GESTIONE DI PERIFERICHE SOTTO SISTEMA OPERATIVO MPX-32 (GOULD) IN COLLEGAMENTO CON PERSONAL COMPUTER

S. Cerri - E. Fantini

B4-42 (1987)

GESTIONE DI PERIFERICHE SOTTO SISTEMA OPERATIVO MPX-32 (GOULD) IN COLLEGAMENTO CON PERSONAL COMPUTER

Sirio Cerri - Enrico Fantini

Istituto di Elaborazione della Informazione del CNR - Pisa

Questo package per la gestione di periferiche sotto il sistema operativo MPX-32 in collegamento con il CP/M e` stato implementato nell'ambito della collaborazione tecnico-scientifica tra la Societa` Aeritalia G.V.T. e l'Istituto di Elaborazione della Informazione del CNR di Pisa per lo studio di un sistema di controllo di qualita` per componenti aeronautici.

Il collegamento tra Gould e PC e` effettuato tramite una normale linea seriale configurata a 19200 baud, 8 bits, no parita`, 1 stop bit.

Le periferiche utilizzate in questo specifico lavoro sono il disco, il nastro magnetico e il sistema video pittorico SVP-2000. La richiesta di un'operazione su una periferica e` una sequenza di 4 byte di cui i primi tre contengono i caratteri "SUB" ed il quarto definisce la routine richiesta; quest'ultimo byte puo` avere un valore tra 0 e 255 e quindi il package puo` essere aggiornato ed ampliato fino ad un massimo di 256 richieste.

Attualmente le richieste utilizzate nell'ambito di questo lavoro sono suddivise come segue:

SVP-2000	SUBx	dove X = 0 ÷ 49
File di disco	SUBx	dove X = 50 ÷ 99
File di nastro	SUBx	dove X = 100 ÷ 149

Poiche` la mole di dati da gestire e` abbastanza rilevante (circa 1 MByte nel caso della massima matrice), e` stato essenziale poter effettuare il collegamento alla massima velocita` consentita dai sistemi per ridurre i tempi di trasferimento.

Alcuni problemi, dall'impossibilita` da parte del sistema MPX-32 di gestire dati binari sulla linea dichiarata "user terminal", alla

necessita` di avere il programma non swappabile dallo stesso MPX-32 (per i problemi di sincronismo con il programma sotto CP/M), all'attivazione sotto MPX-32 del programma gestore senza intervento di operatore, sono stati risolti per il primo scrivendo, una routine con chiamate dirette alle SVC in linguaggio macchina per la gestione della linea di collegamento, il secondo grazie ad alcune specifiche in fase di link, dichiarando alta priorita` e residente in memoria il programma, ed infine per il terzo simulando le richieste ed interpretando le risposte da parte del programma gestore sotto CP/M proprio come se alla linea fosse collegato un normalissimo terminale di lavoro.

Il collegamento, come gia` detto, e` stato concepito senza alcun intervento da parte di operatori sul sistema Gould, per cui il log-in viene effettuato automaticamente dai programmi operanti sul CP/M.

Le operazioni richieste dal programma attivato sotto CP/M si possono riassumere in:

1) richiesta di attenzione del sistema MPX-32
2) log-in con nome e chiave di accesso
3) attivazione di un programma batch che attiva a sua volta il programma di gestione vero e proprio
4) log-out lasciando il programma di gestione attivo in attesa di comandi da parte del programma gestore sotto CP/M.

La sincronizzazione tra i due programmi (CP/M e MPX-32) e` stata demandata ad un hand-shake software (questo per l'impossibilita` da parte del S.O. MPX-32 di gestire le linee di controllo della porta seriale) e quindi ogni richiesta e` seguita da una verifica di consenso da parte dei programmi colloquianti.

Appena attivato il programma sotto MPX-32, viene spedito il comando "STAT" dal CP/M a cui il primo dovra` rispondere con il proprio nome ("ATDSK" per il programma che gestisce i file di disco, "ATSVP" per il programma che gestisce il sistema SVP-2000 e "ATMT" per il programma che gestisce il nastro magnetico) confermando cosi` la disponibilita` di accettare comandi successivi.

Esaminiamo dapprima la parte di log-in che e` comune a tutte le periferiche:

CP/M	MPX-32
Spedisce il carattere per richiedere l'attenzione del sistema. Nel nostro caso il carattere e` ESC (codice 27).	Risponde con il logo del sistema operativo MPX-32 e richiede il nome dell'utente che vuole collegarsi.
Spedisce "MINUS" poiche` tale e` il nome dell'utente previsto.	Richiesta della chiave di accesso.
Spedisce "MINUS" come chiave prevista.	Controlla che l'utente sia abilitato e quindi manda i messaggi di sistema: gli ultimi 4 caratteri del messaggio devono corrispondere al "prompt" del terminal service manager "TSM>".
Controlla se ricevuto "TSM>" e denuncia collegamento attivo.	

Un altro comando comune e` quello di disattivazione del programma:

CP/M	MPX-32
"STOP", richiesta di disattivazione del programma.	Risponde con "K" prima di effettuare lo stop.

Esaminiamo ora quello specifico delle periferiche.

SVP-2000.

CP/M	MPX-32
Spedisce il comando di attivazione programma batch "BATCH BATSVP" + 13 + "X" + 13 dove: "BATCH BATSVP richiesta di attivazione programma BATSVP "X" richiesta di log-out 13 e` il codice del ritorno carrello (CR).	Viene schedulato il programma BATSVP e quindi effettuato il log-off.
Viene attivato il time-out per assicurare l'avvenuto log-off e quindi viene spedito il comando "STAT".	Viene risposto con "ATSVP".
Viene controllato di aver ricevuto "ATSVP" e si procede con i comandi propri della periferica: "SUB"+1 richiesta di reset hardware al sistema SVP-2000.	Effettuato il reset del sistema SVP-2000 risponde con "K".
Aspetta "K" e spedisce "SUB"+2, richiesta di configurazione del banco e della memoria di lavoro.	Risponde con "K".
Aspetta "K" e spedisce due byte: byte 1 = banco byte 2 = memoria	Effettua la configurazione del sistema SVP-2000 e risponde "K".
"SUB"+3 richiesta di interfacciamento della memoria scelta sulle look-up.	Risponde "K".
Aspetta "K" e spedisce 1 byte con il numero della memoria.	Effettua l'interfacciamento e risponde "K".

CP/M	MPX-32
Aspetta "K" e spedisce "SUB"+6, richiesta di cancellazione memoria.	Risponde "K".
Aspetta "K" e spedisce 1 byte con il numero della memoria da cancellare.	Effettua la cancellazione della memoria richiesta e risponde "K".
Aspetta "K" e spedisce "SUB"+14, richiesta di scrittura area su SVP-2000.	Risponde "K".
Aspetta "K" e spedisce 9 byte con byte 1-2 coordinate Y iniziale byte 3-4 coordinate Y finale byte 5-6 coordinate X iniziale byte 7-8 coordinate X finale byte 9 memoria su cui scrivere.	Manda i parametri area a SVP-2000 e risponde "K".
Aspetta "K" e spedisce i valori dell'area riga per riga.	Scrive la riga su SVP-2000 e risponde "K".
. . .	. . .
Aspetta "K" e spedisce l'ultima riga.	Scrive la riga su SVP-2000 e risponde "K".

FILE DI DISCO

CP/M	MPX-32
Spedisce il comando di attivazione programma batch "BATCH BATDSK"+13+ "X"+13 dove: BATCH BATDSK" richiesta di attivazione programma BATDSK "X" richiesta di log-out, il numero 13 e` il codice del ritorno carrello (CR).	Viene schedulato il programma BATDSK e quindi effettuato il log-off.
Viene atteso il time-out per assicurare l'avvenuto log-off e quindi viene spedito il comando "STAT".	Viene risposto con "ATDSK".
Viene controllato di avere ricevuto "ATDSK" e si procede con i comandi propri di gestione file di disco: "SUB"+50 richiesta di creazione file.	Risponde con "K".
Aspetta "K" e spedisce 10 byte che contengono: byte 1-6 il nome del file da creare byte 7-8 lunghezza in byte dei rec. byte 9-10 numero del record.	Effettua la creazione del file come richiesto, se tutto OK risponde con "K" altrimenti risponde con il codice del tipo di errore avvenuto.
Aspetta "K" e spedisce "SUB"+53, richiesta scrittura riga.	Risponde con "K".
+->Aspetta K e spedisce la lunghezza del record e il numero del record da scrivere.	Risponde con "K".
Aspetta "K" e invia i valori del record.	Scrive il record sul file e risponde con "K".
+- Continua fino alla fine dei record.	

CP/M	MPX-32
Aspetta "K" e spedisce "SUB"+51, richiesta di chiusura del file.	Effettua la chiusura del file e risponde con "K".
Aspetta "K".	

FILE SU NASTRO MAGNETICO

CP/M	MPX-32
Spedisce il comando di attivazione programma batch "BATCH BATMT"+13+ "X"+13 dove: "BATCH BATMT" richiesta di attivazione programma BATMT "X" richiesta di log-out e il numero 13 e` il codice del ritorno carrello (CR).	Viene schedulato il programma BATMT e quindi effettuato il log-off.
Viene atteso il time-out per assicurare l'avvenuto log-off e quindi viene spedito il comando "STAT".	Viene risposto con "ATMT".
Viene controllato di aver ricevuto "ATMT" e quindi tramite un messaggio sul display, viene richiesto il montaggio fisico del nastro magnetico sull'unita` 0 del sistema MPX-32; una volta effettuato si prosegue con return dalla consolle del CP/M.	
Viene spedito "SUB"+100, richiesta di rewind.	Effettua il rewind e spedisce "K".

CP/M	MPX-32
Aspetta "K" e spedisce "SUB"+101, richiesta di salto file.	Risponde con "K".
Aspetta "K" e spedisce un byte contenente il num. di file da saltare.	Salta i file richiesti e risponde con "K".
Aspetta "K" e spedisce "SUB"+106, richiesta di scrittura record.	Risponde con "K".
Aspetta "K" e spedisce 4 byte di cui i primi 2 contengono la lunghezza del record da scrivere e gli altri il numero dei record da scrivere.	Risponde con "K".
Vengono spediti i valori da scrivere per il primo record.	Scrive il record sul nastro magnetico e risponde con "K".
. . .	. . .
Vengono spediti i valori da scrivere per l'ultimo record.	Scrive l'ultimo record e risponde con "K".
Aspetta "K" e spedisce "SUB"+108, richiesta di scrittura E.O.F.	Scrive E.O.F. su nastro e risponde con "K".
Aspetta "K".	

```
$JOB CAT/SVP MINUS,MINUS SLOF=EEE
$OPTION 2 3 4 5
$FORT77
C
C*************************************************************
C
C         Programma ATSVP
C
C         Questo programma viene usato per pilotare il
C         sistema SVP-2000 in collegamento con un PC MINUS.
C
C*************************************************************
C
      INTEGER*1 IWR(2048),IOK,IK(2),IGO(6),IZ(2)
      INTEGER*2 MEM,BAN,RIG,KBUF(1024),PWR(1024)
      EQUIVALENCE (IWR(1),PWR(1))
      ICOM=0
C
C     Apertura LFC 'AAA' per colloquio con MINUS
C
      LFC=X'00414141'
      CALL RWB(LFC,0,IWR,1,IST)
      IGO(1)=65 ; IGO(2)=84 ; IGO(3)=03
      IGO(4)=86 ; IGO(5)=80 ; IGO(6)=13
      IK(1)=75 ; IK(2)=15
C
C     Attesa di un comando
C
10    CALL RWB(LFC,1,IWR,4,IST)
C
C     Test per comando 'STOP'
C
      IF(IWR(1).EQ.83.AND.IWR(2).EQ.84.AND.IWR(3).EQ.79
     -.AND.IWR(4).EQ.80) THEN
          CALL RWB(LFC,2,IK,2,IST)
          STOP
      ENDIF
C
C     Test per comando 'STAT' (richiesta dello stato da MINUS)
C     Invio al MINUS dei caratteri 'ATSVP' come risposta alla
C     richiesta di stato.
C
      IF(IWR(1).EQ.83.AND.IWR(2).EQ.84.AND.IWR(3).EQ.65
     -.AND.IWR(4).EQ.84) THEN
           CALL RWB(LFC,2,IGO,6,IST)
           GOTO 10
      ENDIF
C
C     Test per comando 'SUBn' (richiesta esecuzione routine SVP)
C
      IF(IWR(1).EQ.83.AND.IWR(2).EQ.85.AND.IWR(3).EQ.66) THEN
      IF(IWR(4).LT.0.OR.IWR(4).GT.14)GOTO 20
      GOTO(100,101,102,103,104,105,106,107,108,109,110,111,112,113,114)
     +IWR(4)+1
100   STOP
101       CALL TINIZ
          CALL TRSFW
          CALL TRHRW
          CALL WAIT(1000,1,KKK)
          CALL ASINT
       GOTO 200
102   CALL RWB(LFC,2,IK,2,IST)
      CALL RWB(LFC,1,IWR,2,IST)
```

```
      BAN=IWR(1) ; MEM=IWR(2)
      CALL BANCO(BAN,MEM)
      GOTO 200
103   CALL RWB(LFC,2,IK,2,IST)
      CALL RWB(LFC,1,IWR,1,IST)
      MEM=IWR(1)
      CALL INTM(MEM)
      GOTO 200
104   STOP
105   STOP
106   CALL RWB(LFC,2,IK,2,IST)
      CALL RWB(LFC,1,IWR,1,IST)
      MEM=IWR(1)
      CALL MEMC(MEM,0)
      GOTO 200
107   STOP
108   CALL RWB(LFC,2,IK,2,IST)
      CALL RWB(LFC,1,IWR,3,IST)
      MEM=IWR(1) ; RIG=IWR(2)+IWR(3)*256
      CALL RWB(LFC,2,IK,2,IST)
      CALL WAIT(300,1,KKK)
      CALL RWB(LFC,1,IWR,1024,IST)
      J=0
      DO I=1,1024,2 ;  J=J+1 ; KBUF(J)=IWR(I) ; ENDDO
      CALL SRIG(MEM,RIG,KBUF)
      GOTO 200
109   STOP
110   STOP
111   STOP
112   STOP
113   STOP
114   CALL RWB(LFC,2,IK,2,IST)
      CALL RWB(LFC,1,IWR,9,IST)
      KBUF(1)=O'11406'
      KBUF(2)=IWR(2)*256+IWR(1)
      KBUF(3)=IWR(4)*256+IWR(3)
      KBUF(4)=IWR(6)*256+IWR(5)
      KBUF(5)=IWR(8)*256+IWR(7)
      KBUF(6)=IWR(9)-1
      CALL TOUT(KBUF,6,IERR)
      NY=KBUF(3)-KBUF(2)+1
      NX=KBUF(5)-KBUF(4)+1
      LX=(NX+1)/2
      ICON=0
444   CALL RWB(LFC,2,IK,2,IST)
      CALL RWB(LFC,1,IWR,NX,IST)
      ICON=ICON+1
      IF(ICON.EQ.NY) THEN
          CALL TOUT(IWR,LX,IERR)
          GOTO 200
      ENDIF
      CALL RWB(LFC,2,IK,2,IST)
      CALL RWB(LFC,1,IWR(NX+1),NX,IST)
      CALL TOUT(PWR,NX,IERR)
      ICON=ICON+1
      IF(ICON.LT.NY) GOTO 444
200       CALL RWB(LFC,2,IK,2,IST)
          GOTO 10
      ENDIF
C
C         Test per comando di scrittura su SVP1000
C
      IF(IWR(1).EC.87.AND.IWR(2).EQ.68) THEN
```

```
            LUN=IWR(3)*256+IWR(4)
            CALL RWB(LFC,2,1K,2,IST)
            CALL RWB(LFC,1,IWR,LUN,IST)
            LUN=LUN+1
            IWR(LUN)=0
            LUN=LUN/2
            CALL TOUT(IWR,LUN,IERR)
            CALL RWB(LFC,2,IK,2,IST)
            GOTO 10
        ENDIF
C
C       Test per comando di lettura dall' SVP1000
C
        IF(IWR(1).EQ.82.AND.IWR(2).EQ.6E) THEN
            LUN=IWR(3)*256+IWR(4)
            LUX=(LUN+1)/2
            CALL TINP(IWR,LUX,IERR)
            CALL RWB(LFC,2,IWR,LUN,IST)
            GOTO 10
        ENDIF
        GOTO 10
20      IZ(1)=90 ; IZ(2)=13
        CALL RWB(LFC,2,IZ,2,IST)
        GOTO 10
        END
$IFT ABORT EXIT
$AS LIB TO @SYSTEM(LIBRERIE)GEPIM.LIB BLOC=N
$AS DIR TO @SYSTEM(LIBRERIE)GEPIM.DIR BLOC=N
$AS LO1 TO @SYSTEM(LIBRERIE)SVP.LIB   BLOC=N
$AS DO1 TO @SYSTEM(LIBRERIE)SVP.DIR   BLOC=N
$CATALOG
AS AAA TO DEV=TY7EA5
AS TSC TO DEV=UU
AS TSR TO LFC=TSC
ENVIRONMENT RESIDENT
BUIL ATSVP 30 NOM
$EOJ
$$
```

```
$JOB CAT/SVP MINUS,MINUS SLOF=EEE
$OPTION 2 3 4 5
$FORT77
C
C*************************************************************
C
C         Programma ATDSK
C
C         Questo programma e' usato per gestire file
C         di disco in collegamento con un PC MINUS.
C
C*************************************************************
C
       IMPLICIT INTEGER*2 (I-N)
       INTEGER*1 IWR(2000),IOK,IK(2),IGO(6),IZ(2),KER(3)
       INTEGER*2 MEM,BAN,RIG,KBUF(1000),PWR(1000)
       INTEGER*2 NOME(3),IER
       INTEGER*4 LFC
       EQUIVALENCE (IWR(1),PWR(1)),(KER,IER)
C
C      Apertura LFC 'AAA' per colloquio con MINUS
C
       KER(1)=KER(2)=0 ; KER(3)=13
       LFC=X'00414141'
       CALL RWB(LFC,0,IWR,1,IST)
       IGO(1)=65 ; IGO(2)=84 ; IGO(3)=68
       IGO(4)=83 ; IGO(5)=75 ; IGO(6)=13
       IK(1)=75 ; IK(2)=13
C
C      Attesa di un comando
C
10     CALL RWB(LFC,1,IWR,4,IST)
C
C      Test per comando 'STOP'
C
       IF(IWR(1).EQ.83.AND.IWR(2).EQ.84.AND.IWR(3).EQ.79
      -.AND.IWR(4).EQ.80) THEN
           CALL RWB(LFC,2,IK,2,IST)
           STOP
       ENDIF
C
C      Test per comando 'STAT' (richiesta dello stato da MINUS)
C      Invio al MINUS dei caratteri 'ATDSK' come risposta alla
C      richiesta di stato.
C
       IF(IWR(1).EQ.83.AND.IWR(2).EQ.84.AND.IWR(3).EQ.65
      -.AND.IWR(4).EQ.84) THEN
            CALL RWB(LFC,2,IGO,6,IST)
            GOTO 10
       ENDIF
C
C      Test per comando 'SUBn' (richiesta esecuzione routine)
C
       IF(IWR(1).EQ.83.AND.IWR(2).EQ.85.AND.IWR(3).EQ.66) THEN
       IF(IWR(4).LT.50.OR.IWR(4).GT.53)GOTO 20
       GOTO(150,151,152,153)IWR(4)-49
C*
C*   CREAZIONE FILE SENZA COLLOQUIO
C*
150    CALL RWB(LFC,2,IK,2,IST)
       CALL RWB(LFC,1,IWR,10,IST)
       DO I=1,3 ; K=(I-1)*2+1 ; NOME(I)=IWR(K)*256+IWR(K+1) ; ENDDO
       LUN=IWR(7)+IWR(8)*256
```

```
      NREC=IWR(9)+IWR(10)*256
      IOP=-1
      IDCB=77 ; ISP=0
      CALL MAPRE(IDCB,NOME,IOP,PWR,I,I,IER)
      IF(IER.EQ.0) CALL RWB(LFC,2,IK,2,IST)
      IF(IER.NE.0) CALL RWB(LFC,2,KER,3,IST)
      GOTO 10
C*
C*  scrittura record
C*
153   CALL RWB(LFC,2,IK,2,IST)
      CALL RWB(LFC,1,IWR,4,IST)
      LUN=IWR(1)+IWR(2)*256
      IREC=IWR(3)+IWR(4)*256
      CALL RWB(LFC,2,IK,2,IST)
      CALL RWB(LFC,1,IWR,LUN,IST)
      NLU=(LUN+1)/2
      CALL SPACK(IWR,NLU,KBUF)
      CALL MLESC(IDCB,1,KBUF,IREC,LUN,IER)
      IF(IER.EQ.0) CALL RWB(LFC,2,IK,2,IST)
      IF(IER.NE.0) CALL RWB(LFC,2,KER,3,IST)
      GOTO 10
C
C*  CHIUSURA FILE
C*
151   CALL MCHIUD(IDCB,1,PWR,NREC,LUN,IER)
      IF(IER.EQ.0) CALL RWB(LFC,2,IK,2,IST)
      IF(IER.NE.0) CALL RWB(LFC,2,KER,3,IST)
      GOTO 10
152   STOP
20    STOP
      ENDIF
      END
$IFT ABORT EXIT
$AS LIB TO @SYSTEM(LIBRERIE)GEPIM.LIB BLOC=N
$AS DIR TO @SYSTEM(LIBRERIE)GEPIM.DIR BLOC=N
$AS L01 TO @SYSTEM(LIBRERIE)SVP.LIB BLOC=N
$AS D01 TO @SYSTEM(LIBRERIE)SVP.DIR BLOC=N
$AS L02 TO @CART1(MINUS)MGEFI.LIB BLOC=N
$AS D02 TO @CART1(MINUS)MGEFI.DIR BLOC=N
$CATALOG
AS AAA TO DEV=TY7EA5
OPTION LOWER
ENVIRONMENT RESIDENT
BUIL ATDSK 30 NOM
$EOJ
$$
```

```
$JOB MIN/MT SLOF=EEE
$OPTION 2 3 4 5
$FORT77
C
C*************************************************************
C
C         Programma ATMT
C
C         Questo programma viene usato per trasferire
C         files su MTO in collegamento con un PC MINUS.
C
C*************************************************************
C
      IMPLICIT INTEGER*2 (I-N)
      INTEGER*1 IWR(2000),IK(2),IGO(6),KER(3)
      INTEGER*4 LFC,LFCO
      EQUIVALENCE (KER,IER)
C
C     Apertura LFC 'AAA' per colloquio con MINUS
C     e del LFCO '888' per uscita su nastro MT1000
C
      KER(1)=KER(2)=0 ; KER(3)=13
      LFC=X'00414141'
      LFCO=X'00424242'
      CALL RWB(LFC,0,IWR,1,IST)
      IGO(1)=65 ; IGO(2)=84 ; IGO(3)=77
      IGO(4)=84 ; IGO(5)=13
      IK(1)=75 ; IK(2)=13
C
C     Attesa di un comando
C
10    CALL RWB(LFC,1,IWR,4,IST)
C
C     Test per comando 'STOP'
C
      IF(IWR(1).EQ.83.AND.IWR(2).EQ.84.AND.IWR(3).EQ.79
     -.AND.IWR(4).EQ.80) THEN
          CALL RWB(LFC,2,IK,2,IST)
          STOP
      ENDIF
C
C     Test per comando 'STAT' (richiesta dello stato da MINUS)
C     Invio al MINUS dei caratteri 'ATMT' come risposta alla
C     richiesta di stato.
C
      IF(IWR(1).EQ.83.AND.IWR(2).EQ.84.AND.IWR(3).EQ.65
     -.AND.IWR(4).EQ.84) THEN
           CALL RWB(LFC,2,IGO,5,IST)
           GOTO 10
      ENDIF
C
C     Test per comando 'SUBn' (richiesta esecuzione routine)
C
      IF(IWR(1).NE.83.OR.IWR(2).NE.85.OR.IWR(3).NE.66) GOTO 10
      IF(IWR(4).EQ.100) GOTO 150
      IF(IWR(4).EQ.101) GOTO 151
      IF(IWR(4).EQ.106) GOTO 153
      IF(IWR(4).EQ.108) GOTO 154
      GOTO 10
C*
C*  APERTURA FILE NASTRO E REWIND
C*
150   OPEN(LFCO,DEVICE='MT100C',IOSTAT=IER)
```

```
      IF(IER.EQ.0) REWIND(LFC0,IOSTAT=IER)
      IF(IER.EQ.0) THEN
          CALL RWB(LFC,2,IK,2,IST)
      ELSE
          CALL RWB(LFC,2,KER,3,IST)
      ENDIF
      GOTO 10
C*
C*    SKIPFILE DI IXR(1) FILE SU NASTRO MT1000
C*
151   CALL RWB(LFC,1,IWR,1,IST)
      DO K=1,IWR(1)
          SKIPFILE(LFC0,IOSTAT=IER)
          IF(IER.NE.0) GOTO 152
      ENDDO
152   IF(IER.EQ.0) THEN
          CALL RWB(LFC,2,IK,2,IST)
      ELSE
          CALL RWB(LFC,2,KER,3,IST)
      ENDIF
      GOTO 10
C*
C*  scrittura record
C*
153   CALL RWB(LFC,2,IK,2,IST)
      CALL RWB(LFC,1,IWR,4,IST)
      LUN=IWR(1)+IWR(2)*256
      IREC=IWR(3)+IWR(4)*256
      CALL RWB(LFC,2,IK,2,IST)
      CALL RWB(LFC,1,IWR,LUN,IST)
      CALL RWB(LFC0,2,IWR,LUN,IER)
      IF(IER.EQ.0) THEN
          CALL RWB(LFC,2,IK,2,IST)
      ELSE
          CALL RWB(LFC,2,KER,3,IST)
      ENDIF
      GOTO 10
C*
C*  SCRITTURA EOF SU NASTRO MT1000 E
C*  CHIUSURA FILE
C*
154   ENDFILE(LFC0,IOSTAT=IER)
      IF(IER.EQ.0) CLOSE(LFC0,IOSTAT=IER)
      IF(IER.EQ.0) THEN
          CALL RWB(LFC,2,IK,2,IST)
      ELSE
          CALL RWB(LFC,2,KER,3,IST)
      ENDIF
      GOTO 10
      END
$IFT ABORT EXIT
$AS LIB TO @SYSTEM(LIBRERIE)GEPIM.LIB BLOC=N
$AS DIR TO @SYSTEM(LIBRERIE)GEPIM.DIR BLOC=N
$CATALOG
AS AAA TO DEV=TY7EA5
OPTION LOWER
ENVIRONMENT RESIDENT
BUIL ATMT 30 NOM
$EOJ
$$
```

```
C     *-----------------------------------------------------------*
C     !    Routine per trasferire un qualsivoglia numero di     !
C     !    bytes con valori compresi da 0 a 255                 !
C     !                                                         !
C     !    Rev. 1.0  -  15/10/1985  -     E. Fantini            !
C     !                                                         !
C     !    Richiamabile da FORTRAN 77 con formato :             !
C     !              CALL RWB(LFC,MODE,IBUF,NBY,IST)            !
C     !                                                         !
C     !    Parametri :   LFC - Logical file code su cui operare I/O !
C     !                  MODE- Modo in cui deve operare la routine !
C     !                        0 = Open                         !
C     !                        1 = Read                         !
C     !                        2 = Write                        !
C     !                  IBUF- Buffer di I/O dimensionato in bytes !
C     !                  NBY - Numero di bytes da trsferire     !
C     !                  IST - Codice di ritorno che vale :     !
C     !                        0 = Tutto O.K                    !
C     !                        1 = Errore.  0 < MODO > 2        !
C     !                        2 = Errore.  NBY < 1             !
C     !                                                         !
C     !    N.B. - La routine con MODO = 0 deve essere usata una !
C     !           sola volta necessaria per aprire il canale di I/O !
C     *-----------------------------------------------------------*
C
      SUBROUTINE RWB(LFC,MODO,IBUF,NBY,IST)
      INTEGER FCB(16)
      INTEGER*1 IBUF(1)
      IST=0
      IF(MODO.LT.0.OR.MODO.GT.2) IST=1
      IF(MODO.NE.0.AND.NBY.LT.1) IST=2
      IF(IST.NE.0) RETURN
      N=MODO+1
      IF(N.EQ.1) THEN
          DO K=1,16
              FCB(K)=0
          ENDDO
      ENDIF
      FCB(1)=LFC
      FCB(3)=X'22000000'
      IL=LOCF(IBUF)
      FCB(9)=IAND(IL,X'0007FFFF')
      FCB(10)=NBY
      GOTO (100,200,300) N
C
C     Open
C
100   CONTINUE
      INLINE
          LA 1,FCB
          SVC 1,X'30'
      ENDI
      RETURN
C
C     Read
C
200   CONTINUE
      INLINE
          LA 1,FCB
          SVC 1,X'31'
      ENDI
      RETURN
C
```

```
C       Write
C
300     CONTINUE
        INLINE
            LA 1,FCB
            SVC 1,X'32'
        ENDI
        RETURN
        END
```