Volume

1

AxesBrain

Sistema di movimentazione

Sistema di movimentazione

AxesBrain

Sistema di movimentazione

AB&T Tecnologie Informatiche™
Via dell'About, 2/A • 10015 Ivrea
Fax 0125 234397
www.bausano.net
info@bausano.net

Informazioni legali

Sommario

AxesBrain

Il prodotto "AxesBrain" è stato sviluppato per offrire dei servizi di movimentazione agli applicativi utente, utilizzando la tecnologia "DCOM", sarà così possibile avere l'accesso a questi servizi non solo dallo stesso PC, ma anche da PC integrati in rete locale o collegati tramite Internet.

Per utilizzare i servizi di movimentazione possono essere utilizzate essenzialmente due strade o utilizzando le funzioni dirette alle risorse assi, mandrini e i segnali di ingesso uscita oppure avvalendosi di cicli programmati. Tramite i linguaggi di programmazione vengono definite delle procedure o cicli di percorsi degli assi e di operazioni di manipolazione e lavorazione, nel nostro caso abbiamo tre linguaggi disponibili: AxesBrainL, AxesBrainISO e AxesBrainAWL che l'utente può utilizzare a secondo del tipo di applicazione che si viene a presentare, è inoltre possibile integrare come libreria , delle funzionalità o addirittura dei processi per la gestione di cicli di lavorazione personalizzati ad esigenze specifiche, completamente sviluppate dall' utente.

I servizi di movimentazione sono:

- FUNZIONI dirette
    - comandi agli assi e ai segnali d'ingrasso uscita
    - acquisizioni dei valori dal campo , lettura assi, I/O, trasduttori, ecc.
    - impostazioni delle modalità per le traiettorie, le acquisizioni , ecc.
- CICLI di lavorazione e manipolazione con programmazione tramite
    - AxesBrainL - Linguaggio proprietario adatto per descrivere i cicli di manipolazione per l'automazione in genere
    - AxesBrainISO - Linguaggio standard ISO adatto per descrivere i cicli di lavorazione tipici della fresatura, foratura e tornitura.
    - AxesBrainAWL - Linguaggio AWL per descrivere i cicli di PLC
- FUNZIONI speciali "OEM"
- CICLI di lavorazione e manipolazione con la programmazione di procedure speciali "OEM "

Il "AxesBrain" è un prodotto articolato in moduli funzionali, in modo da poter avere una ampia flessibilità sia a livello di gestione dei dispositivi che in termini interfaccia utente che di prestazioni.

Il "AxesBrain" è così suddiviso:

- "AxesBrain OpenEnviroment" modulo per il pilotaggio dei dispositivi d' interfaccia, oltre che gestire alcune gamme di dispositivi commerciali è stato sviluppato in modo da poter ampliare i dispositivi a seconda delle esigenze degli utenti .

- "AxesBrainServer" modulo base, fornitore dei servizi del sistema in tecnologia "DCOM" .

- "AxesBrainOpenTemplate" modulo di classi base per sviluppare nuove prestazioni direttamente dal l'utente, sia in termini di funzionalità che di procedure.

- "AxesBrainBase" modulo d'intefaccia per la connessione con i servizi per applicativi in Visual Basic o Visual C++

- "AxesBrainCom" modulo d'intefaccia per la connessione con i servizi per le pagine utente HTML o per applicativi scritti nei vari linguaggi di programmazione.

- "AxesBrainStudio" ambiente d'interfaccia utente, modulo che permette la realizzazione d'interfacce utente con alto grado di personalizzazione.

Schema dell'architettura AxesBrain

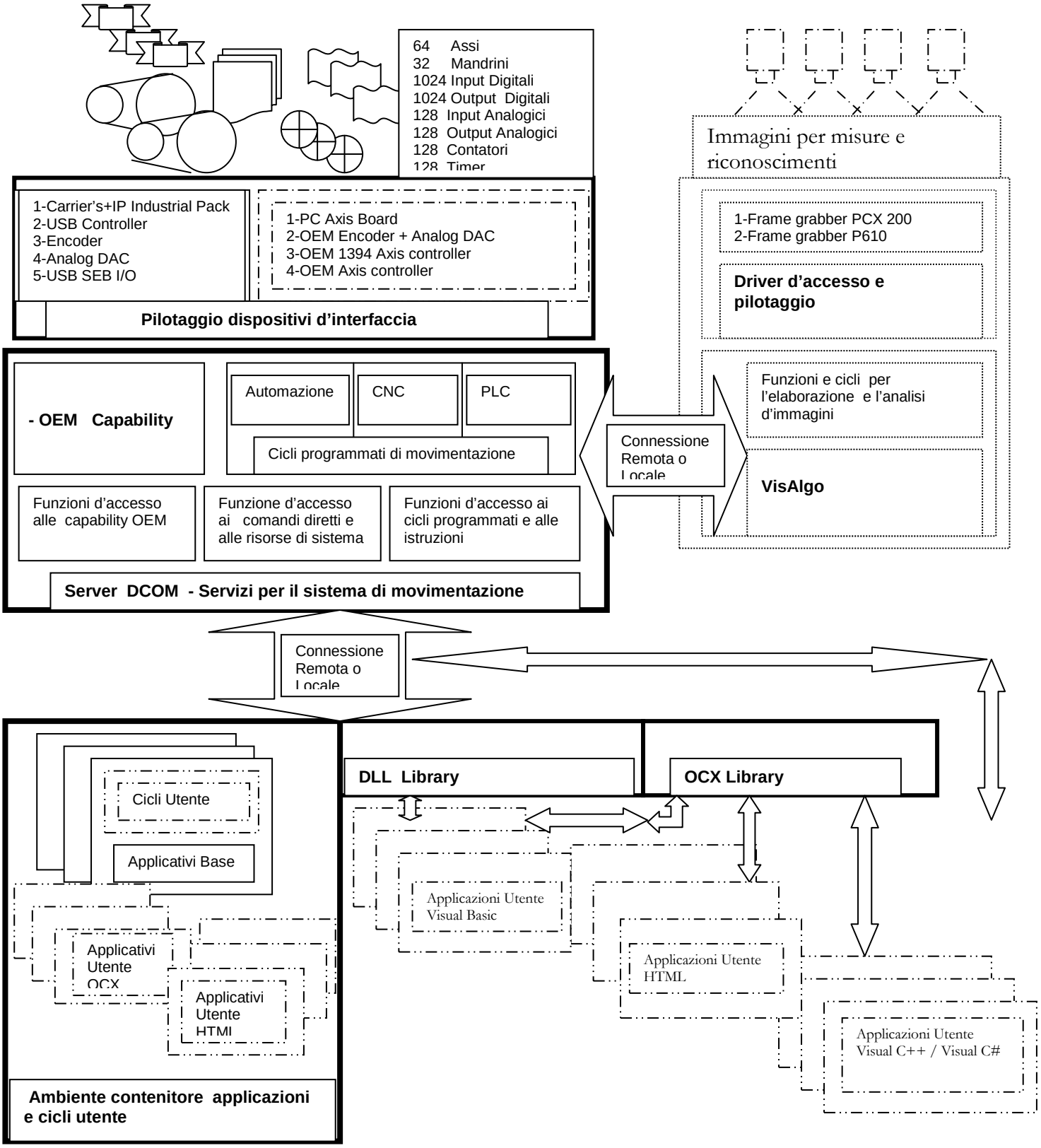


Figura 1

Capitolo

1

Sistema di movimentazione

Il sistema di movimentazione è il gestore di tutte le risorse ed è il supervisore delle funzionalità di movimentazione richieste sia direttamente che tramite i cicli di lavoro, programmati utilizzando i tre linguaggi di programmazione, inoltre può integrare come libreria delle funzionalità e dei processi di gestione cicli di lavorazione, completamente sviluppate del Cliente.Una particolare attenzione è stata data alla "scalabilità" del sistema in termini di dispositivi , di funzionalità , di impianto e d'interfaccia utente.

Le esigenze dell'ambiente in cui un sistema di movimentazione si viene a trovare sono molteplici, vanno:

- dal tipo di tipologia diversa dei motori:

  motori a corrente continua,motori passo,motori brushless, motori lineari, ecc.

- dal loro sistema di pilotaggio:

  azionamenti analogici, azionamenti digitali, ecc.

- dal tipo di dispositivo elettronico che permette di accedere ai loro sistemi di pilotaggio:

  PCAxisBoard, PCFieldBus, PCEncoderBoard, PCAnalogBoard , ecc.

- dal tipo di bus PC:

  PCI, AT, 104,CompactPCI, etc

- dal tipo di sistema operativo

  Windows 98, Windows Me, Windows 2000

Il sistema AxesBrain ha cercato di rendere flessibile il suddetto mondo, creando uno strato chiamato **AxesBrainOpenEnviroment**, dove risiedono una serie di moduli per il pilotaggio di alcuni dispositivi commerciali e una classe di base per poterne facilitare lo sviluppo e l'integrazione di nuovi. Questo permette anche la separazione tra ambiente di pilotaggio e gli applicativi stessi.

Le funzionalità che un sistema di gestione della movimentazione deve avere è quella di fornire i suoi servizi a più utilizzatori, che possono risiedere anche su PC remoti, a questa esigenza è venuta incontro la tecnologia DCOM (Distributed Component Object Model) con cui è stato realizzato il **AxesBrainServer**.

Per quanto le funzionalità siano di ampia fascia, è impossibile sviluppare tutte le esigenze che un sistema di movimentazione dovrebbe avere, per questo motivo è stato progettata la possibilità di avere una espandibilità in termini di prestazioni, che in questo caso sarà lo stesso Cliente a sviluppare e che il sistema integrerà in termini di libreria OEM, libreria che si appoggia su un "Template" di classi specifiche, questa possibilità in di termini di ampliamenti funzionali è realizzata dal modulo **AxesBrainOpenTemplate**.

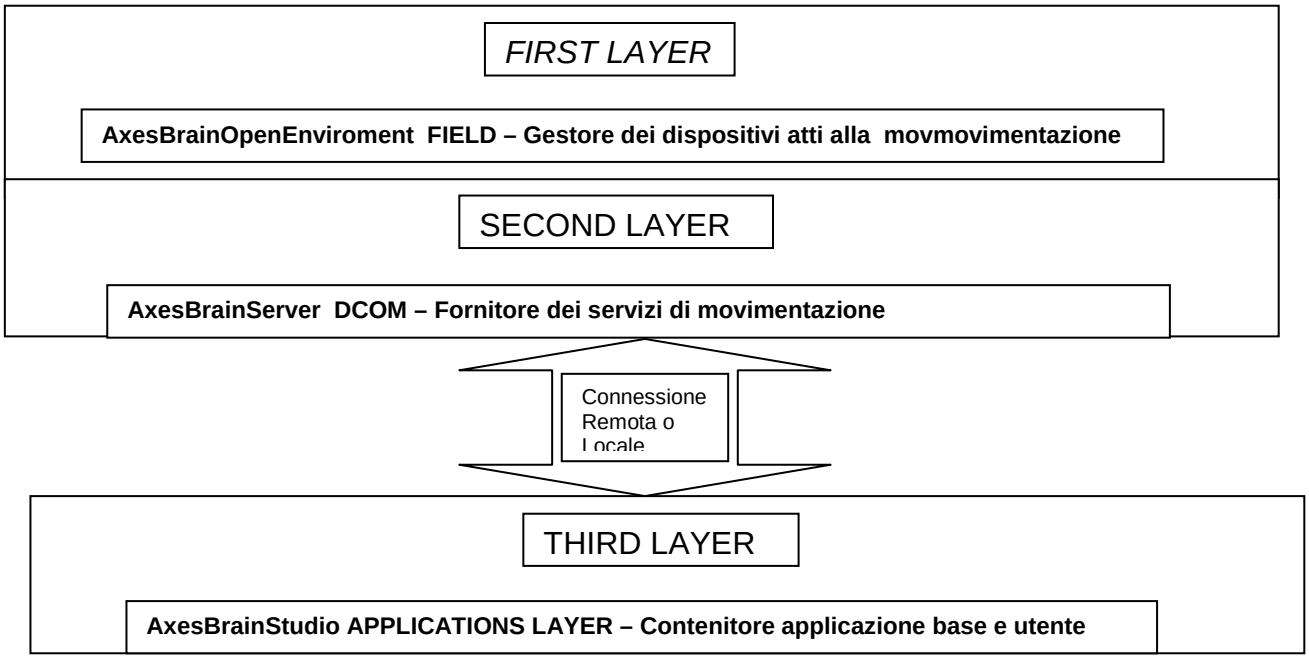
L'interfaccia Utente è realizzata come Client del modulo AxesBrainServer può essere liberamente sviluppata dal Cliente, utilizzando un qualsiasi linguaggio di programmazione in grado di appoggiarsi alla tecnologia DCOM (Distributed Component Object Model), per rendere più efficiente e per aumentare la gamma dei linguaggi di programmazione sono state sviluppate due librerie d'interfacciamento ai servizi DCOM del AxesBrainServer , la prima è una libreria in tecnologia DLL chiamata **AxesBrainBase** e la seconda in tecnologia OCX chiamata **AxesBrainCom** .

Per permettere un utilizzo di alcune interfacce utente di consueto uso come l'editing di programmi, la caratterizzazione dell'impianto, la sua conduzione, il monitoraggio dei segnali e degli stati etc. , sono state realizzate alcune applicazione base, le quali per potersi integrare in modo uniforme alle applicazioni utente sono state inserite in contesto chiamato **AxesBrainStudio**

Architettura

L'architettura del AxesBrain è suddivisa su tre livelli:

- AxesBrainOpenEnviroment (First Layer FIELD) Gestore dei dispositivi atti alla movimentazione.
- AxesBrainServer (SecondLayer DCOM "Distributed Component Object Model ") Fornitore di servizi di movimentazione
- AxeBrainStudio (Third Layer APPLICATION LAYER) Contenitore applicazioni utente e base per la gestione delle funzioni e dei cicli di movimentazione, applicazioni clienti del DCOM AxesBrainServer .
  Nel terzo livello sono collocate le applicazioni utente.



Dispositivi d'interfaccia

Dei moduli software chiamati "Drivers" per la gestione dei dispositivi sono collocati in una libreria e sono stati sviluppati appoggiandosi ad una classe di base, questa architettura aperta ha il compito di facilitare sia lo sviluppo di nuovi moduli che il loro ampliamento prestazionale
Come la parola "OpenEnviroment" tende a sottolineare il fatto che il sistema è stato progettato per poter includere altri dispositivi, man mano che nuove richieste si vengono ad aggiungere.
I "Driver" sono realizzati con la tecnologia "WDM" (Windows Driver Model) , in modo che sono indipendenti dal sistema operativo Windows Microsoft :

Windows 98
Windows Me (Millenium Edition)
Windows 2000 .

I dispositivi attualmente gestiti dal AxesBrainOpenEnviroment si suddividono in gamme e sono:

- Gamma "Industrial Pack" con "Carrier " SBS per PCI, CompactPCI, PC104 , PCAT
  - IP Tews 1,2 assi analogici-encoder
  - IP SBS unidig T 4 timer/contatori
  - IP SBS unidig 48 48 I/O digitali
  - IP SBS 8 DAC 8 Output analogici
  - IP SBS 32DAC 32 Output analogici
  - IP SBS 16DAC 16 Input analogici
- Gamma USB
  - USBControllore Assi "Encoder e Stepping" e IO Digitali e analogici
  - USBControllore IO Digitali e analogici
- Gamma PC Board
  - Assi Encoder
  - Output analogici
  - PC8255 4 Timer + 48 I/O TTL digitali
- Gamma PC AxisBoard
  - (in fase d'analisi)

La gamma "Industrial Pack" con "Carrier " SBS per PCI, CompactPCI, PC104 , PCAT se da una parte è la più onerosa dal punto di vista finanziario, consente tre importanti aperture:

- Compatibilità con tutte le piattaforme PC (PC-AT , PC-104, PCI, CompactPCI).

- Possibilità di raggiungere fisicamente la massima espandibilità in termini di :

    - 64 assi fisici
    - 32 mandrini
    - 1024 input digitali
    - 1024 output digitali
    - 128 input analogici
    - 128 output analogici
    - 128 timer
    - 128 contatori.

- Ottenere una campionatura di **100 microsecondi** per 8 assi, ovviamente utilizzando processori di almeno 500 MHz .

La gamma USB che è la più facile da utilizzare, essendo collegabile al PC attraverso la porta USB, ormai presente su tutti i PC nella loro diversa tipologia, ha un inconveniente in termini di campionatura essendo fissa a 1 milli secondo, L'espandilbilità è di:

- 32 assi fisici
- 32 mandrini
- 512 input digitali
- 512 output digitali
- 64 input analogici
- 64 output analogici
- 16 contatori.

La gamma PC Board se da una parte è la più economica, presenta grossi problemi dal punto di vista dell'espansione e delle compatibilità tra le diverse piattaforme.

La gamma PC AxisBoard è in fase d'analisi.

La piattaforma AxesBrainOpenEnviroment è progettata in modo da poter dare un **"scalabilità**" del sistema di movimentazione **in termini di dispositivi** .

Note:
Particolare attenzione è attualmente posta per i dispositivi 1394 FireWire.

Fornitore di servizi per la movimentazione (DCOM)

Il AxesBrainServer è il fornitore di servizi di movimentazione delle risorse ed acquisizione dei dati dal campo, è stato progettato in architettura DCOM (Distributed Component Object Model) per due importanti esigenze:

- MULTI UTENZA - più applicativi concorrono nello stesso tempo nella richiesta di servizi di gestione movimentazione

- UTENZA REMOTA - alcuni o addirittura tutti gli applicativi possono trovarsi su macchine diverse, purché collegate in rete locale od in "Internet".

Si possono cosi costruire vari tipi di scenari:

- Work Station (User Interface) + Controllore su una solo PC

- Work Station (User interface) su un PC + Controllore su una secondo PC

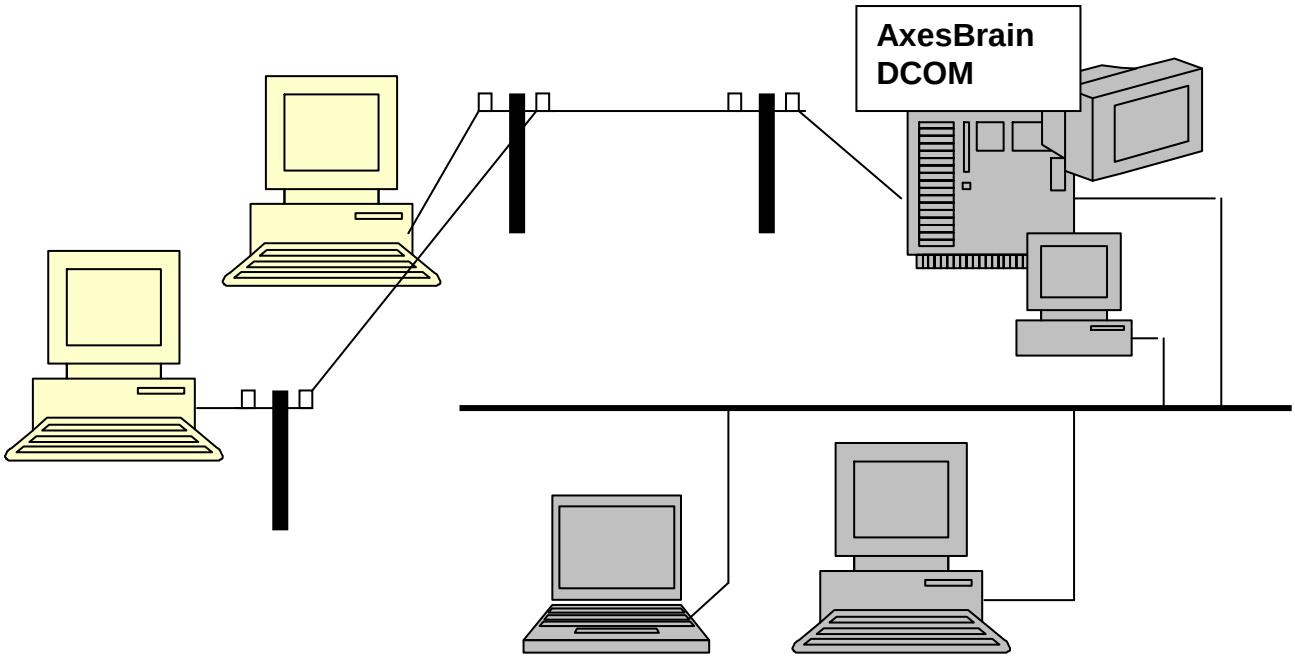
- Controllore senza User Interface, ma collegandolo ad un PC tramite rete è possibile averne la gestione

- Controllore con più di una User Interface, ad esempio una seconda User Interface solo per il scarico/carico.

- Connettersi da qualsiasi PC remoto per poter effettuare operazioni di verifica e controllo

- Controllare in remoto macchine non presidiate.

Il "AxesBrainServer" è stato progettato in modo da poter dare un **"scalabilità**" del sistema di movimentazione **in termini di funzionalità** .



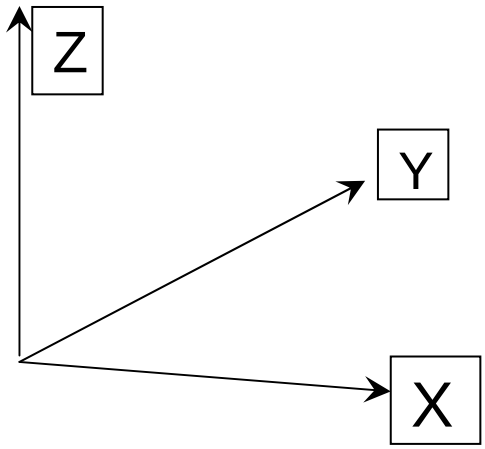
Sistemi di coordinate e anticollisione

Le risorse sono le parti costituenti il campo e sono:

- Assi
- Mandrini
- Input digitali
- Output digitali
- Input analogici
- Output analogici
- Timer
- Contatori

Gli assi possono lavorare in sistemi di coordinate diverse, AxesBrain prevede tre tipologie di sistemi di coordinate più alcune varianti dei medesimi:

Sistema Cartesiano



Gli assi sono sulle tre coordinate X,Y e Z , in questo caso abbiamo normale gruppo di lavoro:

1. X asse della ascissa
2. Y asse della ordinata
3. Z asse della altitudine

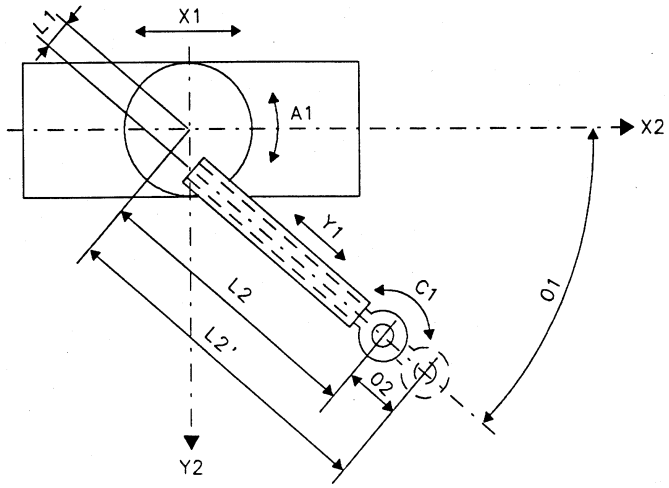
Sistema Polare

Sistema cilindrico tipo “canotto”:

composto da un asse rotante chiamato A1 più un asse a canotto chiamato Y1 asse C1 con compensazione rotante dell’angolo definito dall’asse A1.

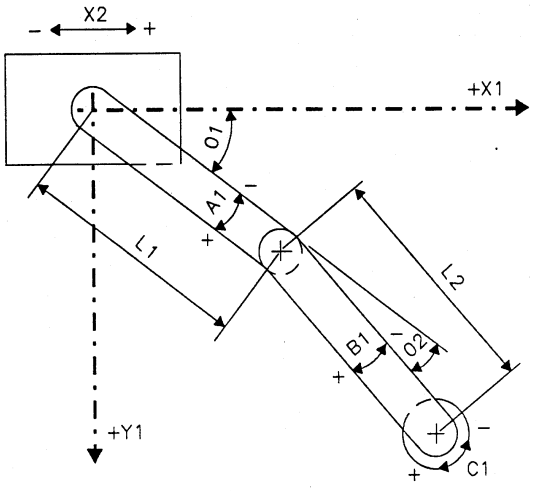
Sistema cilindrico tipo “traslante”

Composto da un asse rotante chiamata A1 più un asse traslante chiamato X1 Asse C1 con compensazione dell’angolo definito dall’asse A1



Sistema Scara

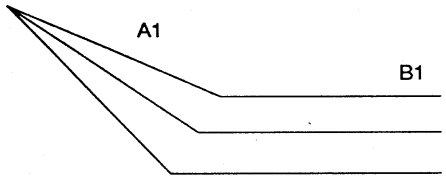
Sistema composto da due assi rotanti A1 e B1 più un terzo asse rotante C1 per la compensazione delle rotazioni di A1 e B1.



Pantografo

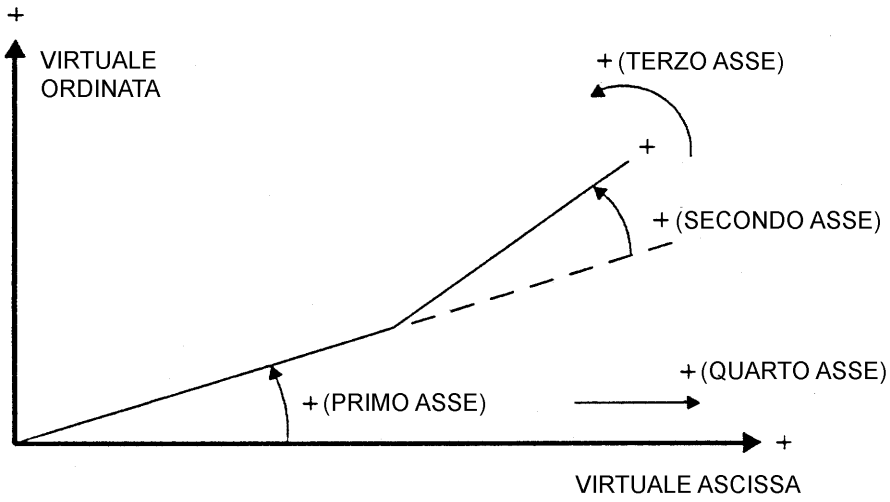
Tipo di Scara con asse B meccanicamente compensato

come la figura rappresenta l'asse B1 rimane meccanicamente vincolato alla posizione rispetto agli assi cartesiani.Anche in questo sistema esiste un asse C1 per la compensazione degli assi A1 e B1



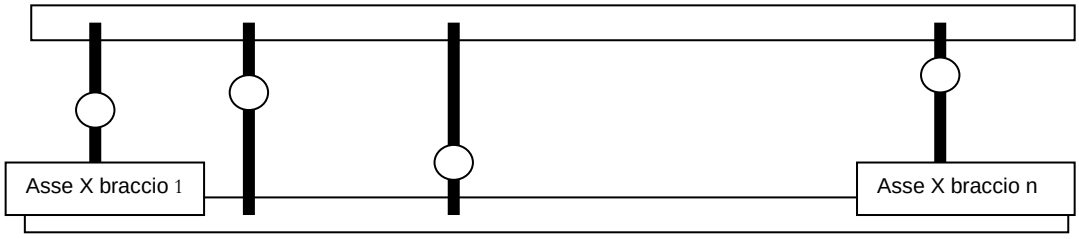
Assi Virtuali

Nei sistemi polari e "SCARA" esiste tuttavia la necessita di poter riferirci con degli assi come fossimo nel sistema di coordinate cartesiane, per cui in questo caso introduciamo il concetto di **assi virtuali** (sistema cartesiano) rispetto agli **assi fisic**i che ne permettono il movimento .



Anticollisione

Gli assi vengono così raggruppati in modo che possono tra loro lavorare separatamente, gli assi possono incidere però su unico asse fisico, esempio portale con più bracci che lavorano sullo stesso asse portante, in questo caso il sistema è in grado di risolvere il problema dell'anticollisione, gestendo l'arresto momentaneo di un asse in attesa che l'altro si porti in una posizione di fuori ingombro.



In questo caso la programmazione avviene con ciclo separato per ogni braccio, altrimenti sarebbe oneroso pensare alla movimentazione di tutto il sistema, sarà il sistema "AxesBrain" che con la funzionalità di anticollisione controllerà la gestione degli ingombri, unica attenzione è che la programmazione dei cicli non porti dei casi di abbraccio mortale "DEAD LOCK".

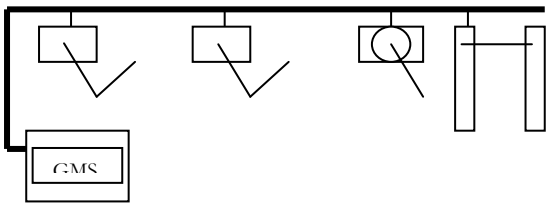
Configurazioni d'impianti

Gli assi possono suddividersi su più sistemi di coordinate, lavorare insieme o separati, su cicli diversi e concorrenti.

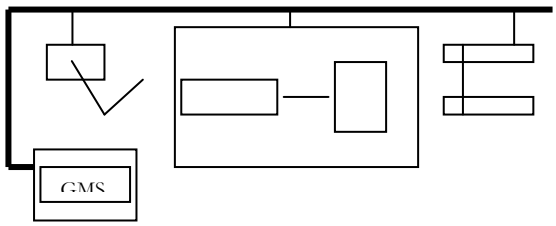
L'aggregazione degli assi è dinamica gestita solo dai comandi o dalle istruzione dei cicli programmati, possiamo quindi avere più sistemi in lavorazione, linee di Robots alcuni con geometria SCARA altri, POLARI, altri ancora con geometria cartesiana. Con questa filosofia possiamo avere una macchina operatrice con sistemi di alimentazione integrata, l'unico limite è nel numero massimo di assi in 64.

Esempi di configurazioni d'impianto:

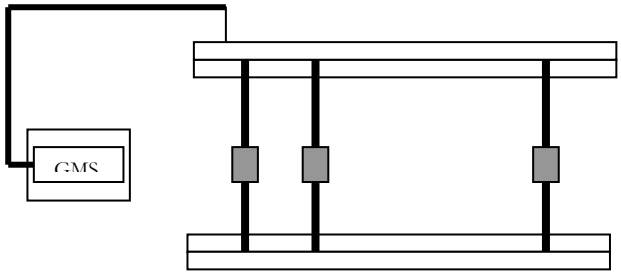
Linea con ROBOT "Scara", Polari , cartesiani



Macchina operatrice con carico effettuato da un Robot "Scara" e scarico con Robot cartesiano a portale



Macchina operatrice con 3 bracci cartesiani incidenti su un unico portale



I sistemi di coordinate e le configurazioni permettono una **"scalabilità**" del sistema di movimentazione **in termini di impianto** .

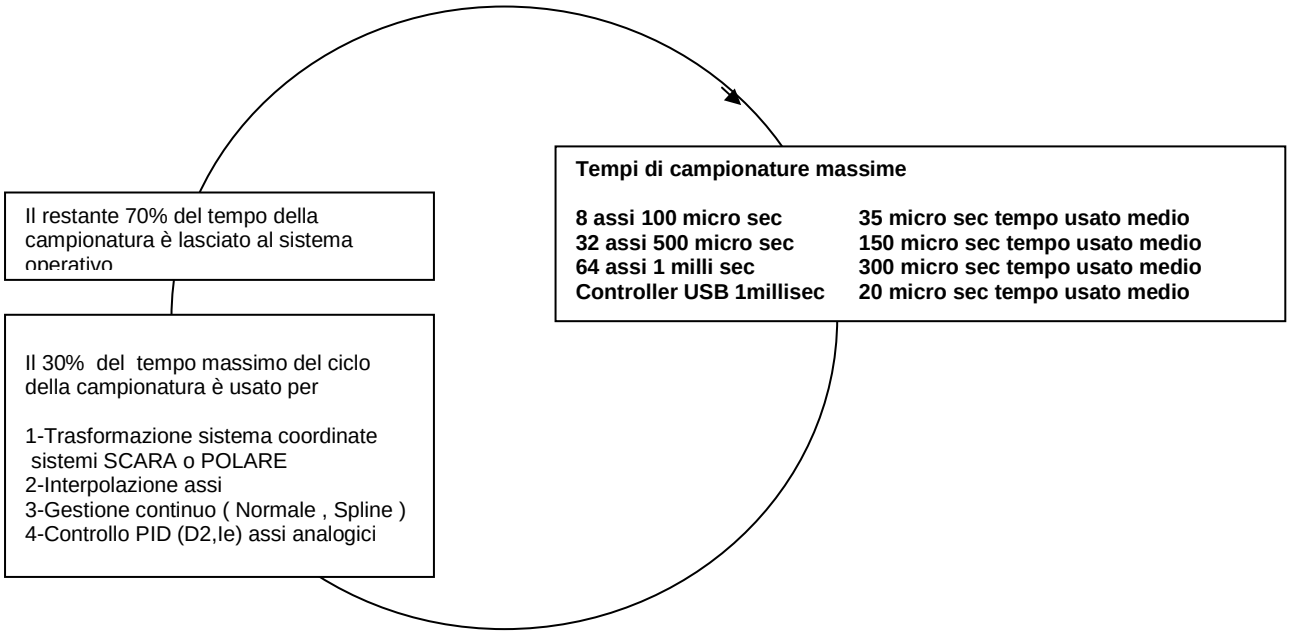
Campionatura

Per la gestione degli assi è necessario un tempo di campionatura, il cui compito è di effettuare ciclicamente i calcoli per la:

1. Trasformazione dei sistemi di coordinate "SCARA" e POLARE
2. Interpolazione degli assi coordinati.
3. Gestione della movimentazione continua degli assi coordinati (Normale, SPLINE)
4. Controllo dell' PID (D2,Ie) per ogni asse analogico

L'impegno della CPU per gestire gli assi è quindi pari al numero di assi e ai calcoli da effettuare, il punto 4 , controllo dell' PID (D2,Ie) con dispositivi EMBEDDED DSP o assi digitali è demandato al dispositivo stesso e quindi non graverà sul tempo d'utilizzo della campionatura.
Per la campionatura è stato progettato un sistema "Real Time" Embedded con il sistema operativo Windows Microsoft, (Windows 98, Windows Me, Windows 2000) che permette di avere campionature su macchine con almeno 500Mhz di 100 microsecondi per 8 Assi o 500 microsecondi per 32 assi o 1 millisecondo per 64 assi, in tutti i casi i tempi sono per assi analogici quindi il caso peggiore.

Schema funzionamento campionatura



Un frequenza maggiore della campionatura, permette di avere una **granularità migliore** non solo per il PID (D2,Ie), ma anche nelle traiettorie non lineari (circolari , Spline , assi virtuali , ecc).Che vengono calcolate nel tempo effettivo di campionatura.(Real Time).

OEM Cabability

Il AxesBrainOpenTemplate permette l'integrazione di funzionalità specifiche del Cliente, in modo che è possibile aggiungere nuove istruzioni, nuovi cicli di lavorazione con i rispettivi comandi, queste funzionalità sono integrate come un specifico DLL di proprietà del Cliente che integrano a quelle del sistema base . Il AxesBrainOpenTemplete mette a disposizioni delle classi specifiche per poter integrare come DLL, le funzionalità del Cliente.

OpenOEMDll
CloseOEMDll

Classi Base

class COEMCabability
- Create(…)
- OnGetStatusOEM(..)
- OnCommandOEM(..)
- OnReadRunningProcessesOEM(..)
- Close(..)

class CRegistryOEM

Metodi :
- Create(..)
- ReadRegistry(int iKindRegistry, int iStartElement, int iNumberElements, Variat)
- WriteRegistry(int iKindRegistry, int iStartElement, int iNumberElements, Variat)
- Close(..)

class CArrayOEM
- Create(..)
- ReadArray(int iKindArray, int iNumArray, iStartElement, iEndElements,Variat)
- WriteArray(int iKindArray, int iNumArray, iStartElement, iEndElements,Variat)
- Close(..)

class CProcessOEM
- Create(..)
- GetStatusProcess(..)
- CommandProcess(..)
- GetParametersProcess(..)
- SetParametersProcess(..)
- ExecProcess(..)
- KillProcess (..)
- SendProcessCompleted (..)
- SendOrClearSubSystemEventSignal (..)
- WriteServiceParametersAndContinue (..)
- WaitSignal (..)
- WaitServiceParameters (..)
- Close(..)

class CInstructionOEM

Create
ExecInstruction( )
KillInstruction( )
SendInstructionCompleted ( )
SendOrClearSubSystemEventSignal (..)
WriteServiceParametersAndContinue (..)
WaitSignal (..)
WaitServiceParameters (..)
Close

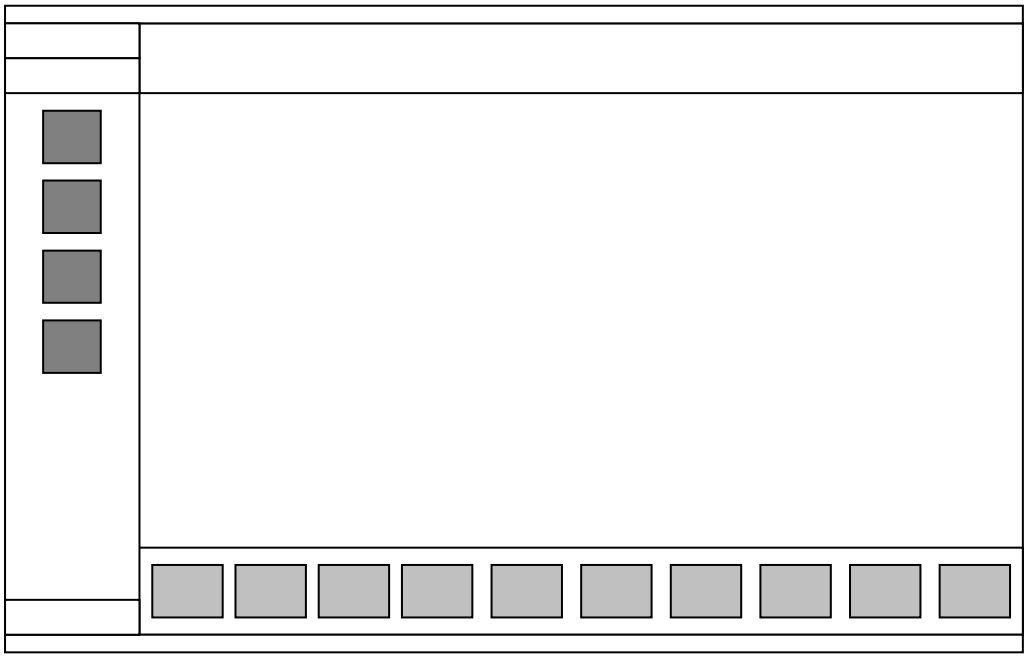
Interfaccia utente

Il "AxesBrainStudio" è un contenitore di applicativi che permettono all'utente d'interfacciare il sistema di movimentazione.

Gli applicativi si suddividono in applicativi base e applicativi utenti, quelli base permettono di condurre l'impianto, di programmare ed editare i cicli di lavorazione e manipolazione, monitorare le risorse controllate, configurare l'impianto, effettuare la sua manutenzione , in sintesi fornire tutte gli strumenti necessari all'utente per gestire l'impianto o la macchina operatrice, gli applicativi utente permettono invece di personalizzare tali strumenti in modo da ritagliare l'interfaccia utente alle esigenze specifiche di ogni singolo sistema.

Gli applicativi utenti sono sviluppabile con la maggior parte dei linguaggi di programmazione (Visual Basic ,Visual C++ ,Java, ecc.), inoltre possono essere anche progettate come pagine di un documento HTML.

Le applicazioni sono raccolte in "Ambienti" , in modo che ognuno di loro diventa una "Vista" o "Pagina" del "Ambiente", rimane così facilitata la navigazione da parte dell'utente.



Il "AxesBrainStudio" è stato progettato in modo da poter dare un **"scalabilità**" del sistema di movimentazione **in termini di interfaccia utente** .

Integrazione con la visione

Il "AxesBrain" ha sviluppato una modalità d'integrazioni con applicativi diversi sia residenti sullo stesso PC o remotati, integrazione a servizi esterni che però sono trasparenti all'interfaccia utente, in modo da dare una omogeneità di tutto il sistema.

Di particolare interesse è l'integrazione con applicativi che utilizzano la libreria "VisAlgo".

La libreria "VisAlgo" fornisce dei moduli per l'analisi d'immagine ottenute tramite "frame grabber" commerciali.

Le funzionalità sono essenzialmente per due tipologie di esigenze: il riconoscimento per la manipolazione di parti e la misurazione di particolari.

L'integrazione avviene ho tramite istruzioni del linguaggio VisAlgoL, oppure tramite funzione di richiesta servizi.(Via utilizzata dai moduli OEM).

La medesima strada è comunque utilizzabile per qualsiasi tipologia di programmi, che sviluppano dei servizi che possano essere utilizzabili dal sistema di movimentazione.

Capitolo

2

Servizi o metodi DCOM del AxesBrainServer

I servizi chiamati anche "Metodi"sono gli strumenti con i quali gli utenti gestiscono la movimentazione della macchina operatrice o dell'impianto di produzione .

Ai metodi o servizi del DCOM "VisAlgoServer" si accede tramite un set di funzioni che si suddividono a seconda dei compiti richiesti in:

- Funzioni sistema base
- Funzioni assi e mandrini
- Funzioni per i segnali
- Funzioni dei sotto sistemi
- Funzioni per le segnalazioni d'impianto
- Funzioni per la configurazione del sistema

A sua volta il DCOM "AxesBrainServer" invia degli eventi per:

- segnalare anomalie
- avvertire che il servizio a lui richiesto è stato completato
- richiedere a sua volta dei servizi da parte degli utenti ad esso collegati

Funzioni sistema base

Per poter gestire le risorse di sistema nella loro globalità con i cicli di lavorazione è necessario avere delle funzioni per ottenere lo stato, dare o togliere potenza agli assi, cancellare i cicli di lavorazione n esecuzione sugli applicativi utente o programmati con i rispettivi linguaggi

1-GetInformations	Legge le informazioni del sistema
2-GetSystemStatus	Legge lo stato del sistema
3-SystemCommand	Invia un comando per accendere la potenza agli assi o toglierla o azzera le richieste di funzionamento sia per i cicli d'automazione che per i cicli di CNC che per i comandi assi in corso

Funzioni assi e mandrini

Per poter gestire le risorse di assi e mandrini abbiamo funzioni che lavorano direttamente sulle risorse stesse riferite tramite il loro nome. Per poter lanciare dei movimenti coordinati con sistemi di riferimento tra loro omogenei e con traiettorie continue o "SPLINE" è necessario aprire degli HANDLE di gestione.

4-ReadAxesRegister	Legge il valore di un registro dell'asse
5-WriteAxesRegister	Scrive un valore di un registro su un asse
6-StartAxesBuffer	Inizia la bufferizzazione dei dati di un asse
7-ReadAxesBuffer	Legge i dati bufferrizzati dell' asse
8-StopAxesBuffer	Ferma la bufferizzazione dei dati di un asse
9-SendCommandAxes	Invia un comando ad un asse
10-ChangeOnFlyDinamicParameterAxes	Cambia al volo i parametri dinamici di un asse
11-OpenHandleAxis	Apertura di un HANDLE di gestione assi
12-CommandMotionAxis	Comando di movimento sugli assi
13-CloseHandleAxis	Chiusura di un HANDLE di gestione assi
14-ChangeDinamicParameterSystem	Cambia i parametri dinamici del sistema

Funzioni per i segnali

Per poter gestire le risorse di Input e di Output digitali o analogiche abbiamo delle funzione che fanno riferimento agli I/O tramite il loro nome.

15-ReadIO	Legge il valore del I/O
16-WriteIO	Scrive il valore di un I/O
17-WaitIO	Attendi il valore di un I/O

Funzioni dei sottosistemi

Funzioni per poter gestire le risorse di sistema nella loro globalità con i cicli di lavorazione e manipolazione programmati con i linguaggi AxesBrainL,AxesBrainISO,AxesBrainAWL e OEM Cabability , che permette la contemporaneità di più attività o cicli, adatto per la sua flessibilità all'automazione in senso lato.

18-GetSubSystemStatus	Legge lo stato di un sotto sistema
19-SubSystemCommand	Invia un comando al sotto sistema
20-GetSubSystemRegistry	Legge i registri delle sotto sistema
21-SetSubSystemRegistry	Scrive i registri di un sotto sistema
22-GetSubSystemArray	Legge le tabelle del sotto sistema
23-SetSubSystemArray	Scrive le tabelle del sotto sistema
24-ExecSubSystemProcess	Richiede l'esecuzione di un processo
25-KillSubSystemProcess	Sospende e cancella l'esecuzione di un processo
26-GetSubSystemRunningProcesses	Legge i processi in esecuzione nel sotto sistema
27-GetSubSystemProcessStatusRegistry	Legge i registri di stato di un processo
28-SetSubSystemProcessStatusRegistry	Scrive i registri di stato di un processo
29-GetSubSystemProcessParameters	Legge i parametri di un processo
30-SetSubSystemProcessParameters	Scrive i parametri di un processo
31-ExecSubSystemInstruction	Richiede l'esecuzione di una istruzione
32-KillSubSystemInstruction	Sospende e cancella l'esecuzione di una istruzione
33-SendOrClearSubSystemEventSignal	Usata come sincronizzazione semplice con il sottosistema segnalando un evento, se il processo del sottosistema era in attesa dell'evento viene riattivato, altrimenti l'evento viene memorizzato.
34-WriteServiceParametersAndContinue	Il processo del sotto sistema si pone in attesa inviando una richiesta si servizio, sarà appunto il l'esecutore del servizio a chiamare il metodo: "WriteServiceParametersAndContinue"

Note:

OnRequestServiceEvent	Richiede un servizio agli utenti connessi, l'applicazione dell'utente predisposto ad eseguire la richiesta, una volta che ha terminato il compito deve segnalare tramite la funzione "WriteServiceParametersAndContinue" che l'operazione è stata eseguita.

Gli eventi di richiesta servizi sono:

WAIT_SERVICE	Richiesta di un servizio generico
WAIT_KEYBOARD_SERVICE	Richiesta di un input da tastiera
WAIT_GVS_FUNCTION	Richiesta di una funzione "VisAlgo"
WAIT_GVS_PROCEDURE	Richiesta di una procedura "VisAlgo"

Funzioni per le segnalazioni d'impianto

Quando il sistema AxesBrain rileva anomalie, stati d'errore o segnalazioni le registra in un file di LOG e le segnale per mezzo di un FIRE apposito, le applicazioni possono catturare queste segnalazioni per poterle a loro volta segnalare in modo grafico.

35-GetEventsLogFile	Legge le segnalazioni del LOGFILE
36-DeleteEventsLogFile	Cancella le segnalazioni dal LOGFILE

Note:

OnSystemEvent	Evento inviato da parte del sistema per segnalare anomalie

Funzioni per la configurazione del sistema

La configurazione delle risorse, dei dispositivi di controllo ed altre caratteristiche di sistema sono registrate nel "AxesBrain", apposite funzione ne permettono la lettura, il cambiamento, la cancellazione e l'aggiunta di nuove configurazioni.

37-ReadSystemConfiguration	Legge la configurazione di sistema
38-EditSystemConfiguration	Scrive la configurazione di sistema
39-GetNumberAndNameKindElement	Legge il numero e i nomi di un tipo di elementi
40-ReadElementConfiguration	Legge la configurazione di un elemento
41-EditElementConfiguration	Cambia,cancella o aggiunge la configurazione di un elemento

Eventi dal Server

Il server invia in modo asincrono, delle segnalazioni agli utenti connessi attraverso degli "EVENTI". Gli eventi si suddividono in tre tipologie:

- Eventi di segnalazione di anomalie del sistema
- Eventi di segnalazione di richieste asincrone espletate
- Eventi di richieste da parte del Server, per ottenere dei servizi dai Client

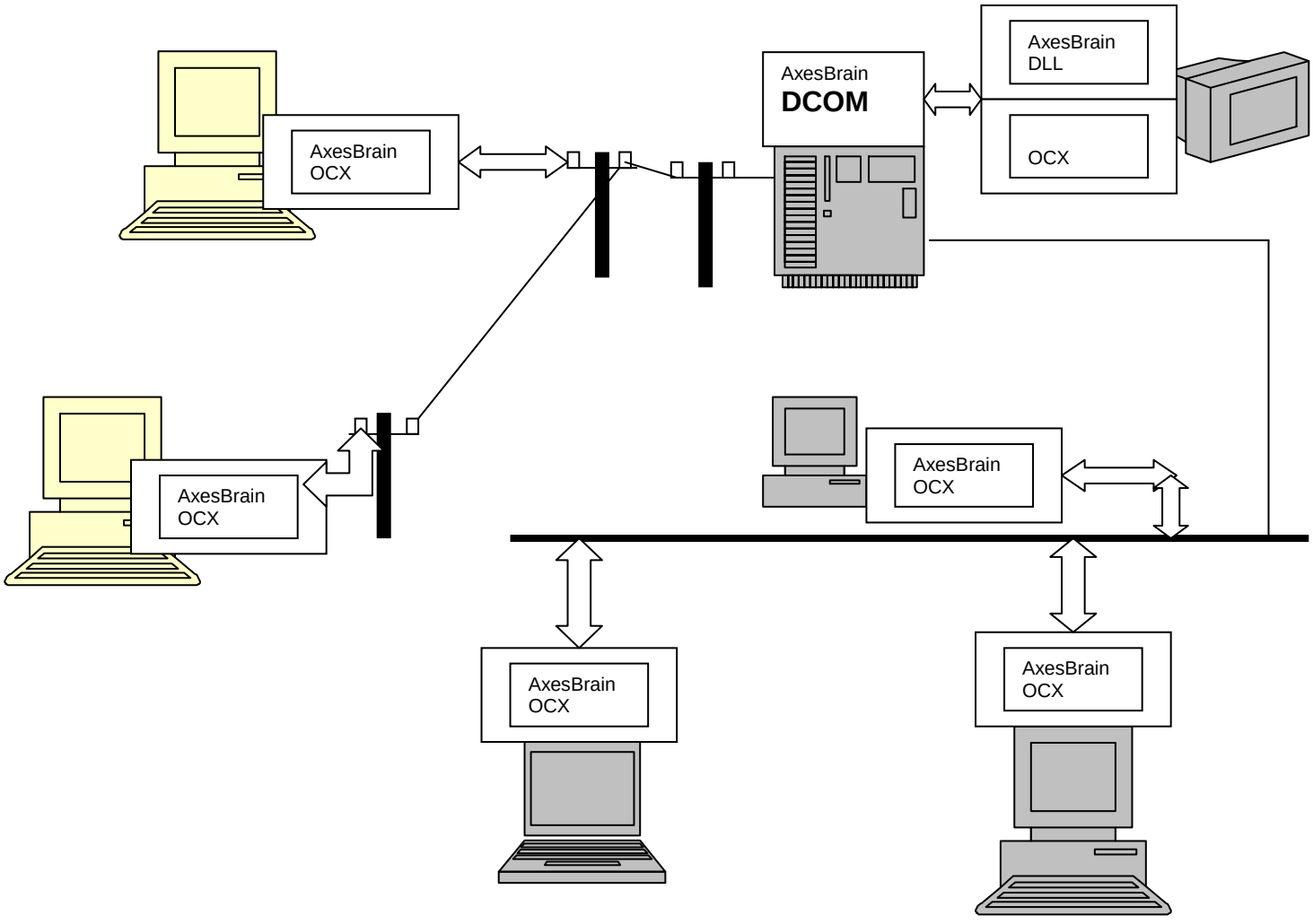
OnSystemEvent	Segnala anomalia sul sistema
OnCompleteCommand	Segnala il completamento di una funzione asincrona (con ID dato all'atto della richiesta)
OnRequestServiceEvent	Richiede un servizio dal applicativo utente in termini di dati, che quando reperiti verranno forniti tramite l'opportuna funzione di scrittura.

Accesso con AxesBrainBase (Libreria DLL)

e con AxesBrainCom (Libreria OCX)

Per permettere una interfaccia facilitata agli applicativi utente sono state sviluppate due librerie di accesso alle funzionalità del "AxesBrainServer" DCOM, una prima libreria di tipo DLL utilizzabile dalle applicazioni scritte in linguaggio Visual C++ o Visual Basic, una seconda libreria di tipo OCX per tutti i linguaggi di programmazione.
Per richiamare le funzioni dalla libreria "AxesBrainBase" (DLL) bisogna anticipare il prefisso Bs_ alle funzioni del "AxesBrainServer" DCOM.
Per richiamare le funzioni dalla libreria "AxesBrainCom" (OCX) bisogna anticipare il prefisso Cx_ alle funzioni del "AxesBrainServer "DCOM.

La tecnologia DCOM che è stata utilizzata per realizzare "AxesBrain", permette di far utilizzare le proprie funzionalità anche dagli applicativi che lavorano nel mondo di INTERNET, quindi sviluppati con il linguaggio HTML, l'uso della libreria OCX comunque ne facilita il compito.



Capitolo

3

Automazione (AxesBrainL)

Il linguaggio AxesBrainL è un linguaggio proprietario per la gestione dei cicli di automazione generici .

Nato dalla specifica del SIGLA (SIGMA LANGUAGE) 1976 della Olivetti , come uno dei primi linguaggi di programmazione delle macchine speciali per l'assemblaggio di parti, è stato ampliato in modo da soddisfare le moderne esigenze d'integrazione e flessibilità. Riferimento a :"Robot technology at Olivetti: the sigma system" Olivetti,Milan 1976.

Una sua prerogativa oltre la semplicità di sintassi è la capacità di avere la multiprogrammazione dei singoli cicli, prerogativa indispensabile per poter effettuare delle operazioni di assemblaggio e di manipolazioni di parti.

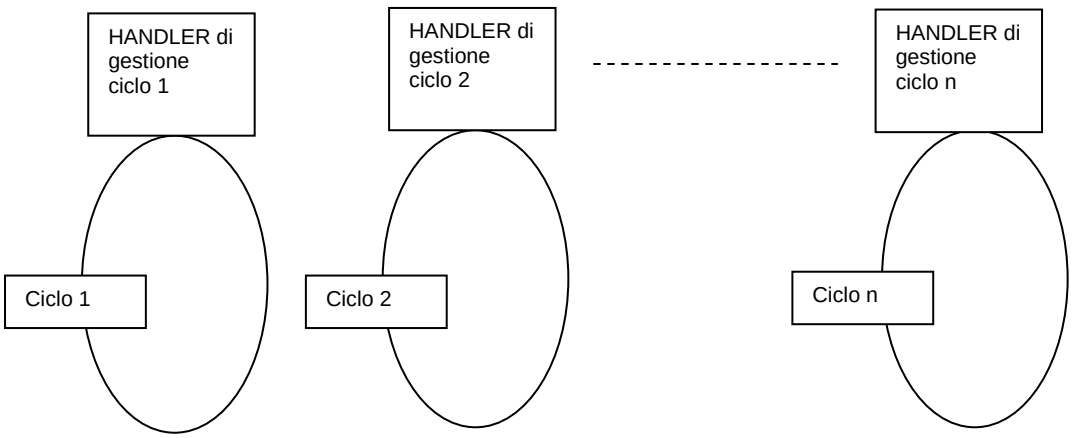
Architettura

L'architettura del sotto sistema di automazione è quella di poter eseguire i cicli di lavoro in parallelo tra loro sincronizzandoli con degli eventi.

Ogni programma ha al proprio interno un area proprietaria dove sono allocate le variabili LOCALI ,su cui tra l'altro vengono depositati i parametri di chiamata.

Quando al programma viene inserito in esecuzione, da un comando esplicito o richiamato da un CALL o da un'istruzione TSK, il suo codice sorgente viene caricato in memoria ed automaticamente precompilato, in memoria vi rimarrà fino ad un comando esplicito di CLOSE o comando al sottosistema di RESET.

I programmi vengono abbinati ad un HANDLER di lavoro proprietario su cui si appenderanno tutte le attività del ciclo, possiamo aver un numero pressoché infinito di HANDLER, con questa modalità si possono avere configurazione d'impianto estremamente flessibili, come linee di ROBOTS, macchine speciali multi testa, macchine con carichi scarichi integrati , macchine multifunzionali, etc.



Sintassi del linguaggio

La sintassi del linguaggio è estremamente semplice un triletterale preceduto da un trattino "-" rappresenta l'istruzione, uno "/" rappresenta il delimitatore dai parametri dell'istruzione che possono essere:

1. Riferimenti diretti a variabili GLOBALI o LOCALI
2. Riferimenti indiretti a variabili GLOBALI
3. Espressioni numeriche con riferimenti alle variabili GLOBALI o LOCALI
4. Espressioni matematiche con riferimenti alle variabili GLOBALI o LOCALI
5. LABEL di salto
6. Nomi di risorse assi , mandrini, input e output
7. Stringhe alfanumeriche

I parametri sono separati da virgole.

Le etichette o LABEL per i salti condizionati o incondizionati sono messe prima del separatore trattino "-" dell'istruzione.

I commenti sono preceduti da carattere punto e virgola ";"

[label] –XXX/[parametro 1],…[parametro n] [; questo è un commento]

Esempio:

```
;  questo è un esempio di programmazione
        -LET/G1,0
qui     -JNE/G1,1,qui           ; attende che il valore della variabile GLOBALE G1 sia posto a 1
; da un altro ciclo in lavorazione parallela
```

Le espressioni matematiche hanno le seguenti funzioni:

abs	Assoluto di un numero
acos	Arco coseno
and	And tra due numeri
asin	Arco seno
atan	Arco tangente
atanw	Arco tangente di Y,X
ceil	Arrotondamento verso l'alto di un numero decimale in un numero intero
cos	Coseno
cosh	Coseno iperbolico
deg	Trasformazione in gradi di un angolo espresso in radianti
exp	Esponenziale
floor	Arrotondamento verso il basso di un numero decimale in un numero intero
logd	Logaritmo decimale
logn	Logaritmo naturale
lshift	Shift verso sinistra di un numero
max	Massimo fra di enne espressioni
min	Minimo fra enne espressioni
mod	Modulo fra due numeri
not	Negazione boleana di un numero
or	Or fra due numeri
pi	P greco
rad	Trasformazione in radianti di un angolo espresso in gradi
rshift	Shift verso destra di un numero
sin	Seno
sinh	Seno iperbolico
sqr	Radice quadrata
tan	Tangente
tanh	Tangente iperbolica
xor	Or esclusivo fra due numeri

Gli operatori in una espressione sono:

+	somma
-	sottrazione
/	divisione
*	moltiplicazione
^	elevato
(	parentesi aperta
)	parentesi chiusa

Esempio di espressione matematica

```
        -LET/L1, MAX(SIN(RAD(G1+12),COS(RAD(G1+12*L1/56))
 ; nella variabile LOCALE L1 viene caricato il risultato della espressione:
;                MAX(SIN(RAD(G1+12),COS(RAD(G1+12*L1/56))
```

Variabili globali e locali

Per poter effettuare delle operazioni logiche, leggere valori numerici sono necessarie le variabili.

Il sotto sistema AxesBrainL prevede due tipi di variabili :

LOCALI

GLOBALI

Ogni programma all'atto dell'entrata in funzione si alloca un numero di variabili pari a quello configurate nel sistema, vengono tutte azzerate e sono a disposizione delle istruzioni di quel programma, le prime variabili vengono impostate con i parametri di chiamata del comando di esecuzione, le variabile rimarranno in memoria a disposizione per le operazioni di interrogazioni o visualizzazione

Un caso particolare sono i programmi richiamati dalle istruzioni di "-CAL" e di "-TSK" con parametri, in questi casi le prime variabili LACALI sono caricate con i parametri di chiamate nello stesso ordine posizionale, se esistono delle LOCALI nei parametri di chiamate verranno caricate con il nuovo valore al ritorno del programma chiamato.

Esempio:

-CAL/,routinemia,L9,12,23,G1+89,L7

quando avviene il ritorno "routinemia " L9 ed L7 avranno il valore definito nella routine, in questo caso

L9 avrà il valore di 11

L7 avrà il valore di 3

Infatti il codice di "routinemia" è il seguente:

-LET/L1,11

-LET/L5,3

All'atto del esecuzione le variabili LOCALI di "routinemia" sono:

L1=valore di L9 del programma chiamante

L2= 12

L3=23

L4=valore della variabile G1 + 89

L5=valore di L7 del programma chiamante

Oltre le variabili LOCALI nel sotto sistema di automazione sono previste 32767 variabili GLOBALI che vengono richiamate con la lettera "G" ed il numero della stessa.

Le variabili indirette rappresentate dalla lettera "I" seguita dal numero indica che il valore che viene trattato è quella della variabile GLOBALE il cui numero è quello della variabile LOCALE indicata dalla lettera "I".

Esempio:

-LET/L12,100

-LET/I12,120.56

In questo caso il valore 120.56 è caricato nella variabile GLOABLE 100, infatti il valore 100 indicante la GLOBALE è stato caricato nella LOCALE 12 , di cui si fa riferimento nella istruzione "-SET/I12,120.56".

Multitask e anticollisione

Una caratteristica importante nell'automazione è il poter effettuare più operazioni insieme, coordinale tra loro o meno, quindi abbiamo la necessità di avere la funzionalità di "MULTITASK".

Un ciclo di attività può essere eseguito con un comando esplicito, oppure da una istruzione "-TSK", il ciclo o programma viene abbinato ad un HANDLER di lavoro proprietario su cui si appenderanno tutte le attività del ciclo, possiamo aver un numero pressoché infinito di HANDLER.

Un "TASK" può essere cancellato da parte di un altro task o da se stesso con l'istruzione "-TKM" oppure quando viene eseguito il comando di RESET per il sotto sistema di automazione.

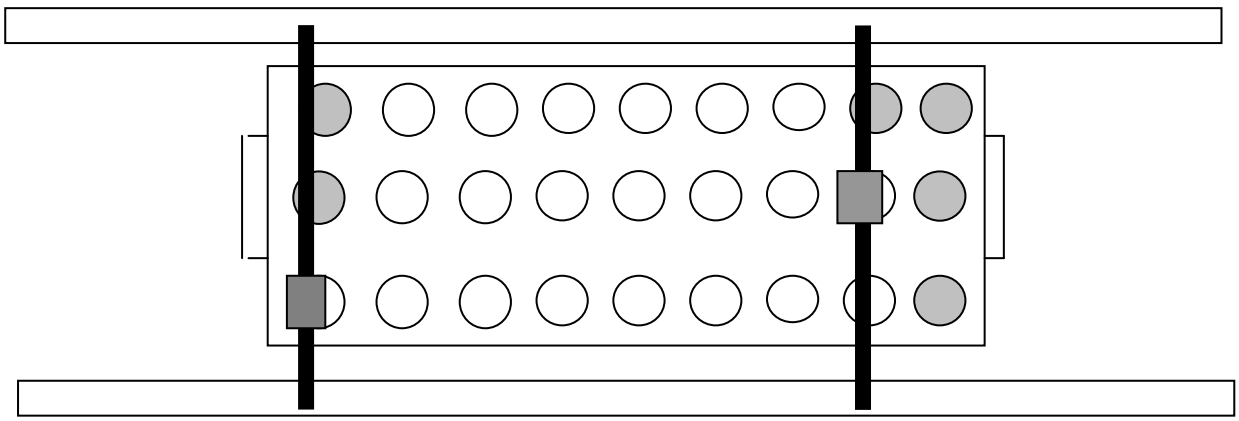
Un task può essere sospeso, riattivato dalla istruzione "-TKM".

Il concetto di HANDLER è utile inoltre vederlo come un canale su cui vengono effettuate operazioni di inizio della movimentazione in continuo , aggregazioni di movimento , attesa che i movimenti siano completati.

Con questa architettura noi siamo in grado di vedere il sistema di movimentazione come una apparecchiatura con numerosi "Bracci" che lavorano insieme coordinati o meno, aggregando dinamicamente gruppi di assi,.

Un esempio interessante può essere immaginare lo riempimento di un vassoio di bicchieri, in una fase iniziale abbiamo due "Bracci" che separatamente riempiono i calici, il sistema gestisce l'anticollisione e abbiamo così due gruppi di assi che lavorano separatamente, quando i calici sono stati riempiti il sistema, raggruppando i due bracci come farebbe un cameriere, porta il vassoio in zona di scarico.

Come prima evidenziato durante il riempimenti dei calici , i due bracci incidendo su uno stesso asse fisico X, grazie alla gestione dell'anticollisione è possibile programmare due cicli indipendenti di riempimento, sincronizzare i due a completamento delle loro rispettive fasi , quindi programmare un unico ciclo di scarico del vassoio con un raggruppamento unico dei due bracci.



Istruzioni logiche matematiche

1-LET	Imposta ad una Variabile il valore dell'espressione (LET)
2-ADD	Somma ad una Variabile il valore dell'espressione (ADDed)
3-SUB	Sottrai ad una Variabile il valore dell'espressione (SUBtract)
4-MUL	Moltiplica una Variabile con il valore dell'espressione (MULtiply)
5-DIV	Dividi una Variabile con il valore dell'espressione (DIVided)
6-NEG	Nega il valore di una Variabile (NEGation)
7-LBF	Imposta ad un vettore di Variabili il valore dell'espressione (Load BuFfer)

Istruzioni di controllo

8-JMP	Salta incondizionatamente ad una Label (JuMP)
9-JEQ	Salta ad una Label se le due espressioni sono uguali (Jump if Equal)
10-JNE	Salta ad una Label se le due espressioni non sono uguali (Jump in Not Equal)
11-JLT	Salta ad una Label se il valore del primo parametro è minore del secondo (Jump if Less Then)
12-JLE	Salta ad una Label se il valore del primo parametro è minore o uguale del secondo (Jump if Less then and Equal)
13-JGT	Salta ad una Label se il valore del primo parametro è maggiore del secondo (Jump if Great Then)
14-JGE	Salta ad una Label se il valore del primo parametro è maggiore o uguale del secondo (Jump if Great then and Equal)
15-JRN	Salta se il valore del parametro è compreso nel range (Jump if RaNge)
16-JNR	Salta se il valore del parametro è fuori del range(Jump if Not Range)
17-JOS	Salta se almeno un bit del valore del parametro è uno (Jump Or if bit Set)
18-JOC	Salta se almeno un bit del valore del parametro è zero (Jump Or if bit Clear)
19-JAS	Salta se tutti i bit del valore del parametro sono uno (Jump And if bit Set)
20-JAC	Salta se tutti i bit del valore del parametro sono zero (Jump And if bit Clear)
21-CAL	Chiama un part program passandogli dei parametri (CALl)
22-RET	Ritorna al chiamante del programma (RETurn)
23-TSK	Esegue in parallelo un ciclo di lavoro (TaSK)
24-TKM	Sospende, ripristina e cancella un ciclo di lavoro (TasK Manegement)

Istruzioni di movimentazione

25-HOM	Origine di un asse (HOMing)
26-MOV	Movimento interpolato linearmente di un gruppo di assi (MOVe)
27-MOC	Movimento interpolato linearmente di un gruppo di assi con condizione di start (MOve Conditional)
28-CIR	Movimento interpolato circolare o ellittico in senso orario di un gruppo di assi (CIrcular Right)
29-CIL	Movimento interpolato circolare o ellittico in senso antiorario di un gruppo di assi (CIrcular Left)
30-CRR	Movimento interpolato circolare o ellittico in senso orario di un gruppo di assi con raggio noto (Circular Radius Right)
31-CRL	Movimento interpolato circolare o ellittico in senso antiorario di un gruppo di assi con raggio noto (Circular Radius Left)
32-STC	Inizio di una movimentazione in continuo con definizione di percorso (STart Continuous)
33-HLC	Attesa del completamento della movimentazione in continuo (HaLt Continuous)
34-ABC	Cancellazione del movimento in continuo (Abort Continuous)
35-CAP	Cambia i parametri asse (Change Axis Parameter)
36-HMS	Gestione Master Slave (Handling Master Slave)
37-HEC	Gestione delle camme (Handling Electronic Cam)
38-CFR	Cambia i parametri dinamici di un asse (Change Feed Rate)
39-CPL	Cambia il loop di posizione (Change Position Loop)
40-PRD	Legge le posizioni dell'asse (Position ReaD)
41-RAV	Legge i parametri dell'asse (Read Axis Value)
42-RSV	Legge la velocità di un mandrino (Read Speed Value)
43-SFP	Imposta la velocità del profilo di movimentazione (Set Feed Profile)
44-SPD	Imposta la velocità di rotazione di un mandrino (SPeeD)
45-TCH	Movimento con tostatura (TouCH)
46-TMT	Movimento con ricerca valore di segnale analogico (Test Movement Trasducer)
47-TMS	Movimento con ricerca valore di sensore digitale (Test Movement Sensor)
48-TPE	Abilita il tastatore (Touch Probe Enable)
49-SZP	Definisci la posizione di un set zeri macchina (Set Zero Point)
50-LZP	Attiva un set di zeri macchina (Load Zero Point)
51-PIN	Flag di incrementale su un asse (Position INcremental)
52-PAB	Flag di assoluto su un asse (Position ABsolute)
53-ANT	Gestione dell' anticollisione(ANTicollision)

Istruzioni per i segnali di input ed output

54-WDI	Attende che un segnale d'ingresso digitale si porti ad un determinato stato (Wait Digital Input)
55-WAI	Attende che un segnale d'ingresso analogico si porti ad un determinato stato (Wait Analog Input)
56-TDI	Effettua un'operazione di test su un segnale d'ingresso digitale (Test Digital Input)
57-TAI	Effettua un'operazione di test su un segnale d'ingresso analogico (Test Analog Input)
58-SDO	Setta o resetta un segnale d'uscita digitale (Set Digital Output)
59-SAO	Scrive il valore di un segnale analogico d'uscita (Set Analog Output)
60-GDI	Legge il valore di un segnale digitale d'ingresso (Get Digital Input)
61-GAI	Legge il valore di un segnale analogico d'ingresso (Get Analog Input)
62-WBD	Scrive un blocco di segnali digitali in uscita (Write Buffer Digital output)
63-RBD	Legge un blocco di segnali digitali in ingresso (Read Buffer Digital input)

Istruzioni di sincronizzazione

64-EVS	Setta degli eventi di sincronizzazione (EVent Set)
65-EVC	Resetta degli eventi di sincronizzazione (EVent Clear)
66-EVW	Attende alcuni eventi di sincronizzazione (EVent Wait)
67-OCS	Apre la sessione critica (Open Critical Section)
68-CCS	Chiudi la sessione critica (Close Critical Section)
69-SCD	Disabilità la schedulazione dei part program GMSL (SCheduler Disable)
70-SCE	Abilita la schedulazione dei part program GMSL (Close Critical Section)

71-FOC	azzera il contenuto un file o la crea se non esiste (File Open Create)
72-FWR	Scrive un record su file (File WRite)
73-FRD	Legge un record da file (File ReaD)
74-TIM	Temporizzatore in secondi (TIMe)
75-TMM	Temporizzatore in millesimi di secondo (TiMe Millisecond)
76-SWA	Inizializza un orologio (Start WAtch)
77-RWA	Legge un orologio (Read WAtch)
78-HWA	Ferma un orologio (Halt WAtch)
79-CWA	Continua un orologio (Continue WAtch)
80-KYB	Attende un'operazione da tastiera (KeYBoard)
81-DRT	Visualizza continuamente i valori di assi, globali, segnali (Display Real Time)
82-DIS	Visualizza una riga di messaggio (DISplay)
83-WND	Crea una finestra grafica (WiNDow)
84-DRW	Disegna degli elementi grafici (DRaW)
85-HLD	Manda il sistema nello stato di Cycle Stop (HoLD)
86-TEA	Autoapprendimento (TEAch)

87-ARI	Richiesta di esecuzione di una istruzione per l'ambiente specificato nel primo parametro e attesa della risposta (ritorno effettuato con la funzione di "WriteServiceParametersAndContinue" al sottosistema di automazione AxesBrainL) (Ambient Request Istruction)
88-AES	Invia degli eventi all'ambiente specificato nel primo parametro (Ambient Event Set)
89-AEC	Azzera gli eventi dall'ambiente specificato nel primo parametro (Ambient Event Clear)
90-AEW	Attendi un evento dall'ambiente specificato nel primo parametro(attesa terminata con la funzione di "SendOrClearSubSystemEventSignal" al sottosistema di automazione AxesBrainL) (Ambient Event Wait)

Gli ambienti sono:

1. Ambiente "CLIENT"
2. Ambiente o sotto sistema OEM
3. Ambiente o sotto sistema CNC AxesBrainISO
4. Ambiente o sotto sistema PLC AxesBrainAWL
5. Ambiente di analisi delle immagine VisAlgo

Capitolo

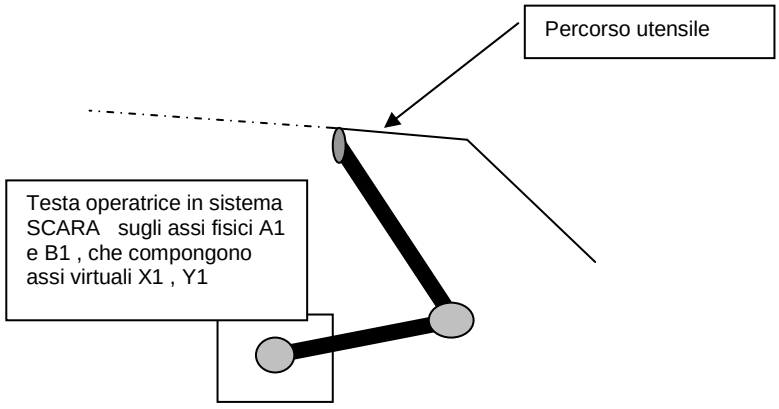
4

CNC (AxesBrainISO)

Il sottosistema CNC AxesBrainISO è in grado di controllare macchine utensili con una programmazione in linguaggio ISO, è un sistema multi testa che ha due funzionalità importanti: una è la capacità di moltiplicare la stessa lavorazione su più teste, oppure di eseguire lavorazioni su ogni testa.

Nel caso del sottosistema AxesBrainISO la macchina utensile è vista come composta da più di teste lavoranti separatamente o parallelamente, questo senza dimenticare che per i dispositivi ausiliari di carico e scarico li possiamo gestire in modo ancora separati dal sottosistema di automazione AxesBrainL.

Ad ogni testa lavorante è abbinato un gruppo di assi uno per ogni coordinata del sistema cartesiano, se gli assi fisici sono in un **sistema polare o "SCARA"** possiamo usare i loro rispettivi assi virtuali nel raggruppamento degli assi .



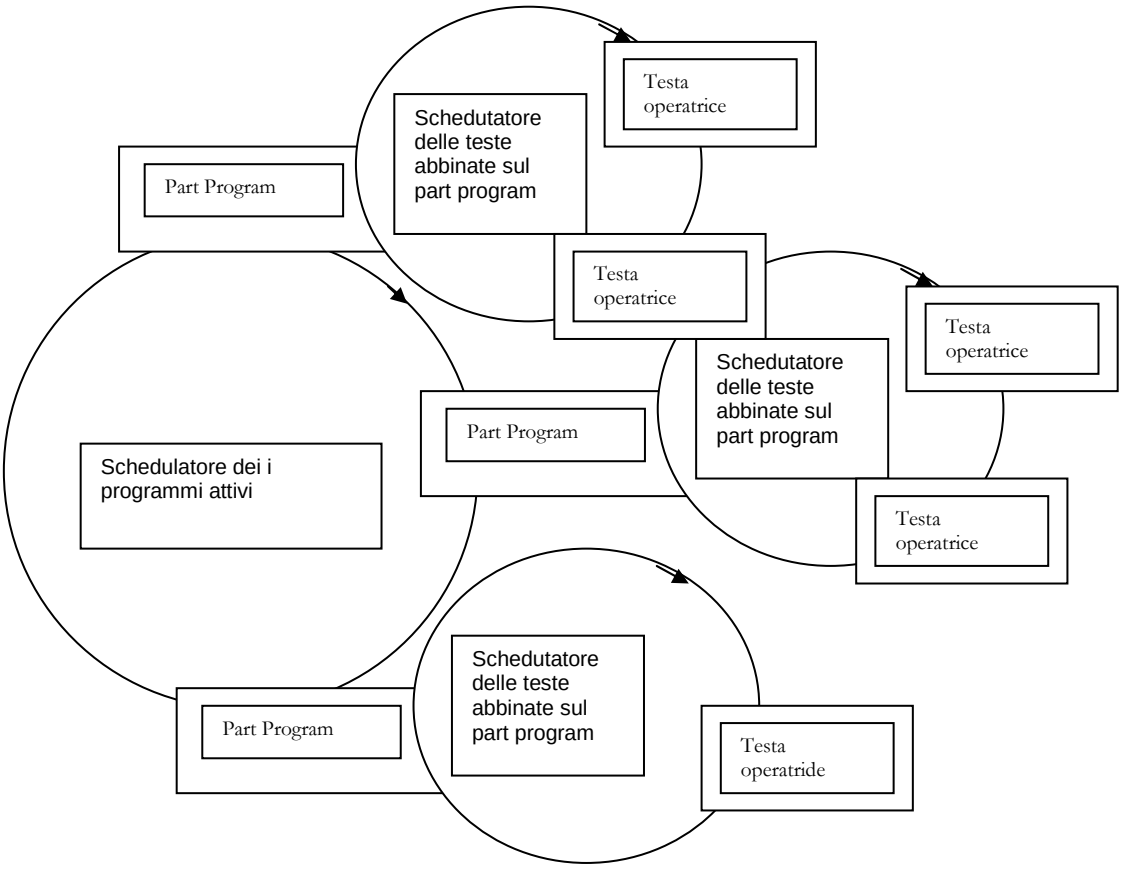
Architettura

Il sottosistema AxesBrainISO è in grado di gestire una macchina utensili multi testa, sia con lavorazione parallela sia lavorazioni indipendenti.

Ogni testa è vista come un abbinamento di assi fisici ad ogni sistema di coordinate rappresentante la testa operatrice o contesto CNC.

Il CNC è il contesto per l'esecuzione di part-program, procedure e blocchi di programmazione singoli.

Il sottosistema CNC schedula le attività richieste per ogni programma di lavorazione in formato ISO attivato a sua volta per ogni programma esiste un altro schedulatore che gestisce la realizzazione blocco per ogni testa abbinata al part program.



Regole di programmazione

Un programma di lavorazione è costituito da un insieme di blocchi che vengono eseguito in sequenza. Ciascun blocco corrisponde ad una sola linea ed è costituito da una o più funzioni, seguite dal loro valore.

Un blocco può contenere informazioni di commento al programma, che non hanno alcuna influenza sul programma stesso. In questo caso il testo deve essere preceduto dal carattere punto e virgola ";" .

Esempio

; inizio fresatura

N 023 X100.56 Y 41.9

Il carattere maggiore ">" all'inizio della riga indica un comando di controllo o di configurazione o di richiamo a procedure specifiche, come la configurazioni di assi, ripetizioni di blocchi ,etc.

La programmazione è a formato variabile. Questo significa che il numero di caratteri necessari per esprimere un valore non è fisso, quindi non è richiesto nessun tipo di incolonnamento. E' possibile introdurre degli spazi in ciascun punto del blocco, poiché vengono ignorati. Il punto decimale va introdotto solo per determinate funzioni. Se non viene indicato il segno, i valori sono considerati positivi, i valori negativi devono essere preceduti dal segno meno "-". Indipendentemente dalla posizione delle funzioni all'interno del blocco l'esecuzione di ciascuna avviene nel seguente ordine:

- funzione N
- funzione G
- funzioni M iniziali
- movimento assi
- funzioni M finali

All'interno di un blocco possono essere programmati più funzioni G e più funzioni M, ma solo una delle altre (X,Y,Z,T,F,ecc.)

Le funzioni G sono in diversi gruppi.

Non è possibile programmare in un blocco due o più funzioni G dello stesso gruppo.

Le funzioni M si dividono in funzioni iniziali, cioè eseguite prima del movimentassi e in funzioni finali, cioè eseguite al termine del movimentassi. Per ogni blocco di programmazione possono essere programmate due funzioni M iniziali e due funzioni M finali. Le funzioni M (miscellanee) servono generalmente per impartire alla macchina utensile dei comandi specifici riguardanti la predisposizione degli assi o del mandrino prima e dopo una certa operazione.

Abbinamenti assi fisici

Il sottosistema AxesBrainISO è in grado di gestire una macchina utensili multi testa, sia con lavorazione parallela sia lavorazioni indipendenti.

Per realizzare questa importante funzione si devono abbinare gli assi fisici o virtuali al sistema di coordinate cartesiane di ciascuna testa , questa operazione può essere fatta staticamente nella configurazione del sistema, oppure come set di comandi nel programma stesso.

Nel programma il comando:

> Start_heads_configuration

predispone il sistema ad accettare la configurazione vera e propria degli assi teorici che sono: X Y Z A B C U V W e il mandrino Spindle

quindi avremo comandi di assegnamento per la testa principale:

> Base_[nome asse teorico tra:X Y Z A B C U V W o Spindle] = [nome asse fisico o virtuale]

quindi avremo comandi di assegnamento per le teste aggiuntive:

> Head_[numero della testa aggiuntiva]_[nome asse teorico tra:X Y Z A B C U V W o Spindle] =

[nome asse fisico o virtuale]

Ala fine della fase di configurazione dovrà esserci il comando:

> End_configuration

Esempio:

> Start_heads_configuration
> Base_X=A1
> Base_Y=B1
> Base_Z=C1
> Base_Spindle=M1
> Head_2_X=A2
> Head_2_Y=B2
> Head_2_Z=C2
> Head_2_Spindle=M2
>End_configuration

Nell'esempio sopraccitato definiamo che la lavorazione sarà effettuata parallelamente da due teste.

Piano lavoro

I nomi degli assi macchina possono essere scelti tra i seguenti: X,Y,Z,A,B,C,U,V,W

Due qualsiasi di questi determinano l'ascissa e l'ordinata del piano di lavoro, su cui si applica la correzione raggio utensile nel piano, mentre un terzo assume la funzione di asse perpendicolare al piano di lavoro, asse a cui viene applicata la correzione lunghezza.

Gli eventuali altri assi sono correlati a questi, ma non sono interessati dai correttori raggio e lunghezza.

Piani di lavoro principali

Le funzioni G17,G18 e G19 specificano i tre piani principali predisposti dal sistema.

G17 Piano di lavoro XY, asse perpendicolare Z, l'ascissa del piano cartesiano è X, l'ordinata è Y

G18 Piano di lavoro ZX, asse perpendicolare Y, l'ascissa del piano cartesiano è Z, l'ordinata è X

G19 Piano di lavoro YZ, asse perpendicolare X, l'ascissa del piano cartesiano è Y, l'ordinata è Z

La programmazione dei pezzi in G18 ed in G19 va fatta ruotando il disegno in modo che l'asse dell'ascissa (Z in G18,Y in G19) diventi l'asse orizzontale con i valori positivi rivolti verso destra. Con questo accorgimento la programmazione risulta coerente con quella fatta in G17.

I valori delle ascisse a destra dello zero sono positivi, quelli a sinistra negativi. I valori delle ordinate sopra lo zero sono positivi, quelli sotto negativi.

Volendo definire piani di lavoro diversi da XY,ZX e YZ, si deve utilizzare la funzione G17 seguita dai nomi degli assi che nell'ordine saranno: ascissa, ordinata ed asse perpendicolare.

Ad esempio la frase G17 UVC assegnerà all'asse U il ruolo di ascissa, all'asse V il ruolo di ordinata ed all'asse C il ruolo di asse perpendicolare.

Le funzioni G17,G18,G19 vanno programmate in un blocco da sole.

Funzioni di movimento

Il movimento degli assi viene realizzato programmando il nome seguito dal valore della coordinata da raggiungere.

I nomi degli assi machina possono essere scelti tra i seguenti: X, Y, Z, A, B, C, U, V, W.

Assi di gruppi meccanici diversi possono avere lo stesso nome.

Il valore della coordinata può essere positivo o negativo (il segno + può non essere programmato); usare il punto decimale e non la virgola per separare la parte intera dai decimali.

Il movimento compiuto dagli assi lineari per raggiungere la posizione programmata dipende dal tipo di funzione preparatoria (G) usata. Se il blocco non contiene funzioni G, il movimento alla posizione programmata è eseguito in interpolazione lineare, cioè seguendo la linea retta che unisce il punto di partenza con il punto d'arrivo programmato. Le funzioni X,Y e Z sono modali e quindi non devono essere programmate nuovamente se la posizione dei rispettivi assi non cambia.

U,V,W sono assi aggiuntivi lineari e paralleli. Definiscono le quote alle quali dovranno posizionare gli assi aggiuntivi. Per convenzione le funzioni U,V,W indicano gli assi aggiuntivi lineari e gli assi paralleli a X, Y e Z. Per la programmazione delle funzioni U, V e W rimane valido ciò che è stato detto riguardo alle funzioni X, Y e Z.

A, B, C sono assi aggiuntivi rotativi. Definiscono le quote alle quali si dovranno posizionare gli assi aggiuntivi identificati con i nomi A, B, e C. Per convenzione le funzioni A, B e C indicano gli assi aggiuntivi rotanti rispettivamente attorno ad X, Y e Z. Le funzioni A, B e C sono modali e poiché esprimono quote angolari, vanno programmati in gradi.

Le funzioni di movimento sono: G00,G01,G02,G03

Funzioni G

Sono funzioni preparatorie che servono a predisporre il sistema o la macchina utensile alle operazioni successive. Esse sono composte dalla lettera G seguita da 2 (in questo ambiente di programmazione) o 3 cifre e possono essere valide nel blocco in cui sono programmate o fino a quando non sono cancellate da un'altra funzione. Quelle utilizzate in questo ambiente di programmazione sono elencate nel seguito:

G00	posizionamento rapido degli assi
G01	interpolazione lineare
G02	interpolazione circolare o elicoidale senso orario
G03	interpolazione circolare o elicoidale senso antiorario
G04	pausa temporizzata, tempo di pausa programmato con la funzione K in decimi sec.
G09	decelerazione alla fine del blocco che la contiene
G17	specifica XY come piano lavoro e Z asse perpendicolare
G18	specifica ZX come piano lavoro e Y asse perpendicolare
G19	specifica YZ come piano lavoro e X asse perpendicolare
G40	annulla G41 e G42
G41	attivazione correzione raggio, utensile a sinistra del profilo
G42	attivazione correzione raggio, utensile a destra del profilo
G48	richiamo correttore lunghezza
G54	memorizzazione origine pezzo
G55	attivazione origine pezzo
G70	programmazione in pollici
G71	programmazione in millimetri
G80	annullamento cicli fissi
G81	ciclo fisso per foratura
G82	ciclo fisso per lamatura
G83	ciclo fisso per foratura profonda
G84	ciclo fisso per maschiatura
G85	ciclo fisso per alesatura
G86	ciclo fisso per barenatura
G88	ciclo fisso GMSL
G89	ciclo fisso OEM
G90	programmazione assoluta
G91	programmazione incrementale
G94	avanzamento F in mm/min o pollici/min
G95	avanzamento F in mm/giro o pollici/giro

Funzioni M

Sono le cosiddette funzioni varie o miscellanee. Sono programmabili con la lettera M seguita da un numero di cifre compreso tra 2 e 4 (da M00 a M9999).

M00	arresto programmato
M03	rotazione oraria del mandrino
M04	rotazione antioraria del mandrino
M05	arresto del mandrino
M06	cambio utensile
M07	attivazione refrigerante secondario
M08	attivazione refrigerante primario
M09	disattivazione refrigerante
M10	attivazione bloccaggio assi
M11	disattivazione bloccaggio assi
M13	rotazione oraria mandrino e attivazione refrigerante
M14	rotazione antioraria mandrino e attivazione refrigerante
M19	orientamento mandrino
M30	fine programma,azzera le funzioni ausiliarie attive
M40 - M44	cambio gamma
M101 - M109	funzioni AxesBrainL
M111 - M119	funzioni OEM

Funzioni T

La funzione T serve per il cambio utensile, manuale o automatico. Le cifre che seguono la funzione T definiscono il numero dell'utensile da richiamare.

Oltre all'utensile la funzione T richiama tutti i parametri memorizzati nella tabella dei dati utensili (correttori lunghezza,raggio,ecc.)

Il modo di programmare il cambio utensile (manuale, automatico sequenziale o no, con o senza braccio scambiatore) dipende da come è stato realizzato dall'utente e da come è stata sviluppata la logica di macchina con AxesBrainL.

Per disattivare la correzione lunghezza attiva programmare G48 L0

Funzioni F, S e O

La funzione F definisce la velocità di avanzamento durante la lavorazione ed è programmata con la lettera F seguita da 3 cifre che esprime la velocità stessa. La funzione F è modale

Programmata dopo G94 definisce la velocità F in mm/min o pollici/min

Programmata dopo G95 definisce la velocità F in mm/giro o in pollici/giro

La funzione definisce la velocità di rotazione del mandrino. Si programma con la lettera S seguita da sei cifre che esprimono la velocità espressa direttamente in giri/min.

La funzione O richiama le origini in fase di azzeramento, da O1 a O99.

La funzione O0 disabilita le origini e le coordinate sono riferite allo zero macchina. La funzione O-1 richiama, dopo la programmazione di O0, l'ultima origine programmata.

Funzioni di controllo programma

Ripetizioni di parte di programma

Utilizzando i codici L è possibile ripetere n volte un programma o parte di esso. Il numero massimo di ripetizioni è 32767.

La parte di programma che si vuole ripetere è racchiusa fra una definizione di riferimento "label" e l'istruzione di salto alla label seguita dal numero di ripetizioni.

Il numero di ripetizioni può essere un numero o un parametro.

Esempio

…….
…….
L=12 definizione della label
…….
…….
L12 K8 salta alla label 12 per 8 volte

Sottoprogrammi interni al programma

Si intende per sottoprogramma una sequenza di blocchi che possono essere richiamati da punti diversi del programma principale (ad esempio la successione dei vari punti su cui applicare i diversi cicli fissi, foratura, carenatura, alesatura, ecc .) o un profilo da richiamare più volte in punti diversi o con correttori raggio diversi.

Il sottoprogramma è richiamato programmando la funzione L seguita dal numero del sottoprogramma.

I sottoprogrammi interni al programma principale vanno programmati alla fine dello stesso, dopo la funzione M30.

Devono iniziare con la funzione L = numero sottoprogramma e terminare con la funzione di ritorno G32.

Un programma può chiamare un altro e così via fino ad massimo di 8 livelli di annodamento.

Il numero massimo di label è 100 L0 a L99, da utilizzare sia per la ripetizione di parti del programma che per i sottoprogrammi interni.

Capitolo

5

PLC (AxesBrainAWL)

Il sottosistema PLC AxesBrainAWL scandisce ogni 10 millesimi di secondo un ciclo di programma PLC, l'utilizzo di una logica ciclica può essere utile utilizzarla per dare delle funzionalità all'impianto una capacità di risposta immediata a segnali di controllo.

Il linguaggio di programmazione del ciclo PLC è AWL , un set di istruzioni mnemoniche che rappresentano le funzioni del sotto sistema.

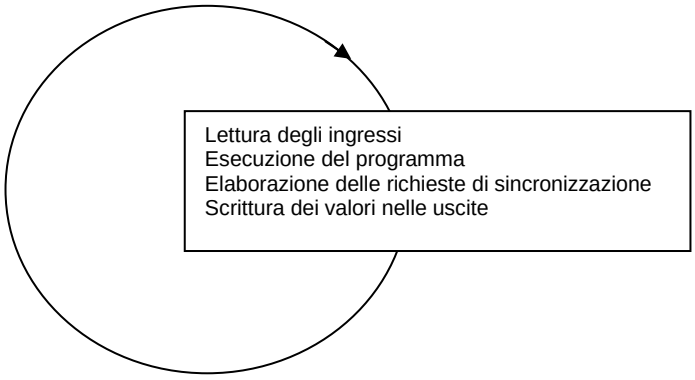
Alcune istruzioni di sincronismo possono essere utilizzate per cooperare con gli altri sottosistemi.

Il sotto sistema è in grado di eseguire una serie di operazioni, compreso il programma, in modo ciclico.

L'esecuzione ciclica delle operazioni viene definito ciclo di scansione. Durante il ciclo di scansione illustrato nella figura sottostante, il sotto sistema alcune o tutte le seguenti operazioni:

- lettura degli ingressi
- esecuzione del programma utente
- elaborazione delle richieste di sincronismo
- scrittura dei valori sulle uscite

Ciclo di scansione 10 millisecondi



Lettura degli ingressi

Ogni ciclo di scansione inizia leggendo il valore corrente degli ingressi digitali e scrivendo poi questi valori nel registro delle immagini di processo degli ingressi.

Nel sotto sistema sono previsti incrementi di otto bit (un byte) per il registro delle immagini di processo degli ingressi. Se le risorse non forniscono un ingresso fisico per ogni bit del byte riservato, non sarà possibile utilizzarli nel programma utente. Il sistema resetta a zero tali ingressi inutilizzati del registro delle immagini di processo all'inizio di ogni ciclo.

Il sistema non aggiorna automaticamente gli ingressi analogici come parte del ciclo di scansione e non memorizza un registro per le immagini di ingresso analogiche. Occorre quindi accedere agli ingressi analogici direttamente dal programma.

Esecuzione del programma

Durante la fase di esecuzione del ciclo di scansione , il sistema esegue il programma iniziando dalla prima operazione e procedendo verso l'operazione finale.

Le operazioni dirette I/O forniscono all'utente un accesso immediato a ingressi e uscite nel corso dell'esecuzione del programma o di una routine d'interrupt.

Se si utilizzano interrupt nel programma utente, le routine associate agli eventi di interrupt vengono memorizzate come parte del programma. Le routine di interrupt non vengono eseguite come parte normale del ciclo di scansione, bensì realizzate quando interviene un evento di interrupt (ciò può verificarsi ad ogni punto del ciclo di scansione).

Elaborazione delle richieste di sincronismo

Le richieste di eventi di sincronismo vengono espletate alla fine del ciclo, in modo da garantire che tutto il ciclo sia terminato e i segnali siano completamente elaborati.

Scrittura dei valori nelle uscite digitali

Al termine di ogni ciclo di scansione il sistema scrive nelle uscite digitali i valori memorizzati nel registro delle immagini di processo delle uscite.

Nel sistema sono previsti incrementi di otto bit (un byte) per il registro delle immagini di processo delle uscite.

Il sistema non aggiorna automaticamente le uscite analogiche come parte del ciclo di scansione e non memorizza un registro delle immagini delle uscite analogiche. Occorre quindi accedere alle uscite analogiche direttamente dal programma utente.

Interruzione del ciclo di scansione

Se si utilizzano interrupt, le routine associate ad ogni evento di interrupt vengono memorizzate come parte del programma. Le routine di interrupt non vengono eseguite come parte normale del ciclo di scansione, bensì realizzate quando interviene un evento di interrupt (ciò può verificarsi ad ogni punto del ciclo di scansione). Gli interrupt vengono elaborati uno dopo l'altro in base al livello di priorità

Registri delle immagini di processo degli ingressi e delle uscite

Solitamente è preferibile utilizzare i l registro delle immagini di processo piuttosto che accedere direttamente agli ingressi o alle uscite durante l'esecuzione de programma e questo per tre ragioni:

- Il caricamento di tutti gli ingressi alla sommità del ciclo sincronizza e congela i valori degli ingressi per la fase di esecuzione del programma all'interno del ciclo di scansione. Le uscite sono aggiornate dal registro delle immagini di processo ad esecuzione del programma completata. Ciò produce un effetto stabilizzante sul sistema.

- Il programma utente può accedere al registro delle immagini di processo molto più velocemente rispetto ai punti di ingresso e uscita, consentendo una maggiore rapidità anche nell'esecuzione del programma.

- I punti di ingresso e uscita sono entità bit, alle quali si accede solo nel formato binario; al registro delle immagini di processo si può invece accedere in bit, byte, parola e doppia parola.Grazie a ciò, i registri delle immagini di processo offrono una maggiore flessibilità.

Un beneficio ulteriore è rappresentato dal fatto che i registri delle immagini di processo sono sufficientemente grandi per gestire il numero massimo di ingressi e uscite. Poiché un sistema reale consiste sia di ingressi che di uscite; rimane sempre un certo numero di irizzi delle immagini di processo non utilizzate. Tali indirizzi possono essere usati come merker interni supplementari.

Controllo diretto ingressi e uscite

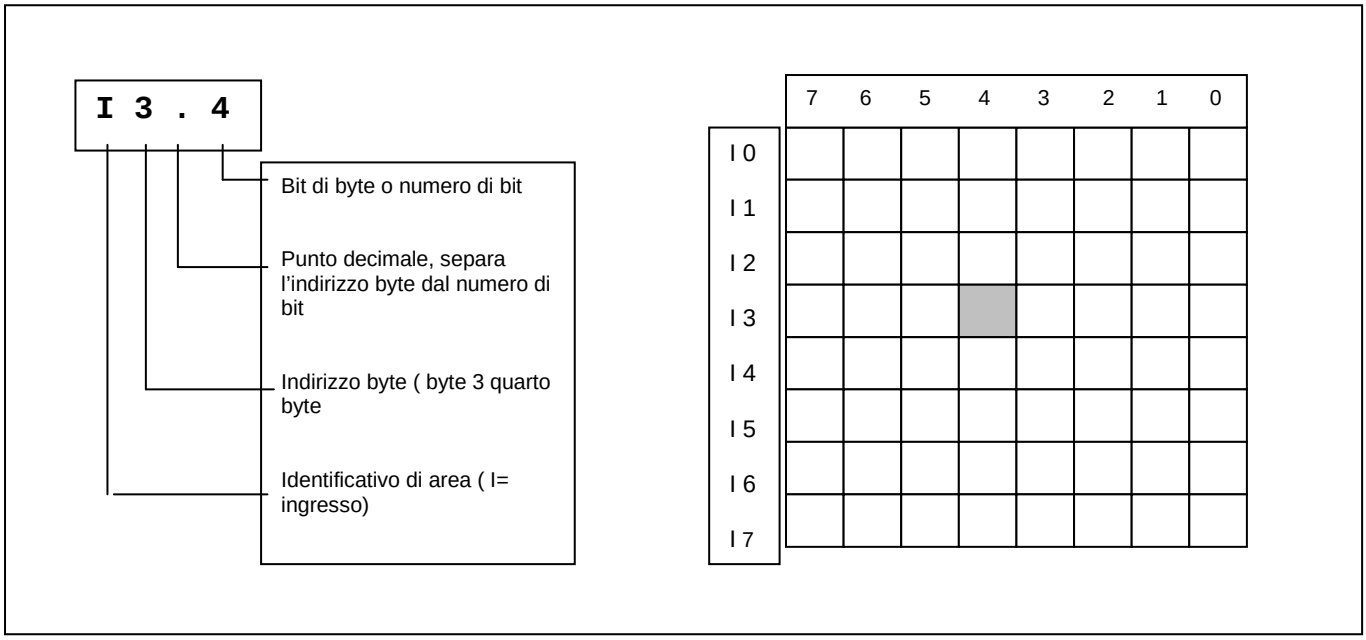
Le operazioni dirette con ingressi e uscite permettono un accesso immediato a ingressi e uscite effettive, malgrado i registri delle immagini di processo vengano normalmente adoperati come sorgente o destinazione per accessi I/O. Se si utilizza una operazione diretta per accedere ad un ingresso, l'indirizzo corrispondente del registro delle immagini di processo degli ingressi non viene modificato. L'indirizzo corrispondente del registro delle immagini di processo delle uscite viene invece aggiornato.

Tipi di dati e modi d' indirizzamento

Utilizzo dell'indirizzo di memoria per l'accesso ai dati

Per accedere ad un bit in un'area di memoria occorre specificare l'indirizzo,che comprende l'identificazione di area di memoria, l'indirizzo byte e il numero di bit.

La figura riporta un esempio di accesso ad un bit (definito anche indirizzamento "byte.bit"). Nel seguente esempio, l'area di memoria e l'indirizzo byte (I = ingresso, 3 = byte 3) sono seguiti da un punto decimale "." per distinguere l'indirizzo del bit.(bit 4).



Indirizzamento del registro delle immagini di processo degli ingressi (I)

Il sistema campiona i punti di ingresso fisici all'inizio di ogni ciclo di scansione e scrive questi valori nel registro delle immagini di processo degli ingressi. Si potrà accedere a tale registro in bit,byte parola e doppia parola.

Formato:	Bit	I[indirizzo byte].[indirizzo bit]	**I10.1**
	Byte	IB[indirizzo byte iniziale]	**IB4**
	Parola	IW][indirizzo byte iniziale]	**IW4**
	DoppiaParola	ID[indirizzo byte iniziale]	**ID4**

Indirizzamento del registro delle immagini di processo delle uscite (Q)

Alla fine del ciclo di scansione il sistema copia nelle uscite fisiche i valori memorizzati nell'immagine di processo delle uscite. Si potrà accedere a tale registro in bit,byte, parola e doppia parola.

Formato:	Bit	Q[indirizzo byte].[indirizzo bit]	**Q10.1**
	Byte	QB[indirizzo byte iniziale]	**QB4**
	Parola	QW][indirizzo byte iniziale]	**QW4**
	DoppiaParola	QD[indirizzo byte iniziale]	**QD4**

Indirizzamento del area di memoria variabile (V)

La memoria V può essere utilizzata per memorizzare i risultati intermedi di operazioni che vengono eseguite dalla logica di controllo del programma utente. Si potrà inoltre usare la memoria V per memorizzare altri dati pertinenti al processo o al compito di interesse dell'utente. Si potrà accedere all'area di memoria V in byte, parola e doppia parola.

Formato:	Bit	V[indirizzo byte].[indirizzo bit]	**V10.1**
	Byte	VB[indirizzo byte iniziale]	**VB4**
	Parola	VW][indirizzo byte iniziale]	**VW4**
	DoppiaParola	VD[indirizzo byte iniziale]	**VD4**

Indirizzamento dell'area di merker (M)

I bit di merker interni(memoria M) possono essere utilizzati come relè di controllo per memorizzare lo stato intermedio di una operazione o altre informazioni di controllo.Si potrà accedere all'area di merker in byte, parola e doppia parola.

Formato:	Bit	M[indirizzo byte].[indirizzo bit]	**M10.1**
	Byte	MB[indirizzo byte iniziale]	**MB4**
	Parola	MW][indirizzo byte iniziale]	**MW4**
	DoppiaParola	MD[indirizzo byte iniziale]	**MD4**

Indirizzamento dell'area di memoria dei temporizzatori (T)

I temporizzatori sono elementi che contano gli incrementi di tempo. I temporizzatori hanno diversa risoluzione da 1 ms a 100ms. Le seguenti variabili sono associate al temporizzatore.

- Valore corrente: numero intero con segno a 16 bit che memorizza l'ammontare di tempo conteggiato dal temporizzatore.

- Bit di temporizzazione: questo bit si attiva (è impostato su 1) se il valore corrente del temporizzatore è maggiore o uguale al valore del bit di default. (Il valore di default viene immesso parte integrante dell'operazione di temporizzazione)

Indirizzamento dell'area di memoria di contatori (C)

I contatori sono elementi che contano ogni transizione da negativa a positiva all'ingresso o agli ingressi del contatore . Ci sono due tipi di contatori: il primo conta solo in avanti, il secondo sia in avanti che indietro(bidirezionale). Le seguenti variabili sono associate al contatore

- Valore corrente: numero intero con segno a 16 bit che memorizza il conteggio finora avuto.

- Bit di temporizzazione: questo bit si attiva (è impostato su 1) se il valore corrente del contatore è maggiore o uguale al valore del bit di default. (Il valore di default viene immesso parte integrante dell'operazione di conteggio)

-

Indirizzamento degli ingressi analogici (AI)

Il sistema converte un valore analogico del mondo reale(ad esempio un valore di temperatura o di potenza) in un valore digitale in formato parola(16 bit). A tali valori si può accedere mediante l'identificazione di area (AI), la dimensione dei dati (W) e l'indirizzo del byte iniziale. Gli ingressi analogici sono parole e iniziano sempre su indirizzi a byte pari. I valori di ingresso analogici sono di sola lettura.

Formato: AIW[indirizzo byte iniziale] **AIW4**

Indirizzamento delle uscite analogiche (AQ)

Il sistema converte un valore digitale in formato parola (16 bit) in una corrente o potenza, proporzionale al valore digitale. Tali valori possono essere scritti indicando l'identificativo di area (AQ), la dimensione dei dati (W) e l'indirizzo del byte iniziale. Gli ingressi analogici sono parole che iniziano sempre su byte pari..

Formato: AQW[indirizzo byte iniziale] **AQW4**

Indirizzamento degli accumulatori (AC)

Gli accumulatori sono dispositivi di lettura/scrittura che possono essere utilizzati come memoria. Ad esempio , con gli accumulatori possono passare parametri dai sottoprogrammi e memorizzare i valori intermedi utilizzati in un calcolo. Il sistema fornisce quattro accumulatori a

Formato: AC[numero accumulatore] **AC0**

Indirizzamento indiretto delle aree di memoria del sotto sistema

L'indirizzamento indiretto usa un puntatore per accedere ai dati in memoria. Il sistema permette di utilizzare indirettamente le seguenti arre di memoria: I,Q,V,M,T e C. Non è ammesso l'indirizzamento indiretto di singoli bit o di valori analogici.

Creazione di un puntatore

Per indirizzare indirettamente un'area in memoria occorre creare dapprima un puntatore che punti alla posizione stessa. I puntatori sono valori di memoria a doppia parola contenente un altro indirizzo di memoria. Si possono utilizzare come puntatori indirizzi di memoria V o accumulatori AC1,AC2,AC3. Per creare un puntatore si utilizza l'operazione Trasferisci doppia parola (MOVD) che trasferisce l'area di memoria indirizzata indirettamente nell'area del puntatore. L'operando di ingresso dell'operazione deve essee preceduto da una "&" indicante che è l'indirizzo di memoria e non il suo valore, che deve essere trasferito nell'indirizzo identificato dall'operando di uscita.

Esempio : MOVD &VB100,VD204

Utilizzo di un puntatore per l'accesso ai dati

L'utilizzo di un asterisco (*) davanti a un operando di una istruzione indica che l'operando è un puntatore.

Modifica di puntatori

L'utente può modificare il valore di un puntatore. Essendo i puntatori valori a 32 bit occorre utilizzare le operazioni a doppia parola per modificare i valori del puntatore. Con semplici operazioni matematiche, come quella di somma e di incremento, tali valori possono essere modificati. Ricordarsi di adeguare alla dimensione dei dati a cui accedere :

Per l'accesso a byte incrementare il valore del puntatore di uno

Per l'accesso a parola o valore corrente di un temporizzatore o contatore, addizionare il valore 2 o incrementare il valore del puntatore di 2

Per l'accesso a doppia parola , addizionare il valore 4 o incrementare il valore del puntatore di 4.

Istruzioni a contatti

LD	Carica operazione
A	Combina il valore di bit tramite AND
O	Combina il valore di bit tramite OR
LDN	Carica il valore del bit negato
AN	Combina il valore di bit negato tramite AND
ON	Combina il valore di bit negato tramite OR
LDI	Carica il valore di bit direttamente
AI	Combina il valore di bit direttamente tramite AND
OI	Combina il valore di bit direttamente tramite OR
LDNI	Carica il valore del bit negato direttamente
ANI	Combina direttamente il valore di bit negato tramite AND
ONI	Combina direttamente il valore di bit negato tramite OR
NOT	Modifica del valore superiore
EU	Rilevamento di fronte positivo
ED	Rilevamento di fronte negativo
LDB=	Confronto di byte uguali, caricando il risultato
AB=	Confronto di byte uguali, combina il risultato tramite AND
OB=	Confronto di byte uguali, combina il risultato tramite OR
LDB>=	Confronto di byte uguali o maggiore, caricando il risultato
AB>=	Confronto di byte uguali o maggiore, combina il risultato tramite AND
OB>=	Confronto di byte uguali o maggiore, combina il risultato tramite OR
LDB<=	Confronto di byte uguali o minore, caricando il risultato
AB<=	Confronto di byte uguali o minore, combina il risultato tramite AND
OB<=	Confronto di byte uguali o minore, combina il risultato tramite OR
LDW=	Confronto di parole uguali, caricando il risultato
AW=	Confronto di parole uguali, combina il risultato tramite AND
OW=	Confronto di parole uguali, combina il risultato tramite OR
LDW>=	Confronto di parole uguali o maggiore, caricando il risultato
AW>=	Confronto di parole uguali o maggiore, combina il risultato tramite AND
OW>=	Confronto di parole uguali o maggiore, combina il risultato tramite OR
LDW<=	Confronto di parole uguali o minore, caricando il risultato
AW<=	Confronto di parole uguali o minore, combina il risultato tramite AND
OW<=	Confronto di parole uguali o minore, combina il risultato tramite OR
LDD=	Confronto di doppie parole uguali, caricando il risultato
AD=	Confronto di doppie parole uguali, combina il risultato tramite AND
OD=	Confronto di doppie parole uguali, combina il risultato tramite OR
LDD>=	Confronto di doppie parole uguali o maggiore, caricando il risultato
AD>=	Confronto di doppie parole uguali o maggiore, combina il risultato tramite AND
OD>=	Confronto di doppie parole uguali o maggiore, combina il risultato tramite OR
LDD<=	Confronto di doppie parole uguali o minore, caricando il risultato
AD<=	Confronto di doppie parole uguali o minore, combina il risultato tramite AND
OD<=	Confronto di doppie parole uguali o minore, combina il risultato tramite OR

LDR=	Confronto di numeri reali uguali, caricando il risultato
AR=	Confronto di numeri reali uguali, combina il risultato tramite AND
OR=	Confronto di numeri reali uguali, combina il risultato tramite OR
LDR>=	Confronto di numeri reali uguali o maggiore, caricando il risultato
AR>=	Confronto di numeri reali uguali o maggiore, combina il risultato tramite AND
OR>=	Confronto di numeri reali uguali o maggiore, combina il risultato tramite OR
LDR<=	Confronto di numeri reali uguali o minore, caricando il risultato
AR<=	Confronto di numeri reali uguali o minore, combina il risultato tramite AND
OR<=	Confronto di numeri reali uguali o minore, combina il risultato tramite OR
=	Assegna, copia nel parametro specificato il valore superiore dello stack
=I	Assegna direttamente, copia immediatamente nel parametro specificato il valore superiore dello stack
S	Imposta il numero di punti specificato
R	Resetta il numero di punti specificato
SI	Imposta direttamente il numero di uscite fisiche specificato
RI	Resetta direttamente il numero di uscite fisiche specificato
NOP	Nessuna operazione

Istruzioni timer e contatori

TON	Avvia temporizzatore come ritardo all'inserzione
TONR	Avvia temporizzatore come ritardo all'inserzione con memoria
CTU	Conta in avanti
CTUD	Conta in avanti ed indietro

Istruzioni matematiche

+I	Somma due numeri interi a 16 bit
-I	Sottrai due numeri interi a 16 bit
+D	Somma due numeri interi a 32 bit
-D	Sottrai due numeri interi a 32 bit
+R	Somma due numeri reali
-R	Sottrai due numeri reali
MUL	Moltiplica due numeri interi
DIV	Dividi due numeri interi
*R	Moltiplica due numeri reali
/R	Dividi due numeri reali
SQRT	Radice quadrata di un numero reale

Istruzioni incremento e trasferimento

INCB	Incrementa un byte di 1
DECB	Decrementa un byte di 1
INCW	Incrementa una parola di 1
DECW	Decrementa una parola di 1
INCD	Incrementa una doppia parola di 1
DECD	Decrementa una doppia parola di 1
MOVB	Trasferisci byte
MOVW	Trasferisci parola
MOVR	Trasferisci numero reale
BMB	Trasferisci blocco di byte
BMW	Trasferisci blocco di parole
BMD	Trasferisci blocco di doppie parole
SWAP	Scambia byte nella parola

Istruzioni controllo programma

END	Termina il programma
JMP	Salta all'etichetta
LBL	Definisce l'etichetta
CALL	Richiama sottoprogramma
SBR	Inizia sottoprogramma
RET	Fine del sottoprogramma
CRET	Fine condizionata del sottoprogramma

Istruzioni stack logico

ALD	Combina il primo e il secondo elemento tramite AND
OLD	Combina il primo e il secondo elemento tramite OR
LPS	Duplicazione logica
LRD	Copiatura logica
LPP	Prelevamento logico

Istruzioni booleane

ANDB	Combina byte tramite AND
ORB	Combina byte tramite OR
XORB	Combina byte tramite OR esclusivo
ANDW	Combina parola tramite AND
ORW	Combina parola tramite OR
XORW	Combina parola tramite OR esclusivo
ANDD	Combina doppia parola tramite AND
ORD	Combina doppia parola tramite OR
XORD	Combina doppia parola tramite OR esclusivo
INVB	Inverti byte
INVW	Inverti parola
INVD	Inverti doppia parola

Capitolo

6

Ambiente d'interfaccia AxesBrainStudio

AxesBrainStudio è una applicazione contenitore di "Pagine" utilizzate per dare una Interfaccia Uomo ai servizi messi a disposizione dal AxesBrainServer, fa parte delle funzionalità quali la gestione delle segnalazioni, la gestione di alcuni Eventi come la tastiera , il suo compito principale è accogliere delle altre applicazioni fornendo un modo di navigazione tra di loro uniforme, raggruppandole in "Ambienti" separati e rappresentandole in una veste grafica uniforme.

Le pagine possono essere fatte in tre modi: generando un controllo OCX con determinate caratteristiche (che può essere sviluppato in Visual Basic o Visual C++), in formato HTML oppure generando una DLL utilizzando delle classi specifiche fornite dall' AB&T.

La configurazione delle pagine e degli ambienti viene scritta nel registro di Windows e viene caricata all' avvio del AxesBrainStudio.

L'interfaccia fornisce la gestione di 10 tasti funzione per ogni pagina, un' area comune sempre visibile per I messaggi di sistema.

Le applicazioni contenute nel "AxesStudio"sono chiamate "Pagine" e vengono raggruppate in "Ambienti", la navigazione permette il passaggio da un ambiente all'altro ritrovando sempre l'ultima pagina selezionata dell'ambiente.

Ad ogni "Pagina" è inoltre abbinata una icona e un scritta di identificazione, in modo che una volta selezionato l'ambiente l'utente può vedere tutte le icone delle "Pagine" ivi contenute e richiamare quella desiderata.

Le informazioni sul numero di "Ambienti", delle "Pagine" abbinate con le loro icone e scritte, della modalità del tipo di applicazione abbinata alla "Pagina" sono inserite nel "Registry " AxesBrain, in modo che solo personale esperto possa cambiare la configurazione del sistema.

Il GMSStudio vede le applicazioni come delle funzioni o librerie da qui la suddivisione di librerie DLL quelle base, che fanno parte del applicativo stesso, le librerie ad oggetti o OCX e applicazioni HTML, in questo caso si comporta come un BROWSER di navigazione su internet.

Le applicazioni o "Pagine" sono di tre tipi:

- Applicazioni base
- Applicazioni utente OCX
- Applicazioni utente HTML

Applicativi base

Per l'interfaccia utente con il sistema di movimentazione sono stati sviluppati alcuni applicativi o "Pagine" , che si suddividono negli ambienti base.

Ambiente di sistema

- Pagina "Gestione OEM Capability"
- Pagina "Percorso assi 2d "
- Pagina "Percorso assi nD "
- Pagina "Gestione segnali ingresso e uscita"
- Pagina "Caratterizzazione sistema"

Ambiente automazione

- Pagina "Edit Part Program AxesBrainL"
- Pagina "Gestione cicli AxesBrainL"
- Pagina "Debug cicli AxesBrainL"
- Pagina "Configurazione risorse AxesBrainL"

Ambiente CNC

- Pagina "Edit Part Program AxesBrainISO"
- Pagina "Gestione programmi AxesBrainISO"
- Pagina "Gestione Tabelle AxesBrainISO"
- Pagina "Gestione manuale assi AxesBrainISO"
- Pagina "Configurazione risorse AxesBrainISO"

Ambiente PLC

- Pagina "Edit Part Program AxesBrainAWL"
- Pagina "Gestione ciclo AxesBrainAWL"
- Pagina "Debug ciclo AxesBrainAWL"
- Pagina "Configurazione risorse AxesBrainAWL"

Ambiente OEM Capability

Pagina “Edit generico di file”

Pagina “Gestione sotto sistema OEM”

Pagina “Configurazione risorse OEM”

Ambiente Visione

Pagina “Misurale”

Pagina “Riconoscimento parti"

Applicativi OCX

Gli applicatici OCX sono sviluppati come una normale libreria, usando il linguaggio di programmazione meglio conosciuto dal Cliente.

Avremo quindi che la libreria sarà gestita come una pagina dal AxesBrainStudio raggruopato da un ambiente specificato nella registrazione.

Le "PAGINE" OCX hanno la possibilità oltre che dei servizi AxesBrainServer DCOM diretti anche usare l'accesso a tali servizi tramite una libreria OCX GMSCOM.

Per avere una User Interface omogenea esiste una libreria funzioni del AxesBrainStudio come la gestione dei tasti funzione, la gestione della messaggistica di sistema, il ricevimento degli eventi da parte del AxesBrainServer e altre funzioni di uso comune.

Applicativi HTML

Gli applicatici HTML sono sviluppati come un normale documento HTML, in questo caso GMSStudio si comporta come un BROWSER .

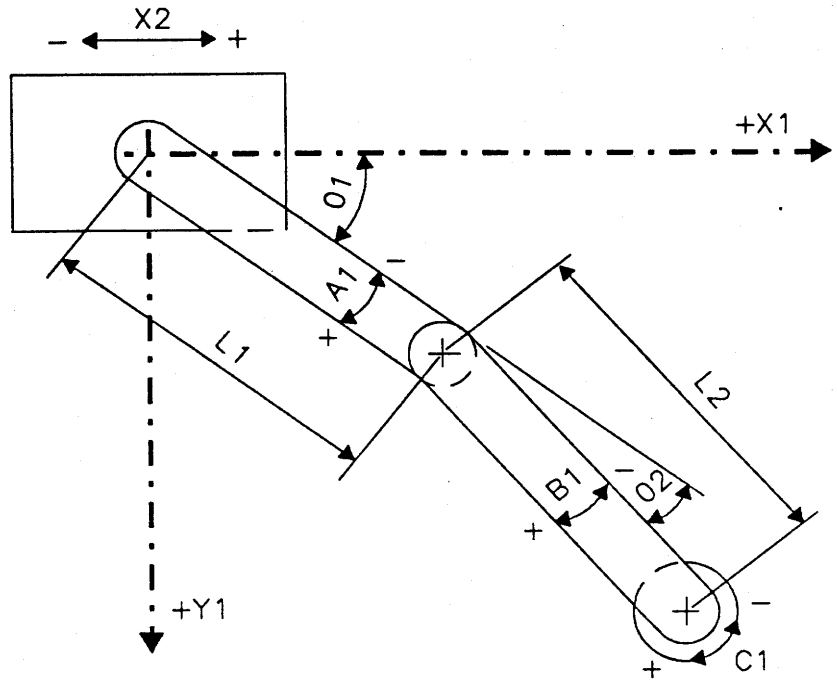
Avremo quindi che il documento sarà gestito come una pagina dal AxesBrainStudio raggruppato da un ambiente specificato nella registrazione.

I documenti HTML hanno la possibilità oltre che dei servizi AxesBrainServer DCOM diretti anche di usare l'accesso a tali servizi tramite una libreria OCX AxesBrainCOM.

Assi Virtuali

Gli assi virtuali, come dice la parola stessa, non sono assi fisici ma assi che per comodità utente vengono caratterizzati su sistemi a geometria SCARA o CILINDRICA in modo che oltre ad effettuare movimenti sugli assi fisici si possa anche effettuare movimenti secondo un sistema cartesiano.
Con tale sistema, inoltre si possono effettuare interpolazioni lineari e circolari.

Sistema SCARA



Numero di assi SCARA: indica il numero di assi da prendere in considerazione; minimo 2 (i primi 2); tre (1,2,3) oppure 4 (tutti)

Nell'esempio riportato, vengono presi in considerazione tutti e 4 (i 2 snodi A1 e B1,il 3° asse rotante C1 posto all'estremità del secondo snodo e il 4° asse traslante X2)

Nome del primo asse: indica il primo asse reale del sistema (1° snodo); nell'esempio l'asse A1

Lunghezza del primo asse: indica la lunghezza del primo asse (distanza fra il primo e secondo snodo); nell'esempio L1

Nome del secondo asse: indica il secondo asse reale del sistema (2° snodo); nell'esempio B1

Lunghezza del secondo asse: indica la lunghezza del secondo asse (distanza fra snodo centrale e asse pinza); nell'esempio L2.
Nel caso sia presente il terzo asse (nell'esempio C1), l'asse pinza deve essere concentrico con l'asse di rotazione del terzo asse (in questo caso la lunghezza del secondo asse è fissa). Se invece vi è eccentricità, la lunghezza del secondo asse (rispetto all'asse pinza) è variabile, essendo dipendente dalla posizione angolare della pinza. Di conseguenza, poiché si imposta una sola lunghezza, il sistema cartesiano ottenuto sarà valido solo in quelle condizioni.

In base a queste considerazioni, in caso di eccentricità, occorre tener presente quanto segue:

- non è possibile ottenere traiettorie rettilinee e circolari e l'errore di traiettoria sarà tanto maggiore quanto maggiore è l'eccentricità.

Nome del terzo asse: indica l'eventuale asse rotante posto all'estremità del secondo asse; nell'esempio C1.
L'indicazione o meno di questo asse in tabella, influisce solo sulla geometria del sistema agli effetti della compensazione automatica di detto rotante e precisamente:

- Asse non indicato in tabella: in caso di richiesta movimento assi (reali o virtuali) tale asse rotante viene ignorato (naturalmente se non richiesto nel movimento)

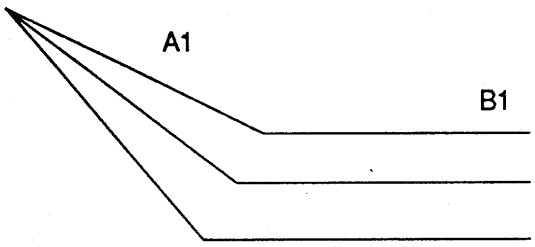
- Asse indicato in tabella: in caso di richiesta movimento assi (reali o virtuali) tale asse rotante (naturalmente se non richiesto nel movimento) viene automaticamente compensato; la pinza posta su tale asse rimane quindi sempre parallela a se stessa.
  E' certo che se questo asse viene richiesto nel movimento, l'indicazione o meno dell'asse nella tabella non ha più alcun significato essendo il movimento richiesto determinante.

Nome del quarto asse: indica l'eventuale asse traslante (traslazione di tutto Io SCARA); nell'esempio X2

Compensazione meccanica: questo dato serve per individuare il tipo di robot secondo la convenzione seguente:

0 = robot di tipo SCARA tradizionale
± 1 = robot di tipo SCARA ma con sistema a pantografo

dove muovendo un solo asse (nell'esempio sottostante A1) il secondo asse (nell'esempio B1) si mantiene parallelo a se stesso.



Il segno algebrico va inteso nel modo seguente:
Con il segno positivo si intende la compensazione automatica sull'asse B1 sommando l'angolo effettuato dall'asse A1.

Con il segno negativo si intende la compensazione automatica sull'asse B1 sottraendo l'angolo effettuato dall'asse A1.

Offset assoluto: questo dato viene utilizzato ogni volta si desideri ottenere uno zero asse pratico diverso da quello teorico.

Massima velocità: corrisponde alla massima velocità voluta sull'asse virtuale; se questa velocità fosse impossibile da effettuare (occorre tener presente che i due assi reali abbinati all'asse virtuale hanno ognuno una massima velocità) per default viene utilizzata la massima possibile.

Massima accelerazione e massima decelerazione:
corrispondono alla massima accelerazione e decelerazione volute sull'asse virtuale; anche per questi dati se i valori indicati fossero superiori al possibile, per default vengono utilizzati i massimi possibili.

Note:

a) Poiché tutti i calcoli per gli assi virtuali vengono eseguiti secondo la geometria classica operando su seni e coseni di angoli espressi in gradi, occorre tenere presente quanto segue:

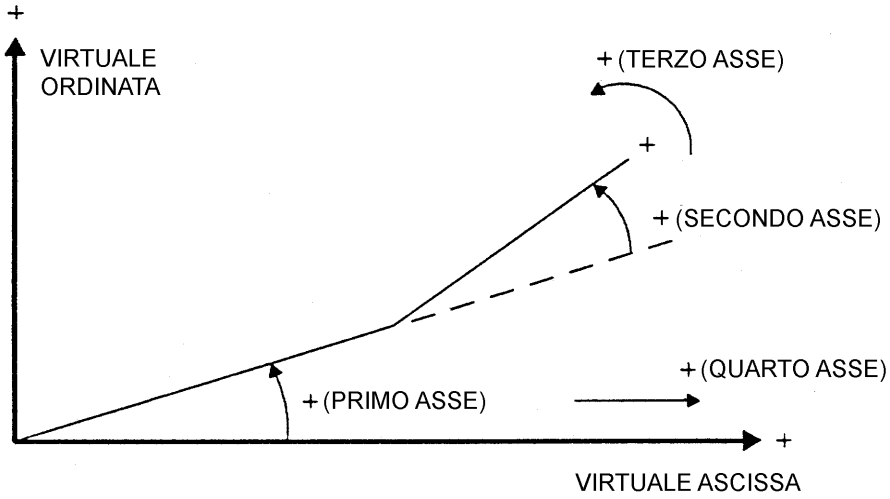
1 - L'unità di misura dei primi 3 assi reali deve essere il grado.

2 - L'unità di misura del quarto asse, può essere il mm o pollici a seconda che le lunghezze primo e secondo asse vengano date rispettivamente in mm o inc.

3 - L'offset assoluto del primo asse reale deve essere dato (se è presente il 4° asse traslante) in modo che l'asse virtuale ascissa sia parallelo al quarto asse SCARA.

4 - L'offset assoluto del secondo asse reale deve essere dato in modo che i primi due assi risultino allineati (braccio tutto teso).

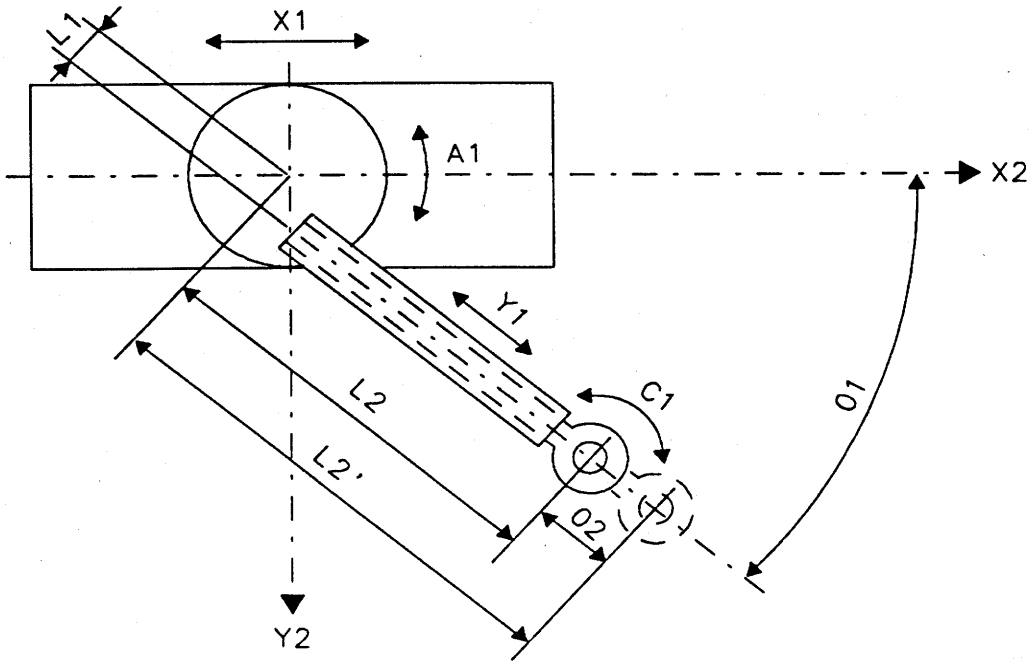
5 - Le direzioni di movimento (positiva, negativa) degli assi del sistema devono essere tutte concordi: per i primi 3 assi rotanti direzione positiva significa andare da ascissa positiva a ordinata positiva; l'eventuale quarto asse deve avere la stessa direzione della ascissa. La figura successiva fornisce un esempio di dette direzioni.



6 - In base alle considerazioni viste nei punti 1-5 precedenti, si avrà che dopo un movimento a zero di tutti e 4 gli assi reali SCARA, l'asse virtuale ascissa si troverà ad una coordinata equivalente alla somma delle lunghezze del primo e secondo asse; l'asse virtuale ordinata a coordinata zero.

b) Con un sistema SCARA, non è possibile richiedere tramite una stessa istruzione di movimento, assi virtuali unitamente ai primi due assi reali del sistema (uno dei due o entrambi.
E' viceversa possibile richiedere assi virtuali unitamente al 3° e/o 4° asse reale del sistema.

Sistema CILINDRICO



Numero di assi del sistema CILINDRICO: indica il numero di assi da prendere in considerazione; 1°+2°,1°+2°+ 3°,1°+2°+3°+ 4°. Si consiglia di indicare sempre 4 e se qualche asse non è presente (2°, 3°, 4°) non indicarlo nelle righe apposite.
Nell'esempio riportato, vengono presi in considerazione tutti e 4 gli assi.

Nome del primo asse: indica il primo asse reale del sistema; nell'esempio A1. Il primo asse è sempre l'asse rotante generale e deve essere sempre presente

Lunghezza del primo asse: indica la distanza (può anche essere 0) fra l'asse di rotazione del primo asse e l'asse pinza; tale distanza va data perpendicolarmente all'asse cannotto (se cannotto mancante, al braccio porta pinza); nell'esempio L1.

Nome del secondo asse: indica il secondo asse reale del sistema; nell'esempio Y1.
Tale asse è sempre l'asse cannotto e potrebbe anche non essere presente; in tale caso deve però essere necessariamente presente il quarto asse

Lunghezza del secondo asse: indica la distanza fra l'asse di rotazione del primo asse e l'asse pinza; tale distanza va data parallelamente all'asse cannotto (se cannotto mancante, al braccio porta pinza); nell'esempio, L2 se il secondo asse (cannotto) non ha offset assoluto; 12', se questo asse ha un suo offset (02).
Nel caso sia presente il terzo asse (nell'esempio C1), l'asse pinza deve essere concentrico con l'asse di rotazione del terzo asse (in questo caso la lunghezza del secondo asse è fissa). Se invece vi è eccentricità, la lunghezza del secondo asse (rispetto all'asse pinza) è variabile, essendo indipendente dalla posizione angolare della pinza. Di conseguenza, poiché si imposta una sola lunghezza, il sistema cartesiano ottenuto sarà valido solo in quelle condizioni.
In base a queste considerazioni, in caso di eccentricità, occorre tener presente quanto segue:

- non è possibile ottenere traiettorie rettilinee e circolari e l'errore di traiettoria sarà tanto maggiore quanto maggiore è l'eccentricità.

Nome del terzo asse: indica l'eventuale asse rotante posto all'estremità del cannotto (o braccio porta pinza se cannotto mancante); nell'esempio C1.
L'indicazione o meno di questo asse in tabella, influisce solo sulla geometria del sistema agli effetti della compensazione automatica di detto rotante e precisamente:

- Asse non indicato in tabella: in caso di richiesta movimento assi (reali o virtuali) tale asse rotante viene ignorato (naturalmente se non richiesto nel movimento)

- Asse indicato in tabella: in caso di richiesta movimento assi (reali o virtuali) tale asse rotante (naturalmente se non richiesto nel movimento) viene automaticamente compensato; la pinza posta su tale asse rimane quindi sempre parallela a se stessa.
  E' certo che se questo asse viene richiesto nel movimento, l'indicazione o meno dell'asse nella tabella non ha più alcun significato essendo il movimento richiesto determinante.

Nome del quarto asse: indica l'eventuale asse traslante (traslazione di tutto il CILINDRICO); nell'esempio X1. Tale asse potrebbe anche non essere presente purché sia presente il secondo asse;

Compensazione meccanica: tale dato va sempre dato con valore 0.

Offset assoluto: questo dato viene utilizzato ogni qualvolta si desideri ottenere uno zero asse pratico diverso da quello teorico.

Massima velocità: corrisponde alla massima velocità voluta sull'asse virtuale; se questa velocità fosse impossibile da effettuare, (ogni asse reale appartenente al sistema ha una propria massima velocità) per default viene utilizzata la massima possibile.

Massima accelerazione e decelerazione: corrispondono alla massima accelerazione e decelerazione volute sull'asse virtuale; anche per questi dati se i valori indicati fossero superiori al possibile, per default vengono utilizzati i massimi possibili.

Note:

a) Poiché tutti i calcoli per gli assi virtuali vengono eseguiti secondo la geometria classica, operando su seni e coseni di angoli espressi in gradi, occorre tenere presente quanto segue:

1 - L'unità di misura degli assi rotanti (primo e terzo asse) reali deve essere il grado.

2 - L'unità di misura del quarto asse reale può essere il mm o l'inc. a seconda che la lunghezza secondo asse venga data rispettivamente in mm o inc.

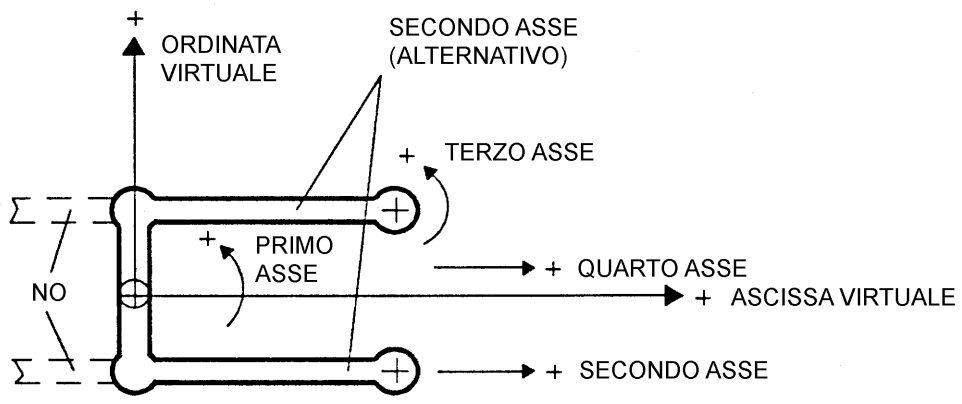
3 - L'offset assoluto del primo asse reale deve essere dato (se è presente il 4 asse traslante) in modo che l'asse virtuale ascissa sia parallelo al quarto asse CILINDRICO.

4 - ll secondo asse (cannotto) può o meno avere offset; tale offset implica chiaramente una diversa lunghezza del secondo asse.

5 - Le direzioni di movimento (positive, negative) degli assi del sistema devono essere tutte concordi: per gli assi rotanti (primo e terzo asse) direzione positiva significa andare da ascissa positiva a ordinata positiva; gli assi lineari (secondo e quarto) devono avere la stessa direzione delle ascisse.

6 - ll secondo asse (cannotto) non può trovarsi in ascissa negativa in caso contrario ruotare il primo asse in modo che si trovi in ascissa positiva.

La figura successiva fornisce un esempio di come devono essere messi gli assi su di un sistema cilindrico



7 - In base alle considerazioni viste nei punti 1-6 precedenti, si avrà che dopo un movimento a zero di tutti e 4 gli assi reali CILINDRICI, l'asse virtuale ascissa si troverà ad una coordinata pari alla lunghezza del secondo braccio e l'asse virtuale ordinata ad una coordinata pari alla lunghezza del primo braccio. (Per quest'ultimo asse la coordinata sarà positiva o negativa)

b) Con un sistema CILINDRICO, non è possibile richiedere tramite una stessa istruzione di movimento, assi virtuali unitamente ai due assi reali (uno dei due o entrambi) costituenti il sistema cilindrico e precisamente: 1° asse del sistema (Rotante generale), 2° asse del sistema (cannotto) o 4° asse del sistema (traslante) nel caso di assenza dell'asse cannotto

E' invece possibile richiedere assi virtuali unitamente al 3° e/o 4° asse (se è presente il 2° asse)
Riepilogo degli assi reali non abbinabili ai virtuali in una stessa istruzione di movimento:

- Sistema cilindrico composto dal 1° e 2° asse (rotante generale + cannotto)
  Non sono ammessi virtuali +1° e/o 2° asse reale

- Sistema cilindrico composto dal 1° e 4° asse (rotante generale + traslante)
  Non sono ammessi virtuali + 1° e/o 4° asse reale

- Sistema cilindrico completo di 1°, 2°, e 4° asse

Non sono ammessi virtuali + 1° e/o 2° asse (da solo o con il 4°).

Assi a portale (Gantry)

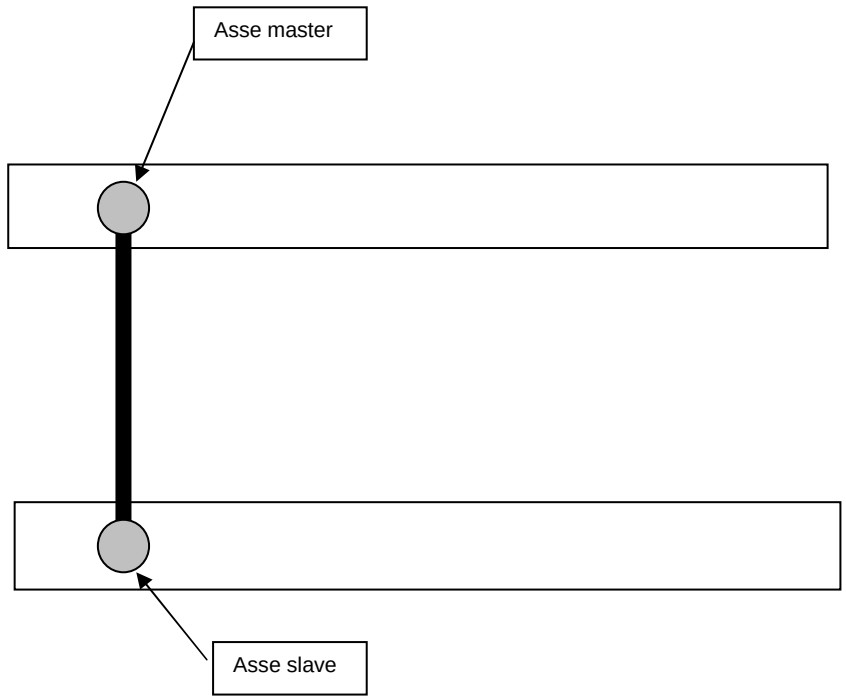
L'asse a portale(Gantry)è una struttura meccanicamente rigida (normalmente è una struttura a portale) e corrisponde pertanto ad un asse unico, ma è trattata dal controllo come se fosse costituita da una coppia di assi (asse master e asse slave, ciascuno con i propri sistemi di conteggio ed il proprio azionamento.
Una delle funzioni del controllo è quella di mantenere la posizione dell'asse "slave" più prossima possibile a quella dell'asse "master".

I movimenti richiesti sugli assi Master-Slave, vengono eseguiti nel modo seguente:

La richiesta di movimento può essere effettuata tramite tutte le istruzioni di movimento.

Nel caso di movimento relativo al solo asse Master, l'asse Slave lo segue o rimane fermo rispettivamente in caso di associazione o disassociazione.

Durante il movimento, l'asse Slave segue il suo Master in tempo reale



Controllo asse con volantino

Il posizionamento di un asse in manuale può essere abbinato ad un dispositivo chiamato volantino che viene visto dal sistema come un asse di sola lettura .

Il valore di posizione letto dal volantino modifica la posizione dell'asse abbinato, è così possibile dare degli incrementi micrometrici all' asse stesso.

Il volantino è visto come un asse di sola lettura, che tramite opportuni comando viene agganciato ad un asse che ne rimarrà controllato.

Nel linguaggio di automazione AxesBrainL, l'istruzione "-HMS" permette l'inserimento e il disinserimento di un asse volantino "master" ad un asse di posizionamento "slave".

Per gestire l'abbinamento asse volantino con asse di posizionamento in ambiente "DCOM", deve essere utilizzato il servizio "WriteAxesRegister" per entrambi gli assi.

Camme elettroniche

La camma elettronica permette di abbinare la posizione di un gruppo di assi ad un asse "master" ed una tabella di posizioni multiple.

E' così possibile simulare elettronicamente il comportamento delle camme, sostituirne il funzionamento meccanico con un sistema analogo formato da un gruppo di assi asservito ad un asse "master" che può essere di sola lettura.

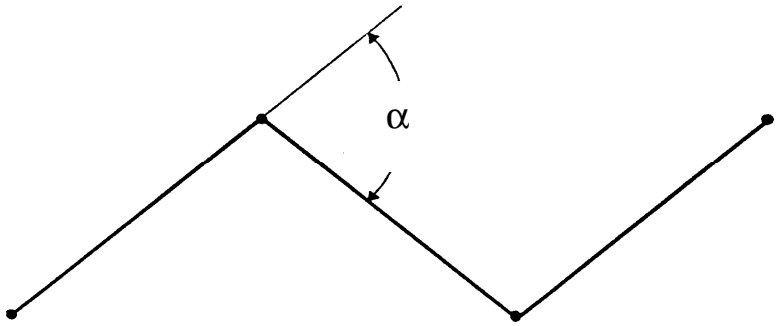
Le leggi del moto dei cedenti è definita come una tabella di vettori, che ne defisse le posizioni rispetto al moto della camma.

Nel linguaggio di automazione AxesBrainL, l'istruzione "-HEC" permette la gestione delle camme elettroniche.

Per gestire le camme elettriche in ambiente "DCOM", deve essere utilizzato il servizio "WriteAxesRegister" per tutti gli assi .

Movimenti in continuo

I vari tipi di continuo, si riferiscono al profilo teorico seguente:



Parametri del continuo

a = switch (da 0 a 3) che determina il tipo di continuo nel modo seguente:
- 0 significa continuo normale
- 1 significa spline di tipo 1
- 2 significa spline di tipo 2
- 3 significa spline di tipo 3

b = coefficiente (da 0 a n) di riduzione velocità ad ogni cambio di direzione fra i vari punti del profilo.

+ = partenza immediata dei movimenti.

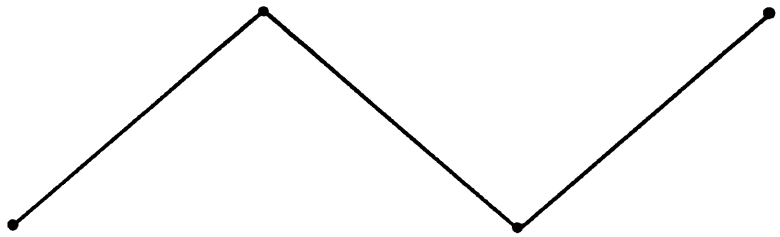
- = partenza dei movimenti condizionata al verificarsi delle seguenti condizioni:
- istruzione HCL trovata nel Part Program;
- nuova istruzione STC con parametro **b** positivo trovata nel Part Program;
- saturazione del buffer contenente gli enti movimenti relativo al continuo;

c = valore (da 0, a 1) che determina il tipo di congiunzione (schiacciamento o meno dell'arco) fra i punti del profilo nel modo seguente:
- 0 significa retta (arco tutto schiacciato) fra i vari punti;
- 1 significa arco normale fra i vari punti;
- da >0 a <1 significa arco più o meno schiacciato; tanto più piccolo è il valore, tanto più schiacciato sarà l'arco;

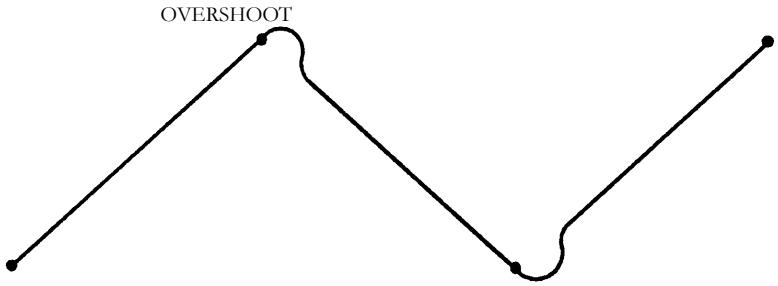
a) Nel caso di continuo normale (parametro **a**= 0), vi è sempre overshoot dopo ogni punto del profilo. La velocità di overshoot dopo ogni movimento è direttamente proporzionale alla accelerazione ed al coefficiente è maggiore (con incremento) o minore (con decremento) in modo non proporzionale rispetto all'angolo α (cambio di direzione fra 2 movimenti successivi). Poichè la massima velocità di overshoot non può comunque superare la massima velocità dell'asse, un valore grande di coefficiente è significativo solamente con un valore piccolo di accelerazione; viceversa con una grande accelerazione ha significato un piccolo valore di coefficiente (se questo valore è grande, viene comunque forzato al massimo ammissibile).

Esempi di profilo con continuo normale
Profilo pratico con coefficiente del parametro **b** = 0

Esempi di profilo con continuo normale
Profilo pratico con coefficiente di riduzione velocità ad ogni cambio di direzione fra i vari punti del profilo. = 0



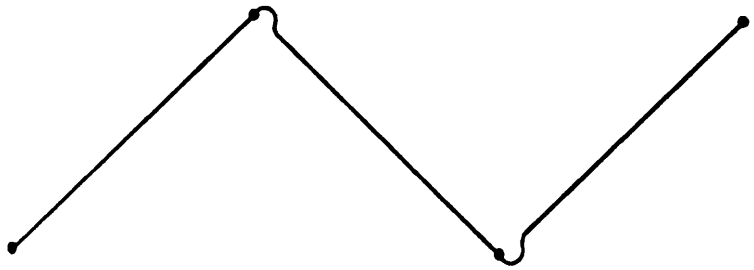
Profilo pratico con coefficiente diverso da 0



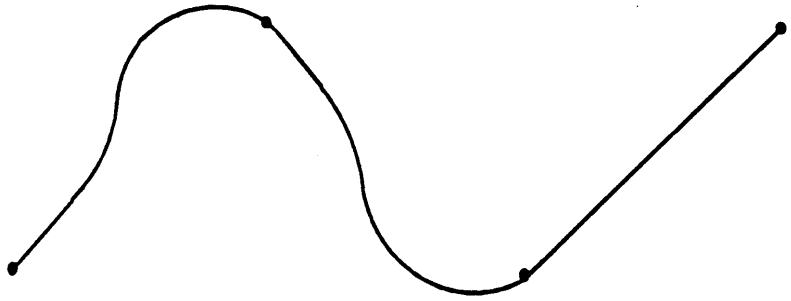
b) Nel caso di spline di tipo 1 ,l'overshoot dopo ogni punto del profilo è nettamente inferiore a quello relativo al continuo normale; praticamente è nullo se si assegna valore 1 al parametro **c**; varia da 0 ad un valore crescente (comunque sempre basso) più il valore del parametro di congiunzione (schiacciamento o meno dell'arco) si avvicina allo 0 (retta con arco tutto schiacciato ,fra i vari punti).
Anche per questo tipo di continuo, si ha una variazione di velocità lungo il profilo, per essa è molto meno evidente rispetto al continuo normale; naturalmente la velocità sul profilo sarà tanto più uniforme tanto minore è l'angolo a, tanto più il valore del parametro o si avvicina a 1 e tanto maggiore (anche se poco influente) è il valore del coefficiente di riduzione velocità

Esempi di profilo con spline di tipo 1

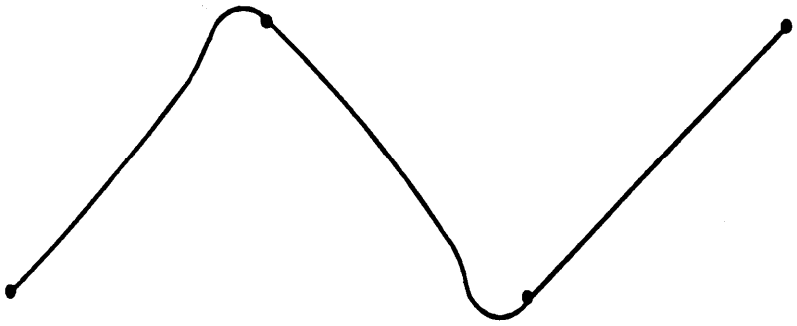
OVERSHOOT PICCOLO



Profilo ottenuto con parametro b=0 e coefficiente di riduzione velocità diverso da 0; se il coefficiente di riduzione velocità è 0, non vi sono naturalmente gli overshoot.



Profilo ottenuto con parametro **c**= i (coefficiente di riduzione velocità diverso da 0)

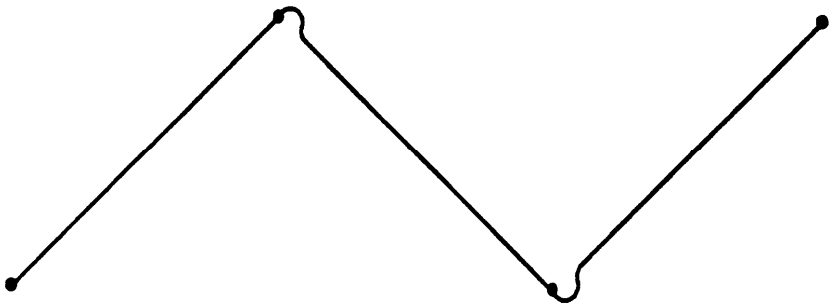


Profilo ottenuto con parametro **c** compreso fra 0 e 1 (coefficiente di riduzione velocità diverso da 0)

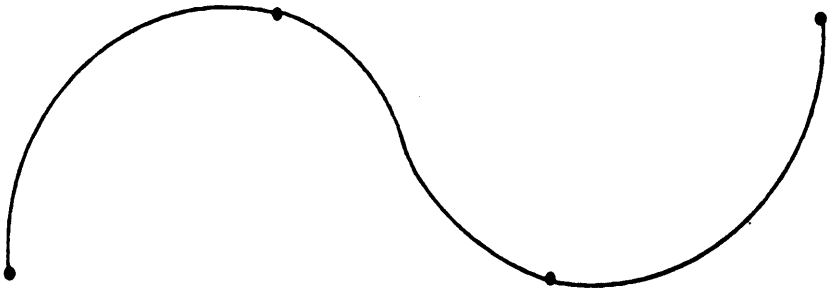
o) Nel caso di spline di tipo 2 (parametro **a** = 2) e di tipo 3 (parametro **a** = 3) si ha un comportamento simile alla spline di tipo 1 ma con la differenza che per la spline di tipo 2 il precalcolo sul profilo viene tatto prendendo in considerazione 3 punti del profilo mentre per la spline di tipo 3 i punti presi in considerazione sono 5.

Esempi di profilo con spline di tipo 2 e 3

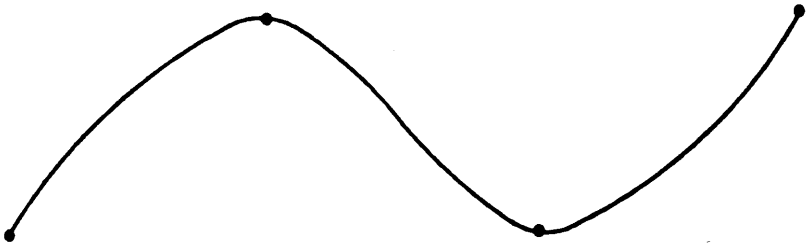
OVERSHOOT PICCOLO



Profilo ottenuto con parametro **c** = 0 e coefficiente STC diverso da 0, se il coefficiente STC è 0, non vi sono naturalmente gli overshoot.



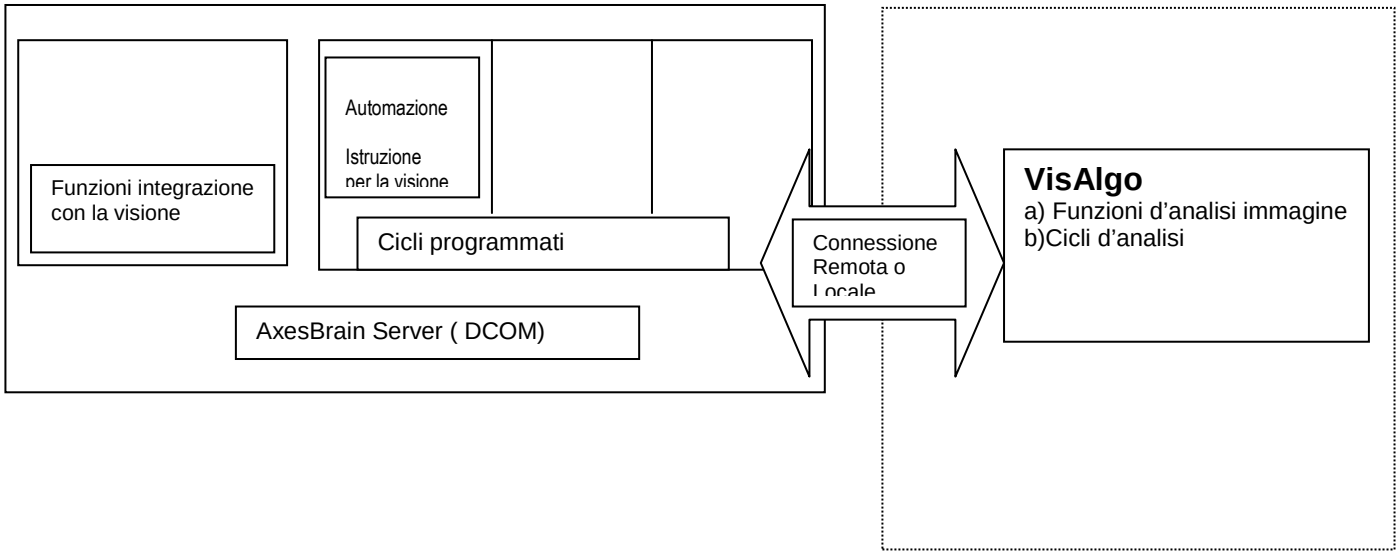
Profilo ottenuto con parametro **c** =1 (coefficiente STC diverso da 0)



Profilo ottenuto con parametro **c** compreso fra 0 e 1 (coefficiente STC diverso da 0).

Integrazione con la visione

L'integrazione con l'ambiente di analisi d'immagine VisAlgo può essere operato in due modalità o attraverso delle specifiche richieste del AxesBrainServer che effettuano delle chiamate alle funzioni base o richieste di eseguire cicli di analisi, una seconda strada è utilizzare le istruzioni d'interfacciamento visione del linguaggio AxesBrainL.



Schema integrazione analisi d'immagine

Come si può notare dallo schema la macchina predisposta per l'analisi d'immagine può essere remotata su un altro PC, naturalmente collegata in rete locale o rete internet.

VisAlgo

Il pacchetto VisAlgo lavora tramite richieste di funzioni d'analisi dell'immagine che effettuano operazioni di :

- Filtro e manipolazione dell'immagina
- Analisi luce e estrazioni automatiche di soglie
- Blobs su rettangoli d'interesse
- Estrazioni di contorni su intersezione di figure su rettangoli d'interesse
- Estrazioni di bordi
- Misurazione di contorni
- Estrazioni di enti sui contorni
- Misurazione sugli enti

Inoltre è previsto un linguaggio AxesBrainL in grado di programmare cicli d'elaborazione di immagini, in modo di ottenere dei risultati mirati alle situazioni reali.