uf.>12 vv. cap
34.	C1, C4	2	1uf+80-10%>12WV Cap
35.	C5	1	.33 uf $\pm 10\%$ >12WV Cap
36.	C7,C8,C15,C16,C17	5	.1uf+80-10%>12WV Cap
37.	C9, C10, C11	3	.0068uf $\pm 10\%$ >12WV
38.	C12	1	.047uf $\pm 10\%$ >12WV
39.	C13	1	.022uf $\pm 10\%$ >12WV
40.	C14	1	.001uf $\pm 10\%$ >12WV
41.		1	44 Pin Edge Conn.
42.	X1	1	1 MHz XTAL
43.		1	PCB.
44.		1	24 Key KBD
45.		6	Rubber Pads
46.		1	Shipping Bag (Static Free)
47.		1	Shipping Box
48.		1	Hardware Manual
49.		1	Software Manual
50.		1	KIM Manual
51.			
52.		1	Wall Chart
53.			
54.		1	Program Card
55.	C18	1	10pf CAP
56.	R53	1	330K $\frac{1}{4}$ W Resistor
57.	U4	1	74LS145 BCD Decoder 1C

APPENDICE B

CONFIGURAZIONE CIRCUITALE DEL KIM - 1



APPENDICE C

IN CASO DI DIFFICOLTA'

SINTOMO: Il display non si illumina

1) Controllate l'alimentatore a +5V, usando un tester per provare la tensione di +5V tra i piedini E-21 ed E-22. Provate anche la tensione di +5V tra i piedini A-A ed A-1.

L'alimentatore dovrà essere stabilizzato a +5V \pm 5%.

2) Controllate il collegamento dell'opzione KB-TTY (fig.1.4).

Il piedino A-21 non deve essere collegato al piedino A-V.

3) Assicuratevi che il decodificatore sia abilitato.

Confrontate la fig. 1.2 ed assicuratevi che il piedino A-K sia connesso a massa.

4) Premete il tasto di reset e tutti gli altri tasti, assicurandovi che nessuno sia bloccato.

5) Provate con un tester tra i piedini E-21 (+5V) ed il piedino E7 (Reset). Premete alternativamente e rilasciate il tasto di reset, controllando se la tensione cambia da più di 4V a meno di 1 V.

6) Provate il piedino E-V (\emptyset_2) con un oscilloscopio e verificate la presenza dell'onda ad 1 MHz.

SINTOMO: Non registra sul nastro audio
Non carica dal nastro audio

1) Misurate l'alimentatore a +12V. Provate con un tester la tensione di +12V tra i piedini A-N (+12V) e A-1 (massa). L'alimentatore

deve essere stabilizzato a $+12V \pm 5\%$. (vedi fig. 1.2)

2) Verificate il controllo di volume del registratore (che deve essere posizionato a mezza corsa).

3) Assicuratevi di avere eseguito gli appropriati collegamenti (vedi fig. 1.3).

4) Provate il circuito d'interfaccia del nastro scollegando il registratore e cortocircuitando il piedino A-P (uscita audio alta) con il piedino A-L (ingresso audio). Posizionate il KIM-1 come se dovesse rilevare una sezione di memoria. Con un oscilloscopio osservate i dati al piedino E-X (PROVA PLL). (Vedi l'appendice E per il corretto formato dei dati e la procedura di calibrazione).

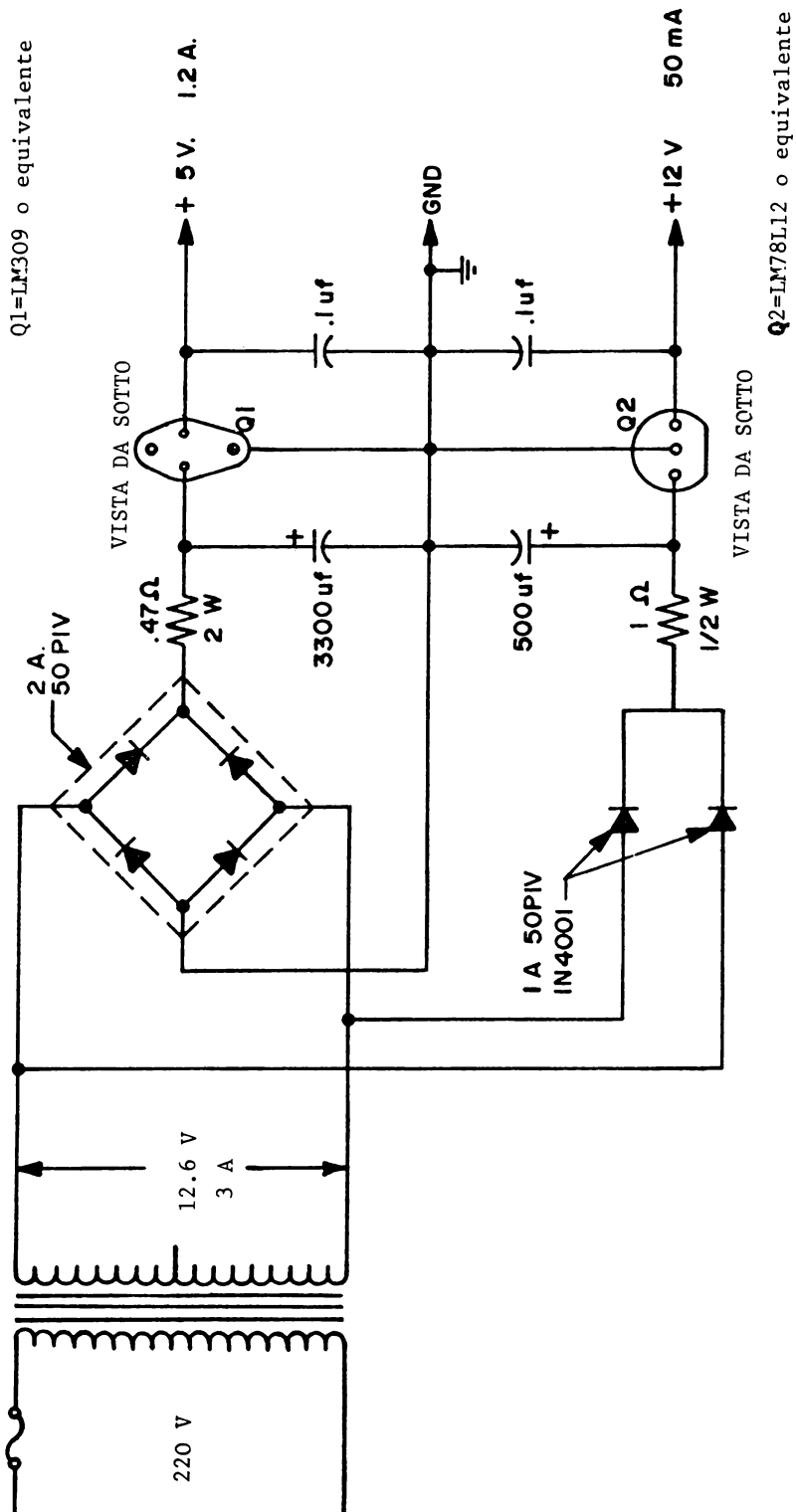
5) Registrate la voce su una sezione di nastro e risentitela per assicurarvi che il registratore funzioni. Collegare un altro registratore al sistema o provate un'altra cassetta.

SINTOMO: Difficoltà nell'interfaccia TTY

1) Assicuratevi che il piedino A-21 sia connesso al piedino A-V (fig. 1.4) per permettere l'operazione in TTY.

2) Confrontate le connessioni di fig. 1.4 con lo schema dell'interfaccia della vostra TTY sul relativo manuale.

3) Premete il tasto di reset sulla tastiera del KIM-1 seguito da un carattere RUBOUT sulla TTY.



APPENDICE E

FORMATO DEI DATI SUL NASTRO AUDIO

I dati sono registrati sull'audiocassetta con uno specifico formato progettato per permettere un recupero esente da errori.

Nell'improbabile caso che avvenga un errore di trascrizione, sono previsti diversi rilevatori di errore per segnalare tale situazione.

I dati sono trasmessi al registratore in forma ASCII seriale (sette bits di dati più un bit di parità). I dati rilevati dalla memoria sono convertiti in questa forma separando ogni byte in due semi-bytes. I semi-bytes sono quindi trasformati nel loro equivalente ASCII.

Ogni registrazione trasmessa comincia con un segnale di sincronizzazione di cento caratteri SYN (ASCII 16) seguiti da un carattere * (ASCII 2A). Durante la trasmissione, questa sequenza permette al microprocessore di rilevare l'inizio di una registrazione valida e la sincronizza all'afflusso dei dati seriali. Dopo il segnale *, viene il numero di identificazione (ID), l'indirizzo di inizio basso (SAL) e quello alto (SAH). I dati compresi tra i limiti di inizio (SAL, SAH) e di fine (EAL, EAH) sono trasmessi successivamente, seguiti da un carattere "/" (ASCII 2F) per indicare la fine della porzione dei dati sulla registrazione. Dopo il "/" sono trasmessi due bytes CHECKSUM per comparazione con il numero checksum calcolato durante il rilievo per assicurare una volta di più che ha avuto luogo la appropriata rilevazione dei dati. Due caratteri EOT (ASCII 04) contrassegnano la fine della trasmissione della registrazione.

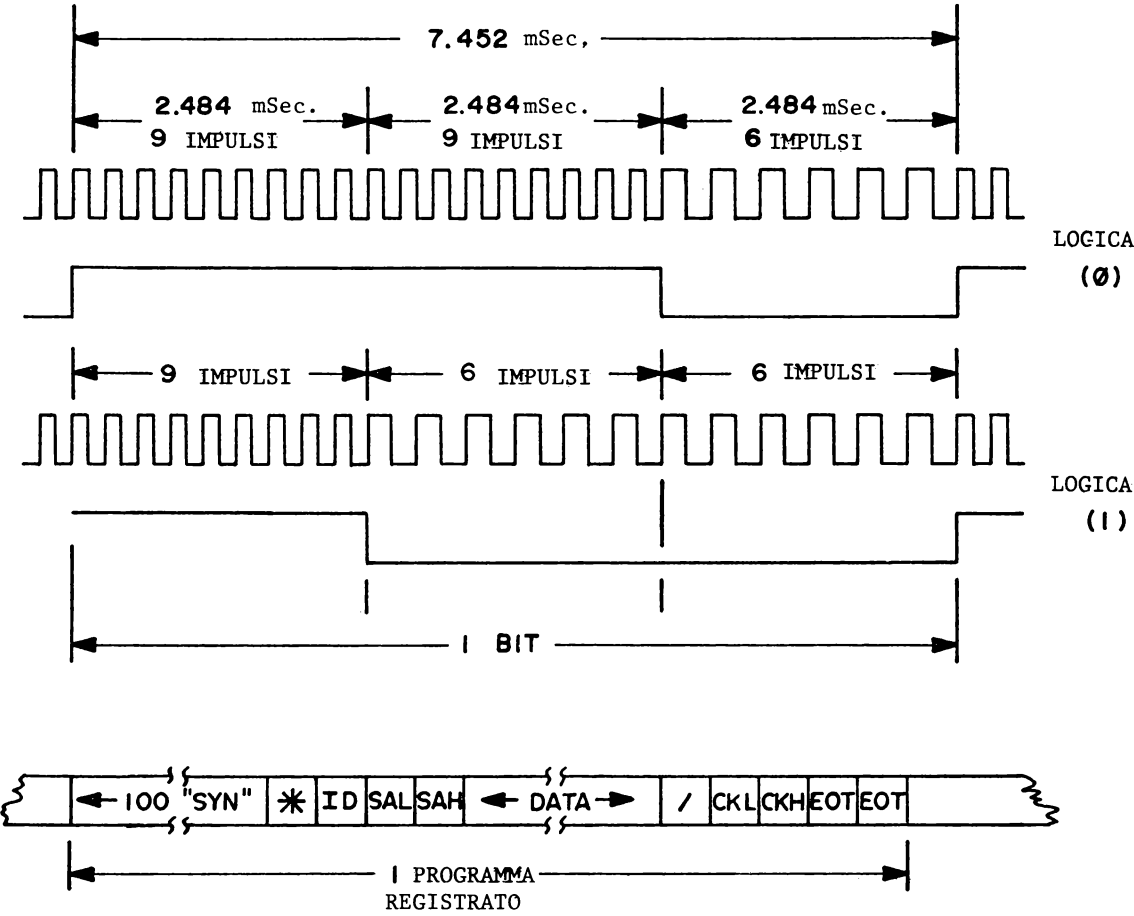
Ciascun bit trasmesso comincia con una nota a 3700 Hz e finisce con una nota a 2400 Hz. Gli 1 presentano una transizione dalla frequenza alta alla bassa ad un terzo del periodo di bit. Gli 0 hanno la transizione a due terzi del periodo di bit. Durante l'ascolto del nastro, il circuito PLL 565 si aggancia e rileva queste due frequenze producendo (at-

traverso il comparatore 311) un impulso logico 1 di un terzo del periodo di bit per l' "1" logico. Allo stesso modo viene prodotto un impulso di due terzi di bit per lo "0" logico. Il microcalcolatore usa un algoritmo controllato da software per convertire questo segnale in parole di dati da 8 bit.

Il metodo della registrazione dei dati con il sistema ad aggancio di fase (PLL) e variazione di frequenza (FSK) è relativamente insensibile a variazioni dell'ampiezza e della fase. La frequenza persistente dell'anello di aggancio di fase è stata aggiustata in fabbrica regolando la a mezza via tra le due frequenze dei dati (è detta frequenza centrale).

Questa regolazione è ottenuta collegando il piedino A-P (uscita audio alta) al piedino A-L (ingresso audio). Un programma che parte dall'indirizzo 1A6B esadecimale, provvede al riferimento per la frequenza centrale che permette all'anello di essere regolato con il potenziometro VR1. Il piedino E-X (PLL-TEST) è controllato con un volmetro mentre il potenziometro è girato finché la lettura sul voltmetro è al punto di transizione tra il livello logico 1 (+5V) e 0 (massa).

QUESTA REGOLAZIONE E' STATA ESEGUITA IN FABBRICA E DEVE ESSERE RIFATTA SOLO IN SEGUITO A SOSTITUZIONE DI COMPONENTI.



FORMATO PER LA REGISTRAZIONE
Figura E - 1

APPENDICE F

FORMATO DEI DATI PER IL NASTRO DI CARTA

Le routines di "carica da nastro" (LOAD) e "perfora su nastro" (DUMP), memorizzano e leggono dati in uno specifico formato progettato per permettere una rilevazione scevra d'errori. Ciascun byte di dati da caricare è convertito in due semibytes. I due semibytes (il cui possibile valore va da Ø ad F esadecimale) sono tradotti nei loro equivalenti ASCII ed in questa forma trasferiti sul nastro di carta.

Ciascuna registrazione emessa comincia con un carattere ";" (ASCII 3B) per segnare l'inizio di una stringa (RECORD) di dati validi. Il successivo byte trasmesso (18 esadec.) o (24 decimale) è il numero dei bytes di dati contenuti nella stringa. L'indirizzo di partenza alto della registrazione (1 byte, 2 caratteri), l'indirizzo di partenza basso (1 byte, 2 caratteri) ed i dati (24 bytes, 48 caratteri) seguono nell'ordine. Ciascuna stringa è terminata da un checksum (2 bytes, 4 caratteri), un segnale di ritorno carrello (ASCII OD), un segnale di linee feed (ASCII ØA) e da sei caratteri NULL (ASCII ØØ).

L'ultima stringa trasmessa ha zero bytes di dati (indicato da ;ØØ). L'indirizzo di partenza è rimpiazzato da un numero esadecimale di 4 cifre, che rappresenta il numero totale di registrazioni di dati contenuto nella trasmissione, seguito dalla registrazione delle solite cifre di checksum. Un carattere XOFF chiude la trasmissione.

;180000FFEEDDCCBBAA0099887766554433221122334455667788990AFC
;0000010001

Durante un "LOAD" tutti i dati che pervengono sono ignorati fino alla ricezione di un carattere ";". La ricezione di dati non ASCII od un disaccordo tra un checksum calcolato e quello letto dal nastro causerà una condizione di errore che sarà rilevata dal sistema KIM. Il checksum è calcolato sommando tutti i dati della registrazione eccetto il carattere ";".

Il formato per il nastro di carta descritto è compatibile con tutti gli altri programmi di supporto software della Most Technology Inc.

APPENDICE G

CARATTERISTICHE DEL 6502

Clocks ($\emptyset 1$, $\emptyset 2$)

L'MCS 6502 è dotato di un generatore di clock interno, la cui frequenza è controllata da un cristallo di quarzo.

Linee degli indirizzi ($A_0 - A_{15}$)

Queste uscite sono TTL compatibili, capaci di pilotare un carico TTL e 130 pF.

Linee dei dati ($D_0 - D_7$)

Per le linee dei dati si usano otto piedini. Si tratta di linee bidirezionali che trasferiscono i dati dal microprocessore alle periferiche e viceversa. Le uscite sono dei buffer a tre stati capaci di pilotare un carico TTL standard e 130 pF.

Ready (RDY)

Questo segnale di ingresso permette all'utente di passare a passo singolo tutti i cicli del microprocessore ad eccezione di quelli di scrittura. Una transizione negativa allo stato basso durante od in coincidenza con la fase 1 ($\emptyset 1$) fermerà il microprocessore con le linee di indirizzo d'uscita che rispecchiano l'indirizzo rilevato al momento. Questa condizione permarrà durante una susseguente fase 2 ($\emptyset 2$) nella quale il segnale Ready è alto. Questa particolarità permette al microprocessore d'interfacciarsi con PROMS a bassa velocità come pure ad accessi di memoria diretti (DMA) rapidi (massimo 2 cicli). Se ready è basso durante un ciclo di scrittura, esso è ignorato sino alla successiva operazione di lettura.

Richiesta di interrupt ($\overline{\text{IRQ}}$)

Questo ingresso a livello TTL richiede al microprocessore una sequenza d'interruzione. Il microprocessore completerà la istruzione in corso e quindi prenderà cognizione della richiesta. In quel momento verrà esaminata la configurazione dei bit d'interruzione nel Registro del Codice di Stato. Se il flag dell'interruzione non è attivato, il microprocessore inizierà una sequenza di interrupt. Il Contatore del Programma ed il Registro dello Stato del Processore sono conservati nello stack. Il microprocessore attiverà quindi il flag dell'interrupt ad 1 (livello alto) in modo che non possano avvenire altre interruzioni. Alla fine di questo ciclo il byte basso del Contatore di Programma sarà prelevato dall'indirizzo FFFE e il byte alto dall'indirizzo FFFF, trasferendo quindi il controllo del programma al vettore di memoria localizzato in questi indirizzi. Il segnale RDY deve essere allo stato alto per ogni interrupt che debba essere riconosciuto. Sarà necessario l'utilizzo di una resistenza esterna da 3K Ohm per avere una corretta operazione dell'OR cablato.

Interrupt non mascherabile ($\overline{\text{NMI}}$)

Una commutazione negativa del segnale su questo ingresso richiede che venga generata entro il microprocessore una sequenza di interruzione non mascherabile.

$\overline{\text{NMI}}$ è un'interruzione incondizionata. Dopo il completamento dell'istruzione in corso, la sequenza di operazioni definita per l'IRQ viene eseguita senza riguardo allo stato del flag dell'interrupt. L'indirizzo del vettore caricato nel contatore di programma, basso ed alto, sono rispettivamente le locazioni FFFA ed FFFB. Le istruzioni caricate a queste locazioni costringono il microprocessore ad eseguire un branch ad una routine di interruzione non mascherabile. Anche $\overline{\text{NMI}}$ richiede una resistenza da 3K Ohm esterna a Vcc, per una regolare operazione dell'OR cablato.

Gli ingressi $\overline{\text{IRQ}}$ ed $\overline{\text{NMI}}$ sono linee di interruzione in hardware, riconosciute durante $\phi 2$ (fase 2) che cominceranno l'appropriata routine di interrupt nella $\phi 1$ (fase 1) dopo il completamento dell'istruzione in corso.

Attiva il flag di overflow (S.O.)

Questo segnale di ingresso a livello TTL permette il controllo esterno del bit di overflow nel registro del codice di stato.

SYNC

Questa linea di uscita è prevista per identificare i cicli nei quali il microprocessore sta attuando un rilievo di un Op Code. La linea SYNC va alta durante il Ø1 del rilievo di un Op Code e rimane alta per il resto del ciclo. Se la linea RDY è mandata bassa durante l'impulso di clock Ø1 nel quale SYNC è andato alto, il processore si fermerà in quello stato e vi rimarrà sinchè la linea RDY andrà alta. In questo modo il segnale SYNC può essere usato per controllare RDY e ottenere l'esecuzione di una singola istruzione.

RESET

Questo ingresso è usato per resettare od avviare il microprocessore da una condizione di mancanza dell'alimentazione. Durante il tempo nel quale questa linea è mantenuta bassa, è impedita la scrittura verso o dal microprocessore. Se viene rilevata all'ingresso una commutazione positiva, il microprocessore comincerà immediatamente la sequenza di reset.

Dopo un tempo di inizializzazione del sistema di sei cicli di clock, il flag dell'interrupt sarà attivato ed il microprocessore caricherà il contatore di programma dalle locazioni del vettore di memoria FFFC ed FFFD. Questa è la locazione di partenza per il controllo del programma.

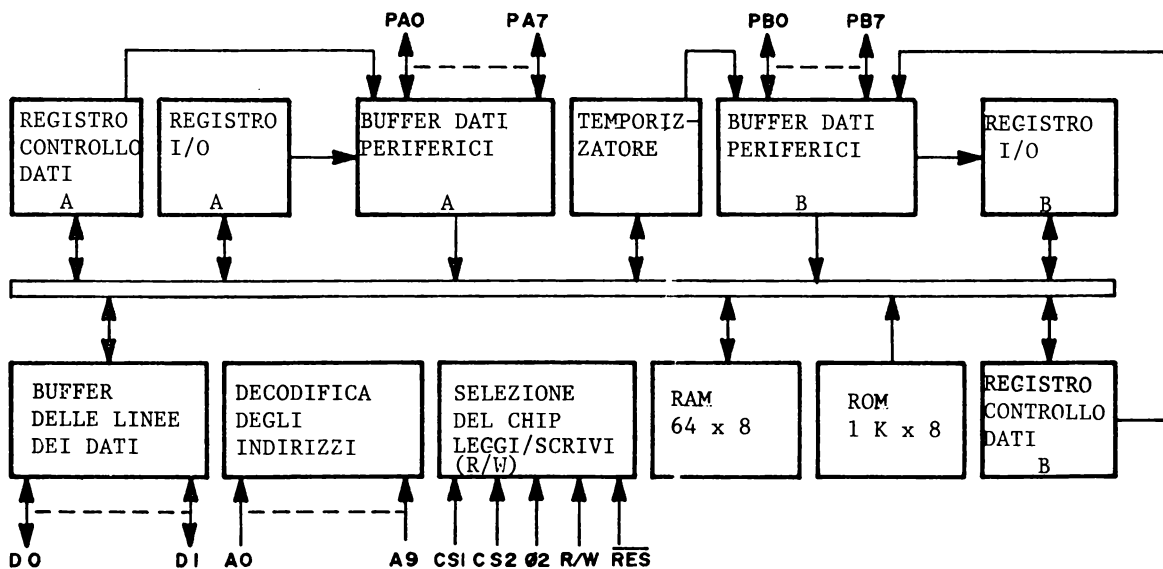
Dopo che Vcc ha raggiunto i 4,75 V nella sequenza di alimentazione, il reset deve essere mantenuto basso per almeno due cicli.

Se il segnale di reset va alto dopo questi due cicli di clock, il microprocessore procederà con la normale routine di reset spiegata in precedenza.

APPENDICE H

CARATTERISTICHE DELL'MCS 6530

L'MCS 6530 è progettato per operare in congiunzione con i microprocessori della famiglia MCS 6500. Comprende una ROM programmabile a maschera da 1024 x 8, una RAM statica da 64 x 8, due port dati bidirezionali da 8 bit controllati dal software che permettono l'interfacciamento diretto tra il microprocessore ed i dispositivi periferici, ed un timer degli intervalli, programmabile da software, con interruzione, capace di temporizzare in vari intervalli da 1 a 262.144 periodi di



MCS 6530 SCHEMA A BLOCCHI

Figura H - 1

Reset ($\overline{\text{RES}}$)

Durante l'inizializzazione del sistema uno "0" logico sull'ingresso $\overline{\text{RES}}$ causerà un azzeramento di tutti e quattro i registri I/O, facendo sì che tutti i bus I/O agiscano come ingressi evitando in volontarie scritture verso l'esterno. I buffers dei bus dei dati sono messi nello stato OFF durante il reset. La possibilità d'interruzione è disabilitata con il segnale $\overline{\text{RES}}$. Quando è richiesto il reset, il segnale $\overline{\text{RES}}$ deve essere mantenuto basso per almeno un periodo di clock.

Clock d'ingresso

Il clock d'ingresso è un sistema a due fasi che può essere sia un clock a basso livello ($V_{\text{IL}} < 0.4$, $V_{\text{IH}} > 2.4$) oppure un clock ad alto livello ($V_{\text{IL}} < 0.2$, $V_{\text{IH}} = V_{\text{CC}} - 0.2$).

Lettura/scrittura (R/W)

Il segnale R/W è fornito dal microprocessore ed è usato per controllare il trasferimento dei dati verso e dal microprocessore e l'MCS 6530. Un livello alto sul piedino R/W permette al microprocessore di leggere (con un appropriato indirizzamento) i dati forniti dall'MCS 6530. Un livello basso sul piedino R/W permette una scrittura (con appropriato indirizzamento) nell'MCS 6530.

Richiesta di interruzione ($\overline{\text{IRQ}}$)

Il piedino $\overline{\text{IRQ}}$ è un piedino di interruzione che può essere attivato dal temporizzatore. Questo stesso piedino, qualora non sia usato come interruzione, può essere usato come piedino I/O per le periferiche (PB7). Se usato come interruzione il piedino può essere determinato come ingresso dal registro della direzione dei dati. Il piedino sarà normalmente a livello 1 mentre un livello 0 indicherà un'interruzione dall'MCS 6530.

Bus dei dati ($D_0 - D_7$)

L'MCS 6530 ha 8 piedini bidirezionali per i dati ($D_0 - D_7$).

Questi piedini si connettono alle linee dei dati del sistema per permettere il trasferimento dei dati verso e dal microprocessore. I buffers di uscita rimangono nello stato di esclusione eccetto quando avviene un'operazione di lettura.

Port dei dati delle periferiche

Sia l'MCS 6530-002 che l'MCS 6530-003, hanno 15 piedini disponibili per le operazioni di I/O delle periferiche. Ciascun piedino è individualmente programmabile dal software per funzionare sia da ingresso che da uscita. I 15 piedini sono divisi in due port da 8 bit (PA0 - PA7 e PB0 - PB7). PB6 è stato usato come selezionatore del chip e non è disponibile per l'utente. I piedini sono destinati ad essere ingressi scrivendo "0" nel corrispondente bit del registro direzione dati. Un "1" scritto in questo registro causerà al corrispondente piedino di diventare un'uscita. Quando sono condizionati come ingressi, i buffers di uscita delle periferiche sono nello stato "1" ed un dispositivo di pull-up agisce sulle linee dei dati delle periferiche con un carico minore di un TTL. Nella operazione di lettura il microprocessore legge i piedini delle periferiche. Quando l'MCS 6530 riceve informazioni dalle periferiche, il microprocessore riceverà i dati immagazzinati nel registro dei dati. Il microprocessore leggerà informazioni esatte se le linee periferiche saranno ad un livello maggiore di 2,0 volt per un 1 e minore di 0,8 volt per uno 0, dal momento che i piedini periferici sono TTL compatibili. I piedini PA0 e PB0 sono pure capaci di erogare 3mA ad 1,5 V, quindi essi sono capaci di pilotare un Darlington. Il piedino PB7 non ha pull-up interno (in modo da permettere un OR cablato con altri dispositivi).

Linee di indirizzamento (A0 - A9)

Oltre ai 10 piedini di indirizzamento, c'è anche un piedino di selezione della ROM. I suddetti piedini (A0 - A9) ed il ROM SELECT sono sempre usati come piedini di indirizzamento.

Due piedini addizionali programmabili a maschera, possono essere usati sia individualmente che insieme come CHIP SELECTS: si tratta

dei piedini PB5 e PB6. Se usati come piedini per dati dalle periferiche, non possono essere usati come CHIP SELECTS. Il PB5 è stato usato come piedino per i dati mentre PB6 è stato usato come chip select e non è disponibile all'utente.

Uno schema a blocchi dell'architettura interna è mostrato in figura H-1. L'MCS 6530 è diviso in quattro sezioni base, RAM, ROM, I/O e temporizzatore.

La RAM e la ROM interfacciano direttamente con il microprocessore attraverso il bus dei dati del sistema e le linee di indirizzamento. La sezione I/O consiste in due port. Ciascun port contiene un registro della direzione dei dati (DDR) ed un registro I/O.

ROM da 1K byte (8K bits)

La ROM da 8K è in configurazione 1024 x 8. Le linee di indirizzamento A₀ - A₉ come pure RS₀ sono destinate ad indirizzare l'intera ROM. Con l'aggiunta di CS₁ e CS₂, possono essere indirizzati sette MCS 6530, fornendo un totale di 7168 x 8 bits di ROM contigue.

RAM da 64 bytes (512 Bits)

Una RAM statica da 64 x 8 è contenuta nell'MCS 6530. Essa è indirizzata da A₀ - A₅ (selezione del byte), da RS₀, A₆, A₇, A₈, A₉ e CS₁.

Registri interni alle periferiche

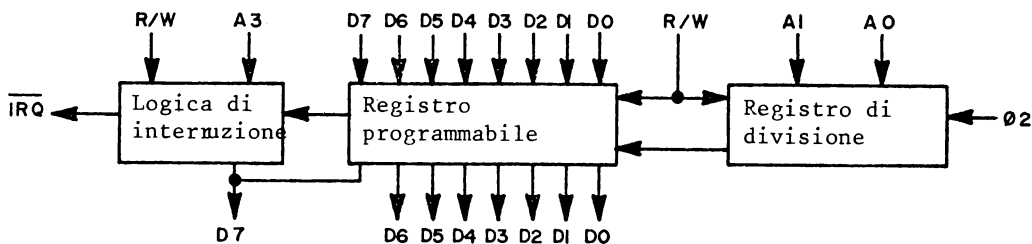
Ci sono quattro registri interni, due dei quali per la direzione dei dati e due per l'I/O dei dati delle periferiche. I due registri direzione dati (lato A e lato B) controllano la direzione dei dati in entrata od in uscita attraverso i piedini delle periferiche.

Un "1" scritto nel registro della direzione dati configura il corrispondente piedino del buffer delle periferiche a funzionare da uscita. Quindi, qualsiasi cosa venga scritta nel registro I/O apparirà sul corrispondente piedino periferico. Uno "0" scritto entro la DDR proibisce al buffer di uscita di trasmettere dati da/o verso il registro I/O. Per esempio, un "1" caricato nel registro direzione dati A, posizione 3, predispone il piedino periferico PA3 come uscita.

Se è stato caricato uno "0", PA3 sarà configurato come entrata e rimarrà allo stato alto. I due registri per l'I/O dei dati sono usati per memorizzare temporaneamente i dati provenienti dal bus dei dati durante un'operazione di scrittura sinchè i dispositivi periferici non possono leggere i dati forniti dal microprocessore. Durante l'operazione di lettura il microprocessore non legge i registri I/O, ma in effetti legge i piedini dei dati periferici. Per i piedini dei dati periferici programmati come uscite, il microprocessore leggerà i bits di dati del registro I/O. Il solo modo nel quale i dati nel registro I/O possono essere cambiati è un'operazione di scrittura da parte del microprocessore. Il registro I/O non è influenzato da una lettura dei dati sui piedini periferici.

Temporizzatore (Interval timer)

La sezione timer dell'MCS 6530 contiene tre parti base: un registro per la divisione preliminare, un registro programmabile ad 8 Bit ed una logica d'interruzione, illustrati nella fig. H-2.



ELEMENTI BASE DEL TEMPORIZZATORE - Figura H 2

Il temporizzatore degli intervalli può essere programmato per contare fino a 256 intervalli di tempo. Ciascun intervallo può essere sia 1T che 8T, 64T oppure 1024T, dove T è il periodo del clock del sistema. Quando viene raggiunto il pieno conteggio, un flag di interruzione è attivato al livello logico "1" ed il clock interno comincerà a contare indietro fino ad un massimo di -255T. Dopo che il flag di interruzione è attivato, una lettura del temporizzatore ci dirà quanto tempo è passato dall'attivazione del flag, fino ad un massimo di 255T.

Il sistema del bus dei dati ad 8 bit è usato per trasferire i dati da e verso il temporizzatore degli intervalli. Se bisogna contare un totale di 52 intervalli di tempo, è necessario inserire nel bus dei dati la configurazione 00110100 (34 HEX) che viene scritta nel registro del temporizzatore.

Nota: HEX = esadecimale.

Mentre questo dato viene scritto nel temporizzatore degli intervalli, gli intervalli di conteggio per 1, 8, 64, 1024T, sono decodificati dalle linee di indirizzo A0 ed A1. Durante un'operazione di lettura o scrittura la linea di indirizzo A3 controlla la capacità di interruzione di PB7. In altri termini A3=1 abilita IRQ su PB7, A3=0 disabilita IRQ su PB7. Se PB7 deve essere usato come flag di interruzione con il temporizzatore degli intervalli, dovrà essere programmato come ingresso. Se PB7 è abilitato da A3 ed avviene un'interruzione, PB7 andrà a 0.

Se il temporizzatore viene letto prima che sia attivato il flag di interruzione, allora sarà letto il numero dei restanti intervalli di tempo, ossia 51, 50, 49, ecc.*

Se il temporizzatore ha contato indietro fino a 0000 0000 (00 HEX), al successivo tempo di conteggio avverrà un'interruzione ed il contatore leggerà 1111 1111 (FF HEX). Dopo l'interruzione il registro del temporizzatore decresce con un tasso di divisione per "1" del clock del sistema. Se, per esempio, dopo l'interruzione viene letto il temporizzatore e vi si vede un valore 1110 0100 (E4 HEX), il tempo

trascorso dall'interruzione è 28T. Il valore letto è in complemento a 2:

```

Valore letto = 1110 0100
Complemento = 0001 1011
      +              1
=====
Complemento a 2 0001 1100 = 28

```

Quindi, per arrivare al tempo totale trascorso, c'è solo da fare il complemento a 2 del tempo originale scritto nel temporizzatore. Supponiamo che il tempo scritto sia 0011 0100 (=52).

Con una divisione per 8 il tempo totale che manca all'interruzione è $(54 \times 8) + 1 = 417T$.

Il tempo totale trascorso sarà $416T + 28T = 444T$ (supponendo che il valore letto dopo l'interruzione sia 1110 0100).

Dopo l'interruzione, il relativo flag è resettato, sia che il temporizzatore sia stato scritto oppure letto. D'altronde la lettura del temporizzatore allo stesso momento in cui avviene l'interruzione, non resetta il flag di interruzione.

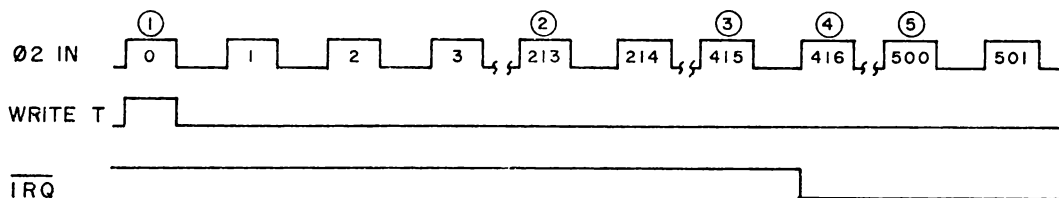


Figura H 3

1. Il dato scritto nel temporizzatore degli intervalli è $0011\ 0100 = 52_{10}$
2. Il dato nel temporizzatore degli intervalli è $0001\ 1001 = 25_{10}$
 $52 - \frac{213}{8} - 1 = 52 - 26 - 1 = 25$

3. Il dato nel temporizzatore degli intervalli è $0000\ 0000 = 0_{10}$
 $52 - \frac{415}{8} - 1 = 52 - 51 - 1 = 0$
4. L'interruzione è avvenuta all'impulso di $\emptyset_2 \neq 416$
 Il dato nel temporizzatore degli intervalli è $= 1111\ 1111$
5. Il dato nel temporizzatore degli intervalli è $1010\ 1100$
 Il complemento a 2 è $0101\ 0100 = 84_{10}$
 $84 + (52 \times 8) = 500_{10}$

Quando si legge il temporizzatore dopo un'interruzione, A3 deve essere basso in modo da disabilitare il piedino $\overline{\text{IRQ}}$, al fine di evitare future interruzioni finchè non avvenga un'altra operazione di scrittura del timer.

Interval timer del KIM-1

Nella memoria MCS 6530-003 del KIM-1 è presente un temporizzatore disponibile per l'uso da parte dell'utilizzatore.

Tale temporizzatore vi consente di prestabilire un conteggio fino a 256 decimale ed un valore di divisione del clock per 1, 8, 64 o 1024 scrivendo alle opportune locazioni di memoria.

Il conteggio del temporizzatore inizia non appena si realizza la scrittura e procederà con il rapporto di divisione stabilito.

Il temporizzatore decrementa il suo valore alla frequenza di clock divisa per il valore stabilito. Il conteggio del timer può essere letto a qualsiasi momento. A vostra scelta, il temporizzatore può essere programmato per generare un'interruzione quando il contatore oltrepassa lo zero nel suo conteggio a ritroso. Quando il conteggio zero è oltrepassato, il rapporto di divisione è automaticamente portato al valore 1 e il contatore continua il suo decremento alla velocità di clock partendo dal valore FF (che è -1 decimale nella rappresentazione aritmetica in complemento a 2).

Questo consente all'utilizzatore di determinare quanti cicli di clock sono passati dal momento che il timer raggiunse il conteggio: zero.

Dal momento che il contatore non si ferma mai, esso arriverà ancora al valore 00, quindi FF, e il conteggio continuerà indefinitamente.

1) Modo di operare:

a) Caricamento del timer.

Il rapporto di divisione e l'opzione di abilitare/disabilitare l'interruzione, sono programmate decodificando i bits meno significativi dell'indirizzo.

La partenza del conteggio per il temporizzatore è determinata dal valore scritto a quell'indirizzo.

<u>Indirizzi ai quali scrivere</u>	<u>Rapporto di divisione</u>	<u>L'interruzione è</u>
1704	1	Disabilitata
1705	8	Disabilitata
1706	64	Disabilitata
1707	1024	Disabilitata
170C	1	Abilitata
170D	8	Abilitata
170E	64	Abilitata
170F	1024	Abilitata

b) Determinazione dello stato del temporizzatore

A temporizzazione avviata, la lettura della locazione di indirizzo 1707 darà lo stato del temporizzatore. Se il contatore ha oltrepassato il conteggio di zero, il bit 7 sarà portato a 1, altrimenti il bit 7 (e tutti gli altri bits della locazione 1707) sarà zero.

Ciò permette al programma di "guardare" la locazione 1707 e determinare quando il temporizzatore ha terminato la temporizzazione.

c) Lettura del conteggio nel temporizzatore

Se il temporizzatore non ha oltrepassato lo zero nel conteggio, la lettura della locazione 1706 darà il conteggio corrente del tempo-

rizzatore e disabiliterà l'interruzione; la lettura della locazione 170E darà il valore corrente del temporizzatore e abiliterà l'interruzione. Così l'opzione di interruzione può essere cambiata mentre il temporizzatore sta procedendo nel suo conteggio.

Se il temporizzatore ha oltrepassato lo "0" durante il suo conteggio, la lettura di entrambe le locazioni di memoria 1706 o 170E ristabilirà il rapporto di divisione al valore precedentemente programmato, disabilitando l'interruzione e lasciando il temporizzatore nel suo attuale valore di conteggio (non il conteggio originalmente scritto al temporizzatore). Poichè il temporizzatore non si ferma mai di contare, il temporizzatore continuerà a decrementare, passerà per lo "0", stabilirà il rapporto di divisione a "1" e continuerà a decrementare alla frequenza di clock, fintanto che una nuova informazione non sarà scritta nel temporizzatore.

d) Uso della possibilità di interruzione

Per poter usare la possibilità di interruzione prima descritta, collegate PB7 (piedino 15 del connettore di applicazione) a $\overline{\text{IRQ}}$ (piedino 4 del connettore di espansione) oppure a $\overline{\text{NMI}}$ (piedino 6 del connettore di espansione) a seconda della funzione di interruzione desiderata. Dovrete programmare PB7 come linea di ingresso (che è lo stato normale dopo un reset).

Nota: Se desiderate usare PB7 come una normale linea di I/O, dovrete disabilitare la possibilità di interruzione del temporizzatore (scrivendo o leggendo l'indirizzo 1706) così che non interferisca con la normale funzione PB7.

Considerate inoltre che PB7 è stato progettato per essere collegato con altre possibili fonti di interruzione. Se non lo desiderate, usate una resistenza di pull-up da 5.1 K collegata tra PB7 e +5 V, ma questo non quando PB7 è collegato a $\overline{\text{NMI}}$ o $\overline{\text{IRQ}}$.

Appendice I

PROGRAMMA CONTENUTO COME “MONITOR” NEL KIM-1

CARD	LOC	CODE	CARD				
3		;		666666	555555	333333	000000
4		;		6	5	3	0 0
5		;		6	5	3	0 0
6		;		666666	555555	333333	0 0
7		;		6 6	5	3	0 0
8		;		6 6	5	3	0 0
9		;		666666	555555	333333	000000
10		;					
11		;					
12		;					
13		;					
14		;			000000	000000	333333
15		;			0 0	0 0	3
16		;		-----	0 0	0 0	3
17		;		-----	0 0	0 0	333333
18		;		-----	0 0	0 0	3
19		;			0 0	0 0	3
20		;			000000	000000	333333
21		;					
22		;					
23		;					
24		;					
25		;		COPYRIGHT			
26		;		MOS TECHNOLOGY INC.			
27		;		DATE OCT. 18 1975 REV. D			
28		;					
29		;					
30		;					
31		;		6530-003 IS AN AUDIO CASSETT TAPE			
32		;		RECORDER EXTENSION OF THE BASIC			
33		;		KIM MONITOR			
34		;					
35		;		IT FEATURES TWO BASIC ROUTINES			
36		;		LOADT-LOAD MEM FROM AUDIO TAPE			
37		;		DUMPT-STOR MEM ONTO AUDIO TAPE			
38		;					
39		;		LOADT			
40		;		ID=00	IGNORE ID		
41		;		ID=FF	IGN. ID USE SA FOR START ADDR		
42		;		ID=01-FE	IGN. ID USE ADDR ON TAPE		
43		;					
44		;		DUMPT			
45		;		ID=00	SHOULD NOT BE USED		
46		;		ID=FF	SHOULD NOT BE USED		
47		;		ID=01-FE	NORMAL ID RANGE		
48		;		SAL	LSB STARTING ADDRESS		
49		;		SAH	MSB		
50		;		EAL	LSB ENDING ADDRESS		
51		;		EAH	MSB		
52		;					

CARD	LOC	CODE	CARD
54		;	
55		;	EQUATES
56		;	SET UP FOR 6530-002 I / O
57		;	
58		SAD	=S1740 6530 A DATA
59		PADD	=S1741 6530 A DATA DIRECTION
60		SBD	=S1742 6530 B DATA
61		PBDD	=S1743 6530 B DATA DIRECTION
62		CLK1T	=S1744 DIV BY 1 TIME
63		CLK8T	=S1745 DIV BY 8 TIME
64		CLK64T	=S1746 DIV BY 64 TIME
65		CLKKT	=S1747 DIV BY 1024 TIME
66		CLKRDI	=S1747 READ TIME OUT BIT
67		CLKRDT	=S1746 READ TIME
68		;	
69	0000		W=S00EF
70		;	MPU REG. SAVX AREA IN PAGE 0
71		;	
72	00EF	PCL	W=W+1 PROGRAM CNT LOW
73	00F0	PCH	W=W+1 PROGRAM CNT HI
74	00F1	PREG	W=W+1 CURRENT STATUS REG.
75	00F2	SPUSER	W=W+1 CURRENT STACK POINT
76	00F3	ACC	W=W+1 ACCUMULATOR
77	00F4	YREG	W=W+1 Y INDEX
78	00F5	XREG	W=W+1 X INDEX
79		;	
80		;	KIM FIXED AREA IN PAGE 0
81		;	
82	00F6	CHKHI	W=W+1
83	00F7	CHKSUM	W=W+1
84	00F8	INL	W=W+1 INPUT BUFFER
85	00F9	INH	W=W+1 INPUT BUFFER
86	00FA	POINTL	W=W+1 LSB OF OPEN CELL
87	00FB	POINTH	W=W+1 MSB OF OPEN CELL
88	00FC	TEMP	W=W+1
89	00FD	TMPX	W=W+1
90	00FE	CHAR	W=W+1
91	00FF	MODE	W=W+1
92		;	
93		;	KIM FIXED AREA IN PAGE 23
94		;	
95	0100		W=S17E7
96	17E7	CHKL	W=W+1
97	17E8	CHKH	W=W+1
98	17E9	SAVX	W=W+3
99	17EC	VEB	W=W+6
100	17F2	CNTL30	W=W+1
101	17F3	CNTH30	W=W+1
102	17F4	TIMH	W=W+1
103	17F5	SAL	W=W+1
104	17F6	SAH	W=W+1
105	17F7	EAL	W=W+1

CARD	LOC	CODE	CARD	
106	17F8	EAH	Ⓢ=Ⓢ+1	HI ENDING ADDRESS
107	17F9	ID	Ⓢ=Ⓢ+1	
108		;		
109		;	INTERRUPT VECTORS	
110		;		
111	17FA	NMIV	Ⓢ=Ⓢ+2	STOP VECTOR (STOP=1C00)
112	17FC	RSTV	Ⓢ=Ⓢ+2	RST VECTOR
113	17FE	IRQV	Ⓢ=Ⓢ+2	IRQ VECTOR (BRK= 1C00)
114		;		

CARD	LOC	CODE	CARD
116	1800		*=\$1800
117			
118			INIT VOLATILE EXECUTION BLOCK
119			DUMP MEM TO TAPE
120			
121	1800	A9 AD	DUMPT LDA =/\$AD LOAD ABSOLUTE INST
122	1802	8D EC 17	STA VEB
123	1805	20 32 19	JSR INTVEB
124			
125	1808	A9 27	LDA =/\$27 TURN OFF DATAIN PB5
126	180A	8D 42 17	STA SBD
127	180D	A9 BF	LDA =/\$BF CONVERT PB7 TO OUTPUT
128	180F	8D 43 17	STA PBDD
129			
130	1812	A2 64	LDX =/\$64 100 CHARS
131	1814	A9 16	DUMPT1 LDA =/\$16 SYN CHAR'S
132	1816	20 7A 19	JSR QUTCHT
133	1819	CA	DEX
134	181A	DO F8	BNE DUMPT1
135			
136			
137	181C	A9 2A	LDA =/'* START CHAR
138	181E	20 7A 19	JSR OUTCHT
139			
140	1821	AD F9 17	LDA ID OUTPUT ID
141	1824	20 61 19	JSR OUTBT
142			
143	1827	AD F5 17	LDA SAL OUTPUT STARTING
144	182A	20 5E 19	JSR OUTBTC ADDRESS
145	182D	AD F6 17	LDA SAH
146	1830	20 5E 19	JSR OUTBTC
147			
148	1833	AD ED 17	DUMPT2 LDA VEB+1 CHECK FOR LAST
149	1836	CD F7 17	CMP EAL DATA BYTE
150	1839	AD EE 17	LDA VEB+2
151	183C	ED F8 17	SBC EAH
152	183F	90 24	BCC DUMPT4
153			
154	1841	A9 2F	LDA =/' / OUTPUT END OF DATA CHR
155	1843	20 7A 19	JSR OUTCHT
156	1846	AD E7 17	LDA CHKL LAST BYTE HAS BEEN
157	1849	20 61 19	JSR OUTBT OUT PUT NOW OUTPUT
158	184C	AD E8 17	LDA CHKH CHKSUM
159	184F	20 61 19	JSR OUTBT
160			
161			
162	1852	A2 02	LDX =/\$02 2 CHAR'S
163	1854	A9 04	DUMPT3 LDA =/\$04 EOT CHAR
164	1856	20 7A 19	JSR OUTCHT
165	1859	CA	DEX
166	185A	DO F8	BNE DUMPT3
167			

CARD	LOC	CODE	CARD		
168	185C	A9 00	LDA	≠ \$00	DISPLAY 0000
169	185E	85 FA	STA	POINTL	FOR NORMAL EXIT
170	1860	85 FB	STA	POINTH	
171	1862	4C 4F 1C	JMP	START	
172					
173	1865	20 EC 17	DUMPT4 JSR	VEB	DATA BYTE OUTPUT
174	1868	20 5E 19	JSR	OUTBTC	
175					
176	186B	20 EA 19	JSR	INCVEB	
177	186E	4C 33 18	JMP	DUMPT2	
178					
179					
180				LOAD MEMORY FROM TAPE	
181					
182	1871	0F 19	TAB	.WORD LOAD12	
183	1873	A9 8D	LOADT LDA	≠ \$8D	INIT VOLATILE EXECUTION
184	1875	8D EC 17	STA	VEB	BLOCK WITH STA ABS.
185	1878	20 32 19	JSR	INTVEB	
186					
187	187B	A9 4C	LDA	≠ \$4C	JUMP TYPE RTRN
188	187D	8D EF 17	STA	VEB+3	
189	1880	AD 71 18	LDA	TAB	
190	1883	8D FO 17	STA	VEB+4	
191	1886	AD 72 18	LDA	TAB+1	
192	1889	8D F1 17	STA	VEB+5	
193					
194	1880	A9 07	LDA	≠ \$07	RESET PB5=0 (DATA IN)
195	188E	8D 42 17	STA	SBD	
196					
197	1891	A9 FF	SYNC LDA	≠ \$FF	CLEAR SAVX FOR SYNC AREA
198	1893	8D E9 17	STA	SAVX	
199					
200	1896	20 41 1A	SYNC1 JSR	RDBIT	GET A BIT
201	1899	4E E9 17	LSR	SAVX	SHIFT BIT INTO CHAR
202	189C	0D E9 17	ORA	SAVX	
203	189F	8D E9 17	STA	SAVX	
204	18A2	AD E9 17	LDA	SAVX	GET NEW CHAR
205	18A5	C9 16	CMP	≠ \$16	SYN CHAR
206	18A7	DO ED	BNE	SYNC1	
207					
208	18A9	A2 0A	LDX	≠ \$0A	TEST FOR 10 SYN CHARS
209	18AB	20 24 1A	SYNC2 JSR	RDCHT	
210	18AE	C9 16	CMP	≠ \$16	
211	18B0	DO DF	BNE	SYNC	IF NOT 10 CHAR RE-SYNC
212	18B2	CA	DEX		
213	18B3	DO F6	BNE	SYNC2	
214					
215					
216	18B5	20 24 1A	LOADT4 JSR	RDCHT	LOOK FOR START OF
217	18B8	C9 2A	CMP	≠ '★	DATA CHAR
218	18BA	FO 06	BEQ	LOAD11	
219	18BC	C9 16	CMP	≠ \$16	IF NOT ★ SHOULD BE SYN

CARD	LOC	CODE	CARD		
220	18BE	DO D1	BNE	SYNC	
221	18CO	FO F3	BEQ	LOADT4	
222					
223	18C2	20 F3 19	LOAD11 JSR	RDBYT	READ ID FROM TAPE
224	18C5	CD F9 17	CMP	ID	COMPARE WITH REQUESTED ID
225	18C8	FO OD	BEQ	LOADT5	
226	18CA	AD F9 17	LDA	ID	DEFAULT OO READ RECORD
227	18CD	C9 OO	CMP	≠ \$00	ANYWAY
228	18CF	FO O6	BEQ	LOADT5	
229	18D1	C9 FF	CMP	≠ \$FF	DEFAULT FF IGNOR SA ON
230	18D3	FO 17	BEQ	LOADT6	TAPE
231	18D5	DO 9C	BNE	LOADT	
232					
233	18D7	20 F3 19	LOADT5 JSR	RDBYT	GET SA FROM TAPE
234	18DA	20 4C 19	JSR	CHKT	
235	18DD	8D ED 17	STA	VEB+1	SAVX IN VEB+1,2
236	18EO	20 F3 19	JSR	RDBYT	
237	18E3	20 4C 19	JSR	CHKT	
238	18E6	8D EE 17	STA	VEB+2	
239	18E9	4C F8 18	JMP	LOADT7	
240					
241	18EC	20 F3 19	LOADT6 JSR	RDBYT	GET SA BUT IGNORE
242	18EF	20 4C 19	JSR	CHKT	
243	18F2	20 F3 19	JSR	RDBYT	
244	18F5	20 4C 19	JSR	CHKT	
245					
246					
247	18F8	A2 O2	LOADT7 LDX	≠ \$02	GET 2 CHARS
248	18FA	20 24 1A	LOAD13 JSR	RDCHT	GET CHAR (X)
249	18FD	C9 2F	CMP	≠ '/'	LOOK FOR LAST CHAR
250	18FF	FO 14	BEQ	LOADT8	
251	1901	20 OO 1A	JSR	PACKT	CONVERT TO HEX
252	1904	DO 23	BNE	LOADT9	Y=1 NON-HEX CHAR
253	1906	CA	DEX		
254	1907	DO F1	BNE	LOAD13	
255					
256	1909	20 4C 19	JSR	CHKT	COMPUTE CHECKSUM
257	190C	4C EC 17	JMP	VEB	SAVX DATA IN MEMORY
258	190F	20 EA 19	LOAD12 JSR	INCVEB	INCREMENT DATA POINTER
259	1912	4C F8 18	JMP	LOADT7	
260					
261	1915	20 F3 19	LOADT8 JSR	RDBYT	END OF DATA COMPARE CHKSUM
262	1918	CD E7 17	CMP	CHKL	
263	191B	DO OC	BNE	LOADT9	
264	191D	20 F3 19	JSR	RDBYT	
265	1920	CD E8 17	CMP	CHKH	
266	1923	DO O4	BNE	LOADT9	
267	1925	A9 OO	LDA	≠ \$00	NORMAL EXIT
268	1927	FO O2	BEQ	LOAD10	
269					
270	1929	A9 FF	LOADT9 LDA	≠ \$FF	ERROR EXIT
271	192B	85 FA	LOAD10 STA	POINTL	

CARD	LOC	CODE	CARD
272	192D	85 FB	STA POINTH
273	192F	4C 4F 1C	JMP START
274			

;

CARD	LOC	CODE	CARD
276			;
277			;
278			;
279			;
280			;
281	1932	AD F5 17	INTVEB LDA SAL
282	1935	8D ED 17	STA VEB+1
283	1938	AD F6 17	LDA SAH
284	193B	8D EE 17	STA VEB+2
285	193E	A9 60	LDA ≠ \$60 RTS INST
286	1940	8D EF 17	STA VEB+3
287	1943	A9 00	LDA ≠ \$00 CLEAR CHKSUM AREA
288	1945	8D E7 17	STA CHKL
289	1948	8D E8 17	STA CHKH
290	194B	60	RTS
291			;
292			;
293			;
294			;
295	194C	A8	CHKT TAY
296	194D	18	CLC
297	194E	6D E7 17	ADC CHKL
298	1951	8D E7 17	STA CHKL
299	1954	AD E8 17	LDA CHKH
300	1957	69 00	ADC ≠ \$00
301	1959	8D E8 17	STA CHKH
302	195C	98	TYA
303	195D	60	RTS
304			;
305			;
306			;
307			;
308	195E	20 4C 19	OUTBTC JSR CHKT COMP CHKSUM
309	1961	A8	OUTBT TAY SAVX DATA BYTE
310	1962	4A	LSR A SHIFT OFF LSD
311	1963	4A	LSR A
312	1964	4A	LSR A
313	1965	4A	LSR A
314	1966	20 6F 19	JSR HEXOUT OUT PUT MSD
315	1969	98	TYA
316	196A	20 6F 19	JSR HEXOUT OUT PUT LSD
317	196D	98	TYA
318	196E	60	RTS
319			;
320			;
321			;
322			;
323	196F	29 0F	HEXOUT AND ≠ \$0F
324	1971	C9 0A	CMP ≠ \$0A
325	1973	18	CLC
326	1974	30 02	BMI HEX1
327	1976	69 07	ADC ≠ \$07

CARD	LOC	CODE	CARD		
328	1978	69 30	HEX1	ADC	≠ \$30
329					
330				OUTPUT TO TAPE ONE ASCII	
331				CHAR USE SUB'S ONE + ZRO	
332					
333	197A	8E E9 17	OUTCHT	STX	SAVX
334	197D	8C EA 17		STY	SAVX+1
335	1980	A0 08		LDY	≠ \$08 START BIT
336	1982	20 9E 19	CHT1	JSR	ONE
337	1985	4A		LSR	A GET DATA BIT
338	1986	B0 06		BCS	CHT2
339	1988	20 9E 19		JSR	ONE DATA BIT=1
340	198B	4C 91 19		JMP	CHT3
341	198E	20 C4 19	CHT2	JSR	ZRO DATA BIT=0
342	1991	20 C4 19	CHT3	JSR	ZRO
343	1994	88		DEY	
344	1995	DO EB		BNE	CHT1
345	1997	AE E9 17		LDX	SAVX
346	199A	AC EA 17		LDY	SAVX+1
347	199D	60		RTS	
348					
349					
350				OUTPUT 1 TO TAPE	
351				9 PULSES 138 MICROSEC EACH	
352					
353	199E	A2 09	ONE	LDX	≠ \$09
354	19A0	48		PHA	SAVX A
355	19A1	2C 47 17	ONE1	BIT	CLKRDI WAIT FOR TIME OUT
356	19A4	10 FB		BPL	ONE1
357	19A6	A9 7E		LDA	≠ 126
358	19A8	8D 44 17		STA	CLK1T
359	19AB	A9 A7		LDA	≠ \$A7
360	19AD	8D 42 17		STA	SBD SET PB7=1
361	19B0	2C 47 17	ONE2	BIT	CLKRDI
362	19B3	10 FB		BPL	ONE2
363	19B5	A9 7E		LDA	≠ 126
364	19B7	8D 44 17		STA	CLK1T
365	19BA	A9 27		LDA	≠ \$27
366	19BC	8D 42 17		STA	SBD RESET PB7=0
367	19BF	CA		DEX	
368	19C0	DO DF		BNE	ONE1
369	19C2	68		PLA	
370	19C3	60		RTS	
371					
372					
373				OUTPUT 0 TO TAPE	
374				6 PULSES 207 MICROSEC EACH	
375					
376	19C4	A2 06	ZRO	LDX	≠ \$06
377	19C6	48		PHA	SAVX A
378	19C7	2C 47 17	ZRO1	BIT	CLKRDI
379	19CA	10 FB		BPL	ZRO1

CARD	LOC	CODE	CARD	
380	19CC A9	C3	LDA	195
381	19CE 8D	44 17	STA	CLK1T
382	19D1 A9	A7	LDA	195 A7
383	19D3 8D	42 17	STA	SBD SET PB7=1
384	19D6 2C	47 17	BIT	CLKRDI
385	19D9 10	FB	BPL	ZR02
386	19DB A9	C3	LDA	195
387	19DD 8D	44 17	STA	CLK1T
388	19E0 A9	27	LDA	195 27
389	19E2 8D	42 17	STA	SBD RESET PB7=0
390	19E5 CA		DEX	
391	19E6 DO	DF	BNE	ZR01
392	19E8 68		PLA	RESTORE A
393	19E9 60		RTS	
394				
395				
396				
397	19EA EE	ED 17	INCVEB	INC VEB+1
398	19ED DO	O3	BNE	INCVE1
399	19EF EE	EE 17	INC	VEB+2
400	19F2 60		INCVE1	RTS
401				
402				
403				
404	19F3 20	24 1A	RDBYT	JSR RDCHT
405	19F6 20	00 1A	JSR	PACKT
406	19F9 20	24 1A	RDBYT2	JSR RDCHT
407	19FC 20	00 1A	JSR	PACKT
408	19FF 60		RTS	
409				
410				
411				
412				
413	1A00 C9	30	PACKT	CMP 195 30
414	1A02 30	1E	BMI	PACKT3
415	1A04 C9	47	CMP	195 47
416	1A06 10	1A	BPL	PACKT3
417	1A08 C9	40	CMP	195 40
418	1A0A 30	03	BMI	PACKT1
419	1A0C 18		CLC	
420	1A0D 69	09	ADC	195 09
421	1A0F 2A		PACKT1	ROL A
422	1A10 2A			ROL A
423	1A11 2A			ROL A
424	1A12 2A			ROL A
425	1A13 AO	04	LDY	195 04
426	1A15 2A		PACKT2	ROL A
427	1A16 2E	E9 17		ROL SAVX
428	1A19 88		DEY	
429	1A1A DO	F9	BNE	PACKT2
430	1A1C AD	E9 17	LDA	SAVX
431	1A1F AO	00	LDY	195 00 Y=0 VALID HEX CHAR

CARD	LOC	CODE	CARD		
432	1A21	60	RTS		Y=0 VALID HEX
433	1A22	C8	PACKT3	INY	Y=1 NOT HEX
434	1A23	60	RTS		
435					
436					
437					
438					
439	1A24	8E EB 17	RDCHT	STX SAVX+2	
440	1A27	A2 08	LDX	≠ \$08	READ 8 BITS
441	1A29	20 41 1A	RDCHT1	JSR RDBIT	GET NEXT DATA BIT
442	1A2C	4E EA 17	LSR	SAVX+1	RIGHT SHIFT CHAR
443	1A2F	OD EA 17	ORA	SAVX+1	OR IN SIGN BIT
444	1A32	8D EA 17	STA	SAVX+1	REPLACE CHAR
445	1A35	CA	DEX		
446	1A36	DO F1	BNE	RDCHT1	
447					
448	1A38	AD EA 17	LDA	SAVX+1	MOVE CHAR INTO A
449	1A3B	2A	ROL	A	SHIFT OFF PARITY
450	1A3C	4A	LSR	A	
451	1A3D	AE EB 17	LDX	SAVX+2	
452	1A40	60	RTS		
453					
454					
455					
456					
457	1A41	2C 42 17	RDBIT	BIT SBD	WAIT FOR END OF START BIT
458	1A44	10 FB	BPL	RDBIT	
459	1A46	AD 46 17	LDA	CLKRDT	GET START BIT TIME
460	1A49	AO FF	LDY	≠ \$FF	A=256-T1
461	1A4B	8C 46 17	STY	CLK64T	SET UP TIMER
462					
463	1A4E	AO 14	LDY	≠ \$14	
464	1A50	88	RDBIT3	DEY	DELAY 100 MICROSEC
465	1A51	DO FD	BNE	RDBIT3	
466					
467	1A53	2C 42 17	RDBIT2	BIT SBD	
468	1A56	30 FB	BMI	RDBIT2	WAIT FOR NEXT START BIT
469					
470	1A58	38	SEC		
471	1A59	ED 46 17	SBC	CLKRDT	(256-T1) - (256-T2)=T2-T1
472	1A5C	AO FF	LDY	≠ \$FF	
473	1A5E	8C 46 17	STY	CLK64T	SET UP TIMER FOR NEXT BIT
474					
475	1A61	AO 07	LDY	≠ \$07	
476	1A63	88	RDBIT4	DEY	DELAY 50 MICROSEC
477	1A64	DO FD	BNE	RDBIT4	
478					
479	1A66	49 FF	EOR	≠ \$FF	COMPLEMENT SIGN OF A
480	1A68	29 80	AND	≠ \$80	MASK ALL EXCEPT SIGN
481	1A6A	60	RTS		

CARD	LOC	CODE	CARD		
483				;	
484				;	DIAGNOSTICS
485				;	MEMORY
486				;	PLLCAL
487				;	
488				;	
489				;	
490				;	PLLCAL OUTPUT 166 MICROSEC
491				;	PULSE STRING
492				;	
493	1A6B	A9 27	PLLCAL	LDA	≠ \$27
494	1A6D	8D 42 17		STA	SBD TURN OFF DATIN PB5=1
495	1A70	A9 BF		LDA	≠ \$BF CONVERT PB7 TO OUTPUT
496	1A72	8D 43 17		STA	PBDD
497				;	
498	1A75	2C 47 17	PLL1	BIT	CLKRDI
499	1A78	10 FB		BPL	PLL1
500	1A7A	A9 9A		LDA	≠ 154 WAIT 166 MICROSEC
501	1A7C	8D 44 17		STA	CLK1T
502	1A7F	A9 A7		LDA	≠ \$A7 OUTPUT PB7=1
503	1A81	8D 42 17		STA	SBD
504				;	
505	1A84	2C 47 17	PLL2	BIT	CLKRDI
506	1A87	10 FB		BPL	PLL2
507	1A89	A9 9A		LDA	≠ 154
508	1A8B	8D 44 17		STA	CLK1T
509	1A8E	A9 27		LDA	≠ \$27 PB7=0
510	1A90	8D 42 17		STA	SBD
511	1A93	4C 75 1A		JMP	PLL1
512				;	
513				;	
514				;	INTERRUPTS PAGE 27
515				;	
516	1A96			★ = ★ +\$0164	RESERVED FOR TEST
517	1BFA	6B 1A	NMIP27	.WORD	PLLCAL
518	1BFC	6B 1A	RSTP27	.WORD	PLLCAL
519	1BFE	6B 1A	IRQP27	.WORD	PLLCAL
520				;	

CARD	LOC	CODE	CARD
522		;	
523		;	
524		;	
525		;	
526		;	666666 555555 333333 000000
527		;	6 5 3 0 0
528		;	6 5 3 0 0
529		;	666666 555555 333333 0 0
530		;	6 6 5 3 0 0
531		;	6 6 5 3 0 0
532		;	666666 555555 333333 000000
533		;	
534		;	
535		;	
536		;	000000 000000 222222
537		;	0 0 0 0 2
538		;	----- 0 0 0 0 2
539		;	----- 0 0 0 0 222222
540		;	----- 0 0 0 0 2
541		;	0 0 0 0 2
542		;	000000 000000 222222
543		;	

CARD	LOC	CODE	CARD
545		;	
546		;	
547		;	
548		;	
549		;	COPYRIGHT
550		;	MOS TECHNOLOGY INC.
551		;	DATE OCT 13 1975 REV. E
552		;	
553		;	KIM : TTY INTERFACE
554		;	: KEYBOARD INTERFACE
555		;	: 7 SEG 6 DIGIT DISPLAY
556		;	
557		;	
558		;	TTY CMDS:
559		;	G GOEXEC
560		;	CR OPEN NEXT CELL
561		;	LF OPEN PREV. CELL
562		;	MODIFY OPEN CELL
563		;	SP OPEN NEW CELL
564		;	L LOAD (OBJECT FORMAT)
565		;	Q DUMP FROM OPEN CELL ADDR TO HI LIMIT
566		;	RO RUB OUT - RETURN TO START (KIM)
567		;	((ALL ILLEGAL CHAR ARE IGNORED))
568		;	
569		;	KEYBOARD CMDS:
570		;	ADDR SETS MODE TO MODIFY CELL ADDRESS
571		;	DATA SETS MODE TO MODIFY DATA IN OPEN CELL
572		;	STEP INCREMENTS TO NEXT CELL
573		;	RST SYSTEM RESET
574		;	RUN GOEXEC
575		;	STOP \$1COO CAN BE LOADED INTO NMIV TO
576		;	USE STOP FEATURE
577		;	PC DISPLAY PC
578		;	
579		;	CLOCK IS NOT DISABLED IN SIGMA 1
580		;	
581		;	
582		;	

CARD	LOC	CODE	CARD	
636	1C4F 20	8C 1E	START JSR	INIT1
637	1C52 A9	01	LDA	≠ \$01
638	1C54 2C	40 17	BIT	SAD
639	1C57 D0	1E	BNE	TTYKB
640	1C59 20	2F 1E	JSR	CRLF
641	1C5C A2	0A	LDX	≠ \$0A
642	1C5E 20	31 1E	JSR	PRTST
643	1C61 4C	AF 1D	JMP	SHOW1
644				
645	1C64 A9	00	CLEAR LDA	≠ \$00
646	1C66 85	F8	STA	INL
647	1C68 85	F9	STA	INH
648	1C6A 20	5A 1E	READ JSR	GETCH
649	1C6D C9	01	CMP	≠ \$01
650	1C6F F0	06	BEQ	TTYKB
651	1C71 20	AC 1F	JSR	PACK
652	1C74 4C	DB 1D	JMP	SCAN
653				
654				
655				
656				
657	1C77 20	19 1F	TTYKB JSR	SCAND
658	1C7A D0	D3	BNE	START
659	1C7C A9	01	TTYKB1 LDA	≠ \$01
660	1C7E 2C	40 17	BIT	SAD
661	1C81 F0	CC	BEQ	START
662	1C83 20	19 1F	JSR	SCAND
663	1C86 F0	F4	BEQ	TTYKB1
664	1C88 20	19 1F	JSR	SCAND
665	1C8B F0	EF	BEQ	TTYKB1
666				
667	1C8D 20	6A 1F	GETK JSR	GETKEY
668	1C90 C9	15	CMP	≠ \$15
669	1C92 10	BB	BPL	START
670	1C94 C9	14	CMP	≠ \$14
671	1C96 F0	44	BEQ	PCCMD
672	1C98 C9	10	CMP	≠ \$10
673	1C9A F0	2C	BEQ	ADDRM
674	1C9C C9	11	CMP	≠ \$11
675	1C9E F0	2C	BEQ	DATAM
676	1CA0 C9	12	CMP	≠ \$12
677	1CA2 F0	2F	BEQ	STEP
678	1CA4 C9	13	CMP	≠ \$13
679	1CA6 F0	31	BEQ	GOV
680	1CA8 0A		DATA ASL	A
681	1CA9 0A		ASL	A
682	1CAA 0A		ASL	A
683	1CAB 0A		ASL	A
684	1CAC 85	FC	STA	TEMP
685	1CAE A2	04	LDX	≠ \$04
686	1CB0 A4	FF	DATA1 LDY	MODE
687	1CB2 D0	0A	BNE	ADDR

PRT CR LF
TYPE OUT KIM

CLEAR INPUT BUFFER

GET CHAR

MAIN ROTINE FOR KEY BOARD
AND DISPLAY

IF A=0 NO KEY

DISPLAY PC
ADDR MODE=1
DATA MODE = 1

STEP

RUN

SHIFT CHAR INTO HIGH
ORDER NIBBLE

STORE IN TEMP

TEST MODE 1=ADDR
MODE=0 DATA

CARD	LOC	CODE	CARD
688	1CB4 B1	FA	LDA (POINTL),Y GET DATA
689	1CB6 06	FC	ASL TEMP SHIFT CHAR
690	1CB8 2A		ROL A SHIFT DATA
691	1CB9 91	FA	STA (POINTL),Y STORE OUT DATA
692	1CBB 4C C3 1C		JMP DATA2
693		;	
694	1CBE 0A	ADDR	ASL A SHIFT CHAR
695	1CBF 26	FA	ROL POINTL SHIFT ADDR
696	1CC1 26	FB	ROL POINTH SHIFT ADDR HI
697	1CC3 CA	DATA2	DEX
698	1CC4 DO	EA	BNE DATA1 DO 4 TIMES
699	1CC6 FO	O8	BEQ DATAM2 EXIT HERE
700		;	
701	1CC8 A9	O1 ADDR	LDA =/\$01
702	1CCA DO	O2	BNE DATAM1
703		;	
704	1CCC A9	O0 DATAM	LDA =/\$00
705	1CCE 85	FF DATAM1	STA MODE
706	1CDO 4C 4F 1C	DATAM2	JMP START
707		;	
708	1CD3 20 63 1F	STEP	JSR INCPT
709	1CD6 4C 4F 1C		JMP START
710		;	
711	1CD9 4C C8 1D	GOV	JMP GOEXEC
712		;	
713		;	
714		;	DISPLAY PC BY MOVING
715		;	PC TO POINT
716		;	
717	1CDC A5	EF PCCMD	LDA PCL
718	1CDE 85	FA	STA POINTL
719	1CEO A5	FO	LDA PCH
720	1CE2 85	FB	STA POINTH
721	1CE4 4C 4F 1C		JMP START
722		;	
723		;	LOAD PAPER TAPE FROM TTY
724		;	
725	1CE7 20 5A 1E	LOAD	JSR GETCH LOOK FOR FIRST CHAR
726	1CEA C9	3B	CMP =/\$3B SMICOLON
727	1CEC DO	F9	BNE LOAD
728	1CEE A9	O0 LOADS	LDA =/\$00
729	1CF0 85	F7	STA CHKSUM
730	1CF2 85	F6	STA CHKHI
731		;	
732	1CF4 20 9D 1F		JSR GETBYT GET BYTE CNT
733	1CF7 AA		TAX SAVE IN X INDEX
734	1CF8 20 91 1F		JSR CHK COMPUTE CHKSUM
735		;	
736	1CFB 20 9D 1F		JSR GETBYT GET ADDRESS HI
737	1CFE 85	FB	STA POINTH
738	1DO0 20 91 1F		JSR CHK
739	1DO3 20 9D 1F		JSR GETBYT GET ADDRESS LO

CARD	LOC	CODE	CARD		
740	1D06 95	FA	STA	POINTL	
741	1D08 20	91 1F	JSR	CHK	
742					
743	1DOB 8A		TXA		IF CNT=0 DONT
744	1DOC FO	OF	BEQ	LOAD3	GET ANY DATA
745					
746	1DOE 20	9D 1F	LOAD2 JSR	GETBYT	GET DATA
747	1D11 91	FA	STA	(POINTL),Y	STORE DATA
748	1D13 20	91 1F	JSR	CHK	
749	1D16 20	63 1F	JSR	INCPT	NEXT ADDRESS
750	1D19 CA		DEX		
751	1D1A DO	F2	BNE	LOAD2	
752	1D1C E8		INX		X=1 DATA RECORD
753					X=0 LAST RECORD
754	1D1D 20	9D 1F	LOAD3 JSR	GETBYT	COMPARE CHKSUM
755	1D20 C5	F6	CMP	CHKHI	
756	1D22 DO	17	BNE	LOADE1	
757	1D24 20	9D 1F	JSR	GETBYT	
758	1D27 C5	F7	CMP	CHKSUM	
759	1D29 DO	13	BNE	LOADER	
760					
761	1D2B 8A		TXA		X=0 LAST RECORD
762	1D2C DO	B9	BNE	LOAD	
763					
764	1D2E A2	OC	LOAD7 LDX	≠/\$0C	X-OFF KIM
765	1D30 A9	27	LOAD8 LDA	≠/\$27	
766	1D32 8D	42 17	STA	SBD	DISABLE DATA IN
767	1D35 20	31 1E	JSR	PRTST	
768	1D38 4C	4F 1C	JMP	START	
769					
770	1D3B 20	9D 1F	LOADE1 JSR	GETBYT	DUMMY
771	1D3E A2	11	LOADER LDX	≠/\$11	X-OFF ERR KIM
772	1D40 DO	EE	BNE	LOAD8	
773					
774				DUMP TO TTY	
775				FROM OPEN CELL ADDRESS	
776				TO LIMHL, LIMHH	
777					
778	1D42 A9	00	DUMP LDA	≠/\$00	
779	1D44 85	F8	STA	INL	
780	1D46 85	F9	STA	INH	CLEAR RECORD COUNT
781	1D48 A9	00	DUMPO LDA	≠/\$00	
782	1D4A 85	F6	STA	CHKHI	CLEAR CHKSUM
783	1D4C 85	F7	STA	CHKSUM	
784					
785	1D4E 20	2F 1E	DUMP1 JSR	CRLF	PRINT CR LF
786	1D51 A9	3B	LDA	≠/\$3B	PRINT SMICOLON
787	1D53 20	A0 1E	JSR	OUTCH	
788	1D56 A5	FA	LDA	POINTL	TEST POINT GT OR ET
789	1D58 CD	F7 17	CMP	EAL	HI LIMIT GO TO EXIT
790	1D5B A5	FB	LDA	POINTH	
791	1D5D ED	F8 17	SBC	EAH	

CARD	LOC	CODE	CARD		
792	1D60 90	18	BCC	DUMP4	
793					
794	1D62 A9	00	LDA	18 \$00	PRINT LAST RECORD
795	1D64 20	3B 1E	JSR	PRTBYT	0 BYTES
796	1D67 20	CC 1F	JSR	OPEN	
797	1D6A 20	1E 1E	JSR	PRTPNT	
798					
799	1D6D A5	F6	LDA	CHKHI	PRINT CHKSUM
800	1D6F 20	3B 1E	JSR	PRTBYT	FOR LAST RECORD
801	1D72 A5	F7	LDA	CHKSUM	
802	1D74 20	3B 1E	JSR	PRTBYT	
803	1D77 4C	64 1C	JMP	CLEAR	
804					
805	1D7A A9	18	DUMP4 LDA	18 \$18	PRINT 24 BYTE CNT
806	1D7C AA		TAX		SAVE AS INDEX
807	1D7D 20	3B 1E	JSR	PRTBYT	
808	1D80 20	91 1F	JSR	CHK	
809	1D83 20	1E 1E	JSR	PRTPNT	
810					
811	1D86 A0	00	DUMP2 LDY	18 \$00	PRINT 24 BYTES
812	1D88 B1	FA	LDA	(POINTL),Y	GET DATA
813	1D8A 20	3B 1E	JSR	PRTBYT	PRINT DATA
814	1D8D 20	91 1F	JSR	CHK	COMP CHKSUM
815	1D90 20	63 1F	JSR	INCPT	INCREMENT POINT
816	1D93 CA		DEX		
817	1D94 D0	F0	BNE	DUMP2	
818					
819	1D96 A5	F6	LDA	CHKHI	PRINT CHKSUM
820	1D98 20	3B 1E	JSR	PRTBYT	
821	1D9B A5	F7	LDA	CHKSUM	
822	1D9D 20	3B 1E	JSR	PRTBYT	
823	1DA0 E6	F8	INC	INL 1	INCREMENT RECORD CNT
824	1DA2 D0	02	BNE	DUMP3	
825	1DA4 E6	F9	INC	INH	
826	1DA6 4C	48 1D	DUMP3 JMP	DUMPO	
827					
828	1DA9 20	CC 1F	SPACE JSR	OPEN	OPEN NEW CELL
829	1DAC 20	2F 1E	SHOW JSR	CRLF	PRINT CR LF
830	1DAF 20	1E 1E	SHOW1 JSR	PRTPNT	
831	1DB2 20	9E 1E	JSR	OUTSP	PRT SPACE
832	1DB5 A0	00	LDY	18 \$00	PRINT DATA SPECIFIED
833	1DB7 B1	FA	LDA	(POINTL),Y	BY POINT AD = LDA EXT
834	1DB9 20	3B 1E	JSR	PRTBYT	
835	1DBC 20	9E 1E	JSR	OUTSP	PRT SPACE
836	1DBF 4C	64 1C	JMP	CLEAR	
837					
838	1DC2 20	63 1F	RTRN JSR	INCPT	OPEN NEXT CELL
839	1DC5 4C	AC 1D	JMP	SHOW	
840					
841	1DC8 A6	F2	GOEXEC LDX	SPUSER	
842	1DCA 9A		TXS		
843	1DCB A5	FB	LDA	POINTH	PROGRAM RUNS FROM

CARD	LOC	CODE	CARD	
844	1DCD 48		PHA	OPEN CELL ADDRESS
845	1DCE A5 FA		LDA POINTL	
846	1DDO 48		PHA	
847	1DD1 A5 F1		LDA PREG	
848	1DD3 48		PHA	
849	1DD4 A6 F5		LDX XREG	RESTORE REGS
850	1DD6 A4 F4		LDY YREG	
851	1DD8 A5 F3		LDA ACC	
852	1DDA 40		RTI	
853				
854	1ddb C9 20	SCAN	CMP ≠ \$20	OPEN CELL
855	1DDD FO CA		BEQ SPACE	
856	1DDF C9 7F		CMP ≠ \$7F	RUB OUT (KIM)
857	1DE1 FO 1B		BEQ STV	
858	1DE3 C9 OD		CMP ≠ \$OD	NEXT CELL
859	1DE5 FO DB		BEQ RTRN	
860	1DE7 C9 OA		CMP ≠ \$OA	PREV CELL
861	1DE9 FO 1C		BEQ FEED	
862	1DEB C9 2E		CMP ≠ '.	MODIFY CELL
863	1DED FO 26		BEQ MODIFY	
864	1DEF C9 47		CMP ≠ 'G	GO EXEC
865	1DF1 FO D5		BEQ GOEXEC	
866	1DF3 C9 51		CMP ≠ 'Q	DUMP FROM OPEN CELL TO HI LIMIT
867	1DF5 FO OA		BEQ DUMPV	
868	1DF7 C9 4C		CMP ≠ 'L	LOAD TAPE
869	1DF9 FO 09		BEQ LOADV	
870	1DFB 4C 6A 1C		JMP READ	IGNORE ILLEGAL CHAR
871				
872	1DFE 4C 4F 1C	STV	JMP START	
873	1EO1 4C 42 1D	DUMPV	JMP DUMP	
874	1EO4 4C E7 1C	LOADV	JMP LOAD	
875				
876	1EO7 38	FEED	SEC	
877	1EO8 A5 FA		LDA POINTL	DEC DOUBLE BYTE
878	1EOA E9 01		SBC ≠ \$01	AT POINTL AND POINTH
879	1EOC 85 FA		STA POINTL	
880	1EOE B0 02		BCS FEED1	
881	1E10 C6 FB		DEC POINTH	
882	1E12 4C AC 1D	FEED1	JMP SHOW	
883				
884	1E15 A0 00	MODIFY	LDY ≠ \$00	GET CONTENTS OF INPUT BUFF
885	1E17 A5 F8		LDA INL	INL AND STOR IN LOC
886	1E19 91 FA		STA (POINTL),Y	SPECIFIED BY POINT
887	1E1B 4C C2 1D		JMP RTRN	
888				
889			END OF MAIN LINE	

CARD	LOC	CODE	CARD		
891				;	SUBROUTINES FOLLOW
892				;	
893				;	
894				;	
895				;	SUB TO PRINT POINTL, POINTH
896				;	
897	1E1E	A5 FB	PRTPNT	LDA	POINTH
898	1E20	20 3B 1E		JSR	PRTBYT
899	1E23	20 91 1F		JSR	CHK
900	1E26	A5 FA		LDA	POINTL
901	1E28	20 3B 1E		JSR	PRTBYT
902	1E2B	20 91 1F		JSR	CHK
903	1E2E	60		RTS	
904				;	
905				;	PRINT STRING OF ASCII CHAR FROM
906				;	TOP+X TO TOP
907				;	
908	1E2F	A2 07	CRLF	LDX	≠ \$07
909	1E31	BD D5 1F	PRTST	LDA	TOP,X
910	1E34	20 AO 1E		JSR	OUTCH
911	1E37	CA		DEX	
912	1E38	10 F7		BPL	PRTST STOP ON INDEX ZERO
913	1E3A	60	PRT1	RTS	
914				;	
915				;	PRINT 1 HEX BYTE AS TWO ASCII CHAR'S
916				;	
917	1E3B	85 FC	PRTBYT	STA	TEMP
918	1E3D	4A		LSR	A SHIFT CHAR RIGHT 4 BITS
919	1E3E	4A		LSR	A
920	1E3F	4A		LSR	A
921	1E40	4A		LSR	A
922	1E41	20 4C 1E		JSR	HEXTA CONVERT TO HEX AND PRINT
923	1E44	A5 FC		LDA	TEMP GET OTHER HALF
924	1E46	20 4C 1E		JSR	HEXTA CONVERT TO HEX AND PRINT
925	1E49	A5 FC		LDA	TEMP RESTORE BYTE IN A AND RETURN
926	1E4B	60		RTS	
927				;	
928	1E4C	29 OF	HEXTA	AND	≠ \$0F MASK HI 4 BITS
929	1E4E	C9 OA		CMP	≠ \$0A
930	1E50	18		CLC	
931	1E51	30 02		BMI	HEXTA1
932	1E53	69 07		ADC	≠ \$07 ALPHA HEX
933	1E55	69 30	HEXTA1	ADC	≠ \$30 DEC HEX
934	1E57	4C AO 1E		JMP	OUTCH PRINT CHAR
935				;	
936				;	GET 1 CHAR FROM TTY
937				;	RETURN FROM SUB WITH CHAR IN A
938				;	X IS PRESERVED AND Y RETURNED = FF
939				;	
940	1E5A	86 FD	GETCH	STX	TMPX SAVE X REG
941	1E5C	A2 08		LDX	≠ \$08 SET UP 8 BIT CNT
942	1E5E	A9 01		LDA	≠ \$01

CARD	LOC	CODE	CARD		
943	1E60	2C 40 17	GET1	BIT	SAD
944	1E63	D0 22		BNE	GET6
945	1E65	30 F9		BMI	GET1
946	1E67	20 D4 1E		JSR	DELAY
947	1E6A	20 EB 1E	GET5	JSR	DEHALF
948	1E6D	AD 40 17	GET2	LDA	SAD
949	1E70	29 80		AND	≠ \$80
950	1E72	46 FE		LSR	CHAR
951	1E74	05 FE		ORA	CHAR
952	1E76	85 FE		STA	CHAR
953	1E78	20 D4 1E		JSR	DELAY
954	1E7B	CA		DEX	
955	1E7C	D0 EF		BNE	GET2
956	1E7E	20 EB 1E		JSR	DEHALF
957					
958	1E81	A6 FD		LDX	TMPX
959	1E83	A5 FE		LDA	CHAR
960	1E85	2A		ROL	A
961	1E86	4A		LSR	A
962	1E87	60	GET6	RTS	
963					
964					
965					
966	1E88	A2 01	INITS	LDX	≠ \$01
967	1E8A	86 FF		STX	MODE
968					
969	1E8C	A2 00	INIT1	LDX	≠ \$00
970	1E8E	8E 41 17		STX	PADD
971	1E91	A2 3F		LDX	≠ \$3F
972	1E93	8E 43 17		STX	PBDD
973	1E96	A2 07		LDX	≠ \$07
974	1E98	8E 42 17		STX	SBD
975	1E9B	D8		CLD	
976	1E9C	78		SEI	
977	1E9D	60		RTS	
978					
979					
980					
981					
982					
983	1E9E	A9 20	OUTSP	LDA	≠ \$20
984	1EA0	85 FE	OUTCH	STA	CHAR
985	1EA2	86 FD		STX	TMPX
986	1EA4	20 D4 1E		JSR	DELAY
987	1EA7	AD 42 17		LDA	SBD
988	1EAA	29 FE		AND	≠ \$FE
989	1EAC	8D 42 17		STA	SBD
990	1EAF	20 D4 1E		JSR	DELAY
991	1EB2	A2 08		LDX	≠ \$08
992	1EB4	AD 42 17	OUT1	LDA	SBD
993	1EB7	29 FE		AND	≠ \$FE
994	1EB9	46 FE		LSR	CHAR

CARD	LOC	CODE	CARD
995	1EBB 69	00	ADC $\neq \$00$
996	1EBD 8D	42 17	STA SBD
997	1ECO 20	D4 1E	JSR DELAY
998	1EC3 CA		DEX
999	1EC4 D0	EE	BNE OUT1
1000	1EC6 AD	42 17	LDA SBD STOP BIT
1001	1EC9 09	01	ORA $\neq \$01$
1002	1ECB 8D	42 17	STA SBD
1003	1ECE 20	D4 1E	JSR DELAY STOP BIT
1004	1ED1 A6	FD	LDX TMPX RESTORE INDEX
1005	1ED3 60		RTS
1006			
1007			
1008			DELAY 1 BIT TIME
1009			AS DETERMEND BY DETCPS
1010	1ED4 AD	F3 17	DELAY LDA CNTH30 THIS LOOP SIMULATES THE
1011	1ED7 8D	F4 17	STA TIMH DETCPS SECTION AND WILL DELAY
1012	1EDA AD	F2 17	LDA CNTL30 1 BIT TIME
1013	1EDD 38		DE2 SEC
1014	1EDE E9	01	DE4 SBC $\neq \$01$
1015	1EE0 B0	03	BCS DE3
1016	1EE2 CE	F4 17	DEC TIMH
1017	1EE5 AC	F4 17	DE3 LDY TIMH
1018	1EE8 10	F3	BPL DE2
1019	1EEA 60		RTS
1020			
1021			
1022	1EEB AD	F3 17	DEHALF LDA CNTH30 DELAY HALF BIT TIME
1023	1EEE 8D	F4 17	STA TIMH DOUBLE RIGHT SHIFT OF DELAY
1024	1EF1 AD	F2 17	LDA CNTL30 CONSTANT FOR A DIV BY 2
1025	1EF4 4A		LSR A
1026	1EF5 4E	F4 17	LSR TIMH
1027	1EF8 90	E3	BCC DE2
1028	1EFA 09	80	ORA $\neq \$80$
1029	1EFC B0	E0	BCS DE4
1030			
1031			
1032			SUB TO DETERMINE IF KEY IS
1033			DEPRESSED OR COMDITON OF SSW
1034			KEY NOT DEP OR TTY MODE A = 0
1035			KEY DEP OR KB MODE A NOT ZERO
1036			
1037	1EFE A0	03	AK LDY $\neq \$03$ 3 ROWS
1038	1FO0 A2	01	LDX $\neq \$01$ DIGIT 0
1039			
1040	1FO2 A9	FF	ONEKEY LDA $\neq \$FF$
1041	1FO4 8E	42 17	AK1 STX SBD OUTPUT DIGIT
1042	1FO7 E8		INX GET NXT DIGIT
1043	1FO8 E8		INX
1044	1FO9 2D	40 17	AND SAD INPUT SEGMENTS
1045	1FOC 88		DEY
1046	1FOD D0	F5	BNE AK1

CARD	LOC	CODE	CARD			
1047				;		
1048	1FOF	AO 07		LDY	≠\$07	
1049	1F11	8C 42 17		STY	SBD	
1050				;		
1051	1F14	09 80		ORA	≠\$80	
1052	1F16	49 FF		EOR	≠\$FF	
1053	1F18	60		RTS		
1054				;		
1055				;		
1056				;		
1057	1F19	AO 00	SCAND	LDY	≠\$00	GET DATA SPECIFIED
1058	1F1B	B1 FA		LDA	(POINTL),Y	BY POINT
1059	1F1D	85 F9		STA	INH	SET UP DISPLAY BUFFER
1060	1F1F	A9 7F	SCANDS	LDA	≠\$7F	CHANGE SEG
1061	1F21	8D 41 17		STA	PADD	TO OUTPUT
1062				;		
1063	1F24	A2 09		LDX	≠\$09	INIT DIGIT NUMBER
1064	1F26	AO 03		LDY	≠\$03	OUTPUT 3 BYTES
1065				;		
1066	1F28	B9 F8 00	SCAND1	LDA	INL,Y	GET BYTE
1067	1F2B	4A		LSR	A	GET MSD
1068	1F2C	4A		LSR	A	
1069	1F2D	4A		LSR	A	
1070	1F2E	4A		LSR	A	
1071	1F2F	20 48 1F		JSR	CONVD	OUTPUT CHAR
1072	1F32	B9 F8 00		LDA	INL,Y	GET BYTE AGAIN
1073	1F35	29 0F		AND	≠\$0F	GET LSD
1074	1F37	20 48 1F		JSR	CONVD	OUTPUT CHAR
1075	1F3A	88		DEY		SET UP FOR NXT BYTE
1076	1F3B	DO EB		BNE	SCAND1	
1077	1F3D	8E 42 17		STX	SBD	ALL DIGITS OFF
1078	1F40	A9 00		LDA	≠\$00	CHANGE SEG
1079	1F42	8D 41 17		STA	PADD	TO INPUTS
1080	1F45	4C FE 1E		JMP	AK	GET ANY KEY
1081				;		
1082				;		
1083				;	CONVERT AND DISPLAY HEX	
1084				;	USED BY SCAND ONLY	
1085	1F48	84 FC	CONVD	STY	TEMP	SAVE Y
1086	1F4A	A8		TAY		USE CHAR AS INDEX
1087	1F4B	B9 E7 1F		LDA	TABLE,Y	LOOK UP CONVERSION
1088	1F4E	AO 00		LDY	≠\$00	TURN OFF SEGMENTS
1089	1F50	8C 40 17		STY	SAD	
1090	1F53	8E 42 17		STX	SBD	OUTPUT DIGIT ENABLE
1091	1F56	8D 40 17		STA	SAD	OUT PUT SEGMENTS
1092				;		
1093	1F59	AO 7F		LDY	≠\$7F	DELAY 500 CYCLES APPROX.
1094	1F5B	88	CONVD1	DEY		
1095	1F5C	DO FD		BNE	CONVD1	
1096				;		
1097	1F5E	E8		INX		GET NEXT DIGIT NUM
1098	1F5F	E8		INX		ADD 2

CARD	LOC	CODE	CARD			
1099	1F60 A4	FC	LDY	TEMP	RESTORE Y	
1100	1F62 60		RTS			
1101			;			
1102			;		SUB TO INCREMENT POINT	
1103			;			
1104	1F63 E6	FA	INCPT	INC	POINTL	
1105	1F65 D0	O2		BNE	INCPT2	
1106	1F67 E6	FB		INC	POINTH	
1107	1F69 60		INCPT2	RTS		
1108			;			
1109			;		GET KEY FROM KEY BOARD	
1110			;		RETURN WITH A=KEY VALUE	
1111			;		A GT. 15 THEN ILLEGAL OR NO KEY	
1112			;			
1113			;			
1114	1F6A A2	21	GETKEY	LDX	≠/\$21	START AT DIGIT 0
1115	1F6C A0	01	GETKE5	LDY	≠/\$01	GET 1 ROW
1116	1F6E 20	O2 1F		JSR	ONEKEY	
1117	1F71 D0	O7		BNE	KEYIN	A=0 NO KEY
1118	1F73 E0	27		CPX	≠/\$27	TEST FOR DIGT 2
1119	1F75 D0	F5		BNE	GETKE5	
1120	1F77 A9	15		LDA	≠/\$15	15=NO KEY
1121	1F79 60			RTS		
1122	1F7A A0	FF	KEYIN	LDY	≠/\$FF	
1123	1F7C 0A		KEYIN1	ASL	A	SHIFT LEFT
1124	1F7D B0	O3		BCS	KEYIN2	UNTIL Y=KEY NUM
1125	1F7F C8			INY		
1126	1F80 10	FA		BPL	KEYIN1	
1127	1F82 8A		KEYIN2	TXA		
1128	1F83 29	OF		AND	≠/\$0F	MASK MSD
1129	1F85 4A			LSR	A	DIV BY 2
1130	1F86 AA			TAX		
1131	1F87 98			TYA		
1132	1F88 10	O3		BPL	KEYIN4	
1133	1F8A 18		KEYIN3	CLC		
1134	1F8B 69	O7		ADC	≠/\$07	MULT (X-1) TIMES A
1135	1F8D CA		KEYIN4	DEX		
1136	1F8E D0	FA		BNE	KEYIN3	
1137	1F90 60			RTS		
1138			;			
1139			;		SUB TO COMPUTE CHECK SUM	
1140			;			
1141	1F91 18		CHK	CLC		
1142	1F92 65	F7		ADC	CHKSUM	
1143	1F94 85	F7		STA	CHKSUM	
1144	1F96 A5	F6		LDA	CHKHI	
1145	1F98 69	00		ADC	≠/\$00	
1146	1F9A 85	F6		STA	CHKHI	
1147	1F9C 60			RTS		
1148			;			
1149			;		GET 2 HEX CHAR'S AND PACK	
1150			;		INTO INL AND INH	

CARD	LOC	CODE	CARD		
1151			;	X PRESERVED	Y RETURNED = 0
1152			;	NON HEX CHAR WILL BE LOADED AS NEAREST HEX EQU	
1153			;		
1154	1F9D 20 5A 1E	GETBYT	JSR	GETCH	
1155	1FA0 20 AC 1F		JSR	PACK	
1156	1FA3 20 5A 1E		JSR	GETCH	
1157	1FA6 20 AC 1F		JSR	PACK	
1158	1FA9 A5 F8		LDA	INL	
1159	1FAB 60		RTS		
1160			;		
1161			;	SHIFT CHAR IN A INTO	
1162			;	INL AND INH	
1163			;		
1164	1FAC C9 30	PACK	CMP	≠/\$30	CHECK FOR HEX
1165	1FAE 30 1B		BMI	UPDAT2	
1166	1FB0 C9 47		CMP	≠/\$47	NOT HEX EXIT
1167	1FB2 10 17		BPL	UPDAT2	
1168	1FB4 C9 40	HEXNUM	CMP	≠/\$40	CONVERT TO HEX
1169	1FB6 30 03		BMI	UPDATE	
1170	1FB8 18	HEXALP	CLC		
1171	1FB9 69 09		ADC	≠/\$09	
1172	1FBB 2A	UPDATE	ROL	A	
1173	1FBC 2A		ROL	A	
1174	1FBD 2A		ROL	A	
1175	1FBE 2A		ROL	A	
1176	1FBF A0 04		LDY	≠/\$04	SHIFT INTO I/O BUFFER
1177	1FC1 2A	UPDAT1	ROL	A	
1178	1FC2 26 F8		ROL	INL	
1179	1FC4 26 F9		ROL	INH	
1180	1FC6 88		DEY		
1181	1FC7 D0 F8		BNE	UPDAT1	
1182	1FC9 A9 00		LDA	≠/\$00	A=0 IF HEX NUM
1183	1FCB 60	UPDAT2	RTS		
1184			;		
1185	1FCC A5 F8	OPEN	LDA	INL	MOVE I/O BUFFER TO POINT
1186	1FCE 85 FA		STA	POINTL	
1187	1FDO A5 F9		LDA	INH	TRANSFER INH-POINTH
1188	1FD2 85 FB		STA	POINTH	
1189	1FD4 60		RTS		
1190			;		
1191			;		
1192			;	END OF SUBROUTINES	

CARD	⋈	LOC	CODE	CARD
1194				;
1195				;
1196				;
1197		1FD5 00	TOP	. BYTE \$00,\$00,\$00,\$00,\$00,\$00,\$0A,\$0D,'MIK'
1197		1FD6 00		
1197		1FD7 00		
1197		1FD8 00		
1197		1FD9 00		
1197		1FDA 00		
1197		1FDB 0A		
1197		1FDC 0D		
1197		1FDD 4D	49 4B	
1198		1FE0 20		. BYTE ' ', \$13, 'RRE', ' ', \$13
1198		1FE1 13		
1198		1FE2 52	52 45	
1198		1FE5 20		
1198		1FE6 13		
1199				;
1200				;
1201				;
1202		1FE7 BF	TABLE	. BYTE \$BF,\$86,\$DB,\$CF,\$E6,\$ED,\$FD,\$87
1202		1FE8 86		
1202		1FE9 DB		
1202		1FEA CF		
1202		1FEB E6		
1202		1FEC ED		
1202		1FED FD		
1202		1FEE 87		
1203				;
1204		1FEF FF		. BYTE \$FF,\$EF,\$F7,\$FC,\$B9,\$DE,\$F9,\$F1
1204		1FF0 EF		
1204		1FF1 F7		
1204		1FF2 FC		
1204		1FF3 B9		
1204		1FF4 DE		
1204		1FF5 F9		
1204		1FF6 F1		

CARD	LOC	CODE	CARD
1206		;	
1207		;	
1208		;	
1209		;	
1210		;	INTERRUPT VECTORS
1211		;	
1212	1FF7		★=\$1FFA
1213	1FFA 1C 1C	NMIENT	.WORD NMIT
1214	1FFC 22 1C	RSTENT	.WORD RST
1215	1FFE 1F 1C	IRQENT	.WORD IRQT
1216			.END

END OF MOS-TECHNOLOGY 650X ASSEMBLY VERSION 4

NUMBER OF ERRORS = 0, NUMBER OF WARNINGS = 0

SYMBOL TABLE

SYMBOL	VALUE	LINE DEFINED		CROSS-REFERENCES						
ACC	OOF3	76	587	851						
ADDR	1CBE	694	687							
ADDRM	1CC8	701	673							
AK	1EFE	1037	1080							
AK1	1FO4	1041	1046							
CHAR	OOF6	90	950	951	952	959	984	994		
CHK	1F91	1141	734	738	741	748	808	814	899	902
CHKH	17E8	97	158	265	289	299	301			
CHKHI	OOF6	82	730	755	782	799	819	1144	1146	
CHKL	17E7	96	156	262	288	297	298			
CHKSUM	OOF7	83	729	758	783	801	821	1142	1143	
CHKT	194C	295	234	237	242	244	256	308		
CHT1	1982	336	344							
CHT2	198E	341	338							
CHT3	1991	342	340							
CLEAR	1C64	645	803	836						
CLKKT	1747	65	★★★★							
CLKRDI	1747	66	355	361	378	384	498	505		
CLKRDT	1746	67	459	471						
CLK1T	1744	62	358	364	381	387	501	508		
CLK64T	1746	64	461	473						
CLK8T	1745	63	★★★★							
CNTH30	17F3	101	613	622	1010	1022				
CNTL30	17F2	100	625	1012	1024					
CONVD	1F48	1085	1071	1074						
CONVD1	1F5B	1094	1095							
CRLF	1E2F	908	640	785	829					
DATA	1CA8	680	★★★★							
DATAM	1CCC	704	675							
DATAM1	1CCE	705	702							
DATAM2	1CDO	706	699							
DATA1	1CBO	686	698							
DATA2	1CC3	697	692							
DEHALF	1EEB	1022	947	956						
DELAY	1ED4	1010	946	953	986	990	997	1003		
DETCPS	1C2A	612	★★★★							
DET1	1C31	615	617							
DET2	1C42	623	621							
DET3	1C3A	619	624							
DE2	1EDD	1013	1018	1027						
DE3	1EE5	1017	1015							
DE4	1EDE	1014	1029							
DUMP	1D42	778	873							
DUMPT	1800	121	★★★★							
DUMPT1	1814	131	134							
DUMPT2	1833	148	177							
DUMPT3	1854	163	166							
DUMPT4	1865	173	152							
DUMPV	1E01	873	867							
DUMPO	1D48	781	826							
DUMPL	1D4E	785	★★★★							

SYMBOL	VALUE	LINE	DEFINED	CROSS-REFERENCES						
DUMP2	1D86	811	817							
DUMP3	1DA6	826	824							
DUMP4	1D7A	805	792							
EAH	17F8	106	151	791						
EAL	17F7	105	149	789						
FEED	1E07	876	861							
FEED1	1E12	882	880							
GETBYT	1F9D	1154	732	736	739	746	754	757	770	
GETCH	1E5A	940	648	725	1154	1156				
GETK	1C8D	667	*****							
GETKEY	1F6A	1114	667							
GETKE5	1F6C	1115	1119							
GET1	1E60	943	945							
GET2	1E6D	948	955							
GET5	1E6A	947	627							
GET6	1E87	962	944							
GOEXEC	1DC8	841	711	865						
GOV	1CD9	711	679							
HEXALP	1FB8	1170	*****							
HEXNUM	1FB4	1168	*****							
HEXOUT	196F	323	314	316						
HEXTA	1E4C	928	922	924						
HEXTA1	1E55	933	931							
HEX1	1978	328	326							
ID	17F9	107	140	224	226					
INCPT	1F63	1104	708	749	815	938				
INCPT2	1F69	1107	1105							
INCVEB	19EA	397	176	258						
INCVE1	19F2	400	398							
INH	00F9	85	647	780	825	1059	1179	1187		
INITS	1E88	966	600	609						
INIT1	1E8C	969	636							
INL	00F8	84	646	779	823	885	1066	1072	1158	1178 1185
INTVEB	1932	281	123	185						
IRQENT	1FFE	1215	*****							
IRQP27	1BFE	519	*****							
IRQT	1C1F	604	1215							
IRQV	17FE	113	604							
KEYIN	1F7A	1122	1117							
KEYIN1	1F7C	1123	1126							
KEYIN2	1F82	1127	1124							
KEYIN3	1F8A	1133	1136							
KEYIN4	1F8D	1135	1132							
LOAD	1CE7	725	727	762	874					
LOADER	1D3E	771	759							
LOADE1	1D3B	770	756							
LOADS	1CEE	728	*****							
LOADT	1873	183	231							
LOADT4	18B5	216	221							
LOADT5	18D7	233	225	228						
LOADT6	18EC	241	230							
LOADT7	18F8	247	239	259						
LOADT8	1915	261	250							
LOADT9	1929	270	252	263	266					

ADC	13
AND	9
ASL	7
BCC	4
BCS	5
BEQ	26
BIT	12
BMI	9
BNE	44
BPL	15
BRK	0
BVC	0
BVS	0
CLC	8
CLD	1
CLI	0
CLV	0
CMP	38
CPX	1
CPY	0
DEC	2
DEX	14
DEY	8
EOR	2
INC	7
INX	5
INY	2
JMP	31
JSR	115
LDA	108
LDX	29
LDY	25
LSR	22
NOP	0
ORA	6
PHA	5
PHP	0
PLA	5
PLP	0
ROL	18
RTI	1
RTS	28
SBC	5
SEC	3
SED	0
SEI	1
STA	81
STX	14
STY	7
TAX	3
TAY	3
TSX	1
TXA	3
TXS	2
TYA	4

\neq SYMBOLS = 204 (LIMIT = 400)
 \neq LINES = 1242 (LIMIT = 1500)
STOP 0

\neq BYTES = 1690 (LIMIT = 4096)
 \neq XREFS = 646 (LIMIT = 900)

MARIO STAVOLTA EDITORE
Via Mazzini, 58 - 33170 PORDENONE

ULTIMISSIME PUBBLICAZIONI

Dott. Giuseppe De Lorenzo

LA TERAPIA CON LE ACQUE MINERALI

Il dott. De Lorenzo ha un'attitudine particolare ed ammirevole a semplificare lo scibile. E poichè, in ambito d'idrologia medica, è tanto competente quanto appassionato, la sa divulgare riunendo l'indispensabile, tralasciando il superfluo.

Il libro, in trattazione sintetica ed agile presenta: le generalità sulle acque minerali e sulle loro applicazioni terapeutiche; le sorgenti minerali italiane; le norme generali per la pratica delle cure termali e le specifiche indicazioni dell'idroterapia per i diversi tipi di patologia.

E' un libro utile non soltanto al medico, bensì anche ai pazienti, che intendono rendersi conto di persona delle più opportune terapie termali, e dei limiti delle stesse.

Presentazione del prof. Enrico Cheli.

Volume di pagg. 80, rilegato in brossura, formato cm. 17 x 25. L. 4.800

Ing. G. Lardinelli

L'ELICOTTERO

Questo libro, unico nel suo genere nel panorama editoriale italiano, descrive in modo molto chiaro ed esauriente l'elicottero nei suoi principi costruttivi ed inoltre i principi dell'aerodinamica, che ne consentono il volo.

Il libro, che si rivolge anzitutto agli studenti degli Istituti Aeronautici, è consigliato anche agli elicotteristi ed a quanti s'interessano di questo straordinario mezzo di volo.

Volume di pagg. 140, con numerosissime illustrazioni nel testo, rilegato in brossura, formato cm. 17 x 25. L. 6.000

Ing. G. Lardinelli

AEROTECNICA

L'aerotecnica, è la scienza che studia l'applicazione dei principi dell'aerodinamica alla costruzione dei mezzi aerei.

Quest'opera dell'ing. G. Lardinelli è frutto di un'esperienza maturata in anni d'insegnamento, e si rivolge in primo luogo agli studenti degli istituti tecnici aeronautici; ma non a loro soltanto, bensì anche a tutti gli appassionati, dai modellisti ai frequentatori di corsi di volo presso i vari aeroclub d'Italia.

L'eccezionale chiarezza del testo, che non va di certo a scapito della serietà scientifica, consente a chiunque abbia volontà di apprendere, di penetrare lo stretto legame esistente tra i principi dell'aerodinamica e la costruzione dei mezzi aerei, dall'aerostato, al velivolo ad elica, all'aviogetto.

3 volumi di pagg. 160 ciascuno, con numerosissime illustrazioni nel testo, rilegati in broccata, L. 14.400 complessive.

LITOGRAFIA ~~igab~~ BASAGLIAPENTA/UD

giugno 1978

Copertina di Ginesio Romano

KIM-1 MODULO MICROCOMPUTER Manual

1