

HANDBOOK

PROGRAMMAZIONE TEACH PENDANT

FANUC ER-4iA — Controller R-30iB Mate

Prof. D.Bertolino — www.handsonstem.it

Classi III, IV, V — Robotica Industriale

Guida completa alla sintassi TP, regole di programmazione, esempi commentati e best practices

Indice

1. Struttura di un programma TP
2. Comandi di movimento: J, L, C
3. Velocità e terminazione
4. I/O digitali: DO e DI
5. Temporizzazioni: WAIT
6. Registri numerici: R[]
7. Logica condizionale: IF, JMP, LBL
8. Sottoprogrammi: CALL
9. Commenti
10. Tabella riassuntiva sintassi
11. Programmi esempio commentati
12. Errori comuni e soluzioni
13. Best practices
14. Formato file .LS

1. Struttura di un programma TP

Ogni programma FANUC TP segue una struttura precisa. L'intestazione dichiara il nome del programma, il corpo contiene le istruzioni numerate, e il terminatore chiude il programma.

Schema base

```

/PROG NOME_PROGRAMMA
    → Intestazione: nome max 8 caratteri, maiuscolo, no spazi
1: J P[1] 100% FINE
    → Prima istruzione: movimento Joint al punto 1
2: L P[2] 250mm/sec FINE
    → Seconda istruzione: movimento Lineare
...
/END
    → Terminatore obbligatorio

```

Regole fondamentali

- Il nome del programma segue /PROG, massimo 8 caratteri alfanumerici e underscore.
- Ogni istruzione è preceduta dal numero di riga seguito da due punti (es. 1: 2: 3:).
- Il programma termina sempre con /END su una riga separata.
- Le righe che iniziano con ! sono commenti e vengono ignorate durante l'esecuzione.
- La numerazione delle righe deve essere crescente ma non necessariamente consecutiva.

⚠ Il controller esegue le istruzioni nell'ordine di numerazione. Se manca /END, il programma non verrà compilato.

2. Comandi di movimento: J, L, C

I comandi di movimento sono il cuore della programmazione TP. Definiscono come il robot si sposta da una posizione all'altra.

2.1 Movimento Joint (J)

Il movimento J (Joint) muove tutti gli assi simultaneamente per raggiungere la posizione target nel minor tempo possibile. La traiettoria del TCP nello spazio cartesiano NON è prevedibile (può essere curva).

N: J P[n] vel% terminazione

```
1: J P[1] 100% FINE
    → Movimento Joint a P[1] al 100% della velocità, arresto preciso
2: J P[2] 50% CNT100
    → Joint a P[2] al 50%, passaggio continuo (non si ferma)
```

Quando usare J: movimenti lunghi, avvicinamenti, ritorni a home, quando la traiettoria non è critica. È il tipo di movimento più veloce.

✅ Usa J per spostamenti "in aria" dove non importa la traiettoria esatta.

2.2 Movimento Lineare (L)

Il movimento L (Linear) muove il TCP in linea retta nello spazio cartesiano. La velocità è espressa in mm/sec. Richiede più calcolo al controller rispetto al J.

N: L P[n] vel mm/sec terminazione

```
3: L P[3] 200mm/sec FINE
    → Lineare a P[3] a 200mm/sec, arresto preciso
4: L P[4] 500mm/sec CNT50
    → Lineare a P[4] a 500mm/sec, smussatura 50%
```

Quando usare L: operazioni di lavorazione (saldatura, incollaggio, taglio), avvicinamento/allontanamento dal pezzo, qualsiasi operazione dove il percorso deve essere rettilineo.

⚠ Il movimento L può generare errore MOTN-023 (singolarità) se il TCP passa vicino all'asse J5≈0°. Evita traiettorie lineari che passano sopra la base del robot.

2.3 Movimento Circolare (C)

Il movimento C (Circular) muove il TCP lungo un arco di cerchio definito da 3 punti: la posizione corrente (inizio), un punto di passaggio (via) e la destinazione (fine).

N: C P[via] P[dest] vel mm/sec terminazione

```
5: C P[5] P[6] 150mm/sec FINE
```

→ Arco da posizione corrente, passando per P[5], fino a P[6]

Quando usare C: saldatura ad arco su profili curvi, smussatura angoli, movimenti di avvicinamento curvi per evitare ostacoli.

Il comando C richiede che i 3 punti non siano collineari (allineati). Se lo sono, il controller fallback su un movimento lineare.

✓ *Per registrare un arco: prima posiziona il robot al punto di partenza, poi usa TEACH C per il punto via, muovi il robot e usa di nuovo TEACH C per il punto finale.*

3. Velocità e terminazione

3.1 Velocità

La velocità si esprime in due modi diversi a seconda del tipo di movimento:

Movimento J: percentuale della velocità massima (1% – 100%). Es: 50% = metà della velocità massima di ciascun asse.

Movimento L e C: millimetri al secondo del TCP. Es: 250mm/sec = il TCP si muove a 250mm al secondo in linea retta.

La velocità effettiva è sempre moltiplicata per l'Override (regolabile dall'operatore, 1–100%). Se il programma dice 100% e l'override è al 50%, la velocità reale è il 50%.

 *In modalità T1 (teach), la velocità massima del TCP è limitata a 250mm/sec indipendentemente da quanto scritto nel programma.*

3.2 Terminazione

La terminazione definisce il comportamento del robot quando raggiunge il punto target:

FINE: il robot si ferma completamente sul punto prima di eseguire l'istruzione successiva. Massima precisione, ma rallenta il ciclo.

CNTn: (Continuous, n = 0–100) il robot "taglia" la curva passando vicino al punto senza fermarsi. Più alto il valore, più lontano dal punto passa.

```
1: L P[1] 200mm/sec FINE
   → Si ferma esattamente su P[1]
2: L P[2] 200mm/sec CNT100
   → Passa vicino a P[2] senza fermarsi, massima smussatura
3: L P[3] 200mm/sec CNT0
   → Passa molto vicino a P[3], quasi come FINE ma senza decelerare a zero
```

 *Usa FINE per prelievo e deposito pezzi (precisione necessaria). Usa CNT50–100 per movimenti intermedi per ridurre il tempo ciclo.*

4. I/O digitali: DO e DI

Gli I/O digitali permettono al robot di comunicare con il mondo esterno: pinze, sensori, PLC, segnali di stato.

4.1 Uscite digitali (DO)

```
N: DO[n]=ON
N: DO[n]=OFF
```

```
3: DO[1]=ON
    → Attiva l'uscita 1 (es. chiude la pinza pneumatica)
8: DO[1]=OFF
    → Disattiva l'uscita 1 (es. apre la pinza)
```

Le uscite digitali sono tipicamente collegate a: pinze pneumatiche, elettrovalvole, segnali al PLC, luci di stato, attivazione utensili.

4.2 Ingressi digitali (DI)

Gli ingressi digitali si leggono con il comando WAIT o IF:

```
2: WAIT DI[1]=ON
    → Attende che l'ingresso 1 diventi ON (es. sensore pezzo presente)
5: IF DI[2]=ON, JMP LBL[3]
    → Se ingresso 2 è ON, salta a LBL[3]
```

4.3 Convenzioni di assegnazione I/O

Una buona pratica è documentare l'assegnazione degli I/O all'inizio del programma con commenti:

```
! DO[1] = Pinza ON/OFF
! DO[7] = Segnale ciclo completato
! DO[8] = Segnale robot pronto
! DI[1] = Sensore pezzo presente
! DI[2] = Sensore qualità OK
! DI[8] = PLC Start ciclo
```

5. Temporizzazioni: WAIT

Il comando WAIT mette in pausa l'esecuzione del programma per un tempo specificato o fino al verificarsi di una condizione.

5.1 WAIT temporizzato

N: WAIT tempo(sec)

4: WAIT 0.3(sec)

→ Attende 300 millisecondi (tempo di chiusura pinza)

7: WAIT 1.5(sec)

→ Attende 1.5 secondi

5.2 WAIT condizionale

N: WAIT DI[n]=ON|OFF

2: WAIT DI[8]=ON

→ Attende che il PLC dia il segnale di start

 **Il WAIT condizionale blocca il programma indefinitamente se la condizione non si verifica mai. In produzione, aggiungi sempre un timeout o un controllo alternativo.**

 **Dopo un DO[n]=ON per la pinza, aggiungi sempre un WAIT 0.2–0.5(sec) per dare tempo al dispositivo di attivarsi fisicamente.**

6. Registri numerici: R[]

I registri R[1]–R[200] sono variabili numeriche che il programma può leggere, scrivere e modificare. Sono persistenti: mantengono il valore tra un'esecuzione e l'altra.

6.1 Assegnazione

```

N: R[n]=valore
N: R[n]=R[m]+valore
N: R[n]=R[m]-valore
N: R[n]=R[m]*valore

```

```

1: R[1]=0
   → Azzera il registro 1 (inizializza contatore)
12: R[1]=R[1]+1
   → Incrementa R[1] di 1 (contatore pezzi)
5: R[3]=R[2]*10
   → R[3] = R[2] moltiplicato per 10

```

6.2 Uso tipico

I registri si usano come contatori di ciclo, accumulatori, indici per posizioni, conteggio pezzi buoni/scarti, tempi ciclo.

✓ Inizializza sempre i registri a inizio programma con $R[n]=0$. Non dare per scontato che siano a zero.

7. Logica condizionale: IF, JMP, LBL

La logica condizionale permette di creare cicli, biforcazioni e decisioni nel programma.

7.1 Etichette (LBL)

N: LBL[n]

Un'etichetta è un segnaposto nel programma. Non esegue nessuna azione, ma serve come destinazione per i salti.

7.2 Salto incondizionato (JMP)

N: JMP LBL[n]

14: JMP LBL[1]

→ Salta a LBL[1] (ripete il ciclo)

⚠ *Un JMP senza condizione di uscita crea un ciclo infinito. Assicurati di avere sempre un IF che porta a un LBL di uscita.*

7.3 Salto condizionale (IF)

N: IF R[n] operatore valore, JMP LBL[m]

N: IF DI[n]=ON|OFF, JMP LBL[m]

Operatori disponibili: >= <= > < = !=

13: IF R[1]>=5, JMP LBL[2]

→ Se R[1] è maggiore o uguale a 5, vai a LBL[2]

10: IF DI[2]=ON, JMP LBL[3]

→ Se l'ingresso 2 è ON, vai a LBL[3]

7.4 Pattern: ciclo FOR con contatore

FANUC TP non ha un vero ciclo FOR, ma si simula con R[] + LBL + IF + JMP:

1: R[1]=0

→ Inizializza contatore

2: LBL[1]

→ Inizio ciclo

3: ... istruzioni del ciclo ...

10: R[1]=R[1]+1

→ Incrementa contatore

11: IF R[1]>=5, JMP LBL[2]

→ Se 5 cicli fatti, esci

12: JMP LBL[1]

→ Altrimenti ripeti

13: LBL[2]

→ *Fine ciclo*

8. Sottoprogrammi: CALL

N: CALL NOME_PROG

Il comando CALL esegue un altro programma TP come sottoprogramma. Al termine, l'esecuzione torna alla riga successiva del programma chiamante.

```
5: CALL PICK_UP
   → Esegue il programma PICK_UP, poi torna qui
6: CALL PUT_DOWN
   → Esegue PUT_DOWN
```

Vantaggi: riutilizzo del codice, programmi più leggibili, manutenzione facilitata. Un ciclo pick & place può essere un sottoprogramma chiamato dal main.

✅ *Tieni i sottoprogrammi brevi e con un singolo scopo. Nomina in modo descrittivo: PICK_UP, PUT_DOWN, HOME_POS, QC_CHECK.*

9. Commenti

```
! Testo del commento
```

Le righe che iniziano con ! sono commenti. Non vengono eseguite dal controller. Servono per documentare il programma.

```
! Programma: Pick & Place con conteggio  
! Autore: Classe IV - Gruppo A  
! Data: 2026-02-14  
! DO[1] = Pinza, DI[1] = Sensore
```

✅ *Inserisci sempre un blocco commenti in testa al programma con: nome, autore, data, mappa I/O.
Commenta anche le sezioni critiche del codice.*

10. Tabella riassuntiva sintassi

COMANDO	SINTASSI	DESCRIZIONE
J	N: J P[n] vel% FINE CNTn	Movimento Joint (spazio giunti)
L	N: L P[n] vel mm/sec FINE CNTn	Movimento Lineare (linea retta)
C	N: C P[via] P[end] vel mm/sec FINE	Movimento Circolare (arco)
DO	N: DO[n]=ON OFF	Imposta uscita digitale
WAIT	N: WAIT tempo(sec)	Pausa temporizzata
WAIT DI	N: WAIT DI[n]=ON OFF	Attende condizione ingresso
R[]=	N: R[n]=valore R[m]±val	Assegnazione registro
IF	N: IF cond, JMP LBL[n]	Salto condizionale
JMP	N: JMP LBL[n]	Salto incondizionato
LBL	N: LBL[n]	Etichetta (destinazione salto)
CALL	N: CALL NOME	Chiamata sottoprogramma
!	! testo	Commento (non eseguito)
/PROG	/PROG NOME	Intestazione programma
/END	/END	Fine programma

11. Programmi esempio commentati

11.1 Pick & Place base

Il ciclo pick & place è l'operazione fondamentale: prelevare un pezzo da una posizione e depositarlo in un'altra.

```

/PROG PICK_PLACE
! Ciclo base pick & place
! DO[1] = Pinza pneumatica
1: J P[1] 100% FINE
    → Vai a posizione Home (sicura, sopra l'area di lavoro)
2: L P[2] 250mm/sec FINE
    → Avvicinamento lineare al pezzo (preciso)
3: DO[1]=ON
    → Chiudi pinza (afferra pezzo)
4: WAIT 0.3(sec)
    → Attendi chiusura completa pinza
5: L P[3] 200mm/sec CNT50
    → Solleva pezzo (smussatura: non serve fermarsi)
6: J P[4] 80% FINE
    → Spostamento rapido verso zona deposito
7: L P[5] 100mm/sec FINE
    → Deposito preciso (FINE: fermata esatta)
8: DO[1]=OFF
    → Apri pinza (rilascia pezzo)
9: WAIT 0.3(sec)
    → Attendi apertura
10: J P[1] 100% FINE
    → Ritorno a Home
/END

```

11.2 Ciclo con contatore

Ripete il pick & place per un numero definito di volte usando un registro come contatore.

```

/PROG CYCLE_5X
! Esegui 5 cicli pick & place
1: R[1]=0
    → Azzerà contatore pezzi
2: LBL[1]
    → Inizio ciclo
3: J P[1] 100% FINE
4: L P[2] 250mm/sec FINE
5: DO[1]=ON
6: WAIT 0.3(sec)
7: L P[3] 200mm/sec CNT50
8: J P[4] 80% FINE
9: L P[5] 100mm/sec FINE
10: DO[1]=OFF
11: WAIT 0.3(sec)

```

```

12: R[1]=R[1]+1
    → Incrementa contatore
13: IF R[1]>=5, JMP LBL[2]
    → Se 5 pezzi fatti, esci dal ciclo
14: JMP LBL[1]
    → Altrimenti ripeti
15: LBL[2]
    → Fine ciclo
16: J P[1] 100% FINE
    → Ritorno finale a Home
/END

```

11.3 Handshake con PLC

Protocollo di comunicazione robot-PLC tramite segnali digitali. Il PLC governa quando il robot può partire.

```

/PROG PLC_SYNC
! Protocollo handshake PLC
! DO[8] = Robot pronto
! DI[8] = PLC Start
! DO[7] = Ciclo completato
1: DO[8]=ON
    → Segnala al PLC: robot pronto
2: WAIT DI[8]=ON
    → Attendi che il PLC dia il via
3: DO[8]=OFF
    → Togli segnale pronto (sto lavorando)
4: J P[1] 100% FINE
... ciclo di lavoro ...
12: J P[1] 100% FINE
13: DO[7]=ON
    → Segnala al PLC: ciclo completato
14: WAIT 0.5(sec)
15: DO[7]=OFF
    → Reset segnale completamento
/END

```

11.4 Smistamento con controllo qualità

Il robot smista pezzi buoni e scarti in base al segnale di un sensore collegato a DI[2].

```

/PROG QC_SORT
! R[1] = contatore pezzi OK
! R[2] = contatore scarti
! DI[1] = pezzo presente
! DI[2] = qualità OK
1: R[1]=0
2: R[2]=0
3: LBL[1]
4: J P[1] 100% FINE
    → Posizione di attesa
5: WAIT DI[1]=ON

```

```

        → Attendi pezzo sul nastro
6: L P[2] 200mm/sec FINE
        → Preleva
7: DO[1]=ON
8: WAIT 0.3(sec)
9: L P[3] 300mm/sec CNT50
        → Sollewa
10: IF DI[2]=ON, JMP LBL[3]
        → Se qualità OK, vai a deposito OK
11: J P[4] 80% FINE
        → Deposito SCARTI
12: R[2]=R[2]+1
13: JMP LBL[4]
14: LBL[3]
        → Deposito pezzi OK
15: J P[5] 80% FINE
16: R[1]=R[1]+1
17: LBL[4]
18: L P[6] 100mm/sec FINE
19: DO[1]=OFF
20: WAIT 0.3(sec)
21: JMP LBL[1]
        → Ripeti (ciclo infinito, fermato dal PLC)
/END

```

12. Errori comuni e soluzioni

MOTN-023: Singolarità

Causa: il TCP passa vicino a $J5=0^\circ$ durante un movimento L. Soluzione: evita traiettorie lineari che passano sopra l'asse J1. Usa J per attraversare queste zone.

MOTN-019: Fuori portata

Causa: il punto target è oltre lo spazio di lavoro (reach 550mm). Soluzione: verifica le coordinate. Riposiziona il pezzo più vicino alla base.

Programma non parte

Cause possibili: manca /END, errore di sintassi nelle righe, punto P[n] non registrato. Soluzione: controlla la console errori, verifica tutti i punti.

Robot si ferma durante CNT

Causa: il punto successivo non è stato calcolato in tempo (look-ahead). Soluzione: riduci la velocità o aumenta il valore CNT.

Pinza non si chiude

Causa: manca il WAIT dopo DO[1]=ON. Il robot parte prima che la pinza abbia chiuso. Soluzione: aggiungi sempre WAIT 0.2–0.5(sec) dopo ogni DO.

Ciclo infinito

Causa: JMP LBL[1] senza condizione IF di uscita, oppure il contatore non viene incrementato. Soluzione: verifica che $R[n]=R[n]+1$ sia presente e che IF $R[n]>=target$, JMP LBL[uscita] sia corretto.

13. Best practices

Struttura del programma

- Inizia sempre con un movimento J a Home: sicuro e ripetibile.
- Termina sempre con un ritorno a Home.
- Documenta il programma con commenti in testa: nome, autore, data, mappa I/O.
- Inizializza tutti i registri usati a inizio programma.

Movimenti

- Usa J per spostamenti lunghi e L per operazioni precise.
- Usa FINE per prelievo/deposito, CNT per movimenti intermedi.
- Testa sempre in T1 a bassa velocità prima di passare ad AUTO.
- Avvicinati al pezzo dall'alto con L, mai lateralmente.

I/O e sicurezza

- Aggiungi sempre WAIT dopo DO per dare tempo ai dispositivi.
- Usa handshake con il PLC: non assumere mai che un dispositivo esterno sia pronto.
- Prevedi sempre una posizione Home sicura raggiungibile da qualsiasi configurazione.

Testing

- Usa STEP per verificare ogni riga prima di RUN.
- Inizia sempre con override al 10–20%, poi aumenta gradualmente.
- Nel simulatore, attiva il TRAIL per visualizzare la traiettoria.
- Controlla i registri dopo ogni ciclo per verificare che i contatori funzionino.

14. Formato file .LS

Il file .LS (Listing) è il formato testuale dei programmi FANUC. Può essere editato con un editor di testo e caricato sul controller via USB o rete.

Struttura del file .LS

```

/ATTR
    → Attributi del programma (autore, data, versione)
OWNER = MNEDITOR;
FILE_NAME = PICK_PLACE;
...
/MN
    → Sezione istruzioni (corpo del programma)
1: J P[1] 100% FINE ;
    → Nota: ogni riga termina con punto e virgola
2: L P[2] 250mm/sec FINE ;
...
/END

/POS
    → Sezione posizioni (coordinate dei punti)
P[1]{
    GP1:
    UF : 0, UT : 1,
    X = 200.000 mm, Y = 0.000 mm, Z = 400.000 mm,
    W = 180.000 deg, P = 0.000 deg, R = 0.000 deg
};
/END

```

Il simulatore supporta l'import e l'export di file .LS tramite i pulsanti nella toolbar TP. L'export genera un file .LS completo con header, corpo e posizioni. L'import legge le sezioni /MN e /POS e ricostruisce il programma.

✓ *Puoi editare i file .LS con qualsiasi editor di testo (Notepad++, VS Code). Utile per modificare coordinate senza il teach pendant.*