ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΤΟΜΕΑΣ ΜΗΧΑΝΟΛΟΓΙΚΩΝ ΚΑΤΑΣΚΕΥΩΝ & ΑΥΤΟΜΑΤΟΥ ΕΛΕΓΧΟΥ

ΕΝΑΡΞΗ ΛΕΙΤΟΥΡΓΙΑΣ KR 15/1

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ονοματεπώνυμο: Άγγελος Στάθης

Αριθμός Μητρώου: 02109672

Υπεύθυνος Καθηγητής: Κωνσταντίνος Κυριακόπουλος

Ημερομηνία: 30- 7- 2018

1. Εισαγωγή

Στο Εργαστήριο Αυτομάτου Ελέγχου και Ρυθμίσεως Μηχανών & Εγκαταστάσεων του Τομέα Μηχανολογικών Κατασκευών & Αυτομάτου Ελέγχου βρίσκεται ο ρομποτικός βραχίονας KR 15/1 της εταιρίας Kuka. Ο βραχίονας είναι 6 βαθμών ελευθερίας και ελέγχεται από τον ελεγκτή KRC1 (Kuka Robot Controller 1).

Επιθυμία του εργαστηρίου είναι η έναρξη λειτουργίας του βραχίονα αυτού και η ένταξη του στο εκπαιδευτικό πρόγραμμα αλλά και η ερευνητική χρήση του, σε σύνδεση με έναν εξωτερικό υπολογιστή.

Η επιθυμία αυτή μπορεί να ικανοποιηθεί με την δημιουργία κατάλληλου λογισμικού που επιτρέπει την επικοινωνία ανάμεσα στους δύο υπολογιστές. Το λογισμικό αυτό συμπερασματικά πρέπει να ακολουθεί το μοντέλο του πελάτη-διακομιστή (client-server).

Έχοντας ως δεδομένο ότι στόχος του εργαστηρίου είναι ο έλεγχος του βραχίονα κατά θέση, ταχύτητα και δύναμη, ο ελεγκτής πρέπει να έχει τον ρόλο του πελάτη και ο εξωτερικός υπολογιστής τον ρόλο του εξυπηρετητή.

Η κατανόηση του συστήματος του βραχίονα με τον ελεγκτή, της υλοποίησης του λογισμικού και της γλώσσας προγραμματισμού της εταιρίας και των δυνατοτήτων που αυτά προσφέρουν είναι επομένως αναγκαία.

Εξίσου αναγκαίο είναι το πρόγραμμα του εξυπηρετητή να μπορεί να συνδυαστεί με τον υπάρχοντα κώδικα του εργαστηρίου ώστε στο μέλλον να μπορεί να ενσωματωθεί σε ένα μεγαλύτερο πακέτο κώδικα που αφορά τα ρομπότ.

Επομένως πρέπει να υλοποιηθεί η επικοινωνία με την ανάπτυξη δύο προγραμμάτων.

Το πρόγραμμα πελάτη, γραμμένο στην γλώσσα του βραχίονα, αναλαμβάνει να φέρει εις πέρας τις κινήσεις κινήσεις που απαιτούνται καθώς και να επιστρέφει σημαντικά δεδομένα, όπως για παράδειγμα την τρέχουσα θέση των αξόνων και την ταχύτητά τους, καθώς επίσης να ανταποκρίνεται στις απαιτήσεις ελέγχου, στο βαθμό που αυτό είναι δυνατό.

Το πρόγραμμα του εξυπηρετητή, γραμμένο στην γλώσσα προγραμματισμού που χρησιμοποιείται στο εργαστήριο, ανταποκρίνεται στην επικοινωνία, μεταφέρει τις χρήσιμες πληροφορίες και είναι δομημένο με συγκεκριμένο τρόπο. Ο τρόπος αυτός δίνει την δυνατότητα επέκτασης του κώδικα από επόμενους χρήστες, μέσω της απλότητας και της αμεσότητας του, ώστε ο κάθε ένας να εφαρμόζει τον έλεγχο που επιθυμεί.

Πέραν από την ανάπτυξη του λογισμικού, είναι ιδιαίτερης σημασίας η πλήρης και ουσιαστική ανάλυση του συστήματος βραχίονα και ελεγκτή. Αυτό επιτυγχάνεται με την δημιουργία ενός εγχειριδίου χρήσης, το οποίο περιλαμβάνει όλες τις απαραίτητες πληροφορίες για τον τρόπο λειτουργίας αυτού. Η σημασία αυτού έγκειται, για την ταχύτερη κατανόηση, την αποφυγή λαθών αλλά και να αποτελεί βάση για την περαιτέρω χρήση του συνόλου των δυνατοτήτων εις βάθος, εφόσον είναι επιθυμία του χρήστη να εντρυφήσει και να αποκομίσει τα μέγιστα.

Συνοψίζοντας, το πρόβλημα αποτελείται από την ανάπτυξη του λογισμικού στα δύο μέρη και την παρουσίαση του συστήματος της Kuka στον χρήστη.

Η επίλυση του προβλήματος αυτού είναι ιδιαίτερης σημασίας για το εργαστήριο.

Στο επίπεδο του εκπαιδευτικού προγράμματος δίνει την δυνατότητα, λόγω της θέσης του, να πραγματοποιείται το εργαστήριο του οπτικού ελέγχου από την μία πλευρά του. Αυτό είναι εφικτό διότι μπορεί να συνεργαστεί με τον ταινιόδρομο και να εκτελέσει την πληθώρα κινήσεων που απαιτούνται με μεγάλη ακρίβεια. Από την άλλη πλευρά του, υπάρχει διαθέσιμος χώρος που μπορούν τα μέλη του εργαστηρίου να εκμεταλλευτούν όπως επιθυμούν, με το σκεπτικό επίσης ότι μπορούν να χρησιμοποιούν στην απόληξη ένα εργαλείο της επιλογής τους.

Σε επίπεδο διπλωματικών εργασιών, μπορούν οι φοιτητές να επιβεβαιώσουν τις γνώσεις τους στην ρομποτική, να αναπτύξουν τους δικούς τους αλγορίθμους ελέγχου και να κατανοήσουν εις βάθος τα ρομποτικά συστήματα.

Η διπλωματική εργασία χωρίζεται σε τέσσερα επιμέρους κεφάλαια.

Στο κεφάλαιο Ανάλυση του Προβλήματος, περιγράφεται συνολικά το σύστημα που χρησιμοποιήθηκε τόσο από άποψη λογισμικού όσο και από τεχνικού εξοπλισμού. Επίσης ορίζεται και ο στόχος της διπλωματικής εργασίας.

Στο κεφάλαιο Τεχνικά Θέματα, περιλαμβάνεται ο οραματισμός της αρχιτεκτονικής για την λύση του προβλήματος. Γίνεται αναφορά στους περιορισμούς που υπάρχουν καθώς και στα αποτελέσματα που έφερε η τελική έκβαση.

Στο κεφάλαιο Ανάλυση Λογισμικού υπάρχει ολοκληρωμένο το εγχειρίδιο χρήσης του συστήματος, ο κώδικας που αναπτύχθηκε καθώς και η ανάλυση του πρωτοκόλλου επικοινωνίας. Το κεφάλαιο αυτό, το οποίο είναι το μεγαλύτερο και σημαντικότερο, αποτελεί και την αναφορά για τον χρήστη που επιθυμεί να χρησιμοποιήσει τον βραχίονα και να τον ελέγξει.

Στο κεφάλαιο Εναπομείναντα Θέματα γίνεται αναφορά της προοπτικής βελτίωσης του κώδικα καθώς και τεχνικά θέματα που έχουν προκύψει.

2. Περιεχόμενα

Πίνακας περιεχομένων

3. Ανάλυση του Προβλήματος

Η ανάλυση του προβλήματος του ελέγχου του βραχίονα KR 15/1 της Kuka από έναν εξωτερικό υπολογιστή συνίσταται στην περιγραφή του συστήματος που χρησιμοποιήθηκε και στον στόχο αυτού.

3.1. Περιγραφή Συστήματος

Αρχικά γίνεται μία συνολική ανάλυση του συστήματος που χρησιμοποιήθηκε τόσο από άποψη τεχνικού εξοπλισμού όσο και λογισμικού.

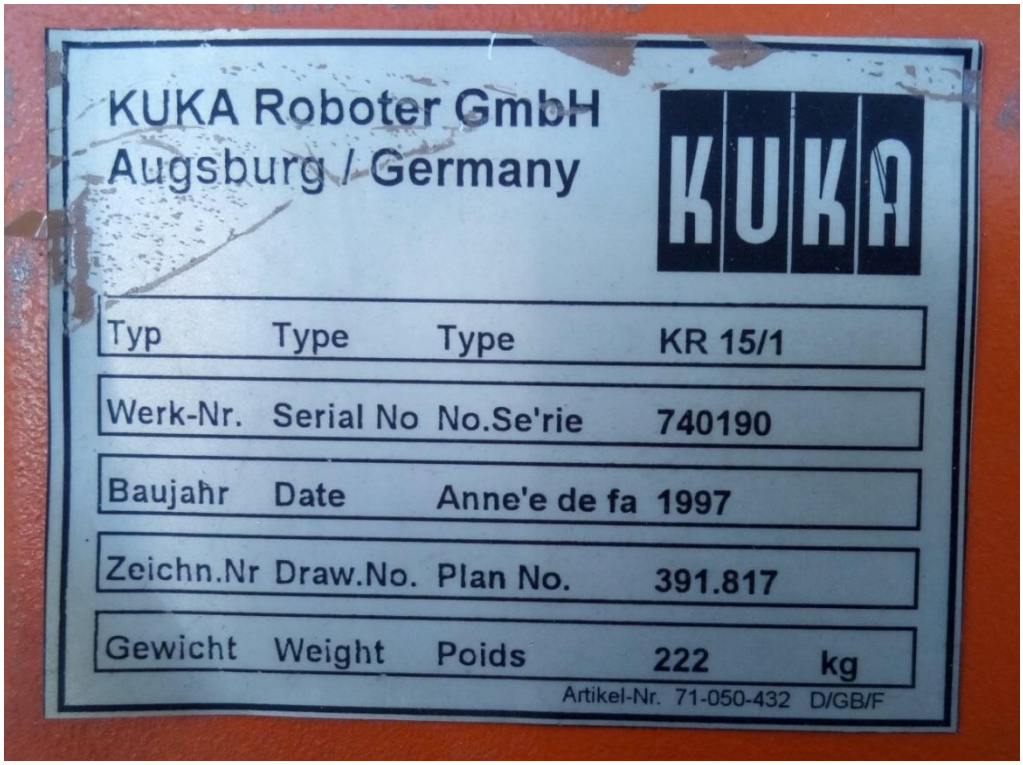
3.1.1. Τεχνικός Εξοπλισμός

Από άποψη εξαρτημάτων και υλικού (**hardware**) χρησιμοποιήθηκαν:

- **Ο ρομποτικός βραχίονας της KUKA KR15/1 (6 βαθμών ελευθερίας) με προδιαγραφές:**

  Serial# 740190

  Weight 222kg



- **Ο ελεγκτής (controller) της KUKA KRC1 με προδιαγραφές:**

Type	(V)KRC1
Serial#	02962
Artikel#	71039279
Date	1997
Plan#	392.700-85.001
1	
Supply Volt	3x400 V
Frequency	50/60 Hz
Rated Current	6 A
Mainsfuse	16 A
Weight	136 kg

- **Φορητός υπολογιστής HP Pavilion dv5000 με προδιαγραφές:**

Επεξεργαστής (CPU):	Genuine Intel® CPU T2500 @2.00GHz
Μνήμη (RAM):	2.0 GB
Γραφικά (GPU):	NVidia GeForce GO 7400
Τύπος Συστήματος (OS type):	32-bit

- **Μετατροπέας USB 2.0 σε 9-pin D-SUB (male to male):**

  pl2303 converter

- **Καλώδιο Null Modem 9-pin D-SUB (female to female)**

3.1.2. Λογισμικό

Από άποψη λογισμικού (**software**) χρησιμοποιήθηκαν:

- **Για τον ελεγκτή (controller) KRC1:**

  Λειτουργικό σύστημα:

  - Microsoft Windows 95

  Εγκατεστημένα τα προγράμματα:

  - Ikarus Antivirus
  - KRC1 V2.3.24 SP07
    - **GUI VERSION V2.3.33**
    - **KERNEL SYSTEM VERSION KS V2.76_23**
    - **ROBOT TYPE #KR15_1 floor ZH01**
    - **R1 Mada V14.4.0/KUKA 2.3**

  για προγραμματισμό στην γλώσσα KRL (Kuka Robot Language)

- **Για τον φορητό υπολογιστή HP Pavilion dv5000:**

  Λειτουργικό σύστημα:

  - Ubuntu 16.04 LTS

  Εγκατεστημένα τα προγράμματα:

  - Conda 4.3.21 (Python)

    Με πακέτα στο περιβάλλον: /home/user/anaconda3/envs/KUKA:

    - certify 2016.2.28 py36_0
    - mkl 2017.0.3 0
    - numpy 1.13.1 py36_0
    - openssl 1.0.2l 0
    - pip 9.0.1 py36_1
    - pyserial 2.7 py36_0
    - python 3.6.2 0
    - readline 6.2 2
    - setuptools 36.4.0 py36_0
    - sqlite 3.13.0 0
    - tk 8.5.18 0
    - wheel 0.29.0 py36_0
    - xz 5.2.3 0

- zlib 1.2.11 0

για προγραμματισμό στην γλώσσα Python 3

Ολοκληρωμένο περιβάλλον ανάπτυξης (integrated development environment, IDE):

- Atom 1.22.1

Και το πρόγραμμα που διαβάζει την σειριακή θύρα:

- CuteCom

3.2. Στόχος

Στόχος της διπλωματικής εργασίας είναι η επικοινωνία του εξωτερικού υπολογιστή με τον ελεγκτή KRC1 για τον έλεγχο θέσης, ταχύτητας και δύναμης του κάθε άξονα του ρομποτικού βραχίονα KR 15/1 από πρόγραμμα του πρώτου.

Η επικοινωνία επιτυγχάνεται μέσω της σειριακής σύνδεσης RS-232, χρησιμοποιώντας τον μετατροπέα USB σε 9-pin D-SUB για τον εξωτερικό υπολογιστή και την αντίστοιχη θύρα του δεύτερου, με παρεμβαλλόμενο το ειδικό καλώδιο Null Modem για να επιτευχθεί η ζεύξη.

Από την μεριά του εξωτερικού υπολογιστή χρησιμοποιείται η γλώσσα προγραμματισμού Python με το πακέτο Pyserial το οποίο περιέχει τις απαραίτητες δομές για την σειριακή σύνδεση , καθώς και το πακέτο numpy για την εκτέλεση των μαθηματικών πράξεων που μπορεί να χρειαστούν στο μέλλον, για παράδειγμα την επίλυση της κινηματικής του βραχίονα.

Από την μεριά του ελεγκτή η γλώσσα προγραμματισμού KRL περιέχει ήδη όλες εκείνες τις εντολές που χρειάζονται για την επικοινωνία αυτή.

4. Τεχνικά Θέματα

4.1. Οραματισμός Αρχιτεκτονικής

Για να επιτευχθεί ο έλεγχος θέσης, ταχύτητας και δύναμης των αρθρώσεων του ρομποτικού βραχίονα, είναι αναγκαία η ανάπτυξη κατάλληλου λογισμικού και από την μεριά του εξωτερικού υπολογιστή και από την μεριά του ελεγκτή. Τα προγράμματα αυτά θα πρέπει να επικοινωνούν μεταξύ τους σε πραγματικό χρόνο (real-time) μέσω τις σειριακής θύρας και να δουλεύουν ταυτόχρονα (τύπου client-server).

Θα πρέπει να δίνεται η δυνατότητα στον χρήστη να μπορεί με ευκολία να επιλέξει εκείνος όποιον έλεγχο θέλει να εφαρμόσει, είτε κατά το αξονικό σύστημα συντεταγμένων είτε κατά το καρτεσιανό. Το πρόγραμμα θα πρέπει να επιστρέφει όλες εκείνες τις απαραίτητες πληροφορίες στον μικρότερο δυνατό χρονικό διάστημα. Επίσης οι πληροφορίες αυτές θα πρέπει να έχουν την κατάλληλη δομή για να είναι άμεσα χρήσιμες στον χρήστη και στους στόχους του.

Για τον λόγο αυτό το πρόγραμμα από την μεριά του εξωτερικού υπολογιστή θα πρέπει να είναι δομημένο με τρόπο τέτοιον ώστε:

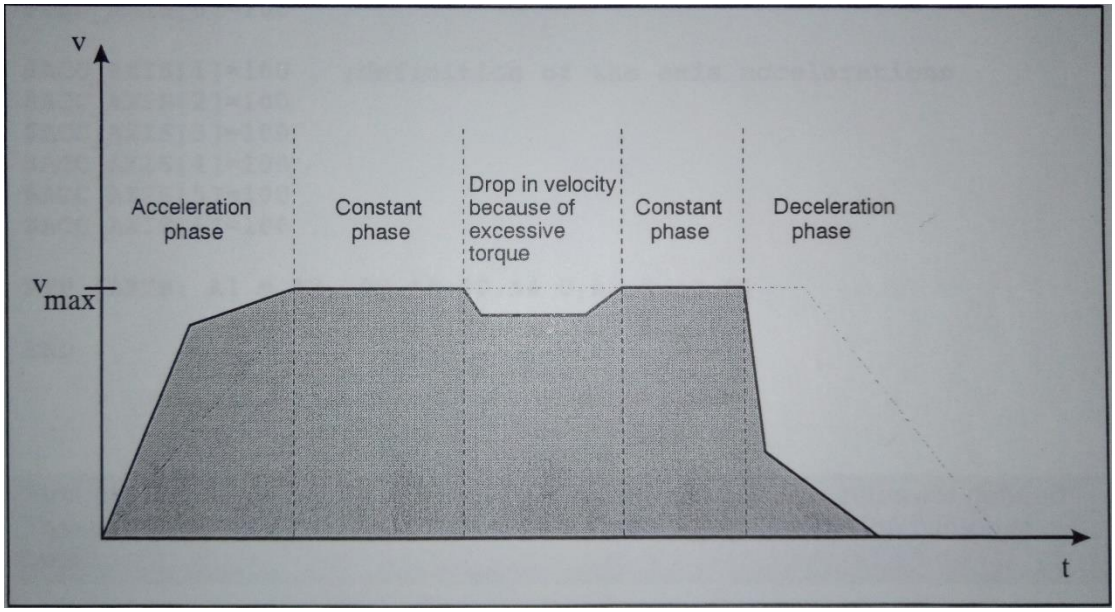
- Ο χρήστης να μπορεί με ευκολία και ταχύτητα να το κατανοήσει
- Να μπορεί να προσθέσει με ευκολία τις δικές του συναρτήσεις (functions) ελέγχου, οι οποίες να συνεργάζονται με τις υπάρχουσες
- Να υπάρχει η δυνατότητα μεταβολής του για μελλοντικές βελτιώσεις – διαφοροποιήσεις

Από την μεριά του ελεγκτή, θα πρέπει το πρόγραμμα:

- Να δίνει την δυνατότητα επιλογής τύπου ελέγχου
- Να μπορεί να προσφέρει όλες εκείνες τις πληροφορίες τις οποίες ο χρήστης θεωρεί χρήσιμες
- Να μπορεί να κατανοηθεί με ευκολία, για τυχούσες βελτιώσεις και αλλαγές
- Να είναι όσο το δυνατόν ταχύτερο, καθότι ο υπολογιστής του ελεγκτή είναι αρκετά παλαιότερος του εξωτερικού

4.2. Περιορισμοί

Το λογισμικό του ελεγκτή έχει δημιουργηθεί με τέτοιον τρόπο κατά τον οποίο δίνεται μόνο η δυνατότητα ελέγχου θέσης του ρομπότ. Υπάρχουν μόνο ως μεταβλητές οι θέσεις στο αξονικό σύστημα συντεταγμένων και στο καρτεσιανό. Δεν δίνεται η δυνατότητα ελέγχου της ταχύτητας ούτε και της δύναμης, παρά μόνο μπορούν να δοθούν τιμές σε μορφή ποσοστού των μέγιστων ταχυτήτων και των επιταχύνσεων για κάθε άξονα. Αυτό συμβαίνει διότι ο ελεγκτής σε κάθε εντολή κίνησης υπολογίζει το προφίλ της ταχύτητας αυτόματα χωρίς να δίνει παραπάνω ελευθερίες στον χρήστη. Παρακάτω φαίνεται ο τρόπος με τον οποίο λειτουργεί:



Για την σύνδεση μέσω της σειριακής θύρας, ο ελεγκτής χρησιμοποιεί το πρωτόκολλο 3964R της Siemens το οποίο αναλύεται παρακάτω με όλες του τις λεπτομέρειες, ενώ το πρωτόκολλο XON/XOFF δυστυχώς δεν έχει εφαρμοστεί, λόγω της παλαιότητας του ελεγκτή. Επιπροσθέτως τα άλλα δύο διαθέσιμα πρωτόκολλα (SRVT, WTC) δεν χρησιμοποιήθηκαν, διότι δεν υπάρχουν πληροφορίες για τον τρόπο λειτουργίας τους.

Επίσης για την σειριακή σύνδεση, το BAUD = 115200 είναι η μέγιστη επιτρεπτή ταχύτητα μεταφοράς δεδομένων, από την πλευρά του ελεγκτή.

Η σειριακή θύρα είναι μια ασύγχρονη θύρα, επομένως ο έλεγχος σε πραγματικό χρόνο δεν είναι εφικτός παρά μόνο ανά μεγάλα (για τα δεδομένα του ελέγχου) χρονικά διαστήματα.

4.3. Επίλυση

Δημιουργήθηκε επομένως λογισμικό στον ελεγκτή το οποίο εφαρμόζει τον έλεγχο θέσης, ενώ δίνεται η δυνατότητα στον χρήστη να επιλέξει τις μέγιστες ταχύτητες και επιταχύνσεις. Επιλέχθηκε μόνο το αξονικό σύστημα συντεταγμένων για χρήση, με σκοπό την ελάφρυνση των πράξεων που χρειάζεται να εκτελεί ο ελεγκτής για τον υπολογισμό των κινήσεων αλλά και για διάυγεια για τον χρήστη.

Στον εξωτερικό υπολογιστή το λογισμικό καλύπτει τις παραπάνω ανάγκες με επιπρόσθετη την υλοποίηση του συγκεκριμένου σειριακού πρωτοκόλλου επικοινωνίας.

4.4. Αποτελέσματα

Πραγματοποιήθηκαν μετρήσεις ώστε να βρεθεί για ποιες τιμές του βήματος, της ταχύτητας και της επιτάχυνσης η κίνηση είναι ομαλή και ο χρόνος επικοινωνίας μειώνεται όσο το δυνατόν περισσότερο.

Αναλυτικά:

Βήμα (°)	Ταχύτητα (%)	Επιτάχυνση (%)	Hiccups	Μέσος Χρόνος (s)
0,1	1	1	έχει, αλλά δεν φαίνονται	0,137
	1	2	αρκετά/ταλαντώσεις	0,11
	1	3	αρκετά	0,093
	1	4	αρκετά	0,087
	1	5	πολλά	0,083
	3	1	πολλά/ταλαντώσεις	0,115
	3	2	πολλά	0,092
	3	5	πολλά	0,077
	3	10	πολλά	0,075
	3	15	πολλά	0,077
0,5	1	1	smooth	0,427
	1	2	smooth	0,311
	1	3	smooth	0,251
	1	4	smooth	0,219
	1	5	smooth	0,198
	1	8	smooth	0,174
	3	3	1	0,138
	3	4	2	0,122
	3	5	2	0,126
	3	8	2	0,101
	5	3	smooth	0,123
	5	5	2	0,108
	5	8	2	0,091
	5	11	6	0,088
	5	14	7	0,081
	7	3	3	0,124
	7	5	2	0,104
	7	8	4	0,092
	7	11	8	0,101
	7	14	5	0,08
1	1	5	smooth	0,344
	1	8	smooth	0,317
	1	11	smooth	0,316
	1	14	smooth	0,317
	3	8	smooth	0,15
	3	11	2	0,133
	3	14	smooth	0,119

	3	17	2	0,14
	3	20	smooth	0,116
	5	5	2	0,134
	5	8	1	0,11
	5	11	1	0,093
	5	14	1	0,098
	5	17	3	0,099
	5	20	1	0,09
	7	5	1	0,129
	7	8	2	0,108
	7	11	2	0,098
	7	14	3	0,095
	7	20	2	0,092

5. Ανάλυση Λογισμικού

Στο κεφάλαιο αυτό γίνεται μια πλήρης ανάλυση του λογισμικού που χρησιμοποιείται, με έμφαση στον ελεγκτή του ρομποτικού βραχίονα. Στόχος είναι να περιλαμβάνει όλες εκείνες τις απαραίτητες πληροφορίες για άμεση χρήση της δομής client-server από τον χρήστη.

Το κεφάλαιο αυτό χωρίζεται σε τρεις ενότητες.

Η πρώτη ενότητα αποτελεί το εγχειρίδιο χρήσης (Operating Manual) του ελεγκτή του ρομπότ. Γίνεται ανάλυση τόσο κατά hardware όσο και κατά software για την εξοικείωση του χρήστη.

Η δεύτερη ενότητα περιλαμβάνει τον κώδικα αυτούσιο σε KRL και Python που αναπτύχθηκε για την δομή αυτή. Η γνώση των λειτουργικών συστημάτων Windows και Linux καθώς και της γλώσσας προγραμματισμού Python θεωρείται δεδομένη.

Η τρίτη ενότητα αναλύει το σειριακό πρωτόκολλο 3964R.

5.1. Εγχειρίδιο Χρήσης

Στο πρώτο μέρος της ενότητας αυτής γίνεται λεπτομερής περιγραφή του ελεγκτή του ρομποτικού βραχίονα καθώς και του χειριστηρίου του, με σκοπό τον σαφή προσδιορισμό της λειτουργίας του. Αναλύεται το ειδικά διαμορφωμένο λογισμικό της KUKA και οι δυνατότητές του.

Στο δεύτερο μέρος γίνεται μια ανάλυση της δομής client – server που αναπτύχθηκε.

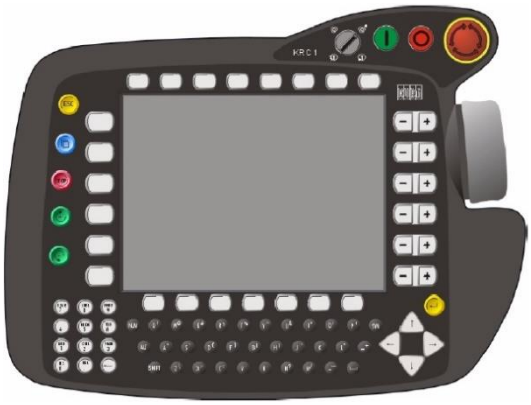
5.1.1. Kuka Robot Controller

Στην ενότητα αυτή καλύπτονται θέματα που αφορούν την ενεργοποίηση και την απενεργοποίηση του ελεγκτή, το ειδικά διαμορφωμένο χειριστήριο του, τα συστήματα συντεταγμένων του ρομποτικού βραχίονα και την κίνησή του από το χρήστη εκτός προγράμματος.

Στο τέλος γίνεται ανάλυση του γραφικού περιβάλλοντος του λογισμικού της Kuka και της λειτουργίας αυτού.

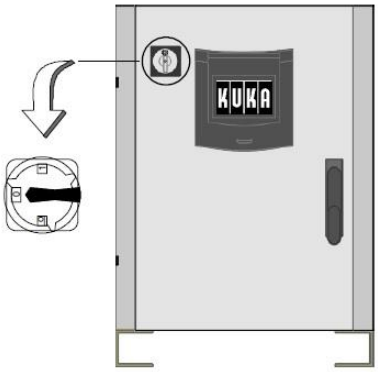
I. Ενεργοποίηση-Απενεργοποίηση του Συστήματος:

Γενικά:

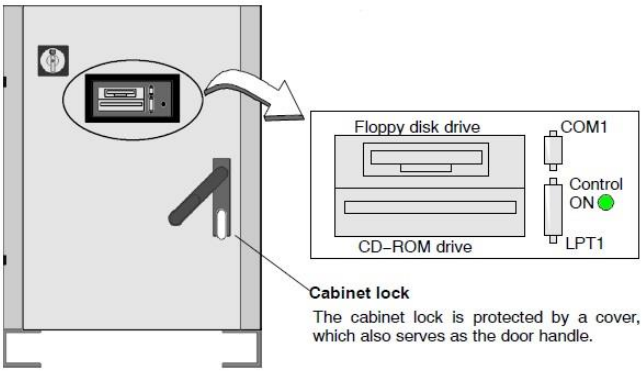


Ο ελεγκτής KRC1 περιέχει τα ηλεκτρονικά ελέγχου για τον ρομποτικό βραχίονα KR 15/1 καθώς και προσφέρει ισχύ σε αυτόν. Πέραν από το κεντρικό διακόπτη, όλα τα στοιχεία ελέγχου του χρήστη βρίσκονται στο χειριστήριο (Kuka Control Panel – KCP). Για διευκόλυνση στη χρήση του ρομπότ, έχουν συνδεθεί επιπλέον μια οθόνη, ένα πληκτρολόγιο και ένα ποντίκι.

Στοιχεία του ελεγκτή KRC1:



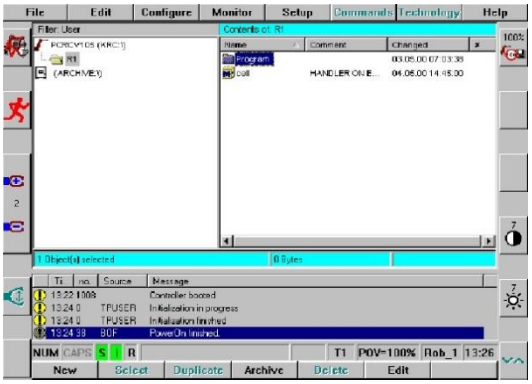
Το σύστημα ελεγκτή-ρομπότ ενεργοποιείται και απενεργοποιείται μέσω του κεντρικού διακόπτη, ο οποίος είναι σχεδιασμένος με τέτοιον τρόπο ώστε να μην είναι δυνατό να πατηθεί κατά λάθος.



Στην μπροστινή όψη του ελεγκτή, κάτω από το πλαστικό κάλυμμα βρίσκονται ο οδηγός για δισκέτες (Floppy Disk Drive), ο οδηγός για συμπαγείς δίσκους (CD-ROM Drive), η σειριακή θύρα COM1 και η παράλληλη θύρα LPT1 καθώς και η λυχνία κατάστασης του ελεγκτή (Control ON).

Ενεργοποίηση του ελεγκτή:

Αφού ο ελεγκτής έχει ενεργοποιηθεί από τον κεντρικό διακόπτη, ο υπολογιστής ξεκινά να φορτώνει το λειτουργικό σύστημα και το λογισμικό για τον έλεγχο του ρομπότ. Αυτή η διαδικασία διαρκεί αρκετά λεπτά. Η πρόοδος της απεικονίζεται στην οθόνη με την μπάρα προόδου.



Η οθόνη μετέπειτα εμφανίζει το κεντρικό μενού για δημιουργία, επεξεργασία ή εκτέλεση ενός προγράμματος.

Απενεργοποίηση του ελεγκτή:

Αφού ο χρήστης έχει απενεργοποιήσει το σύστημα μέσω του κεντρικού διακόπτη, ο ελεγκτής απενεργοποιεί μόνος του το λογισμικό για τον έλεγχο και το λειτουργικό σύστημα. Κατά τη διάρκεια αυτής της διαδικασίας αποθηκεύονται αυτόματα σημαντικά δεδομένα. Αυτό βέβαια συμβαίνει μόνο όταν ο ελεγκτής είχε ενεργοποιηθεί πριν ορθώς και πλήρως.

Συμπεριφορά του ελεγκτή κατά την επανεκκίνηση:

Υπάρχουν δύο τρόποι που το σύστημα εκκινεί:

- Κρύα εκκίνηση (Cold start)
- Ζεστή επανεκκίνηση (Warm restart)

Κρύα επανεκκίνηση:

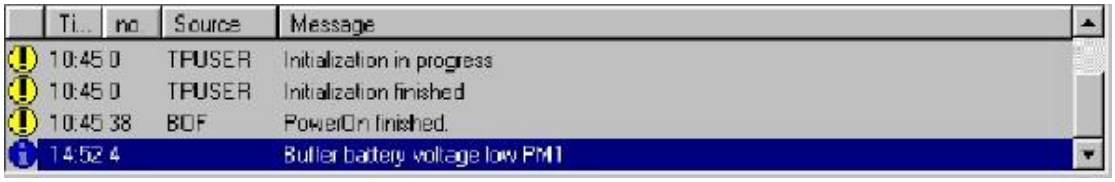
Κατά την κρύα επανεκκίνηση, κανένα πρόγραμμα δεν είναι επιλεγμένο όταν το σύστημα ενεργοποιείται και όλες οι έξοδοι χρήστη (User Outputs) είναι ψευδείς (FALSE). Αλλαγές σε αρχεία συστήματος απαιτούν κρύα επανεκκίνηση (βλέπε “Configure Menu”).

Ζεστή επανεκκίνηση:

Η ζεστή επανεκκίνηση έχει ως στόχο την μείωση του χαμένου χρόνου της παραγωγής σε περίπτωση που χαθεί η ηλεκτρική ισχύς του δικτύου ρεύματος. Όταν το σύστημα ενεργοποιηθεί, επαναφέρεται η θέση στην οποία είχε φτάσει το πρόγραμμα που έτρεχε και οι δίαυλοι (Field Bus) επαναφέρονται (reset). Οι έξοδοι (Outputs) επανέρχονται στις τιμές που είχαν πριν και η εκτέλεση του προγράμματος μπορεί να συνεχιστεί από το σημείο στο οποίο είχε διακοπεί.

Επίβλεψη ισχύος μπαταριών:

Ο ελεγκτής για να μπορέσει να εκτελέσει τις διαδικασίες που περιεγράφηκαν, περιέχει μπαταρίες οι οποίες αποδίδουν συνολική διαφορά δυναμικού 24V. Εάν η τάση αυτή πέσει κάτω από την τιμή των 22V, εμφανίζεται ένα μήνυμα σφάλματος στο παράθυρο μηνυμάτων.



Το μήνυμα σφάλματος εμφανίζεται σε περίπτωση που:

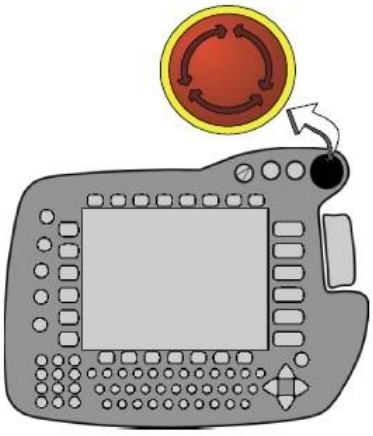
- Οι μπαταρίες δεν είναι πλήρως φορτισμένες, οπότε συνίσταται να μείνει ενεργοποιημένος ο ελεγκτής για 10 ώρες, ή
- Μία ή και οι δύο μπαταρίες είναι ελαττωματικές, οπότε απαιτείται αντικατάσταση.

Προστασία από ιούς:

Κατά την εκκίνηση του ελεγκτή, το λογισμικό κατά ιών Ikarus εκτελείται αυτόματα και ένα παράθυρο με πληροφορίες εμφανίζεται για λίγα δευτερόλεπτα.

II. Χειριστήριο KCP

i. Στοιχεία Ελέγχου του KCP



Κουμπί διακοπής έκτακτης ανάγκης (EMERGENCY STOP):

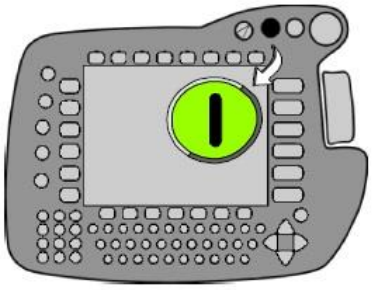
Το κουμπί διακοπής έκτακτης ανάγκης είναι το πιο σημαντικό στοιχείο ασφαλείας. Το κόκκινο κουμπί είναι σχεδιασμένο να πιέζεται σε επικίνδυνες περιπτώσεις και απενεργοποιεί πλήρως τους οδηγούς (drives) του ρομποτικού βραχίονα.

Προκειμένου οι οδηγοί να ενεργοποιηθούν ξανά, το κουμπί έκτακτης ανάγκης πρέπει να απελευθερωθεί. Για να γίνει αυτό, πρέπει να περιστραφεί ωρολογιακά μέχρι να ακουστεί πως απελευθερώθηκε. Το σύστημα παράγει ένα μήνυμα στο παράθυρο μηνυμάτων, το οποίο πρέπει να αναγνωριστεί από τον χρήστη πιέζοντας το πλήκτρο αναγνώρισης (Ack).

Η χρήση του κουμπιού διακοπής έκτακτης ανάγκης ενεργοποιεί πέδηση που διατηρεί την τροχιά της κίνησης.

Πριν την απελευθέρωση του κουμπιού, ο χρήστης θα πρέπει να ελέγξει τους λόγους για τους οποίους χρειάστηκε η χρήση του καθώς και τις συνέπειες της πράξης αυτής.

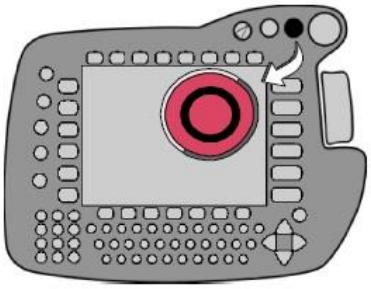
Οδηγοί ενεργοποιημένοι (DRIVES ON):



Η πίεση αυτού του κουμπιού ενεργοποιεί τους οδηγούς του ρομποτικού βραχίονα. Αυτοί μπορούν μόνο να τεθούν σε λειτουργία υπό κανονικές συνθήκες (π.χ. δεν έχει πατηθεί το κουμπί έκτακτης ανάγκης, κ.λπ.)

Εάν έχει τεθεί η χειροκίνητη κατάσταση (manual mode), το κουμπί δεν έχει λειτουργία (βλέπε κεφ. Κίνηση του Βραχίονα από το Χρήστη).

Οδηγοί απενεργοποιημένοι (DRIVES OFF):

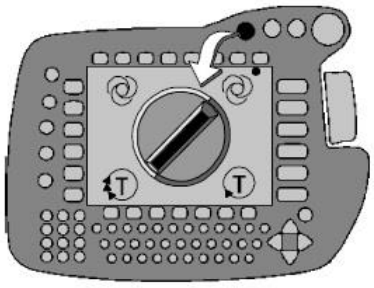


Η πίεση αυτού του κουμπιού απενεργοποιεί τους οδηγούς του ρομποτικού βραχίονα. Η πέδηση των κινητήρων εμπλέκεται με μία μικρή καθυστέρηση για να διατηρήσει τους άξονες στις θέσεις τους.

Εάν έχει τεθεί η χειροκίνητη κατάσταση (manual mode), το κουμπί δεν έχει λειτουργία.

Το κουμπί ενεργοποιεί δυναμική πέδηση (Dynamic Braking).

Επιλογή Κατάστασης (Mode Selection):



Κάνοντας χρήση του διακόπτη με το κλειδί μπορεί ο χρήστης να επιλέξει κατάσταση.

Το ρομπότ κινείται μόνο κατά την διάρκεια που ένα πλήκτρο ενεργοποίησης (enabling switch) διατηρείται πατημένο. Οι κινήσεις πραγματοποιούνται με μειωμένη ταχύτητα (Teaching Mode 1). Οι γραμμές του προγράμματος εκτελούνται ανά μία.

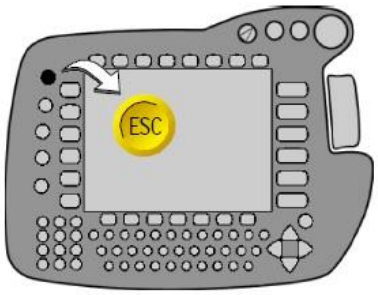
Το ρομπότ κινείται μόνο κατά την διάρκεια που ένα πλήκτρο ενεργοποίησης (enabling switch) διατηρείται πατημένο. Οι κινήσεις πραγματοποιούνται με την προγραμματισμένη ταχύτητα (Teaching Mode 2).

Το ρομπότ εκτελεί το επιλεγμένο πρόγραμμα αυτόματα και επιβλέπεται από τον χρήστη μέσω του χειριστηρίου KCP. Οι κινήσεις πραγματοποιούνται με την προγραμματισμένη ταχύτητα.

Το ρομπότ εκτελεί το επιλεγμένο πρόγραμμα αυτόματα και ελέγχεται μέσω ενός εξωτερικού υπολογιστή ή ενός PLC. Οι κινήσεις πραγματοποιούνται με την προγραμματισμένη ταχύτητα.

Σε περίπτωση που η κατάσταση αλλάξει κατά την διάρκεια που το πρόγραμμα δουλεύει, ενεργοποιείται η δυναμική πέδηση.

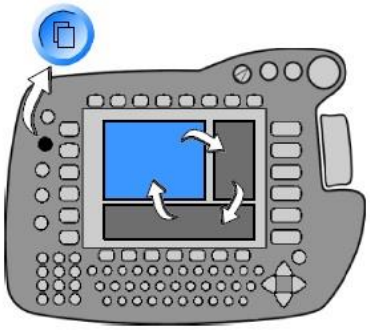
Πλήκτρο Διαφυγής/Ακύρωσης (Escape):



Μία δράση που έχει ξεκινήσει μπορεί να ακυρωθεί σε οποιαδήποτε στιγμή χρησιμοποιώντας το πλήκτρο διαφυγής/ακύρωσης. Αυτό περιλαμβάνει για παράδειγμα, ανοιχτές φόρμες σε γραμμή και παράθυρα κατάστασης.

Τα μενού που έχουν ανοίξει κατά λάθος μπορούν να κλείσουν ξανά, με την σειρά, πατώντας αυτό το πλήκτρο.

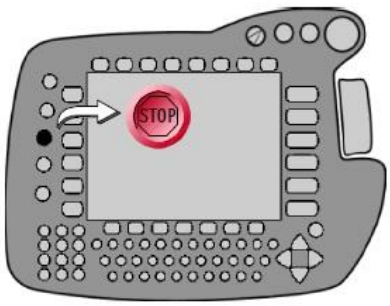
Πλήκτρο Εναλλαγής Παραθύρων (Window Selection Key):



Με αυτό το πλήκτρο, μπορεί να γίνει η εναλλαγή ανάμεσα στο παράθυρο του προγράμματος, στο παράθυρο μηνυμάτων και στο παράθυρο κατάστασης εάν αυτά είναι διαθέσιμα.

Το φόντο (background) του επιλεγμένου (ενεργοποιημένου) παραθύρου τονίζεται (highlight) με χρώμα.

Πλήκτρο Παύσης (Stop):

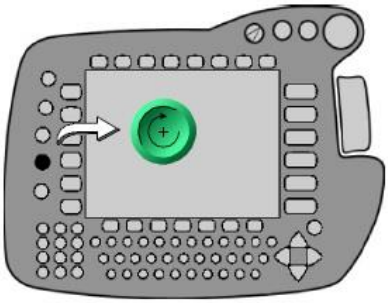


Πιέζοντας αυτό το πλήκτρο σταματάει το πρόγραμμα που τρέχει.

Πραγματοποιείται πέδηση που διατηρεί την τροχιά (path maintaining braking). Είναι δυνατό σε αυτόματη κατάσταση να αναγνωριστεί το γεγονός αυτό από το πρόγραμμα.

Για την συνέχιση ενός προγράμματος που σταμάτησε, πρέπει να πιεστεί το πλήκτρο πρόσω εκκίνησης.

Πλήκτρο Πρόσω Εκκίνησης (Program Start Forwards):

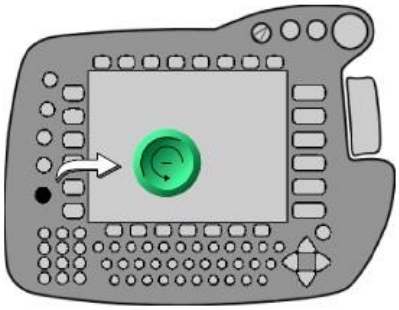


Αυτό το πλήκτρο χρησιμοποιείται για την εκκίνηση ενός επιλεγμένου προγράμματος.

Ένα πρόγραμμα μπορεί μόνο να ξεκινήσει όταν οι οδηγοί είναι ενεργοποιημένοι και δεν υπάρχει κατάσταση έκτακτης ανάγκης (Emergency Stop).

Κατά τις καταστάσεις T1 και T2, το πλήκτρο αυτό πρέπει να μένει πατημένο, αλλιώς ενεργοποιείται πέδηση με διατήρηση της τροχιάς.

Πλήκτρο Οπίσω Εκκίνησης (Program Start Backwards):



Πιέζοντας αυτό το πλήκτρο, οι κινήσεις του επιλεγμένου προγράμματος πραγματοποιούνται βήμα προς βήμα με κατεύθυνση την αρχή του προγράμματος.

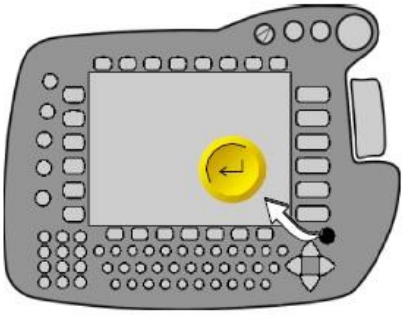
Επομένως ο ρομποτικός βραχίονας κινείται με αντίθετη φορά επάνω στην τροχιά που είχε αρχικώς προγραμματιστεί.

Η κίνηση χρησιμοποιείται, για παράδειγμα, για την εκμάθηση ενδιάμεσων σημείων σε κυκλικές κινήσεις.

Η απελευθέρωση του πλήκτρου ενεργοποιεί πέδηση με διατήρηση της τροχιάς.

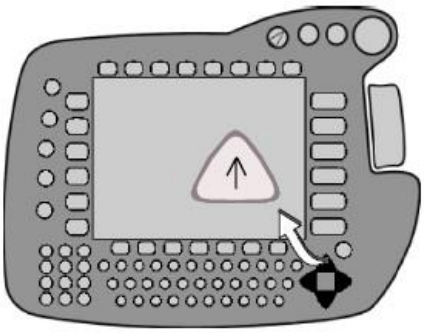
Η λειτουργιά αυτή είναι διαθέσιμη μόνο στις καταστάσεις T1 και T2.

Πλήκτρο Εισαγωγής (Enter Key):



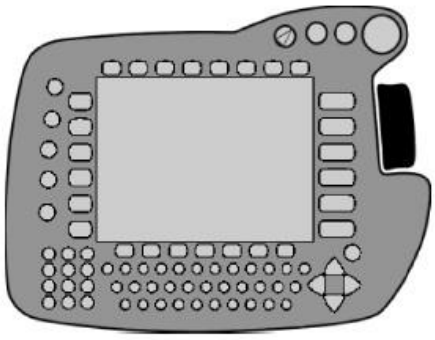
Αυτό το στοιχείο ελέγχου αντιστοιχεί στο πλήκτρο “Enter” ή “Return” από το κανονικό πληκτρολόγιο υπολογιστή.

Πλήκτρα Κατευθύνσεων (Arrow Keys):



Η λειτουργία αυτών των πλήκτρων είναι όμοια με αυτή σε ένα κανονικό πληκτρολόγιο.

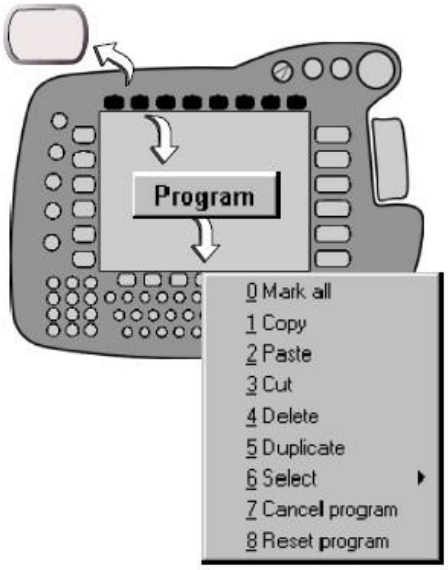
Ποντίκι Χώρου (Space Mouse):



Με το ποντίκι χώρου δίνεται η δυνατότητα ελέγχου και των έξι αξόνων (βαθμών ελευθερίας) του ρομποτικού βραχίονα. Το μέγεθος της μετατόπισης από την αρχική θέση επηρεάζει την ταχύτητα του ρομπότ.

Εναλλακτικά, τα πλήκτρα + και – στην δεξιά πλευρά της οθόνης μπορούν επίσης να χρησιμοποιηθούν.

Πλήκτρα Μενού (Menu Keys):

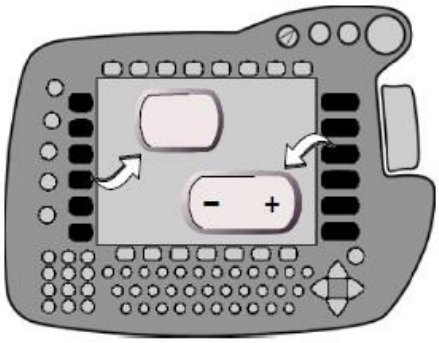


Αυτά τα πλήκτρα χρησιμοποιούνται για να ανοίγουν τα εκάστοτε μενού στην μπάρα των μενού (menu bar), που βρίσκεται στο πάνω μέρος της οθόνης.

Η περιήγηση στα μενού μπορεί να γίνει είτε με τα πλήκτρα κατευθύνσεων είτε με το αριθμητικό πληκτρολόγιο.

Ένα μενού μπορεί να κλείσει ένα βήμα την φορά χρησιμοποιώντας το πλήκτρο διαφυγής.

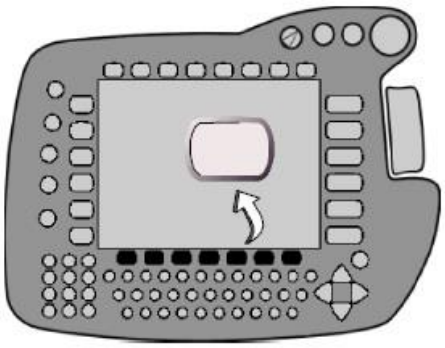
Πλήκτρα Κατάστασης (Status Keys):



Τα πλήκτρα κατάστασης (που βρίσκονται στην αριστερή και στην δεξιά πλευρά της οθόνης) χρησιμοποιούνται για την επιλογή ρυθμίσεων λειτουργίας, την εναλλαγή μεμονωμένων λειτουργιών και την ρύθμιση τιμών.

Οι εκάστοτε λειτουργίες απεικονίζονται γραφικά κάθε φορά με εικονίδια δίπλα στα πλήκτρα.

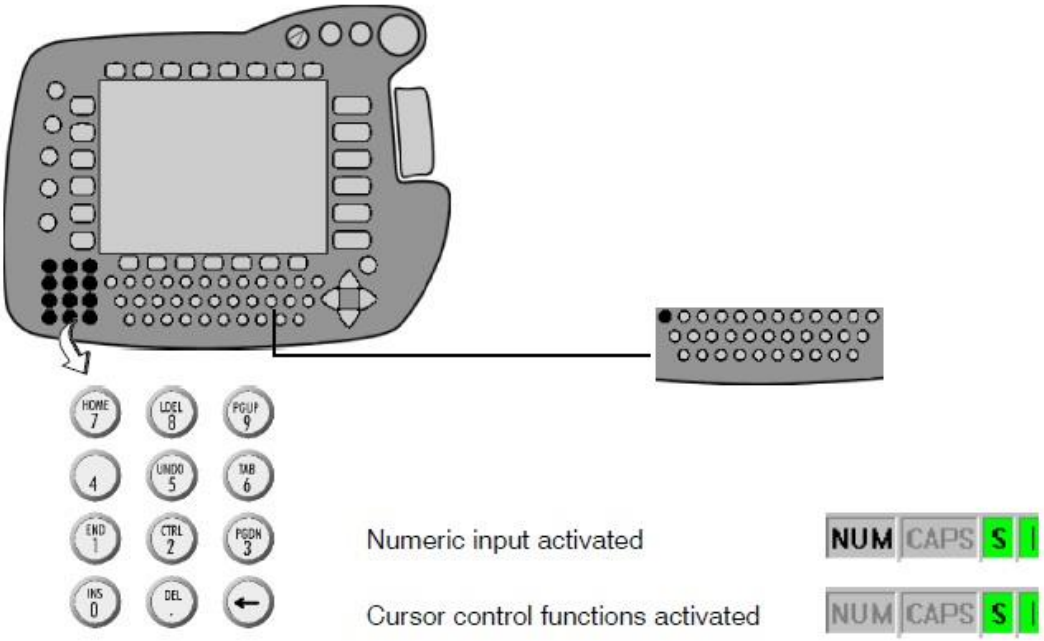
Εναλλασσόμενα Πλήκτρα (Softkeys):



Αυτά τα πλήκτρα χρησιμοποιούνται για την επιλογή των λειτουργιών που απεικονίζονται με εικονίδια πάνω από αυτά.

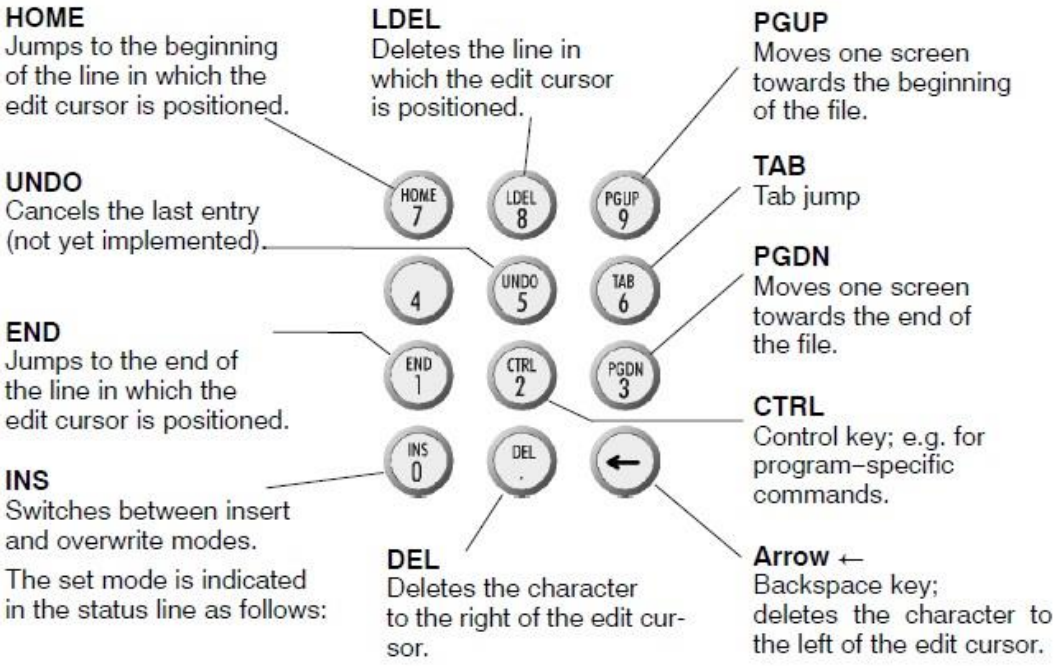
Οι λειτουργίες τους εναλλάσσονται δυναμικά όπως αλλάζουν και τα αντίστοιχα εικονίδια τους.

Αριθμητικό Πληκτρολόγιο (Numeric Keypad):

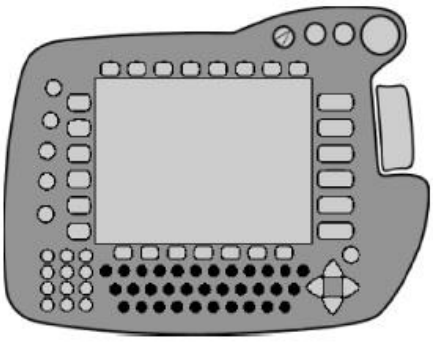


Σε πρώτο επίπεδο, το αριθμητικό πληκτρολόγιο χρησιμοποιείται για την εισαγωγή αριθμών.

Σε δεύτερο επίπεδο, οι λειτουργίες του αφορούν τον έλεγχο του κέρσορα.



Πληκτρολόγιο (Keyboard):



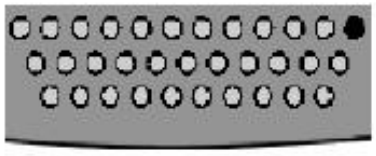
Η εναλλαγή μεταξύ κεφαλαίων και πεζών γραμμάτων γίνεται με την χρήση του πλήκτρου SHIFT.

Εάν το πλήκτρο έχει πατηθεί μία φορά, τότε μόνο ο επόμενος χαρακτήρας θα είναι κεφαλαίος. Για να τυπωθούν όλοι οι χαρακτήρες κεφαλαίοι, πρέπει να μείνει πατημένο.

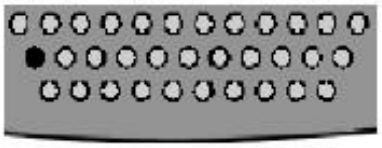
Είναι επίσης δυνατό να χρησιμοποιηθεί η λειτουργία κεφαλαίων (CAPS LOCK). Για να επιτευχθεί αυτό, πρέπει να ενεργοποιηθούν τα πλήκτρα SYM και SHIFT ταυτόχρονα.

Caps Lock inactive 

Caps Lock active 



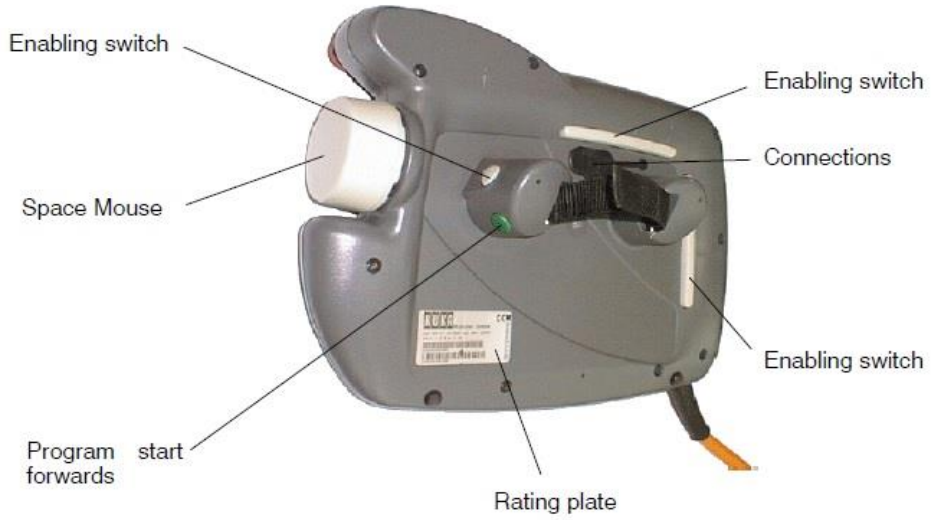
Οι χαρακτήρες στίξης καθώς και οι ειδικοί χαρακτήρες είναι διαθέσιμοι στο δεύτερο επίπεδο του πληκτρολογίου. Για την χρήση αυτών πρέπει να πατηθεί το πλήκτρο SYM. Εάν πατηθεί μία φορά, τότε μόνο ο επόμενος χαρακτήρας θα είναι στίξη ή ειδικός.



Το πλήκτρο ALT βρίσκεται στα αριστερά του πληκτρολογίου. Χρησιμοποιείται για δευτερεύουσες λειτουργίες, όπως πχ ALT + TAB.

Το πλήκτρο ελέγχου CTRL βρίσκεται στο αριθμητικό πληκτρολόγιο. Για να χρησιμοποιηθεί, πρέπει το αριθμητικό πληκτρολόγιο να ελέγχει τις λειτουργίες κέρσορα όπως προαναφέρθηκε.

ii. Πίσω Όψη του KCP



iii. Γραφικό Περιβάλλον Χρήστη (GUI)

Η οθόνη του χειριστηρίου του ρομπότ είναι χωρισμένη σε περιοχές με διάφορες δυνατότητες. Αυτές αλλάζουν δυναμικά ανάλογα με τις ανάγκες κατά την λειτουργία του ελεγκτή.

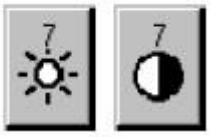
Τα στοιχεία του γραφικού περιβάλλοντος περιλαμβάνουν την μπάρα μενού (menu bar), τα πλήκτρα κατάστασης (status keys), τα εναλλασσόμενα πλήκτρα (softkeys), το παράθυρο του προγράμματος (program window), τις φόρμες εισαγωγής (inline forms), το παράθυρο μηνυμάτων (message window) και την γραμμή κατάστασης (status line).

a. Ρύθμιση Φωτεινότητας και Αντίθεσης

Για μεγαλύτερη ευκρίνεια, τόσο η φωτεινότητα όσο και η αντίθεση της οθόνης LCD μπορούν να τροποποιηθούν.

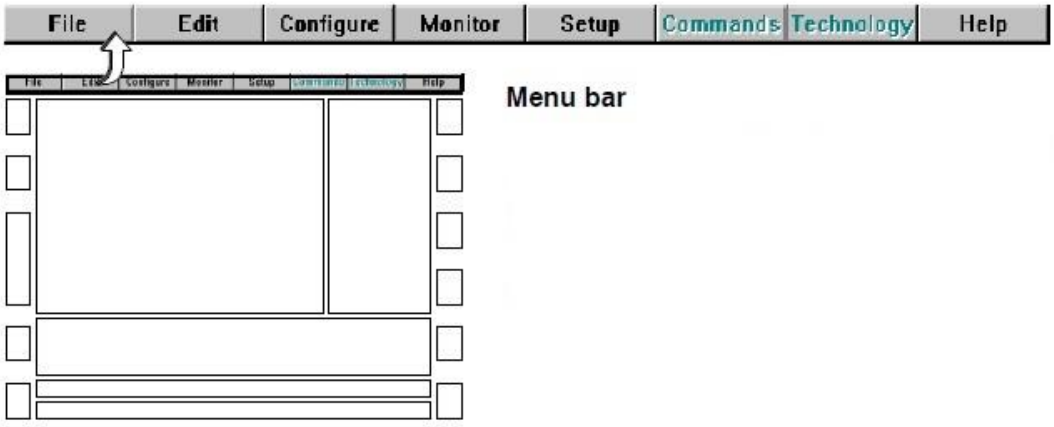
Η χειροκίνητη λειτουργία του ρομπότ πρέπει πρώτα να απενεργοποιηθεί για να γίνει η τροποποίηση.

Το πλήκτρο της χειροκίνητης λειτουργίας βρίσκεται αριστερά προς τα πάνω στην οθόνη.

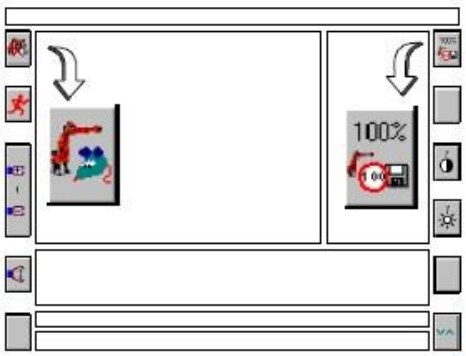


Τα δύο πλήκτρα στα δεξιά αφορούν την φωτεινότητα και την αντίθεση, που μπορούν να αλλάξουν χρησιμοποιώντας τα πλήκτρα + και -.

b. Πλήκτρα ελέγχου

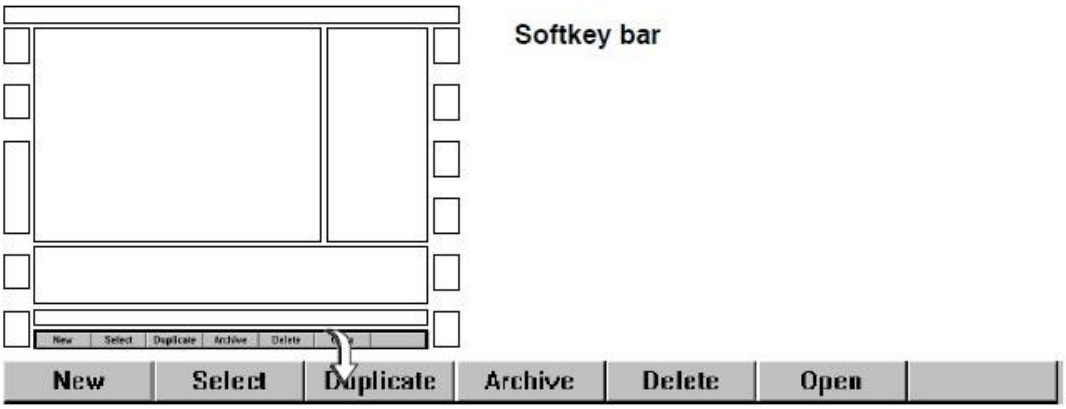


Οι λειτουργίες του ελεγκτή είναι ομαδοποιημένες στην μπάρα μενού. Η κάθε ομάδα μπορεί να επεκταθεί με την χρήση των πλήκτρων μενού.



Status key bars

Οι μπάρες των πλήκτρων κατάστασης εμφανίζουν τις εναλλασσόμενες λειτουργίες στα αριστερά και δεξιά της οθόνης.

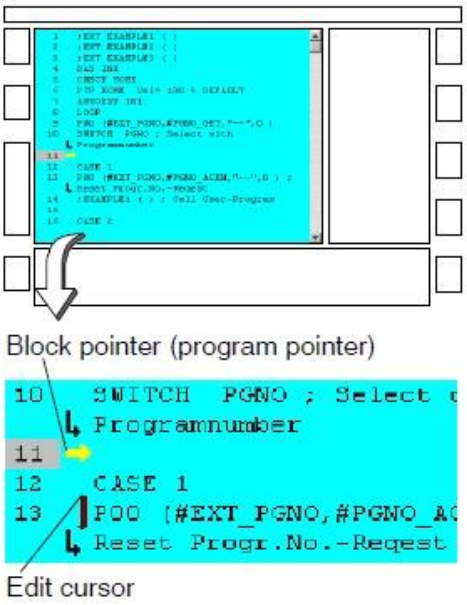


Softkey bar

Η μπάρα των εναλλασσόμενων πλήκτρων βρίσκεται στο κάτω μέρος της οθόνης και αλλάζει ανάλογα με την περίσταση.

c. Παράθυρα Εισόδου/Εξόδου

Παράθυρο προγράμματος (Program Window):

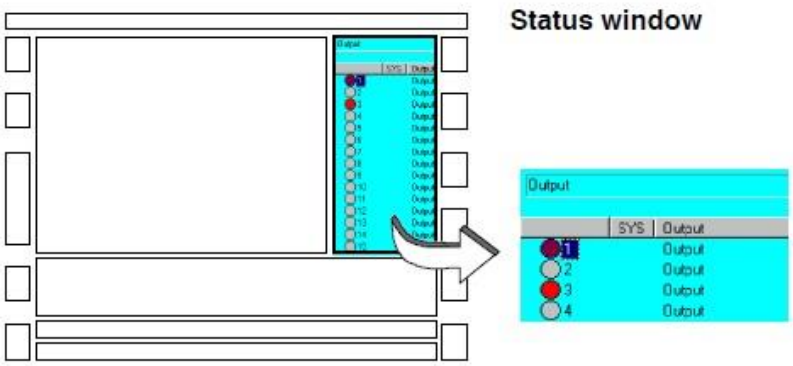


Το παράθυρο προγράμματος εμφανίζει τα περιεχόμενα του επιλεγμένου προγράμματος. Εάν δεν έχει επιλεγεί κάποιο πρόγραμμα, εμφανίζει μία λίστα με όλα τα διαθέσιμα προγράμματα.

Μεταξύ του αριθμού της σειράς και του κώδικα, ένα κίτρινο βέλος που στοχεύει δεξιά υποδεικνύει την σειρά η οποία εκείνη την στιγμή εκτελείται. Το βέλος αυτό ονομάζεται δείκτης εντολής (block pointer).

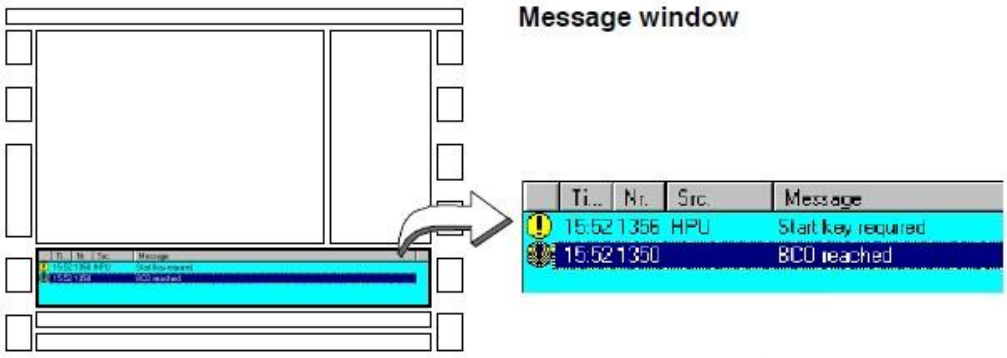
Ένας άλλος δείκτης είναι ο δείκτης επεξεργασίας (edit cursor), μία κατακόρυφη κόκκινη γραμμή που βρίσκεται στην αρχή της σειράς και υποδεικνύει ποια γραμμή είναι υπό επεξεργασία.

Παράθυρο Κατάστασης (Status Window):



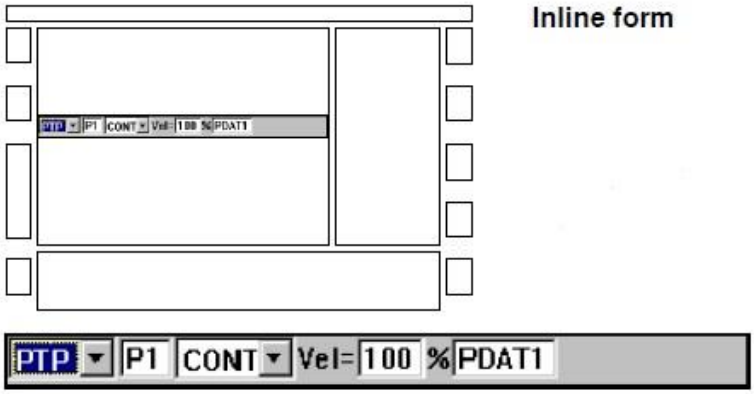
Το παράθυρο κατάστασης εμφανίζεται εάν είναι αναγκαίο για την εμφάνιση πληροφοριών, όπως για παράδειγμα την εκχώρηση των εξόδων (outputs) ή την εισαγωγή δεδομένων, λόγου χάρη την βαθμονόμηση του εργαλείου (tool calibration).

Παράθυρο μηνυμάτων (Message Window):



Ενημερωτικά μηνύματα, μηνύματα κατάστασης, διαλόγου και επιβεβαίωσης εμφανίζονται σε αυτό το παράθυρο.

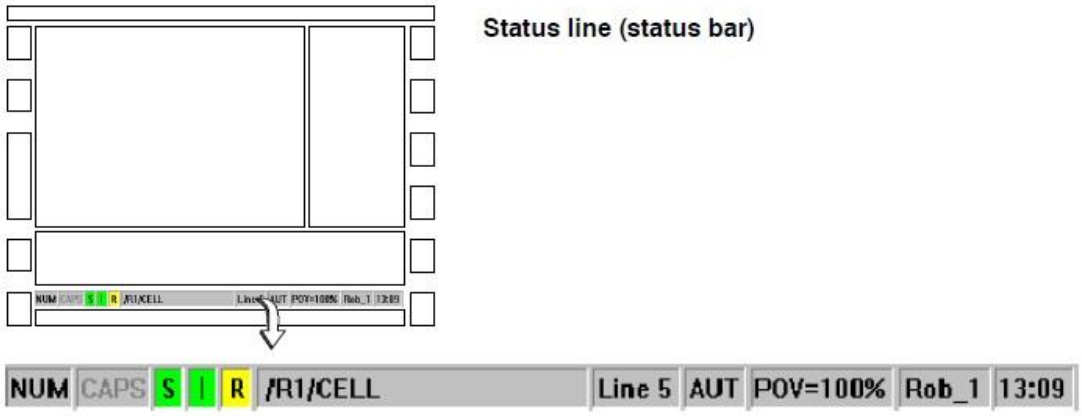
Φόρμα Εισαγωγής (Inline Form):



Μερικές λειτουργίες του προγράμματος απαιτούν να γίνει εισαγωγή τιμών. Οι τιμές αυτές εισάγονται στην φόρμα εισαγωγής (inline form) ή επιλέγονται από τα υπο-μενού της φόρμας.

Με αυτό τον τρόπο, είναι σίγουρο ότι οι εντολές έχουν πάντα την κατάλληλη δομή.

Κατάσταση Συστήματος (Status Bar):



Η γραμμή κατάστασης (Status Line) περιέχει πληροφορίες για σημαντικές καταστάσεις λειτουργίας.

Αυτό περιλαμβάνει πληροφορίες για την κατάσταση του προγράμματος ή του PLC.

d. Μηνύματα

Τα σύμβολα που εμφανίζονται στο παράθυρο μηνυμάτων έχουν την εξής σημασία:

Τα ενημερωτικά μηνύματα (Notification Messages) περιέχουν πληροφορίες ή υποδεικνύουν επιλογές του χειριστή, προγραμματιστικά λάθη και λάθη κατά την λειτουργία. Έχουν καθαρά πληροφοριακό χαρακτήρα και δεν διακόπτουν την εκτέλεση του προγράμματος.

Start key required

This message appears after a program has been selected.

Τα μηνύματα κατάστασης (Status Messages) υποδεικνύουν την κατάσταση του συστήματος. Είναι επίσης πληροφοριακού χαρακτήρα και μπορούν να διακόψουν την εκτέλεση ενός προγράμματος μέχρι ένα βαθμό. Τα μηνύματα αυτά διαγράφονται άμεσα μόλις η κατάσταση που τα προκάλεσε σταματά να ισχύει.

EMERGENCY STOP

This message is generated if, for example, the EMERGENCY STOP button has been pressed or a safety gate opened.

Τα μηνύματα επιβεβαίωσης (Acknowledgement Messages) συνήθως εμφανίζονται μετά από ένα μήνυμα κατάστασης και πρέπει να επιβεβαιωθούν ρητά. Υποδεικνύουν διαταραχή στην εκτέλεση του προγράμματος.

Confirm EMERGENCY STOP

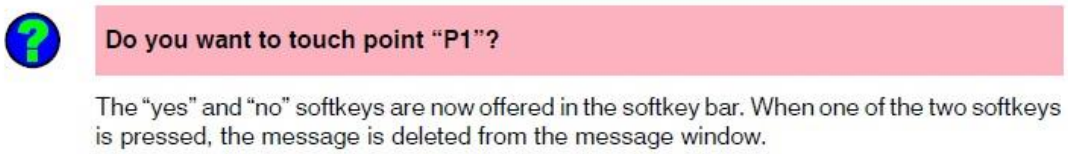
Acknowledgement messages stop robot operation until the cause of the error has been eliminated and the message confirmed.

Μηνύματα αναμονής (Wait Messages) δημιουργούνται εάν κατά την εκτέλεση ενός προγράμματος εκτελείται μια εντολή αναμονής (wait condition).

WAIT FOR $IN[1]==TRUE

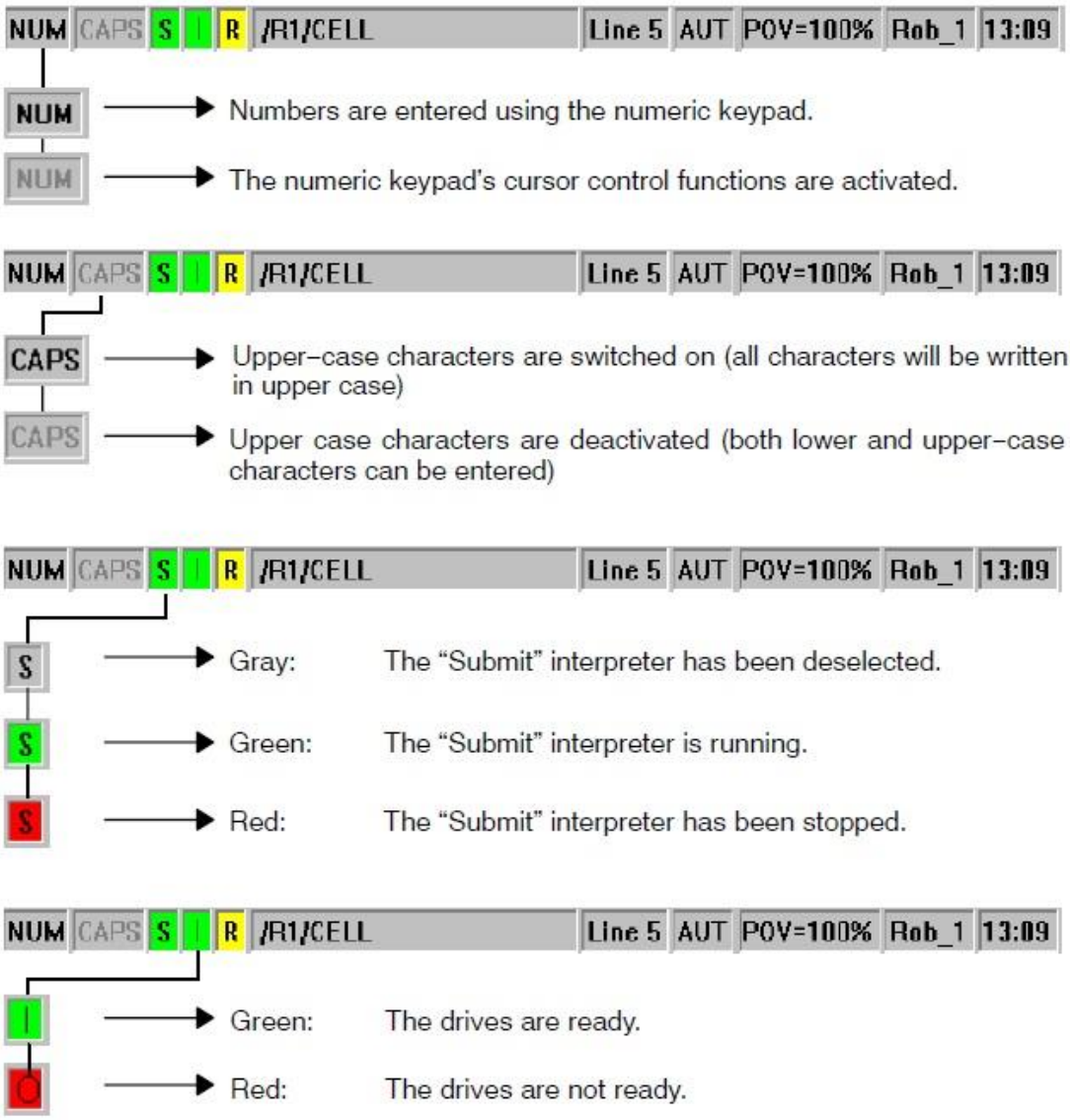
The robot controller is stopped until the condition is fulfilled or the program reset. In this example the system is waiting for a signal at input 1.

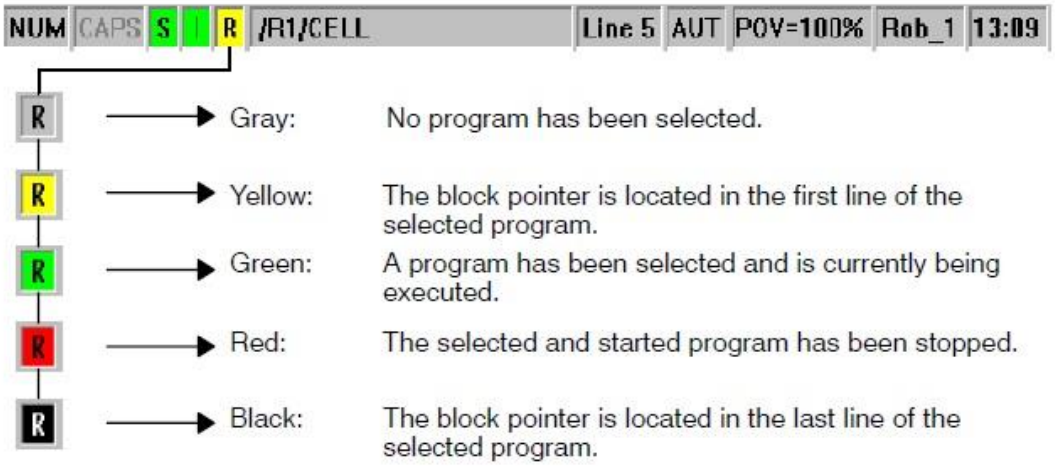
Στα μηνύματα διαλόγου ο χειριστής πρέπει να ανταποκριθεί. Η επιλογή αποθηκεύεται στην εκάστοτε μεταβλητή. Η εκτέλεση του προγράμματος σταματά μέχρι να γίνει η επιλογή και στην συνέχεια συνεχίζει κανονικά.

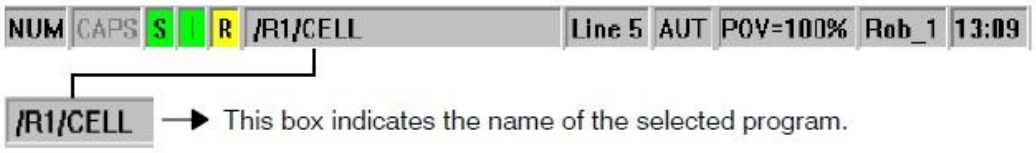


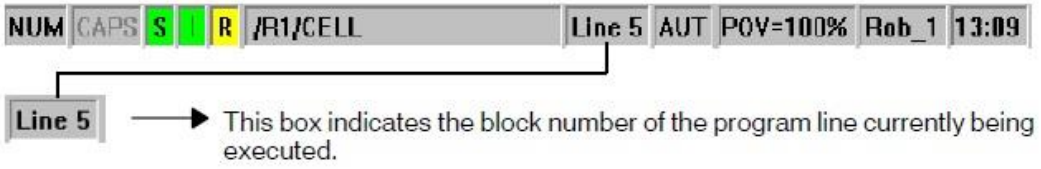
e. Γραμμή Κατάστασης

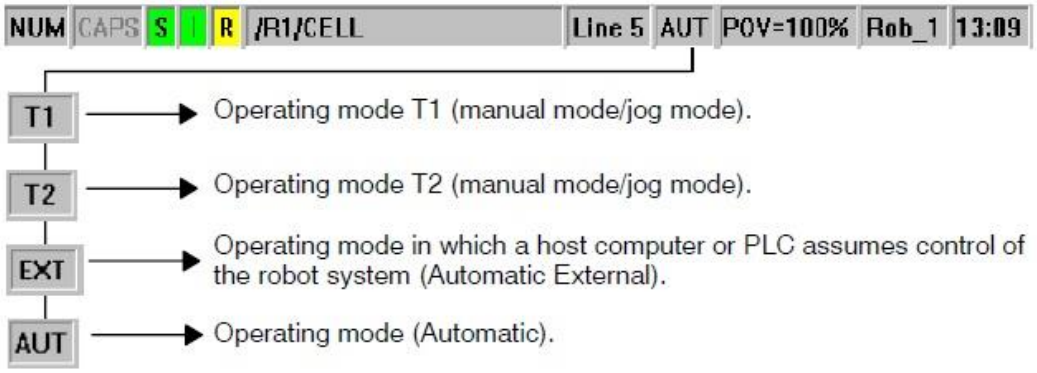
Η γραμμή κατάστασης περιέχει πληροφορίες για σημαντικές λειτουργίες.

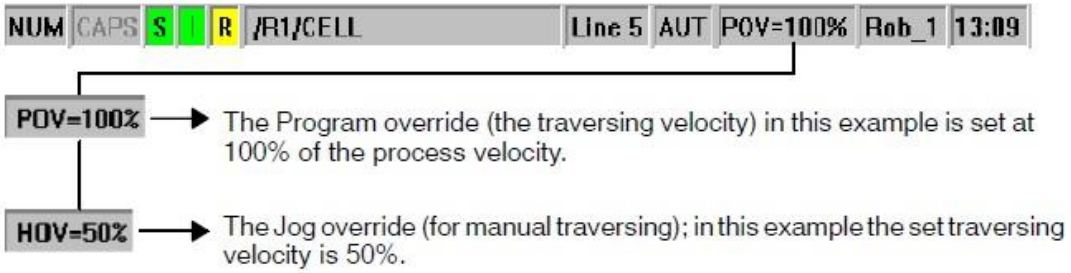


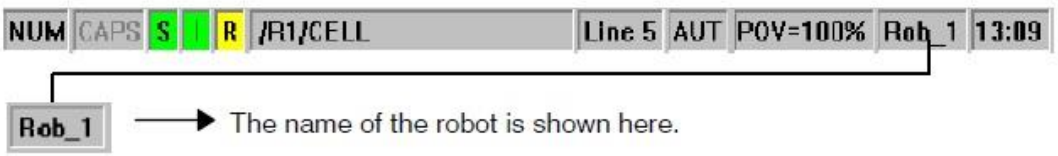

NUM CAPS S R /R1/CELL Line 5 AUT POV=100% Rob_1 13:09
R
R
R
R
R
Gray:
No program has been selected.
Yellow:
The block pointer is located in the first line of the selected program.
Green:
A program has been selected and is currently being executed.
Red:
The selected and started program has been stopped.
Black:
The block pointer is located in the last line of the selected program.

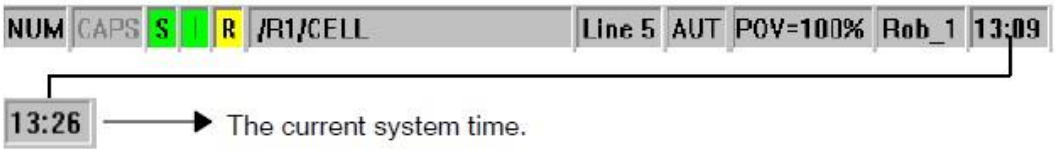

NUM CAPS S R /R1/CELL Line 5 AUT POV=100% Rob_1 13:09
/R1/CELL
This box indicates the name of the selected program.


NUM CAPS S R /R1/CELL Line 5 AUT POV=100% Rob_1 13:09
Line 5
This box indicates the block number of the program line currently being executed.


NUM CAPS S R /R1/CELL Line 5 AUT POV=100% Rob_1 13:09
T1
Operating mode T1 (manual mode/jog mode).
T2
Operating mode T2 (manual mode/jog mode).
EXT
Operating mode in which a host computer or PLC assumes control of the robot system (Automatic External).
AUT
Operating mode (Automatic).

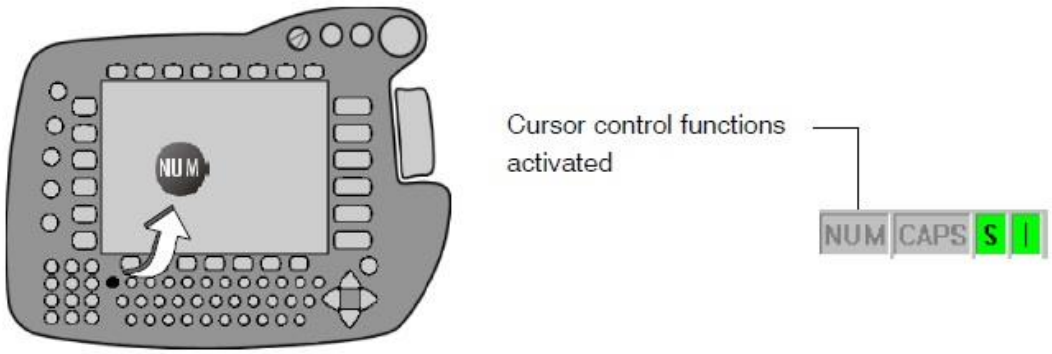

NUM CAPS S R /R1/CELL Line 5 AUT POV=100% Rob_1 13:09
POV=100%
The Program override (the traversing velocity) in this example is set at 100% of the process velocity.
HOV=50%
The Jog override (for manual traversing); in this example the set traversing velocity is 50%.



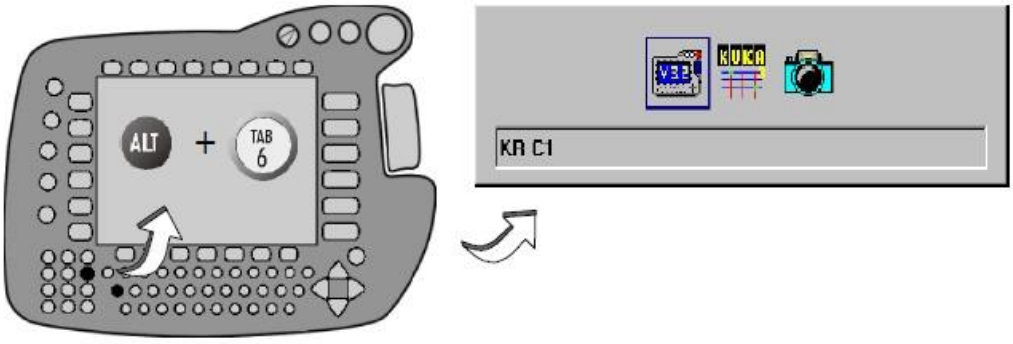


f. Εναλλαγή στην Επιφάνεια Εργασίας των Windows

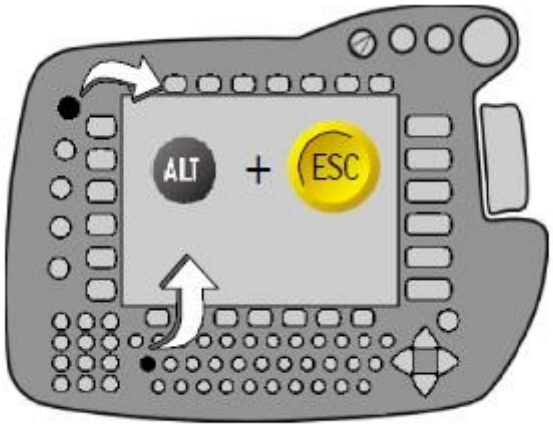
Είναι δυνατή η εναλλαγή στην επιφάνεια εργασίας των Windows χρησιμοποιώντας διάφορες συντομεύσεις (μόνο για τον χρήστη επιπέδου EXPERT). Για να γίνει αυτό, πρέπει η ένδειξη NUM στην γραμμή κατάστασης να είναι απενεργοποιημένη, ώστε να χρησιμοποιούνται οι λειτουργίες του αριθμητικού πληκτρολογίου.



<u>Alt – Tab:</u>

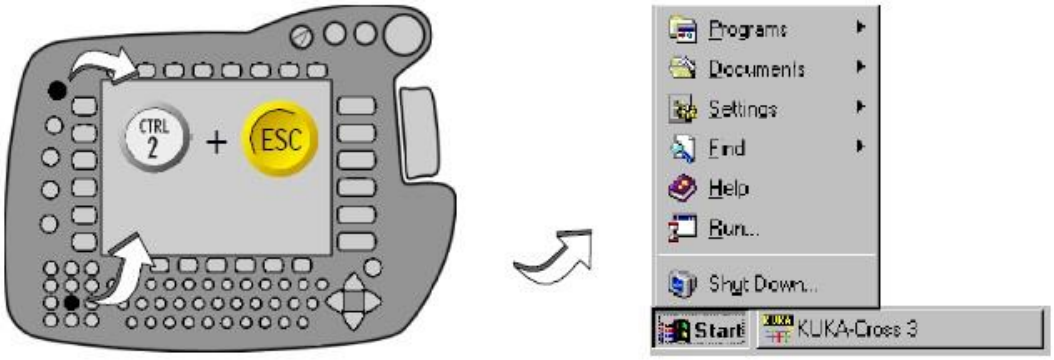


Alt – Escape:



Γίνεται επιστροφή στην προηγούμενη ενεργή εφαρμογή.

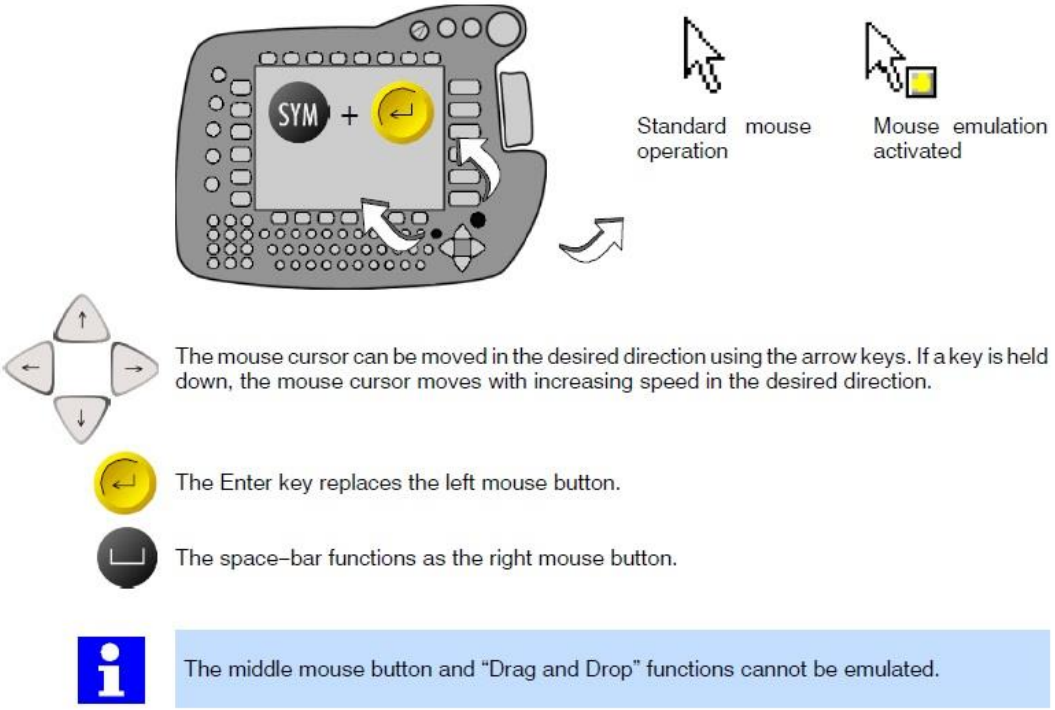
CTRL – Escape:



Ο συνδυασμός αυτός ενεργοποιεί την έναρξη των Windows (Start). Είναι επίσης δυνατή αυτή η λειτουργία απευθείας από το επιπλέον συνδεδεμένο πληκτρολόγιο, πατώντας το πλήκτρο των Windows.

g. Εξομοίωση των Πλήκτρων Ποντικιού

Η λειτουργία αυτή επιτρέπει την εξομοίωση της χρήσης ενός ποντικιού. Μπορεί να ενεργοποιηθεί και να απενεργοποιηθεί χρησιμοποιώντας τον συνδυασμό πλήκτρων SYM και Enter.



Συνίσταται ο χρήστης να χρησιμοποιεί το συνδεδεμένο ποντίκι για περισσότερη ευκολία. Η εξομοίωση του ποντικιού παρατίθεται για πληρότητα.

III. Συστήματα Συντεταγμένων

i. Γενικά

Για να κινηθεί ο βραχίονας από τον χρήστη και όχι από κάποιο πρόγραμμα, πρέπει να επιλεγεί ένα σύστημα συντεταγμένων. Ο χρήστης μπορεί να το κάνει αυτό είτε με το ποντίκι χώρου είτε με τα πλήκτρα + και -.

Joint coordinate system
Each robot axis can be individually moved in positive or negative direction;

WORLD coordinate system
Fixed, rectangular coordinate system whose origin is located at the base of the robot;

BASE coordinate system
Rectangular coordinate system which has its origin on the workpiece that is to be processed;

TOOL coordinate system
Rectangular coordinate system, whose origin is located in the tool.

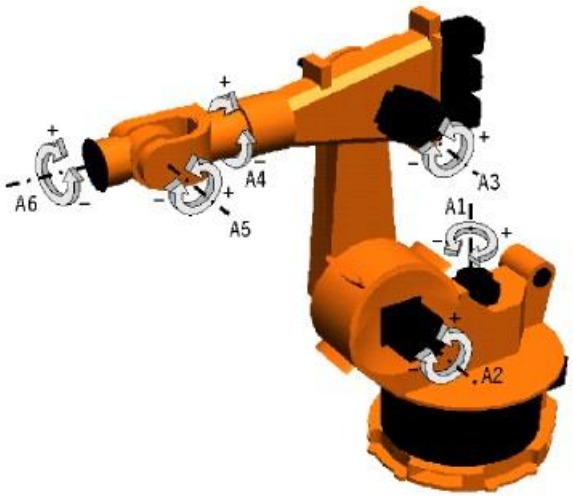
Το σύστημα συντεταγμένων που αναφέρεται ο βραχίονας μια δεδομένη στιγμή, μπορεί να αλλάξει μόνο κατά την χειροκίνητη λειτουργία (T1 ή T2).

Στα αριστερά προς τα επάνω στην οθόνη εμφανίζεται είτε το σύμβολο για το ποντίκι χώρου, είτε για τα πλήκτρα.

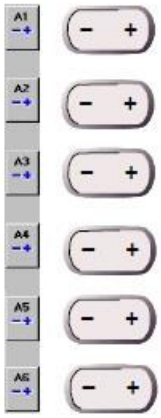
Για την επιλογή του συστήματος συντεταγμένων, πρέπει να πατηθεί επαναλαμβανόμενα το πλήκτρο, μέχρις ότου εμφανίζεται το εικονίδιο που επιθυμούμε δίπλα στο πλήκτρο κατάστασης.

Για την κίνηση του ρομπότ από το χειριστήριο, υπάρχουν περισσότερες πληροφορίες στο επόμενο κεφάλαιο.

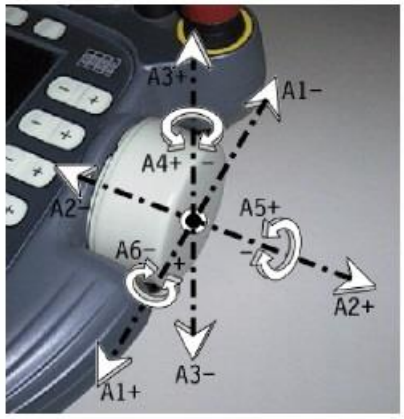
ii. Αξονικό Σύστημα Συντεταγμένων (Joint Coordinate System)



The following traversing keys/movements of the Space Mouse enable each axis to be moved individually:

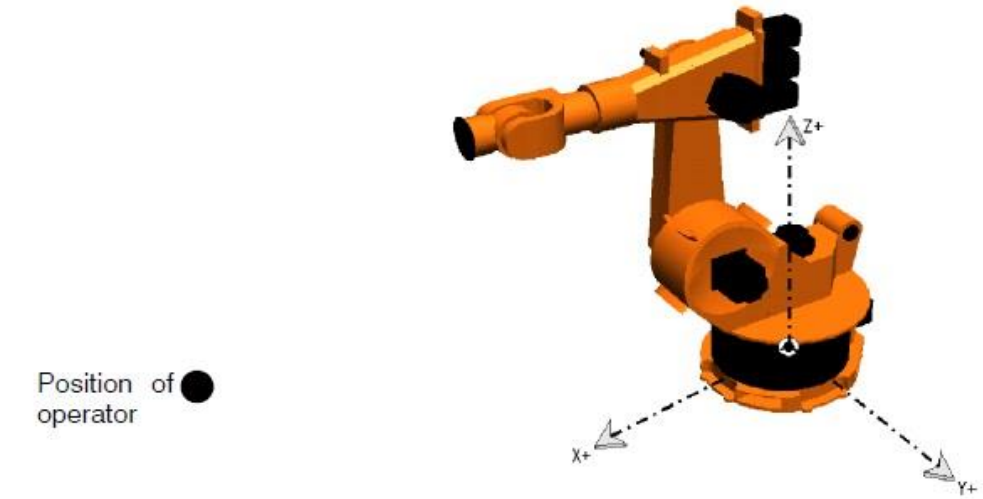


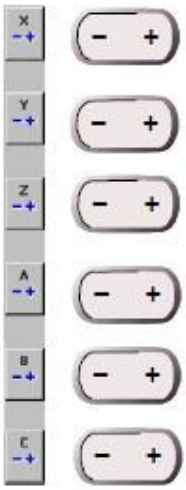
Traversing keys



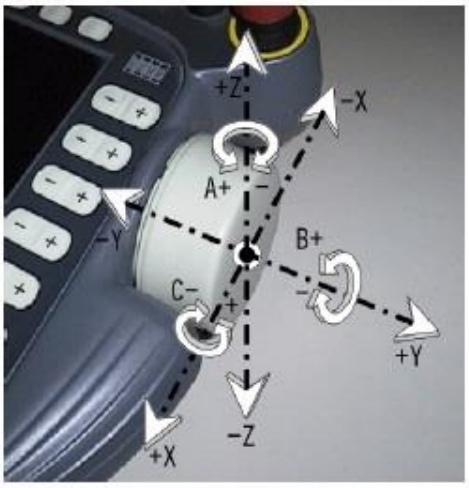
Space Mouse

iii. Σύστημα Συντεταγμένων Κόσμου (World Coordinate System)



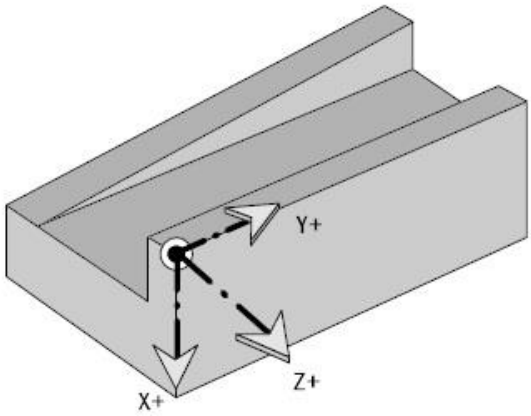


Manual traversing keys



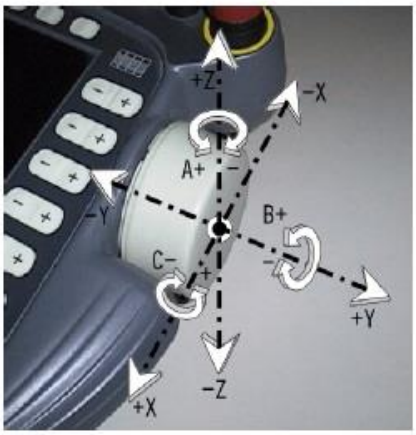
Space Mouse

iv. Σύστημα Συντεταγμένων Βάσης



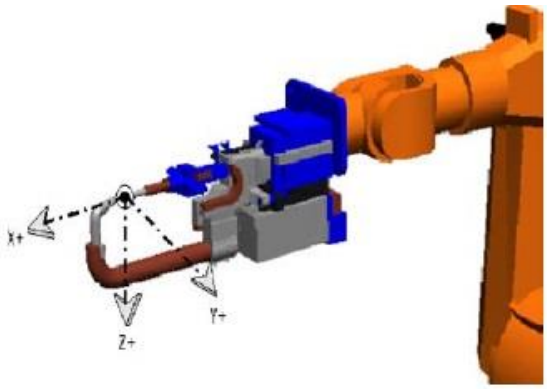
The following traversing keys/movements of the Space Mouse enable each axis to be moved individually:

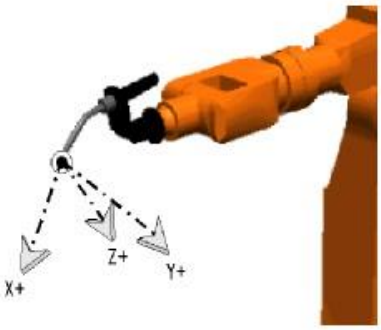
Manual traversing keys

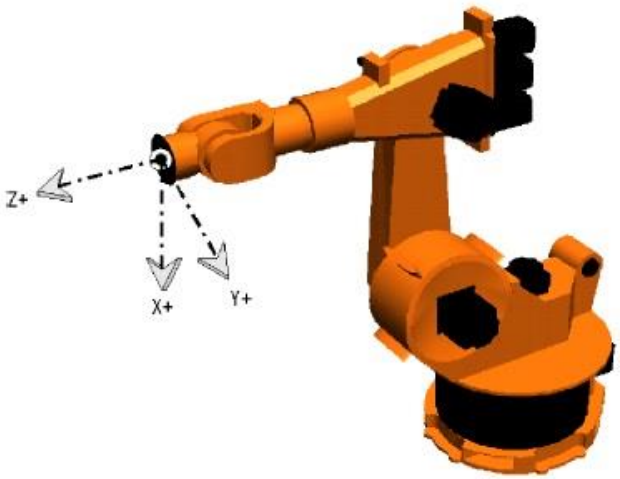


Space Mouse

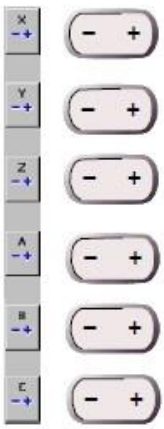
v. Σύστημα Συντεταγμένων Εργαλείου

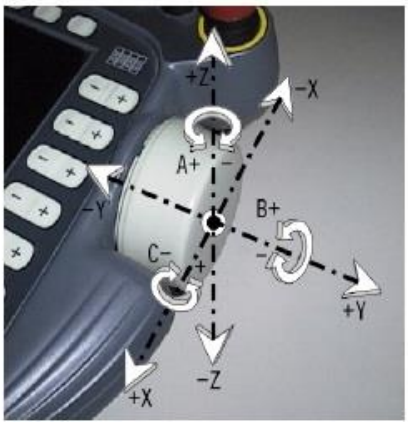






The following traversing keys/movements of the Space Mouse enable each axis to be moved individually:





IV. Κίνηση του Βραχίονα από το Χρήστη

i. Εισαγωγή

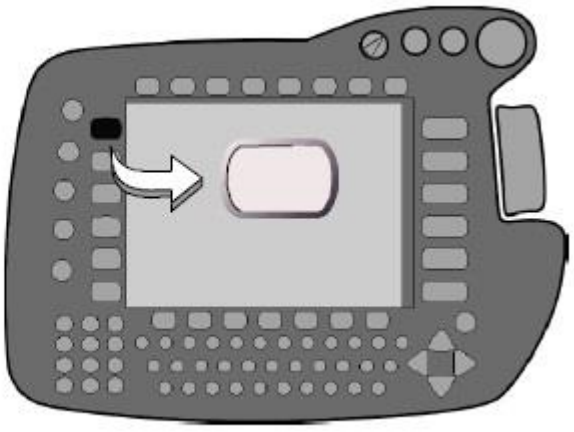
Για να κινηθεί ο βραχίονας από το χρήστη, πρέπει να είναι ενεργοποιημένη η χειροκίνητη λειτουργία T1 ή T2.

Στην συνέχεια πρέπει να επιλεγεί ο τρόπος ελέγχου της κίνησης:

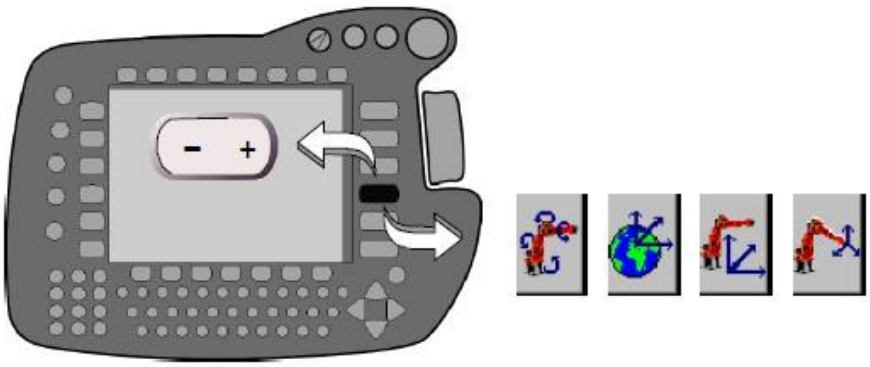
Η χειροκίνητη λειτουργία δεν είναι διαθέσιμη

Το ποντίκι χώρου είναι ενεργοποιημένο

Τα πλήκτρα είναι ενεργοποιημένα



Έπειτα, πρέπει να επιλεγεί το σύστημα συντεταγμένων:



Τέλος, πρέπει να επιλεγεί το κινηματικό σύστημα, καθότι ο ελεγκτής πέραν από τους έξι άξονες του βραχίονα, υποστηρίζει μέχρι άλλους έξι βαθμούς ελευθερίας – εξωτερικούς άξονες (External Axes).

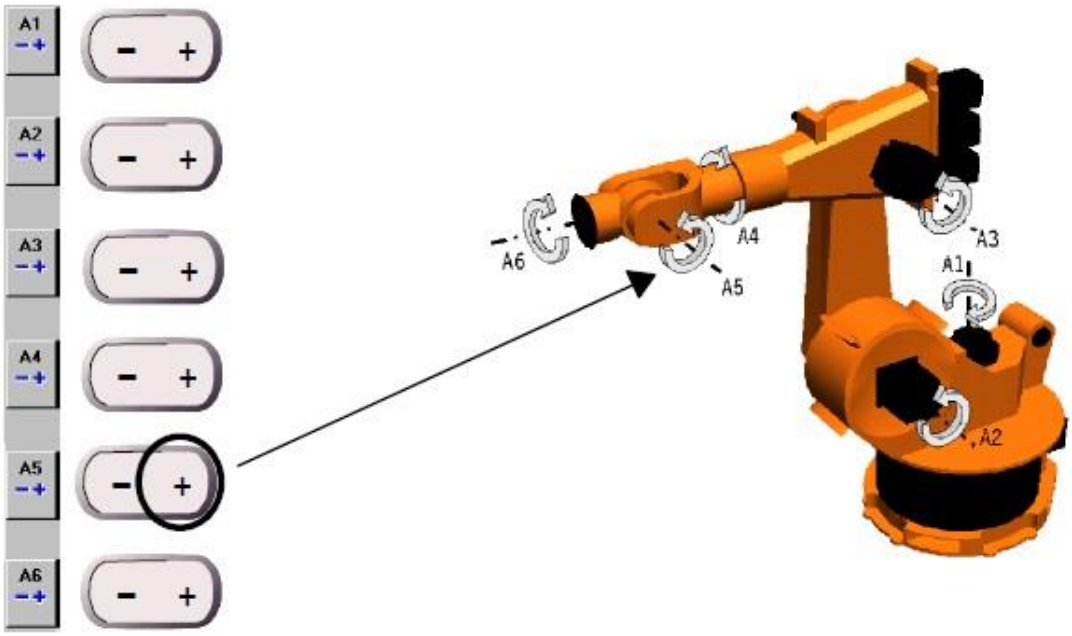
Στο παρόν έντυπο γίνεται επεξήγηση της κίνησης του ρομπότ από τον χρήστη μέσω των πλήκτρων και όχι του ποντικιού χώρου. Για περισσότερες πληροφορίες όσον αφορά το ποντίκι χώρου, ο αναγνώστης καλείται να ανατρέξει στο εγχειρίδιο οδηγιών:

- Operator Control

ii. Σύστημα Συντεταγμένων Αρθρώσεων

Στην περίπτωση που έχει επιλεγεί το σύστημα συντεταγμένων των αρθρώσεων (αξονικό), οι άξονες από A1 μέχρι A6 εμφανίζονται στην δεξιά πλευρά του χειριστηρίου, όταν πατηθεί το ενεργοποιητικό πλήκτρο (enabling switch) στην πίσω πλευρά αυτού.

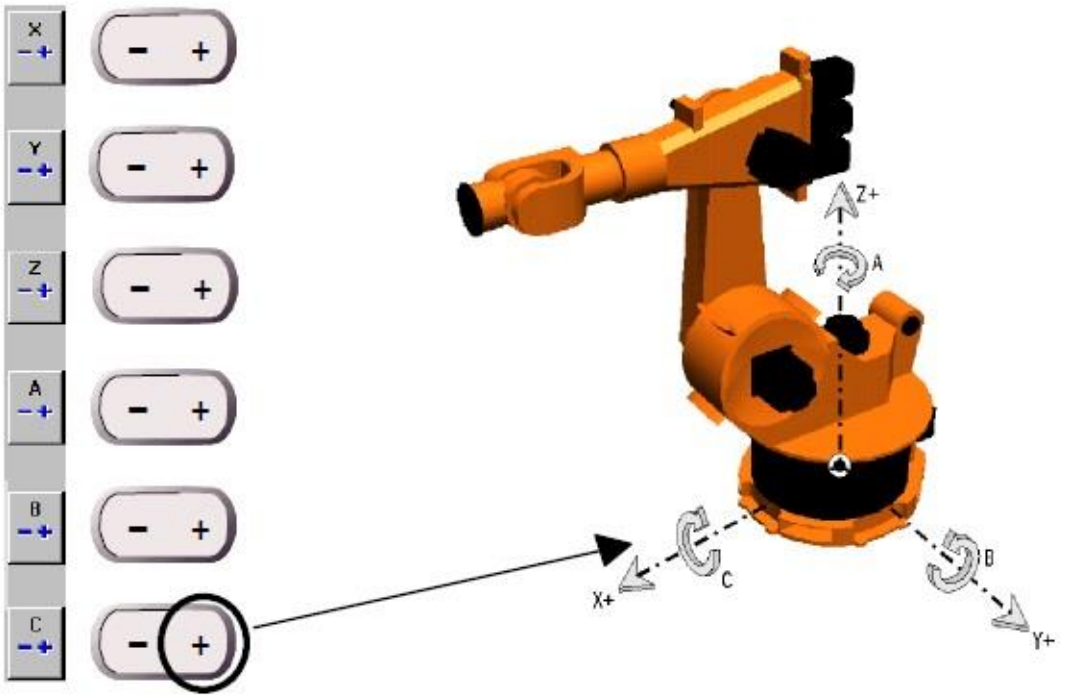
Στο σύστημα αυτό, όπως φαίνεται στο παρακάτω σχήμα, γίνεται η αντιστοίχιση των πλήκτρων με την θετική και την αρνητική φορά του κάθε άξονα.



iii. Συστήματα Συντεταγμένων Εργαλείου, Βάσης, Κόσμου

Εάν έχει επιλεγεί κάποιο από αυτά τα συστήματα, στα δεξιά του χειριστηρίου εμφανίζονται οι βαθμοί ελευθερίας X, Y, Z, A, B, C. Περισσότεροι άξονες του ενός ενεργοποιούνται κατά την κίνηση αυτή ταυτοχρόνως.

Για παράδειγμα, στο σύστημα συντεταγμένων Κόσμου, η αντιστοίχιση έχει ως εξής:



iv. Επιλογή Ταχύτητας

Είναι δυνατή η επιλογή της ταχύτητας με την οποία θα κινηθεί το ρομπότ από τον χρήστη. Το εικονίδιο αυτό υποδεικνύει το ποσοστό της μέγιστης επιτρεπόμενης ταχύτητας και στο αξονικό σύστημα αφορά τον κάθε άξονα ξεχωριστά. Συνίσταται η επιλογή 1% και 3% μέχρι ο χρήστης να εξοικειωθεί με τον βραχίονα.

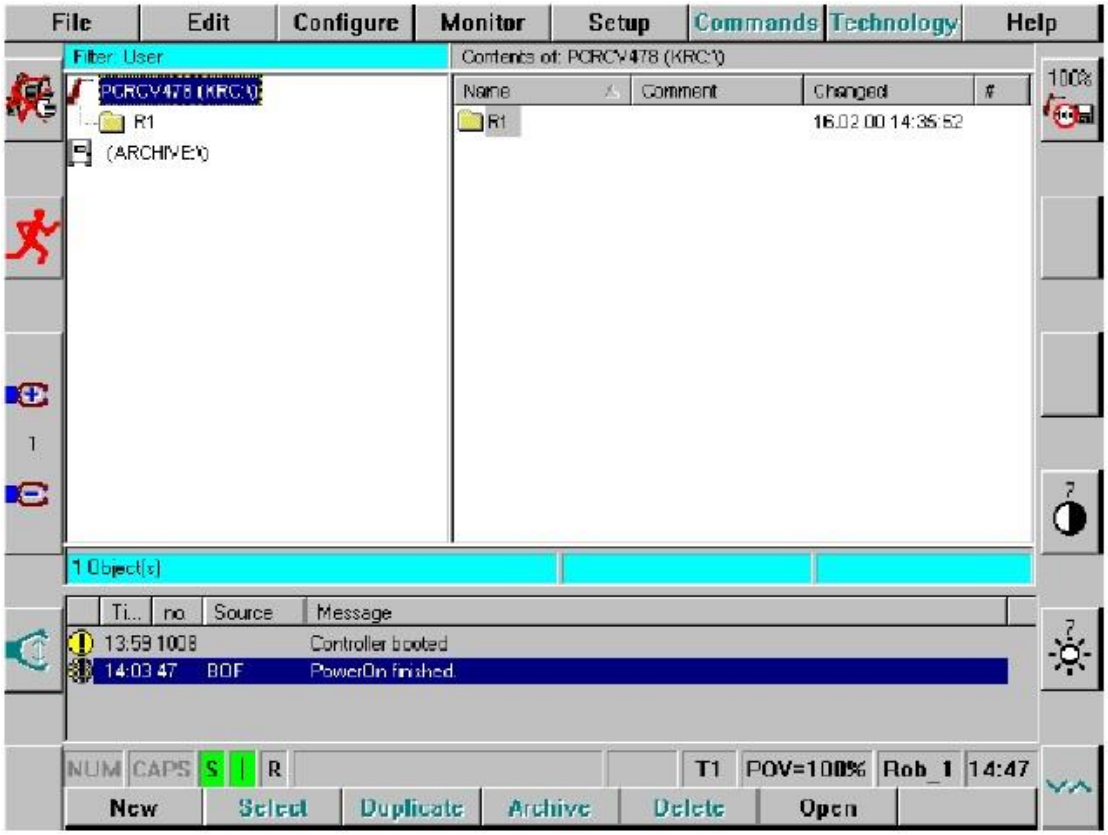
V. Πλοηγός (Navigator)

Στην ενότητα αυτή γίνεται μία ανάλυση της μορφής και της λειτουργίας του πλοηγού του λογισμικού της Kuka.

i. Γενικά

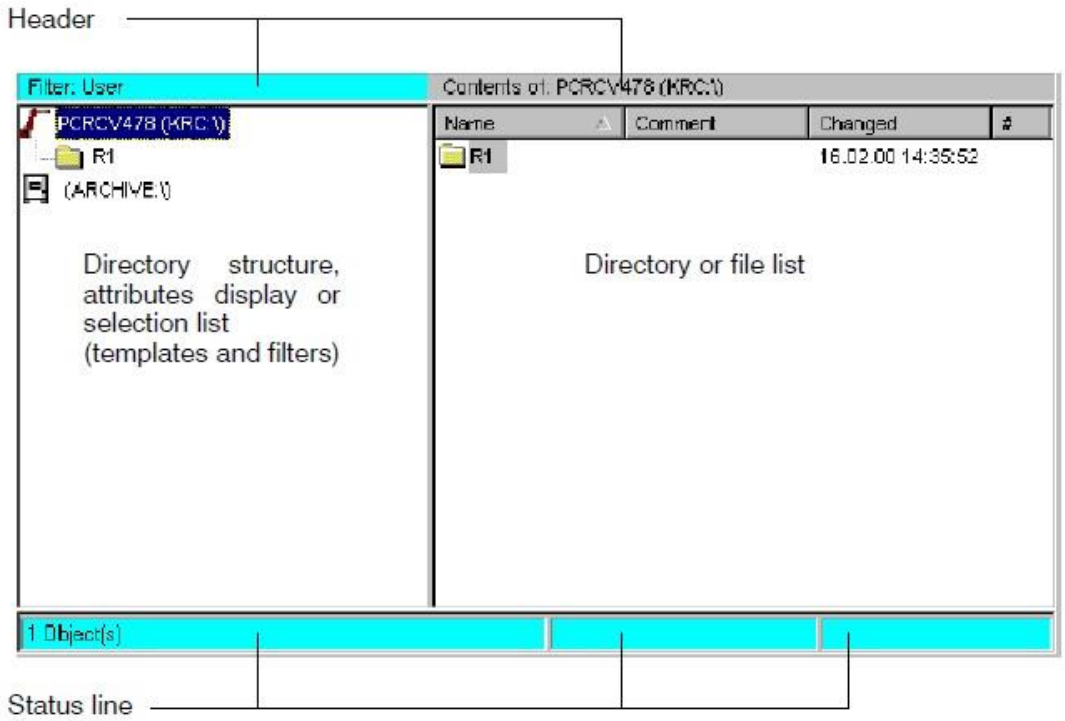
Ο πλοηγός είναι το πρόγραμμα διαχείρισης των αρχείων, τον οποίο ο χρήστης μπορεί να χρησιμοποιήσει για να πλοηγηθεί στους δίσκους και στην δομή των καταλόγων των αρχείων (directories). Μέσω αυτού μπορεί να δημιουργήσει, να επιλέξει, να αντιγράψει, να αποθηκεύσει, να διαγράψει και να «ανοίξει» αρχεία.

Αφού ο ελεγκτής έχει πλήρως ενεργοποιηθεί, εμφανίζεται η παρακάτω οθόνη:



ii. Γραφικό Περιβάλλον (GUI)

Το γραφικό περιβάλλον αποτελείται από 4 επιμέρους τμήματα, την επικεφαλίδα (header), τους δύο καταλόγους (directories) και την γραμμή κατάστασης (status line).



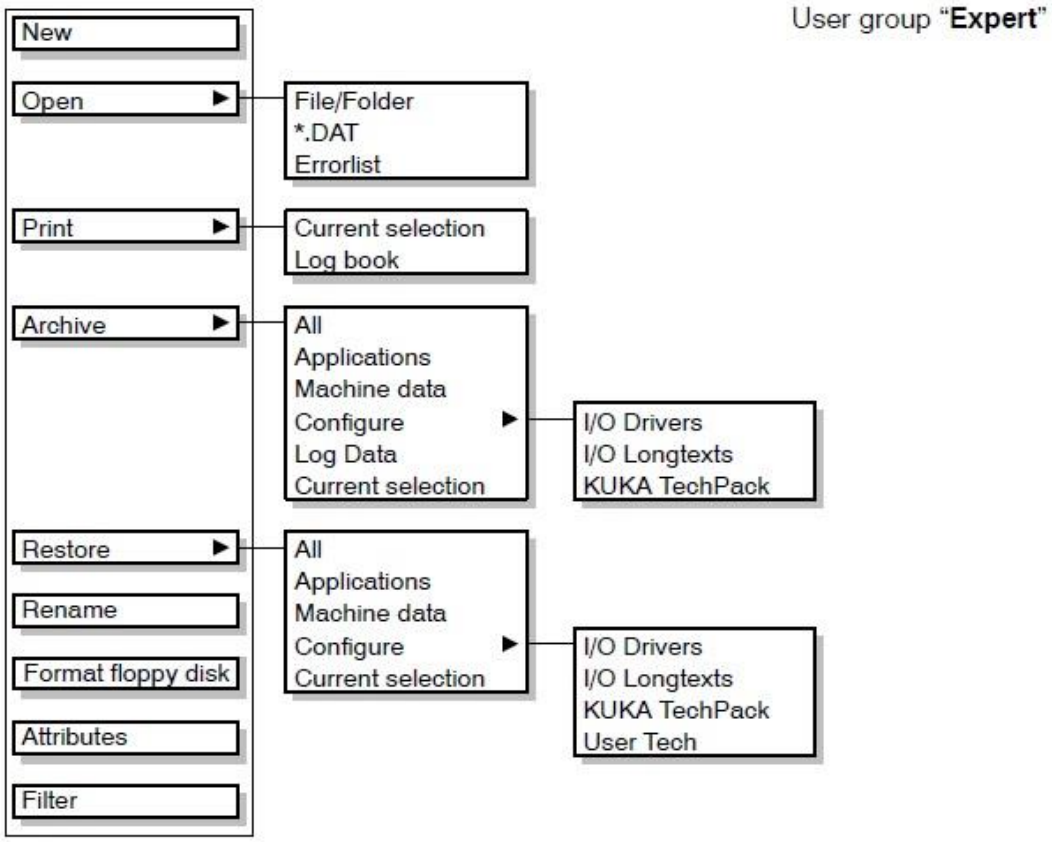
Στα αριστερά της επικεφαλίδας φαίνεται το φίλτρο που είναι ενεργοποιημένο για την αντίστοιχη απεικόνιση (αντίστοιχο του view στα Windows), ενώ στα δεξιά φαίνεται ο κατάλογος που χρησιμοποιείται την προκείμενη στιγμή. Το φίλτρο αλλάζει μόνο σε επίπεδο χρήστη EXPERT.

Ο χρήστης μπορεί να πλοηγηθεί μέσα στους καταλόγους με την χρήση των βελών κατεύθυνσης ή με το ποντίκι.

Στην γραμμή κατάστασης υπάρχουν πληροφορίες σχετικά με τον αριθμό των αρχείων, το μέγεθός τους, την θέση τους, ενώ η γραμμή αυτή αλλάζει δυναμικά ανάλογα με την εκάστοτε χρήση.

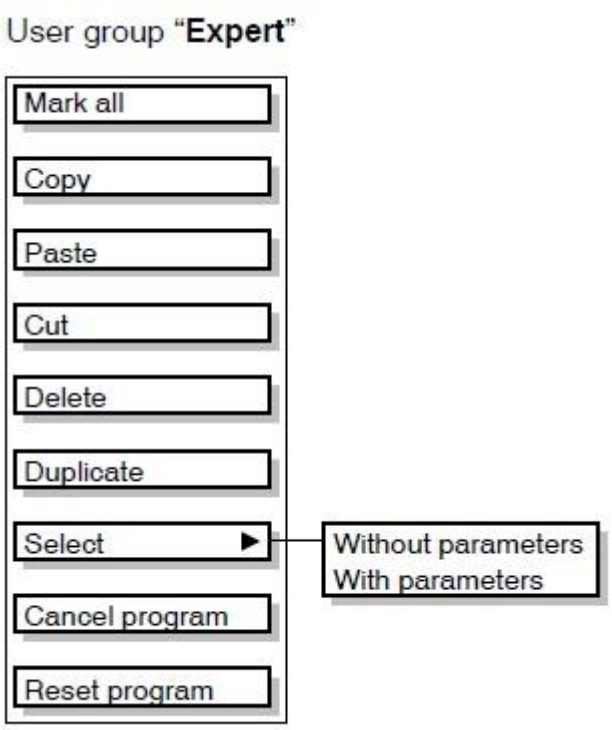
iii. Μενού Αρχείου (File Menu)

Πιέζοντας το πλήκτρο “File” εμφανίζεται το μενού που έχει την παρακάτω δομή:



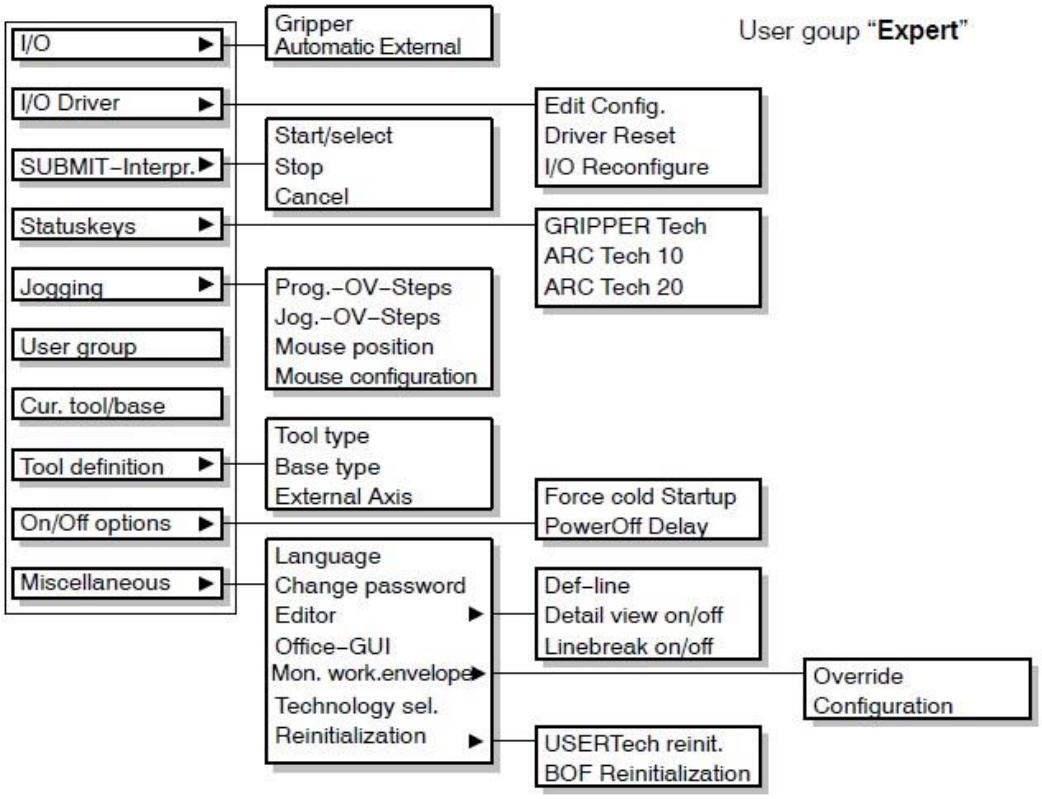
iv. Μενού Επεξεργασίας (Edit Menu)

Πιέζοντας το πλήκτρο “Edit” εμφανίζεται το μενού που έχει την παρακάτω δομή:



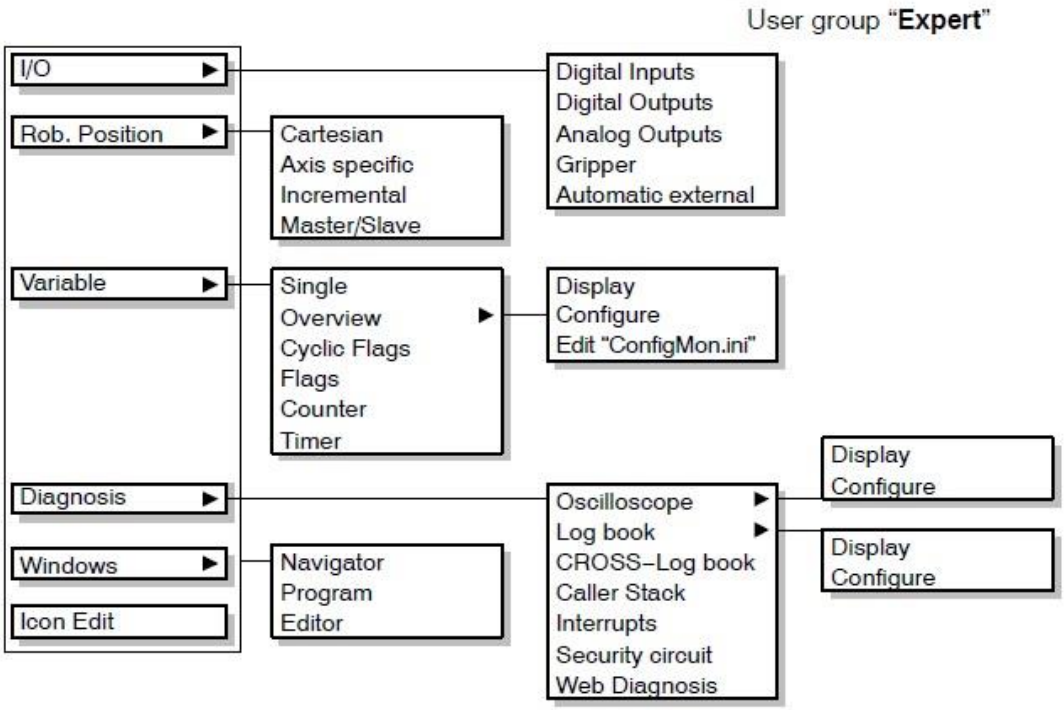
v. Μενού Παραμετροποίησης (Configure Menu)

Πιέζοντας το πλήκτρο “Configure” εμφανίζεται το παρακάτω μενού:



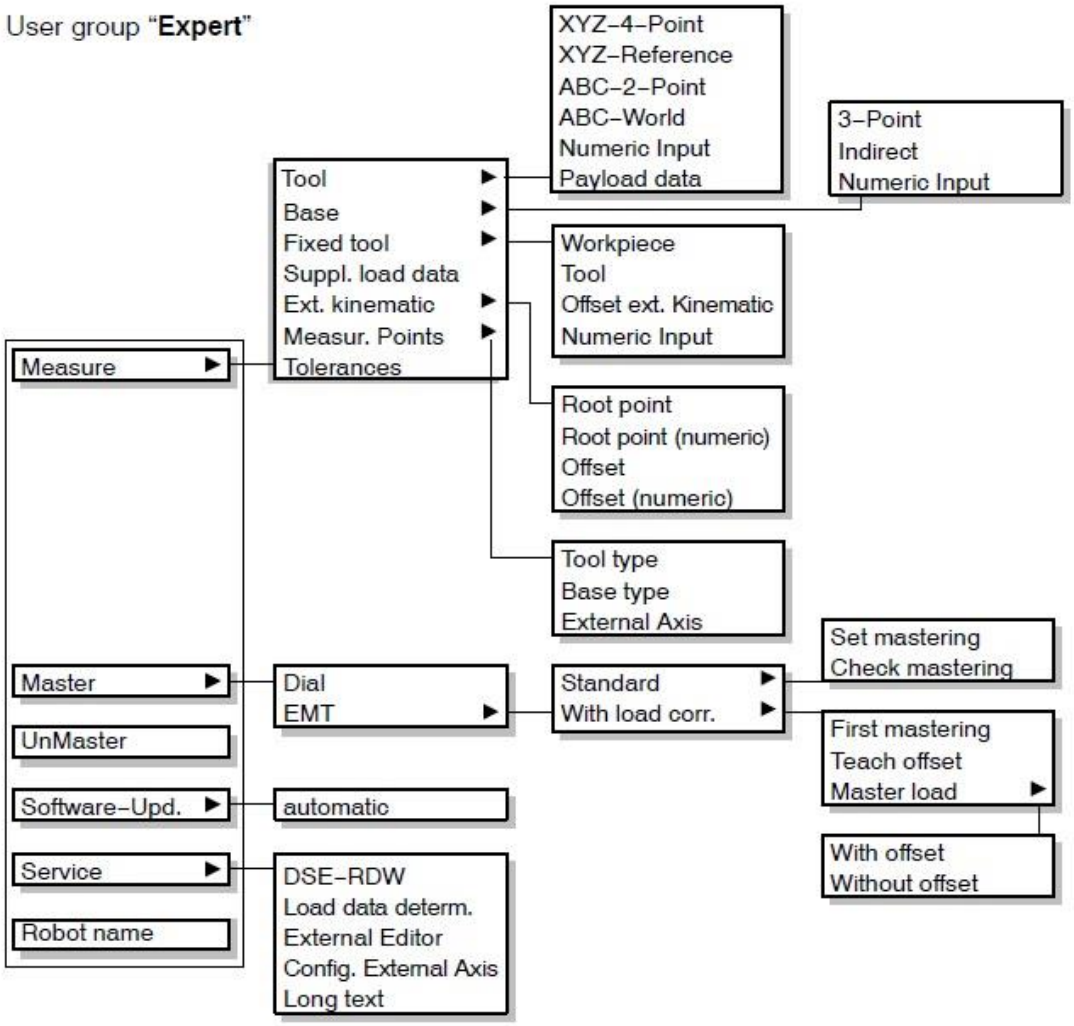
vi. Μενού Επισκόπησης (Monitor Menu)

Στο μενού “Monitor” από τις πιο βασικές λειτουργίες είναι η εμφάνιση της θέσης του ρομπότ (Robot Position).



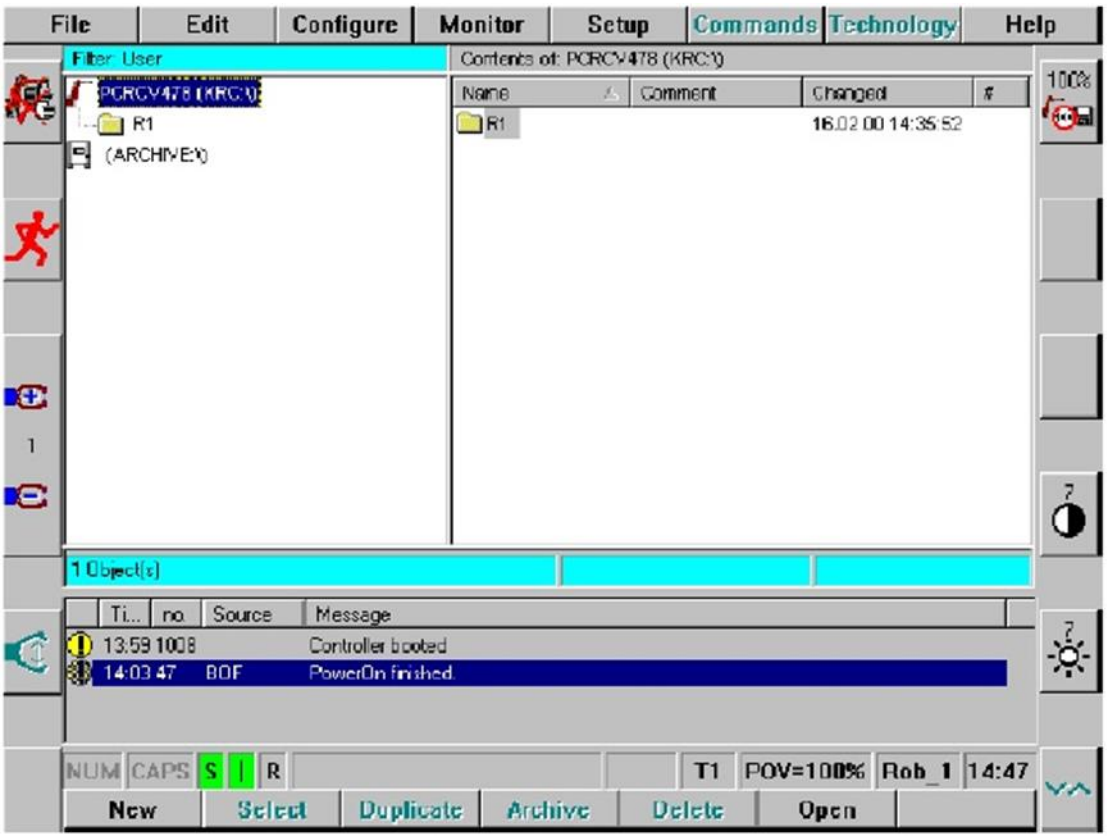
vii. Μενού Εγκατάστασης (Setup Menu)

Σημαντική λειτουργία του μενού αυτού είναι το mastering των αξόνων του ρομπότ, κατά το οποίο αποθηκεύεται η τιμή του encoder του κάθε άξονα ώστε να συμπίπτει το μηχανικό με το ηλεκτρονικό μηδέν. Δηλαδή να γνωρίζει ο βραχίονας που πραγματικά βρίσκεται.



VI. Λειτουργίες Προγράμματος

Μέσα από τον πλοηγό δίνεται η δυνατότητα να δημιουργήσουμε καινούργια προγράμματα, να επεξεργαστούμε παλιά και να τρέξουμε εκείνο το οποίο επιθυμούμε.



i. Άνοιγμα, Επεξεργασία και Αποθήκευση

Για να επεξεργαστούμε ένα πρόγραμμα πρέπει να πατήσουμε το πλήκτρο “Open” αφού το βρούμε στον κατάλογο αρχείων. Όταν γίνει αυτό το παράθυρο αλλάζει και μεταφερόμαστε στο περιβάλλον ανάπτυξης της Kuka. Από εκεί σε επίπεδο EXPERT μπορούμε απευθείας να πληκτρολογήσουμε τις εντολές μας και να διαμορφώσουμε το πρόγραμμά μας. Επίσης μπορούμε να εισάγουμε εντολές χρησιμοποιώντας τις έτοιμες φόρμες εισαγωγής που εμφανίζονται στα εναλλασσόμενα πλήκτρα (“Softkeys”).

Για να αποθηκεύσουμε το πρόγραμμά μας, πατάμε το πλήκτρο “Close” και στην συνέχεια πατάμε το πλήκτρο “Yes” για να αποθηκευτούν οι αλλαγές. Αν υπάρχει κάποιο λάθος στο πρόγραμμα θα δημιουργηθεί ένα αρχείο Error List το οποίο θα περιέχει πληροφορίες σχετικά σε ποια γραμμή είναι το λάθος, καθώς και τον τύπο του.

ii. Επιλογή και Εκτέλεση

Για να τρέξουμε ένα πρόγραμμα, πρέπει πρώτα να το επιλέξουμε με το πλήκτρο “Select” όταν βρισκόμαστε στον πλοηγό.

Στα επίπεδα T1 και T2 πρέπει να μένουν πατημένα τα πλήκτρα enabling switch και program start forwards συνεχώς για να τρέχει το πρόγραμμα.

Στο επίπεδο EXPERT πρέπει αρχικά να έχουν ενεργοποιηθεί οι οδηγοί. Στην συνέχεια πρέπει να πατηθεί συνεχόμενα το πλήκτρο program start forwards μέχρις ότου ο βραχίονας εκτελέσει την κίνηση BCO (Block Coincidence Run), δηλαδή να έρθει στην αρχική θέση (Home Position). Αφού εμφανιστεί το μήνυμα ότι η κίνηση πραγματοποιήθηκε, τότε αρκεί να πιέσουμε μόνο μια φορά το ίδιο πλήκτρο πάλι για να ξεκινήσει το πρόγραμμα να τρέχει.

5.1.2. Δομή Client - Server

Παρακάτω αναλύονται τα βήματα που πρέπει να ακολουθήσει ο χρήστης για να χρησιμοποιήσει την δομή καθώς και την λειτουργία αυτής.

I. Απαραίτητες Προ-ενέργειες

Από την μεριά του εξωτερικού υπολογιστή σε περιβάλλον Linux θα πρέπει να γίνουν οι εξής ενέργειες:

- Πρέπει να συνδεθεί ο μετατροπέας USB 2.0 σε 9-pin D-SUB (pl2303 converter) σε μία θύρα USB.

Αφού γίνει αυτό, πρέπει να εκτελεσθούν οι παρακάτω εντολές σε ένα terminal:

- ~$ lsusb
  - Από την εντολή αυτή βρίσκουμε το ID, που είναι σε μορφή ID 0000:0000

- ~$ modprobe usbserial vendor=0x____ product=0x____
  - Όπου vendor και product είναι το ID που βρήκαμε πριν.

- ~$ dmesg | grep 'ttyUSB'
  - Όπου βρίσκουμε το όνομα της θύρας, π.χ ttyUSB0

- ~$ sudo chmod 777 /dev/ttyUSB0
  - Με αυτή την εντολή παίρνουμε όλα τα δικαιώματα της θύρας

Από την μεριά του ελεγκτή του ρομπότ, πρέπει να γίνουν τα παρακάτω:

Αφού ενεργοποιηθεί πλήρως, πρέπει να επιλέξουμε user group EXPERT, ώστε να εμφανιστούν οι παραπάνω δυνατότητες που χρειαζόμαστε.

- Πιέζουμε το πλήκτρο "Configure" και μετά "User group"

- Πιέζουμε το πλήκτρο "Expert" και πληκτρολογούμε τον κωδικό kuka

- Πιέζουμε το πλήκτρο "Ok"

- Αναζητούμε το πρόγραμμα CSL_RTC (Control Systems Lab – Real Time Control) στην λίστα και το επιλέγουμε με το πλήκτρο "Select"

- Βεβαιωνόμαστε ότι βρισκόμαστε σε λειτουργία Automatic γυρίζοντας τον διακόπτη με το κλειδί στην κατάλληλη θέση (AUT)

Αφού γίνουν αυτά, ξεκινούμε πρώτα το πρόγραμμα server στην Python από το laptop το οποίο περιμένει να συνδεθεί το πρόγραμμα client. Στην συνέχεια από το χειριστήριο του ρομπότ ενεργοποιούμε τους οδηγούς (Drives ON) και πιέζουμε συνεχόμενα το πλήκτρο πρόσω εκκίνησης (Program Start Forwards) μέχρι να εμφανιστεί το μήνυμα ότι ο βραχίονας έφτασε στην θέση BCO (Block Coincidence Run). Έπειτα πιέζουμε πάλι μία φορά το πλήκτρο και το πρόγραμμα τρέχει κανονικά και αυτόματα. Το ρομπότ τώρα ελέγχεται από τις εντολές που δίνουμε μέσω του εξωτερικού υπολογιστή.

Όταν τερματίσει ο χρήστης το πρόγραμμα server, τερματίζεται αυτόματα και το πρόγραμμα client. Ο χρήστης πρέπει στην συνέχεια να ακυρώσει το πρόγραμμα του ρομπότ πατώντας το πλήκτρο “Program” και στην συνέχεια “Cancel Program” για να βρεθεί στο περιβάλλον του πλοηγού.

II. Αρχή Λειτουργίας

Το πρόγραμμα server μπορεί να τρέξει και σε περιβάλλον Linux και σε περιβάλλον Windows, καθώς χρησιμοποιούνται μόνο βασικά πακέτα της γλώσσας Python τα οποία είναι ανεξάρτητα του λειτουργικού συστήματος.

Το πρόγραμμα αρχικά εντοπίζει τις διαθέσιμες θύρες και ο χρήστης θα πρέπει να εισάγει αυτή που επιθυμεί. Αν δεν υπάρχουν θύρες, το πρόγραμμα ενημερώνει τον χρήστη και τερματίζεται.

Αφού γίνουν τα απαραίτητα βήματα, το πρόγραμμα αυτό αναμένει την σύνδεση του client. Μόλις συνδεθεί ο client, καλείται η συνάρτηση handshake έτσι ώστε να γνωρίζουν και τα δύο προγράμματα ότι έχει γίνει η σύνδεση επιτυχώς.

Στην συνέχεια το πρόγραμμα client περιμένει εντολές από το πρόγραμμα server. Ο χρήστης μπορεί ανά πάσα στιγμή να αλλάξει τις τιμές της μέγιστης ταχύτητας και της επιτάχυνσης αλλά μπορεί και να διαλέξει αν θα στέλνει τις εντολές κίνησης σχετικά με το προηγούμενο σημείο (relative) ή απόλυτες τιμές (absolute).

Το πρόγραμμα client είναι δομημένο ώστε συνέχεια να αναμένει εντολές είτε κίνησης είτε για να στείλει δεδομένα. Τερματίζει μόνο κατά εντολή του server.

Παρακάτω παρατίθεται η πιο απλή μορφή προγράμματος που μπορεί να τρέξει:

```
1.  my_robot = kuka()
2.
3.
4.  my_robot.start_program()
5.  #my_robot.enable_print()
6.  my_robot.set_velocity_and_acceleration(7,14)
7.
8.  my_robot.start_moving()
9.
10. x = [ 0.0 , -90.0 , 90.0 , 0.0 , 0.0 , 0.0 ]
```

```
11.
12. a = 0.0
13. b = -90.0
14. c = 90.0
15. d = 0.0
16. e = 0.0
17. f = 0.0
18.
19. step = 1
20.
21. while ( (x[0] < 90 ) or (x[1] > -120) or (x[2] > 60) or (x[3] < 90) or (x[4] < 90) or (x[5] < 90) ):
22.
23.     x = my_robot.read_robot_position()
24.     #my_robot.print_current_positional_values()
25.     my_robot.move_to ( f1 = a , f2 = b , f3 = c , f4 = d , f5 = e , f6 = f )
26.
27.     if (a < 90):
28.         a += step
29.     if (b > -120):
30.         b -= step
31.     if (c > 60):
32.         c -= step
33.     if (d < 90):
34.         d += step
35.     if (e < 90):
36.         e += step
37.     if (f < 90):
38.         f += step
39.
40.
41. my_robot.stop_moving()    #stop
42.
43. my_robot.end_program()
```

Οι εντολές read_robot_position και move_to πρέπει πάντα να ακολουθεί η μία την άλλη με αυτή τη σειρά καθότι με αυτόν τον τρόπο γίνεται η επικοινωνία.

5.2. Κώδικας

Παρακάτω παρατίθεται ο κώδικας για το πρόγραμμα client γραμμένος στην γλώσσα προγραμματισμού KRL και για το πρόγραμμα server στην γλώσσα Python.

Για την λεπτομερή επεξήγηση των εντολών στην γλώσσα της KUKA ο χρήστης θα πρέπει να ανατρέξει στα εγχειρίδια:

- Reference Guide v4_1
- Serial Read Write manual
- Programming User Messages
- KRL System Variables manual

5.2.1. Πρόγραμμα KRL (Client)

Το πρόγραμμα BAS είναι το βασικό πακέτο προγραμμάτων του ρομπότ και συμπεριλαμβάνεται εδώ για τον ορισμό της μέγιστης επιτρεπόμενης ταχύτητας και επιτάχυνσης.

Οι μεταβλητές που χρησιμοποιεί το πρόγραμμα CSL_RTC είναι ορισμένες στο αρχείο **$config.dat** στον ίδιο φάκελο με το πρόγραμμα. Συγκεκριμένα:

```
;==================================
; Control Systems Lab edit - start
;==================================
BOOL THREAD_IS_ENABLED=FALSE
INT RTC_ELEMENT=6
AXIS RTC_BUFFER[6]

INT CSL_HANDLE=1
AXIS CSL_HOME={A1 0.0,A2 -90.0,A3 90.0,A4 0.0,A5 0.0,A6 0.0}
AXIS CSL_AXIS_ACT

INT CSL_READ_ERRORS=0
INT CSL_WRITE_ERRORS=0
;==================================
; Control Systems Lab edit - end
;==================================
```

```
DEF  CSL_RTC ( )

;###################################
;                                   #
; Creator: Angelos Stathis          #
;          mc09672                  #
; E-mail:  angel_stathis@yahoo.gr   #
;                                   #
;###################################

;Import programs

EXT  BAS (BAS_COMMAND  :IN,REAL  :IN )
EXT  SER_OPEN ( )
EXT  SER_CLOS ( )
EXT  SER_READ (CHAR [] :OUT )
EXT  SER_WRIT (CHAR [] :OUT )
EXT  MSG_OUT (CHAR [] :OUT,REAL  :IN )
EXT  THREAD ( )

CHAR PROG_STATE[3],READ_ERRORS[25],WRITE_ERRORS[25]
INT I
DECL AXIS SECURE_A2_A3

$ADVANCE=5
$APO.CPTP=1

CSL_READ_ERRORS=0
CSL_WRITE_ERRORS=0

;Initialize target values
FOR I=1 TO  6
RTC_BUFFER[I]=CSL_HOME
ENDFOR

;No offset between flange and tool coordinate systems
$TOOL={X 0,Y 0,Z 0,A 0,B 0,C 0}

;No base system declared
$BASE={X 0,Y 0,Z 0,A 0,B 0,C 0}

;Set values using BAS - set as % of maximum values

BAS (#ACC_PTP,1 )
BAS (#VEL_PTP,1 )

PTP  CSL_HOME ;BCO run

INITIALIZE ( )

WHILE  TRUE

SER_READ (PROG_STATE[] )
;MSG_OUT (PROG_STATE[],0 )

IF ((PROG_STATE[1]=="E") AND (PROG_STATE[2]=="N") AND
(PROG_STATE[3]=="D")) THEN
EXIT
ENDIF
```

```
IF ((PROG_STATE[1]=="V") AND (PROG_STATE[2]=="A") AND
(PROG_STATE[3]=="V")) THEN
MOVEMENT_VALUES ( )
ENDIF

IF ((PROG_STATE[1]=="P") AND (PROG_STATE[2]=="T") AND
(PROG_STATE[3]=="P")) THEN
MOVE_TO ( )
ENDIF

ENDWHILE

$ADVANCE=0

BAS (#ACC_PTP,3 )
BAS (#VEL_PTP,3 )

SECURE_A2_A3=$AXIS_ACT
SECURE_A2_A3.A2=-90
SECURE_A2_A3.A3=90

PTP  SECURE_A2_A3 ; Secure axes 2 and 3
PTP  CSL_HOME ; Return home

READ_ERRORS[]="Total read errors ="
WRITE_ERRORS[]="Total write errors ="

TERMINATE ( )

MSG_OUT (READ_ERRORS[],CSL_READ_ERRORS )
MSG_OUT (WRITE_ERRORS[],CSL_WRITE_ERRORS )

END

;###################################

DEF  HANDSHAKE ( )

CHAR STR1[3],STR2[3],MES1[25],MES2[15]

MES1[]="Handshake Successful!!"
MES2[]="Handshake Error"
STR1[]="111"
STR2[]="   "

SER_WRIT (STR1[] )
SER_READ (STR2[] )

IF ((STR1[1]==STR2[1]) AND (STR1[2]==STR2[2]) AND (STR1[3]==STR2[3]))
THEN
MSG_OUT (MES1[],0 )

ELSE
MSG_OUT (MES2[],0 )
CONTINUE
HALT

ENDIF

END
```

```
;###################################

DEF  MOVEMENT_VALUES ( )

CHAR VELOCITY[3],ACCELERATION[3]
INT VEL,ACC,OFFSET
DECL STATE_T STATE

;Read the values from serial:

SER_READ (VELOCITY[] )
SER_READ (ACCELERATION[] )

OFFSET=0
CONTINUE
SREAD(VELOCITY[],STATE,OFFSET,"%d",VEL)

OFFSET=0
CONTINUE
SREAD(ACCELERATION[],STATE,OFFSET,"%d",ACC)

;Set the values using BAS - set as % of maximum values

BAS (#VEL_PTP,VEL )
BAS (#ACC_PTP,ACC )

END

;###################################

DEF  MOVE_TO ( )

CHAR KEEP_GOING[1]

WHILE  TRUE

SER_READ (KEEP_GOING[] )

IF (KEEP_GOING[1]=="Y") THEN
MOVE_TO_TARGET ( )
ENDIF

IF (KEEP_GOING[1]=="N") THEN
EXIT
ENDIF

ENDWHILE

END

;###################################

DEF  MOVE_TO_TARGET ( )

IF (RTC_ELEMENT==6) THEN

THREAD_IS_ENABLED=TRUE
THREAD ( )

;PRINT_AXES (RTC_BUFFER[1] )
PTP  RTC_BUFFER[1] C_PTP
```

```
CONTINUE
WAIT FOR  (THREAD_IS_ENABLED==FALSE)
RTC_ELEMENT=1

ELSE

THREAD_IS_ENABLED=TRUE
THREAD ( )

;PRINT_AXES (RTC_BUFFER[RTC_ELEMENT+1] )
PTP  RTC_BUFFER[RTC_ELEMENT+1] C_PTP
CONTINUE
WAIT FOR  (THREAD_IS_ENABLED==FALSE)
RTC_ELEMENT=RTC_ELEMENT+1

ENDIF

END

;###################################

DEF  INITIALIZE ( )

SER_OPEN ( )

HANDSHAKE ( )

END

;###################################

DEF  TERMINATE ( )

CHAR OK[2]

OK[]="OK"

SER_WRIT (OK[] )

SER_CLOS ( )

END

;###################################

DEF  PRINT_AXES (POSITION )

DECL AXIS POSITION
INT OFFSET
DECL STATE_T STATE
CHAR POS_STRING[45]

OFFSET=0

CONTINUE
SWRITE(POS_STRING[],STATE,OFFSET,"%.2f %.2f %.2f %.2f %.2f
%.2f",POSITION.A1,POSITION.A2,POSITION.A3,POSITION.A4,POSITION.A5,POS
ITION.A6)

MSG_OUT (POS_STRING[],0 )
```

```
END
```

```
DEF  THREAD ( )

EXT  MSG_OUT (CHAR [] :OUT,REAL  :IN )
EXT  SER_WRIT (CHAR [] :OUT )
EXT  SER_R6FL (REAL  :OUT,REAL  :OUT,REAL  :OUT,REAL  :OUT,REAL
:OUT,REAL  :OUT )

SEND_POSITION ( )
GET_POSITION ( )

THREAD_IS_ENABLED=FALSE

END

;###################################

DEF  SEND_POSITION ( )

CHAR POS_STRING[60]
INT OFFSET
DECL STATE_T STATE

CONTINUE
CSL_AXIS_ACT=$AXIS_ACT
OFFSET=0

CONTINUE
SWRITE(POS_STRING[],STATE,OFFSET,"%.2f %.2f %.2f %.2f %.2f
%.2f",CSL_AXIS_ACT.A1,CSL_AXIS_ACT.A2,CSL_AXIS_ACT.A3,CSL_AXIS_ACT.A4
,CSL_AXIS_ACT.A5,CSL_AXIS_ACT.A6)

;MSG_OUT (POS_STRING[],0 )
SER_WRIT (POS_STRING[] )

END

;###################################

DEF  GET_POSITION ( )

REAL F1,F2,F3,F4,F5,F6

SER_R6FL (F1,F2,F3,F4,F5,F6 )

RTC_BUFFER[RTC_ELEMENT].A1=F1
RTC_BUFFER[RTC_ELEMENT].A2=F2
RTC_BUFFER[RTC_ELEMENT].A3=F3
RTC_BUFFER[RTC_ELEMENT].A4=F4
RTC_BUFFER[RTC_ELEMENT].A5=F5
RTC_BUFFER[RTC_ELEMENT].A6=F6

END
```

```
DEF  SER_OPEN ( )

EXT  MSG_OUT (CHAR [] :OUT,REAL  :IN )
CHAR MES[40]

WHILE  TRUE

CONTINUE
COPEN(:SER_1,CSL_HANDLE)

IF (CSL_HANDLE==1) THEN
MES[]="Serial Port Open: COM"
MSG_OUT (MES[],CSL_HANDLE )
EXIT

ELSE
MES[]="Error while Opening Serial Port"
MSG_OUT (MES[],CSL_HANDLE )
CONTINUE
HALT

ENDIF

ENDWHILE

END
```

```
DEF  SER_CLOS ( )

EXT  MSG_OUT (CHAR [] :OUT,REAL  :IN )
DECL STATE_T STATE
CHAR MES[40]

WHILE  TRUE

CONTINUE
CCLOSE(CSL_HANDLE,STATE)

IF (STATE.RET1==#CMD_OK) THEN
MES[]="Serial Port Closed: COM"
MSG_OUT (MES[],CSL_HANDLE )
EXIT

ELSE
MES[]="Error while Closing Serial Port: COM"
MSG_OUT (MES[],CSL_HANDLE )
CONTINUE
HALT

ENDIF

ENDWHILE

END
```

```
DEF  SER_READ (READ_CHAR[] :OUT )

EXT  MSG_OUT (CHAR [] :OUT,REAL  :IN )
CHAR MES[40]

INT OFFSET
REAL TIMEOUT
CHAR READ_CHAR[]
DECL STATE_T STATE
DECL MODUS_T MODUS

WHILE  TRUE

MODUS=#ABS
OFFSET=0
TIMEOUT=0.3

;WHILE  TRUE

CONTINUE
WAIT FOR  ($DATA_SER1>0)

;CONTINUE
;MES[]="DATA_SER1 = "
;MSG_OUT (MES[],$DATA_SER1 )

CONTINUE
CREAD(CSL_HANDLE,STATE,MODUS,TIMEOUT,OFFSET,"%s",READ_CHAR[])

IF  NOT ((STATE.RET1==#DATA_OK) OR (STATE.RET1==#DATA_END) OR
(STATE.RET1==#CMD_OK)) THEN
;IF  NOT (STATE.RET1==#CMD_OK) THEN

CSL_READ_ERRORS=CSL_READ_ERRORS+1

;MSG_OUT (READ_CHAR[],0 )

IF (STATE.RET1==#DATA_END) THEN
MES[]="SER_READ: DATA_END"
MSG_OUT (MES[],$DATA_SER1 )
ENDIF
IF (STATE.RET1==#DATA_OK) THEN
MES[]="SER_READ: DATA_OK"
ENDIF
IF (STATE.RET1==#CMD_ABORT) THEN
MES[]="SER_READ: CMD_ABORT"
ENDIF
IF (STATE.RET1==#CMD_REJ) THEN
MES[]="SER_READ: CMD_REJ"
ENDIF
IF (STATE.RET1==#CMD_PART) THEN
MES[]="SER_READ: CMD_PART"
ENDIF
IF (STATE.RET1==#CMD_SYN) THEN
MES[]="SER_READ: CMD_SYN"
ENDIF
IF (STATE.RET1==#CMD_TIMEOUT) THEN
MES[]="SER_READ: CMD_TIMEOUT"
ENDIF
IF (STATE.RET1==#DATA_BLK) THEN
MES[]="SER_READ: DATA_BLK"
```

```
ENDIF
IF (STATE.RET1==#FMT_ERR) THEN
MES[]="SER_READ: FMT_ERR"
ENDIF
IF (STATE.HITS>0) THEN
MES[]="SER_READ: HITS"
ENDIF
IF (STATE.LENGTH>0) THEN
MES[]="SER_READ: LENGTH"
ENDIF

MSG_OUT (MES[],$DATA_SER1 )

ELSE
;MSG_OUT (READ_CHAR[],0 )
EXIT

ENDIF

ENDWHILE

END
```

```
DEF  SER_WRIT (IN_STRING[] :OUT )

EXT  MSG_OUT (CHAR [] :OUT,REAL  :IN )
CHAR MES[40]

DECL CHAR IN_STRING[]
DECL STATE_T STATE
DECL MODUS_T MODUS

MODUS=#SYNC

WHILE  TRUE

CONTINUE
CWRITE(CSL_HANDLE,STATE,MODUS,"%s",IN_STRING[])

IF  NOT ((STATE.RET1==#CMD_OK) OR (STATE.RET1==#DATA_OK)) THEN

CSL_WRITE_ERRORS=CSL_WRITE_ERRORS+1

IF (STATE.RET1==#DATA_OK) THEN
MES[]="SER_WRITE: DATA_OK"
ENDIF
IF (STATE.RET1==#CMD_ABORT) THEN
MES[]="SER_WRITE: CMD_ABORT"
ENDIF
IF (STATE.RET1==#CMD_REJ) THEN
MES[]="SER_WRITE: CMD_REJ"
ENDIF
IF (STATE.RET1==#CMD_PART) THEN
MES[]="SER_WRITE: CMD_PART"
ENDIF
IF (STATE.RET1==#CMD_SYN) THEN
MES[]="SER_WRITE: CMD_SYN"
ENDIF
IF (STATE.RET1==#CMD_TIMEOUT) THEN
MES[]="SER_WRITE: CMD_TIMEOUT"
ENDIF
IF (STATE.RET1==#DATA_BLK) THEN
MES[]="SER_WRITE: DATA_BLK"
ENDIF
IF (STATE.RET1==#DATA_END) THEN
MES[]="SER_WRITE: DATA_END"
ENDIF
IF (STATE.RET1==#FMT_ERR) THEN
MES[]="SER_WRITE: FMT_ERR"
ENDIF

MSG_OUT (MES[],0 )

ELSE
EXIT

ENDIF

ENDWHILE

END
```

```
DEF  SER_R6FL (F1 :OUT,F2 :OUT,F3 :OUT,F4 :OUT,F5 :OUT,F6 :OUT )

EXT  MSG_OUT (CHAR [] :OUT,REAL  :IN )
CHAR MES[40]

INT READ_OFFSET,WRITE_OFFSET
REAL READ_TIMEOUT,F1,F2,F3,F4,F5,F6
CHAR S1[8],S2[8],S3[8],S4[8],S5[8],S6[8]
DECL STATE_T SR_T
DECL MODUS_T MR_T

MR_T=#ABS
READ_OFFSET=0
WRITE_OFFSET=0
READ_TIMEOUT=0.3

WHILE  TRUE

S1[]=" "
S2[]=" "
S3[]=" "
S4[]=" "
S5[]=" "
S6[]=" "

CONTINUE
WAIT FOR  ($DATA_SER1>0)

READ_OFFSET=0
CONTINUE
CREAD(CSL_HANDLE,SR_T,MR_T,READ_TIMEOUT,READ_OFFSET,"%s %s %s %s %s
%s",S1[],S2[],S3[],S4[],S5[],S6[])

;MSG_OUT (S1[],0 )
;MSG_OUT (S2[],0 )
;MSG_OUT (S3[],0 )
;MSG_OUT (S4[],0 )
;MSG_OUT (S5[],0 )
;MSG_OUT (S6[],0 )

WRITE_OFFSET=0
CONTINUE
SREAD(S1[],SR_T,WRITE_OFFSET,"%f",F1)

WRITE_OFFSET=0
CONTINUE
SREAD(S2[],SR_T,WRITE_OFFSET,"%f",F2)

WRITE_OFFSET=0
CONTINUE
SREAD(S3[],SR_T,WRITE_OFFSET,"%f",F3)

WRITE_OFFSET=0
CONTINUE
SREAD(S4[],SR_T,WRITE_OFFSET,"%f",F4)

WRITE_OFFSET=0
CONTINUE
SREAD(S5[],SR_T,WRITE_OFFSET,"%f",F5)

WRITE_OFFSET=0
```

```
CONTINUE
SREAD(S6[],SR_T,WRITE_OFFSET,"%f",F6)

;MES[] = "READ_6_FLOATS: F1="
;MSG_OUT (MES[],F1 )
;MES[] = "READ_6_FLOATS: F2="
;MSG_OUT (MES[],F2 )
;MES[] = "READ_6_FLOATS: F3="
;MSG_OUT (MES[],F3 )
;MES[] = "READ_6_FLOATS: F4="
;MSG_OUT (MES[],F4 )
;MES[] = "READ_6_FLOATS: F5="
;MSG_OUT (MES[],F5 )
;MES[] = "READ_6_FLOATS: F6="
;MSG_OUT (MES[],F6 )

;IF  NOT ((SR_T.RET1==#DATA_OK) OR (SR_T.RET1==#DATA_END)) THEN

IF  NOT ((SR_T.RET1==#DATA_OK) OR (SR_T.RET1==#DATA_END) OR
(SR_T.RET1==#CMD_OK)) THEN

CSL_READ_ERRORS=CSL_READ_ERRORS+1

;IF (SR_T.RET1<>#CMD_OK) THEN

IF (SR_T.RET1==#DATA_END) THEN
MES[]="SER_READ_6_FLOATS: DATA_END"
ENDIF
IF (SR_T.RET1==#DATA_OK) THEN
MES[]="SER_READ_6_FLOATS: DATA_OK"
ENDIF
IF (SR_T.RET1==#CMD_ABORT) THEN
MES[]="SER_READ_6_FLOATS: CMD_ABORT"
ENDIF
IF (SR_T.RET1==#CMD_REJ) THEN
MES[]="SER_READ_6_FLOATS: CMD_REJ"
ENDIF
IF (SR_T.RET1==#CMD_PART) THEN
MES[]="SER_READ_6_FLOATS: CMD_PART"
ENDIF
IF (SR_T.RET1==#CMD_SYN) THEN
MES[]="SER_READ_6_FLOATS: CMD_SYN"
ENDIF
IF (SR_T.RET1==#CMD_TIMEOUT) THEN
MES[]="SER_READ_6_FLOATS: CMD_TIMEOUT"
ENDIF
IF (SR_T.RET1==#DATA_BLK) THEN
MES[]="SER_READ_6_FLOATS: DATA_BLK"
ENDIF
IF (SR_T.RET1==#DATA_END) THEN
MES[]="SER_READ_6_FLOATS: DATA_END"
ENDIF
IF (SR_T.RET1==#FMT_ERR) THEN
MES[]="SER_READ_6_FLOATS: FMT_ERR"
ENDIF

MSG_OUT (MES[],0 )

ELSE
EXIT
```

```
ENDIF

ENDWHILE

END
```

```
DEF  MSG_OUT (MES[] :OUT,VALUE :IN )

DECL MSG_T EMPTY_MSG
DECL CHAR MES[]
REAL VALUE
DECL STATE_T STATE
INT OFFSET

;CONTINUE
EMPTY_MSG={MSG_T: VALID FALSE,RELEASE FALSE,TYP #NOTIFY,MODUL[] "
",KEY[] " ",PARAM_TYP #VALUE,PARAM[] " ",DLG_FORMAT[] " ",ANSWER 0}

OFFSET=0
IF (VALUE<>0) THEN
CONTINUE
SWRITE($MSG_T.KEY[],STATE,OFFSET,"%s %.2f",MES[],VALUE)
ENDIF

IF (VALUE==0) THEN
CONTINUE
SWRITE($MSG_T.KEY[],STATE,OFFSET,"%s",MES[])
ENDIF

$MSG_T.VALID=FALSE
$MSG_T.RELEASE=FALSE
$MSG_T.TYP=#NOTIFY
$MSG_T.PARAM_TYP=#VALUE
$MSG_T.PARAM[]=" "
$MSG_T.DLG_FORMAT[]=" "
$MSG_T.ANSWER=0

$MSG_T.VALID=TRUE
WHILE  $MSG_T.VALID
CONTINUE
WAIT SEC 0.05
ENDWHILE
CONTINUE
WAIT SEC 0.3
$MSG_T=EMPTY_MSG

END
```

5.2.2. Πρόγραμμα Python (Server)

```
#!/Python36/python
# -*- coding: UTF-8 -*-

#########################################################
#                                                       #
#   Creator:    Angelos Stathis                         #
#               mc09672                                 #
#   e-mail:     angel_stathis@yahoo.gr                  #
#                                                       #
#########################################################

##### List of imports

import serial
import sys
import glob
import numbers
from timeit import default_timer as timer
import time
import math
import os

##### End of List

#############################################################################################################

class kuka:

    """       Usable functions of class kuka_robot:

                start_program()
                end_program()
                enable_print()              This function enables all comments for debugging
                disable_print()
                print_current_position()
                move_to()                   Move to an absolute target
                move_to_relative()          Move to a relative target
                reset_position()            Move to home position
    """

    ###########################################################################################################

    def __init__ ( self ):

        # Current position of the robot

        self.__is_A1 = 0.0     # °
        self.__is_A2 = -90.0   # °
        self.__is_A3 = 90.0    # °
        self.__is_A4 = 0.0     # °
        self.__is_A5 = 0.0     # °
        self.__is_A6 = 0.0     # °

        # Target position of the robot

        self.__target_A1 = 0.0     # °
        self.__target_A2 = -90.0   # °
        self.__target_A3 = 90.0    # °
        self.__target_A4 = 0.0     # °
```

```
60.         self.__target_A5 = 0.0     # °
61.         self.__target_A6 = 0.0     # °
62.
63.         # Protocol 3964R bytes:
64.
65.         self.__STX = 0x2
66.         self.__ETX = 0x3
67.         self.__DLE = 0x10
68.         self.__NAK = 0x15
69.
70.         self.__BYTE_STX = bytes(chr(0x2),'ascii')
71.         self.__BYTE_ETX = bytes(chr(0x3),'ascii')
72.         self.__BYTE_DLE = bytes(chr(0x10),'ascii')
73.         self.__BYTE_NAK = bytes(chr(0x15),'ascii')
74.
75.         # Enabling - Disabling printing of all comments
76.
77.         self.__print_value = 0    # Default is disabled
78.
79.         # Initializing velocity and acceleration
80.
81.         self.__velocity = 1
82.         self.__acceleration = 1
83.
84.         # Creating a serial class object
85.
86.         self.robot_ser = serial.Serial()
87.
88.         # Program Status
89.
90.         self.__program_status = 0
91.
92.         # Current Working Directory
93.
94.         self.__cwd = os.getcwd()
95.         self.__path = self.__cwd + "/kuka_robot_log.txt"
96.         self.__path_dbg = self.__cwd + "/kuka_robot_debug.txt"
97.
98.         # Create Log File
99.
100.         self.__log_file =  open ( self.__path , 'w' )
101.         self.__log_file.close()
102.
103.         self.__debugger = open ( self.__path_dbg , 'w' )
104.         self.__debugger.close()
105.
106.         # Timer
107.
108.         self.__start = 0.0
109.
110.         # Counter
111.
112.         self.__counter = 0
113.
114.         self.__total_time = 0.0
115.         self.__mean_time = 0.0
116.
117.     ##########################################################################
    ##################################
118.
119.     def reset_position ( self ):
120.
121.         if self.__program_status == 0:
122.             print ( "---Program not started!!!\n")
123.             return 0
124.
```

```
125.        print ( '---Reseting position...\n' )
126.
127.        velocity_old_value      =   self.__velocity         # backup values
128.        acceleration_old_value  =   self.__acceleration
129.
130.        self.stop_moving ( )                                # exit movement
    loop
131.
132.        self.set_velocity_and_acceleration ( 1 , 10 )       # set new, safer
     values
133.
134.        self.start_moving ( )                               # enter movement
     loop
135.
136.        while ((self.__is_A2 != -90) and (SELF.__is_A3 != 90)):
137.            self.read_robot_position ( )
138.            self.move_to ( f2 = -
    90, f3 = 90 )              # secure axis 2 & 3 from collision
139.
140.        while ((self.__is_A1 != 0.0) and (self.__is_A2 != -
    90) and (self.__is_A3 != 90) and (self.__is_A4 != 0.0) and (self.__is_A5 != 0
    .0) and (self.__is_A6 != 0.0)):
141.            self.read_robot_position ( )
142.            self.move_to ( )                                # move to home p
    osition
143.
144.        self.stop_moving ( )                                #  re-
    exit movement loop
145.
146.        self.set_velocity_and_acceleration ( velocity_old_value , accelerati
    on_old_value )  # restore previous values
147.
148.        self.start_moving ( )                               # re-
    enter movement loop
149.
150.        print ( '---Position reset!!\n' )
151.
152.    ##########################################################################
    ##################################
153.
154.    def __print_all ( self , *var ):
155.
156.        """Enables or disables extra debugging print() calls"""
157.
158.        #self.__debugger.write(*var)
159.
160.        if self.__print_value == 1:
161.            print ( *var )
162.
163.    ##########################################################################
    ##################################
164.
165.    def enable_print ( self ):
166.
167.        if self.__program_status == 0:
168.            print ( "---Program not started!!!\n")
169.            return 0
170.
171.        self.__print_value = 1
172.
173.    ##########################################################################
    ##################################
174.
175.    def disable_print ( self ):
176.
```

```
177.        if self.__program_status == 0:
178.            print ( "---Program not started!!!\n")
179.            return 0
180.
181.        self.__print_value = 0
182.
183.    ##########################################################################
    ################################
184.
185.    def __find_serial_ports ( self ):
186.
187.        """ Lists serial port names
188.            :raises EnvironmentError:
189.                On unsupported or unknown platforms
190.            :returns:
191.                A list of the serial ports available on the system"""
192.
193.        if sys.platform.startswith('win'):
194.
195.            self.__print_all ( '---Your Operating System is Windows\n' )
196.            ports = ['COM%s' % (i + 1) for i in range(256)]
197.
198.        elif sys.platform.startswith ( 'linux' ) or sys.platform.startswith
    ( 'cygwin' ):
199.
200.            self.__print_all ( '---Your Operating System is GNU/Linux\n' )
201.            ports = glob.glob ('/dev/tty[A-Za-
    z]*') # this excludes your current terminal "/dev/tty"
202.
203.        else:
204.            print ( '---
    Unsupported platform! ( Windows or GNU/Linux only! )\n' )
205.            print ( '---Program ended\n' )
206.            #raise EnvironmentError('\nUnsupported platform\n')
207.            return 0
208.
209.        result = []
210.        for port in ports:
211.            try:
212.                ser_test = serial.Serial ( port )
213.                ser_test.close()
214.                result.append ( port )
215.            except ( OSError , serial.SerialException ):
216.                pass
217.        if (result == []):
218.            result = 'No serial ports availiable!'
219.            print ( '---Ports availiable:' , result , '\n' )
220.            print ( '---Program ended\n' )
221.            return 0
222.        else:
223.            print ( '---Ports availiable:' , result , '\n' )
224.            return
225.
226.    ##########################################################################
    ################################
227.
228.    def __open_serial ( self  ):
229.
230.        """Gives user the choice to open an availiable serial port or to exi
    t the program"""
231.
232.        while True:
233.
234.            port_chosen = input ( '---
    Please choose serial port from ports available (type "exit" to exit): ' )
235.
```

```
236.            if port_chosen == 'exit':
237.                print ( '\n---Program ended by user\n' )
238.                return 0
239.            try:
240.                self.robot_ser = serial.Serial(    port_chosen,
241.                                               115200,
242.                                               bytesize = serial.EIGHTBITS,
243.                                               parity = serial.PARITY_EVEN,
244.                                               stopbits = serial.STOPBITS_ONE,
245.                                               timeout = 0.5,
246.                                               rtscts = False,
247.                                               dsrdtr = False ) # open serial port

248.                print ( )
249.                self.__print_all ( '---
    Port used: ',self.robot_ser.name,'\n' ) # check which port was really used
250.                self.__print_all ( '---
    Port info: ',self.robot_ser.isOpen,'\n' )
251.                return
252.            except ( OSError , serial.SerialException ):
253.                print( '\n---Error typing, please check spelling\n' )
254.
255.    ##########################################################################
    ##################################
256.
257.    def __close_serial ( self  ):
258.
259.        """Closes serial port"""
260.
261.        self.__print_all ( '---
    Port info: ' , self.robot_ser.isOpen , '\n' )
262.        self.__print_all ( '---
    Closing port: ' , self.robot_ser.name, '\n' )
263.
264.        self.robot_ser.close ( )
265.
266.        return
267.
268.    ##########################################################################
    ##################################
269.
270.    def __form_3964r_block ( self , input_string ):
271.
272.        """Forms the block of protocol 3964R by adding to the input string (
    now bytes string) the DLE,ETX, BCC bytes"""
273.
274.        bcc = self.__DLE ^ self.__ETX       #BCC stands for Block Check Char
    acter
275.
276.        for i in range ( 0 , len( input_string ) ):
277.
278.            #self.__print_all( '---
    ' , 'bcc :' , bin(bcc) , 'XOR', input_string[i] , ':' , bin( ord( input_strin
    g[i] ) ) , 'is' )
279.
280.            bcc = bcc ^ ord ( input_string[i] )
281.
282.            self.__print_all( '---' , bcc , "\n" )
283.            #self.__print_all( '---new bcc is' , bin( bcc ) )
284.
285.        try:
286.            BYTE_BCC = bytes( chr( bcc ), 'ascii' )
287.            self.__print_all ( "---
    form_3964r_block: BCC is:" , BYTE_BCC , "\n" )
288.            output_string = bytes( input_string , 'ascii' ) + self.__BYTE_DL
    E + self.__BYTE_ETX + BYTE_BCC
```

```
289.
290.             self.__print_all( '---
    form_3964r_block:' , output_string , '\n' )
291.
292.             return output_string
293.
294.         except UnicodeError:
295.             print( '---
    form_3964r_block: String data not compatible with ASCII\n' )
296.             return 'ascii_problem'
297.
298.     ##########################################################################
    ##################################
299.
300.     def __bytes_waiting ( self ):
301.
302.         """Checks if there are data waiting in the input buffer"""
303.
304.         number_of_bytes = 0
305.         start = timer()
306.
307.         while number_of_bytes <= 0:
308.
309.             number_of_bytes = self.robot_ser.inWaiting()
310.             end = timer()
311.
312.             if end-start > 5:
313.                 self.__print_all('---
    bytes_waiting: timeout exceeded: input buffer is empty\n')
314.                 return number_of_bytes
315.
316.         self.__print_all('---bytes_waiting: Data found\n')
317.
318.         return number_of_bytes
319.
320.     ##########################################################################
    ##################################
321.
322.     def __read_string ( self ):
323.
324.         """Forms a string of bytes while reading one byte at a time from the
     serial port"""
325.
326.         x = b''
327.         #start = timer()
328.
329.         while True:
330.
331.             self.__print_all( "---read_string: Currently in use" )
332.             c = self.robot_ser.read(1)
333.             x += c
334.             end = timer()
335.
336.             if c == self.__BYTE_DLE:
337.                 c = self.robot_ser.read(2)
338.                 self.__print_all ( "---
    read_string: DLE received, bytes found:" , c )
339.                 x += c
340.                 break
341.
342.             elif x == self.__BYTE_NAK:
343.                 self.__print_all ('---read_string: NAK received\n')
344.                 break
345.
346.             #elif end-start > 0.1:
347.                 #print('---read_string: timeout exceeded\n')
```

```
348.                 #print(x)
349.                 #return None
350.
351.             else:
352.                 self.__print_all ( "---read_string: byte found:" , x )
353.
354.         return x
355.
356.     ##########################################################################
    ################################
357.
358.     def __send_string ( self , *data ):
359.
360.         """Creates a string from the data given and rounds numbers"""
361.
362.         new_data =''
363.
364.         if not data:
365.
366.             print ( '---send_string: Wrong input data\n' )
367.             return None
368.
369.         for var in data:
370.
371.             if isinstance ( var,numbers.Number ) == True:
372.                 var = round ( var , 2 )                     # If it's a numb
    er, it rounds it by 2 decimal digits
373.
374.             new_data += ' '+ str ( var )
375.
376.         new_data = new_data[1:]                             # Gets rid of th
    e first space character
377.
378.         self.__print_all( '---send_string:' , new_data , '\n' )
379.
380.         self.__debugger.write(new_data)
381.         self.__debugger.write("\n")
382.
383.         return new_data
384.
385.     ##########################################################################
    ################################
386.
387.     def __return_floats ( self , input_string ):
388.
389.         """Reads floats from a string and returns a list of those floats"""

390.
391.         self.__print_all ( "return_floats:" , input_string )
392.
393.         x = input_string.split ( ' ' )
394.
395.         for i in range ( 0 , len ( x ) ):
396.             x[i] = float ( x[i] )
397.
398.         return x
399.
400.     ##########################################################################
    ################################
401.
402.     def read_data ( self ):
403.
404.         """Reads data sent from KUKA or prints an error if data are not comp
    atible with ASCII"""
405.
406.         while True:
```

```
407.
408.            while True:
409.
410.                Number_of_Bytes = self.__bytes_waiting ( )
411.
412.                if Number_of_Bytes > 0:
413.
414.                    self.__print_all ( '---
    read_data: Number_of_Bytes =' , Number_of_Bytes , '\n' )
415.
416.                    x = self.robot_ser.read ( 1 )
417.
418.                    self.__print_all ( '---
    read_data: Bytes =' , x , '\n' )
419.
420.                    if x == self.__BYTE_STX:
421.                        self.__print_all ( '---
    read_data: STX received\n' )
422.
423.                        try:
424.                            self.robot_ser.write ( self.__BYTE_DLE )
425.                            self.__print_all ( '---
    read_data: DLE (1) sent\n' )
426.                            #time.sleep ( 0.001 )
427.                            break
428.
429.                        except serial.SerialTimeoutException:
430.                            print ( '---
    read_data: Write timeout exceeded (DLE)\n' )
431.                            return None
432.
433.                    elif x == self.__BYTE_NAK:
434.                        self.__print_all ( '---
    read_data: NAK (1) received\n' )
435.
436.            while True:
437.
438.                Number_of_Bytes = self.__bytes_waiting ( )
439.
440.                if Number_of_Bytes > 0:
441.
442.                    self.__print_all ( '---
    read_data: Number_of_Bytes =' , Number_of_Bytes , '\n' )
443.
444.                    x = self.__read_string ( )
445.
446.                    self.__print_all( '---read_data: Bytes =' , x , '\n' )
447.
448.                    if x == self.__BYTE_NAK:
449.                        self.__print_all ( '---
    read_data: NAK (2) received\n' )
450.                        break
451.
452.                    elif len ( x ) <= 3:
453.                        print ( '---
    read_data: Failed to receive a character from KUKA side (Number_of_Bytes < 3)
    \n' )
454.                        return None
455.
456.                    else:
457.                        try:
458.                            z = x.decode('ascii')
459.                            z = z[:len(z)-3]
460.
461.                            y = self.__form_3964r_block ( z )
462.
```

```
463.                             if y == x:
464.                                 self.__print_all( '---
    read_data: BCC OK\n' )
465.                                 self.__print_all( '---
    read_data: The data: [',z,'] received from KUKA\n' )
466.
467.                                 try:
468.                                     self.robot_ser.write ( self.__BYTE_DLE )

469.                                     self.__print_all ( '---
    read_data: DLE sent\n' )
470.                                     #time.sleep ( 0.001 )
471.                                     return z
472.
473.                                 except serial.SerialTimeoutException:
474.                                     print ( '---
    read_data: Write timeout exceeded (DLE)\n' )
475.                                     return None
476.
477.                             else:
478.                                 print ( '---read_data: BCC not OK\n' )
479.                                 try:
480.                                     self.robot_ser.write ( self.__BYTE_NAK )

481.                                     self.__print_all ( "---
    read_data: NAK sent")
482.                                     break
483.
484.                                 except serial.SerialTimeoutException:
485.                                     print ( '---
    read_data: Write timeout exceeded (NAK)\n' )
486.                                     return None
487.
488.                         except UnicodeError:
489.                             print ( '---
    read_data: Data not compatible with ASCII\n' )
490.                             return None
491.
492.                 else:
493.                     print ( '---
    read_data: Number_of_Bytes =' , Number_of_Bytes , '\n' )
494.
495.         return None
496.
497.     ##########################################################################
    ##################################
498.
499.     def send_data( self , *data ):
500.
501.         """Sends data to KUKA according to the 3964R protocol"""
502.
503.         my_data = self.__send_string ( *data )
504.
505.         while True:
506.
507.             try:
508.                 self.robot_ser.write ( self.__BYTE_STX )
509.                 self.__print_all ( '---send_data: STX sent\n' )
510.                 #time.sleep ( 0.001 )
511.             except serial.SerialTimeoutException:
512.                 print ( '---send_data: Write timeout exceeded (STX)\n' )
513.                 return None
514.
515.             while True:
516.
517.                 Number_of_Bytes = self.__bytes_waiting ( )
```

```
518.                 if Number_of_Bytes > 0:
519.                     self.__print_all ( '---
    send_data: Number_of_Bytes (1) =' , Number_of_Bytes , '\n' )
520.
521.                     x = self.robot_ser.read(1)
522.                     self.__print_all ( '---
    send_data: Bytes (1) =' , x , '\n' )
523.                     #w = self.robot_ser.read(1)
524.                     #self.__print_all ( '---
    send_data: Bytes (1) =' , w , '\n' )
525.
526.                     if x == self.__BYTE_DLE:
527.                         self.__print_all ( '---
    send_data: DLE (1) received\n' )
528.                         data_to_KUKA = self.__form_3964r_block ( my_data )
529.                         try:
530.                             self.robot_ser.write ( data_to_KUKA )
531.                             self.__print_all ( '---
    send_data: Data sent\n' )
532.                             #time.sleep(0.035)
533.                             while True:
534.
535.                                 Number_of_Bytes = self.__bytes_waiting( )
536.                                 if Number_of_Bytes > 0:
537.                                     self.__print_all ( '---
    send_data: Number_of_Bytes (2) =' , Number_of_Bytes , '\n' )
538.                                     x = self.robot_ser.read(1)
539.                                     self.__print_all ( '---
    send_data: Bytes (2) =' , x , '\n' )
540.
541.                                     if x == self.__BYTE_DLE:
542.                                         self.__print_all ( '---
    send_data: DLE (2) received\n' )
543.                                         return
544.                                     elif x == self.__BYTE_NAK:
545.                                         self.__print_all ( '---
    send_data: NAK (2) received\n' )
546.                                         break
547.
548.
549.                         except serial.SerialTimeoutException:
550.                             print ( '---
    send_data: Write timeout exceeded (data_to_KUKA)\n' )
551.                             return None
552.
553.                         break
554.
555.                     elif x == self.__BYTE_NAK:
556.                         self.__print_all ( '---
    send_data: NAK (1) received\n' )
557.                         break
558.
559.         return
560.
561.     ###############################################################################
    ###################################
562.
563.     def __handshake ( self ):
564.
565.         """Performs a handshake with KUKA robot"""
566.
567.         print ( '---Waiting for handshake...\n' )
568.         x = self.read_data ( )
569.
570.         if x == "111":
571.             self.send_data ( x )
```

```
572.             print ( '---Handshake Successful!!\n' )
573.             return 1
574.
575.         else:
576.             print ( '---Handshake Error\n' )
577.             return 0
578.
579.     ##########################################################################
    ##################################
580.
581.     def __initialize ( self ):
582.
583.         """Initializes the program by using the serial port and handshake fu
    nctions"""
584.
585.         print ( "\n---Initializing...\n" )
586.
587.         self.__debugger =  open ( self.__path_dbg, 'w' )
588.
589.         if self.__find_serial_ports ( ) == 0:        #calls serial_ports
590.             return 0
591.
592.         if self.__open_serial ( ) == 0:           #calls open_serial
593.             return 0
594.
595.         if self.__handshake ( ) == 0:           #calls handshake
596.             return 0
597.
598.         print ( "---Initializing complete!\n" )
599.
600.         return 1
601.
602.     ##########################################################################
    ##################################
603.
604.     def __terminate ( self ):
605.
606.         """Terminates the program and closes the serial port"""
607.
608.         print ( '---Waiting robot to reach home position...\n')
609.
610.         if ( self.read_data() == "OK" ):
611.
612.             print ( '---Robot in position!!\n')
613.             self.__close_serial( )           #calls close_serial
614.
615.             print ( "---Program ended successfully!\n" )
616.             self.__debugger.close()
617.
618.         return
619.
620.     ##########################################################################
    ##################################
621.
622.     def print_current_positional_values ( self ):
623.
624.         if self.__program_status == 0:
625.             print ( "---Program not started!!!\n")
626.             return 0
627.
628.         print( '---
    Current position:\n' , "\n" , 'A1 = ', self.__is_A1 , '°\n' , 'A2 = ', self._
    _is_A2, '°\n' , 'A3 = ', self.__is_A3, '°\n' , 'A4 = ', self.__is_A4, '°\n' ,
     'A5 = ', self.__is_A5, '°\n' , 'A6 = ', self.__is_A6, '°\n' , sep="" )
629.
```

```
630.    ##########################################################################
    ##################################
631.
632.    def set_velocity_and_acceleration ( self , vel = None , acc = None ):
633.
634.        if self.__program_status == 0:
635.            print ( "---Program not started!!!\n")
636.            return 0
637.
638.        if vel != None:                          #set velocity
639.
640.            self.__velocity = vel
641.
642.        if acc != None:                          #set acceleration
643.
644.            self.__acceleration = acc
645.
646.        self.send_data ( 'VAV' )                 # Velocity Acceleration Valu
    es
647.        self.send_data ( self.__velocity )
648.        self.send_data ( self.__acceleration )
649.
650.    ##########################################################################
    ##################################
651.
652.    def read_robot_position ( self ):
653.
654.        if self.__program_status == 0:
655.            print ( "---Program not started!!!\n")
656.            return 0
657.
658.        current_time = timer() - self.__start
659.        self.__counter += 1
660.
661.        self.send_data ( "Y" )      # Keep moving - Y as yes
662.
663.        x = self.read_data()
664.
665.        self.__print_all ( "---Position of Robot: " , x , "\n" )
666.
667.        x = self.__return_floats ( x )
668.
669.        self.__is_A1 = x[0]   # °
670.        self.__is_A2 = x[1]   # °
671.        self.__is_A3 = x[2]   # °
672.        self.__is_A4 = x[3]   # °
673.        self.__is_A5 = x[4]   # °
674.        self.__is_A6 = x[5]   # °
675.
676.        text =  "\n" + str(self.__counter) + ")\n"
677.        text += 'IS:  ' + "{:.2f}".format(self.__is_A1) + '°\t' + "{:.2f}".f
    ormat(self.__is_A2) + '°\t\t' + "{:.2f}".format(self.__is_A3)
678.        text +=  '°\t\t' + "{:.2f}".format(self.__is_A4) + "°\t\t" + "{:.2f}
    ".format(self.__is_A5) + '°\t\t' + "{:.2f}".format(self.__is_A6) + '°\t'
679.        self.__log_file.write(text)
680.
681.        text = "{:.3f}".format(current_time) +" s\n"
682.        self.__log_file.write(text)
683.
684.        return x
685.
686.    ##########################################################################
    ##################################
687.
688.    def move_to ( self , f1 = None , f2 = None , f3 = None , f4 = None , f5
    = None , f6 = None ):
```

```
689.
690.        if self.__program_status == 0:
691.            print ( "---Program not started!!!\n")
692.            return 0
693.
694.        if f1 != None:
695.            self.__target_A1 = f1    # °
696.        if f2 != None:
697.            self.__target_A2 = f2    # °
698.        if f3 != None:
699.            self.__target_A3 = f3    # °
700.        if f4 != None:
701.            self.__target_A4 = f4    # °
702.        if f5 != None:
703.            self.__target_A5 = f5    # °
704.        if f6 != None:
705.            self.__target_A6 = f6    # °
706.
707.        text = 'TO:  ' + "{:.2f}".format(self.__target_A1) + '°\t' + "{:.2f}
    ".format(self.__target_A2) + '°\t\t' + "{:.2f}".format(self.__target_A3)
708.        text +=  '°\t\t' + "{:.2f}".format(self.__target_A4) + "°\t\t" + "{:
    .2f}".format(self.__target_A5) + '°\t\t' + "{:.2f}".format(self.__target_A6)
    + '°\t'
709.        self.__log_file.write(text)
710.
711.        #print('---Moving to:\n')
712.        #print( 'A1 = ', self.__target_A1 , '°\n' , 'A2 = ', self.__target_A
    2, '°\n' , 'A3 = ', self.__target_A3, '°\n' , 'A4 = ', self.__target_A4, '°\n
    ' , 'A5 = ', self.__target_A5, '°\n' , 'A6 = ', self.__target_A6, '°\n' , sep
    ="" )
713.
714.        self.send_data ( self.__target_A1 , self.__target_A2 , self.__target
    _A3 , self.__target_A4 , self.__target_A5 , self.__target_A6 )
715.
716.        current_time = timer() - self.__start
717.        text = "{:.3f}".format(current_time) +" s\n"
718.        self.__log_file.write(text)
719.
720.    ##########################################################################
    ##################################
721.
722.    def move_to_relative ( self , f1 = 0.0 , f2 = 0.0 , f3 = 0.0 , f4 = 0.0
    , f5 = 0.0 , f6 = 0.0 ):
723.
724.        if self.__program_status == 0:
725.            print ( "---Program not started!!!\n")
726.            return 0
727.
728.        self.__target_A1 += f1   # °
729.        self.__target_A2 += f2   # °
730.        self.__target_A3 += f3   # °
731.        self.__target_A4 += f4   # °
732.        self.__target_A5 += f5   # °
733.        self.__target_A6 += f6   # °
734.
735.        text = 'TO:  ' + "{:.2f}".format(self.__target_A1) + '°\t' + "{:.2f}
    ".format(self.__target_A2) + '°\t\t' + "{:.2f}".format(self.__target_A3)
736.        text +=  '°\t\t' + "{:.2f}".format(self.__target_A4) + "°\t\t" + "{:
    .2f}".format(self.__target_A5) + '°\t\t' + "{:.2f}".format(self.__target_A6)
    + '°\t'
737.        self.__log_file.write(text)
738.
739.        #print('---Moving to:\n')
740.        #print( 'A1 = ', self.__target__A1 , '°\n' , 'A2 = ', self.__target_
    A2, '°\n' , 'A3 = ', self.__target_A3, '°\n' , 'A4 = ', self.__target_A4, '°\
```

```
    n' , 'A5 = ', self.__target_A5, '°\n' , 'A6 = ', self.__target_A6, '°\n' , se
    p="" )
741.
742.         self.send_data ( self.__target_A1 , self.__target_A2 , self.__target
    _A3 , self.__target_A4 , self.__target_A5 , self.__target_A6 )
743.
744.         current_time = timer() - self.__start
745.         text = "{:.3f}".format(current_time) +" s\n"
746.         self.__log_file.write(text)
747.
748.     ##########################################################################
    ##################################
749.
750.     def start_moving ( self ):
751.
752.         if self.__program_status == 0:
753.             print ( "---Program not started!!!\n")
754.             return 0
755.
756.         self.send_data ( "PTP" )    # Get in movement loop - Point To Point
    movement
757.         self.__start = timer()
758.
759.     ##########################################################################
    ##################################
760.
761.     def stop_moving ( self ):
762.
763.         if self.__program_status == 0:
764.             print ( "---Program not started!!!\n")
765.             return 0
766.
767.         # Write mean time to log
768.
769.         self.__total_time = timer() - self.__start
770.         self.__mean_time = self.__total_time / self.__counter
771.         text = "\nTotal time: " + "{:.3f}".format(self.__total_time) + " s\n
    " + "Mean time: " + "{:.3f}".format(self.__mean_time) +" s\n"
772.         self.__log_file.write(text)
773.
774.         # Tell the robot to exit move-loop
775.
776.         self.send_data ( 'N' )     # N as no
777.
778.     ##########################################################################
    ##################################
779.
780.     def start_program ( self ):
781.
782.         self.__program_status = self.__initialize ( )
783.
784.         if self.__program_status == 0:
785.             return self.__program_status
786.
787.         self.__log_file =  open ( self.__path , 'w' )
788.
789.     ##########################################################################
    ##################################
790.
791.     def end_program ( self ):
792.
793.         if self.__program_status == 1:
794.
795.             self.send_data ( 'END' )   #end program
796.             self.__terminate ( )
797.             self.__log_file.close()
```

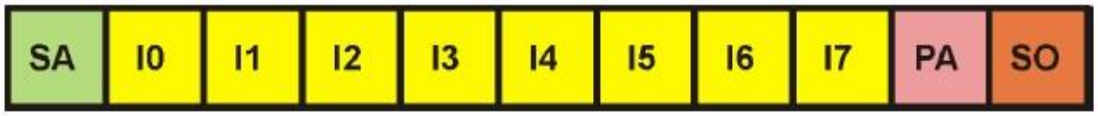
```
798.
799.        elif self.__program_status == 0:
800.            print ( "---Program not started!!!\n")
801.            return 0
802.
803.###############################################################################
    ############################
804.
805.
806.def sine_calcutalor( amplitude , frequency , now, dt = 0):
807.
808.    sine = "{:.2f}".format(0 - (amplitude * math.sin( 2 * math.pi * frequenc
    y * ( now + dt ) ) ) )
809.    sine = float ( sine )
810.    print("---
    -  " , sine , "° in " , "{:.3f}".format(now), "s\n", sep="" )
811.    return sine
812.
813.
814.my_robot = kuka()
815.
816.
817.my_robot.start_program()
818.#my_robot.enable_print()
819.my_robot.set_velocity_and_acceleration(7,14)
820.
821.my_robot.start_moving()
822.
823."""
824.start = timer()
825.k = 0
826.total_time = 0
827.while True:
828.
829.    k +=1
830.
831.    current_time = timer()-start
832.    x = sine_calcutalor ( 45 , 0.03 , current_time)
833.    #print(current_time)
834.
835.    my_robot.read_robot_position()
836.    my_robot.print_current_positional_values()
837.    my_robot.move_to ( f1 = x )
838.
839.    #print ("---timer is" , timer ( ) )
840.    my_time = "{:.3f}".format ( timer() - current_time - start )
841.    my_time = float ( my_time )
842.    print ( "---Time elapsed:" , my_time , "s\n")
843.
844.    total_time += my_time
845.    stop = timer()
846.    if (stop - start >= 30):
847.        break
848.print ( "---Mean time:" , round ( total_time / k , 3 ) , "s\n" )
849."""
850.
851.#'''
852.x = [ 0.0 , -90.0 , 90.0 , 0.0 , 0.0 , 0.0 ]
853.
854.a = 0.0
855.b = -90.0
856.c = 90.0
857.d = 0.0
858.e = 0.0
859.f = 0.0
860.
```

```
step = 1

while ( (x[0] < 90 ) or (x[1] > -
    120) or (x[2] > 60) or (x[3] < 90) or (x[4] < 90) or (x[5] < 90) ):

    x = my_robot.read_robot_position()
    #my_robot.print_current_positional_values()
    my_robot.move_to ( f1 = a , f2 = b , f3 = c , f4 = d , f5 = e , f6 = f )

    if (a < 90):
        a += step
    if (b > -120):
        b -= step
    if (c > 60):
        c -= step
    if (d < 90):
        d += step
    if (e < 90):
        e += step
    if (f < 90):
        f += step

'''
for k in range(1,15):
    my_robot.read_robot_position()
    my_robot.print_current_positional_values()
    my_robot.move_to ( f1 = i, f2 = j, f3 = l , f4 = i, f5 = i , f6 = i )
'''

my_robot.stop_moving()     #stop

my_robot.end_program()
```

5.3. Σειριακό Πρωτόκολλο 3964R

Το πρωτόκολλο 3694R είναι ένα ασύγχρονο σειριακό πρωτόκολλο, το οποίο ελέγχει την ροή δεδομένων μεταξύ δύο υπολογιστών και στην συγκεκριμένη περίπτωση μεταξύ του ελεγκτή KRC1 και του εξωτερικού υπολογιστή.

Η ακολουθία των bit κατά το πρωτόκολλο αυτό έχει ως εξής:



Definitions:
SA Start bit
I0...I7 Information bits
PA Parity bit
SO Stop bit

Οι χαρακτήρες ελέγχου του πρωτοκόλλου αυτού έχουν παρθεί από το DIN 66003 standard για κώδικα μήκους 7 bit. Η μετάδοση όμως χρησιμοποιεί 8 bit, με το bit 7 = 0.

Στο αρχείο serial.ini το οποίο βρίσκεται στην διεύθυνση C:\KRC\Roboter\INIT βρίσκονται όλες οι ρυθμίσεις που αφορούν την σειριακή θύρα COM1. Συγκεκριμένα:

```
[COM1]
BAUD=115200
CHAR_LEN=8   ; 7,8
STOP_BIT=1   ; 1,2  at time not changeable
PARITY=2     ; EVEN=2, ODD=1, NONE=0
PROC=1       ; 3964R=1, SRVT=2, WTC=3
```

Η μεταβλητή PROC ορίζει ποιο πρωτόκολλο θα χρησιμοποιηθεί στην συγκεκριμένη θύρα.

Επίσης μέσα στο ίδιο αρχείο υπάρχουν οι ρυθμίσεις για το πρωτόκολλο:

```
[3964R]
CHAR_TIMEOUT=500   ; msec
QUITT_TIMEOUT=500  ; msec
TRANS_TIMEOUT=2000 ; msec
```

MAX_TX_BUFFER=5 ; 1..5

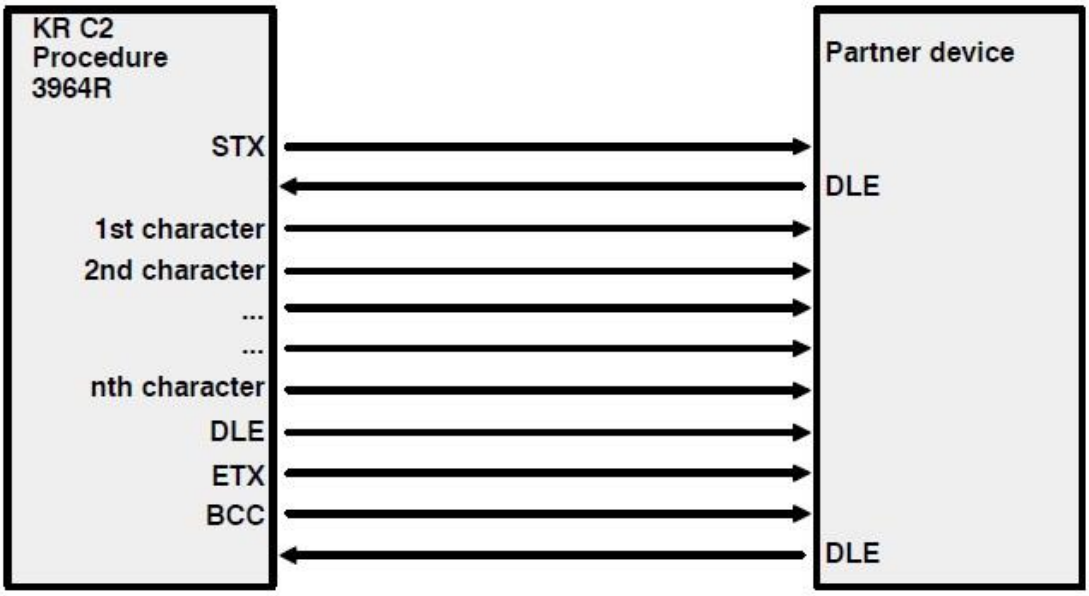
MAX_RX_BUFFER=20 ; 1..20

SIZE_RX_BUFFER=200 ; 1..2048

PROTOCOL_PRIOR=1 ; HIGH=1, LOW=0

Οι μεταβλητές του (CHAR_TIMEOUT, QUITT_TIMEOUT, TRANS_TIMEOUT, PROTOCOL_PRIOR) εξηγούνται παρακάτω.

Η λειτουργία του πρωτοκόλλου έχει ως εξής:



Για να στείλει η μία συσκευή στην άλλη ένα πακέτο δεδομένων, αρχικά στέλνει το byte STX (βλέπε ASCII). Αν η απέναντι συσκευή απαντήσει το byte DLE μέσα στον καθορισμένο χρόνο QUITT_TIMEOUT, τότε αρχίζει η αποστολή δεδομένων. Τα δεδομένα πρέπει να αποστέλλονται μέσα στον χρόνο CHAR_TIMEOUT.

Όταν ολοκληρωθεί το πακέτο δεδομένων, τότε αποστέλλονται 3 bytes ελέγχου, DLE, ETX και BCC. Το byte BCC, ή αλλιώς block check character, δημιουργείται από όλα τα bytes του πακέτου κάνοντας EXCLUSIVE OR σε αυτά.

Η απέναντι συσκευή υπολογίζει εκ νέου το BCC, το συγκρίνει και απαντά DLE αν όλα είναι σωστά.

Σε περίπτωση που υπάρχει κάποιο λάθος, αποστέλλεται το byte NAK και η διαδικασία ξεκινά από την αρχή. Σε αυτή τη περίπτωση ο αποστολέας του NAK περιμένει TRANS_TIMEOUT αλλιώς εμφανίζει μήνυμα σφάλματος.

Αν και οι δύο συσκευές αποστείλουν τον χαρακτήρα STX, τότε η συσκευή με την υψηλή προτεραιότητα (PROTOCOL_PRIOR) συνεχίζει να στέλνει, ενώ η άλλη ξεκινά να περιμένει δεδομένα.

6. Εναπομείναντα Θέματα

6.1. Βελτίωση Κώδικα

Επειδή η σειριακή επικοινωνία είναι αργή, της τάξης των 75 ms και με δεδομένο το γεγονός ότι τα καλύτερα χρονικά αποτελέσματα έγκεινται στα 100 ms περίπου, είναι προτιμότερο να χρησιμοποιηθούν οι έτοιμες εντολές κίνησης του ρομπότ LIN και CIRC για ευθεία και κυκλική κίνηση αντίστοιχα, με τους στόχους να βρίσκονται αρκετά μακριά μεταξύ τους.

Η αλλαγή αυτή μπορεί να τοποθετηθεί στην εντολή WHILE όπου επιλέγεται είτε ο τερματισμός του προγράμματος είτε η MOVEMENT_VALUES ( ) είτε η MOVE_TO ( ).

6.2. Τεχνικά Προβλήματα

Έχουν παρατηρηθεί δύο προβλήματα στο σύστημα της Kuka:

- Μερικές φορές εμφανίζονται στο χειριστήριο σφάλματα που αφορούν τον κινητήρα του τρίτου άξονα του ρομπότ (Motor 3) υποδηλώνοντας ότι στο μέλλον πιθανόν να χρειαστεί αντικατάσταση.

- Το σύστημα κάθε φορά που απενεργοποιείται αποθηκεύει σημαντικές πληροφορίες σε μία πλακέτα που βρίσκεται στην βάση του βραχίονα και ονομάζεται RDC. Αυτή η πλακέτα είναι πρώτης γενιάς και χρησιμοποιεί μνήμη flash, επομένως αναμένεται να γεμίσει και να μην μπορεί πλέον να αποθηκεύει δεδομένα, οπότε θα πρέπει μελλοντικά να αντικατασταθεί αν συμβεί αυτό.