SOFTWARE DI GESTIONE DELLA "CROCE DEL NORD" TRAMITE SCHEDULA
- PROGRAMMA TOSKED -

C. Bortolotti - A. Cattani
G. Grueff - S. Montebugnoli

Collaborazione grafica V. Albertazzi

Rapporto interno IRA 160/92

word file: tosked.doc

INDICE

INTRODUZIONE

In questo rapporto interno viene descritto il programma TOSKED, nato come naturale evoluzione del programma TOS di gestione acquisizione dati per il radiotelescopio 'Croce del Nord'.

Il programma TOS, descritto in un precedente rapporto interno (IRA 143/91), permette di posizionare correttamente ed automaticamente il nastro alla fine dell'ultimo file registrato (piu` esattamente in mezzo ai due file-mark) e guida l'operatore all'input dei dati relativi all'osservazione; dati che vengono poi scritti nella header all'inizio di ogni file. All'ora di 'start osservazione' inizia l'acquisizione dati che perdura fino all'ora di stop acquisizione. La scrittura di due file-mark riaggiorna la fine fisica del nastro. Una esigenza riscontrata nell'uso di questo programma durante campagne di calibrazioni e/o osservazioni, e` stata quella di potere registrare in successione varie sorgenti senza dovere di volta in volta 'entrare' i dati e puntare le antenne a mano riducendo nello stesso tempo le possibilita` di errori. Avendo gia` a disposizione un sistema di puntamento automatico interfacciato con la consolle e gestito da un personal computer AT, si e` pensato di sviluppare, come naturale evoluzione del programma TOS, un pacchetto software in grado di leggere una schedula contenente una lista di sorgenti, di interagire con il calcolatore di puntamento e di gestire le osservazioni.

-TOSKED-

Il programma TOSKED, al contrario del TOS, non posiziona il nastro alla fine dell'ultimo file scritto, ma inizia a scrivere i files dal numero 1 in poi, in successione, dal punto in cui si trova; e` quindi necessario posizionare manualmente (o tramite appositi programmi) il nastro nel punto desiderato, sfruttando i blocchi software REWIND, SKIP ed HEAD descritti in un altro rapporto interno (IRA 143/91). Il programma, fig.1, parte con la inizializzazione dell'interfaccia IEEE488 ed il settaggio dei primi 4 elementi di ogni array da 2884 bytes che verra` inviato al nastro con TARRAY, ogni 9 secondi. Questi contengono informazioni riservate all'unita` nastro (THORN-EMI mod.8900 1600 Bpi), che sono rispettivamente:

- vet(1)=35 (carattere di controllo #)
- vet(2)=69 (comando di scrittura di 1 blocco)
- vet(3)=69 (MSBytes della lunghezza del record)
- vet(4)=64 (LSBytes della lunghezza del blocco)

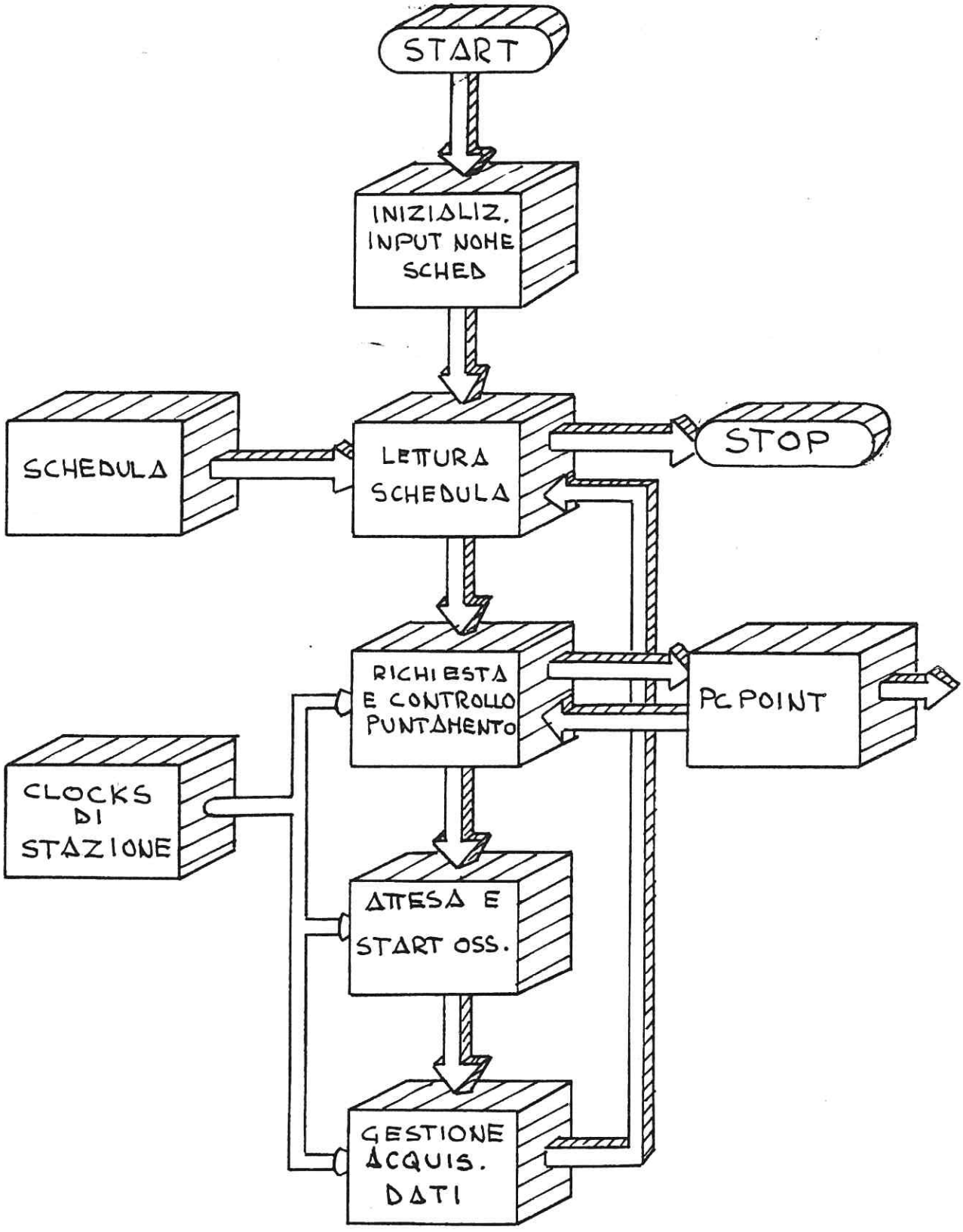
Viene richiesto il numero del nastro ed il nome della schedula poi inizia la fase di lettura dei dati per l'header, relativi alle sorgenti, precedentemente scritti in schedula secondo il formato dell'esempio che segue:

```
start 04 04          (Ora di start operazioni di puntamento)
1                    (Numero file)
3c123                (Nome sorgente 5 chr max)
04 36 35.32          (Ascensione retta gia` precessata)
29 39 22.07          (Declinazione gia` precessata)
calib                (Nome programma 40 chr max)
120                  (Commenti 40 chr max ad esempio flusso)
04 21                (Ora di start acquisizione)
04 51                (Ora di stop  acquisizione)
***                  ('***' indica che non e` l'ultima osservazione)
2
3c295
14 11 02.90
52 14 44.
calib
53.6
13 56
14 26
***
-----------------

DATI HEADER
PER N ALTRE
OSSERVAZIONI

-----------------
N+3
3c409
20 14 03.35
23 33 50.
calib
45.5
19 59
20 29
stop                 ('stop' indica che e` l'ultima osservazione)
```

La fase che segue ha il compito di mandare le informazioni relative alla declinazione da puntare, al personal computer AT di puntamento tramite il blocco software PCPOINT. Se invece il calcolatore di puntamento e` gia` allocato all'altro calcolatore del sistema di acquisizione dati pulsar, PCPOINT ritorna un messaggio MASTER=F (false) e la delta a cui l'antenna punta. Se la delta 'ritornata' e` la stessa richiesta, il programma prosegue, in caso contrario si mette ad aspettare (programma WAIT) tre minuti prima di ripetere la procedura precedente e cosi` via. Si esce da questa fase del programma, se si ottiene l'antenna puntata correttamente sia dal proprio PC (MASTER=T) sia dal PC di acquisizione pulsar (MASTER=F, una delta OK); come conseguenza e` possibile effettuare osservazioni simultanee di pulsar e di survey nel continuo, se si concorda una schedula opportuna. Il

programma si mette quindi a leggere l'orologio siderale di stazione per avviare le procedure di acquisizione dati, all'ora prescritta nella schedula (header). Durante la fase di acquisizione, vengono riportate su video varie informazioni rigurdanti l'acquisizione in corso, tipo indicazioni dell'ora di start, blocchi scritti e quelli da scrivere, nome della schedula, ora (siderale) a cui e` partita l'acquisizione, transito atteso, ora a cui cessera` l'acquisizione ecc., come visibile nella tavola 1 nelle pagine seguenti. All'ora di Stop definita da schedula, TOSKED cessa l'acquisizione e ritorna al blocco di lettura della schedula. Se nella schedula la riga che segue la precedente informazione di ora di stop e` costituita da una serie di 3 asterischi, viene ripresa la procedura sopraddetta, se e` invece presente il messaggio 'Stop', il programma esce per 'schedula terminata'. In fig.2 appare lo schema a blocchi di come il suddetto software si inserisce nel contesto piu` generale del sistema Croce, puntamento e Pulsar. Il modulo PCPOINT di comunicazione tra i calcolatori e quello di puntamento, e` un modulo di gestione della comunicazione via RS232 e fa uso delle routine contenute nella package software NOLIMIT.

I listati dei programmi sopramenzionati, appaiono nelle pagine che seguono.


START
INIZIALIZ. INPUT NOME SCHED
SCHEDULA
LETTURA SCHEDULA
STOP
RICHIESTA E CONTROLLO PUNTAMENTO
PC POINT
CLOCKS DI STAZIONE
ATTESA E START OSS.
GESTIONE ACQUIS. DATI

-TOSKED-

Fig. 1

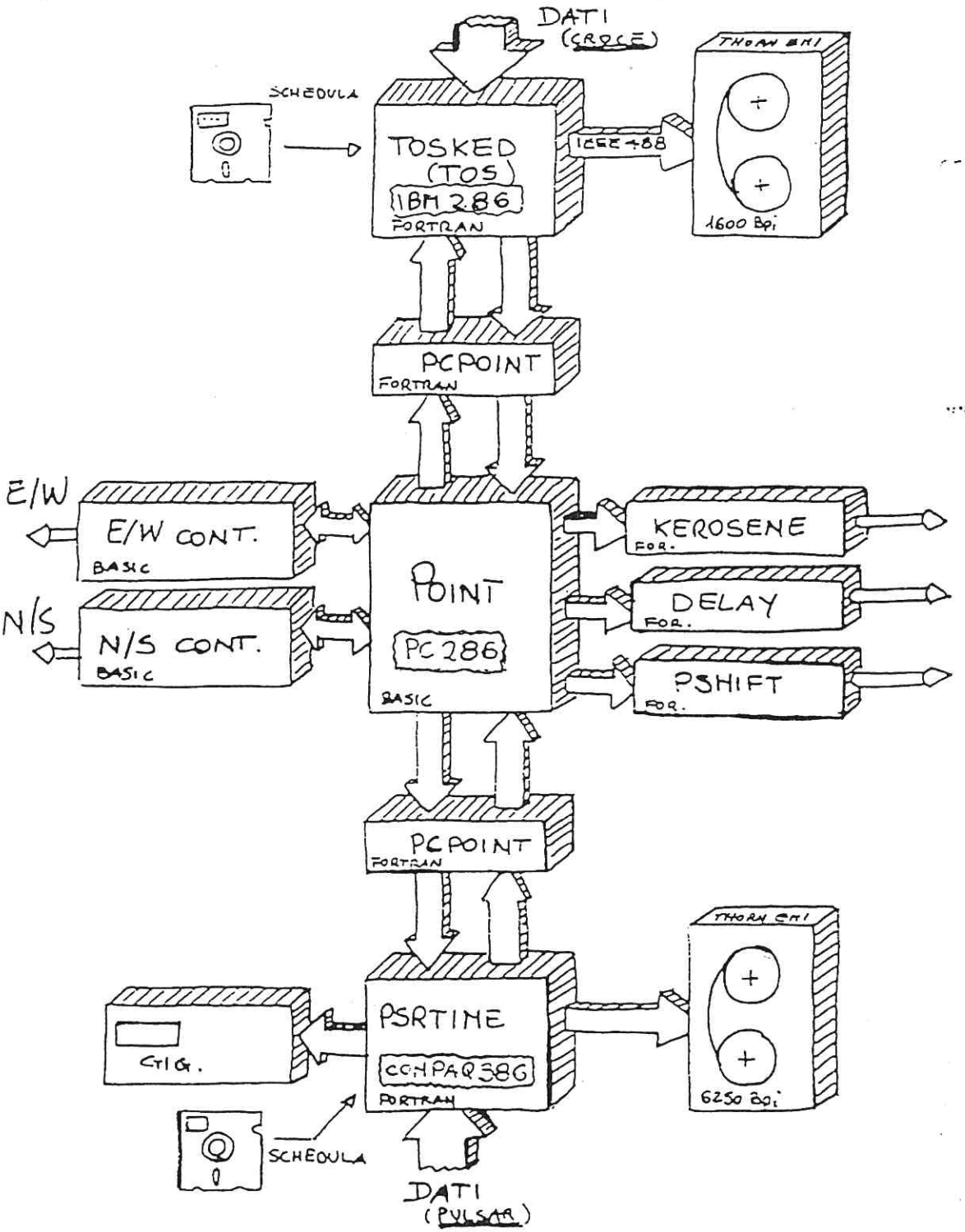

DATI
(CROCE)
SCHEDULA
TOSKED
(TOS)
IBM 286
FORTRAN
ICEE 488
THORN EMI
1600 Bpi
PCPOINT
FORTRAN
E/W
E/W CONT.
BASIC
N/S
N/S CONT.
BASIC
POINT
PC286
BASIC
KEROSENE
FOR.
DELAY
FOR.
PSHIFT
FOR.
PCPOINT
FORTRAN
THORN EMI
PSRTIME
COMPAQ386
FORTRAN
CTIG.
6250 Bpi
SCHEDULA
DATI
(PULSAR)

Fig. 2

```
$DEBUG
$NOTRUNCATE
$DECLARE
c       ========================================
c       ********* SCHED1 OPERATIVE SYSTEM*********
c                 for OSSERV.BAT Program
c       ========================================
c                    name=tosked
c
c          S.Montebugnoli-A.Cattani-G.Grueff
c                       -1991-
c
c       ------------------------------------------------
c       This program has to be linked with:
c       ------------------------------------------------
c       -SUCONV         --> A/D channels conversion
c       -TAPEC          --> Tape control routine
c       -FORT488        --> IEEE 488 routine
c       -CLOCK          --> UT  clock reading
c       -CLOCkSID       --> SID clock reading
c       -NOLIMIT.LIB    --> Nolimit libraries
c       -UCLOCK         --> Fast UT clock reading
c       -PCPOINT        --> Send pointing infor.
c       -PCEND          --> Close serial comunic.
c       ------------------------------------------------

        integer*2 numcom,ibaud,ipar,iwlen,istop,irs,ics,ids,icd,rel3
     * ,ilf,ierr,i1,i2,i3,k
        integer*2 address,level,adr1,adr2,adr3,w,r,key,rel,rel1,rel2
        integer*1 vet(2884),b1,b2,b3,b4,gg,pr
        integer*2 it1,it2,it3,it4,it5,it6,sync
        integer*1 t1,t2,t3,t4,t5,t6,t7,t8,t9,hstart,mstart,hstop,mstop
        integer*4 points,nump,h1,h2
        integer*2 nchr,ltctime,ntape,nfile,delay,num,movant(5)
        integer*1 rcg,rgg,rhh,rmm,rss,rcs,adclk,fs,nast,ki
        integer*4 rec,ski
        integer*2 irow,icolm,ivpage,ifreq,idurr,in,ierror,vrel
        integer*2 i,j,iseg,hst,mst,sst,res,day,cgu,ggu,dgu
        integer*4 aseg
        logical flagi
        real*4    se,decl(5),decl2,decl3,dew,dns,temp,siderr
        real*8  date
        character*5 rinp,t1$,t2$,t3$,t4$,t5$,t6$,t7$,t8$,t9$
        character*5 start$
        character*80 stri(14),out,text(14)
        character*30 idsk,epoch
        character*35 dec$,transit
        character*10 scom$
        character*5 ntap$,nfil$,day$
        character*35 sfitt$,snom$,sked$
        character*1 chr(80),cr,flag
c       ----------------------------------------
        character*1176 text1
        character*12 str(98)
```

```
      equivalence(text1,str(1))
      equivalence(vet(1125),text1)
c     ------------------------------------------


      equivalence(aseg,iseg)
      common/corre/text
      equivalence (out,chr(1))
      equivalence (vet(5),stri(1))

      num=1
      call getarg(num,sked$,ierr)

c     ------------------------------------------------
      numcom=1
      ibaud=6
      ipar=0
      iwlen=7
      istop=1
      irs=0
      ics=0
      ids=0
      icd=0
      ilf=0

     call iopcom(numcom,ibaud,ipar,iwlen,istop,irs,ics,ids
    +,icd,ilf,ierr)
     if(ierr.ne.0)then
      write(*,*)'Error in opening RS232 ',ierr
      stop
     endif
c     ------------------------------------------------

 1    aseg=45056
      adr1=768
      adr2=769
      adr3=770

      w=0
      r=1
      rcg=32
      rgg=33
      rhh=34
      rmm=35
      rss=36
      rcs=37
      ltctime=32
      adclk=38
      fs=0
      res=0
      rel=65
      rel1=64
      rel2=66
      rel3=67
```

```
c       ----  System Reset ----

        vrel=0
        call outport(vrel,rel1)
        call outport(vrel,rel)
        call outport(vrel,rel2)
c       call outport(vrel,rel3)

c       ---- reading time informations from disk ----
        open(unit=15,file='\croce\sider.err')
        read(15,5,err=6)siderr,date
 5      format(f6.2,f10.4)

c       ------------------------------------------

 6      do 2 ki=1,5
        movant(ki)=1
 2      continue


c       ------------------------------------------
c       PC488 interface initializing
        address=1
        level=0
        call initia (address,level)
c       ------------------------------------------
c
c       ---- clear memory buffer ----
        do 10 i=1,2884
        vet(i)=0
 10     continue
c       ---- IEEE 488 first bytes informations for TARRAY ----
c       ---- '#' character ----
        vet(1)=35
c       ---- write one block command ----
        vet(2)=69
c       ---- MSByte's block length ----
        vet(3)=11
c       ---- LSByte's block length ----
        vet(4)=64
c       ---- input mode of operation ----
c       ---- clear screen ----
        nast=0
        call cls
        cr=' '
c       ---- make speaker beep ----
        ifreq=1500
        idurr=5
        call beep2(ifreq,idurr,in)
        write (*,*)'       '
        write (*,15)'NORTHERN CROSS RADIOTELESCOPE DATA ACQUISITION SYST
     &EM'
        write (*,15)'-----------------------------------------------
```

```
     &-'
        write(*,*)
        write(*,*)
 15     format (13x,a53)
        do 20 i=161,639,2
        j=112
 20     call poke (iseg,i,j)
        irow=10
        icolm=28
        ivpage=0
        goto 500
c
c       ************************************
c       ------------End Of Program------------
c       ************************************
 30    call cls
       call wrfmk(b1,b2)
       call decode(b1,b2,b3,b4)
       call beep2(ifreq,idurr,in)
        stop'                                End of program'
c       ---------------------------
 500    write (*,*) '                                                  '

        open(unit=15,file=sked$,status='old')
        write(*,*)' Enter tape number'
        read(*,1771)ntape
   12   continue
 1720   continue
      call cls
c       ******************************
c       *   writing the header -1750- *
c       ******************************
 1750   continue
c       ---- clear screen ----
 1765   continue
c       -------------------------
        ifreq=4000
        idurr=1
        call beep2(ifreq,idurr,in)

c       ---------------------------------------
        stri(1)='-NORTHERN CROSS- RADIOTELESCOPE'
        write (*,*) stri(1)
        write(*,*)
        write(*,*)

        write(*,*)'================================= '
        write(*,*)'WAIT FOR OPERATIONS STARTING TIME '
        write(*,*)'================================= '

c       ---- operation starting time control ----

 329    read(15,'(a5,2i3)',end=3000) start$,hstart,mstart
        write(*,'(1x,a5,2i5.2,5x,12h ESC to exit)')start$,
```

```
     *  hstart,mstart
        write(*,*) ' '
        if (start$.ne.'start') stop' Wrong schedule'
 328    call clocksid(t1,t2,t3,t4,t5,t6,t7,t8,t9)

        hst=t4*10+t5
        mst=t6*10+t7
        sst=t8*10+t9

        irow=9
        icolm=4
        ivpage=0
        call putcur(irow,icolm,ivpage,ierror)

c       -- -O- key control (to go on) --
        call inkey(cr,key,in)
        if(in.eq.1.and.key.eq.24) goto 1760

        write(*,332)hst,mst,sst
 332    format (' Current Time',3i5.2)
        call inkey(cr,key,in)
        if(key.eq.1)stop'Program aborted by user '
        if(hst.ne.hstart.or.mst.ne.mstart) then
          ifreq=6000
          idurr=1
          call beep2(ifreq,idurr,in)
        h1=hstart*3600+(mstart*60)
        h2=hst*3600+(mst*60)+sst
        if (h1.le.h2) then
         h1=h1+86400
        endif
        points=h1-h2

        hst=points/3600
        mst=(points-(hst*3600))/60
        sst=(points-(hst*3600)-(mst*60))

        irow=10
        call putcur(irow,icolm,ivpage,ierror)

        write(*,337)hst,mst,sst
 337    format (' Count-down  ',3i5.2)
c       write(*,341)points
c341    format(' Time to Start  ',1i6)

c       ----------sked informations--------------
        write(*,*)' '
        write(*,*)'            CURRENT SKEDULE'
        write (*,*) '---------------------------------------'
        write(*,*) '            ',sked$
        write (*,*) '---------------------------------------'
        write(*,*)'    '
        write(*,*)' Hit -O- (Operator) key to go on '
```

```
c       -- -O- key control (to go on) --
        call inkey(cr,key,in)
        if(in.eq.1.and.key.eq.24) goto 1760


        go to 328
        endif

 1760   continue
        call cls

        write(*,*)'================================ '
        write(*,*)'POINTING OPERATIONS IN PROGRESS   '
        write(*,*)'================================ '
        write(*,*)'  '

c       ---- tape number ----
        sfitt$='TAPE NUMBER     '

 1771   format(i3)
 1770   write (*,*) sfitt$,ntape
 1790   write (ntap$,1791)ntape
 1791   format(2x,i3.3)
        stri(2)=sfitt$//ntap$

c       ---- file number ----
        sfitt$='FILE  NUMBER     '
       read(15,*,end=30)nfile
 1800   write (*,*) sfitt$,nfile
 1820   write(nfil$,1791)nfile
        stri(3)=sfitt$//nfil$
c       ---- source name ----
        sfitt$='SOURCE  NAME      '
        read(15,'(a35)') snom$
        stri(6)=sfitt$//snom$
        write (*,*) stri(6)
c       ---- getting alfa ----
        sfitt$='ALFA                          '
        read (15,'(a35)') snom$
        transit=snom$
        stri(7)=sfitt$//snom$
        write (*,*) stri(7)
c       ---- getting delta ----
        sfitt$='DELTA                         '
        read(15,'(a35)') snom$
        read (snom$,388)gg,pr,se
 388    format (i2,i3,f5.1)
 398    do 327 ki=1,4
        decl(ki)=gg+pr/60.+se/3600.
        if (ki.eq.2) then
          stri(8)=sfitt$//snom$
          write (*,*) stri(8)
        endif
```

```
 327    continue
        decl(5)=44.57
c  *** decl(5)=ZENITH **********
c       ---- program's name ----
        sfitt$='PROGRAM  NAME                    '
        read (15,'(a35)') snom$
        stri(9)=sfitt$//snom$
        write (*,*) stri(9)
c       ************************************************************


c       ---- send informations ----
c       ----  to pointing  PC  ----
c       pointing informations sent are:
c
c       decl(1)=NS Kerosene pointing
c       decl(2)=EW mechanical pointing
c       decl(3)=NS mechanical pointing
c       decl(4)=NS delays
c       decl(5)=NS phase shifters (ZENITH)

        decl2=decl(2)
        decl3=decl(3)
        write(*,*) decl2
c       write(*,*) decl3

c       ==============================
 1795   call pcpoint(decl,flag,movant)
c       ==============================
        write(*,*)decl(2)

c       --- stop?? ---
        write(*,*)' <ESC> to Quit'

        call inkey(cr,key,in)
        if(in.eq.0)  go to 7771
        if(key.ne.1)then
 1793    call inkey(cr,key,in)
          if(in.ne.0)goto 1793
          goto 7771
        endif
        if(key.eq.1)then
        call pcend(flag)
        call pause(1)
        call clscom(numcom,ierror)
        write(*,*)' Program stopped by user; flag =',flag
        stop
        endif



 7771   dew=abs(decl(2)-decl2)
        dns=abs(decl(3)-decl3)
        write(*,*) dew
```

```
c       ---- flag 1/9 ----
c       -se c'e` errore 6 o 7 (no reply e com.err.) resetta e rientra-

        if(flag.eq.'6'.or.flag.eq.'7') then
          write(*,*)'flag=',flag
          call pcend(flag)
          call pause(1)
          call clscom(numcom,ierror)
          call pause(1)
          call iopcom(numcom,ibaud,ipar,iwlen,istop,irs,ics,ids
     +,icd,ilf,ierr)
          Write(*,*)'Error (6/7) Occured! Waiting for 50 sec.'
          delay=50
          call wait(delay)
          goto 1795
        endif

        if (flag.eq.'1'.or.
     :      flag.eq.'2'.or.
     :      flag.eq.'3'.or.
     :      flag.eq.'4'.or.
     :      flag.eq.'5'.or.
     :      flag.eq.'8'.or.
     :      flag.eq.'9')then
          write (*,*) 'Irrecoverable Error ', flag
          delay=50
          Write(*,*)'Error Occured! Waiting for 50 sec.'
          call wait(delay)
           goto 1795
          endif
c         ------------------------------------------------

c           PARTE DA CONTROLLARE

           if (flag.eq.'B')then
           write(*,*) 'flag = B'

           goto 2223
           write(*,*) '                                    '
           write(*,*) 'NOT POSSIBLE TO HAVE RIGTH POSITION'
           write(*,*) '------------------------------------'
           write(*,*) '                                    '
          write(*,*) 'waiting for 3 min....'
          delay=180
          call wait(delay)
           goto 1795
          endif
c       ------------------------------------------------------

c       ---- flag=A ----
 2223   if(flag.eq.'A'.or.flag.eq.'0') then
         write(*,*)'flag = ',flag
```

```
          if(dew.lt.0.4.and.dns.lt.0.80) goto 995


           write(*,*) 'ANTENNA UNDER PULSAR COMPUTER CONTROL'
           write(*,*) '                                     '
           write(*,*) 'NOT POSSIBLE TO HAVE RIGTH POSITION'
           write(*,*) '-----------------------------------'
           write(*,*) '                                     '
           write(*,*) 'waiting for 3 min.....'

           delay=180
           call wait(delay)

           goto 1795
          endif
c        ************************************************************

 995     continue
c        ---- comments ----
         sfitt$='COMMENTS '
         read (15,'(a10)') scom$
         write(dec$,330)decl,flag
c        write(*,*)decl,flag
 330     format(5f6.2,1x,a1)
         stri(10)=sfitt$//scom$//dec$
         write (*,*) stri(10)
c        ----- observation start ------
         read (15,*) hstart,mstart
         read (15,*)hstop,mstop
         write (stri(11),9020)hstart,mstart
         write(*,9020)hstart,mstart
 9020    format (' Sid. START TIME       ',',             ',2i3.2)
         write (stri(12),9030)hstop,mstop
         write(*,9030)hstop,mstop
 9030    format (' Sid. STOP TIME        ',',             ',2i3.2)

c        ---- correlators Data ----
         day=0
         day$='   '
c        --- 'day' number reading ---
         sync=0
         call uclock (sync,it1,it2,it3,it4,it5,it6)
         cgu=it1/16
         ggu=it2/16
         dgu=it2-ggu*16
         day=(cgu*100)+(dgu*10)+ggu
         write(day$,992) day
 992     format(i3.3)
         write(*,*)day,' ',day$
         idsk='\croce\calib\corr.'//day$
         stri(13)='Corr.xxx NOT present!                  '
         inquire(file=idsk,exist=flagi)
         if(.not.flagi) then
         day=day-1
```

```
        write(day$,992) day
        idsk='\croce\calib\corr.'//day$
        inquire(file=idsk,exist=flagi)
        if(.not.flagi) go to 437
        endif
        stri(13)='Present Corr.'//day$
        write(*,*) idsk
        open(unit=12,file=idsk,status='old ')
        rewind(12)
c       ---- corr. data reading ---
        read(12,431) temp,epoch
 431    format(44x,f4.1/25x,a25//)
        write(stri(14),434)temp,epoch
 434    format('Correlators tested at T=',f4.1,' Deg.  ',a20)
        write(*,*) temp,' ',epoch,' ',idsk
c439    format(f4.1,' ',a21,' ',a30)
        do 432 k=1,96
        read(12,433) i1,i2,i3
 433    format(11x,i3,3x,i3,4x,i3)
        write(str(k),435)i1,i2,i3
 435    format(3(1x,i3))
 432    continue
        write(str(97),'(f6.2)')siderr
        write(str(98),'(f10.4)')date

c       ------------------------------------------------------------
 437    continue
c       ---- points to acquire ----
 9021   format(36x,2i3)
        nump=0
        h1=hstop*3600+(mstop*60)
        h2=hstart*3600+(mstart*60)
        if (h1.le.h2) then
         h1=h1+86400
        endif
        points=1+(h1-h2)/9
c       ------------------------------------
        write(*,*)' Waiting for START TIME'
        write(*,*)' Hit -O- key to go on '


        write(*,*)' <ESC> to quit'
c       ---- start time control ----
 9040   call clocksid (t1,t2,t3,t4,t5,t6,t7,t8,t9)
        hst=t4*10+t5
        mst=t6*10+t7
c       -- -O- key control (to go on) --
        call inkey(cr,key,in)
        if(in.eq.1.and.key.eq.24) goto 9500

        if (hst.eq.hstart.and.mst.eq.mstart) go to 9500
        call inkey(cr,key,in)
        if(in.eq.0)  go to 9040
        if(key.eq.1)then
```

```
c         call wrfmk(b1,b2)
c         call decode(b1,b2,b3,b4)
          stop 'Program aborted'
         endif

c        -- -0- key control (to go on) --
         call inkey(cr,key,in)
         if(in.eq.1.and.key.eq.24) goto 9500

         goto 9040
 9500    continue

c        ************************************
c        ------ write header on tape ---------
c        ************************************
c        ---- read date/clock informations ----
         call clock (t1,t2,t3,t4,t5,t6,t7,t8,t9)

c        --- 'day' number reading ---
         sync=7
         call uclock(sync,it1,it2,it3,it4,it5,it6)
         cgu=it1/16
         ggu=it2/16
         dgu=it2-ggu*16
         day=(cgu*100)+(dgu*10)+ggu

         ifreq=4000
         idurr=3
         call beep2 (ifreq,idurr,in)
         write (stri(4),1822)t1,t2,t3,day
 1822    format('DATE',30x,3i3.2,3x,i3.2)
         ifreq=400
         idurr=3
         call beep2 (ifreq,idurr,in)
         write (stri(5),1823)t4,t5,t6,t7,t8,t9
 1823    format('UT TIME',28x,2i1,':',2i1,':',2i1 )
c        stri(5)=sfitt$//t4$//t5$//t6$//t7$//t8$//t9$
c         write (*,1830)'DATE',t1,t2,t3,day
 1830    format (a5,31x,i2,'/', i2,'/', i2, 3x,i3.2)
c         write (*,1840)'UT TIME',t4,t5,t6,t7,t8,t9
 1840    format (a8,28x,2i1,':',2i1,':',2i1)
         b1=0
         b2=0
         call write (vet,b1,b2)
         call decode (b1,b2,b3,b4)
         call cls

c        ************************************
c        * starting of subconversion routine *
c        ************************************
c
c        read date/clock's information
         call clock (t1,t2,t3,t4,t5,t6,t7,t8,t9)
         t1$=char(t1)
```

```
        t2$=char(t2)
        t3$=char(t3)
        t4$=char(t4)
        t5$=char(t5)
        t6$=char(t6)
        t7$=char(t7)
        t8$=char(t8)
        t9$=char(t9)

        sfitt$='starting      '
        stri(3)=sfitt$//t1$//t2$//t3
        sfitt$='ut time       '
        stri(4)=sfitt$//t4$//t5$//t6$//t7$//t8$//t9$
        write(*,*)'             -NORTHERN CROSS RADIOTELESCOPE DATA ACQU
     :ISITION SYSTEM-'


        write (*,*) '                                -TELESCOPE ON LINE-'
        write (*,*) '                                                   '

c       ---- ON-LINE lamp -ON- ---
        vrel=1
        call outport(vrel,rel)

c       ---- set inverse mode ----
        j=112
        do 1900 i=161,639,2
 1900   call poke (iseg,i,j)
c       -------------------------
        write (*,1996)'start',t1,t2,t3
        write (*,1997)'ut time',t4,t5,t6,t7,t8,t9
 1996   format(29x,a6,3x,i2,'/',i2,'/',i2)
 1997   format(29x,a8,1x,2i1,':',2i1,':',2i1)
        write (*,1999) '------------------------------'
        write (*,1999) ' HIT ALT-F TO STOP ACQUISITION'
        write (*,1999) '------------------------------'
c       ---- set highlight mode ----
        j=15
        do 1998 i=1121,1599,2
 1998   call poke (iseg,i,j)
c       ------------------------
 1999   format(21x,a33)
        write (*,*)'                              tape N.',ntape
        write (*,*)'                              file N.',nfile


c       ----------sked informations--------------
        write(*,*)' '
        write(*,*)' '
        write(*,*)' '
        write (*,2999) '-----------------------------------------'
        write(*,2018) sked$
        write(*,2899) stri(6)
        write(*,2019) hstart,mstart
```

```
        write(*,3009) transit
        write(*,2017)hstop,mstop
        write (*,2999) '-------------------------------------------'
 2018   format(20x,'SKEDULE ',18x,a18)
 2019   format(20x,'OBSERVATION STARTED AT SID.TIME:',4x,i2.2,' ',i2.2)
 3009  format(20x,'EXPECTED SOURCE TRANSIT IS AT:   ',a13)
 2017   format(20x,'OBSERVATION WILL STOP AT SID.TIME:',2x,i2.2,' ',
     #i2.2)
 2899   format(20x,a45)
 2999   format(18x,a45)


c       ---------------------------------------------
c       acquisition routine (9 acquisitions)
c       ---------------------------------------------
c       ---- Record's Counter Reset ----
        rec=0
 2000   ifreq=7000
        idurr=1
        call beep2(ifreq,idurr,in)
c       ---- status control ----
        call sbytes (b1,b2)
c       ----------------------
        call conversion
        rec=rec+1
        nump=nump+1
        write (*,2015)'record N. ',rec,' of',points
 2015   format (28x,a10,i4,a3,i4)
        if (nump.eq.points) go to 2030
 2020   continue
c       ---- "ALT-F" key control ----
        call inkey (rinp,key,in)
        nchr=1
        if (key.eq.33.and.in.eq.2) go to 2030
        go to 2000
c       ************************************
c       ---- End of File  1 fm is  written ----
c       ************************************
        rinp='          '
 2030   continue
        call wrfmk (b1,b2)
        call decode (b1,b2,b3,b4)
        ifreq=4000
        idurr=1
        call beep2 (ifreq,idurr,in)
        write (*,*)'     '
 2035   continue
c       ********************************
c       ---- end of data acquisition ----
c       ********************************
        ifreq=3000
        idurr=1
        call beep2 (ifreq,idurr,in)
```

```
c       ---- End of schedule control ----
        read (15,'(a5)')start$
        if (start$.ne.'stop') go to 1760
        call pcend(flag)
        call clscom(numcom,ierror)


        if(flag.ne.'0') then
          write(*,*)'Failure to end properly ',flag
c       ---- ON-LINE lamp -OFF- ---
        vrel=0
        call outport(vrel,rel)
          stop'    '
        endif

        go to 329

 3000   irow=23
        icolm=1
        ivpage=0
        call putcur(irow,icolm,ivpage,ierror)

c       -- 2th FM is written (END OF SKEDULE)
        call wrfmk (b1,b2)
        call decode (b1,b2,b3,b4)
        ifreq=4000
        idurr=1
        call beep2 (ifreq,idurr,in)


        write(*,3001)'END OF SCHEDULE ',sked$
 3001   format(28x,a17,a35)

c       ---- ON-LINE lamp -OFF- ---
        vrel=0
        call outport(vrel,rel)
c       --- 1 rev. block ---
c       ---(to leave tape between last 2 file marks)---
        ifreq=700
        idurr=6
        call beep2 (ifreq,idurr,in)
        call rv1blk(b1,b2)
        end


        subroutine skip(nskip)

c       ============================================================
c
c       SKIP NUM FILES FORWARD ON THE TAPE UNIT
c
c       ============================================================
        integer*1 b1,b2,b3,b4
        integer*2 ifreq,idurr,in,resp
```

```
        integer*4 nskip
        integer*2 i
c       ------------------------------------
c
        resp=nskip
        ifreq=2500
        idurr=1
        call beep2(ifreq,idurr,in)
      if(resp.le.0)return
        do 5610 i=1,resp
        call fsfwhighsp (b1,b2)
        call decode (b1,b2,b3,b4)
        ifreq=2500
        idurr=1
        call beep2(ifreq,idurr,in)
 5610   continue
      return
        end
```

APPENDICE 1

Vengono riportati per eventuali consultazioni, tutti i listati delle routines usate da tosked, rimandando comunque alla consultazione del rapporto interno IRA 143/91 per una descrizione piu` dettagliata.

```
$DECLARE
c       *********CLOCK'S DATA ACQUISITION ROUTINE**********
c
        subroutine clock(t1,t2,t3,t4,t5,t6,t7,t8,t9)
        integer*2 adr1,adr2,adr3,w,r,i,input
        integer*2 adclk,d(2),wr
        integer*1 c(10),fs,mo(12),t1,t2,t3,t4,t5,t6,t7,t8,t9
        integer*2 sd,day,cgu,dgu,ggu
        integer*1 gm,im,an,dhu,hhu,dmu,mmu,dsu,ssu
        integer*2 rcg,rgg,rhh,rmm,rss,rcs,ltctime,bis,biss
        logical pps
        character*8 string

        data mo/31,28,31,30,31,30,31,31,30,31,30,31/

        call gdate(string)
        read(string,'(6x,i2)') an
        bis=int(an/4)
        biss=an-bis*4
        if(biss.eq.0) then
        data mo/31,29,31,30,31,30,31,31,30,31,30,31/
        endif

        adr1=768
        adr2=769
        adr3=770
        w=0
        r=1
        wr=0
        adclk=38
        rcg=32
        rgg=33
        rhh=34
        rmm=35
        rss=36
        rcs=37
        ltctime=32
        fs=0
c       ----read condition----
        call output (adr3,r)
c       ----clear memory data buffer----
        do 90 i=1,10
        c(i)=0
 90     continue
c       ----sync con il PPS----
 100    call output(adr2,adclk)
        fs=input(adr1)
        pps=btest(fs,0)
        if(pps) go to 100
 200    call output(adr2,adclk)
        fs=input (adr1)
        pps=btest(fs,0)
        if (.not.pps) go to 200
```

```
c       ----ut data/clock information----
        call output (adr2,ltctime)
        call output (adr3,w)
        call output (adr3,r)
c       ----reading data/clock----
        call output (adr2,rcg)
        d(1)= input (adr1)
        call output (adr2,rgg)
        d(2)=input (adr1)
        call output (adr2,rhh)
        c(3)= input (adr1)
        call output (adr2,rmm)
        c(4)=input (adr1)
        call output (adr2,rss)
        c(5)= input (adr1)
        call output (adr2,rcs)
        c(6)= input (adr1)
c       ----data/clock nibble swooping----
        c(3)= ishc(c(3),4)
        c(4)= ishc(c(4),4)
        c(5)= ishc(c(5),4)
        c(6)= ishc(c(6),4)
c       ----clock's display----
c       d(2)=ibclr(d(2),15)
        cgu=d(1)/16
        ggu=d(2)/16
        dgu=d(2)-ggu*16
        day=(cgu*100)+(dgu*10)+ggu
        if (day.gt.366) write (*,*)'clock`s fatal error!!'
        dhu=c(3)/16
        hhu=c(3)-dhu*16
        dmu=c(4)/16
        mmu=c(4)-dmu*16
        dsu=c(5)/16
        ssu=c(5)-dsu*16
        sd=337+mo(2)
        do 290 im=12,1,-1
        sd=sd-mo(im)
        if (day.gt.sd) go to 295
 290    continue
 295    gm=day-sd
c       write (*,*)day
c       write (*,300) gm,im,an,dhu,hhu,dmu,mmu,dsu,ssu
c300    format('*',25x,i2,'/',i2,'/',i2,'/','  ',2i1,':',2i1 ,':',2i1)
c       ----data for main program----
        t1=gm
        t2=im
        t3=an
        t4=dhu
        t5=hhu
        t6=dmu
        t7=mmu
        t8=dsu
        t9=ssu
```

```
        return
        end
```

```
      $NOTRUNCATE
      $DECLARE
      c       ************TAPE CONTROL`S SUBROUTINES************
      c
      c
      c
      c       ..........................................
      c       ****READ ONE BLOCK OF 2880 BYTES LENGTH****
      c       ..........................................
              subroutine read (da)
              integer*2 count1,length,seg,eoi,status
              integer*1 da(2884)
              character*50 cmd
              seg=-1
              eoi=1
              count1=2884
              cmd='UNL UNT LISTEN 1 MTA DATA 35 70|'
              call transm (cmd,status)

              cmd='UNL UNT TALK 1 MLA|'
              call transm (cmd,status)
              call rarray(seg,da(1),count1,length,status)

      c       ----This routine     c       ---- status bytes are da(1) and da(2)
----

              return
              end
      c       ..........................................
      c       ****WRITE ONE BLOCK OF 2880 BYTES LENGTH****
      c       ..........................................
              subroutine write (c,istat1,istat2)
              integer*2 status,istat
              integer*2 seg,count1,count2,length,eoi
              integer*1 istat1,istat2
              integer*1 c(2884)
              character*50 cmd
              eoi=1
              count1=2884
              count2=4
              cmd='UNL UNT LISTEN 1 MTA|'
              call transm (cmd,status)
              call tarray (seg,c(1),count1,eoi,status)
      c       ---- status control ----
              cmd='UNL UNT TALK 1 MLA|'
              call transm(cmd,status)
              length=2
              call receiv(istat,length,status)
              istat2=istat/256
              istat1=(istat-(istat2*256))
              return
```

```
      end
c     .....................................
c     ****SPACE FORWARD 1 BLOCK****
c     .....................................
      subroutine fw1blk (istat1,istat2)
      integer*2 status,istat,length
      integer*1 istat1,istat2
      character*50 cmd
      cmd='UNL UNT LISTEN 1 MTA DATA 35 50|'
      call transm (cmd,status)
c     ---- status control ----
      cmd='UNL UNT TALK 1 MLA|'
      call transm(cmd,status)
      length=2
      call receiv(istat,length,status)
      istat2=istat/256
      istat1=(istat-(istat2*256))
      return
      end
c     .....................................
c     ****REVERSE 1 BLOCK ****
c     .....................................
      subroutine rv1blk (istat1,istat2)
      integer*2 status,istat,length
      integer*1 istat1,istat2
      character*50 cmd
      cmd='UNL UNT LISTEN 1 MTA DATA 35 51|'
      call transm (cmd,status)
c     ---- status control ----
      cmd='UNL UNT TALK 1 MLA|'
      call transm(cmd,status)
      length=2
      call receiv(istat,length,status)
      istat2=istat/256
      istat1=(istat-(istat2*256))
      return
      end
c     .....................................
c     ****REWIND****
c     .....................................
      subroutine rew (istat1,istat2)
      integer*2 istat,length,status
      integer*1 istat1,istat2
      character*50 cmd
      cmd='UNL UNT LISTEN 1 MTA DATA 35 52|'
      call transm (cmd, status)
c     ---- status control ----
      cmd='UNL UNT TALK 1 MLA|'
      call transm(cmd,status)
      length=2
      call receiv(istat,length,status)
      istat2=istat/256
      istat1=(istat-(istat2*256))
      return
```

```
      end
c     ........................................
c     ****WRITE FILE MARK****
c     ........................................
      subroutine wrfmk (istat1,istat2)
      integer*2 status
      integer*2 istat,length
      integer*1 istat1,istat2
      character*50 cmd
      cmd='UNL UNT LISTEN 1 MTA DATA 35 54|'
      call transm (cmd,status)
c     write (*,*)status
c     ---- status control ----
      cmd='UNL UNT TALK 1 MLA|'
      call transm(cmd, status)
      length=2
      call receiv(istat,length,status)
      istat2=istat/256
      istat1=(istat-(istat2*256))
      return
      end
c     ........................................
c     ****STATUS REQUEST****
c     ........................................
      subroutine sbytes(istat1,istat2)
      integer*2 seg,count,status
      integer*2 length,istat
      integer*1 istat1,istat2
      character*50 cmd
      seg=-1
      count=4
      cmd='UNT UNL LISTEN 1 MTA DATA 35 55|'
      call transm (cmd,status)
      cmd='UNT UNL TALK 1 MLA|'
      call transm (cmd,status)
      call receiv (istat,length,status)
      istat2=istat/256
      istat1=(istat-(istat2*256))
      return
      end
c     ........................................
c     ****EXTENDED WRITE 1 BLOCK****
c     ........................................
      subroutine extwr
      integer*2 seg,count2,length,status
      integer*1 sb(4)
      character*50 cmd
      count2=4
      seg=-1
      cmd='UNL UNT LISTEN 1 MTA DATA 35 69|'
      call transm (cmd,status)
c     ================================================
c     call tarray has to be written in these lines,
c     with proper parameters, requested by the program
```

```
c       ================================================
c       ---------------------
c       tape's status control
c       ---------------------
        cmd='UNL UNT TALK 1 MLA|'
        call transm (cmd,status)
        call rarray (seg,sb(1),count2,length,status)
        return
        end
c       ........................................
c       ****EXTENDED READ 1 BLOCK****
c       ........................................
        subroutine extrd
        integer*1 sb(4)
        integer*2 seg,count2,length,status
        character*50 cmd
        seg=-1
        count2=4
        cmd='UNL UNT LISTEN 1 MTA DATA 35 70|'
        call transm (cmd,status)
c       ================================================
c       call rarray has to be written in these lines,
c       with proper parameters, requested by the program
c       ================================================
c       --------------------
c       tape's status control
c       --------------------
        cmd='UNL UNT TALK 1 MLA|'
        call transm (cmd,status)
        call rarray (seg,sb(1),count2,length,status)
        return
        end
c       ........................................
c       ****FILE SEARCH FWD LOW SPEED****
c       ........................................
        subroutine fslowsp (istat1,istat2)
        integer*2 status,istat,length
        integer*1 istat1,istat2
        character*50 cmd
        cmd='UNL UNT LISTEN 1 MTA DATA 35 71|'
        call transm (cmd,status)
c       ---- status control ----
        cmd='UNL UNT TALK 1 MLA|'
        call transm (cmd,status)
        length=2
        call receiv (istat,length,status)
        istat2=istat/256
        istat1=(istat-(istat2*256))
        return
        end
c       ........................................
c       ****FILE SEARCH FWD HIGH SPEED****
c       ........................................
        subroutine fsfwhighsp (istat1,istat2)
```

```
      integer*2 status,istat,length
      integer*1 istat1,istat2
      character*50 cmd
      cmd='UNL UNT LISTEN 1 MTA DATA 35 72|'
      call transm (cmd,status)
c     ---- status control ----
      cmd='UNL UNT TALK 1 MLA|'
      call transm (cmd,status)
      length=2
      call receiv (istat,length,status)
      istat2=istat/256
      istat1=(istat-(istat2*256))
      return
      end
c     ........................................
c     ****FILE SEARCH REV. LOW SPEED****
c     ........................................
      subroutine fsrlowsp (istat1,istat2)
      integer*2 status,istat,length
      integer*1 istat1,istat2
      character*50 cmd
      cmd='UNL UNT LISTEN 1 MTA DATA 35 73|'
      call transm (cmd,status)
c     ---- status control ----
      cmd='UNL UNT TALK 1 MLA|'
      call transm (cmd,status)
      length=2
      call receiv (istat,length,status)
      istat2=istat/256
      istat1=(istat-(istat2*256))
      return
      end
c     ........................................
c     ****FILE SEARCH REV.HIGH SPEED****
c     ........................................
      subroutine fsrhighsp (istat1,istat2)
      integer*2 status,istat,length
      integer*1 istat1,istat2
      character*50 cmd
      cmd='UNL UNT LISTEN 1 MTA DATA 35 74|'
      call transm (cmd,status)
c     ---- status control ----
      cmd='UNL UNT TALK 1 MLA|'
      call transm (cmd,status)
      length=2
      call receiv (istat,length,status)
      istat2=istat/256
      istat1=(istat-(istat2*256))
      return
      end
c     ........................................
c     ****SUB.DECODE****
c     ........................................
c          status bytes decoding
```

```
c       ----------------------------------
c       first byte decoding ....by1
        subroutine decode (by1,by2,by3,by4)
        integer*2 ifreq,idurr,in,irow,icolm,ivpage,i
        integer*1 by1,by2,by3,by4
        integer*1 overr,encerr,b1,b2
        real ccos,co
        encerr=iand(by1,2)

        if(encerr.eq.2) go to 555
        go to 100

c       ---- overal error decoding ----
 555    encerr=iand(by1,60)
        if(encerr.eq.4) go to 10
        if(encerr.eq.8) go to 15
        if(encerr.eq.12) go to 20
        if(encerr.eq.16) go to 25
        if(encerr.eq.20) go to 30
        if(encerr.eq.24) go to 35
        if(encerr.eq.28) go to 40
        if(encerr.eq.32) go to 45
        if(encerr.eq.36) go to 50
        if(encerr.eq.40) go to 55
c       if(encerr.eq.44) go to 60
        if(encerr.eq.48) go to 65
        if(encerr.eq.52) go to 70
        go to 100
 10     write (*,*)'COMMAND NOT RECOGNISED BY SBY'
        go to 100
 15     continue
        ifreq=500
        idurr=90
        call beep2(ifreq,idurr,in)
        write(*,*)'TAPE UNIT NOT READY'
        go to 100
 20     ifreq=500
        idurr=90
        call beep2(ifreq,idurr,in)
        write (*,*)'FILE PROTECTED,WRITE REQUESTED'
        go to 100
 25     write (*,*)'REVERSE REQUESTED AT BOT'
        go to 100
 30     write (*,*)'FILE MARK DETECTED'
        go to 100
 35     write(*,*)'TAPE UNIT DID`T RESPOND TO REWIND OR OFFLINE COMMAND'
        go to 100
 40     write (*,*)'DBY DID NOT SET, TAPE UNIT FAULT'
        goto 100
 45     write (*,*)'IRRECOVERABLE WRITE ERROR'
        go to 100
 50     write (*,*)'IRRECOVERABLE READ ERROR'
        go to 100
 55     continue
```

```
c       write(*,*)'FILE MARK NOT FOUND AT "END" OF FILE SEARCH COMMAND'
        go to 100
c       write (*,*)'RECOVERABLE WRITE ERROR. TAPE SKIPPED, BLOCK
c    &  SUCCESSFULLY REWRITTEN'
        go to 100
 65     write (*,*)'RECOVERABLE READ ERROR. DATA RECOVERED WITHOUT ERROR
     &  AFTER RETRIES'
        go to 100
 70     write (*,*)'DBY FAILED:BLANK TAPE / TOTAL R/W FAILURE'
c       overall error:second byte decoding....by2
 100    by3=encerr
        overr=iand (by2,1)
        if(overr.eq.1) go to 110
        overr=iand(by2,2)
        if(overr.eq.2) go to 115
        if(overr.eq.0.and.encerr.eq.40) go to 118
        overr=iand(by2,4)
        if(overr.eq.4) go to 120
        overr=iand(by2,8)
        if(overr.eq.8) go to 125
        overr=iand(by2,16)
        if(overr.eq.16) go to 130
        overr=iand(by2,32)
        if(overr.eq.32) go to 135
        go to 510
 110    write(*,*)'BOT!'
        go to 510
 115    write (*,*)'END OF TAPE!'
        go to 510
 118    write (*,*)'BLANK TAPE!'
        do 119 i=1,5000
        co=1
        ccos=cos (co)
 119    continue
        go to 700
 120    write (*,*)'TAPE UNIT REJECT!'
        go to 510
 125    continue
        ifreq=5000
        idurr=3
        call beep2 (ifreq,idurr,in)
c       write (*,*) 'FILE MARK DETECTED'
        go to 510
 130    write (*,*)'TAPE FATAL ERROR'
        go to 500
 135    write(*,*)'CONTROL THE BACK-TAPE WRITE PROTECTING RING'
 500    stop
 510    by4=overr
c       if (by3.eq.40 ) go to 520
        go to 700
 520    call cls
c       ifreq=1000
c       idurr=30
c       call beep2(ifreq,idurr,in)
```

```
c       irow=10
c       icolm=29
c       ivpage=0
c       call putcur (irow,icolm,ivpage,in)
c       if (by4.eq.4) write (*,*) 'end of recorded tape (file mark)'
c       write (*,*)'not found!'
c       write (*,*) 'I RETRY AGAIN!!!!!!'
c       ---- rew ----
c       call rew (b1,b2)
 700    return
        end
```

```
$storage:2
$notruncate
      subroutine correct
      character*80 strings(12)
      integer*1 har
      character*1 char,text(80,12)
      common /corre/strings
      equivalence (har,char)
      equivalence(text(1,1),strings(1))
c      call cls
      ivp=0
      do 15 k=1,12
   15 write(*,52)(text(i,k),i=1,78)

   52 format(1x,78a1)
      irow=11
      icol=36
      call putcur(irow,icol,ivp,ierr)
1      call inkey(char,key,nch)
      if(nch.eq.0)goto 1
      if(key.eq.1)goto 100
   50 format(5x,3i5)
      if(nch.eq.2.and.key.eq.72)then
        if(irow.gt.9)irow=irow-1
        call putcur(irow,icol,ivp,ierr)
      goto 1
      endif
      if(nch.eq.2.and.key.eq.80)then
        if(irow.lt.15)irow=irow+1
        call putcur(irow,icol,ivp,ierr)
      goto 1
      endif
      if(nch.eq.2.and.key.eq.75)then
        if(icol.gt.36)icol=icol-1
        call putcur(irow,icol,ivp,ierr)
      goto 1
      endif
      if(nch.eq.2.and.key.eq.77)then
        if(icol.lt.79)icol=icol+1
        call putcur(irow,icol,ivp,ierr)
      goto 1
      endif
      if(ichar(char).ge.32.and.ichar(char).lt.127)then
      call wchar(ivp,char,1,0,0,ierr)
      text(icol,irow-3)=char
      if(icol.lt.79)icol=icol+1
      call putcur(irow,icol,ivp,ierr)
      endif
      goto 1
  100 call putcur(18,1,ivp,ierr)
       return
      end
```

```
$DECLARE
c            *******subroutine name: UCLOCK ********
c       *****************   UT TIME  **********************
c       *********CLOCK'S DATA ACQUISITION ROUTINE**********
c              ------ SYNC. WITH PPS/100 ------
        subroutine uclock(sync,t1,t2,t3,t4,t5,t6)
c       ---- sync=0 ---> sync.with PPS
c       ---- sync=7 ---> sync.with PPS/100
        integer*2 adr1,adr2,adr3,w,r,i,input
        integer*2 adclk,wr,sync
        integer*1 c(10),fs
        integer*1 t1,t2,t3,t4,t5,t6


        integer*2 rcg,rgg,rhh,rmm,rss,rcs,ltctime
        logical pps

        adr1=768
        adr2=769
        adr3=770
        w=0
        r=1
        wr=0
        adclk=38
        rcg=32
        rgg=33
        rhh=34
        rmm=35
        rss=36
        rcs=37
        ltctime=32
        fs=0
c       ----read condition----
        call output (adr3,r)
c       ----clear memory data buffer----
        do 90 i=1,10
        c(i)=0
 90     continue
c       ----sync with PPS/100----
 100    call output(adr2,adclk)
        fs=input(adr1)
        pps=btest(fs,sync)
        if(pps) go to 100
 200    call output(adr2,adclk)
        fs=input (adr1)
        pps=btest(fs,sync)
        if (.not.pps) go to 200
c       ----ut data/clock information----
        call output (adr2,ltctime)
        call output (adr3,w)
        call output (adr3,r)
c       ----reading data/clock----
        call output (adr2,rcg)
        t1= input (adr1)
```

```
      call output (adr2,rgg)
      t2=input (adr1)
      call output (adr2,rhh)
      t3= input (adr1)
      call output (adr2,rmm)
      t4=input (adr1)
      call output (adr2,rss)
      t5= input (adr1)
      call output (adr2,rcs)
      t6= input (adr1)
c     ----data/clock nibble swooping----
c     t1= ishc(t1,4)
c     t2= ishc(t2,4)
      t3= ishc(t3,4)
      t4= ishc(t4,4)
      t5= ishc(t5,4)
      t6= ishc(t6,4)
      return
      end
```

```
$NOTRUNCATE
$DECLARE
c       *********CLOCK'S DATA ACQUISITION ROUTINE**********
c                  ****** SIDERAL TIME *******
        subroutine clocksid(t1,t2,t3,t4,t5,t6,t7,t8,t9)
        integer*2 adr1,adr2,adr3,w,r,i,input
        integer*2 adclk,d(2)
        integer*1 c(10),fs,mo(12),t1,t2,t3,t4,t5,t6,t7,t8,t9
        integer*2 sd,day,cgu,dgu,ggu
        integer*1  gm,im,an,dhu,hhu,dmu,mmu,dsu,ssu
        integer*2 rcg,rgg,rhh,rmm,rss,rcs,ltctime,bis,biss
        logical pps
        character*8 string
        data mo/31,28,31,30,31,30,31,31,30,31,30,31/

        call gdate(string)
        read(string,'(6x,i2)') an
        bis=int(an/4)
        biss=an-bis*4
        if(biss.eq.0) then
        data mo/31,29,31,30,31,30,31,31,30,31,30,31/
        endif


        adr1=768
        adr2=769
        adr3=770
        w=0
        r=1
        adclk=54
        rcg=48
        rgg=49
        rhh=50
        rmm=51
        rss=52
        rcs=53
        ltctime=48
        fs=0
c       ----read condition----
        call output (adr3,r)
c       ----clear memory data buffer----
        do 90 i=1,10
        c(i)=0
 90     continue
c       ----sync con il PPS----
 100    call output(adr2,adclk)
        fs=input(adr1)
        pps=btest(fs,0)
        if(pps) go to 100
 200    call output(adr2,adclk)
        fs=input (adr1)
        pps=btest(fs,0)
        if (.not.pps) go to 200
c       ----ut data/clock information----
```

```
      call output (adr2,ltctime)
      call output (adr3,w)
      call output (adr3,r)
c     ----reading data/clock----
      call output (adr2,rcg)
      d(1)= input (adr1)
      call output (adr2,rgg)
      d(2)=input (adr1)
      call output (adr2,rhh)
      c(3)= input (adr1)
      call output (adr2,rmm)
      c(4)=input (adr1)
      call output (adr2,rss)
      c(5)= input (adr1)
      call output (adr2,rcs)
      c(6)= input (adr1)
c     ----data/clock nibble swooping----
      c(3)= ishc(c(3),4)
      c(4)= ishc(c(4),4)
      c(5)= ishc(c(5),4)
      c(6)= ishc(c(6),4)
c     ----clock's display----
c     d(2)=ibclr(d(2),15)
      cgu=d(1)/16
      ggu=d(2)/16
      dgu=d(2)-ggu*16
      day=(cgu*100)+(dgu*10)+ggu
      if (day.gt.366) write (*,*)'clock`s fatal error!!'
      dhu=c(3)/16
      hhu=c(3)-dhu*16
      dmu=c(4)/16
      mmu=c(4)-dmu*16
      dsu=c(5)/16
      ssu=c(5)-dsu*16
      sd=337+mo(2)
      do 290 im=12,1,-1
      sd=sd-mo(im)
      if (day.gt.sd) go to 295
 290  continue
 295  gm=day-sd
c     write (*,300) gm,im,an,dhu,hhu,dmu,mmu,dsu,ssu
c300  format('*',25x,i2,'/',i2,'/',i2,'/','  ',2i1,':',2i1 ,':',2i1)
c     ----data for main program----
      t1=gm
      t2=im
      t3=an
      t4=dhu
      t5=hhu
      t6=dmu
      t7=mmu
      t8=dsu
      t9=ssu
      return
      end
```

```
$DECLARE
c       **** READING HEADER SUBROUTINE ****
c
        subroutine rdhead
        character*80 text(36)
        integer*2 ivet(1442),i
        integer*1 k
        equivalence(text(1),ivet(3))
        call read (ivet)
        write(*,150)(text(k),k=1,12)
 150    format(1x,a79)
        do 200 i=1,1442
        ivet(i)=0
 200    continue
        return
        end
```

```
$declare
c       lettura numero di file
c       -- nell`header il numero di file e` contenuto --
c       -- in da(202),da(203) e da(204) --
        subroutine nfil(nl)
        integer*2 nl
        character*30 nfile
        character*1  da(2884)
        equivalence(da(202),nfile)
        call read (da)
        read(nfile,10) nl
 10     format(i4)
        return
        end
```

```
      subroutine outsfns(roma_a,roma_d)


      integer*2 roma_a,roma_d
      integer*2 port_a,port_d,strobe,ifreq,idurr,in
      ifreq=1300
      idurr=5
      call beep2(ifreq,idurr,in)

100   continue
      port_a=82
      port_d=81
      strobe=roma_a+128
      call outport(roma_d,port_d)
      call outport(roma_a,port_a)
      call outport(strobe,port_a)
      call outport(roma_a,port_a)
      return
      end
```

```
$DECLARE

        subroutine outport(data,address)
        integer*2 adr1,adr2,adr3,w,r,address,data
        adr1=768
        adr2=769
        adr3=770
        w=0
        r=1
        call output (adr2,address)
        call output(adr1,data)
        call output (adr3,w)
        call output(adr3,r)
        return
        end
```

```
;       MICROSOFT FORTRAN to PC-488 interface package
;-------------------------------------------------------------------
;       Capital Equipment Corporation 1985,1986,1987
;
;       Integer arguments are declared as INTEGER*2, and string
;       arguments are CHARACTER*n variables, where n is from 1 to
;       127.  The strings are terminated with a "|" character
;       (vertical bar).
;


CEC_SEG EQU     0D200H          ; address of PC-488 board


; Data definitions

DGROUP  GROUP   DATA
DATA    SEGMENT WORD PUBLIC 'DATA'
        ASSUME  DS:DGROUP

Desc            DW      ?       ; used to build BASIC-like string descript
r
Ptr1            DW      ?       ; (similar to BASIC Compiler, so long stri
gs
                                ;  can be used)

DATA    ENDS

;-------------------------------------------------------------------
;    FORTRAN interface MACROS
;
;       (note: argnum's are counted R to L here)

GET_ARG_OFFSET  MACRO   argnum          ; get offset in DS of argument
                                        ;  (uses BX, CX)
                                        ; returns offset in BX
        MOV     CX,DS
        MOV     BX,[BP+(argnum-1)*4+8]          ; get segment difference
        SUB     BX,CX
        MOV     CX,4                            ; multiply by 16
        SHL     BX,CL
        MOV     CX,[BP+(argnum-1)*4+6]          ; subtract from offset
        ADD     CX,BX
        MOV     BX,CX

  ENDM

DO_SEGOFS  MACRO argnum                 ; handle seg & ofs (ignore seg)
                                        ; push 2 words on stack
        GET_ARG_OFFSET argnum           ; get offset in DS into BX
        MOV     Desc,BX
        MOV     Ptr1,DS
        MOV     BX,OFFSET DGROUP:Ptr1
```

```
        PUSH    BX
        MOV     BX,OFFSET DGROUP:Desc
        PUSH    BX
  ENDM

CONVERT_SDESC   MACRO   argnum            ; convert string argument to
                                           ; BASIC-like str. desc. format,
                                           ; and replace argument ptr.

        GET_ARG_OFFSET  argnum
        CALL    BUILD                      ; build str. desc.
                                           ; replace argument ptr. on stack
        MOV     CX,DS
        MOV     WORD PTR [BP+(argnum-1)*4+8],CX
        MOV     WORD PTR [BP+(argnum-1)*4+6],OFFSET DGROUP:Desc

   ENDM

ENTRY_LINK      MACRO

        PUSH    BP
        MOV     BP,SP
        PUSH    SI
        PUSH    DI

    ENDM

EXIT_LINK       MACRO   n                  ; exit linkage, n arguments

        POP     DI
        POP     SI
        POP     BP
        RET     4*n

      ENDM
;-----------------------------------------------------------------

; Code

CODE    SEGMENT BYTE PUBLIC 'CODE'
        ASSUME CS:CODE,DS:DGROUP,SS:DGROUP

        PUBLIC  INITIA,TRANSM,RECEIV,SEND,ENTER
        PUBLIC  SPOLL,PPOLL,XMITA,RECVA,DMA
        PUBLIC  TARRAY,RARRAY,DMA2

;-----------------------------------------------------------------

n_routines equ  14
init    dw      0,CEC_SEG
spll    dw      12,CEC_SEG
ppll    dw      15,CEC_SEG
dev     dw      18,CEC_SEG
xa      dw      24,CEC_SEG
```

```
ra      dw      27,CEC_SEG
xmit    dw      30,CEC_SEG
recv    dw      33,CEC_SEG
s       dw      36,CEC_SEG
ent     dw      39,CEC_SEG
dm      dw      42,CEC_SEG
tarr    dw      200,CEC_SEG
rarr    dw      203,CEC_SEG
dm2     dw      206,CEC_SEG

BUILD   PROC    NEAR
; This subroutine finds string length, and puts it in descriptor
        push    es
        push    di
        mov     Ptr1,bx
        mov     di,bx
        mov     ax,ds           ; (find length of string)
        mov     es,ax
        mov     al,124          ; terminator is "|"
        cld
        mov     cx,127
        repnz scasb
        mov     ax,126
        sub     ax,cx
        mov     Desc,ax
        pop     di
        pop     es
        ret
BUILD   ENDP

;-----------------------------------------------------------------
        PUBLIC  PC488S
PC488S  PROC    FAR

        push    bp
        mov     bp,sp
        push    es
        GET_ARG_OFFSET 1
        mov     ax,[bx]                 ; get segment value
        mov     es,ax                   ; test board existence
        mov     ax,0
        cmp     byte ptr es:[50],'C'
        jnz     bad
        cmp     byte ptr es:[51],'E'
        jnz     bad
        cmp     byte ptr es:[52],'C'
        jz      good
bad:    mov     ax,1
        jmp     done
good:   mov     cx,n_routines
        mov     bx,offset init+2
plp:    mov     cs:[bx],es
        add     bx,4
        loop    plp
```

```
done:
        pop     es
        pop     bp
        cld
        ret     4
PC488S  ENDP
;---------------------------------------------------------------

INITIA          PROC    FAR

        ENTRY_LINK
        GET_ARG_OFFSET  2
        PUSH    BX
        GET_ARG_OFFSET  1
        PUSH    BX
        Call     dword ptr init
        EXIT_LINK       2

INITIA          ENDP


TRANSM          PROC    FAR

        ENTRY_LINK
        CONVERT_SDESC   2
        GET_ARG_OFFSET  2
        PUSH    BX
        GET_ARG_OFFSET  1
        PUSH    BX
        PUSH    DS
        POP     ES
        Call     dword ptr xmit
        EXIT_LINK       2

TRANSM          ENDP

RECEIV        PROC    FAR

        ENTRY_LINK
        CONVERT_SDESC   3
        GET_ARG_OFFSET  3
        PUSH    BX
        GET_ARG_OFFSET  2
        PUSH    BX
        GET_ARG_OFFSET  1
        PUSH    BX
        Call     dword ptr recv
        EXIT_LINK       3

RECEIV        ENDP

SEND            PROC    FAR

        ENTRY_LINK
```

```
        CONVERT_SDESC   2
        GET_ARG_OFFSET  3
        PUSH    BX
        GET_ARG_OFFSET  2
        PUSH    BX
        GET_ARG_OFFSET  1
        PUSH    BX
        Call     dword ptr s
        EXIT_LINK       3

SEND            ENDP

SPOLL           PROC    FAR

        ENTRY_LINK
        GET_ARG_OFFSET  3
        PUSH    BX
        GET_ARG_OFFSET  2
        PUSH    BX
        GET_ARG_OFFSET  1
        PUSH    BX
        Call     dword ptr spll
        EXIT_LINK       3

SPOLL           ENDP

PPOLL           PROC    FAR

        ENTRY_LINK
        GET_ARG_OFFSET  1
        PUSH    BX
        Call     dword ptr ppll
        EXIT_LINK       1

PPOLL           ENDP

ENTER           PROC    FAR

        ENTRY_LINK
        CONVERT_SDESC   4
        GET_ARG_OFFSET  4
        PUSH    BX
        GET_ARG_OFFSET  3
        PUSH    BX
        GET_ARG_OFFSET  2
        PUSH    BX
        GET_ARG_OFFSET  1
        PUSH    BX
        Call     dword ptr ent
        EXIT_LINK       4

ENTER           ENDP

XMITA           PROC    FAR
```

```
        ENTRY_LINK
        GET_ARG_OFFSET  3
        PUSH    BX
        GET_ARG_OFFSET  2
        PUSH    BX
        GET_ARG_OFFSET  1
        PUSH    BX
        Call     dword ptr xa
        EXIT_LINK       3

XMITA           ENDP

RECVA           PROC    FAR

        ENTRY_LINK
        GET_ARG_OFFSET  4
        PUSH    BX
        GET_ARG_OFFSET  3
        PUSH    BX
        GET_ARG_OFFSET  2
        PUSH    BX
        GET_ARG_OFFSET  1
        PUSH    BX
        Call     dword ptr ra
        EXIT_LINK       4

RECVA           ENDP

DMA             PROC    FAR

        ENTRY_LINK
        GET_ARG_OFFSET  4
        PUSH    BX
        GET_ARG_OFFSET  3
        PUSH    BX
        GET_ARG_OFFSET  2
        PUSH    BX
        GET_ARG_OFFSET  1
        PUSH    BX
        Call     dword ptr dm
        EXIT_LINK       4

DMA             ENDP

TARRAY          PROC    FAR

        ENTRY_LINK
        DO_SEGOFS       4
        GET_ARG_OFFSET  3
        PUSH    BX
        GET_ARG_OFFSET  2
        PUSH    BX
        GET_ARG_OFFSET  1
```

```
        PUSH    BX
        Call     dword ptr tarr
        EXIT_LINK       5

TARRAY          ENDP

RARRAY          PROC    FAR

        ENTRY_LINK
        DO_SEGOFS       4
        GET_ARG_OFFSET  3
        PUSH    BX
        GET_ARG_OFFSET  2
        PUSH    BX
        GET_ARG_OFFSET  1
        PUSH    BX
        Call     dword ptr rarr
        EXIT_LINK       5

RARRAY          ENDP

DMA2            PROC    FAR

        ENTRY_LINK
        DO_SEGOFS       4
        GET_ARG_OFFSET  3
        PUSH    BX
        GET_ARG_OFFSET  2
        PUSH    BX
        GET_ARG_OFFSET  1
        PUSH    BX
        Call     dword ptr dm2
        EXIT_LINK       5

DMA2            ENDP

;-----------------------------------------
;  Special support routines

        PUBLIC  GPIBSE  ; GPIBSEG

GPIBSE PROC    FAR
        PUSH    BP                      ; Entry linkage
        MOV     BP,SP
        LES     BX,DWORD PTR [BP+6+8]       ; Get array address in ES:BX
        MOV     DX,BX                   ; Make a copy of the offset
        MOV     AX,ES                   ; Get segment
        MOV     CL,4                    ; Form segment+offset/16+1
        SHR     DX,CL
        ADD     AX,DX
        INC     AX
        MOV     DX,BX                   ; Make another copy of the offset
        AND     DX,0FH                  ; Form 16-(offset MOD 16)
        MOV     CX,16
```

```
        SUB     CX,DX
        JNZ     NO_ADJUST               ; (if = 0, adjust to 16)
        MOV     CX,16
NO_ADJUST:
        INC     CX                      ; FORTRAN arrays start at 1 !!
        LES     BX,DWORD PTR [BP+6]       ; Store index value
        MOV     ES:[BX],CX
        LES     BX,DWORD PTR [BP+6+4]     ; Store segment value
        MOV     ES:[BX],AX
        POP     BP                      ; Exit linkage
        RET     12
GPIBSE  ENDP


        PUBLIC  PEEK,POKE,INPUT,OUTPUT

PEEK    PROC    FAR
        PUSH    BP
        MOV     BP,SP
        LES     BX,DWORD PTR [BP+6+4]       ; Get segment
        MOV     CX,0[BX]
        LES     BX,DWORD PTR [BP+6]     ; Get offset
        MOV     BX,0[BX]
        PUSH    ES
        MOV     ES,CX
        MOV     AL,ES:[BX]              ; Get value
        POP     ES
        MOV     AH,0
        POP     BP
        RET     8
PEEK    ENDP

POKE    PROC    FAR
        PUSH    BP
        MOV     BP,SP
        LES     BX,DWORD PTR [BP+6+8]       ; Get segment
        MOV     CX,0[BX]
        LES     BX,DWORD PTR [BP+6]         ; Get value
        MOV     AX,0[BX]
        LES     BX,DWORD PTR [BP+6+4]       ; Get offset
        MOV     BX,0[BX]
        PUSH    ES
        MOV     ES,CX
        MOV     ES:[BX],AL              ; Poke it
        POP     ES
        POP     BP
        RET     12
POKE    ENDP

INPUT   PROC    FAR
        PUSH    BP
        MOV     BP,SP
        LES     BX,DWORD PTR [BP+6]         ; Get port #
        MOV     DX,0[BX]
```

```
        IN      AL,DX                   ; Input byte
        MOV     AH,0
        POP     BP
        RET     4
INPUT   ENDP

OUTPUT  PROC    FAR
        PUSH    BP
        MOV     BP,SP
        LES     BX,DWORD PTR [BP+6+4]       ; Get port #
        MOV     DX,0[BX]
        LES     BX,DWORD PTR [BP+6]         ; Get value
        MOV     AX,0[BX]
        OUT     DX,AL                   ; Output
        POP     BP
        RET     8
OUTPUT  ENDP


;--------

        PUBLIC  SRQ2,SRQ3

SRQ2    PROC    FAR                     ; SRQ TEST FUNCTION (opt. 200 boar
s)
        mov     ax,init+2
        add     ax,400H
        MOV     ES,AX
        MOV     AX,ES:1                 ; get status byte
        AND     AX,2                    ; test SRQ bit
        MOV     AX,0
        JZ      NO_SRQ2
        MOV     AX,1
NO_SRQ2:
        RET
SRQ2    ENDP

SRQ3    PROC    FAR                     ; SRQ TEST FUNCTION (opt. 300 boar
s)
        mov     ax,init+2
        MOV     ES,AX
        MOV     DX,ES:98                ;(get NEC port - note must be >= r
v H)
        ADD     DX,2
        IN      AL,DX                   ; get status byte
        AND     AX,40H                  ; test SRQ bit
        MOV     AX,0
        JZ      NO_SRQ3
        MOV     AX,1
NO_SRQ3:
        RET
SRQ3    ENDP

CODE    ENDS
```

```
        END
```

```
$storage:2

c=====================================================================
        Subroutine NC_TEL_ALLOCATE(telstat)

        implicit none

        integer*4 telstat
        integer*2 comstat

        telstat=0

        call open_com1(comstat)
        if(comstat.ne.0)telstat=1
        return
        end
c=====================================================================
        Subroutine NC_TEL_DEALLOCATE(telstat)

        implicit none

        integer*4 telstat
        character*1 iflag

        telstat=0
        call pcend(iflag)
        if(iflag.ne.'0')telstat=1
        return
        end
c=====================================================================
      subroutine pcpoint(decl,iflag,ndecl)
      integer*1 te(30),tr(30)
      character*1 tex(30),iflag
      character*30 text,tret
      logical master
      dimension decl(5),ndecl(5),dt(5)
      character*3 decnam(5)
      integer*4 icou1,icou2
      equivalence (te(1),text), (tex(1),text),(tr(1),tret)
      data decnam /'ker','e-w','n-s','dly','phs'/
      dt(1)=260.
      dt(2)=4.
      dt(3)=4.
      dt(4)=6.
      dt(5)=6.
c *** 1=ker, 2=ew, 3=ns, 4=delay, 5=phase ****
c *****  default timeout delay *********
      iflag='0'
      num=0
      tret=' '
      text='sta'
      te(4)=13
3      call outcom(1,text,4,nact,ierr)
      call pause(2)
```

```
      call incom(1,tret,4,nact,ierr)
      if(nact.eq.0)then
        iflag='6'
        return
      endif
      if(tret.ne.text)then
c        write(*,*)'Error in communications'
        num=num+1
        if(num.lt.4)goto 3
        write(*,*)(tr(kk),kk=1,10)
c        stop'Communications impossible'
        iflag='7'
        return
      endif
      call pause(2)
      call incom(1,text,30,nact,ierr)
      do  i=1,30
      if(te(i).lt.32)te(i)=32
      enddo
      master=.true.
      read(text,'(2f10.3,3i2)',err=30)dew,dns,ifl1,ifl2,ifl3
      if(ifl3.ne.0)then
        master=.false.
      endif
      write(*,*)'  Master = ',master
      ddew=abs(dew-decl(2))
      ddns=abs(dns-decl(3))
      dt(2)=ddew*8.5+40.
      if(ddew.lt.3.6)dt(2)=220.
      dt(3)=ddns*8.5+40.
      do 10 k=1,5
      if(ndecl(k).ne.0)then
        write(text,'(a3,1x,f7.3)')decnam(k),decl(k)
        te(12)=13
        num=0
    1   call outcom(1,text,12,nact,ierr)
        call pause(2)
        call incom(1,tret,30,nact,ierr)
        if(nact.eq.0)then
          iflag='6'
          return
        endif
        if(tret.ne.text)then
          num=num+1
          if(num.gt.4)then
            iflag='7'
            return
          endif
          goto 1
        endif
        text='go '
        te(4)=13
        call outcom(1,text,4,nact,ierr)
        text=' '
```

```
   13   call timasm(icou1,ierr)
11      call incom(1,text,1,nact,ierr)
        call timasm(icou2,ierr)
        if(ierr.ne.0)icou1=icou1-1573040
        deltt=(icou2-icou1)*65536./1193180.
        if(deltt.gt.dt(k))then
          write(iflag,'(i1)')k
          if (.not.master)then
            if (iflag.eq.'0')then
              iflag='A'
             else
              iflag='B'
            endif
          endif
          return
        endif
        if(nact.eq.0)goto 11
        call pause(2)
        call incom(1,text,30,nact,ierr)
        do j=1,30
        if(te(j).lt.32)te(j)=32
        enddo
        read(text,'(f9.3,f10.3,3i2)',err=30)dew,dens,ifl1,ifl2,ifl3
c        write(*,*)dew,dens,ifl1,ifl2,ifl3
        if(k.eq.1.and.ifl2.eq.1)iflag='8'
        if(k.eq.2)decl(2)=dew
        if(k.eq.3)decl(3)=dens
        if(k.eq.3.and.iflag.eq.'0'.and.ifl1.eq.1)iflag='9'
      endif
  10  continue
      if (.not.master)then
        if (iflag.eq.'0')then
          iflag='A'
         else
          iflag='B'
        endif
      endif
      return
   30 iflag='7'
      if (.not.master)then
        if (iflag.eq.'0')then
          iflag='A'
         else
          iflag='B'
        endif
      endif
      return
      end
c************************************
c
      subroutine pause(num)
      integer*4 icount1,icount2
3     call timasm(icount1,ierr)
    1 call timasm(icount2,ierr)
```

```
      if(ierr.ne.0)icount1=icount1-1573040
      dt=(icount2-icount1)*65536./1193180.
      idt=dt
      if(idt.lt.num) goto 1
      return
      end
c ************************************
c
c   iflag='0'      OK
c   iflag='1'      Kerosene timeout
c   iflag='2'      N-S timeout
c   iflag='3'      E-W timeout
c   iflag='4'      Delay timeout
c   iflag='5'      Phase timeout
c   iflag='6'      No reply
c   iflag='7'      Transmission errors
c   iflag='8'      Kerosene failure
c   iflag='9'      N-S failure
c   iflag='A'      Allocated to other user but pointed as you requested
c   iflag='B'      Allocated to other user and not pointed as you requeste
c
c   Remember that you have to check if dec E-W and N-S are what you want
c****************************************
c
c  decl(1)   Kerosene
c  decl(2)   E-W
c  decl(3)   N-S
c  decl(4)   Delays
c  decl(5)   Phases
c
c*********************************************

      subroutine pcend(iflag)
      integer*1 te(30),tr(30)
      character*1 tex(30),iflag
      character*30 text,tret
      equivalence(tr(1),tret),(te(1),text),(te(1),tex(1))
      num=0
      iflag='0'
      tret=' '
      text='end'
      te(4)=13
  3   call outcom(1,text,4,nact,ierr)
      call pause(2)
      call incom(1,tret,4,nact,ierr)
      if(nact.eq.0)then
        iflag='6'
        return
      endif
      if(tret.ne.text)then
        num=num+1
        if(num.le.4)goto 3
        iflag='7'
        return
```

```
      endif
      return
      end
C=======================================================================
      Subroutine open_com1

      integer*2 comstat
      integer*2 numcom,ibaud,ipar,iwlen,istop,
     :          irs,ics,ids,icd,ilf,ierr
      numcom=1
      ibaud=6
      ipar=0
      iwlen=7
      istop=1
      irs=0
      ics=0
      ids=0
      icd=0
      ilf=0
      call iopcom(numcom,ibaud,ipar,iwlen,istop,irs,ics,ids,
     *icd,ilf,ierr)

      comstat=ierr
      return
      end
```

APPENDICE 2

Viene riportato l'elenco delle routines contenute nella libreria CROCE, quelle precedute da asterisco appartengono al pacchetto NOLIMIT.

```
  ABERR.............astro
* BEEP2.............beep2
* CBKOFF............icomsub3
* CHDIR.............CHDIR
  CHKHRC............CHKHRC
* CIRCA.............CIRCA2
  CLOCKSID..........clocksid
* CLOSIT............CLOSIT
  CLROUT............icomsub3
* CLSATT............CLSATT
* CMPSTG............CMPSTG2
* COMLEN............icomsub3
  CONV..............conv
  CORRE.............correct
* CURDIR............CURDIR
* CURTYP............CURTYP
  DECODE............tapec
* DIRINF............dirinf
  DMA2..............fort488
* DSKINF............DSKINF
  ENTER.............fort488
  EXTRD.............tapec
  FASENS............fasens
  FCBTMPSEG.........OPENUP
* FILDIR............FILDIR
* FILSIZ............FILSIZ
* FLEDEL............FLEDEL3
* FRANDB............FRANDB
* FRNEGB............FRNEGB
* FRNOTB............FRNOTB
* FRORB.............FRORB
* FRROLB............frrolb
* FRRORB............frrorb
* FRSHLB............frshlb
* FRSHRB............frshrb
* FRXORB............FRXORB
  FSFWHIGHSP........tapec
  FSRHIGHSP.........tapec
  FW1BLK............tapec
* GETBUF............GETBUF2
* GETDOS............GETDOS
* GETVID............GETVID
* GSETCL............GSETCL
* GTVRFY............GTVRFY
* HRCCLS............herc10
* INCOM.............icomsub3
* INKEY.............INKEY
  INPUT.............fort488
* INSTR.............INSTR2
* IRANDM............IRANDM
  IUNITOFF..........OPENUP
  JULIAN............astro
* JUSTR.............JUSTR2
* LINEA.............LINEA3
* BCARAY............bcaray
* BEEPIT............BEEPIT
* CBRKON............icomsub3
* CHKDRV............CHKDRV
  CHK_EGA...........CHK_EGA
  CLOCK.............clock
  CLOSFZ............CLOSFZ
  CLRIN.............icomsub3
* CLS...............CLS
* CLSCOM............icomsub3
* CMSTAT............CMSTAT
* CONCAT............CONCAT2
  CONVERSION........suconv
  CORRECT...........correct
* CURDSK............CURDSK
  DATE..............astro
* DIRDIR............DIRDIR
  DMA...............fort488
* DPAGE.............DPAGE
* EGAPRS............EGAPRS
* EQPMNT............EQPMNT
  EXTWR.............tapec
  FCBTMPOFF.........OPENUP
* FEXIST............FEXIST3
* FILINF............FILINF
* FILSTG............FILSTG
* FLEDLZ............FLEDLZ
* FRANDW............FRANDW
* FRNEGW............FRNEGW
  FRNOTW............FRNOTW
* FRORW.............FRORW
* FRROLW............frrolw
* FRRORW............frrorw
* FRSHLW............frshlw
* FRSHRW............frshrw
* FRXORW............FRXORW
  FSLOWSP...........tapec
  FSRLOWSP..........tapec
* GDATE.............GDATE
* GETCUR............GETCUR
* GETSCR............getscr
  GPIBSE............fort488
* GTPATH............gtpath
  HMS...............astro
* HRCOPN............herc10
  INITIA............fort488
* INOCOM............INOCOM
* INSERT............insert
* IOPCOM............icomsub3
* IRNDM2............IRNDM2
  IUNITSEG..........OPENUP
* JUSTL.............JUSTL2
  LINE..............line
* LNEBUF............LNEBUF
```

```
* MEFARC.....MEFARC
* MKDIR.....MKDIR
  MOON.....astro
  MXRECSEG.....OPENUP
  NC_TEL_DEALLOCATE..nctel1
  NFILMX.....OPENUP
  NUTATION.....astro
  OPEN_COM1.....nctel1
* OPNNEW.....OPNNEW
* OUTCOM.....icomsub3
  OUTPORT.....outport
  OUTSFNS.....outsfns
* PACKR.....PACKR2
  PAUSA.....pausa
  PC488S.....fort488
  PCPOINT.....nctel1
* PEEK.....fort488
* PORTIN.....PORTIN2
* PPOLL.....fort488
* PRTSCR.....PRTSCR
* PUTSCR.....putscr
* RCHAR.....rchar
* RDLINE.....RDLINE
* RDRND.....RDRND
  READ.....tapec
  RECVA.....fort488
  REW.....tapec
* RNFLEZ.....RNFLEZ
  RV1BLK.....tapec
  SCLOCK.....sclock
* SCROLL.....SCROLL
* SEGAO1.....segao1
* SELDSK.....SELDSK
* SETBRD.....setbrd
  SPOLL.....fort488
  SRCFCBSEG.....OPENUP
  SRQ3.....fort488
  SUNPOS.....astro
* TEGAIB.....TEGAIB
* TIMEOD.....TIMEOD2
  TMPDTASEG.....OPENUP
  UCLOCK.....uclock
* UPCASE.....UPCASE2
* WCHAR.....wchar2
* WDOT.....wdot
  WRITE.....tapec
* WRTRND.....WRTRND
  XQTDOS.....xqtdos
* XQTPGM.....xqtpgm
* ZFBLNK.....ZFBLNK2
* MIDCHR.....MIDCHR2
* MKPATH.....MKPATH
  MXRECOFF.....OPENUP
  NC_TEL_ALLOCATE...nctel1
  NFIL.....nfil
  NOPEN.....OPENUP
* OPENUP.....OPENUP
* OPNCOM.....OPNCOM
* OPNOLD.....OPNOLD
* OUTLEN.....icomsub3
  OUTPUT.....fort488
* PACKL.....PACKL2
* PAINTA.....PAINTA2
  PAUSE.....nctel1
  PCEND.....nctel1
  PCSTAT.....pcstat
* POKE.....fort488
* PORTOT.....PORTOT2
* PRNASM.....PRNASM
* PUTCUR.....PUTCUR
  RARRAY.....fort488
  RDHEAD.....rdhead
* RDOT.....RDOT
* RDRND2.....rdrnd2
  RECEIV.....fort488
* RENFLE.....RENFLE3
* RMDIR.....RMDIR
* ROTATE.....rotate
  SBYTES.....tapec
* SCREEN.....screen
* SEGAAL.....SEGAAL
* SEGAP1.....SEGAP1
  SEND.....fort488
  SIDG.....astro
  SRCFCBOFF.....OPENUP
  SRQ2.....fort488
* STVRFY.....STVRFY
  TARRAY.....fort488
* TIMASM.....TIMASM
  TMPDTAOFF.....OPENUP
  TRANSM.....fort488
* UCMSTG.....UCMSTG
  WAIT.....wait
* WCHARS.....wchars
  WRFMK.....tapec
* WRTRN2.....wrtrn2
  XMITA.....fort488
  XQTDS2.....XQTDS2
  XQTPM2.....XQTPM2

fort488          Offset: 00000010H  Code and data size: 61cH
  DMA              DMA2               ENTER            GPIBSE
  INITIA           INPUT              OUTPUT           PC488S
```

PEEK POKE PPOLL RARRAY
RECEIV RECVA SEND SPOLL
SRQ2 SRQ3 TARRAY TRANSM
XMITA

wait Offset: 000008c0H Code and data size: 149H
WAIT

nfil Offset: 00000bc0H Code and data size: b8fH
NFIL

BEEPIT Offset: 00000d80H Code and data size: 43H
BEEPIT

CHDIR Offset: 00000e30H Code and data size: 2eH
CHDIR

CHKDRV Offset: 00000ec0H Code and data size: 2c4H
CHKDRV

CHKHRC Offset: 00001030H Code and data size: 2aH
CHKHRC

CHK_EGA Offset: 000010c0H Code and data size: 29H
CHK_EGA

CIRCA2 Offset: 00001150H Code and data size: 3d9H
CIRCA

CLOSFZ Offset: 000019c0H Code and data size: 20H
CLOSFZ

CLOSIT Offset: 00001a40H Code and data size: 157H
CLOSIT

CLS Offset: 00001d30H Code and data size: 2fH
CLS

CLSATT Offset: 00001dc0H Code and data size: 30H
CLSATT

CMPSTG2 Offset: 00001e50H Code and data size: 74H
CMPSTG

CMSTAT Offset: 00001f80H Code and data size: 5dH
CMSTAT

CONCAT2 Offset: 00002040H Code and data size: 71H
CONCAT

CURDIR Offset: 00002170H Code and data size: 39H
CURDIR

CURDSK Offset: 00002210H Code and data size: 1aH

CURDSK

CURTYP Offset: 00002290H Code and data size: 31H
CURTYP

DIRDIR Offset: 00002320H Code and data size: 18eH
DIRDIR

DPAGE Offset: 000025d0H Code and data size: 87H
DPAGE

DSKINF Offset: 00002720H Code and data size: 40H
DSKINF

EGAPRS Offset: 000027c0H Code and data size: 8bH
EGAPRS

EQPMNT Offset: 00002910H Code and data size: a5H
EQPMNT

FEXIST3 Offset: 00002a10H Code and data size: 109H
FEXIST

FILDIR Offset: 00002bc0H Code and data size: 190H
FILDIR

FILINF Offset: 00002e30H Code and data size: f1H
FILINF

FILSIZ Offset: 00002fb0H Code and data size: 12fH
FILSIZ

FILSTG Offset: 00003180H Code and data size: 84H
FILSTG

FLEDEL3 Offset: 000032c0H Code and data size: 109H
FLEDEL

FLEDLZ Offset: 00003470H Code and data size: 1dH
FLEDLZ

FRANDB Offset: 000034f0H Code and data size: 2cH
FRANDB

FRANDW Offset: 00003580H Code and data size: 2cH
FRANDW

FRNEGB Offset: 00003610H Code and data size: 21H
FRNEGB

FRNEGW Offset: 00003690H Code and data size: 21H
FRNEGW

FRNOTB Offset: 00003710H Code and data size: 21H

FRNOTB

FRNOTW Offset: 00003790H Code and data size: 21H
FRNOTW

FRORB Offset: 00003810H Code and data size: 2cH
FRORB

FRORW Offset: 000038a0H Code and data size: 2cH
FRORW

FRXORB Offset: 00003930H Code and data size: 2cH
FRXORB

FRXORW Offset: 000039c0H Code and data size: 2cH
FRXORW

GDATE Offset: 00003a50H Code and data size: 58H
GDATE

GETBUF2 Offset: 00003b20H Code and data size: 1dcH
GETBUF

GETCUR Offset: 00003e50H Code and data size: 9bH
GETCUR

GETDOS Offset: 00003fc0H Code and data size: 2fH
GETDOS

GETVID Offset: 00004050H Code and data size: b2H
GETVID

GSETCL Offset: 00004200H Code and data size: 7eH
GSETCL

GTVRFY Offset: 00004300H Code and data size: 1aH
GTVRFY

INKEY Offset: 00004380H Code and data size: 58H
INKEY

INOCOM Offset: 00004450H Code and data size: 21bH
INOCOM

INSTR2 Offset: 00004800H Code and data size: bdH
INSTR

IRANDM Offset: 00004990H Code and data size: 4eH
IRANDM

IRNDM2 Offset: 00004a50H Code and data size: 93H
IRNDM2

JUSTL2 Offset: 00004b70H Code and data size: 99H

JUSTL

JUSTR2 Offset: 00004cb0H Code and data size: 9fH
JUSTR

LINEA3 Offset: 00004e00H Code and data size: 4b9H
LINEA

LNEBUF Offset: 00005840H Code and data size: 231H
LNEBUF

MEFARC Offset: 00005c00H Code and data size: 70dH
MEFARC

MIDCHR2 Offset: 00006a40H Code and data size: 71H
MIDCHR

MKDIR Offset: 00006b70H Code and data size: 2eH
MKDIR

MKPATH Offset: 00006c00H Code and data size: 92H
MKPATH

OPENUP Offset: 00006d10H Code and data size: 66aH

FCBTMPOFF	FCBTMPSEG	IUNITOFF	IUNITSEG
MXRECOFF	MXRECSEG	NFILMX	NOPEN
OPENUP	SRCFCBOFF	SRCFCBSEG	TMPDTAOFF
TMPDTASEG			

OPNCOM Offset: 00007300H Code and data size: 108H
OPNCOM

OPNNEW Offset: 000074b0H Code and data size: 25H
OPNNEW

OPNOLD Offset: 00007530H Code and data size: 24H
OPNOLD

PACKL2 Offset: 000075b0H Code and data size: 7bH
PACKL

PACKR2 Offset: 000076c0H Code and data size: 88H
PACKR

PAINTA2 Offset: 000077f0H Code and data size: bbcH
PAINTA

PEEK Offset: 000085d0H Code and data size: 33H

POKE Offset: 00008660H Code and data size: 31H

PORTIN2 Offset: 000086f0H Code and data size: 37H
PORTIN

PORTOT2 Offset: 00008780H Code and data size: 34H
PORTOT

PRNASM Offset: 00008810H Code and data size: 33H
PRNASM

PRTSCR Offset: 000088a0H Code and data size: 24H
PRTSCR

PUTCUR Offset: 00008920H Code and data size: e9H
PUTCUR

RDLINE Offset: 00008b60H Code and data size: 5dH
RDLINE

RDOT Offset: 00008c60H Code and data size: 13aH
RDOT

RDRND Offset: 00008f30H Code and data size: 338H
RDRND

RENFLE3 Offset: 00009610H Code and data size: 172H
RENFLE

RMDIR Offset: 00009870H Code and data size: 2eH
RMDIR

RNFLEZ Offset: 00009900H Code and data size: 20H
RNFLEZ

SCROLL Offset: 00009980H Code and data size: 137H
SCROLL

SEGAAL Offset: 00009be0H Code and data size: 56H
SEGAAL

SEGAP1 Offset: 00009c90H Code and data size: 75H
SEGAP1

SELDSK Offset: 00009d80H Code and data size: 25H
SELDSK

STVRFY Offset: 00009e00H Code and data size: 21H
STVRFY

TEGAIB Offset: 00009e80H Code and data size: 44H
TEGAIB

TIMASM Offset: 00009f30H Code and data size: 2bH
TIMASM

TIMEOD2 Offset: 00009fc0H Code and data size: 58H
TIMEOD

UCMSTG Offset: 0000a090H Code and data size: 93H
UCMSTG

UPCASE2 Offset: 0000a1e0H Code and data size: 39H
UPCASE

WRTRND Offset: 0000a280H Code and data size: 3ebH
WRTRND

XQTDS2 Offset: 0000aa80H Code and data size: 33aH
XQTDS2

XQTPM2 Offset: 0000ae10H Code and data size: 2f4H
XQTPM2

ZFBLNK2 Offset: 0000b120H Code and data size: 35H
ZFBLNK

gtpath Offset: 0000b1b0H Code and data size: 5H
GTPATH

setbrd Offset: 0000b270H Code and data size: 55H
SETBRD

wdot Offset: 0000b340H Code and data size: 186H
WDOT

segao1 Offset: 0000b6c0H Code and data size: 54H
SEGAO1

bcaray Offset: 0000b790H Code and data size: 2c6H
BCARAY

screen Offset: 0000bd10H Code and data size: 1c5H
SCREEN

frshlw Offset: 0000c090H Code and data size: 1dH
FRSHLW

frshlb Offset: 0000c110H Code and data size: 1dH
FRSHLB

frshrw Offset: 0000c190H Code and data size: 1dH
FRSHRW

frshrb Offset: 0000c210H Code and data size: 1dH
FRSHRB

frrolw Offset: 0000c290H Code and data size: 1dH
FRROLW

frrolb Offset: 0000c310H Code and data size: 1dH
FRROLB

frrorw Offset: 0000c390H Code and data size: 1dH
FRRORW

frrorb Offset: 0000c410H Code and data size: 1dH
FRRORB

insert Offset: 0000c490H Code and data size: e0H
INSERT

rchar Offset: 0000c6a0H Code and data size: 11fH
RCHAR

herc10 Offset: 0000c8e0H Code and data size: 1a33H
HRCCLS HRCOPN

wrtrn2 Offset: 0000eab0H Code and data size: 230H
WRTRN2

rdrnd2 Offset: 0000ef40H Code and data size: 230H
RDRND2

wchar2 Offset: 0000f3d0H Code and data size: 141H
WCHAR

wchars Offset: 0000f6a0H Code and data size: 1aaH
WCHARS

beep2 Offset: 0000fa30H Code and data size: 6dH
BEEP2

rotate Offset: 0000fb00H Code and data size: 316H
ROTATE

putscr Offset: 00010190H Code and data size: 2e4H
PUTSCR

getscr Offset: 00010800H Code and data size: 2a4H
GETSCR

xqtpgm Offset: 00010dc0H Code and data size: 2ddH
XQTPGM

xqtdos Offset: 00011090H Code and data size: 30bH
XQTDOS

dirinf Offset: 000113b0H Code and data size: f4H
DIRINF

icomsub3 Offset: 00011540H Code and data size: f70H
CBKOFF CBRKON CLRIN CLROUT
CLSCOM COMLEN INCOM IOPCOM
OUTCOM OUTLEN

outport Offset: 000123b0H Code and data size: 6eH

OUTPORT

outsfns Offset: 00012550H Code and data size: 8cH
OUTSFNS

pausa Offset: 00012720H Code and data size: 9fH
PAUSA

pcstat Offset: 00012940H Code and data size: 300H
PCSTAT

astro Offset: 00012ec0H Code and data size: 30ddH
ABERR DATE HMS JULIAN
MOON NUTATION SIDG SUNPOS

sclock Offset: 00018fc0H Code and data size: 256H
SCLOCK

correct Offset: 000193e0H Code and data size: 59fH
CORRE CORRECT

fasens Offset: 00019820H Code and data size: 2ebH
FASENS

line Offset: 00019e10H Code and data size: f22H
LINE

rdhead Offset: 0001b440H Code and data size: b98H
RDHEAD

nctel1 Offset: 0001b610H Code and data size: a8bH
NC_TEL_ALLOCATE NC_TEL_DEALLOCATE OPEN_COM1
PAUSE PCEND PCPOINT

uclock Offset: 0001c7d0H Code and data size: 25eH
UCLOCK

conv Offset: 0001cc00H Code and data size: 20aH
CONV

clock Offset: 0001d0b0H Code and data size: 400H
CLOCK

clocksid Offset: 0001d810H Code and data size: 3f8H
CLOCKSID

suconv Offset: 0001df70H Code and data size: 12d4H
CONVERSION

tapec Offset: 0001ed00H Code and data size: 1388H
DECODE EXTRD EXTWR FSFWHIGHSP
FSLOWSP FSRHIGHSP FSRLOWSP FW1BLK
READ REW RV1BLK SBYTES
WRFMK WRITE

BIBLIOGRAFIA

1) Rapporto interno IRA 143/91 "Software di gestione del sistema di acquisizione dati della 'Croce del Nord', per IBM 286 -PROGRAMMA TOS-
S. Montebugnoli, A. Cattani, G. Grueff

2) NOLIMIT
M I E I F I Enviroment Inc.

NOTE