ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ Μ.Κ. & Α.Ε.
Εργαστήριο Αυτομάτου Ελέγχου

Μεταπτυχιακή Εργασία

ΤΙΤΛΟΣ

Αμφίδρομη επικοινωνία και βαθμονόμηση αισθητήρων για ψηφιακό έλεγχο του τηλεχειριζόμενου ηλεκτροϋδραυλικού ρομπότ 10 β.ε. SARCOS.

Πέτρος Μ. Τσαμπάς

Επιβλέπων Καθηγητής: Ε. Γ. Παπαδόπουλος

ΑΘΗΝΑ 2010

Περίληψη

Σε αυτή τη μεταπτυχιακή εργασία έγινε εφικτή η αμφίδρομη επικοινωνία, ο έλεγχος λειτουργικότητας, η βαθμονόμηση και ο PID ψηφιακός έλεγχος του υλεκτροϋδραυλικού βραχίονα 10 Β.Ε. SARCOS, που διαθέτει το Εργαστήριο Αυτομάτου Ελέγχου του Τμήματος Μηχανολόγων Μηχανικών του Ε.Μ.Π.

Το συγκεκριμένο ρομποτικό σύστημα, δεν διαθέτει τον υπολογιστή υποστήριξης (VME) ούτε και το συνοδό πρόγραμμα (ARGUS) τα οποία να επιτρέπουν την αμφίδρομη επικοινωνία και τον έλεγχο.

Για τον σκοπό αυτό, σχεδιάστηκαν και κατασκευάστηκαν τα απαραίτητα ηλεκτρονικά κυκλώματα για την αποκατάσταση αμφίδρομης επικοινωνίας μεταξύ του ρομποτικού συστήματος SARCOS και ενός PC εξοπλισμένου με κάρτα NI-DIO96, ως υπολογιστή υποστήριξης.

Στην συνέχεια, υλοποιήθηκε εφαρμογή σε ANSI C η οποία επιτρέπει τον έλεγχο λειτουργικότητας των επενεργητών, την βαθμονόμηση των αισθητήρων θέσης και την εφαρμογή PID ψηφιακού ελέγχου, όπου η επιθυμητή θέση παρέχεται είτε χειροκίνητα μέσω ποτενσιομέτρων, είτε μέσω αρχείου.

Abstract

In this MSc thesis, bidirectional communication, functionality tests, calibration and PID digital control of the electro-hydraulic 10 DOF manipulator SARCOS, which is founded in Control System Laboratory of School Of Mechanical Engineering N.T.U.A., have been achieved.

The particular system came with no hosting computer (VME) or supporting software (ARGUS), both important for bidirectional communication and control.

For that purpose, necessary electronic circuits designed and manufactured, allowing the restoration of bidirectional communication between SARCOS and a PC, equipped with a NI-DIO96 digital card, as a hosting computer.

Software, in ANSI C, has been developed, allowing functionality tests for actuators, calibration of position sensors and digital PID control of the manipulator, with desired position signal acquired from either local potentiometers or a file.

Αφιερωμένο στους Γονείς μου

και την Νονά μου

Ευχαριστίες

Θέλω να ευχαριστήσω κατ αρχήν τον επιβλέποντα καθηγητή Ε. Γ. Παπαδόπουλο για την εμπιστοσύνη που έδειξε στο πρόσωπό μου, τόσο όσον αφορά στην ικανότητα μου να διαχειριστώ μια τόσο πολύπλοκη και σημαντική διάταξη, όσο και σε άλλα θέματα που άπτονται του εργαστηρίου. Επιπλέον για την σημαντική στήριξη καθ όλη την διάρκεια της εργασίας.

Τον Ιωάννη Νταβλιάκο, για την καθημερινές συζητήσεις και την συνεργασία μας.

Τον Ιωάννη Ροδίτη, προκάτοχο μου, που με εισήγαγε στον κόσμο του SARCOS.

Τον Ιωάννη Κοντολάτη, για τις πάντα ακριβόλογες και εύστοχες παρατηρήσεις του.

Τον Μιχάλη Μακροδημήτρη, για τις μακρές και γόνιμες συζητήσεις σε ψηφιακά ηλεκτρονικά και μικροεπεξεργαστές.

Όλη την ομάδα του εργαστηρίου για την αμέριστη βοήθεια της, την στήριξη της και την αποδοχή της. Ελπίζω, να επέστρεψα ένα τμήμα από την βοήθεια που μου προσφέρθηκε.

Τον ξάδελφο μου Στυλιανό Τσαμπά του Κωνσταντίνου, προπτυχιακό φοιτητή του Τμήματος Μηχανικών Η/Υ & Πληροφορικής του Πανεπιστημίου Πατρών που με βοήθησε πάρα πολύ σε θέματα Linux και ANSI C.

Περιεχόμενα

Περίληψη ... 3
Abstract ... 4
Ευχαριστίες ... 6
Περιεχόμενα ... 7
Κατάλογος Σχημάτων ... 10
Κατάλογος Πινάκων ... 12
Επεξηγήσεις Συντμήσεων και Ονομασιών ... 13
1 Εισαγωγή ... 14
1.1 Το ρομποτικό σύστημα SARCOS ... 15
1.2 Προηγούμενες Εργασίες ... 18
1.3 Επισκόπηση της παρούσας εργασίας. ... 19
2 Γενική Περιγραφή του ηλεκτρο-μηχανικού υλικού ... 20
2.1 Οι βραχίονες ... 20
2.2 Βαλβίδες ... 25
2.3 Αισθητήρες Θέσης ... 27
2.3.1 RVDT ... 27
2.3.2 Ποτενσιόμετρα ... 28
2.3.3 Παλμογεννήτριες (Optical Encoders) ... 28
2.4 Αισθητήρες Ροπής ... 29
2.5 Συγκεντρωτικές πληροφορίες ... 30
2.6 Παροχή ισχύος ... 30
3 Ανάστροφη Μηχανική και Αναλυτική Περιγραφή των Ηλεκτρονικών ... 31
3.1 Γενική Επισκόπηση - Σχηματικό Διάγραμμα. ... 31
3.2 Οι Κάρτες AJC ... 32
3.2.1 Γενική Περιγραφή Λειτουργίας και Σχηματικό Διάγραμμα Αναλογικού Βρόχου Ελέγχου. ... 33
3.2.2 Ο υπό-χωρος διευθύνσεων των καρτών AJC. ... 37
3.2.3 Επιπλέον επιλογές μέσω γεφυρώσεων ... 45
3.3 Η κάρτα BusRepv4 ... 49
3.3.1 Ο Πλήρης Χώρος Διευθύνσεων του SARCOS - Κλωβοί (CAGEs) ... 50
3.3.2 I/O Δίαυλος Greenspring μεταξύ SARCOS και Host Computer ... 51
3.3.3 Η PROM αποκωδικοποίησης διευθύνσεων και επιλογής AJC ... 54
3.3.4 Διευθύνσεις εντός της BusRepV4 ... 55
3.4 Ο Δίαυλος BackPanel - Το Κύκλωμα Cable Panel (DRINVRB3) ... 56
3.5 Κάρτα National Instruments PCI DIO-96 ... 57

3.5.1 Το ολοκληρωμένο IC8255 PIO 57
3.5.2 Οι Θύρες της Κάρτας National Instruments PCI DIO-96 60
3.5.3 Οι drivers - εντολές DAQmx για ANSI C 61
3.5.4 Πειράματα Αναγνώρισης μέγιστης ταχύτητα ανά τρόπο λειτουργίας 62
3.5.5 Υπολογιστής και Λειτουργικό Σύστημα 63
4. Υλικό (Hardware) που αναπτύχθηκε στα πλαίσια της παρούσας εργασίας 64
4.1 Η Κάρτα διεπαφής (interface) μεταξύ NI DIO-96 και SARCOS 64
4.1.1 Η ανάγκη κατασκευής της κάρτας. 64
4.1.2 Περιγραφή κάρτας και Σχηματικό block διάγραμμα. 65
4.1.3 Δομή των σημάτων - Ο συνολικός χώρος διευθύνσεων 66
4.2.Η κάρτα παρακολούθησης του διαύλου BackPlane (SNIFFER) 70
4.2.1.Η ανάγκη κατασκευής της κάρτας. 70
4.2.2.Διευθύνσεις εντός της κάρτας 71
5 Αλγόριθμοι για Βαθμονόμηση των Αισθητήρων 72
5.1 Μέθοδοι Βαθμονόμησης των Αισθητήρων 72
5.1.1 Μοντέλο κυκλώματος βαθμονόμησης 73
5.1.2 Μοντέλο Μετατροπέα από Αναλογικό σε Ψηφιακό (ADC). 73
5.1.3 Μέθοδος ενός καναλιού 74
5.1.4 Η εύρεση του γινομένου CAB 79
5.1.5 Μέθοδος δύο καναλιών 80
5.2 Μέθοδος Βαθμονόμησης του Αισθητήρα Θέσης 81
5.3 Μέθοδος Βαθμονόμησης του Αισθητήρα Ροπής 82
6 Κώδικας Για Επικοινωνία, Βαθμονόμηση και Έλεγχο 84
6.1 Εντολές DAQmx αρχικοποίησης της NI DIO-96 87
6.2 Αιτήσεις εισόδου/εξόδου - Δομή του Πίνακα των αιτήσεων ελέγχου 87
6.3 Διαδικασία αρχικοποίησης του SARCOS 88
6.4 Η υπορουτίνα queryTime() - Προετοιμασία Για Ανάγνωση χρόνου 89
6.5.Η Υπορουτίνα querySensors() - Προετοιμασία Για Λήψη Δεδομένων 90
6.6 Οι Ρουτίνες Ελέγχου 91
6.6.1 Η Ρουτίνα Διεπαφής με τον Χειριστή 91
6.6.2 Η ρουτίνα Επιλογής Επιθυμητής Εισόδου 93
6.5.3 Η ρουτίνα ControlDigPID() - Ψηφιακού Αλγόριθμου Έλεγχου 93
6.7 Η Ρουτίνα sendget(char action) - Αποστολή και Λήψης σημάτων 94
6.8 Υπορουτίνες υλοποίησης ψηφιακών φίλτρων 95
7 Πειράματα 96
7.1 Αρχικά Πειράματα 96

7.2 Πειράματα Χρονισμού 100
7.3 Βαθμονόμηση Αισθητήρων Θέσης 102
7.3.1 Υπολογισμός Γινομένου CAB 102
7.3.2 Βαθμονόμηση Με την Μέθοδο Μονού Καναλιού 104
7.4.Βαθμονόμηση αισθητήρων ροπής 107
7.4.1.Υπολογισμός γινομένου CAB 107
7.5.Έλεγχος από Ποτενσιόμετρα Μέσω P Ψηφιακού Βρόχου 108
8 Συμπεράσματα 113
8.1 Αποτίμηση αποτελεσμάτων 113
8.2 Επόμενα βήματα και προτάσεις 114
Βιβλιογραφία 116
ΠΑΡΑΡΤΗΜΑ 117
Π.1.Πίνακες 117
Π.2. Ανάστροφη Μηχανική και Ανάλυση Επιλεγμένων κυκλωμάτων 119
Π.2.1.Μετατροπή ψηφιακού σήματος στα DAC σε αναλογικό gain ή bias 119
Π.3.Προγραμματιζόμενη Λογική. 120
Π.3.1.CPLD ATF2500c 120
Π.3.3.Κώδικας interface card 125
Π.3.4.Κώδικας sniffer card 129
Π.4.Ο κώδικας MATLAB 131
Π.5.Ο κώδικας ελέγχου του ρομποτικού βραχίονα 133

Κατάλογος Σχημάτων

Σχήμα 1. 1 Τηλεχειριζόμενος Βραχίονας 10 Β.Ε. SARCOS. 16
Σχήμα 1- 2 τηλε-ρομποτικό χειρουργικό σύστημα daVinci. 16
Σχήμα 1- 3 ArgoMedical ReWalk. 17
Σχήμα 1- 4 CyBErdyne HAL. 17
Σχήμα 1- 5 Raytheon OX2. 17
Σχήμα 1- 6 NASA Robonaut2. 18
Σχήμα 1- 7 HONDA Body Weight Support Assist. 18

Σχήμα 2-1 Οι βραχίονες του slave και του master. 20
Σχήμα 2-2 Πρόσθια και πλάγια όψη του βραχίονα στην θέση ηρεμίας. 21
Σχήμα 2-3 Οι κύριοι Β.Ε. του SLAVE και το εύρος κίνησης τους. 22
Σχήμα 2-4 Επενεργητές και αισθητήρες του MASTER βραχίονα. 23
Σχήμα 2-5 Επενεργητές και αισθητήρες του SLAVE βραχίονα. 24
Σχήμα 2-6 Η βαλβίδα (D3000H/200) του 4^{ου} Β.Ε. εντός της κρύπτης. 25
Σχήμα 2- 7 Όψεις και τομή της βαλβίδας A3000H/50. 25
Σχήμα 2- 8 Όψεις και τομή της βαλβίδας D3000H/200. 26
Σχήμα 2-9 Χαρακτηριστική DC (conditioned) απόκρισης RVDT. 27
Σχήμα 2-10 Χαρακτηριστική ποτενσιόμετρου. 28
Σχήμα 2-11 Παλμογεννήτρια (α) και “interpolation box” 28
Σχήμα 2-12 Χαρακτηριστική Torsional Load Cell. 29

Σχήμα 3. 1 Εποπτικό Διάγραμμα Κυρίων Ηλεκτρονικών Υποσυστημάτων και Διαύλων του SARCOS. 31
Σχήμα 3-2 Κάρτες AJC1.1 και AJC2.1 πρόσθια όψη. 32
Σχήμα 3-3 Block διάγραμμα AJC1.1. 35
Σχήμα 3-4 Block διάγραμμα AJC2.1. 36
Σχήμα 3-5 Κάρτα AJC1.1 47
Σχήμα 3-6 Κάρτα AJC2.1 48
Σχήμα 3-7 BusRepV4 πρόσθια όψη. 49
Σχήμα 3-8 Ο κλωβός (cage) του slave. 50
Σχήμα 3-9 Κύκλος αποστολής πληροφορίας στις κάρτες. 52
Σχήμα 3-10 Κύκλος αποστολής δεδομένων στις κάρτες. 53
Σχήμα 3-11 Κυκλωματική αποκωδικοποίηση διευθύνσεων εντός του κλωβού. 54
Σχήμα 3-12 Δίαυλος BackPlane άνω και κάτω τμήμα. 56
Σχήμα 3-13 Cable Panel (DRINVRB3) και σύνδεση με τον host computer. 57
Σχήμα 3-14 Συνδεσμολογία Strobed Input - Strobed Output. 59
Σχήμα 3-15 Block Διάγραμμα της NI DIO-96. 60
Σχήμα 3-16 Οι connectors της NI DIO. 61
Σχήμα 3-17 Μέγιστη συχνότητα λειτουργίας strobed mode. 63

Σχήμα 4-1 Η κάρτα IFC (κάτω όψη). 64
Σχήμα 4-2 Η κάρτα IFC (άνω όψη). 65
Σχήμα 4-3 Block Διάγραμμα IFC. 66
Σχήμα 4-4 Σήματα χρονισμού παραγόμενα από την IFC. 67
Σχήμα 4-5 Ροή πληροφορίας κατά τον κύκλο αποστολής δεδομένων στο SARCOS 68
Σχήμα 4-6 Ροή πληροφορίας κατά τον κύκλο λήψης δεδομένων από το SARCOS. 69
Σχήμα 4-7 Κάρτα SNIFFER. 71

Σχήμα 5-1 Πρώτη βαθμίδα (στάδιο) βαθμονόμησης του σήματος. 72
Σχήμα 5-2 Απόκριση ADC μετατροπέα από αναλογικό σε ψηφιακό. 74
Σχήμα 5. 3 Σημαντικές θέσεις του 4ου Β.Ε. για βαθμονόμηση θέσης ή ροπής. 81

Σχήμα 6-1 Ροή πληροφορίας 85

Σχήμα 7-1 Σχέση τιμής DAC08 και CH07. 96
Σχήμα 7-2 Παράδειγμα ευστάθειας (1ου Β.Ε.) κατά την πρώτη απόπειρα. 97
Σχήμα 7-3 Μηδενικό σήμα από αισθητήρες. 98
Σχήμα 7-4 Αστάθεια κατά την κίνηση του 2ου Β.Ε. 99
Σχήμα 7-5 Βρόχος υστέρησης θετικής ανάδρασης για τον 2ο Β.Ε. 99
Σχήμα 7-6 Κατανομή χρόνων επιστροφής βρόχου χωρίς φίλτρο. 100
Σχήμα 7-7 Κατανομή χρόνων επιστροφής βρόχου με φίλτρο. 101
Σχήμα 7-8 Ψηφιακά σήματα DAC23 και CH01 στον χρόνο. 102
Σχήμα 7-9 Σήμα DAC23 σε σχέση με το σήμα CH01 (πλήρες εύρος). 102
Σχήμα 7-10 Σήμα DAC23 σε σχέση με το σήμα CH01 (γραμμικό τμήμα). 103
Σχήμα 7-11 Κίνηση Β.Ε. κατά την βαθμονόμηση αισθητήρα θέσης (1). 104
Σχήμα 7-12 Κίνηση Β.Ε. κατά την βαθμονόμηση αισθητήρα θέσης (1). 105
Σχήμα 7-13 Καμπύλη ρεύματος (DAC08) - ρεύματος (CH07) για τον 4ο Β.Ε. 106
Σχήμα 7-14 Καμπύλη τάσης - ρεύματος γι τον 4ο Β.Ε. 106
Σχήμα 7-15 Καμπύλη τάσης - ρεύματος για τον 2ο Β.Ε. 107
Σχήμα 7-16 (α έως θ) Κίνηση βραχίονα με P ψηφιακό έλεγχο από ποτενσιόμετρα. 112

Κατάλογος Πινάκων

Πίνακας 2-1 Λειτουργίες ανά Β.Ε. 21
Πίνακας 2-2 Χαρακτηριστικά βαλβίδων..... 25
Πίνακας 2-3 Χαρακτηριστικά RVDT. 27
Πίνακας 2-4 Συγκεντρωτικές πληροφορίες για τον master βραχίονα..... 30
Πίνακας 2-5 Συγκεντρωτικές πληροφορίες για τον slave βραχίονα. 30

Πίνακας 3-1 Διευθύνσεις AJC1.1 και AJC2.1. 37
Πίνακας 3-2 Είσοδοι απευθείας ελέγχου βαλβίδας. 40
Πίνακας 3-3 Είσοδοι επιθυμητών σημάτων αναλογικού βρόχου. 40
Πίνακας 3-4 Διευθύνσεις Παραμετροποίησης του Αναλογικού Βρόχου.. 40
Πίνακας 3-5 Διευθύνσεις σταδίων βαθμονόμησης. 41
Πίνακας 3-6 Διευθύνσεις χειρισμού AJC. 42
Πίνακας 3-7 Κανάλια (channels) AJC. 43
Πίνακας 3-8 Διευθύνσεις ανάγνωσης από encoder 44
Πίνακας 3-9 Γεφυρώσεις της AJC1.1 46
Πίνακας 3-10 Γεφυρώσεις της AJC2.1 46
Πίνακας 3-11 Κλωβοί και Διευθύνσεις. 50
Πίνακας 3-12 Διευθύνσεις εντός της BusRepV4. 55
Πίνακας 3-13 Τρόποι εισόδου/εξόδου (I/O modes) IC8255PIO. 58
Πίνακας 3-14 Εναλλακτικές αριθμήσεις Θυρών και γραμμών της NI DIO..... 60
Πίνακας 3-15 Συχνότητες μέγιστης λειτουργίας ανα τρόπο I/O. 62

Πίνακας 4-1 Αρχικοποίηση θυρών NI-DIO & IFC..... 66
Πίνακας 4-2 Ο χώρος διευθύνσεων μαζί με την IFC. 67
Πίνακας 4-3 Διευθύνσεις εντός της κάρτας SNIFFER. 71

Πίνακας 5-1 Σύμβολα του μοντέλου κυκλώματος βαθμονόμησης. 73
Πίνακας 5-2 Σύμβολα του μοντέλου ADC. 73

Πίνακας 6-1 Δομές που χρησιμοποιούνται στον βρόχο ελέγχου. 86
Πίνακας 6-2 Συναρτήσεις που υλοποιούν τον βρόχο ελέγχου. 86
Πίνακας 6. 3 Ειδικές Ερμηνείες Λέξεων Κλειδιών. 87
Πίνακας 6-4 Τρόποι λειτουργίας του ρομποτικού βραχίονα..... 91

Πίνακας 7-1 Συμπεριφορά των Β.Ε. πρώτη απόπειρα. 97
Πίνακας 7-2 Εντολές και Κύκλοι. 100
Πίνακας 7-3 Γινόμενο $\left(C \cdot A \cdot B\right)^{-1}$ για τα κυκλώματα βαθμονόμησης θέσης ..... (πρώτου σταδίου). 103
Πίνακας 7-4 Κέρδη και πολώσεις του πρώτου σταδίου βαθμονόμησης θέσης..... 107
Πίνακας 7. 5 Γινόμενο $\left(C \cdot A \cdot B\right)^{-1}$ για τα κυκλώματα βαθμονόμησης ροπής. 108

Επεξηγήσεις Συντμήσεων και Ονομασιών

AJC	Advanced Joint Controller. Κάρτα που εντός του Sarcos μια για κάθε BE υπεύθυνη για τον έλεγχο του.
ANSI C	Η γλώσσα προγραμματισμού C, κατά το πρότυπο ANSI
BackPlane (Upper and Lower)	Δίαυλος που μεταφέρει σήματα μεταξύ της BusRepV4 και των AJC του ίδιου cage
BE	Βαθμός Ελευθερίας.
BusRepV4	Bus Repeater Version 4. Κάρτα εντός του Sarcos μια για κάθε κλωβό (cage) υπεύθυνη για την αμφίδρομη διακίνηση σημάτων μεταξύ AJC και Host Computer
CAGE	Κλωβός. Το Σύνολο μιας κάρτας BusRepV4 και έως 20 καρτών AJC, που επικοινωνούν μέσω διαύλου BackPlane.
CPLD	Complex Programmable Logic Device. Ολοκληρωμένο Κύκλωμα προγραμματιζό-μενης λογικής
DAQmx	Σύνολο εντολών ANSCI C για τον έλεγχο της κάρτας NI-DIO
GreenSpring Bus	Ο Δίαυλος μεταξύ του υπολογιστή υποστήριξης (host computer), και του Sarcos.
Host Computer	Ο υπολογιστής υποστήριξης. Σε αυτόν τρέχει το λογισμικό ψηφιακού ελέγχου.
IFC (InterFace Card)	Κάρτα που σχεδιάστηκε στα πλαίσια της παρούσας εργασίας. Ενδιάμεση μεταξύ της NI-DIO και του διαύλου Greenspring. Παρέχει τα απαραίτητα I/O σήματα που απαιτούνται για αμφίδρομη επικοινωνία.
LINUX	Λειτουργικό σύστημα ανοιχτού πηγαίου κώδικά.
NI DIO - 96	National Instrument Digital Input Output. Κάρτα που τοποθετείται στον PCI δίαυλο του PC και παρέχει έως 96 ψηφιακές γραμμές εισόδου - εξόδου.
PID	Proportional - Integral - Differential. Είδος γραμμικού (αναλογικού ή ψηφιακού) ελέγχου.
RO	Read Only. Θέση μνήμης μόνο ανάγνωσης.
SNIFFER	Κάρτα που σχεδιάστηκε στα πλαίσια της παρούσας εργασίας. Παρέχει υποστήριξη σε θέματα χρονισμού και ελέγχου της λειτουργικότητας του ψηφιακών διαύλων.
WO	Write Only. Θέση μνήμης μόνο εγγραφής.
WR	Write Read. Θέση μνήμης ανάγνωσης και εγγραφής

1 Εισαγωγή

Η λέξη «ρομπότ» περιγράφει μια μεγάλη κατηγορία μηχανισμών οι οποίοι καλούνται, αυτόνομα ή ημιαυτόνομα, να αντικαταστήσουν ή να υποβοηθήσουν τον άνθρωπο σε επικίνδυνες ή επαναλαμβανόμενες εργασίες.

Οι βασικές απαιτήσεις από ένα ρομπότ, είναι η δυνατότητα αυτόνομης κίνησης και η δυνατότητα χειρισμού αντικειμένων, πάντα με τον επιθυμητό τρόπο και χωρίς να αποτελεί κίνδυνο για το άμεσο περιβάλλον του.

Έτσι αναδύεται η έννοια του αυτόματου ελέγχου, δηλαδή, του τρόπου με τον οποίο πληροφορίες από το περιβάλλον λαμβάνονται, μέσω αισθητήρων, από το ρομπότ και στην συνέχεια βάσει αυτών και σε συνδυασμό με τις απαιτήσεις της εντελλόμενης εργασίας, προκύπτει, η απόκριση του ρομπότ, μέσω επενεργητών.

Τα ρομπότ καλούνται να κινηθούν σε χώρους με πιθανά εμπόδια, στην στεριά ή σε υδάτινα περιβάλλοντα, σε χώρους διαμορφωμένους από ανθρώπους για ανθρώπους, ή ακόμα και στο διάστημα, ενώ κάποιες φορές τα ρομπότ είναι αναρτημένα πάνω στο ανθρώπινο σώμα. Καλούνται δε, να χειριστούν αντικείμενα φυσικά ή τεχνητά, συχνά πολύ μικρού ή πολύ μεγάλου μεγέθους.

Η ανάγκη για προσαρμογή των ρομπότ σε διαφορετικά περιβάλλοντα και εργασίες, έχει άμεση επίδραση στο σχήμα τους, στον τρόπο κίνησης τους, στο είδος των αισθητήρων τους και των επενεργητών τους. Υπάρχουν δυο διαφορετικές, αλλά αλληλοσυμπληρούμενες, τάσεις στον τρόπο που ο σχεδιασμός αντιμετωπίζει τις απαιτήσεις αυτές.

Η πρώτη τάση στον σχεδιασμό, είναι η κατασκευή ενός ρομπότ βάσει καθαρά των τεχνολογικών απαιτήσεων με συμβατικά σχήματα και τρόπους κίνησης. Τέτοιου είδους ρομπότ διαθέτουν ρόδες, έλικες ή αεριώθηση (jet) για την κίνηση τους, και γεωμετρικά σχήματα βελτιστοποιημένα για την μέγιστη δυνατή απόδοση.

Η δεύτερη τάση στον σχεδιασμό, είναι η υιοθέτηση τρόπων κίνησης και σχημάτων δοκιμασμένων και βελτιστοποιημένων, από την ίδια την φύση, μέσω της φυσικής επιλογής. Τέτοιου είδους ρομπότ διαθέτουν τεχνητά πτερύγια, χέρια, πόδια και μύες.

Συνήθως οι τάσεις αυτές συνυπάρχουν, προς όφελος της απόδοσης του ρομπότ. Ειδικά στην περίπτωση που τα ρομπότ καλούνται να κινηθούν σε περιβάλλοντα διαμορφωμένα για τον άνθρωπο και να χειριστούν αντικείμενα της ανθρώπινης καθημερινότητας, ο βιομιμητικός ή ανθρωπόμορφος σχεδιασμός είναι προτιμητέος.

1.1 Το ρομποτικό σύστημα SARCOS

Το SARCOS Dexterous Arm, είναι ένα master - slave τηλεχειριζόμενο, ηλεκτροϋδραυλικό ρομποτικό σύστημα, πρότυπο του οποίου είναι ο ανθρώπινος άνω κορμός. Σε πλήρη ανάπτυξη αποτελείται από δυο υποσυστήματα, το (εξωσκελετικό) master και το slave, κάθε ένα εκ των οποίων αποτελείται από κορμό (torso), κεφάλι (head), αριστερό χέρι (left arm) και δεξί χέρι (right arm). Συνολικά αντιστοιχούν 27 βαθμοί ελευθερίας (Β.Ε.) στον master και αντίστοιχα, 27 βαθμοί ελευθερίας στον slave.

Το Εργαστήριο Αυτομάτου Ελέγχου του Ε.Μ.Π. διαθέτει τον δεξιό, master - slave, βραχίονα, ο οποίος φαίνεται στο σχήμα 1-1, και πάνω στον οποίο έγινε και η παρούσα εργασία. Ο βραχίονας έχει 10 Β.Ε. που αντιστοιχούν σε ώμο (shoulder) 3 Β.Ε., αγκώνα (elbow) 1 Β.Ε., καρπό (wrist) 3 Β.Ε., αντίχειρα (thumb) 2 Β.Ε. και δάκτυλο (finger) 1 Β.Ε.

Το ρομποτικό σύστημα SARCOS, το οποίο φαίνεται στο Σχήμα 1-1, ανήκει στην κατηγορία των εξωσκελετικών, τηλεχειριζόμενων, ανθρωπόμορφων ρομπότ. Τα ρομπότ αυτής της κατηγορίας καλούνται να χειριστούν αντικείμενα, φυσικά ή τεχνητά, μιμούμενα τις κινήσεις και τις δεξιότητες του ανθρώπου. Ο έλεγχος του slave μπορεί να γίνει είτε μέσω του master σε γεωμετρική αντιστοιχία 1-1, είτε μέσω υπολογιστή.

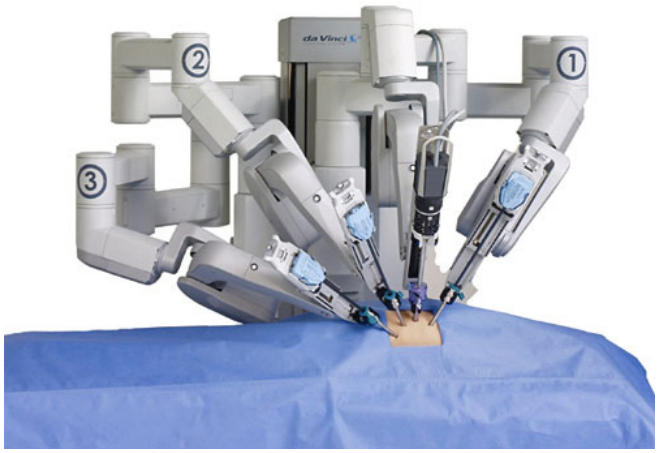
Στον master - slave έλεγχο μέσω τηλεχειρισμού, ο χειριστής φοράει το master εξωσκελετικό υποσύστημα, και μέσω ηλεκτρονικών η κίνηση του χειριστή αντιγράφεται από τον slave. Από τον master παρέχεται πληροφορία θέσης προς τον slave, ενώ από τον slave παρέχεται πληροφορία ροπής προς τον master. Με αυτό τον τρόπο ο χειριστής γνωρίζει, υπό κλίμακα, τις απαιτήσεις σε δύναμη της ζητούμενη κίνησης.

Ένα τέτοιο ρομποτικό σύστημα μπορεί να έχει εφαρμογές

- στην ιατρική
    - τηλε-ρομποτική χειρουργική
    - αποκατάσταση κίνησης μέλους μετά από τραυματισμό
- στη βαριά εργασία
    - πολλαπλασιαστής δύναμης
- στο πεδίο μάχης
    - πολλαπλασιαστής δύναμης
    - υποστήριξη πανοπλίας
- Τηλεχειρισμός εργασιών σε επικίνδυνα περιβάλλοντα
    - πυρηνικά
    - χημικά
    - εκρηκτικά
    - διάστημα

Ρομπότ στην ίδια κατηγορία με το SARCOS φαίνονται στις Σχήματα 1.2 έως 1.7.

Σχήμα 1. 1 Τηλεχειριζόμενος Βραχίονας 10 Β.Ε. SARCOS.

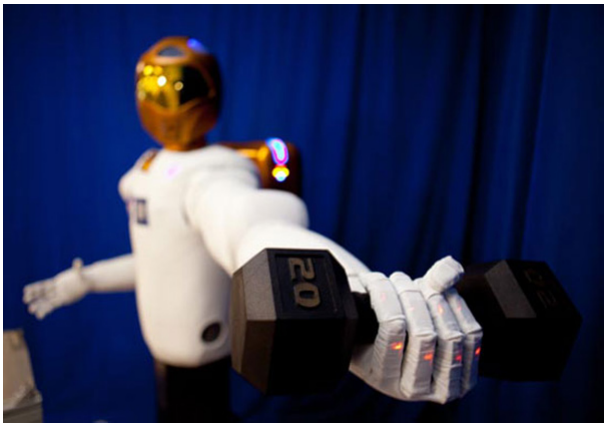


Σχήμα 1- 2 τηλε-ρομποτικό χειρουργικό σύστημα daVinci.

Σχήμα 1- 3 ArgoMedical ReWalk.

Σχήμα 1- 4 CyBErdyne HAL.

Σχήμα 1- 5 Raytheon OX2.



Σχήμα 1- 6 NASA Robonaut2.

Σχήμα 1- 7 HONDA Body Weight Support Assist.

1.2 Προηγούμενες Εργασίες

Το ρομποτικό σύστημα SARCOS έως το 2004 ανήκε στο McGill University του Καναδά. Υπό τον καθηγητή Ε. Γ. Παπαδόπουλου είχε γίνει έρευνα όσον αφορά την μοντελοποίηση των βαλβίδων [1], [2], [3].

Στην συνέχεια το ρομποτικό σύστημα SARCOS ήλθε στην κατοχή του Εργαστηρίου Αυτομάτου Ελέγχου, του Ε.Μ.Π. υπό την επίβλεψη του καθηγητή Ε. Γ. Παπαδόπουλου. Κατά την παραλαβή διαπιστώθηκε ότι ο υπολογιστής υποστήριξης δεν είχε έρθει μαζί με το υπόλοιπο σύστημα.

Για την αποκατάσταση της λειτουργικότητας του συστήματος έγιναν κατά χρονολογική σειρά οι κάτωθι εργασίες στα πλαίσια μεταπτυχιακών εργασιών του ΔΠΜΣ «Συστήματα Αυτοματισμού».

- 2006 Σ. Καπαγερίδης [4]
    - Προσαρμογή του συστήματος σε τάση 380V 50Hz από 575V 60Hz.
    - Σύνδεση αντλίας.
    - Προγραμματισμός PLC για αποκατάσταση χαμηλής πίεσης.
    - Εφαρμογή σε LabView αποστολής τιμών στα DAC.

- 2008 Ι. Ροδίτης [6]
    - Προγραμματισμός PLC για αποκατάσταση υψηλής πίεσης.
    - Ανάστροφη μηχανική ηλεκτρονικών.
    - Εφαρμογή ANSI C Linux για αποστολή τιμών στα DAC.

1.3 Επισκόπηση της παρούσας εργασίας.

Στην παρούσα εργασία επιτυγχάνονται τα εξής:

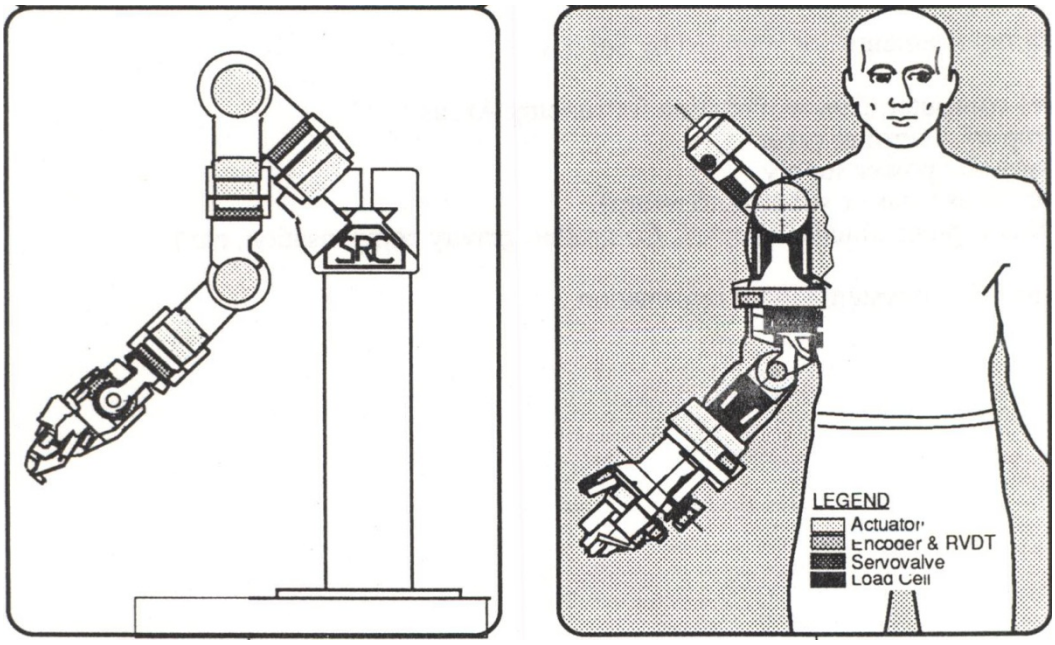
- Η αμφίδρομη επικοινωνία με το ρομποτικό σύστημα SARCOS
    - Για τον σκοπό αυτό αναλύθηκαν ως προς τον τρόπο λειτουργίας τους, τόσο τα ηλεκτρονικά του SARCOS (κάρτες AJC και BusReV4) όσο και η κάρτα NI-DIO
    - Στην συνέχεια κατασκευάστηκαν δυο ηλεκτρονικές κάρτες (IFC και SNIFFER) ως ενδιάμεση και συμπληρωματική, αντίστοιχα, για την αποκατάσταση της επικοινωνίας.
- Η ανάπτυξη μεθόδου βαθμονόμησης των αισθητήρων με χρήση μόνο των ψηφιακών εξόδων του ρομπότ, χωρίς την ανάγκη άμεσης μέτρησης τάσεων επί των ηλεκτρονικών του SARCOS.
- Η ανάπτυξη εφαρμογής (σε ANSI C και Linux) για τον ψηφιακό έλεγχο του ρομπότ με στόχο
    - την εκτέλεση ελέγχου λειτουργικότητας.
    - δοκιμαστικό PID έλεγχο.
    - βαθμονόμηση των αισθητήρων θέσης.
    - βαθμονόμηση των αισθητήρων ροπής (μόνο κώδικας, δεν ολοκληρώθηκε).
    - αναγνώριση παραμέτρων ανοικτού (μόνο κώδικας, δεν ολοκληρώθηκε).

2 Γενική Περιγραφή του ηλεκτρο-μηχανικού υλικού

Το SARCOS είναι ένα τηλεχειριζόμενο ρομποτικό σύστημα το οποίο αποτελείται από δυο κυρίως τμήματα, το τμήμα του master και το τμήμα του slave. Σε πλήρη ανάπτυξη απαρτίζεται από αριστερό και δεξί χέρι με 10 Β.Ε. έκαστο, κορμό με 4 Β.Ε. και κεφάλι με 3 Β.Ε., συνολικά 27 Β.Ε. για το master και 27 Β.Ε. για το slave.

2.1 Οι βραχίονες

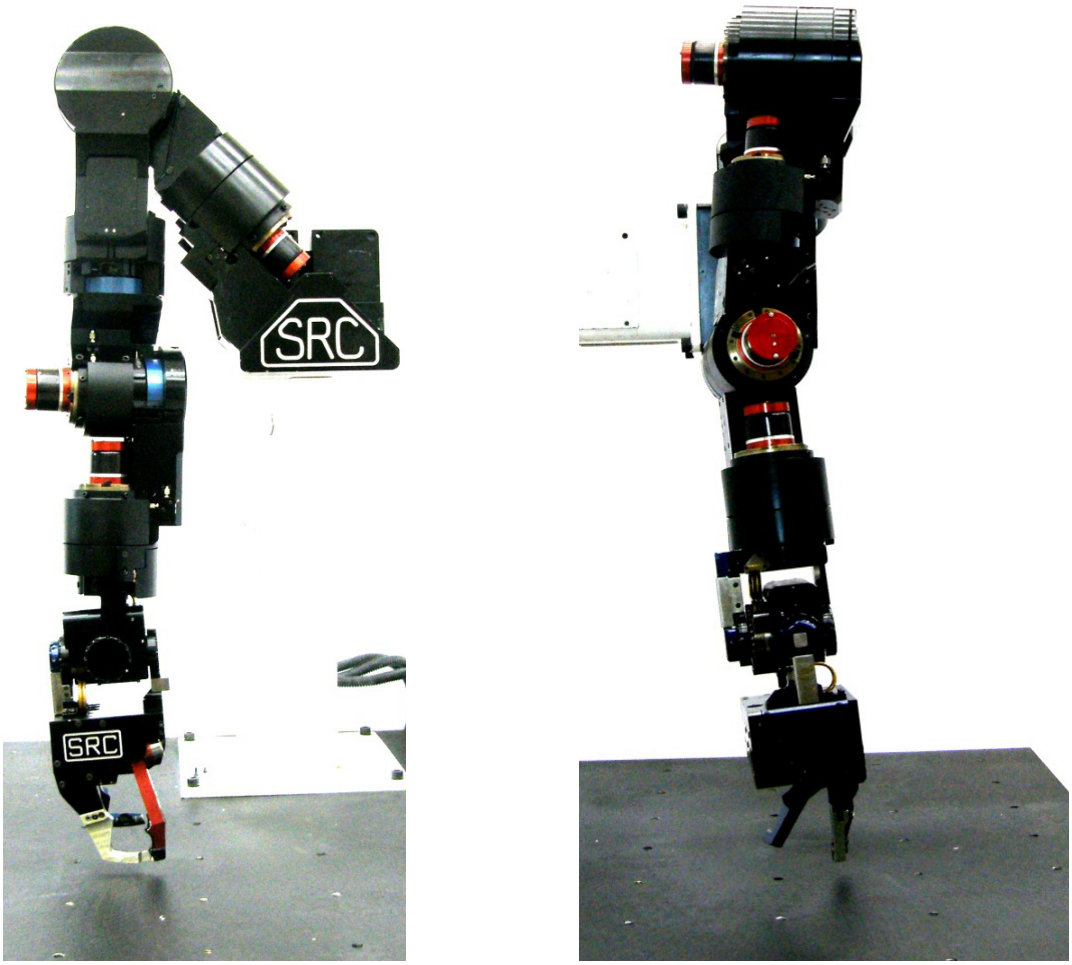
Οι βραχίονες, του master και του slave Σχήμα 2-1, είναι ανοιχτές κινηματικές αλυσίδες με γεωμετρική, **αλλά όχι μηχανική**, ισοδυναμία μεταξύ τους. Για κάθε δυνατή διευθέτηση του master υπάρχει η αντίστοιχη γεωμετρικά ισοδύναμη διευθέτηση του slave.



Σχήμα 2-1 Οι βραχίονες του slave και του master.

Όλες οι αρθρώσεις του slave είναι εφοδιασμένες με υδραυλικούς επενεργητές, περιστροφικούς κινητήρες με ειδικές σερβοβαλίδες, ικανούς να ασκήσουν υψηλές ροπές. Με αντίστοιχους επενεργητές, μικρότερης ισχύος, είναι εφοδιασμένες και οι αρθρώσεις του master για αντιστάθμιση του βάρους του βραχίονα έτσι ώστε να μην γίνεται αντιληπτό από τον χειριστή του.

Η συνολική διάταξη είναι σχεδιασμένη με τέτοιο τρόπο ώστε ο slave να μπορεί να αντιγράφει με ακρίβεια τις κινήσεις του master, λειτουργώντας ως ενισχυτής ροπής (δύναμης). Ταυτόχρονα, ο χειριστής του master αντιλαμβάνεται, υπό κλίμακα, τις απαιτήσεις σε ροπή που απαιτούνται για την κίνηση και μπορεί να έχει ίδια άποψη για την δυσκολία ή την ευκολία της ζητούμενης κίνησης.

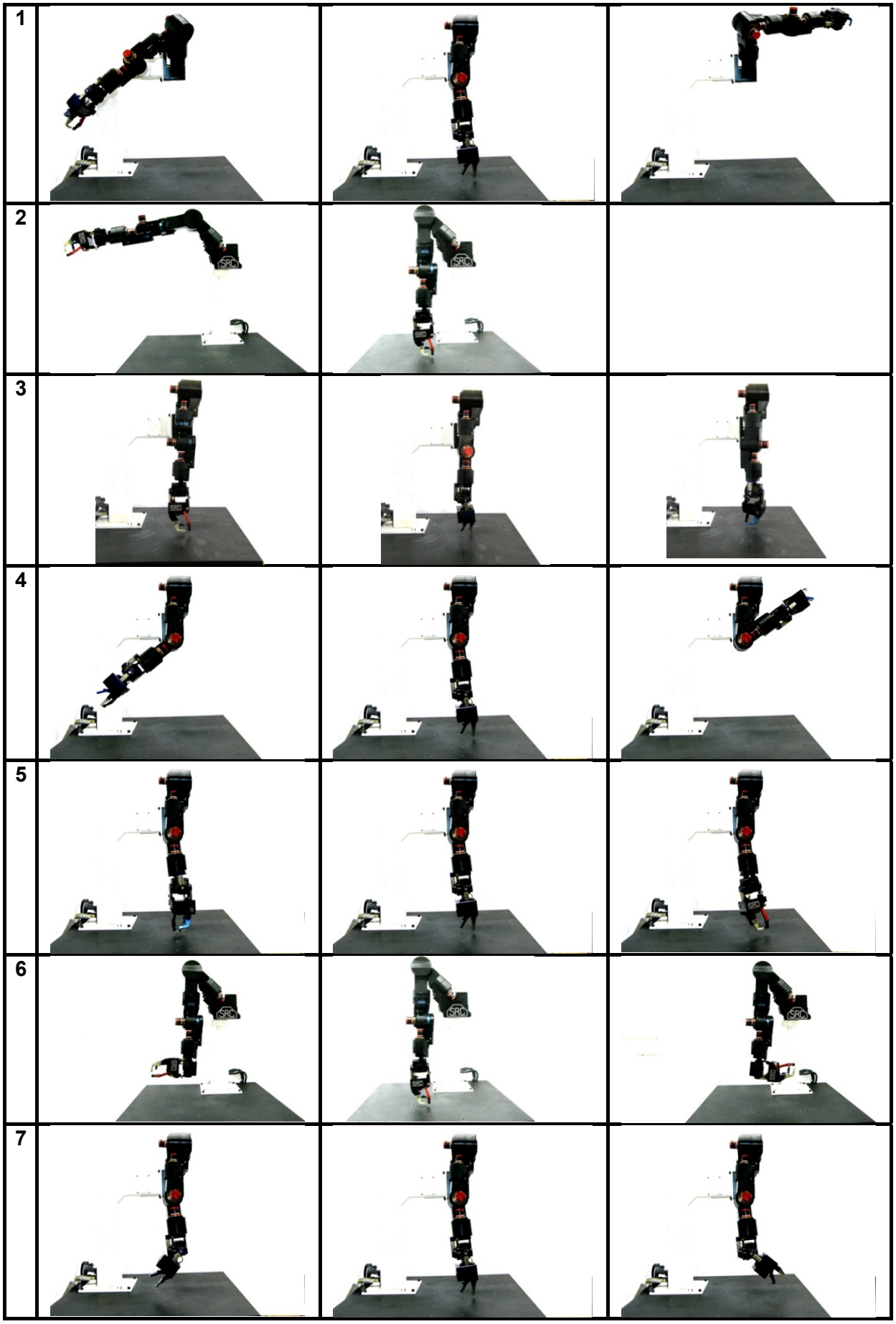


Σχήμα 2-2 Πρόσθια και πλάγια όψη του βραχίονα στην θέση ηρεμίας.

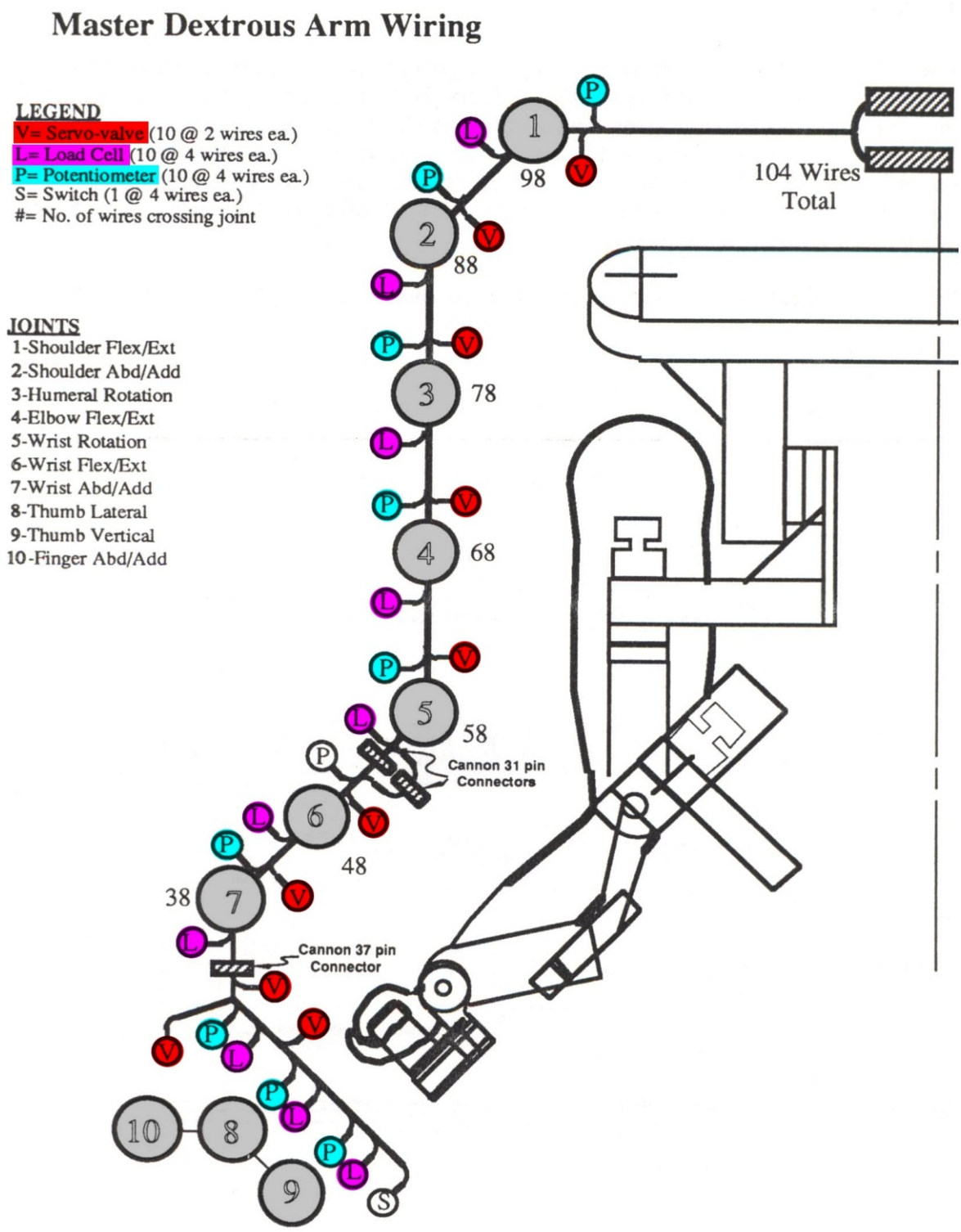
Οι λειτουργία που εκτελεί ο κάθε Β.Ε. περιγράφεται στον Πίνακα 2.1 Ακολουθεί μια διαδοχή φωτογραφιών, στο Σχήμα 2-3, που δείχνει το εύρος της κίνησης του slave βραχίονα για κάθε ένα Β.Ε., και δυο διαγράμματα, Σχήματα 2-4 και 2-5, με πληροφορίες για τους αισθητήρες και τους επενεργητές που διαθέτουν ο master και ο slave βραχίονες.

Πίνακας 2-1 Λειτουργίες ανά Β.Ε.

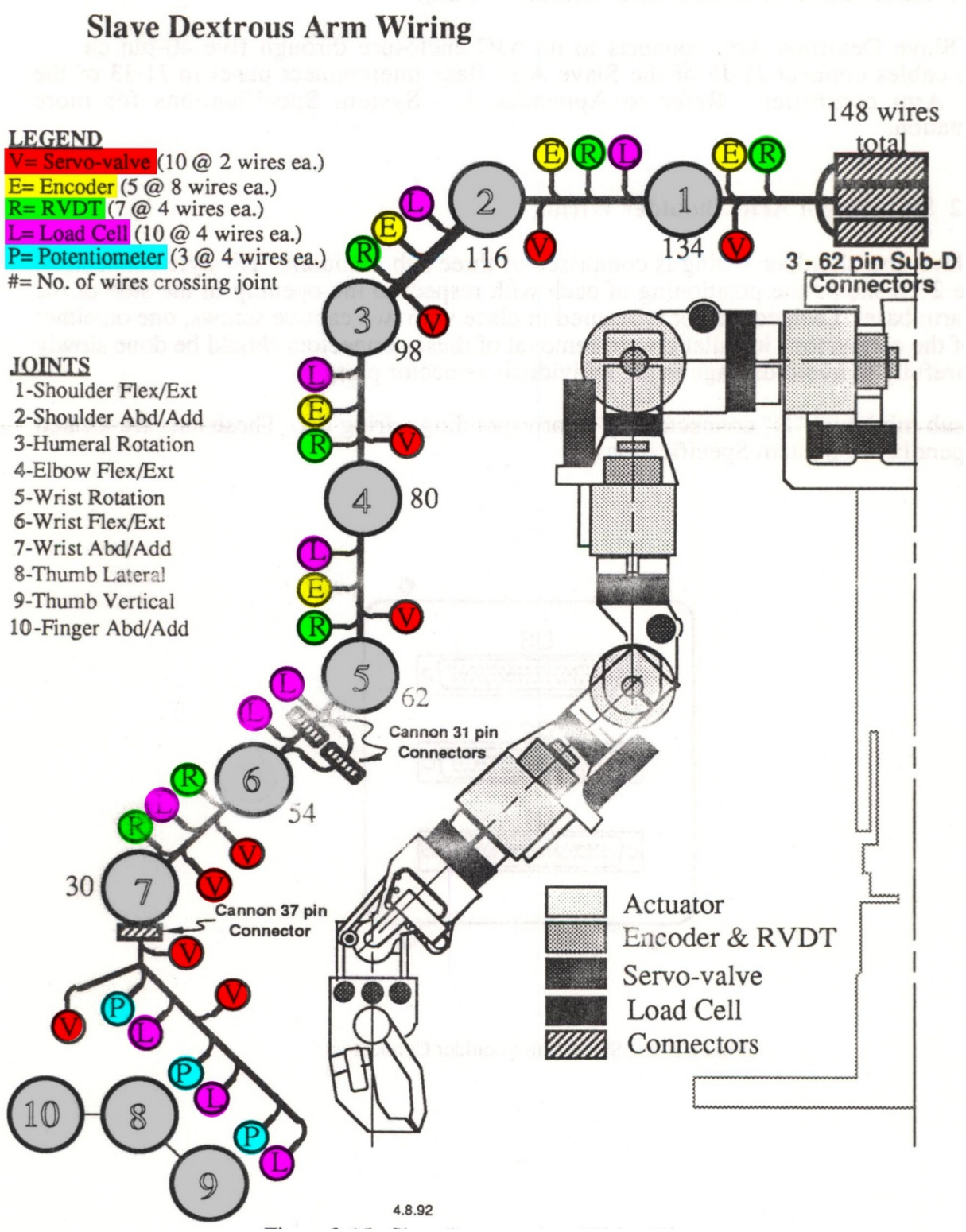
Β.Ε.	θέση	λειτουργία	position	function
1	Ώμος	Έκταση / Κάμψη	Shoulder	Flexion./ Extension
2	Ώμος	Απαγωγή / Προσαγωγή	Shoulder	Abduction / Adduction
3	Βραχίονας	Περιστροφή	Humeral	Rotate
4	Αγκώνας	Κάμψη	Elbow	Flexion
5	Καρπός	Περιστροφή	Wrist	Rotate
6	Καρπός	Έκταση / Κάμψη	Wrist	Flexion./ Extension
7	Καρπός	Απαγωγή / Προσαγωγή	Wrist	Abduction / Adduction
8	Αντίχειρας	Πλευρική κίνηση	Thumb	Lateral
9	Αντίχειρας	Κάθετη	Thumb	Vertical
10	Δάκτυλο	Απαγωγή / Προσαγωγή	Finger	Abduction / Adduction



Σχήμα 2-3 Οι κύριοι Β.Ε. του SLAVE και το εύρος κίνησης τους.



Σχήμα 2-4 Επενεργητές και αισθητήρες του MASTER βραχίονα.



Σχήμα 2-5 Επενεργητές και αισθητήρες του SLAVE βραχίονα.

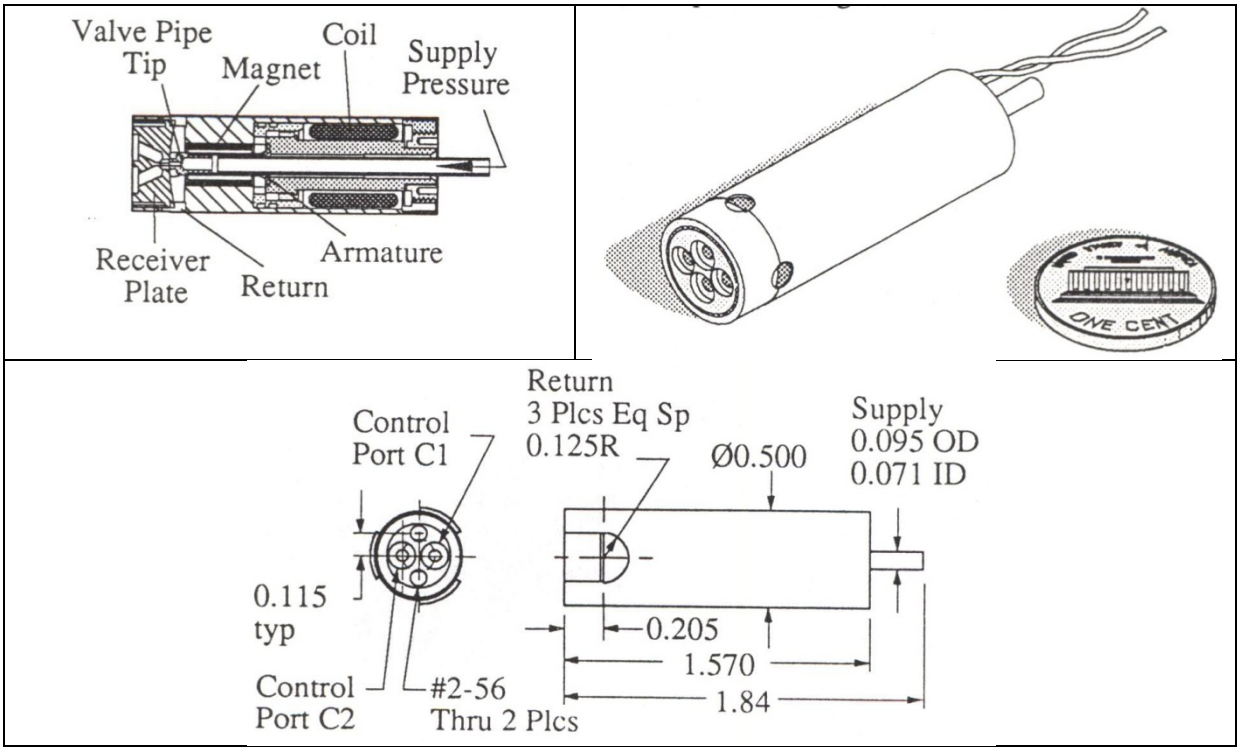
2.2 Βαλβίδες

Οι επενεργητές είναι υδραυλικές βαλβίδες λαδιού τύπου jet pipe, Οι βαλβίδες αυτές, δεν κυκλοφορούν στο εμπόριο, είναι σχεδιασμένες και κατασκευασμένες ειδικά για το SARCOS.

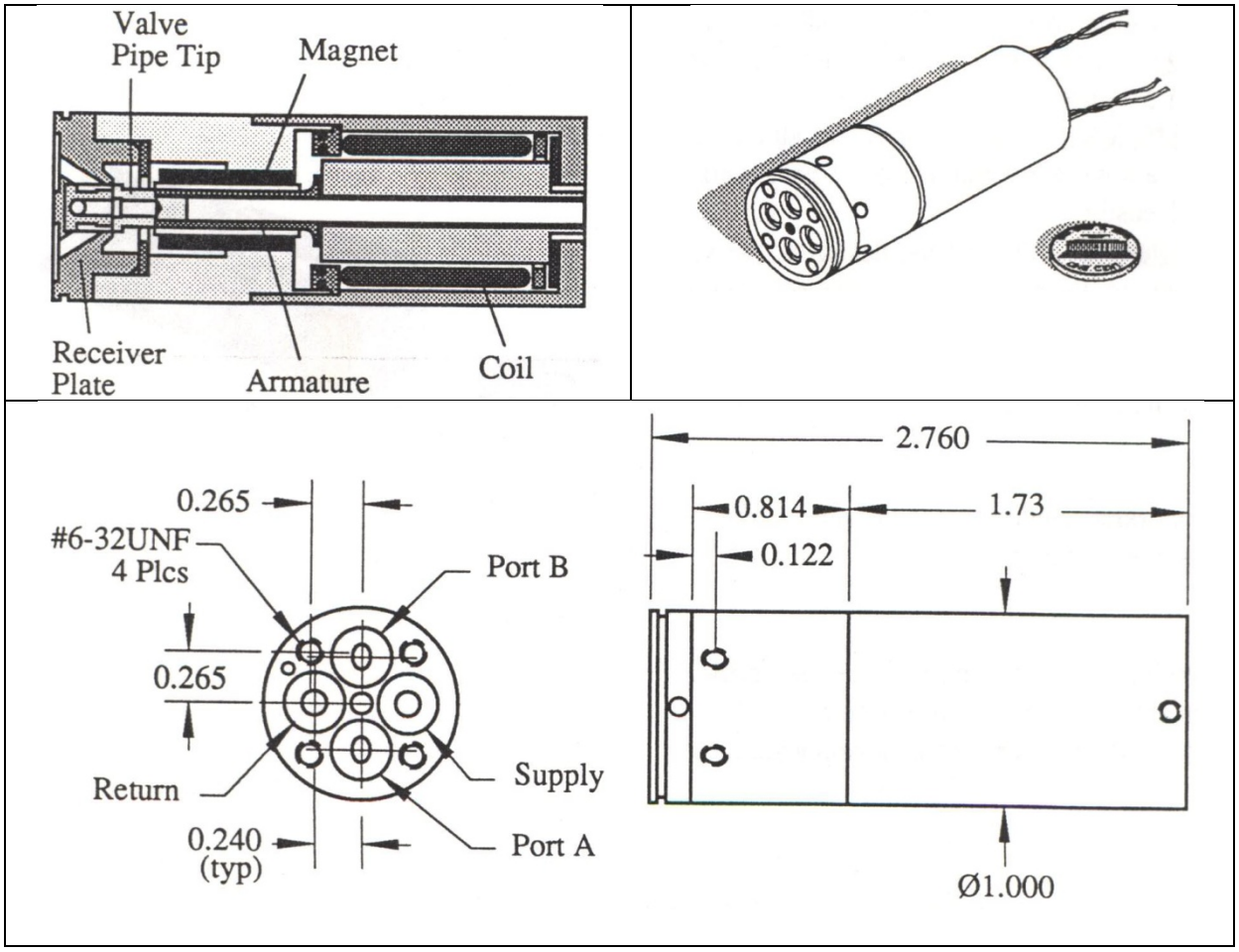
Πίνακας 2-2 Χαρακτηριστικά βαλβίδων

μέγιστη πίεση	204atm	3000psi
μέγιστη ροή		
A30000H/50	1.7lt/min	0.5 gpm
D3000H/200	7.4lt/min	2.0 gpm
μέγιστο ρεύμα οδήγησης	+/- 1A	
DC αντίσταση	8ohm	

Σχήμα 2-6 Η βαλβίδα (D3000H/200) του 4^{ου} Β.Ε. εντός της κρύπτης.



Σχήμα 2- 7 Όψεις και τομή της βαλβίδας A3000H/50.



Σχήμα 2- 8 Όψεις και τομή της βαλβίδας D3000H/200.

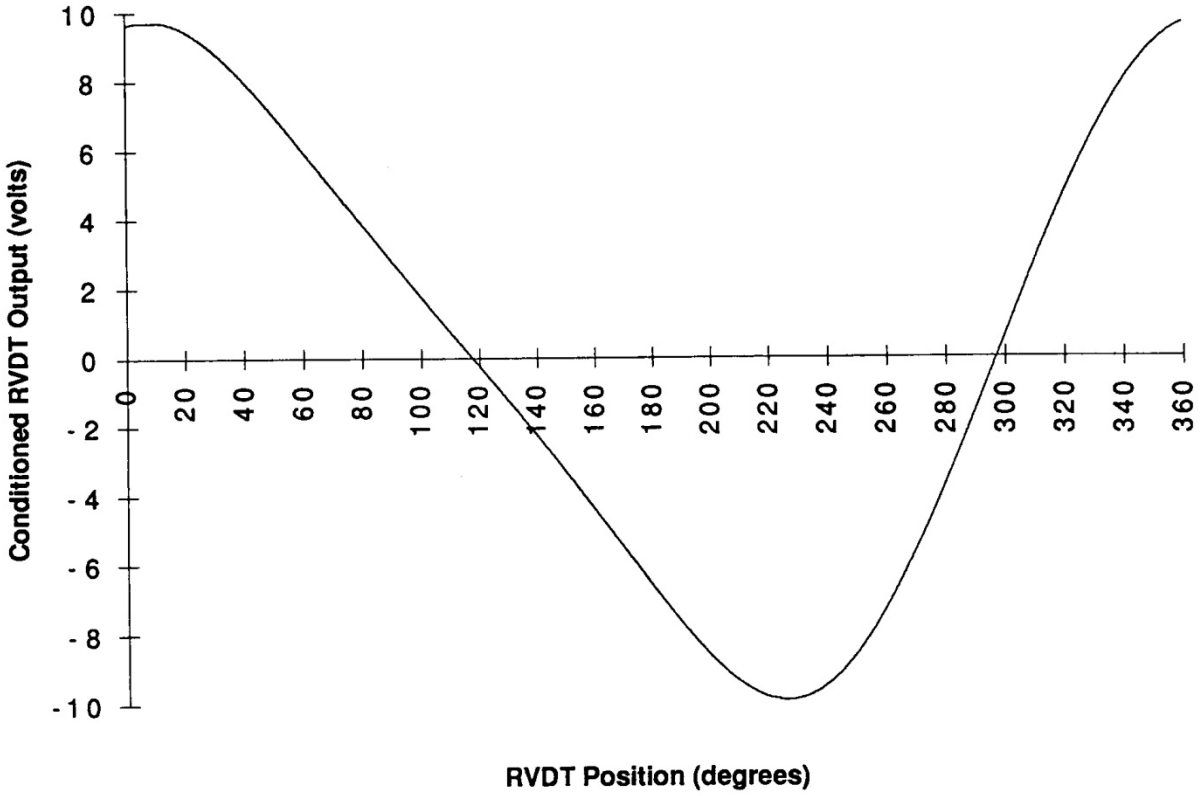
Όσο μεγαλύτερο και, κατά σύμβαση, θετικό το ρεύμα τόσο περισσότερη ροή κατευθύνεται προς την πόρτα Α και λιγότερη προς την πόρτα Β, το αντίθετο ακριβώς συμβαίνει εάν το ρεύμα είναι αρνητικό.

Συγκεντρωτικές πληροφορίες για όλους τους Β.Ε. παρατίθενται στο τέλος του κεφαλαίου.

2.3 Αισθητήρες Θέσης

2.3.1 RVDT

Η χαρακτηριστική των RVDT (Rotational Variable Differential Transducer) φαίνεται στο Σχήμα 2-9 . Τα RVDT διεγείρονται από AC τάση, και στην έξοδο παρέχουν DC τάση. Εκ κατασκευής υπάρχουν δύο εύρη λειτουργίας, 180° και 90° . Με αυτό τον τρόπο, η ίδια συσκευή μπορεί να χρησιμοποιηθεί με δυο διαφορετικές αναλύσεις. Τα χαρακτηριστικά φαίνονται στον Πίνακα 2-3



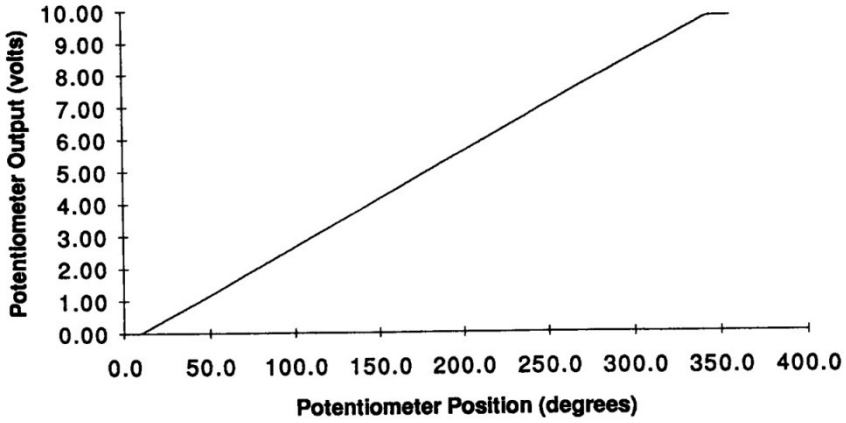
Σχήμα 2-9 Χαρακτηριστική DC (conditioned) απόκρισης RVDT.

Πίνακας 2-3 Χαρακτηριστικά RVDT.

πλάτος AC τάση εισόδου (διέγερσης)	5V	
ρεύμα AC εισόδου (διέγερσης)	20mA	
πλάτος AC τάση εξόδου (απόκρισης)	4.25Vrms +/- 0.05V	
Εύρος DC τάσης εξόδου	(-10 …10) V	
γωνιακό εύρος λειτουργίας	90°	180°
ανάλυση	80 mVrms /^{o}	44mVrms/^{o}

2.3.2 Ποτενσιόμετρα

Τα ποτενσιόμετρα διεγείρονται από DC τάση 10 V και έχουν εύρος λειτουργίας 330°

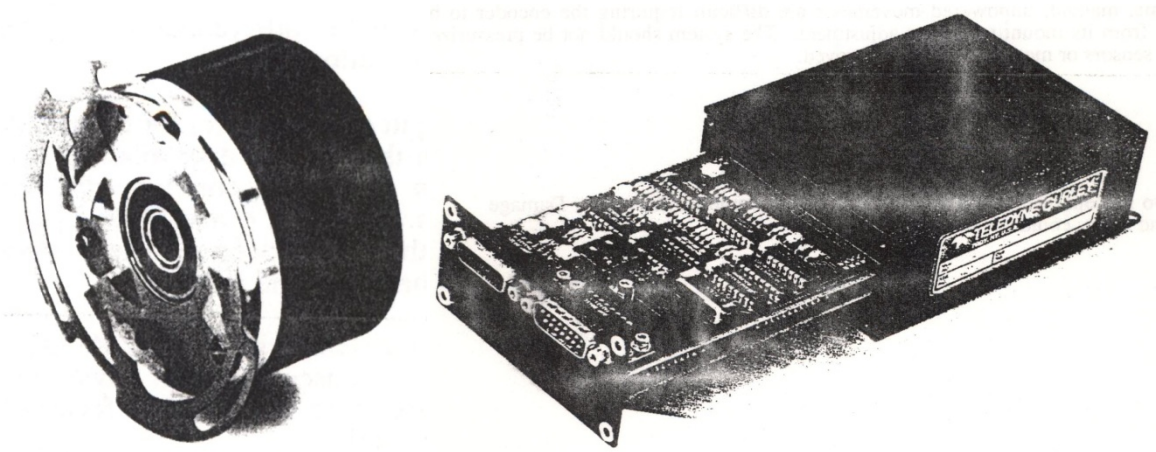


Σχήμα 2-10 Χαρακτηριστική ποτενσιόμετρου.

2.3.3 Παλμογεννήτριες (Optical Encoders)

Υπάρχει η δυνατότητα ανάγνωσης τις θέσεις από παλμογεννήτρια Σχήμα 2-11 (α). Στην υπάρχουσα διάταξη, με παλμογεννήτριες είναι εξοπλισμένοι οι Β.Ε. 1 έως 5.

Οι παλμογεννήτριες αυτές είναι υβριδικού τύπου (αναφέρονται ως αναλογικές στο manual) και παρέχουν τα σήμα με ανάλυση 5000 παλμών ανά περιστροφή. Στην συνέχεια, το σήμα διέρχεται από ένα κύκλωμα με την ονομασία “interpolation box”, Σχήμα 2-11 (β), και αποκτά ανάλυση 200000 παλμών ανά περιστροφή.

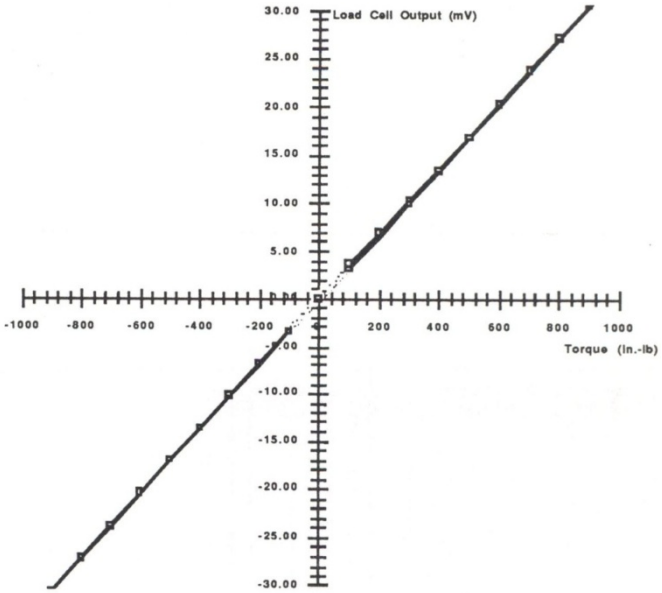


(α) (β)

Σχήμα 2-11 Παλμογεννήτρια (α) και “interpolation box”

2.4 Αισθητήρες Ροπής

Οι αισθητήρες ροπής είναι Load Cells τύπου Tortional, Beam ή Remote Center και διεγείρονται από DC γέφυρα Wheatstone. Οι χαρακτηριστικές όλων των load Cell,όπως αυτές μετρηθήκαν κατά την συναρμολόγηση του SARCOS βρίσκονται στο παράρτημα APPEX I του manual του SARCOS.



Σχήμα 2-12 Χαρακτηριστική Torsional Load Cell.

2.5 Συγκεντρωτικές πληροφορίες

Στους Πίνακες 2-4 και 2-5 βρίσκονται συγκεντρωτικές πληροφορίες για τους βραχίονες των master και slave. Αναλυτικότερες πληροφορίες μπορούν να βρεθούν στα παράρτημα APPEX I του εγχειρίδιου λειτουργίας (manual) του SARCOS.

Πίνακας 2-4 Συγκεντρωτικές πληροφορίες για τον master βραχίονα.

Β.Ε.	εύρος κίνησης	επενεργητής		αισθητήρες θέσης		αισθητήρας ροπής	
		κίνηση επενεργητή	ροή lt/min	αναλογικός αισθητήρας	encoder ανάλυση	είδος	εύρος (Nm)
1	110.4	γραμμική	7.4	POT	-	Torsional	109.2
2	87.5	γραμμική	7.4	POT	-	Torsional	109.2
3	177.9	στροφική	1.7	POT	-	Torsional	55.8
4	104.8	γραμμική	1.7	POT	-	Torsional	55.8
5	177.7	στροφική	1.7	POT	-	Torsional	55.8
6	98.2	γραμμική	1.7	POT	-	Torsional	24.8
7	84.4	γραμμική	1.7	POT	-	Torsional	24.8
8	54.2	γραμμική	1.7	POT	-	Torsional	8.0
9	80.5	γραμμική	1.7	POT	-	Beam	8.0
10	34.6	γραμμική	1.7	POT	-	Beam	8.0

Πίνακας 2-5 Συγκεντρωτικές πληροφορίες για τον slave βραχίονα.

Β.Ε.	εύρος κίνησης	επενεργητής		αισθητήρες θέσης		αισθητήρας ροπής	
		κίνηση επενεργητή	ροή lt/min	αναλογικός αισθητήρας	encoder ανάλυση	είδος	εύρος (Nm)
1	180	στροφική	7.4	RVDT	200000	Torsional	244.5
2	180	στροφική	7.4	RVDT	200000	Torsional	244.5
3	180	στροφική	7.4	RVDT	200000	Torsional	109.2
4	180	στροφική	7.4	RVDT	200000	Torsional	109.2
5	180	στροφική	7.4	RVDT	200000	Torsional	109.2
6	180	στροφική	7.4	RVDT	-	Beam	31.0
7	90	στροφική	7.4	RVDT	-	Beam	31.0
8	106	στροφική	1.7	TLPOT	-	Remote Center	24.8
9	80	γραμμική	1.7	TVPOT	-	Remote Center	24.8
10	30	γραμμική	1.7	FPOT	-	Remote Center	24.8

2.6 Παροχή ισχύος

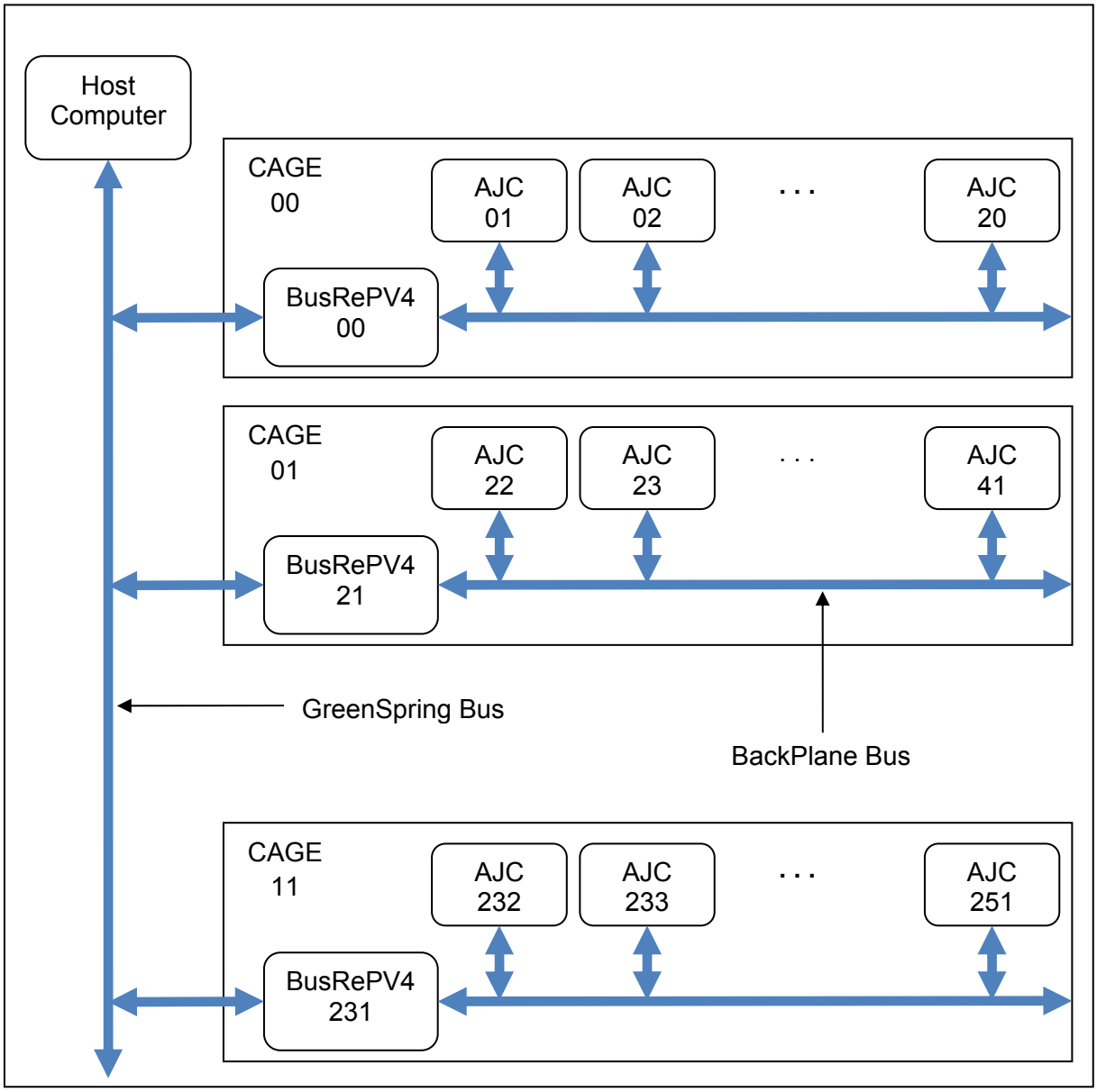
Η απαραίτητη ισχύς για την λειτουργία του συστήματος παρέχεται από μια αντλία PVP-4130-2-R-2-6A2 της εταιρίας Parker. Η αντλία χαρακτηρίζεται ως εμβολοφόρος, μεταβλητού όγκου εμβολισμού και δυνατότητα ελέγχου αποδιδόμενης πίεσης και ισχύος. Η μέγιστη παροχή της αντλίας είναι 41cm^3/rev και μπορεί να αποδώσει πιέσεις 250 - 3000psi.

3 Ανάστροφη Μηχανική και Αναλυτική Περιγραφή των Ηλεκτρονικών

3.1 Γενική Επισκόπηση - Σχηματικό Διάγραμμα.

Σε αυτό το κεφάλαιο γίνεται μια αναλυτική περιγραφή υπαρχόντων των ηλεκτρονικών του SARCOS καθώς και του τρόπου ροής της πληροφορίας, αναλογική και ψηφιακής, όπως φαίνεται και στο εποπτικό σχηματικό διάγραμμα του Σχήματος 3-1

Σε κάθε Β.Ε. αντιστοιχεί μια κάρτα AJC η οποία λαμβάνει πληροφορία από τους αισθητήρες και ελέγχει τον επενεργητή, στέλνοντας ηλεκτρική ισχύ στην βαλβίδα του. Ανά είκοσι, οι κάρτες AJC, ομαδοποιούνται σε ένα κλωβό (CAGE) ο οποίος ελέγχεται από μια κάρτα BusRepV4. Όλοι οι κλωβοί ελέγχονται από τον υπολογιστή υποστήριξης (Host Computer).



Σχήμα 3. 1 Εποπτικό Διάγραμμα Κυρίων Ηλεκτρονικών Υποσυστημάτων και Διαύλων του SARCOS.

3.2 Οι Κάρτες AJC

Οι κάρτες παρέχουν αλληλεπίδραση του χειριστή με κάθε Β.Ε. μέσω ενδεικτικών leds, ενός διακόπτη και ενός ποτενσιόμετρου στην πρόσοψη του, όπως φαίνεται στο Σχήμα 3-2.

AJC1.1		AJC2.1
τηλεχειρισμός / τοπικός έλεγχος		τηλεχειρισμός / τοπικός έλεγχος
επιθυμητή θέση		επιθυμητή θέση
επιθυμητή ροπή		επιθυμητή ροπή
αντιστάθμιση βαρύτητας		αντιστάθμιση βαρύτητας
πόλωση (bias) βαλβίδας		πόλωση (bias) βαλβίδας
πραγματική θέση		πραγματική θέση
πραγματική ροπή		πραγματική ροπή
σφάλμα θέσης στο άνω ή κάτω όριο		σφάλμα θέσης/ροπής στο άνω όριο
σφάλμα ροπής στο άνω ή κάτω όριο		σφάλμα θέσης/ροπής στο κάτω όριο
εντολή βαλβίδας από PD βρόχο		εντολή βαλβίδας από PD βρόχο
περιορισμός ρεύματος PD βρόχου		περιορισμός ρεύματος PD βρόχου
ρεύμα βαλβίδας		ρεύμα βαλβίδας
ασφάλεια		ασφάλεια
ποτενσιόμετρο για τοπικό έλεγχο θέσης		ποτενσιόμετρο για τοπικό έλεγχο θέσης
κάρτα επιλεγμένη		

Σχήμα 3-2 Κάρτες AJC1.1 και AJC2.1 πρόσθια όψη.

3.2.1 Γενική Περιγραφή Λειτουργίας και Σχηματικό Διάγραμμα Αναλογικού Βρόχου Ελέγχου.

Σε κάθε βαθμό ελευθερίας αντιστοιχεί μια κάρτα AJC (Advanced Joint Controller). Όπως θα φανεί και στα επόμενα, οι κάρτες AJC είναι ένα υβρίδιο μεταξύ αναλογικών και ψηφιακών ηλεκτρονικών. Κάθε κάρτα μπορεί να δεχθεί αναλογική ή/και ψηφιακή πληροφορία και να στείλει ψηφιακή πληροφορία, αλλά, οι κυρίως λειτουργίες που υλοποιούνται εντός της κάρτας είναι αναλογικές. Παρ' όλα αυτά, μεγαλύτερο τμήμα της παραμετροποίησης της κάρτας γίνεται ψηφιακά.

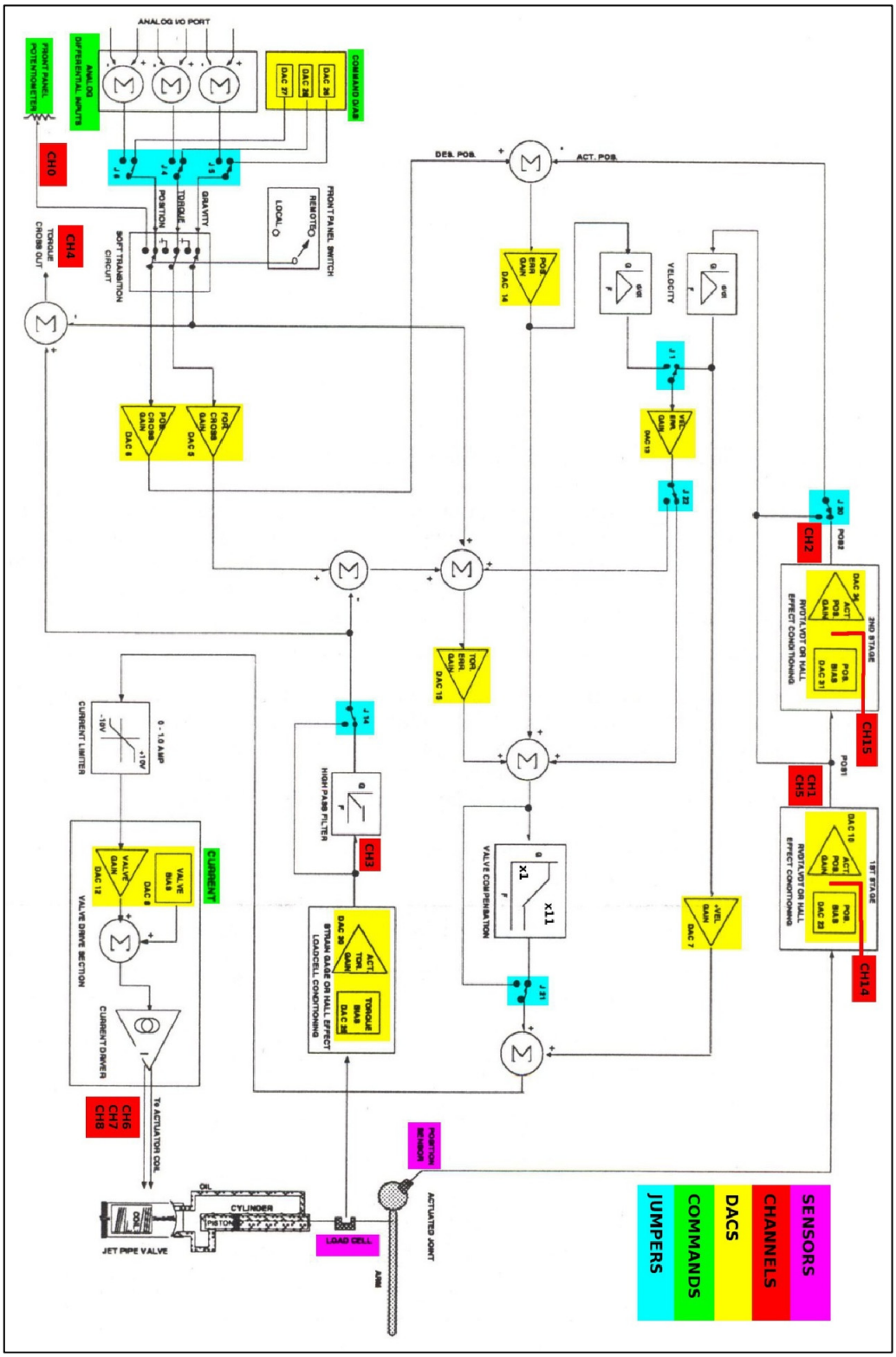
Ο υβριδικός χαρακτήρας των καρτών, αν και καθιστά τον χειρισμό τους αρκετά περίπλοκο, ήταν (για την εποχή που σχεδιάστηκαν) απαραίτητος, αφού προσφέρει αφενός την ταχύτητα του αναλογικού βρόχου ελέγχου, αφετέρου την ψηφιακή δυνατότητα γρήγορης και εύκολης μεταβολής των παραμέτρων και προσαρμογής τους σε αλλαγές της συμπεριφοράς του υλικού (ηλεκτρονικών, αισθητήρων, βαλβίδων), λόγω φθοράς ή εξωγενών παραγόντων. Η ευελιξία των καρτών ενισχύεται περεταίρω, από την δυνατότητα ρυθμίσεων μέσω γεφυρώσεων (jumper) πάνω στην κάρτα.

Οι κάρτες AJC έρχονται σε δύο εκδόσεις 1.1 και 2.1 των οποίων οι λειτουργίες διαφέρουν σε κάποιο βαθμό. Για αυτό το λόγο θα εξεταστούν πρώτα οι κοινές λειτουργίες και στην συνέχεια θα αναλυθούν οι διαφορές τους. Κάθε AJC κάρτα περιέχει έξι κυρίως τμήματα των οποίων η λειτουργία περιγράφεται παρακάτω.

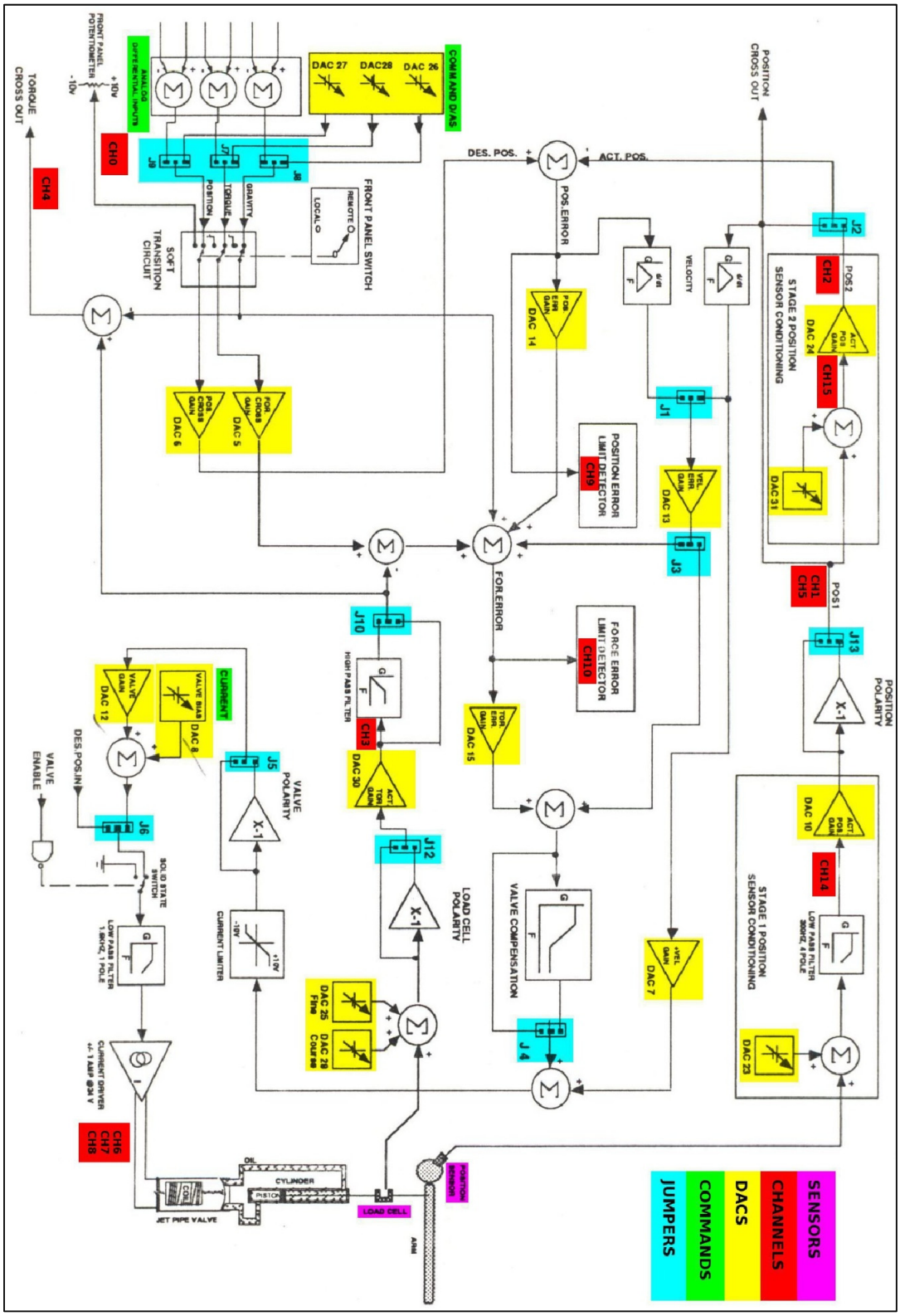
1. Το τμήμα AC/DC, του signal conditioning κυκλώματος, το οποίο είναι υπεύθυνο για την κατασκευή των σημάτων διέγερσης των αισθητήρων (AD ή DC), θέσης και ροπής, και την αναλογική επεξεργασία του σήματος αυτού, για την κατασκευή του DC σήματος εξόδου του αισθητήρα.
2. Το τμήμα βαθμονόμησης, του signal conditioning κυκλώματος, που είναι υπεύθυνο για την αλλαγή στο εύρος του σήματος ώστε να είναι συμβατό με το εύρος των τάσεων [-10…+10]V εντός του αναλογικού βρόχου καθώς και με το εύρος των επιτρεπτών τάσεων [-10…+10]V στον ADC, μετατροπέα από αναλογικό σε ψηφιακό.
3. Το τμήμα των εισόδων ελέγχου, οι οποίες μπορεί να είναι :
   - **Αναλογικά διαφορικά σήματα τηλεχειρισμού** (remote), επιθυμητής θέσης, επιθυμητής ροπής και αντιστάθμισης βαρύτητας. Τα σήματα αυτά, αφού μετατραπούν με αναφορά ως προς την γείωση, χρησιμοποιούνται στον αναλογικό βρόχο ελέγχου.
   - **Αναλογικό σήμα τοπικό** (local), από το ποτενσιόμετρο στη πρόσοψη της κάρτας. Το σήμα χρησιμοποιείται, ως έχει, ως επιθυμητή θέση στον αναλογικό βρόχο ελέγχου.
   - **Ψηφιακά σήματα τηλεχειρισμού** (remote), επιθυμητής, θέσης, επιθυμητής ροπής και αντιστάθμισης βαρύτητας. Οι ψηφιακές αυτές είσοδοι μετατρέπονται σε αναλογικές και τροφοδοτούνται στον βρόχο ελέγχου.
   - **Ψηφιακό σήμα ρεύματος στην βαλβίδα**. Η είσοδος αυτή μετατρέπεται σε αναλογική και τροφοδοτείται απευθείας στο τμήμα ισχύος.
   - Η επιλογή μεταξύ ψηφιακού ή αναλογικού τηλεχειρισμού γίνεται από τις γεφυρώσεις J4 - J6 για τις AJC1.1 ή J7 - J9 για τις AJC2.1.
   - Η επιλογή μεταξύ τηλεχειρισμού ή τοπικού σήματος γίνεται μέσω του διακόπτη στην AJC.

4. Το τμήμα του αναλογικού βρόχου, το οποίο είναι υπεύθυνο για την υλοποίηση PD ελέγχου θέσης και ροπής. Τα κέρδη του αναλογικού βρόχου μπορούν να ρυθμιστούν ψηφιακά από τον χειριστή ανά πάσα στιγμή.
5. Το τμήμα ψηφιακής εξόδου, το οποίο είναι υπεύθυνο για την μετατροπή από αναλογικό σε ψηφιακό συγκεκριμένων τάσεων σε επιλεγμένους κόμβους (channels) του κυκλώματος. Οι γνώση των τάσεων αυτών είναι απαραίτητη για τον ψηφιακό έλεγχο και την παραμετροποίηση της κάρτας.
6. Το τμήμα ισχύος, που είναι υπεύθυνο για την παροχή ρεύματος στην βαλβίδα εξαρτώμενο είτε από την έξοδο του αναλογικού βρόχου είτε από την απευθείας ψηφιακή εντολή.

Στα Σχήματα 3-3, 3-4 φαίνονται για τις AJC1.1 και AJC2.1, σε Block μορφή, τα τμήματα 2 έως 6 στα οποία μπορεί να αποσταλεί ψηφιακή πληροφορία ή να διαβαστεί από αυτά αλλά όχι το τμήμα AC/DC του signal conditioning κυκλώματος με το οποίο δεν υπάρχει αλληλεπίδραση.



Σχήμα 3-3 Block διάγραμμα AJC1.1.



Σχήμα 3-4 Block διάγραμμα AJC2.1.

3.2.2 Ο υπό-χωρος διευθύνσεων των καρτών AJC.

Διευθύνσεις Εγγραφής, Ανάγνωσης και συνοπτική περιγραφή αυτών
Η απαίτηση για αποστολή ψηφιακής πληροφορίας στις κάρτες AJC και τη λήψη ψηφιακής πληροφορίας από αυτές, οδηγεί στην ανάγκη ενός χώρου ψηφιακών διευθύνσεων εντός κάθε κάρτας. Οι διευθύνσεις αυτές είναι ίδιες για τις AJC1.1 και AJC2.1, έτσι ώστε να υπάρχει συμβατότητα προς παλαιότερες εκδόσεις (backward compatibility).

Πρέπει να επιστηθεί η προσοχή στο ότι οι διευθύνσεις αυτές είναι **είτε μόνο ανάγνωσης (RO) είτε μόνο εγγραφής (WO)**, δεν είναι και ανάγνωσης και εγγραφής. Αυτό συνεπάγεται πώς η εγγραφή ψηφιακών δεδομένων **δεν** μπορεί να επιβεβαιωθεί άμεσα, αλλά μόνο έμμεσα, και μόνο εφ όσων επηρεάζει μία ψηφιακά μετρήσιμη ποσότητα (channel) πάνω στην κάρτα.

Γενικά, σχεδόν όλες οι εγγραφές γίνονται σε κάποιο DAC (τύπου AD7547). Τα DAC είναι σε τέτοια συνδεσμολογία ώστε η ψηφιακή τιμή να λειτουργεί ως ισοδύναμο κέρδος (gain) ή πόλωση (bias) πάνω σε κάποιο αναλογικό σήμα. Το εύρος του DAC είναι 12bit άρα το εύρος των ψηφιακών τιμών είναι [0…4095]. Όταν δρα ως κέρδος, το αντίστοιχο αναλογικό εύρος είναι [0…10], με την τιμή 410 να αντιστοιχεί σε κέρδος 1 (μονάδα). Όταν δρα σαν πόλωση, το αντίστοιχο αναλογικό εύρος είναι [-10…+10] οπότε η τιμή 2048 αντιστοιχεί στο 0 (μηδέν).

Οι διευθύνσεις έχουν εύρος 6bit, άρα υπάρχουν 64 διαφορετικές διευθύνσεις από 0 έως 63, από τις οποίες μόνο 28 είναι λειτουργικές. Στον Πίνακα 3-1 παρουσιάζονται οι διευθύνσεις αυτές, καθώς και μια σύντομη περιγραφή τους, ενώ στα επόμενα εδάφια ομαδοποιούνται ως προς την συνάφεια της λειτουργίας που εκτελούν.

Πίνακας 3-1 Διευθύνσεις AJC1.1 και AJC2.1.

Διεύθυνση	Data #bit	RO WO	Έκδοση Κάρτας Θέση στο Σχήμα		Περιγραφή
			AJC1.1	AJC2.1	
00	8	RO		Ναι	**Card ID#.** Ένας κωδικός μοναδικός για κάθε AJC. Εάν είναι διαφορετικός από το αναμενόμενο σημαίνει ότι έχει αλλαχθεί η κάρτα.
01	2	RO		Ναι	**Board Status Register.** Καταχωρητής κατάστασης κάρτας. **bit0** : Μη έγκυρη τιμή στην έξοδο του ADC, εκτελείται μετατροπή. **bit1** : Υψηλό (HI) σημαίνει εντολές θέσης, ροπής και αντιστάθμισης βαρύτητας από απόσταση (master ή ψηφιακά) Χαμηλό (LO) σημαίνει εντολή θέσεις από ποτενσιόμετρο στην πρόσοψη της κάρτας. **bit2 AJC2.1 μόνο**: Χαμηλό (LO) σημαίνει πώς όλοι οι έλεγχοι ασφαλείας είναι εντάξει και επιτρέπεται στο κύκλωμα ισχύος να στείλει ρεύμα στην βαλβίδα. **bit3 AJC2.1 μόνο**: Υψηλό (HI) σημαίνει υπέρβαση του ορίου ασφαλείας σφάλματος θέσης ή ροπής.

02	12	RO	Ναι	Ναι	**Local ADC Read + Convert.** Τοπικός Μετατροπέας αναλογικού σε ψηφιακό. Μια ανάγνωση από αυτή την διεύθυνση θα επιστρέψει τη τιμή της προηγούμενης μετατροπής και θα ξεκινήσει μια νέα.
03	4	WO	Ναι	Ναι	**Local Mux Select.** Καθορίζει, μέσω των δεδομένων, τον αριθμό του αναλογικού καναλιού (Analog Channel) που οδηγείται στην είσοδο του ADC προς μετατροπή.
04	-	-	-	-	Κενή
05	12	WO	Ναι	Ναι	**Force Gross Gain.** Κέρδος για το ψηφιακό σήμα επιθυμητής Ροπής
06	12	WO	Ναι	Ναι	**Position Gross Gain.** Κέρδος για το ψηφιακό σήμα επιθυμητής Θέσης
07	12	WO	Ναι	Ναι	**Velocity Gain.** Κέρδος για το σήμα ταχύτητας, για χρήση εντός του PD αναλογικού ελέγχου.
08	12	WO	Ναι	Ναι	**Valve Command.** Εντολή ρεύματος απευθείας στην βαλβίδα αντιστοιχεί σε (-1 …+1) ampere
09	12	WO	Ναι	Ναι	**TAP threshold.** Δεν χρησιμοποιείται στο slave
10	12	WO	Ναι	Ναι	**Actual Position Gain, Stage 1.** Κέρδος για το 1° στάδιο βαθμονόμησης του σήματος από τον αισθητήρα θέσης
11	12	WO	Ναι	Ναι	**TAP Gain.** Δεν χρησιμοποιείται στο slave
12	12	WO	Ναι	Ναι	**Extension Valve Gain.** Κέρδος του σήματος του PD ελεγκτή πριν τον ενισχυτή ρεύματος.
13	12	WO	Ναι	Ναι	**Velocity Error Gain.** Κέρδος για το σφάλμα ταχύτητας, για χρήση εντός του αναλογικού PD ελέγχου
14	12	WO	Ναι	Ναι	**Position Error Gain.** Κέρδος για το σφάλμα θέσης, για χρήση εντός του PD αναλογικού ελέγχου.
15	12	WO	Ναι	Ναι	**Torque Error Gain.** Κέρδος για το σφάλμα ροπής, για χρήση εντός του PD αναλογικού ελέγχου.
16-22	-	-	-	-	Κενές
23	12	WO	Ναι	Ναι	**Actual Position Bias, Stage 1.** Πόλωση για το 1° στάδιο βαθμονόμησης του σήματος από τον αισθητήρα θέσης
24	12	WO	Ναι	Ναι	**Actual Position Gain, Stage 2.** Πόλωση σταθερά για το 2° στάδιο βαθμονόμησης του σήματος από τον αισθητήρα θέσης
25	12	WO	-	Ναι	**Actual Torque Bias Fine.** Πόλωση για το κύκλωμα βαθμονόμησης του αισθητήρα ροπής.
26	12	WO	Ναι	Ναι	**Gravity Compensation.** Εντολή αντιστάθμισης βαρύτητας
27	12	WO	Ναι	Ναι	**Desired Position In.** Εντολή επιθυμητής θέση.
28	12	WO	Ναι	Ναι	**Desired Force In.** Εντολή επιθυμητής ροπής.

29	12	WO	Ναι	Ναι	**Actual Torque Bias Course.** Πόλωση για το κύκλωμα βαθμονόμησης του αισθητήρα ροπής.
30	12	WO	Ναι	Ναι	**Actual Torque Gain.** Κέρδος για το κύκλωμα βαθμονόμησης του αισθητήρα ροπής.
31	12	WO	Ναι	Ναι	**Actual Position Bias, Stage 2.** Πόλωση για το 2° στάδιο βαθμονόμησης του σήματος από τον αισθητήρα θέσης.
32	4	RO	Ναι	Ναι	**Encoder Output (D0-D3) + lock.** Τα 4 χαμηλότερης αξίας bit από τον encoder. Ο encoder έχει 20bit.
33	16	RO	Ναι	Ναι	**Encoder Output (D4-D19) + unlock.** Τα 16 υψηλότερης αξίας bit από τον encoder. Ο encoder έχει 20bit.
32-47	-	-	-	-	Για μελλοντική χρήση.
48,52,56,60	8	WO	Ναι	Ναι	**World Mux Select.** Μια εγγραφή σε αυτή την διεύθυνση προκαλεί τους MUX σε όλες τις κάρτες ανεξαρτήτως διεύθυνσης σε ένα GAGE_να επιλέξουν το αναλογικό κανάλι που περιγράφεται από τα δεδομένα.
49,53,47,61	1	WO	Ναι	Ναι	**World ADC Convert.** Μια εγγραφή σε αυτή την διεύθυνση προκαλεί τους ADC σε όλες τις κάρτες ανεξαρτήτως διεύθυνσης σε ένα GAGE να εκτελέσουν μια μετατροπή. **Δεν γίνεται καμία ανάγνωση**.
50,54,58,62	-	-	-	-	Κενές
51,55,59,63	-	-	-	-	Κενές

Ψηφιακές διευθύνσεις ομαδοποιημένες βάσει της λειτουργίας τους

Είσοδοι Απευθείας Ελέγχου της Βαλβίδας.
Τα DAC αυτά ελέγχουν το ρεύμα που αποστέλλεται στη βαλβίδα από το στάδιο ισχύος. Το DAC 08 παρέχει το DC offset (πόλωση), ενώ το DAC 12 λειτουργεί ως πολλαπλασιαστική σταθερά, όπως περιγράφεται και στον Πίνακα 3-2.

Για να λειτουργεί σωστά ο αναλογικός βρόχος, πρέπει στο DAC 08 να εγγραφεί η τιμή 2048, που αντιστοιχεί σε ρεύμα offset 0, ενώ στο DAC 12 η τιμή 4095, ώστε να περνάει πλήρως το σήμα του αναλογικού βρόχου.

Αντίθετα, εάν στο DAC12 εγγραφεί η τιμή 0, ο αναλογικός βρόχος αποκόπτεται τελείως. Τότε το ρεύμα στην βαλβίδα ελέγχεται πλήρως από την τιμή που εγγράφεται στο DAC 08 έτσι ώστε το ψηφιακό εύρος [0…4095] να αντιστοιχεί σε ρεύμα [-1 …+1]A (Ampere).

Η δυνατότητα απευθείας ελέγχου του ρεύματος της βαλβίδας, παρέχει την δυνατότητα υλοποίησης ελέγχου ανοιχτού βρόχου, όταν χρειάζεται, αλλά και την δυνατότητα απευθείας ψηφιακού ελέγχου, παρακάμπτοντας πλήρως τον αναλογικό PD βρόχο που υλοποιείται πάνω στην AJC.

Πίνακας 3-2 Είσοδοι απευθείας ελέγχου βαλβίδας.

Διεύθυνση	Data #bit	RO WO	Έκδοση Κάρτας Θέση στο Σχήμα		Περιγραφή
			AJC1.1	AJC2.1	
08	12	WO	Ναι	Ναι	**Valve Command.** Εντολή ρεύματος απευθείας στην βαλβίδα αντιστοιχεί σε [-1 …+1] A
12	12	WO	Ναι	Ναι	**Extension Valve Gain.** Κέρδος του σήματος του PD ελεγκτή πριν τον ενισχυτή ρεύματος.

Είσοδοι Επιθυμητών Σημάτων για τον Αναλογικό Βρόχο.
Τα DAC 26, DAC 27, DAC 28, παρέχουν τα σήματα αντιστάθμισης βαρύτητας, επιθυμητής θέσης, επιθυμητής ροπής για τον αναλογικό PD βρόχο που υλοποιείται πάνω στην AJC. Η τιμή που εγγράφεται σε αυτά DAC μετατρέπεται σε τάση έτσι ώστε το ψηφιακό εύρος [0…4095] να αντιστοιχεί σε τάση [-10…+10] V.

Τα DAC 05, DAC 06, είναι πολλαπλασιαστές (κέρδη) ώστε να φέρουν το πλάτος των αντίστοιχων σημάτων στα επιθυμητά όρια, εάν για κάποιο λόγο αυτό δεν συμβαίνει.

Πίνακας 3-3 Είσοδοι επιθυμητών σημάτων αναλογικού βρόχου.

Διεύθυνση	Data #bit	RO WO	Έκδοση Κάρτας Θέση στο Σχήμα		Περιγραφή
			AJC1.1	AJC2.1	
26	12	WO	Ναι	Ναι	**Gravity Compensation.** Εντολή Αντιστάθμισης Βαρύτητας
27	12	WO	Ναι	Ναι	**Desired Position In.** Εντολή Επιθυμητής θέση.
28	12	WO	Ναι	Ναι	**Desired Force In.** Εντολή επιθυμητής ροπής.
05	12	WO	Ναι	Ναι	**Force Gross Gain.** Κέρδος για το ψηφιακό σήμα επιθυμητής Ροπής
06	12	WO	Ναι	Ναι	**Position Gross Gain.** Κέρδος για το ψηφιακό σήμα επιθυμητής Θέσης

Διευθύνσεις Παραμετροποίησης του Αναλογικού Βρόχου.
Τα DAC αυτά (Πίνακας 3-4), παρέχουν τα κέρδη που εφαρμόζονται στον αναλογικό PD βρόχο. Το εύρος ψηφιακών τιμών [0…4095] αντιστοιχεί σε αναλογικό κέρδος με εύρος [0…10]. Οι τιμές που λαμβάνουν έχουν να κάνουν με τον επιθυμητό τρόπο λειτουργίας του βραχίονα, εφ όσον αυτός ελέγχεται μέσω του αναλογικού βρόχου.

Πίνακας 3-4 Διευθύνσεις Παραμετροποίησης του Αναλογικού Βρόχου..

Διεύθυνση	Data #bit	RO WO	Έκδοση Κάρτας Θέση στο Σχήμα		Περιγραφή
			AJC1.1	AJC2.1	
07	12	WO	Ναι	Ναι	**Velocity Gain.** Κέρδος για το σήμα ταχύτητας, για χρήση εντός του PD αναλογικού ελέγχου.
13	12	WO	Ναι	Ναι	**Velocity Error Gain.** Κέρδος για το σφάλμα ταχύτητας, για χρήση εντός του αναλογικού PD ελέγχου

Διεύθυνση	Data #bit	RO WO	Έκδοση Κάρτας Θέση στο Σχήμα		Περιγραφή
14	12	WO	Ναι	Ναι	**Position Error Gain.** Κέρδος για το σφάλμα θέσης, για χρήση εντός του PD αναλογικού ελέγχου.
15	12	WO	Ναι	Ναι	**Torque Error Gain.** Κέρδος για το σφάλμα ροπής, για χρήση εντός του PD αναλογικού ελέγχου.

Διευθύνσεις Για Βαθμονόμηση των Αισθητήρων.

Τα DAC αυτά (Πίνακας 3-5), είναι τμήμα των κυκλωμάτων αναλογικής επεξεργασίας σήματος (signal conditioning) τα οποία δέχονται σαν εισόδους τα σήματα από τους αισθητήρες. Τα DAC ρυθμίζονται με τέτοιο τρόπο ώστε το σήμα στην έξοδος τους των κυκλωμάτων signal conditioning να είναι εντός των επιθυμητών ορίων τάσης [-10…+10]V. Τα όρια αυτά είναι υποχρεωτικά για την σωστή μετατροπή στον ADC από αναλογικό σε ψηφιακό. Επιπλέον, και η σχεδίαση του αναλογικού βρόχου υποθέτει πως όλα τα σήματα βρίσκονται εντός αυτών των ορίων τάσης.

Υπάρχουν δυο στάδια (stages) ρύθμισης του σήματος θέσης. Εάν είναι απαραίτητο, το ένα από τα δυο ρυθμίζεται ώστε να στέλνει σωστό σήμα στον αναλογικό βρόχο, ενώ το άλλο ρυθμίζεται έτσι ώστε να στέλνει σωστό σήμα στον ADC. Η επιλογή του σήματος που θα μπει στον αναλογικό βρόχο γίνεται μέσω της γεφύρωσης J20 . Συνήθως το δεύτερο στάδιο δεν χρησιμοποιείται.

Πίνακας 3-5 Διευθύνσεις σταδίων βαθμονόμησης.

Διεύθυνση	Data #bit	RO WO	Έκδοση Κάρτας Θέση στο Σχήμα		Περιγραφή
			AJC1.1	AJC2.1	
10	12	WO	Ναι	Ναι	**Actual Position Gain, Stage 1.** Κέρδος για το 1° στάδιο βαθμονόμησης του σήματος από τον αισθητήρα θέσης
23	12	WO	Ναι	Ναι	**Actual Position Bias, Stage 1.** Πόλωση για το 1° στάδιο βαθμονόμησης του σήματος από τον αισθητήρα θέσης
24	12	WO	Ναι	Ναι	**Actual Position Gain, Stage 2.** Πόλωση σταθερά για το 2° στάδιο βαθμονόμησης του σήματος από τον αισθητήρα θέσης
31	12	WO	Ναι	Ναι	**Actual Position Bias, Stage 2.** Πόλωση για το 2° στάδιο βαθμονόμησης του σήματος από τον αισθητήρα θέσης.
29	12	WO	Ναι	Ναι	**Actual Torque Bias Course.** Πόλωση για το κύκλωμα βαθμονόμησης του αισθητήρα ροπής.
25	12	WO	Όχι	Ναι	**Actual Torque Bias Fine.** Πόλωση για το κύκλωμα βαθμονόμησης του αισθητήρα ροπής.
30	12	WO	Ναι	Ναι	**Actual Torque Gain.** Κέρδος για το κύκλωμα βαθμονόμησης του αισθητήρα ροπής.

Ψηφιακές Διευθύνσεις Ανάγνωσης – Έξοδοι

Κανάλια (channels) - Μετατροπή από Αναλογικό σε Ψηφιακό
Όπως έχει ήδη αναφερθεί οι κάρτες AJC μπορούν να στείλουν μόνο ψηφιακή πληροφορία προς τα έξω. Για τον λόγο αυτό, κάθε κάρτα AJC περιλαμβάνει έναν ADC μετατροπέα από αναλογικό σε ψηφιακό. Ο ADC έχει εύρος τάσης εισόδου [-10…+10]V, ανάλυση 12bit και χρόνο μετατροπής 8.33μsec.

Η ανάγνωση της ψηφιακής εξόδου του ADC γίνεται από τη **διεύθυνση 03**. Μια ανάγνωση από αυτή τη διεύθυνση θα επιστρέψει την τιμή της **προηγούμενης** μετατροπής και θα ξεκινήσει μια νέα της οποίας η τιμή αποθηκευτεί μέχρι την επόμενη ανάγνωση από αυτή την διεύθυνση.

Επειδή τα αναλογικά σήματα που πρέπει να αναγνωστούν, είναι περισσότερα του ενός, η είσοδος του ADC είναι συνδεδεμένη με έναν αναλογικό πολυπλέκτη aMUX, ο οποίος συλλέγει όλα τα αναλογικά σήματα (κανάλια) και οδηγεί την είσοδο του ADC με το επιθυμητό κάθε φορά σήμα. Υπάρχουν συνολικά 16 κανάλια και αριθμούνται από το 0 έως το 15. Η επιλογή του επιθυμητού καναλιού γίνεται με εγγραφή στην **διεύθυνση μνήμης 02** και με δεδομένα (data) τον αριθμό του καναλιού, πρέπει δε να προηγείται της ανάγνωσης του ADC.

Συνήθως είναι επιθυμητή η καταγραφή στις στιγμιότυπου (snapshoot), δηλαδή η ανάγνωση του ίδιου καναλιού (π.χ. θέσης) από όλες τις AJCs σε κάποια χρονική στιγμή. Υπάρχουν δυο διευθύνσεις μνήμης οι 48 και 49 που επιτρέπουν να γίνει μια τέτοια διαδικασία.

- Μια εγγραφή στη διεύθυνση 48, με δεδομένα (data) τον αριθμό του επιθυμητού καναλιού θέτει όλους aMUX εντός του κλωβού (CAGE), στην τιμή του συγκεκριμένου καναλιού. Έτσι αποφεύγεται η ανάγκη για εγγραφή στην διεύθυνση μνήμης 03 για κάθε κάρτας ξεχωριστά.
- Στην συνέχεια μια εγγραφή στη διεύθυνση μνήμης 49, αναγκάζει όλους τους ADC του κλωβού, να εκτελέσουν μια μετατροπή την ίδια ακριβώς στιγμή, **χωρίς να λάβει χώρα κάποια ανάγνωση δεδομένων**.
- Για να ολοκληρωθεί η διαδικασία πρέπει με **διαδοχικές αναγνώσεις από την θέση 02 κάθε AJC κάρτας** να αναγνωστούν οι ψηφιακές τιμές.

Οι διευθύνσεις που σχετίζονται με την (ψηφιακή) ανάγνωση αναλογικού σήματος αναφέρονται στον Πίνακα 3-6.

Πίνακας 3-6 Διευθύνσεις χειρισμού AJC.

Διεύθυνση	Data #bit	RO WO	Έκδοση Κάρτας Θέση στο Σχήμα		Περιγραφή
			AJC1.1	AJC2.1	
02	12	RO	Ναι	Ναι	**Local ADC Read + Convert.** Τοπικός Μετατροπέας αναλογικού σε ψηφιακό. Μια ανάγνωση από αυτή την διεύθυνση θα επιστρέψει τη τιμή στις προηγούμενης μετατροπής και θα ξεκινήσει μια νέα.
03	4	WO	Ναι	Ναι	**Local Mux Select.** Καθορίζει, μέσω των δεδομένων, τον αριθμό του αναλογικού καναλιού (Analog Channel) που οδηγείται στην είσοδο του ADC στις μετατροπή.

48,52,56,60	8	WO	Ναι	Ναι	**World Mux Select.** Μια εγγραφή σε αυτή την διεύθυνση προκαλεί τους aMUX όλων των καρτών εντός του GAGE , ανεξαρτήτως διεύθυνσης, να επιλέξουν το αναλογικό κανάλι που περιγράφεται από τα δεδομένα (data).
49,53,47,61	1	WO	Ναι	Ναι	**World ADC Convert.** Μια εγγραφή σε αυτή την διεύθυνση προκαλεί τους ADC σε όλες τις κάρτες σε ένα GAGE, ανεξαρτήτως διεύθυνσης, να εκτελέσουν μια μετατροπή. **Δεν γίνεται καμία ανάγνωση**.

Τα κανάλια και τα σήματα που αντιστοιχούν σε αυτά παρατίθενται στον Πίνακα 3-7 για τις εκδόσεις AJC1.1 και AJC2.1, όπου υπάρχουν αλλαγές μεταξύ των δύο εκδόσεων αναφέρεται ρητά.

Πίνακας 3-7 Κανάλια (channels) AJC.

Κανάλι (CH)	Αντίστοιχο σήμα AJC1.1	Αντίστοιχο σήμα AJC2.1
00	**Front Panel Desired Position Knob.** Τάση από το ποτενσιόμετρο στην πρόσοψη στις κάρτας.	**Front Panel Desired Position Knob.** Τάση από το ποτενσιόμετρο στην πρόσοψη στις κάρτας.
01	**Actual Position Stage 1.** Σήμα θέσης από το στάδιο 1.	**Actual Position Stage 1.** Σήμα θέσης από το στάδιο 1.
02	**Actual Position Stage 2.** Σήμα θέσης από το στάδιο 2.	**Actual Position Stage 2.** Σήμα θέσης από το στάδιο 2.
03	**Actual Torque.** Σήμα ροπής.	**Actual Torque.** Σήμα ροπής.
04	**Torque Out.** Η διαφορά μεταξύ του σήματος ροπής και του σήματος αντιστάθμισης βαρύτητας.	**Torque Out.** Η διαφορά μεταξύ του σήματος ροπής και του σήματος αντιστάθμισης βαρύτητας.
05	**Actual Velocity.** Σήμα ταχύτητας από το από αναλογική διαφόριση του σήματος θέσης του στάδιου 1.	**Actual Velocity.** Σήμα ταχύτητας από το από αναλογική διαφόριση του σήματος θέσης του στάδιου 1.
06	**Extension Valve Voltage Before Fuse.** Η τάση στην έξοδο του ενισχυτή ισχύος στις βαλβίδας έκτασης, πριν την ασφάλεια.	**Extension Valve Voltage Before Fuse.** Η τάση στην έξοδο του ενισχυτή ισχύος πριν την ασφάλεια [-20…20]V
07	**Extension Valve Current.** Το ρεύμα που διαρρέει το πηνίο στις βαλβίδας έκτασης.	**Extension Valve Current.** Το ρεύμα που διαρρέει το πηνίο στις βαλβίδας.
08	**Extension Valve Voltage After Fuse.** Η τάση στην έξοδο του ενισχυτή ισχύος στις βαλβίδας έκτασης , μετά την ασφάλεια.	**Extension Valve Voltage After Fuse.** Η τάση στην έξοδο του ενισχυτή ισχύος μετά την ασφάλεια [-20 …20]V
09	**Flexion Valve Voltage Before Fuse.** Δεν χρησιμοποιείται στο υπάρχον σύστημα.	**Position Error Set Point.** Προκαθορισμένο μέγιστο επιτρεπτό σφάλμα θέσης. Χρησιμοποιείται σε κύκλωμα ασφαλείας εντός του αναλογικού βρόχου.

Κανάλι (CH)	Αντίστοιχο σήμα AJC1.1	Αντίστοιχο σήμα AJC2.1
10	**Flexion Valve Current.** Δεν χρησιμοποιείται στο υπάρχον σύστημα.	**Torque Error Set Point.** Προκαθορισμένο μέγιστο επιτρεπτό σφάλμα ροπής. Χρησιμοποιείται σε κύκλωμα ασφαλείας εντός του αναλογικού βρόχου.
11	**Flexion Valve Voltage After Fuse.** Δεν χρησιμοποιείται	**κενή**
12	**Actual Extension Load Cell Output.** Δεν χρησιμοποιείται	**κενή**
13	**Actual Flexion Load Cell Output.** Δεν χρησιμοποιείται	**κενή**
14	**Actual Position Bias 1.** Το σήμα του αισθητήρα θέσης μετά το βαθυπερατό φίλτρο και το DAC23	**Actual Position Bias 1.** Το σήμα του αισθητήρα θέσης μετά το βαθυπερατό φίλτρο και το DAC23
15	**Actual Position Bias 2.** Το σήμα του αισθητήρα θέσης μετά το DAC31	**Actual Position Bias 2.** Το σήμα του αισθητήρα θέσης μετά το DAC31

Παλμογεννήτριες (Encoders)

Επιπλέον των άλλων αισθητήρων, οι κάρτες υποστηρίζουν οπτικές παλμογεννήτριες (optical encoders). Σε αυτή την περίπτωση, η θέση μπορεί να αναγνωστεί μέσω των διευθύνσεων μνήμης 32 και 33.

Μια ανάγνωση από τη **διεύθυνση μνήμης 33,** θα επιστρέψει τα 16 πλέον σημαντικά (most significant) bits.

Μια ανάγνωση από τη **διεύθυνση μνήμης 32**, θα επιστρέψει τα 4 λιγότερο σημαντικά (less significant) bits και ταυτόχρονα θα κλειδώσει τα 16 πλέον σημαντικά bits τα οποία **πρέπει** να αναγνωστούν από τη διεύθυνση 33, για να ξεκλειδώσει η έξοδος του encoder.

Πίνακας 3-8 Διευθύνσεις ανάγνωσης από encoder

Διεύθυνση	Data #bit	RO WO	Έκδοση Κάρτας Θέση στο Σχήμα		Περιγραφή
			AJC1.1	AJC2.1	
32	4	RO	Ναι	Ναι	**Encoder Output (D0-D3) + lock.** Τα 4 χαμηλότερης αξίας bit από τον encoder. Ο encoder έχει 20bit.
33	16	RO	Ναι	Ναι	**Encoder Output (D4-D19) + unlock.** Τα 16 υψηλότερης αξίας bit από τον encoder. Ο encoder έχει 20bit.
Προσοχή : ανάγνωση από την διεύθυνση μνήμης 32 θα κλειδώσει τον encoder.					

Ο encoder είναι incremental, έχει ανάλυση 200000 παλμών ανά περιστροφή και το μηδέν δεν μπορεί να οριστεί αυθαίρετα, αλλά ορίζετε από το index κανάλι. Κατά την εκκίνηση του συστήματος κάθε encoder είναι κλειδωμένος στην 20bit τιμή 262144 (ή 16bit 16384) και δεν μετρά. Ξεκλειδώνει αφού το κανάλι index δώσει σήμα για πρώτη φορά. Άρα για μπορέσει να λειτουργήσει κανονικά πρέπει να αρχικοποιηθεί, το οποίο σημαίνει, ότι κάθε Β.Ε. πρέπει να στραφεί με τέτοιο τρόπο ώστε να περάσει από το μηδέν του encoder, το οποίο βρίσκεται περίπου στο μέσο στις διαδρομής του Β.Ε.

Εάν είναι επιθυμητά χαμηλά επίπεδα θορύβου εντός του αναλογικού βρόχου ή εάν δεν υπάρχει RVDT, το ψηφιακό σήμα από το του encoder μετατρέπεται σε αναλογικό εντός στις κάρτας, και μπορεί να τροφοδοτηθεί, **αντί** του σήματος από το RVDT, στον αναλογικό βρόχο ελέγχου. Αυτό γίνεται μέσω γεφύρωσης πάνω στην AJC. Με encoder (εκτός από RVDTs) είναι εξοπλισμένοι οι Β.Ε. 1, 2, 3, 4, 5, του slave arm.

Προσοχή, σε αυτή την περίπτωση και μέχρι να αρχικοποιηθούν τα encoder δεν μπορεί να εφαρμοστεί έλεγχος κλειστού βρόχου.

3.2.3 Επιπλέον επιλογές μέσω γεφυρώσεων

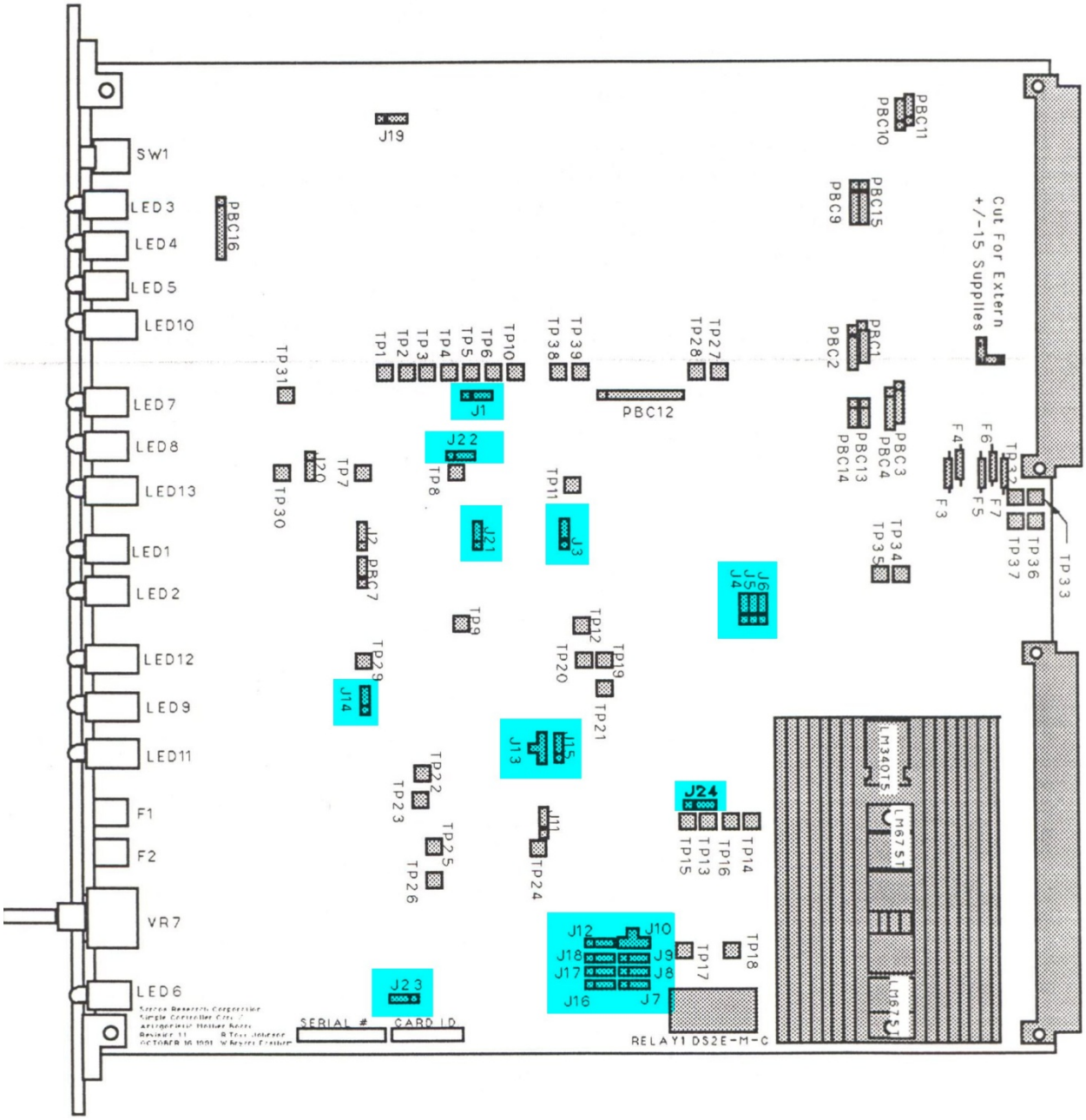
Υπάρχουν επιπλέον ρυθμίσεις οι οποίες μπορούν να γίνουν με την χρήση γεφυρώσεων (jumpers) που υπάρχουν πάνω στις AJC κάρτες. Αυτές αφορούν στο είδος των αισθητήρων θέσης και ροπής που είναι τοποθετημένες πάνω στον βραχίονα και στο πρόσημο των σημάτων μέσα στο βρόχο ελέγχου και γενικότερα πάνω στην κάρτα. Στους πίνακες 3.9, 3.10 παρατίθενται οι γεφυρώσεις κατά αύξοντα αριθμό, για τις δύο εκδόσεις καρτών. Στα Σχήματα 3.3, 3.4 φαίνεται η θέση των γεφυρώσεων πάνω στις κάρτες AJC1.1 και AJC2.1

Πίνακας 3-9 Γεφυρώσεις της AJC1.1

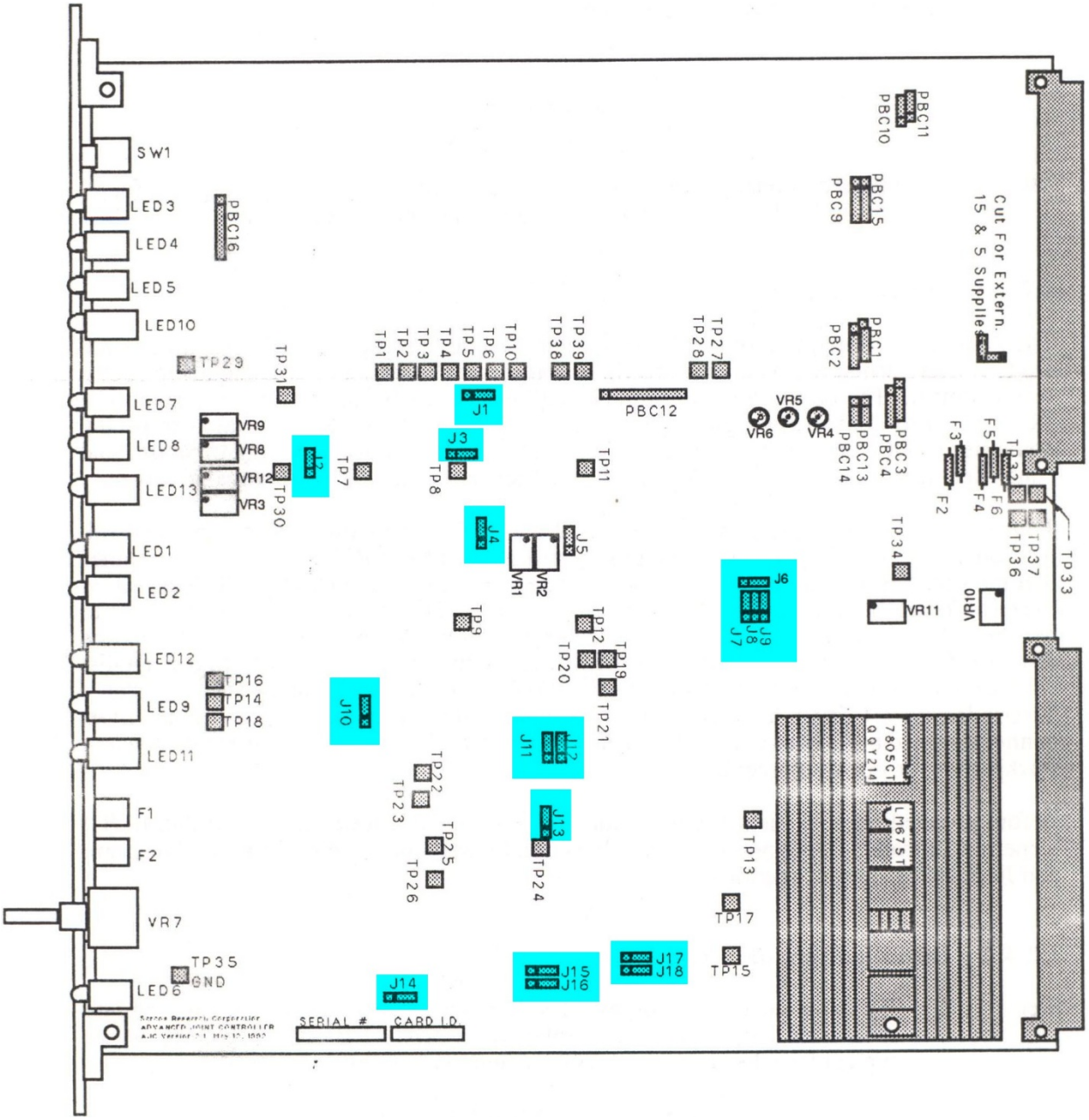
AJC1.1	
γεφύρωση	λειτουργία
J1	Velocity Mode Select
J2	Simple/Antagonistic Select
J3	Valve Driver Polarity Select
J4	Desired Force Input Select
J5	Gravity Compensation Input Select
J6	Desired Position Input Select
J7	- Position Return Select
J8	+ Position Return Select
J9	+ Position Excitation Select
J10	Position Sensor Type Select
J11	Position Polarity Reversal Select
J12	Force Bias Reference Select
J13	Force Sensor Type Select
J14	AC/DC Force Select
J15	Force Polarity Reversal Select
J16	+ Torque Return Select
J17	- Torque Return Select
J18	+ Torque Excitation Select
J19	External/Internal ADC Clock Select
J20	Actual position Stage 1 / 2 Select
J21	Valve Compensation Select (lag/lead)
J22	Velocity Error Mode Select
J23	RVDT/LVTD Internal/External Synchronizing Select
J24	Internal/External Valve Drive Select

Πίνακας 3-10 Γεφυρώσεις της AJC2.1

AJC2.1	
γεφύρωση	λειτουργία
J1	Velocity Mode Select
J2	Actual Position Stage 1/2 Select
J3	Valve Compensation Select
J4	Velocity Error Mode Select
J5	Valve Polarity Reversal Select
J6	Valve Drive Source Select
J7	Desired Torque Input Select
J8	Gravity Compensation Input Select
J9	Desired Position Input Select
J10	AC/DC Force Select
J11	Strain Gage/Hall Effect LC sensor Select
J12	Load Cell Polarity Select
J13	position Polarity Select
J14	RVDT/LVDT Internal/External Synchronization Select
J15	Position Sensor Type Select
J16	Position Excitation Select
J17	-Torque Return Select
J18	+Torque Return Select



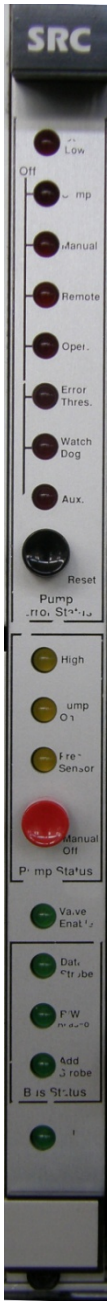
Σχήμα 3-5 Κάρτα AJC1.1



Σχήμα 3-6 Κάρτα AJC2.1

3.3 Η κάρτα BusRepv4

Η κάρτα BusRepV4 (Bus Driver And Repeater) είναι η ενδιάμεση μεταξύ του υπολογιστή υποστήριξης (Host Computer) και των καρτών AJC. Αναλαμβάνει να μοιράσει την πληροφορία από τον υπολογιστή υποστήριξης στις AJC, και αντιστρόφως να συγκεντρώσει τις πληροφορίες από τις AJC και να τις αποστείλει στον υπολογιστή υποστήριξης. Επιπλέον εκτελεί λειτουργίες γενικού ελέγχου και ασφάλειας. Τα ηλεκτρονικά συστήματα του SARCOS είναι σχεδιασμένα έτσι ώστε σε πλήρη ανάπτυξη, να ελέγχουν έως 240 Β.Ε. Το συνολικό ρομποτικό σύστημα περιλαμβάνει δεξί χέρι, αριστερό χέρι, κορμό και κεφάλι.



BusRepV4

χαμηλή πίεση

μηδενισμός (reset)

κουμπί διακοπής λειτουργίας

βαλβίδα λειτουργική

Data Strobe[-]

Write[-] / Read[+]

Address Strobe[-]

κάρτα επιλεγμένη

Σχήμα 3-7 BusRepV4 πρόσθια όψη.

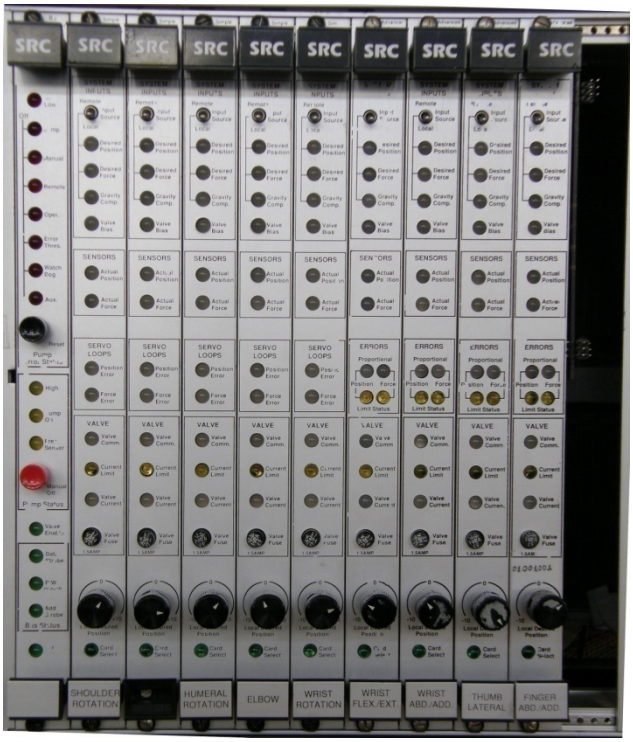
3.3.1 Ο Πλήρης Χώρος Διευθύνσεων του SARCOS - Κλωβοί (CAGEs)

Ο πλήρης χώρος διευθύνσεων του SARCOS διαμερίζεται σε μικρότερα τμήματα επονομαζόμενα κλωβοί (cages). Κάθε κλωβός αποτελείται από μια κάρτα BusRepV4 και έως 20 κάρτες AJC. Στην BusRepV4 αντιστοιχεί η πρώτη διεύθυνση του κλωβού και στις AJC οι επόμενες είκοσι, συνολικά είκοσι-μία διευθύνσεις ανά κλωβό. Βάσει του καταμερισμού αυτού, στους κλωβούς αντιστοιχούν διευθύνσεις όπως φαίνεται στον Πίνακα 3.11

Πίνακας 3-11 Κλωβοί και Διευθύνσεις.

κλωβός	διευθύνσεις	
	BusRepV4	AJC
0	0	1 … 20
1	21	22 … 41
2	42	43 … 62
⋮	⋮	⋮
11	231	232 … 251

Με αυτό τον τρόπο κάθε κλωβό μπορεί να αφιερωθεί σε ένα συγκεκριμένο τμήμα του ρομποτικού συστήματος. Στο υπάρχον σύστημα, ο κλωβός 0 αναφέρεται στον slave arm και κλωβός 1 αναφέρεται στον master arm. Κάθε κλωβός ελέγχει 10 ΒΕ, μέσω 10 αντίστοιχων AJC καρτών. Εάν ο έλεγχος των Β.Ε. ήταν ανταγωνιστικός (antagonistic control), τότε για κάθε Β.Ε. θα χρειάζονταν δυο επενεργητές άρα δυο κάρτες AJC, εξ ου και οι 10 ελεύθερες θέσεις σε κάθε κλωβό.



Σχήμα 3-8 Ο κλωβός (cage) του slave. Φαίνεται η επιπλέον κάρτα στην θέση του 10^{ου} Β.Ε.

Λαμβάνοντας υπόψη και τον υπόχωρο διευθύνσεων των καρτών AJC, ο συνολικός χώρος διευθύνσεων του SARCOS μπορεί να περιγραφεί ως

[x x DAC5…0 DOF8…0]

Όπου τα 8 πρώτα Bit αναφέρονται στην διεύθυνση της κάρτας AJC και τα επόμενα 6bit στον διεύθυνση (DAC ή ADC ή encoder) εντός κάρτας. Τα δύο ελεύθερα bit, 14 και 15, είναι πολύ σημαντικά για την υλοποίηση της κάρτας IFC η οποία παρουσιάζεται στο κεφάλαιο 4.

3.3.2 I/O Δίαυλος Greenspring μεταξύ SARCOS και Host Computer

Το SARCOS επικοινωνεί με τον host computer μέσω του διαύλου Greenspring . Ο δίαυλος αυτός συνδέεται στον δίαυλο BackPlane μέσω του τυπωμένου κυκλώματος Cable Panel (DRINVRB3).

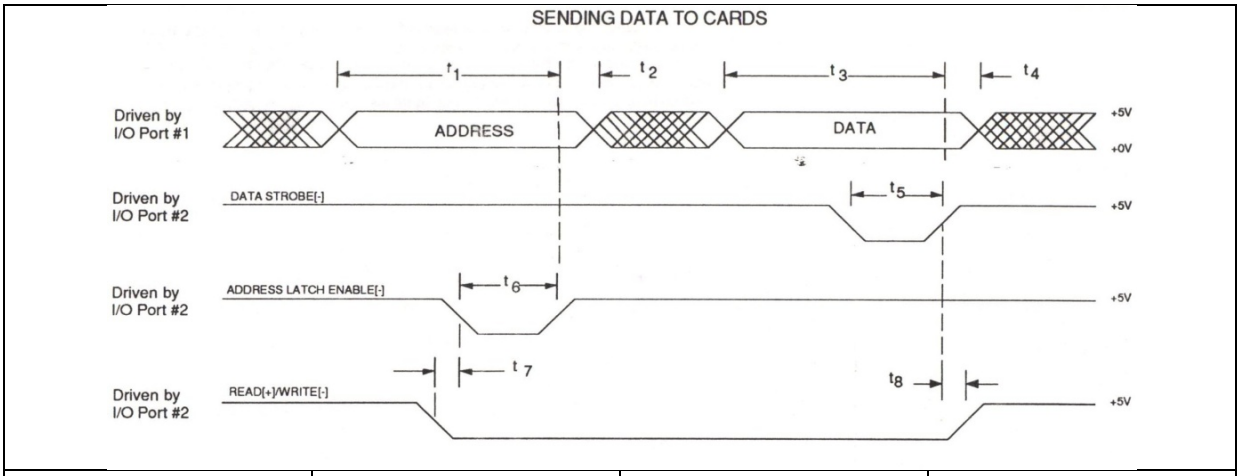
Ο δίαυλος Greenspring αποτελείται από

- 16 αμφίδρομες (bidirectional) γραμμές από τις οποίες μεταφέρεται πληροφορία από και προς το SARCOS
- 3 μονόδρομες γραμμές ελεγχόμενες από τον host computer ALE[-], DS[-], RD[+]/WR[-], τα σήματα στις οποίες ελέγχουν την ροή της πληροφορίας.

Όπως έχει ήδη αναφερθεί, ο χώρος διευθύνσεων του SARCOS είναι 14bit, ενώ το εύρος της πληροφορίας από και προς τις κάρτες AJC είναι 12bit ή, στην περίπτωση των encoder, 16bit. Πρέπει να βρεθεί τρόπος να διέρχεται πληροφορία 30bit (16+14) μέσω ενός διαύλου 16bit. Για το λόγο αυτό, υπάρχουν δυο κύκλοι βάσει των όποιων γίνεται η μεταφορά πληροφορίας, Ο κύκλος αποστολής δεδομένων, Σχήμα 3.9 και ο κύκλος ανάγνωσης δεδομένων, Σχήμα 3.10.

Με την ανάλυση που θα ακολουθήσει θα γίνει φανερό πώς είτε αποστέλλονται δεδομένα στο SARCOS είτε διαβάζονται δεδομένα από το SARCOS, πάντα προηγείται ένας κύκλος αποστολής διευθύνσεων. Μνημονικά μπορεί να περιγραφεί ως:

SENDING DATA TO AJC = SEND ADDRESS + SEND DATA
READING DATA FROM AJC = SEND ADDRESS + READ DATA

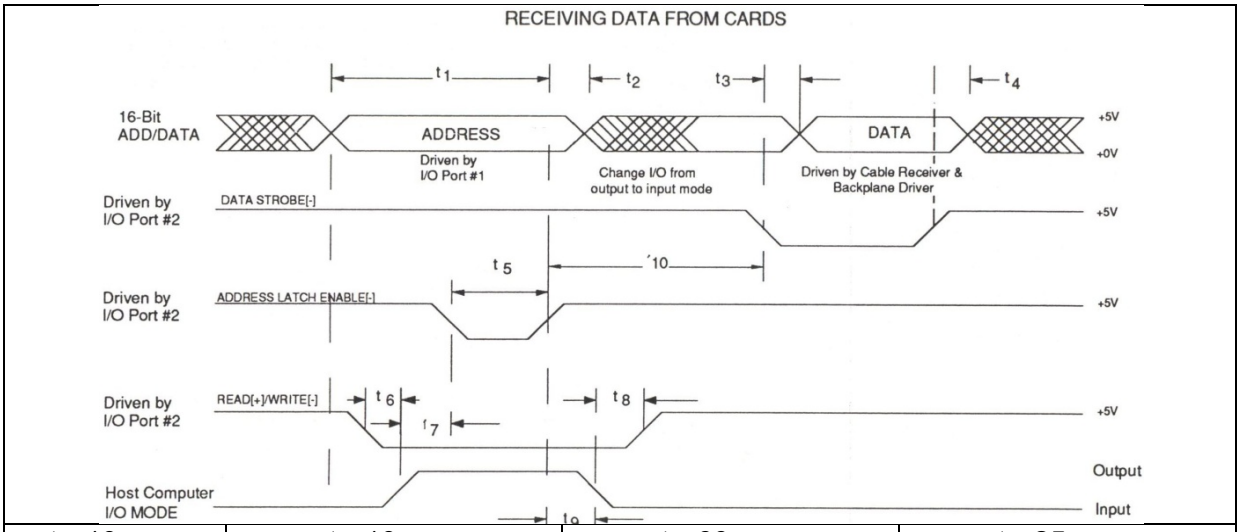


t_1>10nsec	t_2>10nsec	t_3>60 nsec	t_4>25 nsec
t_5>80 nsec	t_6>10 nsec	t_6>0 nsec	t_8>0 nsec

Σχήμα 3-9 Κύκλος αποστολής πληροφορίας στις κάρτες, SENDING DATA TO CARDS.

Για την αποστολή δεδομένων:

1. Ο host computer θέτει τη γραμμή READ[+]/WRITE[-] σε χαμηλό (LOW) και ειδοποιεί ότι θέλει να στείλει πληροφορία (διευθύνσεις ή δεδομένα).
2. Ο host computer τοποθετεί την διεύθυνση στην οποία θέλει να γράψει στον δίαυλο Greenspring .
3. Ο host computer θέτει τη γραμμή ADDRESS LATCH ENABLE[-] σε χαμηλό (LOW). Το στάδιο αυτό είναι προπαρασκευαστικό, δηλαδή, σε αυτό το στάδιο ενεργοποιούνται τα κατάλληλα ολοκληρωμένα εντός της BusRepV4, όμως, **καμία πληροφορία δεν αποθηκεύεται ακόμα.**
4. Ο host computer θέτει τη γραμμή ADDRESS LATCH ENABLE[-] σε υψηλό (HIGH). **Το μέτωπο του παλμού αναγκάζει τα latch διευθύνσεων της BusRepV4 να αποθηκεύσουν την πληροφορία που υπάρχει στον δίαυλο.**
5. Τα 8 λιγότερο σημαντικά bit της πληροφορία αυτής αποκωδικοποιούνται και εάν η διεύθυνση αντιστοιχεί σε Β.Ε. του κλωβού (cage), ενεργοποιείται η αντίστοιχή AJC. Στην συνέχεια τα 6 πλέον σημαντικά bit προωθούνται στην AJC για την επιλογή του αντίστοιχου DAC.
6. Ο host computer τοποθετεί τα δεδομένα στον δίαυλο Greenspring.
7. Ο host computer θέτει τη γραμμή DATA STROBE[-] σε χαμηλό (LOW). Το στάδιο αυτό είναι προπαρασκευαστικό, δηλαδή, σε αυτό το στάδιο ενεργοποιούνται τα κατάλληλα ολοκληρωμένα εντός της προεπιλεγμένης AJC **καμία πληροφορία δεν αποθηκεύεται ακόμα.**
8. Ο host computer θέτει τη γραμμή DATA STROBE[-] σε υψηλό (HI). **Το μέτωπο του παλμού αναγκάζει το προεπιλεγμένο DAC εντός της προεπιλεγμένης AJC να αποθηκεύσει την πληροφορία που υπάρχει στον δίαυλο.**
9. Ο host computer θέτει τη γραμμή READ[+]/WRITE[-] σε υψηλό (HI), το οποίο ολοκληρώνει τον κύκλο.



t_1>10nsec	t_2>10nsec	t_3>60 nsec	t_4>25 nsec
t_5>80 nsec	t_6>10 nsec	t_6>0 nsec	t_8>0 nsec

Σχήμα 3-10 Κύκλος αποστολής δεδομένων στις κάρτες, RECEIVING DATA FROM CARDS.

Για την λήψη δεδομένων:

1. Ο host computer θέτει τη γραμμή READ[+]/WRITE[-] σε χαμηλό (LOW) και ειδοποιεί ότι θέλει να στείλει πληροφορία (διευθύνσεις ή δεδομένα).
2. Ο host computer τοποθετεί την διεύθυνση στην οποία θέλει να γράψει στον δίαυλο Greenspring .
3. Ο host computer θέτει τη γραμμή ADDRESS LATCH ENABLE[-] σε χαμηλό (LOW). Το στάδιο αυτό είναι προπαρασκευαστικό, δηλαδή, σε αυτό το στάδιο ενεργοποιούνται τα κατάλληλα ολοκληρωμένα εντός της BusRepV4, όμως, **καμία πληροφορία δεν αποθηκεύεται ακόμα.**
4. Ο host computer θέτει τη γραμμή ADDRESS LATCH ENABLE[-] σε υψηλό (HIGH). **Το μέτωπο του παλμού αναγκάζει τα latch διευθύνσεων της BusRepV4 να αποθηκεύσουν την πληροφορία που υπάρχει στον δίαυλο.**
5. Ο host computer θέτει τη γραμμή READ[+]/WRITE[-] σε υψηλό (HIGH).
6. Ο host computer θέτει τη γραμμή DATA STROBE[-] σε χαμηλό (LOW), αναγκάζοντας την AJC να στείλει δεδομένα στην BusRepV4 και από εκεί να περάσουν στον δίαυλο Greenspring.
7. Ο host computer διαβάζει τα δεδομένα από τον δίαυλο Greenspring.
8. Ο host computer θέτει τη γραμμή DATA STROBE[-] σε υψηλό (HIGH), οι έξοδοι της BusRepv4 πάνε σε κατάσταση υψηλής εμπέδησης και ολοκληρώνεται ο κύκλος.

Προσοχή: Σε κατάσταση ηρεμίας τα σήματα ADDRESS LATCH ENABLE[-], DATA STROBE[-], READ[+]/WRITE[-] είναι όλα σε χαμηλό (LOW). Αυτό αναγκάζει τα ολοκληρωμένα I/O της BusRepV4 να είναι σε κατάσταση υψηλής εμπέδησης και αποτρέπει την πιθανότητα δύο έξοδοι (μία από το SARCOS και μια από το host computer) να οδηγούν την ίδια γραμμή ταυτόχρονα. Σε αντίθετη περίπτωση, μπορεί να προκληθεί μια κατάσταση γνωστή ως bus concatenation που θα είχε ως αποτέλεσμα μια από τις δυο εξόδους να καεί.

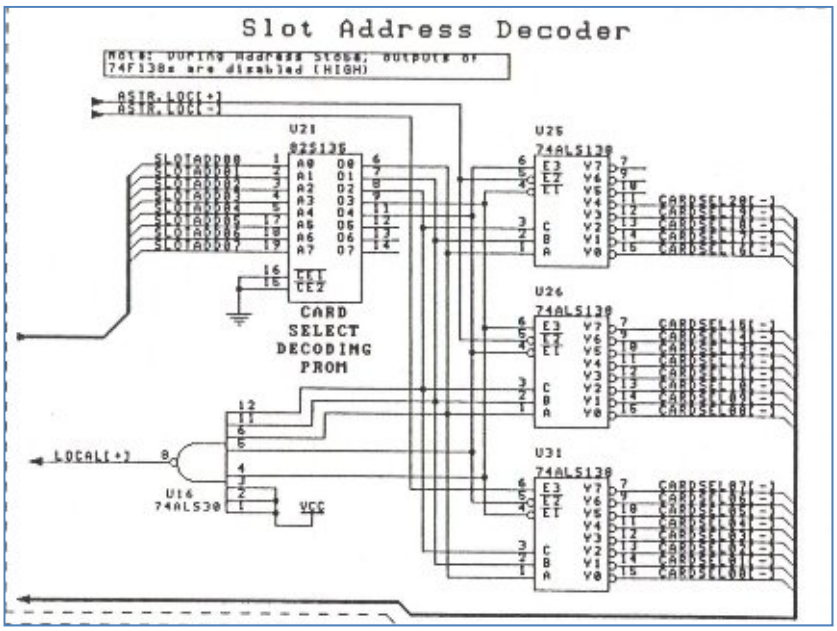
3.3.3 Η PROM αποκωδικοποίησης διευθύνσεων και επιλογής AJC

Όπως έχει αναφερθεί, κάθε κλωβός (cage) μπορεί να υποστηρίζει έως είκοσι κάρτες AJC. Κυκλωματικά όλες οι AJC κάρτες βρίσκονται πάνω σε ένα δίαυλο ονομαζόμενο BackPlane ελεγχόμενο από την BusRepV4. Ο δίαυλος αυτός χωρίζεται σε δύο τμήματα το άνω, το οποίο μεταφέρει ψηφιακά σήματα και το κάτω, το οποίο μεταφέρει αναλογικά σήματα.

Η BusRepv4 εκτελεί την αποκωδικοποίηση των 8 λιγότερο σημαντικών bit της διεύθυνσης, και εάν η διεύθυνση που σχηματίζουν είναι εντός του εύρους διευθύνσεων του κλωβού, ενεργοποιεί την αντίστοιχη AJC μέσω ενός συνόλου από-πολυπλεγμένων γραμμών, μία για κάθε κάρτα AJC. Τότε τα 6 πλέον σημαντικά bit της διεύθυνσης **αποκωδικοποιούνται εντός της AJC** και προετοιμάζεται το αντίστοιχο DAC.

Η αποκωδικοποίηση εντός της BusRepV4 γίνεται μέσω του κυκλώματος που φαίνεται στο Σχήμα 3-11 , περιλαμβάνει μια PROM 82S135 της όποιας τα περιεχόμενα είναι μοναδικά για κάθε κλωβό (cage). Φαίνονται οι 8 γραμμές διευθύνσεων [SLOTADD00 - SLOTADD07], και τα από-πολυπλεγμένα σήματα, ένα για κάθε κάρτα, [CARDSEL00 - CARDSEL20]. Το σήμα CARDSEL00 αντιστοιχεί στην ίδια την BusRepV4 και τα υπόλοιπα στις AJC.

Τα περιεχόμενα των PROM (master και slave) διαβάστηκαν, μέσω του προγραμματιστή XELTEK 3000U, και είναι πλέον διαθέσιμα σε περίπτωση αστοχίας των PROM (αν και λόγω παλαιότητας ίσως πλέον δεν υπάρχουν ισοδύναμα ολοκληρωμένα στην αγορά). Κατά τα άλλα οι BusRepV4 του master κλωβού και του slave κλωβού, είναι εντελώς ισοδύναμες, και με εναλλαγή των PROM μπορεί η μία να αντικαταστήσει την άλλη.



Σχήμα 3-11 Κυκλωματική αποκωδικοποίηση διευθύνσεων εντός του κλωβού.

3.3.4 Διευθύνσεις εντός της BusRepV4

Σε κάθε κάρτα BusRepV4 αντισταθούν τέσσερεις διευθύνσεις (εσωτερικές) οι οποίες είναι είτε μόνο εγγραφής (WO) είτε μόνο ανάγνωσης (RO), όπως περιγράφονται στον Πίνακα 3.12.

Πίνακας 3-12 Διευθύνσεις εντός της BusRepV4.

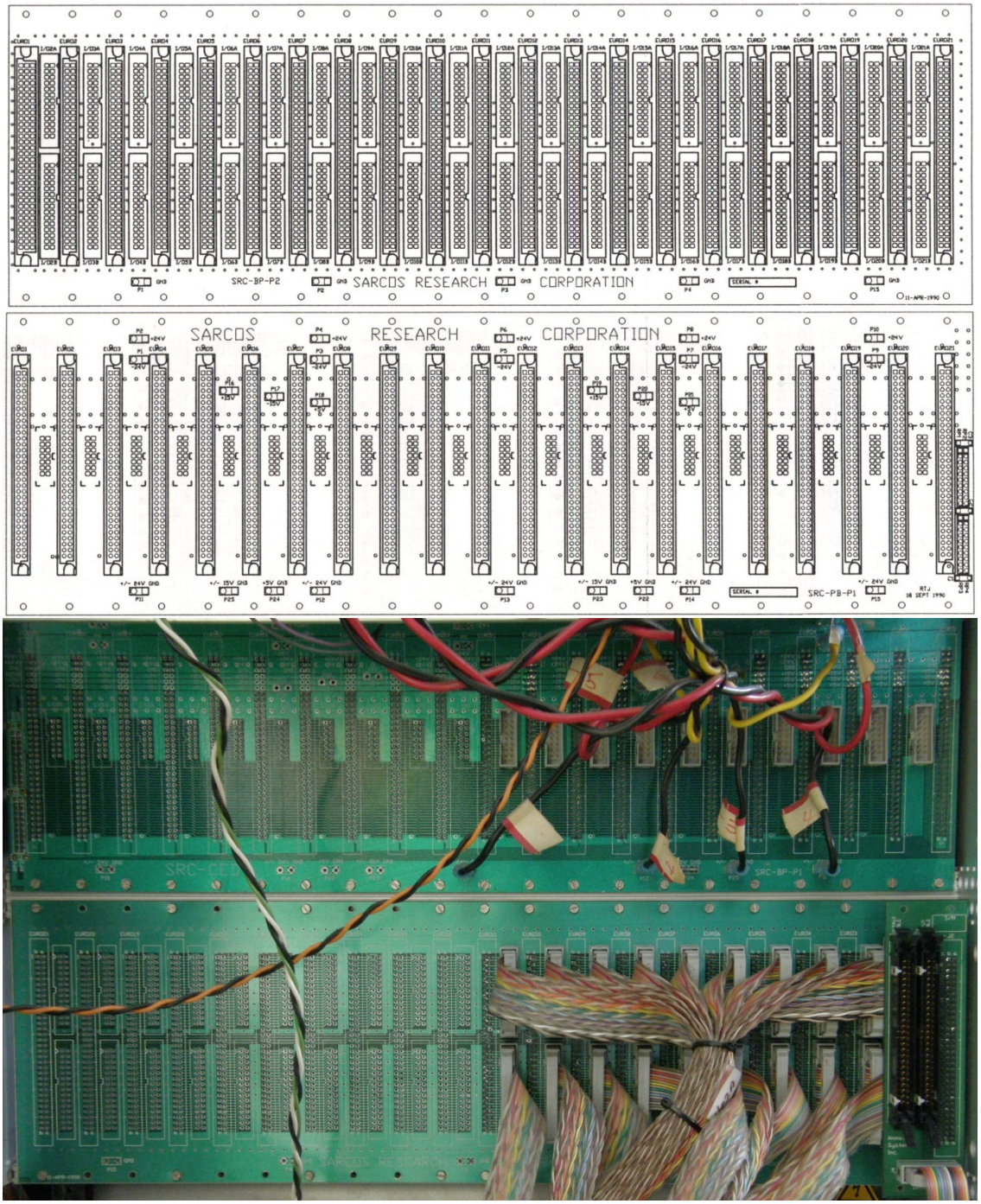
διεύθυνση	WO RO	περιγραφή
00	RO	Τα 16 σήματα (ένα για κάθε bit) σε αυτή την διεύθυνση, είναι ενδεικτικά του τρόπου επείγουσας αναστολής της λειτουργίας της αντλίας για λόγους ασφαλείας (emergency stops). Επειδή τα αντίστοιχα κυκλώματα είναι ανενεργά και παρακάμπτονται, δεν έγινε έλεγχος του τρόπου με τον οποίο τα αυτά σήματα συμπεριφέρονται οπότε οποιαδήποτε περιγραφή κρίνεται επισφαλής.
01	RO	bit00 : Όταν είναι χαμηλό (LO) σημαίνει πώς το κουμπί RESET στην πρόσοψη της κάρτας είναι πιεσμένο.
		bit01 : Όταν είναι χαμηλό (LO) σημαίνει πώς η πίεση στον βραχίονα έχει ξεπεράσει ένα προκαθορισμένο όριο.
		bit02 : Όταν είναι υψηλό (HI) σημαίνει πώς ο κινητήρας της αντλίας είναι σε λειτουργιά.
		bit03 : Όταν είναι χαμηλό (LO) σημαίνει πώς το relay ελέγχου της υψηλής πίεσης είναι ενεργό και πώς η αντλία βρίσκεται σε υψηλή πίεση.
		bit04 : Όταν είναι χαμηλό (LO) σημαίνει πώς το relay ελέγχου ασφαλείας είναι ενεργό και η αντλία μπορεί να ξεκινήσει
		bit05 : Όταν είναι υψηλό (HI) σημαίνει πώς επιτρέπεται στο υποσύστημα ισχύος των AJC να στείλει ρεύμα στις βαλβίδες. Οποιοδήποτε σφάλμα ή η απώλεια πίεσης, αποτρέπει τους ενισχυτές ισχύος από το να στείλουν ρεύμα. **Προσοχή : οι βαλβίδες ψύχονται από το λάδι που ρέει μέσα από αυτές οπότε είναι πολύ σημαντικό να μη διαρρέονται από ρεύμα εάν δεν υπάρχει ροή λαδιού (bit01).**
		bit06 : Όταν είναι υψηλό (HI) σημαίνει ότι υπάρχει ηλεκτρική σύνδεση με την αντλία άρα μπορεί να ελεγχθεί μέσω των υπολοίπων σημάτων.
		bit07 : είναι συνδεδεμένο με την γείωση και πρέπει πάντα να επιστρέφει χαμηλό (LO).
Προσοχή : Επειδή το κύκλωμα ελέγχου και ασφάλειας είναι προς το παρόν αποκομμένο και παρακάμπτεται, δεν είναι όλα τα σήματα λειτουργικά. Στην παρούσα εργασία γίνεται χρήση του bit01.		
02	WO	Μια εγγραφή σε αυτή την διεύθυνση, θα κλείσει την αντλία, εφ όσον αυτό είναι επιτρεπτό μέσω του jumper J2.
03	WO	Μια εγγραφή σε αυτή την διεύθυνση, θα θέσει την αντλία σε χαμηλή πίεση, εφ όσον αυτό είναι επιτρεπτό μέσω του jumper J1.
Προσοχή : Η κάρτα BusRepV4, αν και δέχεται τις εντολές κλεισίματος και χαμηλής πίεσης, κυκλωματικά δεν μπορεί να τις στείλει στην αντλία, καθώς το κύκλωμα ελέγχου προς το παρόν είναι αποκομμένο και ο έλεγχος γίνεται μέσω εξωτερικών διακοπτών		

Ρυθμίσεις της BusRepV4 μέσω γεφυρώσεων (jumpers)

Η συμπεριφορά της BusRepV4, όσον αφορά την απόκριση σε διάφορα σήματα εκτάκτου ανάγκης και ασφάλειας ελέγχεται μέσω 12 γεφυρώσεων. Επειδή, προς το παρόν, το κύκλωμα ασφάλειας είναι ανενεργό και παρακάμπτεται οι γεφυρώσεις αυτές δεν επηρεάζουν τη λειτουργία της κάρτας.

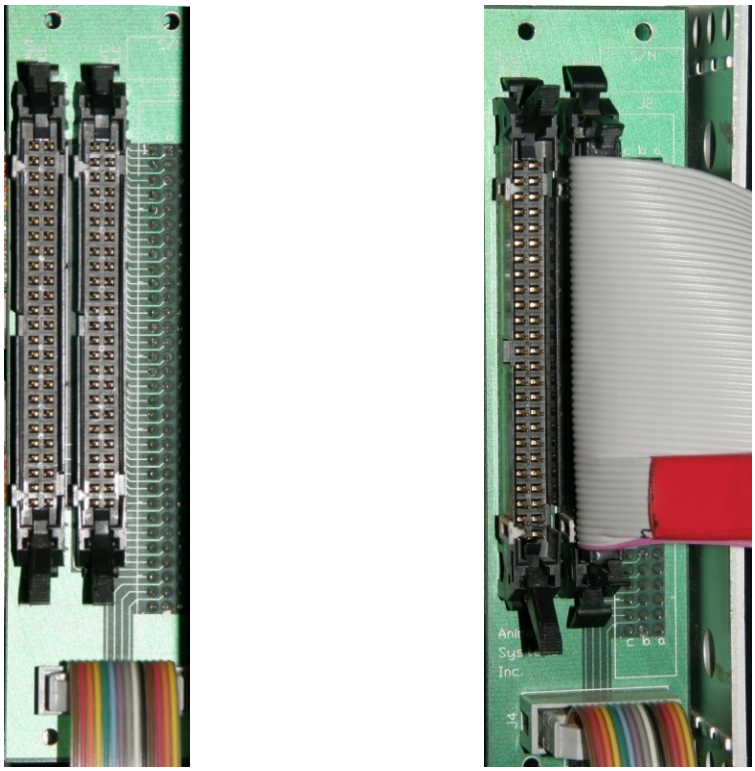
3.4 Ο Δίαυλος BackPanel - Το Κύκλωμα Cable Panel (DRINVRB3)

Σε κάθε κλωβό οι BusRepV4 και οι AJC κάρτες βρίσκονται πάνω στον δίαλο BackPlane. Ο δίαυλος αποτελείται από άνω και κάτω τμήμα. Το άνω τμήμα (P1) μεταφέρει ψηφιακά σήματα, το κάτω τμήμα (P2) μεταφέρει αναλογικά σήματα. Σχεδιάγραμμα και γωτογραφία του διαύλου φαίνονται στο σχήμα 3-12.



Σχήμα 3-12 Δίαυλος BackPlane άνω και κάτω τμήμα.

Στο κάτω αριστερό άκρο του διαύλου P2 συνδέεται το τυπωμένο κύκλωμα Cable Panel (DRINVRB3) και πάνω σε αυτό, στο συνδετήρα 50pin με την ένδειξη INPUT, συνδέεται η καλώδιο-ταινία του διαύλου Greenspring όπως φαίνεται στο σχήμα 3-13..



Σχήμα 3-13 Cable Panel (DRINVRB3) και σύνδεση με τον host computer.

3.5 Κάρτα National Instruments PCI DIO-96

Η NI PCI DIO-96 είναι μια προγραμματιζόμενη κάρτα 96 ψηφιακών εισόδων - εξόδων της National Instruments. Τοποθετείται στον PCI δίαυλο του PC, και υποστηρίζεται από το λογισμικό LabView της ίδιας εταιρίας, καθώς και από τους drivers για ANSI C DAQmx και για λειτουργικά συστήματα windows ή Linux. Λόγω αυτών των δυνατοτήτων της, η κάρτα αυτή είχε επιλεγεί και χρησιμοποιηθεί και στις δυο προηγούμενες εργασίες [4],[6] οι οποίες είχαν εκπονηθεί πάνω στο ρομποτικό σύστημα SARCOS.

3.5.1 Το ολοκληρωμένο IC8255 PIO

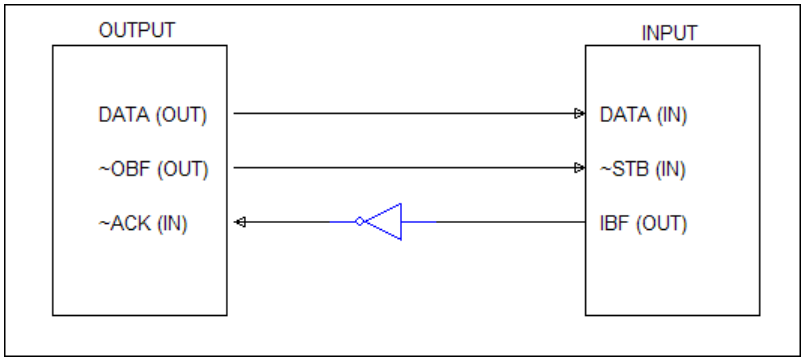
Η κάρτα βασίζεται στο ολοκληρωμένο 8255 PIO (Peripheral Input Output). Το ολοκληρωμένο παρέχει τρείς θύρες των 8bit τις A, B, C και επιτρέπει τρείς βασικούς τρόπους επικοινωνίας (I/O modes), basic (input ή output), strobed (input ή output) και bidirectional. Επιπλέον τεχνικές πληροφορίες για το ολοκληρωμένο μπορούν να βρεθούν στο [8].

Ανάλογα με τον τρόπο (mode) λειτουργίας κάποια θύρες είναι λειτουργικές και κάποιες άλλες παρέχουν τα επιπλέον σήματα ελέγχου (strobe, acknowledge κ.τ.λ.) τα οποία είναι απαραίτητα για την ροή της πληροφορίας. Κάθε θύρα μπορεί να τεθεί ανεξάρτητα σε ένα επιτρεπτό mode, εφ όσον δεν έρχεται σε αντίθεση με κάποια άλλη θύρα. Στον Πινάκα 3-13 φαίνονται τα δυνατά mode λειτουργίας για κάθε port, και τα επιπλέον σήματα που απαιτούνται.

Πίνακας 3-13 Τρόποι εισόδου/εξόδου (I/O modes) IC8255PIO.

	θύρα (port)	λειτουργία
mode 0 - input		
	A	standard input
	B	standard input
	C	standard input
mode 0 – output		
	A	standard output
	B	standard output
	C	standard output
mode 1 - strobed input		
	A	strobed input
	C.4	~STBA
	C.5	IBFA
	C.3	INTRA
	B	
	C.2	~STBB
	C.1	IBFB
	C.0	INTRB
mode 1 - strobed output		
	A	strobed output
	C.7	~OBFA
	C.6	~ACKA
	C.3	INTRA
	B	strobed output
	C.1	~OBFB
	C.2	~ACKB
	C.0	INTRB
mode 2 - bidirectional		
	A	bidirectional
	C.3	INTRA
	C.7	~OBFA
	C.6	~ACKA
	C.4	~STBA
	C.5	IBFA
Ο συμβολισμός «.» (τελεία) σημαίνει bit της αντίστοιχης θύρας. Ο συμβολισμός «~» (τεθλασμένη) σημαίνει το συμπληρωματικό του σήματος.		

Για να γίνουν κατανοητά τα παραπάνω, στο Σχήμα 3-14 φαίνεται η σύνδεση μεταξύ δύο θυρών όπου έχουν τεθεί σε strobed input mode και σε strobed output mode και περιγράφεται η λειτουργία. Οι θύρες μπορεί να βρίσκονται σε διαφορετικά ολοκληρωμένα ή ακόμα και στο ίδιο (παράδειγμα για έλεγχο της κανονικής λειτουργίας του IC)



Σχήμα 3-14 Συνδεσμολογία Strobed Input - Strobed Output.

1. Όταν τα δεδομένα έχουν εγγραφεί ασφαλώς στον καταχωρητή εξόδου DATA(OUT), το σήμα ~OBF (Output Buffer Full) γίνεται LOW (χαμηλό).
2. Το σήματα ~OBF οδηγεί την είσοδο ~STB (Strobe) σε LOW, οπότε τα δεδομένα αποθηκεύονται στον καταχωρητή DATA(IN).
3. Όταν τα δεδομένα έχουν εγγραφεί ασφαλώς στον καταχωρητή εισόδου DATA (IN), το σήμα IBF (Input Buffer Full) γίνεται HIGH (υψηλό).
4. Το σήμα IBF, αφού αντιστραφεί, οδηγεί την είσοδο ~ACK (ACKnowledge) σε LOW οπότε ο κύκλος ξεκινάει από την αρχή

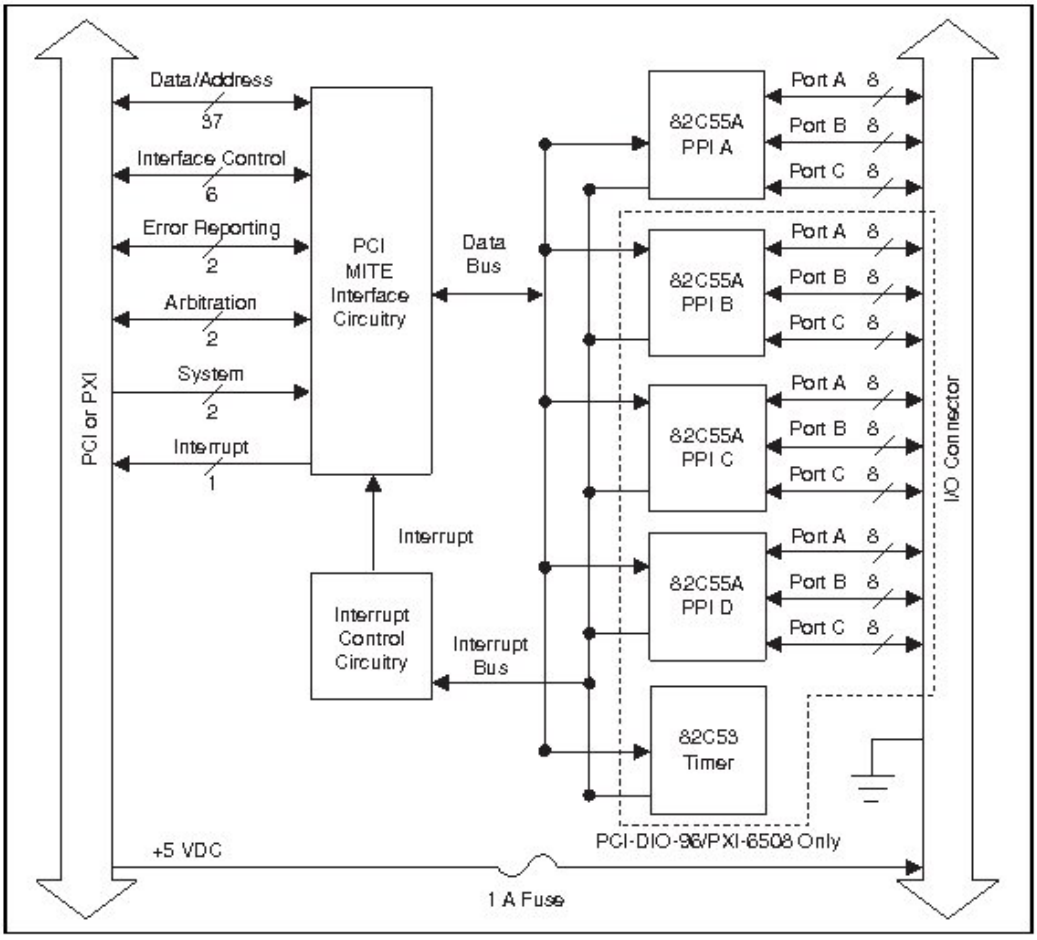
Με αυτό το τρόπο η πληροφορία ρέει από την output θύρα στην input θύρα.

Αντίστοιχα λειτουργεί και το bidirectional mode (mode 2) αλλά, εφ όσον απαιτείται, οι λειτουργίες των θυρών μπορούν να αντιμετατεθούν, ώστε να υπάρχει ροή πληροφορίας και προς τις δυο κατευθύνσεις.

Βασική σχεδιαστική λεπτομέρεια, είναι πως όταν, αλλάζει το mode από input σε output η εμπέδηση της αντίστοιχης θύρας μεταβάλλεται από υψηλή σε χαμηλή. Πολλές input θύρες μπορούν να έχουν κοινό κόμβο, αλλά **μια output θύρα απαγορεύεται να έχει κοινό κόμβο με μια άλλη outpu**t

3.5.2 Οι Θύρες της Κάρτας National Instruments PCI DIO-96

Η DIO-96 έχει 4 IC 8255 PIO, οπότε παρέχει μέχρι 96 προγραμματιζόμενες γραμμές I/O. Κάθε θύρα, σε κάθε IC 8255, μπορεί να οριστεί σαν input, output, strobed ή bidirectional σύμφωνα με τους κανόνες που προαναφέρθηκαν. Η εσωτερική δομή της κάρτας σε Block διάγραμμα φαίνεται στο Σχήμα 3-15.



Σχήμα 3-15 Block Διάγραμμα της NI DIO-96.

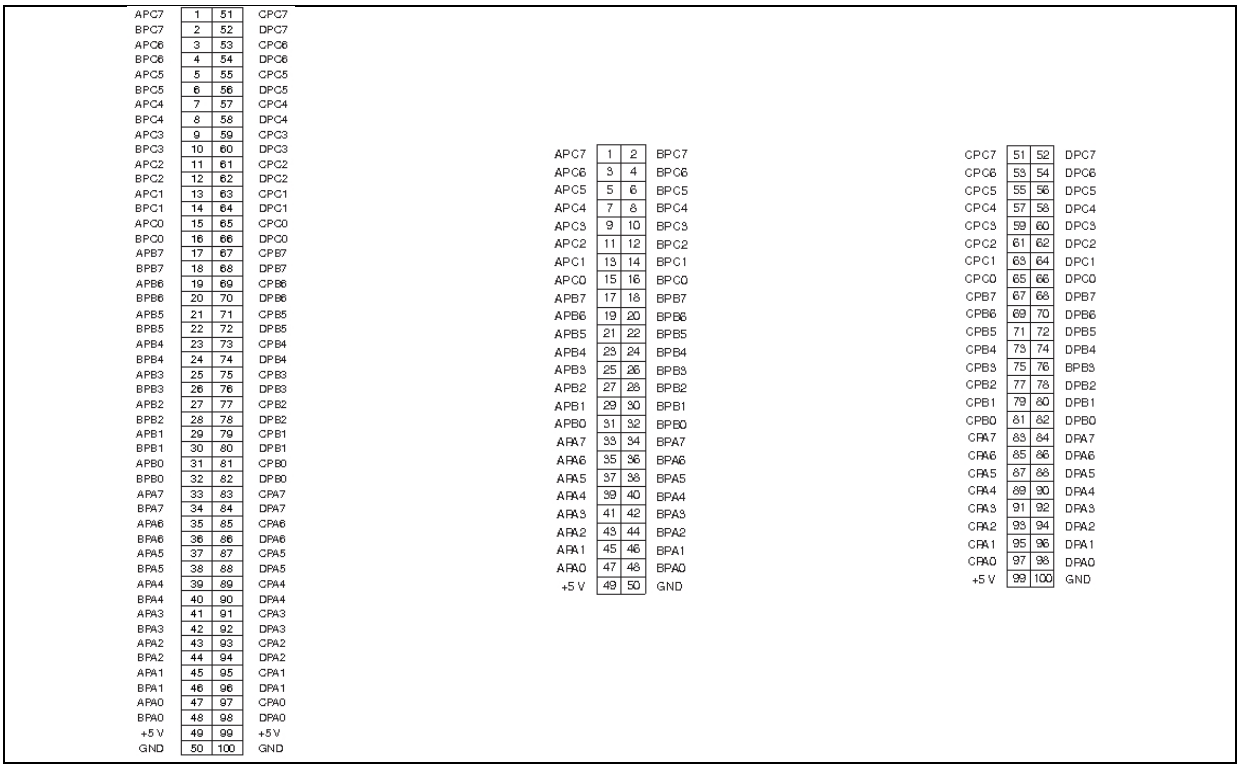
Η αρίθμηση των θυρών και των αντίστοιχων γραμμών γίνεται με τρείς ισοδύναμους τρόπους όπως παρατίθεται στον Πίνακα 3-14

Πίνακας 3-14 Εναλλακτικές αριθμήσεις Θυρών και γραμμών της NI DIO.

Θύρα	A0	B0	C0	A1	…	A3	B3	C3
	P00	P01	P02	P03	…	P09	P10	P11
γραμμή	A0.0 - A0.7	B0.0 - B0.7	C0.0 - C0.7	A1.0 - A1.7	…	A3.0 - A3.7	B3.0 - B3.7	C3.0 … C3.7
	P00.0 - P00.7	P01.0 - P01.7	P02.0 - P02.7	P03.0 - P03.7	…	P09.0 - P09.7	P10.0 - P10.7	P11.0 - P11.7
	1 - 8	9 - 16	17 - 24	25 - 32	…	77 - 84	85 - 92	93-100

Οι θύρες, εφ όσον τα φυσικά χαρακτηριστικά του mode το επιτρέπουν, μπορούν, μέσω λογισμικού (εντολές DAQmx) να ενωθούν σε μια ενιαία οντότητα, η οποία ονομάζεται κανάλι (channel). Τότε, κάθε εντολή που αναφέρεται στο κανάλι αυτό έχει επίπτωση σε όλες τις σχετιζόμενες θύρες.

Για την έξοδο των σημάτων, υπάρχει θηλυκός connector 100pin σε αυτόν συνδέεται ο αντίστοιχος αρσενικός ο οποίος μέσω διπλής καλώδιο-ταινίας καταλήγει σε δυο θηλυκούς connector 50pin. Στο Σχήμα 3-16 φαίνεται η διάταξη και η αρίθμηση των γραμμών. Για λόγους ασφαλείας και σαφήνειας των σημάτων κατά την αρχικοποίηση του υλικού, κάθε γραμμή είναι συνδεδεμένη στα 5V (pulled high) με αντίσταση 100k.



Σχήμα 3-16 Οι connectors της NI DIO.

3.5.3 Οι drivers - εντολές DAQmx για ANSI C

Η NI παρέχει για την DIO-96 τη βιβλιοθήκη DAQmx.h, για ANSI C και για λειτουργικά συστήματα Windows ή Linux, η οποία περιέχει τις εντολές εισόδου/εξόδου, αρχικοποίησης και ελέγχου της κάρτας. Η κάρτα συνοδεύεται από τεκμηρίωση (documentation) σε μορφή αρχείων pdf και HTML, καθώς και με αρχεία παραδειγμάτων σε ANSI C. Στην υλοποίηση της παρούσας εργασίας, χρησιμοποιηθήκαν οι drivers DAQmx8.3 για Linux SUSE9.3 αρχικά και SUSE11.1 αργότερα.

Ένα βασικό πρόβλημα που υπήρξε, ήταν ότι η τεκμηρίωση των εντολών DAQmx καθώς και τα παραδείγματα, δεν αναφέρονταν στην κάρτα DIO-96 συγκεκριμένα, ούτε αναφερόταν σαφώς ποίες εντολές υποστήριζε η συγκεκριμένη κάρτα . Έτσι κάποιες εντολές DAQmx δεν ήταν συμβατές με την DIO-96 ή ήταν μερικώς συμβατές. Η ασάφεια αυτή έπρεπε να αρθεί, ώστε να αποφασισθεί κατά πόσο η κάρτα ήταν ικανή να εκτελέσει τις απαιτούμενες λειτουργίες για επικοινωνία και έλεγχο του ρομποτικού βραχίονα SARCOS.

Για τον λόγω αυτό, εκτός από την ανάλυση των παραδειγμάτων σε ANSI C, έγινε και ανάλυση της βιβλιοθήκης DAQmx. Από την ανάλυση αυτή έγινε φανερό ότι υπάρχουν τρία επίπεδα (level) εντολών DAQmx.

1. Σε πρώτο επίπεδο είναι εντολές που μεταβάλουν ή διαβάζουν κάποια ιδιότητας (property), και μέσω αυτής ελέγχουν την συμπεριφορά της κάρτας.
2. Σε δεύτερο επίπεδο είναι εντολές πιο γενικές που ελέγχουν, μέσω των εντολών πρώτου επιπέδου, μια ομάδα από ιδιότητες (properties) αρχικοποιώντας την κάρτα για μια συγκεκριμένη λειτουργία.
3. Σε τρίτο επίπεδο είναι οι εντολές που εκτελούν τις διαδικασίες I/O.

Αφού έγινε κατανοητή η δομή των ιδιοτήτων (properties) και των εντολών DAQmx, σταδιακά μέσω δοκιμαστικού κώδικα ελέγχθηκε η συμβατότητα τους με την κάρτα DIO-96. Εάν μια ιδιότητα δεν ήταν συμβατή με την κάρτα, τότε η κάρτα επέστρεφε μήνυμα σφάλματος. Επειδή υπάρχει αλληλεξάρτηση μεταξύ κάποιον properties, σημαντικό ρόλο είχε και η σειρά με τις οποίες οι properties αυτές άλλαζαν. Επιπλέον, λόγω της παλαιότητας της κάρτας, δεν υπάρχουν πολλές αναφορές στο διαδίκτυο και όσες υπάρχουν αναφέρονται σε είτε κώδικα LabView είτε σε μια παλαιότερη μορφή εντολών DAQmx μη συμβατή με την παρούσα. Και στις δυο περιπτώσεις δεν υπάρχει προφανής αντιστοιχία μεταξύ των εντολών των παραδειγμάτων και των εντολών DAQmx.

3.5.4 Πειράματα Αναγνώρισης μέγιστης ταχύτητα ανά τρόπο λειτουργίας

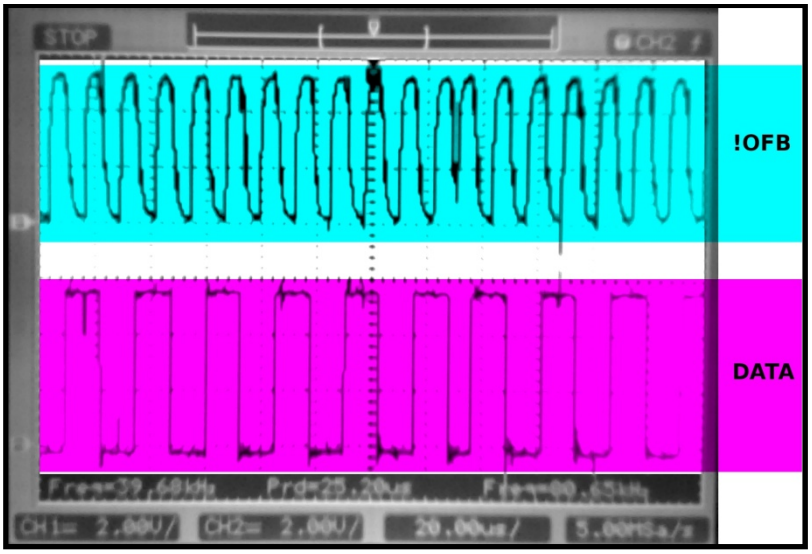
Όταν έγιναν κατανοητές οι εντολές, οι δοκιμές επικεντρώθηκαν στην διερεύνηση των τρόπων (modes 0, 1, 2) εισόδου/εξόδου και στις ταχύτητες με τις οποίες είναι δυνατόν να λειτουργήσουν. Για τον σκοπό αυτό γράφθηκε κώδικας όπου:

- Η θύρα P00 (A0) αρχικοποιήθηκε ως έξοδος
- Η θύρα P03 (A1) αρχικοποιήθηκε ως είσοδος
- Βρόχος αναλάμβανε την αποστολή και την λήψη δεδομένων.
    - Ως δεδομένα αποστέλλονται εναλλάξ οι 8bit τιμές 0x00 και 0xFF
- Σε παλμογράφο διαβάζεται το bit P00.0 το οποίο είναι το λιγότερο σημαντικό ψηφίο των δεδομένων καθώς και το σήμα ~STB, εφ όσον η επικοινωνία είναι strobed.
    - Κάθε φορά που το bit P00.0 αλλάζει σημαίνει ότι έχουν λάβει χώρα ένας κύκλος ανάγνωσης **και** ένας κύκλος εγγραφής.

Ο κώδικας των δοκιμών παρατίθεται στο παράρτημα. Στον επόμενο Πίνακα … γίνεται παρουσίαση των αποτελεσμάτων και στο Σχήμα 3-18 φαίνονται τα παλμογραφήματα των δοκιμών.

Πίνακας 3-15 Συχνότητες μέγιστης λειτουργίας ανα τρόπο I/O.

mode	συχνότητα επικοινωνίας
A0 standard output	5kHz
A3 standard input	
A0 strobed output	80kHz
A3 strobed input	
A0 bidirectional	Δεν έγινε εφικτή η αρχικοποίηση μέσω των εντολών DAQmx.
A3 bidirectional	



Σχήμα 3-17 Μέγιστη συχνότητα λειτουργίας strobed mode.

Το σήμα !OBF (output buffer full) μεταβάλλεται με συχνότητα 80kHz. Τα δεδομένα για εποπτικούς λόγους ήταν εν αλλάξ 0 και 1, οπότε το σήμα DATA μεταβάλλεται με συχνότητα 40kHz. Αυτή είναι η μεγίστη συχνότητα επικοινωνίας που επιτεύχθηκε με το δεδομένο υλικό (PC , NI-DIO και λειτουργικό).

Για την επικοινωνία με το SARCOS απαιτείται **αμφίδρομος δίαυλος δεδομένων** και **μονόδρομος δίαυλος ελέγχου**, ελεγχόμενος από το PC.

Η πλέον κοντινή σε αυτή λειτουργία της NI-DIO, είναι της αμφίδρομης επικοινωνίας, με την προϋπόθεση, ότι τα σήματα του διαύλου ελέγχου από το SARCOS προς την DIO (τα οποία δεν παράγει το SARCOS) θα κατασκευάζονται, ως ψευδοσήματα, μέσω επιπλέον καναλιών της κάρτα DIO.

Όμως, η επιθυμητή λειτουργία, της αμφίδρομής (bidirectional) επικοινωνίας δεν μπόρεσε να επιτευχτεί. Το συμπέρασμα της ανάλυσης είναι πως η κάρτα DAQmx, ως έχει, δεν μπορεί να υποστηρίξει αμφίδρομη επικοινωνία με την κάρτα BusRepV4.

3.5.5 Υπολογιστής και Λειτουργικό Σύστημα

Χρησιμοποιήθηκε προσωπικός υπολογιστής (PC) με :

- επεξεργαστή P4 χρονισμένο στα 1500MHz
- δίαυλο PCI στα 133kHz
- μνήμη 512 MB
- Κάρτα γραφικών MAXTOR NVIDIA 64MB

Το λειτουργικό αρχικά ήταν LINUX Suse9.3 και στην συνέχεια αναβαθμίστηκε στην έκδοση Suse11.0. Έγινε προσπάθεια να φορτωθεί το RTLinux για εφαρμογές πραγματικού χρόνου αλλά δεν ήταν επιτυχής

Η ανάπτυξη του κώδικα έγινε σε περιβάλλον KDEveloper, σε γλώσσα ANSI C. Όπου χρειάστηκε να γίνει αριθμητική μετ-επεξεργασία (post processing) των δεδομένων έγινε σε περιβάλλον MATLAB.

4. Υλικό (Hardware) που αναπτύχθηκε στα πλαίσια της παρούσας εργασίας

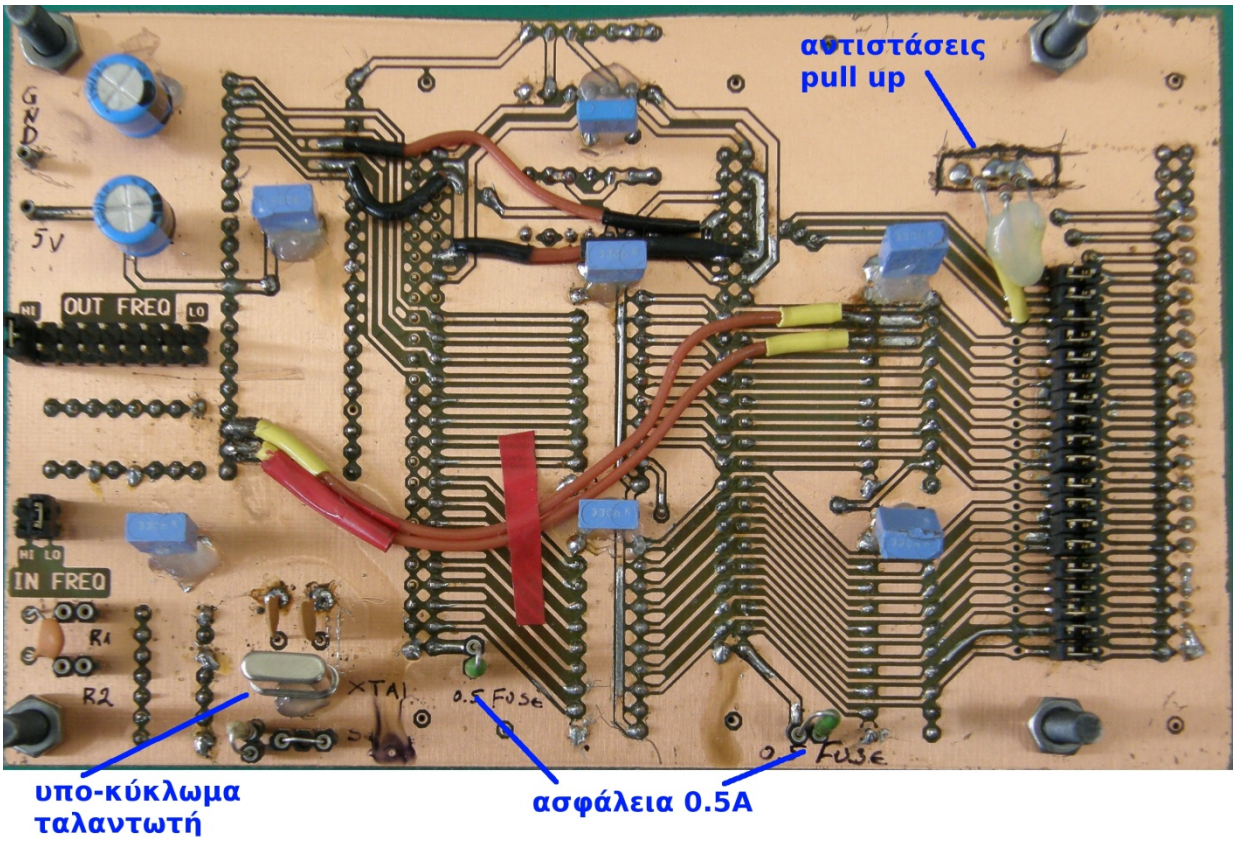
4.1 Η Κάρτα διεπαφής (interface) μεταξύ NI DIO-96 και SARCOS

4.1.1 Η ανάγκη κατασκευής της κάρτας.

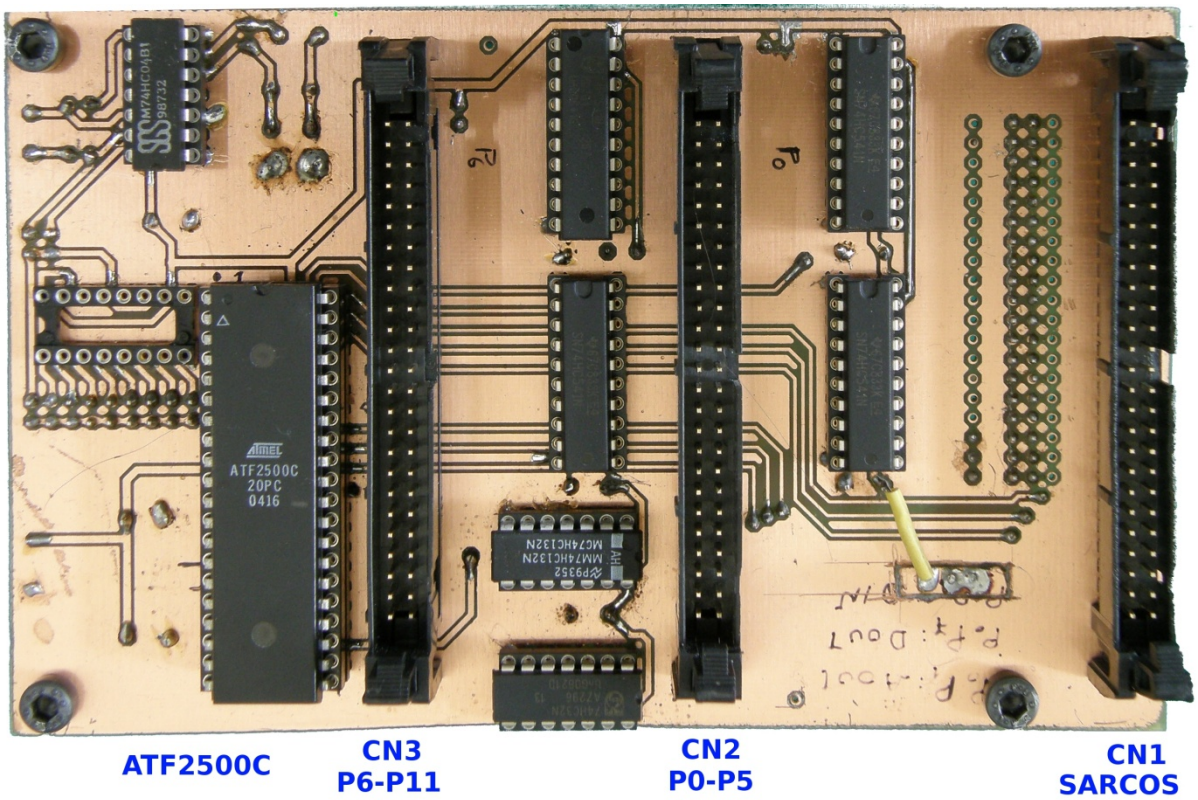
Από την διερεύνηση του κεφαλαίου 3 φαίνεται πως η κάρτα DIO-96 δεν είναι ικανή, ως έχει, για την άμεση αποκατάσταση επικοινωνίας με την BusRepV4 και ο λόγος είναι η αδυναμία αρχικοποίησης των θυρών για αμφίδρομή επικοινωνία (bidirectional mode). Υπήρξε λοιπόν το ερώτημα εάν έπρεπε να αλλαχθεί, ή εάν με κάποιο εξωτερικό κύκλωμα θα γινόταν εφικτή η επικοινωνία. Οι απαιτήσεις που πρέπει να πληρούνται είναι οι εξής:

- Αμφίδρομός Δίαυλος Δεδομένων. Πρέπει η ίδια γραμμή δεδομένων να γίνεται **είτε** είσοδος **είτε** έξοδος ανάλογα με τις ανάγκες.
- Δίαυλος ελεγχόμενος από το εξωτερικό κύκλωμα. Το SARCOS δεν στέλνει κανένα σήμα στον δίαυλο ελέγχου και όλη η επικοινωνία πρέπει να ελέγχεται εξωτερικά. Άρα τα σήματα ελέγχου πρέπει να παράγονται από το εξωτερικό κύκλωμα.
- Υψηλότερη δυνατή συχνότητα επικοινωνίας. Το κύκλωμα πρέπει να συνεργάζεται με την DIO-96 προγραμματισμένη για λειτουργία σε mode με την υψηλότερη δυνατή ταχύτητα.

Το κύκλωμα που επιλέχθηκε, σχεδιάστηκε και κατασκευάστηκε και πλέον θα αναφέρεται ως interface card ή IFC φαίνεται στα Σχήματα 4.1έως 4.3



Σχήμα 4-1 Η κάρτα IFC (κάτω όψη).



Σχήμα 4-2 Η κάρτα IFC (άνω όψη).

4.1.2 Περιγραφή κάρτας και Σχηματικό block διάγραμμα.

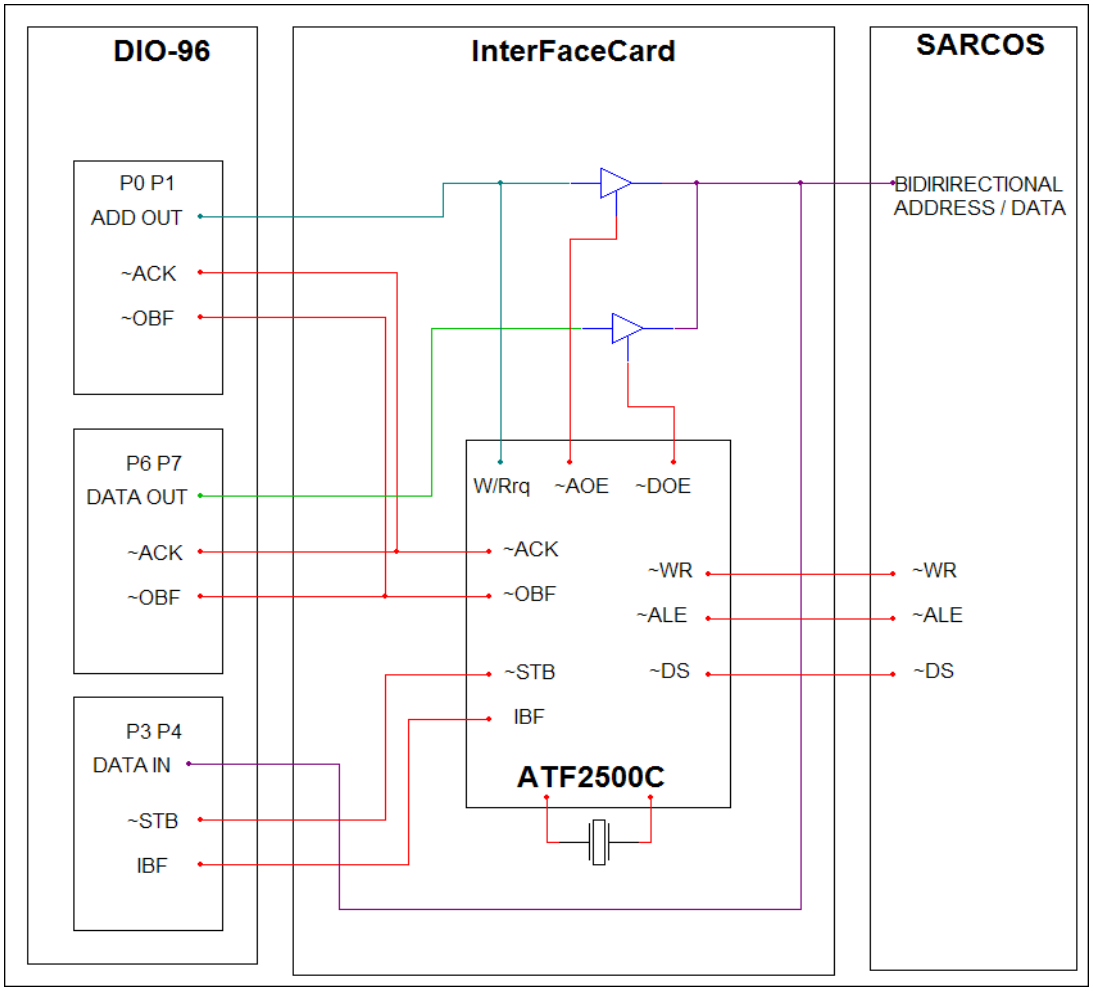
Η κάρτα επικοινωνεί με τις υπόλοιπες βαθμίδες μέσω τριών αρσενικών συνδετήρων 50pin. Σε αυτού συνδέονται 3 καλώδιο-ταινίες μέσω θηλυκών συνδετήρων. Από δεξιά προς τα αριστερά συνδέονται οι

- CN1, Καλώδιο-ταινία προς SARCOS. Ο συνδετήρας αυτός συνδέεται στην κάρτα CablePanel, στον αντίστοιχο αρσενικό συνδετήρα με την ένδειξη INPUT.
- CN2, Καλώδιο-ταινία από NI DIO. Θύρες 0 έως 5 (γραμμές 1 - 50)
- CN3, Καλώδιο-ταινία από NI DIO. Θύρες 6 έως 11 (γραμμές 51- 100)

Η κάρτα τροφοδοτείται με τάση 5V, μέσω δύο ασφαλειών 0.5A έκαστη, από τις γραμμές 49,50 και 99,100 της NI DIO. Η κάρτα σε κανονική λειτουργία καταναλώνει περίπου 250mA.

Η IFC βασίζεται στο ολοκληρωμένο Atmel ATF2500C, το οποίο είναι ένα chip προγραμματιζόμενης λογικής (Complex Programmable Logic Device). Το συγκεκριμένο IC επιλέχθηκε λόγω δυνατοτήτων, αλλά και εξοικείωσης από την πλευρά του γράφοντα.

Παρέχει 14 pin εισόδων, 24 pin προγραμματιζόμενων εισόδων/εξόδων, 48 flip/flop και 2500 πύλες. Ο κώδικας γράφηκε σε γλώσσα CUPL - WinCUPL και ο προγραμματισμός έγινε σε προγραμματιστή XELTEK SUPERPRO 3000U. Περισσότερες πληροφορίες μπορούν να βρεθούν στο Παράρτημα.Π-2



Σχήμα 4-3 Block Διάγραμμα IFC.

Το ATF2500C χρονίζεται στα 8MHz και αναλαμβάνει να στείλει όλα τα απαραίτητα σήματα τόσο προς την DIO-96 όσο και προς το SAROS, ώστε να επιτευχθεί επικοινωνία συμβατή με τις απαιτήσεις που αναφέρθησαν στο προηγούμενο εδάφιο. Σε κάθε περίπτωση, το εύρος των σημάτων προς το SARCOS πρέπει να είναι μεγαλύτερο από 80μs, οπότε μέγιστη ταχύτητα ρολογιού είναι τα 12MHz.

Στο Σχήμα 4-3 φαίνονται τα σήματα που διαχειρίζεται ιη κάρτα IFC. Τα σήματα ~WR, ~ALE, ~DS, ~ACK, ~OBF είναι συνδεδεμένα στα 5Volt (pulled high) μέσω αντιστάσεων 10kΩ για λόγους σαφήνειας των σημάτων κατά την αρχικοποίηση του υλικού.

4.1.3 Δομή των σημάτων - Ο συνολικός χώρος διευθύνσεων

Για να συνεργαστεί η κάρτα IFC με την κάρτα NI-DIO, πρέπει η δεύτερη να έχει αρχικοποιηθεί με συγκεκριμένο τρόπο όπως φαίνεται στον Πίνακα 4-1.

Πίνακας 4-1 Αρχικοποίηση θυρών NI-DIO & IFC.

κανάλι	Θύρα	τρόπος λειτουργίας	πληροφορία
εξόδου 32bit	P0,P1	strobed output	διευθύνσεις SARCOS
	P6,P7	strobed output	δεδομένα προς SARCOS
εισόδου 16bit	P3,P4	strobed input	δεδομένα από SARCOS

Η πληροφορία (σήματα) που το πρόγραμμα στέλνει προς την NI-DIO, μέσω των DAQmx, πρέπει είναι 32bit και έχει την δομή που φαίνεται στον Πίνακα 4-2. Απόλυτη σημασία έχουν τα bit 14 και 15 γιατί ο συνδυασμός τους είναι αυτός που πληροφορεί την κάρτα IFC πώς θα διαχειριστεί την πληροφορία που λαμβάνει.

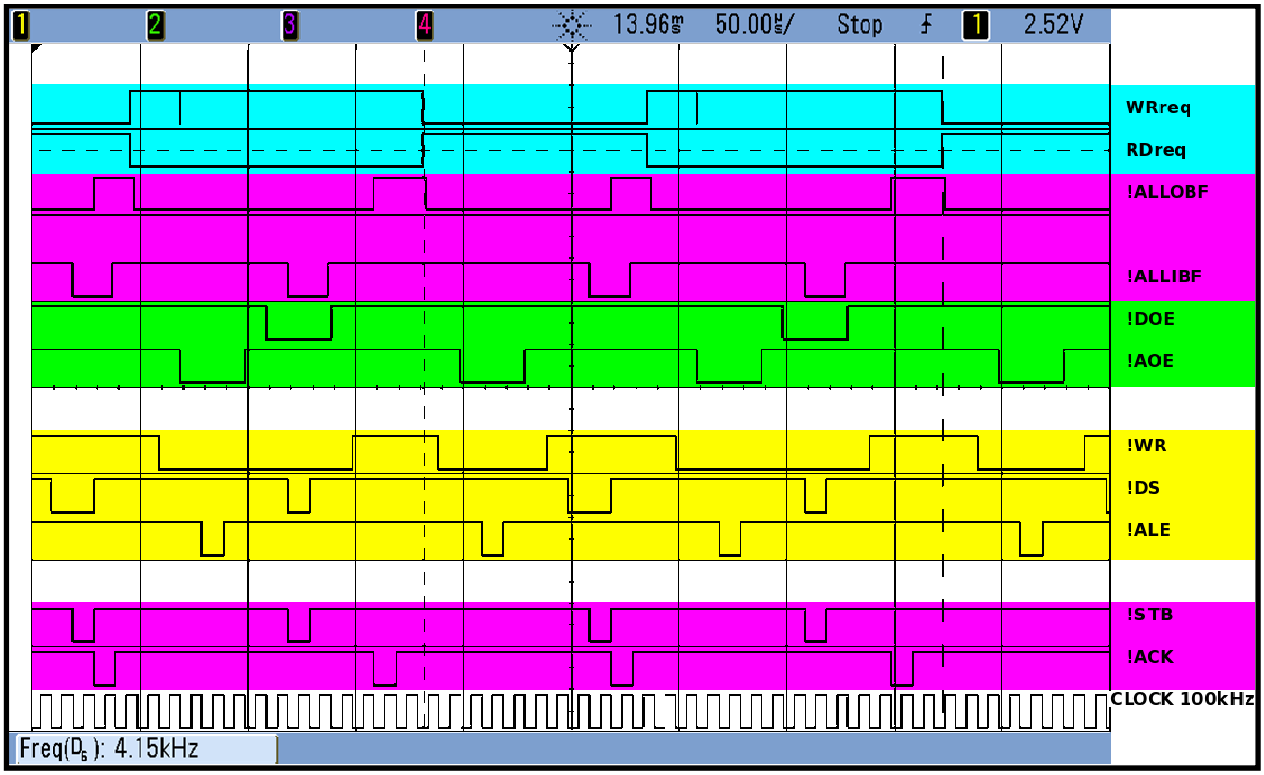
Πίνακας 4-2 Ο χώρος διευθύνσεων μαζί με την IFC.

	δεδομένα	διευθύνσεις			
		IFC		SARCOS	
για αποστολή δεδομένων στο SARCOS	DATA15…0	**0**	**1**	DAC5…0	DOF8…0
για ανάγνωση δεδομένων από το SARCOS	NoC15…0 (αδιάφορα)	**1**	**0**	DAC5…0	DOF8…0

Περιγραφή Λειτουργίας

Η λογική λειτουργίας, κάνει χρήση του γεγονότος, ότι ο χώρος διευθύνσεων του SARCOS έχει εύρος 14bit, οπότε τα δυο περισσευούμενα bit, 14 και 15, χρησιμοποιούνται για την σηματοδότηση ενός κύκλου εγγραφής δεδομένων στο SARCOS ή ενός κύκλου ανάγνωσης δεδομένων από το SARCOS.

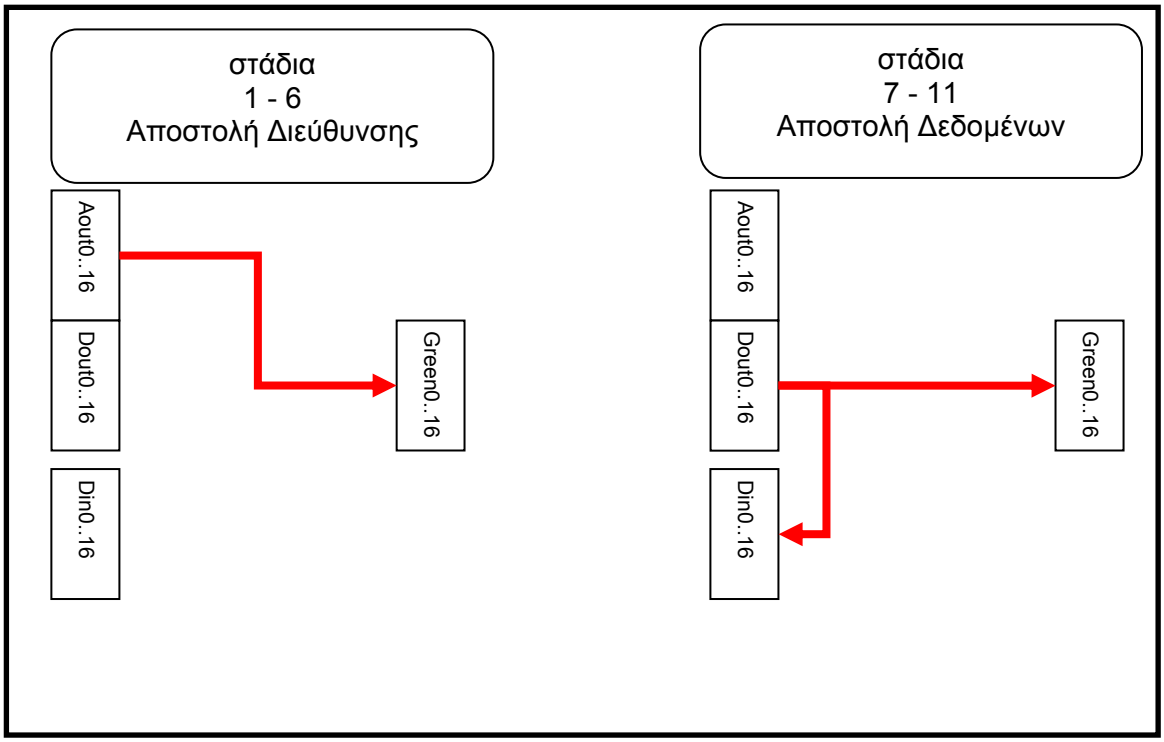
Στο Σχήμα 4-4 φαίνονται τα σήματα από και προς την κάρτα σε ένα κύκλο ανάγνωσης και ένα κύκλο εγγραφής. Η συχνότητα ρολογιού είναι εσκεμμένα πολύ μικρή για να είναι ευδιάκριτα τα πλάτη των σημάτων.



Σχήμα 4-4 Σήματα χρονισμού παραγόμενα από την IFC.

Λειτουργία εγγραφής δεδομένων στο SARCOS

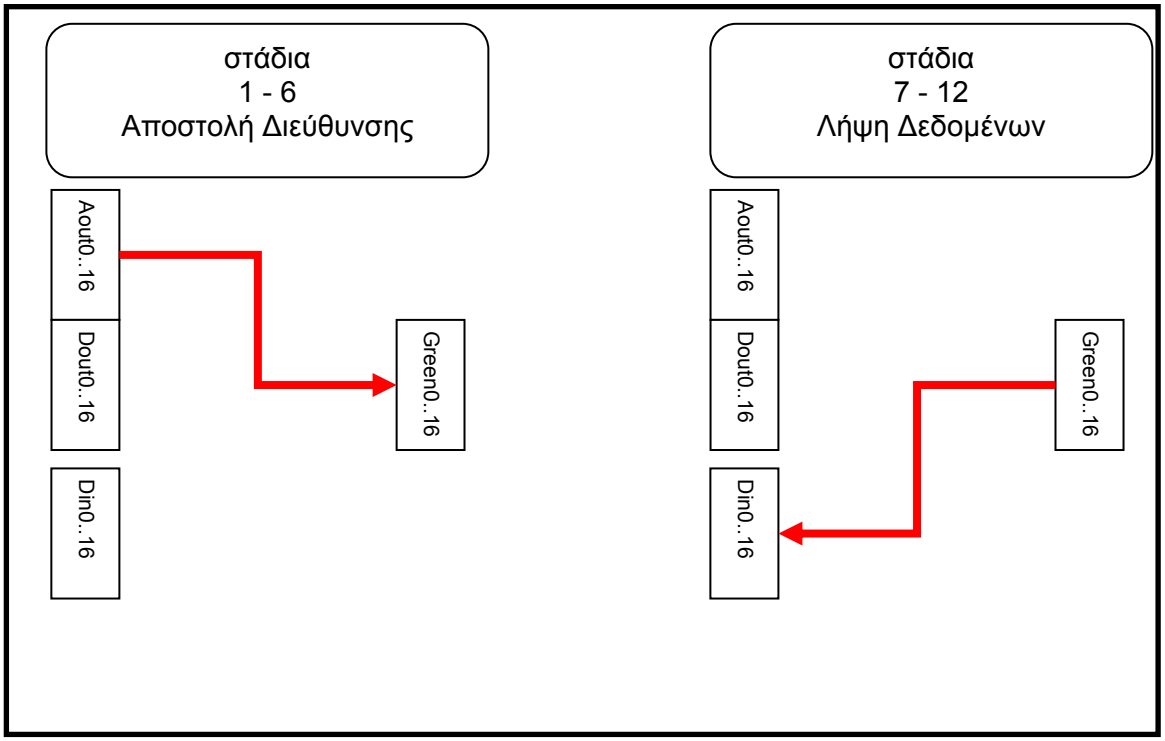
1. Η DIO στέλνει μια πληροφορία 32bit με δομή [data16 **0 1** address14] προς την IFC.
2. Το ATF2500C κάνει χαμηλή τη γραμμή ~AOE και επιτρέπει στα HC571, IC1 και IC2 να στείλουν την πληροφορία από τις P00, P01 στον δίαυλο του SARCOS.
3. Το ATF2500C κάνει χαμηλή τη γραμμή ~WRITE/READ οπότε η BusReV4 προετοιμάζεται να αποθηκεύει πληροφορία (δεν αποθηκεύεται καμία πληροφορία προς το παρόν).
4. Το ATF2500C κάνει χαμηλή τη γραμμή ~ALE οπότε η BusRepV4 αντιλαμβάνεται την πληροφορία ως διεύθυνση (προπαρασκευαστικό στάδιο)
5. Το ATF2500C κάνει υψηλή τη γραμμή ~ALE αποθηκεύει την διεύθυνση στα latch διευθύνσεων.
6. Το ATF2500C κάνει υψηλή τη γραμμή ~AOE και θέτει τα HC571, IC1 και IC2 σε κατάσταση υψηλής εμπέδησης.
7. Το ATF2500C κάνει χαμηλή τη γραμμή ~DOE και επιτρέπει στα HC571, IC3 και IC4 να στείλουν την πληροφορία από τις P06, P07 στον δίαυλο του SARCOS
8. Το ATF2500C κάνει χαμηλή τη γραμμή ~DS οπότε η BusRepV4 αντιλαμβάνεται την πληροφορία ως δεδομένα (προπαρασκευαστικό στάδιο)
9. Το ATF2500C κάνει υψηλή τη γραμμή ~DS οπότε η BusrepV4 αποθηκεύει την πληροφορία στην διεύθυνση που προηγουμένως είχε κρατήσει.
10. Το ATF2500C κάνει υψηλή τη γραμμή ~WRITE/READ
11. Το ATF2500C κάνει υψηλή τη γραμμή ~DOE και θέτει τα HC571, IC3 και IC4 σε κατάσταση υψηλής εμπέδησης και τερματίζεται ο κύκλος.



Σχήμα 4-5 Ροή πληροφορίας κατά τον κύκλο αποστολής δεδομένων στο SARCOS .

Λειτουργία ανάγνωσης δεδομένων από το SARCOS

1. Η DIO στέλνει πληροφορία 32bit με δομή [NoC16 **1 0** address14] προς την IFC.
2. Όταν όλες οι γραμμές ~OBF γίνουν χαμηλές σημαίνει πως και οι τέσσερις θύρες έχουν χρήσιμη πληροφορία. Τότε ο συνδυασμός 1 0 στα δυο πλέον σημαντικά ψηφία της διεύθυνσης ενεργοποιεί τον κύκλο.
3. Το ATF2500C κάνει χαμηλή τη γραμμή ~AOE και επιτρέπει στα HC571, IC1 και IC2 να στείλουν την πληροφορία από τις P00, P01 στον δίαυλο του SARCOS
4. Το ATF2500C κάνει χαμηλή τη γραμμή ~WRITE/READ οπότε η BusReV4 προετοιμάζεται να αποθηκεύει πληροφορία (δεν αποθηκεύεται καμία πληροφορία προς το παρόν).
5. Το ATF2500C κάνει χαμηλή τη γραμμή ~ALE οπότε η BusRepV4 αντιλαμβάνεται την πληροφορία ως διεύθυνση (προπαρασκευαστικό στάδιο)
6. Το ATF2500C κάνει υψηλή τη γραμμή ~ALE αποθηκεύει την διεύθυνση στα latch διευθύνσεων.
7. Το ATF2500C κάνει υψηλή τη γραμμή ~AOE και θέτει τα HC571, IC1 και IC2 σε κατάσταση υψηλής εμπέδησης.
8. Το ATF2500C κάνει χαμηλή τη γραμμή ~DOE και επιτρέπει στα HC571, IC3 και IC4 να στείλουν την πληροφορία από τις P06, P07 στον δίαυλο του SARCOS
9. Το ATF2500C κάνει χαμηλή τη γραμμή ~DS οπότε η BusRepV4 αντιλαμβάνεται την πληροφορία ως δεδομένα (προπαρασκευαστικό στάδιο)
10. Το ATF2500C κάνει υψηλή τη γραμμή ~DS οπότε η BusrepV4 αποθηκεύει την πληροφορία στην διεύθυνση που προηγουμένως είχε κρατήσει.
11. Το ATF2500C κάνει υψηλή τη γραμμή ~WRITE/READ
12. Το ATF2500C κάνει υψηλή τη γραμμή ~DOE και θέτει τα HC571, IC3 και IC4 σε κατάσταση υψηλής εμπέδησης και τερματίζεται ο κύκλος.



Σχήμα 4-6 Ροή πληροφορίας κατά τον κύκλο λήψης δεδομένων από το SARCOS.

4.2.Η κάρτα παρακολούθησης του διαύλου BackPlane (SNIFFER)

4.2.1.Η ανάγκη κατασκευής της κάρτας.

Το πρώτο σημαντικό ερώτημα που παρουσιάστηκε, ήταν αυτό της σωστής λειτουργία του ψηφιακού διαύλου στο σύνολο του, δηλαδή, host computer ←→ Greenspring Bus ←→ BusRepV4 ←→BackPlane Bus←→ AJC. Η αδυναμία άμεσης απάντησης στο προηγούμενο ερώτημα, οφείλεται στο ότι **δεν υπάρχει δυνατότητα εγγραφής ψηφιακής πληροφορίας σε μια διεύθυνση μνήμης και άμεσης ανάγνωσης της πληροφορίας από την θέση αυτή**. Προφανώς ένας τέτοιος έλεγχος είναι εκ των ουκ άνευ, ειδικά στην προκείμενη περίπτωση, όπου η κάρτα BusRepV4 έχει καταπονηθεί και επισκευαστεί.

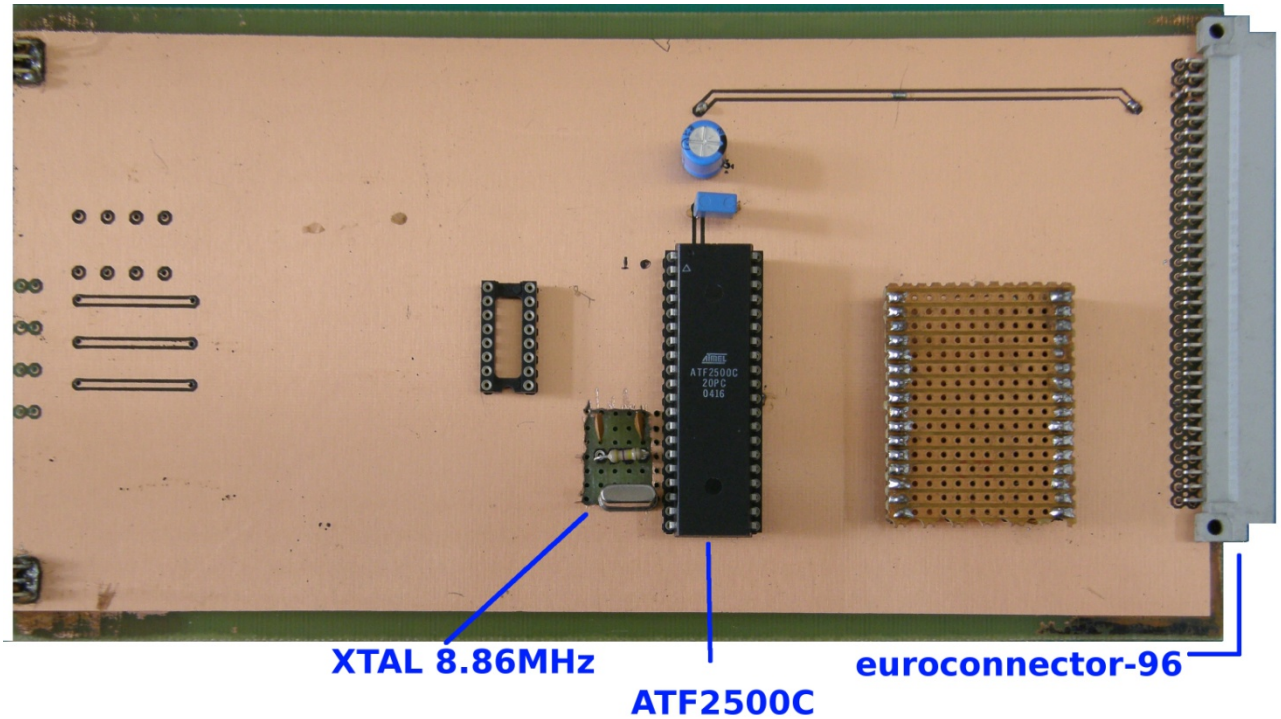
Για τον λόγο αυτό κατασκευάστηκε η κάρτα παρακολούθησης του διαύλου (SNIFFER), η οποία βασίζεται στο CPLD ATF2500C. Η κάρτα μέσω ενός συνδετήρα euroconnector 96pin, μπορεί να τοποθετηθεί στην θέση μιας κάρτας AJC. Παρέχει μια διεύθυνση εγγραφής και ανάγνωσης 16bit, στην οποία μπορούν να εγγραφούν δεδομένα και στην συνέχεια να αναγνωστούν και να εξεταστούν ως προς την ορθότητα τους. **Κάθε διαφορά μεταξύ των απεσταλμένων και των ληφθέντων δεδομένων σηματοδοτεί κάποιο πρόβλημα στον ψηφιακό δίαυλο**.

Για τον έλεγχο του διαύλου, η κάρτα SNIFFER τοποθετήθηκε διαδοχικά στις θέσεις όλων των AJC και εκτελέστηκαν δοκιμαστικοί κύκλοι εγγραφής και ανάγνωσης κατά τους οποίους φάνηκε ότι η ροή ψηφιακής πληροφορίας (έως και πριν τις AJC) λαμβάνει χώρα απρόσκοπτα. Στην συνέχεια η κάρτα SNIFFER τοποθετήθηκε μόνιμα στη θέση 11, η οποία αντιστοιχεί στον 10° Β.Ε., και χρησιμοποιείται για έλεγχο της BusRepV4 κάθε φορά που αρχικοποιείται το SARCOS.

Στη πορεία έγινε φανερό ότι, κυρίως λόγω έλλειψης real time λειτουργικού, δεν υπήρχε δυνατότητα γνώσης του χρόνου του βρόχου ελέγχου. Η άγνοια αυτή δεν επέτρεπε την διερεύνηση και τον έλεγχο της δυναμικής του ρομποτικού συστήματος.

Για τον λόγο αυτό, στην κάρτα SNIFFER υλοποιήθηκε και ένας ψηφιακός μετρητής (counter) 31bit στα 8.86MHz. Ο counter φαίνεται ως θέση μνήμης η οποία μπορεί να αρχικοποιείται (μηδενισμός), και να διαβάζεται. Το εύρος των 31bit παρέχουν την δυνατότητα μέτρησης έως $2^{31} \times (8.86MHz)^{-1} = 242.38\,\mathrm{s}$.

Με τον τρόπο αυτό πλέον υπάρχει η δυνατότητα γνώσης του χρόνου του βρόχου. **Η τιμή που διαβάζεται από τον counter αναφέρεται στο προηγούμενο στιγμιότυπο του βρόχου ελέγχου.** Οι μετρήσεις που έγινα θα παρατεθούν στο κεφάλαιο των πειραμάτων.



Σχήμα 4-7 Κάρτα SNIFFER.

4.2.2.Διευθύνσεις εντός της κάρτας

Οι λειτουργίες της κάρτας SNIFFER υλοποιούνται μέσω εγγραφών και αναγνώσεων, σε διευθύνσεις εντός του χώρου διευθύνσεων του SARCOS. Η τοποθετείται σε μια υποδοχή κάρτας AJC (έως στιγμής αντικαθιστά την κάρτα #11) και κληρονομεί την διεύθυνση της. Οι διευθύνσεις και οι λειτουργίες τους παρατίθενται στον Πίνακα 4-3.

Πίνακας 4-3 Διευθύνσεις εντός της κάρτας SNIFFER.

διεύθυνση	RW	περιγραφή
0	RW	**Register 16bit:** εγγραφής και ανάγνωσης χρήσιμη για έλεγχο του ψηφιακού διαύλου.
1	R	**counter LSB 16bit + lock:** μια ανάγνωση επιστρέφει τα λιγότερο σημαντικά bit του counter και κλειδώνει τον counter.
	W	**reset + unlock:** Μηδενισμός και εκκίνηση του counter
2	R	**counter MSB 15bit + lock** μια ανάγνωση επιστρέφει τα περισσότερο σημαντικά bit του counter και κλειδώνει τον counter
	W	**reset + unlock:** Μηδενισμός και εκκίνηση του counter
προσοχή: Κατά την ανάγνωση του counter, αυτός κλειδώνει ώστε να μην υπάρξουν σφάλματα . Για να ξεκλειδώσει πρέπει να γίνει μια εγγραφή στις θέσεις 1 ή 2		

Το χρονικό διάστημα που παρήλθε, μεταξύ μηδενισμού και μέτρησης, δίνεται από την σχέση

$$\Delta t = \frac{counter(LO) + counter(HI) \times 65536}{8.86 \times 10^6} \text{s}$$

5 Αλγόριθμοι για Βαθμονόμηση των Αισθητήρων

Στο παρόν κεφάλαιο, παρατίθενται και περιγράφονται μαθηματικά οι αλγόριθμοι και η μεθοδολογία που χρησιμοποιούνται στην παρούσα εργασία. Για λόγους σαφήνειας, οι αλγόριθμοι αναλύονται ξεχωριστά από τον κώδικα που τους υλοποιεί. Ο κώδικας αναλύεται στο Κεφάλαιο 6, ενώ πειραματικά αποτελέσματα παρατίθενται στο Κεφάλαιο 7.

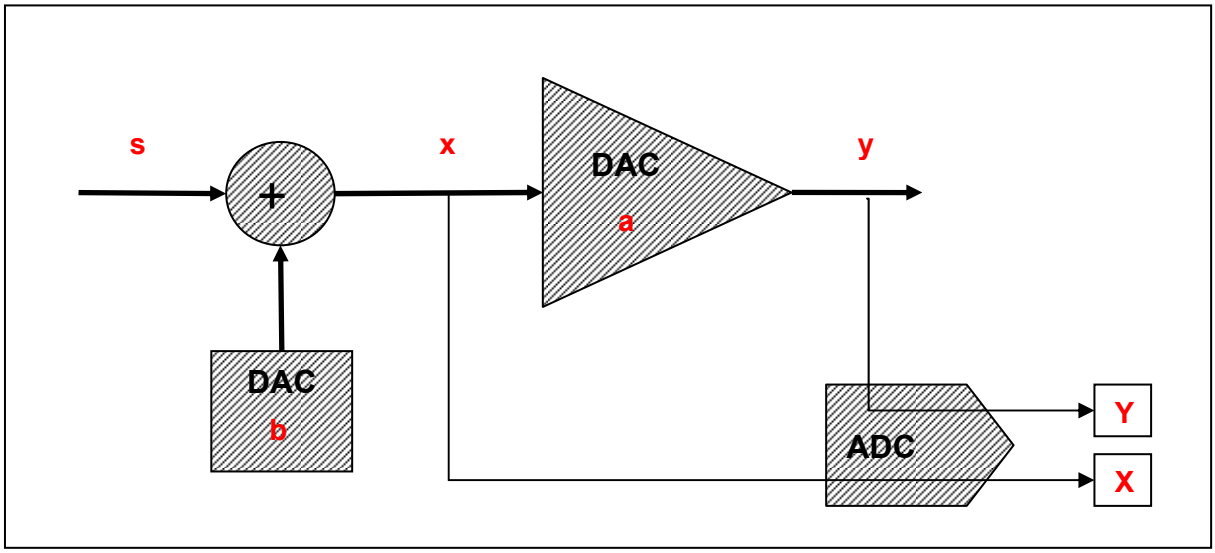
5.1 Μέθοδοι Βαθμονόμησης των Αισθητήρων

Το αναλογικό σήμα, s , που λαμβάνεται από ένα αισθητήρα, θέσης ή ροπής, συνήθως δεν βρίσκεται εντός επιθυμητών ορίων τάσης. Τις περισσότερες φορές είναι απαραίτητη η κλιμάκωση και η μετατόπιση του σήματος, έτσι ώστε να βρεθεί εντός των λειτουργικών ορίων τάσης των επόμενων βαθμίδων επεξεργασίας.

Μια τέτοια διαδικασία καλείται βαθμονόμηση (calibration) και υλοποιείται, για κάθε αισθητήρα ξεχωριστά σε ένα στάδιο βαθμονόμησης. Σκοπός της μεθόδου είναι η βαθμονόμηση των αισθητήρων, χωρίς να απαιτείται χειρονακτική μέτρηση τάσεων πάνω στις κάρτες AJC, αλλά μόνο με την χρήση των παρατηρήσιμων μεγεθών, ψηφιακά μετρουμένων μέσω του ADC.

Το στάδιο βαθμονόμησης αποτελείται από μια ψηφιακά ελεγχόμενη πόλωση (offset) και ένα ψηφιακά ελεγχόμενο κέρδος (gain), όπως στο Σχήμα 5-1, και σκοπός του είναι να φέρει το σήμα του αισθητήρα στο επιθυμητό εύρος τάσης [-10 … +10] V. Για να γίνει αυτό, το κέρδος και η πόλωση πρέπει να ρυθμιστούν κατάλληλα.

Συγκεντρωτικές πληροφορίες για όλους τους Β.Ε. παρατίθενται στο τέλος του κεφαλαίου



Σχήμα 5-1 Πρώτη βαθμίδα (στάδιο) βαθμονόμησης του σήματος.

Για να μπορέσει να λάβει χώρα η βαθμονόμηση, πρέπει να διαβαστεί πληροφορία (σήμα) από το στάδιο βαθμονόμησης. Το σήμα αυτό διαβάζεται μέσω του μετατροπέα από αναλογικό σε ψηφιακό (ADC).

5.1.1 Μοντέλο κυκλώματος βαθμονόμησης

Το μοντέλο του κυκλώματος βαθμονόμησης του Σχήματος 5.1 είναι

$$\begin{cases} y = s + B \cdot (b - b_0) \\ x = a \cdot A \cdot y = a \cdot A \cdot (s + B \cdot (b - b_0)) \end{cases} \qquad \textbf{εξ. 5-1}$$

Τα σύμβολα που χρησιμοποιούνται περιγράφονται στον Πίνακα 5-1.

Πίνακας 5-1 Σύμβολα του μοντέλου κυκλώματος βαθμονόμησης.

σύμβολο	επεξήγηση	παρατηρήσεις
s	σήμα απευθείας από τον αισθητήρα	
y	σήμα μετά τον αθροιστή	
b , b^k	ψηφιακή τιμή του offset	
b_0	ψηφιακή τιμή του offset του μηδενός	περίπου 2048*
B	σταθερά μετατροπής του offset από ψηφιακό σε αναλογικό	περίπου 20/4096*
x	σήμα μετά το κέρδος	
a , a^k	ψηφιακή τιμή του κέρδους	
A	σταθερά μετατροπής του κέρδους από ψηφιακός σε αναλογικό	περίπου 10/4096*
*Σύμφωνα με το manual : Το διάστημα (0 … 4095), του ψηφιακού offset, αντιστοιχεί στο διάστημα (-10 … +10), του αναλογικού offset Το διάστημα (0…4095) του ψηφιακού gain αντιστοιχεί στο διάστημα (0…10), του αναλογικού gain		

5.1.2 Μοντέλο Μετατροπέα από Αναλογικό σε Ψηφιακό (ADC).

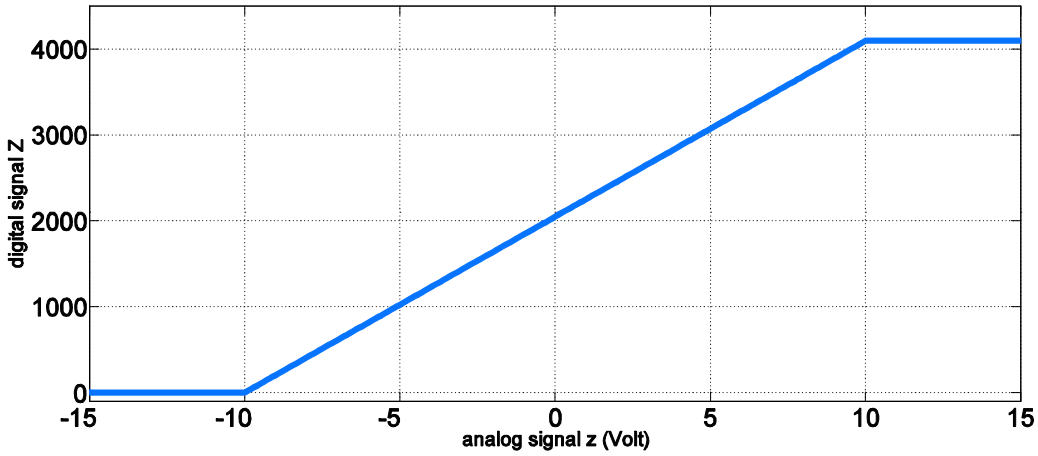
Το μοντέλο του ADC, είναι κατά τμήματα γραμμικό και περιγράφεται από την

$$Z = \begin{cases} 0, \varepsilon\acute{\alpha}\nu, C \cdot (z + d) < 0 \\ C \cdot (z + d) \\ 4095, \varepsilon\acute{\alpha}\nu, C \cdot (z + d) > 4095 \end{cases} \qquad \textbf{εξ. 5-2}$$

Τα σύμβολα που χρησιμοποιούνται περιγράφονται στον Πίνακα 5-2.

Πίνακας 5-2 Σύμβολα του μοντέλου ADC.

z	αναλογική τιμή σήματος στην είσοδο του ADC	
Z	ψηφιακή τιμή σήματος στην έξοδο του ADC	
C	σταθερά αναλογίας της μετατροπής από αναλογικό σε ψηφιακό*	περίπου 4096/20*
d	αναλογική σταθερά offset της μετατροπής από αναλογικό σε ψηφιακό*	περίπου 10Volt*
*Σύμφωνα με το manual : Το διάστημα (-10 … 10), της αναλογικής εισόδου z, αντιστοιχίζεται στο διάστημα (0 …4095) της ψηφιακής εξόδου Z		



Σχήμα 5-2 Απόκριση ADC μετατροπέα από αναλογικό σε ψηφιακό.

Υπάρχουν δυο σημαντικά σημεία τα οποία μπορεί να εισάγουν σφάλματα στην διαδικασία βαθμονόμησης.

1. Οι τιμές των παραμέτρων A, B, b_0, C, d δεν είναι επακριβώς γνωστές γιατί τα κυκλώματα υπόκεινται σε φθορές λόγω γήρανσης. Οπότε, μια μέθοδος βαθμονόμησης δεν πρέπει να στηρίζεται σε κάποια εκ των προτέρων εκτίμηση των παραμέτρων αυτών.
2. Ο ADC μπορεί να ψαλιδίσει το ψηφιακό σήμα εξόδου οπότε η μέθοδος βαθμονόμησης πρέπει να εξασφαλίζει ότι το σήμα βρίσκεται εντός της γραμμικής περιοχής και όχι στις περιοχές κορεσμού.

5.1.3 Μέθοδος ενός καναλιού

Η μέθοδος αυτή κάνει χρήση μόνο του καναλιού x. Χρησιμοποιεί μια επαναληπτική μέθοδο κατά την οποία **εναλλάξ κεντράρει και διευρύνει** την ψηφιακή έξοδο του ADC $\left[X_{\max}, X_{\min}\right]$ έτσι ώστε τελικά να συμπέσει με το επιθυμητό διάστημα.

Εάν δεν υπήρχαν οι μη γραμμικότητες, λόγω του ADC, η μέθοδος αυτή θα ήταν μια μέθοδος ενός σταδίου, αφού ουσιαστικά είναι ένας απλός γραμμικός μετασχηματισμός. Για λόγους που θα φανούν στην συνέχεια υποτίθεται πως η βαθμονόμηση είναι μια διαδικασία πολλών σταδίων.

Έστω ότι λαμβάνει χώρα μια επαναληπτική μέθοδος βαθμονόμησης, η οποία βρίσκεται στην k επανάληψη κατά την οποία είναι γνωστά τα a^k, b^k (Πίνακας 5-1) και πρέπει να υπολογιστούν τα a^{k+1}, b^{k+1}.

Έστω ότι $X_{\max}, X_{\min}$ η ψηφιακή έξοδος X του ADC, όταν η έξοδος s του αισθητήρα λαμβάνει τις ακρότατες τιμές $s_{\max}, s_{\min}$. Τότε ισχύουν οι

$$X_{\max}^{k} = C \cdot (a^{k} \cdot A \cdot (s_{\max} + B \cdot (b^{k} - b_{0})) + d) \qquad \textbf{εξ. 5-3}$$

$$X_{\min}^{k} = C \cdot (a^{k} \cdot A \cdot (s_{\min} + B \cdot (b^{k} - b_{0})) + d) \qquad \textbf{εξ. 5-4}$$

Έστω ότι το διάστημα $\left[X_{\min}^{k}, X_{\max}^{k}\right]$ δεν είναι το επιθυμητό, αλλά απαιτείται η ψηφιακή έξοδος να βρίσκεται στο διάστημα $\left[X_{\min}^{des}, X_{\max}^{des}\right]$ και έστω ότι αυτό συμβαίνει όταν $a = a^{des}, b = b^{des}$.

Τότε θα ισχύει

$$X_{\max}^{des} = C \cdot (a^{des} \cdot A \cdot (s_{\max} + B \cdot (b^{des} - b_{0})) + d) \qquad \textbf{εξ. 5-5}$$

$$X_{\min}^{des} = C \cdot (a^{des} \cdot A \cdot (s_{\min} + B \cdot (b^{des} - b_{0})) + d) \qquad \textbf{εξ. 5-6}$$

Αφαιρώντας την εξ.5.3 από την εξ.5.4 , και την εξ.5.5 από την εξ.5.6 ,προκύπτει

$$\Delta^{k} = X_{\max}^{k} - X_{\min}^{k} = C \cdot a^{k} \cdot A \cdot (s_{\max} - s_{\min}) \qquad \textbf{εξ. 5-7}$$

$$\Delta^{des} = X_{\max}^{des} - X_{\min}^{des} = C \cdot a^{des} \cdot A \cdot (s_{\max} - s_{\min}) \qquad \textbf{εξ. 5-8}$$

Ορίζεται το νέο κέρδος μέσω της

$$a^{k+1} = a^{k} \cdot \frac{\Delta^{des}}{\Delta^{k}} \qquad \textbf{εξ. 5-9}$$

και ισχύει

$$\Delta^{k} = \Delta^{des} \Leftrightarrow a^{k+1} = a^{k} \qquad \textbf{εξ. 5-10}$$

Οπότε καθορίζεται και το **σημείο ισορροπίας της διαδικασίας διεύρυνσης.**

Στην περίπτωση που το διάστημα $\left[X_{\max}, X_{\min}\right]$ είναι γνωστό χωρίς σφάλματα, ή προηγούμενη διαδικασία αποτελείται από ένα μόνο βήμα.

Πράγματι,

$$\Delta^{k+1} = X_{\max}^{k+1} - X_{\min}^{k+1} = C \cdot a^{k+1} \cdot A \cdot (s_{\max} - s_{\min}) \qquad \textbf{εξ. 5-11}$$

οπότε

$$\Delta^{k+1} = C \cdot a^{k} \cdot \frac{\Delta^{des}}{\Delta^{k}} \cdot A \cdot (s_{\max} - s_{\min}) = \frac{\Delta^{des}}{\Delta^{k}} \cdot \Delta^{k} = \Delta^{des} \qquad \textbf{εξ. 5-12}$$

Προσθέτοντας κατά μέλη τις εξισώσεις 5-3, 5-4 και τις 5-5, 5-6 προκύπτει

$$X_{\max}^{k} + X_{\min}^{k} = C \cdot (a^{k} \cdot A \cdot ((s_{\max} + s_{\min}) + B \cdot (b^{k} - b_{0})) + d) \qquad \textbf{εξ. 5-13}$$

$$X_{\max}^{des} + X_{\min}^{des} = C \cdot (a^{des} \cdot A \cdot ((s_{\max} + s_{\min}) + B \cdot (b^{des} - b_{0})) + d) \qquad \textbf{εξ. 5-14}$$

Αφαιρώντας κατά μέλη τις εξισώσεις 5-13 και 5-14 προκύπτει

$$\left(X_{\max}^{k} + X_{\min}^{k}\right) - \left(X_{\max}^{des} + X_{\min}^{des}\right) = C \cdot (a^{k} - a^{des}) \cdot A \cdot (s_{\max} + s_{\min}) + C \cdot A \cdot B \cdot (b^{k} - b^{des}) \qquad \textbf{εξ. 5-15}$$

Οι ποσότητες

$$X_{center}^{k} = \frac{X_{\max}^{k} + X_{\min}^{k}}{2}, \quad X_{center}^{des} = \frac{X_{\max}^{des} + X_{\min}^{des}}{2} \qquad \textbf{εξ. 5-16}$$

καθορίζουν τα κέντρα των διαστημάτων $\left[X_{\min}^{k}, X_{\max}^{k}\right]$, και $\left[X_{\min}^{des}, X_{\max}^{des}\right]$ αντίστοιχα, οπότε η εξ 5.15 γίνεται

$$X_{center}^{k} - X_{center}^{des} = \frac{C}{2} \cdot (a^{k} - a^{des}) \cdot A \cdot (s_{\max} + s_{\min}) + \frac{C \cdot A \cdot B}{2} \cdot (b^{k} - b^{des}) \qquad \textbf{εξ. 5-17}$$

Εάν οι ποσότητες $(s_{\max} + s_{\min})$ και $C \cdot A \cdot B$ ήταν γνωστές, τότε η εξίσωση 5-17 θα λυνόταν ως προς b^{des}. Η ποσότητα $C \cdot A \cdot B$ μπορεί να προσδιοριστεί από ανεξάρτητο πείραμα που θα περιγραφεί αργότερα, όμως, η ποσότητα $(s_{\max} + s_{\min})$ παραμένει άγνωστη και η μόνη περίπτωση που δεν επηρεάζει την λύση είναι όταν $a^{k} = a^{des}$.

Στην (ειδική) περίπτωση όπου $a^{k} = a^{des}$ η εξίσωση 5-17 λύνεται και δίνει

$$b^{k+1} = 2 \cdot \frac{X_{center}^{des} - X_{center}^{k}}{C \cdot A \cdot B} + b^{k} \qquad \textbf{εξ. 5-18}$$

Τότε το σημείο ισορροπίας της διαδικασίας καθορίζεται μονοσήμαντα από την σχέση

$$X_{center}^{k} = X_{center}^{des} \Leftrightarrow b^{k+1} = b^{k} \qquad \textbf{εξ. 5-19}$$

το οποίο είναι και **σημείο ισορροπίας της διαδικασίας κεντραρίσματος.**

Στην περίπτωση που το διάστημα $\left[X_{\max}, X_{\min}\right]$ είναι γνωστό χωρίς σφάλματα, ή προηγούμενη διαδικασία αποτελείται από ένα μόνο βήμα. Πράγματι από την εξ 5-17 για τις επαναλήψεις k και k+1 προκύπτει

$$X_{center}^{k+1} - X_{center}^{k} = \frac{C}{2} \cdot (a^{k+1} - a^{k}) \cdot A \cdot (s_{\max} + s_{\min}) + \frac{C \cdot A \cdot B}{2} \cdot (b^{k+1} - b^{k}) \qquad \textbf{εξ. 5-20}$$

και λόγω της εξίσωσης 5-16 για k και k+1

$$X_{center}^{k+1} - X_{center}^{k} = X_{center}^{des} - X_{center}^{k} \qquad \textbf{εξ. 5-21}$$

που οδηγεί στην

$$X_{center}^{k+1} = X_{center}^{des} \qquad \textbf{εξ. 5- 22}$$

Όμως η ψηφιακή έξοδος X, υπόκειται σε κορεσμό (saturation), εάν το αναλογικό σήμα x υπερβεί τα όρια [-10 … +10]V. Ο κορεσμός αυτός μπορεί να προκύψει ανά πάσα στιγμή και μπορεί να οφείλεται σε ακραίες τιμές των παραμέτρων a,b, εισάγει δε σφάλμα στην διαδικασία οπότε πρέπει να μελετηθεί η ευαισθησία της μεθόδου.

Έστω ότι η πραγματική έξοδος του ADC είναι

$$\begin{cases} \hat{X}_{\max} = X_{\max} - e_{\max} \\ \hat{X}_{\min} = X_{\min} + e_{\min} \end{cases}, e_{\max} > 0, e_{\min} > 0 \qquad \textbf{εξ. 5-23}$$

όπου $e_{\max}, e_{\min}$ τα σφάλματα λόγω κορεσμού.

Ορίζονται, σε αντιστοιχία με τις εξισώσεις 5-7, 5-15 και 5-16, τα :

$$\begin{aligned} \hat{\Delta}^k &= \hat{X}_{\max}^k - \hat{X}_{\min}^k \\ \hat{X}_{center}^k &= \frac{\hat{X}_{\max}^k + \hat{X}_{\min}^k}{2} = X_{center}^k + \frac{e_{\max} + e_{\min}}{2} \end{aligned} \qquad \textbf{εξ. 5-24}$$

Πλέον η μέθοδος εφαρμόζεται για τις βαρύμενες μεταβλητές. $(\hat{\bullet})$

Η εξίσωση 5-9 γίνεται

$$a^{k+1} = a^k \cdot \frac{\Delta^{des}}{\hat{\Delta}^k} \qquad \textbf{εξ. 5-25}$$

και είναι η πλέον ευαίσθητη στα σφάλματα. Αυτό φαίνεται εύκολα εάν υποτεθεί ότι, λόγω εντελώς εσφαλμένων a,b, υπάρχει πλήρης κορεσμός της ψηφιακής εξόδου, δηλαδή, $X_{\max} = X_{\min} = 0$ ή $X_{\max} = X_{\min} = 4095$.

Τότε, $\Delta^k = 0 \Rightarrow a^{k+1} = \infty$.

Γενικότερα, εάν $\hat{\Delta}^k < \Delta^k$ τότε $a^{k+1} > a^k$, το οποίο θα αυξήσει περαιτέρω τα σφάλματα οπότε η διαδικασία δεν θα συγκλίνει. Για να εφαρμοστεί η εξίσωση 5-23 χωρίς προβλήματα, πρέπει να εξασφαλιστεί ότι το ψηφιακό σήμα δε είναι σε πλήρη κορεσμό με κατάλληλη επιλογή των αρχικών a^0, b^0.

Η εξίσωση 5-21 γίνεται

$$X_{center}^{k+1} - X_{center}^k = X_{center}^{des} - \hat{X}_{center}^k \qquad \textbf{εξ. 5-26}$$

και λόγω της εξ 5-22 τελικά

$$X_{center}^{k+1} = X_{center}^{des} - \hat{X}_{center}^{k} + X_{center}^{k} = X_{center}^{des} - \frac{e_{\max} - e_{\min}}{2} \qquad \textbf{εξ. 5-27}$$

Η εξίσωση 5-26 περιγράφει ότι το κέντρο θα μετατοπιστεί, αλλά το πρόσημο της μετατόπισης εξαρτάται από τις δυο άγνωστες ποσότητες $e_{\max}, e_{\min}$. Υπάρχουν τρείς περιπτώσεις

1. $e_{\max} > 0, e_{\min} = 0$.

   Τότε ο αισθητήρας εμφανίζει κορεσμό στην πάνω περιοχή . Η εξίσωση 5-25 γίνεται

   $$X_{center}^{k+1} = X_{center}^{des} - \frac{e_{\max}}{2}$$

   και αναγκάζει το κέντρο να κινηθεί χαμηλότερα από το επιθυμητό. Αυτό όμως αυξάνει την πιθανότητα ο αισθητήρας να τεθεί εντός ορίων, οπότε στην επόμενη επανάληψη να κεντραριστεί σωστά.

2. $e_{\max} = 0, e_{\min} > 0$.

   Τότε ο αισθητήρας εμφανίζει κορεσμό στην κάτω περιοχή . Η εξίσωση 5-25 γίνεται

   $$X_{center}^{k+1} = X_{center}^{des} + \frac{e_{\min}}{2}$$

   και αναγκάζει το κέντρο να κινηθεί υψηλότερα από το επιθυμητό. Αυτό όμως αυξάνει την πιθανότητα ο αισθητήρας να τεθεί εντός ορίων, οπότε στην επόμενη επανάληψη να κεντραριστεί σωστά.

3. $e_{\max} > 0, e_{\min} > 0$.

   Τότε ο αισθητήρας παρουσιάζει αμφίπλευρο κορεσμό όποτε δεν είναι δυνατό το κεντράρισμα.

Βάσει των παραπάνω κατασκευάζεται μια επαναληπτική διαδικασία, η k επανάληψη της οποίας περιλαμβάνει τα εξής στάδια.

1. ο Β.Ε. κινείται στις τερματικές του θέσεις, οπότε καθορίζονται
   1.1. το διάστημα $\left[X_{\min}^{k}, X_{\max}^{k}\right]$
   1.2. το εύρος $\Delta^{k} = X_{\min}^{k} - X_{\max}^{k}$
   1.3. το κέντρο $X_{center}^{k} = \frac{X_{\max}^{k} + X_{\min}^{k}}{2}$
2. συγκρίνονται τα $X_{center}^{k}, X_{center}^{des}$
   2.1. εάν $X_{center}^{k} \neq X_{center}^{des}$, τότε ο **αισθητήρας κεντράρεται**
      2.1.1. $a^{k+1} = a^{k}$,οπότε, $\Delta^{k+1} = \Delta^{k}$
      2.1.2. $b^{k+1} = 2 \cdot \frac{X_{center}^{k} - X_{center}^{des}}{C \cdot A \cdot B} + b^{k}$, οπότε $X_{center}^{k+1} = X_{center}^{des} - \frac{e_{\max} - e_{\min}}{2}$

2.2. εάν $X_{center}^{k} = X_{center}^{des}$, **τότε ρυθμίζεται το εύρος**

2.2.1. $b^{k+1} = b^{k}$

2.2.2. $a^{k+1} = a^{k} \cdot \frac{\Delta^{req}}{\Delta^{k}}$,

Οπότε $\Delta^{k+1} = \Delta^{des}$

και $X_{center}^{k+1} = \frac{C}{2} \cdot (a^{k+1} - a^{k}) \cdot A \cdot (s_{\max} + s_{\min}) + X_{center}^{des}$

5.1.4 Η εύρεση του γινομένου CAB

Η εύρεση του γινομένου $C \cdot A \cdot B$ γίνεται εφικτή εάν το σήμα s του αισθητήρα παραμείνει σταθερό, οπότε $\delta s = 0$, ενώ μεταβάλλεται το ψηφιακό offset b σε όλο το δυνατό εύρος [0…4095].

Γενικά ισχύει πώς

$$\delta X = \delta(C \cdot (a \cdot A \cdot (s + B \cdot (b - b_0)) + d)) = \alpha \cdot C \cdot A \cdot B \cdot \delta b \qquad \textbf{εξ. 5.-28}$$

Με εφαρμογή ελαχίστων τετραγώνων μεταξύ b, X το γινόμενο $a \cdot C \cdot A \cdot B$ μπορεί να υπολογιστεί. Από αυτό, επειδή το a είναι γνωστό, υπολογίζεται το γινόμενο $C \cdot A \cdot B$, εφόσον πληρούνται οι ακόλουθες προϋποθέσεις.

1. Ο βραχίονας να είναι ακίνητος και σε τέτοια θέση ώστε το σήμα X να μην είναι σε κορεσμό, για μια αξιόλογη περιοχή τιμών του b. Η θέση αυτή εξαρτάται από το είδος του αισθητήρα που βαθμονομείται.
   a. Εάν είναι αισθητήρας θέσης μια καλή θέση είναι στο μέσο του φυσικού εύρους κίνησης του Β.Ε.
   b. Εάν είναι αισθητήρας ροπής τότε πρέπει να είναι στην θέση μηδενικής ροπής.
2. Τα ζεύγη (b, X) με $X = 0$ ή $X = 4095$ πρέπει να παραλείπονται από την μέθοδο των ελάχιστων τετράγωνων γιατί είναι ύποπτα για κορεσμό.
3. Το ψηφιακό κέρδος α πρέπει να έχει τέτοια (γνωστή) τιμή ώστε το σήμα X, να μην έρχεται γρήγορα σε κορεσμό, αλλά ταυτόχρονα να έχει και αξιόλογο εύρος. Μια τιμή μεταξύ 400 και 2000 είναι ικανοποιητική.

Η μέθοδος των ελαχίστων τετραγώνων κατ' αρχής εφαρμόζεται σε MATLAB offline σε δεδομένα που συλλέγονταν κατά την λειτουργία. Σκοπός είναι να αναπτυχθεί κώδικας που χρησιμοποιεί τα ελάχιστα τετράγωνα αυτόνομα και αποθηκεύει την τιμή του γινομένου σε συγκεκριμένο αρχείο, για μελλοντική χρήση από την ρουτίνα βαθμονόμησης.

Τα πειράματα αυτά έγιναν τόσο για τα κυκλώματα βαθμονόμησης των αισθητήρων θέσης, όσο κυκλώματα βαθμονόμησης των αισθητήρων ροπής. Τα αποτελέσματα παρατίθενται στο αντίστοιχο κεφάλαιο.

5.1.5 Μέθοδος δύο καναλιών

Σε αυτή την περίπτωση, μπορεί να διαβαστεί, μέσω του ADC, πληροφορία τόσο για το σήμα x όσο και για το σήμα y. Η μέθοδος αυτή προτείνεται από την εταιρία για την βαθμονόμηση του αισθητήρα θέσης. Αναφέρεται ως έχει, και γίνεται προσπάθεια να αναλυθεί βάσει των μοντέλων.

Εάν διαβαστεί το αναλογικό σήμα x, μέσω του ADC, λαμβάνεται το ψηφιακό σήμα X, αντίστοιχα, εάν διαβαστεί το αναλογικό σήμα y, μέσω του ADC, λαμβάνεται το ψηφιακό σήμα Y τότε ισχύουν οι εξισώσεις

$$\begin{cases} Y = C\cdot(y+d) = C\cdot(s+B\cdot(b-b_0)+d) \\ X = C\cdot(x+d) = C\cdot(a\cdot A\cdot y+d) = C\cdot(a\cdot A\cdot(s+B\cdot(b-b_0))+d) \end{cases} \qquad \text{εξ. 5-29}$$

Η μέθοδός βαθμονόμησης αποτελείται από τα εξής βήματα :

1. Τοποθέτηση του Β.Ε. με έλεγχο ανοικτού βρόχου (open loop) στην ακραία θέση με $s = s_{\min}$.
2. Μεταβολή του b έως την τιμή b_{cal} όπου $Y = 0$, οπότε ισχύει

$$C\cdot(s_{\min}+B\cdot(b_{cal}-b_0)+d) = 0 \Leftrightarrow B\cdot(b_{cal}-b_0) = -s_{\min}-d \qquad \text{εξ. 5-30}$$

3. Τοποθέτηση του Β.Ε. με έλεγχο ανοικτού βρόχου (open loop)στην ακραία θέση με $s = s_{\max}$.
4. Μεταβολή του a έως την τιμή a_{cal} όπου $X = 4095$, οπότε ισχύει

$$C\cdot(a_{cal}\cdot A\cdot(s_{\max}+B\cdot(b_{cal}-b_0))+d) = 4095 \qquad \text{εξ. 5-31}$$

Και λόγω της εξίσωσης 5-28,

$$C\cdot(a_{cal}\cdot A\cdot(s_{\max}-s_{\min}-d)+d) = 4095 \qquad \text{εξ. 5-32}$$

Οπότε λύνοντας ως προς a_{cal},

$$a_{cal} = \frac{\frac{4095}{C}-d}{A\cdot(s_{\max}-s_{\min}-d)} \qquad \text{εξ. 5-33}$$

5. πλέον η έξοδος του ψηφιακού καναλιού X είναι

$$X = C\left(\frac{\frac{4095}{C}-d}{(s_{\max}-s_{\min}-d)}(s-s_{\min}-d)+d\right) \qquad \text{εξ. 5-34}$$

Η μέθοδός αυτή δεν μπορεί να λειτουργήσει εάν $d \neq 0$ γιατί

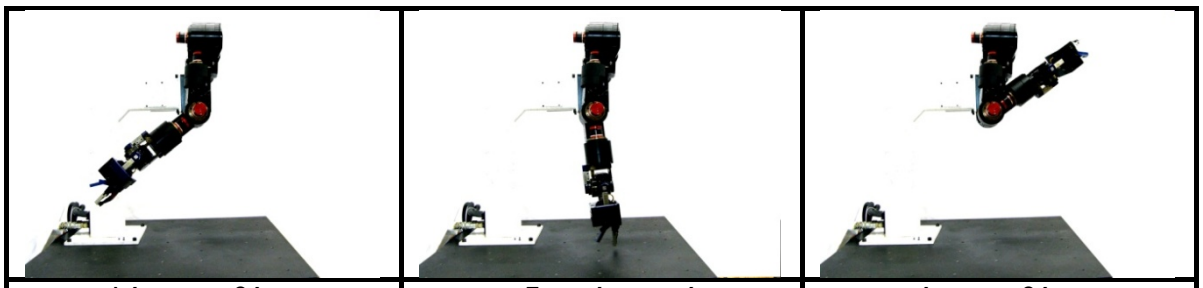
$$\begin{cases} s = s_{\max} \Rightarrow X_{\max} = 4095 \\ s = s_{\min} \Rightarrow X_{\min} \neq 0 \end{cases}$$

οπότε δεν απεικονίζεται σωστά το σήμα του αισθητήρα και μάλιστα η ελάχιστη τιμή είναι άγνωστη.

5.2 Μέθοδος Βαθμονόμησης του Αισθητήρα Θέσης

Οι αισθητήρες θέσεις που χρησιμοποιούνται στις AJC μπορεί να είναι RVDT, Hall Effect ή Potentiometer. Τα χαρακτηριστικά εξόδου των συσκευών αυτών δεν είναι τα ίδια. Ακόμα και μεταξύ όμοιων αισθητήρων μπορεί να υπάρχουν σημαντικές αποκλίσεις οπότε είναι απαραίτητη η βαθμονόμηση τους.

Για την βαθμονόμηση του αισθητήρα θέσης χρησιμοποιείται η μέθοδος μονού καναλιού, όπως αυτή περιγράφεται στην Παράγραφο 5.1.3. Εμπλέκεται το κύκλωμα βαθμονόμησης του πρώτου σταδίου (Σχήματα 3-1 και 3-2). Η βαθμονόμηση γίνεται μέσω των DAC10 (κέρδος) και DAC 23 (πόλωση). Στο Σχήμα 5-3 φαίνονται οι θέσεις που λαμβάνει κατά την κίνηση του ο βραχίονας και η σημασία τους (θεωρείται η περίπτωση που βαθμονομείται ο 4^{ος} Β.Ε.).



ελάχιστης θέσης ελάχιστης ροπής	μηδενικής ροπής	μέγιστης θέσης μέγιστης ροπής

Σχήμα 5. 3 Σημαντικές θέσεις του 4^{ου} Β.Ε. για βαθμονόμηση θέσης ή ροπής.

Το σήμα s είναι το σήμα του αισθητήρα θέσης, ενώ το σήμα X είναι το ψηφιακό του ισοδύναμο το οποίο διαβάζεται από τον DAC. Τα πειραματικά δεδομένα παρατίθενται στο Κεφάλαιο 7.

Οι Β.Ε. βαθμονομούνται, για μέγιστη ανάλυση των αισθητήρων, έτσι ώστε το εύρος της φυσικής κίνησης να αντιστοιχεί σε εύρος ψηφιακής εξόδου [0 … 4095]. Η ανάλυση προκύπτει από την σχέση

$$r_i = \frac{\Theta_i}{[X_{\max} - X_{\min}]} \qquad \textbf{εξ. 5-35}$$

Όπου Θ_i το φυσικό εύρος κίνησης του Β.Ε. i. Επειδή το φυσικό εύρος κίνησης κάθε Β.Ε. είναι διαφορετικό, η ανάλυση κάθε αισθητήρα είναι διαφορετική. Εάν είναι επιθυμητή συγκεκριμένη ανάλυση, σε όλους τους Β.Ε., δηλαδή σταθερή αναλογία $\delta\theta / \delta X$ μεταξύ ψηφιακής εξόδου και γωνίας τότε, το εύρος των ψηφιακών εξόδων των αισθητήρων $[X_{\max}, X_{\min}]$ πρέπει να είναι διαφορετικό για κάθε Β.Ε.

5.3 Μέθοδος Βαθμονόμησης του Αισθητήρα Ροπής

Για την βαθμονόμηση του αισθητήρα ροπής χρησιμοποιείται παραλλαγή της μεθόδου μονού καναλιού (Παράγραφος 5.1.3). Εμπλέκεται το κύκλωμα βαθμονόμησης του αισθητήρα ροπής (Σχήματα 3-1 και 3-2). Η βαθμονόμηση γίνεται μέσω των DAC 31, DAC29 και DAC25. Στο Σχήμα 5-3 φαίνονται οι θέσεις που λαμβάνει κατά την κίνηση του ο βραχίονας και η σημασία τους (θεωρείται η περίπτωση που βαθμονομείται ο 4^{ος} Β.Ε.).

Σημαντική αδυναμία που πρέπει να ξεπεραστεί είναι πως οι μάζες και οι ροπές αδρανείας των μελών της ρομποτικής αλυσίδας είναι άγνωστες. Στην περίπτωση που τα μεγέθη αυτά ήταν γνωστά, τότε η τοποθέτηση του βραχίονα σε θέσεις όπου, λόγω γεωμετρίας η ασκούμενη ροπή θα ήταν μέγιστη, μηδενική ή ελάχιστη, αλλά πάντα γνωστή, θα ήταν αρκετή για την βαθμονόμηση του αισθητήρα μέσω της μεθόδου μονού καναλιού.

Για να ξεπεραστεί αυτό το εμπόδιο προτείνεται η μέθοδος που ακολουθεί, η οποία χρησιμοποιεί ένα επιπλέον γνωστό βάρος, το μέγιστο το οποίο θα κληθεί να διαχειριστεί ο βραχίονας. Τα μόνα γνωστά πρέπει να είναι τα μήκη των μελών, για τον υπολογισμό των ροπών.

Έστω ότι βαθμονομείται ο αισθητήρα ροπής του συνδέσμου k. Τότε η διαδικασία αποτελείται από τα εξής βήματα:

1. Ο βραχίονας φορτίζεται με το γνωστό βάρος.
2. Ο βραχίονας τοποθετείται σε τέτοια διάταξη όπου
   2.1. Η κίνηση του συνδέσμου k να είναι τέτοια ώστε να περνάει από τις θέσεις μέγιστης, μηδενικής και ελάχιστης ροπής που, λόγω βάρους, ασκείται στο μέλος.
   2.2. Οι απαιτήσεις σε ροπή των άλλων Β.Ε. να είναι οι ελάχιστες, μηδενικές εάν είναι εφικτό.
3. Ο βραχίονας κινείται πολύ αργά, σε όλο το εύρος κίνησης οπότε το αναλογικό σήμα από τον αισθητήρα ροπής λαμβάνει τιμές στο διάστημα $\left[s_{\max}, s_{\min}\right]$. Χρησιμοποιείται το σύμβολο s σε συμφωνία με την παράγραφο 5.1.3
   3.1. Μέσω του καναλιού CH03 μετρούνται η μέγιστη και η ελάχιστη ψηφιακή έξοδος $\left[T_{\max}, T_{\min}\right]$. Λόγω της αργής κίνησης του βραχίονα, θεωρείται πώς οι ροπές αυτές είναι στατικές και οφείλονται μόνο στην δύναμη της βαρύτητας.
   3.2. Μέσω του καναλιού CH01, μετρούνται οι γωνίες στις οποίες εμφανίζονται τα ακρότατα.
   3.3. Γίνεται μια εκτίμηση των μεγίστων ροπών, υποθέτοντας πως ο βραχίονας καλείται να δώσει ροπές n φορές (π.χ. n=5) μεγαλύτερες από τις στατικές. Άρα οι ακρότατες ροπές είναι $\left[n \cdot s_{\max}, n \cdot s_{\min}\right]$
   3.4. Τότε το εύρος $\left[n \cdot T_{\max}, n \cdot T_{\min}\right]$ πρέπει να αντιστοιχεί σε ψηφιακό εύρος $\left[0, 4095\right]$ οπότε ισχύει

$$\left[n \cdot T_{\max}, n \cdot T_{\min}\right] = [0, 4095] \Leftrightarrow \left[T_{\max}, T_{\min}\right] = [0, \frac{4095}{n}] \qquad \textbf{εξ. 5. 36}$$

   3.5. Γίνεται υπολογισμός των νέων κερδών με $\left[T_{\max}^{req}, T_{\min}^{req}\right] = [0, \frac{4095}{n}]$ και εάν είναι απαραίτητο επαναλαμβάνεται το βήμα 3

Μετά το πέρας της επαναληπτικής διαδικασίας, ο αισθητήρας ροπής έχει βαθμονομηθεί με την μέγιστη διακριτική ικανότητα, έτσι ώστε η μέγιστη και η ελάχιστη δυνατή ροπή, που προκύπτουν για το μέγιστο φορτίο σε μέγιστη επιτάχυνση να αντιστοιχούν σε ψηφιακή έξοδο $[0, 4095]$.

Όμως, η αντιστοιχία μεταξύ πραγματικής ροπής και ψηφιακής εξόδου παραμένει άγνωστή. Για να υπολογιστεί η σταθερά αναλογίας, αφαιρείται το βάρος και ο βραχίονας κινείται στις ίδιες θέσεις μέγιστης και ελάχιστης ροπής

Έστω ότι:

$$\left[T_{\max}^{load}, T_{\min}^{load}\right]$$

είναι η μέγιστη και η ελάχιστη ψηφιακή ένδειξη όταν ο βραχίονας φέρει βάρος

$$\left[T_{\max}^{unload}, T_{\min}^{unload}\right]$$

είναι η μέγιστη και η ελάχιστη ψηφιακή ένδειξη όταν ο βραχίονας δεν φέρει βάρος.

Τότε ισχύει

$$\begin{cases} T_{\max}^{load} - T_{\max}^{unload} = K \cdot s_{\max} \\ T_{\min}^{unload} - T_{\min}^{load} = K \cdot s_{\min} \end{cases} \qquad \textbf{εξ. 5-37}$$

Όπου $s_{\max}, s_{\min}$ οι ροπές που οφείλονται αποκλειστικά στο επιπλέον βάρος, οι οποίες όμως είναι γνωστές λόγω γεωμετρίας. Τότε η σταθερά αναλογίας υπολογίζεται από την

$$K = \frac{\left(T_{\max}^{load} - T_{\max}^{unload}\right) + \left(T_{\min}^{unload} - T_{\min}^{load}\right)}{s_{\max} + s_{\min}} \qquad \textbf{εξ. 5-38}$$

Απαραίτητη προϋπόθεση είναι να υπάρξει απόλυτη ταύτιση μεταξύ των θέσεων του βραχίονα με βάρος και χωρίς βάρος. Μόνο τότε οι διαφορά των ροπών θα οφείλεται αποκλειστικά στο επιπρόσθετο βάρος και όχι σε διαφορετική γεωμετρική διευθέτηση. Για το λόγο αυτό είναι απαραίτητο να υπάρχει και ένας όρος I (integral) στον νόμο ελέγχου (έλεγχος PID), έτσι ώστε να μηδενίζονται τα σφάλματα θέσης.

6 Κώδικας Για Επικοινωνία, Βαθμονόμηση και Έλεγχο

Όπως έχει αναφερθεί, το SARCOS διαθέτει έναν υβριδικό βρόχο ελέγχου.

- Ο κύριος βρόχος είναι PD αναλογικός, αλλά παραμετροποιείται μέσω DAC, τα οποία κυκλωματικά είναι έτσι συνδεδεμένα, ώστε να επηρεάζουν το σήμα είτε ως κέρδη (gain) είτε ως πολώσεις (bias).
- Τα σήματα των επιθυμητών εισόδων θέσης, ροπής, αντιστάθμισης βαρύτητας είναι δυνατόν να στέλνονται στο SARCOS σε ψηφιακή μορφή, εκτός από αναλογική.
- Τα σήματα θέσης, ροπής ταχύτητας και άλλα είναι δυνατόν μέσω ADC να διαβαστούν σε ψηφιακή μορφή.

Το SARCOS, εν μέρει λόγω κατασκευής, και ως υδραυλικά κινούμενο, είναι ιδιάζων σε σχέση με τα ηλεκτρικά κινούμενα ρομπότ. Εφ όσον βρεθεί υπό πίεση, και δεν υπάρχει κλειστός βρόχος ελέγχου, αναγκαστικά οι Β.Ε. θα κινηθούν προς κάποια ,εν γένει, τυχαία κατεύθυνση και ο βραχίονας πιθανώς να χτυπήσει με αντικείμενα που τον περιβάλλουν ή τα μέλη να φτάσουν στα μηχανικά τους όρια με μεγάλη ταχύτητα. Και στις δυο περιπτώσεις, αυτό συνεπάγεται ακραία μηχανική καταπόνηση και ίσως ζημιά του βραχίονα.

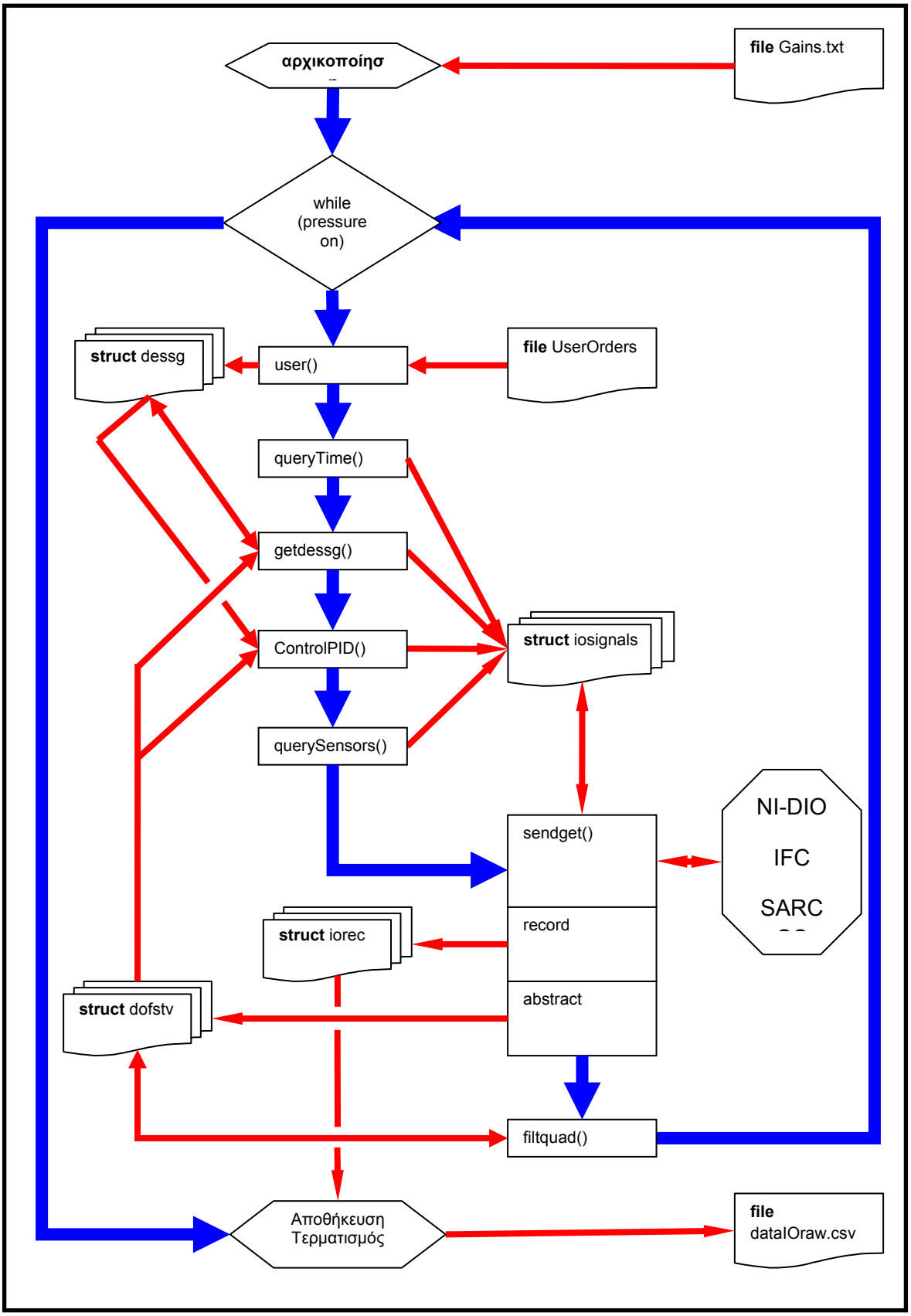
Από την άλλη, πολλές διαδικασίες, όπως η βαθμονόμηση των αισθητήρων, απαιτούν την κίνηση ενός Β.Ε. σε όλο το δυνατό εύρος. Σε μια αυτοματοποιημένη διαδικασία βαθμονόμησης αναγκαστικά ο βραχίονας πρέπει να είναι υπό πίεση. Αυτό σημαίνει πως πρέπει να υπάρχει ένας ελάχιστος βρόχος ελέγχου ο οποίος θα καθορίζει την θέση όλου του βραχίονα.

Επιπλέον, όπως συμβαίνει και στην προκειμένη περίπτωση, εάν για κάποιο λόγο ο αναλογικός βρόχος δεν είναι ευσταθής, και γίνεται προσπάθεια ανάστροφής μηχανικής είναι πολύ βολικό να μπορούν να ελεγχθούν οι κάρτες AJC κατά τμήματα. Έτσι, ένας ψηφιακός βρόχος ελέγχου ο οποίος παρακάμπτει εντελώς τον αναλογικό έλεγχο και επικεντρώνεται μόνο στις εισόδους και τις εξόδους δηλαδή θέση, ροπή, ρεύμα, μπορεί να δώσει πολύτιμα στοιχεία για πιθανή βλάβη.

Τέλος, ο ψηφιακός έλεγχος έχει πλέον αντικαταστήσει τον αναλογικό, οπότε μια προσπάθεια καθαρά ψηφιακού ελέγχου εκτός από διδακτική είναι και ως ένα βαθμό επιβεβλημένη ειδικά σε ένα τέτοιο ρομποτικό σύστημα με πολλές μη γραμμικότητες.

Στην προκείμενη περίπτωση, η υλοποίηση του ψηφιακού βρόχου ελέγχου, θα διευκόλυνε σε πολύ μεγάλο βαθμό τόσο τις διαδικασίες ελέγχου και βαθμονόμησης, αλλά ακόμα και αυτή την απλή διαδικασία εφαρμογής υψηλής πίεσης στο σύστημα (και γλίτωσε τον γράφοντα από πολλά προβλήματα).

Ο πλήρης κώδικας παρατίθεται στο παράρτημα. Σε αυτό το κεφάλαιο αναλύεται η λογική και η φιλοσοφία του κώδικα και παρατίθενται επιλεγμένα τμήματα κώδικα τα οποία είναι σημαντικά και ενδεικτικά της λειτουργίας του βρόχου ελέγχου. Κατ' αρχή θα περιγραφούν ξεχωριστά τμήματα κώδικα και οι λειτουργίες τους και στην συνέχεια θα περιγραφεί η χρήση τους εντός του βρόχου ελέγχου.



Σχήμα 6-1 ροή πληροφορίας (λεπτά κόκκινα βέλη) και

ροή κώδικα (παχιά μπλε βέλη) του κυρίως βρόχου.

Οι δομές και οι συναρτήσεις που χρησιμοποιούνται περιγράφονται περιληπτικά στον Πίνακες 6-1 και 6-2.

Πίνακας 6-1 Δομές που χρησιμοποιούνται στον βρόχο ελέγχου.

Δομή	**Επίθεμα**	**Επεξήγηση**
iosignals	.o	σήματα προς την κάρτα NI-DIO
	.i	σήματα καπό την κάρτα NI-DIO
dofstv	.channels	πληροφορία από τα κανάλια σε raw μορφή, FIFO circular buffer
	.fpos	φιλτραρισμένη θέση
	.fvel	φιλτραρισμένη ταχύτητα
	.facc	φιλτραρισμένη επιτάχυνση
dessg	.inp	επιλογή προέλευσης σημάτων επιθυμητών εισόδων
	.move	επιλογή τρόπου κίνησης
	.pos	επιθυμητή θέση
	.trq	επιθυμητή ροπή
	.cur	επιθυμητο ρεύμα
iorec	.o	αρχείο σημάτων προς την κάρτα NI-DIO
	.i	αρχείο σημάτων από την κάρτα NI-DIO

Πίνακας 6-2 Συναρτήσεις που υλοποιούν τον βρόχο ελέγχου.

Συνάρτηση	**Λειτουργία**
user()	είσοδος από χειριστή
queryTime()	έλεγχος χρονομέτρου
getdessg()	ανάγνωση σήματος επιθυμητής κατάστασης
ControlPID	PID βρόχος ελέγχου
querySensors()	ανάγνωση αρό αισθητήρες
sendget()	αποστολή λήψη σημάτων μεταξύ PC και SARCOS

6.1 Εντολές DAQmx αρχικοποίησης της NI DIO-96

Οι εντολές DAQmx αυτές, χρησιμοποιούνται για την αρχικοποίηση της κάρτας NI DIO-96. Εκτελούνται μια φορά πριν την είσοδο στον βρόχο ελέγχου.

- Καθορίζουν ποιες θύρες θα είναι είσοδοι, ποιες έξοδοι
- πώς θα ομαδοποιηθούν σε tasks (εργασίες)
- σε τι I/O mode θα λειτουργούν.
- καθορίζουν το μέγεθος των buffer για τα task εισόδου και εξόδου.

Επιπλέον πληροφορίες μπορούν να βρεθούν στα HTML αρχεία βοήθειας για την NI-DIO.

```
DAQmxResetDevice ("Dev1");

DAQmxCreateTask("Read",&taskRead);
DAQmxCreateTask("Write",&taskWrite);

DAQmxCreateDOChan(taskWrite,"Dev1/port0, Dev1/port1, Dev1/port6,
Dev1/port7 " ,"WriteChannel", DAQmx_Val_ChanForAllLines);

DAQmxCreateDIChan(taskRead, "Dev1/port3, Dev1/port4 " ,"ReadChannel",
DAQmx_Val_ChanForAllLines);

DAQmxSetBufOutputBufSize(taskWrite,0);// set to zero for continue IO
cycles

DAQmxSetBufInputBufSize(taskRead,512);// no more than 512 commands on
one cycle

DAQmxSetSampTimingType(taskWrite,DAQmx_Val_Handshake);
DAQmxSetSampTimingType(taskRead,DAQmx_Val_Handshake);

DAQmxSetSampQuantSampMode(taskWrite,DAQmx_Val_ContSamps);
DAQmxSetSampQuantSampMode(taskRead,DAQmx_Val_ContSamps);

DAQmxStartTask(taskRead );
```

6.2 Αιτήσεις εισόδου/εξόδου - Δομή του Πίνακα των αιτήσεων ελέγχου

Αφού αρχικοποιηθούν τα tasks, τότε μέσω των εντολών DAQmWrite και DAQmxRead μπορεί να γίνει αποστολή και λήψη πληροφορίας.

Γίνεται προσπάθεια να διαφοροποιηθούν οι έννοιες «αίτηση» και «εντολή». Η έννοια «εντολή» δηλώνει αμεσότητα μεταξύ αιτίου και αποτελέσματος, για αυτό το λόγο αποφεύγεται η χρήση της στο κείμενο. Για αποφυγή σύγχυσης, ισχύουν οι εξής ερμηνείες όπως παρουσιάζονται στον πίνακα 6-3.

Πίνακας 6. 3 Ειδικές Ερμηνείες Λέξεων Κλειδιών.

Λέξη ή Φράση	**Ερμηνεία**	**Επεξήγηση**
«αίτηση»	τμήμα κώδικα	κατασκευάζει ένα σήμα εξόδου 32bit και το αποθηκεύει σε προσωρινό Πίνακα, χωρίς να το αποστέλλει.
«σήμα εξόδου»	πληροφορία	περιέχει τις απαραίτητες πληροφορίες διευθύνσεις και δεδομένα για • αποστολή δεδομένων στο SARCOS • λήψη δεδομένων από το SARCOS

Λέξη ή Φράση	Ερμηνεία	Επεξήγηση
«σήμα εισόδου»	πληροφορία	το αποτέλεσμα αποστολής ή λήψης δεδομένων
«αποστολή δεδομένων»	διαδικασία της κάρτας IFC <u>μετά από αίτηση</u>	το σήμα αναλύεται σε τμήματα διεύθυνσης και δεδομένων τα όποια αποστέλλονται διαδοχικά στο SARCOS Τα δεδομένα επιστρέφουν μέσω του σήματος εισόδου.
«ανάγνωση δεδομένων»	διαδικασία κάρτας IFC <u>μετά από αίτηση</u>	το σήμα αναλύεται σε τμήμα ADDRESS και από το SARCOS λαμβάνονται δεδομένα μέσω του σήματος εισόδου.

Σε γενικές γραμμές ισχύουν τα εξής:

- Λόγω της απαίτησης για υψηλή ταχύτητα, τα σήματα εγγραφής/ανάγνωσης προς το SARCOS αποστέλλονται σε πακέτα (packages).
- τα <u>σήματα</u>, έχουν εύρος 32bit, είναι κωδικοποιημένα σύμφωνα με τις απαιτήσεις της κάρτας IFC και περιέχονται μέσα στον Πίνακα *iosignals.o*
- Ο πίνακάς *iosignals.o* ΔΕΝ κατασκευάζεται ενιαία σε ένα σημείο του κώδικα αλλά κατά τμήματα από διάφορες υπορουτίνες, μέσω αιτήσεων.
- Ανάλογα με τις τρέχουσες απαιτήσεις του ελέγχου, κάποια σήματα παραλείπονται και άλλα προστίθενται.
- Πολλές φορές επιπλέον σήματα πρέπει να μπουν εμβόλιμα στα σήματα ελέγχου.
- Οι ρουτίνες που κατασκευάζουν τον Πίνακα ΔΕΝ αποστέλλουν σήματα προς το SARCOS.
- Στο τέλος μια ρουτίνα (sendget()) αποστέλλει σε ένα πακέτο όλα τα σήματα εξόδου, και λαμβάνει τα αντίστοιχα σήματα εισόδου.

Αυτή η ιδιομορφία, επιβάλλει την ύπαρξη ενός επιπέδου αφαίρεσης (abstraction layer), δηλαδή, μιας μεθόδου διαλογής των σημάτων εισόδου, συσχετισμού τους με τα αντίστοιχα σήματα εξόδου που τα προκάλεσαν, και αποθήκευση τους σε αντίστοιχους πίνακες οι οποίοι χρησιμοποιούνται από όλες τις άλλες ρουτίνες ελέγχου.

Σε ένα κώδικα που είναι κατασταλαγμένος με συγκεκριμένες διαδικασίες εισόδου - εξόδου, μια τέτοια τεχνική ίσως να είναι πλεονασμός. Επειδή στην παρούσα υλοποίηση σκοπός είναι ο έλεγχος λειτουργίας όλων των υποσυστημάτων, υπήρχαν συχνές αλλαγές στις απαιτήσεις εισόδου - εξόδου και του κώδικα. Αυτό δημιουργούσε προβλήματα στην συνολική δομή του κώδικα, γιατί δε υπήρχε εύκολος συσχετισμός μεταξύ αιτήσεων προς το SARCOS και επιστρεφόμενης πληροφορίας.

6.3 Διαδικασία αρχικοποίησης του SARCOS

Η διαδικασία εκτελεί όλες τις απαραίτητες εργασίες αρχικοποίησης, απαραίτητες για την ομαλή είσοδο στον κύριο βρόχο ελέγχου περιλαμβάνει

- Αποστολή αρχικών κερδών στις AJC μέσω της ρουτίνας ReadConfig(). Τα αναλογικά κέρδη είναι αποθηκευμένα στο αρχείο gains. Εκεί είναι τα κέρδη για όλα τα DAC κάθε κάρτας AJC ξεχωριστά.
- Ανάγνωση των αισθητήρων και ενημέρωση των αντίστοιχων πινάκων.

- Αποθήκευση της αρχικής θέσης μέσω της ρουτίνας storepos(). Η αρχική θέση θεωρείται από το πρόγραμμα ως επιθυμητή θέση ισορροπίας. Για λόγους ευκολίας, ο βραχίονας επιστρέφει σε αυτή στο τέλος κάθε πειράματος.
- Επιβολή ασφαλούς μεθόδου λειτουργίας. Κατά την εκκίνηση του βρόχου, ο βραχίονας είναι κλειδωμένος στην αρχική θέση και για να ξεκλειδώσει πρέπει να πλησιάσει το σήμα της επιθυμητή στην πραγματική θέση. Αυτή η δέσμευση επιβάλλεται για να αποφευχθούν απότομες κινήσεις κατά την εφαρμογή πίεσης στο σύστημα.
- Εμφανίζεται μήνυμα που ενημερώνει πως πρέπει να εφαρμοστεί πίεση μέσα σε ορισμένο χρόνο διαφορετικά ο βρόχος ελέγχου θα τερματίσει
- Εκκίνηση του βρόχου ελέγχου.

6.4 Η υπορουτίνα queryTime() - Προετοιμασία Για Ανάγνωση χρόνου

Για τον έλεγχο αλλά και για την αναγνώριση παραμέτρων, απαιτείται η γνώση του χρόνου εκτέλεσης του βρόχου ελέγχου (loop time). Εάν είναι γνωστός ο χρόνος επιστροφής του βρόχου τότε μπορεί να αναπαρασταθεί χρονικά όλη η κίνηση, χρησιμοποιώντας ως χρόνο το άθροισμα των χρόνων επιστροφής.

Για αυτό το λόγο χρησιμοποιείται ο counter πάνω στην κάρτα SNIFFER. Οι αιτήσεις χειρισμού του counter, προηγούνται όλων των άλλων σε κάθε επανάληψη του βρόχου ελέγχου είναι οι ακόλουθες

```
void queryTime()
{
iosignals.o[iosignals.indx] =   0x00000000   |ReadMask| CntLo| sniffer;
iosignals.indx++;//read counter low  from SNIFFER CARD

iosignals.o[iosignals.indx] =   0x00000000   |ReadMask| CntHi| sniffer;
iosignals.indx++;//read counter high from SNIFFER CARD

iosignals.o[iosignals.indx] =   0x00000000   |WriteMask| CntLo|sniffer;
iosignals.indx++;//reset counter
}
```

Αυτή η σειρά είναι απαραίτητη ώστε να μετρηθεί όχι μόνο ο χρόνος αποστολής των σημάτων αλλά ο χρόνος ολόκληρου του βρόχου ελέγχου.

Οι αιτήσεις ερμηνεύονται εξής

- ανάγνωση του counter (ο counter σταματά να μετράει)
    - πρώτα τα 16 λιγότερο σημαντικά ψηφία
    - μετά τα 15 πλέον σημαντικά ψηφία
- μηδενισμός του counter, εκκίνηση νέας μέτρησης.

Πρέπει να σημειωθεί, πώς

- **Με αυτό τον τρόπο διαβάζεται ο χρόνος επιστροφής του προηγούμενου βρόχου.**
- Κανένα σήμα εξόδου δεν αποστέλλεται στο SARCOS

6.5.Η Υπορουτίνα querySensors() - Προετοιμασία Για Λήψη Δεδομένων

Βάσει των κανόνων των παραγράφων 4.1.3. και 6.1.2 παρατίθεται οι αιτήσεις που κατασκευάζουν τμήμα του Πίνακα iosignals.o για λήψη δεδομένων από τον ADC.

Ανάγνωση θέσης από το κανάλι CH01, και Ροπής από το κανάλι CH03 για όλους τους Β.Ε.

```
channel = Apos1<<16;

iosignals.o[iosignals.indx] = channel |WriteMask | GlobalMUXsel    | 1;
iosignals.indx++;//prepare MUXs to read Position from first stage

iosignals.o[iosignals.indx] =   0    |WriteMask | GlobalADC       | 1 ;
iosignals.indx++;   //force all ADCs to convert

for (dof=1; dof<sniffer; dof++)
  {
iosignals.o[iosignals.indx] =  0   |ReadMask  | LocalADC  | dof;
iosignals.indx++;} //read from all dofs

channel = Atrq<<16;

iosignals.o[iosignals.indx] =   channel   |WriteMask | GlobalMUXsel| 1;
iosignals.indx++;//prepare MUXs to read Torque

iosignals.o[iosignals.indx] =   channel   |WriteMask | GlobalADC   | 1;
iosignals.indx++;//force all ADCs to convert

for (dof=1; dof<sniffer; dof++)
  {
   iosignals.o[iosignals.indx] = channel   |ReadMask  | LocalADC  |dof;
   iosignals.indx++;
   } //read from all dofs
```

Εάν δεν είναι απαραίτητο να διαβαστεί το συγκριμένο κανάλι για όλους τους Β.Ε. τότε ο κώδικάς μπορεί να αλλάξει ως εξής:

Ανάγνωση του ποτενσιόμετρου από το κανάλι CH00 μόνο για όσους Β.Ε. ελέγχονται από το ποτενσιόμετρο

```
channel = PKnod<<16; ask = TRUE;

for (dof=1; dof<sniffer; dof++)
  {
  if (dessg.inp[dof]=='k' | dessg.inp[dof]=='K'  )
  {
  if (ask)
  {
 // prepare ADCs + convert only once

iosignals.o[iosignals.indx] = channel |WriteMask | GlobalMUXsel | 1;

iosignals.indx++;//prepare MUXs to read PanelKnod

iosignals.o[iosignals.indx] = WriteMask | GlobalADC    | 1;

iosignals.indx++;//force all ADCs to convert
```

```
ask = FALSE;
  }
iosignals.o[iosignals.indx] = ReadMask  | LocalADC        | dof;
iosignals.indx++; //read from dof
  }
}
```

6.6 Οι Ρουτίνες Ελέγχου

Ο έλεγχος γίνεται μέσω τριών ιεραρχικών επιπέδων (ρουτινών) τα οποία προς το παρόν εκτελούνται σειριακά. Τα όρια των λειτουργιών, που το κάθε επίπεδο αναλαμβάνει, είναι μερικώς ασαφή. Η κατανομή των λειτουργιών έγινε με γνώμονα την ευκολία κατανόησης του κώδικα, αλλά και την μελλοντική παραλληλοποίηση κάποιων ρουτινών. Μια άλλη υλοποίηση θα μπορούσε να κατανείμει κάποιες λειτουργίες διαφορετικά.

6.6.1 Η Ρουτίνα Διεπαφής με τον Χειριστή

Πρώτο επίπεδο είναι η διεπαφή με το χειριστή. Η ρουτίνα αυτή αναλαμβάνει την επικοινωνία με τον χειριστή και μέσω αυτής ο χειριστής επιλέγει τον τρόπο (method) λειτουργίας του ρομπότ. Η επιλογή λειτουργίας βασίζεται στην δομή *dessg* και στην τροποποίηση κάποιον σηματωρών (flags) που περιέχονται σε αυτή. Η σημασία τους περιγράφεται στον Πίνακα 6-4.

Πίνακας 6-4 Τρόποι λειτουργίας του ρομποτικού βραχίονα

επιθυμητή είσοδος *dessg.inp*	επιθυμητός τρόπος κίνησης *dessg.move*	σήμα ελέγχου	προέλευση σήματος ελέγχου	παρατηρήσεις
k	-	*dessg.pos*	ποτενσιόμετρο	κίνηση κάθε Β.Ε. ξεχωριστά
K	-	*dessg.pos*	ποτενσιόμετρο	Β.Ε. αρχικά κλειδωμένος
a	-	-	εσωτερικά παραγόμενο	υπολογισμός του γινομένου CAB
c	-	*dessg.cur*	εσωτερικά παραγόμενο	βαθμονόμηση αισθητήρα θέσης
C	-	*dessg.pos*	εσωτερικά παραγόμενο	βαθμονόμηση αισθητήρα ροπής
i		-	εσωτερικά παραγόμενο	identification open loop
I			εσωτερικά παραγόμενο	identification closed loop
f	f	*dessg.pos*	αρχείο	νέα επιθυμητή θέση, μετά από καθορισμένο χρόνο
	p	*dessg.pos*	αρχείο	νέα επιθυμητή θέση, μετά από καθορισμένο χρόνο και αφού όλοι οι Β.Ε. έχουν πλησιάσει την τρέχουσα επιθυμητή θέση
F	-	*dessg.pos*	αρχείο	όλοι οι Β.Ε. αρχικά κλειδωμένοι
o	f	*stored.pos*	εσωτερικά παραγόμενο	Β.Ε. προς κλείδωμα
	s		εσωτερικά παραγόμενο	Β.Ε. προς κλείδωμα σειριακά
O	-	*stored.pos*	εσωτερικά παραγόμενο	Β.Ε. κλειδωμένος

Προς το παρόν για λόγους ταχύτητας και ευκολίας ως προς την υλοποίηση, η ρουτίνα αυτή δεν διαβάζει εντολές από το πληκτρολόγιο, αλλά από ένα κωδικοποιημένο αρχείο. Απώτερος σκοπός είναι η ρουτίνα αυτή να τρέχει σαν ανεξάρτητο thread ώστε να μην επιβαρύνει υπολογιστικά τον βρόχο ελέγχου.

- k η επιθυμητή θέση δίνεται από τα ποντεσιόμετρο στην πρόσοψη της κάρτας
- K η επιθυμητή θέση δίνεται από τα ποντεσιόμετρο στην πρόσοψη της κάρτας.
    - Για αποφυγή απότομων κινήσεων, ο Β.Ε. παραμένει κλειδωμένος στην τρέχουσα θέση ώσπου η επιθυμητή θέση πλησιάσει την πραγματική οπότε ξεκλειδώνει
- a υπολογισμός του γινομένου CAB
    - Ταυτόχρονος για όλα τα στάδια βαθμονόμησης θέσης και ροπής.
- c διαδικασία βαθμονόμησης του αισθητήρα θέσης
- C διαδικασία βαθμονόμησης του αισθητήρα ροπής
- ι Διαδικασία αναγνώρισης παραμέτρων ανοιχτού βρόχου
    - Ο βραχίονας τοποθετείται στην επιθυμητή θέση, και στις βαλβίδες αποστέλλεται διέγερση ρεύματος με προφίλ λευκού θορύβου και με ελεγχόμενα χαρακτηριστικά
        - Επιθυμητής μέσης τιμής
        - Επιθυμητής συχνότητας
        - Επιθυμητού εύρους
- I Διαδικασία αναγνώρισης παραμέτρων κλειστού βρόχου
    - Ο βραχίονας τοποθετείται στην επιθυμητή θέση και στον βρόχο PD αποστέλλεται διέγερση θέσης με προφίλ λευκού θορύβου και με ελεγχόμενα χαρακτηριστικά
        - Επιθυμητής μέσης τιμής
        - Επιθυμητής συχνότητας
        - Επιθυμητού εύρους
- f,f Επιθυμητή θέση από αρχείο.
    - Το αρχείο περιέχει την επιθυμητή θέση για κάθε Β.Ε. και τον χρόνο παραμονής σε αυτή. Η επόμενη κίνηση γίνεται αφού παρέλθει ο χρόνος παραμονής
- f,p, Επιθυμητή θέση από αρχείο
    - Το αρχείο περιέχει την επιθυμητή θέση για κάθε Β.Ε. και τον χρόνο παραμονής σε αυτή. Η επόμενη κίνηση γίνεται αφού παρέλθει ο χρόνος παραμονής και εφ όσον όλοι οι Β.Ε. έχουν λάβει την επιθυμητή θέση.
- F Επιθυμητή θέση από αρχείο
    - Για αποφυγή απότομων κινήσεων, ο Β.Ε. παραμένει κλειδωμένος στην τρέχουσα θέση ώσπου η επιθυμητή θέση πλησιάσει την πραγματική οπότε ξεκλειδώνει
- o,f Β.Ε. προς κλείδωμα
    - ο Β.Ε. κινείται προς την θέση αποθηκευμένη στον Πίνακα *stored.pos* αυτόνομα και κλειδώνει
- o,s Β.Ε. προς κλείδωμα
    - ο Β.Ε. κινείται προς την θέση αποθηκευμένη στον Πίνακα *stored.pos* αφού ο προηγούμενος Β.Ε. έχει κλειδώσει. Η σειρά είναι από τον 10° προς τον 1°. Αυτός ο τρόπος είναι ο πλέον ελεγχόμενος και επιτρέπει στον βραχίονα να ξεδιπλώνει.

- O Β.Ε. κλειδωμένος στην θέση *stored.pos*
- H Οι εντολές προς τον Β.Ε. δεν μεταβάλλονται το ρεύμα παραμένει σταθερό.

Τα flags αυτά αναφέρονται σε κάθε Β.Ε. ξεχωριστά και επηρεάζουν την λειτουργία κάθε Β.Ε. ξεχωριστά. Έτσι υπάρχει η δυνατότητα ένας Β.Ε. να είναι σε διαδικασία βαθμονόμησης ή αναγνώρισης παραμέτρων και όλοι οι άλλοι σε PD έλεγχο. Με αυτό τον τρόπο, επιτυγχάνεται καλύτερος έλεγχος του βραχίονα και αποφυγή δυσάρεστων καταστάσεων.

6.6.2 Η ρουτίνα Επιλογής Επιθυμητής Εισόδου

Η ρουτίνα αυτή υλοποιεί όσα περιγράφονται στο εδάφιο 6.5.2.

- Ανάλογα με τις τιμές των
    - *dessg.inp*,
    - *dessg.move*,
- κατασκευάζει
    - την επιθυμητή θέση *dess.pos*,
    - ή το επιθυμητό ρεύμα *dess.cur*.
- Είναι υπεύθυνη για την τήρηση του χρονισμού, όπου αυτό απαιτείται, χρησιμοποιώντας σαν χρόνο την τιμή του counter που βρίσκεται πάνω στην κάρτα SNIFFER.
    - Η δομή *counter* κρατάει τους χρόνους.
- Ελέγχει το κλείδωμα ή το ξεκλείδωμα των Β.Ε.

6.5.3 Η ρουτίνα ControlDigPID() - Ψηφιακού Αλγόριθμου Έλεγχου

Η ρουτίνα αυτή υλοποιεί τον ψηφιακό έλεγχο. Ο έλεγχος είναι PID, ο αναλογικός έλεγχος των AJC είναι PD, σε κάθε άρθρωση ξεχωριστά (δεν υπάρχουν διαγώνιοι όροι).

Χρησιμοποιεί τις μεταβλητές

- *dofstv.fpos*
- *dofstv.fvel*
- *dessg.pos*
- *dessg.cur*

για να υπολογίσει, μέσω PID νόμου ελέγχου, τα ρεύματα που στέλνονται στις βαλβίδες. Στην συνέχεια κατασκευάζει εκείνο το τμήμα του Πίνακα *iosignals.o* που στέλνει αντίστοιχα τα σήματα στο SARCOS.

Ο P (proportional) όρος του ελέγχου είναι πάντα ενεργός και παρέχει ο κατεξοχήν όρος από τον οποίο υπολογίζεται το απαιτούμενο ρεύμα προς τις βαλβίδες.

Για να λειτουργήσει απρόσκοπτα ο D (derivative) όρος του έλεγχου, πρέπει το σήμα να φιλτραριστεί ψηφιακά. Αυτό το αναλαμβάνουν οι ρουτίνες του εδαφίου 6.8

Ο I (integral) όρος του έλεγχου, είναι απαραίτητος εφ όσον ζητείται μηδενισμός του σφάλματος. Ο I όρος είναι φραγμένος σε μια προσπάθεια να μην εμφανίζεται αστάθεια. Ενεργοποιείται μόνο εφόσον υπάρχει πίεση. Στην αντίθετη περίπτωση το αντίστοιχο ολοκλήρωμα μηδενίζεται έτσι ώστε, να μην συσσωρεύεται σφάλμα, καθώς ο βραχίονας αδυνατεί να κινηθεί.

6.7 Η Ρουτίνα sendget(char action) - Αποστολή και Λήψης σημάτων.

Όταν ο Πίνακας *iosignals.o* συμπληρωθεί και είναι έτοιμος προς αποστολή στην NI-DIO καλείται η ρουτίνα sendget. Ο Πίνακας αυτός περιέχει σήματα, δηλαδή όλη την πληροφορία που απαιτείται για ένα κύκλο αποστολής δεδομένων στο SARCS ή ένα κύκλο λήψης δεδομένων από το SARCOS

Η αποστολή και η λήψη σημάτων καθώς και ο έλεγχος σφαλμάτων επικοινωνίας γίνεται με τις εντολές

```
werr = DAQmxWriteDigitalU32 (taskWrite, iosignals.indx, 1, 0.10,
DAQmx_Val_GroupByChannel, &iosignals.o[0],       &written, NULL);

rerr = DAQmxReadDigitalU16 (taskRead,  iosignals.indx,    0.10,
DAQmx_Val_GroupByChannel, &iosignals.i[0], 1024, &read,    NULL);

if (werr!=0 | rerr!=0) printf("*** WARNING DAQmx WRITE %u READ %u
error\n\a *** ",werr,rerr);
```

Οι κύκλοι αποστολή και λήψης ακολουθούν το πρωτόκολλο που περιγράφεται στις σελίδες 65 -66.

- Αποστέλλονται τα σήματα *iosignals.o* και λαμβάνεται τα σήματα *iosignals.i*
- Κάθε φορά που η αποστέλλεται ένα σήμα εξόδου 32bit στην NI-DIO αναμένεται να ληφθεί ένα σήμα εισόδου 16bit.
- Εάν το σήμα εξόδου αποστέλλει δεδομένα στο SARCOS, τότε τα 16bit του σήματος εισόδου είναι <u>αντίγραφο</u> των δεδομένων που αποστέλλονται.
- Εάν το σήμα εξόδου διαβάζει δεδομένα από το SARCOS, τότε τα 16bit του σήματος εισόδου είναι η ζητούμενη πληροφορία.

Επομένως, υπάρχει μια αντιστοιχία 1-1 μεταξύ σημάτων εξόδου και σημάτων εισόδου και για να ερμηνευτεί σωστά το περιεχόμενο ενός σήματος εισόδου, πρέπει να είναι γνωστό το αντίστοιχο σήμα εξόδου.

Οι εντολές DAQmxWrite και DAQmxRead χρησιμοποιούνται μόνο μια φορά μέσα στο βρόχο γιατί έχουν μεγάλο κόστος χρόνου αρχικοποίησης κατά την κλήση τους.

Επειδή οι αιτήσεις που αποστέλλονται προς το SARCOS δεν έχουν σταθερή περιοδικότητα και αλλάζουν ανάλογα με τον τρόπο (method) λειτουργίας του ρομπότ, τα σήματα που επιστρέφονται στον Πίνακα *iosignals.i* έχουν ασάφεια ως προς την αίτηση που τα προκάλεσε.

Για να αρθεί η ασάφεια αυτή το υπόλοιπο τμήμα της ρουτίνας ελέγχει τον Πίνακα *iosignals.o* και ανάλογα με την αίτηση διαχωρίζει τα σήματα εισόδου *iosignals.ι* και τα αποθηκεύει στις αντίστοιχες θέσεις του Πίνακα *dofstv.channels[dof][channel][time].* Με αυτό τον τρόπο κατασκευάζεται ένας κυκλικός buffer (FIFO) ο οποίος κρατάει τα τελευταία 8 στιγμιότυπα των καναλιών.

Αυτό το επίπεδο αφαίρεσης (abstraction layer) είναι απαραίτητο, προκειμένου να υπάρχει ευκολία προσθήκης αιτήσεων προς το SARCOS σε τυχαίο σημείο του κώδικά, χωρίς να επηρεάζονται άλλα σημεία του κώδικα. Όλες οι άλλες ρουτίνες που απαιτούν γνώση των

καναλιών διαβάζουν την πληροφορία αυτή από την δομή *dofstv* και όχι κατευθείαν από τον Πίνακα *iosignals.i* .

Στην συνέχεια, εάν είναι επιθυμητό, ενημερώνεται η δομή *iorec*. Στη δομή αυτή αποθηκεύονται, προσωρινά, τα σήματα εξόδου και εισόδου και στο τέλος του πειράματος όλη η δομή αποθηκεύεται στο αρχείο *dataIOraw.csv* για offline επεξεργασία. Η ανάγνωση του αρχείου *dataIOraw.csv* γίνεται από τον matlab κώδικα που παρατίθεται στο παράρτημα.

6.8 Υπορουτίνες υλοποίησης ψηφιακών φίλτρων

Τα δεδομένα που διαβάζονται από τα κανάλια αποθηκεύονται σε ένα προσωρινό κυκλικό buffer, τον Πίνακα dofstv.channels[dof][channel][time]. Εάν κριθεί σκόπιμο μπορούν να εφαρμοστούν ψηφιακά φίλτρα για την καταστολή του θορύβου.

Οι ρουτίνες αυτές χρησιμοποιούν τον Πίνακα dofstv.channels και κατασκευάζουν τα φιλτραρισμένα σήματα,

- θέσης, *dofstv.pos[dof]*
- ταχύτητας, *dofstv.vel [dof]*
- επιτάχυνσης, *dofstv.acc[dof]*

Προς το παρόν έχουν αναπτυχθεί δυο ρουτίνες

- filtnone(), δεν φιλτράρει τα δεδομένα, παρέχει μόνο πληροφορία θέσης
- filtquad(), εφαρμόζει μια παρεμβολή δεύτερης τάξης στις τελευταίες 8 μετρήσεις

Κατά την εφαρμογή του αλγορίθμου παρεμβολής υποτέθηκαν σταθεροί χρόνοι βρόχου, προς απλοποίηση και επιτάχυνση των υπολογισμών.

Το κύριο πρόβλημα που εμφανίστηκε είναι πως, λόγω απουσίας ελέγχου πραγματικού χρόνου, δεν είναι σταθερός ο χρόνος επιστροφής του βρόχου, και το χρονικό βήμα δεν είναι σταθερό. Οπότε τα αποτελέσματα των φίλτρων δεν ήταν πάντα ακριβή. Παρέχουν όμως μια εκτίμηση της ταχύτητας, χρήσιμη για τον PID βρόχο.

Αν και η μέθοδος της παρεμβολή μπορεί να αντιμετωπίσει αυτό το πρόβλημα των άνισων χρόνων επιστροφής του βρόχου ελέγχου, οι επιπλέον υπολογισμοί που απαιτούνται κάνουν την μέθοδο πολύ αργή και αναποτελεσματική με τον υπάρχοντα εξοπλισμό. Όμως, με εφαρμογή ελαφρύτερου λειτουργικού συστήματος πραγματικού χρόνου, αναμένεται να αποδώσει καλυτέρα.

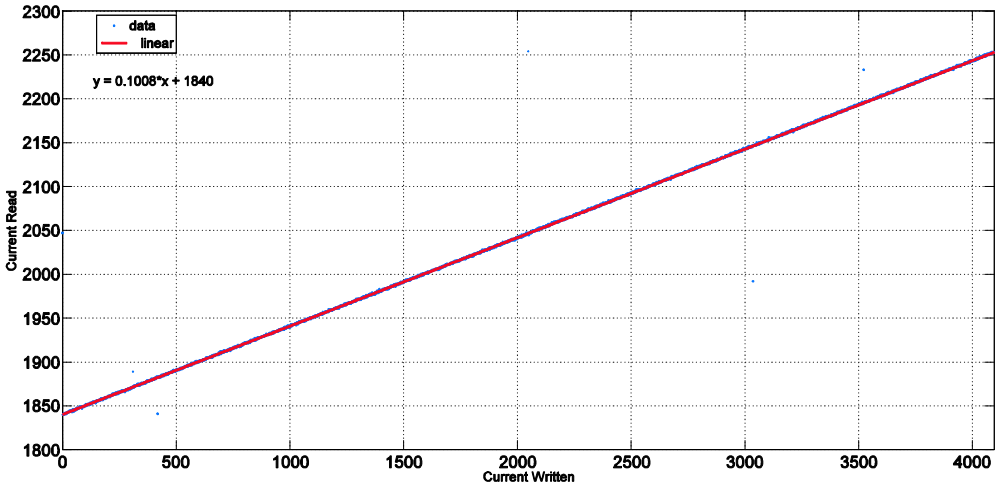
7 Πειράματα

7.1 Αρχικά Πειράματα

Τα πρώτα πειράματα ήταν ανοικτού βρόχου και διερευνούσαν την δυνατότητα εγγραφής ψηφιακών δεδομένων στις κάρτες AJC και ανάγνωσης ψηφιακών δεδομένων από αυτές. Τα DAC που επιλέχθηκαν για την δοκιμή είναι τα DAC12 (valve gain) και DAC08 (valve bias), επειδή επιδρούν απ' ευθείας στο ρεύμα, το οποίο μπορεί να διαβαστεί από το κανάλι CH07 (valve current) και επειδή η κίνηση του βραχίονα θα επιβεβαίωνε, οπτικά, ότι η εντολή έφτασε στην βαλβίδα.

Στο DAC12 φορτώθηκε η τιμή 0 η οποία αντιστοιχεί σε αναλογικό κέρδος 0 και αποκόπτει πλήρως τον αναλογικό έλεγχο. Στην συνέχεια στο DAC08 φορτωθήκαν διαδοχικά οι τιμές 0 έως 4095 και κάθε φορά διαβαζόταν η τιμή του CH7. Οι μετρήσεις έγιναν σε όλους τους Β.Ε.. Κατά την διάρκεια των πειραμάτων ο βραχίονας βρισκόταν στην χαμηλή πίεση έτσι ώστε να είναι ενεργοποιημένοι οι ενισχυτές ισχύος. Τα αποτελέσματα είναι σχεδόν ίδια για όλους τους Β.Ε. Στο Σχήμα 7-1 παρουσιάζονται για τον 2^{ο} Β.Ε.

Παρατηρείται μια, μη αναμενόμενη, αναλογία 1/10 μεταξύ τις τιμής που εγγραφόταν και της εντολής που διαβαζόταν, η αναλογία αυτή οφείλεται στο κύκλωμα εισόδου του σταδίου ισχύος. Περισσότερες λεπτομέρειες βρίσκονται στο Παράρτημα.



Σχήμα 7-1 Σχέση τιμής DAC08 και CH07.

Αντίστοιχη δοκιμή μπορεί να γίνει και μέσω των DAC10 και DAC31 τα οποία επιδρούν άμεσα πάνω στο κανάλι CH15. Το DAC10 μπορεί να φορτωθεί με την η τιμή 0 η οποία αντιστοιχεί σε αναλογικό κέρδος 0 και αποκόπτει πλήρως το σήμα από τον αισθητήρα θέσης. Στην συνέχεια στο DAC08 μπορούν να φορτωθούν οι τιμές [0…4095] και κάθε φορά να διαβάζεται η τιμή του CH15.

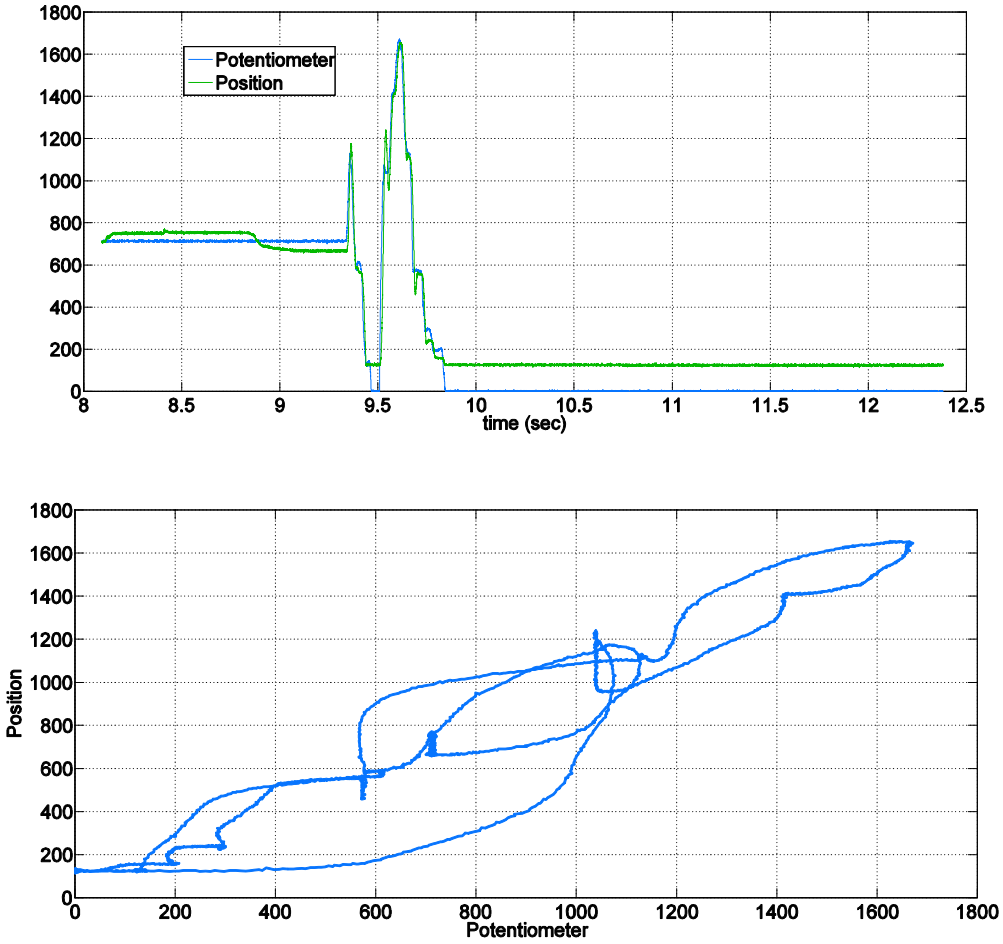
Στην συνέχεια, έγινε προσπάθεια **ψηφιακού** (όχι αναλογικού) **P έλεγχου** μέσω των ποτενσιόμετρων. Για κάθε Β.Ε., ως επιθυμητή θέση λαμβανόταν το ψηφιακό σήμα από το CH0 (ποτενσιόμετρο), και ως πραγματική θέση το ψηφιακό σήμα από το CH1 (αισθητήρας θέσης). Σκοπός της προσέγγισης αυτής, ήταν να κατασκευαστεί ένας απλός βρόχος

ελέγχου, με την πλέον απλή είσοδο, ώστε να διαπιστωθεί η λειτουργικότητα του συστήματος.

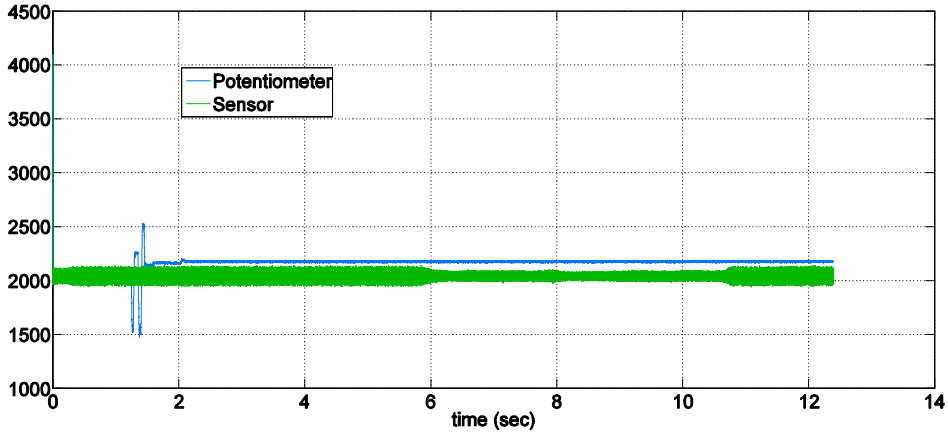
Σε κάθε βαθμό ελευθερίας χρησιμοποιήθηκε κέρδος βρόχου K_p = 1 και το σύστημα τέθηκε, για λόγους ασφαλείας, σε χαμηλή πίεση 500psi. Οι αισθητήρες δεν είχαν βαθμονομηθεί ακόμα σωστά οπότε δεν υπήρχαν προσδοκίες πλήρους ευστάθειας. Τα αποτελέσματα φαίνονται στον Πίνακα 7.1.

Πίνακας 7-1 Συμπεριφορά των Β.Ε. πρώτη απόπειρα.

Β.Ε.	**συμπεριφορά**
1, 3, 5, 6	ευστάθεια
2, 4	αστάθεια
7, 8, 9	μηδενικό σήμα από τους αισθητήρες
10	δεν υπάρχει AJC



Σχήμα 7-2 Παράδειγμα ευστάθειας (1^{ου} Β.Ε.) κατά την πρώτη απόπειρα.



Σχήμα 7-3 Μηδενικό σήμα από αισθητήρες.

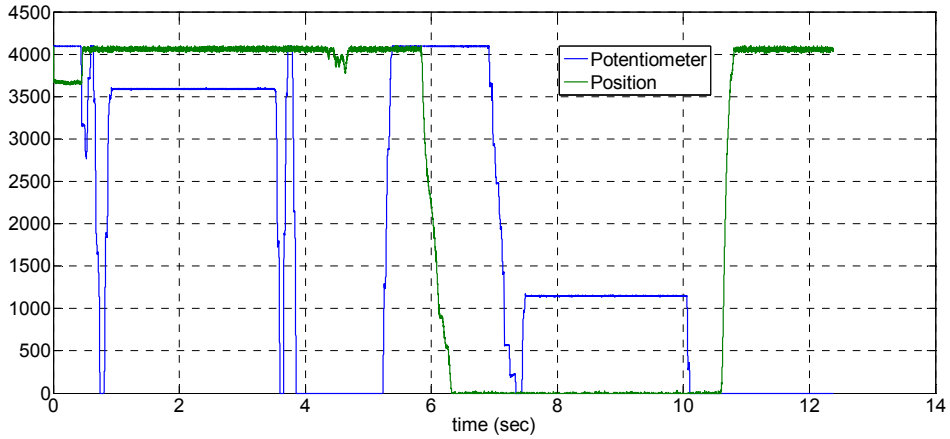
Το γεγονός ότι εμφανίστηκε ευστάθεια σε κάποιους Β.Ε. ήταν ενθαρρυντικό, οπότε η επόμενη κίνηση ήταν ο έλεγχος των Β.Ε. που δεν συμπεριφέρονταν κατά το αναμενόμενο.

Η διερεύνηση έδειξε πώς οι αισθητήρες θέσης των Β.Ε. 7, 8, 9 είχαν ρυθμιστεί ως να ήσαν optical encoder ενώ είναι ποτενσιόμετρα. Μια αλλαγή στον jumper J15 (AJC2.1) έλυσε το πρόβλημα και οι Β.Ε. 7, 8, 9 απέκτησαν ευστάθεια.

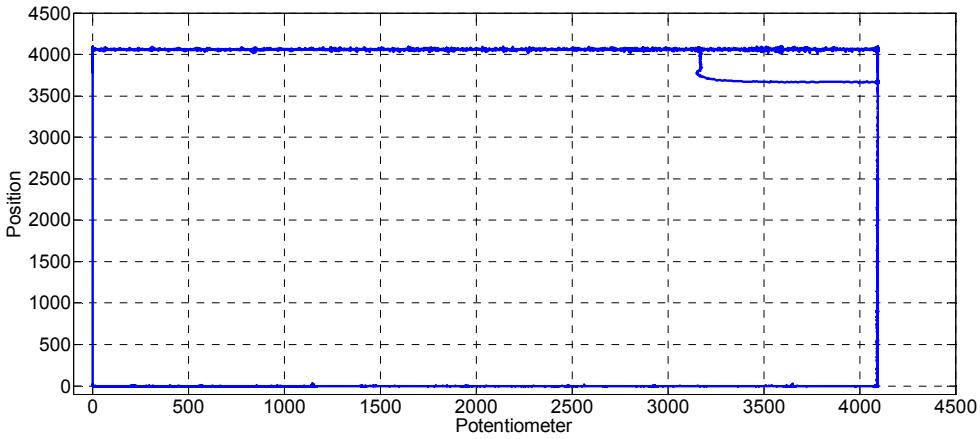
Στη συνέχεια η προσοχή εστιάστηκε στους Β.Ε. 2 και 4. Μια πρώτη σκέψη, ήταν ότι κάπου στην διαδρομή του κυκλώματος ισχύος υπήρχε ηλεκτρικό πρόβλημα. Αν και η ανάγνωση από το CH7 (ρεύμα βαλβίδας) έδειχνε πως το ρεύμα ήταν το αναμενόμενο, οι βαλβίδες αυτών των Β.Ε. είχαν κατά το παρελθόν αφαιρεθεί για την εκτέλεση πειραμάτων και αυτό ίσως να είχε δημιουργήσει προβλήματα.

Αφού ανοίχτηκαν τα καπάκια των διαμερισμάτων των βαλβίδων, φάνηκε ότι όντως ο Β.Ε. 4 είχε ηλεκτρικό πρόβλημα, αφού τα καλώδια είχαν συνθλιβεί σε κάποια σημεία και εμφανιζόντουσαν σπινθήρες κατά την λειτουργία. Έγινε παρέμβαση, και το ελαττωματικό τμήμα των καλωδίων απομακρύνθηκε. Όμως, ο Β.Ε. 2 δεν παρουσίαζε κανένα ηλεκτρικό πρόβλημα.

Παρά την παρέμβαση, το πρόβλημα της αστάθειας παρέμεινε και για τους δυο Β.Ε., 2° και 4°. Για αυτό τον λόγο, έγινε μια δοκιμή σε υψηλή πίεση, μήπως η αστάθεια εμφανιζόταν λόγω της χαμηλής πίεσης. Η αστάθεια παρέμεινε, αλλά στο γράφημα μεταξύ επιθυμητής θέσης και πραγματικής θέσης, εμφανίστηκε ένας ευδιάκριτος βρόχος υστέρησης, χαρακτηριστικός θετικής ανάδρασης όπως φαίνεται στο Σχήμα 7-5 .



Σχήμα 7-4 Αστάθεια κατά την κίνηση του 2ου Β.Ε.



Σχήμα 7-5 Βρόχος υστέρησης θετικής ανάδρασης για τον 2ο Β.Ε.

Αυτό σήμαινε πως το ρεύμα στις βαλβίδες είχε αντίθετο πρόσημο από το αναμενόμενο. Πιθανή αιτία είναι ανάποδη σύνδεση στην καλωδίωση των βαλβίδων. Η αλλαγή του πρόσημου του κέρδους K_p του ψηφιακού βρόχου, μόνο για τον 2ο και 40 Β.Ε., από 1 σε -1 έλυσε το πρόβλημα οπότε και επιτεύχθηκε ευστάθεια.

Αυτό το εύρημα οδήγησε στην σκέψη στους Β.Ε. 2 και 4 να αναστραφεί η πολικότητα του σήματος εξόδου του αναλογικού βρόχου μέσω του jumper J14 (AJC1.1). Η απόφαση αυτή θα αξιολογηθεί με πειράματα ελέγχου του αναλογικού βρόχου.

7.2 Πειράματα Χρονισμού

Ο χρόνος που απαιτείται για να κλείσει ο βρόχος ελέγχου, θεωρώντας σταθερή την διαθέσιμη υπολογιστική ισχύ, εξαρτάται από δυο παράγοντες :

- από τον χρόνο επικοινωνίας με την NI DIO-96
- από τον χρόνο εκτέλεσης του κώδικα

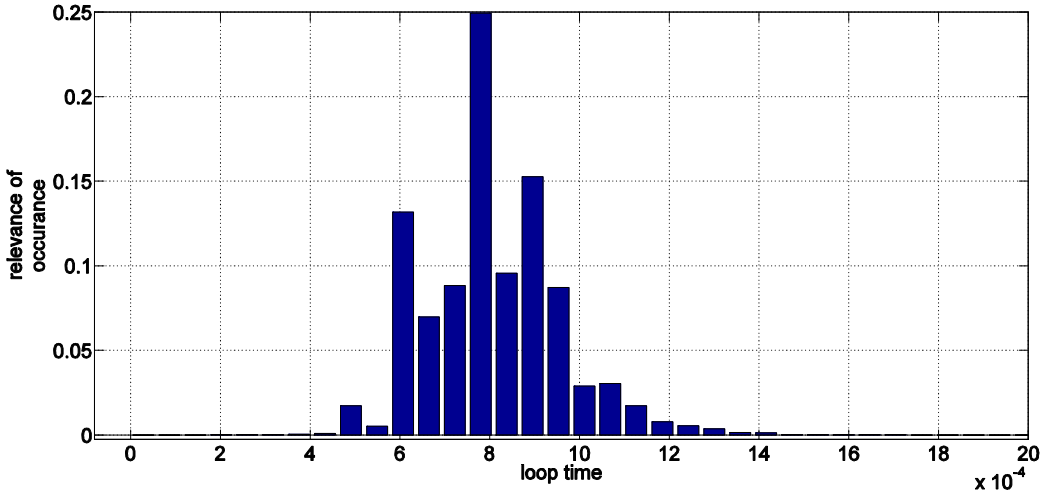
Προφανώς, όσο πιο πολλά δεδομένα προς λήψη και αποστολή τόσο μεγαλύτερος ο χρόνος επικοινωνίας με την DIO, και όσο πιο πολύπλοκος ο κώδικας ελέγχου τόσο πιο αργή η εκτέλεση του. Επιπλέον στοιχεία που επιβαρύνουν την εκτέλεση του κώδικα είναι προγράμματα που τρέχουν παράλληλα όπως το γραφικό περιβάλλον ή άλλα προγράμματα του λειτουργικού συστήματος Linux.

Λόγω του ότι το λειτουργικό προς το παρόν δεν είναι πραγματικού χρόνου (real time) η διάρκεια του βρόχου δεν είναι σταθερή. Παρ όλα αυτά, είναι δυνατή η κατασκευή μιας κατανομής των χρόνων που κάνει ο βρόχος να κλείσει, από την οποία μπορούν να συναχθούν συμπεράσματα.

Οι αιτήσεις που αποτελούν το πακέτο (package) και αποστέλλονται στις κάρτες, είναι ένα τυπικό σύνολο αιτήσεων στην περίπτωση ελέγχου θέσης και ροπής. Σε συνοπτική μορφή είναι οι εξής :

Πίνακας 7-2 Εντολές και Κύκλοι.

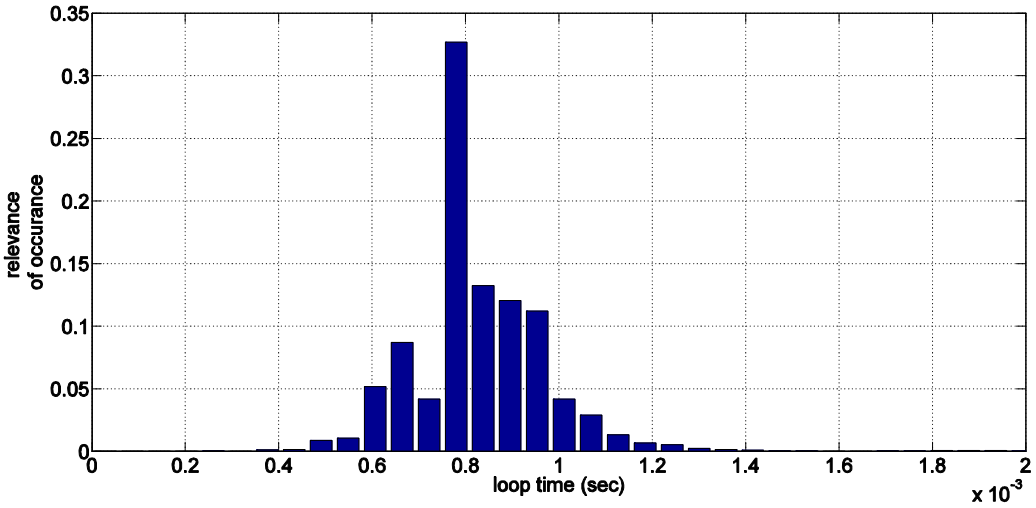
Εντολή	Αριθμός Κύκλων	
	send	**receive**
ανάγνωση counter	2	0
μηδενισμός counter	1	0
εγγραφή ρευμάτων	9	0
ανάγνωση ποτενσιόμετρων	2	9
ανάγνωση θέσεων	2	9
ανάγνωση ροπών	2	9
σύνολο	18	29
ολικό σύνολο		45



Σχήμα 7-6 Κατανομή χρόνων επιστροφής βρόχου χωρίς φίλτρο.

Στο εν λόγω πείραμα δεν χρησιμοποιήθηκε κάποιο φίλτρο. Η κατανομή των χρόνων δίνει μια μέση τιμή 0.82ms με διασπορά 0.25ms χρόνος ικανός ακόμα και για έλεγχο δύναμης. Όμως η διασπορά είναι της τάξης του 30% και τίθενται σοβαρά ερωτηματικά για την ευρωστία του βρόχου ελέγχου.

Για σύγκριση αντίστοιχο πείραμα έγινε και με την εφαρμογή φίλτρου το οποίο υπολογιστικά επιβαρύνει το βρόχο. Τα αποτελέσματα φαίνονται στο Σχήμα 7-7



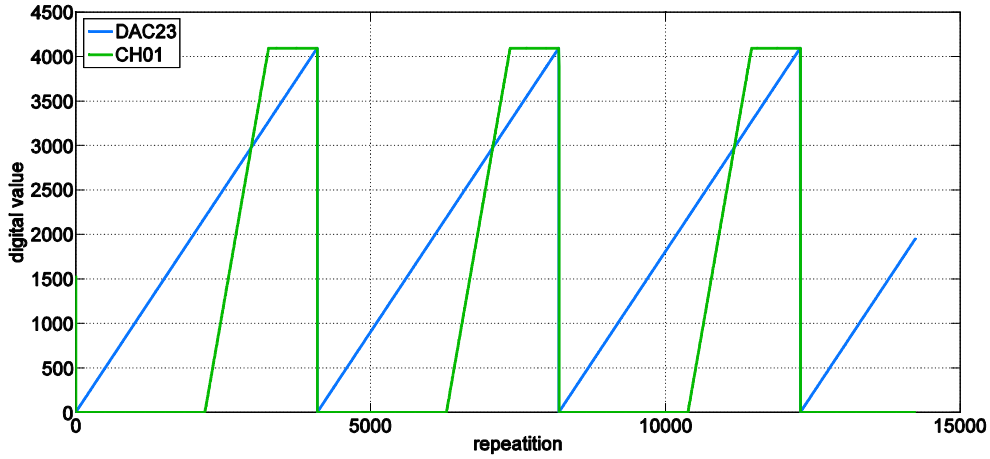
Σχήμα 7-7 Κατανομή χρόνων επιστροφής βρόχου με φίλτρο.

Η κατανομή των χρόνων δίνει μια μέση τιμή 0.84msec με διασπορά 0.27msec χρόνος ικανός ακόμα και για έλεγχο δύναμης. Η διασπορά είναι ξανά της τάξης του 30%. Όμως παρά τον υπολογιστικό φόρτο λόγω φίλτρου, ο μέσος χρόνος και η διασπορά άλλαξαν πολύ λίγο, το οποίο σημαίνει ότι το βασικό αίτιο αργοπορίας πρέπει να εντοπιστεί στην κάρτα NI-DIO.

7.3 Βαθμονόμηση Αισθητήρων Θέσης

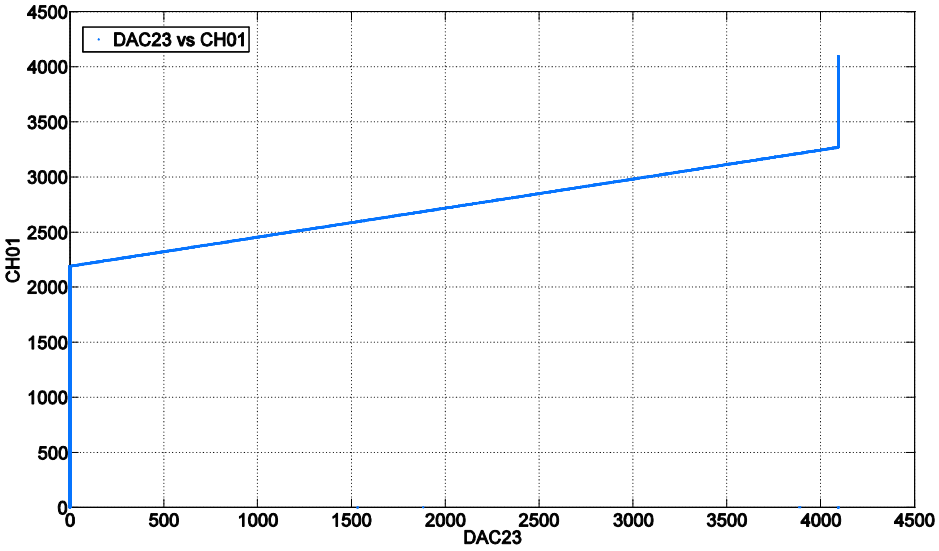
7.3.1 Υπολογισμός Γινομένου CAB

Για τον υπολογισμό του γινομένου $C \cdot A \cdot B$ φορτώνεται στο DAC10 η τιμή 110, έτσι ώστε το εύρος του ψηφιακού σήματος X να είναι περίπου 1000. Ο βραχίονας μένει σταθερός στην θέση ηρεμίας του και μεταβάλλεται, αργά, η τιμή του DAC23. Αυτό έχει σαν αποτέλεσμα την μεταβολή του σήματος από το κανάλι CH01. Στο Σχήμα 7-8 φαίνονται οι αντίστοιχες μεταβολές.



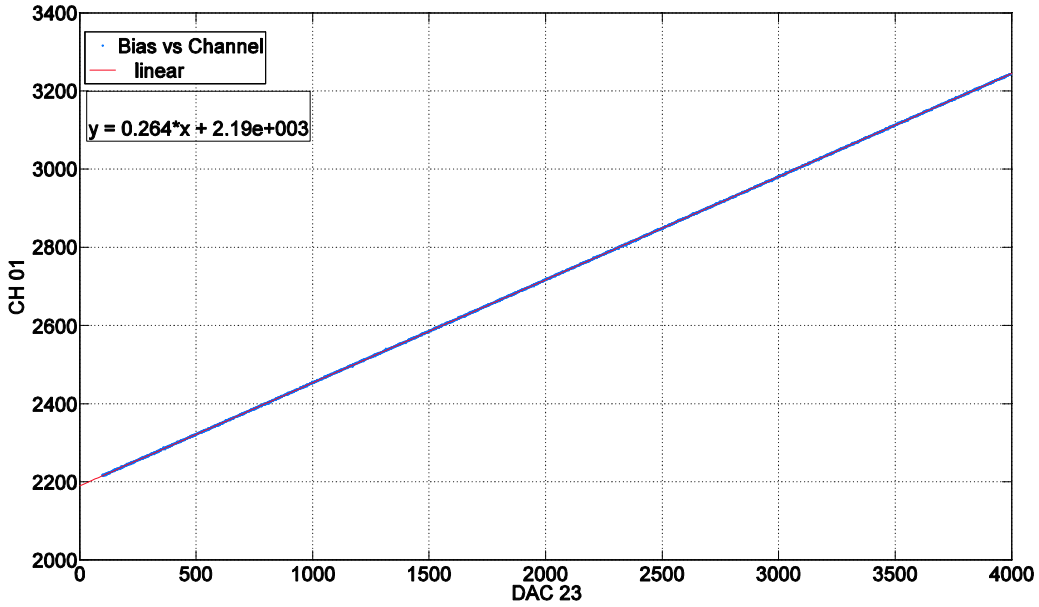
Σχήμα 7-8 Ψηφιακά σήματα DAC23 και CH01 στον χρόνο.

Η γραφική παράσταση μεταξύ της ψηφιακής τιμής που εγγράφεται στο DAC23 και της ψηφιακής τιμής που αναγιγνώσκεται από το κανάλι CH01 φαίνεται στο Σχήμα 7-9.



Σχήμα 7-9 Σήμα DAC23 σε σχέση με το σήμα CH01 (πλήρες εύρος).

Απορρίπτοντας τις τιμές που βρίσκονται σε κορεσμό λαμβάνεται η γραφική παράσταση του Σχήματος 7-10.



Σχήμα 7-10 Σήμα DAC23 σε σχέση με το σήμα CH01 (γραμμικό τμήμα).

Οπότε σύμφωνα με την μέθοδο που περιγράφεται στο Κεφάλαιο 5.1.4 υπολογίζεται το γινόμενο

$$C \cdot A \cdot B = \frac{0.264}{110} \Rightarrow \frac{1}{C \cdot A \cdot B} = 417 .$$

Ο υπολογισμός του όρου $(C \cdot A \cdot B)^{-1}$ είναι προτιμότερος επειδή μπορεί να παρασταθεί με ακρίβεια ως ακέραιος και να χρησιμοποιηθεί με ευκολία στον κώδικα.

Τα πειράματα που γίνανε σε όλους τους Β.Ε. καταλήξανε στις ακόλουθες τιμή για τον όρο $(C \cdot A \cdot B)^{-1}$

Β.Ε.	1	2	3	4	5	6	7	8	9	10
$(C \cdot A \cdot B)^{-1}$	417	425	416	414	411	412	413	411	410	;

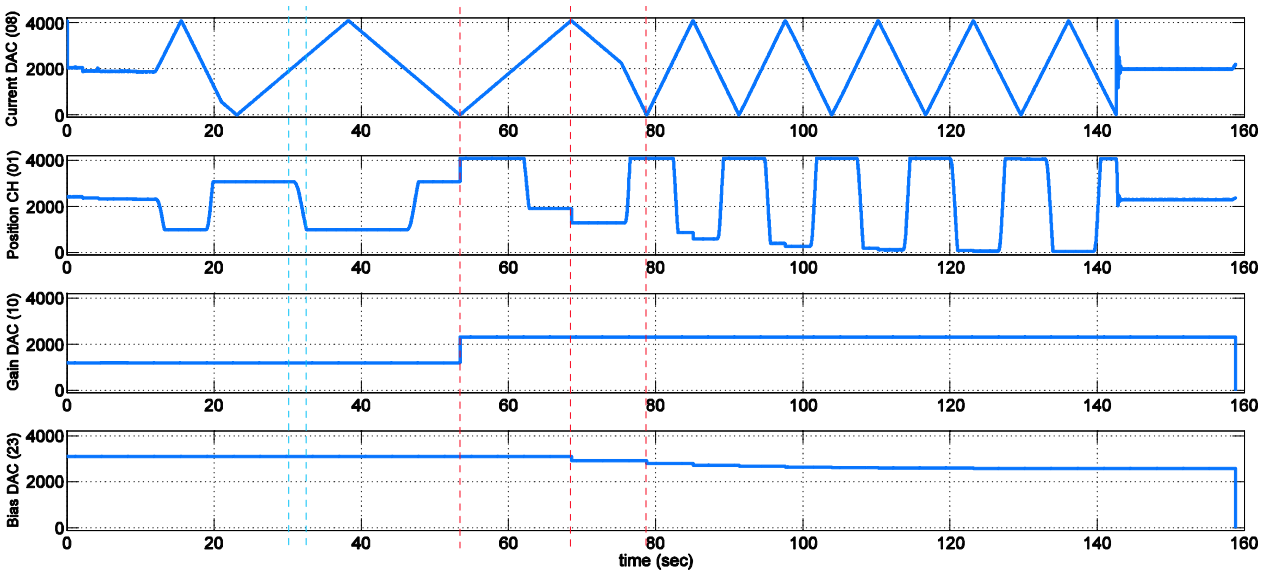
Πίνακας 7-3 Γινόμενο $(C \cdot A \cdot B)^{-1}$ για τα κυκλώματα βαθμονόμησης θέσης (πρώτου σταδίου).

7.3.2 Βαθμονόμηση Με την Μέθοδο Μονού Καναλιού

Η ρουτίνα βαθμονόμησης θέσης διαβάζει, θέση, ρεύμα και τάση από τα κανάλια 01, 06, 07, 08 αντίστοιχα. Ο λόγος είναι πως εκτός από βαθμονόμηση μπορεί με αυτό τον τρόπο να γίνει ένας αρχικός έλεγχος της λειτουργικότητας το Β.Ε.

Παρουσιάζεται η βαθμονόμηση με την μέθοδο μονού καναλιού για τον 4° Β.Ε. λόγω του ότι είχε παρουσιάσει πρόβλημα στο ηλεκτρικό κύκλωμα οπότε ήταν ο .καλύτερος υποψήφιος για ενδελεχή έλεγχο. Η βαθμονόμηση δεν έγινε στο πλήρες διάστημα [0…4095] αλλά στο διάστημα [32 … 4064] για να υπάρχει απόλυτη βεβαιότητα ότι το σήμα δεν θα παρουσιάσει κορεσμό. Στο Σχήμα 7-11 παρουσιάζονται

1. Η ψηφιακή εντολή ρεύματος ρεύμα που αποστέλλεται στην βαλβίδα (DAC 08)
2. Η θέση όπως διαβάζεται από τον αισθητήρα (CH 01)
3. Το ψηφιακό κέρδος (gain) του σταδίου βαθμονόμησης (DAC 10)
4. Η ψηφιακή πόλωση (bias) του σταδίου βαθμονόμησης (DAC 23)



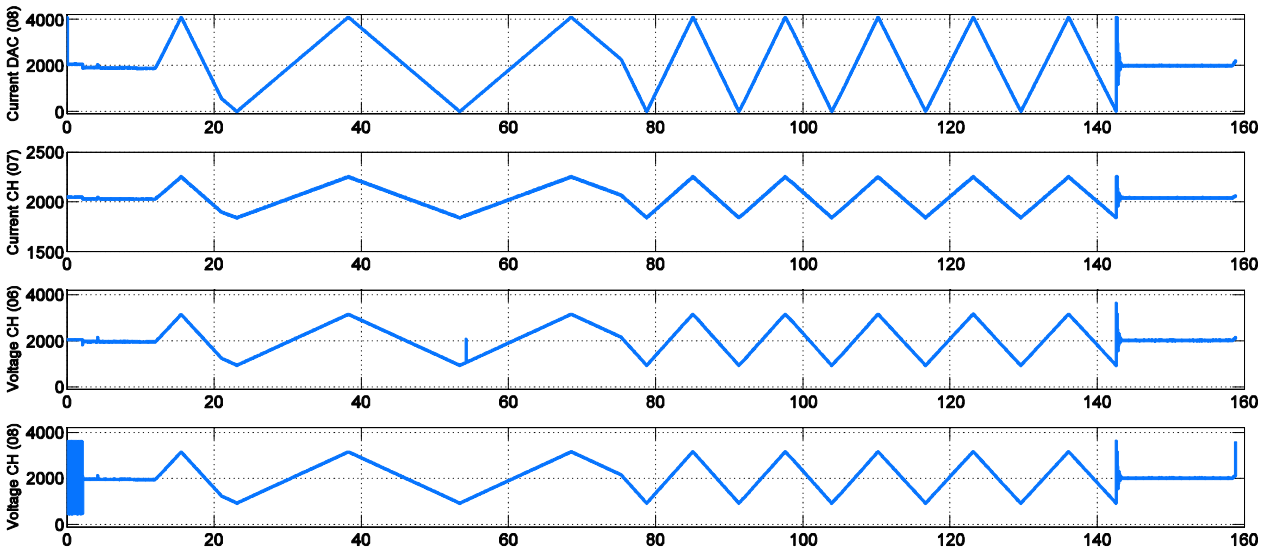
Σχήμα 7-11 Κίνηση Β.Ε. κατά την βαθμονόμηση αισθητήρα θέσης (1).

Η διαδικασία περιγράφεται με τα ακόλουθα βήματα

1. Αρχικά ο βραχίονας είναι σε P control, ώστε να τοποθετηθεί σε θέση βολική για βαθμονόμηση το Β.Ε.
   - μακριά από εμπόδια
   - σε θέση που η κίνηση του Β.Ε. έχει μικρές απαιτήσεις ροπής
   - Για να μπορεί ο βραχίονας να είναι σε P control πρέπει το σήμα θέσης όλων των Β.Ε. να μην είναι σε κορεσμό, ώστε να μπορούν να κινηθούν από τα ποτενσιόμετρα. Άρα μια σχετικά καλή αρχική επιλογή των κερδών και των πολώσεων είναι απαραίτητη.
2. Μετά από προκαθορισμένο χρόνο (στο παράδειγμα ~15sec), ο Β.Ε. μπαίνει σε κατάσταση βαθμονόμησης, μέσω απευθείας ελέγχου του ρεύματος, ενώ όλοι οι άλλοι παραμένουν σε P control.

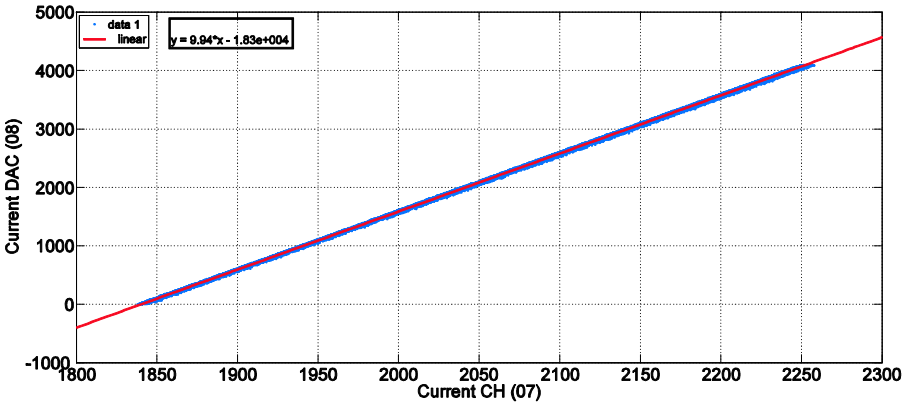
3. Καθώς το ρεύμα μεταβάλλεται από μέγιστο σε ελάχιστο ο Β.Ε. κινείται και λαμβάνει εναλλάξ τις δυο ακραίες θέσεις. Κατά την διάρκεια της κίνησης υπολογίζονται η μέγιστη και ελάχιστη τιμή του ψηφιακού σήματος θέσης (CH01)
    - Παράδειγμα κίνησης είναι μεταξύ των δυο κάθετων γαλάζιων διακεκομμένων γραμμών
4. Αφού ο Β.Ε. κινηθεί δυο φορές σε όλο το εύρος ξεκινάει η διαδικασία βαθμονόμησης όπως περιγράφεται στο αντίστοιχο κεφάλαιο.
    - Οι τρείς πρώτες επαναλήψεις της διαδικασίας ορίζονται από τις τρείς κόκκινες διακεκομμένες γραμμές
    - **Προσοχή**, η απότομη μεταβολή στην έξοδο του αισθητήρα θέσης **δεν** οφείλεται σε μετακίνηση, αλλά στην μεταβολή λόγω βαθμονόμησης είτε του κέρδους είτε της πόλωσης.
5. Αρχικά αλλάζει το κέρδος (gain) και αυτό έχει ως αποτέλεσμα η έξοδος να εμφανίζει κορεσμό άνω
6. Στην συνέχεια το σήμα κεντράρεται με διαδοχικές μεταβολές της πόλωσης (bias)
7. Στο τέλος γίνονται μικρο-ρυθμίσεις στο κέρδος και στην πόλωση, η έξοδος του αισθητήρα σταθεροποιείται στο επιθυμητό διάστημα η ρουτίνα τερματίζει (στο παράδειγμα ~142sec).
8. Ο βραχίονας παραμένει υπό έλεγχο, σε P control, μέχρι να κλείσει η παροχή λαδιού οπότε τερματίζεται το πείραμα.

Λόγω του ότι το ρεύμα μεταβάλλεται πολύ αργά μπορεί να γίνει αξιολόγηση των DC χαρακτηριστικών του κυκλώματος ισχύος. Για τον λόγο αυτό, διαβάζονται τα κανάλια CH06, CH07, CH08. Η απόκριση στον χρόνο σε αντιπαράθεση με την ψηφιακή είσοδο ρεύματος (DAC08) φαίνεται στο Σχήμα 7-12



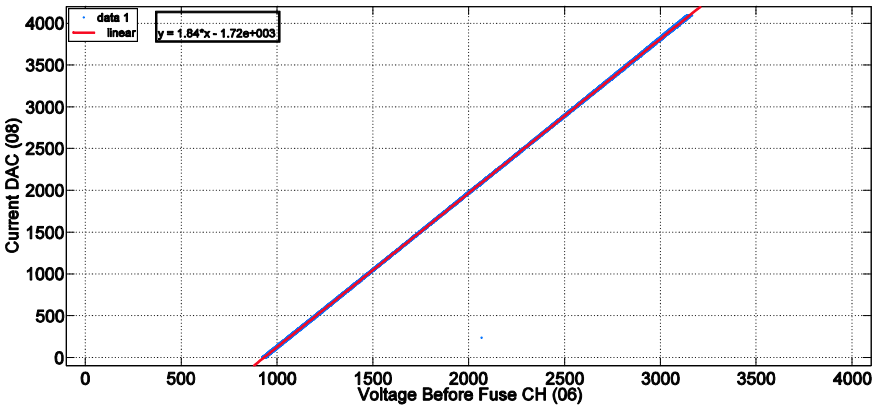
Σχήμα 7-12 Κίνηση Β.Ε. κατά την βαθμονόμηση αισθητήρα θέσης (1).

Ενδιαφέρουσα είναι η γραφική παράσταση μεταξύ ψηφιακής εντολής ρεύματος (DAC08) και του ρεύματος που ο ADC διαβάζει από το κανάλι CH07, Σχήμα 7-13. Η αναλογία δεν είναι η αναμενόμενη $1:1$ αλλά $1:10$. Η ερμηνεία είναι κυκλωματική και δίνεται στο Παράρτημα.



Σχήμα 7-13 Καμπύλη ρεύματος (DAC08) - ρεύματος (CH07) για τον 4° Β.Ε.

Η γραφική παράσταση μεταξύ ρεύματος και τάσης παρέχει μια καλή εκτίμηση της DC αντίστασης του κυκλώματος της βαλβίδας.



Σχήμα 7-14 Καμπύλη τάσης - ρεύματος γι τον 4° Β.Ε.

Η τάση που παρέχει το κύκλωμα εξόδου έχει εύρος [-22..22]V, ενώ ο ADC δέχεται τάσεις με εύρος [-10…10]. Για αυτό τον λόγο, γίνεται μια διαίρεση της τάσης εξόδου δια του δυο πριν τον ADC. Οπότε, κατ εξαίρεση για τα CH6 και CH8, το εύρος ψηφιακό εύρος [0…4095] αντιστοιχεί σε εύρος τάσης [-20 ... 20]V.

Εάν v, i είναι οι αναλογικές τιμές της τάσης και του ρεύματος, ενώ V, I είναι οι αντίστοιχες ψηφιακές τιμές τότε

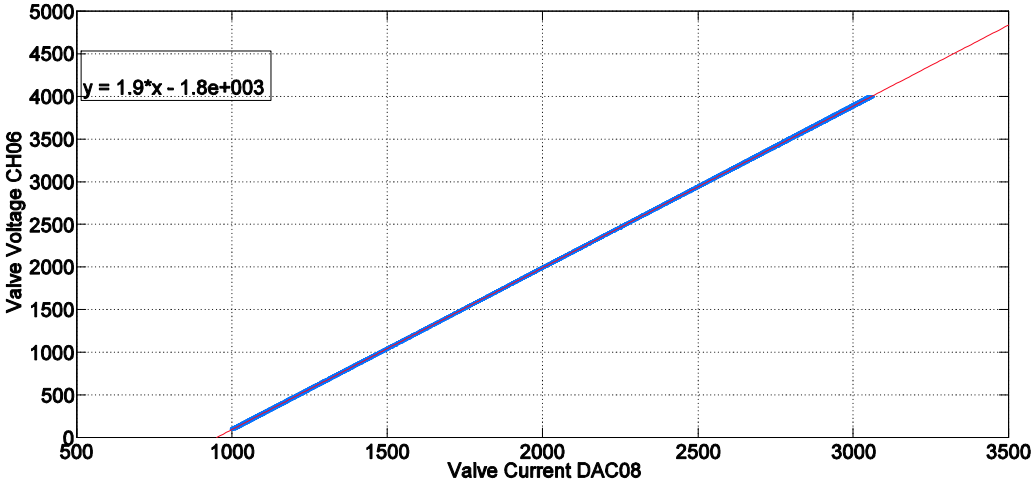
$$\left\{\begin{array}{l} \dfrac{\Delta V}{\Delta v} = \dfrac{4096}{40} \\ \dfrac{\Delta \mathrm{I}}{\Delta \imath} = \dfrac{4096}{2} \end{array}\right\} \Rightarrow \frac{\Delta V}{\Delta \mathrm{I}} = \frac{\Delta v}{\Delta i} \cdot \frac{1}{20} \Rightarrow \frac{\Delta v}{\Delta i} = 20 \cdot \frac{\Delta V}{\Delta \mathrm{I}}$$

Όμως, από το Σχήμα 7-14

$$\frac{\Delta I}{\Delta V}=1.84 \Rightarrow \frac{\Delta V}{\Delta \mathrm{I}}=\frac{1}{1.84} \Rightarrow \frac{\Delta v}{\Delta i}=\frac{20}{1.84} \Rightarrow \frac{\Delta v}{\Delta i}=10.9$$

Άρα η βαλβίδα εμφανίζει με μια DC αντίσταση 10.9Ω. Τα τεχνικά χαρακτηριστικά της βαλβίδας προδιαγράφουν αντίσταση 8Ω. Σε αυτή πρέπει να προστεθεί και μια αντίσταση 1ohm που είναι σε σειρά για την μέτρηση του ρεύματος, άρα συνολικά έπρεπε να μετρηθούν περίπου 9ohm. Η μετρούμενη τιμή είναι 20% υψηλότερη της αναμενόμενης. Η διαφορά αυτή πιθανώς οφείλεται σε προβλήματα του ηλεκτρικού κυκλώματος.

Αντίστοιχα πειράματα έγιναν και στον 2° Β.Ε. ο οποίος επίσης παρουσίαζε προβλήματα η γραφική παράσταση φαίνεται στο Σχήμα 7-15.



Σχήμα 7-15 Καμπύλη τάσης - ρεύματος για τον 2° Β.Ε.

Η αντίσταση που υπολογίστηκε είναι 10.5Ω ποσοστό 17% υψηλότερο από το αναμενόμενο.

Μετά το πέρας τις βαθμονόμησης για όλους τους βαθμούς ελευθερίας τα κατάλληλα κέρδη και πολώσεις για όλους τους Β.Ε. περιγράφονται στον Πίνακα 7-4

Πίνακας 7-4 Κέρδη και πολώσεις του πρώτου σταδίου βαθμονόμησης θέσης

Β.Ε.	1	2	3	4	5	6	7	8	9	10
DAC10	2225	2264	2203	2318	2310	2278	2412	2018	2464	;
DAC23	2590	2667	2609	2589	2564	2569	2571	2541	2804	;

7.4.Βαθμονόμηση αισθητήρων ροπής

7.4.1.Υπολογισμός γινομένου CAB

Για τον υπολογισμό του γινομένου $C \cdot A \cdot B$ φορτώνεται στο DAC30 η τιμή 410. Ο βραχίονας μένει σταθερός στην θέση ηρεμίας του και μεταβάλλεται, αργά, η τιμή του DAC25. Αυτό έχει σαν αποτέλεσμα την μεταβολή του σήματος από το κανάλι CH03. Στο Σχήμα 7-8 φαίνονται οι αντίστοιχες μεταβολές.

Με την ίδια μεθοδολογία που ακολουθείται και στην Παράγραφο 7.3.1 υπολογίζεται το γινόμενο $(C \cdot A \cdot B)^{-1}$ και παρατίθεται στον Πίνακα 7-5

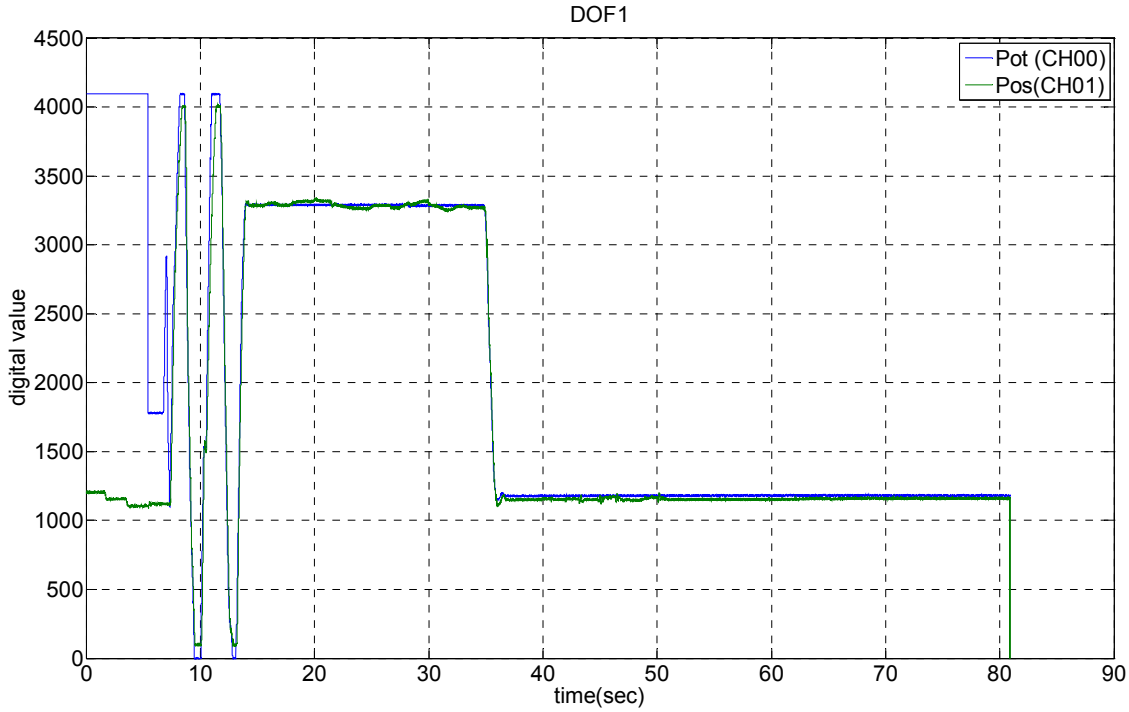
Πίνακας 7. 5 Γινόμενο $(C \cdot A \cdot B)^{-1}$ για τα κυκλώματα βαθμονόμησης ροπής.

Β.Ε.	1	2	3	4	5	6	7	8	9	10
$(C \cdot A \cdot B)^{-1}$	823	1209	1165	1126	1209	3628	3575	3513	;	;

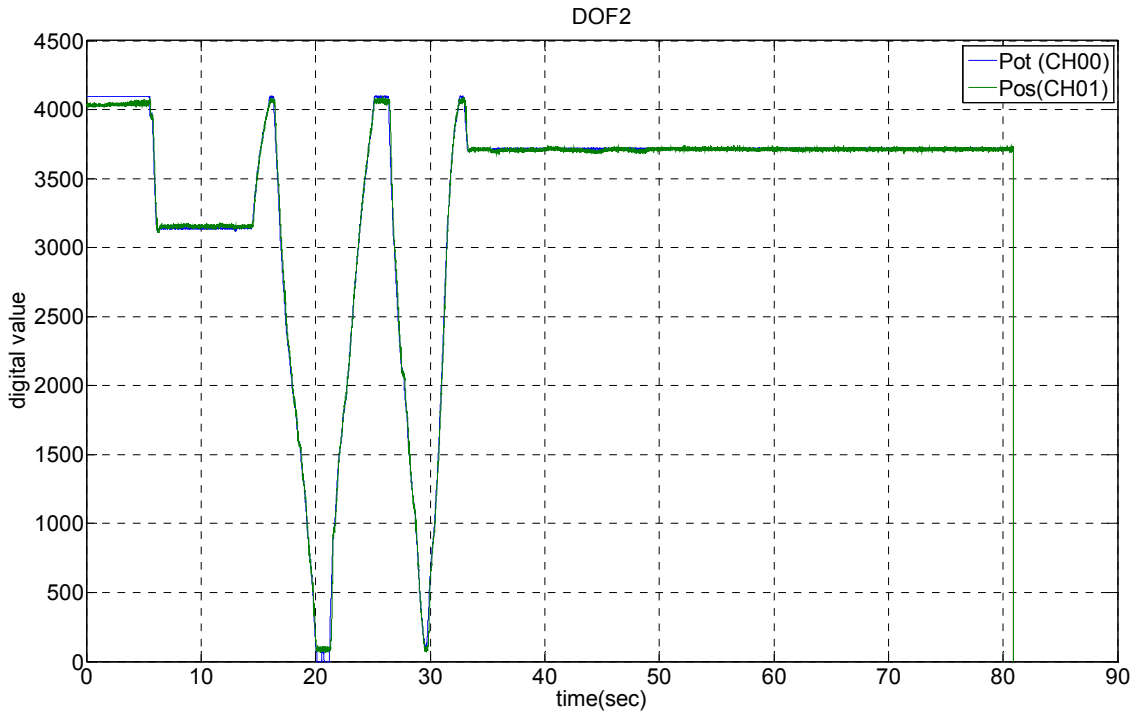
7.5.Έλεγχος από Ποτενσιόμετρα Μέσω Ρ Ψηφιακού Βρόχου

Παρακάτω παρατίθενται γραφικές παραστάσεις από πείραμα ελέγχου του βραχίονα , με το σήμα επιθυμητής εισόδου να παρέχεται από τα ποτενσιόμετρα που βρίσκονται στις προσόψεις των καρτών AJC.

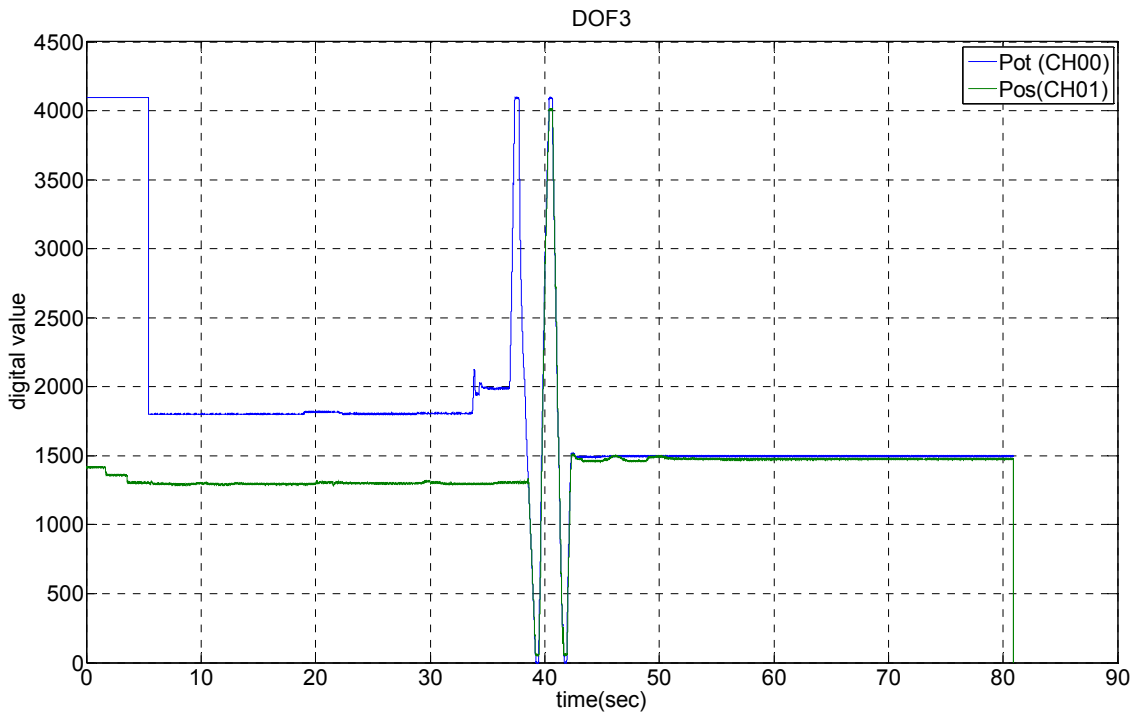
- Αρχικά όλοι οι Β.Ε. είναι κλειδωμένοι (Ο) στην θέση ηρεμίας
- Στην συνέχεια ο έλεγχος μεταφέρεται στα ποτενσιόμετρα (Κ).
- Οι Β.Ε. κινούνται διαδοχικά από τον 1° έως τον 9°
    - Για αποφυγή απότομων κινήσεων, κάθε Β.Ε. παραμένει κλειδωμένος στην αρχική του θέση μέχρι το σήμα από το αντίστοιχο ποτενσιόμετρο (μπλε) να πλησιάσει το σήμα θέσης (πράσινο)
    - Κάθε βραχίονας ξεκλειδώνει (k) και ακολουθεί το σήμα από το ποτενσιόμετρο.
- Στο τέλος όλοι οι Β.Ε. αυτόματα τοποθετούνται στην θέση ηρεμίας και κλειδώνουν εκεί
- Το πείραμα τερματίζει με την διακοπή της πίεσης στον βραχίονα.



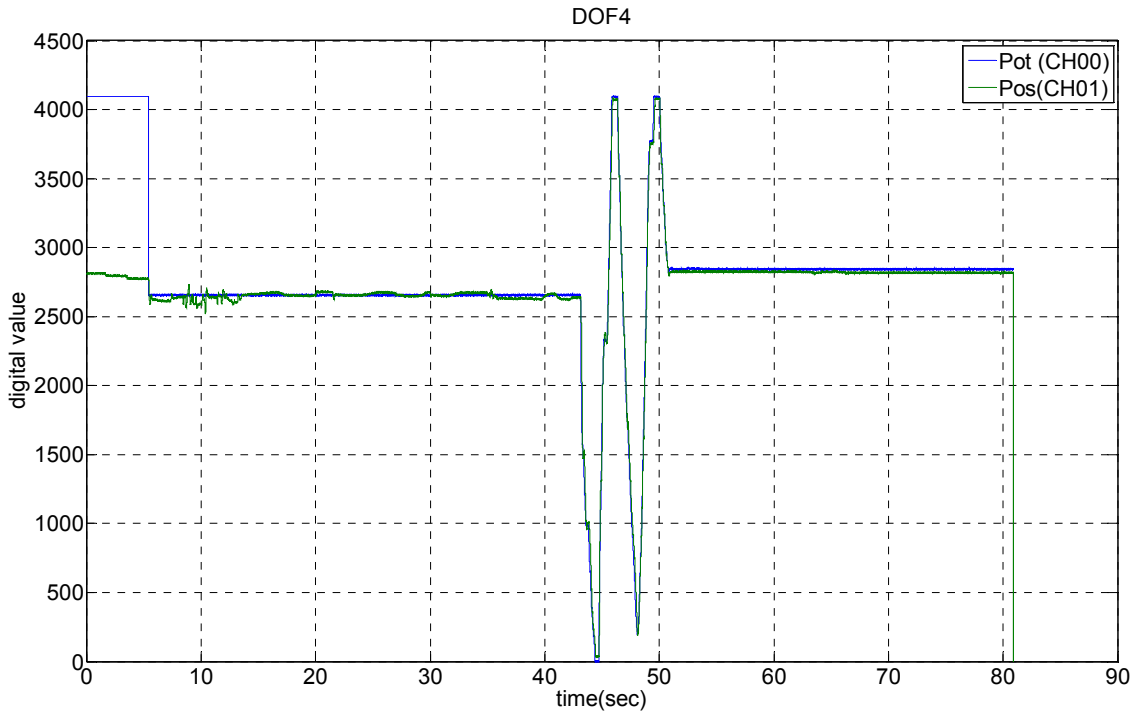
(α)


DOF2
Pot (CH00)
Pos(CH01)
digital value
time(sec)
4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
10
20
30
40
50
60
70
80
90

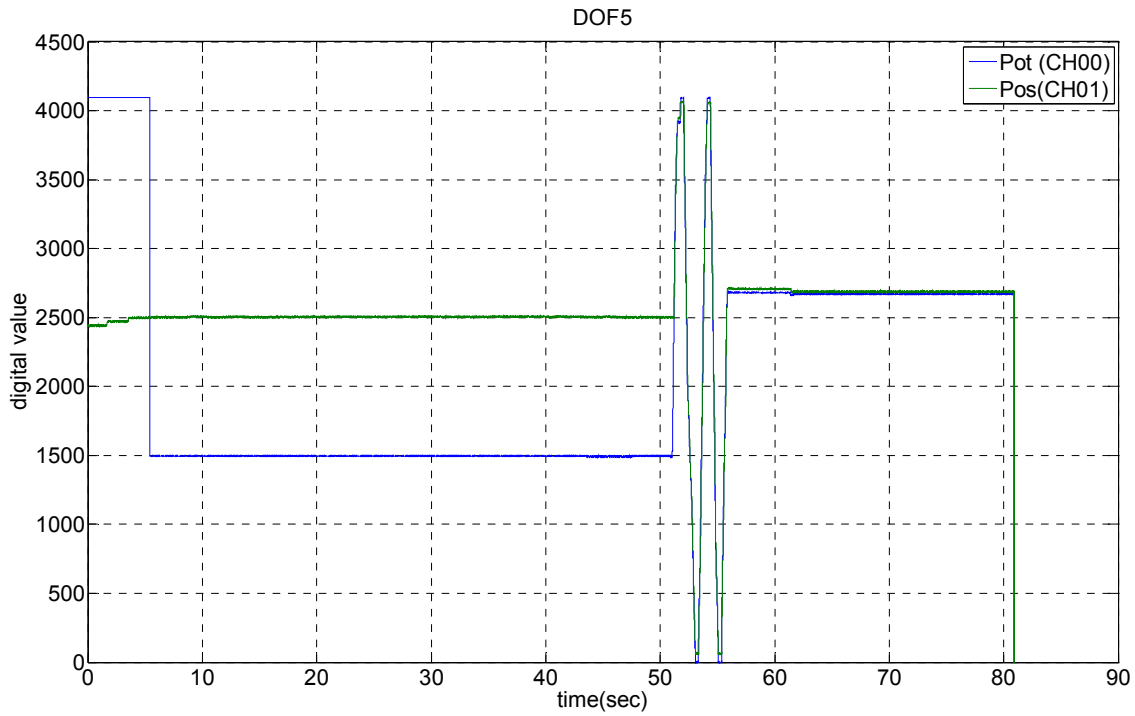
(β)


DOF3
Pot (CH00)
Pos(CH01)
digital value
time(sec)
4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
10
20
30
40
50
60
70
80
90

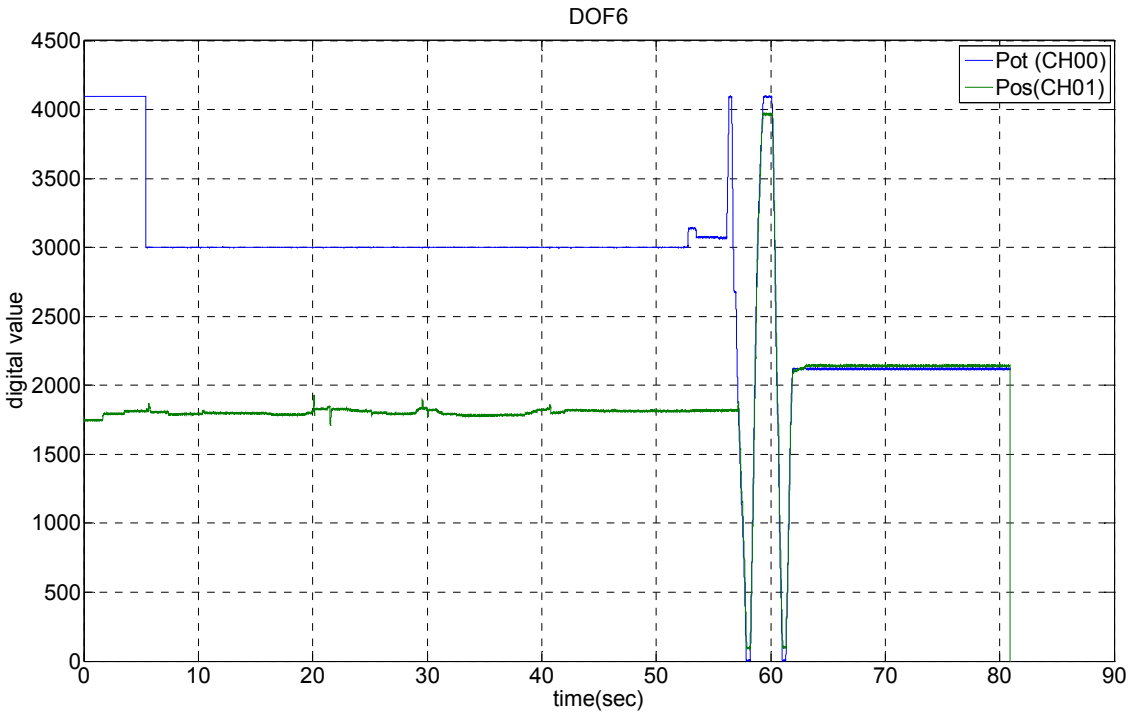
(γ)


DOF4
Pot (CH00)
Pos(CH01)
digital value
time(sec)
4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
10
20
30
40
50
60
70
80
90

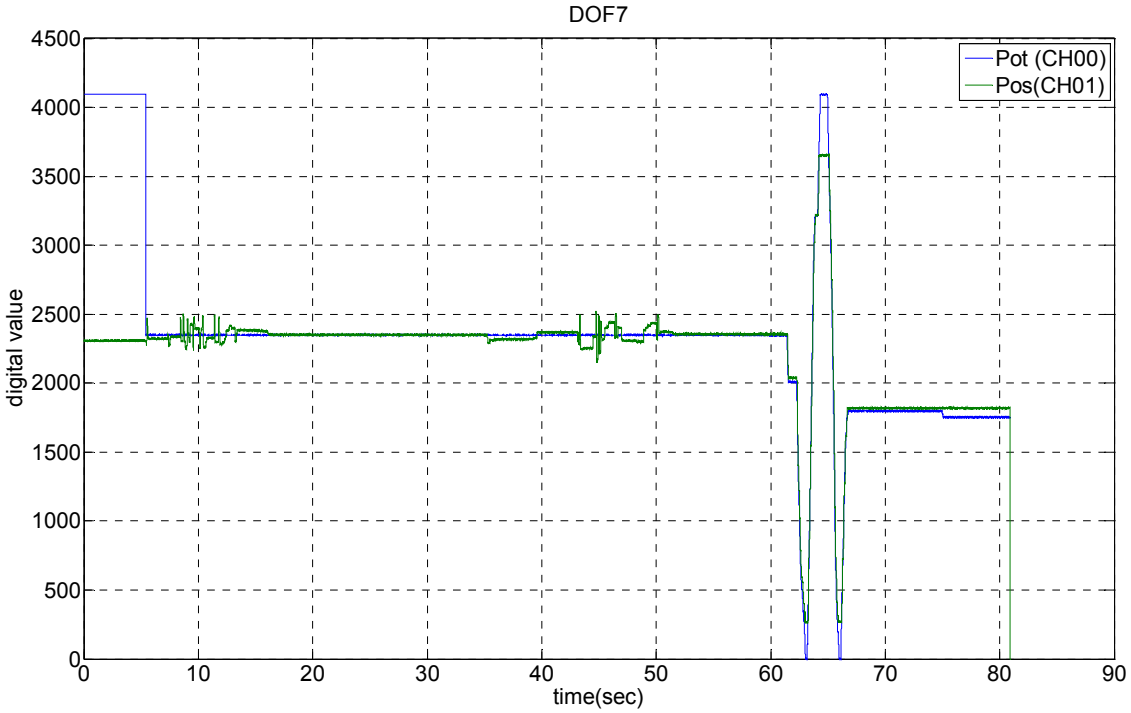
(δ)


DOF5
Pot (CH00)
Pos(CH01)
digital value
time(sec)
4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
10
20
30
40
50
60
70
80
90

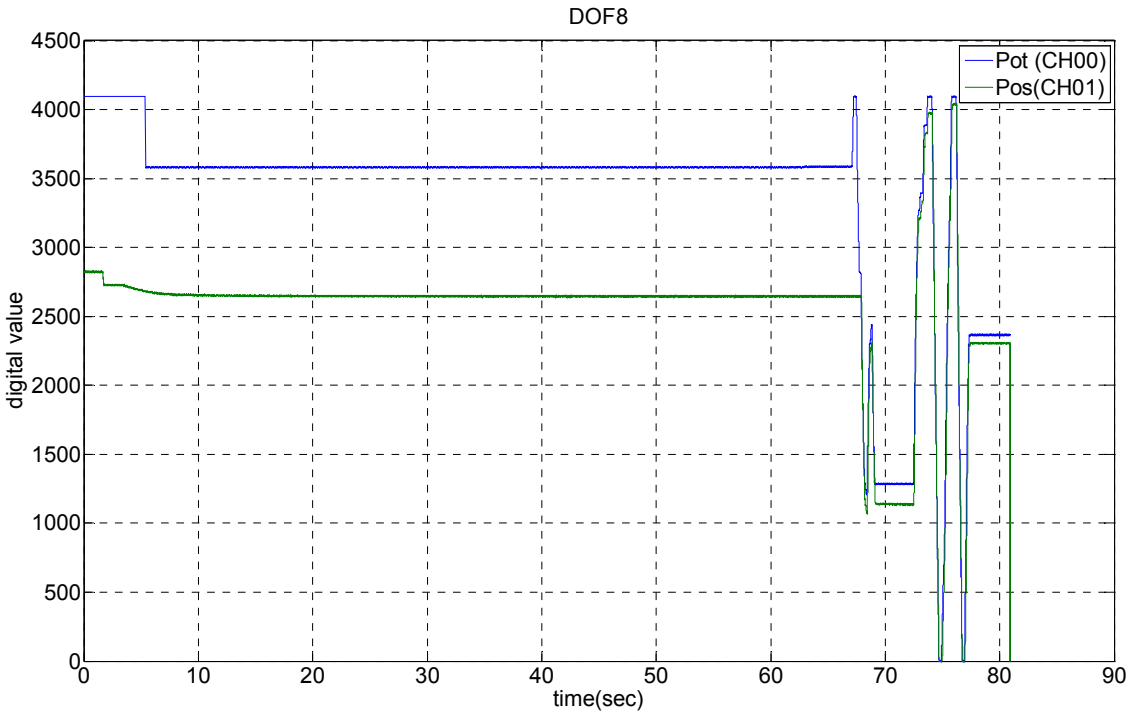
(ε)


DOF6
Pot (CH00)
Pos(CH01)
digital value
time(sec)
0
500
1000
1500
2000
2500
3000
3500
4000
4500
0
10
20
30
40
50
60
70
80
90

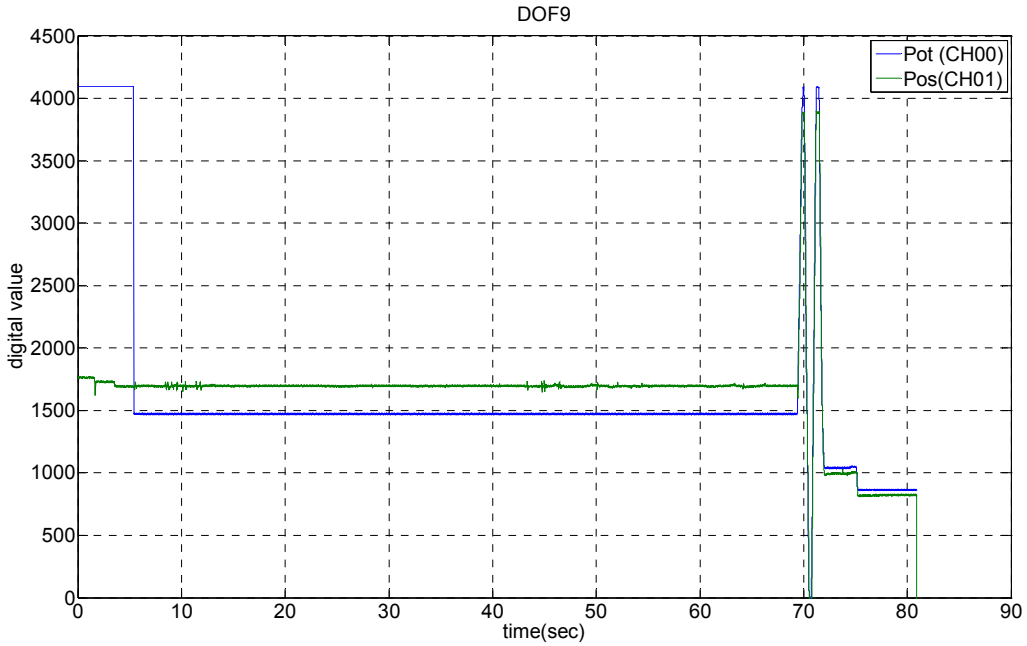
(στ)


DOF7
Pot (CH00)
Pos(CH01)
digital value
time(sec)
0
500
1000
1500
2000
2500
3000
3500
4000
4500
0
10
20
30
40
50
60
70
80
90

(ζ)



(η)



(θ)

Σχήμα 7-16 (α έως θ) Κίνηση βραχίονα με P ψηφιακό έλεγχο από ποτενσιόμετρα.

8 Συμπεράσματα

8.1 Αποτίμηση αποτελεσμάτων

Το σύστημα είχε έρθει από το πανεπιστήμια McGill χωρίς τον συνοδεύοντα υπολογιστή υποστήριξης (host computer) και χωρίς το λογισμικό. Ως εκ τούτου έπρεπε να αναπτυχθούν εκ νέου τόσο τα ηλεκτρονικά για την επικοινωνία με υπολογιστή όσο και το σχετικό λογισμικό.

Σαν κύριους στόχους της η εργασία αυτή είχε την αποκατάσταση της αμφίδρομης επικοινωνίας μεταξύ του SARCOS και ενός PC, εξοπλισμένου με μια κάρτα NI-DIO96, ως υπολογιστή υποστήριξης, για την υλοποίηση ψηφιακού ελέγχου. Απαραίτητα ενδιάμεσα στάδια ήταν ο έλεγχος λειτουργικότητας των επενεργητών και η βαθμονόμηση των αισθητήρων. Για την επίτευξη των παραπάνω, σχεδιάστηκαν και κατασκευάστηκαν ηλεκτρονικά, καθώς και το απαραίτητο λογισμικό.

Τα συμπεράσματα που αποκομίσθηκαν κατά την διαδικασία κατασκευής των ηλεκτρονικών, του λογισμικού και κατά την διενέργεια των πειραμάτων είναι τα εξής:

- Η χρήση της κάρτας NI-DIO είναι περιοριστική όσον αφορά τόσο τα επιτρεπόμενα πρωτόκολλα επικοινωνία που παρέχει, όσο και την ευχέρεια πρόσβασης στους οδηγούς (drivers) και του τρόπου λειτουργίας τους.
    - Λόγω της αδυναμίας άμεσης συνεργασίας της NI-DIO με τον δίαυλο του SARCOS αναπτύχθηκε ένα επιπλέον κύκλωμα, ενδιάμεσο, το οποίο αναλαμβάνει τον χειρισμό των απαραιτήτων σημάτων.
    - Το γεγονός ότι οι οδηγοί της κάρτας NI-DIO δεν είναι ανοιχτού κώδικά, δεν επιτρέπει την ανάπτυξη εφαρμογών πραγματικού χρόνου, αφού, ακόμα και εάν το λειτουργικό είναι πραγματικού χρόνου η συμπεριφορά της κάρτας θα είναι πάντα υπό αμφισβήτηση.
    - Για να μπορέσει η NI-DIO να λειτουργεί σε ικανοποιητικές συχνότητες, λειτουργεί σε strobed mode το οποίο σημαίνει ότι, με μια εντολή DAQmx I/O ανταλλάσσεται ένα πακέτο (package) πληροφοριών. Επιπλέον οι κάρτες AJC έχουν ένας ιδιόμορφο τρόπο απόδοσης διευθύνσεων και αναφοράς στα ψηφιακά κανάλια. Ο συνδυασμός των παραπάνω οδηγεί στην ανάγκη ενός επιπέδου αφαίρεσης στο λογισμικό (abstraction layer) το οποίο κοστίζει υπολογιστικά.
- Οι έλεγχοι που έχουν γίνει ως τώρα αφορούν τον slave βραχίονα και έχουν δείξει πως τα ηλεκτρομηχανικά τμήματα του συστήματος είναι σε καλή κατάσταση.
- Η εφαρμογή που αναπτύχθηκε εφαρμόζει PID έλεγχο στον βραχίονα έτσι ώστε οι κινήσεις του, κατά τις διαδικασίες ελέγχου και βαθμονόμησης, να είναι εντός επιθυμητών ορίων προς αποφυγή συγκρούσεων.
    - Παρέχεται η δυνατότητα ελέγχου της κίνησης των Β.Ε., καθώς και των DC χαρακτηριστικών τους. Οι βαλβίδες λειτουργούν κανονικά και όλοι οι Β.Ε. κινούνται απρόσκοπτα σε όλο το δυνατό εύρος κίνησης.
    - Οι αισθητήρες θέσης είναι λειτουργικοί, βαθμονομημένοι και ανά πάσα στιγμή μπορούν επαναβαθμονομηθούν σύμφωνα με τον την προτεινόμενη μέθοδο ανάλογα με τις απαιτήσεις της εφαρμογής.

- Οι αισθητήρες ροπής δεν είναι βαθμονομημένοι.
- Οι encoder των 1^{ου} , 2^{ου} , 4^{ου} , 5^{ου} Β.Ε. είναι λειτουργικοί, όμως ο encoder του 3^{ου} Β.Ε. είναι μη λειτουργικός. Μια επισκόπηση στην κρύπτη της βαλβίδας του 2^{ου} Β.Ε., για λόγο μη σχετιζόμενο με τον encoder, είχε αποκαλύψει κάποια κομμένα καλώδια που ίσως να ερμηνεύουν αυτή την συμπεριφορά.
- Η εφαρμογή που αναπτύχθηκε παρέχει την δυνατότητα για αναγνώριση παραμέτρων ανοιχτού ή κλειστού βρόχου. Έγιναν μόνο δοκιμαστικά πειράματα χωρίς προσπάθεια επεξεργασίας των αποτελεσμάτων.

8.2 Επόμενα βήματα και προτάσεις

Μελλοντική εργασία η οποία μπορεί να υλοποιηθεί άμεσα με σειρά προτεραιότητας είναι

- Βαθμονόμηση αισθητήρων ροπής
- Εγκατάσταση λειτουργικού σε πραγματικού χρόνου.
- Παραλληλοποίηση τμημάτων του κώδικα
- Εκτέλεση πειραμάτων αναγνώρισης παραμέτρων.

Παράλληλα προτείνεται η ανακατασκευή των ηλεκτρονικών έτσι ώστε να αυξηθεί η αξιοπιστία του συστήματος. Οι λόγοι που συνηγορούν προς αυτή την κατεύθυνση είναι οι ακόλουθοι.

- Η σύγχρονη τάση είναι ο έλεγχος να γίνεται ψηφιακά. Επιπλέον, η ανάπτυξη αλγορίθμων ελέγχου βάσει μοντέλου (model based control), απαιτεί την ανάγνωση των σημάτων από τους αισθητήρες και τον απ' ευθείας έλεγχο του ρεύματος των βαλβίδων. Άρα ένας μοντέρνος έλεγχος παρακάμπτει τελείως το τμήμα του αναλογικού PD ελέγχου των καρτών AJC.
- Η απαίτηση για έλεγχο πραγματικού χρόνου οδηγεί στην χρήση λειτουργικού πραγματικού χρόνου και σε βιβλιοθήκες πραγματικού χρόνου για όλο το συνοδό υλικό (hardware). Αυτή η απαίτηση εξαιρεί κάθε κάρτα με κλειστές βιβλιοθήκες (πχ NI-DIO).
- Η μέθοδος ανάγνωσης πληροφορίας από τα κανάλια, καθώς και ο μέθοδος εγγραφής πληροφορίας στα DAC είναι αμφισβητήσιμες, με τα σύγχρονα δεδομένα.
- Η ηλεκτρική σύνδεση μεταξύ υπολογιστή υποστήριξης και ηλεκτρονικών δεν επιτρέπει πραγματικό τηλεχειρισμό, αφού η μεταφορά σημάτων μέσω καλώδιο-ταινίας σε μεγάλες αποστάσεις είναι σαφώς προβληματική.
- Επιπλέον αυτού, δεν υπάρχει κάποιο σήμα επιστροφής από το SARCOS ώστε να μπορεί να υπολογιστεί η καθυστέρηση (lag) λόγω απόστασης, η γνώση της οποίας είναι σημαντική για τον βρόχο ελέγχου.
- Τα αναλογικά ηλεκτρονικά του τμήματος signal conditioning και του τμήματος ισχύος είναι πλέον παλιά και η απόδοση τους πιθανώς να είναι υποδεέστερη αντίστοιχων σύγχρονων industrial grade ηλεκτρονικών.
- Λόγω παλαιότητας των καρτών, η πιθανότητα βλάβης είναι αυξημένη. Η δυνατότητα επιδιόρθωσης περιορισμένη λόγω της έλλειψης, πλέον, αντίστοιχων εξαρτημάτων και της αυξημένης πυκνότητας εξαρτημάτων πάνω στις πλακέτες.

Προς αντιμετώπιση των παραπάνω ζητημάτων προτείνεται αντικατάσταση των καρτών BusRepV4 και AJC. Οι νέες κάρτες θα σχεδιαστούν με γνώμονα τα εξής

- Πλήρης γνώση των κυκλωμάτων και επάρκεια ανταλλακτικών

- Τοπικός έλεγχος, για ασφάλεια, σε περίπτωση διακοπής της επικοινωνίας.
- Τοπική υλοποίηση ψηφιακών φίλτρων για μείωση του φόρτου εργασίας του υπολογιστή υποστήριξης
- Συγχρονισμένη ανάγνωση των αισθητήρων.
- Συγχρονισμένη εγγραφή σημάτων ελέγχου.
- Βελτίωση, εάν κριθεί απαραίτητο, του τμήματος ισχύος
- Μέτρηση του χρόνου υστέρησης (lag) λόγω απόστασης
- Νέο στάδιο signal conditioning, με σύγχρονα ηλεκτρονικά εξαρτήματα που πιθανώς να επιτρέψουν την αύξηση της ευκρίνειας των υπαρχόντων αισθητήρων και την μείωση του λόγου σήματος προς θόρυβο (NSR).
- Σειριακής επικοινωνία, RS-424 ή RS-485, μέσω Ethernet ή αντίστοιχου πρωτόκολλου, ανοιχτού πηγαίου κώδικα, και με χρήση καρτών PCI 100Mbps Ethernet.

Για την υλοποίηση των κυκλωμάτων προτείνεται ο συνδυασμός

- OPAMP ή H-BRIDGE
    - Υποσύστημα ισχύος, έλεγχος ρεύματος στην βαλβίδα.

- FPAA (Field Programmable Analog Array)
    - Κατασκευή σημάτων διέγερσης αισθητήρων
    - Υλοποίηση προγραμματιζόμενων αναλογικών φίλτρων.
    - .Αναλογική επεξεργασία των σημάτων απόκρισης των αισθητήρων

- ADC ανάλυσης 12bit ή ανώτερης
    - ξεχωριστό για κάθε αισθητήρα.

- FPGA (Field Programmable Grid Array) ή CPLD
    - Διαχείριση (low level) ψηφιακών σημάτων.
    - Υλοποίηση ψηφιακών φίλτρων.

- Microcontroller
    - Δυνατότητα Δυναμικού προγραμματισμού των FPGA ,FPAA
    - Υλοποίηση πρωτοκόλλου σειριακής επικοινωνίας
    - Εκτέλεση του τοπικού βρόχου ελέγχου
    - Υλοποίηση ψηφιακών φίλτρων
    - Ανάγνωση των ADC

Μια τέτοια προσέγγιση προσφέρει ευελιξία στον σχεδιασμό, την κατασκευή και την τροποποίηση, αλλά και δυνατότητα άμεσης χρησιμοποίησης του εν λόγω κυκλώματος σε άλλες εφαρμογές.

Βιβλιογραφία

[1] Bilodeau, G. and Papadopoulos, E., "Development of a Hydraulic Manipulator Servoactuator Model: Simulation and Experimental Validation," Proc. IEEE International Conference on Robotics and Automation, (ICRA '97), April 21-27 1997, Albuquerque, NM.

[2] Bilodeau, G. and Papadopoulos, E., "Modelling, Identification and Experimental Validation of a Hydraulic Manipulator Joint for Control," Proceedings of the 1997 International Conference on Intelligent Robots and Systems (IROS '97), Grenoble, France, September 1997.

[3] Bilodeau, G. and Papadopoulos, E., "Experiments on a High Performance Hydraulic Manipulator Joint: Modelling for Control," Proceedings of the 1997 International Symposium on Experimental Robotics (ISER '97), Barcelona, Spain, June 1997.

[4] Bilodeau, G. and Papadopoulos, E. , "A Model-Based Impedance Control Scheme for High-Performance Hydraulic Joints," Proceedings of the 1998 International Conference on Intelligent Robots and Systems (IROS '98), Victoria, BC, Canada, October 1998.

[5] Καπαγερίδης Σ. «Αντίστροφη Σχεδίαση, Κινηματική Μελέτη Και Υλοποίηση Επαναλειτουργίας Του Ηλεκτροϋδραυλικού Τηλερομποτικού Συστήματος Sarcos» Μεταπτυχιακή Εργασία ΔΠΜΣ «Συστήματα Αυτοματισμού» 2004

[6] Παπαδόπουλος Ε., Κυριακόπουλος Κ., Σημειώσεις Ρομποτικής, Σχολή Μηχανολόγων Μηχανικών, ΕΜΠ.

[7] Ροδίτης Ι. «Ανάπτυξη εφαρμογής για εγγραφή κερδών ελέγχου και αποκατάσταση ελέγχου πίεσης του ηλεκτροϋδραυλικού τηλερομποτικού συστήματος Sarcos» Μεταπτυχιακή Εργασία ΔΠΜΣ «Συστήματα Αυτοματισμού» 2006

[8] SARCOS Research Corporation – SRC, Dextrous Teleoperation System Manual.

ΠΑΡΑΡΤΗΜΑ

Π.1.Πίνακες

Σε αυτό το τμήμα παρουσιάζονται διάφοροι πίνακες με λειτουργικές τιμές ή./και ρυθμίσεις για διάφορα υποσυστήματα του SARCOS

Π 1. 1 Ψηφιακά κέρδη AJC, προτεινόμενες τιμές ανά κάρτα.

Β.Ε. DAC	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
6	410	410	410	410	410	410	410	410	410	410
5	0	0	0	0	0	0	0	0	0	0
14	1230	1230	1230	1230	1230	820	820	410	410	410
13	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
10	2225	2264	2203	2318	2310	2278	2412	2018	2464	907
23	2590	2667	2609	2589	2564	2569	2571	2541	2804	2221
24	0	0	0	0	0	0	0	0	0	0
31	2048	2048	2048	2048	2048	2048	2048	2048	2048	2048
30	0	0	0	0	0	0	0	0	0	0
25	2048	2048	2048	2048	2048	2048	2048	2048	2048	2048
29	2048	2048	2048	2048	2048	2048	2048	2048	2048	2048
11	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0
28	2048	2048	2048	2048	2048	2048	2048	2048	2048	2048
27	2048	2048	2048	2048	2048	2048	2048	2048	2048	2048

Π 1. 2 Κέρδη Ψηφιακού PID ελέγχου.

Β.Ε. όρος	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Kp	3	-3	3	-3	3	2	2	1	1	1
Kd	8	-8	8	-10	8	3	3	1	1	1
Ki	3	-3	3	-2	3	1	1	1	1	1

Π 1. 3 Γινόμενο $(C \cdot A \cdot B)^{-1}$ για τα κυκλώματα βαθμονόμησης θέσης (πρώτο στάδιο)

Β.Ε.	1	2	3	4	5	6	7	8	9	10
$(C \cdot A \cdot B)^{-1}$	417	425	416	414	411	412	413	411	410	;

Π 1. 4 Γινόμενο $(C \cdot A \cdot B)^{-1}$ για τα κυκλώματα βαθμονόμησης θέσης (πρώτο στάδιο)

B.E.	1	2	3	4	5	6	7	8	9	10
$(C \cdot A \cdot B)^{-1}$	823	1209	1165	1126	1209	3628	3575	3513	;	;

Παρακάτω παρατίθενται σε δεκαεξαδικό σύστημα τα περιεχόμενα των PROM ,όπως αυτά διαβαστήκανε από τον προγραμματιστή XELTEK3000U. Η αριστερή στήλη αντιστοιχεί στην διεύθυνση του αριστερότερου byte. Σε κάθε σειρά παρατίθενται 16 bytes.

Π 1. 5 Περιεχόμενα PROM master

```
MASTER ARM PROM

00000000  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
00000010  FF FF FF FF FF 00 01 02-03 04 05 06 07 08 09 0A
00000020  0B 0C 0D 0E 0F 10 11 12-13 14 FF FF FF FF FF FF
00000030  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
00000040  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
00000050  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
00000060  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
00000070  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
00000080  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
00000090  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
000000A0  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
000000B0  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
000000C0  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
000000D0  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
000000E0  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
000000F0  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
```

Π 1. 6 Περιεχόμενα PROM slave

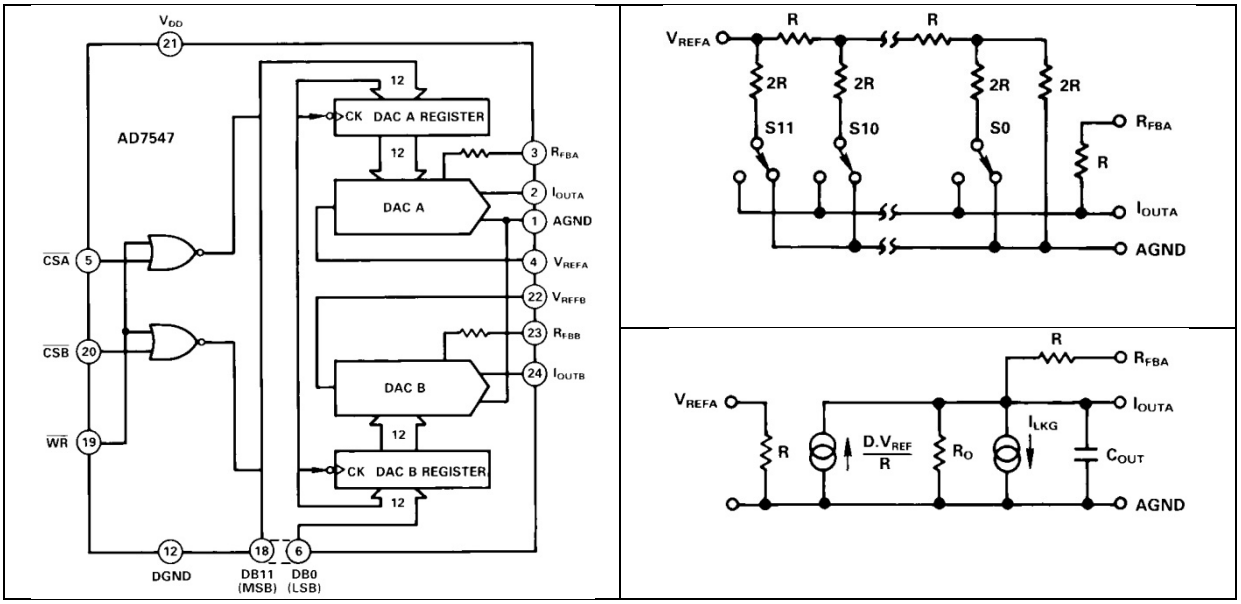
```
SLAVE ARM PROM

00000000  00 01 02 03 04 05 06 07-08 09 0A 0B 0C 0D 0E 0F
00000010  10 11 12 13 14 FF FF FF-FF FF FF FF FF FF FF FF
00000020  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
00000030  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
00000040  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
00000050  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
00000060  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
00000070  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
00000080  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
00000090  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
000000A0  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
000000B0  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
000000C0  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
000000D0  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
000000E0  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
000000F0  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
```

Είναι φανερό πως υπάρχει κανονικότητα που σχετίζεται άμεσα με το πεδίο των διευθύνσεων κάθε κλωβού (cage).

Π.2. Ανάστροφη Μηχανική και Ανάλυση Επιλεγμένων κυκλωμάτων

Π.2.1.Μετατροπή ψηφιακού σήματος στα DAC σε αναλογικό gain ή bias

Το DAC είναι το AD7547, το οποίο είναι 12bit και έρχεται σε DIP συσκευασία. Κάθε IC περιέχει δύο DACs. Το ψηφιακό (δίαυλοι και σήματα ελέγχου), και το αναλογικό ισοδύναμο, φαίνονται στο σχήμα …



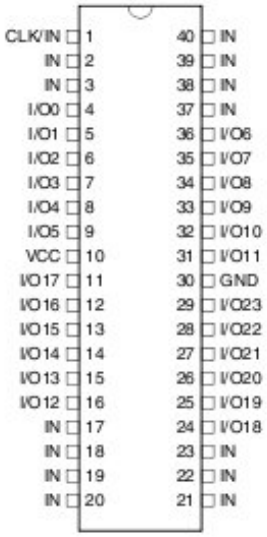
σχήμα Π.2. 1 Ψηφιακό και αναλογικό ισοδύναμο του DAC7547

Το DAC μετατρέπει την ψηφική τιμή D σε <u>ρεύμα</u> με αναλογία $I_{out} = D \cdot \frac{V_{ref}}{R}$. Στην συνέχεια απαιτήται μια βαθμίδα μετροπής από ρεύμα σε τάση για να παραχθεί μία τάση ανάλογη του D και V_{ref}.

Π.3.Προγραμματιζόμενη Λογική.

Π.3.1.CPLD ATF2500c

Το CPLD ATF2500C έρχεται σε δυο συσκευασίες PLCC44 και DIP40. Στην παρούσα εργασία χρησιμοποιήθηκαν οι συσκευασίες DIP40συσκευασία DIP40. Ακολουθεί μια περιγραφή του ολοκληρωμένου και των δυνατοτήτων του.



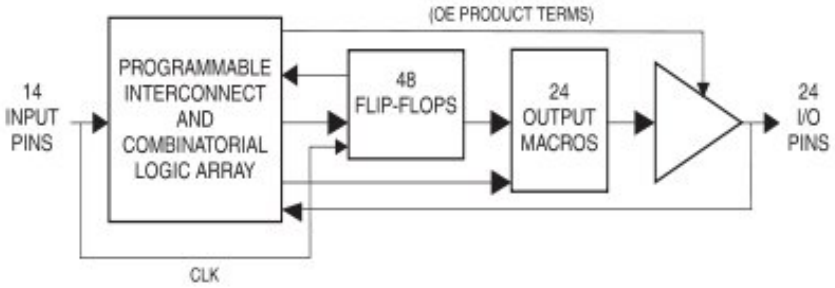
Σχήμα Π.3-1 Το ολοκληρωμένο ATMELATF2500C

Εξωτερικά, το ολοκληρωμένο έχει 40 pin σε DIP που χωρίζονται σε 3 κατηγορίες όπως παρουσιάζονται στον πίνακα Π.3-1

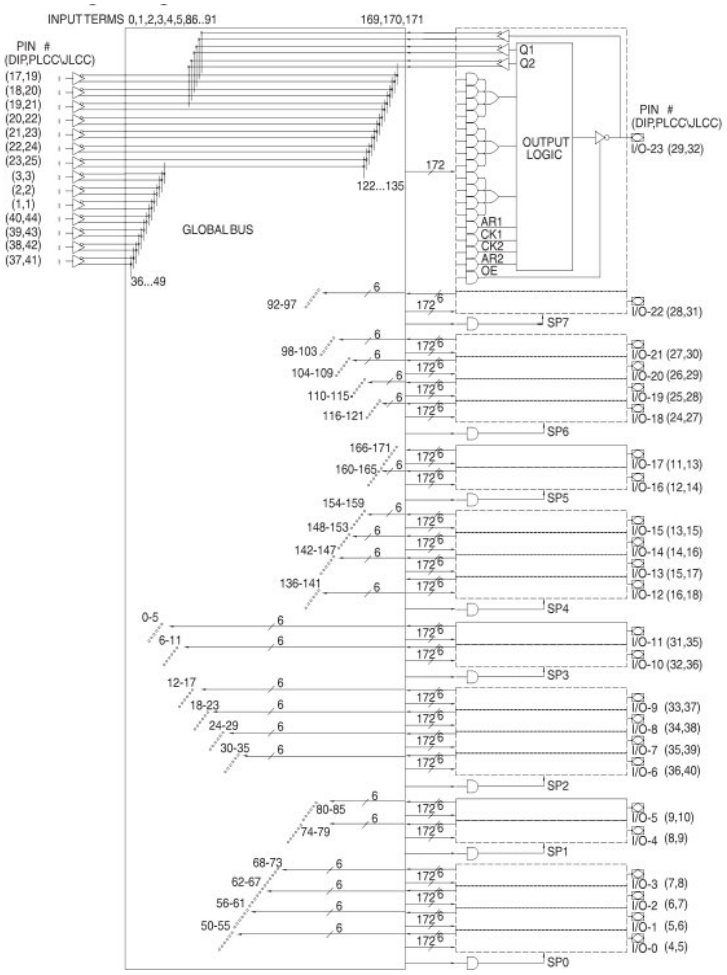
Πίνακας Π.2- 1 Οι ακροδέκτες του ATF2500C

Κατηγορία	pin#	Σύνολο
IN μόνο είσοδοι	1,2,3, 17,18,19,20 21,22,23, 37,38,39,40	14
IN / OUT Προγραμματιζόμενα ως είσοδοι ή έξοδοι ή αμφίδρομα (bidirectional)	4,5,6,7,8,9, 11,12,13,14,15,16, 23,24,25,26,27,28,29, 31,32,33,34,35,36	24
POWER	10(5Vcc) , 30 (GND)	2

- Εσωτερικά, το ολοκληρωμένο βασίζεται
  - σε ένα δίαυλο (global bus) εύρους 62 bit
  - σε 24 οργανωμένες μονάδες λογικής (macrocells).
- Στον δίαυλο καταλήγουν
  - τα 14 pin των εισόδων (IN)
  - οι $2 \times 24 = 48$ έξοδοι των macrocells



Σχήμα Π.3- 2 Block διάγραμμα macrocell

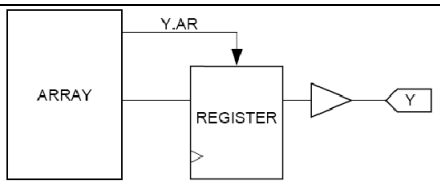
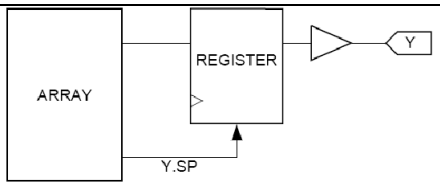
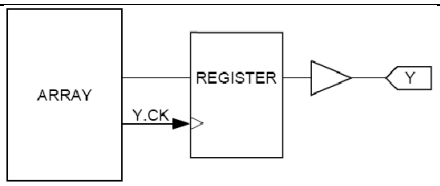


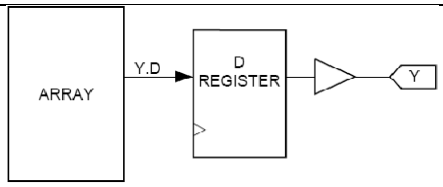
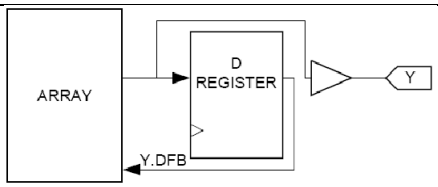
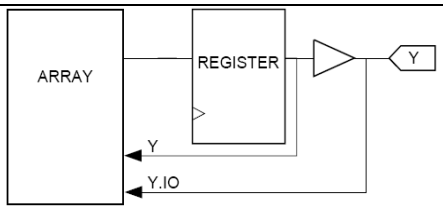
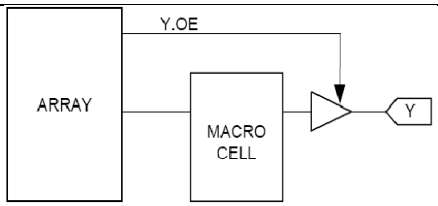
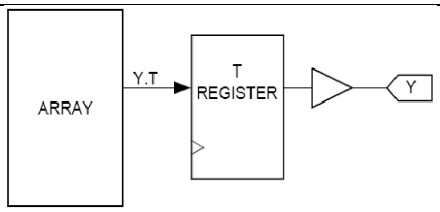
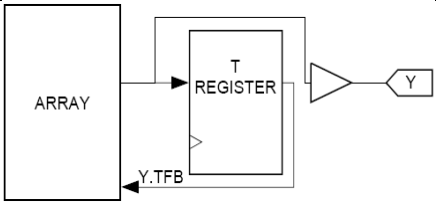
Σχήμα Π.3.-3 Εσωτερικός δίαυλος CPLD (global Bus)

- Κάθε macrocell μπορεί να οδηγήσει έως δυο γραμμές του διαύλου ανεξάρτητα και να δεχθεί κάθε μια από τις 62 γραμμές του διαύλου σαν είσοδο.
- Κάθε macrocell μπορεί να δεχθεί ως εισόδους, δυο SOPs (sum of products)
    - ένα για κάθε έξοδο, 8 και 4 όρων αντίστοιχα
    - να τα ενοποιήσει σε ένα 12 όρων, θυσιάζοντας την μία έξοδο,
    - ή να τα συνδυάσει σε δυο γινόμενα, 12 και 4 όρων αντίστοιχα, εάν ζητηθεί από τον χρήστη (product term shearing)
    - Κάθε γινόμενο (product) παράγεται από μια πύλη **AND** 62 εισόδων, που μπορεί δεχθεί όλες αδιακρίτως τις γραμμές του διαύλου.
- Κάθε macrocell περιέχει δυο flip-flop τα οποία μπορούν να προκαθοριστούν ως D ή T. Οι είσοδοι των flip-flop οδηγούνται από SOPs.
- Οι είσοδοι clock των flip-flop μπορούν να οδηγηθούν από ανεξάρτητα SOPs ή από ένα κοινό clock από το pin με αριθμό 1
- Τα I/O pin είναι bidirectional και ελέγχονται από ανεξάρτητους όρους γινομένων OE (output enable products) ή, εάν είναι επιθυμητή μεγαλύτερη πολυπλοκότητα, από SOPs από τον global bus
- Η είσοδος ενός I/O pin μπορεί να προσπελασθεί ανεξάρτητα από την έξοδο του εν λόγω pin.
- Υπάρχει η δυνατότητα προσαρμογής σε positive ή negative λογική, χωρίς να αλλάξει ο υφιστάμενος κώδικας, αλλάζοντας μόνο την πολικότητα των σημάτων εισόδου ή εξόδου κατά την δήλωση των αντίστοιχων pin.

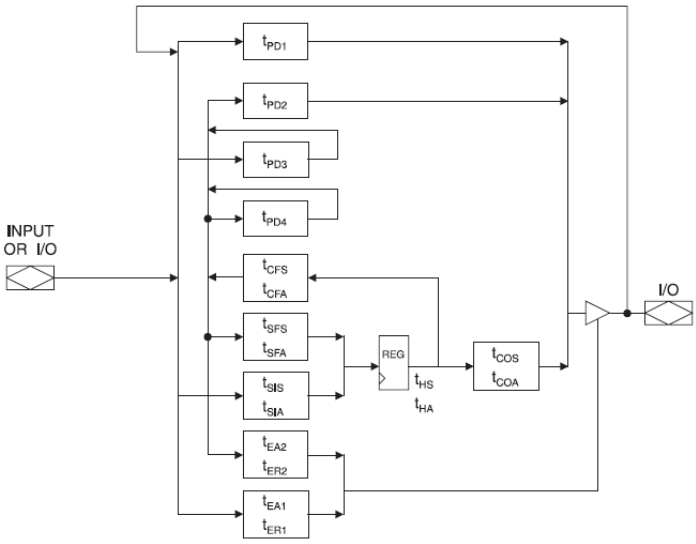
Παραδείγματα των παραπάνω δυνατοτήτων φαίνονται στον Πίνακα Π.2-2, (από το manual της ATMEL). Κάθε διαφορετικό προσάρτημα (extension) της μεταβλητής Y υποδηλώνει διαφορετική λειτουργία.

Πίνακας Π.2- 2 Προσαρτήματα και λειτουργείες των macrocells

	Y.AR SOP ασύγχρονο reset
	Y.SP SOP σύγχρονο preset
	Y.CK SOP είσοδος clock του flip-flop

	Y.D SOP είσοδος καταχωρητή διαμορφωμένου ως D, συνδεδεμένου σε pin εξόδου
	Y.DFB Καταχώρησης διαμορφωμένος ως D ανατροφοδοτούμενος στο global bus <u>Το SOP εισόδου του καταχωρητή συνδέεται και στο pin εξόδου</u>
	Y.IO Η <u>είσοδος</u> από το pin οδηγείται στον global bus, ανεξάρτητα από την τιμή της εξόδου
	Y.OE SOP output enable. Δυνατότητα bidirectional λειτουργίας (σε συνεργασία με το Y.IO) και σύνδεσης σε εξωτερικό δίαυλο παράλληλα με άλλες I/O συσκευές
	Y.T SOP είσοδος καταχωρητή διαμορφωμένου ως T, συνδεδεμένου σε pin εξόδου
	Y.TFB Καταχώρησης διαμορφωμένος ως T ανατροφοδοτούμενος στο global bus <u>Το SOP εισόδου του καταχωρητή συνδέεται και στο pin εξόδου</u>

Στο σχήμα … φαίνονται οι δυνατές διαδρομές των σημάτων μέσα στο CPLD και οι χρόνοι που αυτές απαιτούν για την ασφαλή διάδοση των σημάτων. Περαιτέρω πληροφορίες μπορούν να βρεθούν στις παραπομπές [4],[5],[9]



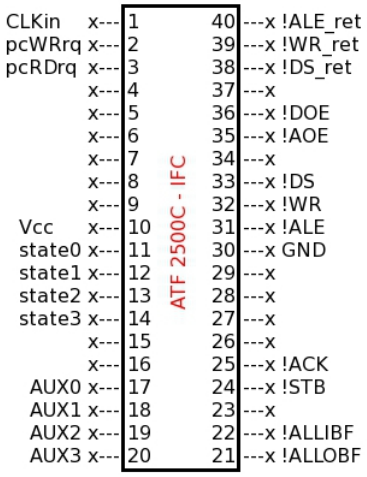
Σχήμα Π.3. 4 Εσωτερική διασύνδεση όρων και χρόνοι μετάδοσης σημάτων

Από πλευράς ηλεκτρικών χαρακτηριστικών

- οι είσοδοι έχουν μέγιστο ρεύμα 10μA,το ίδιο και τα I/O pin όταν λειτουργούν ως είσοδοι
- οι έξοδοι, όταν παρέχουν ρεύμα έως 20 mA εμφανίζουν ισοδύναμη αντίσταση 75ohm, και όταν τραβάνε ρεύμα έως 20 mA εμφανίζουν ισοδύναμη αντίσταση 38ohm.

Π.3.3.Κώδικας interface card

Το ATF2500C προγραμματίστηκε και λειτουργεί όπως στο Σχήμα Π.3-6 . Ο κώδικάς παρατίθεται αυτούσιος.



Σχήμα Π.3- 5 Pin διάγραμμα ATF2500C - IFC

```
Name     DIO2SRC_BFR_2500_v3;
PartNo   00 ;
Date     11/12/2009 ;
Revision 01 ;
Designer ptsampas ;
Company   ;
Assembly DIOtoSarcosInterface ;
Location NTUA MECH csl-ep ;
Device   v2500c ;

/* C:\Wincupl\Shared>cupl  -j -u atmel.dl ../projects/NIonSARCOS/DIO2SRC_BFR_v3.pld */

/*******************************************************************************************/
/* signals are defined at pins in that way so we use only POSITIVE LOGIC into the design  */
/*******************************************************************************************/


/****************/
/*      clocks  */
/****************/

pin 1                  =      CLKin;

/*********************************************************************/
/* handshaking signals and I/O cycle signals between NI-DIO and CPLD */
/*********************************************************************/

/* output ports ( data P6 P7 , address P0 P1 ) handshaking signals */

pin [2, 3]     =      [pcWRrq, pcRDrq];      /* these signals specify the type of cycle and
comes from the two            */
                              /* unused higher bits of the address bus
        */
                              /* pcWRreq = P7.6 = A14 signals for DATA write from PC to
SARCOS  */
                              /* pcRDreq = P7.7 = A15 signals for DATA read  from SARCOS to
PC      */


pin [17..20]  =       [AUX0..AUX3];  /* auxilary signals    AUX0..4 = D13..15 = P7.4..7
        */
                                     /* are the unused 4 higher bits of the data bus     */

pin 21  =       !ALLOBF;             /* active LOW, DIO --> CPLD all output buffers full
        */
                              /* !AOBF = !OBF0 #!OBF1 # !OBF2 #!OFB3 comes from IC 74HC32 */
```

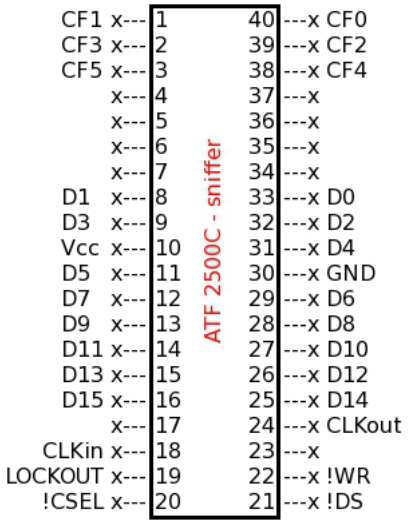
```
                            /*  AOBF =  OBF0 & OBF1 &  OBF2 & OBF3                    */

pin 25  =      !ACK;        /* active LOW, CPLD --> DIO signals 82C55 that data have been
read, SAME FOR ALL PORTS */


/* input ports ( data P3 P4 ) handshaking signals */
pin 22  = !ALLIBF;          /* active  HI, DIO --> CPLD signals CPLD that DATA have been
read by 82C55               */
                      /* AIBF = IBF0 & IBF1 comes from IC 74HC132 (NAND) if an AND gate is
used negation (!)     */
                      /* must be removed
       */
pin 24  =!STB;        /* active LOW, CPLD --> DIO signals 82C55 that DATA are available to
read, SAME FOR ALL PORTS      */

/* Bidirectional Bus Control Signals */

pin 31  =!ALE;        /* active LOW, CPLD --> BUSREPV signals BUSREPV for address latching
               */
pin 32  =!WR;         /* active LOW, CPLD --> BUSREPV signals BUSREPV for SEND cycle
               */
pin 33  =!DS;         /* active LOW, CPLD --> BUSREPV signals BUSREPV for data    latching
               */
                      /*enables data output from BUSREPV to                            */
                      /*Bidirectional Bus when  !WR is HI                              */

pin 35  =!AOE;        /* active LOW, CPLD --> 74HC541 enables address output from INTERFACE
CARD to Bidirectional Bus */
pin 36  =!DOE;        /* active LOW, CPLD --> 74HC541 enables data    output from INTERFACE
CARD to Bidirectional Bus */


/* Return Signals usefull whenever transmition latencies    */
/* needed to be taken in mind for ex long flatcables               */

pin 38  =!DS_ret;
pin 39  =!WR_ret;
pin 40  =!ALE_ret;

/*****************************************************************************************/
/* signals are defined at pins in that way so we use only POSITIVE LOGIC into the design  */
/*****************************************************************************************/

/* there are TWO DIFFERENT cycles, initiated */
/* by START and pcWRrq or pcRDrq respectivly */

/* 1. WRITE data from PC to SARCOS           */
/* 2. READ  data from SARCOS to PC           */

/* We use */
/* 1. a binary finite state machine which changes states at clock's RISING egde          */
/* 2. a lookup table to translate between states and output signals                      */
/* 3. an output latching circuitry at clock's FALLING edge to avoid contradictioning signals
       */

node START;

START   = ALLOBF & (pcWRrq$pcRDrq);                  /* ALL Output Buffers are Full       */
                                            /* and a legal action is requested   */
                                            /* this signal initiates  the cycle  */


pin [11..14] =  [state0..3];                         /* declare state as pin to gain more
product terms  */

[state0..3].ck = CLKin; min [state0..3] = 4;

sequenceD [state0..3]
{

present 'd'00

if  START&!ALLIBF next 'd'01; /* begin       */
if !START# ALLIBF next 'd'00; /* loop */
```

```
/****************/
/* SEND ADDRESS */
/****************/
present 'd'01  next 'd'02;
present 'd'02  next 'd'03;
present 'd'03  next 'd'04;
present 'd'04  next 'd'05;

present 'd'05

if pcWRrq             next 'd'06; /* continue with a SEND    DATA cycle   */
if pcRDrq             next 'd'12; /* continue with a RECEIVE DATA cycle   */


/****************/
/* SEND DATA    */
/****************/
present 'd'06  next 'd'07;
present 'd'07  next 'd'08;
present 'd'08  next 'd'09;
present 'd'09  next 'd'10;
present 'd'10  next 'd'11;
present 'd'11  next 'd'00;


/****************/
/* READ DATA    */
/****************/
present 'd'12  next 'd'13;
present 'd'13  next 'd'14;
present 'd'14   next 'd'15;
present 'd'15  next 'd'00;

}

field OutSignals =  [AOE, DOE, WR, ALE, DS, ACK, STB];

OutSignals.ck = !CLKin; min OutSignals=4;

/*ACK.oe = START;     /* enable signals from CPLD to NI only if proper signals are present
        */
/*STB.oe = START;     /* this is a measure to avoid wrong usage of NI when IFC is present*/
               /* that way 6 bits must take the rigth value 4 OBFs, pcWRrq, pcRDrq*/

table [state0..3]=>OutSignals.d
{
'd'00 => 'b'0000000;  /* steady state */

/* PC WRITE cycle consists of an ADDRESS SEND cycle and a DATA SEND    cycle      */
/* PC READ  cycle consists of an ADDRESS SEND cycle and a DATA RECEIVE cycle      */

/*     1      2      3      4      5      6      7      */
/*     AOE    DOE    WR     ALE    DS     ACK    STB    */

/* ADDRESS SEND       cycle*/
/*          1234567                                    */
'd'01 => 'b'0010000;  /*  set WR , signal BUSREPV for a SEND DATA cycle   */

'd'02 => 'b'1010000;  /*  set AOE, enable ADDRESS into bidirectiona bus   */

'd'03 => 'b'1011000;  /*  set ALE, prepeare BUSREPV to store ADDRESS      */

'd'04 => 'b'1010000;  /*reset ALE, !ALE goes HI ADDRESS stored now */

'd'05 => 'b'0010000;  /*reset AOE*/

/* DATA SEND cycle                                    */
/*          1234567                                    */
'd'06 => 'b'0110000;  /*  set DOE, enable DATA into bidirectional bus     */

'd'07 => 'b'0110101;  /*  set DS , prepeare AJC to store DATA             */
               /*  set STB, force NI-DIO input ports to READ DATA  */
               /*           useful for validation                  */
               /** posible wait stage for the FSM ***************  */
```

```
'd'08 => 'b'0110000;  /*reset DS, !DS goes HI DATA stored now      */
               /*reset STB                   */

'd'09 => 'b'0010000;  /*reset DOE                   */

'd'10 => 'b'0000000;  /*reset WR                    */

'd'11 => 'b'0000010;  /*  set ACK, signal NI-DIO output ports that DATA   */
               /*  have been read                          */
               /** posible wait stage for the FSM ******************/


/* DATA READ cycle                                          */
/*          1234567                                         */
'd'12 => 'b'0000000;  /*reset WR , signal BUSREPV for a DATA READ cycle   */
'd'13 => 'b'0000100;  /*  set DS , enable DATA from AJC into bidirectional */
               /*           bus                            */
'd'14 => 'b'0000101;  /*  set STB, force NI-DIO input ports to READ DATA  */
               /** posible wait stage for the FSM ******************/

'd'15 => 'b'0000010;  /*reset DS                                 */
               /*reset STB                           */
               /*  set ACK, this signal is neccecary for NI-DIO    */
               /*           output ports                           */
               /** posible wait stage for the FSM ******************/

}
```

Π.3.4.Κώδικας sniffer card

Το ATF2500C προγραμματίστηκε και λειτουργεί όπως στο Σχήμα Π.3-7. Ο κώδικάς παρατίθεται αυτούσιος.



Σχήμα Π.3. 6 Pin διάγραμμα ATF2500C - sniffer

```
Name     AJC_sniffer_V1 ;
PartNo   00 ;
Date     1/8/2010 ;
Revision 01 ;
Designer ptsampas ;
Company  MSc NTUA;
Assembly SARCOS ;
Location csl_ep NTUA ;
Device   v2500c ;

/* this code impelments              */
/* 1. a digital 16bit R/W register           */
/* 2. a 24bit counter                */

/* operation                 */

/* register                          */
/* write at  'b'000000               */
/* read from 'b'000000        */

/* counter                           */
/* read from 'b'00001x stops the counter     */
/* read from 'b'000010 reads C00..15 */
/* read from 'b'000011 reads C16..23 */
/* write at  'b'00001x resets the counter    */


/* data bus */
pin [ 8,  9, 11, 12, 13, 14, 15, 16] = [D1 , D3, D5, D7, D9, D11, D13, D15];

pin [33, 32, 31, 29, 28, 27, 26, 25] = [D0 , D2, D4, D6, D8, D10, D12, D14];


/* address bus */
pin [1, 2, 3, 40, 39, 38] = [CF1 ,CF3, CF5, CF0, CF2, CF4];


pin 19 = LOCKOUT;
/************************************/
/* into this design use positive     */
/* logic for control bus signals     */
```

```
/*************************************/
pin 21 = !DS;         /* DataStrobe  */

pin 22 = !WR;         /* Write              */

pin 20 = !CSEL;               /* CardSelected       */


node [Q0..15];                /* internal register  */
node [C0..30];                /* internal counter   */

/* clock */
pin 18 = CLKin;
pin 24 = CLKout;

node CLKhold;

/* clock */
CLKhold.d  = 'b'1;
CLKhold.ck =  CSEL & DS & !WR & CF1 ;        /* read  at 'b'xxxx1x stops  the counter     */
CLKhold.ar =  CSEL & DS &  WR & CF1 ;        /* write at 'b'xxxx1x starts the counter     */

CLKout =  CLKhold & CLKout.io # !CLKhold&!CLKin;     /* if CLKhold = 1 hold    counter    */
                                       /* if CLKhold = 0 release counter    */
/* internal register */
[Q0..15].d  = [D0..15].io;
[Q0..15].ck = CSEL & DS & WR & ([CF0..CF1]:'b'00);  /* write to internal register at
'b'xxxx00      */


/* internal counter */

[C0..30].ar = CSEL & DS & WR & CF1 ;         /* write at 'b'xxxx1x resets the counter     */

[C0..30].ck = CLKout;

C0.t = 'b'1;
$repeat i=[1..30]
C{i}.t = [C0..{i-1}]:& ;
$repend

C31 ='b'0;

/* data outpout */
[D0..15].oe = CSEL & DS & !WR ;                              /* read , enable output
        */

                                             /* register    => 'b'xxxx00          */
                                             /* counter low => 'b'xxxx10   */
                                             /* counter hi  => 'b'xxxx11   */

[D0..15]    = [Q0..15]&([CF0..1]:'b'00) # [C0..15]&([CF0..1]:'b'10) #
[C16..31]&([CF0..1]:'b'11) ;
```

Π.4.Ο κώδικας MATLAB

Περιγράφεται ο κώδικας MATLAB που χρησιμοποιείται στην εξόρυξη δεδομένων από το RAW αρχείο εξόδου του προγράμματος ελέγχου του ρομποτικού βραχίονα.

Ο κώδικας διαβάζει το RAW αρχείο και κατασκευάζει τους εξής πίνακες όπως περιγράφονται στον Πίνακα Π.4-1

Π.4- 1 Πίνακες που κατασκευάζει η ρουτίνα MATLAB

πίνακας	περιεχόμενα	μονάδες
dt	χρόνος βρόχου	counts
t	απόλυτος χρόνος	sec
CHLval	τιμες από κανάλια	ψηφιακές
DACval	τιμές σε/από DAC	ψηφιακές

```
%find file
[n,p]=uigetfile([    '*raw.*']);cd(p);

fid=fopen([p n],'r');

datao=uint32(fscanf(fid,'%u%u',[2 inf]))';
datai=uint16(datao(:,2));datao(:,2)=[];

%check for READ COUNTER LOW READ
indx =  find (datao == bin2dec('1000001000001010') );

L = length(indx);

%get time
dt = uint32(datai(indx) + datai(indx+1)*65536);
dt(1)=0;
 t=[ 0; cumsum(double(dt)) ] /8.86e6;

CHLval = repmat(uint16(4096),[L 16 11]);
DACval = repmat(uint16(4096),[L 64 11]);

%extract channels
 DACmask = uint32(bin2dec('00000000000000000000000011111111'));%8  bits
 DOFmask = uint32(bin2dec('00000000000000000011111100000000'));%6  bits
  WRmask = uint32(bin2dec('00000000000000001100000000000000'));%2  bits
DATAmask = uint32(bin2dec('11111111111111110000000000000000'));%16 bits
iter = 1; i=1;
for i=i:min( indx(end)+diff(indx)-1 , length(datao))

    dof = uint8( bitand( datao(i) , DACmask) );%get dof

    dac = uint8( bitshift( bitand( datao(i) , DOFmask ) , -8) );%get dac (address)

     wr = uint8( bitshift( bitand (datao(i) ,  WRmask ), -14) ) ;%get  write/read command


    dataout=uint16( bitshift( bitand( datao(i), DATAmask) , -16 ) ) ;%get doata out


    %new iteration
    if ( wr == 2 & dof == 10 & dac == 2)
        if iter>1 % fill in unread DACs and CHANNELS
            for DOF=0:10

                for DAC=0:63
                    if DACval(iter,DAC+1,DOF+1)==4096 %not updated use previous value
                        DACval(iter,DAC+1,DOF+1)=DACval(iter-1,DAC+1,DOF+1);
                    end
                end

                for CHANNEL=0:15
                    if CHLval(iter,CHANNEL+1,DOF+1)==4096 %not updated use previous value
```

```
                        CHLval(iter,CHANNEL+1,DOF+1)=CHLval(iter-1,CHANNEL+1,DOF+1);
                    end
                end

            end
        end
        iter=iter+1;%next iteration
    end

    %channel selection , set
    if ( wr == 1 & dac == 48)
        channel = dataout;
    end

    %READ channel
    if ( wr == 2 & dac == 2 & dof~=10)
        CHLval(iter,channel+1,dof+1) = bitand( datai(i), bin2dec('0000111111111111'));
    end

    %WRITE to dac
    if ( wr == 1 )
        DACval(iter,dac+1,dof+1) = dataout;
    end

    %READ from dac (or encoder)
    if ( wr == 2 )
        DACval(iter,dac+1,dof+1) = datai(i);
    end

end

save([ n 'mat'])
```

Π.5.Ο κώδικας ελέγχου του ρομποτικού βραχίονα

```
/*********************************************************************
*

This code is for establising BIDIRECTIONAL communication
and control between PC and SARCOS using  NI-DIO card and
the custom made INTERFACE card.

To do so the following requirments MUST MEET

*** WARNING ****** WARNING ****** WARNING ****** WARNING ****** WARNING
***
***       WRONG CONFIGURATION WILL DAMAGE NI-DIO AND INTERFACE CARD
***
*********************************************************************
*

0. SARCOS uses a 14bit addressing mode explaind  hereby
      0.a Every CAGE has 21 cards 1 BUSREPV and (as many as) 20 AJCs
      0.b That way every cage owns a unique address range [0...20] ,
[21...41] , [42...62] and so on until the all range of 8bit addresses (256
in total) is covered.
      0.c Inside BUSREPV and AJCs are digital addresses that can be
written or read. This addresses form a 14 bit address space. The 2 most
significant bits are not used by SARCOS so are suitable to indicate
pcWRITE or pcREAD cycle (see 1.b)

1. Ports 0,1,6,7 must be set as STROBED OUTPUTS (totaly 32 bits = 16 + 16)
      1.a Ports 0 [0.0 ... 0.7], 1 [1.0 ... 1.7] outputs 16bit address
ACCORDING to SARCOS addressing convetions creating a 14bit address space
      1.b The two MOST SIGNIFICANT address bits [1.6 1.7] are not used as
address bit by SARCOS so the are suitable for signaling PC DATA WRITE TO
SARCOS [0 1] (hex mNoQueryCode 0x4000) or PC DATA READ FROM SARCOS [1 0]
(hex mNoQueryCode 0x8000) cycles. For safety reasons combinations [0 0] or
[1 1] are ignored by the hardware.
      1.c Ports 6 , 7  outputs 12+4 bit data.
      1.d The FOUR MOST SIGNIFICANT bits [7.4 ... 7.7] are ignored by AJCs
while they input DATA (PcWrite cycle). They may be used by the SNIFFER
CARD or as auxilary bits by the IFC card for further expansion

2. Ports 3,4 must be set as STROBED INPUTS (totaly 16 bits)
      2.a Ports 3 [3.0 ... 3.7], 4 [4.0 ... 4.7] input 16bit data from
SARCOS. ADCs use only 12bit but ENCODERs use 16bit

3. This protocol is selected for the following reasons
      3.a the INTERFACE CARD is responsible of managing control signals
for both SARCOS and DIOs 82C55. That way the I/O speed of the NI-DIO card
(max 80kHz) doesn't affect duration and timming of control signals, so
more bandwidth is dedicated to true data transfers
      3.b a single DAQmxwrite call can output 32bit (addresse + data) at
one cycle so we have significant gain in speed (x2)

4. For proper cooperation of NI-DIO and INTERFACE cards, a STROBED WRITE
cycle must allways followed by a STROBED READ cycle, according to the
following rules
      4.a To READ DATA from SARCOS to PC
            - Write to NI-DIO with DAQmxwrite 32bit integer = [ NotCare16
[1 0] Address14 ]
            - Read from NI-DIO with DAQmxread a 16bit integer.
```

```
            - The returned value is the value sent by SARCOS to PC

      4.b To WRITE DATA from PC to SARCOS
            - Write to NI-DIO with DAQmxwrite 32bit integer = [ NotCare4
Data12 [0 1] Address14   ]
            - Read from NI-DIO with DAQmxread a 16bit integer.
            - The returned value is the data sent by PC to SARCOS so this
is a way to test the integrity of the BIDIRECTIONAL BUS (external bus)
      4.c That way a block DAQmxwrite and a block DAQmxread functions are
combined to get block bidirectional I/O in speeds up to 120kHz

5.Ports (0,1,2) (6,7,8) (3,4,5) are reserved the third port in each
triplet used for communication signals. Pors (9,10,11) are free for
further expansion
*************************************************************************
**************************/
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <NIDAQmx.h>


#define DAQmxErrChk(functionCall) if( DAQmxFailed(error=(functionCall)) )
goto Error; else

#define recordlength    0x00F00000

// delay for current variation when calibrating
#define calDelay  0x8000

/* SarcosRead            D16 10xx xxxx xxxx xxxx*/
#define ReadMask  0x00008000

/* SarcosWrite           D16 01xx xxxx xxxx xxxx*/
#define WriteMask 0x00004000

#define dofs            10
#define sniffer         10
#define busrepv         0
// define addresses inside SNIFFER

// read-write digital register
#define     DiReg       0x0000

// 31bit long counter
// read from CntHi or CntLo will stop  the counter
// write to  CntHi or CntLo will reset the counter
#define CntLo   0x0200
#define     CntHi 0x0300


// define addresses inside AJC

// read only
#define     LocalADC    0x0200
#define     EncoderHi   0x2100
#define EncoderLo       0x2000
```

```
// write only
#define LocalMUXsel     0x0300

#define GlobalADC       0x3100
#define GlobalMUXsel    0x3000

#define ValveGain 0x0C00
#define ValveBias 0x0800

#define ActPosGain1     0x0A00
#define ActPosBias1     0x1700

#define ActPosGain2     0x1800
#define ActPosBias2     0x1F00

#define TorqueGain      0x1E00
#define TorqueBias1     0x1900
#define TorqueBias2     0x1D00

#define CardID          0x0000
#define BoardStatusReg  0x0100

#define GravComp  0x1A00
#define DesPosIn  0x1B00
#define DesTrqIn  0x1C00

// IO DATA USED BY DAQmx
struct iosignals_
{
unsigned short    indx;

uInt32      o[1024];
uInt16      i[1024];

}  iosignals;

// LONG TERM RECORD
struct iorec_
{
unsigned long     indx;

uInt32      o[recordlength];//output data
uInt16      i[recordlength];//input  data
char  a[recordlength];//calling routine identifier
} iorec;


//temporary record for all channels, mean values, velocities, etc.
//use channel's ID as matrix index to store temporal values into it
//use one extra for encoder and one extra for clock(?)
#define     PKnod 0
#define     Apos1 1
#define Apos2     2
#define Atrq      3
#define Otrq      4
#define Avel      5

#define Vvaf      6
#define Vcur      7
#define     Vvbf  8
```

```
#define Bpos1     14
#define Bpos2     15

#define Enchann   16

// the length of the record
#define reclen    8

// DOF STATE VARIABLES
struct dofstv_
{

int indx;               //Circular index. Points to the NEXT element to be
updated
                        //Each time an IO circle takes place the index is
increased by
                        //one, so matrices are filled up sequentially and
circularly.

unsigned short    channels[dofs+1][17][reclen]; //raw channels record
[dof][channel]['time']

signed short fpos[dofs+1];          //filtered position
signed short fvel[dofs+1];          //filtered velocity
signed short facc[dofs+1];          //filtered acceleration

signed short ftrq[dofs+1];          //filtered torque
signed short dtrq[dofs+1];          //filtered torque first direvative

signed short perr[dofs+1];                //position error

signed short mx[dofs+1];            //maximum values
signed short mn[dofs+1];            //minimum values

signed long  integ[dofs+1];         // integral term

unsigned long     cnt[reclen];      //counts
unsigned short    chk[reclen];      //check, digital register

char unsigned status;
} dofstv;

struct dessg_
{
char uinp[dofs+1];
char  inp[dofs+1];            // holds the the input method by which each
dof will be governed in Digital Mode

                        // 'k', 'K' =     position by Panel knod
                        // 'o', 'O' =     position by strored positions
                        // 'm', 'M' =     position by master
                        // 'f', 'F' =     position by file

                        // 'c', 'C' =     position sensors calibration
routine
                        // 'i', 'I' =     open loop identification

                        // 'n', 'N' =     current by Panel Knod
```

```
char umove;
char  move;             // movement type
                        // WHEN INPUT FROM FILE
                        // 'f'            after a given dt time interval
                        // 'p'            after a given dt time interval
and
                        //          after last desired position is in
proximity

                        // WHEN INPUT FROM STORED
                        // 'f'            free
                        // 's'            sequential


char mode[dofs+1];            // holds the mode analog or digital of
control
                        // 'a' = analog control  via AJCs      with
digital input of desired position and torque
                        // 'd' = digital control via ALGORITHM with
digital input of desired valve current

char grvf;
char posf;
char trqf;
char curf;

signed short      grv[dofs+1];      //desired gravity compansation
signed short      pos[dofs+1];      //desired position
signed short      trq[dofs+1];      //desired torque
signed short      cur[dofs+1];      //desired current

} dessg;

struct dist_
{
signed int  ad[dofs+1]; //actual to desired

char        ok[dofs+1]; //boolean used for proximity
char        okdof;            //last dof with proximity
char        okall;            //boolean used for total proximity

} dist;

// into this structure 16 positions can be strored
// for quick reference and positioning

// index 0 is reserved for zero torque position.      The position of
torque free ARM   (don't use elsewere)
// inedx 1 is reserved for half range position.       The position of 1/2
dof range   (don't use elsewere)

struct stored_
{
signed short      dist[dofs+1];     //distance from selected position to
preset position

unsigned short    pos[dofs+1][16];  //stored positions for all DOFs
unsigned short    cur[dofs+1][16];  //stored currents  for all DOFs
unsigned short    pot[dofs+1][16];  //stored potKnods  for all DOfs
```

```
char working;                 //working position index
char zeroT;             //zero Torque index
char zeroP;             //zero position index

} stored;

struct AB_
{
unsigned int a[dofs+1][2];    //gain and biases for calibration procudure
unsigned int b[dofs+1][2];
} AB;


unsigned int      invAB[dofs+1] = {0,410, 425, 416, 414, 411, 412, 413,
411, 410, 410};// position sensor -- > stage 1

char         reps[dofs+1]={0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, };
signed int   di  [dofs+1]={0,1, 1, 1, 1, 1, 1, 1, 1, 1, 1, };


// timmming variables
struct counts_
{
long unsigned int in;   //accumulative counts from clock

long unsigned int exp;  //counts between movements

long unsigned int req;
} counts;

double timeEllapsed, timePressureIgnore, timeFinal, timeDelta; time_t
timeStart, timeLastDisplay;

char expirimentID, expirimentStatus;

//Digital PID control gains
short int   Kp[dofs+1] = {0,3,-3, 3,-3, 3, 2, 2, 1, 1, 1}, //dof 2 and 4
need the minus sign!!!
            Kd[dofs+1] = {0,3,-3, 3,-3, 3, 2, 2, 1, 1, 1},

            Ki[dofs+1] = {0,3,-3, 3,-3, 3, 2, 2, 1, 1, 1};

unsigned    error;
char        errBuff[2048]={'\0'}, AbordKey;


FILE        *cfPtr1, *cfPtr2, *pathfile, *UserOrders;

unsigned long     TaskRead,TaskWrite;



//////////////////////////SUBROUTINES//////////////////////////////////////
///
//////////////////////////////////////////////////////////////////////////
///
// uses no filter
void fltnone()
{
```

```
unsigned char dof, j, aindx;

aindx = (dofstv.indx+reclen-1)%reclen; // allign index

// get position from first stage
for ( dof=1;dof<sniffer ;dof++) //for all dofs
{
      dofstv.facc[dof] = 0;
      dofstv.fvel[dof] = 0;
      dofstv.fpos[dof] = dofstv.channels[dof][Apos1][aindx];


}
}

// gets raw position and torque information
// and filters it using linear interpolation
void fltlinear()
{


}

// gets raw position and torque information
// and filters it using quadratiq interpolation
void fltquad()
{

// define matrices used for interpolation
//#if reclen==8
signed long int Q[reclen][3]=
{
{        2352,      -21167,       39984, },
{         336,       -5711,       21168, },
{       -1007,        5040,        7057, },
{       -1679,       11088,       -2351, },
{       -1679,       12432,       -7055, },
{       -1008,        9072,       -7055, },
{         336,        1009,       -2352, },
{        2352,      -11759,        7056, },
},

det= +56448;
//#endif

/*#if reclen==16
signed long int Q[reclen][3]=
{
{        +1374688,                -354144,           +19040, },
{        +1039584,                -230765,           +11424, },
{         +742560,                -123705,            +4896, },
{         +483616,                 -32966,             -544, },
{         +262752,                 +41453,            -4896, },
{          +79968,                 +99552,            -8160, },
{          -64736,                +141331,           -10336, },
{         -171360,                +166790,           -11424, },
{         -239904,                +175930,           -11424, },
{         -270368,                +168749,           -10336, },
{         -262752,                +145248,            -8160, },
{         -217056,                +105427,            -4896, },
```

```
{         -133280,                +49286,             -544, },
{          -11424,                -23174,            +4896, },
{         +148512,               -111955,           +11424, },
{         +346528,               -217056,           +19040, },
},
det = +3107328;
#endif

#if reclen==32
#endif*/

long int S[3]; unsigned char dof, j, aindx;

// get position from first stage
for ( dof=1;dof<sniffer ;dof++) //for all dofs
      {
      S[0]=0;S[1]=0;S[2]=0;

      for ( j=0;  j<reclen;  j++)
      {
      aindx = (dofstv.indx  + j )%reclen;       // allign index: when j==0
        aindx = dofstv.indx   (initial position)
                                          //           when j==reclen-1
aindx = dofstv.indx-1   (final   position)
      S[2] += ((signed long)dofstv.channels[dof][Apos1][aindx] * Q[j][0]);
//interpollation coefficient for x^0
      S[1] += ((signed long)dofstv.channels[dof][Apos1][aindx] * Q[j][1]);
//interpollation coefficient for x^1
      S[0] += ((signed long)dofstv.channels[dof][Apos1][aindx] * Q[j][2]);
//interpollation coefficient for x^2


      }

      S[0]=S[0]/det; S[1]=S[1]/det; S[2]=S[2]/det;

      dofstv.facc[dof] = (2*S[2]
      );//acceleration

//    dofstv.fvel[dof] = (  2*S[2]*reclen       +S[1]
      );//velocity prediction at next moment
      dofstv.fvel[dof] = (  S[2]*reclen         +S[1]
      );//velocity at the meedle of movement

//    dofstv.fpos[dof] = (  S[2]*reclen*reclen  +S[1]*reclen      +S[0]
      );//position prediction at next moment
      dofstv.fpos[dof] = (  dofstv.fvel[dof]*reclen   +S[0]
      );//position prediction at next moment with less calculations

      }

}

// this routine stores the current positions and currents to the specified
// location into the matrix storedpos maximum 16 positions
// ****** WARNING may cause damage to the robot or to personnel
*************************
//
// stored.pos[:][0] MUST BE the zero torque position. This is ASSUMED by
the program
```

```
// Its value must be modifided causiously with the robotic arm
// UNPRESSURIZED and MANUALLY PLACED into the ZERO POSITION ( don't use
elsewere )
//
//
**************************************************************************
*************
void storesgs(char n, char r)
{

char dof, aindx;
aindx = (dofstv.indx+reclen-1)%reclen; //allign index

//printf("\nposition %d\n",n);

for (dof=1; dof<sniffer; dof++)
      {

      stored.pos[dof][n] = dofstv.channels[dof][Apos1][aindx];    // store
position
      stored.cur[dof][n] = dessg.cur[dof];                        // store
current
      stored.pot[dof][n] = dofstv.channels[dof][PKnod][aindx];    // store
PanelKnod

      if (r=='r') printf("dof %d pos %d cur %d pot %d
\n",dof,stored.pos[dof][n],stored.cur[dof][n],stored.pot[dof][n]);

      }


}

// sends and receives data
void sendget(char action)
{
// This function sends orders to dofs and receives data back.
// Then parces orders to find read commands and stores returned
// values into a temporary record (dofstv.channel) for further usage
// Each channel on each dof has its own record which is updated
// whenever a read command for this particular combination of
// channel and dof is encountered

short unsigned int written, read, werr,rerr;

werr = DAQmxWriteDigitalU32 (TaskWrite, iosignals.indx, 1, 0.10,
DAQmx_Val_GroupByChannel, &iosignals.o[0],       &written, NULL);

rerr = DAQmxReadDigitalU16  (TaskRead,  iosignals.indx,    0.10,
DAQmx_Val_GroupByChannel, &iosignals.i[0], 1024, &read,    NULL);

if (werr!=0 | rerr!=0) printf("*** WARNING DAQmx WRITE %u READ %u
error\n\a *** ",werr,rerr);

unsigned int j,data,rdad=0; unsigned char dof=255, channel=255;

//write iosignals to record
if ( (iorec.indx + iosignals.indx < recordlength) & action=='r')
      for (j=0;j<iosignals.indx;j++)
      {
```

```
      iorec.o[iorec.indx] = iosignals.o[j];
      iorec.i[iorec.indx] = iosignals.i[j];

      iorec.indx++;
      }

// scan iosignals.o to find read commands
for (j=0;j<iosignals.indx;j++)
{

      //*****************************************************************
********************************
      // warning this code works either with a single globalADC read or
two subsequent localADC reads
      // it can not handle 'shifted' single reads inside the same dof and
will write to record wrong data
      //*****************************************************************
********************************

      // find channel selection order, and extract channel number
      // channel selection is done by a WRITE to localMUX or globalMUX so
search for that first

      if  ( (iosignals.o[j]&0x00003F00)==GlobalMUXsel )
            channel  = (unsigned char)((iosignals.o[j]&0x000F0000)>>16);

      // all subsequent reads will be directed to that channel's record
      //unless a read from encoder or counter encountered
      if ( (iosignals.o[j]&ReadMask)==ReadMask )      // check for read
command
            {
            dof =  iosignals.o[j]&0x000000FF;   //extract dof
            rdad=  iosignals.o[j]&0x00003F00;   //extract read address

            switch (dof)
                  {

                  case 1:case 2:case 3:case 4:case 5:
            //dofs
                  case 6:case 7:case 8:case 9:

                  if (rdad==LocalADC)

      dofstv.channels[dof][channel][dofstv.indx]=iosignals.i[j]&0x0FFF;
      //12bit value from ADC

                  if (rdad==EncoderHi)

      dofstv.channels[dof][Enchann][dofstv.indx]=iosignals.i[j];
      //16bit value from encoder
                  break;



                  case sniffer:
                  if (rdad==DiReg) dofstv.chk[dofstv.indx] =
iosignals.i[j];               // register

                  if (rdad==CntLo)
```

```
                  {dofstv.cnt[dofstv.indx] = iosignals.i[j];    } //
counter low
                                                                        //
counter low  can be read alone
                                                                        //
and alwayw before counter high
                                                                        //
because counter high is added to that value
                  if (rdad==CntHi)
                  {dofstv.cnt[dofstv.indx]+= iosignals.i[j]<<16;} //
counter high
                                                                        //
counter high must be read after
                                                                        //
a counter low read.
                  break;

                  case busrepv:
                  dofstv.status = iosignals.i[j];
            //status
                  break;

                  default:break;
                  }


                  }

            }
            counts.in += dofstv.cnt[dofstv.indx];     //this serves as time
            dofstv.indx++;dofstv.indx %=reclen; //increase index if index
goes higher than reclen sent it back to zero
}


//data reading routines

//NoQueryCodes the time interval (in counts of 1/4.4e6 sec) of the overall
loop
void queryTime()
{
iosignals.o[iosignals.indx] =   0x00000000   |ReadMask  | CntLo    |
sniffer ;iosignals.indx++;//read counter low  from SNIFFER CARD
iosignals.o[iosignals.indx] =   0x00000000   |ReadMask  | CntHi    |
sniffer ;iosignals.indx++;//read counter high from SNIFFER CARD

iosignals.o[iosignals.indx] =   0x00000000   |WriteMask | CntLo    |
sniffer ;iosignals.indx++;//reset counter

}

void querySensors()
{
// the following code is proper for reading various channels using global
ADC
// convertion take care of the 16bit shift operation applied on channels.
// Î•NCODERS NEED SPECIAL TREATMENT since addressing is different

// when reading channels
// the MUX GLOBAL SELECT  command,
```

```
// the ADC GLOBAL CONVERT command and
// the ADC LOCAL READ     command,
// take as argument the channel's value .
// This is NOT necesary for ADC GLOBAL CONVERT and ADC READ but is useful.
JUST DO IT

unsigned int channel; char dof, noADCcode;

iosignals.o[iosignals.indx] =   0         |ReadMask  |         0x0100  | 0
;iosignals.indx++;//check if arm is under pressure


/* various read commands

//read panel Knod
channel = PKnod<<16; noADCcode = TRUE;

iosignals.o[iosignals.indx] =   channel   |WriteMask | GlobalMUXsel    | 1
;iosignals.indx++;//prepare MUXs to read Position from first stage
iosignals.o[iosignals.indx] =   channel   |WriteMask | GlobalADC       | 1
;iosignals.indx++;//force all ADCs to convert

for (dof=1; dof<sniffer; dof++)
      {iosignals.o[iosignals.indx] =  channel     |ReadMask  | LocalADC
| dof;iosignals.indx++;} //read from all dofs



//read Current
channel = Vcur<<16; noADCcode = TRUE;

iosignals.o[iosignals.indx] =   channel   |WriteMask | GlobalMUXsel    | 1
;iosignals.indx++;//prepare MUXs to read Position from first stage
iosignals.o[iosignals.indx] =   channel   |WriteMask | GlobalADC       | 1
;iosignals.indx++;//force all ADCs to convert

for (dof=1; dof<sniffer; dof++)
      {iosignals.o[iosignals.indx] =  channel     |ReadMask  | LocalADC
| dof;iosignals.indx++;} //read from all dofs


//read Voltage after fuse
channel = Vvaf<<16; noADCcode = TRUE;

iosignals.o[iosignals.indx] =   channel   |WriteMask | GlobalMUXsel    | 1
;iosignals.indx++;//prepare MUXs to read Position from first stage
iosignals.o[iosignals.indx] =   channel   |WriteMask | GlobalADC       | 1
;iosignals.indx++;//force all ADCs to convert

for (dof=1; dof<sniffer; dof++)
      {iosignals.o[iosignals.indx] =  channel     |ReadMask  | LocalADC
| dof;iosignals.indx++;} //read from all dofs


//read Voltage before fuse
channel = Vvbf<<16; noADCcode = TRUE;

iosignals.o[iosignals.indx] =   channel   |WriteMask | GlobalMUXsel    | 1
;iosignals.indx++;//prepare MUXs to read Position from first stage
```

```
iosignals.o[iosignals.indx] =   channel   |WriteMask | GlobalADC       | 1
;iosignals.indx++;//force all ADCs to convert

for (dof=1; dof<sniffer; dof++)
      {iosignals.o[iosignals.indx] =  channel     |ReadMask  | LocalADC
| dof;iosignals.indx++;} //read from all dofs


//channel = Apos2<<16;

iosignals.o[iosignals.indx] =   channel   |WriteMask | GlobalMUXsel    | 1
;iosignals.indx++;//prepare MUXs to read Position from first stage
iosignals.o[iosignals.indx] =   channel   |WriteMask | GlobalADC       | 1
;iosignals.indx++;//force all ADCs to convert

for (dof=1; dof<sniffer; dof++)
      {iosignals.o[iosignals.indx] =  channel     |ReadMask  | LocalADC
| dof;iosignals.indx++;} //read from all dofs


iosignals.o[iosignals.indx] =   channel   |WriteMask | GlobalMUXsel    | 1
;iosignals.indx++;//prepare MUXs to read Position from first stage
iosignals.o[iosignals.indx] =   channel   |WriteMask | GlobalADC       | 1
;iosignals.indx++;//force all ADCs to convert

for (dof=1; dof<sniffer; dof++)
      {iosignals.o[iosignals.indx] =  channel     |ReadMask  | LocalADC
| dof;iosignals.indx++;} //read from all dofs


*/

// read on demand (more sophisticate code)

//read Panel Knob
channel = PKnod<<16; noADCcode = TRUE;

for (dof=1; dof<sniffer; dof++)
      {
      //if (dessg.inp[dof]=='k' | dessg.inp[dof]=='K'  )
            {
            if (noADCcode)
                  { // prepare ADCs + convert only once
                  iosignals.o[iosignals.indx] = channel     |WriteMask  |
GlobalMUXsel | 1 ;iosignals.indx++;//prepare MUXs to read Panel Knod
                  iosignals.o[iosignals.indx] =             WriteMask   |
GlobalADC    | 1 ;iosignals.indx++;//force all ADCs to convert
                  noADCcode = FALSE;
                  }

            iosignals.o[iosignals.indx] = ReadMask  | LocalADC        |
dof;iosignals.indx++; //read from dof
            }
      }

// read Valve Current
channel = Vcur<<16; noADCcode = TRUE;

for (dof=1; dof<sniffer; dof++)
```

```
      {
      if (dessg.inp[dof]=='c' | dessg.inp[dof]=='C'  )
            {
            if (noADCcode)
                  { //prepare + convert only once
                  iosignals.o[iosignals.indx] = channel     |WriteMask |
GlobalMUXsel | 1 ;iosignals.indx++;//prepare MUXs to read PanelKnod
                  iosignals.o[iosignals.indx] = WriteMask | GlobalADC    |
1 ;iosignals.indx++;//force all ADCs to convert
                  noADCcode = FALSE;
                  }

            iosignals.o[iosignals.indx] = ReadMask  | LocalADC        |
dof;iosignals.indx++; //read from dof
            }
      }

// read Valve Voltage Before Fuse
channel = Vvbf<<16; noADCcode = TRUE;

for (dof=1; dof<sniffer; dof++)

      {
      if (dessg.inp[dof]=='c' | dessg.inp[dof]=='C'  )
            {
            if (noADCcode)
                  { //prepare + convert only once
                  iosignals.o[iosignals.indx] = channel     |WriteMask |
GlobalMUXsel | 1 ;iosignals.indx++;//prepare MUXs to read PanelKnod
                  iosignals.o[iosignals.indx] = WriteMask | GlobalADC    |
1 ;iosignals.indx++;//force all ADCs to convert
                  noADCcode = FALSE;
                  }

            iosignals.o[iosignals.indx] =  ReadMask  | LocalADC        |
dof;iosignals.indx++; //read from dof
            }
      }

// read Valve Voltage Before Fuse
channel = Vvaf<<16; noADCcode = TRUE;

for (dof=1; dof<sniffer; dof++)

      {
      if (dessg.inp[dof]=='c' | dessg.inp[dof]=='C'  )
            {
            if (noADCcode)
                  { //prepare + convert only once
                  iosignals.o[iosignals.indx] = channel     |WriteMask |
GlobalMUXsel | 1 ;iosignals.indx++;//prepare MUXs to read PanelKnod
                  iosignals.o[iosignals.indx] = WriteMask | GlobalADC    |
1 ;iosignals.indx++;//force all ADCs to convert
                  noADCcode = FALSE;
                  }

            iosignals.o[iosignals.indx] = ReadMask  | LocalADC        |
dof;iosignals.indx++; //read from dof
            }
      }
```

```
// always read position from first stage

channel = Apos1<<16;

iosignals.o[iosignals.indx] = channel     |WriteMask | GlobalMUXsel    | 1
;iosignals.indx++;//prepare MUXs to read Position from first stage
iosignals.o[iosignals.indx] = WriteMask | GlobalADC       | 1
;iosignals.indx++;//force all ADCs to convert

for (dof=1; dof<sniffer; dof++)
      {iosignals.o[iosignals.indx] =  ReadMask  | LocalADC        |
dof;iosignals.indx++;} //read from all dofs


//always read torque

channel = Atrq<<16;

iosignals.o[iosignals.indx] = channel   |WriteMask | GlobalMUXsel    | 1
;iosignals.indx++;//prepare MUXs to read Torque
iosignals.o[iosignals.indx] = WriteMask | GlobalADC       | 1
;iosignals.indx++;//force all ADCs to convert

for (dof=1; dof<sniffer; dof++)
      {iosignals.o[iosignals.indx] = ReadMask  | LocalADC        |
dof;iosignals.indx++;} //read from all dofs

}

// control routines
// PID
void controlDigPID()
{

char dof, dofod; long unsigned int channel;


for (dof=1; dof<sniffer; dof++)
      {


      // calculate current
      switch( dessg.inp[dof] )
            {
            // position by panel knod , by stored position, or by file
            case 'k':case 'K':case 'o':case 'O':case 'f':case 'F':case
'I':

            dofstv.perr[dof] = dessg.pos[dof] - dofstv.fpos[dof];

            //accumylate error only if pressure in on
            if ((dofstv.status & 0x02)!=0)
            {
            dofstv.integ[dof] = dofstv.integ[dof] + dofstv.perr[dof];
            //saturate integral
            if (dofstv.integ[dof]> 4096) dofstv.integ[dof]= 4096;
            if (dofstv.integ[dof]<-4096) dofstv.integ[dof]=-4096;
            }
            else  dofstv.integ[dof]=0;
```

```
            dessg.cur[dof] = Kp[dof]*dofstv.perr[dof] -
Kd[dof]*dofstv.fvel[dof] + (Ki[dof]*dofstv.integ[dof])/4096 + 2048;


            break;

            case 'a': case'A':
            dessg.cur[dof] = 2048;
            break;

            default: break;

            }//switch

            // current limiter
            if (dessg.cur[dof]<=0)    dessg.cur[dof] = 0;
            if (dessg.cur[dof]>=4095) dessg.cur[dof] = 4095;

            //send current
            iosignals.o[iosignals.indx] =   (dessg.cur[dof]<<16)
|WriteMask | ValveBias    | dof ;iosignals.indx++;// send current to valve
dac
      }

}

// This routine implements PD ANALOG control via analog circuitry of AJCs
void controlAnaPID()
{

char dof; unsigned int channel;


for (dof=1; dof<sniffer; dof++)
      {
      iosignals.o[iosignals.indx] =   (dessg.grv[dof]<<16)   |WriteMask |
GravComp    | dof ;iosignals.indx++;// send desired Gravity Compensation
to AJCs

      }

for (dof=1; dof<sniffer; dof++)
      {
      iosignals.o[iosignals.indx] =   (dessg.pos[dof]<<16)   |WriteMask |
DesPosIn    | dof ;iosignals.indx++;// send desired POSITION to AJCs

      }

for (dof=1; dof<sniffer; dof++)
      {
      iosignals.o[iosignals.indx] =   (dessg.trq[dof]<<16)   |WriteMask |
DesTrqIn    | dof ;iosignals.indx++;// send desired TORQUE to AJcs

      }

}

int zeroCurrent()
```

```
{
char dof;

iosignals.indx = 0;

//iosignals.o[iosignals.indx] =  0       | WriteMask | 0x0200       | 0
;iosignals.indx++;// shut down pump

for ( dof=1; dof<sniffer; dof++)
{
iosignals.o[iosignals.indx] =  2048<<16| WriteMask | ValveBias    | dof
;iosignals.indx++;// send zero current to valve dac
}
}

int keypressed()
{
char key; int nread;

if ( (nread=read(0, &key,0))!=EOF ) return(key);
}



void user()
// 1. user() for the moment accepts orders from file
// 2. each bach of orders form one experiment
// 3. experiment must have
//    3.a expiriment ID
//    3.b Input Method
//    3.c Moving Method
//    3.d Time Duration
//    3.e Set Point (optional)
// 4. been, for the moment, in series with control loop user() steels
time,so , orders are in compact form
{

char dof, i, action, statusReq; int nread; fpos_t UserOrdersPos;

//timeFinal       holds the time of switching between experiments
//stored.working  index of desired working position into matrix
strored.pos
//dofstv.status         status signal from BusREpV4

//begin new expiriment

if (timeFinal < timeEllapsed  | expirimentStatus =='e')          //  if
expiriment time expired, or expiriment ends, read new orders

{

fgetpos(UserOrders, &UserOrdersPos);      //save current file position


fscanf(UserOrders,"%d",&expirimentID);    //read expiriment's ID number
fscanf(UserOrders,"%d",&statusReq); //read status requirments for current
experiment

//printf("%d %d",expirimentID,statusReq);
```

```
if ((dofstv.status & 0x02) == statusReq)//if status is as requested
continue
                              //available options are
                              //0: pressure off
                              //2: pressure on
      {


      if (expirimentID!=0)
            {

            expirimentStatus = 'b';       // a new expiriment beggins
                                    // this is a flag used by getdess() to
initialize or finalize expireiment variables

            printf("\n\a");
            printf("%d\n",expirimentID);
            printf("%d\n",statusReq);

            for (dof=1;dof<sniffer;dof++)
            {fscanf(UserOrders,"%c",&dessg.inp[dof]); printf("%d : %c
",dof,dessg.inp[dof]);} //set input method for all dofs
            printf("\n");

            fscanf(UserOrders,"%c",&dessg.move);
                  //set moving method for all dofs
            printf("%c\n",dessg.move);

            fscanf(UserOrders,"%c",&action);
            //set working position
            printf("%c\n",action);



            fscanf(UserOrders,"%d",&stored.working);
            printf("%d\n",stored.working);


            if (action=='r')
      //read desired working position from UserOrders
            for (dof=1;dof<sniffer;dof++)
fscanf(UserOrders,"%d",&stored.pos[dof][stored.working]);

            if (action=='c')
      //use current position as desired working position
            storesgs(stored.working,'r');


            fscanf(     UserOrders,"%d",&counts.exp );
                  //counts for identification loops
            printf("%d\n",counts.exp);

            fscanf(     UserOrders,"%lf",&timeDelta);
            //read experiment duration
            timeFinal = timeEllapsed + timeDelta;
            //set final time for experiment

            printf("resumming \nexperiment %d ... working position %d ...
for %f sec\n\n",expirimentID,stored.working,timeDelta);
```

```
            }

       else if (stored.working !=0)// expirimentID == 0  so set ARM to
zero position and wait for pressure down

            {
            stored.working = 0; dessg.move='s';

            for (dof=1;dof<sniffer;dof++) if (dessg.inp[dof]!='O' &
dessg.inp[dof]!='o') {dessg.inp[dof]='o';}

            }

}

else // pressure status is not as requested
      {
      fsetpos(UserOrders, &UserOrdersPos);            //Backward File
Position

      if ( difftime(time(NULL),timeLastDisplay)>= 5)  //display rarely
(every 5 sec)
            {
            time(&timeLastDisplay); //save current display time
            if (statusReq==2) printf("\aexpiriment %d turn pressure on  to
continue\n" ,expirimentID);
            if (statusReq==0) printf("\aexpitiment %d turn pressure off to
continue\n" ,expirimentID);
            }
      }

}

}

void getdessg()
{
//Uses an appropriate method and an input to construct
//desired positions torques or currents used by the control
//routine to compute valve currents
char dof, method, i; int mx,mn, data;

if (expirimentStatus=='b')    //upper part of global initialization code
                        //goes along with
                        //local initialization codes
                        //lower part of global initialization code at the
end of getdess()
                        //into this code various parametrers initialized

{
counts.in = 0;//printf("zero : %u\n",counts.in);

dist.okall = FALSE;

for (dof=1 ; dof<sniffer; dof++) reps[dof]=0;

}
```

```
// podition from file
// file form
// counts pos(1) pos(2) ...pos(10)
// counts : counts of 8.86MHz counter for each position
// pos(i) : position of DOF i

// position input from file, taking in mind of time
if ( dessg.inp[1]=='f' )

if ( counts.in >= counts.exp)

      {

      switch(dessg.move)
      {
      case 'p':
      if ( !dist.okall ) break;                                    //if not
all DOFs in proximity don't read new desired position
                                                              //otherwise
      case 'f':
      counts.in -= counts.exp; fscanf(pathfile,"%d",&counts.exp);
      //keep overflowing counts, read new counter interval

      for (dof=1;dof<sniffer;dof++) fscanf(pathfile,"%d",&dessg.pos[dof]);
      //read new desired position

      break;

      default:break;

      }

      }

if ( dessg.inp[1]=='F' ) //in 'F' read fast forward from file (don't take
mind of time)
      {
      counts.in = 0;                       //reset counts
      fscanf(pathfile,"%d",&counts.exp);  //read new counter interval

      for (dof=1;dof<sniffer;dof++) fscanf(pathfile,"%d",&dessg.pos[dof]);
      //read new desired position

      }

// all other input methods
for (dof=1 ; dof<sniffer; dof++) // for each dof select desired input
signals
      {

      switch( dessg.inp[dof] )
            {


            case 'a':case 'A':      // CAB product
                                 // in this mode the BIAS of position and
torque
                                 // signal conditioning circuits will vary
                                 // variation must be slow, all DOFs must be
at zero position
```

```
            if (expirimentStatus=='b')
                  {
                  //initialize DACs

                  iosignals.o[iosignals.indx] = 0             | WriteMask
      | ValveGain       | dof;iosignals.indx++; //disable valves
                  iosignals.o[iosignals.indx] = 2048<<16    | WriteMask |
ValveBias   | dof;iosignals.indx++;

                  iosignals.o[iosignals.indx] = 410<<16     | WriteMask |
ActPosGain1       | dof;iosignals.indx++;//position 1 stage gain

//                iosignals.o[iosignals.indx] = 205<<16     | WriteMask |
ActPosGain2       | dof;iosignals.indx++;//position 1 stage gain

                  iosignals.o[iosignals.indx] = 410<<16     | WriteMask |
TorqueGain  | dof;iosignals.indx++;//Torque           stage gain

                  AB.b[dof][1] = 0;

                  if (dof==sniffer-1) //after all dof initiallization
start main exipriment
                        {
                        expirimentStatus = 'w';
                        printf("\nexpiriment initialization done
%d\n",dof);
                        }

                  }

            if (expirimentStatus=='w' & counts.in>=counts.exp)    //wait
some time for next  variation
                  {

                  iosignals.o[iosignals.indx] = AB.b[dof][1]<<16  |
WriteMask   | ActPosBias1 | dof;iosignals.indx++;
                  iosignals.o[iosignals.indx] = AB.b[dof][1]<<16  |
WriteMask   | TorqueBias1 | dof;iosignals.indx++;

                  AB.b[dof][1] ++; AB.b[dof][1] %=4096;

                  if (dof==sniffer-1) counts.in -= counts.exp; //keep
overflowing counts

                  }//if counts
            break;

            case 'c':case 'C':      // calibration proccess
                              // in this mode the current varies from max
to min slowly
                              // allowing for full range dof movements with
low speed

                              // ****** only one dof must be calibrated at
a time ******
                              //the operator must attend the arm for proper
calibration
                              //and ensure FULL RANGE MOTIONS WITHOUT
COLLISIONS
```

```
            if (expirimentStatus == 'b') expirimentStatus='w'; //

            if (dofstv.mx[dof]<dofstv.fpos[dof])
dofstv.mx[dof]=dofstv.fpos[dof]; // look for maximum
            if (dofstv.mn[dof]>dofstv.fpos[dof])
dofstv.mn[dof]=dofstv.fpos[dof]; // look for minumum


            if (counts.in>=counts.exp & expirimentStatus=='w')    //wait
some time for next current variation
                  {
                  dessg.cur[dof] = dessg.cur[dof] + di[dof];// new current

                  // special treatmen when current gets extreme values
                  if (dessg.cur[dof]<=0  |   dessg.cur[dof]>=4095)
                        {
                        // change motion direction
                        if (dessg.cur[dof]<=0    )  {dessg.cur[dof]=0;
di[dof] = 1;}
                        if (dessg.cur[dof]>=4095 )  {dessg.cur[dof]=4095;
di[dof] =-1;}

                        if (reps[dof]>2)
                        if (dofstv.mx[dof]>dofstv.mn[dof])
                                          //reps is the number of full
rotations in any direction. When
                                          //at least one  full range motion
has been achived calculation can be done
                        {

                        if (abs(4096-dofstv.mx[dof]-dofstv.mn[dof])<32) //
symmetric configuration calculate new gain and bias
                        {
                        AB.a[dof][1] = ( 4032 *AB.a[dof][0]
)/(dofstv.mx[dof]-dofstv.mn[dof]);
                        AB.b[dof][1] = AB.b[dof][0];
                        }

                        else// use old gain and change only bias
                        {
                        AB.a[dof][1] = AB.a[dof][0];
                        AB.b[dof][1] =  (long int)( (4096  -
(dofstv.mx[dof]+dofstv.mn[dof]) )*invAB[dof]) / (long int)
(2*AB.a[dof][0])  + AB.b[dof][0];
                        }

                        //prepare iosignals
                        iosignals.o[iosignals.indx] = AB.a[dof][1]<<16  |
WriteMask   | 0x0A00 | dof;iosignals.indx++;
                        iosignals.o[iosignals.indx] = AB.b[dof][1]<<16  |
WriteMask   | 0x1700 | dof;iosignals.indx++;


                        // inspect for changes in a, b. End calibration
when changes are minor
                        if (  abs(AB.a[dof][1]-AB.a[dof][0])<=4 &
                              abs(AB.b[dof][1]-AB.b[dof][0])<=4 &
                              abs(4096-(dofstv.mx[dof]+dofstv.mn[dof]))<32
)
```

```
                              {dessg.inp[dof] = 'K'; printf("dof : %d
calibration     succefull\n",dof); expirimentStatus='e';}

                        printf("dof : %d,repetition : %d, max : %d, min :
%d, old a : %d, new a : %d, old b : %d, new b : %d\n",
dof,reps[dof],dofstv.mx[dof],dofstv.mn[dof],AB.a[dof][0],AB.a[dof][1],AB.b
[dof][0],AB.b[dof][1]);

                        AB.a[dof][0]=AB.a[dof][1];
AB.b[dof][0]=AB.b[dof][1];
                        }

                        else // inappropriate condition
dofstv.mx[dof]<dofstv.mn[dof]
                        {
                        dessg.inp[dof] = 'H'; //suspent motion
                        printf("\a\n");
                        printf("WARNING maximum value EQUALS minimum value
in DOF %d ...\n\n", dof);
                        printf("Motion is suspended.\n\n");
                        printf("Either DOF is blocked or SENSOR is in deep
saturation\n\n");
                        printf("ARM, still UNDER PRESSURE\n\n\a");
                        printf("CAREFULLY DEPRESSURIZE ARM, then check ARM
and/or change initial GAIN and BIAS\n\n\a");
                        printf("\a");
                        }


                        if (reps[dof]==25 & di[dof]!=0) //after 25 full
range motions end experiment automatically
                              {
                              dessg.inp[dof] = 'K'; //control DOF from
potensiometer and LOCK
                              printf("dof : %d calibration NOT
succefull\n",dof);
                              }



                        if (dessg.inp[dof] == 'c')                      //
if calibration is not finished prepare for new movement.
                              {dofstv.mx[dof] = 0;dofstv.mn[dof]=4095;} //
gain and bias have changed so reset maximum and minimum
                                                                        //
otherwise keep max and min values
                        reps[dof]++;
                        }//if (dessg.cur<0 | dess.cur>4095)

                  counts.in -= counts.exp;
      //keep overflowing counts

            }//if counts
            break;

            case 'k':case 'K':      // position by panelknod or locked

            if (expirimentStatus=='b') expirimentStatus='w'; //
```

```
            dessg.pos[dof] = dofstv.channels[dof][PKnod][(dofstv.indx +
reclen-1)%reclen];
            break;

            case 'i':         // open loop identification

            if (expirimentStatus=='b') expirimentStatus='w'; //

            if (counts.in >= counts.exp)
            {
            if (dof==sniffer-1) counts.in -= counts.exp;
      //after last dof reset counter (but keep overflow counts)
            dessg.cur[dof] = (signed int)stored.cur[dof][stored.working] +
(signed int)(rand()/(RAND_MAX/1024)) -512 ;

            // this input method does not modify desired position,
            // which remains constant until the input method changes
            // that way desired position can be used as safety position
            // around of which the identification can take place

            }

            break;

            case 'I':         // closed loop identification

            if (expirimentStatus=='b') expirimentStatus='w'; //

            if (counts.in >= counts.exp)
            {
            if (dof==sniffer-1)
            {
            counts.in -= counts.exp;      //after last dof reset counter
(but keep overflow counts)

            }
            dessg.pos[dof] = (signed int)stored.pos[dof][stored.working] +
rand()/(RAND_MAX/256) - 128;
            //printf("%d %d \n",dof,dessg.pos[dof]);
            }

            break;

            case 'o':   // in 'o' and 'O' mode the locking position is
given by stored.pos[dof][stored.working]
                        // consider of a stored.lock index ???

            if (expirimentStatus=='b') expirimentStatus='w'; //

            if ( dessg.move=='f' )
            {
            dessg.pos[dof] = stored.pos[dof][stored.working];
            }

            if ( dessg.move=='s' )

                  if ( dessg.inp[dof+1]=='O' | dof==(sniffer-1) )
                  {
                  dessg.pos[dof] = stored.pos[dof][stored.working];
                  }
```

```
            if ( abs(dofstv.fpos[dof] -
stored.pos[dof][stored.working])<128 )
            {
            dessg.inp[dof] = 'O';
//          printf("dof %d locked at position
%d\n",dof,stored.pos[dof][stored.working]);
            }

            break;

            case 'O':

            if (expirimentStatus=='b') expirimentStatus='w'; //

            dessg.pos[dof] = stored.pos[dof][stored.working];

            break;

            default: break;

            }
      }


//proximity check
dist.okall = TRUE;

for (dof=sniffer-1 ; dof>0; dof--) // for each dof calculate distances and
proximity
      {
      dist.ad[dof] = abs(dofstv.fpos[dof] - dessg.pos[dof]);

      dist.ok[dof] = dist.ad[dof]<128;

      dist.okall = dist.okall & dist.ok[dof]; if (dist.okall) dist.okdof =
dof;

      }

// lock or unlock dofs
for (dof=1 ; dof<sniffer; dof++)
      {

      switch( dessg.inp[dof] )
            {
            case'K':          //position by panelknod locked

            if ( dist.ok[dof] )
                  {
                  dessg.inp[dof] = 'k';
                  printf("unlocking dof : %d\a\n\a",dof);   // unlock
                  }
            else dessg.pos[dof] = stored.pos[dof][stored.working];// if no
proximity keep DOF locked

            break;

            case 'F':         //position by file locked
```

```
            if (dist.okall) dessg.inp[dof]='f'; //unlock
            else dessg.pos[dof] = stored.pos[dof][0];// if no proximity
keep DOF locked

            break;

            case 'i':
            if (!dist.okall)
            {
            // in 'i' desired position does not changes so is a good
safety point
            dessg.inp[dof]='o'; dessg.move = 's'; stored.working=0; //lock
to zero position
            Kp[dof] = Kp[dof]/abs(Kp[dof]);
            printf(
            "position out of bounds while identification dof %d set to %c
mode  at stored position [%d]\a\n"
                  ,dof,dessg.inp[dof],stored.working
                  );

            }

            break;

            default: break;
            }
      }


}//getdessg



//Reads The Initialization Configuration File
char ReadConfig()

{

unsigned int DACvalue, DACtot, DACind[31],  DOFtot, dof, i, j;

cfPtr1=fopen("uploadlog","w");
      if( (cfPtr2=fopen("gains_Trial","r") )==NULL )
      {
            fprintf(cfPtr1,"Can not find file named 'gains_Trial'\nNo
gains uploading\n");
            fclose(cfPtr1);
            return 1;
      }

      fscanf(cfPtr2,"%d",  &DOFtot);      /* read total number of DOFs  to
be updated */
      fscanf(cfPtr2,"%d",  &DACtot);      /* read total number of DACs  to
be updated  */

            fprintf(cfPtr1,"\t");
            for (j=0;j<DACtot;j++)
            {
            fscanf(cfPtr2,"%d",&DACind[j]); /*load DAC indeces (first row)
      */
            fprintf(cfPtr1,"%d \t",DACind[j]);
```

```
            }
            fprintf(cfPtr1,"\n");

            iosignals.indx = 0;

            for (i=1;i<=DOFtot;i++)
            {
            fscanf(cfPtr2,"%d", &dof);/* read dof (first column) */
            fprintf(cfPtr1,"%d \t",dof);

            for (j=0;j<DACtot;j++)
            {
            fscanf(cfPtr2,"%d",  &DACvalue);
            fprintf(cfPtr1,"%d \t",DACvalue);

            iosignals.o[iosignals.indx] =   DACvalue<<16   |WriteMask |
DACind[j]<<8    | dof ;iosignals.indx++;

            if (DACind[j]==10) AB.a[dof][0]=DACvalue;
            if (DACind[j]==23) AB.b[dof][0]=DACvalue;

            }
            fprintf(cfPtr1,"\n");
            }

            for (j=0;j<11;j++) fscanf(cfPtr2,"%d",  &Kp[j]);
            for (j=0;j<11;j++) fscanf(cfPtr2,"%d",  &Kd[j]);
            for (j=0;j<11;j++) fscanf(cfPtr2,"%d",  &Ki[j]);
            for (j=0;j<11;j++) printf("%d %d %d %d \n",j
,Kp[j],Kd[j],Ki[j]);


      fclose(cfPtr2);

      fprintf(cfPtr1,"DOFs :%d DACs :%d ",DOFtot, DACtot);

      fclose(cfPtr1);

}

void WriteConfig()
{
//this routine is to Write Configuration into a file
}

int main(void)
{

/*********************************************/
/* DAQmx Configure Code             ******/
/*********************************************/
printf("initializing NI-DIO...");

DAQmxResetDevice ("Dev1");                      // reset device

DAQmxErrChk (DAQmxCreateTask("Read",&TaskRead));      // associate name
"Read"  with pointer TaskRead
DAQmxErrChk (DAQmxCreateTask("Write",&TaskWrite));    // associate name
"Write" with pointer TaskWrite
```

```
// create  a channel for writting with name "WriteChannel" and associate
it with pointer TaskWrite
DAQmxErrChk (DAQmxCreateDOChan(TaskWrite,"Dev1/port0, Dev1/port1,
Dev1/port6, Dev1/port7 " ,"WriteChannel", DAQmx_Val_ChanForAllLines));

// create  a channel for Reading with name "ReadChannel" and associate it
with pointer TaskRead
DAQmxErrChk (DAQmxCreateDIChan(TaskRead, "Dev1/port3, Dev1/port4 "
,"ReadChannel", DAQmx_Val_ChanForAllLines));


DAQmxErrChk (DAQmxSetBufOutputBufSize(TaskWrite,0));// set to zero for
continue IO cycles
DAQmxErrChk (DAQmxSetBufInputBufSize(TaskRead,512));// no more than 512
commands on one cycle

DAQmxErrChk (DAQmxSetSampTimingType(TaskWrite,DAQmx_Val_Handshake));
DAQmxErrChk (DAQmxSetSampTimingType(TaskRead,DAQmx_Val_Handshake));

DAQmxErrChk (DAQmxSetSampQuantSampMode(TaskWrite,DAQmx_Val_ContSamps));
DAQmxErrChk (DAQmxSetSampQuantSampMode(TaskRead,DAQmx_Val_ContSamps));


DAQmxErrChk (DAQmxStartTask(TaskRead ));

printf("done\n");
/*********************************************/
/* Sarcos Configure Code            ******/
/*********************************************/

//printf("Turn of pressure, Set manually to ZERO POSITION then press c and
ENTER ...");
//char key; while( (key = getchar()) !='c');

printf("initializing SARCOS... \n");
iorec.indx=0; iosignals.indx = 0;//reset iosignals.indx

queryTime(); ReadConfig(); querySensors(); sendget('r');

char i;

for (i=1;i<reclen;i++); {queryTime(); querySensors(); sendget('r');} //
fill in temporary record

printf("done\n");

char dof;


if ( (UserOrders=fopen("UserOrders.txt","r") )!=NULL )

      {
      fscanf(UserOrders,"%lf",&timePressureIgnore);printf("watchTime :
%lf\n",timePressureIgnore);
      }
else
      {
      printf("\n****** program ABORDED ******\nUserOrders.txt does not
exist\n\a" );
```

```
      goto finalize;
      }


counts.in=0; counts.exp=0;

if( (pathfile=fopen("paths.txt","r"))!=NULL)
      printf("\n********* file paths.txt opened succefully
*********\n\n");
else
      printf("\n********* file paths.txt does not exists
*********\n\n");


time(&timeStart);       //store start time

timeLastDisplay = timeStart;

timeFinal = 0;                      //set expiration time for current
experiment to 0sec

expirimentID = 255;                 //must not be 0 to enter user()
correctly

storesgs(0,'r');              //save zero position in stored.pos[0]
storesgs(1,'r');stored.working = 1; //make one temporary copy in
stored.pos[1] and set as desired position

//lock ARM in zero position
for (dof=1; dof<sniffer; dof++) dessg.inp[dof] = 'o'; dessg.move = 's';
for (dof=1; dof<sniffer; dof++) printf("dof :%d, preset input method :%c,
zero position :%d \n",dof,dessg.inp[dof],stored.pos[dof][0]);

dist.okall = FALSE; dofstv.status = 0;

printf("\n\apressurize the ARM within %lf seconds otherwise the loop will
end automatically\a\n",timePressureIgnore);

printf("begin of control loop ... \n");

// call filtering routine, create dofstv.pos dofstv.vel, dofstv.acc
fltquad();


// enter control loop
// loop continues as long the is pressure to keep the ARM stable
// pressure is not in consideration for a short time in the beggining just
// to give a change to opperator to turn pressure on, while the loop is
active
while( ( timeEllapsed = difftime(time(NULL),timeStart) ) <
timePressureIgnore   |  (dofstv.status & 0x02)!=0 /**/)
{

      iosignals.indx = 0;     //prepare siganls to send,reset
iosignals.indx
      user();                 //get user orders

      // start filling up iosignals matrix

      queryTime();            //request for timming information
```

```
      getdessg();       //depending of oparation mode and method
                        //get desired signals


      controlDigPID();  //control routine

      querySensors();   //request for sensor information

      sendget('r');           //send output signals, get input signals

      fltquad();        //filter input signals from last sendget

}

printf("\n\a");
for (dof=1; dof<sniffer; dof++)
      printf("dof :%d, input :%c, zero position :%d current position
%d\n",dof,dessg.inp[dof],stored.pos[dof][0],dofstv.fpos[dof]);

finalize:
if (UserOrders!=NULL) fclose(UserOrders);
//finalize
iosignals.indx = 0;     //reset iosignals.indx
queryTime();
zeroCurrent();
querySensors();

sendget('r');printf("\n****** WARNING ******\nCURRENTS SET TO ZERO\n\a");

printf("..... end of control loop\n");
printf("duration : %f sec\n",timeEllapsed);
printf("*****************************************************************\n")
;
printf("**** DO NOT PRESSURIZE THE ARM BEFORE RESTART THE PROGRAM
****\n");
printf("*****************************************************************\n")
;


DAQmxErrChk (DAQmxStopTask(TaskRead ));
DAQmxErrChk (DAQmxStopTask(TaskWrite));
DAQmxResetDevice ("Dev1");


if (pathfile!=NULL) fclose(pathfile);// closing files


if (TRUE){

      printf("reporting please wait...\n");

      cfPtr1 = fopen("dataIOraw.csv","w");
      cfPtr2 = fopen("dataIOflt.csv","w");
      pathfile = fopen("KnodOut.csv","w");

      unsigned int i=0;
      while (i<iorec.indx)
      {
      /*write raw data to file*/
```

```
      fprintf(cfPtr1,"%d\t%d\n",iorec.o[i],iorec.i[i]);
      /*write filtered data to file*/
      fprintf(cfPtr2,"%d\t%d\t%d\t%d\t%d\t%d\t%d\n"
                                    ,(iorec.o[i]&0x0000C000)>>14  /*W or
R*/
                                    , iorec.o[i]&0x000000ff       /*DOF*/
                                    ,(iorec.o[i]&0x00003f00)>>8   /*DAC*/
                                    ,(iorec.o[i]&0xffff0000)>>16
      /*Dout16bit*/
                                    , iorec.i[i]                  /*Din
16bit*/
                                    ,(iorec.o[i]&0x0fff0000)>>16
      /*Dout12bit*/
                                    , iorec.i[i]&0x0fff     );    /*Din
12bit*/

      if (  iorec.o[i]&0x0000C000 == ReadMask   &
            iorec.o[i]&0x00003f00 == 0          &
            iorec.o[i]&0x000000ff<sniffer             )
fprintf(pathfile,"%d \n",iorec.i[i]);

      i++;
      }

      fclose(cfPtr1);
      fclose(cfPtr2);
      fclose(pathfile);
      printf(".......................> %d cycles\n",i);
      }


Error:
      if( DAQmxFailed(error) )
            DAQmxGetExtendedErrorInfo(errBuff,2048);
      if( TaskWrite!=0 ) {
            /*********************************************/
            // DAQmx Stop Code
            /*********************************************/
            DAQmxStopTask(TaskWrite);
            DAQmxClearTask(TaskWrite);
      }
      if( TaskRead!=0 ) {
            /*********************************************/
            // DAQmx Stop Code
            /*********************************************/
            DAQmxStopTask(TaskRead);
            DAQmxClearTask(TaskRead);
      }


      if( DAQmxFailed(error) )
      printf("DAQmx Error: %s\n",errBuff);

      DAQmxResetDevice ("Dev1");

      printf("End of program, press Enter key to quit\n");
      getchar();
      return 0;
}
```