Εγχειρίδιο λειτουργίας ηλιοστάτη,

Εργαστηρίου Φυσικής της Ατμόσφαιρας.

Sun tracker manual, LAP.

Νάτσης Αθανάσιος

natsisa@auth.gr

Καραγκιοζίδης Δημήτριος

dkaragki@auth.gr

Θεσσαλονίκη

2025-2-19

Ηλεκτρονική έκδοση: thanasisn.github.io

Contents

List of Figures

List of Tables

Αρχείο pdf

Στοιχειοθεσία έγινε με: bookdown, R, Rmarkdown, Pandoc, XeTeX.
Build: 7e932593

Σημείωση: **Ο πηγαίος κώδικας (source code) που παρατίθεται εδώ είναι ένα παράδειγμα εφαρμογής κάποιων τεχνικών. Σε περίπτωση που θέλετε να τον χρησιμοποιήσετε, προτείνουμε να δοκιμαστεί η καταλληλότητα του για την επιθυμητή χρήση, πρώτα σε μη σημαντικές εφαρμογές (non-critical). Τα παραδείγματα κώδικα εδώ, διαφέρουν από αυτά που χρησιμοποιούμε, λόγω της συνεχής εξέλιξης και βελτίωσης κατά την χρήση τους.**

Εισαγωγή

Το παρόν είναι ένας ανεπίσημος οδηγός για τη χρήση και τη λειτουργία του ηλιοστάτη του ΕΦΑ. Συντάχτηκε από της διαθέσιμες πληροφορίες για τον ηλιοστάτη καθώς και από την εμπειρία που αποκομίσαμε κατά τη χρήση του. Σκοπός μας είναι να βοηθήσει στην περαιτέρω ανάπτυξη και χρήση του, γι' αυτό ζητούμε την συνεισφορά σας στη βελτίωσή του. Για διορθώσεις και προσθήκες μπορείτε να επικοινωνήσετε με τον ThanasisN (natsisphysicist@gmail.com) ή το Εργαστήριο Φυσικής της Ατμόσφαιρας του Α.Π.Θ.

1 Ο ηλιοστάτης του Εργαστηρίου Φυσικής της Ατμόσφαιρας

1.1 Περιγραφή tracker (ηλιοστάτη)

Η συσκευή αποτελείται από σώμα αλουμινίου, δύο άξονες κίνησης, δύο κινητήρες και ηλεκτρονικό σύστημα ελέγχου. Οι άξονες έχουν ομόκεντρα γρανάζια τα οποία τίθενται σε κίνηση από stepper motors μέσω κατάλληλου ιμάντα (Εικόνα 1.1). Η επικοινωνία με το σύστημα ελέγχου γίνεται μέσω σειριακής θύρας (πραγματικής ή εικονικής με τη χρήση μετατροπέα usb/serial).

Και οι δύο κάθετοι άξονες (αζιμούθιου και ζενίθ) έχουν τις ίδιες δυνατότητες και χαρακτηριστικά μεταξύ τους, η λειτουργία τους είναι πανομοιότυπη και μπορούν να κινηθούν ταυτόχρονα και ανεξάρτητα ο ένας από τον άλλο.

Οι κινητήρες του tracker μπορούν να κινηθούν με σταθερό βήμα 0.6° ο καθένας. Στο πρώτο μοντέλο του ηλιοστάτη, για τη μετάδοση της κίνησης, οι κινητήρες έχουν στο γρανάζι τους 12 δόντια και κινούν μέσω ιμάντα τους άξονες, που έχουν από 70 δόντια ο καθένας. Αυτό δίνει την δυνατότητα οι άξονες του ηλιοστάτη να κινούνται με διακριτό βήμα $0.6^\circ \cdot 12/70 \simeq 0.10286^\circ$ για το παλιό μοντέλο. Τα καινούρια μοντέλα του ηλιοστάτη, αντίστοιχα έχουν βήμα 0.125°.

Για την εκτέλεση μιας πλήρους περιστροφής του κάθε άξονα ο κινητήρας πρέπει να κάνει $360^\circ/(0.6^\circ \cdot 12/70) = 3500$ βήματα στο παλιό μοντέλο και $360^\circ/0.125^\circ = 2880$ βήματα στα καινούρια μοντέλα. Ενώ η μέγιστη ταχύτατα περιστροφής των κινητήρων είναι περίπου μία περιστροφή ανά 15 δευτερόλεπτα.

1.2 Επικοινωνία με τον tracker

Οι κινητήρες ελέγχονται από μικροελεγκτή (microcontroller), ο οποίος βρίσκεται μέσα στο κύριο σώμα της συσκευής. Ο μικροελεγκτής είναι ήδη προγραμματισμένος να εκτελεί άμεσα τις εντολές που λαμβάνει, μέσω της σειριακής επικοινωνίας.

Το λειτουργικό σύστημα του υπολογιστή αναγνωρίζει τη σειριακή σύνδεση με τον microcontroller ως κάποια θύρα `COM#` στα Windows ή ως `/dev/ttyUSB#`, `/dev/ttyS#` σε GNU/Linux. Οι εντολές μεταδίδονται στη συσκευή, γράφοντάς τις σε κάποια από αυτές τις διευθύνσεις. Αντίστοιχα, οι απαντήσεις των εντολών διαβάζονται από την ίδια διεύθυνση. Τυπικά, η επικοινωνία είναι πανομοιότυπη με την εγγραφή (write) και την ανάγνωση (read) κειμένου (string) από αρχείο, με βήμα, μία γραμμή ανά εντολή. Κάθε γραμμή τερματίζεται με τον χαρακτήρα `'\r'` (Carriage return, CR).

Εικόνα 1.1: *Εσωτερικό του πρώτου μοντέλου tracker. Φαίνονται τα κύρια γρανάζια των αξόνων, ο κινητήρας του αζιμούθιου άξονα και το ηλεκτρονικό σύστημα ελέγχου, καθώς και το CHP 1.*

Η σειριακή επικοινωνία μπορεί να επιτευχθεί με άμεση σύνδεση του μικροελεγκτή σε σειριακή θύρα του υπολογιστή. Είτε, μέσω μετατροπέα USB UART (adapter usb to serial) σε θύρα USB του ηλεκτρονικού υπολογιστή (Σχήμα 1.1). Στην περίπτωσή μας χρησιμοποιούμε τον μετατροπέα 'FT232 USB-Serial (UART) IC'. Υπάρχει το ενδεχόμενο κάποιοι μετατροπείς UART να μην επιτυγχάνουν σωστή επικοινωνία, λόγω των τεχνικών τους χαρακτηριστικών.

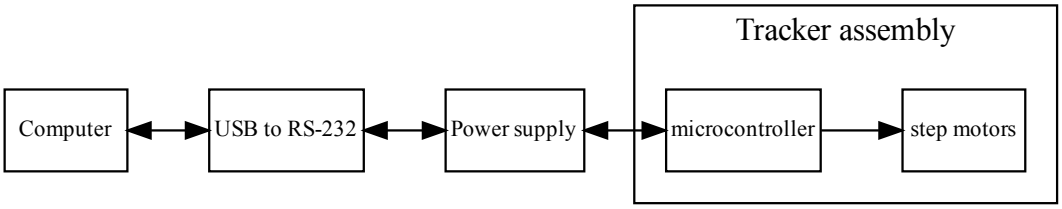


Figure 1.1: *Διάγραμμα επικοινωνίας του ηλιοστάτη με τον υπολογιστή ελέγχου.*

1.2.1 Επικοινωνία με τον Tracker σε περιβάλλον MatLab

Σε αυτό το στάδιο περιγράφονται οι εντολές και οι παράμετροι που απαιτούνται προκειμένου να επιτευχθεί επικοινωνία με τον Tracker σε περιβάλλον MatLab. Το MatLab διαθέτει έτοιμες συναρτήσεις και εντολές για την επικοινωνία με ένα Serial Object, με την προϋπόθεση να

έχουμε εγκατεστημένο το Instrument Control Toolbox. Προτού ανοίξουμε την σειριακή θύρα, πρέπει να δημιουργηθεί ένα Serial Object χρησιμοποιώντας την εντολή `serial`, εισάγοντας τις αντίστοιχες ρυθμίσεις του Tracker. Στο MatLab snippet που ακολουθεί, ορίζεται μια struct με όνομα 'Tracker', η οποία περιλαμβάνει τόσο τις ρυθμίσεις της σειριακής επικοινωνίας (SerialConfiguration), όσο και αυτό καθαυτό το Serial Object (SerialPort). Σημειώνεται ότι στον συγκεκριμένο κώδικα έχει γίνει hard coding της Communication Port σε `COM4`, καθώς ξέρουμε ότι σε αυτήν την θύρα είναι συνδεδεμένο το καλώδιο USB του Tracker. Για την αυτόματη εύρεση της COM Port μπορεί να χρησιμοποιηθεί η εντολή `instrfindall` που εντοπίζει όλες τις ενεργές συνδέσεις και επιστρέφει τις θύρες COM, στις οποίες βρίσκονται συνδεδεμένες συσκευές.

```
% Set Tracker's Serial Port Configuration and create the Serial Object
Tracker.SerialConfiguration.COM             = 'COM4';
Tracker.SerialConfiguration.BaudRate        = 4800;
Tracker.SerialConfiguration.DataBit         = 8;
Tracker.SerialConfiguration.StopBit         = 1;
Tracker.SerialConfiguration.Parity          = 'none';
Tracker.SerialConfiguration.Terminator      = 'CR';
Tracker.SerialConfiguration.Timeout         = 50;
Tracker.SerialConfiguration.InputBufferSize = 5000;

Tracker.SerialPort = serial(Tracker.SerialConfiguration.COM,...
    'BaudRate',Tracker.SerialConfiguration.BaudRate,...
    'DataBit',Tracker.SerialConfiguration.DataBit, ...
    'StopBit',Tracker.SerialConfiguration.StopBit, ...
    'Parity',Tracker.SerialConfiguration.Parity, ...
    'Terminator',Tracker.SerialConfiguration.Terminator, ...
    'Timeout',Tracker.SerialConfiguration.Timeout, ...
    'InputBufferSize',Tracker.SerialConfiguration.InputBufferSize);
```

Η παράμετρος `InputBufferSize` επιτρέπει τον έλεγχο του αριθμού των bytes που μπορούν να κρατηθούν στον buffer του μικροεπεξεργαστή και εφόσον δοθεί μια μεγάλη τιμή (όπως στην περίπτωσή μας, 5000), μπορούμε να δίνουμε εντολές στον Tracker την μία πίσω από την άλλη σε stack. Υπόψιν ότι ο μικροεπεξεργαστής σε αυτήν την περίπτωση θα επιστρέψει ως πρώτη απάντηση το αποτέλεσμα της εντολής που ολοκληρώθηκε πρώτη και στην συνέχεια με την ίδια λογική τις υπόλοιπες. Αφού δημιουργηθεί το Serial Object, χρησιμοποιώντας τις εντολές `fopen` και `fclose` μπορούμε να ανοίξουμε και να κλείσουμε αντίστοιχα την σειριακή θύρα. Με τις εντολές `fprintf` και `fscanf` δίνουμε εντολή στον Tracker και λαμβάνουμε την

απάντησή του αντίστοιχα σε μορφή ASCII. Παρακάτω παρουσιάζεται ένα παράδειγμα για την εύρεση της θέσης (step) του Αζιμούθιου άξονα

```
% Send the Command to the Tracker
fprintf(Tracker.SerialPort, 'AZ?');
```

```
% Wait for the Tracker's microntroller to fill the buffer (Typically 0.5 sec)
pause(time_delay)
```

```
% Get the Tracker's answer
[answer, bytes, msg_err] = fscanf(Tracker.SerialPort);
```

Με τις παραπάνω εντολές, ο Tracker θα απαντήσει με την θέση σε βήματα του Αζιμούθιου άξονα (πχ AZ:6000) στην μεταβλητή `answer`, τον αριθμό των bytes που στάλθηκαν (8 στην συγκεκριμένη περίπτωση) στην μεταβλητή `bytes` και τυχόν μηνύματα σφάλματος κατά την σειριακή επικοινωνία στην μεταβλητή `msg_err`. Άλλες χρήσιμες εντολές και παράμετροι για τον έλεγχο και την ορθή επικοινωνία με τον Tracker είναι η `Tracker.SerialPort.BytesAvailable`, η οποία επιστρέφει τον αριθμό των bytes που υπάρχουν διαθέσιμα στον buffer του microcontroller, η `flushinput(Tracker.SerialPort)` και η `flushoutput(Tracker.SerialPort)` που αδειάζουν τον Input και Output buffer αντίστοιχα.

1.3 Συνδεσμολογία

Έχει διαπιστωθεί εμπειρικά, ότι η σειρά της συνδεσμολογίας του ηλιοστάτη παίζει ρόλο στην ορθή ενεργοποίηση των επιμέρους στοιχείων και την επιτυχή λειτουργία του ηλιοστάτη. Για την επιτυχή επικοινωνία, προτείνουμε την αλληλουχία σύνδεσης (1-2-3-4) ή (1-4-3-2) όπως φαίνεται στο Σχήμα 1.2.

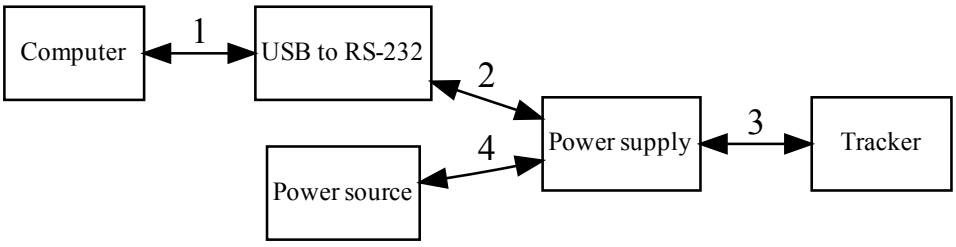


Figure 1.2: *Διάγραμμα συνδέσεων του ηλιοστάτη. Η κατάλληλη αλληλουχία σύνδεσης των στοιχείων είναι: (1-2-3-4) ή (1-4-3-2).*

2 Λειτουργία - Προγραμματισμός tracker

2.1 Γενικές πληροφορίες

Κατά τη λειτουργία πρέπει να δοθεί προσοχή, διότι δεν υπάρχει μηχανικός περιορισμός που να αποτρέπει την περιστροφή του άξονα κατά γωνίες μεγαλύτερες του πλήρη κύκλου. Αυτό μπορεί να προκαλέσει πρόβλημα γιατί το καλώδιο του tracker αλλά και τον οργάνων μπορεί να τυλιχθούν γύρω από τον άξονα και να καταστραφούν. Επίσης, σε περίπτωση που η συσκευή δεν έχει τοποθετηθεί μόνιμα π.χ. σε τρίποδα, μπορεί η περιστροφή να τραβήξει τα καλώδια και να ρίξει τη συσκευή.

Ο tracker όταν του δοθεί εντολή (reset), θα προσπαθήσει να μηδενίσει τον μετρητή βήματος αφού μετακινηθεί σε μία γνωστή θέση. Αυτό γίνεται με την περιστροφή του άξονα, μέχρι να φτάσει στη συγκεκριμένη θέση, την οποία αναγνωρίζει με μία φωτοδίοδο. Προσοχή, σε περίπτωση που δεν πάρει σήμα από τη φωτοδίοδο, θα συνεχίσει να περιστρέφεται ασταμάτητα. Αυτό μπορεί να συμβεί, αν το καπάκι είναι ανοιχτό σε έντονο φως ή αν υπάρχει κάποιο πρόβλημα επικοινωνίας με τη φωτοδίοδο.

Ο tracker θα μηδενίσει τον κάθε άξονα στη θέση `'5000'`. Από εκεί και πέρα μπορεί να κινηθεί και προς τις δύο κατευθύνσεις. Προσοχή, αν το βήμα γίνει αρνητικό από κάποια εντολή ή αν γίνει μεγαλύτερο του `'9999'` ο tracker θα κολλήσει και θα συνεχίσει να περιστρέφεται ασταμάτητα. Αυτό πρέπει να αποφευχθεί μέσω του προγραμματισμού του. Μελλοντικά μπορεί να προστεθεί κάποιος μηχανικός ή ηλεκτρικός μηχανισμός περιορισμού.

Εάν το βήμα γίνει ακατάλληλο και ο tracker περιστρέφεται ασταμάτητα, μπορεί να σταματήσει με την εντολή `'stop'`. Σε αυτήν την περίπτωση, αν ερωτηθεί για τη θέση του επιστρέφει την τιμή `'9999'`.

Όταν ο ηλιοστάτης τροφοδοτείται και είναι σε λειτουργία, οι κινητήρες δεν θα επιτρέψουν την ελεύθερη περιστροφή των αξόνων. Σε περίπτωση που χρειάζεται να περιστραφούν ελεύθερα, πρέπει να διακοπεί η τροφοδοσία. **Σημείωση**: για το πρώτο/παλιό μοντέλο του ηλιοστάτη (ορθογώνιο παραλληλόγραμμο καπάκι) η αντίσταση των κινητήρων σε εξωτερική περιστροφή είναι πολύ μικρή, με αποτέλεσμα να είναι εύκολη η τυχαία μετατόπισή του από εξωτερικούς παράγοντες. Σε αυτή την περίπτωση, είναι καλό να γίνει επανεκκίνηση του προγράμματος ελέγχου του tracker, ώστε να ξαναβρεί τη θέση του κάνοντας reset των αξόνων και στην συνέχεια, να ελεγχθεί αν η διόπτευση (sighting) συνεχίζει να είναι σωστή.

2.2 Παράμετροι επικοινωνίας με τον microcontroller του tracker

Στον Πίνακα 2.1 περιγράφονται οι απαραίτητες παράμετροι για την σύνδεση με τη σειριακή θύρα του tracker. Κατά την επικοινωνία, είναι χρήσιμο να υπάρχει και κάποιο άνω όριο στον χρόνο που περιμένει ο υπολογιστής, πριν αποφασίσει ότι η επικοινωνία δεν είναι εφικτή. Διαφορετικά, αν κάποιο μήνυμα περιμένει να διαβαστεί από τη θύρα και αυτό δεν γίνει, μπορεί να παγώσει η επικοινωνία.

Table 2.1: *Παράμετροι επικοινωνίας του tracker μέσω σειριακής θύρας RS-232.*

bits per second	**4800** (baudrate)
data bits	**8** (bytesize)
parity	**none**
stop bits	**1**
Flow control	**none**
timeout (optional)	90 sec

2.3 Εντολές tracker

2.3.1 Μορφή εντολών (format)

Η επικοινωνία με τον microcontroller του tracker γίνεται με εντολές που στέλνονται μέσω σειριακής θύρας επικοινωνίας. Οι εντολές έχουν την μορφή κειμένου 'string' το οποίο τερματίζεται με το σύμβολο `'cr'` ή `'\r'` ('carriage return' το τυπικό 'enter' των MS Windows).

Οι εντολές που έχουν ως όρισμα αριθμό βημάτων, πρέπει να δίνονται με διαμόρφωση τεσσάρων ψηφίων. Όταν είναι αναγκαίο πρέπει να προπορεύονται από τα αντίστοιχα μηδενικά (padded with zero). Για παράδειγμα 1000, 0999, 0099, 0001. Αυτό ισχύει για την απόλυτη θέση (π.χ. `'AZ=0100'`) και για τη σχετική μετατόπιση (`'AZ+0100'`).

Αντίστοιχα, οι εντολές που αφορούν ταχύτητες έχουν την ίδια λογική μόνο που χρησιμοποιούν δύο ψηφία.

Προσοχή: Η απάντηση κάθε εντολής (ανάλογα και με τις ρυθμίσεις της σειριακής επικοινωνίας) πρέπει να διαβαστεί από τη σειριακή σύνδεση γιατί διαφορετικά δεν θα αδειάσει το buffer της σειριακής θύρας και μπορεί να παγώσει η επικοινωνία.

2.3.2 Μηδενισμός του βήματος του microcontroller

Οι εντολές reset του Πίνακα 2.2 επιστρέφουν τους κινητήρες στην αρχική τους ('μηδενική') θέση. Ο microcontroller αναλαμβάνει την κίνηση. Τους περιστρέφει μέχρι την φωτοδίοδο

και εκεί ρυθμίζει το εσωτερικό βήμα κάθε άξονα στη τιμή 5000. Μετά την εκτέλεση της εντολής ο tracker επιστρέφει τη διαφορά στη θέση από τον προηγούμενο μηδενισμό. Αυτές οι εντολές συνήθως χρησιμοποιούνται κατά την εκκίνηση της λειτουργίας ώστε να βεβαιωθούμε για την απόλυτη θέση του tracker, αφού δεν υπάρχει άλλος μηχανισμός αναγνώρισης της θέσης. Προσοχή, για την σωστή λειτουργία του μηδενισμού, οι παρακάτω εντολές πρέπει να δοθούν ξεχωριστά (όχι σε stack) και να περιμένουμε τον μηδενισμό του κάθε άξονα με την αντίστοιχη απάντηση από τον Tracker προτού δοθεί η επόμενη εντολή. Έπειτα από τον μηδενισμό του κάθε άξονα πρέπει να δοθεί εντολή κίνησης του ίδιου άξονα πριν δοθεί εντολή μηδενισμού του επόμενου άξονα, αλλιώς δημιουργείται πρόβλημα στην επικοινωνία και ο Tracker στην συνέχεια δεν επιστρέφει την απάντηση OK όταν μεταβεί σε κάποια ζητούμενη θέση. Υποθέτοντας ότι η θέση του αζιμούθιου, του ζενίθ και του φίλτρου της φωτοδιόδου είναι 5000, προτείνεται μετά από τον μηδενισμό του κάθε άξονα η κίνηση του ίδιου άξονα στην θέση 5000 (δεν θα μετακινηθεί αφού ήδη βρίσκεται σε αυτήν την θέση, ωστόσο αποφεύγεται το πρόβλημα της επικοινωνίας).

Table 2.2: *Εντολές μηδενισμού (reset) των αξόνων του tracker.*

Εντολή	Λειτουργία	Απάντηση tracker
`DA<cr>`	reset azimuth motor	`eA:####` (όταν βρει τη φωτοδίοδο)
`DF<cr>`	reset filters (not used)	`eF:####` (όταν βρει τη φωτοδίοδο)
`DZ<cr>`	reset zenith motor	`eZ:####` (όταν βρει τη φωτοδίοδο)

2.3.3 Κίνηση των αξόνων

Η εντολές για την κίνηση των αξόνων έχουν τη γενική μορφή: `xx@####<cr>` και αποτελούνται από τρία τμήματα (εδώ τα συμβολίζουμε με 'x', '@' και '#'). Η εξήγηση των επιμέρους στοιχείων γίνεται στον Πίνακα 2.3 και 2.4. Παραδείγματα έγκυρων εντολών βρίσκονται στον Πίνακα 2.6. Το σύνολο των εντολών του ηλιοστάτη παρατίθεται στον Πίνακα 2.5.

Table 2.3: *Τα μέρη της εντολής του ηλιοστάτη.*

Παράμετρος	Λειτουργία
xx	Η παράμετρος στην οποία αναφέρεται
@	Τελεστής της λειτουργίας που εκτελείται (?, =, +, -)
####	Αριθμητική τιμή της εντολής (##: για τις ταχύτητες)
<cr>	Ο χαρακτήρας τερματισμού της εντολής

Table 2.4: *Παράμετροι ελέγχου (operators/modifiers) των εντολών του ηλιοστάτη.*

Τελεστής (@)	Λειτουργία	Απάντηση
?	Ζητά την τιμή της μεταβλητής από τον ηλιοστάτη, χωρίς ####	xx:####<cr> ή xx:##<cr>
=	Θέτει την τιμή της μεταβλητής, ## για παραμέτρους ταχύτητας και #### για θέσης των αξόνων	OK<cr>
+	Αυξάνει τη μεταβλητή κατά τη δοσμένη ποσότητα, ## για παραμέτρους ταχύτητας και #### για θέσης των αξόνων	OK<cr>
-	Μειώνει τη μεταβλητή κατά τη δοσμένη ποσότητα ## για παραμέτρους ταχύτητας και #### για θέσης των αξόνων	OK<cr>

Table 2.5: *Εντολές ηλιοστάτη.*

Εντολές	Μέγεθος
AZ@####<cr>	Βήμα αζιμούθιου άξονα
ZE@####<cr>	Βήμα ζενίθιου άξονα
FR@####<cr>	Βήμα τροχού φίλτρων
SA@##<cr>	Ταχύτητα αζιμούθιου άξονα
SZ@##<cr>	Ταχύτητα ζενίθιου άξονα
SF@##<cr>	Ταχύτητα τροχού φίλτρων
OA@####<cr>	Τροποποίηση αρχικής αζιμούθιας θέσης
OZ@####<cr>	Τροποποίηση αρχικής ζενίθειας θέσης
OF@####<cr>	Τροποποίηση αρχικής θέσης φίλτρου
IA@####<cr>	Τροποποίηση της θέσης της φωτοδιόδου του αζιμούθου άξονα
IZ@####<cr>	Τροποποίηση της θέσης της φωτοδιόδου του ζενίθιου άξονα
IF@####<cr>	Τροποποίηση της θέσης της φωτοδιόδου του τροχού φίλτρων
TA@####<cr>	Τροποποίηση της ζητούμενης αζιμούθιας θέσης (μόνο ερώτηση)
TZ@####<cr>	Τροποποίηση της ζητούμενης ζενίθειας θέσης (μόνο ερώτηση)
TF@####<cr>	Τροποποίηση της ζητούμενης θέσης του τροχού φίλτρων (μόνο ερώτηση)
GO XXXX,YYYY<cr>	Μετακινεί ταυτόχρονα την αζιμούθια και ζενίθεια θέση σε XXXX και YYYY βήματα αντίστοιχα
STOP<cr>	Σταματά την κίνηση του ηλιοστάτη
DEBUG ON/OFF<cr>	Ενεργοποιεί και απενεργοποιεί την κατάσταση αποσφαλμάτωσης (debuging). Στην περίπτωση του ON, ο ηλιοστάτης αποκρίνεται με την εντολή που του δόθηκε και την αντίστοιχη απάντηση

Η εντολή `stop<cr>` σταματά την κίνηση των αξόνων. Λειτουργεί ακόμα και όταν ο ηλιοστάτης έχει σφάλμα θέσης. Αυτό μπορεί να συμβεί, στην περίπτωση που το βήμα ενός άξονα έχει φτάσει σε μη επιτρεπτή τιμή και ο τελευταίος περιστρέφεται ασταμάτητα. Με την παραπάνω εντολή ο ηλιοστάτης σταματά την κίνηση και αποκρίνεται με την τιμή 9999 στο ερώτημα της θέσης.

Table 2.6: *Παράδειγμα έγκυρης αλληλουχίας εντολών ηλιοστάτη.*

Εντολή	Αποτέλεσμα
DA<cr>	Μηδενισμός της θέσης του αζιμούθιου άξονα
DZ<cr>	Μηδενισμός της θέσης του ζενίθιου άξονα
AZ=6000<cr>	Μετακίνηση αζιμούθιου άξονα στη θέση 6000
ZE+0010<cr>	Μετακίνηση άξονα ζενίθ 10 βήματα στη θετική φορά
AZ-0100<cr>	Μετακίνηση άξονα αζιμούθιου 100 βήματα στην αρνητική φορά
AZ?<cr>	Ερώτημα θέσης, με απόκριση: AZ:5900<cr>
SA=60<cr>	Ταχύτητα του αζιμούθιου άξονα 60
?<cr>	θέσεις των δύο αξόνων και του φίλτρου.
	Απαντήσεις: AZ:5010<cr>, ZE:5900<cr>, FR:####<cr>

2.4 Permissions to access USB serial σε GNU/Linux

Με σύνδεση USB-to-serial, η συσκευή φαίνεται ως `/dev/ttyUSB0`. Ως συσκευή συστήματος, συνήθως χρειάζεται να έχετε τα κατάλληλα δικαιώματα στο σύστημα για να μπορείτε να την χρησιμοποιήσετε.

```
## Add current user to dialout group (needs reboot)
sudo usermod -a -G dialout $USER
## Allow anyone to use the device (less secure, not permanent)
sudo chmod 666 /dev/ttyUSB0
```

2.5 Resetting USB/serial interface σε GNU/Linux

Κάποιες ιδέες και προτάσεις για το πως μπορεί να γίνει επανασύνδεση της επικοινωνίας χωρίς να αποσυνδεθεί η φυσική σύνδεση. Έχουν παραχθεί και τα αντίστοιχα 'bash script' που μπορούν να βρουν την κατάλληλη USB συσκευή και να εκτελέσουν τις παρακάτω ενέργειες.

2.5.1 Αποσύνδεση των αντίστοιχων module/firmware από τον kernel

Αυτές οι εντολές απενεργοποιούν και ενεργοποιούν τα κομμάτια του συστήματος που χειρίζονται τις σειριακές συσκευές και τις συσκευές USB.

```
rmmod  ftdi_sio
rmmod  usbserial
modprobe ftdi_sio
modprobe usbserial
```

2.5.2 Αποσύνδεση της συσκευής από το σύστημα

Οι παρακάτω εντολές στέλνουν ένα σήμα σύνδεσης και αποσύνδεσης στην συσκευή USB.

```
sudo sh -c "echo 0 > /sys/bus/usb/devices/1-1/authorized"
sudo sh -c "echo 1 > /sys/bus/usb/devices/1-1/authorized"
```

2.5.3 Αποσύνδεση του mountpoint από το σύστημα (untested)

Αν και δεν έχουν δοκιμαστεί, οι εντολές αυτής της μορφής μπορούν να αποσυνδέσουν και να επανασυνδέσουν την διαδρομή αρχείου που αντιστοιχεί στην συσκευή και επομένως να ανανεώσουν την επικοινωνία.

```
unbind /dev/ttyUSB0
bind   /dev/ttyUSB0
```

2.6 Προγραμματισμός

Ο έλεγχος και ο αυτοματισμός της λειτουργίας του tracker, έχει γίνει στη γλώσσα προγραμματισμού Python.

2.6.1 Python

Η Python επιλέχτηκε ως μία σχετικά εύκολη γλώσσα προγραμματισμού με ευρεία σε χρήση σε ποικιλία εφαρμογών. Είναι ελεύθερο λογισμικό και είναι διαθέσιμη για όλα τα συστήματα και αρχιτεκτονικές υπολογιστών. Είναι δοκιμασμένη σε εξειδικευμένες, αλλά και γενικές επιστημονικές εφαρμογές, με πλήθος αντίστοιχων εργαλείων/βιβλιοθηκών (modules).

Είναι interpreting γλώσσα (τρέχει scripts χωρίς compiling) και αυτό δίνει αμεσότητα στη χρήση της.

Τα βασικότερα module που χρησιμοποιήσαμε, από επιστημονικής άποψης, είναι αυτά για τον υπολογισμό της θέσης του ήλιου στον ουρανό στη τοποθεσία μέτρησης. Αυτά είναι το astropy και το pyephem (ephem). Στην τελική εφαρμογή, αναγκαστήκαμε να χρησιμοποιήσουμε το pyephem λόγω κάποιο απροσδιόριστου προβλήματος λόγω λειτουργικού συστήματος (windows).

Το astropy είναι πιο σύγχρονο και ενημερωμένο με υψηλή ακρίβεια και χρήση στην αστρονομία. Το pyephem βασίζεται σε παλιότερο κώδικα (παραχώρηση από τον προγραμματιστή του xephem) που όμως είναι δοκιμασμένος και σε άλλες εφαρμογές.

Ο κώδικας τεκμηριώνεται από αυτοματοποιημένο κείμενο που παράγετε με το pydoc ή το doxygen από τα σχόλια που υπάρχουν μέσα σε αυτόν. Η python υποστηρίζει κάποιες προδιαγραφές για τα σχόλια ώστε να τα χρησιμοποιεί ως τεκμηρίωση του προγράμματος. Εδώ θα περιγραφούν πιο γενικές έννοιες της χρήσης και των λειτουργιών του, και όποια σχόλια που είναι χρήσιμα για την γενικότερη κατανόηση του συστήματος. Προφανώς αυτό το έγγραφο είναι συμπληρωματικό της ερευνητικής εργασίας για την οποία γράφτηκε.

2.6.2 `sun_tracker_main.py`

Το κύριο πρόγραμμα λειτουργίας του tracker. Αυτό το πρόγραμμα ελέγχει την λειτουργία του tracker και πρέπει να εκτελείτε συνεχώς για όσο χρειάζεται τα όργανα να ακολουθούν τον ήλιο. Στον πηγαίο κώδικα υπάρχουν λεπτομέρειες των λειτουργιών και της εφαρμογή τους.

2.6.2.1 Απαραίτητα και Βοηθητικά προγράμματα

- `sun_vector_astropy.py`
- `sun_vector_ephem.py`
- `param_location.py`
- `tracker_functions.py`

2.6.2.2 Επιγραμματικά οι λειτουργίες του προγράμματος

- Σειριακή επικοινωνία
- Επικοινωνία tracker
- Έλεγχος και επανασύνδεση επικοινωνίας
- Ορατότητα του ήλιου
- Θέση του ήλιου
- Logging (και για το βοηθητικά προγράμματα)
- Αποστολή κατάστασης tracker στο δίκτυο
- Λήψη παραμέτρων sighting

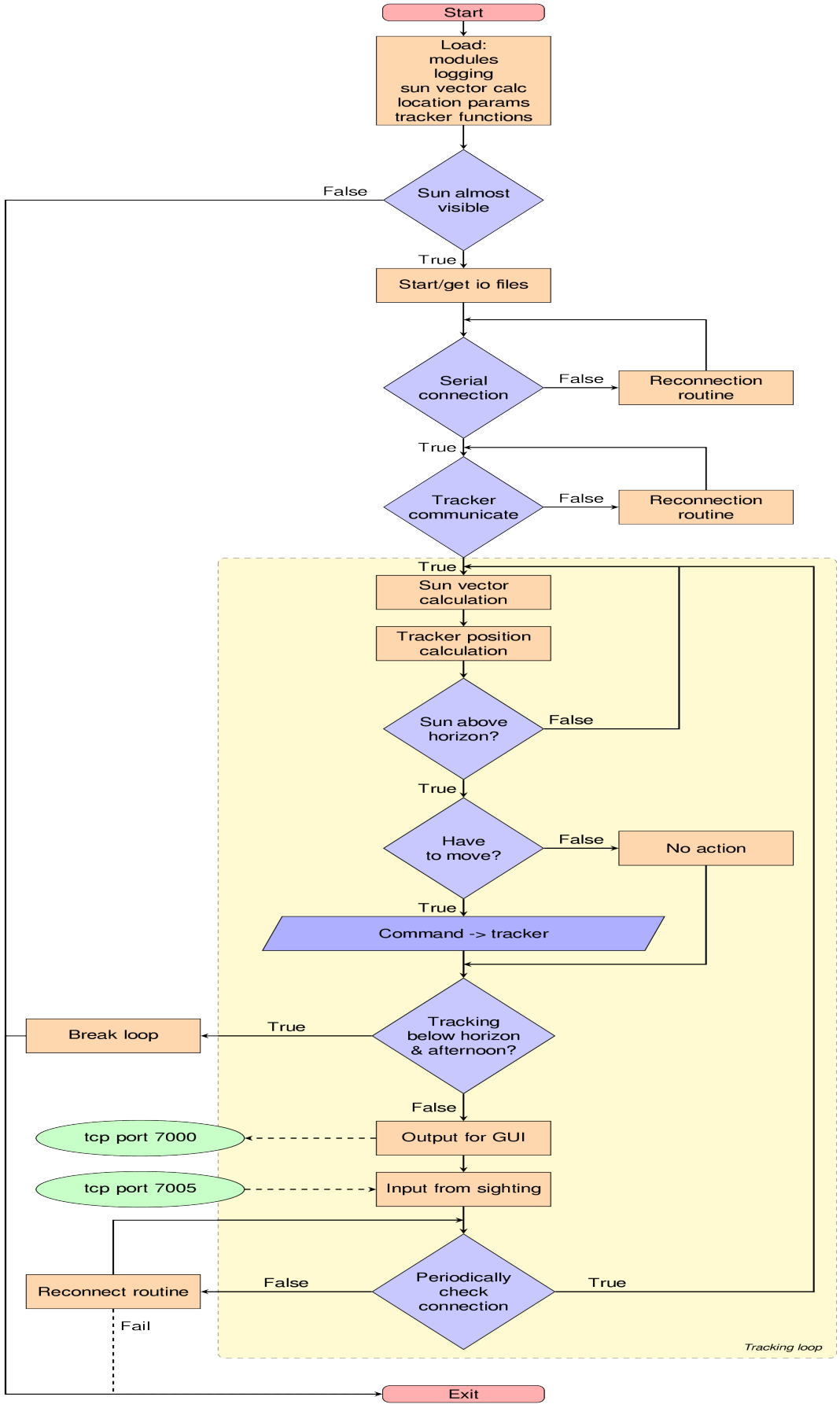


Figure 2.1: *Λογικό διάγραμμα προγράμματος*

2.6.3 `sun_vector_astropy.py` και `sun_vector_ephem.py`

Απλά wrapper προγράμματα που περιέχουν από μία μέθοδο υπολογισμού της θέσης του ήλιου.

Ο σκοπός τους είναι να προετοιμάζουν τις κατάλληλες βιβλιοθήκες και να παρουσιάζουν την ίδια λειτουργία ως προς την χρήση και τα αποτελέσματα στο `sun_tracker_main.py`.

- Για λειτουργικό σύστημα windows το astropy δεν λειτουργεί κανονικά.
- Το ephem δεν λειτουργεί καλά κοντά στον ορίζοντα.
- Υπάρχουν διαθέσιμες και άλλες μέθοδοι στη python για αυτός τους υπολογισμούς.
- Υπάρχουν έτοιμα και κάποια binary γραμμένα σε C, Fortan και Basic που μπορούν να ενσωματωθούν σε κάποιο wrapper.

Πιθανών να χρησιμοποιηθεί άλλη μέθοδος από το ephem που χρησιμοποιείται τώρα.

2.6.4 `param_location.py`

Περιέχει τις παραμέτρους τις τοποθεσίας, γεωγραφικό πλάτος, μήκος και υψόμετρο και τα όρια του τοπικού ορίζοντα. Επίσης περιέχει κάποια κλιματολογία της τοποθεσίας η οποία γίνεται διαθέσιμη στην μέθοδο υπολογισμού της θέσης του ήλιου.

2.6.5 `tracker_functions.py`

Περιέχει κάποιες βοηθητικές μεθόδους για το `sun_tracker_main.py` καθώς και κάποιες μεθόδους για την λειτουργία του tracker. Για τον tracker έχει αποθηκευμένες κάποιες τυπικές τιμές για την λειτουργία του. Δίνει την δυνατότητα αποθήκευσης, αλλαγής και φόρτωσης των ρυμθμίσεών του. Και μπορεί να κάνει ελέγχους των τιμών των παραμέτρων του και να προστατεύσει από εκτός ορίων τιμές.

2.6.6 `tracker_sighting_no_ui_tcp-port.py`

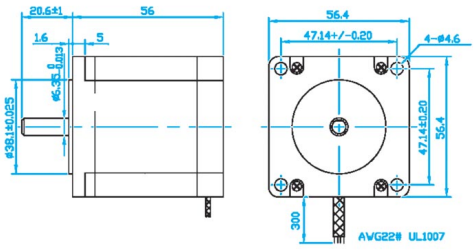
Αυτό το πρόγραμμα ελέγχει το offset (γωνία σε μοίρες) των δύο αξόνων του tracker. Τρέχει σε τερματικό και στέλνει τις τιμές στο `sun_tracker_main.py` οι αλλαγές εφαρμόζονται άμεσα στη θέση του tracker. Επίσης μπορεί να δώσει εντολή στο κύριο πρόγραμμα να αποθηκεύσει, να μηδενίσει ή να διαβάσει τις τιμές από τον δίσκο. Αν οι αλλαγές δεν σωθούν την επόμενη φορά που θα τρέξει το κύριο πρόγραμμα, θα διαβάσει τις αποθηκευμένες τιμές από τον δίσκο. Οι αλλαγές παρόλα αυτά θα γραφτούν στο log κατά την κανονική έξοδο του κύριου προγράμματος. Το βήμα αλλαγής της τιμής του offset καθορίζεται στον κωδικά του προγράμματος και μπορεί να είναι οποιαδήποτε δεκαδική τιμή.

3 Ηλεκτρικά - Μηχανικά Χαρακτηριστικά

3.1 Κινητήρες tracker (57SH56-4AM)

57SH56

HIGH TORQUE HYBRID - 400 STEP — Step Motor



Dimensions in mm.

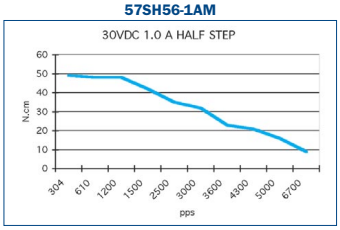
Specifications

	MODEL		57SH56-1AM	57SH56-2AM	57SH56-3AM	57SH56-4AM
1	STEP ANGLE		0,9°	0,9°	0,9°	0,9°
2	STEP ANGLE ACCURACY (FULL STEP, NO LOAD)	%	± 5%	± 5%	± 5%	±5%
3	RATED VOLTAGE	V	7,4	3,6	2,3	2,5
4	CURRENT/PHASE	A	1	2	3	2,8
5	RESISTANCE/PHASE	Ω	7,4	1,8	0,75	0,9
6	INDUCTANCE/PHASE	mH	10	2,5	1,1	2,5
7	DETENT TORQUE	mNm	40	40	40	40
8	HOLDING TORQUE	Ncm	90	90	90	126
9	ROTOR INERTIA	g-cm²	300	300	300	300
10	WEIGHT	Kg	0,7	0,7	0,7	0,7
11	NUMBER OF LEADS	n°.	6	6	6	4

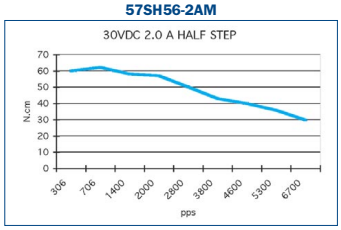
57SH..-A single shaft • **57SH...-B** double shaft

Speed vs. Torque Characteristics

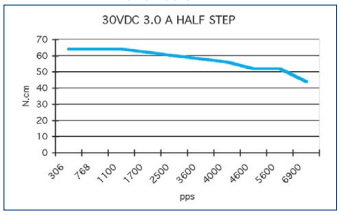
57SH56-1AM



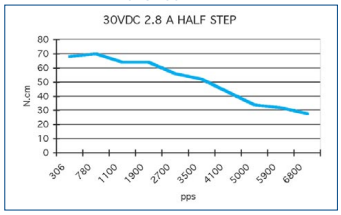
57SH56-2AM



57SH56-3AM



57SH56-4AM



Characteristics

RESISTANCE ACCURACY
± 10%

INDUCTANCE ACCURACY
± 20%

TEMPERATURE RISE
80° C max. (rated current, 2 phase on)

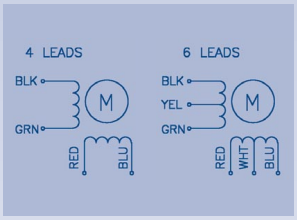
AMBIENT TEMPERATURE
-10° C - + 50° C

INSULATION RESISTANCE
100 M Ω min., 500 VDC

DIELECTRIC STRENGTH
500 VAC for one minute

SHAFT RADIAL PLAY
0,06 max. (450 g-load)

SHAFT AXIAL PLAY
0,08 max. (450 g-load)



AVAILABLE OPTIONS

Motor modifications:
custom winding, special bearing, special tap.
Shaft modifications:
flat, pinion, keyway, length.
Leadwire modifications:
wire type, wire color, wire length, connector installation.

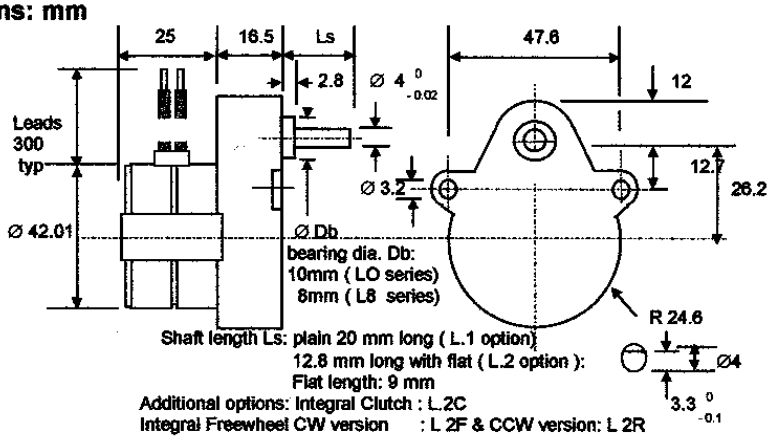
3.2 Κινητήρες filter wheel (G11)

Geared Stepper Motor P542-M48 Series

The P5-M48 series provides a combination of optimum performance and price for use in instrumentation applications which require digital control of position and speed. Features include:

- High performance permanent magnet stepper motor
- Precision Ovoid gearhead incorporating metal gears for optimum torque transmission
- Wide range of standard gear ratio options from stock
- Choice of output shaft options
- Optional integral freewheel and clutch
- Special shaft and gear ratios to meet customer special requirements

Dimensions: mm



P542-M48 geared stepper motor performance

Geared Stepper Motor	Ratio	Steps per rev. at output	Holding Torque (Ncm)	Max Working Torque (Ncm)	Typical Working Torque (Ncm)
P542-M48 -G01....	25:6	200	19.8	13.5	6.0
-G03	25:4	300	29.7	20.3	9.0
-G04	25:3	400	39.6	27.0	12.0
-G05	10:1	480	42.9	29.3	13.0
-G06	25:2	600	53.6	36.6	16.3
-G08	50:3	800	71.5	48.7	21.7
-G09	20:1	960	85.8	58.5	26.0
-G11	25:1	1,200	100.0	73.1	32.5
-G14	100:3	1,600	100.0	97.5	43.3
-G16	125:3	2,000	100.0	100.0	54.2
-G17	50:1	2,400	100.0	100.0	65.0
-G19	125:2	3,000	100.0	100.0	72.5
-G21	250:3	4,000	100.0	100.0	90.0
-G23	125:1	6,000	100.0	100.0	100.0
-G27	250:1	12,000	Use P535-M48 series for ratios of 250:1 and above		

Standard Versions:		**P542-M482U**	**P42-M481U**	Step rate @
Number of phases		4	4	typical working
Rated voltage (L/R Drive)		12	5	torque
Current per phase	(mA)	230	550	L/R : 300 Hz
Resistance per phase	(Ohms)	52.4	9.1	L/4R: 550 Hz
Inductance per phase	(mH)	51.7	8.1	

Instruction Leaflet
Bedienungsanleitung
Hojas de instrucciones
Feuille d'instructions
Foglio d'instruzioni

Wiring instructions GB

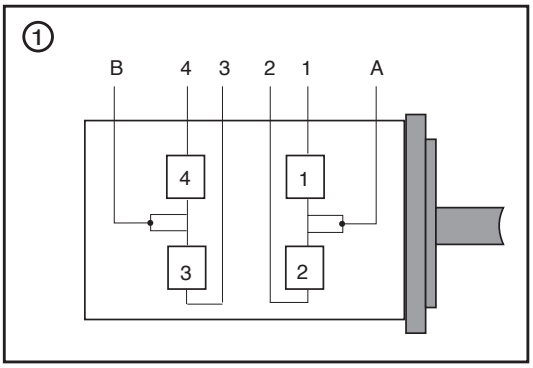
Anweisungen zur Beschaltung D

Instrucciones de cableado E

Instructions de câblage F

Istruzioni di cablaggio I

Figures / Abbildung / Figures / Figura



GB — RS Stock No.

351-4574, 351-4580, 351-4603, 351-4619
351-4625, 351-4631, 351-4647

Lead colours (Figure 1)

1	A	2	3	B	4
Yellow	Red	Orange	Brown	Green	Black

Permanent magnet stepper motor specification

RS stock no	Description	Steps/ Revs	Maximum Holding Torque	Maximum Phase Current Uni-polar Operation
351-4574	motor	48	2.5 Ncm	0.18 Amp
351-4580	motor	48	6.6 Ncm	0.55 Amps
351-4603	geared motor	200	19.8 Ncm	0.55 Amps
351-4619	geared motor	600	53.6 Ncm	0.55 Amps
351-4625	geared motor	1200	100 Ncm*	0.55 Amps
351-4631	geared motor	2400	100 Ncm *	0.55 Amps
351-4647	geared motor	6000	100 Ncm*	0.55 Amps

Note*: Limited gearhead maximum torque rating

D — RS Best-Nr.

351-4574, 351-4580, 351-4603, 351-4619
351-4625, 351-4631, 351-4647

Leiterfarben (Abbildung 1)

1	A	2	3	B	4
Gelb	Rot	Orange	Braun	Grün	Schwarz

Technische Daten der Schrittmotoren mit Permanent-Magnet

RS Best.-Nr.	Beschreibung	Schritte/ Umdrehung	Max. Halte-drehmoment	Max. Phasenstrom Unipolar-betrieb
351-4574	motor	48	2,5 Ncm	0,18 A
351-4580	motor	48	6,6 Ncm	0,55 A
351-4603	Getriebemotor	200	19,8 Ncm	0,55 A
351-4619	Getriebemotor	600	53,6 Ncm	0,55 A
351-4625	Getriebemotor	1200	100 Ncm*	0,55 A
351-4631	Getriebemotor	2400	100 Ncm *	0,55 A
351-4647	Getriebemotor	6000	100 Ncm*	0,55 A

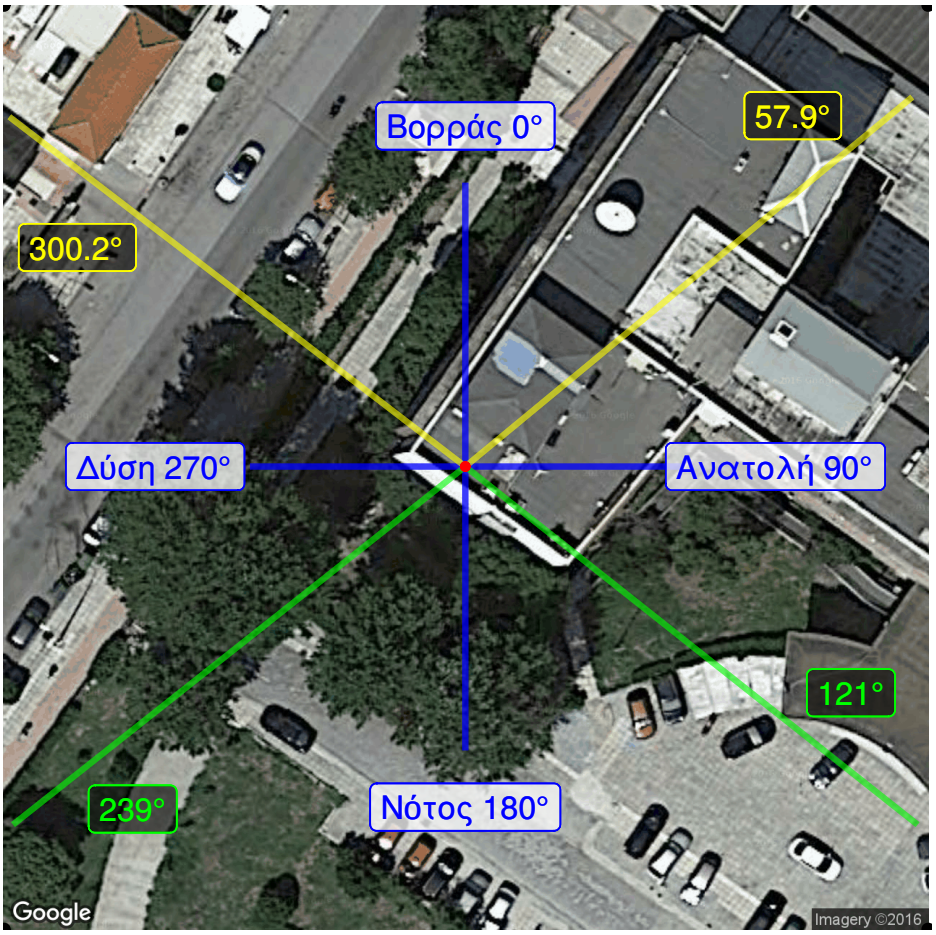
Hinweis*: Auf Getriebe begrenztes maximales Drehmoment.

3.3 Τροφοδοσία ισχύος

Το τροφοδοτικό του tracker παρέχει συνεχή διαφορά δυναμικού $19 - 20\ V$. Για την ώρα δεν γνωρίζουμε την δυνατότητα σε ισχύ της συσκευής ούτε και το μέγιστο ηλεκτρικό ρεύμα που μπορεί να χρησιμοποιήσει. Στην ηλεκτρονική πλακέτα, αναμένουμε περίπου $18\ V$ μετά από την πτώση τάσης στο καλώδιο τροφοδοσίας.

4 Εγκατάσταση του tracker

Είναι αναγκαίο το επίπεδο τις συσκευής να είναι απολύτως οριζόντιο, αλλά και να διατηρείται κατά τη λειτουργία του συστήματος tracker-οργάνου. Γι' αυτό η βάση, πρέπει να είναι αρκετά σταθερή αλλά και να έχει τη δυνατότητα λεπτομερούς ρύθμισης του επιπέδου. Στη συνέχεια θα περιγράψουμε την ρύθμιση του οργάνου για την παρακολούθηση της πορείας του Ήλιου.



Εικόνα 4.1: *Δορυφορική φωτογραφία της τοποθεσίας μετρήσεων (κόκκινο). Έχουν σχεδιαστεί οι κατευθύνσεις που αντιστοιχούν στα σημεία του ορίζοντα (μπλε), το αζιμούθιο του ήλιου κατά την ανατολή και τη δύση στις 21 Ιουλίου (κίτρινο) και αντίστοιχα στις 21 Δεκεμβρίου (πράσινο). Όλες οι γωνίες είναι μετρημένες με αρχή την κατεύθυνση του Βορρά.*

4.1 Ρύθμιση αζιμούθιου άξονα

Πρώτο βήμα, είναι η ευθυγράμμιση του μηδέν του αζιμούθιου της συσκευής με τον Βορρά. Αυτό μπορεί να γίνει με άμεση ευθυγράμμιση, αν η θέση του Βορρά είναι γνωστή (Εικόνα 4.1). Εναλλακτικά, μπορεί να γίνει όταν η συσκευή είναι ενεργή και ακολουθεί τον ήλιο. Τότε, η διόπτευση του ήλιου μπορεί να χρησιμοποιηθεί για να ευθυγραμμίσει το αζιμούθιο με τον Βορρά. Και στις δύο περιπτώσεις η συσκευή πρέπει να τοποθετηθεί στη βάση της και να μπορεί να περιστραφεί πριν σταθεροποιηθεί στην τελική της θέση. Περαιτέρω, βελτίωση

της θέσης του μηδενός του αζιμουθιακού άξονα, μπορεί να γίνει προγραμματιστικά, με τις μεταβλητές του offset στο πρόγραμμα που τις ελέγχει.

4.2 Ρύθμιση ζενίθιου άξονα

Για τον ζενίθιο άξονα, δεν θα χρειαστεί κάποια διαδικασία από τη στιγμή που η συσκευή είναι οριζοντιωμένη. Αν γνωρίζουμε τη θέση αναφοράς όπου μηδενίζει το όργανο, μπορούμε να υπολογίσουμε την θέση που ο άξονας θα βρίσκεται στο ζενίθ ή κάθετα σε αυτό (ανάλογα με τη βάση πρόσδεσης του οργάνου). Έτσι, μπορούμε να θέσουμε αυτήν την παράμετρο στο πρόγραμμα που ελέγχει τη συσκευή. Μία άλλη προσέγγιση, για να βρούμε το offset της στόχευσης, είναι η προσάρτηση του οργάνου να γίνει ενώ ο tracker ακολουθεί τον ήλιο, ώστε να ευθυγραμμιστεί με αυτόν. Αυτό προϋποθέτει ότι η στερέωση του οργάνου μπορεί να γίνει σε οποιαδήποτε γωνία σε σχέση με τον άξονα.

4.3 Οριζοντίωση του ηλιοστάτη (leveling)

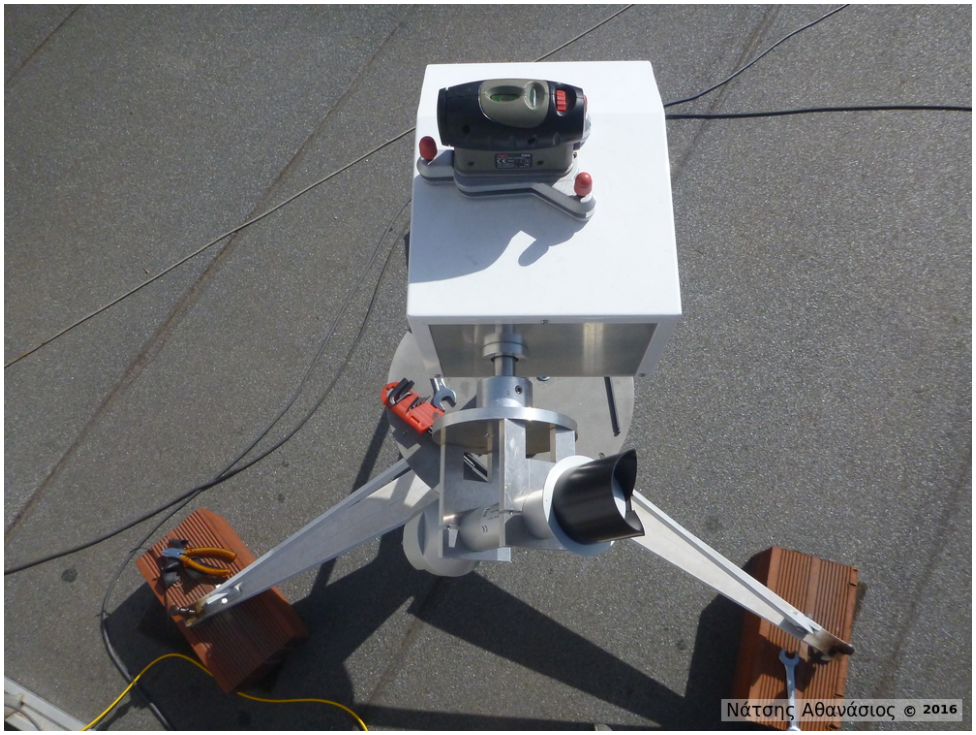
Τέλος, έχουμε την καθετοποίηση του οργάνου, ώστε ο άξονας του αζιμούθιου να είναι κατακόρυφος στη Γη. Μετά την τοποθέτηση των οργάνων μέτρησης, αλλά και περιοδικά, η καθετότητα του άξονα πρέπει να ελέγχεται. Καθώς, αποκλίσεις μπορούν να προκαλέσουν εσφαλμένες μετρήσεις κατά τη διάρκεια της ημέρας. Η ανάγκη αυτή, μπορεί να φανεί όταν μετά από διόπτευση (sighting) του Ήλιου, ο tracker χάνει γρήγορα την ευθυγράμμισή του.

Ξεκινάμε με τον tracker χωρίς τροφοδοσία, ώστε να μπορεί να περιστραφεί ελεύθερα γύρω από τον κατακόρυφο άξονα. Για τον έλεγχο της στάθμης χρησιμοποιούμε αλφάδι φυσαλίδας (αεροστάθμη) με δυνατότης ανεξάρτητης ρύθμισης του επιπέδου του σε σχέση με το όργανο, όπως φαίνεται στην Εικόνα 4.2.

Θα ρυθμίσουμε κάθε σημείο ελευθερίας του επιπέδου (εδώ, κάθε πόδι της βάσης) διαδοχικά, μέχρι το επίπεδο του οργάνου να συμφωνεί με το επίπεδο αναφοράς της στάθμης. Σε όλη τη διαδικασία δεν πρέπει να αλλάξουμε τη θέση του οργάνου της στάθμης πάνω στο όργανο.

Η διαδικασία για κάθε πόδι έχει ως εξής: Φέρνουμε τον άξονα της φυσαλίδας παράλληλο με την διεύθυνση του ποδιού. Αλλάζουμε το ύψος του και ρυθμίζουμε ξανά το όργανο στάθμης, προσπαθώντας να μοιράσουμε την μεταβολή ισόποσα μεταξύ των δύο. Περιστρέφουμε τον tracker κατά 180° και επαναλαμβάνουμε τη ρύθμιση και των δύο.

Το παραπάνω βήμα, το επαναλάβουμε μέχρι η στάθμη να είναι ικανοποιητικά επίπεδη. Με τον ίδιο τρόπο ρυθμίζουμε διαδοχικά και τα υπόλοιπα πόδια. Πιθανότατα, θα χρειαστεί παραπάνω από ένας κύκλος ρυθμίσεων και για το κάθε πόδι, αλλά και για τα τρία σε



Εικόνα 4.2: *Οριζοντίωση του tracker. Φαίνεται ο tracker εγκατεστημένος σε ρυθμιζόμενη βάση (τρίποδο) και η αεροστάθμη (αλφάδι) με την οποία ελέγχεται η οριζοντίωση.*

αλληλουχία. Το πλήθος τους, θα εξαρτηθεί από την ευαισθησία του μηχανισμού ρύθμισης, την ικανότητα του χειριστή αλλά και το επιθυμητό επίπεδο ακρίβειας.

Επαναλαμβάνουμε ότι είναι σημαντική η περιστροφή tracker μαζί με όργανο στάθμης κατά 180°, διότι πρέπει να ληφθεί υπόψη ότι το κάλυμμα του οργάνου δεν είναι κατ' ανάγκη παράλληλο με τη βάση του (π.χ. μπορεί να έχει παραμορφωθεί).

A CHP-1 Quick startup (on 'radmon' computer)

Σύνδεση του tracker.

Στο τροφοδοτικό του tracker συνδέονται τρία καλώδια. Συνήθως, η σύνδεση πετυχαίνει αν τα καλώδια συνδεθούν με τη σειρά από δεξιά προς τα αριστερά ή το αντίθετο. Η εκτέλεση του προγράμματος θα δείξει αν η επικοινωνία μπορεί να γίνει σωστά.

Reset tracker.

Όταν ο tracker τροφοδοτείται τα ηλεκτρονικά του είναι ενεργά, μερικές φορές, αυτό αποτρέπει την επανασύνδεση. Για να λυθεί αυτό ή πρέπει να μείνει για μερικά λεπτά χωρίς τροφοδοσία, ή να βραχυκυκλωθούν οι επαφές που τροφοδοτούν την ηλεκτρονική πλακέτα του tracker. **ΠΡΟΣΟΧΗ: εφαρμόστε λογική και αποσυνδέστε πρώτα την τροφοδοσία ρεύματος**. Αν το περίβλημα του tracker είναι ανοικτό, οι επαφές είναι προσβάσιμες. Εναλλακτικά, μπορούν να βραχυκυκλωθούν οι επαφές του καλωδίου με κάποιο μεταλλικό αντικείμενο και μόνο από τη μεριά του καλωδίου που συνδέεται στο τροφοδοτικό, ώστε να είστε σίγουροι για τη κατάσταση τροφοδοσίας.

Εκκίνηση του προγράμματος ελέγχου

Το πρόγραμμα ελέγχου του tracker (**sun_tracker_main.py**) μπορεί να ξεκινήσει:

- Χειροκίνητα (link στην επιφάνεια εργασίας)
- Μέσω του 'Scheduled Tasks' (properties -> run)
- Εκτέλεση από command line (cmd.exe):

```
c:\><python path>\python.exe <script path>\sun_tracker_main.py
```

Διόπτευση (sighting)

Το sighting του Ήλιου μπορεί να γίνει με το πρόγραμμα **tracker_sighting_no_ui_tcp-port.py** (υπάρχει link στην επιφάνεια εργασίας). Το πρόγραμμα εμφανίζει τις δυνατές επιλογές στην οθόνη και περιμένει είσοδο από το πληκτρολόγιο. Οι ρυθμίσεις επιδρούν άμεσα στη θέση του tracker (το πρόγραμμα **sun_tracker_main.py** πρέπει να εκτελείται και ο tracker πρέπει να έχει αρχίσει την κανονική λειτουργία). Ανάδραση (feedback) των ρυθμίσεων μπορούμε να έχουμε παρακολουθώντας το terminal που τρέχει το **sun_tracker_main.py**. **ΠΡΟΣΟΧΗ**. Οι ρυθμίσεις πρέπει να σωθούν (`'s'`) για να αποθηκευτούν στον δίσκο και να διατηρηθούν στην επόμενη εκτέλεση (ημέρα).

Σημείωση: Είναι βολικό το sighting να γίνεται με κάποιο smart phone μέσω remote desktop ('rdp'). Έτσι, μπορεί να υπάρχει άμεση επίβλεψη της κίνησης του tracker και των διορθώσεων και διευκολύνεται ο συγχρονισμός της κίνησης του tracker με τη θέση του ήλιου.

Σύνδεση πολυμέτρου 'Protek 506'

Το πολύμετρο μετράει την αντίσταση του pt100 ή του θερμίστορ (χρωματισμός καλωδίων στο manual του CHP) και η μέτρηση διαβάζεται μέσω 'RS-232'. Στο πολύμετρο πρέπει να

είναι ενεργές οι επιλογές '*KEEP ON*' και '*RS232*'. Αυτό γίνεται με τα πλήκτρα 'menu' και 'enter' για κάθε μία από τις ρυθμίσεις.

Εκκίνηση προγράμματος πολυμέτρου

Το πρόγραμμα (**protek506_measurments.py**) παίρνει μετρήσεις της αντίστασης - θερμοκρασίας του CHP1 και μπορεί να εκτελεστεί με όμοιο τρόπο με το πρόγραμμα του tracker. Καθώς το πρόγραμμα παίρνει μετρήσεις, στη οθόνη του πολυμέτρου πρέπει να αναβοσβήνουν εναλλάξ οι ενδείξεις '*RX*' και '*TX*'. Αν αυτό δεν γίνεται τότε η επικοινωνία έχει κολλήσει και το πολύμετρο πρέπει να κλείσει, να ξανανοίξει και να ρυθμιστεί (όπως παραπάνω).

Σε περίπτωση διακοπής ρεύματος.

Έχει παρατηρηθεί ότι ο tracker μπορεί να ξεκινήσει κανονικά όταν επανέλθει η τροφοδοσία και ξεκινήσει ο υπολογιστής. Αλλά αυτό δεν γίνεται πάντα.

Scheduled script description

Τα παρακάτω script εκτελούνται συνεχώς από το 'Task Scheduler' και σχετίζονται με τη καθημερινή λειτουργία του tracker και του CPH1, και είναι:

- **sun_tracker_main.py** (απαραίτητο, ελέγχει την κίνηση του tracker)
- **protek506_measurments.py** (σημαντικό, μετράει την θερμοκρασία του CPH1)
- **data_pub_win.py** (βοηθητικό, εκπέμπει δεδομένα λειτουργίας του tracker)

Σημείωση: Σε περίπτωση που κάποιο από αυτά ήδη εκτελείται δεν θα ξανατρέξει, διότι υπάρχει εσωτερικός έλεγχος στον κώδικα του κάθε προγράμματος.

Σημείωση: Πρέπει να είναι είναι ενεργή η επιλογή: `"Disable error reporting"` ώστε να μην κολλάει ο tracker περιμένοντας τον διάλογο του "Error reporting" όταν crashάρει. Πιθανόν και η επιλογή `"But notify me when critical error occurs"` (υπό δοκιμή για την ώρα). Για *Windows XP: Control Panel > System > Advanced > Error reporting*.

B Tracker microcontroler source code

```
#include    <htc.h>
__CONFIG(FOSC_INTOSC&WDTE_ON&PWRTE_OFF&MCLRE_OFF&CP_ON&CPD_ON&BOREN_ON&CLKOUTEN_OFF&IESO_OFF&FCMEN_OFF);
__CONFIG(WRT_ALL&PLLEN_OFF&STVREN_ON&LVP_OFF);//&BORV_25

#pragma warning disable 340

#define _XTAL_FREQ 16000000

#define BIT0        0b00000001
#define BIT1        0b00000010
#define BIT2        0b00000100
#define BIT3        0b00001000
#define BIT4        0b00010000
#define BIT5        0b00100000
#define BIT6        0b01000000
#define BIT7        0b10000000
#define BIT0_       0b11111110
#define BIT1_       0b11111101
```

```
#define BIT2_       0b11111011
#define BIT3_       0b11110111
#define BIT4_       0b11101111
#define BIT5_       0b11011111
#define BIT6_       0b10111111
#define BIT7_       0b01111111


#define PIN_A_A1_L              LATE1
#define PIN_A_A1_H              LATE2
#define PIN_A_A2_L              LATE0
#define PIN_A_A2_H              LATA5
#define PIN_A_B1_L              LATA1
#define PIN_A_B1_H              LATA4
#define PIN_A_B2_L              LATA2
#define PIN_A_B2_H              LATA3


#define PIN_Z_A1_L              LATD0
#define PIN_Z_A1_H              LATD1
#define PIN_Z_A2_L              LATC3
#define PIN_Z_A2_H              LATC2
#define PIN_Z_B1_L              LATC0
#define PIN_Z_B1_H              LATC1
#define PIN_Z_B2_L              LATA6
#define PIN_Z_B2_H              LATA7

#define PIN_F_A1_L              LATD3
#define PIN_F_A1_H              LATD2
#define PIN_F_A2_L              LATC4
#define PIN_F_A2_H              LATC5
#define PIN_F_B1_L              LATD5
#define PIN_F_B1_H              LATD4
#define PIN_F_B2_L              LATD6
#define PIN_F_B2_H              LATD7


#define PIN_A_OPTO              RB2
#define PIN_Z_OPTO              RB3
#define PIN_F_OPTO              RB4



#define OA_Mem                  10
#define OZ_Mem                  20
#define OF_Mem                  30

#define SpeedA_Mem              40
#define SpeedZ_Mem              41
#define SpeedF_Mem              42

#define DA_Mem                  50
#define DZ_Mem                  60
#define DF_Mem                  70

#define AZ_Mem                  90



const unsigned char StepMotorMatrix[5]={  // A1L A1H A2L A2H B1L B1H B2L B2H
0b00000000,// OFF
0b10010000,//
0b00001001,//
0b01100000,//
0b00000110 //
};



unsigned char TX_out;
unsigned char RX_in;
```

```
unsigned char Mem_Data;
unsigned char Mem_Address;

unsigned int  TxFSM;
unsigned char TxChar;
unsigned int  TxStringCounter;

unsigned int  RxFSM;
unsigned char RxChar;
unsigned int  RxCounter;

/*
unsigned int  ADC_tempResult;
unsigned char Counter_ADC_Sum;
unsigned int  I1_Sum;
unsigned int  I2_Sum;
unsigned int  I3_Sum;
unsigned int  I4_Sum;
unsigned int  I5_Sum;
unsigned int  I1_Sum_temp;
unsigned int  I2_Sum_temp;
unsigned int  I3_Sum_temp;
unsigned int  I4_Sum_temp;
unsigned int  I5_Sum_temp;
*/


unsigned char Counter100usec;
unsigned char Counter1msec;
unsigned char Counter10msec;
unsigned char Counter100msec;
unsigned char Counter1sec=50;

unsigned int  ITD_Input;
unsigned char ITD_Output[5];

unsigned char ADCFSM=0;



unsigned int  Timer_Init=20;


unsigned char MainFSM=0;
unsigned int  MainTimer=0;


unsigned char ReceivedByte;
unsigned char ReceiveFIFO[20];
unsigned char ReceiveValid;
unsigned char ReceiveCommand;
unsigned char ReceivedBytesCounter=0;

unsigned char TX_Mirror_Matrix[10];
unsigned int  TX_Mirror_Counter=0;

unsigned char TX_Matrix[81];
unsigned int  TX_Counter;
unsigned int  TX_Counter_Max;
unsigned char SendFlag;

unsigned char TXFSM;

unsigned char StepMotorStateA=0;
unsigned char StepMotorStateZ=0;
unsigned char StepMotorStateF=0;

unsigned int  AZ=5000;
unsigned int  ZE=5000;
unsigned int  FR=5000;

unsigned int  OA=0;
unsigned int  OZ=0;
```

```
unsigned int  OF=0;

unsigned int  Init_DA=0;
unsigned int  Init_DZ=0;
unsigned int  Init_DF=0;

unsigned int  AZtarget=100;
unsigned int  ZEtarget=100;
unsigned int  FRtarget=100;

unsigned int  AZterminal=100;
unsigned int  ZEterminal=100;
unsigned int  FRterminal=100;


unsigned int  AZtarget_temp=100;
unsigned int  ZEtarget_temp=100;
unsigned int  FRtarget_temp=100;

unsigned char SpeedA=10;
unsigned char SpeedZ=10;
unsigned char SpeedF=10;

unsigned char SpeedAnow=1;
unsigned char SpeedZnow=1;
unsigned char SpeedFnow=1;

unsigned char SpeedAtarget=1;
unsigned char SpeedZtarget=1;
unsigned char SpeedFtarget=1;

unsigned char SpeedAcalc=0;
unsigned char SpeedZcalc=0;
unsigned char SpeedFcalc=0;

unsigned int  Counter_Speed_A=0;
unsigned int  Counter_Speed_Z=0;
unsigned int  Counter_Speed_F=0;

unsigned int  ReceiveNumber=0;

unsigned int  AZ_Memory=0;

unsigned int  FSM=0;

unsigned int  FSM_A=0;
unsigned int  FSM_Z=0;
unsigned int  FSM_F=0;

unsigned int  Timer_FSM_A=0;
unsigned int  Timer_FSM_Z=0;
unsigned int  Timer_FSM_F=0;



unsigned int  A_OPTO_Shift=0b0101010101010101;
unsigned int  Z_OPTO_Shift=0b0101010101010101;
unsigned int  F_OPTO_Shift=0b0101010101010101;


unsigned int  tempint;
unsigned char tempB;




bit FLAG1msec=0;

bit Moving=0;
bit OKpending=0;

bit AZsync=0;
bit ZEsync=0;
```

```
bit FRsync=0;

bit SendAll=0;

bit SendAZ=0;
bit SendZE=0;
bit SendFR=0;

bit SendOA=0;
bit SendOZ=0;
bit SendOF=0;

bit SendDA=0;
bit SendDZ=0;
bit SendDF=0;

bit SendSA=0;
bit SendSZ=0;
bit SendSF=0;

bit SendTA=0;
bit SendTZ=0;
bit SendTF=0;

bit SendOK=0;
bit SendLF=0;

bit SendDebugOff=0;
bit SendDebugOn=0;

bit Debug=0;

bit GoToTargetA=0;
bit GoToTargetZ=0;
bit GoToTargetF=0;
bit GoToTargetAll=0;

bit TargetReachedA=0;
bit TargetReachedZ=0;
bit TargetReachedF=0;
bit TargetReachedAll=0;

bit GoToTerminalA=0;
bit GoToTerminalZ=0;
bit GoToTerminalF=0;
bit GoToTerminalAll=0;

bit TerminalReachedA=0;
bit TerminalReachedZ=0;
bit TerminalReachedF=0;
bit TerminalReachedAll=0;

bit StopA=0;
bit StopZ=0;
bit StopF=0;
bit StopAll=0;

bit CounterApulse=0;
bit CounterZpulse=0;
bit CounterFpulse=0;

bit PositionValid_A=0;
bit PositionValid_Z=0;
bit PositionValid_F=0;

bit A_OPTO=0;
bit Z_OPTO=0;
bit F_OPTO=0;

void SPI(void);
void MemRead(void);
void MemWrite(void);
void ReadParameters(void);
```

```
void WriteParameters(void);
void SMS_Number_Server(void);
void SMS_Number_Mobile(void);
void ReadUPSfromFlash(void);
void SaveUPStoFlash(void);
void ReadLSfromFlash(void);
void SaveLStoFlash(void);

void SCI_Receive(void);

void ReceiveByteRoutine(void);
void TX_Routine(void);

void TX_Mirror(unsigned char TXM_byte);


void My_EE_Write(unsigned char My_EE_Adr,unsigned char My_EE_Data);
unsigned char My_EE_Read(unsigned char My_EE_Adr);
void IntToDec(void);
void ReadKeys(void);
void ADC_Routine(void);

void SetStepMotorStates(void);

void StepUpA(void);
void StepDownA(void);
void StepUpZ(void);
void StepDownZ(void);
void StepUpF(void);
void StepDownF(void);

void MoveToTargetA(void);
void MoveToTargetZ(void);
void MoveToTargetF(void);

void DummyMoving(void);

void FSM_A_Routine(void);
void FSM_Z_Routine(void);
void FSM_F_Routine(void);

void ReadOPTO(void);

main()
{
unsigned char i;

CLRWDT();

IRCF3=1;
IRCF2=1;
IRCF1=1;
IRCF0=1;


PORTA =0b00000000;
TRISA =0b00000001;
ANSELA=0b00000000;

PORTB =0b00000000;
TRISB =0b11111100;
ANSELB=0b00100000;
WPUB  =0b11011100;

PORTC =0b00000000;
TRISC =0b11000000;

PORTD =0b00000000;
TRISD =0b00000000;
ANSELD=0b00000000;

PORTE =0b00000000;
TRISE =0b00001000;
```

```
ANSELE=0b00000000;
WPUE  =0b00001000;

nWPUEN=0;

GIE = 0;
//------------------------------------------------


//------------------------------------------------
FVRCON=0b10000011; // 4.096 V
ADCON1=0b10100011;
//------------------------------------------------

//------------------------------------------------
_delay(10000);

AZ=(My_EE_Read(AZ_Mem)<<8)+My_EE_Read(AZ_Mem+1);
if (AZ>9999) {AZ=100;}
AZtarget=AZ;

OA=(My_EE_Read(OA_Mem)<<8)+My_EE_Read(OA_Mem+1);
if (OA>9999) {OA=0;}
OZ=(My_EE_Read(OZ_Mem)<<8)+My_EE_Read(OZ_Mem+1);
if (OZ>9999) {OZ=0;}
OF=(My_EE_Read(OF_Mem)<<8)+My_EE_Read(OF_Mem+1);
if (OF>9999) {OF=0;}

Init_DA=(My_EE_Read(DA_Mem)<<8)+My_EE_Read(DA_Mem+1);
if (Init_DA>9999) {Init_DA=5000;}
Init_DZ=(My_EE_Read(DZ_Mem)<<8)+My_EE_Read(DZ_Mem+1);
if (Init_DZ>9999) {Init_DZ=5000;}
Init_DF=(My_EE_Read(DF_Mem)<<8)+My_EE_Read(DF_Mem+1);
if (Init_DF>9999) {Init_DF=5000;}


SpeedA=My_EE_Read(SpeedA_Mem);
if (SpeedA>99) {SpeedA=80;}
SpeedZ=My_EE_Read(SpeedZ_Mem);
if (SpeedZ>99) {SpeedZ=80;}
SpeedF=My_EE_Read(SpeedF_Mem);
if (SpeedF>99) {SpeedF=80;}


//------------------------------------------------
TxFSM=0;
RxFSM=0;


//--------------------------------------------------------------------------------------
//--------------------------------------------------------------------------------------
//--------------------------------------------------------------------------------------
//--------------------------------------------------------------------------------------
//--------------------------------------------------------------------------------------
//--------------------------------------------------------------------------------------

//--------------------------------------------------------------------------------------
//--------------------------------------------------------------------------------------
//--------------------------------------------------------------------------------------
//--------------------------------------------------------------------------------------
//--------------------------------------------------------------------------------------
//--------------------------------------------------------------------------------------


//------------------------------------------------
SPBRG=207;//4800
BAUDCON=0b00000000;
TXSTA=0b00000100;
RCSTA=0b00000000;
SPEN=1;
TXEN=1;
CREN=1;
tempB=RCSTA;
```

```
tempB=RCREG;
//-----------------------------------------------

//-----------------------------------------------
PSA=0;
PS2=0;
PS1=0;
PS0=0;
T0CS=0;
TMR0IE=1;
//-----------------------------------------------

GIE = 1;            // enable global interrupts



while (1)
    {
    CLRWDT();


    if (FLAG1msec) // ---------------1msec-------------
        {
        CLRWDT();
        FLAG1msec=0;

        ReadOPTO();

        if (RCIF) {ReceivedByte=RCREG;ReceiveByteRoutine();}

        if (OERR)
            {
            CREN=0;
            CREN=0;
            CREN=1;
            }

        FSM_A_Routine();
        FSM_Z_Routine();
        FSM_F_Routine();

        SetStepMotorStates();

        TX_Routine();


        if ( (AZ==AZtarget) && (ZE==ZEtarget) && (FR==FRtarget) ) {Moving=0;}

        if ( (Moving==0) && (OKpending==1) )
            {
            OKpending=0;
            SendOK=1;
            }
/*
        if ((A_OPTO)&&(AZtarget==0))
            {
            AZ=100;
            AZtarget=AZ;
            }
        if ((Z_OPTO)&&(ZEtarget==0))
            {
            ZE=100;
            ZEtarget=ZE;
            }
*/
//      SpeedAtarget=SpeedA;
//      SpeedZtarget=SpeedZ;
//      SpeedFtarget=SpeedF;
        if (Counter_Speed_A>=5000) {Counter_Speed_A=0;CounterApulse=1;}
        if (Counter_Speed_Z>=5000) {Counter_Speed_Z=0;CounterZpulse=1;}
        if (Counter_Speed_F>=5000) {Counter_Speed_F=0;CounterFpulse=1;}
```

```
        Counter1msec++;
        if (Counter1msec>9)
            {Counter1msec=0; // ----------10msec---------


            if (SpeedAtarget>SpeedAnow) {SpeedAnow++;}
            if (SpeedAtarget<SpeedAnow) {SpeedAnow--;}
            if (SpeedZtarget>SpeedZnow) {SpeedZnow++;}
            if (SpeedZtarget<SpeedZnow) {SpeedZnow--;}
            if (SpeedFtarget>SpeedFnow) {SpeedFnow++;}
            if (SpeedFtarget<SpeedFnow) {SpeedFnow--;}


            if (Timer_FSM_A) {Timer_FSM_A--;}
            if (Timer_FSM_Z) {Timer_FSM_Z--;}
            if (Timer_FSM_F) {Timer_FSM_F--;}


//______________________________________________________________________________________________
//______________________________________________________________________________________________
//______________________________________________________________________________________________
//          DummyMoving();
//______________________________________________________________________________________________
//______________________________________________________________________________________________
//______________________________________________________________________________________________


            Counter10msec++;
            if (Counter10msec>9)
                {Counter10msec=0; // ----------100msec---------



                Counter100msec++;
                if (Counter100msec>9)
                    {Counter100msec=0; // ----------1sec---------

                    AZ_Memory=(My_EE_Read(AZ_Mem)<<8)+My_EE_Read(AZ_Mem+1);
                    if (AZ!=AZ_Memory) {My_EE_Write(AZ_Mem,AZ>>8);My_EE_Write(AZ_Mem+1,AZ);}

                    Counter1sec++;
                    if (Counter1sec>59)
                        {Counter1sec=0; // ----------1min---------



                        }
                    }
                }
            }
        }
    }
}


/* service routine for timer 0 interrupt */
void interrupt
timer0_isr(void)
{

if (INTF) {INTF=0;}

    if (TMR0IF)  //  100 usec
        {
        TMR0IF = 0;
        TMR0=65;

        Counter100usec++;
        if (Counter100usec>9) {Counter100usec=0;FLAG1msec=1;}

//      ADC_Routine();
```

```
        Counter_Speed_A=Counter_Speed_A+SpeedAnow;
        Counter_Speed_Z=Counter_Speed_Z+SpeedZnow;
        Counter_Speed_F=Counter_Speed_F+SpeedFnow;

        }
}


void ReadOPTO(void)
{

A_OPTO_Shift<<=1;
A_OPTO_Shift=A_OPTO_Shift+PIN_A_OPTO;
if (A_OPTO_Shift==0xFFFF) {A_OPTO=1;}
if (A_OPTO_Shift==0x0000) {A_OPTO=0;}

Z_OPTO_Shift<<=1;
Z_OPTO_Shift=Z_OPTO_Shift+PIN_Z_OPTO;
if (Z_OPTO_Shift==0xFFFF) {Z_OPTO=1;}
if (Z_OPTO_Shift==0x0000) {Z_OPTO=0;}

F_OPTO_Shift<<=1;
F_OPTO_Shift=F_OPTO_Shift+PIN_F_OPTO;
if (F_OPTO_Shift==0xFFFF) {F_OPTO=1;}
if (F_OPTO_Shift==0x0000) {F_OPTO=0;}

}



void My_EE_Write(unsigned char My_EE_Adr,unsigned char My_EE_Data)
{
while(WR)continue;
EEADR=My_EE_Adr;
EEDAT=My_EE_Data;
GIE=0;
WREN=1;
EECON2=0x55;
EECON2=0xAA;
WR=1;
WREN=0;
GIE=1;
while(WR)continue;
}
unsigned char My_EE_Read(unsigned char My_EE_Adr)
{
EEADR=My_EE_Adr;
RD=1;
return EEDAT;

}

void IntToDec(void)
{
unsigned int ITDtemp;
ITDtemp=ITD_Input;
ITD_Output[0]=0;
ITD_Output[1]=0;
ITD_Output[2]=0;
ITD_Output[3]=0;
ITD_Output[4]=0;

while (ITDtemp>=10000)
    {
    ITD_Output[4]++;
    ITDtemp=ITDtemp-10000;
    }
while (ITDtemp>=1000)
    {
    ITD_Output[3]++;
    ITDtemp=ITDtemp-1000;
    }
while (ITDtemp>=100)
```

```
    {
    ITD_Output[2]++;
    ITDtemp=ITDtemp-100;
    }
while (ITDtemp>=10)
    {
    ITD_Output[1]++;
    ITDtemp=ITDtemp-10;
    }
ITD_Output[0]=ITDtemp;
}


/*

void ADC_Routine(void)
{
switch(ADCFSM)
    {
    case 0:
        I1_Sum_temp=0;
        I2_Sum_temp=0;
        I3_Sum_temp=0;
        I4_Sum_temp=0;
        I5_Sum_temp=0;
        Counter_ADC_Sum=0;
        ADCON0=0b00001101; // AN3
        ADCFSM=1;
    break;
    case 1:
        ADGO=1;
        ADCFSM=2;
    break;
    case 2:
        ADC_tempResult=0b0000001111111111&ADRES;
        I1_Sum_temp=I1_Sum_temp+ADC_tempResult;
        ADCON0=0b00010001; // AN4
        ADCFSM=3;
    break;
    case 3:
        ADGO=1;
        ADCFSM=4;
    break;
    case 4:
        ADC_tempResult=0b0000001111111111&ADRES;
        I2_Sum_temp=I2_Sum_temp+ADC_tempResult;
        ADCON0=0b00010101; // AN5
        ADCFSM=5;
    break;
    case 5:
        ADGO=1;
        ADCFSM=6;
    break;
    case 6:
        ADC_tempResult=0b0000001111111111&ADRES;
        I3_Sum_temp=I3_Sum_temp+ADC_tempResult;
        ADCON0=0b00011001; // AN6
        ADCFSM=7;
    break;
    case 7:
        ADGO=1;
        ADCFSM=8;
    break;
    case 8:
        ADC_tempResult=0b0000001111111111&ADRES;
        I4_Sum_temp=I4_Sum_temp+ADC_tempResult;
        ADCON0=0b00011101; // AN7
        ADCFSM=9;
    break;
    case 9:
        ADGO=1;
        ADCFSM=10;
    break;
```

```
    case 10:
        ADC_tempResult=0b0000001111111111&ADRES;
        I5_Sum_temp=I5_Sum_temp+ADC_tempResult;
        ADCON0=0b00001101; // AN3

        Counter_ADC_Sum++;

        if (Counter_ADC_Sum>=120)
            {
            I1_Sum=I1_Sum_temp;
            I2_Sum=I2_Sum_temp;
            I3_Sum=I3_Sum_temp;
            I4_Sum=I4_Sum_temp;
            I5_Sum=I5_Sum_temp;
            I1_Sum_temp=0;
            I2_Sum_temp=0;
            I3_Sum_temp=0;
            I4_Sum_temp=0;
            I5_Sum_temp=0;
            Counter_ADC_Sum=0;
            }
        ADCFSM=1;
    break;

    default:
        ADCFSM=0;
    break;
    }
}
*/

void SetStepMotorStates(void)
{

if (StepMotorMatrix[StepMotorStateA]&BIT7) {PIN_A_A1_L=1;} else {PIN_A_A1_L=0;}
if (StepMotorMatrix[StepMotorStateA]&BIT6) {PIN_A_A1_H=1;} else {PIN_A_A1_H=0;}
if (StepMotorMatrix[StepMotorStateA]&BIT5) {PIN_A_A2_L=1;} else {PIN_A_A2_L=0;}
if (StepMotorMatrix[StepMotorStateA]&BIT4) {PIN_A_A2_H=1;} else {PIN_A_A2_H=0;}
if (StepMotorMatrix[StepMotorStateA]&BIT3) {PIN_A_B1_L=1;} else {PIN_A_B1_L=0;}
if (StepMotorMatrix[StepMotorStateA]&BIT2) {PIN_A_B1_H=1;} else {PIN_A_B1_H=0;}
if (StepMotorMatrix[StepMotorStateA]&BIT1) {PIN_A_B2_L=1;} else {PIN_A_B2_L=0;}
if (StepMotorMatrix[StepMotorStateA]&BIT0) {PIN_A_B2_H=1;} else {PIN_A_B2_H=0;}

if (StepMotorMatrix[StepMotorStateZ]&BIT7) {PIN_Z_A1_L=1;} else {PIN_Z_A1_L=0;}
if (StepMotorMatrix[StepMotorStateZ]&BIT6) {PIN_Z_A1_H=1;} else {PIN_Z_A1_H=0;}
if (StepMotorMatrix[StepMotorStateZ]&BIT5) {PIN_Z_A2_L=1;} else {PIN_Z_A2_L=0;}
if (StepMotorMatrix[StepMotorStateZ]&BIT4) {PIN_Z_A2_H=1;} else {PIN_Z_A2_H=0;}
if (StepMotorMatrix[StepMotorStateZ]&BIT3) {PIN_Z_B1_L=1;} else {PIN_Z_B1_L=0;}
if (StepMotorMatrix[StepMotorStateZ]&BIT2) {PIN_Z_B1_H=1;} else {PIN_Z_B1_H=0;}
if (StepMotorMatrix[StepMotorStateZ]&BIT1) {PIN_Z_B2_L=1;} else {PIN_Z_B2_L=0;}
if (StepMotorMatrix[StepMotorStateZ]&BIT0) {PIN_Z_B2_H=1;} else {PIN_Z_B2_H=0;}

if (StepMotorMatrix[StepMotorStateF]&BIT7) {PIN_F_A1_L=1;} else {PIN_F_A1_L=0;}
if (StepMotorMatrix[StepMotorStateF]&BIT6) {PIN_F_A1_H=1;} else {PIN_F_A1_H=0;}
if (StepMotorMatrix[StepMotorStateF]&BIT5) {PIN_F_A2_L=1;} else {PIN_F_A2_L=0;}
if (StepMotorMatrix[StepMotorStateF]&BIT4) {PIN_F_A2_H=1;} else {PIN_F_A2_H=0;}
if (StepMotorMatrix[StepMotorStateF]&BIT3) {PIN_F_B1_L=1;} else {PIN_F_B1_L=0;}
if (StepMotorMatrix[StepMotorStateF]&BIT2) {PIN_F_B1_H=1;} else {PIN_F_B1_H=0;}
if (StepMotorMatrix[StepMotorStateF]&BIT1) {PIN_F_B2_L=1;} else {PIN_F_B2_L=0;}
if (StepMotorMatrix[StepMotorStateF]&BIT0) {PIN_F_B2_H=1;} else {PIN_F_B2_H=0;}

}
/*
void StepUpA(void)
{
if (StepMotorStateA<4) {StepMotorStateA++;} else {StepMotorStateA=1;}
if (AZ<9998) {AZ++;}
}
void StepDownA(void)
{
if (StepMotorStateA>1) {StepMotorStateA--;} else {StepMotorStateA=4;}
if (AZ>1) {AZ--;}
}
```

```
*/
void StepUpA(void)
{
if (StepMotorStateA<4) {StepMotorStateA++;} else {StepMotorStateA=1;}
if (AZ<9999) {AZ++;}
}
void StepDownA(void)
{
if (StepMotorStateA>1) {StepMotorStateA--;} else {StepMotorStateA=4;}
if (AZ>0) {AZ--;}
}


void StepUpZ(void)
{
if (StepMotorStateZ<4) {StepMotorStateZ++;} else {StepMotorStateZ=1;}
if (ZE<9999) {ZE++;}
}
void StepDownZ(void)
{
if (StepMotorStateZ>1) {StepMotorStateZ--;} else {StepMotorStateZ=4;}
if (ZE>0) {ZE--;}
}

void StepUpF(void)
{
if (StepMotorStateF<4) {StepMotorStateF++;} else {StepMotorStateF=1;}
if (FR<9999) {FR++;}
}
void StepDownF(void)
{
if (StepMotorStateF>1) {StepMotorStateF--;} else {StepMotorStateF=4;}
if (FR>0) {FR--;}
}


void DummyMoving(void)
{
if (AZ>AZtarget) {AZ--;}
if (AZ<AZtarget) {AZ++;}
if (ZE>ZEtarget) {ZE--;}
if (ZE<ZEtarget) {ZE++;}
if (FR>FRtarget) {FR--;}
if (FR<FRtarget) {FR++;}


}
void MoveToTargetA(void)
{
if (AZ>AZtarget) {StepDownA();}
if (AZ<AZtarget) {StepUpA();}
}
void MoveToTargetZ(void)
{
if (ZE>ZEtarget) {StepDownZ();}
if (ZE<ZEtarget) {StepUpZ();}
}
void MoveToTargetF(void)
{
if (FR>FRtarget) {StepDownF();}
if (FR<FRtarget) {StepUpF();}
}


void ReceiveByteRoutine(void)
{
//ReceiveValid=1;
ReceiveFIFO[19]=ReceiveFIFO[18];
ReceiveFIFO[18]=ReceiveFIFO[17];
ReceiveFIFO[17]=ReceiveFIFO[16];
ReceiveFIFO[16]=ReceiveFIFO[15];
ReceiveFIFO[15]=ReceiveFIFO[14];
ReceiveFIFO[14]=ReceiveFIFO[13];
```

```
ReceiveFIFO[13]=ReceiveFIFO[12];
ReceiveFIFO[12]=ReceiveFIFO[11];
ReceiveFIFO[11]=ReceiveFIFO[10];
ReceiveFIFO[10]=ReceiveFIFO[9];
ReceiveFIFO[9]=ReceiveFIFO[8];
ReceiveFIFO[8]=ReceiveFIFO[7];
ReceiveFIFO[7]=ReceiveFIFO[6];
ReceiveFIFO[6]=ReceiveFIFO[5];
ReceiveFIFO[5]=ReceiveFIFO[4];
ReceiveFIFO[4]=ReceiveFIFO[3];
ReceiveFIFO[3]=ReceiveFIFO[2];
ReceiveFIFO[2]=ReceiveFIFO[1];
ReceiveFIFO[1]=ReceiveFIFO[0];
ReceiveFIFO[0]=ReceivedByte;


//-----------------------------------------------------------------------------------------------------
//-----------------------------------------------------------------------------------------------------
//-----------------------------------------------------------------------------------------------------
//-----------------------------------------------------------------------------------------------------
//TXREG=ReceivedByte;
if (Debug) {TX_Mirror(ReceivedByte);}
//-----------------------------------------------------------------------------------------------------
//-----------------------------------------------------------------------------------------------------
//-----------------------------------------------------------------------------------------------------
//-----------------------------------------------------------------------------------------------------

if (ReceiveFIFO[0]==13)
        {
        SendLF=1;
        switch(ReceivedBytesCounter)
            {
            case 1:
                if (ReceiveFIFO[1]=='?')
                    {
                    SendAll=1;
                    }
            break;
            case 2:
                if ( (ReceiveFIFO[1]=='A') && (ReceiveFIFO[2]=='D') ) {GoToTerminalA=1;}
                if ( (ReceiveFIFO[1]=='Z') && (ReceiveFIFO[2]=='D') ) {GoToTerminalZ=1;}
                if ( (ReceiveFIFO[1]=='F') && (ReceiveFIFO[2]=='D') ) {GoToTerminalF=1;}

            break;
            case 3:

                if (ReceiveFIFO[1]=='?')
                    {
                    if ((ReceiveFIFO[2]=='Z')&&(ReceiveFIFO[3]=='A')) {SendAZ=1;}
                    if ((ReceiveFIFO[2]=='E')&&(ReceiveFIFO[3]=='Z')) {SendZE=1;}
                    if ((ReceiveFIFO[2]=='R')&&(ReceiveFIFO[3]=='F')) {SendFR=1;}

                    if ((ReceiveFIFO[2]=='A')&&(ReceiveFIFO[3]=='O')) {SendOA=1;}
                    if ((ReceiveFIFO[2]=='Z')&&(ReceiveFIFO[3]=='O')) {SendOZ=1;}
                    if ((ReceiveFIFO[2]=='F')&&(ReceiveFIFO[3]=='O')) {SendOF=1;}

                    if ((ReceiveFIFO[2]=='A')&&(ReceiveFIFO[3]=='I')) {SendDA=1;}
                    if ((ReceiveFIFO[2]=='Z')&&(ReceiveFIFO[3]=='I')) {SendDZ=1;}
                    if ((ReceiveFIFO[2]=='F')&&(ReceiveFIFO[3]=='I')) {SendDF=1;}

                    if ((ReceiveFIFO[2]=='A')&&(ReceiveFIFO[3]=='T')) {SendTA=1;}
                    if ((ReceiveFIFO[2]=='Z')&&(ReceiveFIFO[3]=='T')) {SendTZ=1;}
                    if ((ReceiveFIFO[2]=='F')&&(ReceiveFIFO[3]=='T')) {SendTF=1;}

                    if ((ReceiveFIFO[2]=='A')&&(ReceiveFIFO[3]=='S')) {SendSA=1;}
                    if ((ReceiveFIFO[2]=='Z')&&(ReceiveFIFO[3]=='S')) {SendSZ=1;}
                    if ((ReceiveFIFO[2]=='F')&&(ReceiveFIFO[3]=='S')) {SendSF=1;}

                    }

            break;
            case 4:
```

```
                if ( ( (ReceiveFIFO[1]=='P') && (ReceiveFIFO[2]=='O') && (ReceiveFIFO[3]=='T') && (ReceiveFIFO[4]=='S') )  || ( (ReceiveFIFO[1]=='p') && (ReceiveFIFO
                    {
//                  AZtarget=AZ;
//                  ZEtarget=ZE;
//                  FRtarget=FR;
                    StopA=1;
                    StopZ=1;
                    StopF=1;

                    OKpending=0;
                    SendLF=1;
                    }
            break;

            case 5:
                if ( ( (ReceiveFIFO[1]=='G') && (ReceiveFIFO[2]=='U') && (ReceiveFIFO[3]=='B') && (ReceiveFIFO[4]=='E') && (ReceiveFIFO[5]=='D') ) || ( (ReceiveFIFO[
                    {
                    if (Debug)
                            {
                            Debug=0;
                            SendDebugOff=1;
                            }
                        else
                            {
                            Debug=1;
                            SendDebugOn=1;
                            }
                    }

                if ( ( (ReceiveFIFO[1]>='0') && (ReceiveFIFO[2]>='0') ) && ( (ReceiveFIFO[1]<='9') && (ReceiveFIFO[2]<='9') ) )
                    {
                    ReceiveNumber=(ReceiveFIFO[1]-48)+(ReceiveFIFO[2]-48)*10;
                    if ( (ReceiveFIFO[3]=='=') && (ReceiveFIFO[4]=='A') && (ReceiveFIFO[5]=='S') ) {SpeedA=ReceiveNumber;My_EE_Write(SpeedA_Mem,SpeedA);}
                    if ( (ReceiveFIFO[3]=='=') && (ReceiveFIFO[4]=='Z') && (ReceiveFIFO[5]=='S') ) {SpeedZ=ReceiveNumber;My_EE_Write(SpeedZ_Mem,SpeedZ);}
                    if ( (ReceiveFIFO[3]=='=') && (ReceiveFIFO[4]=='F') && (ReceiveFIFO[5]=='S') ) {SpeedF=ReceiveNumber;My_EE_Write(SpeedF_Mem,SpeedF);}
                    SendLF=1;
                    }


            break;

            case 7:
                if ( ( (ReceiveFIFO[1]>='0') && (ReceiveFIFO[2]>='0') && (ReceiveFIFO[3]>='0') && (ReceiveFIFO[4]>='0') ) && ( (ReceiveFIFO[1]<='9') && (ReceiveFIFO[
                    {
                    ReceiveNumber=(ReceiveFIFO[1]-48)+(ReceiveFIFO[2]-48)*10+(ReceiveFIFO[3]-48)*100+(ReceiveFIFO[4]-48)*1000;
                    if ( (ReceiveFIFO[5]=='=') && (ReceiveFIFO[6]=='Z') && (ReceiveFIFO[7]=='A') ) {AZtarget=ReceiveNumber;OKpending=1;Moving=1;GoToTargetA=1;}
                    if ( (ReceiveFIFO[5]=='=') && (ReceiveFIFO[6]=='E') && (ReceiveFIFO[7]=='Z') ) {ZEtarget=ReceiveNumber;OKpending=1;Moving=1;GoToTargetZ=1;}
                    if ( (ReceiveFIFO[5]=='=') && (ReceiveFIFO[6]=='R') && (ReceiveFIFO[7]=='F') ) {FRtarget=ReceiveNumber;OKpending=1;Moving=1;GoToTargetF=1;}

                    if ( (ReceiveFIFO[5]=='+') && (ReceiveFIFO[6]=='Z') && (ReceiveFIFO[7]=='A') ) {AZtarget=AZ+ReceiveNumber;OKpending=1;Moving=1;GoToTargetA=1;}
                    if ( (ReceiveFIFO[5]=='+') && (ReceiveFIFO[6]=='E') && (ReceiveFIFO[7]=='Z') ) {ZEtarget=ZE+ReceiveNumber;OKpending=1;Moving=1;GoToTargetZ=1;}
                    if ( (ReceiveFIFO[5]=='+') && (ReceiveFIFO[6]=='R') && (ReceiveFIFO[7]=='F') ) {FRtarget=FR+ReceiveNumber;OKpending=1;Moving=1;GoToTargetF=1;}

                    if ( (ReceiveFIFO[5]=='-') && (ReceiveFIFO[6]=='Z') && (ReceiveFIFO[7]=='A') ) {AZtarget=AZ-ReceiveNumber;OKpending=1;Moving=1;GoToTargetA=1;}
                    if ( (ReceiveFIFO[5]=='-') && (ReceiveFIFO[6]=='E') && (ReceiveFIFO[7]=='Z') ) {ZEtarget=ZE-ReceiveNumber;OKpending=1;Moving=1;GoToTargetZ=1;}
                    if ( (ReceiveFIFO[5]=='-') && (ReceiveFIFO[6]=='R') && (ReceiveFIFO[7]=='F') ) {FRtarget=FR-ReceiveNumber;OKpending=1;Moving=1;GoToTargetF=1;}

                    if ( (ReceiveFIFO[5]=='=') && (ReceiveFIFO[6]=='A') && (ReceiveFIFO[7]=='O') ) {OA=ReceiveNumber;My_EE_Write(OA_Mem,OA>>8);My_EE_Write(OA_Mem+1,O
                    if ( (ReceiveFIFO[5]=='=') && (ReceiveFIFO[6]=='Z') && (ReceiveFIFO[7]=='O') ) {OZ=ReceiveNumber;My_EE_Write(OZ_Mem,OZ>>8);My_EE_Write(OZ_Mem+1,O
                    if ( (ReceiveFIFO[5]=='=') && (ReceiveFIFO[6]=='F') && (ReceiveFIFO[7]=='O') ) {OF=ReceiveNumber;My_EE_Write(OF_Mem,OF>>8);My_EE_Write(OF_Mem+1,O

                    if ( (ReceiveFIFO[5]=='=') && (ReceiveFIFO[6]=='A') && (ReceiveFIFO[7]=='I') ) {Init_DA=ReceiveNumber;My_EE_Write(DA_Mem,Init_DA>>8);My_EE_Write(
                    if ( (ReceiveFIFO[5]=='=') && (ReceiveFIFO[6]=='Z') && (ReceiveFIFO[7]=='I') ) {Init_DZ=ReceiveNumber;My_EE_Write(DZ_Mem,Init_DZ>>8);My_EE_Write(
                    if ( (ReceiveFIFO[5]=='=') && (ReceiveFIFO[6]=='F') && (ReceiveFIFO[7]=='I') ) {Init_DF=ReceiveNumber;My_EE_Write(DF_Mem,Init_DF>>8);My_EE_Write(


                    SendLF=1;
                    }
            break;
            case 12:
                if ( ( (ReceiveFIFO[1]>='0') && (ReceiveFIFO[2]>='0') && (ReceiveFIFO[3]>='0') && (ReceiveFIFO[4]>='0') ) && ( (ReceiveFIFO[1]<='9') && (ReceiveFIFO[
                    {
                    if ( ( (ReceiveFIFO[6]>='0') && (ReceiveFIFO[7]>='0') && (ReceiveFIFO[8]>='0') && (ReceiveFIFO[9]>='0') ) && ( (ReceiveFIFO[6]<='9') && (ReceiveF
```

```
                        {
                        if ( (ReceiveFIFO[5]==',') && (ReceiveFIFO[10]==' ') && (ReceiveFIFO[11]=='O') && (ReceiveFIFO[12]=='G') )
                            {
                            ReceiveNumber=(ReceiveFIFO[1]-48)+(ReceiveFIFO[2]-48)*10+(ReceiveFIFO[3]-48)*100+(ReceiveFIFO[4]-48)*1000;
                            ZEtarget=ReceiveNumber;
                            ReceiveNumber=(ReceiveFIFO[6]-48)+(ReceiveFIFO[7]-48)*10+(ReceiveFIFO[8]-48)*100+(ReceiveFIFO[9]-48)*1000;
                            AZtarget=ReceiveNumber;
                            OKpending=1;
                            Moving=1;
                            GoToTargetA=1;
                            GoToTargetZ=1;
                            }
                        }
                    }
            break;

            default:

            break;
            }



        ReceivedBytesCounter=0;
        }
    else
        {
        ReceivedBytesCounter++;
        }




}


void TX_Routine(void)
{
if (TXIF)
    {
    switch(TXFSM)
        {
        case 0:

        if (TX_Mirror_Counter)
            {
            TX_Mirror_Counter--;
            TXREG=TX_Mirror_Matrix[0];
            TX_Mirror_Matrix[0]=TX_Mirror_Matrix[1];
            TX_Mirror_Matrix[1]=TX_Mirror_Matrix[2];
            TX_Mirror_Matrix[2]=TX_Mirror_Matrix[3];
            TX_Mirror_Matrix[3]=TX_Mirror_Matrix[4];
            TX_Mirror_Matrix[4]=TX_Mirror_Matrix[5];
            TX_Mirror_Matrix[5]=TX_Mirror_Matrix[6];
            TX_Mirror_Matrix[6]=TX_Mirror_Matrix[7];
            TX_Mirror_Matrix[7]=TX_Mirror_Matrix[8];
            TX_Mirror_Matrix[8]=TX_Mirror_Matrix[9];
            break;
            }

        if (SendLF)
            {
            SendLF=0;
            if (Debug) {TXREG=10;}
            break;
            }
        if (SendAll)
            {
            SendAll=0;
            SendAZ=0;
            SendZE=0;
            SendFR=0;
            if (Debug) {TXREG=10;TXFSM=99;break;} else {TXFSM=100;}
```

```
    break;
    }
if (SendAZ)
    {
    SendAZ=0;
    if (Debug) {TXREG=10;}
    TXFSM=101;
    break;
    }
if (SendZE)
    {
    SendZE=0;
    if (Debug) {TXREG=10;}
    TXFSM=102;
    break;
    }
if (SendFR)
    {
    SendFR=0;
    if (Debug) {TXREG=10;}
    TXFSM=103;
    break;
    }
if (SendOA)
    {
    SendOA=0;
    if (Debug) {TXREG=10;}
    TXFSM=110;
    break;
    }
if (SendOZ)
    {
    SendOZ=0;
    if (Debug) {TXREG=10;}
    TXFSM=111;
    break;
    }
if (SendOF)
    {
    SendOF=0;
    if (Debug) {TXREG=10;}
    TXFSM=112;
    break;
    }
if (SendDA)
    {
    SendDA=0;
    if (Debug) {TXREG=10;}
    TXFSM=113;
    break;
    }
if (SendDZ)
    {
    SendDZ=0;
    if (Debug) {TXREG=10;}
    TXFSM=114;
    break;
    }
if (SendDF)
    {
    SendDF=0;
    if (Debug) {TXREG=10;}
    TXFSM=115;
    break;
    }

if (SendTA)
    {
    SendTA=0;
    if (Debug) {TXREG=10;}
    TXFSM=120;
    break;
```

```
            }
        if (SendTZ)
            {
            SendTZ=0;
            if (Debug) {TXREG=10;}
            TXFSM=121;
            break;
            }
        if (SendTF)
            {
            SendTF=0;
            if (Debug) {TXREG=10;}
            TXFSM=122;
            break;
            }

        if (SendSA)
            {
            SendSA=0;
            if (Debug) {TXREG=10;}
            TXFSM=130;
            break;
            }
        if (SendSZ)
            {
            SendSZ=0;
            if (Debug) {TXREG=10;}
            TXFSM=131;
            break;
            }
        if (SendSF)
            {
            SendSF=0;
            if (Debug) {TXREG=10;}
            TXFSM=132;
            break;
            }


        if (SendOK)
            {
            SendOK=0;
//          TXREG=10;
            TXFSM=190;
            break;
            }
        if (SendDebugOff)
            {
            SendDebugOff=0;
//          if (Debug) {TXREG=10;}
            TXREG=10;
            TXFSM=200;
            break;
            }
        if (SendDebugOn)
            {
            SendDebugOn=0;
//          if (Debug) {TXREG=10;}
//          TXREG=10;
            TXFSM=210;
            break;
            }

        if (TerminalReachedA)
            {
            TerminalReachedA=0;
            if (Debug) {TXREG=10;}
            TXFSM=140;
            break;
            }
        if (TerminalReachedZ)
            {
            TerminalReachedZ=0;
```

```
    if (Debug) {TXREG=10;}
    TXFSM=141;
    break;
    }
if (TerminalReachedF)
    {
    TerminalReachedF=0;
    if (Debug) {TXREG=10;}
    TXFSM=142;
    break;
    }

break;

case 99:
    TX_Counter=0;

    TX_Matrix[0]='A';
    TX_Matrix[1]='Z';
    TX_Matrix[2]=':';
    ITD_Input=AZ;
    IntToDec();
    TX_Matrix[3]=ITD_Output[3]+48;
    TX_Matrix[4]=ITD_Output[2]+48;
    TX_Matrix[5]=ITD_Output[1]+48;
    TX_Matrix[6]=ITD_Output[0]+48;
    TX_Matrix[7]=13;
    TX_Matrix[8]=10;

    TX_Matrix[9]='Z';
    TX_Matrix[10]='E';
    TX_Matrix[11]=':';
    ITD_Input=ZE;
    IntToDec();

    if (PositionValid_Z)
            {
            TX_Matrix[12]=ITD_Output[3]+48;
            TX_Matrix[13]=ITD_Output[2]+48;
            TX_Matrix[14]=ITD_Output[1]+48;
            TX_Matrix[15]=ITD_Output[0]+48;
            }
        else
            {
            TX_Matrix[12]='?';
            TX_Matrix[13]='?';
            TX_Matrix[14]='?';
            TX_Matrix[15]='?';
            }
    TX_Matrix[16]=13;
    TX_Matrix[17]=10;

    TX_Matrix[18]='F';
    TX_Matrix[19]='R';
    TX_Matrix[20]=':';
    ITD_Input=FR;
    IntToDec();
    if (PositionValid_F)
            {
            TX_Matrix[21]=ITD_Output[3]+48;
            TX_Matrix[22]=ITD_Output[2]+48;
            TX_Matrix[23]=ITD_Output[1]+48;
            TX_Matrix[24]=ITD_Output[0]+48;
            }
        else
            {
            TX_Matrix[21]='?';
            TX_Matrix[22]='?';
            TX_Matrix[23]='?';
            TX_Matrix[24]='?';
            }
    TX_Matrix[25]=13;
    TX_Matrix[26]=10;
```

```
            TX_Counter_Max=27;
            TXFSM=250;
        break;
        case 100:
            TX_Counter=0;

            TX_Matrix[0]='A';
            TX_Matrix[1]='Z';
            TX_Matrix[2]=':';
            ITD_Input=AZ;
            IntToDec();
            TX_Matrix[3]=ITD_Output[3]+48;
            TX_Matrix[4]=ITD_Output[2]+48;
            TX_Matrix[5]=ITD_Output[1]+48;
            TX_Matrix[6]=ITD_Output[0]+48;
            TX_Matrix[7]=13;
//          TX_Matrix[8]=10;

            TX_Matrix[8]='Z';
            TX_Matrix[9]='E';
            TX_Matrix[10]=':';
            ITD_Input=ZE;
            IntToDec();
            TX_Matrix[11]=ITD_Output[3]+48;
            TX_Matrix[12]=ITD_Output[2]+48;
            TX_Matrix[13]=ITD_Output[1]+48;
            TX_Matrix[14]=ITD_Output[0]+48;
            TX_Matrix[15]=13;
//          TX_Matrix[17]=10;

            TX_Matrix[16]='F';
            TX_Matrix[17]='R';
            TX_Matrix[18]=':';
            ITD_Input=FR;
            IntToDec();
            TX_Matrix[19]=ITD_Output[3]+48;
            TX_Matrix[20]=ITD_Output[2]+48;
            TX_Matrix[21]=ITD_Output[1]+48;
            TX_Matrix[22]=ITD_Output[0]+48;
            TX_Matrix[23]=13;
//          TX_Matrix[24]=10;

            TX_Counter_Max=24;
            TXFSM=250;
        break;
        case 101:
            TX_Counter=0;

            TX_Matrix[0]='A';
            TX_Matrix[1]='Z';
            TX_Matrix[2]=':';
            ITD_Input=AZ;
            IntToDec();
            TX_Matrix[3]=ITD_Output[3]+48;
            TX_Matrix[4]=ITD_Output[2]+48;
            TX_Matrix[5]=ITD_Output[1]+48;
            TX_Matrix[6]=ITD_Output[0]+48;
            TX_Matrix[7]=13;
//          TX_Matrix[8]=10;
            TX_Counter_Max=8;
            TXFSM=250;
        break;
        case 102:
            TX_Counter=0;

            TX_Matrix[0]='Z';
            TX_Matrix[1]='E';
            TX_Matrix[2]=':';
            ITD_Input=ZE;
            IntToDec();
            TX_Matrix[3]=ITD_Output[3]+48;
            TX_Matrix[4]=ITD_Output[2]+48;
```

```
            TX_Matrix[5]=ITD_Output[1]+48;
            TX_Matrix[6]=ITD_Output[0]+48;
            TX_Matrix[7]=13;
//          TX_Matrix[8]=10;
            TX_Counter_Max=8;
            TXFSM=250;
        break;
        case 103:
            TX_Counter=0;

            TX_Matrix[0]='F';
            TX_Matrix[1]='R';
            TX_Matrix[2]=':';
            ITD_Input=FR;
            IntToDec();
            TX_Matrix[3]=ITD_Output[3]+48;
            TX_Matrix[4]=ITD_Output[2]+48;
            TX_Matrix[5]=ITD_Output[1]+48;
            TX_Matrix[6]=ITD_Output[0]+48;
            TX_Matrix[7]=13;
//          TX_Matrix[8]=10;
            TX_Counter_Max=8;
            TXFSM=250;
        break;
        case 110:
            TX_Counter=0;

            TX_Matrix[0]='O';
            TX_Matrix[1]='A';
            TX_Matrix[2]=':';
            ITD_Input=OA;
            IntToDec();
            TX_Matrix[3]=ITD_Output[3]+48;
            TX_Matrix[4]=ITD_Output[2]+48;
            TX_Matrix[5]=ITD_Output[1]+48;
            TX_Matrix[6]=ITD_Output[0]+48;
            TX_Matrix[7]=13;
//          TX_Matrix[8]=10;
            TX_Counter_Max=8;
            TXFSM=250;
        break;
        case 111:
            TX_Counter=0;

            TX_Matrix[0]='O';
            TX_Matrix[1]='Z';
            TX_Matrix[2]=':';
            ITD_Input=OZ;
            IntToDec();
            TX_Matrix[3]=ITD_Output[3]+48;
            TX_Matrix[4]=ITD_Output[2]+48;
            TX_Matrix[5]=ITD_Output[1]+48;
            TX_Matrix[6]=ITD_Output[0]+48;
            TX_Matrix[7]=13;
//          TX_Matrix[8]=10;
            TX_Counter_Max=8;
            TXFSM=250;
        break;
        case 112:
            TX_Counter=0;

            TX_Matrix[0]='O';
            TX_Matrix[1]='F';
            TX_Matrix[2]=':';
            ITD_Input=OF;
            IntToDec();
            TX_Matrix[3]=ITD_Output[3]+48;
            TX_Matrix[4]=ITD_Output[2]+48;
            TX_Matrix[5]=ITD_Output[1]+48;
            TX_Matrix[6]=ITD_Output[0]+48;
            TX_Matrix[7]=13;
//          TX_Matrix[8]=10;
            TX_Counter_Max=8;
```

```
            TXFSM=250;
        break;
        case 113:
            TX_Counter=0;

            TX_Matrix[0]='I';
            TX_Matrix[1]='A';
            TX_Matrix[2]=':';
            ITD_Input=Init_DA;
            IntToDec();
            TX_Matrix[3]=ITD_Output[3]+48;
            TX_Matrix[4]=ITD_Output[2]+48;
            TX_Matrix[5]=ITD_Output[1]+48;
            TX_Matrix[6]=ITD_Output[0]+48;
            TX_Matrix[7]=13;
//          TX_Matrix[8]=10;
            TX_Counter_Max=8;
            TXFSM=250;
        break;
        case 114:
            TX_Counter=0;

            TX_Matrix[0]='I';
            TX_Matrix[1]='Z';
            TX_Matrix[2]=':';
            ITD_Input=Init_DZ;
            IntToDec();
            TX_Matrix[3]=ITD_Output[3]+48;
            TX_Matrix[4]=ITD_Output[2]+48;
            TX_Matrix[5]=ITD_Output[1]+48;
            TX_Matrix[6]=ITD_Output[0]+48;
            TX_Matrix[7]=13;
//          TX_Matrix[8]=10;
            TX_Counter_Max=8;
            TXFSM=250;
        break;
        case 115:
            TX_Counter=0;

            TX_Matrix[0]='I';
            TX_Matrix[1]='F';
            TX_Matrix[2]=':';
            ITD_Input=Init_DF;
            IntToDec();
            TX_Matrix[3]=ITD_Output[3]+48;
            TX_Matrix[4]=ITD_Output[2]+48;
            TX_Matrix[5]=ITD_Output[1]+48;
            TX_Matrix[6]=ITD_Output[0]+48;
            TX_Matrix[7]=13;
//          TX_Matrix[8]=10;
            TX_Counter_Max=8;
            TXFSM=250;
        break;
        case 120:
            TX_Counter=0;

            TX_Matrix[0]='T';
            TX_Matrix[1]='A';
            TX_Matrix[2]=':';
            ITD_Input=AZtarget;
            IntToDec();
            TX_Matrix[3]=ITD_Output[3]+48;
            TX_Matrix[4]=ITD_Output[2]+48;
            TX_Matrix[5]=ITD_Output[1]+48;
            TX_Matrix[6]=ITD_Output[0]+48;
            TX_Matrix[7]=13;
//          TX_Matrix[8]=10;
            TX_Counter_Max=8;
            TXFSM=250;
        break;
        case 121:
            TX_Counter=0;
```

```
            TX_Matrix[0]='T';
            TX_Matrix[1]='Z';
            TX_Matrix[2]=':';
            ITD_Input=ZEtarget;
            IntToDec();
            TX_Matrix[3]=ITD_Output[3]+48;
            TX_Matrix[4]=ITD_Output[2]+48;
            TX_Matrix[5]=ITD_Output[1]+48;
            TX_Matrix[6]=ITD_Output[0]+48;
            TX_Matrix[7]=13;
//          TX_Matrix[8]=10;
            TX_Counter_Max=8;
            TXFSM=250;
        break;
        case 122:
            TX_Counter=0;

            TX_Matrix[0]='T';
            TX_Matrix[1]='F';
            TX_Matrix[2]=':';
            ITD_Input=FRtarget;
            IntToDec();
            TX_Matrix[3]=ITD_Output[3]+48;
            TX_Matrix[4]=ITD_Output[2]+48;
            TX_Matrix[5]=ITD_Output[1]+48;
            TX_Matrix[6]=ITD_Output[0]+48;
            TX_Matrix[7]=13;
//          TX_Matrix[8]=10;
            TX_Counter_Max=8;
            TXFSM=250;
        break;

        case 130:
            TX_Counter=0;

            TX_Matrix[0]='S';
            TX_Matrix[1]='A';
            TX_Matrix[2]=':';
            ITD_Input=SpeedA;
            IntToDec();
            TX_Matrix[3]=ITD_Output[1]+48;
            TX_Matrix[4]=ITD_Output[0]+48;
            TX_Matrix[5]=13;
//          TX_Matrix[6]=10;
            TX_Counter_Max=6;
            TXFSM=250;
        break;
        case 131:
            TX_Counter=0;

            TX_Matrix[0]='S';
            TX_Matrix[1]='Z';
            TX_Matrix[2]=':';
            ITD_Input=SpeedZ;
            IntToDec();
            TX_Matrix[3]=ITD_Output[1]+48;
            TX_Matrix[4]=ITD_Output[0]+48;
            TX_Matrix[5]=13;
//          TX_Matrix[6]=10;
            TX_Counter_Max=6;
            TXFSM=250;
        break;
        case 132:
            TX_Counter=0;

            TX_Matrix[0]='S';
            TX_Matrix[1]='F';
            TX_Matrix[2]=':';
            ITD_Input=SpeedF;
            IntToDec();
            TX_Matrix[3]=ITD_Output[1]+48;
            TX_Matrix[4]=ITD_Output[0]+48;
            TX_Matrix[5]=13;
```

```
//          TX_Matrix[6]=10;
            TX_Counter_Max=6;
            TXFSM=250;
        break;

        case 140:
            TX_Counter=0;

            TX_Matrix[0]='e';
            TX_Matrix[1]='A';
            TX_Matrix[2]=':';
            ITD_Input=AZterminal;
            IntToDec();
            TX_Matrix[3]=ITD_Output[3]+48;
            TX_Matrix[4]=ITD_Output[2]+48;
            TX_Matrix[5]=ITD_Output[1]+48;
            TX_Matrix[6]=ITD_Output[0]+48;
            TX_Matrix[7]=13;
//          TX_Matrix[8]=10;
            TX_Counter_Max=8;
            TXFSM=250;
        break;
        case 141:
            TX_Counter=0;

            TX_Matrix[0]='e';
            TX_Matrix[1]='Z';
            TX_Matrix[2]=':';
            ITD_Input=ZEterminal;
            IntToDec();
            TX_Matrix[3]=ITD_Output[3]+48;
            TX_Matrix[4]=ITD_Output[2]+48;
            TX_Matrix[5]=ITD_Output[1]+48;
            TX_Matrix[6]=ITD_Output[0]+48;
            TX_Matrix[7]=13;
//          TX_Matrix[8]=10;
            TX_Counter_Max=8;
            TXFSM=250;
        break;
        case 142:
            TX_Counter=0;

            TX_Matrix[0]='e';
            TX_Matrix[1]='F';
            TX_Matrix[2]=':';
            ITD_Input=FRterminal;
            IntToDec();
            TX_Matrix[3]=ITD_Output[3]+48;
            TX_Matrix[4]=ITD_Output[2]+48;
            TX_Matrix[5]=ITD_Output[1]+48;
            TX_Matrix[6]=ITD_Output[0]+48;
            TX_Matrix[7]=13;
//          TX_Matrix[8]=10;
            TX_Counter_Max=8;
            TXFSM=250;
        break;

        case 190:
            TX_Counter=0;

            TX_Matrix[0]='O';
            TX_Matrix[1]='K';
            TX_Matrix[2]=13;
//          TX_Matrix[3]=10;

            TX_Counter_Max=3;
            TXFSM=250;
        break;

        case 200:
            TX_Counter=0;

            TX_Matrix[0]='D';
```

```
            TX_Matrix[1]='E';
            TX_Matrix[2]='B';
            TX_Matrix[3]='U';
            TX_Matrix[4]='G';
            TX_Matrix[5]=' ';
            TX_Matrix[6]='O';
            TX_Matrix[7]='F';
            TX_Matrix[8]='F';
            TX_Matrix[9]=13;
//          TX_Matrix[10]=10;

            TX_Counter_Max=10;
            TXFSM=250;
        break;
        case 210:
            TX_Counter=0;

            TX_Matrix[0]='D';
            TX_Matrix[1]='E';
            TX_Matrix[2]='B';
            TX_Matrix[3]='U';
            TX_Matrix[4]='G';
            TX_Matrix[5]=' ';
            TX_Matrix[6]='O';
            TX_Matrix[7]='N';
            TX_Matrix[8]=' ';
            TX_Matrix[9]=13;
//          TX_Matrix[10]=10;

            TX_Counter_Max=10;
            TXFSM=250;
        break;

        case 250:
            TXREG=TX_Matrix[TX_Counter];
            TX_Counter++;
            if (TX_Counter==TX_Counter_Max) {TXFSM=251;break;}
        break;
        case 251:
            if (Debug) {TXREG=10;}
            TXFSM=0;
        break;

        default:
            TXFSM=0;
        break;
        }
    }
}

void TX_Mirror(unsigned char TXM_byte)
{
if (TX_Mirror_Counter<10)
    {
    TX_Mirror_Matrix[TX_Mirror_Counter]=TXM_byte;
    TX_Mirror_Counter++;
    }
}


void FSM_A_Routine(void)
{
switch (FSM_A)
    {
    case 0:
        SpeedAnow=1;
        GoToTargetA=0;
        TargetReachedA=0;
        GoToTerminalA=0;
        TerminalReachedA=0;
        FSM_A=1;
    break;
```

```
case 1:
    if (GoToTerminalA)
        {
        GoToTerminalA=0;
        FSM_A=200;
        break;
        }
    if (GoToTargetA)
        {
        GoToTargetA=0;
        FSM_A=100;
        break;
        }
    if (StopA) {FSM_A=250;break;}
break;

case 100:
    SpeedAnow=10;
    SpeedAtarget=SpeedA;
    TargetReachedA=0;
    if (AZ>AZtarget) {FSM_A=110;break;}
    if (AZ<AZtarget) {FSM_A=120;break;}
    FSM_A=130;
break;
case 110:
    if (StopA) {FSM_A=250;break;}
    if (AZ==AZtarget) {FSM_A=130;break;}

    if (AZ<AZtarget)
        {
        FSM_A=100;
        break;
        }

    if (CounterApulse)
        {
        CounterApulse=0;
        StepDownA();
        }
    if (AZ-AZtarget<180)
            {
            SpeedAcalc=((AZ-AZtarget)>>1)+20;
            if (SpeedAcalc>SpeedA) {SpeedAcalc=SpeedA;}
            SpeedAtarget=SpeedAcalc;
            }
        else
            {
            SpeedAtarget=SpeedA;
            }
break;

case 120:
    if (StopA) {FSM_A=250;break;}
    if (AZ==AZtarget) {FSM_A=130;break;}

    if (AZ>AZtarget)
        {
        FSM_A=100;
        break;
        }

    if (CounterApulse)
        {
        CounterApulse=0;
        StepUpA();
        }
    if (AZtarget-AZ<180)
            {
            SpeedAcalc=((AZtarget-AZ)>>1)+20;
            if (SpeedAcalc>SpeedA) {SpeedAcalc=SpeedA;}
            SpeedAtarget=SpeedAcalc;
            }
        else
```

```
            {
            SpeedAtarget=SpeedA;
            }
break;

case 130:
    TargetReachedA=1;
    FSM_A=1;
break;

case 200:
    SpeedAnow=10;
    SpeedAtarget=SpeedA;
    TerminalReachedA=0;
    if (A_OPTO==0) {SpeedAnow=10;SpeedAtarget=SpeedA;FSM_A=202;break;}
    if (AZ<9799) {AZtarget_temp=AZ+100;} else {AZtarget_temp=9900;}
    FSM_A=201;
break;
case 201:
    if (StopA) {FSM_A=250;break;}
    if (AZ==AZtarget_temp) {SpeedAnow=10;SpeedAtarget=SpeedA;FSM_A=202;break;}

    if (CounterApulse)
        {
        CounterApulse=0;
        StepUpA();
        }
    if (AZtarget_temp-AZ<180)
            {
            SpeedAcalc=((AZtarget_temp-AZ)>>1)+20;
            if (SpeedAcalc>SpeedA) {SpeedAcalc=SpeedA;}
            SpeedAtarget=SpeedAcalc;
            }
        else
            {
            SpeedAtarget=SpeedA;
            }
break;
case 202:
    if (StopA) {FSM_A=250;break;}
    if (A_OPTO) {SpeedAnow=5;SpeedAtarget=10;AZtarget_temp=AZ+10;FSM_A=203;break;}

    if (CounterApulse)
        {
        CounterApulse=0;
        StepDownA();
        }
break;
case 203:
    if (StopA) {FSM_A=250;break;}
    if (AZ==AZtarget_temp) {SpeedAnow=2;SpeedAtarget=2;FSM_A=204;break;}

    if (CounterApulse)
        {
        CounterApulse=0;
        StepUpA();
        }
break;
case 204:
    if (StopA) {FSM_A=250;break;}
    if (A_OPTO) {FSM_A=205;break;}

    if (CounterApulse)
        {
        CounterApulse=0;
        StepDownA();
        }
break;
case 205:
    TerminalReachedA=1;
    AZterminal=AZ;
    AZ=Init_DA;
    FSM_A=1;
```

```
    break;

    case 250:
        StopA=0;
        TargetReachedA=1;
        FSM_A=1;
    break;

    default:
        FSM_A=0;
    break;
    }

}


void FSM_Z_Routine(void)
{
switch (FSM_Z)
    {
    case 0:
        SpeedZnow=1;
        GoToTargetZ=0;
        TargetReachedZ=0;
        GoToTerminalZ=0;
        TerminalReachedZ=0;
        FSM_Z=1;
    break;

    case 1:
        if (GoToTerminalZ)
            {
            GoToTerminalZ=0;
            FSM_Z=200;
            break;
            }
        if (GoToTargetZ)
            {
            GoToTargetZ=0;
            FSM_Z=100;
            break;
            }
        if (StopZ) {FSM_Z=250;break;}
    break;

    case 100:
        SpeedZnow=10;
        SpeedZtarget=SpeedZ;
        TargetReachedZ=0;
        if (ZE>ZEtarget) {FSM_Z=110;break;}
        if (ZE<ZEtarget) {FSM_Z=120;break;}
        FSM_Z=130;
    break;
    case 110:
        if (StopZ) {FSM_Z=250;break;}
        if (ZE==ZEtarget) {FSM_Z=130;break;}

        if (ZE<ZEtarget)
            {
            FSM_Z=100;
            break;
            }

        if (CounterZpulse)
            {
            CounterZpulse=0;
            StepDownZ();
            }
        if (ZE-ZEtarget<180)
                {
                SpeedZcalc=((ZE-ZEtarget)>>1)+20;
                if (SpeedZcalc>SpeedZ) {SpeedZcalc=SpeedZ;}
                SpeedZtarget=SpeedZcalc;
```

```
                }
            else
                {
                SpeedZtarget=SpeedZ;
                }
    break;

    case 120:
        if (StopZ) {FSM_Z=250;break;}
        if (ZE==ZEtarget) {FSM_Z=130;break;}

        if (ZE>ZEtarget)
            {
            FSM_Z=100;
            break;
            }

        if (CounterZpulse)
            {
            CounterZpulse=0;
            StepUpZ();
            }
        if (ZEtarget-ZE<180)
                {
                SpeedZcalc=((ZEtarget-ZE)>>1)+20;
                if (SpeedZcalc>SpeedZ) {SpeedZcalc=SpeedZ;}
                SpeedZtarget=SpeedZcalc;
                }
            else
                {
                SpeedZtarget=SpeedZ;
                }
    break;

    case 130:
        TargetReachedZ=1;
        FSM_Z=1;
    break;

    case 200:
        SpeedZnow=10;
        SpeedZtarget=SpeedZ;
        TerminalReachedZ=0;
        if (Z_OPTO==0) {SpeedZnow=10;SpeedZtarget=SpeedZ;FSM_Z=202;break;}
//      if (ZE<9799) {ZEtarget_temp=ZE+50;} else {ZEtarget_temp=9850;}
        if (ZE>200) {ZEtarget_temp=ZE-100;} else {ZEtarget_temp=100;}
        FSM_Z=201;
    break;
    case 201:
        if (StopZ) {FSM_Z=250;break;}
        if (ZE==ZEtarget_temp) {SpeedZnow=10;SpeedZtarget=SpeedZ;FSM_Z=202;break;}

        if (CounterZpulse)
            {
            CounterZpulse=0;
            StepDownZ();
            }
        if (ZE-ZEtarget_temp<180)
                {
                SpeedZcalc=((ZE-ZEtarget_temp)>>1)+20;
                if (SpeedZcalc>SpeedZ) {SpeedZcalc=SpeedZ;}
                SpeedZtarget=SpeedZcalc;
                }
            else
                {
                SpeedZtarget=SpeedZ;
                }
    break;
    case 202:
        if (StopZ) {FSM_Z=250;break;}
        if (Z_OPTO) {SpeedZnow=5;SpeedZtarget=10;ZEtarget_temp=ZE-10;FSM_Z=203;break;}

        if (CounterZpulse)
```

```
            {
            CounterZpulse=0;
            StepUpZ();
            }
    break;
    case 203:
        if (StopZ) {FSM_Z=250;break;}
        if (ZE==ZEtarget_temp) {SpeedZnow=2;SpeedZtarget=2;FSM_Z=204;break;}

        if (CounterZpulse)
            {
            CounterZpulse=0;
            StepDownZ();
            }
    break;
    case 204:
        if (StopZ) {FSM_Z=250;break;}
        if (Z_OPTO) {FSM_Z=205;break;}

        if (CounterZpulse)
            {
            CounterZpulse=0;
            StepUpZ();
            }
    break;
    case 205:
        TerminalReachedZ=1;
        ZEterminal=ZE;
        PositionValid_Z=1;
        ZE=Init_DZ;
        FSM_Z=1;
    break;

    case 250:
        StopZ=0;
        TargetReachedZ=1;
        FSM_Z=1;
    break;

    default:
        FSM_Z=0;
    break;
    }

}
void FSM_F_Routine(void)
{
switch (FSM_F)
    {
    case 0:
        SpeedFnow=1;
        GoToTargetF=0;
        TargetReachedF=0;
        GoToTerminalF=0;
        TerminalReachedF=0;
        FSM_F=1;
    break;

    case 1:
        if (GoToTerminalF)
            {
            GoToTerminalF=0;
            FSM_F=200;
            break;
            }
        if (GoToTargetF)
            {
            GoToTargetF=0;
            FSM_F=100;
            break;
            }
        if (StopF) {FSM_F=250;break;}
    break;
```

```
case 100:
    SpeedFnow=10;
    SpeedFtarget=SpeedF;
    TargetReachedF=0;
    if (FR>FRtarget) {FSM_F=110;break;}
    if (FR<FRtarget) {FSM_F=120;break;}
    FSM_F=130;
break;
case 110:
    if (StopF) {FSM_F=250;break;}
    if (FR==FRtarget) {FSM_F=130;break;}

    if (FR<FRtarget)
        {
        FSM_F=100;
        break;
        }

    if (CounterFpulse)
        {
        CounterFpulse=0;
        StepDownF();
        }
    if (FR-FRtarget<180)
            {
            SpeedFcalc=((FR-FRtarget)>>1)+20;
            if (SpeedFcalc>SpeedF) {SpeedFcalc=SpeedF;}
            SpeedFtarget=SpeedFcalc;
            }
        else
            {
            SpeedFtarget=SpeedF;
            }
break;

case 120:
    if (StopF) {FSM_F=250;break;}
    if (FR==FRtarget) {FSM_F=130;break;}

    if (FR>FRtarget)
        {
        FSM_F=100;
        break;
        }

    if (CounterFpulse)
        {
        CounterFpulse=0;
        StepUpF();
        }
    if (FRtarget-FR<180)
            {
            SpeedFcalc=((FRtarget-FR)>>1)+20;
            if (SpeedFcalc>SpeedF) {SpeedFcalc=SpeedF;}
            SpeedFtarget=SpeedFcalc;
            }
        else
            {
            SpeedFtarget=SpeedF;
            }
break;

case 130:
    TargetReachedF=1;
    FSM_F=1;
break;

case 200:
    SpeedFnow=10;
    SpeedFtarget=SpeedF;
    TerminalReachedF=0;
    if (FR<9799) {FRtarget_temp=FR+200;} else {FRtarget_temp=9998;}
```

```
        FSM_F=201;
    break;
    case 201:
        if (StopF) {FSM_F=250;break;}
        if (FR==FRtarget_temp) {SpeedFnow=10;SpeedFtarget=SpeedF;FSM_F=202;break;}

        if (CounterFpulse)
            {
            CounterFpulse=0;
            StepUpF();
            }
        if (FRtarget_temp-FR<180)
                {
                SpeedFcalc=((FRtarget_temp-FR)>>1)+20;
                if (SpeedFcalc>SpeedF) {SpeedFcalc=SpeedF;}
                SpeedFtarget=SpeedFcalc;
                }
            else
                {
                SpeedFtarget=SpeedF;
                }
    break;
    case 202:
        if (StopF) {FSM_F=250;break;}
        if (F_OPTO) {SpeedFnow=5;SpeedFtarget=10;FRtarget_temp=FR+10;FSM_F=203;break;}

        if (CounterFpulse)
            {
            CounterFpulse=0;
            StepDownF();
            }
    break;
    case 203:
        if (StopF) {FSM_F=250;break;}
        if (FR==FRtarget_temp) {SpeedFnow=2;SpeedFtarget=2;FSM_F=204;break;}

        if (CounterFpulse)
            {
            CounterFpulse=0;
            StepUpF();
            }
    break;
    case 204:
        if (StopF) {FSM_F=250;break;}
        if (F_OPTO) {FSM_F=205;break;}

        if (CounterFpulse)
            {
            CounterFpulse=0;
            StepDownF();
            }
    break;
    case 205:
        TerminalReachedF=1;
        FRterminal=FR;
        PositionValid_F=1;
        FR=Init_DF;
        FSM_F=1;
    break;

    case 250:
        StopF=0;
        TargetReachedF=1;
        FSM_F=1;
    break;

    default:
        FSM_F=0;
    break;
    }

}
```