POLITECNICO MILANO 1863 | DIPARTIMENTO DI ELETTRONICA INFORMAZIONE E BIOINGEGNERIA

Fondamenti di robotica

Programmazione del moto

29.03.2026 | Paolo Rocco

Contenuti

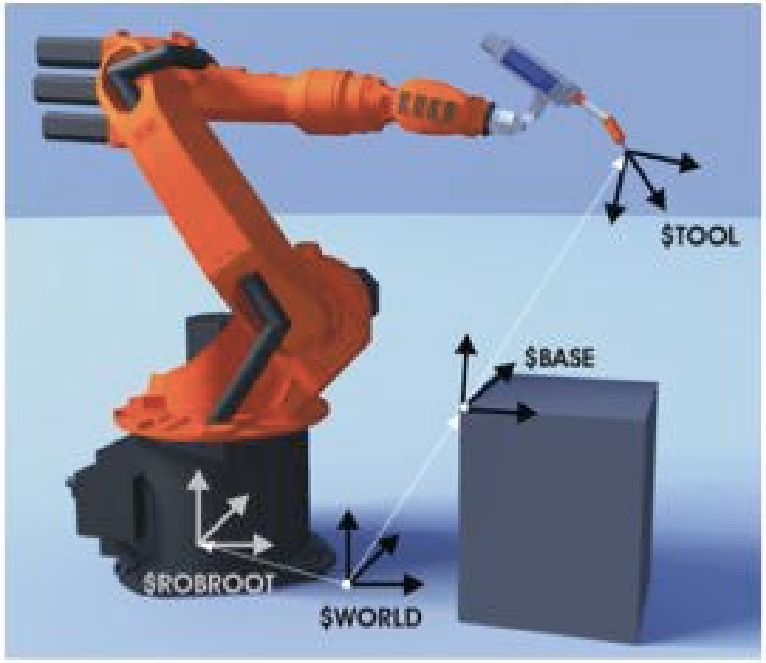
1. Pianificazione del movimento
2. Programmazione del robot

Pianificazione del movimento

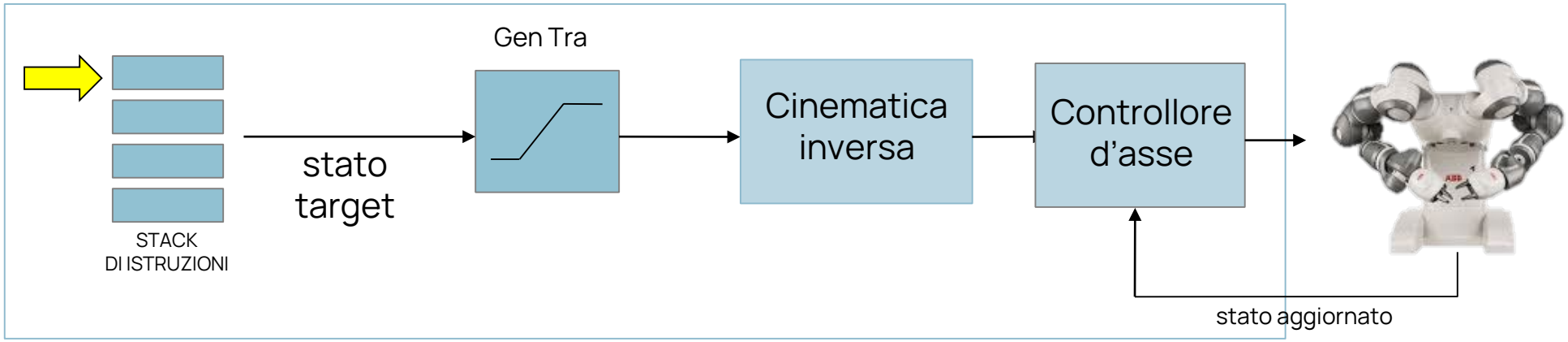
01

Pianificazione del movimento

- Esaurita la descrizione del moto del robot attraverso il suo modello cinematico, passiamo ora al problema della **pianificazione del movimento**
- Si vuole definire la modalità in cui il robot evolve da una postura iniziale a una finale
- La pianificazione del movimento è uno dei problemi essenziali della robotica. Buona parte del successo sul mercato di un robot dipende dalla qualità della pianificazione del movimento.

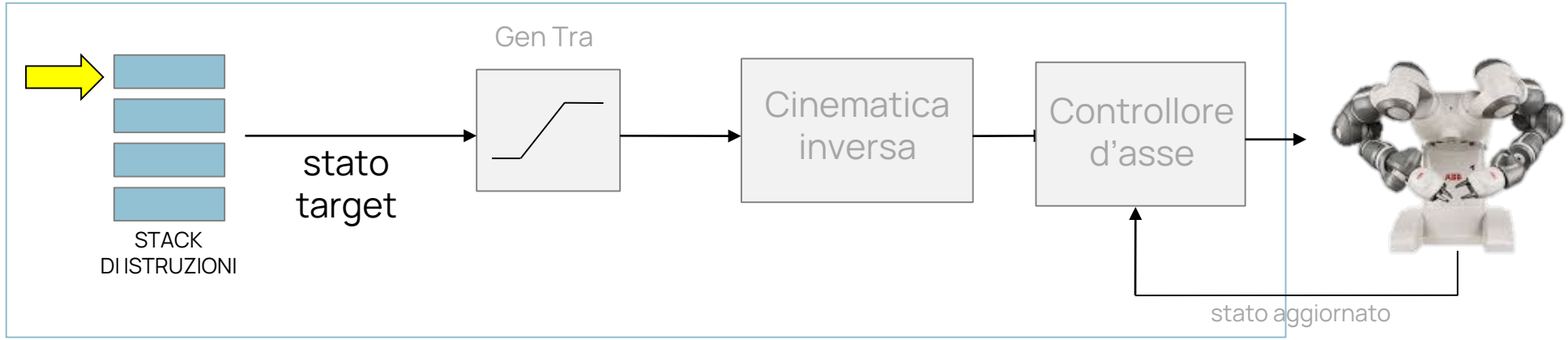


Elementi di un sistema di pianificazione e controllo del movimento



- **Stack di istruzioni**: elenco di istruzioni da eseguire, specificato utilizzando il linguaggio di programmazione proprietario
- **Generazione traiettoria**: converte un'istruzione in una traiettoria da eseguire
- **Cinematica inversa**: mappa la traiettoria dallo spazio cartesiano allo spazio dei giunti (se necessario)
- **Controllore d'asse**: chiude l'anello di controllo garantendo prestazioni di inseguimento

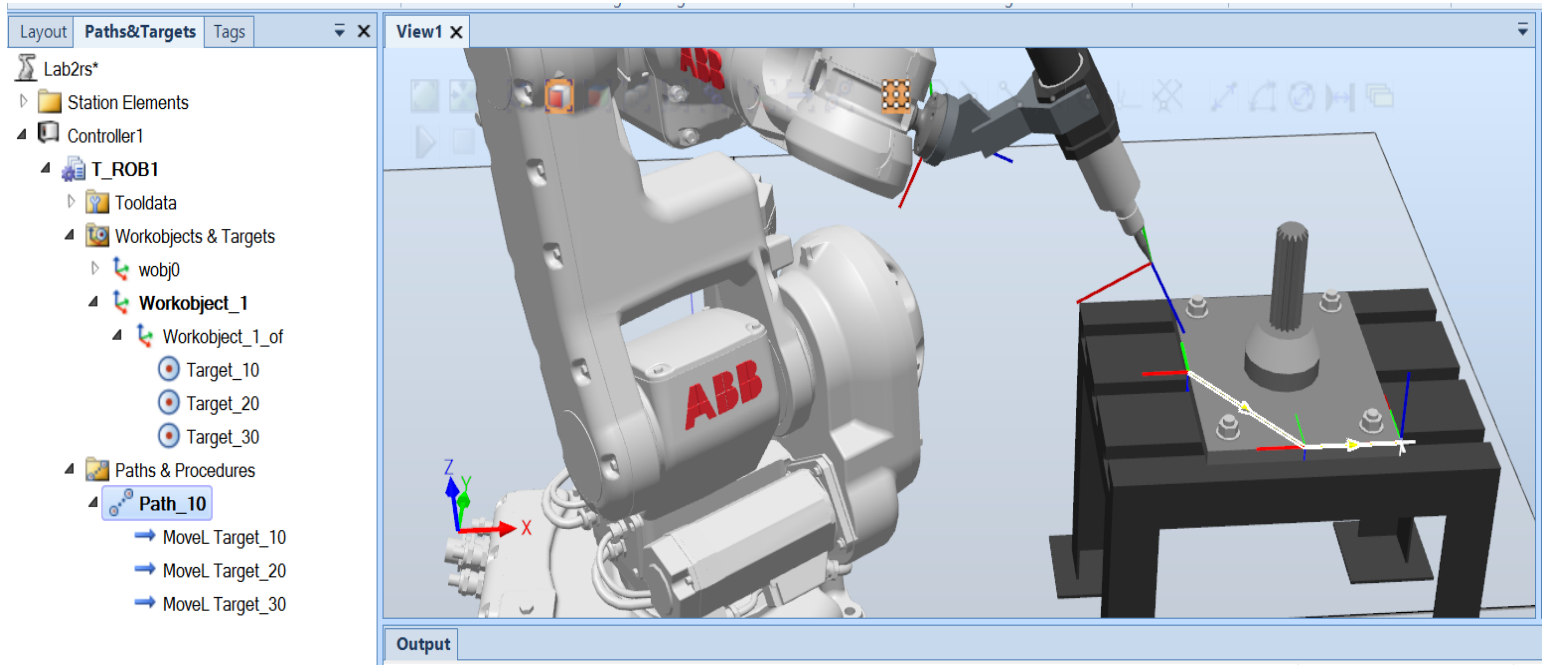
Elementi di un sistema di pianificazione e controllo del movimento

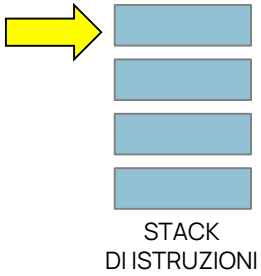


- **Stack di istruzioni**: elenco di istruzioni da eseguire, specificato utilizzando il linguaggio di programmazione proprietario
- **Generazione traiettoria**: converte un'istruzione in una traiettoria da eseguire
- **Cinematica inversa**: mappa la traiettoria dallo spazio cartesiano allo spazio dei giunti (se necessario)
- **Controllore d'asse**: chiude l'anello di controllo garantendo prestazioni di inseguimento

Programmazione del moto

- Per far compiere dei movimenti appropriati a un manipolatore, occorre istruirlo.
- Questo lo si fa con opportuni comandi che inducono il robot a portarsi successivamente nei punti che corrispondono all'esecuzione del compito desiderato.
- Un programma robot è una **sequenza di comandi di movimento attraverso i successivi punti** («target»)





Programmazione per insegnamento

Una prima modalità di programmazione del moto per un robot è la programmazione cosiddetta **teaching-by-showing**.

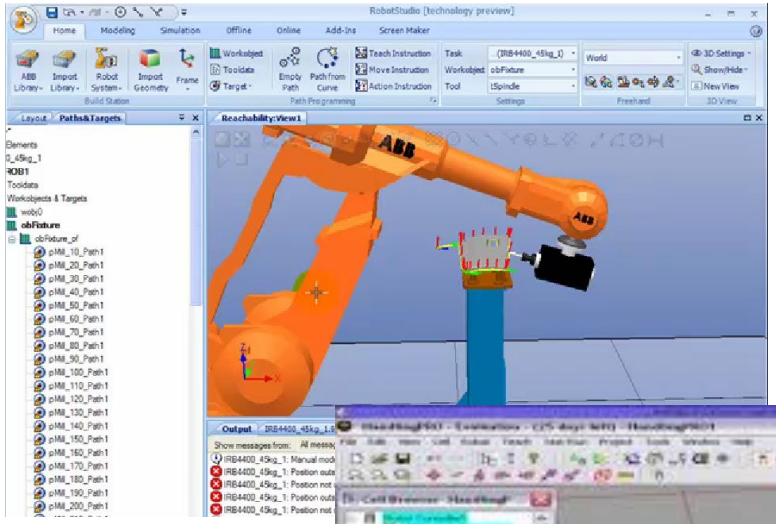
L'operatore con il teach pendant muove il manipolatore lungo il percorso desiderato. I trasduttori di posizione memorizzano le posizioni che il robot deve raggiungere, che diventano i punti target delle istruzioni di movimento.
Il robot sarà quindi in grado di ripetere autonomamente il movimento insegnato con apprendimento sul campo.

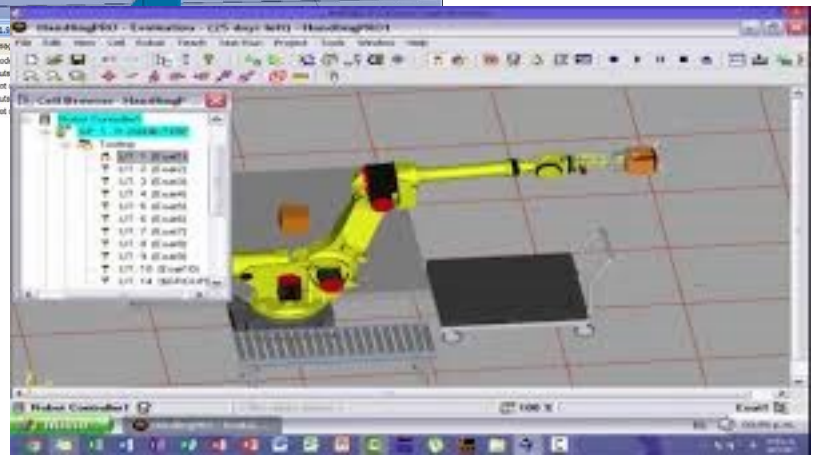
Si tratta di una modalità di programmazione molto semplice, ma la messa a punto del programma richiede che l'operatore abbia a disposizione il robot (che quindi non è operativo).

Ambienti di programmazione

- La generazione del programma robot può essere eseguita anche in un ambiente di programmazione robot. Il programmatore può movimentare il robot in un ambiente virtuale con rendering ad alta fedeltà del movimento del robot nella cella robotica.
- È possibile registrare le posizioni e far muovere il robot lungo un percorso formato da tali posizioni
- L'ambiente di programmazione robot produce il codice pronto per essere scaricato nel controller del robot.

ABB RobotStudio





FANUC RoboGuide

Linguaggi di programmazione

Il programmatore del robot può anche scrivere il programma robot direttamente utilizzando un **linguaggio di programmazione** robotico.

Con un linguaggio di programmazione l'operatore può programmare il movimento del robot e gestire operazioni complesse in cui il robot, all'interno di una cella di lavoro, interagisce con altre macchine e dispositivi. In aggiunta a un linguaggio di programmazione generico, il linguaggio fornisce **funzionalità specifiche orientate ai robot**.

Tutti i robot delle diverse case hanno il proprio linguaggio di programmazione (RAPID per ABB, PDL2 per COMAU, KRL per KUKA, KAREL per Fanuc, AS per Kawasaki). Noi discuteremo alcuni elementi del linguaggio di programmazione **RAPID** di **ABB**.

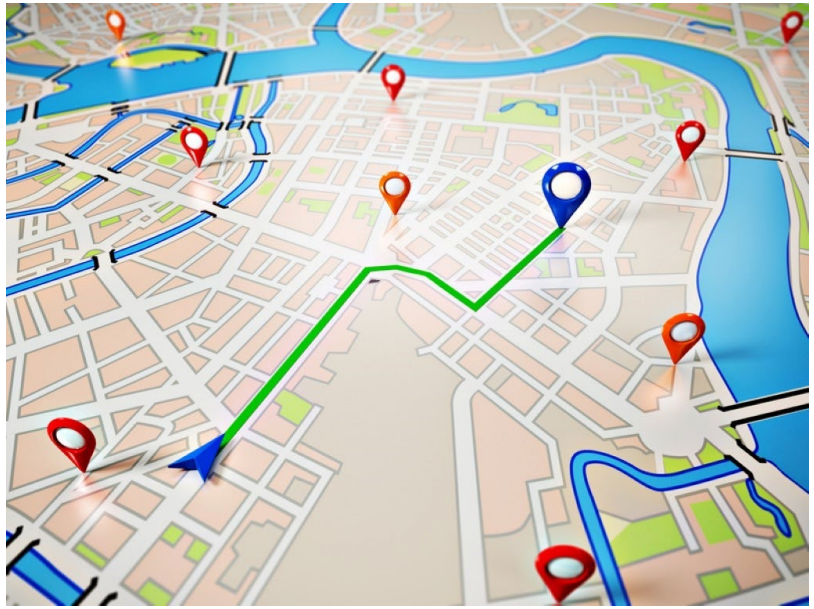
Prima però occorre fare alcune precisazioni.

Definizioni

- **Percorso**: è un concetto geometrico e si riferisce a una linea in un certo spazio (tipicamente lo spazio 3D delle posizioni cartesiane) che l'oggetto il cui moto viene pianificato deve seguire

- **Legge oraria**: è la dipendenza dal tempo con cui si vuole che l'oggetto viaggi lungo il percorso assegnato

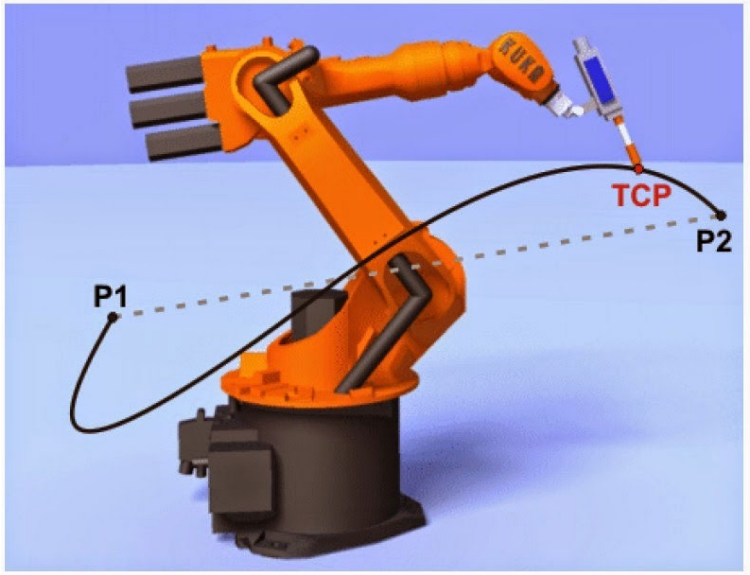
- **Traiettoria**: è un percorso su cui è stata assegnata una legge oraria



Traiettorie nello spazio operativo

Traiettorie nello **spazio operativo**: si definisce il movimento del robot in termini di moto (posizione e orientamento) dell'end-effector.
Occorre specificare:

- il punto terminale del movimento
- il percorso che l'end effector deve seguire



- la descrizione dei compiti del robot è naturale
- si può tenere conto di vincoli (ostacoli) sul percorso
- è necessaria l'inversione cinematica online
- punti di singolarità o gradi di libertà ridondanti generano problemi

Traiettorie nello spazio dei giunti

Traiettorie nello **spazio dei giunti**: il movimento è specificato in termini di posizioni di giunto desiderate.
Occorre specificare:

- il punto terminale del movimento

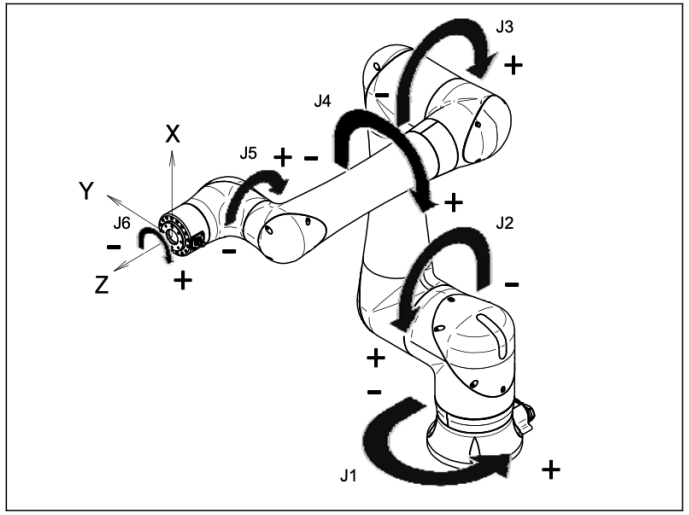


Fig. 3.1 (b) Each axes coordinates and mechanical interface coordinates

- l'inversione cinematica online non è necessaria
- non vi sono problemi legati alle singolarità cinematiche e ai gradi di libertà ridondanti
- il movimento risultante dell'end effector può essere difficile da prevedere
- è una modalità di interesse quando vogliamo solo portare il robot in una posizione finale e non siamo interessati al moto dell'end effector

Programmazione del robot

02

Esempio di programma (ABB RAPID)

```
PROC main()

    MoveAbsJ jHome, v200, z50, tVentosa;
    MoveL Offs(tPresa, 0, 0, 100), v200, z50, tVentosa;
    MoveL tPresa, v100, fine, tVentosa;
    WaitRob \InPos;
    Set DO_Ventosa;
    WaitTime 0.5;
    MoveL Offs(tPresa, 0, 0, 100), v100, z50, tVentosa;
    MoveL Offs(tRilascio, 0, 0, 100), v100, z50, tVentosa;
    MoveL tRilascio, v100, fine, tVentosa;
    WaitRob \InPos;
    Reset DO_Ventosa;
    WaitTime 0.5;
    MoveL Offs(tRilascio, 0, 0, 100), v100, z50, tVentosa;

ENDPROC
```

Esempio di programma (ABB RAPID)

Oltre agli abituali tipi di dati di un qualsiasi linguaggio di programmazione (bool, string, num, …), in RAPID sono definite alcune classi specifiche per applicazioni robotiche.

Tra queste:

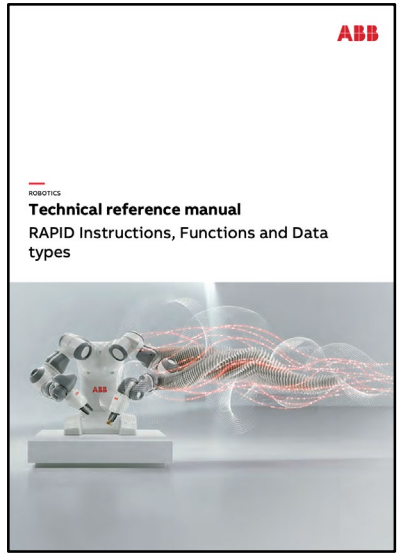
pos: rappresentazione di un vettore di posizione attraverso tre componenti

orient: quattro componenti di orientamento (quaternione)

pose: posa, costituita da una posizione e un orientamento

robtarget: come per pose ma con l'aggiunta di una stringa di configurazione (che indica se la configurazione è di spalla/gomito/polso alto o basso)

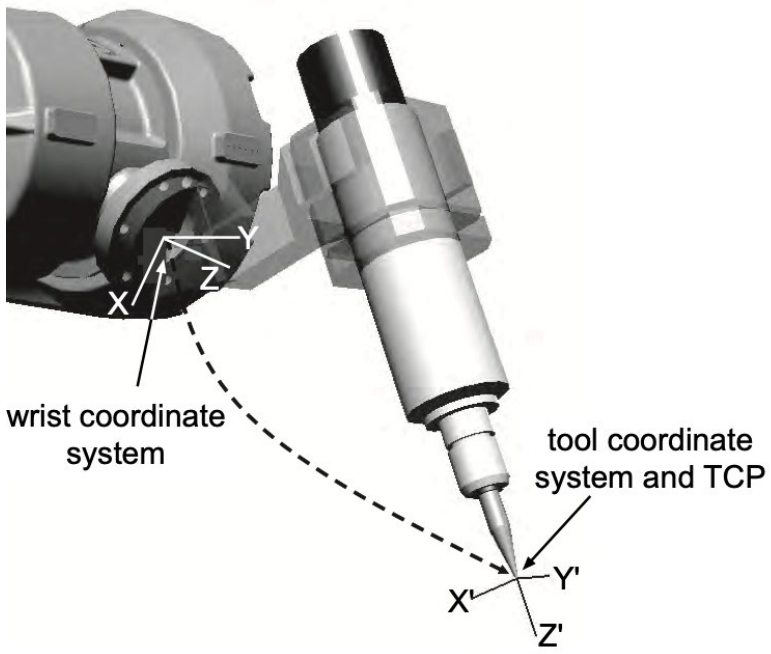
jointtarget: posizioni dei giunti, misurate in gradi

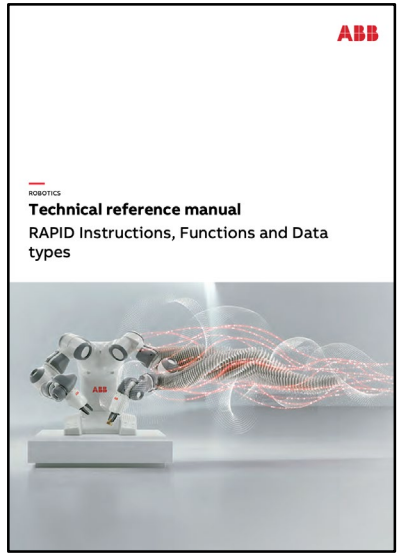


Esempio di programma (ABB RAPID)

In RAPID per ogni manipolatore viene predefinita una **terna di riferimento universale.** L'operatore può ridefinire la **terna di riferimento di base** relativamente alla terna universale. Questo è utile qualora si debba riposizionare il robot nell'area di lavoro, perché evita di ricalcolare tutte le posizioni.

Inoltre il programmatore può definire una terna o più terne relativamente alle corrispondenti **terne utensile**, utili quando viene cambiato l'attrezzo montato sulla flangia oppure il manipolatore monta diversi utensili contemporaneamente.





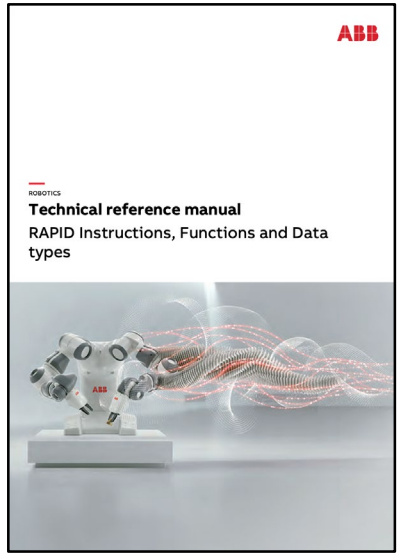
Le istruzioni MOVE

Con le istruzioni di movimentazione vengono impartiti i **comandi per il moto** dei bracci. Ad esempio, con:

```
MoveAbsJ ToJointPos Speed Zone;
```

La posizione da raggiungere (`ToJointPos`) è specificata mediante **jointtarget** (spazio dei giunti) e l'interpolazione avviene nello spazio dei giunti.

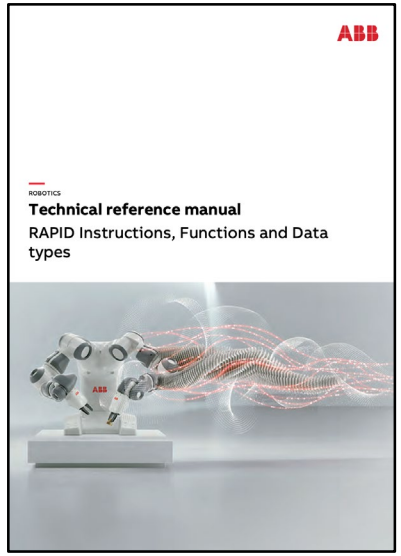
I parametri `Speed` e `Zone` verranno discussi più avanti.



Le istruzioni MOVE

Altre istruzioni:

```
MoveJ ToPoint Speed Zone Tool [\WObj];
MoveL ToPoint Speed Zone Tool [\WObj];
MoveC CirPoint ToPoint Speed Zone Tool [\WObj];
```



La posizione da raggiungere (`ToPoint` o `CirPoint`) è specificata mediante **robtarget** (spazio operativo) e il percorso è interpolato rispettivamente

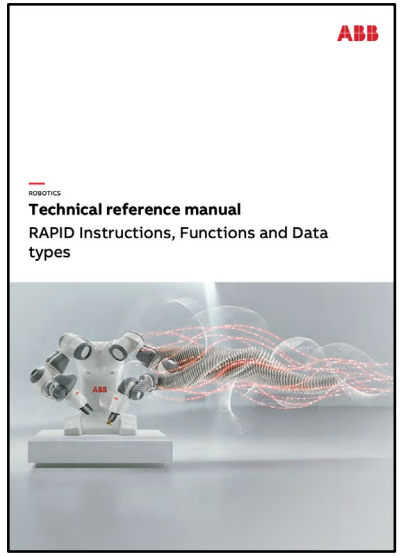
- nello spazio dei giunti (J = joint)
- nello spazio cartesiano tramite segment (L = linear)
- nello spazio cartesiano tramite arco di circonferenza (C = circular)

Il parametro `Tool` indica quale **terna utensile** deve raggiungere le posizioni specificate. Il parametro (opzionale) `WObj` indica la terna di riferimento (se non specificato, coincide con la terna di riferimento assoluta) nella quale sono definite posizione e orientamento da raggiungere.

Le istruzioni MOVE

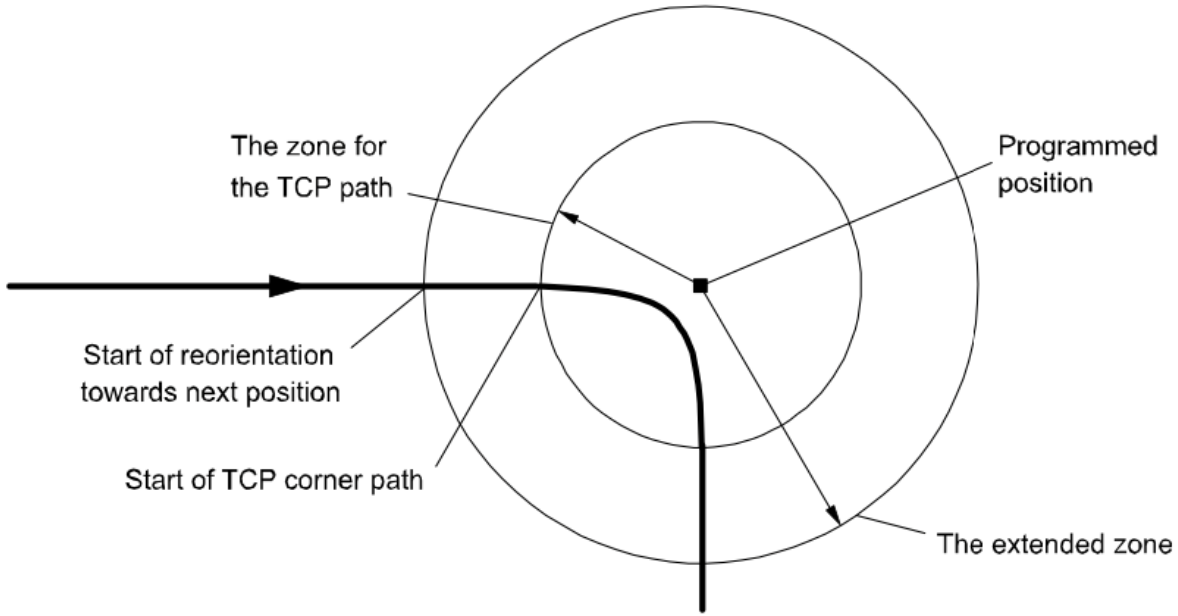
Il parametro `Speed` definisce la velocità lineare del TCP e quella di riorientamento dell'utensile. Nel seguito una tabella di alcuni valori preimpostati:

Name	TCP speed	Orientation
v5	5 mm/s	500°/s
v10	10 mm/s	500°/s
v20	20 mm/s	500°/s
v30	30 mm/s	500°/s
v40	40 mm/	
		°/s
	1500 mm/s	500°/s
v2000	2000 mm/s	500°/s
v2500	2500 mm/s	500°/s

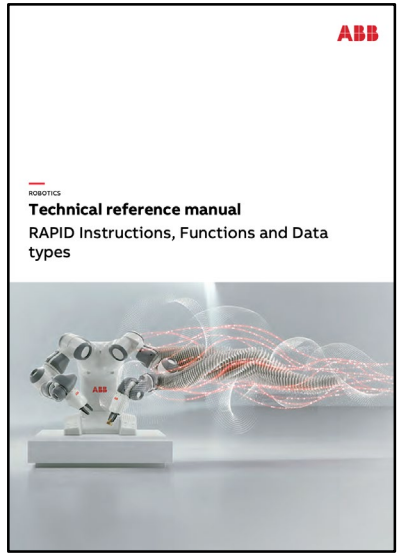


Le istruzioni MOVE

Il parametro `Zone` definisce se il movimento sarà seguito da un altro movimento. In questo caso, il braccio non si arresterà alla prima destinazione, ma si sposterà dal punto di partenza del primo movimento fino al punto finale del secondo, senza fermarsi sul punto comune ai due movimenti. Nel seguito una tabella di alcuni valori preimpostati:



Name	TCP path
z0	0.3 mm
z1	1 mm
z5	5 mm
z100	100 mm
z150	150 mm
z200	200 mm



Se necessario fermarsi, si usa il valore preimpostato `fine`.

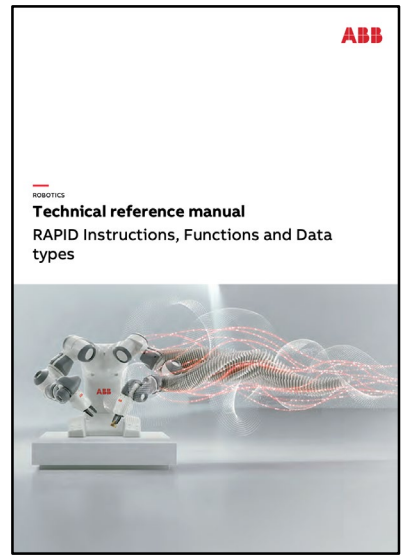
Offsets

La funzione

```
Offs(Point XOffset YOffset ZOffset)
```

viene utilizzata per aggiungere un offset nel sistema di coordinate dell'oggetto alla posizione del robot.
È particolarmente utile per definire posizione di approccio relative ad una posizione finale da raggiungere oppure per definire posizioni multiple:

```
PROC palletize(robtarget palettpos, num row, num column, num dist)
   rilascio := Offs(palettpos, (row-1)*dist, (column-1)*dist, 0);
   MoveL Offs(rilascio, 0, 0, 200), v100, z50, tool0;
   MoveL rilascio, v100, fine, tool0;
   WaitRob \InPos;
   Reset DO_Ventosa;
   WaitTime 0.5;
   MoveL Offs(rilascio, 0, 0, 200), v100, z50, tool0;
ENDPROC
```



Esempio di programma (ABB RAPID)

```
PROC main()

    MoveAbsJ jHome, v200, z50, tVentosa;
    MoveL Offs(tPresa, 0, 0, 100), v200, z50, tVentosa;
    MoveL tPresa, v100, fine, tVentosa;
    WaitRob \InPos;
    Set DO_Ventosa;
    WaitTime 0.5;
    MoveL Offs(tPresa, 0, 0, 100), v100, z50, tVentosa;
    MoveL Offs(tRilascio, 0, 0, 100), v100, z50, tVentosa;
    MoveL tRilascio, v100, fine, tVentosa;
    WaitRob \InPos;
    Reset DO_Ventosa;
    WaitTime 0.5;
    MoveL Offs(tRilascio, 0, 0, 100), v100, z50, tVentosa;

ENDPROC
```

POLITECNICO MILANO 1863 | DIPARTIMENTO DI ELETTRONICA INFORMAZIONE E BIOINGEGNERIA

Contatti

Paolo Rocco
paolo.rocco@polimi.it