Interploazione lineare e circolare su 4 assi cartesiani

La libreria di Proteus per i controllori della serie VK e 4000 comprende un pacchetto completo per l'interpolazione lineare e circolare su 4 assi.

Con questa funzione è possibile interpolare su 2, 3 o 4 assi, con una precisione che puó essere impostata fino a 0.5 micron con passo di targa dell' encoder uguale a 1/1000 di mm.

Normalmente il periodo della routine PID di interpolazione è impostato a 4 millisecondi, ed il tempo del PID non supera gli 800 microsecondi. Il PID esegue tutte le operazioni in virgola mobile a doppia precisione.

Durante l'interpolazione è possibile variare al volo la velocità da 0 all 100% (funzione JOG) e correggere al volo gli offset degli assi (funzione VOLANTINO).

Il programma di interpolazione accetta i comandi ISO standard G00, G1 G40, G2 G40, G3 G40 piú una serie di comandi specifici semplificati, gestiti con un buffer circolare di 8000 passi.

caratteristiche dell'interpolazione:

- Pilotaggio di motori DC e Brushless comandati in tensione e ingresso di abilitazione.
- Gestione della camma di zero e della tacca di zero degli encoders
- Gestione dei fine corsa minimo e massimo hardware e software.
- PID con guadagno integrale automatico.
- Guadagno percentuale (KC) impostabile da 0 a 100%
- Precisione impostabile (massima precisione = ½ impulso encoder)
- Trimmer per regolazione runtime della velocità da 0 a 100%
- Volantini di offset delle quote runtime sui 4 assi
- Autocalibrazione automatica dei parametri dei motori.
- Ciclo di controllo a tempo impostabile (da 2 a 10 mSec, per default 4 mSec.)
- Pilotaggio di motori con tempo di risposta (inerzia) da 0 a 100 mSec.
- Pilotaggio di gruppi di motori con tempi di inerzia differenti tra loro.
- Pilotaggio di gruppi di motori con risoluzione encoder differenti tra loro.
- I dati fissi (salvo quelli meccanici come il passo dell' encoder, la posizione dei fine corsa e le velocità di ricerca della camma di zero) sono ridotti al minimo e sono normalizzati, ovvero nella maggior parte dei casi possono essere messi ad 1.
- È raggiungibile una precisione migliore del micron con velocitá fino a 30 m/min.
- La velocitá massima in lineare non è funzione della precisione né dell' inerzia dei motori.
- La velocitá massima in circolare è funzione solo della precisione.
- Le rampe di accelerazione e decelerazione lineari e circolari sono funzione solo dell'inerzia dei motori. Usando l'autoapprendimento la precisione è garantita per qualsiasi inerzia del motore fino a 100 mSec.

Funzioni dell'interpolazione:

Il task di interpolazione è accessibile dal programmatore mediante le seguenti funzioni:

void F_Savefix(char *name)

Salva i dati fissi dell' interpolazione in un file su SD

void F_Loadfix(char *name)

Carica dati fissi dell' interpolazione da un file su SD

void Inter(int T,int Q)

Lancia il task di interpolazione con tempo di ciclo T millisecondi (normalmente T=4) e con parametro di quantizzazione del cerchio pari a Q (normalmente Q=3)

void Exec_Cmd(int CMD,int Arg,float p0, float p1,float p2,float p3,float p4)

Esegue un comando di interpolazione. I comandi supportati sono:

CMD mnemonico

0 S_END se un comando ha una durata, esce settando Cmd a 0 altrimenti esce quando si riceve S_END. Questo comando è richiesto dai comandi 2, 3, 7 e 8.

1 S_AUTO esegue delle figure di prova

```
Arg     SETTA SCALA QUOTE PASSATE 10=MILLESIMI,100=CENTESIMI,1000=DECIMI,10000=mm.)
p0      SETTA SCALA ANGOLI    (10=gradi/1000,100=gradi/100,1000=gradi/10,10000=gradi)
p2      NUMERO PROGRAMMA      (attualmente solo=1)
```

3* S_MANU controlla gli assi in manuale. Non tiene conto dei fine corsa software e disabilita il controllo della quota a loop chiuso.

Arg: 0=fermo
1=indietro veloce
2=indietro lento
3=avanti lento
4=avanti veloce

p0 = asse da muovere (1=X, 2=Y, 3=Z, 4=A)
p1 = SPEED NORM (0…32000)
p2 = SPEEDLENTO (0…32000)

4 S_CALIB autoapprendimento dei parametri dei motori

Arg: 0=considera tutti i motori con la stessa inerzia
1=calcola separatamente le inerzie dei motori
p0 moltiplicatore inerzia (inerzia a 10 volts/inerzia a tensione applicata).

8* F_SEMI porta gli assi a una quota programmata:

Arg: piano di lavoro (0=lineare, 1=XY, 2= XZ, 3=YZ)
p0 velocitá in mm/min.
p1 quota X in mm/100
p2 quota Y in mm/100
p3 quota Z in mm/100
p4 quota A in mm/100

9 F_AUTO esegue un programma in automatico. Il programma deve essere giá almeno parzialmente caricato mediante le funzioni Passo0 e PassoR

Arg SETTA SCALA QUOTE PASSATE 10=MILLESIMI,100=CENTESIMI,1000=DECIMI,10000=mm.)
p0 SETTA SCALA ANGOLI (10=gradi/1000,100=gradi/100,1000=gradi/10,10000=gradi)
p2 NUMERO RIGA INIZIO (normalmente=1)

10 F_ZEROMICRO esegue la funzione di ricerca del micro di zero, e se presente della tacca di zero di un asse.

Arg numero dell' asse (1=X, 2=Y, 3=Z, 4=A)

11 F_SETASSE setta brutalmente la quota di un asse ad un dato valore

Arg numero dell' asse (1=X, 2=Y, 3=Z, 4=A)
p0 quota dell' asse (in mm/100)

12 F_LOOPON abilita il posizionamento a loop chiuso

13 F_LOOPOFF disabilita il posizionamento a loop chiuso

19 F_STOP abortisce un posizionamento in automatico o semiautomatico. Se il posizionamento era a loop chiuso resta a loop chiuso

21 F_ENABLE commando che specifica i motori da abilitare. all'inizio tutti i motori sono disabilitati.

Arg: non usato
p0 1= asse X abilitato
p1 1= asse Y abilitato
p2 1= asse Z abilitato
p3 1= asse A abilitato

23 F_RESETERROR resetta l'errore in corso.

void Passo0(void)

Specifica l'inizio di un nuovo programma, e prepara i puntatori per la ricezione del programma.

void PassoR(int,float,float,float,float)

Accoda un passo di programma al buffer. Gli operandi possono essere:

MNEMONICI FUNZIONI DI INTERPOLAZIONE

Op	COMANDO	CODICE	Para1	Para2	Para3	Para4	NOTE
-	START	START					
0	STOP	STOP					
1	PIANO XY	PIA	1				
1	PIANO XZ	PIA	2				
1	PIANO YZ	PIA	3				
2	POSIZ. ASS.	PA	XF	YF	ZF	AF	
3	POSIZ. REL.	PR	DX	DY	DZ	DA	
4	POS. TANG.	PT	DIST	DP			**

5	CERCHIO C.R.	CCR	ANG	IC	JC	PF	**
6	CERCHIO TANG.	CT	ANG	RAG	PF		**
7	TUFFO ASS.	ZA	VEL	PF			**
8	TUFFO REL.	ZR	VEL	DP			**
9	ANGOLO ASS.	AA	ANG				**
10	ANGOLO REL.	AR	ANG				**
12	VELOCE ASS.	FA	XF	YF	ZF	AF	
13	VELOCE REL.	FR	DX	DY	DZ	DA	
14	VELOCE TANG.	FT	DIST	DP			**
15	CERCHIO TAN.R	CTR	ANG	RAG	DP		**
20	V.RACCORDO	RR	VEL				
21	VEL. INT.	VP	VEL				
22	WAIT****	WAIT	TmSec	0			
22	USER****	WAIT	TmSec	flag			
23	LOOP	LOOP	N.ITE	N.IST			
25	VEL.MAX.	VF	VEL				
26	RAGGIO RACC.	RAC	RAG				
27	ERR. MAX.	EP	ERR				
28	ERR. Collis.	EC	ERR				
104	WAIT IN	WAITIN	LEV	MASK			
105	SET OUT	SETOUT	LEV	MASK			
106	PUSH ***	STORE	n				
107	POP ***	GO	n	speed			

dove P e' l' asse profondita' (Z se nel piano XY) ,A e' il 4^ asse
IC e' la prima coordinata del centro (X se nel piano XY ecc.) e
JC e' la seconda coordinata del centro.

** I seguenti comandi hanno senso solo dopo aver scelto un piano
di lavoro (XY, XZ o YZ).

*** PUSH (STORE) salva la quota attuale in un registro n (n=1...32)
POP (GO) va alla quota salvata nel registro n (n=1...32)
se speed=0 va in veloce (G00), se speed=1 va a
velocitá di interpolazione (G01)

**** WAIT attende un tempo t in millisecondi
**** USER stoppa il programma con STFLAG=FLAG. Se esiste USER_SUB la
esegue chiamando USER_SUB(FLAG) FLAG deve essere un intero
> 1 altrimenti mette Stp=-2 e si deve porre Stp=2 per
ripartire

void
PassoISO(int,float,float,float,float,float,int)

Accoda un passo di programma ISO al buffer. Gli operandi possono essere:

MNEMONICI FUNZIONI DI INTERPOLAZIONE								
Op	COMANDO	CODICE	Para1	Para2	Para3	Para4	Para5	Para6
100	G00 G40	G00	XF	YF	ZF	AF	--	mask
101	G01 G40	G01	XF	YF	ZF	AF	--	mask
102	G02 G40	G02	ANG	IC	JC	PF	AF	mask
103	G03 G40	G03	ANG	IC	JC	PF	AF	mask

Il parametro mask indica gli assi effettivamente utilizzati nell' istruzione. Gli assi corrispondenti ad uno zero in mask si considerano fermi:

Per G00 e G01:

1= asse X
2= asse Y
4= asse Z
8= asse A

Per G02 e G03:

piano XY:	piano XZ:	piano YZ
4= asse Z	2= asse Y	1= asse X
8= asse A	8= asse A	8= asse A

Esempi di uso delle funzioni dell'interpolazione:

```
F_Loadfix("C:\\PARA_INT.BIN");
Inter(4,3
Exec_Cmd(F_ENABLE,0,1,1,1,1,0);

Exec_Cmd(S_CALIB,MB250,0,0,0,0,0);
Exec_Cmd(F_ZEROMICRO,1,0,0,0,0,0);
Exec_Cmd(S_MANU,2,1,8000,1000,0,0);
Exec_Cmd(S_END,0,0,0,0,0,0);
Exec_Cmd(F_SEMI,1,speed,qx,qy,qz,qa);
Exec_Cmd(F_STOP,1,0,0,0,0,0);
Exec_Cmd(S_AUTO,mm/10,gradi,1,0,0,0);
Exec_Cmd(F_LOOPON,0,0,0,0,0,0);
Exec_Cmd(F_LOOPOFF,0,0,0,0,0,0);

USER_SUB=controlla;  //definisce user subroutine
START;               //INIZIO
PIA(1);              //PIANO X-Y
FR(40,40,0,25);      //SPOST. VELOCE X=X0+80 Y=Y0+80
USER(10);            //funzione USER_SUB con parametro 10
FR(40,40,0,25);      //SPOST. VELOCE X=X0+80 Y=Y0+80
PUSH(1);             //salva posizione
FR(80,80,0,0);       //SPOST. VELOCE X=X0+160 Y=Y0+160
AA(0);               //ANGOLO ASSOLUTO 0 GRADI
CTR(360,75,10);      //CERCHIO TANGENTE 360 GRADI R=75
AR(30);              //ANGOLO RELATIVO +30 GRADI
LOOP(12,2);          //RIPERI 12 VOLTE LE 2 PRECEDENTI ISTRUZIONI
SETOUT(1<<15,1<<15); //accende o16
WAITIN(1<<15,1<<15); //attende i16 on
SETOUT(0,1<<15);     //spegne o16
FR(320,150,0,0);     //SPOST. VELOCE X=X0+480 Y=Y0+310
AA(285);             //ANGOLO ASSOLUTO =285 GRADI
PT(300,-10);         //POSIZ. TANGENTE D=300
AR(210);             //ANGOLO RELATIVO +210 GRADI
LOOP(12,2);          //RIPETI 12 VOLTE LE 2 PRECEDENTI ISTRUZIONI
POP(1,0);            //ritorna in posizione salvata in veloce
WAIT(2000);          //attende 2 sec.
FR(-80,-80,0,-50);   //ritorna al punto di start
STOP;                //FINE PROGRAMMA
Exec_Cmd(F_AUTO,mm/100,gradi,1,0,0,0);
```

Le variabili del posizionamento:

```
// ------------------------------------------
// variabili globali usate dal posizionamento
// ------------------------------------------
//  redirect i/o nulli
U8  Low_Level=0,Hi_Level=1;
U8  Dummy_1,Dummy_2,Dummy_3,Dummy_4,Dummy_5,No_Out;
U16 No_Anal=0;
U16 No_Pot=32767;
U32 Not_Used=0;

// RITOCCO ORIGINI
volatile floatf ORG1=0,ORG2=0;ORG3=0,ORG4=0;

// VARIABILI DEI COMANDI
volatile U16 Cmd;         //COMANDO
volatile U16 Arg;         //ARGOMENTO
volatile S32 Para[5];     //PARAMETRI
void(*USER_SUB)(int arg)=0; //user subroutine callback

// VARIABILI STATISTICHE
volatile int TInt;
volatile int PInt;
volatile char CAUSA[]="NOP                                                                ";

// ALTRE VARIABILI PUBBLICHE
volatile floatf Q1,Q2,Q3,Q4;     //LATCH QUOTE REALI ASSI (mm/100)
volatile floatf HF,HG;           //COMPONENTI DELL' ERRORE ASSIALE/RADIALE
volatile char Flagint;           //FLAG STATO DI INTERPOLAZIONE
volatile char Motors;            //FLAG MODO  DI INTERPOLAZIONE
volatile char V_Oper;            //OPERAZ. IN CORSO DI ESECUZ.
volatile char Istat;             //STATO DELL' INTERPOLAZIONE (1=ATTIVA)
volatile short int  Stp;         //FLAG Passo-Passo
volatile short int  Passo;       //LINEA ATTUALE IN ESECUZIONE
volatile short int  Passoend;    //ULTIMA LINEA INVIATA
volatile short int  Pass;        //Passo IN COSTRUZIONE (NON modulo 8192 ma progressivo)
volatile char  PIANO;            //PIANO DI INTERPOLAZIONE
volatile short int ERRORE ;      //CODICE DI ERRORE
volatile short int TOT_PASSI;    //N. PASSI TOTALI DI PROGR.
volatile short int STFLAG;       //FLAG PAUSA
volatile float V_TRIM ;          //VEL. % TRIMMER 0...1
volatile short int RUN_PASSO,RUN_COMANDO; //Passo E COMANDO ATTUALI
volatile short int RUN_NLINE;             //LINEA ATTUALE (NON modulo 8192 ma progressivo)
volatile short int RUN_C,RUN_P,RUN_N;     //Passo E COMANDO IN ESECUZIONE
volatile short int RUN_C2,RUN_P2,RUN_N2;  //Passo E COMANDO IN PREPARAZIONE
volatile short int ASSIOK;                //SE 15 ASSI CALIBRATI
volatile short int F_HOLD ;      //1=HOLD 0=RUN
volatile short int F_TRIM;       //1=TRIM ATTIVO 0=TRIM OFF (100%)
volatile short int F_HALT ;      //SOFT STOP SU TRAIETTORIA
volatile char KP1,KP2,KP3,KP4 ;  //ASSI ABILITABILI
volatile char HP1,HP2,HP3,HP4;   //ASSI ABILITATI
volatile char CP1,CP2,CP3,CP4;   //ASSI IN COPPIA

// PARAMETRI FISICI DELLA MACCHINA
 float integr=0.3;       //coeff. integrale (0.3)
 float autoamp=2000;     //vel. autocalibrazione (8000.)
```

```
 int *asse1=&Not_Used;   //asse 1 (&encoder1 oppure &Not_Used)
 int *asse2=&Not_Used;   //asse 2 (&encoder2 oppure &Not_Used)
 int *asse3=&Not_Used;   //asse 3 (&encoder3 oppure &Not_Used)
 int *asse4=&Not_Used;   //asse 4 (&encoder4 oppure &Not_Used)
 short *rampa1=&No_Anal; //uscita anal. 1 (&anal1... oppure &No_Anal)
 short *rampa2=&No_Anal;//uscita anal. 2 (&anal1... oppure &No_Anal)
 short *rampa3=&No_Anal;  //uscita anal. 3 (&anal1... oppure &No_Anal)
 short *rampa4=&No_Anal;  //uscita anal. 4 (&anal1... oppure &No_Anal)
 char *tz1=&Low_Level;    //tacca zero 1 (&i1... oppure &Low_Level,&Hi_Level)
 char *tz2=&Low_Level;    //tacca zero 2 (&i1... oppure &Low_Level,&Hi_Level)
 char *tz3=&Low_Level;    //tacca zero 3 (&i1... oppure &Low_Level,&Hi_Level)
 char *tz4=&Low_Level;    //tacca zero 4 (&i1... oppure &Low_Level,&Hi_Level)
 char *mz1=&Low_Level;    //micro zero 1 (&i1... oppure &Low_Level,&Hi_Level)
 char *mz2=&Low_Level;    //micro zero 2 (&i1... oppure &Low_Level,&Hi_Level)
 char *mz3=&Low_Level;    //micro zero 3 (&i1... oppure &Low_Level,&Hi_Level)
 char *mz4=&Low_Level;    //micro zero 4 (&i1... oppure &Low_Level,&Hi_Level)
 char *fcmin1=&Low_Level; //fine corsa min. 1 (&i1... oppure &Low_Level,&Hi_Level)
 char *fcmin2=&Low_Level; //fine corsa min. 2 (&i1... oppure &Low_Level,&Hi_Level)
 char *fcmin3=&Low_Level; //fine corsa min. 3 (&i1... oppure &Low_Level,&Hi_Level)
 char *fcmin4=&Low_Level; //fine corsa min. 4 (&i1... oppure &Low_Level,&Hi_Level)
 char *fcmax1=&Low_Level; //fine corsa max. 1 (&i1... oppure &Low_Level,&Hi_Level)
 char *fcmax2=&Low_Level; //fine corsa max. 2 (&i1... oppure &Low_Level,&Hi_Level)
 char *fcmax3=&Low_Level; //fine corsa max. 3 (&i1... oppure &Low_Level,&Hi_Level)
 char *fcmax4=&Low_Level; //fine corsa max. 4 (&i1... oppure &Low_Level,&Hi_Level)
 char *emerg1=&Hi_Level;  //emergenza 1 (n.c.) (&i1... oppure &Low_Level,&Hi_Level)
 char *emerg2=&Hi_Level;  //emergenza 2 (n.c.) (&i1... oppure &Low_Level,&Hi_Level)
 char *arcond=&Low_Level; //arresto condizionato (&i1... oppure &Low_Level,&Hi_Level)
 char *abi1=&No_Out;      //abilitaz. mot. 1 (&o1,... oppure &No_Out)
 char *abi2=&No_Out;      //abilitaz. mot. 2 (&o1,... oppure &No_Out)
 char *abi3=&No_Out;      //abilitaz. mot. 3 (&o1,... oppure &No_Out)
 char *abi4=&No_Out;      //abilitaz. mot. 4 (&o1,... oppure &No_Out)
 char *test=&No_Out;      //uscita di test   (&o1,... oppure &No_Out)
 char *mandr=&No_Out;     //uscita mandrino  (&o1,... oppure &No_Out)
 char *ready=&No_Out;     //uscita ready     (&o1,... oppure &No_Out)
 short *pote1=&No_Pot;    //potenziometro 1  (trimm. velocita') {&pot1... oppure &No_Pot)
 short *pote2=&No_Pot;    //potenziometro 1  (tastatore)        {&pot1... oppure &No_Pot)

//DATI FISSI
//VAR. COMUNI AI 3 ASSI
 float Gl,Gq;             //COEFF. CORR. LINEARE E DERIVATO
 float Velint;            //VEL. DI INTERPOLAZ. IMPOSTATA M/MIN
 float Ramp;              //RAMPA ACC. CIRCOLARE  (0.4...5)
 float Rali;              //RAMPA ACC. LINEARE    (0.4...5)
 float Damp;              //RAMPA DEC. CIRCOLARE  (0.4...3)
 float Dali;              //RAMPA DEC. LINEARE    (0.4...3)
 float Vmin;              //VELOC. MINIMA DI INVERSIONE MOTO (0.2...0.5)
 float E;                 //ERRORE MAX. IMPOSTATO  (4...10)
 float Tau;               //TEMPO DI COMPENSAZIONE ACCELERAZIONI (20 mSEC.)
 float Molla;             //RIDUZ. GUADAGNO CORREZIONE RADIALE/ASSIALE (TIP. 1)
 float Cent;              //CORREZ. CENTRIFUGA (1...3)
 float Segm;              //N. DI MICROCAMP. PER UN LATO DEL POLIGONO (3..4)
 float Collis;            //ERRORE DI COLLISIONE
 float Collit;            //ERRORE DI TEMPO DI POSIZ.
 float Collix;            //ERRORE TEMPO-SPAZIO
 float Velpp;             //VEL. MAX. IN VELOCE
 float Smorz;             //SMORZAMENTO FINALE CERCHIO (TIP. 0.5)
 float Inema;             //INERZIA MASSIMA
//DATI FISSI
//VAR. SEPARATE PER I 4 ASSI
```

```
 float R[4];              //VEL. MAX. MOTORI
 float Corr[4];           //OFFSET MOTORI
 float Ine[4];            //INERZIE MOTORI
 float Passoenc[4];       //Passo DELL' ENCODER
//DATI FISSI
//FINE CORSA SOFTWARE
 long int Fcmin[4];       //FINE CORSA MINIMI
 long int Fcmax[4];       //FINE CORSA MASSIMI
//DATI FISSI
//TACCHE DI ZERO
 long int Zero[4];        //QUOTE AZZ. ASSI
 long int Vzero[4];       //VEL.  AZZ. ASSI
 long int Vtacc[4];       //VELOC.RICERCA TACCA ZERO ENCODERS
//DATI FISSI: FINE

//VETTORI PASSI DI PROGRAMMA:
//Op = OPERAZIONE        15     8 7      0
//                       ........ ........
//                         assi    codice
//bit 8 = asse 1 inattivo
//bit 9 = asse 2 inattivo ecc.
//
//Para1 ...Para5   parametri (quote,angoli,tempi a seconda di Op)
//Nline            numero fisico della linea in esecuzione
//Nflag            bit 0...31 flags di esecuzione immediata
//Para5            usato solo in semiautomatico

volatile  short int Op[VT_SIZE];     //OPERAZIONE
volatile  long int  Para1[VT_SIZE];  //PRIMO Arg.
volatile  long int  Para2[VT_SIZE];  //SECONDO Arg.
volatile  long int  Para3[VT_SIZE];  //TERZO Arg.
volatile  long int  Para4[VT_SIZE];  //QUARTO Arg.
volatile  long int  Para5[VT_SIZE];  //QUINTO Arg.
volatile  long int  Nflag[VT_SIZE];  //FLAGS DI LINEA
volatile  short int Nline[VT_SIZE];  //N. DI LINEA
```

inizializzazione:

```
#define QUOTA1 battram.quota[0]
#define QUOTA2 battram.quota[1]
#define QUOTA3 battram.quota[2]
#define QUOTA4 battram.quota[3]
//
// parametri geometrici dell' interpolazione:
//
integr=0.1;        //coeff. integrale (default 0.1)
autoamp=2000.;     //vel. autocalibrazione (default 2000.)
asse1=&QUOTA1;     //asse 1 (&encoder1... oppure default &Not_Used)
asse2=&QUOTA2;     //asse 2 (&encoder2... oppure default &Not_Used)
asse3=&QUOTA3;     //asse 3 (&encoder3... oppure default &Not_Used)
asse4=&QUOTA4;     //asse 4 (&encoder4... oppure default &Not_Used)
rampa1=&anal1;     //uscita anal. 1 (&anal1... oppure default &No_Anal)
rampa2=&anal2;     //uscita anal. 2 (&anal1... oppure default &No_Anal)
rampa3=&anal3;     //uscita anal. 3 (&anal1... oppure default &No_Anal)
rampa4=&anal4;     //uscita anal. 4 (&anal1... oppure default &No_Anal)
tz1=0;             //tacca zero 1 (&i1... oppure 1..4 (tacche zero hw) oppure default &Low_Level)
tz2=0;             //tacca zero 2 (&i1... oppure 1..4 (tacche zero hw) oppure default &Low_Level)
tz3=0;             //tacca zero 3 (&i1... oppure 1..4 (tacche zero hw) oppure default &Low_Level)
tz4=0;             //tacca zero 4 (&i1... oppure 1..4 (tacche zero hw) oppure default &Low_Level)
mz1=&i1;           //micro zero 1 (&i1... oppure default &Low_Level)
mz2=&i2;           //micro zero 2 (&i1... oppure default &Low_Level)
mz3=&i3;           //micro zero 3 (&i1... oppure default &Low_Level)
mz4=&i4;           //micro zero 4 (&i1... oppure default &Low_Level)
fcmin1=&i6;        //fine corsa min. 1 (&i1... oppure default &Low_Level)
fcmin2=&i8;        //fine corsa min. 2 (&i1... oppure default &Low_Level)
fcmin3=&i10;       //fine corsa min. 3 (&i1... oppure default &Low_Level)
fcmin4=&i12;       //fine corsa min. 4 (&i1... oppure default &Low_Level)
fcmax1=&i7;        //fine corsa max. 1 (&i1... oppure default &Low_Level)
fcmax2=&i9;        //fine corsa max. 2 (&i1... oppure default &Low_Level)
fcmax3=&i11;       //fine corsa max. 3 (&i1... oppure default &Low_Level)
fcmax4=&i13;       //fine corsa max. 4 (&i1... oppure default &Low_Level)
emerg1=&Hi_Level;  //emergenza 1 (n.c.) (&i1... oppure default &Hi_Level)
emerg2=&Hi_Level;  //emergenza 2 (n.c.) (&i1... oppure default &Hi_Level)
arcond=&Low_Level; //arresto condizionato (&i1... oppure default &Low_Level)
abi1=&o1;          //abilitaz. mot. 1 (&o1,... oppure default &No_Out)
abi2=&o2;          //abilitaz. mot. 2 (&o1,... oppure default &No_Out)
abi3=&o3;          //abilitaz. mot. 3 (&o1,... oppure default &No_Out)
abi4=&o4;          //abilitaz. mot. 4 (&o1,... oppure default &No_Out)
test=&o5;          //uscita di test   (&o1,... oppure default &No_Out)
mandr=&o6;         //uscita mandrino  (&o1,... oppure default &No_Out)
ready=&o7;         //uscita ready     (&o1,... oppure default &No_Out)
pote1=&pot1;       //trimmer velocita' {&pot1... oppure default &No_Pot)
pote2=&No_Pot;     //potenziometro tastatore  {&pot1... oppure default &No_Pot)
//
F_Loadfix("C:\\PARA_INT.BIN");  //carica dati fissi
Inter(4,3);                     //abilita interpolazione a 4 mSec. con camp. circolare=3 (12 mSec)
Exec_Cmd(F_ENABLE,0,1,1,1,1,0); //abilita motori 1,2,3 e 4
```

Alias per i comandi

```
#define START          Passo0()
#define STOP           PassoR(0,0,0,0,0)
#define PIA(a)         PassoR(1,a,0,0,0)
#define PA(x,y,z,a)    PassoR(2,x,y,z,a)
#define PR(x,y,z,a)    PassoR(3,x,y,z,a)
#define PT(d,h)        PassoR(4,d,h,0,0)
#define CCR(a,i,j,p)   PassoR(5,a,i,j,p)
#define CT(a,r,p)      PassoR(6,a,r,p,0)
#define ZA(v,p)        PassoR(7,v,p,0,0)
#define ZR(v,p)        PassoR(8,v,p,0,0)
#define AA(a)          PassoR(9,a,0,0,0)
#define AR(a)          PassoR(10,a,0,0,0)
#define FA(x,y,z,a)    PassoR(12,x,y,z,a)
#define FR(x,y,z,a)    PassoR(13,x,y,z,a)
#define FT(d,p)        PassoR(14,d,p,0,0)
#define CTR(a,r,p)     PassoR(15,a,r,p,0)
#define RR(v)          PassoR(20,v,0,0,0)
#define VP(v)          PassoR(21,v,0,0,0)
#define WAIT(t)        PassoR(22,t,0,0,0)
#define USER(x)        PassoR(22,1,x,0,0)
#define LOOP(n,i)      PassoR(23,n,i,0,0)
#define MICRO(n,i)     PassoR(24,n,i,0,0)
#define RAC(r)         PassoR(26,r,0,0,0)
#define VF(v)          PassoR(25,v,0,0,0)
#define EP(e)          PassoR(27,e,0,0,0)
#define EC(e)          PassoR(28,e,0,0,0)
#define DU(d)          PassoR(30,d,0,0,0)
#define WAITIN(l,m)    PassoR(104,l,m,0,0)
#define SETOUT(l,m)    PassoR(105,l,m,0,0)
#define PUSH(n)        PassoR(106,n,0,0,0)
#define POP(n,speed)   PassoR(107,n,speed,0,0)
#define STORE(n)       PassoR(106,n,0,0,0)
#define GO(n,speed)    PassoR(107,n,speed,0,0)

#define G00(x,y,z,a,m)     PassoISO(100,x,y,z,a,0,m)
#define G01(x,y,z,a,m)     PassoISO(101,x,y,z,a,0,m)
#define G02(a,i,j,p,q,m)   PassoISO(102,a,i,j,p,q,m)
#define G03(a,i,j,p,q,m)   PassoISO(103,a,i,j,p,q,m)

#define S_END 0
#define S_AUTO 1
#define S_SEMI 2
#define S_MANU 3
#define S_CALIB 4
#define F_END 0
#define F_MANU 7
#define F_SEMI 8
#define F_AUTO 9
#define F_ZEROMICRO 10
#define F_SETASSE 11
#define F_LOOPON 12
#define F_LOOPOFF 13
```

```
#define F_STOP 19
#define F_ENABLE 21
#define F_RESETERROR 23
#define F_EXIT 99

#define mm    10000
#define gradi 10000
```