5. SYMBOLIC EDITOR

5.1. INTRODUZIONE

Il "Symbolic Editor" è un programma usato per creare e modificare programmi simbolici (sorgente) per mezzo di semplici comandi dalla tastiera.

L'Editor è disponibile in banda perforata, EPROM, PROM, ROM e occupa approssimativamente 3 K bytes della memoria indirizzabile.

Il suo text buffer ha la capacità di circa 3.000 (decimali) caratteri.

L'Editor è orientato a caratteri e ogni riga del testo è terminata dal codice RETURN.

Tutte le righe nel buffer sono numerate in decimale ed iniziano da 1.

5.2. MODI DI OPERAZIONE

L'Editor ha due modi di operazione, cioè:

- Command Mode
- Text Mode

Nel Command Mode l'input battuto sulla tastiera è interpretato come comando dall'Editor per eseguire operazioni o permettere all'utente di correggere il programma memorizzato nel buffer.

Nel Text mode l'input battuto è interpretato come testo e viene inserito o viene aggiunto ai contenuti del buffer.

5.3. COMANDI DELL'EDITOR

Un comando indica all'Editor di eseguire l'operazione desidera ta.

Ogni comando consiste di una sola lettera che può essere segui ta da uno o due argomenti.

La lettera di comando dice all'Editor cosa fare; gli argomenti specificano per quale riga o righe del testo il comando sarà effettuato.

I comandi all'Editor devono avere uno dei formati specificati in Tabella A.

E indica una lettera di comando.

L'Editor indica all'utente che è in Command Mode stampando il segno (#).

Nota: Eccetto quando specificato altrimenti, ogni comando deve essere terminato dal tasto RETURN.

TABELLA A

Tipo di comando	Formato comando	Significato
Nessun argomento	E	Esegue l'operazione E
Un argomento	n E	Esegue l'operazione E sulla riga n
Due argomenti	m, n E	Esegue l'operazione E dalla riga m alla n compresa.

./..

Gli argomenti m e n, riferiti alle righe numerate in memoria, devono essere positivi e n deve essere più grande di m.

Due argomenti devono essere separati da una virgola e il comando E deve seguire immediatamente l'argomento (o gli argomenti).

Nota: L'Editor considera solo la prima lettera del comando, il resto della riga è ignorato.

Perciò:

```
# L          (RETURN)
# LIST       (RETURN)
# LIST ALL   (RETURN)
```

sono equivalenti.

Dove le opzioni sono valide, il carattere dell'opzione deve seguire il carattere del comando.

Esempio:

W T (RETURN)

Questo comando indica all'Editor di scrivere sull'output device il programma in memoria cambiando ogni TAB's in SPACEs.

L'Editor stamperà messaggi diagnostici ogniqualvolta l'utente richiederà informazioni inesistenti oppure darà un comando non corretto.

5.4. DESIGNAZIONE INPUT/OUTPUT DEVICE

x (RETURN)
INPUT (L, H, C) =
OUTPUT (L, C) =

L'utente indica all'Editor che vuole specificare l'input/output device stampando un asterisco.
L'Editor risponde con:
INPUT (L, H, C) =
e aspetta che l'utente batta il nome del dispositivo (seguito da RETURN) e poi stampa:
OUTPUT (L, C) =
aspetta il nome del dispositivo (seguito da RETURN) e ritorna in "Command Mode".
L = Low speed reader (TTY)
H = High speed reader (HSR)
C = Cassetta magnetica
Qualsiasi altro carattere sarà interpretato dall'Editor come L (TTY).
Default è L (TTY)

La designazione I/O può essere data o cambiata in qualsiasi momento.
Quando l'Editor inizia, entra nel modo di designazione I/O.

./..

Nota sulla cassetta output

L'utente usando la cassetta come output device, prima di dare il comando (RETURN) procede in questo modo:

1. Accende il registratore
2. Riavvolge la cassetta magnetica alla posizione di inizio
3. Preme i bottoni del registratore (REC) e START
4. Da' il comando (RETURN)

Il nastro incomincia a girare e l'Editor esegue una "routine" che posiziona il nastro al suo livello giusto, ferma il nastro e ritorna in "Command Mode".

Nota sulla banda perforata in uscita

Se come device di uscita è stato indicato il perforatore TTY, l'utente deve ricordare che la tastiera stampante e la banda perforata lavorano in parallelo; quindi il perforatore deve essere sulla posizione OFF fino a quando l'utente è pronto a dare il comando di perforazione.
Quando l'utente dà uno dei comandi che trasferiscono il testo all'output device (W, N, Q, E) o il comando LEADER/TRAILER (T), l'Editor aspetta che l'utente metta in "ON" il perforatore prima di eseguire il comando.
L'utente comunica all'Editor quanto sopra premendo un tasto qualunque sulla tastiera.
Dopo aver eseguito il comando, l'Editor aspetta ancora che l'utente metta in "OFF" il perforatore prima di tornare in "Command Mode".
Questo è anche comunicato all'Editor premendo un tasto sulla tastiera (Le eccezioni sono: comandi "Q" e "E").

5.5. COMANDI DEL TEXT INPUT

5.5.1. Comando APPEND

A (RETURN)

Questo comando permette di scrivere un nuovo testo tramite tastiera oppure di aggiungerlo a quello esistente.
L'Editor entra in "text mode" dopo avere ricevuto il comando e l'utente può scrivere qualsiasi numero di righe di testo.
Il nuovo "text" sarà aggiunto all'informazione già nel "buffer" fino a quando l'utente desidera terminare il "text mode" premendo il tasto ESCAPE o ALT.

Se questo tasto è premuto alla fine della riga, l'Editor ritorna in "Command Mode" ignorando quella riga.
Battendo la combinazione del tasto CTRL/U, mentre si scrive una riga del text, si cancella tutta la riga e l'utente può scriverne una nuova.

Qualsiasi RUBOUT, mentre si è in "text mode", cancella l'ultimo carattere stampato.

Battendo più volte il tasto "RUBOUT" si cancelleranno tanti caratteri (da destra a sinistra) quante saranno le battute fino a raggiungere l'inizio della riga.

5.5.2. COMANDO READ

R o READ(RETURN)

L'Editor leggerà l'informazione dall'input device scelto finchè sarà riconosciuto il codice FORM FEED (combinazione tasto CTRL/L) oppure fino a quando l'Editor rivelerà la condizione di riempimento del buffer.
Tutto il testo di entrata, tranne il FORM FEED, è aggiunto ai contenuti del text buffer.
Durante un comando "READ" (R) i RUBOUTS, NULLS e LINE FEEDS che si incontrano vengono ignorati dall'Editor.
Nel caso di input dal lettore veloce o dalla cassetta magnetica l'Editor rimarrà in un ciclo senza fine se alla fine del testo di entrata il codice FORM FEED è mancante.
(L'utente deve assicurarsi di stampare l'input text usando un Editor che aggiunga il codice FORM FEED indicante la fine dell'input).
Nel caso di un lettore lento (telescrivente), se la banda perforata non ha alla fine il codice FORM FEED, l'utente può caricare un codice FORM FEED tramite la tastiera, facendo ritornare l'Editor in "Command Mode".
Se questo non è fatto, il comando "READ" è ancora valido e tutti i comandi successivi saranno interpretati erroneamente come testo e caricati su quanto già letto dal nastro.
Durante un comando "READ" l'Editor rivela una condizione di buffer pieno quando l'ultima riga caricata lascia lo spazio per circa 100 caratteri che possono essere caricati o inseriti nel testo finchè la condizione di buffer pieno è raggiunta.

./..

L'Editor stampa:

XXXX LINES READ IN:

e ritorna in "Command Mode"

XXXX è il numero delle righe nel buffer.

A questo punto, ogni successivo comando READ, farà ristampare dall'Editor il sopra indicato messaggio.

5.5.3. EDITING COMMANDS

I seguenti comandi permettono cancellazioni, alterazioni o espansioni del text nel buffer.

5.5.4. COMANDO KILL

K (RETURN)

KILL ?

Il comando KILL cancella l'intero testo dal buffer.

Dopo il comando "K" l'Editor stampa il messaggio "KILL ?" prima di eseguire il comando.

Questo per evitare un "KILL" casuale del buffer.

Se l'utente risponde con il carattere "Y" l'Editor esegue il comando e ritorna in "Command Mode".

Qualsiasi altro carattere fa in modo che l'Editor ignori il comando (K) e ritorni in "Command Mode".

5.5.5. COMANDO DELETE

nD (RETURN)

Cancella la riga n. La riga è rimossa dal text buffer e il numero di tutte le righe successive e del contatore delle righe viene diminuito di una unità.

m, n D (RETURN)

Cancella dalla riga m alla n inclusa.

Il valore del contatore della riga corrente (.) è uguale al numero della riga che segue l'ultima riga cancellata.

L'Editor rimane in "Command Mode" dopo un'operazione di "DELETE".

D (RETURN)

Questo comando è uguale a:

. D (RETURN)

e cancella la riga corrente

5.5.6. COMANDO INSERT # n I [INSERT] (RETURN)

Inserisce un nuovo testo prima della riga n, fino a quando si incontra un comando di ESCAPE o ALT.

Con questo comando l'Editor entra in "text mode" per accettare l'input e funziona come il comando "APPEND" mentre inserisce il testo.

La prima riga scritta diventa la nuova riga n.

Sia il contatore delle righe che il numero di tutte le righe che seguono l'inserzione sono aumentate del numero delle righe inserite.

Il valore del punto (.) contatore della riga corrente- è uguale all'ultima riga inserita.

Se il numero della riga manca, l'Editor inserisce sopra la riga corrente.

./..

5.5.7. COMANDO OVERLAY

#n O[VERLAY] (RETURN)

Con questo comando l'Editor sostituisce la riga n (se manca n l'Editor assume come argomento la riga corrente) con tante righe quante sono quelle scritte dall'utente.

Questo comando è equivalente a:

#n D

#n I

Nel caso che n coincida con l'ultima riga l'Editor la cancella ed aggiunge quanto scritto.

.5.8. COMANDO CHANGE

#n C[HANGE] (RETURN)

OLD STRING (RETURN)

NEW STRING (RETURN)

NEW LINE

Con questo comando l'Editor ricerca sulla riga n (default riga corrente) la combinazione dell'OLD STRING e stampa la nuova riga.

Se OLD STRING non viene trovato sulla riga, l'Editor stampa NO MATCH e ritorna in "Command Mode".

Se non viene dato "NEW STRING" (l'utente stampa RETURN), OLD STRING è cancellato dalla riga.

Se non viene dato "OLD STRING" (l'utente stampa RETURN), NEW STRING è inserito all'inizio della riga.

./..

Esempio:

#L (RETURN)

HAPPY DAYS WERE HERE ONCE

#C (RETURN)

WERE HERE ONCE (RETURN)

ARE HERE AGAIN (RETURN)

HAPPY DAYS ARE HERE AGAIN

La riga cambiata è stampata dall'Editor.

5.5.9. CARATTERI STRING SEARCH

#S[EARCH] (RETURN)

STRING (RETURN)

L'Editor esamina ogni riga del buffer per ricercare la combinazione dei caratteri "STRING" e,se la trova, stampa la riga e ritorna in "Command Mode".

Se non trova "STRING", l'Editor stampa "NO MATCH" e ritorna in "Command Mode".

Una volta che "STRING" è trovato, il contatore della riga corrente (.) è uguale alla riga dove si trova "STRING".

./..

5.5.10. BUFFER LOCATE

B (RETURN)
STRING (RETURN)

Con questo comando l'Editor ricerca per l'intero input text il buffer che contiene "STRING".

Quando l'Editor trova "STRING" stampa la riga che la contiene e ritorna in "Command Mode". Nell'eseguire questo comando l'Editor nel proprio "buffer" incomincia a cercare "STRING"e se non la trova, trasferisce il "buffer" sull'output device, cancella il "buffer", legge il nuovo testo dall'input device e cerca ancora "STRING".

Questo si ripete. fino a quando la sequenza viene trovata oppure fino a quando viene incontrato END OF TEXT (FORM FEED).

L'ultimo buffer (o quello dove si trova "STRING") non è trasferito sull'output device.

L'Editor stampa "NO MATCH" se non trova "STRING".

5.6. COMANDI DI TEXT OUTPUT

5.6.1. Comando LIST

L[IST] (RETURN)

Scrive l'intero buffer sul terminale

n L[IST] (RETURN)

Scrive la riga n del text buffer sul terminale

m, n L[IST] (RETURN)

Scrive dalla riga m a quella n del text buffer sul terminale.

Il comando LIST aggiorna il contatore della riga corrente (.)

Dopo aver eseguito il comando, l'Editor ritorna in "Command Mode".

5.6.2. COMANDO WRITE

W[RITE] (RETURN)

Scrive l'intero text buffer sull'output device.

n W[RITE] (RETURN)

Scrive la riga n del text buffer sull'output device.

m, n W[RITE] (RETURN)

Scrive dalla riga m a quella n compresa, sull'output device.

Il comando WRITE aggiorna il contatore della riga corrente (.)

Dopo aver eseguito il comando, l'Editor ritorna in "Command Mode".

5.6.3. COMANDO NEXT

N[EXT] (RETURN)

L'Editor scrive il buffer corrente sull'output device, cancella il buffer, poi legge del nuovo testo dall'input device e ritorna in "Command Mode".

5.6.4. COMANDO QUIT

Q[UIT] (RETURN)

L'Editor aggiunge un codice END OF TEXT (FORM FEED) al text buffer, scrive il buffer all'output device, e ritorna in MONITOR (M0-Z).

5.6.5. COMANDO EXIT

E[XIT] (RETURN)

L'Editor scrive il buffer corrente sull'output device e trasferisce tutto il testo dall'input all'output device e ritorna in MONITOR (M0-Z)

Nota: Il comando READ deve essere dato almeno una volta prima che il comando EXIT possa funzionare; altrimenti, il comando EXIT si comporta come un comando QUIT.
I comandi NEXT e EXIT possono avere uno o due argomenti e il comando è eseguito come un comando "WRITE" con argomenti.
Nel caso di comando EXIT gli argomenti sono validi solo per il buffer corrente.

./..

5.6.6. OPZIONE "T"

Con l'opzione "T" dopo un comando di text output (W, N, Q, E) l'Editor converte i TAB's in spazi.

L'opzione "T" deve essere data subito dopo la prima lettera del comando.

Esempio: # W T (RETURN)

5.6.7. LEADER/TRAILER

T[RAILER] (RETURN)

In esecuzione di questo comando l'Editor fa uscire circa 50 "frames" di nastro vuoto (valido solo sull'output della banda perforata).

Nota: Per tutti i comandi di text output (inclusa T) vedere le note sulla banda perforata e la cassetta output (punto 5.4.).

5.7. FUNZIONI E COMANDI TASTI SPECIALI

5.7.1. TASTO RETURN

Sia in "Command Mode" che in "Text Mode", battendo il tasto "RETURN" si segnala all'Editor di processare l'informazione già stampata.

In "Command Mode" l'Editor esegue il comando.

Un comando non sarà eseguito fino a quando non sarà battuto il tasto "RETURN" (con eccezione di "LINE FEED" e segno di minore "<" spiegato più avanti).

In "text mode","RETURN" fa in modo che la riga di testo sia caricata nel "text buffer".

5.7.2. ERASE (CTRL/U)

Il carattere ERASE (combinazione CTRL/U) è usato per rimediare ad errori compiuti in "Command Mode" e "Text Mode".
Lo si genera tenendo premuto il tasto CTRL mentre si batte una U.

Se usato in "Command Mode" o in "Text Mode", CTRL/U cancella la riga, esegue un ritorno ed avanzamento carrello (CR/LF): a questo punto l'utente può ristampare la nuova riga.

5.7.3. TASTO RUBOUT O DELETE

RUBOUT (DELETE) è usato per correggere errori sia in "Command Mode" che in "Text Mode"(durante un comando di READ questi due codici sono ignorati dall'Editor).

In qualsiasi momento, battendo un RUBOUT (DELETE),l'Editor stampa una barra (\) seguita dall'ultimo carattere e cancella il carattere stampato.
RUBOUTs (DELETE) ripetuti cancellano caratteri contigui da sinistra a destra fino all'inizio della riga.

5.7.4. TASTO ALT MODE O ESCAPE

Uno qualsiasi di questi comandi fa ritornare l'Editor in "Command Mode".
In "Command Mode" battendo "ALT MODE" si segnala all'Editor di ignorare il comando e di aspettarne uno nuovo.

./..

Nel caso che l'Editor si trovi in Text Mode, il tasto ESCAPE fa ritornare in "Command Mode" ignorando qualsiasi testo scritto prima del tasto ALT Mode o del tasto ESCape.
Dopo i comandi LIST e WRITE il tasto ESCAPE interrompe l'esecuzione del comando precedente ritornando in "Command Mode" (il contatore della riga corrente (.) non è aggiornato).

5.7.5. CONTATORE DELLA RIGA CORRENTE

L'Editor tiene in evidenza il numero decimale della riga sulla quale sta operando.

In qualsiasi momento (mentre si è in "Command Mode"), il punto (.) indica il numero della riga corrente e può essere usato come un argomento.

Esempio:

#. L (RETURN)
significa stampare la riga corrente

1. Dopo un comando READ o APPEND, il punto è uguale all'ultima riga nel buffer.
2. Dopo un comando INSERT, CHANGE o OVERLAY, il punto è uguale all'ultima riga entrata.
3. Dopo un comando LIST il punto è uguale all'ultima riga del buffer.
4. Dopo un comando SEARCH, il punto è uguale alla riga dove c'è stato il MATCH.

5. Dopo un comando DELETE il punto è uguale alla riga successiva a quella cancellata.
6. Dopo un comando "KILL" il punto è uguale a zero.

5.7.6. SLASH (/)

Il segno di slash (/) ha in ogni momento valore dell'ultima riga del buffer e può essere usato come argomento.
Esempio:

/ L (RETURN)
L'Editor elenca l'ultima riga del buffer.

5.7.7. TASTO LINE FEED

Battendo il tasto "LINE FEED" in risposta al simbolo "#" (che segnala all'utente di essere in "Command Mode") si fa in modo che l'Editor stampi la riga successiva alla riga corrente ed incrementi il valore del contatore della riga corrente (.).
In tal modo il contenuto di tale contatore individua il numero della riga appena stampato.
Battendo il tasto "LINE FEED" mentre si sta completando un comando o una riga di testo si fa in modo che l'Editor ristampi la riga: questo permette all'utente di verificare la correttezza della riga dopo l'uso di uno o più "RUBOUTS".

5.7.8. TASTO CON SEGNO DI MINORE (<)

Battendo questo tasto in risposta al simbolo (#) si fa in modo che l'Editor stampi la riga che precede quella corrente.

Il valore del contatore della riga corrente viene quindi diminuito di uno in modo da riferirsi alla riga stampata.

5.7.9. SEGNO UGUALE (=)

Il segno uguale è usato insieme al punto e allo slash (/).

Quando viene battuto dopo il punto o lo slash, fa in modo che l'Editor stampi rispettivamente il valore della riga corrente o dell'ultima riga del buffer.
Esempio:

```
# . = (RETURN)
XXXX
# / = (RETURN)
YYYY
```

5.7.10. CTRL/BELL

Se viene dato dopo la ricerca di una "STRING" fa in modo che l'Editor cerchi la successiva ricomparsa della stessa "STRING" sul buffer corrente.
Esempio:

```
# S (RETURN)
HL (RETURN)
LD HL; BUFF
# CTRL/BELL
LD A, (HL)
# CTRL/BELL
NO MATCH
```

./..

5.7.11. TABULAZIONE (CTRL/TAB)

L'Editor è scritto in modo tale da simulare la tabulazione orizzontale con intervalli di 8 spazi.

Nel caso che l'utente tenga premuto il tasto CTRL e batta il tasto TAB (o HT), l'Editor fa una tabulazione.

La tabulazione porta il carrello al punto di tabulazione creando da 1 a 8 spazi secondo necessità.

Nel text buffer, l'Editor memorizza solo il codice TAB e sull'output scrive il codice del TABs, a meno che venga data l'opzione "T".

5.7.12. CTRL/C

Con la combinazione del tasto "CTRL", "C", l'Editor ritorna all'M0-Z(MONITOR).

5.8. MESSAGGI DIAGNOSTICI

MESSAGGIO	AVVENIMENTO
SYNTAX or LINE ERROR	- L'utente scrive un comando illegale. - Gli argomenti di un comando non sono validi. - L'utente richiede informazioni inesistenti (es. elenca una riga inesistente). - In "Command Mode" o in "Text Mode" la riga caricata ha più di 80 caratteri, o la riga è nulla. - Durante un comando "READ", la riga è più lunga degli 80 caratteri consentiti o la riga è nulla.
KILL ?	- Dopo un comando KILL se l'utente stampa "Y" (e RETURN), l'Editor cancella il buffer e ritorna in "Command Mode". Qualsiasi altro carattere farà ritornare l'Editor in "Command Mode". Il buffer rimane intatto.
NO MATCH	- Dopo un comando SEARCH o CHANGE se non avviene nessun MATCH della STRING.

./..

BUFFER FULL	- L'Editor non può più accettare altro testo. L'utente deve quindi scrivere tutto o parte del buffer sull'output device e poi liberare spazio del buffer usando il comando KILL o DELETE.
BUFFER EMPTY	- Il buffer non contiene nessun testo quando l'utente prova delle operazioni riguardante il testo (cioè L,I,C,O ecc).
XXXX LINES READ IN	- Avviene dopo un comando READ. XXXX è il numero totale delle righe nel buffer.

Dopo aver scritto qualsiasi messaggio diagnostico l'Editor ritorna in "Command Mode".

Cap. 6-1 - BINARY DUMP

INDICE

		PAG.
6.	Binary Dump	6. - 1
6.1.	Uso del Binary Dump	6. - 1÷3

BINARY DUMP

6. INTRODUZIONE

Lo scopo del BINARY DUMP è quello di perforare su nastro di carta o registrare su cassetta magnetica un programma binario che si trova in memoria.

Il nastro che viene creato da questo programma è dello stesso formato di quello che si ottiene con l'ASSEMBLER perciò può essere caricato con il LOADER.

Questo programma può risiedere su memorie non volatili (ROM - PROM - EPROM) oppure su RAM.

6.1. USO DEL BINARY DUMP

Alla partenza il programma pone una serie di domande.
La prima riguarda la periferica sulla quale si vuole ottenere il binario.
Il calcolatore stampa sulla teletype:
OUT (L/C)
ed attende come risposta una lettera che può essere:

C per avere il binario su cassetta
L per avere il binario su nastro di carta

Qualsiasi altro carattere viene interpretato come L.
A questo punto il programma inizia la perforazione del trailer che precede ogni binario.
Alla fine il calcolatore si ferma ed attende che si prema un tasto

qualsiasi della tastiera per proseguire.

Questo permette all'operatore di disabilitare il perforatore del nastro per non rovinare il binario con le successive domande poste dal calcolatore sulla Teletype.

Queste domande sono:

STR:

END:

Le risposte devono essere l'indirizzo di inizio e di fine del binario da perforare (in esadecimale) seguiti dal RETURN.

Dopo aver risposto alla domanda END, il calcolatore aspetta che si prema un tasto qualsiasi della tastiera per permettere all'operatore di abilitare il perforatore del nastro, dopo di che inizia la perforazione del binario.

Alla fine di questo il calcolatore attende un carattere qualsiasi della tastiera per proseguire; poi ripropone le domande STR: e END per la perforazione di un eventuale altro blocco, non contiguo con quello già perforato.

Se alla domanda END, si risponde con il carattere X seguito dal RETURN, il calcolatore prende l'indirizzo dato come risposta a STR: ,come l'indirizzo di AUTO-START e dopo aver atteso un carattere dalla tastiera perfora il blocchetto contenente l'autostart, il trailer finale e cede il controllo all M0-Z.

Se alla domanda STR: si risponde con il carattere X viene perforato il trailer finale dopo di che si cede il controllo all'M0-Z.

Nota: Se si seleziona come periferica di uscita il registratore a cassetta, il programma non attende i caratteri per permettere l'accensione e lo spegnimento del perforatore lento, perchè non c'è pericolo di rovinare il nastro magnetico con caratteri indesiderati.

Il registratore usato è lo stesso usato dall'EDITOR e dall'ASSEMBLER per scrivere.

Cap. 7-1 - INPUT/OUTPUT PORTS

INDICE

		PAG.
7.	Input/Output Ports	7. -1
7.1.	Interfaccia seriale (USART)	7.1. -1÷2
	7.1.1. Programmazione dell'USART	-3÷4
7.2.	Interfaccia parallelo (PIO)	7.2. -1
	7.2.1. Indirizzi di porta	-1
	7.2.2. PIO Interrupt	-2
	7.2.3. Catena della priorità del PIO Interrupt	-3
	7.2.4. Sequenza caricamento vettore	-4÷5
7.3.	Z80 Interrupts	7.3. -1
	Modi di Interrupts	
	7.3.1. Non - Maskable	-1
	7.3.2. Maskable	-1÷2
	7.3.3. Servizio di interruzione	-3

7. INPUT / OUTPUT PORTS

La scheda del microcomputer è provvista di una interfaccia seriale (USART) e di due interfacce parallelo (PIO).

Il software residente inizializza quei dispositivi e li configura per le operazioni di sviluppo del software.

L'utente può inizializzare di nuovo le interfacce e configurarle per soddisfare le sue esigenze.

7.1. INTERFACCIA SERIALE (USART)

L'interfaccia seriale I/O usata è l'INTEL 8251 USART.

L'USART è programmabile mediante un byte di controllo caricato dal CPU.

I parametri controllabili sono:

- Bits per carattere.
- Numero di bits di stop per operazione asincrona.
- Controllo di parità (dispari, pari, nessuna).
- Trasmissione o ricezione dati a frequenze 1, 16, 64 volte il "Baud rate".

L'USART accetta i caratteri di dati dal CPU in formato parallelo e li converte nel formato di trasmissione.

Può simultaneamente ricevere dati seriali e convertirli in caratteri di dati paralleli per lo Z80 CPU.

Gli indirizzi di porta USART usati nella scheda sono:

INDIRIZZO	FUNZIONE
ØØ11	Input / Output Data
Ø111	Stato
	Bit Ø - Output Flag
	Bit 1 - Input Flag
	Bit 1 - Avanzamento lettore di nastro

7.1.1. Programmazione dell'USART

All'accensione (o quando viene dato un nuovo RESTART) si forma un impulso di RESET e l'M0-Z software inizializza l'USART per stabilire il modo e il formato dei dati da trasferire.

Per le operazioni con TTY, l'USART viene configurato per operare in modo asincrono con un carattere lungo 8 bits, con un fattore di "Baud rate" di 16 X e 2 bits di stop inseriti dopo il carattere trasferito.

Il programma usato dall'M0-Z software per inizializzare l'USART (l'utente potrebbe usarne uno simile per configurare l'USART secondo le sue necessità) è il seguente:

```
; USART INIT ROUTINE
;
INIT:  XOR A; CLEAR ACC.
       OUT (Ø1H), A ; OUTPUT THREE
       OUT (Ø1H), A ; TIMES TO MAKE SURE
       OUT (Ø1H), A ; THE USART IS IN
                    ; A KNOWN STATE
       LD A, 4ØH    ; RESET BIT
       OUT (Ø1H), A ; RESET USART
       LD A, CEH    ; SET MODE
       OUT (Ø1H), A ; CE IMPLIES
                    ; ASYN MODE (x 16)
                    ; 8 DATA BITS
                    ; 2 STOP BITS
       LD A, Ø5H    ; ENABLE
       OUT (Ø1H), A ; TRANSMIT/RECEIVE
```

I programmi READ/WRITE usati dall'M0-Z sono:

```
; TTY READ ROUTINE
; RETURN WITH BYTE IN ACCUMULATOR
```



```
READ:  IN A, (Ø1H)   ; CHECK STATUS
       BIT 1, A      ; FLAG BIT 1 = 1 ?
       JR Z, READ    ; NO WAIT
       IN A, (ØØH)   ; LOAD ACC. WITH
       RET           ; DATA AND RETURN
; TTY WRITE ROUTINE
; CALL WITH BYTE IN ACC.

WRITE: OUT (ØØH), A ; LOAD DATA REGIS.
                    ; TER WITH ACC.
       NOP          ; DELAY REQUIRED
       NOP          ; BY THE USART

FLAGW: IN A, (Ø1H)   ; CHECK STATUS
       BIT Ø, A      ; BIT Ø = 1 ??
       JR Z, FLAGW   ; NO WAIT
       RET           ; YES, RETURN
```

Per ulteriori informazioni consultare la documentazione sull'USART 8251.

7.2. INTERFACCIA PARALLELO (PIO)

La scheda è provvista di 2 interfacce parallelo (PIO) di 2 porte ciascuna per l'interfacciamento dello Z8Ø con le periferiche.

Il PIO contiene 2 porte indipendenti di 8 bits che possono essere configurate dal CPU ad operare in 4 modi.

Nell'Output Mode (MODE Ø) i dati sono scritti dallo Z8Ø sul "data bus" delle porte.

Nell'Input Mode (MODE 1) il dispositivo periferico fornisce i dati alla porta accessibile al CPU.

Il Modo Bidirezionale (MODE 2) permette che una porta (PORT A) sia bidirezionale usando i segnali di "handshake" dell'altra porta (PORT B).

Il Control Mode (MODE 3) permette il controllo diretto dei bits delle porte.
Questo Modo permette a qualsiasi bit di essere individualmente programmato come Bit di ingresso o Bit di uscita.

7.2.1. Indirizzi delle porte

Ogni porta del PIO ha 2 indirizzi, uno per il controllo e uno per i dati.
Gli indirizzi di porta per i 2 PIO sulla scheda possono essere così riassunti:

	PIO N° 1		PIO N° 2	
	PORT A	PORT B	PORT A	PORT B
DATA	Ø4	Ø5	Ø8	Ø9
CONTROL	Ø6	Ø7	ØA	ØB

7.2.2. PIO Interrupt

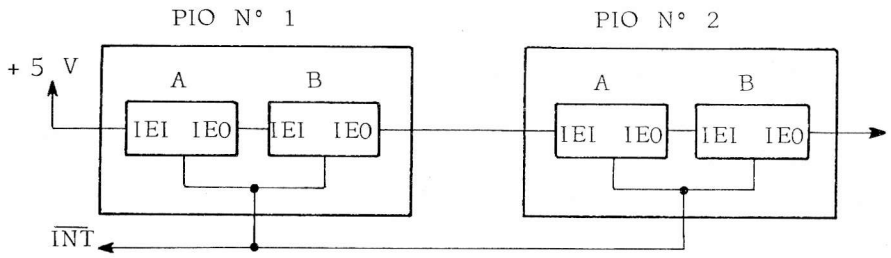
L'interfaccia PIO è disegnata per usare l'interrupt MODE 2 del CPU Z8Ø.
Questo è il più potente dei 3 modi di interrupt perchè permette una chiamata indiretta alla locazione individuata dal vettore di 8 bits fornito dalla periferica.

In questo modo (MODE 2) la periferica genera l'interrupt e mette il vettore sul "data bus" in risposta ad un riconoscimento di interrupt fornito dal CPU.
Il CPU accetta questo vettore e lo interpreta come la parte meno significativa (la più significativa è fornita dal contenuto del registro I) di un "pointer" che individua la locazione di memoria in cui è contenuto l'indirizzo di partenza del programma di servizio dell'interrupt. (vedi par. 7.2.4.).

La priorità dell'interrupt è determinata collegando i dispositivi (PIO) in modo seriale.

Due linee (Interrupt Enable In - IEI- e Interrupt Enable Out - IEO) sono previste in ogni periferica per una configurazione a "daisy chain".

7.2.3. Catena delle Priorità delle Interrupt PIO



La linea IEI del PIO n. 1 è collegata a + 5 Volts per indicare che ha la più alta priorità.

Internamente la Porta A di un PIO ha sempre priorità più alta della Porta B

Un dispositivo, per generare un interrupt, deve avere la sua linea IEI ad un livello alto.

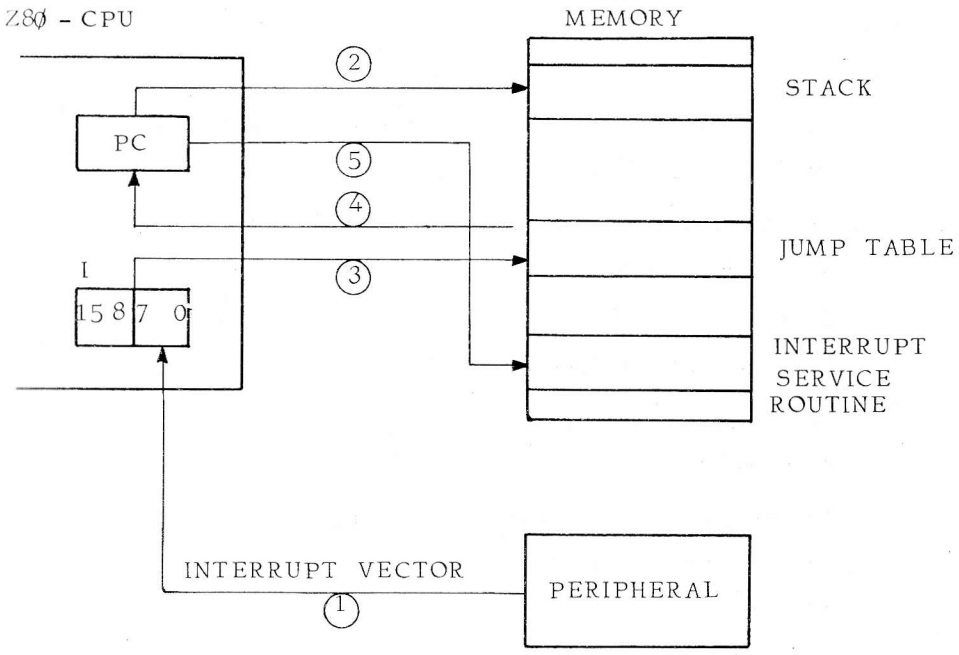
Con questa struttura di interrupt è possibile un "nesting" ad ogni livello.

L'istruzione di ritorno da un interrupt (RETI) facilita il "nesting" permettendo a periferiche con priorità più alta di sospendere temporaneamente le routines di servizio di periferiche con priorità minore.

Dopo il completamento del programma di servizio dell'interrupt,

l'istruzione RETI è usata per restituire i contenuti del Program Counter (POP the stack) e sistemare la logica interna del dispositivo con priorità più alta.

7.2.4. Sequenza Caricamento Vettore



1. Con interruzioni abilitate (EI), la periferica genera una richiesta di interrupt portando la linea $\overline{INT}$ ad un livello basso e mette il vettore di interrupt sul "data bus" quando il CPU riconosce la richiesta di interrupt.

2. Lo Z8Ø - CPU termina l'esecuzione della istruzione corrente e risponde con un riconoscimento di interrupt e legge il vettore dell'interrupt.
   I contenuti del Program Counter (PC) sono messi nello STACK esterno.

   Il flip-flop dell'abilitazione dell'interrupt è risettato; in una struttura con priorità di interruzione i programmi di servizio con priorità più bassa devono permettere, abilitando nuovamente (EI) l'interrupt, a periferiche con priorità più alta di essere serviti prima.

3. Il CPU, dal "pointer" formato dal registro I e dal Vettore, individua la locazione di memoria in cui è contenuto l'indirizzo di partenza del programma di servizio dell'interrupt, e

4. Carica il Program Counter con l'indirizzo del programma di servizio dell'interrupt e perciò:

5. Esegue un CALL a quella locazione.

   N.B. - Tutti i programmi di servizio dell'interrupt devono ritornare al programma principale con l'istruzione di ritorno dall'interrupt (RETI)

7.3. Z8Ø INTERRUPT

Lo scopo di un interrupt è quello di permettere a dispositivi periferici di sospendere, sotto controllo del software, le operazioni del CPU e di forzare il CPU ad iniziare un programma di servizio per la periferica.

Di solito questi programmi di servizio scambiano dati o informazioni tra il CPU e la periferica.

Una volta che il programma di servizio è completato, il CPU ritorna a quelle operazioni la cui esecuzione era stata sospesa.

MODI DI INTERRUZIONE

7.3.1. Non - Maskable

Una interruzione "nonmaskable" sarà sempre accettata dal CPU.
Quando questo avviene, il CPU termina l'esecuzione dell'istruzione corrente e esegue un CALL alla locazione 66 H.
L'istruzione di ritorno del programma di servizio di un nonmaskable interrupt è RETN.

7.3.2. Maskable

Mode Ø

In questo modo la periferica che richiede l'interrupt può mettere sul "data bus" qualsiasi istruzione che il CPU eseguirà.

Ciò significa che è la periferica e non la memoria a fornire la prossima istruzione da eseguire.
Tipicamente questa è un'istruzione RESTART.

Mode 1

Quando è stato selezionato questo modo, il CPU risponde ad un interrupt eseguendo un RESTART alla locazione ØØ38H.

Mode 2

Questo modo è il più potente modo di interrupt.
Con un singolo byte di 8 bits fornito dall'utente si può fare un CALL indiretto a qualsiasi locazione di memoria.
Con questo modo il programmatore forma una tabella di indirizzi di 16 bits che individua la partenza di tutti i programma di servizio dell'interrupt.
Questa tabella può essere localizzata in memoria dovunque.
Quando un interrupt è accettato, si deve formare un "pointer" a 16 bits per ottenere l'indirizzo di partenza della "routine" di servizio dell'interrupt dalla tabella.

Gli 8 bits più significativi del "pointer" sono formati dal contenuto del registro I.

Gli 8 bits meno significativi devono essere forniti dalla periferica che richiede l'interruzione.

7.3.3. Servizio dell'interrupt

Ad una predeterminata condizione, una periferica genera una condizione per interrompere il CPU.

Durante questo tempo la linea di interrupt ($\overline{INT}$) viene forzata verso il basso dalla periferica che richiede l'interruzione.

Il CPU finirà l'esecuzione della istruzione corrente, e manderà un segnale di riconoscimento dell'interrupt ($\overline{INTA} = \overline{IORQ} \cdot \overline{M1}$).

Durante $\overline{INTA}$ la logica dell'interrupt della periferica determinerà la porta con la priorità più alta la quale sta richiedendo un interrupt.

Questo dispositivo mette poi il contenuto del suo vettore di interrupt (8 bits) sul data bus.

La condizione di interrupt è mantenuta fino alla fine del ciclo $\overline{INTA}$.

Le periferiche con priorità più basse sono inibite fino a quando la periferica con priorità più alta decodifica una istruzione RETI.

COD. ORD.: CLZ80 - MAN - ISTR

L. 6.000
(IVA compresa)