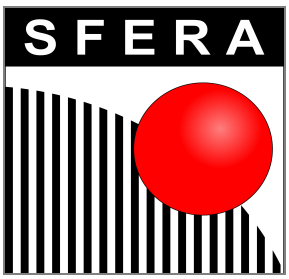


SFERA srl
Controllore assi MC6
Guida utente

versione 1.0.0

Indice

Indice delle tabelle

Indice delle figure

1 Introduzione

1.1 Caratteristiche principali

- Controllore assi standalone per montaggio su quadro elettrico.
- Gestisce fino a sei assi servo con periodo di controllo di 122 us e generazione di traiettorie con periodo di 976 us.
- Sei ingressi encoder con filtri digitali programmabili, contatori a 32 bit da 10 Mhz.
- Sei uscite analogiche per comando azionamenti servo +/-10V, 14 bit di risoluzione
- Sedici uscite e sedici ingressi optoisolati di cui uno specializzato per la cattura veloce della posizione (200 ns max.).
- Gestione di sei ulteriori assi con azionamenti passo-passo
- Posizionamenti indipendenti assoluti e relativi.
- Camma elettronica.
- Albero elettrico (gearing).
- Frizione elettronica.
- Movimenti coordinati lineari e circolari.
- Esecuzione di programmi residenti a bordo

1.2 Come usare la guida

Questa guida è stata scritta con l'obiettivo di rendere semplice e veloce l'utilizzo della scheda controllo assi MC6. Considerando che normalmente le persone interessate sono tecnici che hanno già qualche esperienza automazione industriale, si è cercato di approfondire gli aspetti specifici della scheda piuttosto che utilizzare un approccio didattico sul controllo assi, per il quale si rimanda alla letteratura esistente.
Per gli aspetti hardware, come ad esempio la piedinatura dei connettori, si rimanda al *Manuale di riferimento hardware.*

2 Guida all'uso

2.1 Collegamenti elettrici

Il controllore MC6 presenta sul pannello superiore 6 connettori a vaschetta in cui sono disponibili i segnali di interfaccia agli azionamenti servo e stepper:

- ingresso encoder
- riferimento analogico
- passo e direzione stepper

Lateralmente sono disposti dei connettori a morsetto per le altre funzioni:

- alimentazione a 24V
- inuput digitali
- output digitali
- input analogici
- CAN bus
- seriale RS232
- seriale RS485

2.1.1 Encoder

è possibile utilizzare encoder incrementali in quadratura alimentati a 5V con uscite complementari, oppure encoder “virtuali” equivalenti ricostruiti dall’azionamento.
Se possibile adottate encoder con uscite “line driver RS422”, che garantiscono la migliore qualità dei segnali anche a frequenze elevate e in presenza di disturbi; in ogni caso l'encoder deve essere in grado di sostenere il carico della resistenza di terminazione (150 Ohm) presente nella scheda.
Per il collegamento utilizzate un cavo schermato, meglio se di tipo a coppie per trasmissione dati.

Usate conduttori interni anche per collegare le alimentazioni (GND e +5V) e saldate lo schermo direttamente sul corpo del connettore. Utilizzate gusci metallici. Se il cavo è del tipo a coppie usatele con segnali complementari, ad esempio A1+ e A1- su una coppia, B1+ e B1- su un’altra e così via.
Ovviamente l’alimentazione +5V va collegata solo nel caso di encoder fisici, se l’encoder viene ricostruito dall’azionamento non collegatela. Assicuratevi che l’assorbimento totale dei sei encoder non superi 900 mA, altrimenti è necessario utilizzare un alimentatore esterno; anche in questo caso non collegate insieme le due alimentazioni.
Se avete la necessità di invertire la direzione dell’encoder per adeguarlo alla direzione convenzionale dell’asse, non agite sul collegamento dei segnali perchè potete comodamente farlo da software.
Gli encoder degli assi 1-6 vanno collegati rispettivamente ai connettori P1-P6 della MC6.

2.1.2 Azionamenti

Gli azionamenti devono avere un ingresso di riferimento analogico +/-10V, preferibilmente di tipo differenziale. L'ingresso di abilitazione va collegato ad un

output digitale della MC6. Se l'interfaccia elettrica non fosse compatibile utilizzate un relè esterno.
A volte gli azionamenti permettono la scelta tra funzionamento in modo coppia (la coppia fornita è proporzionale all'ingresso di riferimento) e modo velocità (la velocità del motore è proporzionale all'ingresso di riferimento); si consiglia di scelgliere il modo coppia solo se l'applicazione richiede di regolare o limitare la forza (ad esempio operazioni di piantaggio, inserimento o presa), altrimenti preferite il modo velocità.
Per il riferimento usate un cavetto schermato con lo schermo collegato alla massa dell'azionamento. Se dovete invertire il comando non agite sul cablaggio, soprattutto se l'azionamento non ha un ingresso differenziale perchè provochereste un corto circuito, ma utilizzate l'apposito comando software.

2.1.3 Sensori di zero, extracorsa ed altri input/output

Sono previsti sedici ingressi e sedici uscite a 24V. I sensori di zero e gli extracorsa possono essere collegati a qualunque ingresso in quanto è possibile mappare da software queste funzioni. I rimanenti quattro ingressi e le uscite sono disponibili per l'applicazione. Ad esempio possono comandare un attuatore in relazione molto stretta al movimento degli assi o gestire delle semplici sincronizzazioni con dispositivi esterni.
Se avete la necessità di acquisire delle quote in base ad un segnale esterno collegatelo all'ingresso numero 1 che permette dei tempi di reazione estremamente rapidi.
Gli ingressi sono optoisolati e attivati applicando una tensione compresa tra 15V e 30V rispetto a GNDH (P5.30). Le uscite sono optoisolate di tipo source (PNP), protette al corto circuito e al surriscaldamento e adatte a pilotare carichi fino a 700 mA (ad esempio relè, elettrovalvole e lampade). L'alimentazione a 24V dc nominali (min. 15V max 30V) deve essere fornita esternamente al punto VCCH (P5.21) e deve essere riferita a GNDH.
I fotoaccoppiatori separano l'alimentazione dell'MC6 dal 24V di macchina solo ai fini di aumentare l'immunità ai disturbi, ma per il corretto funzionamento della scheda e per motivi di sicurezza entrambe le alimentazioni devono essere riferite a terra.

2.2 Configurazione e taratura

2.2.1 Impostazione DIP switch

DIP switch S1

Il dip switch ha 8 vie, di cui solo l'ultima è utilizzata dal sistema (se attivo all'accensione pone il controllore in modo programmazione); gli altri 7 sono disponibili per l'applicazione.

2.2.2 Comunicazione

Normalmente il controllore MC6 viene collegato a dei pannelli operatore sui quali è implementata una interfaccia grafica che permette di effettuare la configurazione e

taratura degli assi. In caso contrario è possibile collegare sulla seriale RS232 un emulatore di terminale attraverso il quale si possono dare dei comandi tramite una shell testuale.

Da questo punto del manuale in avanti verranno introdotti diversi comandi che, usati interattivamente, serviranno a verificare e tarare il sistema. In seguito gli stessi comandi potranno essere usati dal vostro software per realizzare l'applicazione.
La sintassi dei comandi è:

- <*Comando*> [*Parametro1*] ... [*ParametroN*] [; *Commento*]

Il trattino iniziale è il prompt del programma e non deve essere digitato. Il commento è facoltativo e non ha alcun effetto sul funzionamento. La shell vi consente di usare indifferentemente lettere maiuscole e minuscole, ma ricordate che nel software applicativo non sarà possibile. La risposta al comando, se prevista, viene visualizzata sulle righe seguenti.

2.2.3 Interfaccia Encoder

Spegnete il controllore e collegate gli encoder. Riaccendete e date i comandi:

- GetActPos 1 ; lettura posizione reale asse 1
  -2

GetActPos restituisce la posizione reale dell'asse, cioè quella letta dal contatore encoder, che all'accensione vale 0 o un valore piccolo se l'asse si e mosso leggermente per qualche vibrazione. Muovete l'asse a mano in direzione positiva (secondo la vostra applicazione) in modo che l'encoder faccia un giro e rileggete la posizione:

- GetActPos 1
  4085

La nuova posizione dovrebbe corrispondere all'incirca al numero degli impulsi dell'encoder moltiplicato per quattro, ad esempio un encoder da 1024 impulsi/giro deve dare circa 4096 conteggi.
Potrebbe succedere che il segno sia negativo, ad esempio -4107; in questo caso occorre configurare l'asse con il comando *ConfigServo* in modo da invertire la direzione del contatore encoder. Oltre alla direzione il comando ConfigServo imposta anche la banda del filtro digitale, utile per evitare problemi dovuti al rumore generato dagli azionamenti. Ad esempio, se un azionamento ha velocità massima 4500 rpm, l'encoder ha 1024 impulsi/giro e a causa della trasmissione fa un 5 giri per ogni giro motore, avremo:

$$Fmax = \frac{4500}{60} * 5 * (1024 * 4) = 1536000\ conteggi/secondo\ (1.536\ MHz)$$

Impostando il parametro EncDir del comando ConfigServo a -1 invertiamo la direzione del contatore e con il parametro EncFilter a 4 filtriamo frequenze oltre i 2 Mhz (vedi la guida di riferimento del comando ConfigServo per maggiori dettagli).

- ConfigServo 1 1 -1 4; imposta inversione encoder 1, Fmax 2 Mhz

A questo punto è opportuno verificare la correttezza della direzione ripetendo la

procedura. Infine prendete nota del codice scelto (dovrete usare *ConfigServo* con lo stesso codice nel programma applicativo) e ripetete il procedimento per tutti gli altri encoder.

2.2.4 Interfaccia azionamento

Scollegate il motore dalla meccanica, ad esempio smontando le cinghie di trasmissione, in modo che possa girare senza rischi per cose o persone. Assicuratevi inoltre di poter togliere rapidamente potenza, ad esempio con un pulsante di emergenza.
Ora impostate il PID a zero e la soglia dell'errore di posizione a 10000, quindi mettete in coppia l'asse:

- Pid 1 0 0 0
- ErrorLimit 1 10000
- MotorOn 1

Controllate che l'azionamento sia abilitato. Il motore potrebbe rimanere fermo o muoversi lentamente a causa degli offset. Impostate un leggero offset positivo fino a quando il senso di rotazione del motore risulta chiaro, ad esempio:

- DacOffset 1 100; 100 mV

Se il motore ruota in direzione negativa, occorre invertire la tensione di riferimento con il parametro RefDir del comando ConfigServo:

- ConfigServo 1 -1 -1 4; +1= riferimento normale, -1= riferimento invertito

Ora il motore deve girare nella direzione corretta. Non rimane che regolare l'offset con:

- DacOffset 1 0

Prendete nota dell'inversione o meno e ripetete il procedimento per gli altri assi.

2.2.5 Protezioni

Per ogni asse sono previsti gli ingressi di extracorsa avanti e indietro. La rilevazione di un fronte di salita su uno di questi ingressi manda in errore l'asse con conseguente disabilitazione immediata. Da notare che essendo l'errore provocato dal fronte e non dal livello del segnale è possibile, alla successiva riabilitazione, riportare l'asse nel campo di lavoro sotto il controllo del software applicativo.
Un'altra protezione è costituita dalla sorveglianza continua dell'errore di posizione che, se supera in valore assoluto il limite impostato, provoca l'errore dell'asse.
Il comando per impostare il limite dell'errore di posizione è:

- ErrorLimit <*asse*> <*valore*>

Va comunque tenuto presente che ci sono delle tipologie di guasto che non sono gestibili efficacemente dal controllore MC6, come ad esempio l'interruzione dei segnali encoder o dei guasti interni, che possono provocare la fuga dell'asse. è quindi necessario disporre delle opportune protezioni (freni, ammortizzatori, barriere

ecc.) per la salvaguardia del sistema e delle persone che vi operano.

2.2.6 Taratura PID

Il regolatore PID è da molto tempo usato nell'industria come metodo di controllo di sistemi in retroazione e rimane ancora oggi molto diffuso. Per l'impostazione si usano i comandi:

- Kp <*asse*> <*valore*> ; fattore proporzionale
- Ki <*asse*> <*valore*> ; fattore integrale
- Kd <*asse*> <*valore*> ; fattore derivativo
- Kvel <*asse*> <*valore*> ; fattore feed forward velocità
- Kacc <*asse*> <*valore*> ; fattore feed forward accelerazione
- DacLimit <*asse*> <*valore*> ; limitazione comando azionamento

La letteratura è ricca di procedure per la taratura dei sistemi PID, ma l'esperienza personale rimane la risorsa più importante. Comunque riportiamo nel seguito qualche consiglio.

Azionamenti in modo velocità

1. Impostare a zero Kp, Ki, Kd, Kvel, e Kacc.
2. Impostare DacLimit a 1 V.
3. Abilitare l'asse.
4. Effettuare dei posizionamenti con profilo trapezoidale e valori di decelerazione elevati, incrementando Kp fino al limite dell'oscillazione.
5. Continuare incrementando Kd per stabilizzare il sistema, se ci si riesce ripetere il punto 4.
6. Dimezzare i valori di Kp e Kd ottenuti in precedenza, quindi effettuare altri posizionamenti incrementando il valore di Ki per diminuire l'errore.
7. Continuare incrementare Kd per stabilizzare il sistema, se ci si riesce ripetere il punto 6.
8. Portare DacLimit a 10 V. Salvo casi particolari, non dovrebbe essere necessario impostare Kv e Ka.

Azionamenti in modo coppia

1. Impostare a zero Kp, Ki, Kd, Kvel, e Kacc.
2. Impostare DacLimit a 1 V.
3. Abilitare l'asse.
4. Effettuare dei posizionamenti con profilo trapezoidale e valori di decelerazione elevati, incrementando Kp fino al limite dell'oscillazione.
5. Continuare incrementando Kd per stabilizzare il sistema, se ci si riesce ripetere il punto 4.
6. Dimezzare i valori di Kp e Kd ottenuti in precedenza.
7. Portare DacLimit a 10 V. Salvo casi particolari, non dovrebbe essere necessario impostare Ki, Kv e Ka.

2.3 Movimenti indipendenti

2.3.1 Introduzione

Lo schema a blocchi seguente illustra la struttura dell'hardware e del software che permettono di muovere un asse in modo controllato:

Drawing 1: Schema a blocchi controllo asse

Drawing 2pppppop

I blocchi rappresentano i componenti software (programma applcativo), firmware (generatore profili, controllo servo) e hardware (controllore MC6, azionamento, motore, encoder).
Abbiamo già visto come fare i collegamenti hardware e come tarare il PID, ottenendo un sistema *controllato in posizione*, cioè un sistema in cui la *posizione effettiva* dell'asse (*ActPos*) ricalca più o meno fedelmente la *posizione di riferimento* (*RefPos*). La differenza tra posizione di riferimento e quella effettiva si chiama *errore di posizione* (*PosErr*) e, con una buona taratura del PID, dovrebbe essere molto contenuta.
A questo punto rimane da esaminare il funzionamento del *generatore di profili*, il blocco che provvede a determinare, istante per istante, il valore della posizione di riferimento RefPos. In pratica il programma applicativo deve impostare i parametri di accelerazione (*NomAcc*), decelerazione (*NomDec*), velocità (*NomVel*) e posizione da raggiungere (*Target*); al comando di start il generatore di profili calcolerà una serie di posizioni intermedie che porteranno l'asse dalla posizione iniziale a quella target nel modo desiderato.
Grazie alla potenza del processore RISC il controllore MC6 ha la capacità di ricalcolare il profilo del movimento in qualsiasi momento. Questo significa che è possibile cambiare velocità, accelerazione, decelerazione e posizione target anche durante il movimento; ad esempio si può iniziare un movimento ad alta velocità verso una posizione target nominale, diminuire la velocità quando si arriva ad una certa distanza ed infine modificare leggermente la posizione target in base alla lettura di un trasduttore.
Nello schema a blocchi è rappresentato un solo asse, ma in realtà il controllore MC6 contiene un generatore di profilo per ciascuno dei sei assi, che possono essere mossi in modo indipendente uno dall'altro. Per questo motivo vengono chiamati *generatori di profili indipendenti* e il tipo di movimento che ne deriva viene chiamato *movimento indipendente* dell'asse.

2.3.2 Parametri del profilo

I parametri del profilo si impostano con il comando:

- Profile <*asse*> <*velocità*> <*accelerazione*> <*decelerazione*>

Ad esempio:

- Profile 1 5000 300000 20000

imposta sull'asse 1 una velocità di 5000 conteggi/secondo, una accelerazione di 300000 conteggi/secondo^2, una decelerazione di 200000 conteggi/secondo^2.
In alternativa si possono impostare i singoli parametri con i comandi:

- Vel <*asse*> <*velocità*>
- Acc <*asse*> <*accelerazione*>
- Dec <*asse*> <*decelerazione*>
- Smoothing <*asse*> <*smoothing*>

che sono comodi anche per cambiare alcuni dei parametri durante il movimento (tranne che per lo smoothing che si può cambiare soltanto ad asse fermo).
Ci sono poi i comandi che restituiscono il valore corrente dei parametri:

- GetVel <*asse*> ; restituisce la velocità nominale
- GetAcc <*asse*> ; restituisce la accelerazione nominale
- GetDec <*asse*> ; restituisce la decelerazione nominale
- GetSmoothing <*asse*> ; restituisce lo smoothing

2.3.3 Posizionamenti

Un posizionamento assoluto è un movimento che porta l'asse a fermarsi in una posizione specificata, detta posizione target. Per eseguirlo si deve abilitare l'asse, impostare i parametri del profilo e dare il comando:

- MoveAbs <*asse*> <*target*> ; muove l'asse alla posizione target

Ad esempio per muovere l'asse 2 a 10000 conteggi encoder:

- MoveAbs 2 10000

Il movimento può essere interrotto in ogni momento con il comando:

- Stop <*asse*> ; interrompe il movimento

che provoca una frenata controllata con la decelerazione nominale. Il movimento può essere completato con un successivo comando di start. La posizione target, come tutte le posizioni, si esprime in conteggi encoder e deve essere compresa tra -2147483648 e 2147483647 (32 bit con segno).
Un posizionamento relativo invece è un movimento in cui si specifica *di quanto* si vuole spostare l'asse rispetto alla posizione attuale. Il comando è:

- MoveRel <*asse*> <*delta*> ; movimento relativo

Ad esempio per muovere l'asse 1 indietro di 1000 conteggi encoder:

- MoveRel 1 -1000

Anche in questo i limiti consentiti sono di -2147483648 e 2147483647 conteggi encoder.
è anche possibile dare un comando di movimento relativo mentre è in corso un movimento assoluto e viceversa: l'asse si muoverà in base all'ultimo comando dato.

2.3.4 Regolazione velocità

Alcune applicazioni richiedono di muovere l'asse indefinitamente ad una velocità prefissata. In questo caso si usa il comando:

- jog <*asse*> <*velocità*>

La velocità va specificata in conteggi encoder al secondo. Per terminare il movimento si può utilizzare il comando *stop* che provoca una decelerazione fino all'arresto oppure dare un comando di posizionamento se si vuole fermare l'asse in una posizione precisa.

2.3.5 Interrogazione asse

Ci sono diverse situazioni in cui è necessario interrogare l'asse, ad esempio si può essere interessati a conoscere quando il movimento è concluso oppure qual'è la posizione reale dell'asse. Per questo sono disponibili una serie di comandi:

- GetPos <*asse*> ; restituisce la posizione nominale
- GetActPos <*asse*> ; restituisce la posizione effettiva
- GetPosErr <*asse*> ; restituisce l'errore di posizione
- GetRefVel <*asse*> ; restituisce la velocita` istantanea del riferimento
- Status <*asse*> ; restituisce i bit di stato

Il comando Status restituisce una word i cui bit sono dei flag di stato molto importanti per sincronizzare il flusso del programma applicativo con il movimento dell'asse; in particolare il flag *Done* risulta molto utile.
L'esempio mostra il tipico andamento dei flag di stato durante un movimento:

- MoveRel 1 1000000 ; muovi l'asse 1 di 1 milione di conteggi
- Status 1 ; asse abilitato e in movimento
  6
- Status 1
  6
- Status 1 ; raggiunta velocità nominale
  22
- Status 1
  22
- Status 1
  22
- Status 1
  6
- Status 1
  6
- Status 1
  10

- Status 1
  42
- Status 1
  42

L'esempio seguente mostra come dare una sequenza di movimenti:

- MoveAbs 1 10000 ; muovi l'asse 1 a 10000 conteggi
- WaitDone ; attendi l'arrivo in posizione target 10000
- Jog 1 1000 ; muovi l'asse 1 a 1000 conteggi al secondo
- WaitDone 1 ; attendi la fine dell'accelerazione

2.3.6 Azzeramento

In precedenza abbiamo visto come effettuare dei posizionamenti specificando il target in termini di posizione assoluta.
All'accensione della scheda non è possibile conoscere la posizione effettiva dell'asse visto che gli encoder di tipo incrementale non forniscono questo genere di informazione, quindi si assume arbitrariamente come zero il punto in cui si trova l'asse in quel momento.
In alcune applicazioni questo può essere accettabile, ad esempio se si eseguono solo posizionamenti relativi o regolazioni di velocità. In altri casi invece è indispensabile che lo zero dell'asse, chiamato anche *origine*, corrisponda ad una posizione fisica ben precisa. Per ottenerlo si esegue una operazione chiamata *sequenza di azzeramento* che corrisponde al comando:

- Home <*asse*> <*modo*> <*velocità1*> <*velocità2*>

dove il parametro *modo* seleziona una delle seguenti sequenze:

Modo = 1

1. Movimento con direzione e velocità definita dal parametro *velocità1,* terminato con frenata controllata quando l'ingresso di home dell'asse diventa attivo.
2. Movimento con direzione e velocità definita dal parametro *velocità2,* terminato con frenata controllata al primo indice encoder incontrato dopo che l'ingresso di home è diventato inattivo.
3. Posizionamento con velocità definita dal parametro *velocità2* sull'indice encoder citato.
4. Assunzione della posizione raggiunta come nuova origine dell'asse.

Modo = 2

1. Movimento con direzione e velocità definita dal parametro *velocità1,* terminato con frenata controllata quando l'ingresso di home dell'asse diventa attivo.
2. Movimento con direzione e velocità definita dal parametro *velocità2,* terminato con frenata controllata quando l'ingresso di home diventa inattivo.
3. Posizionamento con velocità definita dal parametro *velocità2* sul punto di commutazione dell'ingresso di home.
4. Assunzione della posizione raggiunta come nuova origine dell'asse.

Modo = 3

1. Movimento con direzione e velocità definita dal parametro *velocità1,* terminato con frenata controllata quando si incontra il primo indice encoder.
2. Posizionamento con velocità definita dal parametro *velocità1* sull'indice encoder citato.
3. Assunzione della posizione raggiunta come nuova origine dell'asse.

Oltre al comando Home sono disponibili i comandi ArmIndex, ArmHome e SetOrigin che permettono di definire delle sequenze di reset personalizzate. Per ulteriori informazioni consultate la guida di riferimento software.

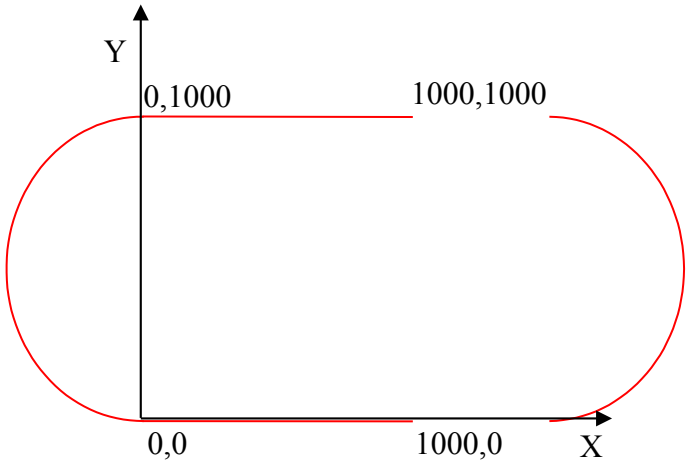
2.4 Movimenti coordinati

2.4.1 Generalita`

I movimenti coordinati consentono di muovere piu` assi contemporaneamente lungo un percorso predeterminato nello spazio. Il controllore MC6 permette due sistemi di assi coordinati attivi contemporaneamente, uno per ciascun asse virtuale. Ogni sistema prevede tre assi cartesiani X, Y, Z piu` un asse di rotazione W, che possono essere associati a qualunque asse servo o stepper della scheda con il comando *ConfigCoord*:

```
- ConfigCoord 13 2 3 0 8        ; asse virtuale 13 coordina asse 2 (X), asse 3(Y)
                                ; asse 8 (W). L'asse Z non e` presente.
```

Rispetto ai movimento indipendenti, i movimenti coordinati richiedono che venga definito un *percorso*, cioe` un insieme di punti che devono essere attraversati in successione. Il percorso viene definito da una serie di *segmenti* (tratti di linea e archi di cerchio) ciascuno dei quali inizia dalla fine del segmento precedente.
Ecco un esempio di percorso e di comandi per definirlo:




```
- BeginCoord 13                 ; inizio percorso associato all'asse virtuale 13
- LineAbs 13 1000 0 0 0         ; segmento lineare alla posizione x=1000,
y=0
 1000
- ArcAbs 13 1000 500 180000 0   ; arco di  180° con centro x=1000,y=500
 1570
```

- LineAbs 13 0 1000 0 0 ; segmento lineare alla posizione x=0, y=1000
  1000
- ArcAbs 13 0 500 0 180000 0 ; arco di 180° con centro x=0,y=500
  1570
- Profile 13 1000 100000 200000 ; imposta velocita` e accelerazioni
- MoveAbs 13 5140 ; posizionamento a fine percorso
- WaitEnd 13 ; attesa completamento

Ogni segmento ha una *posizione iniziale* (coordinate del punto iniziale), una *posizione finale* (coordinate del punto finale) ed una *lunghezza* (spazio percorso lungo il segmento). Anche l'intero percorso ha una posizione iniziale, una posizione finale ed una lunghezza, quest'ultima pari alla somma delle lunghezze dei segmenti che lo compongono.

Una volta definito il percorso gli assi sono vincolati a muoversi lungo di esso, analogamente ad un treno che e` obbligato a muoversi sui binari. Come avviene per le linee ferroviarie, qualsiasi punto lungo il percorso e` identificato dalla distanza dalla stazione iniziale (es. casello al km 127), come se il percorso venisse "raddrizzato" e allineto con un asse di riferimento. Viene allora naturale considerare un movimento coordinato come l'insieme del percorso e di un asse virtuale che viene usato per muoversi lungo la coordinata curvilinea con i comandi gia` disponibili per gli assi indipendenti.

L'origine dell'asse virtuale viene automaticamente ridefinita nella posizione corrente dal comando *BeginCoord*, in modo che la posizione 0 corrisponda al punto iniziale del percorso. Muovendo l'asse virtuale in direzione positiva, gli assi coordinati vengono automaticamente mossi lungo il percorso in modo che la coordinata curvilinea (la distanza percorsa calcolata secondo le usuali formule matematiche) coincida con la posizione corrente dell'asse virtuale. Velocita`, accelerazione e decelerazione scalari (cioe` la componente di velocita`, accelerazione e decelerazione nella direzione del percorso) corrispondono esattamente a quelle del'asse virtuale.

L'asse virtuale puo` essere controllato con tutti i comandi per assi indipendenti, ad esempio *MoveRel*, *MoveAbs*, *Jog*, *Vel*, *Acc*, *Dec*, *Smoothing*, *Profile*, *Start*, *Stop*, *GetPos,* ma escusivamente in direzione positiva, altrimenti viene generato un errore sugli assi coordinati. Una volta raggiunta la fine del percorso gli assi coordinati si arrestano anche se l'asse virtuale procede ulteriormente.

Per effettuare il percorso dell'esempio precedente bisogna muovere l'asse virtuale 13 nella posizione 5140 (i tratti lineari sono lunghi 1000, i semicerchi sono lunghi 500*3.14=1570 per cui D=1000+1570+1000+1570=5140). Tutti i comandi che definiscono un segmento ne ritornano la lunghezza, che puo` essere utilmente impiegata nel programma applicativo per sincronizzare degli eventi (ad esempio l'attivazione di una uscita) al raggiungimento di particolari posizioni.

Il movimento puo` essere arrestato prima della fine con un comando *Stop* all'asse virtuale. Successivamente puo` essere riavviato con il comando *Start* o, in alternativa, si puo` cancellare la parte rimanente del movimento con un ulteriore comando *BeginCoord*.

2.4.2 Segmenti lineari

Per definire un segmento lineare e` sufficiente specificare le coordinate della posizione finale del segmento in termini assoluti o relativi alla posizione finale del segmento precedente. Da notare che l'asse W viene considerato come un quarto asse cartesiano e quindi significativo nel calcolo della lunghezza del segmento.
I comandi sono:

- LineAbs <*handle*> < *xf*> < *yf*> <*zf*> <*wf*>
- LineRel <*handle*> < *dx*> < *dy*> <*dz*> <*dw*>

2.4.3 Movimenti circolari

Per definire un segmento circolare e` sufficiente specificare le coordinate del centro in termini assoluti o relativi alla posizione finale del segmento precedente, l'angolo di rotazione e la rotazione dell'asse w.
I comandi sono:

- ArcAbs < *handle* > <*center_x*> <*center_y*> <*angle*> <*dw*>
- ArcRel < *handle* > <*center_dx*> <*center_dy*> <*angle*> <*dw*>

La rotazione dell'asse w non viene considerata nel calcolo della lunghezza del segmento e puo` essere utile per mantenere l'asse w tangente al cerchio durante il movimento.

2.4.4 Movimenti continui

Il controllore MC6 permette di definire il percorso in tempo reale aggiungendo segmenti via via che ne vengono completati altri. Tale prestazione permette di eseguire percorsi composti da un numero arbitrario di segmenti e quindi approssimare una curva arbitraria con una serie di piccoli segmenti lineari. A questo scopo risulta utile il comando *Segments* che ritorna il numero di segmenti liberi (all'interno della scheda ne possono venire memorizzati fino a 500 per asse virtuale). Questo modo operativo va usato con cautela perche` e` responsabilita` del programma applicativo definire i nuovi segmenti in tempo utile. In caso contrario gli assi coordinati verranno arrestati istantaneamente nel punto finale dell'ultimo segmento.

2.5 Funzioni speciali

2.5.1 Ingranaggio elettronico

La funzione di *ingranaggio elettronico*, chiamata anche *albero elettrico* o *gearing,*, consiste essenzialmente nel muovere un asse *slave* con una velocita` proporzionale a quella di un asse *master*. Una volta definito il rapporto di trasmissione e abilitato il *gearing*, l'asse *slave* si muovera` come se esistesse una trasmissione meccanica

che lo collega al *master*.
Si utilizza il comando seguente:

- Gearing <*Slave*> <*Master*> <*Num*> <*Den*>

Dove *Slave* e *Master* individuano gli assi interessati, mentre *Num* e *Den* definiscono il rapporto di trasmissione.
Ad esempio:

- Gearing 4 1 2000 3000

definisce il gearing tra l'asse 4 (slave) e l'asse 1 (master). L'asse 4 si muovera` di 2000 conteggi encoder per ogni 3000 conteggi percorsi dall'asse 1.
Il controllore MC6 permette di definire un gearing per ogni asse (come slave), che puo` utilizzare come master la posizione nominale o effettiva di qualunque altro asse. Ciascun gearing ha un proprio rapporto di trasmissione che puo` essere anche negativo: in questo caso lo slave si muovera` in direzione opposta al master.

Il gearing puo` essere disabilitato indicando come master l'asse 0.

2.5.2 Frizione elettronica

La funzione di *frizione elettronica* o *clutch* si utilizza in abbinamento al gearing e permette, analogamente ad una frizione meccanica, di innestare o disinnestare gradualmente la trasmissione con l'asse master in movimento, evitando di accelerare bruscamente l'asse slave.
Una caratteristica importante del comando e` di realizzare l'innesto/disinnesto della frizione sulla base della posizione del master, garantendo la transizione avvenga nell'intervallo compreso tra una posizione iniziale (del master) ed una finale specificate.
Altra caratteristica importante e` che lo spazio percorso dallo slave nelle fasi di innesto/disinnesto e` noto a priori, permettendo l'aggancio al master con una relazione di posizione o fase predeterminata; questo per esempio consente di realizzare in modo estremamente semplice ed accurato applicazioni di taglio al volo.
Per innestare la frizione si utilizza il comando:

- ClutchOn <*Slave*> <*Mode*> <*BeginPos*> <*EndPos*>

Dove *Slave* identifica l'asse, *Mode* se le posizioni sono assolute o relative, *BeginPos* la posizione del master in cui inizia l'innesto della frizione, *EndPos* la posizione del master in cui si conclude l'innesto della frizione. Lo spazio percorso dallo slave in accelerazione e` :

$$S = \frac{(EndPos - BeginPos)}{2} * GearRatio$$

Ad esempio:

- ClutchOn 4 1 7000 10000

Innesta il gearing tra lo slave 4 ed il relativo master quando quest'ultimo percorre il tratto compreso tra le posizioni assolute 7000 e 10000. In pratica lo slave rimane fermo finche` il master non raggiunge la posizione 7000, quindi accelera

opportunamente per essere in gearing con il rapporto di trasmissione richiesto nel momento in cui il master raggiunge la posizione 10000. Se il rapporto di trasmissione fosse 2000/3000, lo slave percorrerebbe in accelerazione lo spazio:

$$S = \frac{(10000 - 7000)}{2} * 2000/3000 = 1000$$

Quindi nel momento in cui il master raggiunge la posizione 10000, lo slave raggiunge la posizione 1000; da quel punto in poi i due assi sono agganciati con il rapporto di trasmissione 2000/3000.

In modo del tutto analogo funziona il comando che permette di disinnestare la frizione :

- ClutchOff <*Slave*> <*Mode*> <*BeginPos*> <*EndPos*>

Ad esempio:

- ClutchOff 4 0 0 1000

Inizia immediatamente a disinnestare la frizione (posizione relativa 0) concludento tra 1000 conteggi encoder del master.

Da notare che i due comandi possono essere attivi contemporaneamente, purche` gli intervalli di innesto e disinnesto della frizione non si sovrappongano.
La frizione elettronica richiede che il master si muova in direzione positiva.
All'accensione per default tutti gli assi hanno la frizione elettronica innestata.

2.5.3 Camma elettronica

La funzione di camma elettronica consiste nel muovere un asse *slave* in base alla posizione di un asse *master* secondo una legge definita da una tabella dove e` memorizzato il profilo desiderato della camma. Questo e` il comportamento tipico di una camma meccanica, in cui la posizione angolare dell'albero e la forma della camma determinano la posizione dell'organo condotto.
Ogni asse (compresi anche gli stepper e quelli virtuali) puo` essere agganciato come *slave* di qualunque altro asse (*master)* della scheda. Inoltre e` possibile agganciare o sganciare automaticamente ciascuna camma quando il relativo asse master raggiunge una posizione prestabilita.

Per utilizzare questa prestazione occorre innanzitutto definire il profilo della camma in forma di tabella, ad esempio utilizzando un foglio elettronico del PC. Il numero di punti della tabella e` arbitrario e dipende dal grado di precisione richiesto; in genere bastano alcune decine di punti perche` il controllore MC6 provvede a generare tutti i valori intermedi per interpolazione lineare, ma e` possibile definire anche tabelle di migliaia di punti.

Le tabelle sono allocate in *blocchi* di memoria del controllore sui quali è possibile scrivere da remoto (via can bus o seriale) per la massima flessibilità. Ovviamente a livello di programma applicativo le tabelle possono anche venire calcolate in modo algoritmico.
Il comando per definire una camma è;

Cam <*Handle*> <*Master*> <*Modulo*> <*Block*>

Dove *Handle* identifica l'asse slave, *Master* identifica l'asse master, *Modulo* e` il numero di count dell'asse master che corrisponde ad un giro della camma, *Block* e` l'handle del blocco che contiene il profilo della camma.

Infine va inserita la camma con il comando:

- Engage <*Handle*> <*Mode*> <*Pos*>

Dove *Handle* identifica l'asse, *Mode* se la posizione e` assoluta (1) o relativa (0), *Pos* la posizione del master in cui va innestata la camma. Ad esempio per innestare immediatamente la camma in cui l'asse 2 e` lo slave:

- Engage 2 0 0

In modo analogo si puo` disinnestare un camma con il comando:

- Disengage <*Handle*> <*Mode*> <*Pos*>

Il significato dei parametri e` simile al comando *Engage.*