
# i trucchi del CP/M

*a cura di Pierluigi Panunzi*

## Le funzioni del BDOS

*Siamo oramai arrivati quasi al termine della nostra analisi delle funzioni del BDOS: delle residue funzioni sono particolarmente importanti, come vedremo, la «Read Random» e la «Write Random», particolarmente usate dai programmi applicativi che girano in ambiente CP/M.*

### Funzione 32: Set/Get User Number

Come si può arguire dal nome, questa funzione in realtà è «doppia», nel senso che permette sia di predisporre un determinato «user number», sia di leggere (e perciò sapere) sotto quale «user» ci si trovi in un certo istante.

In particolare, dopo aver posto nel registro C (come di consueto) il numero della funzione, 32 in questo caso, a seconda del valore che porremo nel registro E otterremo la prima o la seconda operazione possibile; rispettivamente, ponendo in E il valore esadecimale ØFFH, otterremo in output nell'accumulatore il numero d'utente selezionato (un valore compreso tra Ø e 15), viceversa ponendo nel registro E un valore compreso tra Ø e 15 (ovviamente il numero d'utente che desideriamo selezionare), automaticamente otterremo la selezione dell'«user number» desiderato.

Per quest'ultima operazione, già sappiamo che esiste il comando «USER n» (con «n» compreso tra Ø e 15, altrimenti il CP/M risponderà con un pronto «?»...), che si può inviare da console per entrare così in un altro «ambiente»: l'unico inconveniente è che in tal modo l'utente stesso può dimenticarsi in quale «ambiente» si trovi, specie dopo aver effettuato parecchi «salti» da un user all'altro. In tale frangente è più sicuro tornare sui propri passi e cioè forzare con «USER Ø» il ritorno all'«ambiente» di default. Evidentemente la funzione in esame è utilizzabile in un programma in assembler, laddove cioè non è possibile all'utente lavorare interattivamente.

Ad esempio la funzione in esame può essere usata in un programma a ricerca di un dato file nei sedici possibili «user», settando via via in un loop il numero d'utente da Ø a 15 ed andando a vedere se in tale utente esiste il file in esame: è proprio quello che facciamo nel programma di figura 1, che ora andiamo brevemente ad illustrare.

Innanzitutto troviamo il controllo se nel comando è stata rispettata la sintassi e cioè se si è digitato correttamente

```
SEARCH FILENAME.TYP
```

dove «filename.typ» è appunto il nome del file che desideriamo cercare: può essere eventualmente preceduto dall'indicazione dell'unità a dischi desiderata ed inoltre può contenere dei caratteri «?». In tal caso la ricerca terminerà (in caso favorevole) quando viene trovato il primo file il cui nome si adatta alla «maschera» impostata.

Ad esempio, supponiamo di cercare il file «ESEMPIO.DAT», in directory dove sappiamo esistono anche i file «E.DAT» ed «ESAMI.DAT»: se nel comando impostiamo

```
SEARCH E*.DAT
```

potrebbe capitare di fermarsi al programma «E.DAT», se questo viene prima nella directory. Ricordiamo infatti che la funzione del BDOS «Open File», nel caso in cui nel nome del file compaiano dei «?», attiva il primo file della directory che soddisfa il «filename ambiguo».

Successivamente, dopo il setup dell'FCB con il nome impostato nel comando, si ha la prima utilizzazione della funzione in esame, per leggere l'user number corrente, con lo scopo di reimpostarlo al termine del programma: è questa una buona prassi in casi del genere in quanto altrimenti l'utente rimarrebbe nell'ambito di un «user» ignoto, all'interno del quale potrebbe cancellare senza volerlo dei file, specie se non è particolarmente «edotto».

Ecco dunque il loop nel quale si setta un «user number», progressivamente da Ø a 15, si cerca il file desiderato e si esce in caso positivo.

In caso negativo, solo dopo aver analizzato tutti gli user fino all'user 15, allora si stamperà su video un messaggio indicante l'esito sfavorevole della ricerca.

Invece in caso positivo si effettuerà una semplicissima conversione del numero (esadecimale) contenuto in E in uno o due caratteri ASCII: in questo caso si gioca sul fatto che se il numero è formato da una cifra, allora questa sarà preceduta da un «blank», mentre viceversa sarà preceduta da un «1».

È in questo caso infine che viene inviato al video un messaggio indicante appunto il numero dell'«user» in cui

si trova il file desiderato: l'utente poi continuerà come più gli aggrada...

## Funzioni 33 e 34: «Read Random» e «Write Random»

Abbiamo volutamente raggruppato in un unico paragrafo queste due funzioni in quanto, a dispetto del differente significato intrinseco, vengono gestite in modo identico (a parte le ovvie differenze sulle quali ci soffermeremo).

In particolare le due funzioni permettono rispettivamente di leggere e di scrivere un dato record (di tipo «CP/M» e perciò di 128 byte) da un file random, nel quale, come sappiamo, si può appunto accedere al singolo record senza essere costretti a leggere i precedenti, come invece accade per i file sequenziali.

In entrambi i casi, il CP/M riterrà che l'utente abbia già «aperto» il file in esame (con la funzione «Open File»), abbia settato correttamente il «DMA Address» (con la funzione «Set DMA Address»), nonché abbia settato il numero del record desiderato nell'apposito campo dell'FCB. In mancanza di tutto ciò, la migliore delle ipotesi è un'operazione errata, per poi arrivare al «crash» del sistema, oppure peggio la cancellazione di file incolpevoli: attenzione quindi...

Eseguiti correttamente questi passi iniziali, la funzione prescelta (Read o Write) calcolerà l'«extent» contenente il record desiderato e tenterà di aprire tale extent per leggere o scrivere il record verso o dal buffer il cui indirizzo è proprio il «DMA Address» (come già sappiamo dalle puntate precedenti). Nel caso della «Read Random» al termine dell'operazione si otterrà in accumulatore (A) un valore rispecchiante appunto l'esito della funzione. In particolare si ha la seguente tabella di corrispondenza:

| valore di A | significato |
|---|---|
| 0 | operazione completata con successo |
| 1 | tentata la lettura di un record non scritto (e cioè inesistente) |
| 3 | il CP/M non è riuscito a chiudere l'extent |
| 4 | tentata la lettura di un extent non esistente |
| 6 | tentata la lettura oltre la fine del disco |

Invece nel caso della «Write Random» si possono avere i seguenti valori di A, legati a particolari esiti dell'operazione:

| valore di A | significato |
|---|---|
| 0 | operazione completata con successo |
| 3 | il CP/M non è riuscito a chiudere l'extent |
| 5 | directory full |
| 6 | tentata la scrittura oltre la fine del disco |

In entrambi i casi, e a differenza delle rispettive funzioni relative ai file sequenziali, le due funzioni in esame non effettuano automaticamente l'incremento del numero del record, in quanto, a pensarci bene, ciò sarebbe in questo caso deleterio: ricordiamo infatti che una delle caratteristiche dei file random è di essere «sparso» e cioè con record sparsi qua e là ed aventi numero *non consecutivo*.

Può capitare infatti un caso abbastanza singolare: supponiamo infatti di creare, nell'ambito di un certo file random, i record 1 e 5000 soltanto.

Il CP/M in questo caso creerà correttamente due extent, il primo contenente il record 1 ed il secondo contenente il record 5000: inutile dire che i due extent *non* sono consecutivi.

Infatti mentre il record 1 si trova nell'extent 0, il secondo (il record numero 5000) si troverà in un extent il cui numero può essere calcolato in base alla conoscenza del valore di alcuni parametri fondamentali del CP/M.

A titolo di esempio, supponiamo che il nostro computer dotato di CP/M preveda «Allocation block» di 2048 byte ed un numero totale di essi inferiore a 256: non ritorniamo certo sul significato di questi parametri, ma rimandiamo senz'altro alle puntate precedenti.

Facciamo dunque un po' di conti: in una directory entry trovano posto 16 allocation block (relative ad un certo file) ed inoltre in un singolo allocation block (da 2048 byte) troveranno posto 2048/128 = 16 record del nostro file random.

Ecco che nell'extent 0 troveremo indicazione dei 16 allocation block relativi ai record compresi tra il numero 0 e 255: dato che dunque vi sono 256 record per extent, il nostro record 5000 si troverà nel diciannovesimo extent (e perciò quello di numero 18). Nel nostro esempio dunque il CP/M creerà i due extent di numero 0 e 18: evidentemente se il nostro file contenesse solo il record numero 5000 allora esisterebbe *soltanto* l'extent numero 18 (strano ma vero...).

```
BDOS    EQU 5
CPM     EQU 0
FILNAM  EQU 5CH
PRINT   EQU 9
OPEN    EQU 15
USER    EQU 32
;
SEARCH  LD A,(FILNAM+1)
        CP 20H
        JR NZ,FILEOK
        LD DE,NOFILE    ;manca il filename
        LD C,PRINT
        CALL BDOS
        JP CPM
FILEOK  LD BC,12
        LD HL,FILNAM
        LD DE,FCB
        LDIR            ;setup del File Control Block
        LD BC,24
        LD DE,FILNAM+14
        LD (HL),0
        LDIR            ;azzeramento byte rimanenti
        LD E,0FFH
        LD C,USER
        CALL BDOS       ;get user number
        PUSH AF         ;salvato nello stack
        LD E,0          ;user 0
LOOP    PUSH DE
        LD C,USER
        CALL BDOS       ;set user number
        LD DE,FCB
        LD C,OPEN
        CALL BDOS       ;ricerca 'file.typ'
        POP DE
        INC A           ;se a=FF allora "file not found"
        JR NZ,FOUND
        INC E           ;si incrementa l'user number
        LD A,16
        CP E
        JR NZ,LOOP      ;se minore di 16 si effettua un altro loop
        LD DE,NOTFND    ;'file.typ' non e' stato trovato
EXIT    LD C,PRINT
        CALL BDOS
        POP AF          ;ripristino dell'user number iniziale
        LD E,A
        LD C,USER
        CALL BDOS
        JP CPM          ;ritorno al CP/M
FOUND   LD A,E          ;'file.typ' e' nell'user (E)
        AND 0FH         ;non si sa mai...
        CP 0AH          ;conversione in due caratteri ASCII
        LD HL,3020H
        JR C,DIGIT
        LD L,31H        ;numero di due cifre
        SUB 0AH
DIGIT   ADD A,H
        LD H,A
        LD (VALUE),HL
        LD DE,FND       ;messaggio "file found"
        JR EXIT
;
NOTFND  DEFM 'FILE NOT FOUND$'
FND     DEFM 'FILE FOUND IN USER
VALUE   DEFM '  $'
NOFILE  DEFM 'NO FILENAME$'
FCB     DEFS 36
;
        END SEARCH
```

*Figura 1 - Listato del programma «SEARCH», che consente di cercare un file nell'ambito dei 16 «user».*

```
; questa routine prevede due parametri in input
;    DE  =  indirizzo dell'FCB contenente il nome del file
;    HL  =  numero di record da 128 byte
;
; in uscita si ha l'esito dell'operazione
;    NZ  =  esito positivo
;     Z  =  errore nell'operazione
;
; la chiamata si effettua nel modo seguente:
;
;       ...
;       LD HL,NUMBER        ;numero record
;       LD DE,FCB           ;FCB del file da costruire
;       CALL SETUP
;       JP Z,ERRORE         ;routine di errore
;       ...
;
BDOS    EQU 5
ERASE   EQU 19
CREATE  EQU 22
SETDMA  EQU 26
WRSEQ   EQU 21
;
SETUP   PUSH HL
        PUSH DE
        LD C,ERASE
        CALL BDOS           ;si cancella il file gia' esistente
        POP DE
        PUSH DE
        LD C,CREATE
        CALL BDOS           ;si crea il file desiderato
        INC A
        JR Z,EXIT           ;se errore si termina
        LD DE,BUFFER
        LD C,SETDMA
        CALL BDOS           ;setup del DMA address
        LD BC,128
        LD HL,BUFFER
        LD DE,BUFFER+1
        LD (HL),0
        LDIR                ;azzeramento buffer
LOOP    POP DE
        POP HL
        LD A,L
        OR H
        JR NZ,NOEND
        INC A
        RET                 ;fine del programma con condizione 'NZ'
NOEND   DEC HL
        PUSH HL
        PUSH DE
        LD C,WRSEQ
        CALL BDOS           ;scrittura del record nullo nel file
        INC A
        JR NZ,LOOP
EXIT    POP DE
        POP HL
        RET                 ;uscita per "directory full"
;
BUFFER  DEFS 128
;
        END
```

*Figura 2 - Listato della subroutine SETUP che permette di creare un file random formato da (NUMBER) record «nulli».*

Questo fatto, che lavorando solo con funzioni «random» non comporta alcun inconveniente, viceversa crea un notevole scompiglio nel caso si operi con le funzioni «sequenziali».

Infatti se si tentasse di copiare il nostro file random (quello con due extent) per mezzo di un programma utilizzante le funzioni sequenziali, allora questo copierà soltanto il primo extent in quanto, non trovando l'extent *successivo* (quello di numero 1), segnalerebbe erroneamente l'«end of file», non essendo predisposto a leggere file sparsi: tutto questo perché le funzioni sequenziali eseguono automaticamente l'incremento del «record number» al termine dell'operazione.

Altra considerazione da fare è che, anche nel caso in cui il file random non abbia l'extent numero Ø (era il secondo caso del nostro esempio precedente), si deve aprire (con la «Open File») comunque l'extent Ø: anche se ciò può sembrare un controsenso nel caso indicato (che non è però per nulla anomalo, anzi potrebbe capitare nella realtà), solo aprendo l'extent Ø si può operare correttamente.

Per evitare inconvenienti nel caso di copiatura di file «random sparsi», allora si può utilizzare il programma della figura 2, che consente, a partire dal numero di record da 128 byte desiderati, di creare un file sequenziale «completo» in cui cioè eventuali record mancanti sono tutti nulli: ciò comporta evidentemente un inutile spreco di spazio sul dischetto, ma ci salvaguarda dai problemi di cui abbiamo parlato più sopra.

A completamento dei commenti riportati nel listing diciamo che il programma presentato è una subroutine a cui fa riferimento un programma applicativo: dal momento che è una subroutine ad un certo punto del programma principale ci sarà una CALL, preceduta dal setup dei registri HL e DE, rispettivamente con il numero di record del nostro file «random-sequenziale» e con l'indirizzo di un FCB, che il programma precedente dovrà provvedere ad inizializzare correttamente con il nome del file in questione e con i rimanenti byte a Ø.

Ecco che perciò la subroutine effettuerà le seguenti operazioni:

— dapprima cancellerà incondizionatamente il file eventualmente già presente ed avente lo stesso nome di quello che desideriamo creare: se ci sono problemi basta rinominare o il file già esistente oppure il nuovo file...

— successivamente crea il file in questione ed in caso di segnalazione di «directory full» subito uscirà con la condizione di «zero», indicante appunto un esito negativo

— invece in caso positivo si avrà il setup del DMA Address e l'azzeramento del buffer, che verrà poi trasferito nel file appena creato

— all'interno del loop principale, da eseguire tante volte quanto è il valore posto in HL all'inizio, troviamo la scrittura di un record in un file sequenziale: anche in questo caso la condizione di «directory full» comporterà la fine della subroutine con la condizione di «zero»

— invece se tutto va bene si effettuerà un nuovo ciclo e ciò continuerà fino a che HL arriva a Ø, nel qual caso si uscirà dalla subroutine con la condizione di «not zero», indicante un esito positivo.

Come si vede il programmino proposto è molto semplice ma efficace; successivamente il programma principale provvederà a scrivere i record desiderati, con la funzione «Write Random» oppure con la funzione «Write Sequential», a seconda che si desideri di non effettuare oppure di effettuare l'incremento automatico del numero del record al termine dell'operazione di scrittura.

Prima di concludere la puntata, ricordiamo che finora si è parlato di record da 128 byte: nulla vieta all'abile programmatore di considerare record di lunghezza differente, sia più piccoli che più grandi di 128 byte.

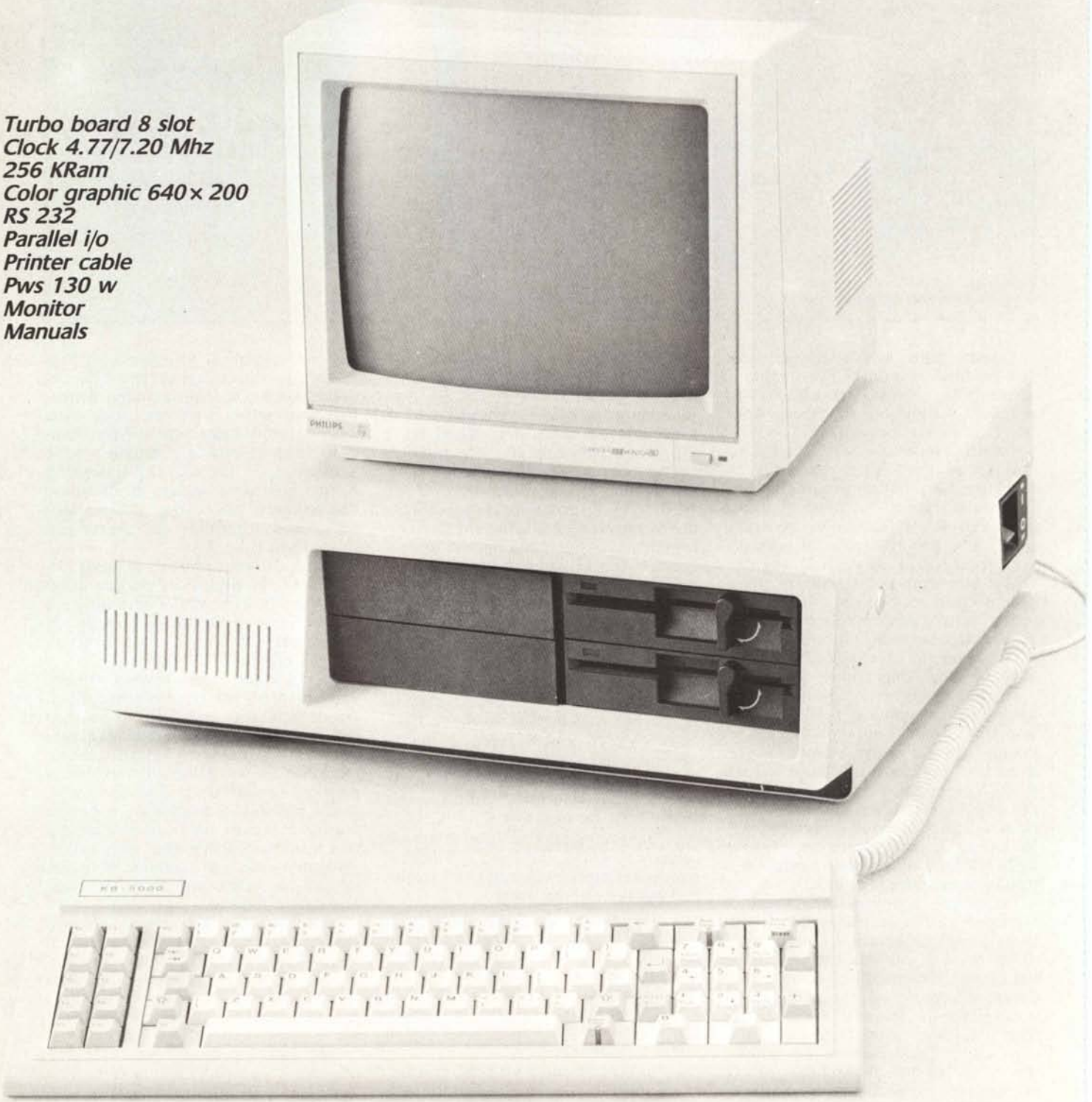
In entrambi i casi però è sempre conveniente lavorare con record «logici» di lunghezza pari ad una potenza di 2, in modo tale da evitare situazioni in cui un record logico è posto a cavallo di due record «fisici» di 128 byte, o peggio a cavallo tra la fine di un allocation block ed il successivo, o peggio ancora a cavallo tra la fine di un extent e l'inizio dell'extent successivo: evidentemente il programma in questi tre casi, dovrebbe prevedere altrettante strade differenti.

Invece con record «logici» di lunghezza pari ad una potenza di 2, si è almeno sicuri che in un record «fisico» ne entreranno sempre un numero intero, nel caso in cui si sceglie per i record «logici» un sottomultiplo di 128.

Nel caso opposto invece (e cioè quando il record «logico» avrà una lunghezza pari ad un multiplo di 128) si dovrà organizzare il tutto in modo tale da effettuare più letture o scritture per arrivare all'ampiezza del record «logico». MC

# IL COMPATIBILE È APPARSO!

*Turbo board 8 slot*
*Clock 4.77/7.20 Mhz*
*256 KRam*
*Color graphic 640 × 200*
*RS 232*
*Parallel i/o*
*Printer cable*
*Pws 130 w*
*Monitor*
*Manuals*




*Cerchiamo distributori per zone libere*
***QUASAR S.r.l. - 13050 Pratrivero (VC) - Tel. 015/778804/377 - Tx 211401 MILFIL***