ΕΙΣΑΓΩΓΙΚΕΣ ΕΝΝΟΙΕΣ ΕΡΓΑΣΤΗΡΙΟΥ ΡΟΜΠΟΤΙΚΗΣ

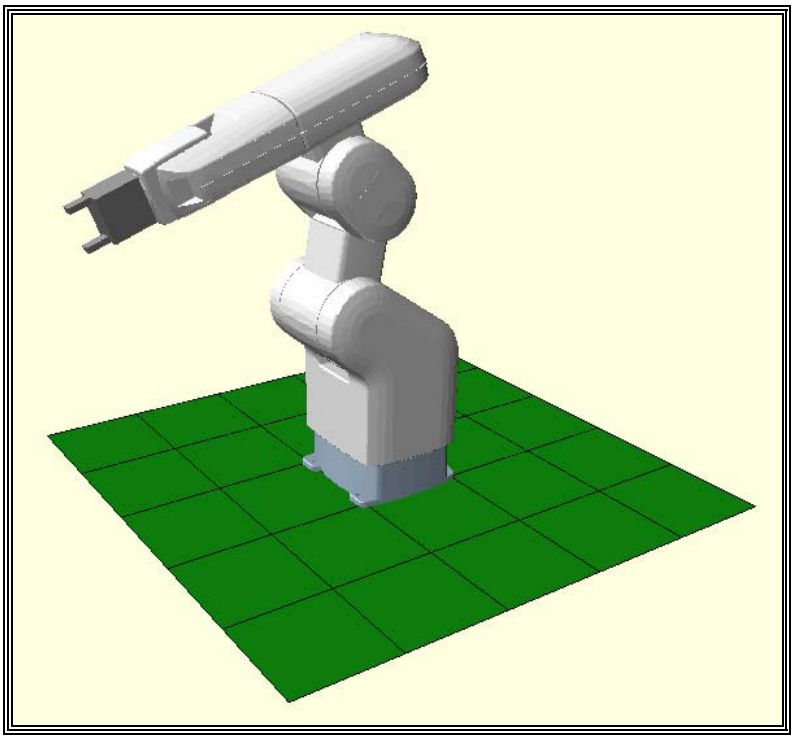
Σήμερα τα ρομπότ χρησιμοποιούνται κυρίως για να εκτελούν βαριές, ανιαρές ,επικίνδυνες και επαναλαμβανόμενες εργασίες, κυρίως στη βιομηχανία. Υπολογίζεται ότι το 90% των ρομπότ είναι εγκατεστημένα σε βιομηχανικούς χώρους. Για παράδειγμα, οι βραχίονες που συναρμολογούν και συγκολλούν τον σκελετό ενός αυτοκινήτου στη γραμμή παραγωγής.

Στόχους της εφαρμογής των ρομποτικών συστημάτων στη βιομηχανία αποτελούν η υποκατάσταση της χειρονακτικής εργασίας η βελτίωση και η σταθεροποίηση της ποιότητας και η αύξηση της παραγωγικότητας.

Άλλοι τομείς, όπου χρησιμοποιούνται ρομπότ, είναι Διαστημικά προγράμματα, Στρατιωτικές εφαρμογές, Ιατρικές εφαρμογές, Ερευνητικά Ιδρύματα και πολλοί άλλοι.

Πρωτοπόρες χώρες στην εγκατάσταση ρομποτικών συστημάτων αποτελούν η Ιαπωνία (352.200 βιομηχανικά ρομπότ στο τέλος του 2004), η Γερμανία (121.500), η Αμερική (121.300), η Ιταλία (53.100), η Γαλλία (28.400) και η Αγγλία (14.600). Συνολικά στην Ευρωπαϊκή Ένωση υπολογίζεται ότι στο τέλος του 2004 υπήρχαν 266.100 εγκατεστημένα ρομπότ στη βιομηχανία.

Τα τελευταία χρόνια έχουν αναπτυχθεί ρομπότ που προσπαθούν να μιμηθούν ανθρώπινα χαρακτηριστικά, όπως το περπάτημα και την όραση. Τα αποτελέσματα είναι ενθαρρυντικά και σίγουρα τις επόμενες δεκαετίες θα δούμε σημαντικότατα επιτεύγματα στον τομέα της Ρομποτικής.



Οργάνωση και λειτουργία

Τα στοιχεία που αποτελούν ένα Ρομποτικό Σύστημα είναι το Μηχανικό μέρος και ο Ελεγκτής. Στην ανάλυση μας, θα έχουμε ως παράδειγμα το ρομποτικό βραχίονα του εργαστηρίου (RV-2A), για να γίνουν πιο εύκολα κατανοητά τα μέρη που περιγράφουμε, χωρίς να χάνεται η γενικότητα.

Το μηχανικό μέρος στο ρομπότ του εργαστηρίου μας αποτελεί ο βραχίονας. Ο βραχίονας αποτελείται από:

- ***Αρθρώσεις*** : Οι αρθρώσεις είναι είτε γραμμικές που επιτρέπουν την κίνηση κατά μήκος ενός άξονα, είτε περιστροφικές, επιτρέπουν την κίνηση γύρω από τον άξονα τους. Παρατηρήστε ότι το RV-2A έχει έξι περιστροφικές αρθρώσεις.

- ***Κινητήρες*** : Κάθε άρθρωση χρειάζεται και από ένα κινητήρα όπως είναι φανερό. Ο κινητήρας μπορεί να είναι ηλεκτρικός (βηματικός ,σερβοκινητήρας), υδραυλικός ή πνευματικός.

- ***Αισθητήρια*** : Για να ελέγχουμε τη θέση του ρομπότ χρειαζόμαστε πληροφορίες για την θέση και την ταχύτητα της κάθε άρθρωσης. Έτσι χρησιμοποιούνται διάφοροι τύποι αισθητηρίων, όπως ποτενσιόμετρα , ταχύμετρα ή encoders (ψηφιακοί οπτικοί κωδικοποιητές θέσης).

  Στην περίπτωση του RV-2A κάθε άρθρωση έχει από ένα σερβοκινητήρα ψηφιακά ελεγχόμενο (PID έλεγχος-software). Το σύστημα ελέγχου θέσης κάθε άρθρωσης (γωνία στροφής - γωνιακή ταχύτητα κλπ.) αποτελείται από έναν υψηλής ακρίβειας ψηφιακό οπτικό κωδικοποιητή θέσης ο οποίος σε μία πλήρη περιστροφή της άρθρωσης αποδίδει **8192** παλμούς.

- ***Τελικό στοιχείο δράσης*** : όλοι οι βραχίονες έχουν προσαρμοσμένο στο άκρο τους ένα μηχανικό εξάρτημα κατάλληλα σχεδιασμένο και επιλεγμένο , προκειμένου να εκτελούν την εργασία για την οποία έχουν προγραμματιστεί, που μπορεί να είναι μια αρπάγη για τη μεταφορά αντικειμένων, συγκολλητήτης αντίστασης ,τόξου ,εργαλεία για διαφορες βιομηχανικές κατεργασίες (λείανση κοπή, τρύπημα, βαφή, συναρμολόγηση, παλετοποίηση κ.ο.κ). Το RV-2A έχει μια υποτυπώδη αρπάγη (βεντούζα) που λειτουργεί με κενό αέρος.

Ο ελεγκτής είναι η μονάδα που μας δίνει τη δυνατότητα να προγραμματίσουμε το ρομπότ και ο οποίος ελέγχει την κίνηση του και την εκτέλεση της εργασίας του. Αποτελείται από:

- ***Ηλεκτρονικά (Hardware)*** : Αυτά είναι συνήθως ένας υπολογιστής, όπου αποθηκεύεται το πρόγραμμα που θα εκτελεστεί. Τα ηλεκτρονικά επικοινωνίας (Interface), που χρησιμεύουν στην επικοινωνία του ελεγκτή με το μηχανικό μέρος του ρομπότ και το εξωτερικό περιβάλλον. Οι ενισχυτές ισχύος, που ενισχύουν τα σήματα ελέγχου στο επίπεδο που απαιτείται, ώστε οι κινητήρες να κινούν τις αρθρώσεις.

- ***Λογισμικό (Software)*** : Κυρίως το λογισμικό ευθύνεται για τη δημιουργία των κατάλληλων σημάτων ελέγχου, σύμφωνα με κάποιον αλγόριθμο, παίρνοντας υπόψη του το φορτίο, την ταχύτητα, την θέση του ρομπότ και άλλες μεταβλητές. Επίσης, στο λογισμικό περιλαμβάνεται και κάποιο βοηθητικό πρόγραμμα (για το RV-2A υπάρχει το περιβάλλον προγραμματισμού COSIROP) που επιτρέπει τον προγραμματισμό του ρομπότ σε μία γλώσσα υψηλού επίπεδου. Ακόμη φροντίζει για την παρακολούθηση της λειτουργίας του και την ενημέρωση του χρήστη.

Βαθμός Ελευθερίας

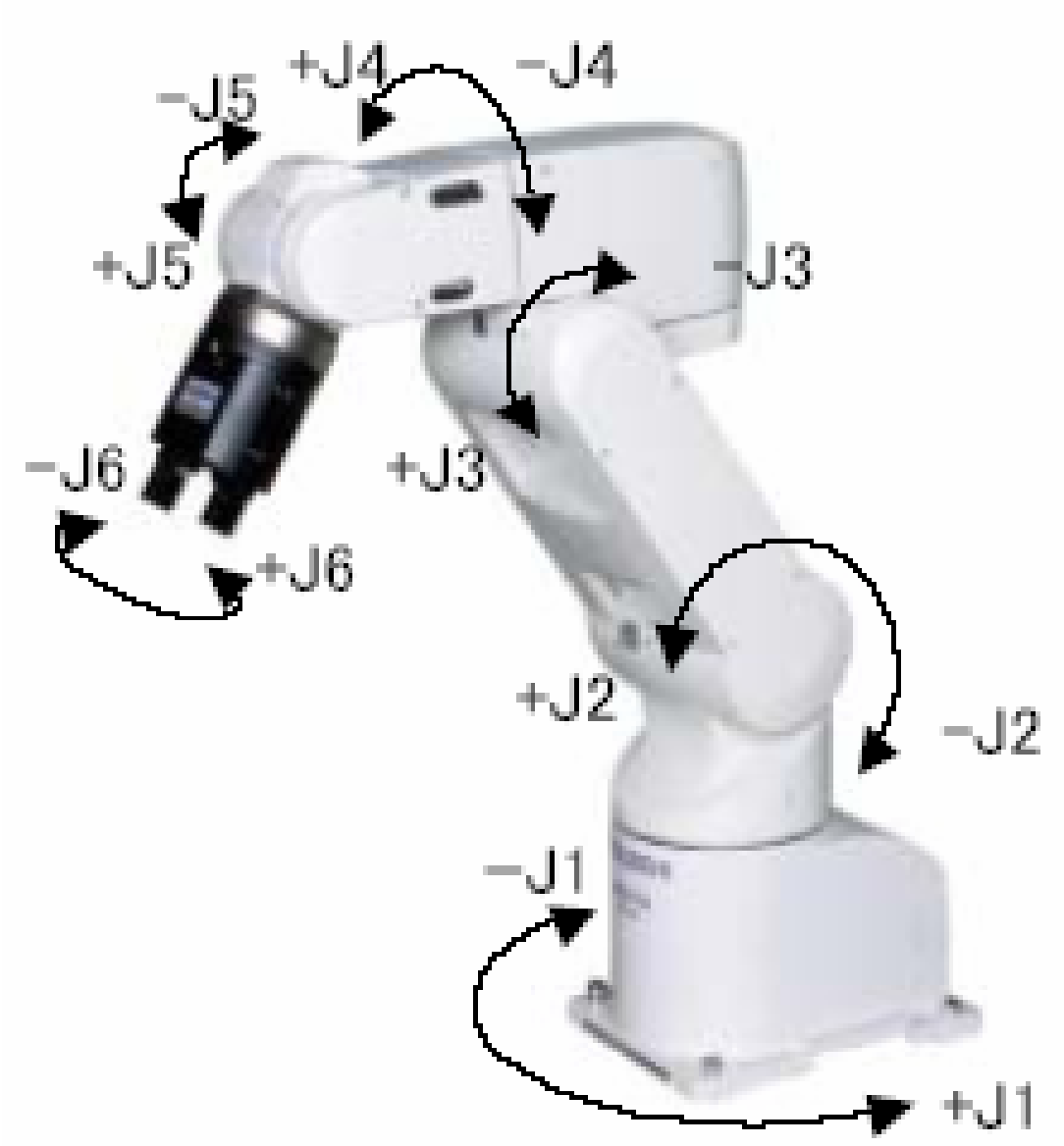
Βασικό γνώρισμα κάθε ρομπότ αποτελεί ο Βαθμός Ελευθερίας (Degree Of Freedom – DOF) του. Σε γενικές γραμμές δηλώνει το πόσο ευκίνητο είναι ένα ρομπότ στο χώρο. Συνήθως, κάθε ανεξάρτητα κινούμενη άρθρωση προσθέτει ένα βαθμό ελευθερίας στο ρομπότ.

Ο ορισμός λέει:
¨Ο αριθμός των ανεξάρτητων παραμέτρων, που προσδιορίζουν τη θέση ενός σώματος στο χώρο, ονομάζεται Βαθμός Ελευθερίας¨

Για να περιγράψουμε ακριβώς τη θέση ενός στερεού σώματος στο χώρο, χρειαζόμαστε 6 μεταβλητές, 3 για την θέση και 3 για τον προσανατολισμό του. Άρα, σύμφωνα με τον παραπάνω ορισμό, για να μπορεί ένα ρομπότ να κινηθεί οπουδήποτε στο χώρο με οποιοδήποτε προσανατολισμό, πρέπει να έχει τουλάχιστον 6 βαθμούς ελευθερίας. Ο ρομποτικός βραχίονας του εργαστηρίου RV-2A έχει 6 βαθμούς ελευθερίας. Δηλαδή, κάθε περιστροφική του άρθρωση προσφέρει από ένα βαθμό ελευθερίας.

Ο ανθρώπινος βραχίονας υπολογίζεται ότι έχει 7 βαθμούς ελευθερίας. Στα βιομηχανικά ρομπότ σπάνια συναντάμε πάνω από 6 βαθμούς ελευθερίας, αφού ναι μεν θα βελτιωνόταν η ευελιξία τους, αλλά θα γινόταν πιο περίπλοκος ο αλγόριθμος ελέγχου τους χωρίς να επεκτείνεται ο χώρος δράσης τους.

Ο βραχίονας που διαθέτει το εργαστήριο ,RV-2A της Mitsubishi, είναι ένας αρθρωτός βιομηχανικός βραχίονας 6 βαθμών ελευθερίας. Έχει πεδίο δράσης ακτίνας 65 cm περίπου και μπορεί να χειριστεί αντικείμενα βάρους έως 2-2,5 kgr αναπτύσσοντας μέγιστη ταχύτητα 3,5 m/sec.



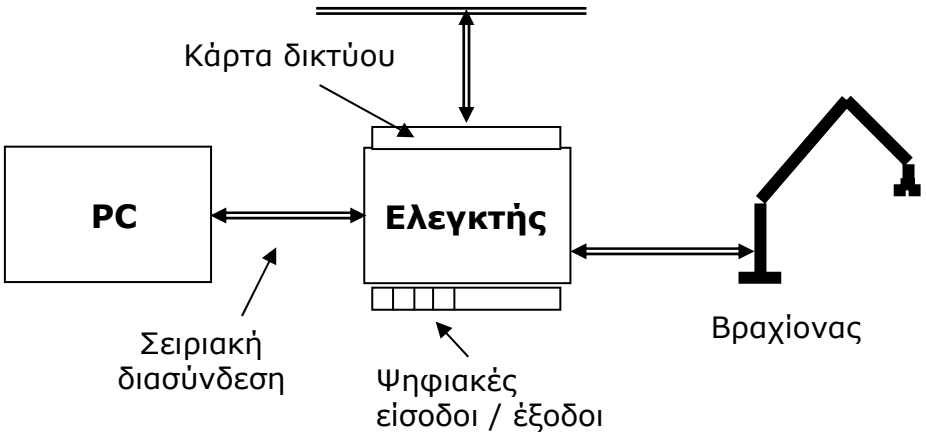
Πρώτη επαφή με τον βραχίονα RV-2A και το περιβάλλον προγραμματισμού COSIROP

Σύντομη περιγραφή του βραχίονα RV-2A

Ο ρομποτικός βραχίονας RV-2A, της Mitsubishi, είναι ένας βραχίονας 6 βαθμών ελευθερίας. Μπορεί να αναπτύξει μέγιστη ταχύτητα 3.5 m/sec για ένα μέγιστο φορτίο έως 2.5 kgr και έχει 'ακτίνα' δράσης περίπου 65 cm.

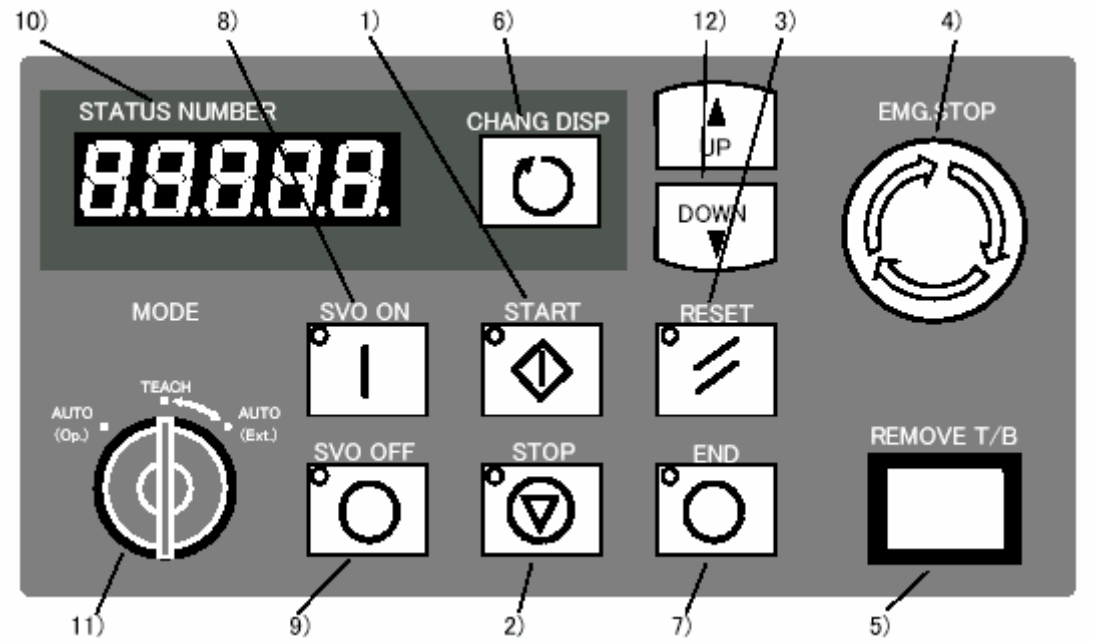
Ο ελεγκτής του διαθέτει:

- Σειριακή θύρα για σύνδεση με υπολογιστή.
- Κάρτα δικτύου για σύνδεση με τοπικό δίκτυο.
- Κάρτες που ενσωματώνουν ψηφιακές εισόδους και εξόδους, γενικής χρήσης.

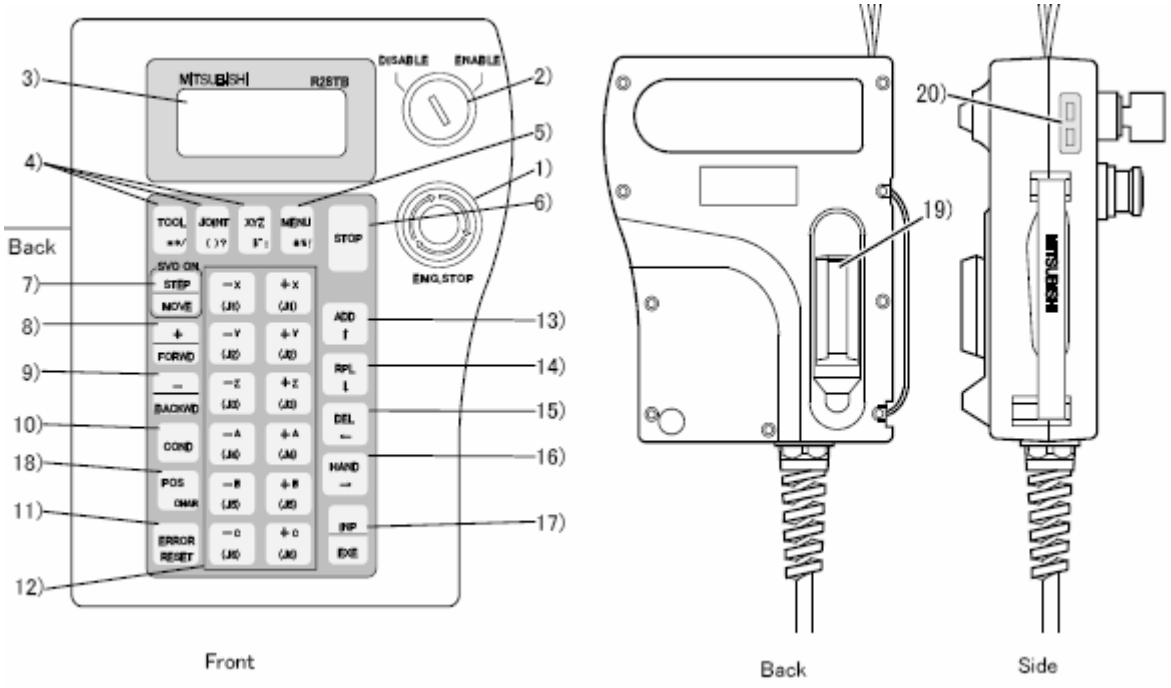


Ο ελεγκτής

Η όψη του ελεγκτή φαίνεται στη παρακάτω φωτογραφία.



Teaching Pendant



A/A	Όνομα	Λειτουργία
1	START	Ξεκινάει το πρόγραμμα που έχουμε επιλέξει. Το πρόγραμμα τρέχει στον ελεγκτή συνεχώς. Κατά την εκτέλεση του, το πράσινο λαμπάκι στο κουμπί ανάβει.
2	STOP	Σταματάει τον βραχίονα αμέσως, χωρίς να απενεργοποιεί τους σερβοκινητήρες. Το κόκκινο φωτάκι ανάβει.
3	RESET	Σε περίπτωση σφάλματος, το κόκκινο φωτάκι του κουμπιού ανάβει και η εκτέλεση του προγράμματος τερματίζεται. Πατώντας το κουμπί, επαναφέρουμε τον ελεγκτή σε κατάσταση λειτουργίας.
4	EMG. STOP	Σταματάει τον βραχίονα αμέσως και απενεργοποιεί τους σερβοκινητήρες.
5	REMOVE T/B	Χρησιμοποιείται όταν συνδέουμε ή αποσυνδέουμε το χειριστήριο εκπαίδευσης.
6	CHNG. DISP.	Εναλλαγή των ενδείξεων στην οθόνη του ελεγκτή.
7	END	Τερματίζει την εκτέλεση του προγράμματος.
8	SVO ON	Επιτρέπει την λειτουργία των σερβοκινητήρων παρέχοντάς τους ισχύ. Το πράσινο φωτάκι ανάβει.
9	SVO OFF	Απενεργοποιεί τους σερβοκινητήρες.
10	STATUS NUMBER	Δίνει της εξής πληροφορίες: Αριθμός Προγράμματος, Γραμμή κώδικα που εκτελείται ανά πάσα στιγμή. Την ταχύτητα εκτέλεσης(ο αριθμός που αναφέρεται στην ταχύτητα είναι η επί τοις εκατό αντιστοίχιση της τρέχουσας ,με τη μέγιστη ταχύτητα που έχουμε ορίσει μέσα στο πρόγραμμα μας. Τον Αριθμό σφάλματος ή προβλήματος όταν ενεργοποιείται κάποιος συναγερμός (ανατρέχουμε στο σχετικό manual).
11	MODE	Teach : Δέχεται εντολές μόνο από το χειριστήριο εκπαίδευσης(teaching pendant). Auto (Op.) : Δέχεται εντολές μόνο από τον ελεγκτή(φάση εκτέλεσης ήδη αποθηκευμένων προγραμμάτων στον controller). Auto (Ext.) : Δέχεται εντολές μόνο από εξωτερική πηγή (Διασύνδεση με το τερματικό Υπολογιστή).
12	UP & DOWN	Ανάλογα με την πληροφορία που έχουμε επιλέξει να εμφανίζεται στην οθόνη του ελεγκτή, μπορούμε να επιλέξουμε πρόγραμμα, να ρυθμίσουμε την ταχύτητα κίνησης και να προσδιορίσουμε τα πιθανά λάθη.

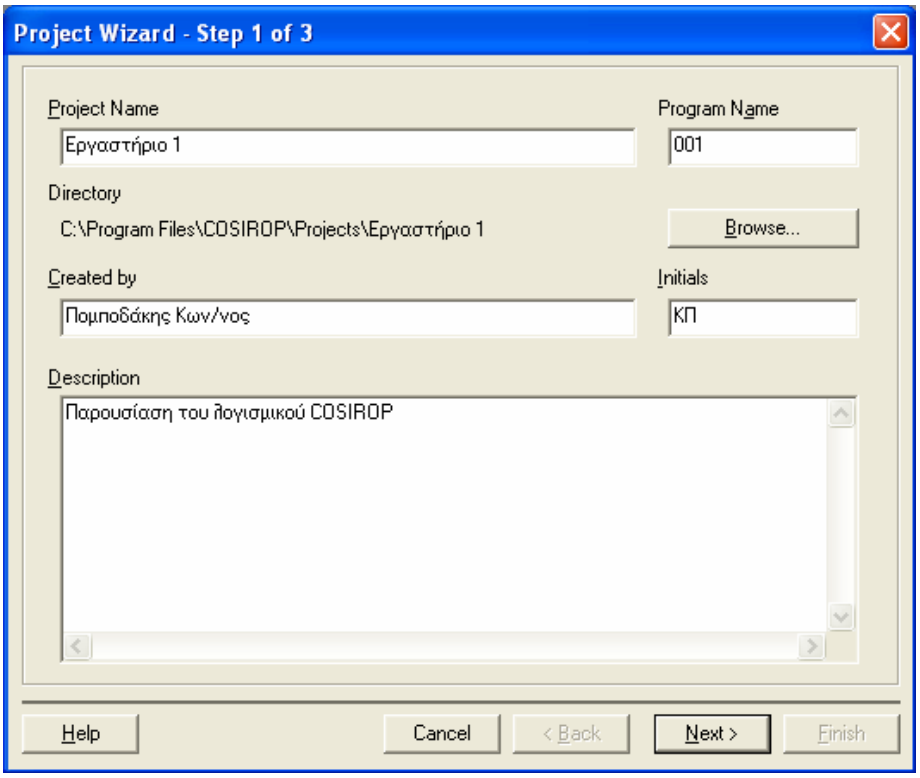
Το περιβάλλον προγραμματισμού COSIROP

Ο βραχίονας RV-2A μπορεί να προγραμματισθεί είτε με την βοήθεια του ειδικού χειριστηρίου εκπαίδευσης (***Teaching Pendant***) είτε με την βοήθεια ενός υπολογιστή (PC).

Για την δεύτερη αυτή περίπτωση υπάρχει διαθέσιμο το «περιβάλλον» προγραμματισμού και παρακολούθησης **COSIROP**. Μέσα από το εν λόγω περιβάλλον, ο χειριστής μπορεί να προγραμματίσει πολλούς από τους βραχίονες της Mitsubishi, μεταξύ των οποίων και τον βραχίονα του εργαστηρίου RV-2A. Εκτός του προγραμματισμού, το COSIROP παρέχει την δυνατότητα ελέγχου και αποσφαλμάτωσης (*Debugging*) του προγράμματος, καθώς και παρακολούθησης του βραχίονα κατά την φάση της λειτουργίας του (παρακολούθηση κινηματικών και ηλεκτρικών μεγεθών). Η διασύνδεση του PC με τον ελεγκτή του βραχίονα πραγματοποιείται είτε μέσω σειριακής θύρας (**RS232** ***Serial Interface***) είτε μέσω δικτύου βάση του πρωτοκόλλου (***TCP/IP***)[1].

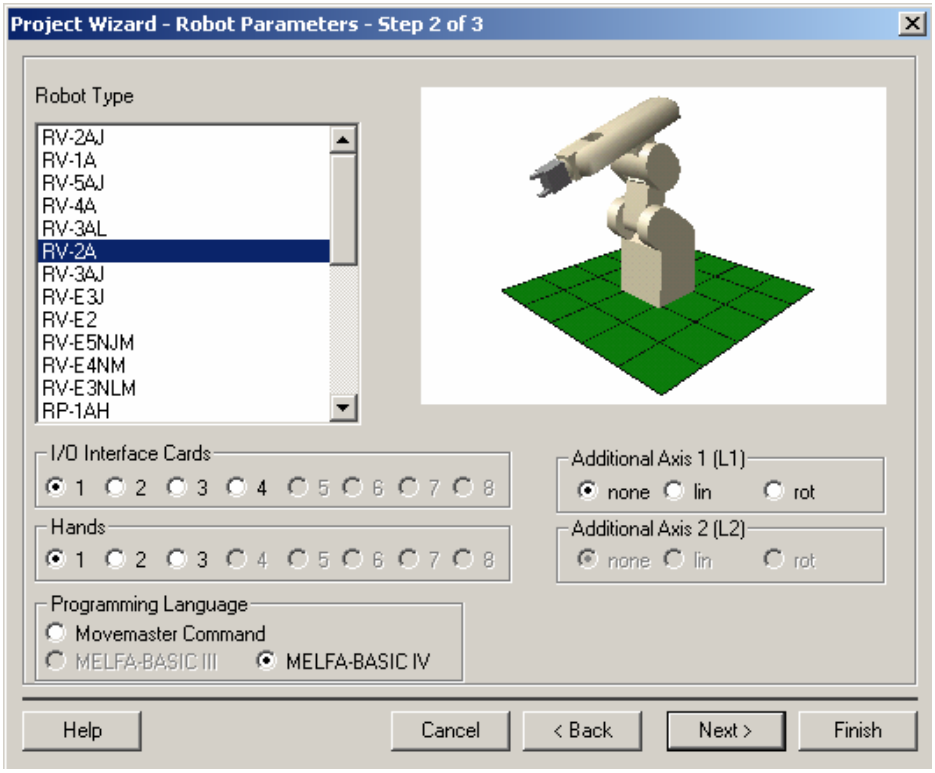
Πρώτη επαφή

Αφού τρέξουμε το COSIROP, από το μενού *File*, επιλέγουμε ***Project Wizard***. Στο πεδίο ***Project Name***, δίνουμε το όνομα του *project*, όπως αυτό θα αποθηκευτεί στον φάκελο του υπολογιστή που έχουμε επιλέξει. Στο πεδίο ***Program Name***, δίνουμε το όνομα του προγράμματος, όπως αυτό θα αποθηκευτεί στον ελεγκτή (ένας αριθμός μέχρι τέσσερα ψηφία). Προαιρετικά συμπληρώνουμε τα υπόλοιπα πεδία. Ένα παράδειγμα φαίνεται στη παρακάτω φωτογραφία.



[1] Στον ελεγκτή του εργαστηρίου δεν έχει εγκατασταθεί τέτοια κάρτα.

Πατάμε *Next*. Στην επόμενη καρτέλα, επιλέγουμε τον τύπο του βραχίονα που θέλουμε να προγραμματίσουμε και δίνουμε ορισμένες πληροφορίες για το σύστημα του ρομπότ. Ο τύπος του βραχίονα (***Robot Type***) που χρησιμοποιούμε είναι ο **RV-2A**, έχει μία (1) αρπάγη (*Hands*) και μία (1) κάρτα ψηφιακών εισόδων / εξόδων (*I/O Interface Cards*). Η γλώσσα προγραμματισμού (*Programming Language*) που θα χρησιμοποιήσουμε είναι η **Melfa – Basic IV**. Πρόσθετος άξονας (*Additional Axis*) δεν υπάρχει οπότε επιλέγουμε *none*.



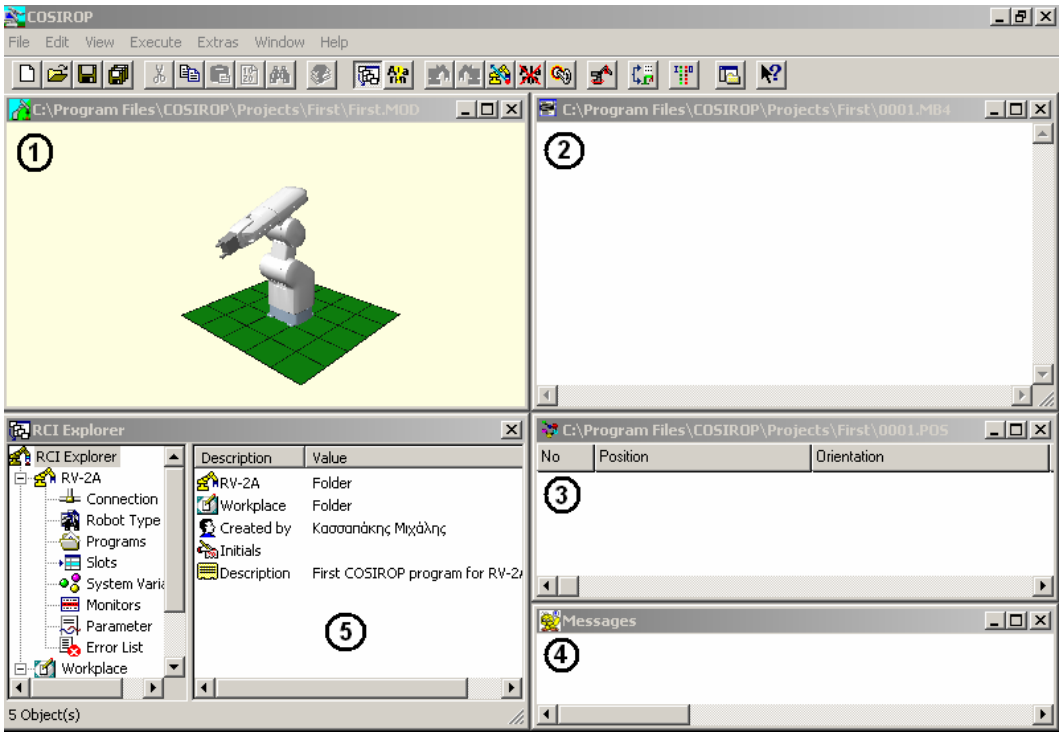
Τελειώνοντας το στάδιο αυτό (*Finish*), μεταφερόμαστε στην κυρίως οθόνη.

Στην περίπτωση που δεν επιθυμούμε να ξεκινήσουμε νέο *project*, αλλά να συνεχίσουμε την εργασία μας σε κάποιο προηγούμενο, τότε στην αρχική οθόνη του COSIROP δεν επιλέγουμε *Project Wizard*. Από το μενού *File*, επιλέγουμε *Open…* και αφού πάμε στον κατάλογο που θέλουμε, ανοίγουμε το αποθηκευμένο *project* που μας ενδιαφέρει.

Το κυρίως προγραμματιστικό περιβάλλον

Στην οθόνη εμφανίζονται πέντε παράθυρα :

1. Φαίνεται ένα τρισδιάστατο μοντέλο του βραχίονα. Εδώ μπορούμε, κάνοντας δεξί κλικ, να δούμε τον βραχίονα από διαφορετικές όψεις να τον μεγεθύνουμε . Επίσης με μία διαδικασία που θα δούμε στην πορεία μπορούμε να έχουμε προσομοίωση της διάταξης του βραχίονα σε κάθε σημείο που έχουμε αποθηκεύσει .
2. Το παράθυρο προγραμματισμού (όπου γράφουμε τον κώδικα του προγράμματος). Το αρχείο, αποθηκεύεται στον υπολογιστή και έχει το όνομα που έχουμε προηγουμένως ορίσει με κατάληξη **MB4**.
3. Η λίστα με τα σημεία (*Position List*) που χρησιμοποιούνται από το πρόγραμμα μας. Αρχείο με κατάληξη **POS**.
4. Παράθυρο μηνυμάτων μεταξύ ελεγκτή και υπολογιστή.
5. Το παράθυρο επικοινωνίας με τον ελεγκτή του βραχίονα : ***R**obot **C**ontroller Interface **(RCI) Explorer***.



Λειτουργίες του RCI Explorer

Ο *RCI Explorer* υλοποιεί την σύνδεση μεταξύ του ρομπότ και του υπολογιστή μας. Είναι ένα κέντρο ανταλλαγής πληροφοριών και δεδομένων με τον ελεγκτή του ρομπότ. Δίνει πληροφορίες για την τρέχουσα κατάσταση του βραχίονα, μεταφέρει προγράμματα από και προς τον ελεγκτή και παρακολουθεί την εκτέλεση τους.

Το παράθυρο του *Explorer* χωρίζεται σε δύο φακέλους. Στον πρώτο, που έχει το όνομα του βραχίονα, περιέχονται πληροφορίες και δεδομένα του ελεγκτή. Αναλυτικότερα, οι λειτουργίες κατά σειρά εμφάνισης είναι:

- ***Connection*** : Μας πληροφορεί για το είδος της σύνδεσης που έχει επιλεγεί (σειριακή ή δικτυακή) και την κατάσταση της. Από εδώ μπορούμε, κάνοντας δεξί κλικ, να ενεργοποιήσουμε ή να απενεργοποιήσουμε την σύνδεση του υπολογιστή με τον ελεγκτή. Επισημαίνουμε ότι, πριν ξεκινήσουμε οποιαδήποτε ανταλλαγή πληροφοριών, μεταξύ του ελεγκτή και του PC, είναι απαραίτητη η ενεργοποίηση της σύνδεσης.

- ***Robot Type*** : Εμφανίζει στοιχεία του ελεγκτή, όπως την ελεύθερη μνήμη του, των αριθμό των προγραμμάτων που είναι υποθηκευμένα στον ελεγκτή κ.α.

- ***Programs*** : Περιέχει τα προγράμματα που είναι αποθηκευμένα στον ελεγκτή. Από εδώ, κάνοντας δεξί κλικ σε ένα πρόγραμμα, μπορούμε να το τρέξουμε, να το σταματήσουμε, να το φορτώσουμε στον υπολογιστή, να ξεκινήσουμε την αποσφαλμάτωση του (*Debugging*) κ.ο.κ.

- ***Slots*** : Ο συγκεκριμένος τύπος ελεγκτή υποστηρίζει την σύγχρονη εκτέλεση (*Multitasking*) έως 8 διαφορετικών προγραμμάτων, τα οποία μπορούμε να αποθηκεύσουμε στις 8 διαφορετικές θύρες (*Slots-περιοχές μνήμης*) του ελεγκτή. Η προεπιλεγμένη θύρα, για την αποθήκευση του προγράμματος μας, είναι η πρώτη.

- ***System Variables*** : Μας ενημερώνει για διάφορες μεταβλητές συστήματος του ελεγκτή και τις τρέχουσες τιμές τους.

- ***Monitors*** : Εδώ έχουμε τα εργαλεία παρακολούθησης του βραχίονα. Κάνοντας διπλό κλικ σε ένα από αυτά εμφανίζεται το αντίστοιχο παράθυρο. Πρόκειται για σημαντικά εργαλεία, που μας δίνουν την δυνατότητα να παρακολουθούμε την κάθε λεπτομέρεια του ρομπότ και να ελέγχουμε την πορεία του προγράμματος μας. Τα σημαντικότερα από αυτά είναι: Παρακολούθηση της εντολής του προγράμματος που εκτελείται (*Programs*), Παρακολούθηση της τιμής των μεταβλητών του προγράμματος (*Variables*), Παρακολούθηση και αλλαγή της κατάστασης των ψηφιακών εισόδων / εξόδων (*Inputs, Outputs*).

- ***Parameter*** : Περιέχει τις παραμέτρους που χρησιμοποιούνται από το σύστημα του ελεγκτή και την τιμή αυτών. Η τιμές αυτών είναι προκαθορισμένες και δεν πρέπει να αλλαχτούν.

- ***Error List*** : Εμφανίζει τα σφάλματα που έχουν συμβεί κατά την λειτουργία του συστήματος, ξεκινώντας από το πιο πρόσφατο. Εάν κάνουμε διπλό κλικ σε κάποιο από αυτά θα εμφανιστεί ο λόγος που προκάλεσε το σφάλμα και τι προτείνεται να κάνουμε.

Προχωράμε, στο δεύτερο φάκελο, που ονομάζεται *Workplace* και αφορά τον υπολογιστή που είναι συνδεμένος με τον ελεγκτή, στον οποίο εργαζόμαστε. Οι λειτουργίες εδώ είναι οι εξής δύο:

- ***Programs*** : Βλέπουμε τα προγράμματα που περιέχονται στο *project* το οποίο δουλεύουμε και είναι αποθηκευμένα στον σκληρό δίσκο του υπολογιστή.

- ***Tools*** : Πρόκειται για τα εργαλεία που προσφέρει το περιβάλλον COSIROP, τα οποία κάνουν τον προγραμματισμό του βραχίονα ευκολότερο.
  Το ***Command Tool*** είναι ένα εργαλείο αποστολής μεμονωμένων εντολών είτε στον ελεγκτή του βραχίονα, είτε στο παράθυρο προγραμματισμού. Στην περίπτωση που στείλουμε την εντολή στον ελεγκτή, μπορούμε να δούμε και την απόκριση του, στο πλαίσιο *Robot*.
  Το εργαλείο ***Jog Operation*** είναι πάρα πολύ χρήσιμο και γι' αυτό θα ασχοληθούμε διεξοδικά παρακάτω. Επιγραμματικά αναφέρουμε ότι μέσω αυτού μπορούμε να κινήσουμε τον βραχίονα σε οποιαδήποτε θέση και με οποιοδήποτε προσανατολισμό.
  Το ***Project Management*** δίνει μία συνολική εικόνα της εργασίας μας και των αρχείων των οποίων εμπλέκονται σε αυτή.
  Το *Terminal* είναι ένα παράθυρο απ' ευθείας επικοινωνίας με τον ελεγκτή του βραχίονα και το πρόγραμμα το οποίο τρέχει.

Πρώτα βήματα στον προγραμματισμό

Σχεδιασμός προβλήματος

Βρισκόμαστε λοιπόν στην κυρίως οθόνη του COSIROP και έχουμε σκεφτεί τι θέλουμε να προγραμματίσουμε τον βραχίονα να κάνει. Στην προκειμένη περίπτωση, θέλουμε ο βραχίονας να ξεκινήσει από ένα σημείο (έστω P1), να κινηθεί γραμμικά σε ένα δεύτερο σημείο (έστω P2). Στο σημείο αυτό, να δώσει εντολή να λειτουργήσει ο ταινιόδρομος για 2sec και στη συνέχεια, αφού απενεργοποιήσει τον ταινιόδρομο να κινηθεί πίσω στο πρώτο σημείο (P1).

Αποκατάσταση επικοινωνίας

Πρώτα απ' όλα, εάν δεν το έχουμε ήδη κάνει, πρέπει να αποκαταστήσουμε την επικοινωνία του ελεγκτή με τον υπολογιστή. Κάνοντας δεξί κλικ στην λειτουργία ***Connection*** **του** ***RCI Explorer***, επιλέγουμε ***Connect***. Εμφανίζεται ένα παράθυρο που μας δίνει πληροφορίες για τον ελεγκτή και τον τύπο του βραχίονα, καθώς μας προειδοποιεί και για τις οδηγίες ασφαλείας που πρέπει να γνωρίζουμε προτού χειριστούμε το ρομπότ.

Βασικές αρχές ασφάλειας

Στο σημείο αυτό, θα επισημάνουμε κάποιες βασικές αρχές που πρέπει να έχουμε συνεχώς στο μυαλό μας, ώστε να χειριστούμε το ρομπότ με ασφάλεια, τόσο για εμάς και τους γύρω μας, όσο και για το ίδιο το ρομπότ.

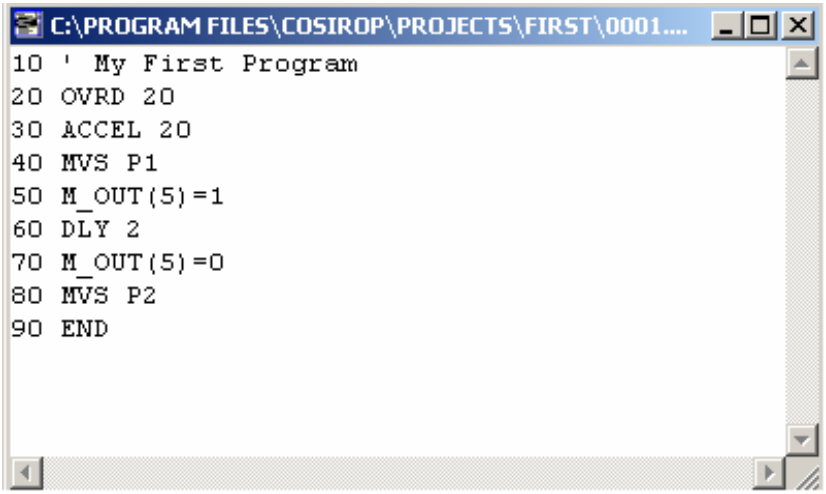
Κύρια αιτία τραυματισμού, όπως καταλαβαίνεται, είναι η ανεξέλεγκτη κίνηση του βραχίονα. Για αυτό πρέπει όταν δίνουμε εντολές στον ελεγκτή να είμαστε εντελώς σίγουροι για το ποια θα είναι η αντίδραση του βραχίονα. Πρώτα 'εκτελούμε' την εντολή στο μυαλό μας και μετά τη στέλνουμε για εκτέλεση στο ρομπότ. Με το ίδιο σκεπτικό, πριν από την τελική εκτέλεση κάθε προγράμματος μας στο ρομπότ, θα πρέπει να έχει προηγηθεί εκτέλεση βήμα-βήμα του προγράμματος, πρώτα νοητά στο μυαλό μας και στη συνέχεια στη πράξη χρησιμοποιώντας την διαδικασία αποσφαλμάτωσης. Εφόσον βεβαιωθούμε ότι έχουμε ερευνήσει διεξοδικά την λειτουργία του προγράμματος, τότε και μόνο τότε, το μεταφέρουμε στον ελεγκτή και ξεκινάμε την εκτέλεση του.

Τελευταίο, αλλά σημαντικότερο, είναι πάντα να βρισκόμαστε σε απόσταση ασφαλείας από τον βραχίονα και να είμαστε σε ετοιμότητα να σταματήσουμε την κίνηση του, πατώντας το κουμπί **STOP** ή **EMG. STOP** του ελεγκτή, εάν χρειαστεί.

Επίσης κατά την πρώτη εκτέλεση από τον controller, ρυθμίζουμε την **ταχύτητα εκτέλεσης του προγράμματος σε χαμηλά επίπεδα** εξασφαλίζοντας το χρονικό περιθώριο και τη δυνατότητα επέμβασής μας για τη διακοπή της ροής του προγράμματος μόλις αντιληφθούμε ότι κάτι δεν εξελίσσεται όπως θα περιμέναμε.

Περιγραφή του προγράμματος

Προχωράμε παραθέτοντας το πρόγραμμα που θα χρειαστεί να γράψουμε στο παράθυρο προγραμματισμού, ώστε το ρομπότ να εκτελέσει όσα είπαμε προηγουμένως.

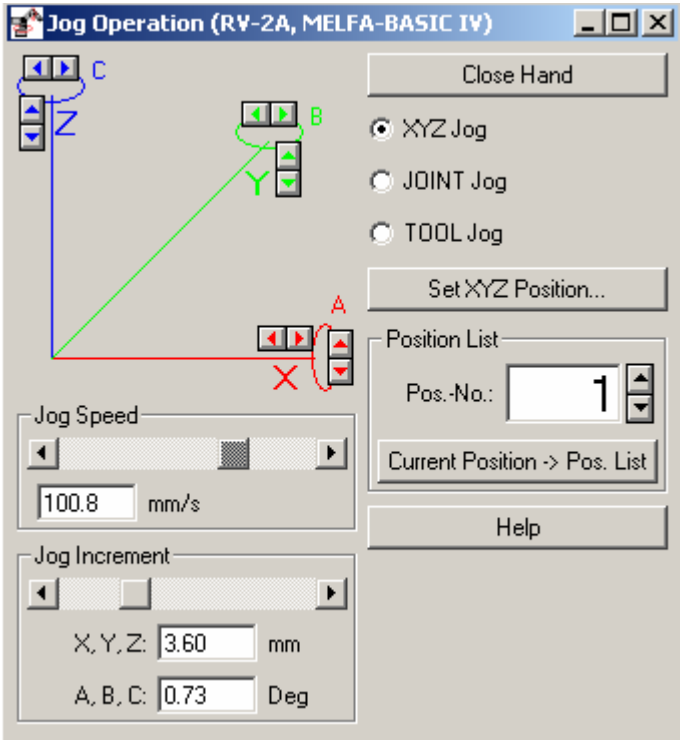


Ας κάνουμε μία σύντομη περιγραφή των εντολών που αποτελούν το πρόγραμμα μας. Το πρώτο χαρακτηριστικό των προγραμμάτων σε γλώσσα *Basic*, είναι ότι η κάθε σειρά αριθμείται με ένα μοναδικό νούμερο. Έχει καθιερωθεί, τα νούμερα αυτά να αυξάνονται κατά 10, χωρίς να απαγορεύεται να ακολουθήσουμε άλλο τρόπο (π.χ. 1, 3, 5…). Αριθμώντας αυξάνοντας κατά δέκα έχουμε τη δυνατότητα να προσθέσουμε επιπλέον κώδικα στην πορεία…

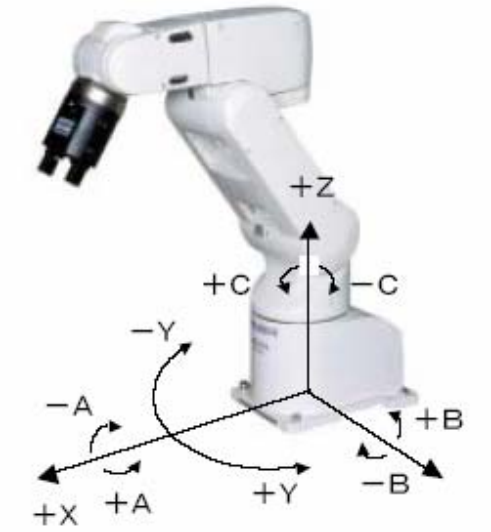
10 : Στη γραμμή αυτή κάνουμε μόνο κάποια σχόλια.
20 : Ορίζουμε την ταχύτητα κίνησης στο 20%.
30 : Ορίζουμε την επιτάχυνση / επιβράδυνση στο 20%.
40 : Κινήσου γραμμικά στο σημείο P1.
50 : Η ψηφιακή έξοδος 5 να έρθει στην κατάσταση λογικό ¨1¨ (Ενεργοποίηση του ταινιόδρομου).
60 : Καθυστέρησε 2 sec.
70 : Η ψηφιακή έξοδος 5 να έρθει στην κατάσταση λογικό ¨0¨ (Απενεργοποίηση του ταινιόδρομου).
80 : Κινήσου γραμμικά στο σημείο P2.
90 : Δηλώνει το τέλος του προγράμματος.

Ορισμός σημείων

Επόμενο βήμα, είναι να ορίσουμε τα σημεία P1 και P2 στα οποία θα κινηθεί ο βραχίονας. Ένα πολύ χρήσιμο εργαλείο που μας προσφέρει το COSIROP, για αυτή τη δουλεία είναι το ***Jog Operation***. Ανοίγοντας το βλέπουμε το παρακάτω παράθυρο.



Εδώ, αρχικά, διακρίνουμε τους 3 άξονες και τα αντίστοιχα βέλη που χρησιμεύουν στο να κινήσουμε το ρομπότ. Παρατηρήστε ότι έχουμε 2 ζευγάρια βέλη ανά άξονα. Το ένα χρησιμεύει στην μετακίνηση πάνω στον άξονα και το άλλο στην περιστροφή γύρω από τον άξονα. Το σημείο αναφοράς είναι το άκρο (βεντούζα) του βραχίονα. Στην παρακάτω φωτογραφία φαίνονται οι άξονες και πως επηρεάζεται η κίνηση από την κάθε μεταβλητή (x, y, z, a, b, c), μόνο που θα έπρεπε η αρχή των αξόνων να βρίσκεται στην άκρη του βραχίονα.

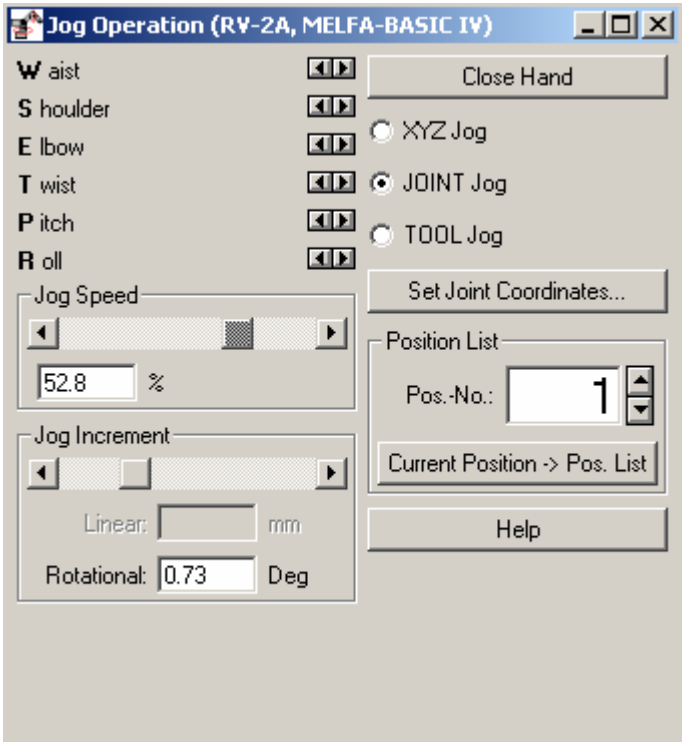


Δύο πράγματα που πρέπει να τους δώσουμε ιδιαίτερη προσοχή, είναι η ταχύτητα κίνησης **(*Jog Speed*)** και το βήμα **(*Jog Increment*).** Προτείνεται να ξεκινάμε από χαμηλές τιμές και να αυξάνουμε σταδιακά, ανάλογα τις απαιτήσεις. Επίσης, πάντα

πρέπει να περιμένουμε να ολοκληρωθεί η εντολή που έχουμε δώσει (δηλ. να σταματήσει την κίνηση του ο βραχίονας) και μετά να προχωρήσουμε στην επόμενη.

Προχωρώντας, βλέπουμε το κουμπί *Close Hand*, που στην περίπτωση μας δεν έχει εφαρμογή, καθώς δεν έχουμε προσαρμόσει τέτοιο εργαλείο στον βραχίονα. Κάτω από αυτό, βρίσκονται τρεις επιλογές.

- ***XYZ Jog*** **:** Είναι το παράθυρο που περιγράψαμε προηγουμένως και χρησιμεύει στο να κινήσουμε τον βραχίονα μεταβάλλοντας τις συντεταγμένες (x, y, z) και τον προσανατολισμό (a, b, c) του άκρου. Επιτυγχάνουμε κίνηση της βάσης του τελικού στοιχείου δράσης κατά μήκος των αξόνων (x,y,z).Παρατηρούμε συνδυασμό κινήσεων όλων των αρθρώσεων ταυτόχρονα.
- ***JOINT Jog*** **:** Μας μεταφέρει στο παράθυρο που φαίνεται παρακάτω. Εδώ κινούμε κάθε άρθρωση χωριστά.
- ***TOOL Jog*** **:** Χρησιμεύει στην περίπτωση που έχουμε προσαρμόσει κάποιο τελικό στοιχείο δράσης στην άκρη του βραχίονα. Αυτό μπορεί να είναι αρκετά περίπλοκο επηρεάζοντας την μορφολογία του βραχίονα, αρχικά προσθέτοντάς του κάποιο επιπλέον μήκος ή ακόμη και βαθμό ελευθερίας εφόσον διαθέτει κάποια επιπλέον άρθρωση. Πληροφορίες για τα τεχνικά χαρακτηριστικά του εξαρτήματος αυτού έχουμε δώσει στο λογισμικό σε σχετικό παράθυρο διαλόγου κατά την εκκίνηση της εφαρμογής όπως θυμάστε.
  Χρησιμοποιούμε λοιπόν το *TOOL Jog για να επιτύχουμε κίνηση του τελικού στοιχείου δράσης κατά μήκος των αξόνων*(x,y,z).



Εάν λοιπόν επιλέξουμε την λειτουργία *Joint Jog*, μπορούμε να κινήσουμε ξεχωριστά τον κάθε σερβοκινητήρα του ρομπότ με θετική ή αρνητική φορά. Αυτό γίνεται πατώντας ένα από τα δύο βελάκια που βρίσκονται δίπλα στο όνομα της κάθε άρθρωσης. Οι αρθρώσεις είναι γραμμένες κατά σειρά, δηλ. *J1=Waist*, *J2=Shoulder*, *J3=Elbow* κ.ο.κ.

Όπως προαναφέραμε, έτσι και εδώ, πρέπει πριν δώσουμε οποιαδήποτε εντολή κίνησης να δώσουμε αρχικά στην ταχύτητα (*Jog Speed*) και στο βήμα (*Jog Increment*) χαμηλές τιμές.

Όταν έχουμε φέρει στην επιθυμητή θέση τον βραχίονα, είτε με τον πρώτο τρόπο που περιγράψαμε είτε με τον δεύτερο, πρέπει να την αποθηκεύσουμε. Αφού λοιπόν επιλέξουμε τον αριθμό που θέλουμε να δώσουμε στην θέση αυτή (*Pos. No.*), κάνουμε κλικ στο κουμπί ***Current Position -> Pos. List***. Έτσι, μεταφέρεται η παρούσα θέση του βραχίονα στην Λίστα σημείων που έχει δημιουργηθεί για το τρέχων πρόγραμμα. Στην περίπτωση μας, πρέπει να δημιουργήσουμε δύο σημεία, τα P1 και P2.

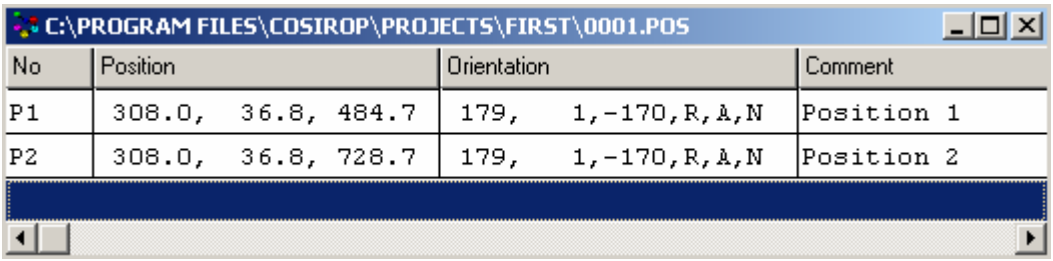
Αυτό που πρέπει να προσέχουμε, κάθε φορά που ορίζουμε κάποια σημεία, είναι πως αυτά θα χρησιμοποιηθούν μέσα στο πρόγραμμα και πως θα κινηθεί ο βραχίονας για να φτάσει σε αυτά. Δηλαδή, στην περίπτωση μας, θα πρέπει να έχουμε στο μυαλό μας ότι ο βραχίονας θα κινηθεί γραμμικά από το σημείο P1 στο σημείο P2. Άρα θα πρέπει γύρω από την νοητή ευθεία που ενώνει τα δύο αυτά σημεία να μην παρεμβάλλεται κάτι που θα μπορούσε να ενοχλήσει την κίνηση του ρομπότ.

Λίστα σημείων

Αφού δημιουργήσουμε τα σημεία μας, το παράθυρο, που περιέχει τη Λίστα σημείων του αντίστοιχου προγράμματος, θα έχει την παρακάτω μορφή. Εδώ φαίνεται το νούμερο που έχουμε δώσει στο κάθε σημείο (*No.*), οι τιμές των συντεταγμένων x ,y ,z (*Position*) και οι τιμές του προσανατολισμού a, b, c (*Orientation*). Κάνοντας δεξί κλικ σε ένα από αυτά μπορούμε να το αλλάξουμε, να το διαγράψουμε κ.τ.λ.

Επίσης κάνοντας διπλό κλικ επάνω στο P1 ή στοP2 μπορούμε να δούμε στο πρώτο παράθυρο, την οπτική προσομοίωση της διάταξης του βραχίονα στο συγκεκριμένο σημείο. Η δυνατότητα αυτή αποτελεί σημαντικό βοήθημα , στην περίπτωση που η λίστα των σημείων έχει μεγαλώσει αρκετά ,με αποτέλεσμα να μην είναι δυνατόν να θυμόμαστε όλα τα σημεία .

Στη συνέχεια και εφόσον έχουμε την εικόνα ενός επιλεγμένου σημείου στο παράθυρο της προσομοίωσης ,έχουμε τη δυνατότητα επίσης να στείλουμε το βραχίονα στο σημείο αυτό. Από το μενού ***Execute***,επιλέγουμε ***PC Position→Robot***.

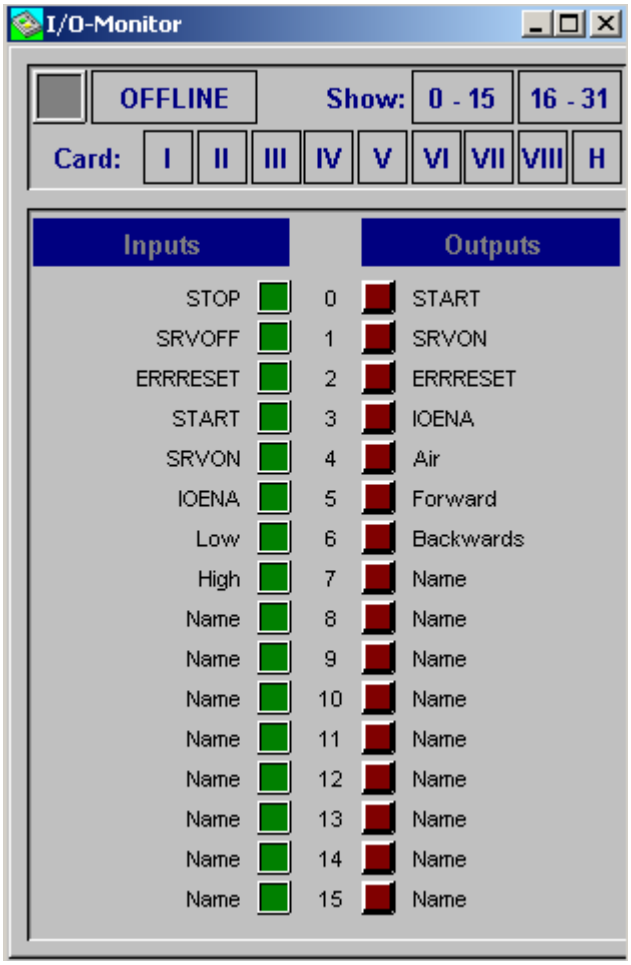


Παρακολούθηση ψηφιακών εισόδων / εξόδων

Επίσης, θα πρέπει να ελέγξουμε εάν η ψηφιακή έξοδος 5 είναι συνδεμένη με τον ταινιόδρομο και ότι το λογικό “1” τον θέτει σε λειτουργία. Για τον λόγο αυτό, υπάρχει το εργαλείο ***I/O Monitor***, που βρίσκεται στον *RCI Explorer* και στο φάκελο που περιέχει τα εργαλεία παρακολούθησης της κατάστασης του ελεγκτή **(*RV-2A -> Monitors*)**.

Το παράθυρο που εμφανίζεται μόλις το τρέξουμε, φαίνεται παρακάτω. Εδώ βλέπουμε ότι μπορούμε να επιλέξουμε την κάρτα (*Card*) που θα εξετάσουμε, οπότε εμφανίζονται οι αντίστοιχοι είσοδοι (*Inputs*) και έξοδοι (*Outputs*) με τα ονόματα αυτών

που χρησιμοποιούνται ήδη. Στην περίπτωση μας μόνο μία κάρτα υπάρχει συνδεμένη στον ελεγκτή και μπορούμε να δούμε τις πρώτες 16 (0-15) ή τις επόμενες 16 (16-31).

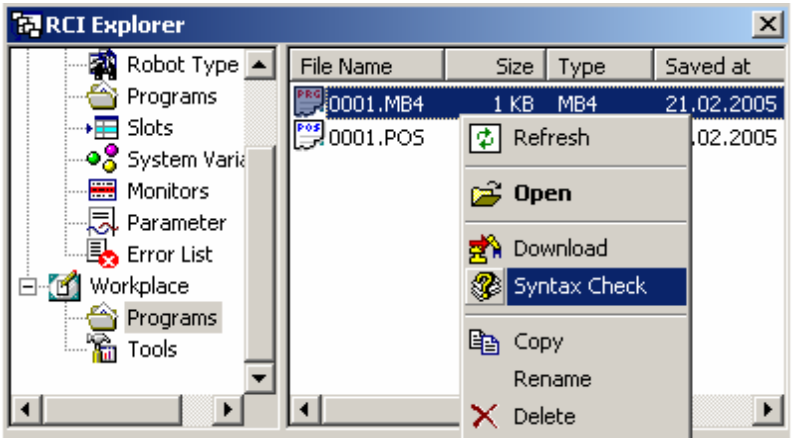


Πατώντας το αντίστοιχο κουμπί για να αρχίσει η λειτουργία *ONLINE*, βλέπουμε την τρέχουσα κατάσταση των εισόδων / εξόδων. Αναμμένη ένδειξη δίπλα στο όνομα συμβολίζει λογικό “1”, ενώ το αντίθετο λογικό “0”. Την κατάσταση των εισόδων είναι προφανές ότι δεν μπορούμε να την επηρεάσουμε, μόνο να την παρατηρήσουμε. Ενώ, για τις εξόδους, μπορούμε να βλέπουμε την τρέχουσα κατάσταση τους και κάνοντας κλικ πάνω στη ένδειξη να την αλλάζουμε.

Όσο αφορά το πρόγραμμα μας, μπορούμε να κάνουμε κλικ στην έξοδο 5 (*Forward*) ώστε να επαληθεύσουμε ότι ο ταινιόδρομος κινείται στη φορά που θέλουμε.

Συντακτικός έλεγχος

Έχοντας γράψει λοιπόν το πρώτο μας πρόγραμμα και αφού ορίσαμε τα απαραίτητα σημεία για την λειτουργία του, πρέπει να ελέγξουμε εάν υπάρχουν συντακτικά λάθη στο πρόγραμμα μας.



Στο παράθυρο του *RCI Explorer*, και στο φάκελο που αφορά τα προγράμματα του υπολογιστή (*Workplace -> Programs*) κάνουμε δεξί κλικ στο πρόγραμμα μας και επιλέγουμε ***Syntax Check***. Εάν το πρόγραμμα είναι σωστό, στο παράθυρο μηνυμάτων θα πρέπει να εμφανιστεί το μήνυμα ***"0 Error(s), 0 Warning(s)"***. Σε διαφορετική περίπτωση, μας λέει πόσα λάθη βρέθηκαν και σε ποιες γραμμές.

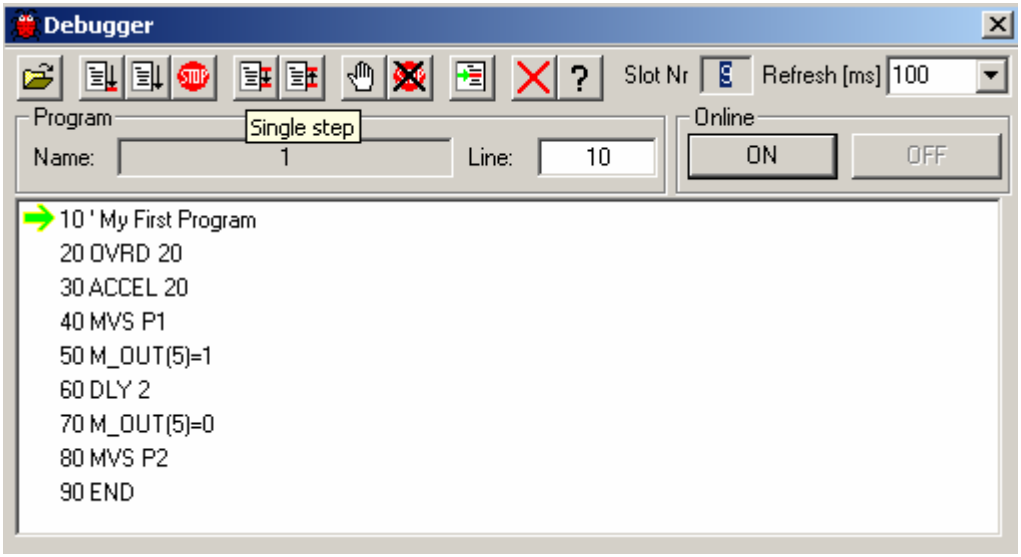
Φόρτωση του προγράμματος στον ελεγκτή

Το επόμενο βήμα είναι να κατεβάσουμε το πρόγραμμα και την αντίστοιχη λίστα σημείων που ετοιμάσαμε, στον ελεγκτή του βραχίονα, ώστε να μπορέσουμε να το τρέξουμε. Ακολουθούμε παρόμοια διαδικασία με αυτήν που περιγράψαμε στην προηγούμενη παράγραφο. Η διαφορά είναι ότι τώρα από το μενού όπου εμφανίζεται επιλέγουμε *Download*, τόσο για το πρόγραμμα (.MB4) όσο και για τη λίστα σημείων (.POS). Στο παράθυρο, που εμφανίζεται μόλις επιλέξουμε *Download*, πατάμε OK. Εφόσον, δεν υπάρχουν λάθη στο πρόγραμμα μας, η διαδικασία ολοκληρώνεται.

Αποσφαλμάτωση του προγράμματος

Πριν το τελικό βήμα, που είναι η εκτέλεση του προγράμματος, πρέπει να προηγηθεί η αποσφαλμάτωση του. Πρόκειται για ένα επιπλέον βήμα, ώστε να βεβαιωθούμε ότι το ρομπότ αντιδρά στις εντολές μας όπως είχαμε σχεδιάσει. Αυτό επιτυγχάνεται με το να εκτελέσουμε το πρόγραμμα σειρά-σειρά ώστε να δούμε άμεσα και βήμα-βήμα τις αντιδράσεις του ρομπότ.

Πηγαίνουμε, λοιπόν, στον *RCI Explorer* και στο φάκελο που περιέχει τα προγράμματα του ελεγκτή (*RV-2A -> Programs*). Εκεί θα πρέπει να υπάρχει το πρόγραμμα που γράψαμε και το οποίο πριν λίγο κατεβάσαμε στον ελεγκτή. Κάνουμε δεξί κλικ πάνω του και επιλέγουμε ***Debug***, οπότε εμφανίζεται το παρακάτω παράθυρο.



Εδώ μας προσφέρονται οι κυριότερες λειτουργίες, που συναντάμε στους αποσφαλματωτές των περισσότερων γλωσσών προγραμματισμού. Μία από αυτές είναι η διαδικασία εκτέλεσης του προγράμματος μας σειρά-σειρά (***Single Step***). Η εντολή αυτή είναι το πέμπτο εικονίδιο από τ' αριστερά όπως φαίνεται στο παραπάνω παράθυρο. Επίσης, όταν δεν βρισκόμαστε *Online*, μπορούμε να ορίσουμε ποια θα είναι η γραμμή εκκίνησης της αποσφαλμάτωσης, π.χ. όταν φτάσουμε στο τέλος του προγράμματος και θέλουμε να ξαναρχίσουμε.

Ακόμη και σε αυτό το στάδιο, θα πρέπει να είμαστε ιδιαίτερα προσεχτικοί με την εκτέλεση των εντολών. Κανείς, ούτε ο προγραμματιστής, δεν θα πρέπει να βρίσκεται στην ακτίνα δράσης του ρομπότ και πάντα να είμαστε σε ετοιμότητα να σταματήσουμε (*EMG. STOP*) την κίνηση του βραχίονα.

Εκτέλεση του προγράμματος

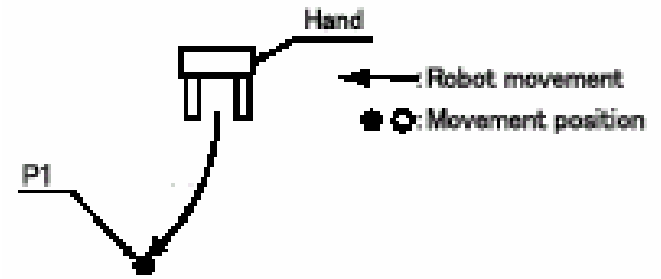
Εφόσον είμαστε ικανοποιημένοι από το πρόγραμμα μας, έφτασε η στιγμή να το εκτελέσουμε κανονικά. Όπως πριν, κάνοντας δεξί κλικ στο πρόγραμμα μας, επιλέγουμε ***Start (CYC)*** **ή** ***Start (REP)***. Στην πρώτη περίπτωση, το πρόγραμμα θα τρέξει μία φορά και όταν συναντήσει την εντολή *END* θα σταματήσει. Στην δεύτερη περίπτωση, θα εκτελείται συνεχώς, δηλ. μόλις συναντήσει την εντολή *END* θα επιστρέψει ξανά στην πρώτη γραμμή για να ξαναρχίσει, εωσότου εμείς σταματήσουμε την εκτέλεση του με κάποιο εξωτερικό τρόπο. Ένας τέτοιος τρόπος είναι να κάνουμε δεξί κλικ στο πρόγραμμα που εκτελείται και να επιλέξουμε *Stop*.

~~~~~~~~~~~~~~~~~Εντολές Ελέγχου Κίνησης~~~~~~~~~~~~~~~~

Πρώτα θα ασχοληθούμε με εντολές που χρησιμεύουν στο να κινήσουμε το ρομπότ, καθώς και στον ορισμό απαραίτητων μεταβλητών ελέγχου της κίνησης αυτής.

~~~~~~~~~~~~JOINT INTERPOLATION MOVEMENT~~~~~~~~~~
(Κίνηση ελεύθερης παρεμβολής)
Εντολή MOV

Η εντολή αυτή κινεί τον βραχίονα στο προκαθορισμένο σημείο (π.χ. *MOV P1*) χωρίς να μας ενδιαφέρει η διαδρομή που θα ακολουθηθεί. Το ρομπότ επιλέγει από μόνο του την διαδοχική κίνηση των αρθρώσεων βάση ενός συγκεκριμένου βέβαια αλγόριθμου που έχει δημιουργηθεί από τον κατασκευαστή για τις περιπτώσεις κινήσεων ελεύθερης παρεμβολής. Ο αλγόριθμος αυτός έχει στόχο να επιτύχει για παράδειγμα το μικρότερο συνδυασμό κινήσεων των αρθρώσεων, συντομία χρόνου, μικρότερη κατανάλωση ενέργειας κλπ. Εύκολα καταλαβαίνουμε ότι σπάνια αυτή η διαδρομή θα είναι ευθεία και φυσικά δεν μπορούμε να την γνωρίζουμε. Συνεπώς συμπεραίνουμε ότι η χρήση της εντολής αυτής εμπεριέχει και κάποιο κίνδυνο.



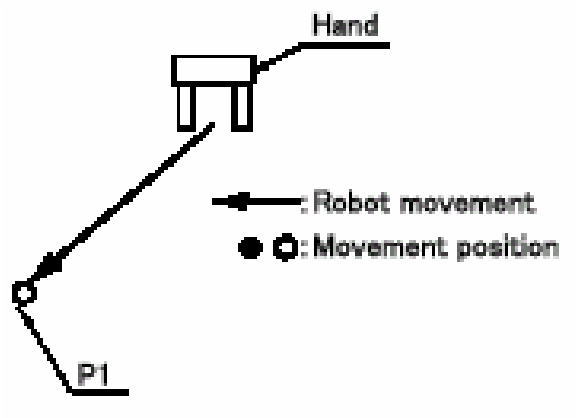
- **MOV P1** : κινήσου ελεύθερα προς το P1
- **MOV P1,-50** : κινήσου ελεύθερα προς το P1,αλλά σταμάτα στο σημείο που βρίσκεται 50 χιλιοστά πριν από το P1.
- **MOV P1 WTH M_OUT(5)=1** : κινήσου ελεύθερα προς το P1 και ταυτόχρονα ενεργοποίησε (ON) την έξοδο 5.
- **MOV P1 WTHIF M_IN(6)=1,SKIP** : κινήσου ελεύθερα προς το P1μόνο εφόσον η είσοδος 6 βλέπει λογικό 1, διαφορετικά αναίρεσε την εκτέλεση της εντολής και προχώρησε στην εκτέλεση της επόμενης γραμμής .
- **MOV P1+P2** : κινήσου ελεύθερα προς το σημείο που προκύπτει από την άθροιση των συντεταγμένων θέσης των σημείων P1 και P2.
- **MOV P1*P2**: ελεύθερη κίνηση προς το P2 σχετική ως προς το P1.

Παράδειγμα

```
1 OVRD 20
5 ACCEL 50,50
10 MOV P1
20 MOV P2,-50
30 MOV P2
40 MOV P3,-100 WTH M_OUT(5)=1
50 MOV P3
60 MOVP3,-100
70 END
```

~~~~~~~~~~~LINEAR INTERPOLATION MOVEMENT~~~~~~~~~~
(Κίνηση γραμμικής παρεμβολής)
Εννολή (MVS)

Το άκρο του βραχίονα, κατά συνέπεια και το τελικό στοιχείο δράσης κινείται γραμμικά από την θέση που βρίσκεται στο προκαθορισμένο σημείο. Μεταξύ δύο σημείων ,η συντομότερη διαδρομή είναι η ευθεία. Αυτό όμως για το βραχίονα δε συνεπάγεται ευκολία στην επίτευξη της πορείας αυτής. Υπολογιστικά ,θα πρέπει να λύσει την εξίσωση της ευθείας και να υπολογίσει τις συντεταμένες όλων των ενδιάμεσων σημείων (όσο το δυνατόν περισσότερων σημείων ώστε να μην έχουμε μεγάλη καθυστέρηση).Έπειτα με συνδυασμό κινήσεων πιθανώς όλων των αρθρώσεων να κινηθεί σε κάθε ένα από τα σημεία αυτά, κατά τέτοιο τρόπο όμως ώστε να μασ δίδει την εντύπωσης μιας γρήγορης ,συντονισμένης ,ομαλής και συνεχούς κίνησης.



- **MVS P1** : κινήσου γραμμικά προς το P1
- **MVS P1,-50** : κινήσου γραμμικά προς το P1,αλλά σταμάτα στο σημείο που βρίσκεται 50 χιλιοστά πριν από το P1.
- **MVS P1 WTH M_OUT(5)=1** : κινήσου γραμμικά προς το P1 και ταυτόχρονα ενεργοποίησε (ON) την έξοδο 5.
- **MVS P1 WTHIF M_IN(6)=1,SKIP** : κινήσου γραμμικά προς το P1μόνο εφόσον η είσοδος 6 βλέπει λογικό 1, διαφορετικά αναίρεσε την εκτέλεση της εντολής και προχώρησε στην εκτέλεση της επόμενης γραμμής .
- **MVS P1+P2** : κινήσου γραμμικά προς το σημείο που προκύπτει από την άθροιση των συντεταγμένων θέσης των σημείων P1 και P2.
- **MVS P1*P2**: γραμμική κίνηση προς το P2 σχετική ως προς το P1.
- **MVS P1 TYPE 0,1** κίνηση προς το P1 με τριών αξόνων γραμμική ορθογώνια παρεμβολή.

Παράδειγμα

```
1 OVRD 20
5 ACCEL 50,50
10 MVS P1
20 MVS P2,-50 TYPE 0,1
30 MVS P2
40 MVS P3,-100 WTH M_OUT(5)=1
50 MVS P3
60 MVS P3,-100
70 END
```

~~~~~~~~~~~CIRCULAR INTERPOLATION MOVEMENT~~~~~~~
(Κίνηση κυκλικής παρεμβολής)
Εντολές MVR, MVR2, MVR3, MVC

Οι εντολές αυτές κινούν τον βραχίονα στην τροχιά του τόξου ενός κύκλου που ορίζεται από τρία (3) σημεία.

Η πρώτη συντάσσεται ως εξής ***MVR P1, P2, P3***. Στην περίπτωση αυτή το άκρο του βραχίονα θα κινηθεί από το αρχικό σημείο P1 **(start point)**και, αφού περάσει από το ενδιάμεσο P2 **(transit point)**, θα καταλήξει στο τελικό P3 **(end point)**, κινούμενο στο τόξο του κύκλου που ορίζεται από τα σημεία P1, P2 και P3.

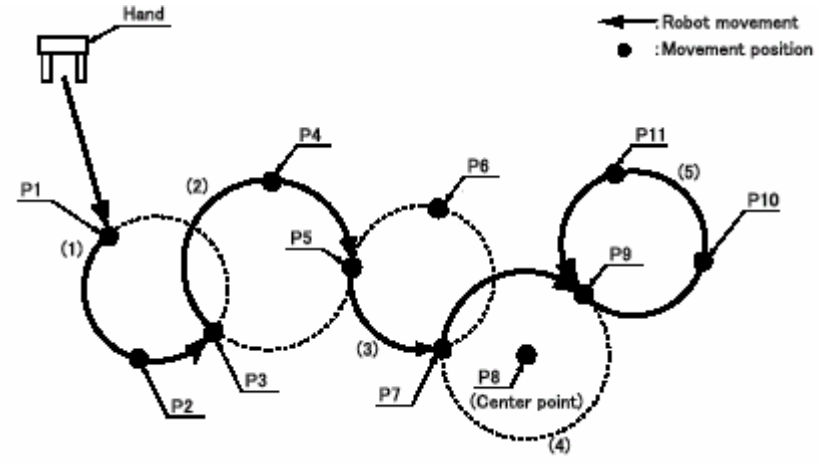
Η δεύτερη εντολή συντάσσεται ως εξής ***MVR2 P1, P3, P2***. Τα τρία αυτά σημεία ορίζουν πάλι ένα κύκλο. Το άκρο του βραχίονα θα κινηθεί γραμμικά στο αρχικό σημείο P1 και θα καταλήξει στο τελικό P3, κινούμενο στο τόξο του κύκλου που έχει οριστεί, αποφεύγοντας όμως να περάσει από το σημείο P2.

Για την τρίτη εντολή έχουμε, ***MVR3 P1, P3, P2***. Εδώ ορίζουμε ένα κύκλο με κέντρο το σημείο P2 και τα δύο άλλα (P1 και P3) να βρίσκονται πάνω στον κύκλο. Το άκρο του βραχίονα θα κινηθεί από το αρχικό σημείο P1 και θα καταλήξει στο τελικό P3, κινούμενο στο τόξο του κύκλου που έχουμε ορίσει.

Εντολή, ***MVC P1, P2, P3*** Η διαφορά είναι ότι εδώ, το άκρο του βραχίονα κινείται κατά μήκος όλου του κύκλου που ορίζουν τα σημεία P1, P2 και P3. Δηλαδή, το άκρο του βραχίονα θα κινηθεί από το αρχικό σημείο P1 και θα καταλήξει πάλι σε αυτό, αφού περάσει από τα άλλα δύο. Δηλαδή εδώ το P1 είναι start και end point και τα άλλα δύο P2,P3 είναι τα ενδιάμεσα (transit points).

Για την καλύτερη κατανόηση όλων των παραπάνω, παραθέτω ένα πρόγραμμα που χρησιμοποιεί τις πιο πάνω εντολές. Στο διάγραμμα που ακολουθεί βλέπουμε την πορεία του άκρου του βραχίονα.

```
1 OVRD 20
5 ACCEL 50,50
10 MVR  P1, P2, P3
20 MVR  P3, P4, P5
30MVR2  P5, P7, P6
40 MVR3  P7, P9, P8
50 MVC  P9, P10, P11
60 END
```



~~~~~~~~~~~~~~~~~~CONTINUOUS MOVEMENT ~~~~~~~~~~~~~~~~~
(Συνεχής κίνηση)
Εντολή CNT

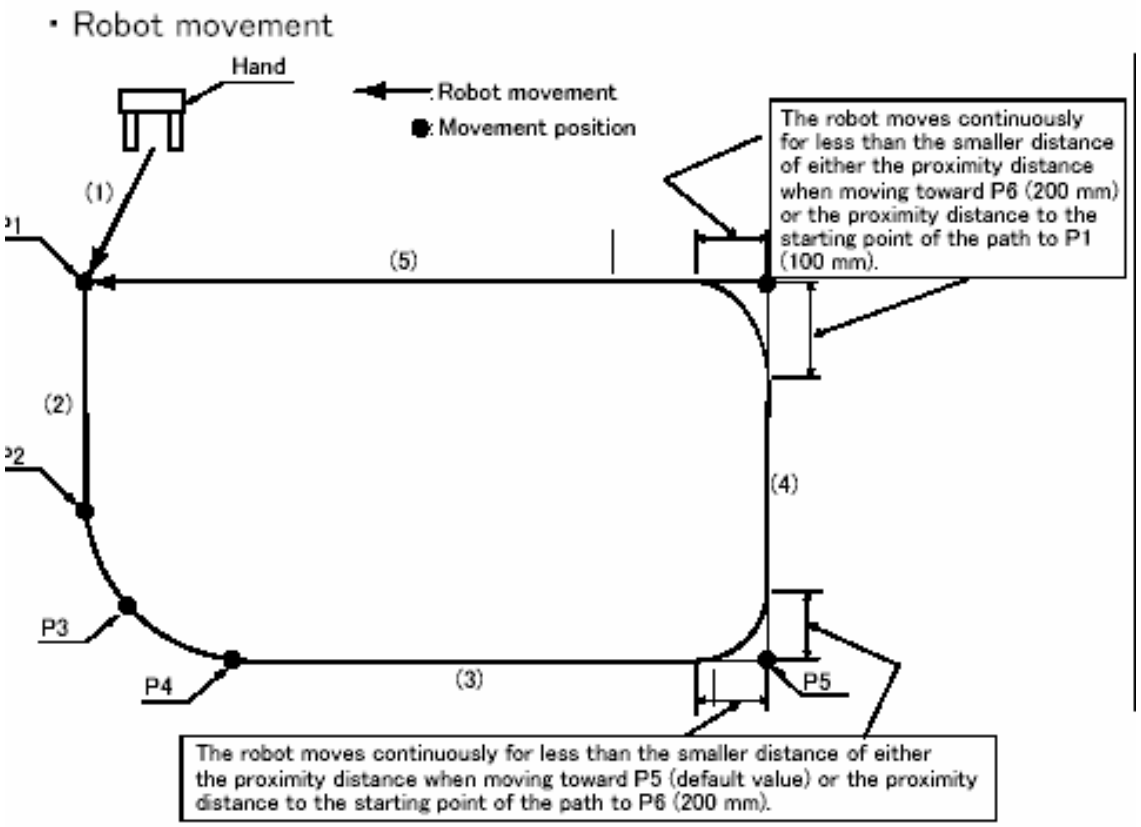
Ορίζοντας μια διαδρομή με αρκετά σημεία που έχουν αποθηκευτεί, με τη λειτουργία του Continuous Movement έχουμε τη δυνατότητα συνεχούς κίνησης του βραχίονα χωρίς stop σε κάθε ένα από τα σημεία αυτά. Η αρχή και το τέλος του Continuous Movement ορίζονται με εντολή και η ταχύτητα κίνησης μεταξύ των ενδιάμεσων σημείων μπορεί να μεταβάλλεται.

Η λειτουργία αυτή σκοπεύει στο να αποφευχθεί το άσκοπο χάσιμο χρόνου που επιφέρει η επιβράδυνση και η στιγμιαία ακινητοποίηση του βραχίονα σε κάθε ένα από τα ενδιάμεσα σημεία. Κίνηση με σταθερή ταχύτητα .Συντομότερη εκτέλεση της εφαρμογής.

Με την εντολή **CNT 1** ορίζουμε μέσα στο πρόγραμμά μας ότι αρχίζει Continuous Movement και με την **CNT 0** το τέλος του.

Με την εντολή **CNT 1,200,100** ορίζουμε ότι ο βραχίονα δε θα περνά από τα ενδιάμεσα σημεία αλλά κοντά σε αυτά, προσπερνόντας τα διαγράφοντας τόξα, όπως φαίνεται στο σχήμα.

10 OVRD 50
20 ACCEL 60,80
30 MOV P1
40 CNT 1
50 MVR P2,P3,P4
60 CNT 1,100,200
70 MVS P5
80 CNT 1,200,100
90 MVS P6
100 CNT 0
110 MVS P1
120 END



~~~~~~~~~~ΕΝΤΟΛΕΣ ΕΛΕΓΧΟΥ ΤΑΧΥΤΗΤΑΣ~~~~~~~~

SPD

Με αυτή την εντολή ορίζουμε ποια είναι η μέγιστη ταχύτητα, σε *mm/sec*, που επιτρέπεται να φτάσει το άκρο του βραχίονα(τελικό στοιχείο δράσης) κατά την εκτέλεση των εντολών που αφορούν την γραμμική και την κυκλική κίνηση. Η σύνταξη της είναι, για παράδειγμα, ***SPD 80***, και δηλώνει μέγιστη ταχύτητα *80 mm/sec*.

OVRD

Χρησιμοποιούμε την εντολή αυτή, για να ορίσουμε την ταχύτητα σε ένα ποσοστό της μέγιστης δυνατής . Εάν δηλαδή γράψουμε, ***OVRD 50***, το άκρο του βραχίονα επιτρέπεται να κινηθεί με ταχύτητα που φτάνει μέχρι το 50% της μέγιστης, που ορίστηκε προηγουμένως.

JOVRD

Συντάσσεται όπως η προηγούμενη εντολή, αλλά χρησιμεύει στο να περιορίσουμε την ταχύτητα κίνησης των αρθρώσεων, σε ένα ποσοστό της μέγιστης.

~~~~ΕΝΤΟΛΗ ΕΛΕΓΧΟΥ ΕΠΙΤΑΧΥΝΣΗΣ-ΕΠΙΒΡΑΔΥΝΣΗΣ~~~~~

ACCEL

Σε αντιστοιχία με τη προηγούμενη εντολή, εδώ ορίζουμε το ποσοστό της μέγιστης επιτάχυνσης και επιβράδυνσης, που επιτρέπουμε στον βραχίονα. Ο λιγότερος χρόνος που χρειάζεται το άκρο του βραχίονα για να φτάσει στην μέγιστη ταχύτητα (ή αντίστοιχα από την μέγιστη ταχύτητα να σταματήσει) είναι 0.2 sec. Έτσι, εάν γράψουμε *ACCEL 40, 80*, ορίζουμε ότι η επιτάχυνση θα φτάνει μέχρι το 40% της μέγιστης και η επιβράδυνση το 80% της μέγιστης. Δηλαδή, τα ποσοστά αυτά σε χρόνους είναι, 0.2/40%=0.5 sec για να επιταχύνει και 0.2/80%=0.25 sec για να επιβραδύνει.

~~~~~~~~~~~~~~~~~~~~~~ΚΑΘΥΣΤΕΡΗΣΕΙΣ~~~~~~~~~~~~~~~~~~~

DLY 2 :καθυστέρηση δύο δευτερολέπτων

WAIT A=15 Το Α είναι μια μεταβλητή (π.χ. A,B,X….) που την τιμή της την ορίζουμε μέσα στη ροή του προγράμματος .Ο βραχίονας ακινητοποιείται εωσότου η μεταβλητή πάρει την τιμή 15.

WAIT M_IN(4)=1 Ο βραχίονας ακινητοποιείται εωσότου η είσοδος 4 δει λογικό 1.Μπορεί λοιπόν στην είσοδο 4 να έχουμε συνδέσει ένα αισθητήριο. Στην περίπτωση αυτή μπλοκάρουμε το βραχίονα εωσότου το αισθητήριο ανιχνεύσει κάτι.

M_OUT(10)=1 DLY 5 Ενεργοποιεί την έξοδο 10 για 5 δευτερόλεπτα.

ΕΝΤΟΛΗ ΕΝΕΡΓΟΠΟΙΗΣΗΣ ΚΙΝΗΣΗΣ ΜΕΓΑΛΗΣ ΑΚΡΙΒΕΙΑΣ
High Accurancy path mode

PREC ON
:
:
PREC OFF

Το μπλοκ εντολών που περιλαμβάνεται στο βρόχο ,εκτελείται με μεγάλη ακρίβεια. Συγκεκριμένα αυξάνονται τα feedback συστήματα (συστήματα ελέγχου-ανατροφοδότηση) του μηχανήματος, επηρεάζοντας βέβαια την ταχύτητά του.

~~~~~~~~~~Εντολές Ελέγχου Ροής Προγράμματος~~~~~~~~~~

Σκοπός

Θα περιγράψουμε εντολές που σχετίζονται με την ροή εκτέλεσης των εντολών του προγράμματος μας. Θα δούμε πως μπορούμε εφόσον χρειαστεί να αλλάζουμε την φυσιολογική σειρά εκτέλεσης του προγράμματός μας , μεταβαίνοντας κατ'επιλογή σε όποια γραμμή θελήσουμε, να επιστρέψουμε πίσω κλπ.. Επίσης να δημιουργήσουμε βρόχους επαναλήψεων (loop) και μπλοκ εντολών που η εκτέλεση τους ή όχι θα εξαρτάται από το εάν ισχύει κάποια συνθήκη, από την τρέχουσα τιμή κάποιας μεταβλητής ή από το αποτέλεσμα κάποιων λογικών πράξεων. Έτσι, θα είμαστε σε θέση να εμπλουτίσουμε τα προγράμματα μας και να υλοποιήσουμε πιο πολύπλοκες εργασίες.

Αναλυτικός Πίνακας Αριθμητικών & Λογικών Πράξεων

Class	Operator	Meaning	Statement example	
Substitution	=	The right side is substituted in the left side.	P1 = P2 P5 = P_CURR P10.Z = 100.0 M1 = 1 STS$="OK"	; Substitute P2 in position variable P1. ; Substitute the current coordinate value in current position variable P5. ; Set the position variable P10 Z coordinate value to 100.0. ; Substitute value 1 in numeric variable M1. ; Substitute the character string OK in the character string variable STS$.
Arithmetic operation	+	Add	P10 = P1 + P2 MOV P8 + P9 M1 = M1 + 1 STS$ =" ERR" +" 001"	; Substitute the results obtained by adding the P1 and P2 coordinate elements to position variable P10. ; Move to the position obtained by adding the position variable P8 and P9 coordinate elements. ; Add 1 to the numeric variable M1. ; Add the character string 001 to the character string ERR and substitute in character string variable STS$.
	-	Subtract	P10 = P1 - P2 MOV P8 - P9 M1 = M1 - 1	; Substitute the results obtained by subtracting the P2 coordinate element from P1 in position variable P10. ; Move to the position obtained by subtracting the P9 coordinate element from the position variable P8. ; Subtract 1 from the numeric variable M1.
	*	Multiply	P1 =P10 * P3 M1 = M1 * 5	; Substitute the relative conversion results from P10 to P3 in position variable P1. ; Multiple the numeric variable M1 value by 5.
	/	Divide	P1 =P10 / P3 M1 = M1 / 2	; Substitute the reverse relative conversion results from P10 to P3 in position variable P1. ; Divide the numeric variable M1 value by 2.
	^	Exponential operation	M1 = M1 ^ 2	; Square the numeric variable M1 value.
	¥	Integer division	M1 = M1¥3	; Divide the numeric variable M1 value by 3 and make an integer (round down).
	MOD	Remainder operation	M1 = M1 MOD 3	; Divide the numeric variable M1 value by 3 and leave redundant.
	-	Sign reversal	P1 = - P1 M1 = - M1	; Reverse the sign for each coordinate element in position variable P1. ; Reverse the sign for the numeric variable M1 value.

Comparison operation	=	Compare whether equal	IF M1 = 1 THEN 200 IF STS$ =" OK" THEN 100	; Branch to line 200 if numeric variable M1 value is 1. ; Branch to line 100 if character string in character string variable STS$ is OK.
	< > or > <	Compare whether not equal	IF M1 < > 2 THEN 300 IF STS$ < >" OK" THEN 900	; Branch to line 300 if numeric variable M1 value is 2. ; Branch to line 900 if character string in character string variable STS$ is not OK.
	<	Compare whether smaller	IF M1 < 10 THEN 300 IF LEN(STS$) < 3 THEN 100	; Branch to line 300 if numeric variable M1 value is less than 10. ; Branch to line 100 if No. of characters in character string STS$ variable is less than 3.
	>	Compare whether larger	IF M1 > 9 THEN 200 IF LEN(STS$) > 2 THEN 300	; Branch to line 200 if numeric variable M1 value is more than 9. ; Branch to line 300 if No. of characters in character string variable STS$ is more than 2.
	= < or < =	Compare whether equal to or less than	IF M1 < = 10 THEN 200 IF LEN(STS$) < = 5 THEN 300	; Branch to line 200 if numeric variable M1 value is equal to or less than 10. ; Branch to line 300 if No. of characters in character string variable STS$ is equal to or less then 5.
	= > or > =	Compare whether equal to or more than	IF M1 > = 11 THEN 200 IF LEN(STS$) > = 6 THEN 300	; Branch to line 200 if numeric variable M1 value is equal to or more than 11. ; Branch to line 300 if No. of characters in character string variable STS$ is equal to or more than 6.
Logical operation	AND	Logical AND operation	M1 = M_INB(1) AND & H0F ; Convert the input signal bit 1 to 4 status and substitute in numeric variable M1. (Input signal bits 5 to 8 remain OFF.)	
	OR	Logical OR operation	M_OUTB(20) = M1 OR & H80 ; Output the numeric variable M1 value to output signal bit 20 to 27. Output bit signal 27 is always ON at this time.	
	NOT	NOT operation	M1 = NOT M_INW(1) ; Reverse the status of input signal bit 1 to 16 to create a value, and substitute in numeric variable M1.	
	XOR	Exclusive OR operation	M2 = M1 XOR M_INW(1) ; Obtain the exclusive OR of the states of M1 and the input signal bits 1 to 16, convert into a value and substitute in numeric variable M2.	
	< <	Logical left shift operation	M1 = M1 < < 2 ; Shift numeric variable M1 two bits to the left.	
	> >	Logical right shift operation.	M1 = M1 > > 1 ; Shift numeric variable M1 bit to the right.	

GOTO

Η κλασικότερη εντολή ελέγχου της ροή ενός προγράμματος. Για παράδειγμα, γράφοντας *GOTO 50*, η εκτέλεση του προγράμματος συνεχίζει από την γραμμή 50, χωρίς κανένα έλεγχο.

Μία εναλλακτική γραφή, για λόγους ευχρηστίας, είναι, αντί για τον αριθμό της γραμμής, να δώσουμε κάποια ετικέτα. Ένα παράδειγμα είναι:

```
      …
30 GOTO *LABEL
      …
70 *LABEL
80 MOVS P1
      …
```

Στο παραπάνω παράδειγμα η γραμμή 70 έχει πάρει το όνομα *LABEL*. Έτσι στη γραμμή 30 μπορούμε να γράψουμε είτε *GOTO 70* είτε *GOTO *LABEL*, έχοντας το ίδιο αποτέλεσμα. Προσοχή στον αστερίσκο, ο οποίος δηλώνει ότι το όνομα που ακολουθεί είναι ετικέτα κάποιας γραμμής.

FOR, NEXT

Ο πιο απλός τρόπος για τη δημιουργία ενός βρόχου επαναλήψεως.(Loop).
Η σύνταξη της φαίνεται στο παρακάτω παράδειγμα:

```
FOR M1=0 TO 10 STEP 2  επίσης αντί για τον αριθμό 10 μπορούμε να
:                      χρησιμοποιήσουμε μια μεταβλητή ,την τιμή
 BLOCK εντολών         οποίας να ελέγχουμε κάπου μέσα στο
:                      πρόγραμμα μας πχ. FOR M1=0 TO A STEP 2
NEXT M1
```

Εκτελεί το BLOCK εντολών, ωσότου η μεταβλητή M1 φτάσει από την τιμή 0 στην τιμή 10. Ο ρυθμός αύξησης (βήμα) της μεταβλητής είναι 2 και αυξάνεται κάθε φορά που εκτελείται η εντολή *NEXT*. Εάν παραλείψουμε το *STEP*, ο ρυθμός αύξησης είναι 1. Μόλις η μεταβλητή M1 ξεπεράσει την καθορισμένη τιμή 10, η εκτέλεση του προγράμματος μας συνεχίζεται στην επόμενη γραμμή από αυτή της εντολής *NEXT*.

IF THEN (ELSE)

Πρόκειται για εντολή ελέγχου, η οποία εξετάζει εάν αληθεύει μια ορισμένη συνθήκη και ανάλογα εκτελεί ή όχι κάποιες εντολές.

Συντάσσεται σε μία γραμμή ως εξής:

IF (Συνθήκη ελέγχου) THEN εντολή 1 ELSE εντολή 2

Για παράδειγμα, *IF (M1=1) THEN 80 ELSE M2=2*. Εδώ, ελέγχουμε την τιμή της μεταβλητής M1. Εάν είναι ίση με 1, τότε η ροή του προγράμματος συνεχίζει από την γραμμή 80 (προσέξτε ότι το *GOTO* παραλείπεται), αλλιώς η μεταβλητή M2 γίνεται

ίση με 2. Μπορούμε, σε περίπτωση που δεν χρειάζεται κάποια ενέργεια, όταν η συνθήκη δεν αληθεύει, να παραλείπουμε το κομμάτι της *ELSE*. Επίσης, για σύνθετες Συνθήκες Ελέγχου, μπορούμε να χρησιμοποιήσουμε τους λογικούς τελεστές *OR*, *AND*, *NOT*, *XOR*.

Όταν θέλουμε, αντί για μία εντολή, να εκτελεστεί κάποιο σετ εντολών, η σύνταξη γίνεται ως εξής:

IF* (Συνθήκη ελέγχου) *THEN
:
BLOCK εντολών 1
:
ELSE
:
BLOCK εντολών 2
:
ENDIF

Σε περίπτωση που το *ELSE* παραληφθεί, η σύνταξη γίνεται:

IF* (Συνθήκη ελέγχου) *THEN
:
BLOCK εντολών
:
ENDIF

SELECT, CASE

Με την εντολή αυτή, επιλέγουμε μία μεταβλητή (πχ.M1) την τιμή της οποίας ελέγχουμε μέσα από το πρόγραμμά μας. Αναλόγως λοιπόν της τρέχουσα τιμής της, εκτελείται και κάποιο διαφορετικό BLOCK εντολών. Συντάσσεται ως εξής:

```
SELECT M1
        CASE 1
        BLOCK εντολών 1
        BREAK
                CASE 2
                BLOCK εντολών 2
                BREAK
        CASE 5
        BLOCK εντολών 3
        BREAK
                DEFAULT
                BLOCK εντολών 4
                BREAK
END SELECT
```

Έτσι, εάν η τιμή της μεταβλητής M1 είναι 1, θα εκτελεστεί το πρώτο σετ εντολών. Εάν είναι 2 το δεύτερο σετ εντολών, εάν είναι 5 το τρίτο σετ εντολών. Σε όλες τις άλλες περιπτώσεις (π.χ. η τιμή της M1 είναι 3) θα εκτελεστεί το τέταρτο σετ εντολών. Εάν δεν συμπεριλάβουμε BLOCK εντολών DEFAULT και η τιμή της μεταβλητής δεν αντιστοιχεί στην τιμή κάποιου CASE ,αυτονόητο είναι ότι δεν θα εκτελεστεί κανένα από αυτά και το πρόγραμμα θα συνεχίσει στην επόμενη γραμμή.

WHILE, WEND

Η σύνταξη αυτής της εντολής είναι:

***WHILE* (Συνθήκη ελέγχου)**
:
BLOCK εντολών
:
WEND

Το BLOCK εντολών εκτελείται όσο ισχύει η συνθήκη ελέγχου. Αυτό σημαίνει ότι μπορεί να μην εκτελεστεί ποτέ, εάν κατά τον πρώτο έλεγχο η συνθήκη δεν αληθεύει. Μπορεί, όμως, να εκτελείται και συνεχώς (ατέρμον βρόχος) όταν, για παράδειγμα το σετ εντολών δεν επηρεάζει την τιμή της μεταβλητής που βρίσκεται στη συνθήκη ελέγχου.

Ένα παράδειγμα φαίνεται παρακάτω.

M1=3
***WHILE* (M1>=0) AND (M1<=10)**
:
BLOCK εντολών
:
M1=M1+1
WEND

Εδώ, το BLOCK εντολών θα εκτελείται μέχρι η μεταβλητή M1 να ξεπεράσει την τιμή 10. Προσέξτε ότι η αρχική τιμή της μεταβλητής ορίζεται έξω από τον βρόγχο, ίση με 3. Οπότε επιτρέπει στον βρόχο να ξεκινήσει. Επίσης, μέσα στο BLOCK εντολών έχουμε αύξηση της τιμής της κατά 1, κάθε φορά που εκτελείται ο βρόχος. Οπότε, κάποια στιγμή, θα ξεπεράσει την τιμή 10 και η εκτέλεση του προγράμματος θα προχωρήσει στην γραμμή που έπεται της *WEND*.

Υπορουτίνες

Οι υπορουτίνες είναι ένα είδος υποπρογραμμάτων, όπου η εκτέλεση τους ενεργοποιείται εάν πληρούνται κάποιες συνθήκες. Η χρήση τους αποσκοπεί στην κατάτμηση του προγράμματος και την ομαδοποίηση κάποιων τμημάτων κώδικα ,για την επίτευξη καλύτερης οργάνωσης σε μεγάλα προγράμματα. Με τη δημιουργία υπορουτίνων ,ο κώδικάς μας γίνεται ευανάγνωστος και περισσότερο ξεκάθαρος για τον προγραμματιστή που μπορεί ευκολότερα να οργανώσει τη σκέψη του.

GOSUB

Όταν συναντήσουμε στο πρόγραμμα μας αυτή την εντολή, η εκτέλεση του προγράμματος μεταβαίνει στην σειρά που μας υποδεικνύεται. Λειτουργεί δηλαδή περίπου όπως την εντολή *GOTO*. Η κύρια διαφορά όμως είναι, ότι μόλις εκτελεστεί το τμήμα των εντολών που αποτελεί την υπορουτίνα, ο έλεγχος του προγράμματος

επιστρέφει στην σειρά απ' όπου αρχικά καλέσαμε την συγκεκριμένη υπορουτίνα. Το παρακάτω παράδειγμα είναι διαφωτιστικό:

```
…
80 GOSUB *LABEL
90 MOV P3
…
130 END
…
500 *LABEL
510 MOVS P1
…
550 RETURN
```

Στο παραπάνω παράδειγμα, οι πρώτες γραμμές, μέχρι τη γραμμή 130 αποτελούν το κυρίως πρόγραμμα. Αυτό δηλώνεται με την εντολή *END*. Από εκεί και κάτω, μπορούν να ακολουθήσουν μια σειρά από υποπρογράμματα, όπου, εάν χρειαστεί, θα τα καλέσουμε χρησιμοποιώντας κατάλληλα την εντολή *GOSUB*. Εδώ, στη γραμμή 500 με ετικέτα *LABEL*, ξεκινάει η υπορουτίνα, η οποία καλούμε, και τελειώνει στη σειρά 550 με την εντολή *RETURN*.

Έτσι, όταν η εκτέλεση του κυρίως προγράμματος φτάσει στην γραμμή 80, τότε θα μεταβεί στην υπορουτίνα *LABEL*. Θα εκτελεστούν οι γραμμές από 500 έως 550 και στη συνέχεια η εκτέλεση του προγράμματος θα συνεχίσει κανονικά στη γραμμή 90.

Προσέχουμε πάντα να οριοθετούμε την υπορουτίνα μας, βάζοντας στο τέλος την εντολή *RETURN*. Επίσης, θα μπορούσαμε εναλλακτικά της εντολής *GOSUB *LABEL*, να γράφαμε *GOSUB 500*.

ΕΚΦΡΑΣΕΙΣ :

ON, GOSUB

ON M_IN(5)=1 GOSUB 500
*ON M_IN(5)=1 GOSUB *LABEL*

ON COM(1) GOSUB 300
*ON COM(1) GOSUB *lABEL*

ON, GOTO

ON M_IN(5)=1 GOTO 500
*ON M_IN(5)=1 GOTO *LABEL*

ON COM(1) GOTO 300
*ON COM(1) GOTO *lABEL*

ON M1 GOTO 100,200,300,350,320,…… αναλόγως την τιμή της μεταβλητής M1(1,2,3,4,5,…) μετάβαση στη γραμμή 100,200,300,……..