Consiglio Nazionale delle Ricerche

ISTITUTO DI
ELABORAZIONE DELLA
INFORMAZIONE

Pisa

Contratto di collaborazione tecnico-scientifica
Alenia-GAT/IEI-CNR

Progetto software del pacchetto per l'ispezione US

E. Bozzi, E. Fantini, A. Marchetti, A. Ribolini

Nota Interna B4-13

Maggio1993

Contratto di collaborazione tecnico-scientifica Alenia-GAT/IEI-CNR

Progetto software del pacchetto per l'ispezione US

Edoardo Bozzi, Enrico Fantini, Andrea Marchetti, Alberto Ribolini

Istituto di elaborazione della Informazione-CNR

Introduzione

In questo lavoro sono riportati i programmi sorgente di alcuni dei moduli che compongono il pacchetto sofwtare per l'ispezione a ultrasuoni descritto in [1].

Ciascun modulo del pacchetto è costituito da un programma di tipo A, che risiede sull'unita di controllo S3, e da un programma di tipo B, che risiede sul calcolatore della stazione di lavoro [2]. Il programma di tipo A provvede al controllo dello spostamento del braccio meccanico del robot, mentre il programma di tipo B svolge le funzioni:di interfacciamento, controllo e immagazzinamento dei dati.

Prima di richiamare il pacchetto software è necessario mettere il robot IRB2000 della stazione in condizione di reset. Questa operazione dev'essere eseguita una sola volta all'atto dell'accensione iniziale ed è costituita dalle seguenti fasi [3].

Mediante l'interruttore generale posto sul fianco destro dell'unità di controllo S3 si alimenta il sistema, che provvede a eseguire le operazioni di autodiagnosi.

Dopo circa 30 secondi si illumina la spia 7 sul pannello di comando di S3 (v. fig. 1); l'utente deve quindi:

premere il pulsante 8 per il ripristino del sistema: la spia 8 si spenge;

premere il pulsante 2, per togliere il sistema dalla condizione di STANDBY: la spia 2 si accende e la spia 3 inizia a lampeggiare;

premere il pulsante 3 per sincronizzare il braccio meccanico: la spia 3 si spenge e sul monitor di S3 compare il programma 0..

Passando dalla condizione di STANDBY a quella di sincronizzazione, il braccio meccanico si muove per raggiungere la posizione di reset: è opportuno assicurarsi in precedenza che non ci siano ostacoli sulla traiettoria, oppure spostare il braccio mediante il joy-stick della tastiera di S3: questa operazione dev'essere compiuta dopo aver premuto il pulsante 2 e prima di premere il pulsante 3..

Il pacchetto software viene richiamato digitando sulla tastiera del calcolatore il codice ROBOT e ciascun modulo viene attivato digitando il corripondente numero di codice sul menù (v. fig. 2); al termine dell'esecuzione viene ripresentato il menù per una ulteriore scelta.

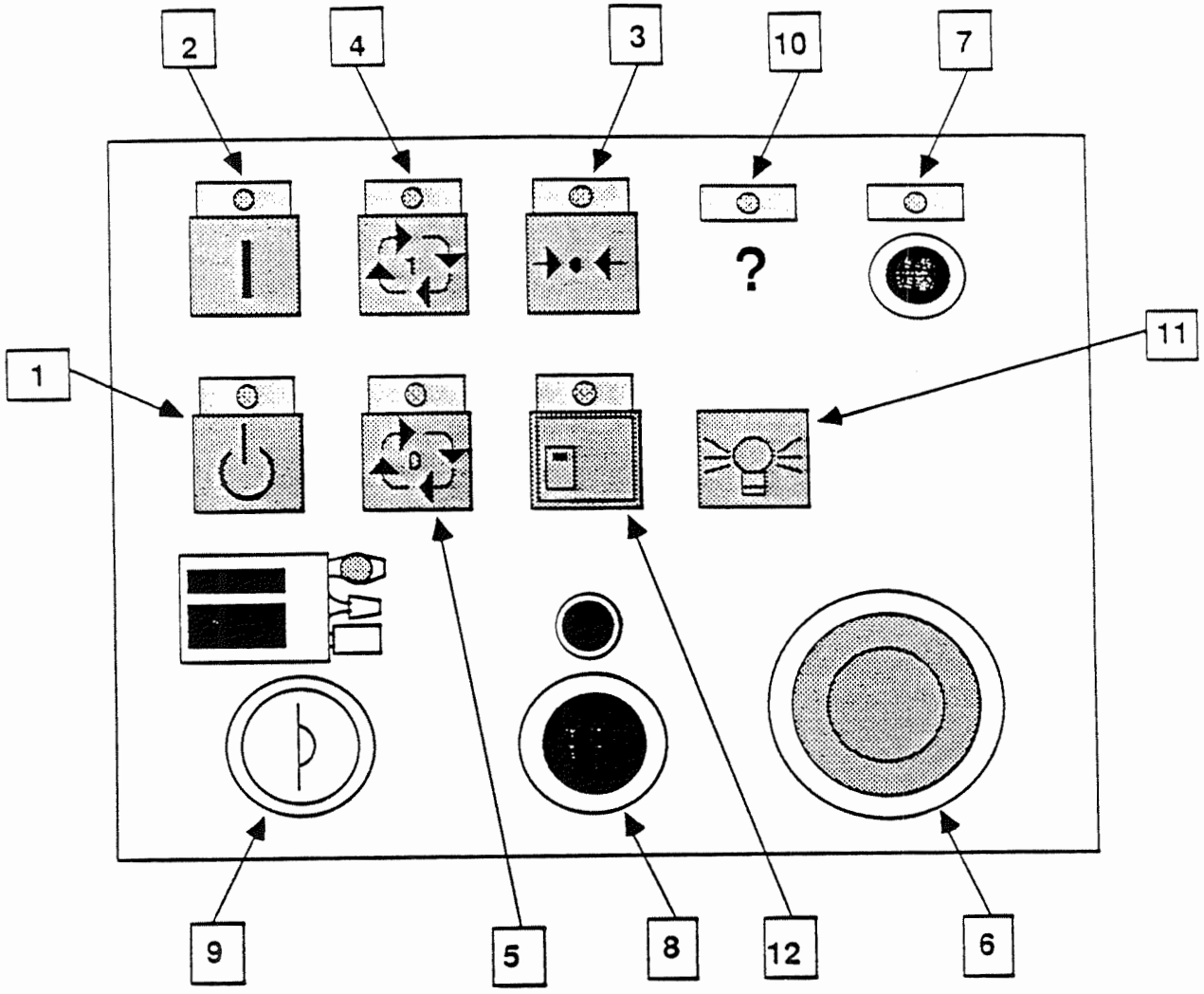

Fig. 1: Pannello di controllo dell'unità S3 del sistema IRB2000

Programmi eseguibili per gestione ROBOT

0. MONISP = spostamento della sonda nello spazio
1. MONISPC = spostamento della sonda nello spazio con correz. di coord.
2. BEMOVI4 = spostamento della sonda parallela all'asse z
3. SULFA35.1 = scansione di un pezzo piano in xy
4. SCA_SF = scansione di un pezzo piano senza limitazioni di coordinate
5. ONDA1 = scansione ad onda completa
6. ONDAF1 = scansione ad onda completa con creazione di file dati " ONDC.DA
7. INCFR = scans. di una superf. inclinata con modifica del frame
8. PIAORI = scans. di una superf. inclinata con modifica del TCP
9. OCINCFR = onda completa per una superf. inclinata con modifica del
10. BEPI7 = scansione del control tab
11. BEPICAD5 = scansione del control tab su dati rilevati automaticamente
12. CILI = scansione del cilindro nel piano xy
13. BERU1 = scansione del cilindro solidale con il TCP
14. ONDABERU = onda completa per il cilindro solidale con il TCP

20. Ritorno a DOS

Programmi eseguibili per gestione ROBOT

0. MONISP = spostamento della sonda nello spazio
1. MONISPC = spostamento della sonda nello spazio con correz. di coord.
2. BEMOVI4 = spostamento della sonda parallela all'asse z
3. SULFA35.1 = scansione di un pezzo piano in xy
4. SCA_SF = scansione di un pezzo piano senza limitazioni di coordinate
5. ONDA1 = scansione ad onda completa

7. INCFR = scans. di una superf. inclinata con modifica del frame
8. PIAORI = scans. di una superf. inclinata con modifica del TCP
9. OCINCFR = onda completa per una superf. inclinata con modifica del
10. BEPI7 = scansione del control tab
11. BEPICAD5 = scansione del control tab su dati rilevati automaticamente
12. CILI = scansione del cilindro nel piano xy
13. BERU1 = scansione del cilindro solidale con il TCP
14. ONDABERU = onda completa per il cilindro solidale con il TCP

20. Ritorno a DOS

Fig. 2: Menù di selezione del pacchetto software

1.Programmi di tipo A

Nelle pagine seguenti sono riportati i seguenti listati.

MOVI1000.IRB
SUL3515.IRB
BEPI4014.IRB
FR1008.IRB
BEP3616.IRB

Questi programmi sono scritti usando il linguaggio ARLA [4]; ciascun programma viene scritto e compilato sul calcolatore della stazione e successivamente trasferito su S3 attraverso la porta di comunicazione seriale. Il programma trasferito viene immagazzinato nella memoria permanente di S3 e individuato dal solo codice numerico; il programma .IRB viene richiamato dal programma di tipo B in esecuzione.

Programma 1000

```
        UNIT = METRIC
0010  COMMENT  SCANSIONE NON PIANA CON PARAMETRI DEFINITI IN
MODO REMOTO;
0020   V = 500 MAX = 1000
0030  ROBOT COORD
0040   TCP 0
0050   FRAME 0
0060   SUCTRL
0070   STOP ;
.
```

Programma 3515

```
      UNIT = METRIC
0010  COMMENT  SCANSIONE NON PIANA CON PARAMETRI DEFINITI IN
MODO REMOTO;
0020   V = 500 MAX = 1000
0030  ROBOT COORD
0040   TCP 0
0050   FRAME 0
0060   SUCTRL
0080  COMMENT *SONDA ALL'ORIGINE  MANUAL.*;
0090   LET R1 = 0
0100  COMMENT REGISTRO COORD. X ;
0110   LET R2 = 0
0120  COMMENT REGISTRO COORD. Y ;
0130   LET R3 = 0
0140  COMMENT REGISTRO COORD. Z ;
0150   LET R9 = 0
0160  COMMENT CONTATORE MISURE X;
0170   LET R10 = 0
0180  COMMENT CONTATORE       Y  ;
0190   LET R11 = 0
0200  COMMENT CONTATORE       Z;
0210   LET R13 = 5
0220   LET R14 = 0
0230   LET R4 = 0
0240  COMMENT REGISTRO PASSO INCR.;
0250   LET R6 = 0
0260  COMMENT REGISTRO MISURE  X;
0270   LET R7 = 0
0280  COMMENT REGISTRO        Y;
0290   LET R8 = 0
0300  COMMENT REGISTRO        Z;
0310   SUCTRL
0320  COMMENT RICHIESTA DI P,PX,PY,PZ ;
0330   LET R9 = R6
0340  COMMENT CONTATORE       X ;
0350   LET R10 = R7
0360  COMMENT CONTATORE       Y ;
0370   LET R11 = R8
0380  COMMENT CONTATORE       Z ;
0390  COMMENT *SONDA ALLE COORDINATE DI SCANSIONE*;
0400   SUCTRL
0550   CALL PROG 4014 REP 99
0560   CALL PROG 4014 REP 99
.
```

Programma 4014

```
      UNIT = METRIC
0010  COMMENT  SCANSIONE NON PIANA CON PARAMETRI DEFINITI IN
MODO REMOTO;
0015  POS V = 100.0 % STORE POSITION 20
0020  LET R13 = 0
0030  LET R1 = 0
0040  LET R2 = 0
0050  LET R3 = 0
0055  LET R9 = R6
0060  POS V = 100.0 % FINE POSITION 20 OFFSET SC = 0.1
        X = (R1) Y = (R2) Z = (R3)
0070  SET OUTP 1
0080  RESET OUTP 1
0090  SUCTRL CONT
0100  LET R1 = R1 - R4
0110  LET R9 = R9 -1
0120  JUMP TO 0060  IF R9 > 0
0125  SUCTRL
0130  RETURN
.
```

Programma1008

```
      UNIT = METRIC
0010  COMMENT  SCANSIONE CON PARAMETRI DEFINITI IN MODO
REMOTO;
0020   V = 500 MAX = 1000
0030  ROBOT COORD
0040   TCP 0
0050   FRAME 0
0060   POS V = 100.0 % REFPOINT OFF #0001
0065   SUCTRL
0070   POS V = 100.0 % FINE POSITION 50 OFFSET X = 0.0 Y = 0.0 Z = 0.0
0087   POS V = 100.0 % STORE POSITION 50
0088   SUCTRL
0089   FRAME 1
0090  COMMENT *SONDA ALL'ORIGINE  MANUAL.*;
0100   LET R1 = 0
0110  COMMENT REGISTRO COORD. X ;
0120   LET R2 = 0
0130  COMMENT REGISTRO COORD. Y ;
0140   LET R3 = 0
0150  COMMENT REGISTRO COORD. Z ;
0160   LET R9 = 0
0170  COMMENT CONTATORE MISURE X;
0180   LET R10 = 0
0190  COMMENT CONTATOREY  ;
0200   LET R11 = 0
0210  COMMENT CONTATOREZ;
0220   LET R4 = 0
0230  COMMENT REGISTRO PASSO INCR.;
0240   LET R6 = 0
0250  COMMENT REGISTRO MISURE  X;
0260   LET R7 = 0
0270  COMMENT REGISTRO Y;
0280   LET R8 = 0
0290  COMMENT REGISTRO Z;
0300   SUCTRL
0310  COMMENT RICHIESTA DI P,PX,PY,PZ ;
0320   LET R9 = R6
0330  COMMENT CONTATORE   X ;
0340   LET R10 = R7
0350  COMMENT CONTATORE   Y ;
0360   LET R11 = R8
0370  COMMENT CONTATORE   Z ;
0380  COMMENT *SONDA ALLE COORDINATE DI SCANSIONE*;
0390   POS V = 100.0 % FINE POSITION 50 OFFSET SC = 0.1
```

```
  X = (R1) Y = (R2) Z = (R3)
0400  SET OUTP 1
0405  SUCTRL CONT
0410  RESET OUTP 1
0420  LET R1 = R1 - R4
0430  LET R9 = R9 -1
0440  JUMP TO 0390  IF R9 > 0
0450  SUCTRL
0460  LET R2 = R2 + R4
0470  LET R10 = R10 -1
0480  LET R1 = 0
0490  LET R9 = R6
0500  JUMP TO 0390  IF R10 > 0
0510  STOP ;
.
```

Programma 3016

```
      UNIT = METRIC
0010  COMMENT  SCANSIONE NON PIANA CON PARAMETRI DEFINITI IN
MODO REMOTO;
0020   V = 500 MAX = 1000
0030  MODRECT COORD
0040   TCP 0
0050   FRAME 0
0060   POS V = 100.0 % FINE #0001
0070   POS V = 100.0 % STORE POSITION 20
0080  COMMENT *SONDA ALL'ORIGINE  MANUAL.*;
0090   LET R1 = 0
0100  COMMENT REGISTRO COORD. X ;
0110   LET R2 = 0
0120  COMMENT REGISTRO COORD. Y ;
0130   LET R3 = 0
0140  COMMENT REGISTRO COORD. Z ;
0150   LET R9 = 0
0160  COMMENT CONTATORE MISURE X;
0170   LET R10 = 0
0180  COMMENT CONTATORE  Y  ;
0190   LET R11 = 0
0200  COMMENT CONTATORE  Z;
0210   LET R13 = 5
0220   LET R14 = 0
0230   LET R4 = 0
0240  COMMENT REGISTRO PASSO INCR.;
0250   LET R6 = 0
0260  COMMENT REGISTRO MISURE  X;
0270   LET R7 = 0
0280  COMMENT REGISTROY;
0290   LET R8 = 0
0300  COMMENT REGISTROZ;
0310   SUCTRL
0320  COMMENT RICHIESTA DI P,PX,PY,PZ ;
0330   LET R9 = R6
0340  COMMENT CONTATORE X ;
0350   LET R10 = R7
0360  COMMENT CONTATORE Y ;
0370   LET R11 = R8
0380  COMMENT CONTATORE Z ;
0390  COMMENT *SONDA ALLE COORDINATE DI SCANSIONE*;
0400   POS V = 100.0 % FINE POSITION 20 OFFSET SC = 0.1
X = (R1) Y = (R2) Z = (R3)
0410   SET OUTP 1
```

```
0420  RESET OUTP 1
0430  SUCTRL CONT
0440  LET R1 = R1 - R4
0450  LET R9 = R9 -1
0460  JUMP TO 0400  IF R9 > 0
0470  SUCTRL
0480  LET R2 = R2 + R4
0490  LET R10 = R10 -1
0500  LET R1 = 0
0510  LET R9 = R6
0520  JUMP TO 0400  IF R10 > 0
0530  POS V = 100.0 % FINE #0002
0540  POS V = 100.0 % STORE POSITION 20
0550  CALL PROG 3014
0560  POS V = 100.0 % FINE #0003
0570  POS V = 100.0 % STORE POSITION 20
0580  CALL PROG 3014
0590  POS V = 100.0 % FINE #0004
0600  POS V = 100.0 % STORE POSITION 20
0610  CALL PROG 3014
0620  POS V = 100.0 % FINE #0005
0630  POS V = 100.0 % STORE POSITION 20
0640  CALL PROG 3014
0650  POS V = 100.0 % FINE #0006
0660  POS V = 100.0 % STORE POSITION 20
0670  CALL PROG 3014
0680  POS V = 100.0 % FINE #0007
0690  POS V = 100.0 % STORE POSITION 20
0700  CALL PROG 3014
0710  POS V = 100.0 % FINE #0008
0720  POS V = 100.0 % STORE POSITION 20
0730  CALL PROG 3014
0740  POS V = 100.0 % FINE #0009
0750  POS V = 100.0 % STORE POSITION 20
0760  CALL PROG 3014
0770  POS V = 100.0 % FINE #0010
0780  POS V = 100.0 % STORE POSITION 20
0790  CALL PROG 3014
0800  POS V = 100.0 % FINE #0011
0810  POS V = 100.0 % STORE POSITION 20
0820  CALL PROG 3014
0830  POS V = 100.0 % FINE #0012
0840  POS V = 100.0 % STORE POSITION 20
0850  CALL PROG 3014
0860  POS V = 100.0 % FINE #0013
0870  POS V = 100.0 % STORE POSITION 20
```

```
0880  CALL PROG 3014
0890  POS V = 100.0 % FINE #0014
0900 POS V = 100.0 % STORE POSITION 20
0910  CALL PROG 3014
0920  POS V = 100.0 % FINE #0015
0930 POS V = 100.0 % STORE POSITION 20
0940  CALL PROG 3014
0950  POS V = 100.0 % FINE #0016
0960 POS V = 100.0 % STORE POSITION 20
0970  CALL PROG 3014
0980  POS V = 100.0 % FINE #0017
0990 POS V = 100.0 % STORE POSITION 20
1000  CALL PROG 3014
1010  POS V = 100.0 % FINE #0018
1020 POS V = 100.0 % STORE POSITION 20
1030  CALL PROG 3014
1040  POS V = 100.0 % FINE #0019
1050 POS V = 100.0 % STORE POSITION 20
1060  CALL PROG 3014
1070  POS V = 100.0 % FINE #0020
1080 POS V = 100.0 % STORE POSITION 20
1090  CALL PROG 3014
1100  POS V = 100.0 % FINE #0021
1110 POS V = 100.0 % STORE POSITION 20
1120  CALL PROG 3014
1130  POS V = 100.0 % FINE #0022
1140 POS V = 100.0 % STORE POSITION 20
1150  CALL PROG 3014
1160  POS V = 100.0 % FINE #0023
1170 POS V = 100.0 % STORE POSITION 20
1180  CALL PROG 3014
1190  POS V = 100.0 % FINE #0024
1200 POS V = 100.0 % STORE POSITION 20
1210  CALL PROG 3014
1220  POS V = 100.0 % FINE #0025
1230 POS V = 100.0 % STORE POSITION 20
1240  CALL PROG 3014
1250  POS V = 100.0 % FINE #0026
1260 POS V = 100.0 % STORE POSITION 20
1270  CALL PROG 3014
1280  POS V = 100.0 % FINE #0027
1290 POS V = 100.0 % STORE POSITION 20
1300  CALL PROG 3014
1310  POS V = 100.0 % FINE #0028
1320 POS V = 100.0 % STORE POSITION 20
1330  CALL PROG 3014
```

```
1340  POS V = 100.0 % FINE #0029
1350  POS V = 100.0 % STORE POSITION 20
1360  CALL PROG 3014
1370  POS V = 100.0 % FINE #0030
1380  POS V = 100.0 % STORE POSITION 20
1390  CALL PROG 3014
1400  POS V = 100.0 % FINE #0031
1410  POS V = 100.0 % STORE POSITION 20
1420  CALL PROG 3014
1430  POS V = 100.0 % FINE #0032
1440  POS V = 100.0 % STORE POSITION 20
1450  CALL PROG 3014
1460  POS V = 100.0 % FINE #0033
1470  POS V = 100.0 % STORE POSITION 20
1480  CALL PROG 3014
1490  POS V = 100.0 % FINE #0034
1500  POS V = 100.0 % STORE POSITION 20
1510  CALL PROG 3014
1520  POS V = 100.0 % FINE #0035
1530  POS V = 100.0 % STORE POSITION 20
1540  CALL PROG 3014
1550  POS V = 100.0 % FINE #0036
1560  POS V = 100.0 % STORE POSITION 20
1570  CALL PROG 3014
1580  STOP ;
.
```

2.Programmi di tipo B

Nelle pagine seguenti sono riportati i seguenti listati.

MOVISP
MOVISPC
BEMOVI4
SCA_SF
INCFR
PIAORI
OCINCFR
BERU1
ONDABERU

Questi programmi sono scritti in linguaggio C e fanno uso di librerie di funzioni di tipo generale e di librerie specializzate per il trasferimento di dati e comandi tra il calcolatore e i componenti della stazione di lavoro: in particolare vengono usate la libreria Robotics Communication Tools per il collegamento con l'unità di controllo del robot [5], e la libreria GPIB per il collegamento con l'oscilloscopio digitale [6].

I listati degli altri programmi di tipo A o B appartenenti ai moduli descritti in [1] e non presenti in questa nota sono riportati in [7].

Programma MOVISP

```
/***movimentazione della sonda nello spazio*/

  #include  <stdio.h>
  #include  <graph.h>
  #include  <math.h>
  .#include  "c:\robinc\ctsttype.h"
   #include  "c:\robinc\ctstfunc.h"

  #define cy        'y'
  #define ccy       'Y'
  #define cn        'n'
  #define ccn       'N'
   #define  PI  3.14159265359

    move_array  move_dat;
    stat_array  stat_dat;
    spon_array  spon_dat;

   main ()
  {
   int frame_id;
   int tcp_id;
   int i,ind2;
         int  enbl_port,enbl_robot,out_no,out_dat,robot_mode,status;
        int prog_id,starter,reg_id;
    short pos[7];
    unsigned char ys;
     float  coordx,coordy,coordz,coordq1,coordq2,coordq3,coordq4;
    double  w,x,y,z,s,a,b,c,d,e,f,g,t,j,k,m,n,o,p,q,r,u,v;

      prog_id=1000;          /* Numero del programma S3 */
      starter=0;             /* 0=inizia, 1= continua programma S3 */

   /****** controllo S3***/
      enbl_robot=0;
      enbl_port=0;
        set_robot_on(enbl_robot,enbl_port,&status);

       cntrl_status(status);
        read_rob_stat(stat_dat,&status);
       cntrl_status(status);

 if(stat_dat[2]==1||stat_dat[3]==0||stat_dat[4]==1||stat_dat[5]==1||stat_dat[6
 ]!=0)
```

```
  {
    printf("\a");
     leggi_stampa_stato();
    exit(-1);

    /************ sincronizzazione automatica ******/

 robot_mode=1;
  cmd_set_mode(robot_mode,&status);

/************** movimentazione remota *******/

 found:
 ind2=1;
  while(ind2)
{
     printf("\nIntroduci coord.X   :  ");
    scanf("%f",&coordx);

     printf("\nIntroduci coord.Y   :  ");
    scanf("%f",&coordy);

     printf("\nIntroduci coord.Z   :  ");
    scanf("%f",&coordz);

      printf("\nIntroduci angolo di rotazione intorno asse Y  ");
      printf("\nSingolarmente sposta la sonda nel piano xz:   ");
    scanf("%lf",&t);

      t=(t/2.0)*PI/180.0;
    w= cos (t);
    x= sin (t) * 0;
    y= sin (t) * 1;
    z= sin (t) * 0;

      printf("\nIntroduci angolo di rotazione t intorno all'asse Z   ");
      printf("\nSingolarmente sposta la sonda nel piano xy:    ");
    scanf("%lf",&t);

      t=(t/2.0)*PI/180.0;
    s= cos (t);
    a= sin (t) * 0;
    b= sin (t) * 0;
    c= sin (t) * 1;
    d=w*s-(x*a+y*b+z*c);
    e=w*a+x*s+y*c-z*b;
```

```
f=w*b+y*s+z*a-x*c;
g=w*c+z*s+x*b-y*a;

  printf("\nIntroduci angolo di rotazione t intorno all'asse X   ");
  printf("\nSingolarmente sposta la sonda nel piano yz:    ");

scanf("%lf",&t);

  t=(t/2.0)*PI/180.0;
m= cos (t);
n= sin (t) * 1;
o= sin (t) * 0;
p= sin (t) * 0;
q=d*m-(e*n+f*o+g*p);
r=d*n+e*m+f*p-g*o;
u=d*o+f*m+g*n-e*p;
v=d*p+g*m+e*o-f*n;
 printf("\nRisultanti q  r  u  v:\n  %lf %lf %lf %lf",q,r,u,v);

/************** start programma robot ***************/

   cmd_prog_start(prog_id,starter,&status);
  cntrl_status(status);
 att_supcon();

/********** sonda sulla posizione di partenza **********/

    pos[0]=(int)coordx*8;
    pos[1]=(int)coordy*8;
    pos[2]=(int)coordz*8;
    pos[3]=q*16393;
    pos[4]=r*16393;
    pos[5]=u*16393;
    pos[6]=v*16393;

    move_dat[8]= pos[0];    /*X*/
    move_dat[9]= pos[1];     /*Y*/
    move_dat[10]= pos[2];    /*Z*/
    move_dat[11]= pos[3];     /*Q1*/
    move_dat[12]= pos[4];     /*Q2*/
    move_dat[13]= pos[5];     /*Q3*/
    move_dat[14]= pos[6];     /*Q4*/

  printf("\nRisultanti q  r  u  v:\n  %d %d %d
%d",move_dat[11],move_dat[12],move_dat[13],move_dat[14]);
```

```
        movi(move_dat);
        leggi1_stampa_stato();
/*************************************************/

      while(1)
     {
     printf("\n VUOI TERMINARE ?  (y/n)  ?:  ");
     scanf("%1s",&ys);
      if(ys==cy||ys==ccy||ys==cn||ys==ccn) break;
     printf("\a");
     }
     if(ys==cy||ys==ccy) ind2=0;

 }

i=7;
  set_robot_off(enbl_robot,&status);
  cntrl_status(status,i);

}

  cntrl_status(status,i)
     int status;
    int i;
{
     if(status <=0 )
   {
          printf("\n\nERRORE stato di ritorno (%d) : %d",i,status);
        exit(-1);
   }
 return(0);
}

/*****************  leggi stampa stato  *************/

  leggi_stampa_stato()
{
   int status;
     read_rob_stat(stat_dat,&status);

  printf("\n\nProgramma   selezionato  :  %d",stat_dat[0]);
  printf("\nIstruzione  selezionata  :  %d",stat_dat[1]);
 if(stat_dat[2] == 0)
     printf("\nStandby : NO");
 else
      printf("\nStandby : SI");
```

```
if(stat_dat[3] == 0)
    printf("\nOperation : NO");
else
    printf("\nOperation : SI");
if(stat_dat[4] == 0)
    printf("\nProgram exec : NO");
else
    printf("\nProgram exec : SI");
if(stat_dat[5] == 0)
    printf("\nEmergency robot : NO");
else
    printf("\nEmergency robot : SI");
if(stat_dat[6] == 0)
    printf("\nTeach pendant : IN");
else
   printf("\nTeach pendant : OUT,DISCONNECTED");
if(stat_dat[7] == 0)
    printf("\nInterrupt : ENABLED");
else
    printf("\nInterrupt : DISABLED");
printf("\nX, Y, Z      : %d  %d  %d",stat_dat[8],stat_dat[9],stat_dat[10]);
printf("\nQ1, Q2, Q3, Q4 : %d  %d  %d
%d",stat_dat[11],stat_dat[12],stat_dat[13],stat_dat[14]);
if(stat_dat[29] == 0)
    printf("\nOrientation : WRIST");
else
    printf("\nOrientation : TOOL");
printf("\nTCP corrente : %d",stat_dat[30]);
 printf(" ");
printf("\n\nPREMERE UN TASTO PER CONTINUARE");
getch();
 return(0);
}

 leggi1_stampa_stato()
{
  int status;
    read_rob_stat(stat_dat,&status);

printf("\nX, Y, Z      : %d  %d  %d",stat_dat[8],stat_dat[9],stat_dat[10]);
printf("\nQ1, Q2, Q3, Q4 : %d  %d  %d
%d",stat_dat[11],stat_dat[12],stat_dat[13],stat_dat[14]);

 return(0);
}
 /******** BEMOVI:C unione di movimentazione remota ***/
```

```
 move_array  move_dat;
 stat_array  stat_dat;
 spon_array  spon_dat;

 movi(move_dat)
 move_array  move_dat;

{
 int frame_id;
 int tcp_id;
int i;
      int  enbl_port,enbl_robot,out_no,out_dat,robot_mode,status,a;
    int  prog_id,starter,reg_id;

 move_dat[0]=0;
 move_dat[1]=0;
 move_dat[2]=0;
 move_dat[3]=1;
 move_dat[4]=0;
 move_dat[5]=0;
 move_dat[6]=800;
 move_dat[7]=8142;
 move_dat[15]=0;
 move_dat[16]=0;
 move_dat[17]=0;
 move_dat[18]=0;
 move_dat[19]=0;
 move_dat[20]=0;
 move_dat[21]=0;
 move_dat[22]=0;
 move_dat[23]=0;
 move_dat[24]=0;
 move_dat[25]=0;
 move_dat[26]=0;
 move_dat[27]=0;
 move_dat[28]=0;
 move_dat[29]=0;

i=3;
 move_dat[2]=2;
  cmd_move(move_dat,&status);
  cntrl_status(status,i);

i=4;
 move_dat[2]=0;
```

```
  cmd_move(move_dat,&status);
  cntrl_status(status,i);

i=5;
 move_dat[2]=3;
  cmd_move(move_dat,&status);
  cntrl_status(status,i);
 return(0);

/*************** attesa supercontr. ********************/
 att_supcon()
{
    int status;
do
  {  read_spon(spon_dat,&status);
    if (status != -22) cntrl_status(status); }
    while (spon_dat[0] !=0 || status == -22);
      cmd_prog_stop(&status);
      cntrl_status(status);
     return(0);
}
```

Programma MOVISPC

```
/*movimentazione della sonda nello spazio che punta sull'origine*/

  #include  <stdio.h>
  #include  <graph.h>
  #include  <math.h>
   #include  "c:\robinc\ctsttype.h"
   #include  "c:\robinc\ctstfunc.h"

  #define cy        'y'
  #define ccy       'Y'
  #define cn        'n'
  #define ccn       'N'
   #define  PI  3.14159265359

   move_array  move_dat;
   stat_array  stat_dat;
   spon_array  spon_dat;

  main ()
 {
  int frame_id;
  int tcp_id;
  int i,ind2;
        int  enbl_port,enbl_robot,out_no,out_dat,robot_mode,status;
       int  prog_id,starter,reg_id;
   short pos[7];
   unsigned char ys;
    float  coordx,coordy,coordz,coordq1,coordq2,coordq3,coordq4;
   double
L,w,x,y,z,s,a,b,c,d,e,f,g,t,j,k,m,n,o,p,q,r,u,v,tx,ty,tz,dx,dy,dz,alfa,beta,rad;
     L=330.0;
     prog_id=1000;                    /* Numero del programma S3 */
     starter=0;                 /* 0=inizia, 1= continua programma S3 */

  /****** controllo S3***/
      enbl_robot=0;
      enbl_port=0;
        set_robot_on(enbl_robot,enbl_port,&status);

       cntrl_status(status);
        read_rob_stat(stat_dat,&status);
       cntrl_status(status);
```

```
if(stat_dat[2]==1||stat_dat[3]==0||stat_dat[4]==1||stat_dat[5]==1||stat_dat[6
]!=0)
    {
      printf("\a");
       leggi_stampa_stato();
      exit(-1);
    }
     /************** sincronizzazione automatica **********/

   robot_mode=1;
    cmd_set_mode(robot_mode,&status);

  /***************** movimentazione remota *****************/

   found:
   ind2=1;
    while(ind2)
  {
      printf("\nIntroduci coord.X   :  ");
     scanf("%f",&coordx);

      printf("\nIntroduci coord.Y   :  ");
     scanf("%f",&coordy);

      printf("\nIntroduci coord.Z   :  ");
     scanf("%f",&coordz);

       printf("\nIntroduci angolo di rotazione intorno asse Y ");
       printf("\nSingolarmente sposta la sonda nel piano xz:  ");
     scanf("%lf",&ty);

       beta=(ty*PI/180.0);

     ty=ty+90.0;

       t=(ty/2.0)*PI/180.0;
     w= cos (t);
     x= sin (t) * 0;
     y= sin (t) * 1;
     z= sin (t) * 0;

       printf("\nIntroduci angolo di rotazione t intorno all'asse Z  ");
       printf("\nSingolarmente sposta la sonda nel piano xy:   ");
     scanf("%lf",&tz);
```

```
        t=(tz)*PI/180.0;
         t=atan(tan(t)*cos(beta));
       alfa=t;

    t=(t/2.0);
   s= cos (t);
   a= sin (t) * 0;
   b= sin (t) * 0;
   c= sin (t) * 1;
   d=w*s-(x*a+y*b+z*c);
   e=w*a+x*s+y*c-z*b;
   f=w*b+y*s+z*a-x*c;
   g=w*c+z*s+x*b-y*a;

    printf("\nprima beta alfa:\n   %lf %lf",beta, alfa);

    dx =- L*cos(alfa)*sin(beta);
   dy = L*sin(alfa);
    dz = L*(1-cos(alfa)*cos(beta));

   printf("\nRisultanti   beta alfa:\n   %lf %lf",beta, alfa);
    printf("\nRisultanti dx dy   dz:\n %lf %lf %lf",dx, dy,dz);
   getch();

tx=0.0;
     t=(tx/2.0)*PI/180.0;
   m= cos (t);
   n= sin (t) * 1;
   o= sin (t) * 0;
   p= sin (t) * 0;
   q=d*m-(e*n+f*o+g*p);
   r=d*n+e*m+f*p-g*o;
   u=d*o+f*m+g*n-e*p;
   v=d*p+g*m+e*o-f*n;
    printf("\nRisultanti q   r   u   v:\n   %lf %lf %lf %lf",q,r,u,v);

/**************** start programma robot ***************/

      cmd_prog_start(prog_id,starter,&status);
     cntrl_status(status);
    att_supcon();

/********** sonda sulla posizione di partenza ***********/

      pos[0]=(coordx-dx)*8;
       pos[1]=(coordy-dy)*8;
```

```
        pos[2]=(coordz-dz)*8;
        pos[3]=q*16393;
        pos[4]=r*16393;
        pos[5]=u*16393;
        pos[6]=v*16393;

        move_dat[8]= pos[0];     /*X*/
        move_dat[9]= pos[1];      /*Y*/
        move_dat[10]= pos[2];     /*Z*/
        move_dat[11]= pos[3];      /*Q1*/
        move_dat[12]= pos[4];      /*Q2*/
        move_dat[13]= pos[5];      /*Q3*/
        move_dat[14]= pos[6];      /*Q4*/

   printf("\nRisultanti q  r  u  v:\n  %d %d %d
%d",move_dat[11],move_dat[12],move_dat[13],move_dat[14]);

        movi(move_dat);
        leggi1_stampa_stato();
/*****************************************************/
    do
    {
          printf("\nVuoi terminare ? (s/n): ");
        i=getch();
        ind2=-1;
        if(i == 's' || i == 'S')  ind2=0;
        if(i == 'n' || i == 'N')  ind2=1;
    }
     while(ind2 < 0);
 }

 i=7;
   set_robot_off(enbl_robot,&status);
   cntrl_status(status,i);

 }
 /*************************************************/

   cntrl_status(status,i)
     int status;
    int i;
 {
     if(status <=0 )
    {
          printf("\n\nERRORE stato di ritorno (%d) : %d",i,status);
        exit(-1);
```

```
    }
  return(0);
}
 /*****************   leggi stampa stato   *************/

  leggi_stampa_stato()
{
   int status;
     read_rob_stat(stat_dat,&status);

  printf("\n\nProgramma   selezionato : %d",stat_dat[0]);
  printf("\nIstruzione selezionata : %d",stat_dat[1]);
 if(stat_dat[2] == 0)
     printf("\nStandby : NO");
 else
      printf("\nStandby : SI");
 if(stat_dat[3] == 0)
     printf("\nOperation : NO");
 else
      printf("\nOperation : SI");
 if(stat_dat[4] == 0)
     printf("\nProgram exec : NO");
 else
      printf("\nProgram exec : SI");
 if(stat_dat[5] == 0)
     printf("\nEmergency robot : NO");
 else
      printf("\nEmergency robot : SI");
 if(stat_dat[6] == 0)
      printf("\nTeach pendant : IN");
 else
    printf("\nTeach pendant : OUT,DISCONNECTED");
 if(stat_dat[7] == 0)
     printf("\nInterrupt : ENABLED");
 else
      printf("\nInterrupt : DISABLED");
 printf("\nX, Y, Z      : %d  %d  %d",stat_dat[8],stat_dat[9],stat_dat[10]);
 printf("\nQ1, Q2, Q3, Q4 : %d  %d  %d
%d",stat_dat[11],stat_dat[12],stat_dat[13],stat_dat[14]);
  if(stat_dat[29] == 0)
      printf("\nOrientation : WRIST");
 else
     printf("\nOrientation : TOOL");
  printf("\nTCP corrente : %d",stat_dat[30]);
   printf(" ");
 printf("\n\nPREMERE UN TASTO PER CONTINUARE");
```

```
 getch();
 return(0);
}

 leggi1_stampa_stato()
{
  int status;
    read_rob_stat(stat_dat,&status);

 printf("\nX,  Y,  Z      : %d  %d  %d",stat_dat[8],stat_dat[9],stat_dat[10]);
 printf("\nQ1, Q2, Q3, Q4 : %d  %d  %d
%d",stat_dat[11],stat_dat[12],stat_dat[13],stat_dat[14]);

 return(0);
}
 /******** BEMOVI:C unione di movimentazione remota ***/

  move_array  move_dat;
  stat_array  stat_dat;
  spon_array  spon_dat;

  movi(move_dat)
  move_array  move_dat;

{
 int frame_id;
 int tcp_id;
 int i;
      int  enbl_port,enbl_robot,out_no,out_dat,robot_mode,status,a;
     int  prog_id,starter,reg_id;

 move_dat[0]=0;
 move_dat[1]=0;
 move_dat[2]=0;
 move_dat[3]=1;
 move_dat[4]=0;
 move_dat[5]=0;
 move_dat[6]=800;
 move_dat[7]=8142;
 move_dat[15]=0;
 move_dat[16]=0;
 move_dat[17]=0;
 move_dat[18]=0;
 move_dat[19]=0;
 move_dat[20]=0;
 move_dat[21]=0;
```

```
 move_dat[22]=0;
 move_dat[23]=0;
 move_dat[24]=0;
 move_dat[25]=0;
 move_dat[26]=0;
 move_dat[27]=0;
 move_dat[28]=0;
 move_dat[29]=0;

i=3;
 move_dat[2]=2;
  cmd_move(move_dat,&status);
  cntrl_status(status,i);

i=4;
 move_dat[2]=0;
  cmd_move(move_dat,&status);
  cntrl_status(status,i);

i=5;
 move_dat[2]=3;
  cmd_move(move_dat,&status);
  cntrl_status(status,i);
 return(0);

}
/************** attesa supercontr. ********************/

 att_supcon()
{
    int status;
do
  { read_spon(spon_dat,&status);
    if (status != -22) cntrl_status(status); }
    while (spon_dat[0] !=0 || status == -22);
      cmd_prog_stop(&status);
      cntrl_status(status);
     return(0);
}
```

Programma BEMOVI4

```
/*unione di movimentazione remota e richiami programmi*/

#include <stdio.h>
#include  "c:\robinc\ctsttype.h"
#include  "c:\robinc\ctstfunc.h"

#define cy        'y'
#define ccy       'Y'
#define cn        'n'
#define ccn       'N'

move_array  move_dat;
stat_array  stat_dat;
spon_array  spon_dat;

main ()
{
int  frame_id;
int  tcp_id;
int  i,ind2;
     int  enbl_port,enbl_robot,out_no,out_dat,robot_mode,status;
    int prog_id,starter,reg_id;
 short pos[7];
 unsigned char ys;
float  coordx,coordy,coordz,coordq1,coordq2,coordq3,coordq4;

   prog_id=1000;          /* Numero del programma S3 */
   starter=0;              /* 0=inizia, 1= continua programma S3 */

/****** controllo S3***/
    enbl_robot=0;
    enbl_port=0;
      set_robot_on(enbl_robot,enbl_port,&status);

     cntrl_status(status);
      read_rob_stat(stat_dat,&status);
     cntrl_status(status);

if(stat_dat[2]==1||stat_dat[3]==0||stat_dat[4]==1||stat_dat[5]==1||stat_dat[6
]!=0)
  {
    printf("\a");
     leggi_stampa_stato();
    exit(-1);
```

```
  }
   /************** sincronizzazione automatica **

robot_mode=1;
cmd_set_mode(robot_mode,&status);

/*************** movimentazione remota ******

found:
ind2=1;
while(ind2)
{
    printf("\nIntroduci coord.X   :   ");
    scanf("%f",&coordx);

    printf("\nIntroduci coord.Y   :   ");
    scanf("%f",&coordy);

    printf("\nIntroduci coord.Z   :   ");
    scanf("%f",&coordz);

/***************** start programma robot ******

    cmd_prog_start(prog_id,starter,&status);
     cntrl_status(status);
    att_supcon();

/********** sonda sulla posizione di partenza ***

       pos[0]=(int)coordx*8;
       pos[1]=(int)coordy*8;
       pos[2]=(int)coordz*8;
       pos[3]=11591;
      pos[4]=0;
       pos[5]=11591;
      pos[6]=0;

       move_dat[8]= pos[0];       /*X*/
       move_dat[9]= pos[1];       /*Y*/
       move_dat[10]= pos[2];      /*Z*/
       move_dat[11]= pos[3];      /*Q1*/
       move_dat[12]= pos[4];      /*Q2*/
       move_dat[13]= pos[5];      /*Q3*/
       move_dat[14]= pos[6];      /*Q4*/

        movi(move_dat);
```

```
        leggi_stampa_stato();
/*************************************************/

      while(1)
     {
    printf("\n VUOI TERMINARE ?  (y/n)  ?:  ");
    scanf("%1s",&ys);
     if(ys==cy||ys==ccy||ys==cn||ys==ccn) break;
     printf("\a");
     }
    if(ys==cy||ys==ccy) ind2=0;
 }

i=7;
set_robot_off(enbl_robot,&status);
cntrl_status(status,i);

}
/*************************************************/

cntrl_status(status,i)
     int status;
    int i;
{
     if(status <=0 )
    {
          printf("\n\nERRORE stato di ritorno (%d) : %d",i,status);
         exit(-1);
    }
return(0);
}

  /******************  leggi stampa stato  **************/

    leggi_stampa_stato()
  {
     int status;
       read_rob_stat(stat_dat,&status);

    printf("\n\nProgramma  selezionato : %d",stat_dat[0]);
    printf("\nIstruzione selezionata : %d",stat_dat[1]);
   if(stat_dat[2] == 0)
       printf("\nStandby : NO");
   else
        printf("\nStandby : SI");
   if(stat_dat[3] == 0)
```

```
        printf("\nOperation : NO");
    else
        printf("\nOperation : SI");
    if(stat_dat[4] == 0)
        printf("\nProgram exec : NO");
    else
        printf("\nProgram exec : SI");
    if(stat_dat[5] == 0)
        printf("\nEmergency robot : NO");
    else
        printf("\nEmergency robot : SI");
    if(stat_dat[6] == 0)
        printf("\nTeach pendant : IN");
    else
       printf("\nTeach pendant : OUT,DISCONNECTED");
    if(stat_dat[7] == 0)
        printf("\nInterrupt : ENABLED");
    else
        printf("\nInterrupt : DISABLED");
    printf("\nX, Y, Z      : %d  %d  %d",stat_dat[8],stat_dat[9],stat_dat[10]);
    printf("\nQ1, Q2, Q3, Q4 : %d  %d  %d
%d",stat_dat[11],stat_dat[12],stat_dat[13],stat_dat[14]);
    if(stat_dat[29] == 0)
        printf("\nOrientation : WRIST");
    else
        printf("\nOrientation : TOOL");
    printf("\nTCP corrente : %d",stat_dat[30]);
    printf(" ");
    printf("\n\nPREMERE UN TASTO PER CONTINUARE");
    getch();
    return(0);
  }

leggi1_stampa_stato()
{
    int status;
     read_rob_stat(stat_dat,&status);

printf("\nX, Y, Z      : %d  %d  %d",stat_dat[8],stat_dat[9],stat_dat[10]);
printf("\nQ1, Q2, Q3, Q4 : %d  %d  %d
%d",stat_dat[11],stat_dat[12],stat_dat[13],stat_dat[14]);

return(0);
}
/******** BEMOVI:C unione di movimentazione remota ***/
```

```
move_array  move_dat;
stat_array  stat_dat;
spon_array  spon_dat;

movi(move_dat)
move_array  move_dat;

{
int  frame_id;
int  tcp_id;
int i;
     int  enbl_port,enbl_robot,out_no,out_dat,robot_mode,status,a;
    int  prog_id,starter,reg_id;

move_dat[0]=0;
move_dat[1]=0;
move_dat[2]=0;
move_dat[3]=1;
move_dat[4]=0;
move_dat[5]=0;
move_dat[6]=800;
move_dat[7]=8142;
move_dat[15]=0;
move_dat[16]=0;
move_dat[17]=0;
move_dat[18]=0;
move_dat[19]=0;
move_dat[20]=0;
move_dat[21]=0;
move_dat[22]=0;
move_dat[23]=0;
move_dat[24]=0;
move_dat[25]=0;
move_dat[26]=0;
move_dat[27]=0;
move_dat[28]=0;
move_dat[29]=0;

i=3;
move_dat[2]=2;
cmd_move(move_dat,&status);
cntrl_status(status,i);

i=4;
move_dat[2]=0;
cmd_move(move_dat,&status);
```

```
cntrl_status(status,i);

i=5;
move_dat[2]=3;
cmd_move(move_dat,&status);
cntrl_status(status,i);
return(0);
}

/**********  attesa supercontr.  **************/

att_supcon()
{
   int status;
do
{  read_spon(spon_dat,&status);
  if (status != -22) cntrl_status(status); }
  while (spon_dat[0] !=0 || status == -22);
    cmd_prog_stop(&status);
    cntrl_status(status);
   return(0);
}
```

Programma SCA_SF

```
/********* AQUISIZIONE E MOVIMENTAZIONE REMOTA *********/
  /* esame pezzo comandato dal calcolatore lettura dati dopo sync
LeCroy, */
  /* con introduzione delle coordinate di partenza con possibilità di
cambiarle*/
  /* con il calcolo delle coordinate X */

   #include "\gpib-pc\decl.h"
  #include <stdio.h>
  #include <conio.h>
  #include <math.h>
  #include <stdlib.h>
   #include "c:\robinc\ctsttype.h"
   #include "c:\robinc\ctstfunc.h"
  #include <time.h>
   #include <sys\types.h>
   #include <sys\timeb.h>

   #define PI 3.14159265359
  #define cc      'c'
  #define ccc     'C'
  #define ce      'e'
  #define cce     'E'
  #define cf      'f'
  #define ccf     'F'
  #define cy      'y'
  #define ccy     'Y'
  #define cn      'n'
  #define ccn     'N'
  #define esc      '\x1b'

   float coordx,coordy,coordz,leggi_tensione();

  FILE *stream;
   stat_array stat_dat;
   spon_array spon_dat;
   move_array move_dat;
 /*****************************************************/

  main()
  {
       int cr,i,l,m,n,num,nd,nr,p,nc,reg_dat,res,j,mm,val;
      int lr,xx,yy,ncc,nrr,mask,ind1,ind2;
     long sleeper,swap;
```

```
    int kbhit(void);

    int zz,zz1,brr,brr1;
    double  ang,ang1,ang2;

      int enbl_port,enbl_robot,status,robot_mode,a;
     int prog_id,starter,reg_id;
    float x,min,max,wd[512];
    unsigned char ys,buf1[8];
   short vs[512];
    char buf[30],str[10],stringa[13];

   short pos[7];
     float tensione,tensione_max;
    float pa,fp,increy,increx;
    long timer(),fine;
     double tempo,max_tempo;
    unsigned coerr;
     struct timeb timebuffer;
    short start,stop;
    max_tempo=50;
   coerr=0;
  zz=0;
  zz1=0;

/****************** inizializzazione schermo *********************/
    clrscrn();
   printf("\n            *** ACQUISIZIONE E MOVIMENTAZIONE
REMOTA***");
    printf("\n                        *lettura dati dopo sync LeCroy* \n");

    prog_id=2015;                    /* Numero del programma S3 */
    starter=0;                /* 0=inizia, 1= continua programma S3 */

 /****** controllo GPIB***/
    if((nd=ibfind("DEV4"))<0)
  {
       printf("\nInterfaccia GPIB :   Errore di canale !");
      exit(-1);
  }
 /****** controllo S3***/
    enbl_robot=0;
    enbl_port=0;
      set_robot_on(enbl_robot,enbl_port,&status);
     cntrl_status(status);
      read_rob_stat(stat_dat,&status);
```

```
        cntrl_status(status);

if(stat_dat[2]==1||stat_dat[3]==0||stat_dat[4]==1||stat_dat[5]==1||stat_dat[6
]!=0)
    {
      printf("\a");
       leggi_stampa_stato();
     exit(-1);
    }

 /***************apertura file *************/
  aprif();
 /*************************************/
   /*****inserimento  parametri****/

  found:
  ind2=1;
   while(ind2)
 {
      printf("\nIntroduci coord.X di partenza (1083) :  ");
     scanf("%f",&coordx);

      printf("\nIntroduci coord.Y di partenza (-439.5) :  ");
     scanf("%f",&coordy);

      printf("\nIntroduci coord.Z di partenza (1033.6) :  ");
     scanf("%f",&coordz);
  /******* sposta robot *************/

  /****************** start programma robot **************/

       cmd_prog_start(prog_id,starter,&status);
      cntrl_status(status);
     att_supcon();

  /********** sonda sulla posizione di partenza ************/

       pos[0]=(int)coordx*8;
       pos[1]=(int)coordy*8;
       pos[2]=(int)coordz*8;
      pos[3]=11591;
     pos[4]=0;
      pos[5]=11591;
     pos[6]=0;

       move_dat[8]= pos[0];     /*X*/
```

```
        move_dat[9]= pos[1];      /*Y*/
        move_dat[10]= pos[2];     /*Z*/
        move_dat[11]= pos[3];     /*Q1*/
        move_dat[12]= pos[4];     /*Q2*/
        move_dat[13]= pos[5];     /*Q3*/
        move_dat[14]= pos[6];     /*Q4*/

         movi(move_dat);
/*****************************************/

      while(1)
     {
    printf("\n POSIZIONE O.K ?  (y/n)  ?:  ");
    scanf("%1s",&ys);
     if(ys==cy||ys==ccy||ys==cn||ys==ccn) break;
     printf("\a");
     }
     if(ys==cy||ys==ccy) ind2=0;
 }
 /******** prosegue inserimento parametri ********/

     while(1)
      {
    printf("\nSCANSIONE ASSE X  ([1,600]mm.)   ?:  ");
      res=getint(3,1,27,1,0,&nc,1,1,600);
     if(res == -4) exit(-1);
    if(nc>0 && nc<601) break;
      }

     while(1)
      {
    printf("\nSCANSIONE ASSE Y  ([1,999]mm.)   ?:  ");
      res=getint(3,1,27,1,0,&nr,1,1,999);
     if(res == -4) exit(-1);
     if(nr>0 && nr<1000) break;
      }

     while(1)
      {
    printf("\nPASSO INCREMENTALE ([1,100]decimi)  ?:  ");
      res=getint(3,1,27,1,0,&p,1,1,100);
     if(res == -4) exit(-1);
    if(p>0 && p<101) break;
      }

     ncc=nc*10/p;
```

```
    nrr=nr*10/p;

    fp=(float)p;

     increy = fp/10.0*8.0;
     increx = fp/10.0*8.0;
cr='c';

    while(1)
     {
    printf("\nSOGLIA VALORE MASSIMO [0.1 , 9.9]  ?:  ");
      res=getfloat(1,1,1,27,1,0,&max,1,0.,10.);
     if(res == 27) exit(-1);
    if(max > 0.  &&  max < 10.) break;
     printf("\a");
     }
     while(1)
     {
    printf("\nSOGLIA VALORE MIMIMO [-1 , < max  ; CR=0.]    ?:  ");
      res=getfloat(2,1,1,27,1,0,&min,1,-1.,max);
     if(res == 27) exit(-1);
    if(min >= -1.  &&  min < max) break;
     printf("\a");
     }
    ind1=1;
     while(ind1)
     {
    printf("\nMEMORIA DI QUADRO (0,1,2,3)   ?:  ");
      res=getint(1,1,27,1,0,&mm,1,0,3);
     if(res == -4) exit(-1);
     if(mm>-1&&mm<4)
    {
     a_displ(mm);
      while(1)
     {
     printf("\nO.K  (y/n)  ?:  ");
    scanf("%1s",&ys);
     if(ys==cy||ys==ccy||ys==cn||ys==ccn) break;
     printf("\a");
     }
     if(ys==cy||ys==ccy) ind1=0;
    }
      }
    val=0;
     a_set(val);
```

```
    printf("\nNumero righe = %d    Numero colonne = %d\n",nrr,ncc);
   xx=100;
   yy=99;
    memset(buf,0,30);
    memset(buf1,0,8);
    memset(pos,0,7);
   ibclr(nd);
   ibwrt(nd,"CHDR OFF",8);            /*elimina testate messaggi*/
   ibwrt(nd,"ACAL OFF",8);            /*inibisce autocalibrazione*/
   ibwrt(nd,"CRMS OFF",8);            /*inibisce visione parametri*/

/********** sonda sulla posizione di partenza ************/

      pos[0]=(int)coordx*8;
      pos[1]=(int)coordy*8;
      pos[2]=(int)coordz*8;
      pos[3]=11591;
     pos[4]=0;
      pos[5]=11591;
     pos[6]=0;

      move_dat[8]= pos[0];    /*X*/
      move_dat[9]= pos[1];     /*Y*/
      move_dat[10]= pos[2];    /*Z*/
      move_dat[11]= pos[3];     /*Q1*/
      move_dat[12]= pos[4];     /*Q2*/
      move_dat[13]= pos[5];     /*Q3*/
      move_dat[14]= pos[6];     /*Q4*/

       movi(move_dat);
/*****************************************/

  switch(cr)
       {
          case 'c': case 'C':
           strcpy(stringa,"C1:PAVA? MAX");
          mask=1;
          break;

          case 'e': case 'E':
           strcpy(stringa,"FE:PAVA? MAX");
           mask=1024;
          break;

          case 'f': case 'F':
           strcpy(stringa,"FF:PAVA? MAX");
```

```
            mask=2048;
            break;
          }

/******************** ciclo di lettura dati **********************/

  for (m=0; m<(nrr); m++)       /*  ciclo avanz. asse Y*/

  {
     for (n=0; n<ncc; n++)        /*ciclo avanz. asse X */
     {

       move_dat[8]=pos[0]-n*increx;    /*X*/
       movi(move_dat);
       impulso_tras();

         ftime(&timebuffer);
           start=timebuffer.millitm; /*controllo tm*/

   while(1)          /*  attesa per bit di INR=1 (segnale acquisito, etc..) */
    {  ibwrt(nd,"INR?",4);
       ibrd(nd,buf1,8);
        if(atoi(buf1)&mask)  break;

       ftime(&timebuffer);
         stop=timebuffer.millitm; /*controllo tm*/
      if(stop<start)
      stop+=1000;
       tempo=stop-start;
       if(tempo>max_tempo)
       {
          printf("\n %d  %d %d %f ",m,n,(atoi(buf1)&mask),tempo);
       coerr++;
        break;
       }
         start=stop;
   }
       ibwrt(nd,stringa,12);
       ibrd(nd,buf,16);
      wd[n]=atof(buf+4);
  }                              /* For interno */

/******************** scrittura riga su PIP ****************/

    for (n=0; n<ncc; n++)
    {
```

```
            vs[n]=((wd[n]-min)/(max-min))*255;
          if(vs[n] < 0) vs[n]=0;
           else if(vs[n] > 255) vs[n]=255;
      }
       yy=yy+1;
        a_rowwi(xx,yy,ncc,vs);

       if(kbhit() != 0  &&  getch() == esc)
      {
         nrr=m+1;
         break;
      }

  /* comando da calcolatore*/

     pos[0]=(int)coordx*8;
    move_dat[9]= pos[1] + m*increy;       /*Y*/

    move_dat[8]= pos[0];     /*X*/

        movi(move_dat);

      }                              /* For esterno */

       printf("\nNum righe = %d     Num colonne = %d",(nrr+27),ncc);
        printf("\nNUM.ACQUISIZIONI = %.0f",(double)nrr*(double)ncc);
      printf("\nNUM.ERRORI = %d",coerr);

    /************ disconnessione robot ed oscilloscopio ***********/
      scon(nd,status,enbl_robot);
}
/******************** controllo dello stato **********************/

  cntrl_status(status)
      int status;
{
      if (status <= 0)
     {
        printf("\n\nControllo S3 :    Errore di stato  %d",status);
        leggi_stampa_stato();
       exit (-1);
     }
       return(0);
}
 /************* disconnessione robot ed oscilloscopio ***********/
```

```
  scon(nd,status,enbl_robot)
       int nd,status,enbl_robot;
{
     cmd_prog_stop(&status);
     cntrl_status(status);
      set_robot_off(enbl_robot,&status);
     cntrl_status(status);
   ibwrt(nd,"BUZZ BEEP",9);
   ibloc(nd);                          /* rilascia canale */
    return(0);
}
/******************  attesa supercontr.  *********************/

 att_supcon()
{
    int status;
do
 {  read_spon(spon_dat,&status);
   if (status != -22) cntrl_status(status); }
   while (spon_dat[0] !=0 || status == -22);
     cmd_prog_stop(&status);
     cntrl_status(status);
    return(0);
}

/******************  leggi stampa stato  *************/

 leggi_stampa_stato()
{
   int status;
     read_rob_stat(stat_dat,&status);

 printf("\n\nProgramma  selezionato : %d",stat_dat[0]);
 printf("\nIstruzione selezionata : %d",stat_dat[1]);
if(stat_dat[2] == 0)
    printf("\nStandby : NO");
else
     printf("\nStandby : SI");
if(stat_dat[3] == 0)
    printf("\nOperation : NO");
else
     printf("\nOperation : SI");
if(stat_dat[4] == 0)
    printf("\nProgram exec : NO");
else
     printf("\nProgram exec : SI");
```

```
 if(stat_dat[5] == 0)
     printf("\nEmergency robot : NO");
 else
     printf("\nEmergency robot : SI");
 if(stat_dat[6] == 0)
     printf("\nTeach pendant : IN");
 else
    printf("\nTeach pendant : OUT,DISCONNECTED");
 if(stat_dat[7] == 0)
     printf("\nInterrupt : ENABLED");
 else
     printf("\nInterrupt : DISABLED");
 printf("\nX, Y, Z      : %d  %d  %d",stat_dat[8],stat_dat[9],stat_dat[10]);
 printf("\nQ1, Q2, Q3, Q4 : %d  %d  %d
%d",stat_dat[11],stat_dat[12],stat_dat[13],stat_dat[14]);
 if(stat_dat[29] == 0)
     printf("\nOrientation : WRIST");
 else
     printf("\nOrientation : TOOL");
 printf("\nTCP corrente : %d",stat_dat[30]);
 printf(" ");
 printf("\n\nPREMERE UN TASTO PER CONTINUARE");
 getch();
 return(0);
}

 /******** BEMOVI:C unione di movimentazione remota ***/

 move_array move_dat;
 stat_array stat_dat;
 spon_array spon_dat;

 movi(move_dat)
 move_array move_dat;

{
 int frame_id;
 int tcp_id;
 int i;
      int enbl_port,enbl_robot,out_no,out_dat,robot_mode,status,a;
     int prog_id,starter,reg_id;

 move_dat[0]=0;
 move_dat[1]=0;
 move_dat[2]=0;
 move_dat[3]=1;
```

```
 move_dat[4]=0;
 move_dat[5]=0;
 move_dat[6]=1800;
 move_dat[7]=8142;
 move_dat[15]=0;
 move_dat[16]=0;
 move_dat[17]=0;
 move_dat[18]=0;
 move_dat[19]=0;
 move_dat[20]=0;
 move_dat[21]=0;
 move_dat[22]=0;
 move_dat[23]=0;
 move_dat[24]=0;
 move_dat[25]=0;
 move_dat[26]=0;
 move_dat[27]=0;
 move_dat[28]=0;
 move_dat[29]=0;

i=3;
 move_dat[2]=2;
  cmd_move(move_dat,&status);
  cntrl_status(status,i);

i=4;
 move_dat[2]=0;
  cmd_move(move_dat,&status);
  cntrl_status(status,i);

i=5;
 move_dat[2]=3;
  cmd_move(move_dat,&status);
  cntrl_status(status,i);
  return(0);
}
 /***invio impulso al trasmettitore  ***/

  impulso_tras()
{
     int out_no,out_dat,status;
    out_no=1;
     out_dat=1;
        write_output(out_no,&out_dat,&status);
     out_dat=0;
        write_output(out_no,&out_dat,&status);
```

```
 return(0);
}

 aprif()
{
     if ((stream=fopen("ondc.dat","wb"))==NULL)
   {
        printf("file non aperto\n");
      exit(0);
   }
     return(0);
}

 chiudif()
{
    fclose(stream);
}
```

Programma INCFR

```
/********* AQUISIZIONE E MOVIMENTAZIONE REMOTA *********/
   /* esame pezzo comandato dal calcolatore lettura dati dopo sync
LeCroy, */
  /* con introduzione delle coordinate di partenza con possibilità di
cambiarle*/
  /* piano inclinato con cambiamento del piano di riferimento*/
   #include "\gpib-pc\decl.h"
  #include <stdio.h>
  #include <conio.h>
  #include <math.h>
  #include <stdlib.h>
   #include "c:\robinc\ctsttype.h"
   #include "c:\robinc\ctstfunc.h"
  #include <time.h>
   #include <sys\types.h>
   #include <sys\timeb.h>

   #define PI 3.14159265359
  #define cc       'c'
  #define ccc      'C'
  #define ce       'e'
  #define cce      'E'
  #define cf       'f'
  #define ccf      'F'
  #define cy       'y'
  #define ccy      'Y'
  #define cn       'n'
  #define ccn      'N'
  #define esc       '\x1b'

   float coordx,coordy,coordz,leggi_tensione();

   posi_array posi_dat;
   stat_array stat_dat;
   spon_array spon_dat;
   move_array move_dat;
 /*************************************************/

  main()
  {
       int cr,i,l,am,an,num,nd,nr,p,nc,reg_dat,res,j,mm,val;
      int lr,xx,yy,ncc,nrr,mask,ind1,ind2;
     long sleeper,swap;
     int kbhit(void);
```

```
    int zz,zz1,brr,brr1;
    double  ang,ang1,ang2;

      int enbl_port,enbl_robot,status,robot_mode,a;
     int prog_id,posi_id,starter,reg_id;
    float min,max,wd[512];
    unsigned char ys,buf1[8];
   short vs[512];
    char buf[30],str[10],stringa[13];

  float coordx,coordy,coordz,coordq1,coordq2,coordq3,coordq4;
 double w,x,y,z,s,aa,b,c,d,e,f,g,t,k,m,n,o,ap,q,r,u,v;
double L,Z,Y,X,dx,dy,dz,zi,alfa,beta,gamma;
  short pos[7];
    float tensione,tensione_max;
   float pa,fp,increy;
   long timer(),fine;
    double tempo,max_tempo;
   unsigned coerr;
    struct timeb timebuffer;
   short start,stop;
   max_tempo=50;
  coerr=0;
 zz=0;
 zz1=0;
L=380.0;
/***************** inizializzazione schermo ****************/
   clrscrn();
 printf("\n**** ACQUISIZIONE E MOVIMENTAZIONE REMOTA ****");
  printf("\n                          *lettura dati dopo sync LeCroy* \n");

  posi_id=50;
    prog_id=1008;   // Numero del programma S3

   starter=0;          /* 0=inizia, 1= continua programma S3 */

/****** controllo GPIB***/

  if((nd=ibfind("DEV4"))<0)
 {
     printf("\nInterfaccia GPIB :   Errore di canale !");
    exit(-1);
 }

/****** controllo S3***/
```

```
        enbl_robot=0;
        enbl_port=0;
           set_robot_on(enbl_robot,enbl_port,&status);
         cntrl_status(status);
          read_rob_stat(stat_dat,&status);
         cntrl_status(status);

if(stat_dat[2]==1||stat_dat[3]==0||stat_dat[4]==1||stat_dat[5]==1||stat_dat[6
]!=0)
    {
      printf("\a");
       leggi_stampa_stato();
      exit(-1);
    }

   /*****inserimento  parametri****/

    cmd_prog_start(prog_id,starter,&status);

   att_supcon();

  found:
  ind2=1;
   while(ind2)
 {
      printf("\nIntroduci coord.X di partenza (1083) :  ");
     scanf("%f",&coordx);

      printf("\nIntroduci coord.Y di partenza (-439.5) :  ");
     scanf("%f",&coordy);

      printf("\nIntroduci coord.Z di partenza (1033.6) :  ");
     scanf("%f",&coordz);

  /********** sonda sulla posizione di partenza ***********/

        pos[0]=(int)coordx*8;
        pos[1]=(int)coordy*8;
        pos[2]=(int)coordz*8;
        pos[3]=11591;
       pos[4]=0;
        pos[5]=11591;
       pos[6]=0;

        move_dat[8]= pos[0];     /*X*/
```

```
        move_dat[9]= pos[1];       /*Y*/
        move_dat[10]= pos[2];      /*Z*/
        move_dat[11]= pos[3];      /*Q1*/
        move_dat[12]= pos[4];      /*Q2*/
        move_dat[13]= pos[5];      /*Q3*/
        move_dat[14]= pos[6];      /*Q4*/

         movi(move_dat);
/*********************************************/

       while(1)
     {
    printf("\n POSIZIONE O.K ?  (y/n)  ?:  ");
    scanf("%1s",&ys);
     if(ys==cy||ys==ccy||ys==cn||ys==ccn) break;
     printf("\a");
     }
     if(ys==cy||ys==ccy) ind2=0;
 }
 /******** prosegue inserimento parametri ********/

   X=coordx;
   Y=coordy;
   Z=coordz;

  posi_dat[0]=(int)coordx*8;
  posi_dat[1]=(int)coordy*8;
  posi_dat[2]=(int)coordz*8;
   posi_dat[3]=0.7071*16393;
 posi_dat[4]=0;
  posi_dat[5]=0.7071*16393;
 posi_dat[6]=0;

       write_position(posi_id,posi_dat,&status);

  starter=1;
   cmd_prog_start(prog_id,starter,&status);

     att_supcon();

      printf("\nIntroduci angolo nel piano xz ");
     printf("\nBETA:  ");

    scanf("%lf",&t);

       t=(t/2.0)*PI/180.0;
```

```
    w= cos (t);
    x= sin (t) * 0;
    y= sin (t) * 1;
    z= sin (t) * 0;

    beta=t * 2.0;

t=0.0;

      t=(t/2.0)*PI/180.0;
    s= cos (t);
    aa= sin (t) * 0;
    b= sin (t) * 0;
    c= sin (t) * 1;
    d=w*s-(x*aa+y*b+z*c);
    e=w*aa+x*s+y*c-z*b;
    f=w*b+y*s+z*aa-x*c;
    g=w*c+z*s+x*b-y*aa;

     printf("\nIntroduci angolo nel piano yz  ");
    printf("\nALFA:   ");

    scanf("%lf",&t);

        t=(t)*PI/180.0;
         t=atan(tan(t)*cos(beta));
       alfa=t;

     t=(t/2.0);
    m= cos (t);
    n= sin (t) * 1;
    o= sin (t) * 0;
    ap= sin (t) * 0;
     q=d*m-(e*n+f*o+g*ap);
     r=d*n+e*m+f*ap-g*o;
     u=d*o+f*m+g*n-e*ap;
     v=d*ap+g*m+e*o-f*n;

     printf("\n X Y Z:\n %lf  %lf %lf",X,Y,Z);

      dx  =  (Z-L)*sin(beta)*cos(alfa)+X*(cos(beta)-1)+Y*sin(beta)*sin(alfa);
     dy =- (Z-L)*sin(alfa)+Y*(cos(alfa)-1);
      dz  =  (Z-L)*(cos(alfa)*cos(beta)-1)-X*sin(beta)+Y*cos(beta)*sin(alfa);

     printf("\nbeta alfa  dx dy  dz:\n  %lf %lf  %lf %lf
%lf",beta,alfa,dx,dy,dz);
```

```
    getch();

      cambia_frame(dx,dy,dz,q,r,u,v);

  /*****inserimento  parametri****/

    while(1)
     {
    printf("\nSCANSIONE ASSE X  ([1,400]mm.)   ?:  ");
      res=getint(3,1,27,1,0,&nc,1,1,400);
    if(res == -4) exit(-1);
    if(nc>0 && nc<401) break;
     }

    while(1)
     {
    printf("\nSCANSIONE ASSE Y  ([1,400]mm.)   ?:  ");
      res=getint(3,1,27,1,0,&nr,1,1,400);
    if(res == -4) exit(-1);
    if(nr>0 && nr<401) break;
     }

    while(1)
     {
    printf("\nPASSO INCREMENTALE ([1,100]decimi)  ?:  ");
      res=getint(3,1,27,1,0,&p,1,1,100);
    if(res == -4) exit(-1);
    if(p>0 && p<101) break;
     }

    ncc=nc*10/p;
     nrr=nr*10/p;

   if(ncc > 400  ||  nrr > 400)
   {
     printf("\a");
      printf("\n\nERRORE : Il numero effettivo di righe o colonne e'
maggiore di 400 !!\n");
    goto found;
   }

 cr='c';
    while(1)
     {
    printf("\nSOGLIA VALORE MASSIMO [0.1 , 9.9]  ?:  ");
      res=getfloat(1,1,1,27,1,0,&max,1,0.,10.);
```

```
 if(res == 27) exit(-1);
if(max > 0.  &&  max < 10.) break;
 printf("\a");
  }
 while(1)
  {
printf("\nSOGLIA VALORE MIMIMO [-1 , < max  ; CR=0.]    ?:  ");
   res=getfloat(2,1,1,27,1,0,&min,1,-1.,max);
 if(res == 27) exit(-1);
if(min >= -1.  &&  min < max) break;
 printf("\a");
  }
ind1=1;
 while(ind1)
  {
printf("\nMEMORIA DI QUADRO (0,1,2,3)   ?:  ");
   res=getint(1,1,27,1,0,&mm,1,0,3);
 if(res == -4) exit(-1);
 if(mm>-1&&mm<4)
{
 a_displ(mm);
   while(1)
 {
 printf("\nO.K  (y/n)  ?:  ");
 scanf("%1s",&ys);
  if(ys==cy||ys==ccy||ys==cn||ys==ccn) break;
 printf("\a");
 }
 if(ys==cy||ys==ccy) ind1=0;
}
   }
 val=0;
 a_set(val);

  printf("\nNumero righe = %d    Numero colonne = %d\n",nrr,ncc);
 xx=100;
 yy=99;
  memset(buf,0,30);
  memset(buf1,0,8);
  memset(pos,0,7);
 ibclr(nd);
 ibwrt(nd,"CHDR OFF",8);          /*elimina testate messaggi*/
 ibwrt(nd,"ACAL OFF",8);          /*inibisce autocalibrazione*/
 ibwrt(nd,"CRMS OFF",8);          /*inibisce visione parametri*/

/******************** start programma robot ***************/
```

```
 starter=1;
      cmd_prog_start(prog_id,starter,&status);
     cntrl_status(status);
    att_supcon();

/******************** trasferimento parametri **************/
          reg_id=4;
          reg_dat=p ;                    /*passo incr. p*/
             write_reg(reg_id,&reg_dat,&status);
            cntrl_status(status);

          reg_id=6;
          reg_dat=ncc;                   /*passi X,px*/
             write_reg(reg_id,&reg_dat,&status);
            cntrl_status(status);

          reg_id=7;
          reg_dat=nrr;                   /*passi Y,py*/
             write_reg(reg_id,&reg_dat,&status);
            cntrl_status(status);

/*************************************************/
           starter = 1;         /* prosegue prog in S3 */
              cmd_prog_start(prog_id,starter,&status);
            cntrl_status(status);
/*************************************************/

  switch(cr)
        {
          case 'c': case 'C':
           strcpy(stringa,"C1:PAVA? MAX");
          mask=1;
          break;

          case 'e': case 'E':
           strcpy(stringa,"FE:PAVA? MAX");
           mask=1024;
          break;

          case 'f': case 'F':
           strcpy(stringa,"FF:PAVA? MAX");
           mask=2048;
          break;
        }

/******************** ciclo di lettura dati ********************/
```

```
 for (am=0;am<nrr; am++)                    /*ciclo avanz. asse Y*/
   {
    for (an=0; an<ncc; an++)              /*ciclo avanz. asse X */
   {

       do              /*attesa sincr. di pos.*/
      {
  read_spon(spon_dat,&status);
 if (status != -22) cntrl_status(status);
}
 while (spon_dat[0] !=0 || status == -22);

        ftime(&timebuffer);
          start=timebuffer.millitm; /*controllo tm*/

 while(1)    /*  attesa per bit di INR=1 (segnale acquisito, etc..) */
 {
    ibwrt(nd,"INR?",4);
     ibrd(nd,buf1,8);
      if(atoi(buf1)&mask)  break;

      ftime(&timebuffer);
        stop=timebuffer.millitm; /*controllo tm*/
     if(stop<start)
     stop+=1000;
      tempo=stop-start;
      if(tempo>max_tempo)
      {
          printf("\n %d  %d %d %f ",am,an,(atoi(buf1)&mask),tempo);
      coerr++;
       break;
      }
        start=stop;
 }
       ibwrt(nd,stringa,12);
      ibrd(nd,buf,16);
     wd[an]=atof(buf+4);

 }                              /* For interno */
        att_supcon();

/********************** scrittura riga su PIP ***************/

          for (an=0; an<ncc; an++)
          {
```

```
                vs[an]=((wd[an]-min)/(max-min))*255;
              if(vs[an] < 0) vs[an]=0;
               else if(vs[an] > 255) vs[an]=255;
          }

     yy=yy+1;
      a_rowwi(xx,yy,ncc,vs);
     if(kbhit() != 0  &&  getch() == esc)
    {
       nrr=am+1;
       break;
    }

     starter = 1;                          /* prosegue prog in S3 */
        cmd_prog_start(prog_id,starter,&status);

    }                                  /* For esterno */

     printf("\nNum righe = %d     Num colonne = %d",(nrr+27),ncc);
      printf("\noNUM.ACQUISIZIONI = %.0f",(double)nrr*(double)ncc);
     printf("\nNUM.ERRORI = %d",coerr);

   /*************** disconnessione robot ed oscilloscopio *********/

     scon(nd,status,enbl_robot);
}
/**************** controllo dello stato **********************/

  cntrl_status(status)
      int status;
{
     if (status <= 0)
    {
      printf("\n\nControllo S3 :   Errore di stato  %d",status);
      leggi_stampa_stato();
     exit (-1);
    }
      return(0);
}
 /************* disconnessione robot ed oscilloscopio **********/

  scon(nd,status,enbl_robot)
       int nd,status,enbl_robot;
{
     cmd_prog_stop(&status);
     cntrl_status(status);
```

```
      set_robot_off(enbl_robot,&status);
      cntrl_status(status);
   ibwrt(nd,"BUZZ BEEP",9);
    ibloc(nd);                          /* rilascia canale */
     return(0);
/********************  attesa supercontr.  ********************/

 att_supcon()
{
    int status;
do
  {  read_spon(spon_dat,&status);
    if (status != -22) cntrl_status(status); }
   while (spon_dat[0] !=0 || status == -22);
     cmd_prog_stop(&status);
     cntrl_status(status);
    return(0);
}
/******************   leggi stampa stato   **************/

  leggi_stampa_stato()
{
   int status;
     read_rob_stat(stat_dat,&status);

  printf("\n\nProgramma   selezionato  :  %d",stat_dat[0]);
  printf("\nIstruzione  selezionata  :  %d",stat_dat[1]);
 if(stat_dat[2] == 0)
     printf("\nStandby : NO");
 else
     printf("\nStandby : SI");
 if(stat_dat[3] == 0)
     printf("\nOperation : NO");
 else
     printf("\nOperation : SI");
 if(stat_dat[4] == 0)
     printf("\nProgram exec : NO");
 else
     printf("\nProgram exec : SI");
 if(stat_dat[5] == 0)
     printf("\nEmergency robot : NO");
 else
     printf("\nEmergency robot : SI");
 if(stat_dat[6] == 0)
     printf("\nTeach pendant : IN");
 else
```

```
    printf("\nTeach pendant : OUT,DISCONNECTED");
  if(stat_dat[7] == 0)
      printf("\nInterrupt : ENABLED");
  else
      printf("\nInterrupt : DISABLED");
  printf("\nX, Y, Z      : %d  %d  %d",stat_dat[8],stat_dat[9],stat_dat[10]);
  printf("\nQ1, Q2, Q3, Q4 : %d  %d  %d
%d",stat_dat[11],stat_dat[12],stat_dat[13],stat_dat[14]);
  if(stat_dat[29] == 0)
      printf("\nOrientation : WRIST");
  else
     printf("\nOrientation : TOOL");
   printf("\nTCP corrente : %d",stat_dat[30]);
    printf(" ");
  printf("\n\nPREMERE UN TASTO PER CONTINUARE");
   getch();
    return(0);
  }

   /******** BEMOVI:C unione di movimentazione remota ***/

    move_array move_dat;
    stat_array stat_dat;
    spon_array spon_dat;

    movi(move_dat)
    move_array move_dat;

  {
   int frame_id;
   int tcp_id;
  int i;
        int enbl_port,enbl_robot,out_no,out_dat,robot_mode,status,a;
       int prog_id,starter,reg_id;

    move_dat[0]=0;
    move_dat[1]=0;
    move_dat[2]=0;
    move_dat[3]=1;
    move_dat[4]=0;
    move_dat[5]=0;
     move_dat[6]=5800;
     move_dat[7]=8142;
    move_dat[15]=0;
    move_dat[16]=0;
    move_dat[17]=0;
```

```
 move_dat[18]=0;
 move_dat[19]=0;
 move_dat[20]=0;
 move_dat[21]=0;
 move_dat[22]=0;
 move_dat[23]=0;
 move_dat[24]=0;
 move_dat[25]=0;
 move_dat[26]=0;
 move_dat[27]=0;
 move_dat[28]=0;
 move_dat[29]=0;

i=3;
 move_dat[2]=2;
  cmd_move(move_dat,&status);
  cntrl_status(status,i);

i=4;
 move_dat[2]=0;
  cmd_move(move_dat,&status);
  cntrl_status(status,i);

i=5;
 move_dat[2]=3;
  cmd_move(move_dat,&status);
  cntrl_status(status,i);
 return(0);
}

  cambia_frame(dx,dy,dz,q,r,u,v)
 double dx,dy,dz,q,r,u,v;
{
  frame_array  frame_dat;

 int frame_id,status;
 frame_id=1;

 frame_dat[0]=-dx*8;
 frame_dat[1]=-dy*8;
 frame_dat[2]=-dz*8;
  frame_dat[3]=q*16393;
  frame_dat[4]=r*16393;
  frame_dat[5]=u*16393;
  frame_dat[6]=v*16393;
```

```
    write_frame(frame_id,frame_dat,&status);
   cntrl_status(status);

  return(0);
}
```

Programma PIAORI

```
/********* AQUISIZIONE E MOVIMENTAZIONE REMOTA *********/
  /* esame pezzo comandato dal calcolatore lettura dati dopo sync
LeCroy, */
  /* con introduzione delle coordinate di partenza con possibilità di
cambiarle*/
  /* con il calcolo delle coordinate X */
  /* con superficie di scansione comunque orientata*/

  #include "\gpib-pc\decl.h"
 #include <stdio.h>
 #include <conio.h>
 #include <math.h>
 #include <stdlib.h>
  #include "c:\robinc\ctsttype.h"
  #include "c:\robinc\ctstfunc.h"
 #include <time.h>
  #include <sys\types.h>
  #include <sys\timeb.h>

  #define PI 3.14159265359
 #define cc      'c'
 #define ccc     'C'
 #define ce      'e'
 #define cce     'E'
 #define cf      'f'
 #define ccf     'F'
 #define cy      'y'
 #define ccy     'Y'
 #define cn      'n'
 #define ccn     'N'
 #define esc      '\x1b'

  float coordx,coordy,coordz,leggi_tensione();

  FILE *stream;
  stat_array stat_dat;
  spon_array spon_dat;
  move_array move_dat;
/*************************************************/

  main()
  {
       int cr,i,l,m,n,num,nd,nr,p,nc,reg_dat,res,mm,val;
```

```
    int lr,xx,yy,ncc,nrr,mask,ind1,ind2;
    long sleeper,swap;
    int kbhit(void);

    int zz,zz1,brr,brr1;
    double  ang,ang1,ang2;
    double L,w,xq,y,z,s,aq,b,c,d,e,f,g,t,k,mq,nq,o,pq,q,r,u,v;
    double tx,ty,tz,dx,dy,dz,alfa,beta;
     int enbl_port,enbl_robot,status,robot_mode,a;
    int prog_id,starter,reg_id;
    float x,min,max,wd[512];
    unsigned char ys,buf1[8];
    short vs[512];
    char buf[30],str[10],stringa[13];

   short pos[8];
    float tensione,tensione_max;
    float pa,fp,increy,increx,increzy,increzz;
    long timer(),fine;
    double tempo,max_tempo;
    unsigned coerr;
    struct timeb timebuffer;
    short start,stop;
    max_tempo=50;
   coerr=0;
  zz=0;
  zz1=0;
 L= 380.0; /* distanza del TCP dalla superficie in esame */
/*************** inizializzazione schermo ***************/
   clrscrn();
  printf("\n*** ACQUISIZIONE E MOVIMENTAZIONE REMOTA ****");
   printf("\n                          *lettura dati dopo sync LeCroy* \n");

   prog_id=2015;                    /* Numero del programma S3 */
   starter=0;                 /* 0=inizia, 1= continua programma S3 */

/****** controllo GPIB***/

   if((nd=ibfind("DEV4"))<0)
  {
      printf("\nInterfaccia GPIB :   Errore di canale !");
     exit(-1);
  }
/****** controllo S3***/
   enbl_robot=0;
   enbl_port=0;
```

```
        set_robot_on(enbl_robot,enbl_port,&status);
       cntrl_status(status);
        read_rob_stat(stat_dat,&status);
       cntrl_status(status);

if(stat_dat[2]==1||stat_dat[3]==0||stat_dat[4]==1||stat_dat[5]==1||stat_dat[6
]!=0)
   {
     printf("\a");
      leggi_stampa_stato();
     exit(-1);
   }

 /***************apertura file **************/
  aprif();
 /*************************************/
   /*****inserimento  parametri****/

  found:
  ind2=1;
   while(ind2)
  {
      printf("\nIntroduci coord.X di partenza (1083) :  ");
     scanf("%f",&coordx);

      printf("\nIntroduci coord.Y di partenza (-439.5) :  ");
     scanf("%f",&coordy);

      printf("\nIntroduci coord.Z di partenza (1033.6) :  ");
     scanf("%f",&coordz);

      printf("\nIntroduci angolo di rotazione intorno asse Y ");
      printf("\nSingolarmente sposta la sonda nel piano xz:  ");
     scanf("%lf",&ty);

      beta=(ty*PI/180.0);

    ty=ty+90.0;

      t=(ty/2.0)*PI/180.0;
     w= cos (t);
     xq= sin (t) * 0;
     y= sin (t) * 1;
     z= sin (t) * 0;

      printf("\nIntroduci angolo di rotazione t intorno all'asse Z  ");
```

```
    printf("\nSingolarmente sposta la sonda nel piano xy:    ");
   scanf("%lf",&tz);

        t=(tz)*PI/180.0;
         t=atan(tan(t)*cos(beta));
       alfa=t;

    t=(t/2.0);
   s= cos (t);
   aq= sin (t) * 0;
   b= sin (t) * 0;
   c= sin (t) * 1;

    d=w*s-(xq*aq+y*b+z*c);
    e=w*aq+xq*s+y*c-z*b;
    f=w*b+y*s+z*aq-xq*c;
    g=w*c+z*s+xq*b-y*aq;

    printf("\nprima beta alfa:\n  %lf %lf",beta, alfa);

     dx =-L*cos(alfa)*sin(beta);
    dy = L*sin(alfa);
     dz = L*(1-cos(alfa)*cos(beta));

    printf("\nRisultanti  beta alfa:\n  %lf %lf",beta, alfa);
     printf("\nRisultanti dx dy  dz:\n %lf %lf %lf",dx, dy,dz);
    getch();

tx=0.0;
     t=(tx/2.0)*PI/180.0;
   mq= cos (t);
   nq= sin (t) * 1;
   o= sin (t) * 0;
   pq= sin (t) * 0;

    q=d*mq-(e*nq+f*o+g*pq);
    r=d*nq+e*mq+f*pq-g*o;
    u=d*o+f*mq+g*nq-e*pq;
    v=d*pq+g*mq+e*o-f*nq;
    printf("\nRisultanti q  r  u  v:\n  %lf %lf %lf %lf",q,r,u,v);
/****************** start programma robot ************/

      cmd_prog_start(prog_id,starter,&status);
     cntrl_status(status);
    att_supcon();
/********** sonda sulla posizione di partenza ************/
```

```
        pos[0]=(coordx-dx)*8;
        pos[1]=(coordy-dy)*8;
        pos[2]=(coordz-dz)*8;
        pos[3]=q*16393;
        pos[4]=r*16393;
        pos[5]=u*16393;
        pos[6]=v*16393;

        move_dat[8]= pos[0];     /*X*/
        move_dat[9]= pos[1];      /*Y*/
        move_dat[10]= pos[2];     /*Z*/
        move_dat[11]= pos[3];      /*Q1*/
        move_dat[12]= pos[4];      /*Q2*/
        move_dat[13]= pos[5];      /*Q3*/
        move_dat[14]= pos[6];      /*Q4*/

        movi(move_dat);
/************************************************/

      while(1)
     {
    printf("\n POSIZIONE O.K ?  (y/n)  ?:  ");
    scanf("%1s",&ys);
     if(ys==cy||ys==ccy||ys==cn||ys==ccn) break;
     printf("\a");
     }
     if(ys==cy||ys==ccy) ind2=0;
 }
 /******** prosegue inserimento parametri ********/

    while(1)
     {
    printf("\nSCANSIONE ASSE X  ([1,600]mm.)   ?:  ");
      res=getint(3,1,27,1,0,&nc,1,1,600);
     if(res == -4) exit(-1);
     if(nc>0 && nc<601) break;
      }

    while(1)
     {
    printf("\nSCANSIONE ASSE Y  ([1,999]mm.)   ?:  ");
      res=getint(3,1,27,1,0,&nr,1,1,999);
     if(res == -4) exit(-1);
     if(nr>0 && nr<1000) break;
      }
```

```
      while(1)
       {
      printf("\nPASSO INCREMENTALE ([1,100]decimi)  ?:  ");
        res=getint(3,1,27,1,0,&p,1,1,100);
      if(res == -4) exit(-1);
      if(p>0 && p<101) break;
       }

      ncc=nc*10/p;
       nrr=nr*10/p;

       fp=(float)p;

        increx = fp/10.0*8.0;
        increy = fp/10.0*8.0;
         increzy = ((fp/10.0) * (float)tan(tz*PI/180.0))*8.0;
        increzz = ((fp/10.0) * (float)tan(beta))*8.0;
       printf("\nincrex increy increzy increzz:\n  %lf %lf %lf
%lf",increx,increy,increzy,increzz);

  cr='c';

      while(1)
       {
      printf("\nSOGLIA VALORE MASSIMO [0.1 , 9.9]  ?:  ");
        res=getfloat(1,1,1,27,1,0,&max,1,0.,10.);
      if(res == 27) exit(-1);
      if(max > 0.  &&  max < 10.) break;
       printf("\a");
       }
      while(1)
       {
      printf("\nSOGLIA VALORE MIMIMO [-1 , < max  ; CR=0.]    ?:  ");
        res=getfloat(2,1,1,27,1,0,&min,1,-1.,max);
      if(res == 27) exit(-1);
      if(min >= -1.  &&  min < max) break;
       printf("\a");
       }
      ind1=1;
       while(ind1)
       {
      printf("\nMEMORIA DI QUADRO (0,1,2,3)   ?:  ");
        res=getint(1,1,27,1,0,&mm,1,0,3);
      if(res == -4) exit(-1);
      if(mm>-1&&mm<4)
```

```
{
 a_displ(mm);
  while(1)
 {
 printf("\nO.K  (y/n)  ?:  ");
 scanf("%1s",&ys);
  if(ys==cy||ys==ccy||ys==cn||ys==ccn) break;
 printf("\a");
 }
 if(ys==cy||ys==ccy) ind1=0;
}
  }
val=0;
 a_set(val);

  printf("\nNumero righe = %d    Numero colonne = %d\n",nrr,ncc);
xx=100;
yy=99;
  memset(buf,0,30);
  memset(buf1,0,8);

 ibclr(nd);
 ibwrt(nd,"CHDR OFF",8);          /*elimina testate messaggi*/
 ibwrt(nd,"ACAL OFF",8);          /*inibisce autocalibrazione*/
 ibwrt(nd,"CRMS OFF",8);          /*inibisce visione parametri*/

/********** sonda sulla posizione di partenza ***********/

    move_dat[8]= pos[0];    /*X*/
    move_dat[9]= pos[1];     /*Y*/
    move_dat[10]= pos[2];    /*Z*/
    move_dat[11]= pos[3];     /*Q1*/
    move_dat[12]= pos[4];     /*Q2*/
    move_dat[13]= pos[5];     /*Q3*/
    move_dat[14]= pos[6];     /*Q4*/

     movi(move_dat);
/***********************************************/

  switch(cr)
      {
        case 'c': case 'C':
          strcpy(stringa,"C1:PAVA? MAX");
        mask=1;
        break;
```

```
            case 'e': case 'E':
              strcpy(stringa,"FE:PAVA? MAX");
              mask=1024;
             break;

            case 'f': case 'F':
              strcpy(stringa,"FF:PAVA? MAX");
              mask=2048;
             break;
          }
       pos[8]=pos[2];
/******************** ciclo di lettura dati **************/

  for (m=0; m<(nrr); m++)       /*  ciclo avanz. asse Y*/

  {
     for (n=0; n<ncc; n++)        /*ciclo avanz. asse X */
     {

      move_dat[8]= pos[0] - n*increx;    /*X*/
      move_dat[10]= pos[8] + n*increzz;      /*Z*/
      movi(move_dat);

      impulso_tras();

        ftime(&timebuffer);
          start=timebuffer.millitm; /*controllo tm*/

   while(1)           /*  attesa per bit di INR=1 (segnale acquisito, etc..) */
    {  ibwrt(nd,"INR?",4);
       ibrd(nd,buf1,8);
        if(atoi(buf1)&mask)  break;

       ftime(&timebuffer);
         stop=timebuffer.millitm; /*controllo tm*/
      if(stop<start)
      stop+=1000;
       tempo=stop-start;
       if(tempo>max_tempo)
       {
          printf("\n %d  %d %d %f ",m,n,(atoi(buf1)&mask),tempo);
       coerr++;
        break;
       }
         start=stop;
   }
```

```
        ibwrt(nd,stringa,12);
        ibrd(nd,buf,16);
        wd[n]=atof(buf+4);
   }                              /* For interno */

/****************** scrittura riga su PIP ***************/

     for (n=0; n<ncc; n++)
     {
            vs[n]=((wd[n]-min)/(max-min))*255;
        if(vs[n] < 0) vs[n]=0;
         else if(vs[n] > 255) vs[n]=255;
     }
      yy=yy+1;
       a_rowwi(xx,yy,ncc,vs);

      if(kbhit() != 0  &&  getch() == esc)
     {
        nrr=m+1;
        break;
     }

 /****** comando da calcolatore*******/

 pos[8]= pos[2] + m*increzy;
   move_dat[8]= pos[0];     /*X*/
   move_dat[9]= pos[1] + m*increy;       /*Y*/
   move_dat[10]= pos[2] + m*increzy;      /*Z*/

    movi(move_dat);

     }                              /* For esterno */

      printf("\nNum righe = %d     Num colonne = %d",(nrr+27),ncc);
       printf("\nNUM.ACQUISIZIONI = %.0f",(double)nrr*(double)ncc);
      printf("\nNUM.ERRORI = %d",coerr);

     /********** disconnessione robot ed oscilloscopio ******/
      scon(nd,status,enbl_robot);
}
/***************  controllo dello stato  ***************/

  cntrl_status(status)
      int status;
{
     if (status <= 0)
```

```
    {
      printf("\n\nControllo S3 :   Errore di stato  %d",status);
      leggi_stampa_stato();
     exit (-1);
    }
      return(0);

/********** disconnessione robot ed oscilloscopio *********/

  scon(nd,status,enbl_robot)
       int nd,status,enbl_robot;
{
     cmd_prog_stop(&status);
     cntrl_status(status);
      set_robot_off(enbl_robot,&status);
     cntrl_status(status);
   ibwrt(nd,"BUZZ BEEP",9);
   ibloc(nd);                          /* rilascia canale */
    return(0);
}
/*************  attesa supercontr.  ********************/

 att_supcon()
{
    int status;
do
  {  read_spon(spon_dat,&status);
    if (status != -22) cntrl_status(status); }
   while (spon_dat[0] !=0 || status == -22);
     cmd_prog_stop(&status);
     cntrl_status(status);
    return(0);
}
/******************  leggi stampa stato  **************/

  leggi_stampa_stato()
{
   int status;
     read_rob_stat(stat_dat,&status);

 printf("\n\nProgramma   selezionato : %d",stat_dat[0]);
 printf("\nIstruzione selezionata : %d",stat_dat[1]);
if(stat_dat[2] == 0)
    printf("\nStandby : NO");
else
     printf("\nStandby : SI");
```

```
if(stat_dat[3] == 0)
    printf("\nOperation : NO");
else
    printf("\nOperation : SI");
if(stat_dat[4] == 0)
    printf("\nProgram exec : NO");
else
    printf("\nProgram exec : SI");
if(stat_dat[5] == 0)
    printf("\nEmergency robot : NO");
else
    printf("\nEmergency robot : SI");
if(stat_dat[6] == 0)
    printf("\nTeach pendant : IN");
else
   printf("\nTeach pendant : OUT,DISCONNECTED");
if(stat_dat[7] == 0)
    printf("\nInterrupt : ENABLED");
else
    printf("\nInterrupt : DISABLED");
printf("\nX, Y, Z      : %d  %d  %d",stat_dat[8],stat_dat[9],stat_dat[10]);
printf("\nQ1, Q2, Q3, Q4 : %d  %d  %d
%d",stat_dat[11],stat_dat[12],stat_dat[13],stat_dat[14]);
if(stat_dat[29] == 0)
    printf("\nOrientation : WRIST");
else
    printf("\nOrientation : TOOL");
printf("\nTCP corrente : %d",stat_dat[30]);
printf(" ");
printf("\n\nPREMERE UN TASTO PER CONTINUARE");
getch();
return(0);
}

/******** BEMOVI:C unione di movimentazione remota ***/

move_array move_dat;
stat_array stat_dat;
spon_array spon_dat;

movi(move_dat)
move_array move_dat;

{
int frame_id;
int tcp_id;
```

```
int i;
      int  enbl_port,enbl_robot,out_no,out_dat,robot_mode,status,a;
     int  prog_id,starter,reg_id;

 move_dat[0]=0;
 move_dat[1]=0;
 move_dat[2]=0;
 move_dat[3]=1;
 move_dat[4]=0;
 move_dat[5]=0;
  move_dat[6]=1800;
  move_dat[7]=8142;
 move_dat[15]=0;
 move_dat[16]=0;
 move_dat[17]=0;
 move_dat[18]=0;
 move_dat[19]=0;
 move_dat[20]=0;
 move_dat[21]=0;
 move_dat[22]=0;
 move_dat[23]=0;
 move_dat[24]=0;
 move_dat[25]=0;
 move_dat[26]=0;
 move_dat[27]=0;
 move_dat[28]=0;
 move_dat[29]=0;

i=3;
 move_dat[2]=2;
   cmd_move(move_dat,&status);
   cntrl_status(status,i);

i=4;
 move_dat[2]=0;
   cmd_move(move_dat,&status);
   cntrl_status(status,i);

i=5;
 move_dat[2]=3;
   cmd_move(move_dat,&status);
   cntrl_status(status,i);
  return(0);
}

 /********invio impulso al trasmettitore  **********/
```

```
 impulso_tras()
{
     int out_no,out_dat,status;
    out_no=1;
     out_dat=1;
        write_output(out_no,&out_dat,&status);
     out_dat=0;
        write_output(out_no,&out_dat,&status);
 return(0);
}

 aprif()
{
     if ((stream=fopen("ondc.dat","wb"))==NULL)
  {
        printf("file non aperto\n");
      exit(0);
  }
    return(0);
}

 chiudif()
{
    fclose(stream);
}
```

Programma OCINCFR

```
/********* AQUISIZIONE E MOVIMENTAZIONE REMOTA *********/
  /*onda completa */
  /* esame pezzo comandato dal calcolatore lettura dati dopo sync
LeCroy, */
  /* con introduzione delle coordinate di partenza con possibilità di
cambiarle*/
  /* piano inclinato con cambiamento del piano di riferimento*/
   #include "\gpib-pc\decl.h"
  #include <stdio.h>
  #include <conio.h>
  #include <math.h>
  #include <stdlib.h>
   #include "c:\robinc\ctsttype.h"
   #include "c:\robinc\ctstfunc.h"
  #include <time.h>
   #include <sys\types.h>
   #include <sys\timeb.h>

   #define PI 3.14159265359
  #define cc      'c'
  #define ccc     'C'
  #define ce      'e'
  #define cce     'E'
  #define cf      'f'
  #define ccf     'F'
  #define cy      'y'
  #define ccy     'Y'
  #define cn      'n'
  #define ccn     'N'
  #define esc      '\x1b'

   float coordx,coordy,coordz,leggi_tensione();

   posi_array posi_dat;
   stat_array stat_dat;
   spon_array spon_dat;
   move_array move_dat;
 /*********************************************/

  main()
 {
       int cr,i,l,am,an,num,nd,nr,p,nc,reg_dat,res,j,mm,val,bb;
      int lr,xx,yy,ncc,nrr,mask,ind1,ind2;
     long sleeper,swap;
```

```
    int kbhit(void);

    int zz,zz1,brr,brr1;
    double  ang,ang1,ang2;

      int enbl_port,enbl_robot,status,robot_mode,a;
     int prog_id,posi_id,starter,reg_id;
    float min,max;
     unsigned char ys,buf1[8];
    short vs[512];
     char buf[600],str[10],stringa[13];
     unsigned char wd[512];
   float coordx,coordy,coordz,coordq1,coordq2,coordq3,coordq4;
  double w,x,y,z,s,aa,b,c,d,e,f,g,t,k,m,n,o,ap,q,r,u,v;
  double L,Z,Y,X,dx,dy,dz,zi,alfa,beta,gamma;
   short pos[7];
    float tensione,tensione_max;
    float pa,fp,increy;
    long timer(),fine;
     double tempo,max_tempo;
    unsigned coerr;
     struct timeb timebuffer;
    short start,stop;
  int ycount=20;
    max_tempo=50;
   coerr=0;
  zz=0;
  zz1=0;
  L=380.0;
   mask=1;        /*** attesa oscill. pronto*********/
/************* inizializzazione schermo ***************/

   clrscrn();
  printf("\n *** ACQUISIZIONE E MOVIMENTAZIONE REMOTA ****");
   printf("\n                          *lettura dati dopo sync LeCroy* \n");

   posi_id=50;
    prog_id=1008;  // Numero del programma S3

   starter=0;              /* 0=inizia, 1= continua programma S3 */

/****** controllo GPIB***/
   if((nd=ibfind("DEV4"))<0)
  {
      printf("\nInterfaccia GPIB :   Errore di canale !");
     exit(-1);
```

```
    }

  /****** controllo S3***/
      enbl_robot=0;
      enbl_port=0;
        set_robot_on(enbl_robot,enbl_port,&status);
       cntrl_status(status);
        read_rob_stat(stat_dat,&status);
       cntrl_status(status);

if(stat_dat[2]==1||stat_dat[3]==0||stat_dat[4]==1||stat_dat[5]==1||stat_dat[6
]!=0)
    {
      printf("\a");
       leggi_stampa_stato();
      exit(-1);
    }

   /*****inserimento parametri****/

    cmd_prog_start(prog_id,starter,&status);

   att_supcon();

  found:
  ind2=1;
   while(ind2)
 {
      printf("\nIntroduci coord.X di partenza (1083) :  ");
     scanf("%f",&coordx);

      printf("\nIntroduci coord.Y di partenza (-439.5) :  ");
     scanf("%f",&coordy);

      printf("\nIntroduci coord.Z di partenza (1033.6) :  ");
     scanf("%f",&coordz);

  /********** sonda sulla posizione di partenza ***********/

        pos[0]=(int)coordx*8;
        pos[1]=(int)coordy*8;
        pos[2]=(int)coordz*8;
        pos[3]=11591;
       pos[4]=0;
        pos[5]=11591;
       pos[6]=0;
```

```
        move_dat[8]= pos[0];     /*X*/
        move_dat[9]= pos[1];      /*Y*/
        move_dat[10]= pos[2];     /*Z*/
        move_dat[11]= pos[3];      /*Q1*/
        move_dat[12]= pos[4];      /*Q2*/
        move_dat[13]= pos[5];      /*Q3*/
        move_dat[14]= pos[6];      /*Q4*/

         movi(move_dat);

       while(1)
      {
     printf("\n POSIZIONE O.K ?  (y/n)  ?:  ");
     scanf("%1s",&ys);
      if(ys==cy||ys==ccy||ys==cn||ys==ccn) break;
      printf("\a");
      }
      if(ys==cy||ys==ccy) ind2=0;
}
/******** prosegue inserimento parametri ********/

   X=coordx;
   Y=coordy;
   Z=coordz;

 posi_dat[0]=(int)coordx*8;
 posi_dat[1]=(int)coordy*8;
 posi_dat[2]=(int)coordz*8;
 posi_dat[3]=0.7071*16393;
posi_dat[4]=0;
 posi_dat[5]=0.7071*16393;
posi_dat[6]=0;

      write_position(posi_id,posi_dat,&status);

 starter=1;
  cmd_prog_start(prog_id,starter,&status);

    att_supcon();

     printf("\nIntroduci angolo nel piano xz ");
    printf("\nBETA:  ");

    scanf("%lf",&t);
```

```
    t=(t/2.0)*PI/180.0;
  w= cos (t);
  x= sin (t) * 0;
  y= sin (t) * 1;
  z= sin (t) * 0;

  beta=t * 2.0;

t=0.0;

    t=(t/2.0)*PI/180.0;
  s= cos (t);
  aa= sin (t) * 0;
  b= sin (t) * 0;
  c= sin (t) * 1;
  d=w*s-(x*aa+y*b+z*c);
  e=w*aa+x*s+y*c-z*b;
  f=w*b+y*s+z*aa-x*c;
  g=w*c+z*s+x*b-y*aa;

   printf("\nIntroduci angolo nel piano yz  ");
  printf("\nALFA:   ");

  scanf("%lf",&t);

      t=(t)*PI/180.0;
       t=atan(tan(t)*cos(beta));
    alfa=t;

  t=(t/2.0);
  m= cos (t);
  n= sin (t) * 1;
  o= sin (t) * 0;
  ap= sin (t) * 0;
  q=d*m-(e*n+f*o+g*ap);
  r=d*n+e*m+f*ap-g*o;
  u=d*o+f*m+g*n-e*ap;
  v=d*ap+g*m+e*o-f*n;

  printf("\n X Y Z:\n %lf  %lf %lf",X,Y,Z);

  dx  =  (Z-L)*sin(beta)*cos(alfa)+X*(cos(beta)-1)+Y*sin(beta)*sin(alfa);
 dy =- (Z-L)*sin(alfa)+Y*(cos(alfa)-1);
  dz  =  (Z-L)*(cos(alfa)*cos(beta)-1)-X*sin(beta)+Y*cos(beta)*sin(alfa);
```

```
    printf("\nbeta alfa  dx dy  dz:\n  %lf %lf  %lf %lf
%lf",beta,alfa,dx,dy,dz);
     getch();

       cambia_frame(dx,dy,dz,q,r,u,v);

   /*****inserimento  parametri****/

     while(1)
      {
     printf("\nSCANSIONE ASSE X  ([1,400]mm.)   ?:  ");
       res=getint(3,1,27,1,0,&nc,1,1,400);
     if(res == -4) exit(-1);
     if(nc>0 && nc<401) break;
      }

     while(1)
      {
     printf("\nSCANSIONE ASSE Y  ([1,400]mm.)   ?:  ");
       res=getint(3,1,27,1,0,&nr,1,1,400);
     if(res == -4) exit(-1);
     if(nr>0 && nr<401) break;
      }

     while(1)
      {
     printf("\nPASSO INCREMENTALE ([1,100]decimi)  ?:  ");
       res=getint(3,1,27,1,0,&p,1,1,100);
     if(res == -4) exit(-1);
     if(p>0 && p<101) break;
      }

     ncc=nc*10/p;
      nrr=nr*10/p;

    if(ncc > 512  ||  nrr > 512)
    {
      printf("\a");
       printf("\n\nERRORE : Il numero effettivo di righe o colonne e'
maggiore  di  400  !!\n");
     goto found;
    }

     ind1=1;
      while(ind1)
       {
```

```
   printf("\nMEMORIA DI QUADRO (0,1,2,3)   ?:  ");
     res=getint(1,1,27,1,0,&mm,1,0,3);
    if(res == -4) exit(-1);
    if(mm>-1&&mm<4)
    {
     a_displ(mm);
      while(1)
     {
    printf("\nO.K  (y/n)  ?:  ");
    scanf("%1s",&ys);
     if(ys==cy||ys==ccy||ys==cn||ys==ccn) break;
     printf("\a");
     }
     if(ys==cy||ys==ccy) ind1=0;
    }
      }
   val=0;
    a_set(val);
       printf("\nNUM.ACQUISIZIONI = %.0f",(double)nrr*(double)ncc);

   xx=100;
   yy=99;
     memset(buf,0,30);
     memset(buf1,0,8);

    ibclr(nd);
    ibwrt(nd,"CHDR OFF",8);          /*elimina testate messaggi*/
    ibwrt(nd,"ACAL OFF",8);          /*inibisce autocalibrazione*/
    ibwrt(nd,"CRMS OFF",8);          /*inibisce visione parametri*/

/************** start programma robot ************/
 starter=1;
      cmd_prog_start(prog_id,starter,&status);
     cntrl_status(status);
    att_supcon();

/*************** trasferimento parametri **********/
          reg_id=4;
          reg_dat=p ;                    /*passo incr. p*/
             write_reg(reg_id,&reg_dat,&status);
            cntrl_status(status);

          reg_id=6;
          reg_dat=ncc;                   /*passi X,px*/
             write_reg(reg_id,&reg_dat,&status);
            cntrl_status(status);
```

```
          reg_id=7;
          reg_dat=nrr;                    /*passi Y,py*/
             write_reg(reg_id,&reg_dat,&status);
           cntrl_status(status);

          starter = 1;                    /* prosegue prog in S3 */
             cmd_prog_start(prog_id,starter,&status);
           cntrl_status(status);

 switch(cr)
       {
          case 'c': case 'C':
           strcpy(stringa,"C1:PAVA? MAX");
          mask=1;
          break;

          case 'e': case 'E':
           strcpy(stringa,"FE:PAVA? MAX");
           mask=1024;
          break;

          case 'f': case 'F':
           strcpy(stringa,"FF:PAVA? MAX");
           mask=2048;
          break;
       }

/*************** ciclo di lettura dati **************/
    ibwrt(nd,"WFSU  SP,8,np,1600,FP,0,sn,0",27);

  for (am=0;am<nrr; am++)                 /*ciclo avanz. asse Y*/
    {
     for (an=0; an<ncc; an++)             /*ciclo avanz. asse X */
    {

       do             /*attesa sincr. di pos.*/
      {
    read_spon(spon_dat,&status);
   if (status != -22) cntrl_status(status);
  }
   while (spon_dat[0] !=0 || status == -22);

         ftime(&timebuffer);
           start=timebuffer.millitm; /*controllo tm*/
```

```
while(1)          /*  attesa per bit di INR=1 (segnale acquisito, etc..) */
{
    ibwrt(nd,"INR?",4);
     ibrd(nd,buf1,8);
      if(atoi(buf1)&mask)  break;

      ftime(&timebuffer);
        stop=timebuffer.millitm; /*controllo tm*/
     if(stop<start)
    stop+=1000;
      tempo=stop-start;
      if(tempo>max_tempo)
      {
          printf("\n %d  %d %d %f ",am,an,(atoi(buf1)&mask),tempo);
      coerr++;
       break;
      }
        start=stop;
}

      ibwrt(nd,"C1:WF? dat1",11);
      ibrd(nd,buf,524);

     for (bb=22,i=0; bb<524; bb+=2,i++)  wd[i]=buf[bb]+127;
          a_roww(0,ycount,i-1,wd);
       ycount++;

}                              /* For interno */
       att_supcon();

   if(kbhit() != 0  &&  getch() == esc)
  {
      nrr=am+1;
      break;
  }

   starter = 1;                              /* prosegue prog in S3 */
       cmd_prog_start(prog_id,starter,&status);

  }                              /* For esterno */

   printf("\nNum righe = %d     Num colonne = %d",(nrr+27),ncc);
    printf("\noNUM.ACQUISIZIONI = %.0f",(double)nrr*(double)ncc);
  printf("\nNUM.ERRORI = %d",coerr);

/******* disconnessione robot ed oscilloscopio *********/
```

```
    scon(nd,status,enbl_robot);
}
/************  controllo dello stato  *****************/

  cntrl_status(status)
      int status;
{
     if (status <= 0)
    {
       printf("\n\nControllo S3 :   Errore di stato  %d",status);
       leggi_stampa_stato();
     exit (-1);
    }
      return(0);
}
 /******* disconnessione robot ed oscilloscopio *****/

  scon(nd,status,enbl_robot)
       int nd,status,enbl_robot;
{
      cmd_prog_stop(&status);
      cntrl_status(status);
       set_robot_off(enbl_robot,&status);
      cntrl_status(status);
   ibwrt(nd,"BUZZ BEEP",9);
   ibloc(nd);                          /* rilascia canale */
    return(0);
}
/***********  attesa supercontr.  **************/

 att_supcon()
{
    int status;
do
  {  read_spon(spon_dat,&status);
    if (status != -22) cntrl_status(status); }
   while (spon_dat[0] !=0 || status == -22);
     cmd_prog_stop(&status);
     cntrl_status(status);
    return(0);
}
/***********   leggi stampa stato   **************/

  leggi_stampa_stato()
{
```

```
   int status;
     read_rob_stat(stat_dat,&status);

  printf("\n\nProgramma   selezionato : %d",stat_dat[0]);
  printf("\nIstruzione selezionata : %d",stat_dat[1]);
 if(stat_dat[2] == 0)
     printf("\nStandby : NO");
 else
     printf("\nStandby : SI");
 if(stat_dat[3] == 0)
     printf("\nOperation : NO");
 else
     printf("\nOperation : SI");
 if(stat_dat[4] == 0)
     printf("\nProgram exec : NO");
 else
     printf("\nProgram exec : SI");
 if(stat_dat[5] == 0)
     printf("\nEmergency robot : NO");
 else
     printf("\nEmergency robot : SI");
 if(stat_dat[6] == 0)
     printf("\nTeach pendant : IN");
 else
    printf("\nTeach pendant : OUT,DISCONNECTED");
 if(stat_dat[7] == 0)
     printf("\nInterrupt : ENABLED");
 else
     printf("\nInterrupt : DISABLED");
 printf("\nX,  Y,  Z      : %d  %d  %d",stat_dat[8],stat_dat[9],stat_dat[10]);
 printf("\nQ1, Q2, Q3, Q4 : %d  %d  %d
%d",stat_dat[11],stat_dat[12],stat_dat[13],stat_dat[14]);
  if(stat_dat[29] == 0)
     printf("\nOrientation : WRIST");
 else
     printf("\nOrientation : TOOL");
  printf("\nTCP corrente : %d",stat_dat[30]);
   printf(" ");
 printf("\n\nPREMERE UN TASTO PER CONTINUARE");
  getch();
  return(0);
}

 /******** BEMOVI:C unione di movimentazione remota ***/

  move_array  move_dat;
```

```
 stat_array  stat_dat;
 spon_array  spon_dat;

 movi(move_dat)
 move_array  move_dat;

{
 int frame_id;
 int tcp_id;
int i;
      int  enbl_port,enbl_robot,out_no,out_dat,robot_mode,status,a;
    int  prog_id,starter,reg_id;

 move_dat[0]=0;
 move_dat[1]=0;
 move_dat[2]=0;
 move_dat[3]=1;
 move_dat[4]=0;
 move_dat[5]=0;
  move_dat[6]=5800;
  move_dat[7]=8142;
 move_dat[15]=0;
 move_dat[16]=0;
 move_dat[17]=0;
 move_dat[18]=0;
 move_dat[19]=0;
 move_dat[20]=0;
 move_dat[21]=0;
 move_dat[22]=0;
 move_dat[23]=0;
 move_dat[24]=0;
 move_dat[25]=0;
 move_dat[26]=0;
 move_dat[27]=0;
 move_dat[28]=0;
 move_dat[29]=0;

  i=3;
 move_dat[2]=2;
   cmd_move(move_dat,&status);
   cntrl_status(status,i);

i=4;
  move_dat[2]=0;
   cmd_move(move_dat,&status);
   cntrl_status(status,i);
```

```
i=5;
 move_dat[2]=3;
  cmd_move(move_dat,&status);
  cntrl_status(status,i);
 return(0);
}

  cambia_frame(dx,dy,dz,q,r,u,v)
 double  dx,dy,dz,q,r,u,v;
{
  frame_array  frame_dat;

 int  frame_id,status;
 frame_id=1;

 frame_dat[0]=-dx*8;
 frame_dat[1]=-dy*8;
 frame_dat[2]=-dz*8;
  frame_dat[3]=q*16393;
  frame_dat[4]=r*16393;
  frame_dat[5]=u*16393;
  frame_dat[6]=v*16393;

   write_frame(frame_id,frame_dat,&status);
  cntrl_status(status);

 return(0);
}
```

Programma BERU1

```
  /* esame pezzo comandato dal calcolatore lettura dati dopo sync
LeCroy, */
  /* con introduzione delle coordinate di partenza */

   #include "\gpib-pc\decl.h"
  #include <stdio.h>
  #include <conio.h>
  #include <math.h>
  #include <stdlib.h>
   #include "c:\robinc\ctsttype.h"
   #include "c:\robinc\ctstfunc.h"
  #include <time.h>
   #include <sys\types.h>
   #include <sys\timeb.h>

   #define PI 3.14159265359
  #define cc      'c'
  #define ccc     'C'
  #define ce      'e'
  #define cce     'E'
  #define cf      'f'
  #define ccf     'F'
  #define cy      'y'
  #define ccy     'Y'
  #define cn      'n'
  #define ccn     'N'
  #define esc      '\x1b'

   float coordx,coordy,coordz,leggi_tensione();

   stat_array stat_dat;
   spon_array spon_dat;
   move_array move_dat;
 /*****************************************************/

  main()
  {
       int cr,i,l,m,n,num,nd,nr,p,nc,reg_dat,res,j,mm,val;
       int lr,xx,yy,ncc,nrr,mask,ind1,ind2;
      long sleeper,swap;
      int kbhit(void);

      int zz,zz1,brr,brr1;
      double  ang,ang1,ang2,raggio;
```

```
    int enbl_port,enbl_robot,status,robot_mode,a;
   int prog_id,starter,reg_id;
  float x,min,max,wd[512];
  unsigned char ys,buf1[8];
 short vs[512];
  char buf[30],str[10],stringa[13];

 short pos[7];
  float tensione,tensione_max;
  float pa,fp,increy,increz;
 long timer(),fine;
  double tempo,max_tempo;
 unsigned coerr;
  struct timeb timebuffer;
  short start,stop;
  max_tempo=70;
 coerr=0;
zz=0;
zz1=0;

/*************** inizializzazione schermo **************/
  clrscrn();
printf("\n**** ACQUISIZIONE E MOVIMENTAZIONE REMOTA ****");
  printf("\n                    *lettura dati dopo sync LeCroy* \n");

  prog_id=2015;                    /* Numero del programma S3 */
  starter=0;                 /* 0=inizia, 1= continua programma S3 */

/****** controllo GPIB***/

 if((nd=ibfind("DEV4"))<0)
 {
     printf("\nInterfaccia GPIB :   Errore di canale !");
    exit(-1);
 }

/****** controllo S3***/

  enbl_robot=0;
  enbl_port=0;
    set_robot_on(enbl_robot,enbl_port,&status);
   cntrl_status(status);
    read_rob_stat(stat_dat,&status);
   cntrl_status(status);
```

```
if(stat_dat[2]==1||stat_dat[3]==0||stat_dat[4]==1||stat_dat[5]==1||stat_dat[6
]!=0)
    {
      printf("\a");
       leggi_stampa_stato();
      exit(-1);
    }

   /*****inserimento  parametri****/

  found:
  ind2=1;
   while(ind2)
  {

       printf("\nIntroduci coord.X di partenza (1000) :  ");
      scanf("%f",&coordx);

       printf("\nIntroduci coord.Y di partenza (0) :  ");
      scanf("%f",&coordy);

       printf("\nIntroduci coord.Z di partenza (1500) :  ");
      scanf("%f",&coordz);

  /********* start programma robot ******************/

        cmd_prog_start(prog_id,starter,&status);
       cntrl_status(status);
      att_supcon();

  /********** sonda sulla posizione di partenza ***********/

        pos[0]=(int)coordx*8;
        pos[1]=(int)coordy*8;
        pos[2]=(int)coordz*8;
        pos[3]=11591;
       pos[4]=0;
        pos[5]=11591;
       pos[6]=0;

        move_dat[8]= pos[0];     /*X*/
        move_dat[9]= pos[1];      /*Y*/
        move_dat[10]= pos[2];     /*Z*/
        move_dat[11]= pos[3];      /*Q1*/
        move_dat[12]= pos[4];      /*Q2*/
```

```
      move_dat[13]= pos[5];       /*Q3*/
      move_dat[14]= pos[6];       /*Q4*/

       movi(move_dat);

     while(1)
    {
   printf("\n POSIZIONE O.K ?  (y/n)  ?:  ");
   scanf("%1s",&ys);
    if(ys==cy||ys==ccy||ys==cn||ys==ccn) break;
    printf("\a");
    }
    if(ys==cy||ys==ccy) ind2=0;
}

     printf("\nIntroduci  raggio del cilindro (mm): ");
   scanf("%lf",&raggio);

    while(1)
     {
   printf("\nSCANSIONE ASSE Z  ([1,400]mm.)  ?:  ");
     res=getint(3,1,27,1,0,&nr,1,1,400);
    if(res == -4) exit(-1);
    if(nr>0 && nr<401) break;
     }

    while(1)
     {
    printf("\nPASSO INCREMENTALE ([1,100]decimi)  ?:  ");
     res=getint(3,1,27,1,0,&p,1,1,100);
    if(res == -4) exit(-1);
    if(p>0 && p<101) break;
     }
/********** calcolo angolo  ***/

      ang1=((double)p/10.0)/raggio;
      printf("\nAngolo: %lf  gradi",ang1*180.0/3.14159265);
    ncc=(2*PI)/ang1;

     nrr=nr*10/p;
     fp=(float)p;

      increz = fp/10.0*8.0;

   if(ncc > 400  ||  nrr > 400)
  {
```

```
      printf("\a");
       printf("\n\nERRORE : Il numero effettivo di righe o colonne e'
maggiore di 400 !!\n");
     goto found;
    }

  cr='c';

      while(1)
       {
      printf("\nSOGLIA VALORE MASSIMO [0.1 , 9.9]  ?:  ");
        res=getfloat(1,1,1,27,1,0,&max,1,0.,10.);
       if(res == 27) exit(-1);
      if(max > 0.  &&  max < 10.) break;
       printf("\a");
       }
      while(1)
       {
      printf("\nSOGLIA VALORE MIMIMO [-1 , < max  ; CR=0.]    ?:  ");
        res=getfloat(2,1,1,27,1,0,&min,1,-1.,max);
       if(res == 27) exit(-1);
      if(min >= -1.  &&  min < max) break;
       printf("\a");
       }
      ind1=1;
       while(ind1)
       {
      printf("\nMEMORIA DI QUADRO (0,1,2,3)   ?:  ");
        res=getint(1,1,27,1,0,&mm,1,0,3);
       if(res == -4) exit(-1);
       if(mm>-1&&mm<4)
      {
       a_displ(mm);
         while(1)
       {
       printf("\nO.K  (y/n)  ?:  ");
      scanf("%1s",&ys);
        if(ys==cy||ys==ccy||ys==cn||ys==ccn) break;
       printf("\a");
       }
       if(ys==cy||ys==ccy) ind1=0;
      }
        }
      val=0;
       a_set(val);
```

```
   printf("\nNumero righe = %d    Numero colonne = %d\n",nrr,ncc);
  xx=100;
  yy=99;
   memset(buf,0,30);
   memset(buf1,0,8);
   memset(pos,0,7);
  ibclr(nd);
  ibwrt(nd,"CHDR OFF",8);            /*elimina testate messaggi*/
  ibwrt(nd,"ACAL OFF",8);            /*inibisce autocalibrazione*/
  ibwrt(nd,"CRMS OFF",8);            /*inibisce visione parametri*/

/********** sonda sulla posizione di partenza ***********/

     pos[0]=(int)coordx*8;
     pos[1]=(int)coordy*8;
     pos[2]=(int)coordz*8;
     pos[3]=11591;
    pos[4]=0;
     pos[5]=11591;
    pos[6]=0;

     move_dat[8]= pos[0];    /*X*/
     move_dat[9]= pos[1];     /*Y*/
     move_dat[10]= pos[2];    /*Z*/
     move_dat[11]= pos[3];     /*Q1*/
     move_dat[12]= pos[4];     /*Q2*/
     move_dat[13]= pos[5];     /*Q3*/
     move_dat[14]= pos[6];     /*Q4*/

     movi(move_dat);

 switch(cr)
      {
        case 'c': case 'C':
          strcpy(stringa,"C1:PAVA? MAX");
        mask=1;
        break;

        case 'e': case 'E':
         strcpy(stringa,"FE:PAVA? MAX");
         mask=1024;
        break;

        case 'f': case 'F':
         strcpy(stringa,"FF:PAVA? MAX");
         mask=2048;
```

```
            break;
          }

/************* ciclo di lettura dati *************/

   for (m=0; m<nrr; m++)                    /*ciclo avanz. asse Z*/
      {
     for (n=0; n<=ncc; n++)                /*ciclo avanz. CILINDRO */
     {
      ang=PI-n*ang1;
        calquaterncili(ang,move_dat);
       movi(move_dat);
       impulso_tras();

         ftime(&timebuffer);
           start=timebuffer.millitm; /*controllo tm*/

   while(1)          /*  attesa per bit di INR=1 (segnale acquisito, etc..) */
    {  ibwrt(nd,"INR?",4);
       ibrd(nd,buf1,8);
        if(atoi(buf1)&mask)  break;

       ftime(&timebuffer);
         stop=timebuffer.millitm; /*controllo tm*/
      if(stop<start)
      stop+=1000;
        tempo=stop-start;
       if(tempo>max_tempo)
       {
          printf("\n %d  %d %d %f ",m,n,(atoi(buf1)&mask),tempo);
       coerr++;
        break;
       }
         start=stop;
   }
        ibwrt(nd,stringa,12);
       ibrd(nd,buf,16);
      wd[n]=atof(buf+4);

   }                              /* For interno */

/************ scrittura riga su PIP ************/

     for (n=0; n<ncc; n++)
    {
            vs[n]=((wd[n]-min)/(max-min))*255;
```

```
            if(vs[n] < 0) vs[n]=0;
             else if(vs[n] > 255) vs[n]=255;
        }
         yy=yy+1;
          a_rowwi(xx,yy,ncc,vs);
         if(kbhit() != 0  &&  getch() == esc)
        {
            nrr=m+1;
            break;
        }

  /******** comando da calcolatore******/

     move_dat[8]= pos[0];     /*X*/
     move_dat[9]= pos[1];      /*Y*/
     move_dat[10]= pos[2] + m*increz;      /*Z*/
     move_dat[11]= pos[3];      /*Q1*/
     move_dat[12]= pos[4];      /*Q2*/
     move_dat[13]= pos[5];      /*Q3*/
     move_dat[14]= pos[6];      /*Q4*/
         movi(move_dat);

        }                              /* For esterno */

%d",(nrr+27),ncc);
          printf("\nNUM.ACQUISIZIONI = %.0f",(double)nrr*(double)ncc);
         printf("\nNUM.ERRORI = %d",coerr);

        /***** disconnessione robot ed oscilloscopio ******/
         scon(nd,status,enbl_robot);
  }

  /**********  controllo dello stato  **************/

     cntrl_status(status)
         int status;
  {
         if (status <= 0)
        {
           printf("\n\nControllo S3 :    Errore di stato  %d",status);
           leggi_stampa_stato();
         exit (-1);
        }
          return(0);
  }
   /******** disconnessione robot ed oscilloscopio *****/
```

```
 scon(nd,status,enbl_robot)
      int nd,status,enbl_robot;
{
    cmd_prog_stop(&status);
    cntrl_status(status);
     set_robot_off(enbl_robot,&status);
    cntrl_status(status);
  ibwrt(nd,"BUZZ BEEP",9);
  ibloc(nd);                          /* rilascia canale */
   return(0);
}

/***********  attesa supercontr.  **************/

 att_supcon()
{
   int status;
do
 {  read_spon(spon_dat,&status);
   if (status != -22) cntrl_status(status); }
   while (spon_dat[0] !=0 || status == -22);
     cmd_prog_stop(&status);
     cntrl_status(status);
    return(0);
}

/**************  leggi stampa stato  *********/

 leggi_stampa_stato()
{
  int status;
    read_rob_stat(stat_dat,&status);

 printf("\n\nProgramma  selezionato : %d",stat_dat[0]);
 printf("\nIstruzione selezionata : %d",stat_dat[1]);
if(stat_dat[2] == 0)
    printf("\nStandby : NO");
else
     printf("\nStandby : SI");
if(stat_dat[3] == 0)
    printf("\nOperation : NO");
else
     printf("\nOperation : SI");
if(stat_dat[4] == 0)
    printf("\nProgram exec : NO");
```

```
 else
      printf("\nProgram exec : SI");
 if(stat_dat[5] == 0)
      printf("\nEmergency robot : NO");
 else
      printf("\nEmergency robot : SI");
 if(stat_dat[6] == 0)
      printf("\nTeach pendant : IN");
 else
     printf("\nTeach pendant : OUT,DISCONNECTED");
 if(stat_dat[7] == 0)
      printf("\nInterrupt : ENABLED");
 else
      printf("\nInterrupt : DISABLED");
 printf("\nX,  Y,  Z      : %d  %d  %d",stat_dat[8],stat_dat[9],stat_dat[10]);
 printf("\nQ1, Q2, Q3, Q4 : %d  %d  %d
%d",stat_dat[11],stat_dat[12],stat_dat[13],stat_dat[14]);
 if(stat_dat[29] == 0)
      printf("\nOrientation : WRIST");
 else
     printf("\nOrientation : TOOL");
  printf("\nTCP corrente  : %d",stat_dat[30]);
   printf(" ");
 printf("\n\nPREMERE UN TASTO PER CONTINUARE");
 getch();
  return(0);
}

  move_array  move_dat;
  stat_array  stat_dat;
  spon_array  spon_dat;

  movi(move_dat)
  move_array  move_dat;

{
 int frame_id;
 int tcp_id;
int i;
      int  enbl_port,enbl_robot,out_no,out_dat,robot_mode,status,a;
     int prog_id,starter,reg_id;

 move_dat[0]=0;
 move_dat[1]=0;
 move_dat[2]=0;
 move_dat[3]=1;
```

```
move_dat[4]=0;
move_dat[5]=0;
 move_dat[6]=8000;
 move_dat[7]=8142;
move_dat[15]=0;
move_dat[16]=0;
move_dat[17]=0;
move_dat[18]=0;
move_dat[19]=0;
move_dat[20]=0;
move_dat[21]=0;
move_dat[22]=0;
move_dat[23]=0;
move_dat[24]=0;
move_dat[25]=0;
move_dat[26]=0;
move_dat[27]=0;
move_dat[28]=0;
move_dat[29]=0;

i=3;
 move_dat[2]=2;
  cmd_move(move_dat,&status);
  cntrl_status(status,i);

i=4;
 move_dat[2]=0;
  cmd_move(move_dat,&status);
  cntrl_status(status,i);

i=5;
 move_dat[2]=3;
  cmd_move(move_dat,&status);
  cntrl_status(status,i);
  return(0);

}

 /***invio impulso al trasmettitore  ***/

  impulso_tras()
{
     int out_no,out_dat,status;
    out_no=1;
     out_dat=1;
        write_output(out_no,&out_dat,&status);
```

```
    out_dat=0;
       write_output(out_no,&out_dat,&status);
 return(0);
}

 float leggi_tensione(mask,stringa)
int mask;
char stringa[13];
{
      unsigned char buf1[8];
     char buf[30];
     int nd,n;
     nd=ibfind("DEV4");

      while(1)  /*  attesa per bit di INR=1 (segnale acquisito, etc..) */
    {
       ibwrt(nd,"INR?",4);
       ibrd(nd,buf1,8);
        if(atoi(buf1)&mask)  break;
    }

      ibwrt(nd,stringa,12);
     ibrd(nd,buf,16);
      return(atof(buf+4));
}

/********** calcolo quaternioni ******/


  calquaterncili(ang,move_dat)


double ang;
 move_array  move_dat;

{
   double w,x,y,z,s,a,b,c,d,e,f,g,t;

     t=(90/2.0)*PI/180.0;
  w= cos (t);
  x= sin (t) * 0;
  y= sin (t) * 1;
  z= sin (t) * 0;

   t=(ang/2.0);
  s= cos (t);
```

```
    a= sin (t) * 1;
    b= sin (t) * 0;
    c= sin (t) * 0;
    d=w*s-(x*a+y*b+z*c);
    e=w*a+x*s+y*c-z*b;
    f=w*b+y*s+z*a-x*c;
    g=w*c+z*s+x*b-y*a;

     d=d*16393;      /*Q1*/
     e=e*16393;      /*Q2*/
     f=f*16393;      /*Q3*/
     g=g*16393;      /*Q4*/
/************** dati in uscita **************/

    move_dat[11]=d;         /*Q1*/
    move_dat[12]=e;         /*Q2*/
    move_dat[13]=f;         /*Q3*/
    move_dat[14]=g;         /*Q4*/

  return(0);
}
```

Programma ONDABERU

```
/********* AQUISIZIONE E MOVIMENTAZIONE REMOTA *********/
  /* esame pezzo comandato dal calcolatore lettura dati dopo sync
LeCroy, con introduzione delle coordinate di partenza; onda completa*/

   #include "\gpib-pc\decl.h"
  #include <stdio.h>
  #include <conio.h>
  #include <math.h>
  #include <stdlib.h>
   #include "c:\robinc\ctsttype.h"
   #include "c:\robinc\ctstfunc.h"
  #include <time.h>
   #include <sys\types.h>
   #include <sys\timeb.h>

   #define PI 3.14159265359
  #define cc      'c'
  #define ccc     'C'
  #define ce      'e'
  #define cce     'E'
  #define cf      'f'
  #define ccf     'F'
  #define cy      'y'
  #define ccy     'Y'
  #define cn      'n'
  #define ccn     'N'
  #define esc      '\x1b'

   float coordx,coordy,coordz,leggi_tensione();

   stat_array stat_dat;
   spon_array spon_dat;
   move_array move_dat;
 /*****************************************************/

  main()
  {
       int cr,i,l,m,n,num,nd,nr,p,nc,reg_dat,res,j,mm,val,b;
      int lr,xx,yy,ncc,nrr,mask,ind1,ind2;
     long sleeper,swap;
     int kbhit(void);

     int zz,zz1,brr,brr1;
     double  ang,ang1,ang2,raggio;
```

```
        move_dat[13]= pos[5];       /*Q3*/
        move_dat[14]= pos[6];       /*Q4*/

         movi(move_dat);

       while(1)
      {
     printf("\n POSIZIONE O.K ?  (y/n)  ?:  ");
     scanf("%1s",&ys);
      if(ys==cy||ys==ccy||ys==cn||ys==ccn) break;
      printf("\a");
      }
      if(ys==cy||ys==ccy) ind2=0;
}

       printf("\nIntroduci  raggio del cilindro (mm): ");
     scanf("%lf",&raggio);

     while(1)
      {
     printf("\nSCANSIONE ASSE Z  ([1,400]mm.)  ?:  ");
       res=getint(3,1,27,1,0,&nr,1,1,400);
     if(res == -4) exit(-1);
     if(nr>0 && nr<401) break;
      }

     while(1)
      {
     printf("\nPASSO INCREMENTALE ([1,100]decimi)  ?:  ");
       res=getint(3,1,27,1,0,&p,1,1,100);
     if(res == -4) exit(-1);
     if(p>0 && p<101) break;
      }
/********** calcolo angolo  ***/

       ang1=((double)p/10.0)/raggio;
       printf("\nAngolo: %lf  gradi",ang1*180.0/3.14159265);
     ncc=(2*PI)/ang1;

     nrr=nr*10/p;
     fp=(float)p;

      increz = fp/10.0*8.0;

    ind1=1;
     while(ind1)
```

```
      {
     printf("\nMEMORIA DI QUADRO (0,1,2,3)   ?:  ");
       res=getint(1,1,27,1,0,&mm,1,0,3);
      if(res == -4) exit(-1);
      if(mm>-1&&mm<4)
     {
      a_displ(mm);
       while(1)
      {
      printf("\nO.K  (y/n)  ?:  ");
     scanf("%1s",&ys);
      if(ys==cy||ys==ccy||ys==cn||ys==ccn) break;
      printf("\a");
      }
      if(ys==cy||ys==ccy) ind1=0;
     }
       }
     val=0;
      a_set(val);
        printf("\nNUM.ACQUISIZIONI = %.0f",(double)nrr*(double)ncc);
%d\n",nrr,ncc);
     xx=100;
     yy=99;
      memset(buf,0,30);
      memset(buf1,0,8);
      memset(pos,0,7);
     ibclr(nd);
     ibwrt(nd,"CHDR OFF",8);            /*elimina testate messaggi*/
     ibwrt(nd,"ACAL OFF",8);            /*inibisce autocalibrazione*/
     ibwrt(nd,"CRMS OFF",8);            /*inibisce visione parametri*/

  /******** sonda sulla posizione di partenza ***********/

        pos[0]=(int)coordx*8;
        pos[1]=(int)coordy*8;
        pos[2]=(int)coordz*8;
        pos[3]=11591;
       pos[4]=0;
        pos[5]=11591;
       pos[6]=0;

        move_dat[8]= pos[0];    /*X*/
        move_dat[9]= pos[1];     /*Y*/
        move_dat[10]= pos[2];    /*Z*/
        move_dat[11]= pos[3];     /*Q1*/
        move_dat[12]= pos[4];     /*Q2*/
```

```
{
     cmd_prog_stop(&status);
     cntrl_status(status);
      set_robot_off(enbl_robot,&status);
     cntrl_status(status);
   ibwrt(nd,"BUZZ BEEP",9);
   ibloc(nd);                          /* rilascia canale */
    return(0);
}

/**************  attesa supercontr.  ****************/

 att_supcon()
{
    int status;
do
  {  read_spon(spon_dat,&status);
    if (status != -22) cntrl_status(status); }
   while (spon_dat[0] !=0 || status == -22);
     cmd_prog_stop(&status);
     cntrl_status(status);
    return(0);
}

/***************  leggi stampa stato  ************/

 leggi_stampa_stato()
{
   int status;
     read_rob_stat(stat_dat,&status);

 printf("\n\nProgramma  selezionato : %d",stat_dat[0]);
 printf("\nIstruzione selezionata : %d",stat_dat[1]);
if(stat_dat[2] == 0)
    printf("\nStandby : NO");
else
     printf("\nStandby : SI");
if(stat_dat[3] == 0)
    printf("\nOperation : NO");
else
     printf("\nOperation : SI");
if(stat_dat[4] == 0)
    printf("\nProgram exec : NO");
else
     printf("\nProgram exec : SI");
if(stat_dat[5] == 0)
```

```
        printf("\nEmergency robot : NO");
    else
        printf("\nEmergency robot : SI");
    if(stat_dat[6] == 0)
        printf("\nTeach pendant : IN");
    else
       printf("\nTeach pendant : OUT,DISCONNECTED");
    if(stat_dat[7] == 0)
        printf("\nInterrupt : ENABLED");
    else
        printf("\nInterrupt : DISABLED");
    printf("\nX,  Y,  Z      : %d  %d  %d",stat_dat[8],stat_dat[9],stat_dat[10]);
    printf("\nQ1, Q2, Q3, Q4 : %d  %d  %d
%d",stat_dat[11],stat_dat[12],stat_dat[13],stat_dat[14]);
    if(stat_dat[29] == 0)
        printf("\nOrientation : WRIST");
    else
        printf("\nOrientation : TOOL");
     printf("\nTCP corrente : %d",stat_dat[30]);
      printf(" ");
    printf("\n\nPREMERE UN TASTO PER CONTINUARE");
     getch();
      return(0);
    }

      move_array move_dat;
      stat_array stat_dat;
      spon_array spon_dat;

      movi(move_dat)
      move_array move_dat;

    {
     int frame_id;
     int tcp_id;
     int i;
          int enbl_port,enbl_robot,out_no,out_dat,robot_mode,status,a;
        int prog_id,starter,reg_id;

      move_dat[0]=0;
      move_dat[1]=0;
      move_dat[2]=0;
      move_dat[3]=1;
      move_dat[4]=0;
      move_dat[5]=0;
       move_dat[6]=8000;
```

```
 move_dat[7]=8142;
 move_dat[15]=0;
 move_dat[16]=0;
 move_dat[17]=0;
 move_dat[18]=0;
 move_dat[19]=0;
 move_dat[20]=0;
 move_dat[21]=0;
 move_dat[22]=0;
 move_dat[23]=0;
 move_dat[24]=0;
 move_dat[25]=0;
 move_dat[26]=0;
 move_dat[27]=0;
 move_dat[28]=0;
 move_dat[29]=0;

i=3;
 move_dat[2]=2;
  cmd_move(move_dat,&status);
  cntrl_status(status,i);

i=4;
 move_dat[2]=0;
  cmd_move(move_dat,&status);
  cntrl_status(status,i);

i=5;
 move_dat[2]=3;
  cmd_move(move_dat,&status);
  cntrl_status(status,i);
 return(0);

}

 /***invio impulso al trasmettitore  ***/

 impulso_tras()
{
     int out_no,out_dat,status;
    out_no=1;
     out_dat=1;
        write_output(out_no,&out_dat,&status);
     out_dat=0;
        write_output(out_no,&out_dat,&status);
 return(0);
```

```
}

 float leggi_tensione(mask,stringa)
int mask;
char stringa[13];
{
      unsigned char buf1[8];
     char buf[30];
     int nd,n;
     nd=ibfind("DEV4");

      while(1)  /*  attesa per bit di INR=1 (segnale acquisito, etc..) */
     {
       ibwrt(nd,"INR?",4);
       ibrd(nd,buf1,8);
        if(atoi(buf1)&mask)  break;
     }

      ibwrt(nd,stringa,12);
     ibrd(nd,buf,16);
      return(atof(buf+4));
}

/********** calcolo quaternioni ******/

 calquaterncili(ang,move_dat)

double ang;
 move_array  move_dat;

{
   double w,x,y,z,s,a,b,c,d,e,f,g,t;

     t=(90/2.0)*PI/180.0;
  w= cos (t);
  x= sin (t) * 0;
  y= sin (t) * 1;
  z= sin (t) * 0;

   t=(ang/2.0);
  s= cos (t);
  a= sin (t) * 1;
  b= sin (t) * 0;
  c= sin (t) * 0;
  d=w*s-(x*a+y*b+z*c);
  e=w*a+x*s+y*c-z*b;
```

```
    f=w*b+y*s+z*a-x*c;
    g=w*c+z*s+x*b-y*a;

     d=d*16393;      /*Q1*/
     e=e*16393;      /*Q2*/
     f=f*16393;      /*Q3*/
     g=g*16393;      /*Q4*/
/************** dati in uscita **************/

    move_dat[11]=d;         /*Q1*/
    move_dat[12]=e;         /*Q2*/
    move_dat[13]=f;         /*Q3*/
    move_dat[14]=g;         /*Q4*/

  return(0);
}
```

Riferimenti

1) E. Bozzi, M. Chimenti: "Pacchetto software per l'ispezione a soglia singola e onda completa mediante stazione US" Collaborazione Scientifica IEI-Alenia, Nota Interna B4-12 1993

2) L. Azzarelli, E. Bozzi, M. Chimenti: "Progetto e sviluppo di una struttura per il CND a ultrasuoni" Collaborazione Scientifica IEI-Alenia, Nota Interna B4-64, 1990

3) ABB ROBOTICS: "Programming Manual Robot Control System S3 - 6397013-121", 1990

4) ASEA ROBOTICS: "ABB Robot Language - ARLA Reference Manual 2.0", 1988

5) ABB ROBOTICS: "Communication Tools - User's Guide CK 09-1425E", 1988

6) LECROY: "Remote Control Manual for models 9420/24/50 Digital Oscilloscopes", 1990

7) E. Bozzi, M. Chimenti: "Progetto Software 'Programmi di scansione e acquisizione US' " Collaborazione Scientifica IEI-Alenia, Nota Interna B4-26, 1992