i trucchi del CP/M

di Pierluigi Panunzi

Le funzioni del BDOS

Continuiamo in questo numero lo studio delle varie funzioni del BDOS, studio iniziato due numeri fa e interrotto lo scorso numero da un "ritorno" dell'MBasic.

Prima di proseguire ricordiamo, rimandando per i dettagli al nº 37 di MC, che il "numero della funzione", così come appare nel titolo di ogni paragrafo, è il valore da porre nel registro C prima di effettuare la chiamata alla locazione 5 (CALL 005).

Funzione 3: Read "Reader" Byte

È una funzione particolare del BDOS in quanto consente di leggere un carattere dal dispositivo logico "Reader" (RDR:).

In pratica, bisogna vedere qual è l'effettivo dispositivo fisico "connesso" al CP/M con l'attributo "RDR:"; infatti, mentre le prime macchine utilizzanti il CP/M erano dotate di un dispositivo di lettura/scrittura di nastro perforato, per cui aveva senso leggere un byte da tale supporto fisico, oggigiorno nessun personal è dotato di tale arcaico dispositivo di lettura/scrittura. Questo fatto comporta che, a livello BIOS, tale funzione è implementata o meno a seconda delle scelte del costruttore e a seconda di eventuali dispositivi prescelti.

Infatti se il dispositivo "Reader" non è in alcun modo implementato, allora la routine in questione in generale risponde come se avesse letto il byte esadecimale 1AH, usato nel CP/M come indicatore di "End of file".

Viceversa si potrebbe usare il dispositivo logico "Reader" come mezzo con cui utilizzare ad esempio un joystick, oppure un canale di ricezione parallelo ad 8 bit, eccetera, ma il tutto richiede una buona dose di esperienza sia a livello hardware, per la costruzione di un'adatta interfaccia, sia a livello software in quanto questa routine attende, sì, un carattere dal dispositivo, ma fatalmente inchioderebbe tutto il sistema se tali caratteri non arrivassero più: questo perché la routine "non" aspetta un certo numero di caratteri, "né" si ferma alla ricezione di un certo carattere.

Funzione 4: Write "Punch" Byte

Questa funzione è analoga alla precedente, solo che tratta l'output di un certo carattere al dispositivo "PUN:".

Anche in questo caso si tratta di una "reliquia" dei primi sistemi su cui operava il CP/M, dotati per l'appunto di un dispositivo perforatore di nastro (Puncher), quale supporto di memoria non volatile.

Nel nostro caso, ancora una volta, il funzionamento della "Write Punch Byte" è legata all'hardware della macchina, ma in maniera meno determinante: in questo caso infatti basta che la routine di gestione di tale dispositivo (a livello BIOS) sia formata dalla sola istruzione RET (Return).

Se si vuole, si può utilizzare il dispositivo "Puncher" ad esempio come dispositivo di uscita (seriale o parallelo) oppure come interfaccia verso un programmatore di E-PROM: anche in questo caso sta al costruttore del sistema o all'utente gestire l'opportuno programma di comunicazione all'interno del BIOS.

Comunque, qualsiasi sia la scelta, il carattere da inviare in output deve essere posto nel registro E, per poi chiamare la routine in questione con il valore 4 nel registro C.

Funzione 5: Write List Byte

Come dice il nome, questa funzione invia il byte, posto nel registro E, al dispositivo "LST:".

Anche se a prima vista non sembrerebbe, dato che praticamente tutti i sistemi dotati di CP/M sono dotati di interfaccia per stampante, anche in questo caso bisogna stare attenti.

Infatti questa funzione non è assolutamente in grado di segnalare al computer lo stato della stampante, come dire che se tale dispositivo non è "On Line" (come capita spesso!), il programma chiamante aspetterà all'infinito, oppure fino a che l'operatore non abbia la curiosità di vedere se la stampante è effettivamente in linea.

Altro problema è dato ad esempio se finisce la carta, ma in genere quasi tutte le stampanti danno un'indicazione ottica (led lampeggiante) o acustica (un sonoro bip) in casi simili.

Anche in questo caso, perciò, il corretto funzionamento della routine di output dipenderà da come è stata implementata la routine a livello BIOS.

Funzione 6: Direct Console I/O

Questa è una funzione che permette l'input o l'output di un byte da o verso la console: è praticamente l'insieme delle due routine di input e di output per il dispositivo "CON:". Vi sono però alcune limitazioni di cui ora parleremo: vediamo come si effettua la chiamata a tale routine.

In questo caso si utilizzano il registro E e l'accumulatore A: nel caso di input di un byte, nel registro E dobbiamo porre il valore ØFFH, mentre in A avremo Ø se non vi è alcun carattere in arrivo oppure un valore diverso da Ø, e cioè proprio il carattere arrivato.

Viceversa, per quanto riguarda l'output, basta che il registro E "non" contenga ØFFH, ma proprio il byte da inviare: in questo caso A non serve.

Ecco dunque quali sono le limitazioni: non si può ricevere il byte nullo (ØØ) né si può inviare il byte ØFFH; in quanto, come visto, tali valori servono da "segnalatori".

È chiaro che questo comportamento può non essere quello desiderato, ad esempio se desiderassimo ricevere o inviare anche i due byte visti: viceversa, con un'opportuna programmazione dell'IOBYTE (vedere le prossime due funzioni) si potrebbe, ad esempio, "connettere" il dispositivo considerato RDR: alla console per aversi in questo caso il test del dispositivo, ricevendo in A o il byte oppure un valore nullo nel caso in cui tale dispositivo non sia in grado di fornire un byte.

Un break prima di proseguire

Abbiamo parlato dell'IOBYTE e del fatto che le prossime due funzioni lo utilizzano: ebbene, dato che l'argomento è alquanto vasto e delicato, abbiamo pensato di trattarlo per intero nella prossima puntata, soprattutto per non essere costretti ad interromperlo questa volta per i soliti motivi di spazio: saltiamo perciò le routine 7 ed 8 (rispettivamente "Get IOBYTE Setting" e "Set IOBYTE"), rimandando al prossimo numero di MCmicrocomputer, mentre continuiamo la nostra "rassegna".

Funzione 9: Display "$" Terminated String

Questa funzione è molto utile per visualizzare sullo schermo della console dei messaggi, segnalazioni varie o richieste di input di dati via tastiera: infatti tale funzione invia in output al dispositivo "CON:" una stringa di caratteri terminante con il simbolo "$".

Come si vede, tale routine è molto utile, ma appare anche evidente che il messaggio da inviare al video non potrà contenere al suo interno il carattere "$", pena il troncamento del messaggio stesso al sopraggiungere di tale delimitatore.

La chiamata a tale funzione si effettua ponendo nella coppia di registri DE l'indirizzo della stringa "$" - terminated e ponendo in C il valore 9: con la consueta CALL ØØ5 inneschereMo per l'appunto la nostra funzione di output.

Funzione 10: Read Console String

È questa una funzione molto utile in quanto consente l'input dalla console (dalla tastiera, cioè) di una stringa, in genere un comando, terminante come è ovvio con un "Carriage Return" o "Return" o "Enter" che dir si voglia.

La potenza di questa routine risiede nel fatto che consente tutti i possibili controlli

di editing, quali il "Backspace", ecc.

In questo caso l'operatore può infatti "ritornare sui suoi passi" cancellando quanto scritto finora, tramite il Control-U o Control-X, per poi riscrivere una nuova stringa.

Vediamo ora la chiamata a tale funzione: nella coppia di registri DE dovrà essere posto l'indirizzo di memoria del Buffer che conterrà la stringa impostata dall'operatore; nel registro C invece andrà il valore (decimale) 10 o meglio (dato che presumibilmente lavoriamo in linguaggio macchina) 0AH, come ormai sappiamo.

Ma vediamo ora più da vicino la struttura di questo Buffer: stabiliamo di avere a disposizione, per il comando da inviare da tastiera (la stringa in input), 128 caratteri (80H) dei quali poi presumibilmente useremo solo una parte: al limite uno solo, come pure nessuno.

Allora il nostro buffer sarà così concepito:

— il primo byte indicherà la lunghezza del buffer stesso: nel nostro caso 80H, valore che dovremo impostare noi.

— Il secondo byte invece ci indicherà "alla fine" quanti sono i caratteri impostati da tastiera e ci potrà essere utile come contatore nella successiva analisi di quanto digitato.

— I successivi (nel nostro caso 128) byte costituiscono il buffer stesso.

Dunque come regola bisogna prevedere due byte in più in testa al buffer, byte che verranno gestiti dalla routine stessa del BDOS e che poi utilizzeremo noi alla fine.

Vediamo ora alcuni casi: se ad esempio digitiamo qualcosa, cancelliamo alcune parti, le riscriviamo, ecc, alla fine il secondo byte del Buffer ci darà la lunghezza corretta della stringa impostata, come dire che nel Buffer vedremo l'ultima versione della stringa e nel secondo byte la sua lunghezza.

Attenzione che il carattere "di consenso" e cioè il "Return", NON comparirà nella stringa finale; come dire: se in risposta ad un certo prompt rispondiamo con un semplice "Return", il nostro buffer conterrà tutti zeri ed il secondo byte sarà anche lui nullo, ad indicare che la "stringa in input" era di lunghezza nulla, dal momento che il "Return" non conta. Viceversa però un secondo byte nullo vuol dire che l'operatore ha premuto "Return", cosa che può essere utile in certi casi.

Funzione 11: Read Console Status

Anche questa è una funzione molto utile: ci dice se "in quel momento" c'è un carattere proveniente dalla console ed in attesa di essere elaborato.

È importante notare la differenza con la funzione n. 1 (Read Console Byte), la quale invece aspetta che ci sia un carattere da elaborare, con un'attesa che può diventare molto lunga.

Nel nostro caso invece, la routine testa lo stato della tastiera (o meglio della console) ponendo in accumulatore il valore 0 se non vi è alcun carattere in attesa e viceversa ponendo il valore 0FFH.

Inutile segnalare l'importanza di questa funzione, molto veloce nel senso che non può inchiodare il sistema ed utile laddove l'elaborazione non si deve fermare per attendere un dato da tastiera, ma viceversa deve ad esempio essere fermata del tutto alla pressione di un certo tasto.

In quest'ultimo caso alla chiamata della routine in questione dovrà seguire il test dell'accumulatore: se è nullo l'elaborazione proseguirà per la sua strada.

Se viceversa il contenuto è 0FFH allora si dovrà procedere all'effettiva lettura del byte con una chiamata alla routine n. 1, che perciò non dovrà attendere la pressione del tasto.

Funzione 12: Get CP/M Number

Ecco una funzioncina semplice semplice, il cui uso è tutto sommato alquanto limitato, solamente nei casi in cui è effettivamente determinante per il corretto proseguimento dell'elaborazione il sapere con quale versione del CP/M stiamo lavorando. La routine in esame fornirà nella coppia di registri HL un codice ben determinato, dato da:

— H vale 0 se si tratta del CP/M, mentre vale 01H se si ha a che fare con l'MP/M.

— L indica il numero della versione:

— 00H per tutte le versioni precedenti la 2.0 (ad es. la 1.4)

— 20H per la versione 2.0

— 21H per la versione 2.1

— 22H per la versione 2.2 e così via.

Come dicevamo l'utilizzazione pratica di tale funzione si ha in un numero ristretto di casi: ad esempio quando si trasportano programmi, che girano in CP/M versione 2.X, in vecchi CP/M (ad esempio il già citato 1.4) non dotati ad esempio di routine di I/O di File Random.

Funzione 13: Reset Disk System

L'ultima routine che analizziamo in questo numero riguarda il reset del sistema a livello dischi.

Non ci sono parametri in ingresso o in uscita e la funzione effettua le seguenti operazioni:

— resetta le tabelle interne del BDOS

— seleziona A: come disco di default

— riporta il valore del DMA Address a 0080H

— stabilisce lo stato di Read/Write per tutti i dischi.

È una funzione molto utile quando l'operatore deve effettuare il cambio di un dischetto: normalmente infatti, quando l'operatore compie questa (a volte) deleteria operazione, il CP/M non ne sa assolutamente niente, dato che non c'è alcun modo per lui di sapere che lo sportellino del driver è stato aperto! Ecco che dunque per lui il dischetto "non" è stato cambiato... Però il CP/M ha in memoria una tabella opportunamente legata al dischetto vecchio ed in particolare alla situazione di allocazione dei vari settori del disco stesso.

Ora, cambiando tale dischetto, con elevatissima probabilità la nuova situazione di allocazione dei settori sarà completamente diversa da quella memorizzata: ecco che dunque il CP/M, per evitare danni, segnala (con il ben noto quanto odiato messaggio "BDOS ERROR ON A:R/O") che non può scrivere sul dischetto, nel caso fosse comandato in tal senso.

Usando invece la funzione in esame, l'operatore può cambiare il dischetto sotto controllo del programma: ad esempio tale programma potrebbe:

— inviare un messaggio alla console del tipo "Cambiare il disco e premere Return quando si è pronti"

— attendere la pressione del tasto "Return"

— chiamare la routine 13 del BDOS per resettare il sistema dei dischi, per assicurare un corretto funzionamento nel caso di successive scritture del dischetto da parte del CP/M.

Dimenticavamo di ricordare, ma il lettore che è incappato in questa situazione lo sa bene, che nella maggioranza dei casi in risposta al fatidico messaggio di R/O (che sta per Read Only) si può soltanto premere Control-C con conseguenze ovvie: il reset del sistema e la perdita del programma in esecuzione... Concludiamo perciò la puntata con l'ultima delle funzioni generali: fra due puntate ci occuperemo delle restanti funzioni, relative alla gestione dei file su disco. Dato che l'argomento è enormemente vasto e alquanto complicato cercheremo di illustrare il tutto con piccoli esempi di programmi. MC

Funzioni del BDOS in dettaglio

Valore di C	Nome	Input	Output
3	Reader Input	—	A = carattere ASCII
4	Punch Output	E = carattere ASCII	—
5	List Output	E = carattere ASCII	—
6	Direct Console Input	E = 0 FFH	A = carattere ASCII
	Output	E = carattere ASCII	—
9	Print String($)	DE = indirizzo stringa	—
10	Read Console Buffer	DE = indirizzo buffer	dati nel buffer
11	Get Console Status	—	A = stato
12	Get Version Number	—	HL = numero versione
13	Reset Disk System	—	—

Tabella 1

STAMPANTI Epson,

una scelta prestigiosa, senza compromessi

FX-80

Indispensabile nelle applicazioni in cui la versatilità e la qualità di stampa sono un imperativo.

Possibilità di creare qualsiasi carattere su una matrice di 11x9 punti. Memoria RAM da 4 Kbyte. 256 tipi di caratteri definibili dall'utente. 136 tipi di caratteri a corredo. Alta velocità di stampa a 160 caratteri al secondo su 80 colonne. Ben 9 modi di stampa grafica punto a punto selezionabili sulla stessa riga contemporaneamente.
Inseritori automatici di fogli singoli a singola e doppia vaschetta.

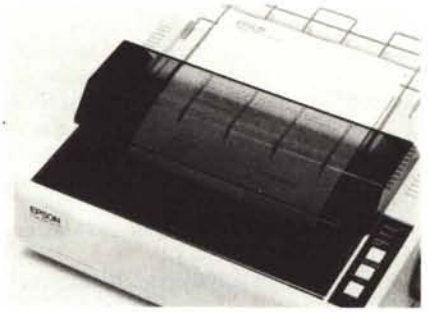
FX-100

Con 132 colonne e 160 caratteri al secondo, la FX-100 è la stampante ideale per data processing e tabulati, specialmente in ufficio grazie anche alla possibilità di accettare fogli singoli e moduli continui di qualsiasi formato. La matrice di punti 11x9 consente prestazioni grafiche e la formazione di una grandissima varietà di tipi di carattere, fino a 256, definibili anche dall'utente e memorizzati nei 3 kbyte di RAM interna. La FX-100 non teme la fatica: la testina di stampa è garantita per oltre 100 milioni di caratteri ed è facilmente sostituibile.
Inseritori automatici di fogli singoli.

Scegliere una stampante è facile?

C'è una sola regola, pretendere sempre il massimo delle prestazioni, senza compromessi: materiali e componenti di prima qualità, disegno elegante, grande affidabilità, robustezza, facilità e flessibilità d'impiego, prezzo adeguato e la garanzia di un grande costruttore leader mondiale.

Così, con Epson, la scelta **è facile e sicura.**
Epson il più grande costruttore al mondo di stampanti vi offre una gamma di prodotti prestigiosi che soddisfano ogni vostra necessità. Epson, una soluzione raffinata, in esclusiva per il vostro computer.



RX-80 / RX-100

Le migliori prestazioni da stampanti, semplici, versatili, silenziose e veloci con 100 caratteri al secondo. 128 tipi di caratteri selezionabili e 11 set internazionali. 80 o 132 colonne. 6 diverse possibilità grafiche. Tutti i tipi di carta, modulo continuo, foglio singolo. Inseritori automatici di fogli singoli.

studio martinetti

Epson dunque, senza compromessi.

EPSON

EPSON CORPORATION
HEAD OFFICE
80 Hirooka, Shiojiri-shi, Nagano
399-07 JAPAN

EPSON, computer e periferiche
sono prodotti distribuiti,
assistiti e garantiti
da SEGI S.p.A. - Milano Via Timavo, 12

segi SERVIZI GENERALI PER L'INFORMATICA