Α.Ε.Ι. ΠΕΙΡΑΙΑ Τ.Τ.
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ Τ.Ε.

" ΧΡΗΣΗ PLC ΓΙΑ ΑΥΤΟΜΑΤΟΠΟΙΗΣΗ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΕΡΓΑΣΙΩΝ – ΕΝΔΕΙΚΤΙΚΑ ΠΑΡΑΔΕΙΓΜΑΤΑ"

Επιβλέπων Καθηγητής: Δρ. Καμινάρης Σταύρος, Αναπλ. Καθηγητής
Σπουδαστής: Νικολάοβιτς Ευθύμιος ΑΜ: 42185

ΑΙΓΑΛΕΩ

ΦΕΒΡΟΥΑΡΙΟΣ 2018

ΕΥΧΑΡΙΣΤΙΕΣ

Η περάτωση της παρούσης πτυχιακής εργασίας σηματοδοτεί το τέλος των σπουδών μου στο τμήμα Ηλεκτρολόγων Μηχανικών Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. (πρώην ΤΕΙ Πειραιά). Δράττομαι της ευκαιρίας να ευχαριστήσω τους συμφοιτητές και τους φίλους μου που μου στάθηκαν κατά τη διάρκεια των σπουδών μου αλλά και τους καθηγητές μου, οι οποίοι πέραν από τις τεχνικές γνώσεις που μου παρείχαν, με βοήθησαν να αναπτύξω τον τρόπο σκέψης μου. Ιδιαίτερα θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κ. Σταύρο Καμινάρη, με τον οποίο είχα άριστη συνεργασία και βοήθεια όποτε χρειαζόμουν το οτιδήποτε.

Τέλος, και πάνω από όλα θα ήθελα να ευχαριστήσω την οικογένειά μου για όλα όσα μου έχει προσφέρει αυτά τα χρόνια και για την ψυχολογική υποστήριξη που μου παρέχει.

ΠΕΡΙΕΧΟΜΕΝΑ

Περιεχόμενα

SUMMARY

This thesis constitutes an effort for the simplification of the programming process of a programmable logic controller (PLC).

In the introduction are given certain historical data, and an effort is made for a first look in the world of PLCs, without a report in their more specific characteristics.

The definition and the advantages of programmable logic controllers are existing in Chapter One. In the same chapter there is an extensive analysis of the PLC components, their mode of operation and their nomenclature, aiming to gradually understand them.

In the second chapter, a reference is made to the programmable logic controllers of "Siemens" and is analyzed the necessary equipment that needed to set up a PLC station.

The third chapter constitutes a continuity of the previous chapter, in which the structure of a program is developed using the software that provided by the company and the programming languages that used for writing it.

In the last chapter, using the knowledge that was acquired by the precedent ones we present the programs that was produced for the applications of the programmable logic controller against the classical automation.

Keywords: automation, PLC, LADDER, industrial applications

ΠΡΟΛΟΓΟΣ

Η παρούσα πτυχιακή εργασία έχει σκοπό την απλοποίηση της διαδικασίας προγραμματισμού ενός προγραμματιζόμενου λογικού ελεγκτή (PLC).

Στην εισαγωγή δίνονται κάποια εισαγωγικά ιστορικά στοιχεία, και γίνεται προσπάθεια για μία πρώτη είσοδο στον κόσμο των αυτοματισμών.

Ο ορισμός και τα πλεονεκτήματα των προγραμματιζόμενων λογικών ελεγκτών παρουσιάζονται στο πρώτο κεφάλαιο. Στο ίδιο κεφάλαιο γίνεται μία εκτεταμένη ανάλυση των δομικών στοιχείων των PLC, του τρόπου λειτουργίας αλλά και της ονοματολογίας τους, με σκοπό την σταδιακή κατανόηση τους.

Στο δεύτερο κεφάλαιο γίνεται μια αναφορά στους προγραμματιζόμενους λογικούς ελεγκτές της εταιρείας “Siemens” και αναλύεται ο απαιτούμενος εξοπλισμός που χρειάζεται για να δημιουργηθεί ένας σταθμός PLC.

Συνέχεια του προηγούμενου κεφαλαίου αποτελεί το τρίτο κεφάλαιο όπου αναπτύσσεται η δομή ενός προγράμματος με την χρήση του αντίστοιχου λογισμικού που προσφέρει η εταιρεία αλλά και οι γλώσσες προγραμματισμού που χρησιμοποιούνται για την σύνταξη ενός προγράμματος.

Στο τελευταίο κεφάλαιο, χρησιμοποιώντας τις γνώσεις που αποκτήθηκαν από τα προηγούμενα, παρουσιάζονται τα προγράμματα που παράχθηκαν για την εφαρμογή του προγραμματιζόμενου λογικού ελεγκτή έναντι του κλασσικού αυτοματισμού.

Λέξεις κλειδιά: αυτοματισμός, PLC, LADDER, βιομηχανικές εφαρμογές

1Ο ΚΕΦΑΛΑΙΟ

"Η ΕΞΕΛΙΞΗ ΤΩΝ ΑΥΤΟΜΑΤΙΣΜΩΝ ΚΑΙ ΟΙ ΠΡΟΓΡΑΜΜΑΤΙΖΟΜΕΝΟΙ ΛΟΓΙΚΟΙ ΕΛΕΓΚΤΕΣ"

1.1 Εισαγωγή

Ο Αυτοματισμός είναι μια παλιά, πολύ παλιά ιστορία. Και σε μεγάλο βαθμό είναι Ελληνική ιστορία. Η λέξη «αυτόματο» είναι Ελληνική και τη συναντάμε κατ' αρχάς στα Ομηρικά έπη.

Στην αρχαιότητα οι Έλληνες, αρχικά φαντάζονταν, οραματίζονταν αυτόματα συστήματα και στη συνέχεια οι Έλληνες Μηχανικοί της αρχαιότητας μελετούσαν, σχεδίαζαν και κατασκεύαζαν αυτόματα και επιπλέον έγραφαν γι αυτά.

Ιδιαίτερα, άνθηση γνώρισε η Τέχνη του Αυτοματισμού κατά την Ελληνιστική περίοδο. Στα γραπτά των μηχανικών της εποχής όπως του Κτησίβιου, τους Φίλωνος του Βυζάντιου και κυρίως του Ήρωνος του Αλεξανδρέως βασίστηκε η εξέλιξη του Αυτοματισμού για όλο το επόμενο διάστημα μέχρι την Αναγέννηση. Μετά τη Βιομηχανική Επανάσταση ο Αυτοματισμός άρχισε να εφαρμόζεται ευρέως στις παραγωγικές διαδικασίες. Ο ηλεκτρισμός έδωσε ώθηση στις δυνατότητες των αυτόματων συστημάτων και ήταν πλέον ένα όπλο στα χέρια των μηχανικών που μπορούσαν να υλοποιήσουν τη «λογική» του συστήματος με τις γνωστές διατάξεις του «κλασσικού αυτοματισμού».

Στη συνέχεια η ανάπτυξη της ηλεκτρονικής και ειδικά η ανακάλυψη των ημιαγωγών, κυριολεκτικά απογείωσε τις δυνατότητες και άνοιξε, μέχρι την εποχή μας, νέους ορίζοντες στο χώρο.

Τα PLCs χρησιμοποιήθηκαν για πρώτη φορά στις αρχές της δεκαετίας του '60, με σκοπό να αποτελέσουν μία πιο οικονομική, ευέλικτη και αξιόπιστη λύση, για τα πολύπλοκα συστήματα ελέγχου και προστασίας, που βασίζονταν μέχρι τότε σε ηλεκτρομηχανικούς ηλεκτρονόμους (relays). Τα μειονεκτήματα των συστημάτων αυτών ήταν πολλά. Τα relays, ως ηλεκτρομηχανικές συσκευές έχουν περιορισμένη διάρκεια ζωής, απαιτούν μεγάλη κατανάλωση ενέργειας για την λειτουργία τους και ευθύνονται για σημαντικό ηλεκτρικό θόρυβο. Κατά δεύτερο λόγο, η εύρεση και η διόρθωση βλαβών σε συστήματα με πολλά

relays, είναι μια επίπονη διαδικασία. Οι μικρές διαστάσεις των PLCs, η ταχύτητα και η ευκολία στον προγραμματισμό τους, αλλά και η μεγάλη διάρκεια ζωής τους, αποτέλεσαν τις αιτίες για να διαδοθούν ταχύτατα και να παράγονται από μεγάλο αριθμό εταιρειών. Το βασικό τους πλεονέκτημα είναι πως οι οποιεσδήποτε αλλαγές στον τρόπο ελέγχου, γίνονται αλλάζοντας μόνο το πρόγραμμα στη μνήμη του προγραμματιζόμενου ελεγκτή, χωρίς να χρειαστεί να αλλάξει η συνδεσμολογία του, όπως συμβαίνει στον έλεγχο με ηλεκτρονόμους, όπου οποιαδήποτε αλλαγή είναι πολύ δύσκολη και χρονοβόρα.

Παρ' όλο όμως που οι πρώτοι απλοί ηλεκτρονικοί λογικοί ελεγκτές εμφανίσθηκαν πριν από 50 χρόνια, η επανάσταση έγινε το 1984. Τη χρονιά αυτή παρουσιάσθηκε hardware «με ευφυΐα» και δυνατότητα προγραμματισμού με χρήση γλωσσών υψηλού επιπέδου.

Οι ελεγκτές προγραμματιζόμενης λογικής εξακολουθούν σήμερα να είναι το βασικό σύστημα σε κάθε εξελιγμένη λύση αυτοματισμού. Τα πιο σημαντικά πλεονεκτήματα των PLC που είναι, η αντοχή, η στιβαρή κατασκευή και η απλότητα στο χειρισμό εξακολουθούν να αποτελούν την εγγύηση για την επιτυχημένη χρήση τους στις εφαρμογές.

Τα PLC μαζί με μια σειρά περιφερειακών, συνεργαζόμενων και παρελκόμενων συστημάτων και προϊόντων δίνουν την ευκαιρία στους σημερινούς μηχανικούς να σχεδιάζουν και να υλοποιούν ολοκληρωμένες εφαρμογές αυτοματισμού, γρήγορα και εύκολα αλλά και στους τελικούς πελάτες να διαθέτουν συστήματα φιλικά, ανοιχτά σε συνεργασία με υφιστάμενα συστήματα, εύκολα στη διάγνωση και αποκατάσταση βλαβών, τη συντήρηση αλλά και τις μελλοντικές επεκτάσεις.

1.2 Τι είναι ο προγραμματιζόμενος λογικός ελεγκτής (PLC) και τα πλεονεκτήματά του

Το PLC είναι μία ηλεκτρονική διάταξη η οποία από την άποψη της λειτουργίας θα μπορούσε να προσομοιωθεί με έναν πίνακα αυτοματισμού. Έχει δηλαδή εισόδους και εξόδους που συνδέονται με τα στοιχεία μιας εγκατάστασης και βέβαια έναν αλγόριθμο που καθορίζει ότι κάποιος συνδυασμός εισόδων παράγει ένα αποτέλεσμα στις εξόδους (π.χ. η ενεργοποίηση ενός τερματικού διακόπτη σταματά τον κινητήρα μιας μεταφορικής ταινίας). Οι ομοιότητες όμως σταματούν εδώ, αφού το ιδιαίτερο χαρακτηριστικό των PLC είναι ότι οι "κανόνες" πού καθορίζουν την συμπεριφορά των εξόδων δεν είναι σταθεροί και "συρματωμένοι" όπως σε ένα κλασσικό πίνακα αυτοματισμού αλλά μπορούν να μεταβάλλονται με την επέμβαση στο πρόγραμμα του PLC χωρίς καμία επέμβαση στο Hardware του συστήματος. Δηλαδή η λογική της λειτουργίας πού ενσωματώνεται στο PLC μέσω του προγραμματισμού του είναι μεταβαλλόμενη.

Έτσι σε ότι αφορά το υλικό όλα τα PLC αποτελούνται από την CPU, η οποία περιέχει την λογική του αυτοματισμού και η οποία αφού διαβάσει την κατάσταση των καρτών εισόδου (input modules) ενεργοποιεί τις κάρτες εξόδου (output modules) σύμφωνα με τους κανόνες (πρόγραμμα) πού έχουμε αποθηκεύσει στην μνήμη του. Βέβαια το σύστημα συμπληρώνεται από το τροφοδοτικό και πιθανόν από διατάξεις ενδείξεων και χειρισμών (operator panel, operator display). Η CPU με την βοήθεια της κάρτας εισόδου γνωρίζει κάθε στιγμή την κατάσταση ενός διακόπτη εάν δηλαδή είναι διεγερμένος η όχι. Επιπλέον με τη βοήθεια της κάρτας εξόδου οπλίζει ένα ρελέ και μέσω αυτού ενεργοποιεί μία διάταξη κίνησης, φωτισμού κλπ. Αυτό που απομένει είναι η "λογική" δηλαδή πότε πρέπει να οπλίσει το ρελέ. Αυτή η λογική είναι το πρόγραμμα του PLC πού συντάσσεται σε συγκεκριμένη γλώσσα με την βοήθεια ειδικού λογισμικού (programing software), και αποθηκεύεται στην μνήμη του PLC.

Πλεονεκτήματα των PLC

- Είναι συσκευές γενικής χρήσεως δεν είναι κατασκευασμένοι για ένα συγκεκριμένο είδος παραγωγής.
- Δεν ενδιαφέρει ο συνολικός αριθμός των επαφών, χρονικών, απαριθμητών κλπ. που θα χρησιμοποιηθούν μιας και αποτελούν στοιχεία μνήμης της CPU και όχι φυσικές οντότητες.
- Η λειτουργία του αυτοματισμού μπορεί ν' αλλάξει σε οποιοδήποτε στάδιο θελήσουμε (μελέτη, κατασκευή, θέση σε λειτουργία ή αργότερα) χωρίς επέμβαση στο υλικό.
- Εύκολος οπτικός εντοπισμός με μία ματιά, της λειτουργίας ή μη στοιχείων της εγκατάστασης με τη βοήθεια των LED που υπάρχουν σε όλες τις κάρτες εισόδου / εξόδου. Με τη βοήθεια συσκευής προγραμματισμού μπορεί να παρακολουθηθεί και η ροή εκτέλεσης του προγράμματος.
- Η κατασκευή του πίνακα που θα τοποθετηθεί το PLC γίνεται παράλληλα με τον προγραμματισμό του, πράγμα το οποίο οδηγεί στη συντομότερη παράδοση του αυτοματισμού.
- Πολύ συχνό είναι το φαινόμενο ο τεχνικός να κληθεί να επισκευάσει μια βλάβη και να δει έκπληκτος ότι άλλα υπάρχουν στα σχέδια και άλλα βλέπει αυτός στην εγκατάσταση. Το πρόβλημα αυτό δεν υπάρχει στα PLC αφού πάντα υπάρχει μόνο ένα "σχέδιο" αποθηκευμένο - το τελευταίο πρόγραμμα που του έχουμε περάσει. Εάν θέλουμε να έχουμε περισσότερα προγράμματα, αυτό είναι δυνατό με τη χρήση δισκετών.
- Τα PLC ως ηλεκτρονικές συσκευές καταλαμβάνουν πολύ μικρότερο χώρο στο πίνακα σε σχέση με τα υλικά του κλασσικού αυτοματισμού, καταναλώνουν δε πολύ λιγότερη ενέργεια από αυτά.
- Τοποθετούνται άφοβα και σε πεδία ισχύος ο κατασκευαστής δίνει οδηγίες γι' αυτές τις περιπτώσεις οι οποίες πρέπει να τηρούνται (αποστάσεις, γειώσεις κλπ).
- Οι γλώσσες προγραμματισμού καλύπτουν όλο το φάσμα των ανθρώπων που καλούνται να ασχοληθούν με την τεχνολογία αυτή - Υπάρχει γλώσσα για ανθρώπους με γνώση στο συμβατικό αυτοματισμό (Ladder), γλώσσες για όσους έχουν υπόβαθρο σε υπολογιστές (Statement List, SCL, FBD,

C++) καθώς και γλώσσες εξειδικευμένες για διάφορες τεχνολογίες (GRAPH 7, HIGRAPH, CSF).

- Τέλος, σαν ψηφιακές συσκευές σήμερα πια μας δίνουν τη δυνατότητα να συνδέσουμε επάνω τους οθόνες, εκτυπωτές, πληκτρολόγια και να καταργήσουμε έτσι τα κλασσικά μιμικά διαγράμματα και τους πίνακες χειρισμών. Εύκολη είναι επίσης και η διασύνδεση μεταξύ τους για ανταλλαγή πληροφοριών, ο τηλεχειρισμός και η τηλεποπτεία, ο εξ' αποστάσεως προγραμματισμός τους και η σύνδεσή τους στο Internet.

Συνοπτικά:

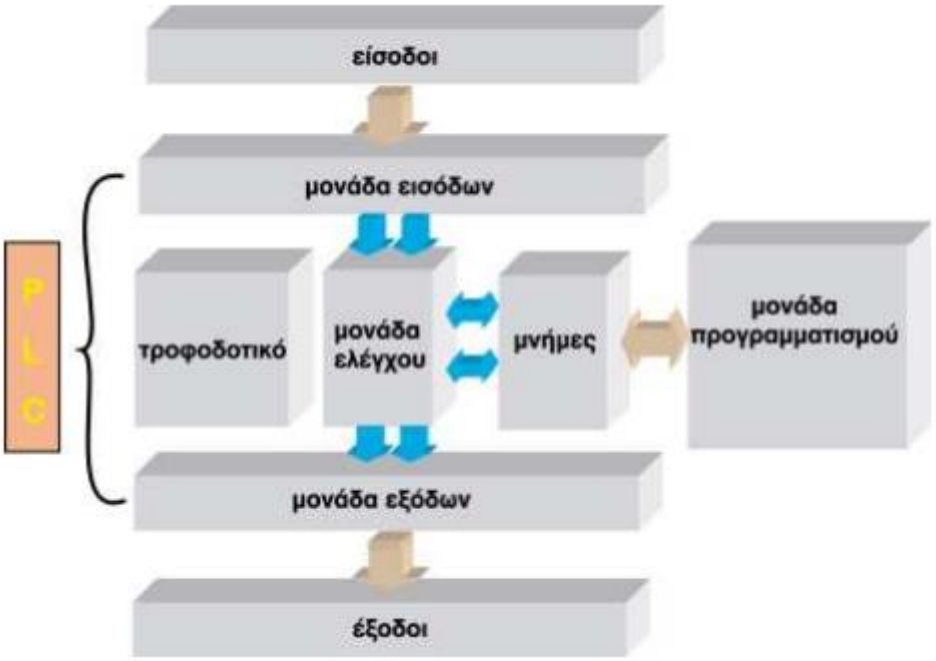
1. Μικρό κόστος
2. Δοκιμαστικός έλεγχος
3. Ταχύτητα λειτουργίας
4. Παρακολούθηση κυκλωμάτων
5. Εγκατάσταση σε πεδία ισχύος
6. Αποθήκευση προγράμματος στο plc
7. Δυνατότητα σύνδεσης με οθόνες , πληκτρολόγια ,εκτυπωτές

1.3 Δομικά στοιχεία των PLC

Για την καλύτερη κατανόηση του PLC και των λειτουργιών του είναι απαραίτητη η ανάλυση της δομής του, των στοιχείων δηλαδή, που το αποτελούν, ειδικά εφ' όσον η οποιαδήποτε επιλογή ενός PLC εξαρτάται κυρίως από το πλήθος των εισόδων, των εξόδων, αλλά και των λειτουργιών που είναι απαραίτητο να πραγματοποιήσει ο αυτοματισμός και αφορούν κατά κύριο λόγο την μνήμη όπως και τις δυνατότητες της κεντρικής μονάδας.

Ένας τυπικός προγραμματιζόμενος λογικός ελεγκτής μπορεί να διαιρεθεί σε κομμάτια, όπως φαίνεται και στο σχήμα 1.3. Αυτά τα τμήματα είναι:

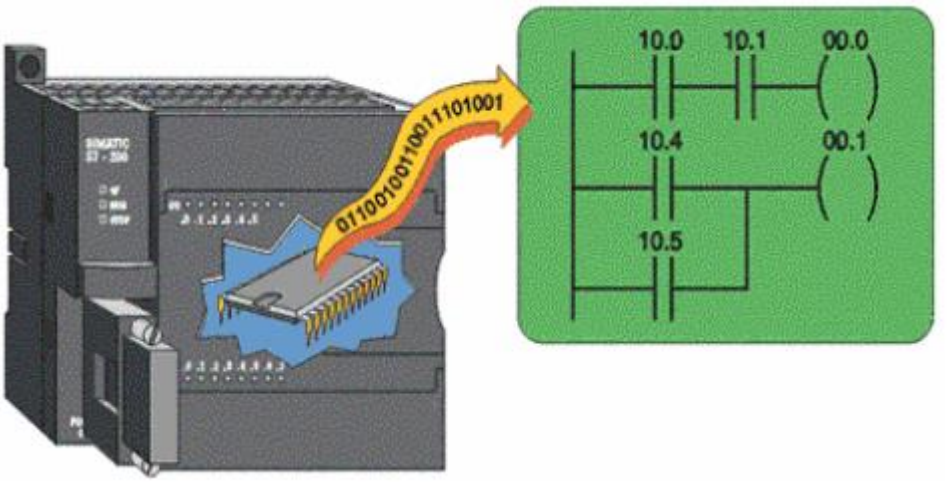
- Η κεντρική μονάδα επεξεργασίας (Central Processing Unit, CPU)
- Η μονάδα τροφοδοσίας
- Μνήμες
- Οι μονάδες εισόδων/εξόδων (Input/Output modules)
- Μονάδα προγραμματισμού



Σχήμα 1.3.1 *Δομή ενός PLC*

1.3.1 Κεντρική μοναδα επεξεργασίας (CPU)

Η Κεντρική Μονάδα Επεξεργασίας (Central Processing Unit – CPU) του PLC είναι ένα ψηφιακό κύκλωμα, ένας μικροεπεξεργαστής συγκεκριμένα (micropro- cessor) που αποτελεί τον "εγκέφαλο" του PLC.



Σχήμα 1.3.2 *Κεντρική Μονάδα Επεξεργασίας*

Πρόκειται για το μέρος του PLC που υλοποιεί τη λογική και παίρνει τις αποφάσεις με βάση τις εντολές του προγράμματος και την κατάσταση των εισόδων και των εξόδων που συνεχώς επιτηρεί. Στη CPU υλοποιούνται λειτουργίες αντίστοιχες με τους συνδυασμούς επαφών στα συμβατικά κυκλώματα απαριθμήσεις, χρονομετρήσεις, συγκρίσεις δεδομένων, μαθηματικές πράξεις και άλλες λειτουργίες.

1.3.2 Μονάδα τροφοδοσίας

Η χρησιμότητα της μονάδας τροφοδοσίας έγκειται στην δημιουργία των απαραίτητων για τη λειτουργία της εγκατάστασης εσωτερικών τάσεων από το δίκτυο. Οι εσωτερικές αυτές τάσεις, τροφοδοτούν αποκλειστικά τα ηλεκτρικά στοιχεία που υπάρχουν μέσα στον ελεγκτή, όπως είναι τα τρανζίστορ. Η μονάδα τροφοδοσίας επιπλέον, είναι αυτή που βοηθάει στην διατήρηση του περιεχομένου της μνήμης RAM σε μία διακοπή ρεύματος, μαζί με μία ενσωματωμένη μπαταρία.

Μία μονάδα τροφοδοσίας αποτελείται από κάποια τεχνικά χαρακτηριστικά τα πιο σημαντικά εκ των οποίων παρατίθενται παρακάτω:

Είσοδος: Ονομαστική τάση, συχνότητα, προστασία, ανοχές τάσης, απορροφούμενο ρεύμα.
Έξοδος: Ονομαστική τάση, ονομαστικό ρεύμα, προστασία βραχυκυκλώματος.

1.3.3 Μνήμες

Η μνήμη μιας CPU χωρίζεται σε 3 γενικές κατηγορίες :

- Μνήμη φόρτωσης,
- Μνήμη εργασίας
- Μνήμη συστήματος

1.3.3.1 Μνήμη φόρτωσης (Load Memory)

Στη μνήμη αυτή αποθηκεύεται ολόκληρο το πρόγραμμα του χρήστη, η περιγραφή του συστήματος καθώς και οι παράμετροι των καρτών. Η μνήμη αυτή προϋπάρχει σε κάθε CPU και μπορεί να επεκταθεί με εξωτερικές μνήμες RAM ή Flash EPROM (FEPROM) ή MMC.

1.3.3.2 Μνήμη εργασίας (Work Memory)

Αυτή είναι πάντα ενσωματωμένη RAM υψηλής απόδοσης (ταχύτητας) και διατηρεί το περιεχόμενό της μόνο με τη βοήθεια μπαταρίας. Εδώ υπάρχουν και εκτελούνται μόνο τα άκρως απαραίτητα στοιχεία από το πρόγραμμά μας καθώς και τα μπλοκ δεδομένων χώροι όπου αποθηκεύονται τιμές (αριθμοί). Η εργασία αυτή η μεταφορά των απαραιτήτων στοιχείων από εξωτερική μνήμη στη μνήμη εργασίας γίνεται από το λειτουργικό σύστημα της CPU.

Στην περίπτωση που το πρόγραμμα μεταφέρεται από συσκευή προγραμματισμού, τότε αυτό καταχωρείται στη μνήμη φόρτωσης και εργασίας ταυτόχρονα. Η μνήμη αυτή προϋπάρχει σε κάθε CPU στη μεγίστη τιμή της, πράγμα που σημαίνει ότι δεν επεκτείνεται με εξωτερικές μνήμες (RAM ή Flash EPROM)

1.3.3.3 Μνήμη συστήματος (System Memory)

Η μνήμη συστήματος περιέχει τις μεταβλητές στις οποίες αναφερόμαστε στο πρόγραμμά μας. Αυτές υπάρχουν ομαδοποιημένες στη μνήμη και το μέγεθος τους εξαρτάται από την εκάστοτε χρησιμοποιούμενη CPU. Η μνήμη συστήματος στη CPU περιλαμβάνει τις παρακάτω περιοχές:

- Μνήμη απεικόνισης εισόδων

- Μνήμη απεικόνισης εξόδων
- Βοηθητικά
- Χρονικά
- Απαριθμητές
- Τοπικά βοηθητικά
- Διαγνωστικά

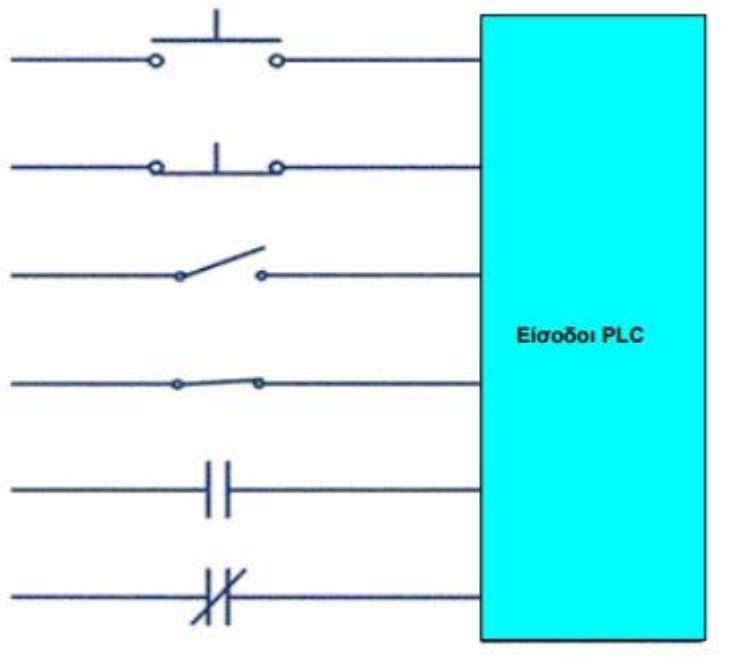
1.3.4 Μονάδες εισόδων/εξόδων

Οι μονάδες εισόδων και εξόδων χωρίζονται σε δύο κατηγορίες στις αναλογικές και στις ψηφιακές αντίστοιχα.

1.3.4.1 Μονάδες ψηφιακών εισόδων (Digital Input – D/I)

Ο ρόλος τους είναι να μεταφέρουν την εικόνα της εγκατάστασης στη CPU, όπως για παράδειγμα ότι πατήθηκε ένας τερματικός ή ότι ο χειριστής πάτησε ένα μπουτόν. Η πληροφορία αυτή μεταφέρεται ηλεκτρικά σε κλέμα της κάρτας εισόδου, ψηφιοποιείται και αποθηκεύεται στη μνήμη απεικόνισης εισόδων.

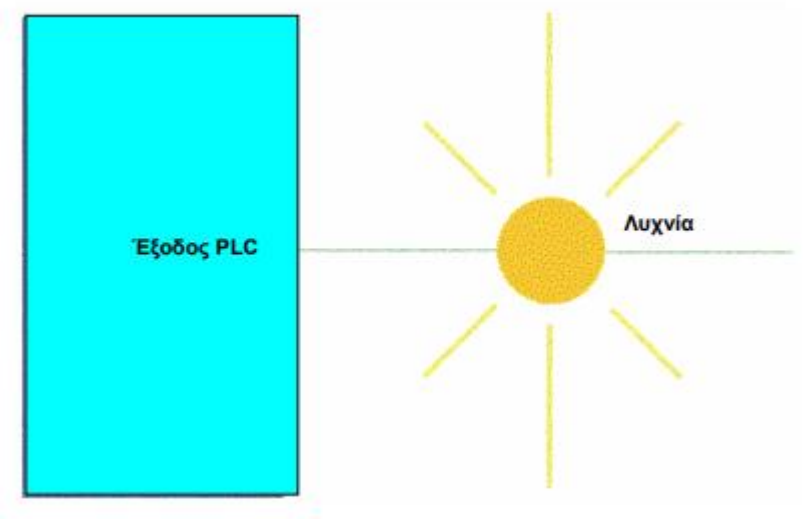
Σαν ψηφιακή πληροφορία εννοούμε αυτήν πού μπορεί να πάρει μόνο 2 διακριτές τιμές. Έτσι για παράδειγμα, σ’ ένα έμβολο ο τερματικός του διακόπτης είτε θα είναι ενεργοποιημένος είτε όχι. Λογικά αλλά και κατασκευαστικά καμία άλλη ενδιάμεση κατάσταση δεν είναι δυνατή. Σε ηλεκτρική υλοποίηση σημαίνει ότι ο τερματικός διακόπτης είναι μία επαφή η οποία μπορεί να είναι είτε ανοικτή είτε κλειστή. Αν τροφοδοτήσουμε την επαφή αυτή με τάση τότε η τάση αυτή, όταν κλείσει η επαφή, θα εμφανισθεί και στην αντίστοιχη κλέμα εισόδου του PLC. Οταν έχουμε τάση σε μια κλέμα κάρτας εισόδου του PLC τότε λέμε ότι εκεί έχουμε σήμα «1» ενώ στην αντίθετη περίπτωση ότι έχουμε σήμα «0».



Σχήμα 1.3.3 (Α) *Ψηφιακή Είσοδος (Digital Input)*

1.3.4.2Μονάδες ψηφιακών εξόδων (Digital Output – D/O)

Ο ρόλος τους είναι να μετατρέπουν τις αποφάσεις που πήρε η CPU σε εντολές προς την εγκατάσταση, όπως για παράδειγμα να εκκινήσει ένας κινητήρας, ν' ανάψει μια λυχνία ή να ηχήσει μια κόρνα. Οι αποφάσεις αυτές βρίσκονται καταχωρημένες στη μνήμη απεικόνισης εξόδων στη CPU και μετατρέπονται σε ηλεκτρικά σήματα από τις κάρτες εξόδων. Οι κάρτες εξόδου λειτουργούν σαν διακόπτες, στους οποίους δίνουμε εμείς τη τάση και όταν κλείσει ο διακόπτης η τάση περνάει και πηγαίνει προς το υπόλοιπο κύκλωμα. Έτσι, η τροφοδοσία των καρτών γίνεται εξωτερικά, ανάλογα με το τι τάση χρησιμοποιείται.



Σχήμα 1.3.3 (Β) *Ψηφιακή έξοδος (Digital Output)*

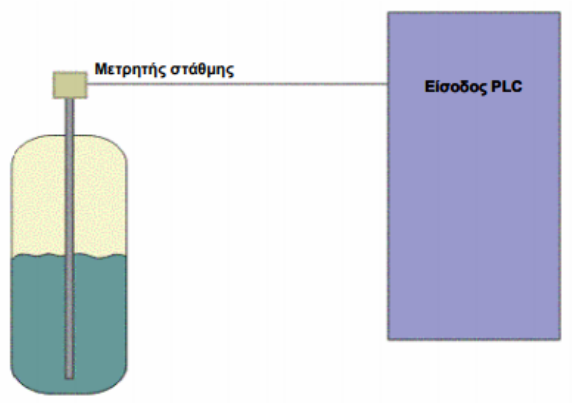
1.3.4.3Μονάδες αναλογικών εισόδων (Analog Input - A/I)

Οι κάρτες που εξετάσαμε μέχρι τώρα επεξεργάζονταν σήματα πού έχουν 2 μόνο καταστάσεις. Στην καθημερινή πρακτική όμως μας ενδιαφέρουν και σήματα τα οποία έχουν διαρκή μεταβολή. Έτσι στην παρακολούθηση της στάθμης μίας δεξαμενής μας ενδιαφέρει η ακριβής τιμή της (π.χ. 88 cm) και όχι μόνο το αν αυτή είναι πάνω η κάτω από ένα όριο, πληροφορία που θα μας έδινε εύκολα ένα φλοτέρ τοποθετημένο στο κατάλληλο σημείο. Τέτοιου είδους μεγέθη πού έχουν ένα συνεχώς μεταβαλλόμενο φάσμα τιμών, λέγονται αναλογικά. Για την επεξεργασία τους μετατρέπουμε πρώτα το φυσικό μέγεθος σε ηλεκτρικό, με την βοήθεια του κατάλληλου αισθητήρα και στη συνέχεια το εισάγουμε στο PLC.

Οι μονάδες πού έχουν την δυνατότητα να επεξεργαστούν τέτοια συνεχή μεγέθη και να τα μετατρέψουν σε μορφή κατάλληλη να τα κατανοήσει η CPU είναι οι κάρτες αναλογικών εισόδων. Οι κάρτες αναλογικών εισόδων λοιπόν δέχονται ρεύματα η τάσεις που παράγονται από τα αναλογικά αισθητήρια. Οι τυποποιημένες τιμές για τα ρεύματα είναι 0-20mA και 4-20 mA ενώ για τις τάσεις το 0-10V,+-5V κ.α.

Όταν δεν επιβάλλεται συγκεκριμένο πρότυπο από ήδη υπάρχουσα εγκατάσταση, το πιο βολικό πρότυπο είναι αυτό των 4-20mA γιατί αφ' ενός μπορεί να χρησιμοποιηθεί μήκος καλωδίου μέχρι και 1 km αφ’ ετέρου, επιτρέπει την ανίχνευση κομμένου καλωδίου.

Αυτό γίνεται πολύ απλά μια και όταν αντιστοιχίσουμε ένα φυσικό μέγεθος π.χ. πίεση 0-10Bar σε ένα ρεύμα 4-20mA τότε ακόμα και όταν η πίεση είναι μηδέν στον αγωγό μας θα κυκλοφορεί το ρεύμα πού αντιστοιχεί σε αυτήν δηλ 4mA. Έτσι αν το PLC διαπιστώσει ότι σε μια του αναλογική είσοδο διαβάζει ρεύμα 0 mA γνωρίζει ότι δεν έχει μηδενισθεί το φυσικό μέγεθος που παρακολουθεί αλλά έχει κοπεί ο αγωγός μεταφοράς οπότε μπορεί να λάβει τα απαραίτητα μέτρα προστασίας. Επιπλέον τα πρότυπα ρεύματος είναι περισσότερο ανεκτικά σε ηλεκτρομαγνητικό θόρυβο από τα αντίστοιχα τάσης.



Σχήμα 1.3.3(C) *Αναλογική Είσοδος (Analog Input)*

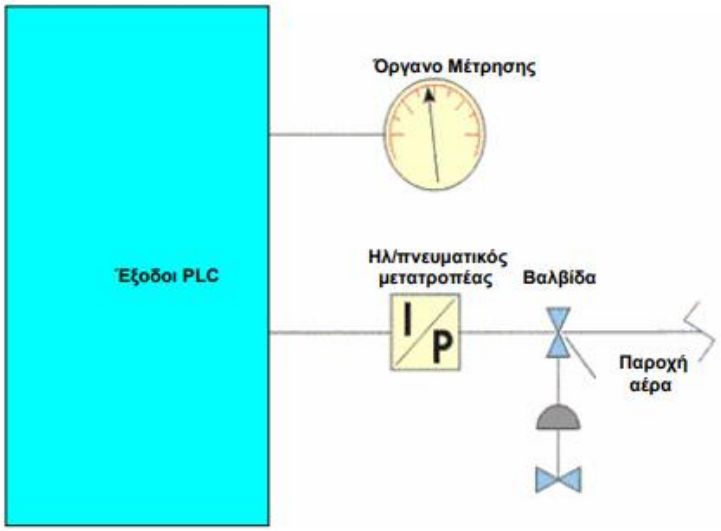
1.3.4.4Μονάδες αναλογικών εξόδων (Analog Output – A/O)

Στις παραγωγικές διαδικασίες πολλές φορές απαιτείται να ελέγξουμε το ποσοστό λειτουργίας μιας μονάδος και όχι μόνο αν θα λειτουργεί η όχι.

Έτσι για παράδειγμα η παροχή καυσίμου σε ένα μπέκ θα μπορούσε να ρυθμίσει την θερμοκρασία ενός λέβητα, η ο έλεγχος των στροφών ενός κινητήρα την ταχύτητα μιας μεταφορικής ταινίας. Σε όλες αυτές τις περιπτώσεις επιβάλλεται η χρήση μιας κάρτας αναλογικών εξόδων. Η κάρτα αυτή αναλαμβάνει να μετατρέψει το αριθμητικό μέγεθος με το οποίο «σκέπτεται» η CPU στην κατάλληλη τιμή ρεύματος ή τάσης ώστε να μπορεί να οδηγηθεί το ανάλογο εξάρτημα πού ελέγχει το φυσικό μέγεθος της εγκατάστασης μας.

Έτσι, αν θέλουμε να ελέγξουμε το άνοιγμα μιας αναλογικής δικλείδας από 0-100 % (και άρα την παροχή ρευστού σε έναν αγωγό), η κάρτα αναλογικών εξόδων θα τροφοδοτήσει την αναλογική δικλείδα με ρεύμα 4-20 mA. Οι μεταβολές του ρεύματος θα αντιστοιχούν σε μεταβολές (0-100%) του ποσοστού ανοίγματος της δικλείδας. Όλα τα χαρακτηριστικά των

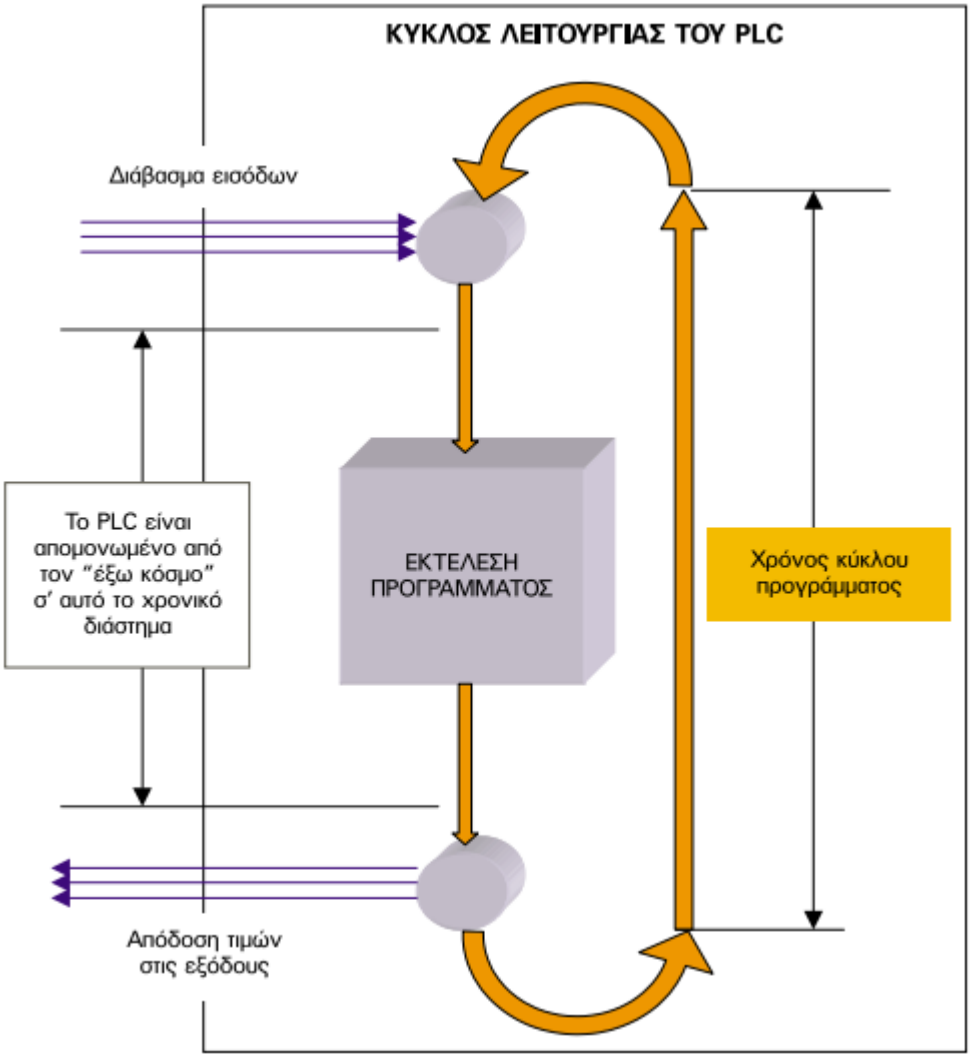
καρτών είναι σε πλήρη αντιστοιχία με αυτά των αναλογικών εισόδων μια και εκτελούν απλώς την αντίστροφη διαδικασία



Σχήμα 1.3.3(D) *Αναλογική Έξοδος(Analog Output)*

1.4 Αρχή λειτουργίας ενός προγραμματιζόμενου λογικού ελεγκτή

Η εκτέλεση του προγράμματος του PLC, είναι μέρος μιας επαναλαμβανόμενης διαδικασίας που ονομάζεται κύκλος του PLC. Ο κύκλος ξεκινά με ανίχνευση (διάβασμα) της κατάστασης των εισόδων του PLC. Στη συνέχεια και με βάση την πληροφορία αυτή εκτελείται το πρόγραμμα. Μετά το PLC εκτελεί εσωτερικές διαγνωστικές λειτουργίες και λειτουργίες επικοινωνιών. Τέλος ενημερώνεται (τροποποιείται ή παραμένει η ίδια) η κατάσταση των εξόδων και ο κύκλος ξεκινά από την αρχή. Ο χρόνος που απαιτείται για να ολοκληρωθεί ένας κύκλος του PLC εξαρτάται από το μέγεθος του προγράμματος, το πλήθος των εισόδων και των εξόδων και επίσης από τον όγκο των επικοινωνιών που ίσως πρέπει να υλοποιηθούν.



Σχήμα 1.4 *Κύκλος λειτουργίας PLC*

1.5 Ονοματολογία των PLC

Είναι γνωστό από τα παραπάνω πως τα PLC χρησιμοποιούν το δυαδικό σύστημα, γεγονός που συμβαδίζει απόλυτα με την τεχνολογία των αυτοματισμών και γενικότερα των υπολογιστικών συστημάτων, από την στιγμή που όλα τα συστήματα αυτού του τύπου μπορούν μόνο να αντιληφθούν αν τα στοιχεία τους διαρρέονται ή όχι από ηλεκτρικό ρεύμα, με άλλα λόγια αν είναι ανοιχτοί, ή κλειστοί οι συμβατικοί αυτοματισμοί.

Όπως έχει ήδη αναφερθεί, τα PLC έχουν την ίδια οργάνωση μνήμης με τους προσωπικούς υπολογιστές σε bits, και έπειτα σε bytes. Ακολουθώντας την ίδια λογική ονοματίζονται και οι διευθύνσεις των εισόδων και των εξόδων σε ένα PLC.

1.5.1 Ονοματολογία εισόδων/εξόδων

1.5.1.1Είσοδοι (Input – I)

Οι είσοδοι ενός PLC συμβολίζονται με το γράμμα I (Input). Μονοσήμαντα μια είσοδος χαρακτηρίζεται από δύο στοιχεία - σε ποια οκτάδα ανήκει (byte) και στα όρια αυτής της οκτάδας σε ποια επιμέρους θέση (bit).

Χαρακτηρισμός

I x.y όπου:

χ - Διεύθυνση byte (0 ... η, ανάλογα με τη χρησιμοποιούμενη CPU)

y - Διεύθυνση bit (0 ... 7)

1.5.1.2Έξοδοι (Output-Q)

Οι έξοδοι ενός PLC συμβολίζονται με το γράμμα Q (Output). Μονοσήμαντα μια έξοδος χαρακτηρίζεται από δύο στοιχεία - σε ποια οκτάδα ανήκει (byte) και στα όρια αυτής της οκτάδας σε ποια επιμέρους θέση (bit).

Χαρακτηρισμός

Q x.y όπου:

χ - Διεύθυνση byte {0 ... π, ανάλογα με τη χρησιμοποιούμενη CPU)

y - Διεύθυνση bit (0 ... 7)

1.5.2 Ονοματολογία χρονικών

Με τον όρο χρονικό εννοούμε μία λέξη μέσα σε μία ειδική περιοχή της μνήμης, την περιοχή των χρονικών. Σε αυτή τη περιοχή τοποθετείται με το πρόγραμμα η τιμή του χρόνου.

Όταν ορίζεται από το πρόγραμμα, ξεκινάει ή αντίστροφη μέτρηση, η μείωση, δηλαδή, της τιμής που έχει δηλωθεί με τον ρυθμό της χρονικής μονάδας. Την στιγμή που η τιμή του χρονικού μηδενιστεί, λαμβάνουμε ένα αντίστοιχο αξιοποιήσιμο σήμα.

Χαρακτηρισμός

Τ χ όπου:

χ - ο αριθμός του χρονικού

1.5.3 Ονοματολογία απαριθμητών

Με τον όρο απαριθμητής εννοούμε μία λέξη μέσα σε μία ειδική περιοχή της μνήμης, την περιοχή των απαριθμητών.

Σε αυτή τη λέξη τοποθετείται κάθε στιγμή το περιεχόμενο του απαριθμητή, το οποίο αυξάνεται ή ελαττώνεται κατά ένα, με κατάλληλες εντολές από το πρόγραμμα. Το περιεχόμενο αυτό μπορεί να ζητηθεί και να αξιολογηθεί.

Χαρακτηρισμός

C χ όπου:

χ – ο αριθμός του απαριθμητή

2Ο ΚΕΦΑΛΑΙΟ

"SIMATIC S7"

2.1 Η οικογένεια Simatic S7

Η σειρά Simatic S7 είναι μια οικογένεια Προγραμματιζόμενων Λογικών Ελεγκτών έτσι σχεδιασμένων ώστε να μπορούν να χρησιμοποιηθούν σε πλήθος εφαρμογών αυτοματισμού. Ο συμπαγής σχεδιασμός τους, το χαμηλό κόστος και το πολύ ισχυρό σετ εντολών κάνει το Simatic S7 ιδανική λύση για μικρές εφαρμογές ελέγχου. Επίσης οι πολλαπλές δυνατότητες προγραμματισμού μας παρέχουν την απαραίτητη ευελιξία για να προσαρμόσετε τα προϊόντα της σειράς Simatic S7 στις ανάγκες σας.

Σχήμα 2.1 *LOGO!*

Σχήμα 2.1(A) *S7-200*

Σχήμα 2.1(B) *S7-300*

Σχήμα 2.1(C) *S7-400*

Σχήμα 2.1(D) *S7-1200*

Σχήμα 2.1(E) *S7-1500*

Τύποι PLC

Τα PLCs χωρίζονται σε δύο κατηγορίες (ανεξάρτητες των εταιρειών). Τα compact (συμπαγούς μορφής) και τα modular (δομοστοιχειωτής δομής).

Τα πρώτα είναι μία συμπαγής συσκευή με CPU, τροφοδοτικό και συγκεκριμένο αριθμό I/O (που ποικίλει ανάλογα με την εταιρεία).

Τα δεύτερα περιλαμβάνουν μία βάση, στην οποία "κουμπώνουν" οι μονάδες επεξεργασίας, τροφοδοσίας, εισόδων, εξόδων. Ένα PLC μπορεί να διαθέτει περισσότερες από μια μονάδες εισόδων και εξόδων, ανάλογα με τον επιθυμητό αριθμό εισόδων ή εξόδων. Επομένως, αν σε κάποιο αυτοματισμό, προκειμένου να τον επεκτείνουμε, χρειαστούμε κι άλλες εισόδους ή εξόδους, που δεν υπάρχουν στην αρχική κατασκευή, έχουμε τη δυνατότητα να προσθέσουμε μία ή περισσότερες μονάδες εισόδων ή εξόδων, διατηρώντας την ίδια CPU και το ίδιο τροφοδοτικό.

S7-300

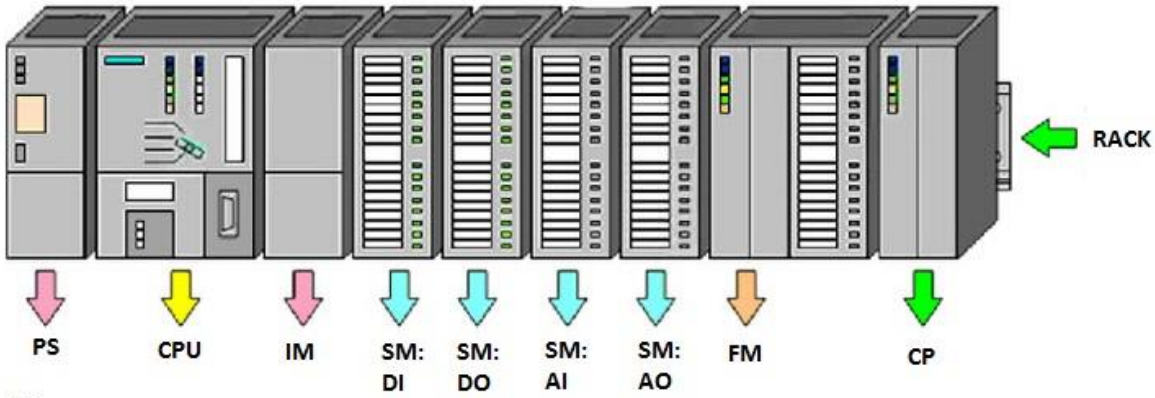
Μοντέλο από αυτή τη σειρά θα χρησιμοποιήσουμε για την δημιουργια των εφαρμογών και πάνω σ' αυτό θα αναφερόμαστε στο εξής. Απευθύνεται για μεσαίας κλίμακας εφαρμογές στις οποίες συγκαταλέγονται και οι περισσότερες των εφαρμογών στην ελληνική αγορά. Τα κυριότερα χαρακτηριστικά του είναι:

- Modular μορφή
- Μεγάλη ποικιλία από CPU για τη βέλτιστη επιλογή ανάλογα με την εττιθυμητή απόδοση
- Επεκτασιμότητα με έως 32 μονάδες
- Δικτυώνεται με όλα τα πρότυπα δίκτυα (Profibus, Industrial Ethernet)
- Δεν έχει περιορισμό για τη θέση των εκ'μέρους μονάδων
- Δεν υπάρχουν μικροδιακόπτες (Dip switches) για την παραμετροποίηση όλα γίνονται μέσω λογισμικού
- Έχει πλήρες 32-bit σετ εντολών (ακόμα και για τριγωνομετρικές εξισώσεις)
- Ενσωματωμένη δυνατότητα δικτύωσης (MPI) στην κεντρική μονάδα
- Ενσωματωμένες δυνατότητες διασύνδεσης με HMI
- Μνήμη διαγνωστικών αυτόματη αποθήκευση με χρόνο και ημερομηνία όλων των συμβάντων στο PLC
- Μια μόνο μονάδα για τα αναλογικά , η επιλογή γίνεται μέσω του λογισμικού

2.2 Απαιτούμενος εξοπλισμός

Οι μονάδες από τις οποίες μπορεί να αποτελείται το S7 300 που χρησιμοποιείται στην εφαρμογή μας, περιγράφονται παρακάτω:

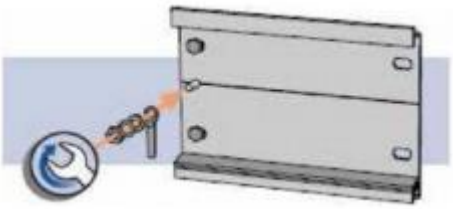
- Το πλαίσιο στήριξης (Rack)
- Τροφοδοτικό PS (Power Supply)
- Κεντρική Μονάδα Επεξεργασίας CPU (Central Processing Unit)
- Κάρτες Εισόδων / Εξόδων (ψηφιακές ή αναλογικές)
- Μονάδες ειδικών λειτουργιών FM (Function Module)
- Επεξεργαστής επικοινωνίας CP
- Κάρτα διασύνδεσης των Rack Interface Module IM.
- Καλώδιο Profibus δικτύο με τους Bus Connector.



'Οπου:
Rack : Πλαίσιο στήριξης
PS: Τροφοδοτικο
IM: Κάρτα διασύνδεσης
DI: Ψηφιακή είσοδος
DO: Ψηφιακή έξοδος
AI: Αναλογική είσοδος
AO: Αναλογική έξοδος
FM: Μονάδα ειδικών λειτουργιών
CP: Επεξεργαστής επικοινωνίας

Σχήμα 2.2 *Απαιτούμενος εξοπλισμός*

2.2.1 Πλαίσιο στήριξης (Rack)

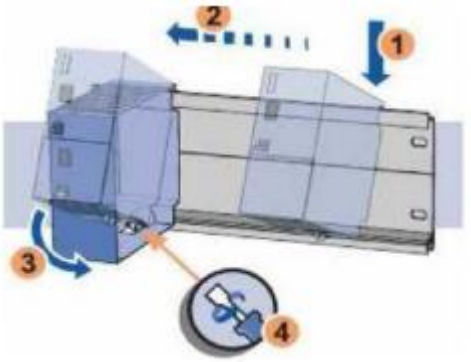


Σχήμα 2.3.1 *Πλαίσιο στήριξης (RACK)*

Ο ρόλος του είναι να στηρίζει απλά τις διάφορες κάρτες που θα συνδέσουν το σύστημα αυτοματισμού. Η επικοινωνία μεταξύ καρτών γίνεται με ένα συνδετήρα σχήματος "Π" στο πίσω μέρος των καρτών. Μέσω αυτού υλοποιούνται οι δίαυλοι επικοινωνίας.

Στο Rack, υπάρχει συγκεκριμένη σειρά που πρέπει να τοποθετείται ο εξοπλισμός. Στην πρώτη θέση κουμπώνουμε την κάρτα του τροφοδοτικού, στη δεύτερη θέση τοποθετούμε την CPU, την τρίτη θέση είτε χρησιμοποιούμε είτε όχι κάρτα διασύνδεσης των Rack (IM) πρέπει να την διαθέσουμε γι' αυτήν, από την τέταρτη θέση και πέρα πάνω στο Rack συνδέονται τα υπόλοιπα στοιχεία.

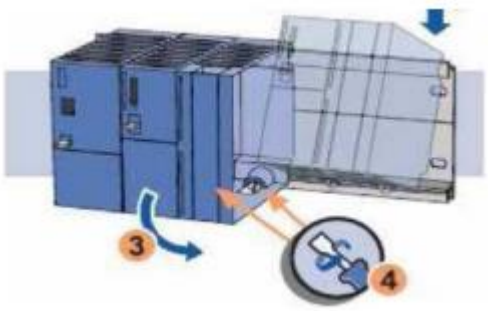
2.2.2 Τροφοδοτικό (Power Supply)



Σχήμα 2.4.2 *Τροφοδοτικό*

Ο σκοπός του είναι από την υπάρχουσα τάση δικτύου να δημιουργήσει τις απαραίτητες τάσεις για την λειτουργία του ίδιου του PLC

2.2.3 Κεντρική μονάδα επεξεργασίας CPU



Σχήμα 2.1.3 *Κεντρική μονάδα επεξεργασίας*

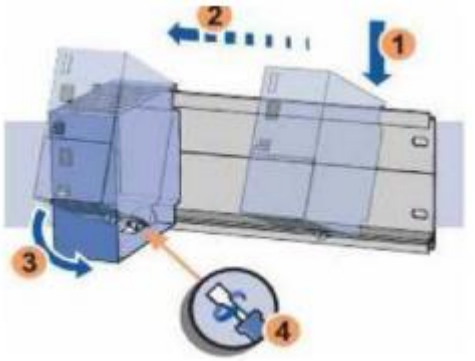
Η κεντρική μονάδα επεξεργασίας CPU (Central Processing Unit), είναι ταυτόχρονα ο εγκέφαλος και η κινητήριος δύναμη ενός PLC.

Η κεντρική μονάδα επεξεργασίας πραγματοποιεί πολλαπλές βασικές λειτουργίες:

- Διάβασμα, ερμηνεία και εκτέλεση, με τη σωστή διαδοχή, των οδηγιών που περιέχονται στη μνήμη
- Έλεγχο του πρωτοκόλλου επικοινωνίας που έχουμε καθορίσει στο σύστημά μας.
- Αποθήκευση των πληροφοριών
- Εκτέλεση αριθμητικών πράξεων.

Κατά μια άποψη, εάν συγκρίνουμε τη CPU με την καλωδιωμένη λογική τότε η CPU είναι το στοιχείο εκείνο το οποίο πραγματοποιεί τις καλωδιώσεις οι οποίες ζητούνται από τον κύκλο εργασίας της μηχανής ή της εγκατάστασης. Σε αντίθεση όμως από την καλωδιωμένη λογική της οποίας η λειτουργία είναι «παράλληλη», το PLC εκτελεί τις λειτουργίες του με «σειριακό» τρόπο. Για το λόγο αυτό στα PLC είναι χαρακτηριστική η ταχύτητα λειτουργίας των κυκλωμάτων.

2.2.4 Μονάδες εισόδων/εξόδων



Σχήμα 2.1.4 *Κάρτες εισόδων/εξόδων*

Προσαρμόζουν τα ηλεκτρικά σήματα από το εξωτερικό περιβάλλον προς τη CPU και αντιστρόφως.

Μια κάρτα ψηφιακών εισόδων των 24 VDC αναγνωρίζει σα σήμα «+1» τα 24 V_{DC} και σαν σήμα «0» τα 0 V. Στις περιπτώσεις εκείνες που υπάρχει διακύμανση στην τάση (μη σταθεροποιημένο τροφοδοτικό) οι ψηφιακές κάρτες έχουν ανοχές. Έτσι σαν σήμα «+1» καταλαβαίνει τις τάσεις 13+30 V_{DC} και σα σήμα «0» τις τάσεις -3 +5 VDC. Οι αναλογικές τιμές που μπορούν να διαβάσουν οι αναλογικές είσοδοι, έξοδοι είναι 0+10 V ή ±10 V για την τάση και 0+20 mA ή 4+20 mA για την ένταση. Ένα άλλο μέγεθος που μας ενδιαφέρει στην επιλογή μιας κάρτας αναλογικών εισόδων είναι η διακριτική τους ικανότητα (ακρίβεια). Όσο μεγαλύτερο φάσμα αριθμών μπορεί να χρησιμοποιήσει μια αναλογική κάρτα για την μετατροπή ενός αναλογικού μεγέθους τόσο μεγαλύτερη θα είναι η ακρίβεια. Το πλήθος των δυαδικών ψηφίων που χρησιμοποιούνται για την αναπαράσταση του αναλογικού μεγέθους καθορίζει το μέγιστο αριθμό των βημάτων μετατροπής. Συνήθως χρησιμοποιούνται 16 bits.

Ένα ακόμα μεγαλύτερο πλεονέκτημα της σειράς S7 είναι ότι μια αναλογική κάρτα εισόδων μπορεί να γίνει τάσης ή έντασης και να μεταβάλουμε την περιοχή μέτρησης της επεμβαίνοντας τόσο εξωτερικά πάνω στην ίδια τη κάρτα όσο και στο software.

3ο Κεφάλαιο

"ΛΟΓΙΣΜΙΚΟ SIMATIC MANAGER S7"

3.1 Δομή Προγράμματος

Κατά τη φάση του σχεδιασμού ενός project, μία από τις πρώτες ενέργειες είναι να αποφασιστεί ο τρόπος δόμησης του προγράμματος, δηλαδή, τι block θα περιέχει και πώς θα συνδέονται αυτά μεταξύ τους.

Κάθε CPU περιλαμβάνει δύο προγράμματα ανεξάρτητα το ένα από το άλλο:

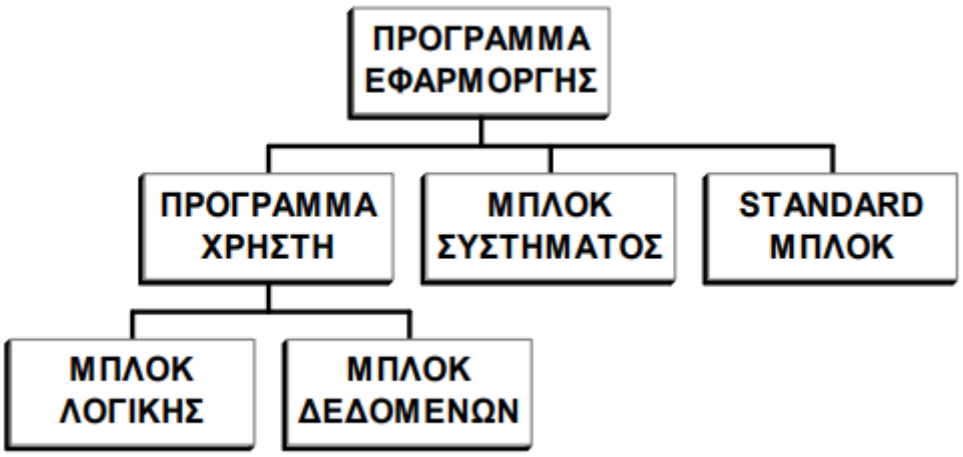
- **Λειτουργικό σύστημα**

  Το λειτουργικό σύστημα είναι το σύνολο των ορισμών και εντολών που ελέγχουν τους πόρους του συστήματος. Είναι αυτό που ενημερώνει το ρολόι του πραγματικού χρόνου στη CPU, που ελέγχει την κατάσταση του διακόπτη της CPU (RUN, STOP κλπ), ελέγχει την ενεργοποίηση των led στη CPU και ρυθμίζει τις επικοινωνίες μέσα από το MPI Interface. Στο λειτουργικό σύστημα δε γίνονται μεταβολές αλλά είναι δυνατή η ανάγνωση ή η χρήση ορισμένων αποτελεσμάτων αυτού.

- **Πρόγραμμα εφαρμογής**

  Το πρόγραμμα εφαρμογής είναι το σύνολο των εντολών και ορισμών που χρειάζεται το PLC για τον έλεγχο της εγκατάστασης. Αυτό μπορεί να περιέχει block λογικής (εντολές) και block δεδομένων (όπου καταχωρούνται λίστες με αριθμούς). Επίσης υπάρχουν και έτοιμα block τα οποία περιέχουν λειτουργίες από πριν ορισμένες και καταχωρημένες στο λειτουργικό σύστημα του PLC. Στο πρόγραμμά του ο χρήστης καλεί αυτά τα block σε οποιοδήποτε σημείο τα χρειαστεί, δίνοντας τους κάποιες παραμέτρους και λαμβάνει τα αποτελέσματα, χωρίς να ενδιαφέρει το πώς παρήχθησαν. Ανάλογα με τον τρόπο που χτίζεται ένα πρόγραμμα υπάρχουν τρία διαφορετικά είδη δόμησης.

3.1.1 Πρόγραμμα Εφαρμογής



Σχήμα 3.1.1 *Δομή προγράμματος εφαρμογής*

3.1.1.1 Πρόγραμμα Χρήστη

Είναι το πρόγραμμα που εμείς γράφουμε για τις λειτουργικές ανάγκες της εγκατάστασης και του αυτοματισμού. Αυτό μπορεί να περιέχει block λογικής (εντολές) και block δεδομένων (όπου καταχωρούνται λίστες με αριθμούς).

3.1.1.2 Block Συστήματος

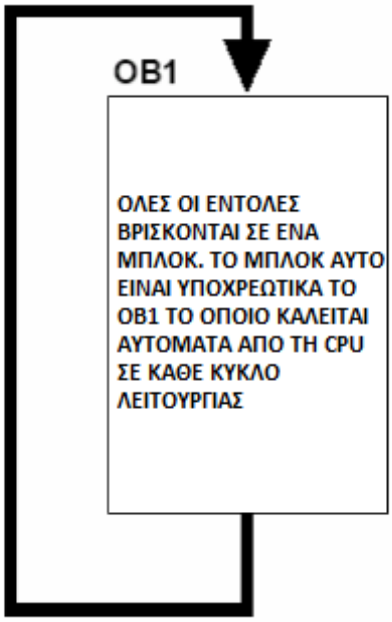
Είναι λειτουργίες που είναι από πριν ορισμένες και καταχωρημένες στο λειτουργικό σύστημα του PLC. Στο πρόγραμμα του ο χρήστης καλεί αυτά τα block σε οποιοδήποτε σημείο θέλει, τους δίνει κάποιες παραμέτρους και παίρνει μόνο τα αποτελέσματα, χωρίς να ενδιαφέρεται για το πώς έχουν αυτά παραχθεί.

3.1.1.3 Standard Block

Είναι block που μας προσφέρουν έτοιμες λύσεις για τυποποιημένες εργασίες αυτοματισμού που πιθανόν να μας ενδιαφέρουν.

3.1.2 Γραμμικό Προγραμμα

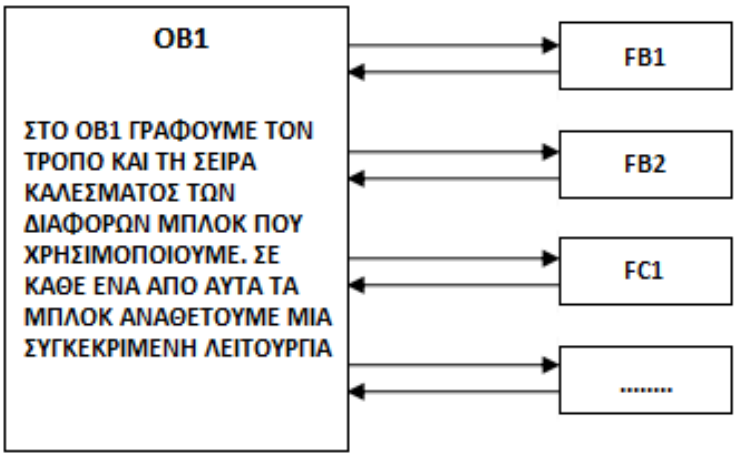
Όλο το πρόγραμμα του χρήστη βρίσκεται σ' ένα συνεχόμενο block (OB1 που καλείται αυτόματα σε κάθε κύκλο λειτουργίας). Η CPU επεξεργάζεται τις εντολές την μια μετά την άλλη μέχρι το τέλος του block και ξαναρχίζει η ίδια διαδικασία πάλι από την αρχή(κύκλος λειτουργίας). Έχει το πλεονέκτημα ότι εύκολα και γρήγορα αρχίζει κάποιος τη φάση του προγραμματισμού. Έχει το μειονέκτημα ότι σε μεγάλα προγράμματα είναι δύσκολο να εντοπίσουμε που γίνεται μια συγκεκριμένη εργασία.



Σχήμα 3.1.2 *Γραμμικό Πρόγραμμα*

3.1.3 Τμηματοποιημένο Πρόγραμμα

Το πρόγραμμα χωρίζεται σε block όπου κάθε ένα από αυτά υλοποιεί μια συγκεκριμένη εργασία. Για τον τρόπο κλήσης, την σωστή λειτουργία τους καθώς και την σωστή σειρά εκτέλεσης τους φροντίζει ένα ειδικό block το οποίο λέγεται block οργάνωσης (OB1).



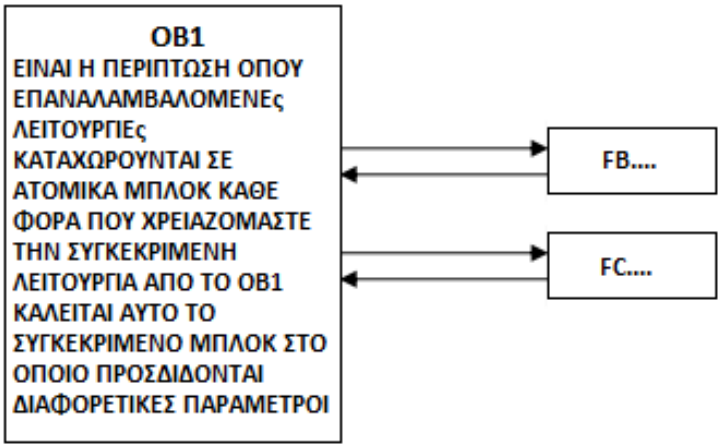
Σχήμα 3.1.3 *Τμηματοποιημένο Πρόγραμμα*

3.1.4 Δομημένο Πρόγραμμα

Ένα δομημένο πρόγραμμα περιλαμβάνει οπωσδήποτε block με παραμέτρους. Αυτά τα block είναι σχεδιασμένα έτσι ώστε να μπορούν να είναι γενικής χρήσης. Όταν καλείται ένα τέτοιο block του δίνουμε τις τρέχουσες παραμέτρους για μια διαδικασία (διευθύνσεις εισόδων, εξόδων, χρονικά κλπ).

Ο δομημένος προγραμματισμός μας προσφέρει πολλά προτερήματα:

- Εξοικονόμηση μνήμης (δεν επαναλαμβάνουμε το γράψιμο ίδιων προγραμμάτων)
- Οποιαδήποτε αλλαγή στη λογική του αυτοματισμού την περνάμε μια φορά στο πρόγραμμα και αυτόματα γίνεται η διόρθωση της λειτουργίας οπού χρειάζεται (εξοικονόμηση χρόνου και ελαχιστοποίηση της πιθανότητας σφάλματος από λανθασμένη πληκτρολόγηση).



Σχήμα 3.1.4 *Δομημένο Πρόγραμμα*

3.2 Τύποι Διαθέσιμων BLOCK

Για το χτίσιμο μιας εφαρμογής ο προγραμματιστής έχει στην διάθεση του διαφορετικά είδη block προγραμματισμού. Το τι θα χρησιμοποιήσει και πως θα τα διασυνδέσει είναι τις περισσότερες φορές υποκειμενική υπόθεση και εξαρτάται από την εφαρμογή που πρέπει να προγραμματίσει.

Σε γενικές γραμμές ένα block που περιέχει κώδικα αποτελείται από τα εξής μέρη :

- Την κεφαλή του block (Block Header) όπου περιλαμβάνει τις ιδιότητες του block και το όνομα του
- Την περιοχή των δηλώσεων (Declarations) οπου δηλώνονται οι τοπικές μεταβλητές του block (Local Variables)
- Την περιοχή η οποία περιλαμβάνει τον κώδικα του χρήστη αλλα και τα τυχόν σχόλια (Code section)

3.2.1 BLOCK Οργάνωσης OB (Organazation Block)

Έχουν τον ρόλο του διαμεσολαβητή μεταξύ του λειτουργικού συστήματος και του προγράμματος του χρήστη. Κατά την εκδήλωση κάποιων ειδικών γεγονότων, όπως για παράδειγμα μιας χρονικής διακοπής, μιας διακοπής τροφοδοσίας, το λειτουργικό σύστημα της CPU καλεί το αντίστοιχο block οργάνωσης.

Ένα από τα διάφορα block οργάνωσης, σημαντικότερο απ’ όλα είναι το OB1. Αυτό είναι ένα block το οποίο η CPU καλεί αυτόματα και το εκτελεί συνεχώς κυκλικά. Μέσα σ’ αυτό το block βρίσκεται το κύριο πρόγραμμα του χρήστη.

Άλλο σημαντικό block είναι το OB100 που εκτελείται μία φορά όταν δίνουμε τάση στο σύστημα. Τα block οργάνωσης έχουν τάξεις προτεραιότητας από 0 ως 29. Αν ένα block έχει μεγαλύτερη προτεραιότητα από κάποιο άλλο, τότε μπορεί να το διακόψει και να εκτελεστεί το ίδιο. Π.χ. το OB1 που έχει προτεραιότητα 1 μπορεί να διακοπεί από όλα τα άλλα block.

Στον κάτω πίνακα δίνονται όλα τα διαθέσιμα block οργάνωσης SIMATIC S7, το καθένα μαζί με την προτεραιότητα του.

Πίνακας 3.1.1

BLOCK ΟΡΓΑΝΩΣΗΣ	**ΣΥΝΘΗΚΕΣ ΚΛΗΣΗΣ**	**ΠΡΟΤΕΡΑΙΟΤΗΤΑ**	
		ΠΡΟΕΠΙΛΟΓΗ	**ΤΡΟΠΟΠΟΙΗΣΙΜΗ**
Ελεύθερος Κύκλος OB1	Κυκλικά μέσω του λειτουργικού συστήματος	1	Όχι
Χρονικές διακοπές(TOD) OB10 έως OB17	Σε συγκεκριμένη ώρα της ημέρας ή σε τακτά χρονικά διαστήματα	2	2 έως 24
Διακοπές καθυστέρησης OB20 έως OB23	Μετά απο προγραμματισμένο χρόνο ελεγχόμενο απο το πρόγραμμα χρήστη	3 έως 6	2 έως 24
Διακοπές χρονιστή επιτήρησης OB30 έως OB38	Τακτικά σε προγραμματισμένα χρονικά διαστήματα (π.χ καθε 100ms)	7 έως 15	2 έως 24
Διακοπές επεξεργασίας OB40 έως OB47	Σε σήματα διακοπών υπό τις βαθμίδες I/O	16 έως 23	2 έως 24
Διακοπή πολυεπεξεργασίας OB60	Κλήση υπο συνθήκες μέσω του προγράμαματος χρήστη σε κατάσταση πολυεπεξεργασίας	25	Όχι
Εφεδρικά σφάλματα OB70	Στην περίπτωση απώλειας στοιχείου που απορρέει απο σφάλμα I/O	25	2 έως 26
OB72 OB73	Στην περίπτωση εφεδρικού σφάλματος της CPU	28	2 έως 28
	Στην περίπτωση εφεδρικού σφάλματος επικοινωνιών	25	2 έως 26
Ασύγχρονα σφάλματα OB80	Λάθη που δεν σχετίζονται με την εκτέλεση του προγράμματος (π.χ. χρονικά σφάλματα, σφάλματα SE, διαγνωστικές διακοπές, διακοπές εγκατάστασης ή απεγκατάστασης βαθμίδων, αποτυχία βάσης στήριξης ή σταθμού)	26	26
OB81 έως OB84		26	2 έως 26
OB86			
OB87		26	24 έως 26
OB85			
Εκτέλεση στο παρασκήνιο OB90	Ελάχιστη διάρκεια χρονικού κύκλου που δεν έχει επιτευχθεί ακόμα	29	Όχι
Ρουτίνα εκκίνισης OB100, OB101, OB 102	Σε προγραμματιζόμενη εκκίνηση ελεγκτή	27	Όχι
Σύγχρονα σφάλματα OB121, OB122	Σφάλματα που σχετίζονται με την εκτέλεση του προγράμματος	Η προτεραιότητα των OB που προκαλέι τα σφάλματα	

3.2.2 Συναρτήσεις FC (Functions)

Οι συναρτήσεις είναι block τα οποία προγραμματίζονται από το χρήστη. Τα FC είναι block κώδικα «στερούμενου μνήμης». Οι προσωρινές μεταβλητές (temporary variables) των FC αποθηκεύονται στην περιοχή των τοπικών δεδομένων (local data stack). Μετά την επεξεργασία των FC αυτά τα δεδομένα χάνονται. Για την αποθήκευση των δεδομένων τα FC μπορούν να χρησιμοποιήσουν DB (shared data blocks).

Ένα FC περιέχει ένα πρόγραμμα το οποίο εκτελείται όταν το FC καλείται από ένα άλλο block που περιέχει κώδικα. Τα FC χρησιμοποιούνται για:

- υπολογισμό κάποιας συνάρτησης και απόδοσης τιμής στο block που το έχει καλέσει (πχ υπολογισμός μαθηματικών συναρτήσεων)
- έλεγχος μιας τεχνολογικής συνάρτησης (πχ έλεγχος ανεξάρτητων τμημάτων εγκατάστασης).

Τα FC παραμετροποιούνται και επομένως μπορούν να χρησιμοποιηθούν για περιπτώσεις στις οποίες έχουμε επαναλαμβανόμενη λογική στο πρόγραμμά μας με διαφορετικές παραμέτρους.

3.2.3 BLOCK Συναρτήσεων (Function Block)

Τα block συναρτήσεων προγραμματίζονται και αυτά από τον χρήστη και περιέχουν κώδικα. Ένα block συνάρτησης «έχει μνήμη», δηλαδή σε αυτό διατίθεται ένα block δεδομένων (DB) σαν δικιά του μνήμη. Αυτό το DB λέγεται (instance data block) και είναι μόνιμα δεσμευμένα με το block συνάρτησης και για την ακρίβεια με την κλήση (call) του block συνάρτησης. Επίσης είναι δυνατόν σε κάθε κλήση block συνάρτησης να εκχωρηθεί ένα διαφορετικό block δεδομένων (με την ίδια δομή αλλά με διαφορετικές τιμές).

Τα FB παραμετροποιούνται όπως και τα FC επομένως και αυτά χρησιμοποιούνται σε περιπτώσεις που έχουν επαναλαμβανόμενη λογική. Όταν δεν παραμετροποιούνται η λειτουργία τους δεν διαφέρει σε τίποτα από τα FC. Τόσο οι παράμετροι οι οποίες μεταβιβάζονται στα FB όσο και οι στατικές μεταβλητές (static variables) αποθηκεύονται στο instance data block. Οι προσωρινές μεταβλητές (temporary variables) αποθηκεύονται στην περιοχή των τοπικών δεδομένων.

Στο τέλος της επεξεργασίας του FB όσα δεδομένα αποθηκεύτηκαν στο instance data block δεν χάνονται ενώ αυτά τα δεδομένα τα οποία αποθηκεύονται στην περιοχή των τοπικών δεδομένων (local data stack) χάνονται. Τα FB περιέχουν πρόγραμμα το οποίο εκτελείται κάθε

φορά που τα FB καλείται από άλλο block που περιέχει κώδικα. Τα block συναρτήσεων (FB) διευκολύνουν τον προγραμματισμό συχνά χρησιμοποιούμενων και σύνθετων συναρτήσεων.

3.2.4 BLOCK Δεδομένων (Data Block)

Τα block δεδομένων δεν περιέχουν κώδικα, αλλά περιέχουν δεδομένα του προγράμματος μας. Προγραμματίζοντας τα block δεδομένων καθορίζουμε σε ποια μορφή θα αποθηκευτούν τα δεδομένα (σε ποια block, με ποια σειρά και με ποιο τύπο δεδομένων).

Υπάρχουν δύο βασικοί τρόποι χρησιμοποίησης των block δεδομένων :

- Block γενικών δεδομένων (Global Data Block)
- Πρότυπα block δεδομένων (Instance Data Block)

3.2.4.1 Block γενικών δεδομένων (Global Data Block)

Προγραμματίζονται για κοινή χρήση σε όλο το πρόγραμμα. Ένα block γενικών δεδομένων είναι, κατά κάποιο τρόπο, ένα «ελεύθερο», block μέσα στο πρόγραμμα του χρήστη και δεν εκχωρείται σε κάποιο block «κώδικα».

3.2.4.2 Πρότυπα Block δεδομένων (Instance Data Block)

Για κάθε κάλεσμα ενός block συναρτήσεων (FB) το οποίο μεταφέρει παραμέτρους δεσμεύεται ένα πρότυπο block δεδομένων (instance data block). Στο instance data block αποθηκεύονται οι πραγματικές παράμετροι και τα στατικά δεδομένα ενός FB. Οι μεταβλητές οι οποίες δηλώνονται στο FB καθορίζουν τη δομή ενός instance data block. Για να μπορέσουμε να δημιουργήσουμε ένα instance data block πρέπει να υπάρχει ήδη ένα FB στο οποίο θα δεσμευτεί.

Υπάρχουν τρεις διαφορετικοί τρόποι χρήσης των instance data block:

- Για ένα block συνάρτησης FB χρήση περισσοτέρων προτύπων δεδομένων (instance data block). Αυτόν τον τρόπο τον χρησιμοποιούμε όταν επιθυμούμε ίδιες λειτουργίες για διαφορετικές μονάδες πχ έλεγχος τριών αντλιών.
- Με ένα πρότυπο block δεδομένων (instance data block) χρήση περισσοτέρων καλεσμάτων του ίδιου block συνάρτησης (FB). Έτσι εξοικονομούμε χώρο μνήμης και κάνουμε καλύτερη διαχείριση των block δεδομένων.
- Σε ένα block συνάρτησης μπορούμε να καλέσουμε instance άλλων FB που έχουν ήδη δημιουργηθεί. Τα instance δεδομένα τα οποία είναι απαραίτητα γι’ αυτόν το σκοπό μπορούν να δηλωθούν στο instance data block του FB που καλεί και

επομένως σε αυτήν την περίπτωση δε χρειάζονται συμπληρωματικά DB (data block) για τα FB που καλούνται.

3.2.5 BLOCK συστήματος (SFC, SFB, SDB)

Αυτά τα block είναι στοιχεία του λειτουργικού συστήματος και περιέχουν προγράμματα [συναρτήσεις συστήματος (SFC) ή block συναρτήσεων συστήματος (SFB) ή δεδομένα (block δεδομένων συστήματος SDB)]. Τα blocks συστήματος μας δίνουν πρόσβαση σε ένα αριθμό σημαντικών συναρτήσεων συστήματος όπως η διαχείριση του εσωτερικού ρολογιού της CPU ή διάφορες συναρτήσεις επικοινωνίας. Μπορούμε να καλέσουμε block SFC και SFB αλλά δεν μπορούμε να τα τροποποιήσουμε ούτε να τα προγραμματίσουμε εμείς οι ίδιοι. Τα blocks αυτά δεν δεσμεύουν χώρο από την μνήμη χρήστη, οι κλήσεις όμως αυτών των blocks και τα πρότυπα blocks δεδομένων (instance data block) των SFB βρίσκονται στη μνήμη του χρήστη. Τα blocks SDB περιέχουν διάφορες χρήσιμες πληροφορίες, όπως η διαμόρφωση του συστήματος αυτοματισμού ή η παραμετροποίηση των βαθμιδών. Τα blocks αυτά δημιουργούνται από την ίδια την STEP 7. Ωστόσο εμείς μπορούμε να καθορίσουμε τα περιεχόμενα τους (για παράδειγμα όταν διαμορφώνουμε τους σταθμούς). Κατά κανόνα τα SDB βρίσκονται στη μνήμη φόρτωσης και δεν μπορούμε να τα προσπελάσουμε από το πρόγραμμα του χρήστη.

3.2.6 Standard (IEC) Block

Αυτά είναι έτοιμα block που μας προσφέρουν έτοιμες λύσεις για τυποποιημένες εφαρμογές αυτοματισμού. Τέτοιες εφαρμογές για παράδειγμα είναι η αντιγραφή μιας περιοχής μνήμης σε μια νέα περιοχή, συγκρίσεις χρόνων/ ημερομηνιών, μέγιστο, ελάχιστο σε λίστα αριθμών κλπ. Βρίσκονται στο πακέτο της βιβλιοθήκης «Standard Library» το οποίο συνοδεύει το πακέτο S7.

3.3 S7-300 Προγραμματισμός

Προγραμματισμός ενός PLC είναι η δημιουργία μιας σειράς εντολών, οι οποίες είναι σε θέση να λύνουν έναν συγκεκριμένο αλγόριθμο που αντιστοιχεί σε μια λειτουργία ενός συστήματος αυτοματισμού. Η διαδικασία που ακολουθείται για την συγγραφή αυτών των εντολών, αποτελεί το πρόγραμμα.

Απαραίτητη προϋπόθεση για την κατανόηση του τρόπου προγραμματισμού του PLC, είναι η αντίληψη της «φιλοσοφίας» στην οποία στηρίζεται η λειτουργία του. Κάθε ενέργεια του PLC υπαγορεύεται από τις εντολές που του δίνονται από το πρόγραμμα, όπου ορθές θεωρούνται φυσικά εκείνες που είναι σε θέση το PLC να «κατανοήσει» και να εκτελέσει.

Το πρώτο βήμα πριν αυτοματοποιήσουμε μια εγκατάσταση είναι να ξεχωρίσουμε τα επιμέρους τμήματα τα οποία είναι συνδεδεμένα μεταξύ τους. Αρχικά λοιπόν πρέπει να χωριστεί η διαδικασία παραγωγής σε επιμέρους εργασίες :

- **Hardware**
    1. Αριθμό και τύπο εισόδων και εξόδων
    2. Αριθμό και τύπο μονάδων
    3. Αριθμό των απαιτούμενων rack
    4. Χωρητικότητα και τύπο CPU
    5. HMI συστήματα
    6. Συστήματα δικτύωσης.

- **Software**
    1. Δομή προγράμματος
    2. Διαχείριση δεδομένων για τη διαδικασία αυτοματισμού
    3. Δεδομένα διαμόρφωσης (Configuration)
    4. Δεδομένα επικοινωνίας
    5. Τεκμηρίωση προγράμματος και project.

3.3.1 Γλώσσες Προγραμματισμού

Ο προγραμματισμός των PLC δεν απαιτεί κάποια ανώτερη γλώσσα προγραμματισμού. Οι πιο διαδεδομένες τυποποιημένες μορφές προγραμματισμού που έχουν επικρατήσει διεθνώς είναι:

- Σχέδιο επαφών (LAD – Ladder Diagram)
- Γλώσσα λογικών εντολών (STL – Statement List)
- Διάγραμμα λογικών πυλών (FBD – Function Block Diagram)

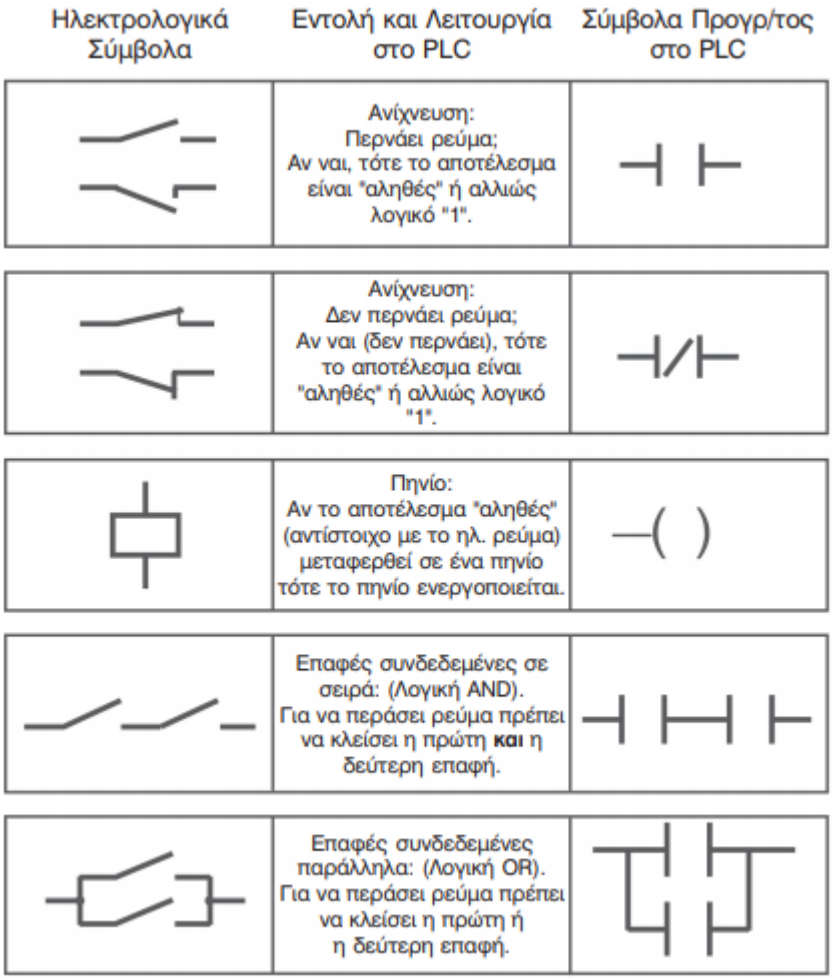
Αυτό συμβαίνει διότι με τις συγκεκριμένες γλώσσες προγραμματισμού είναι εύκολη η παρακολούθηση της ροής του προγράμματος και συνεπώς ο εντοπισμός κάποιου σφάλματος.

3.3.1.1Σχέδιο επαφών LADDER

Η πρώτη γλώσσα προγραμματισμού είναι η Ladder Logic (LAD) που είναι μια γλώσσα γραφικών που χρησιμοποιεί ηλεκτρομηχανικά σύμβολα και επιτρέπει ουσιαστικά τη μεταφορά του ηλεκτρολογικού σχεδίου στο PLC.

Με τη γλώσσα αυτή η εκπαίδευση των τεχνικών, που ήταν συνηθισμένοι στον κλασσικό αυτοματισμό, γινόταν εύκολα και γρήγορα, αφού δεν άλλαζε ουσιαστικά την εργασία σχεδιασμού του αυτοματισμού.

Η γλώσσα LADDER χρησιμοποιεί όχι την Ευρωπαϊκή τυποποίηση στο σχεδιασμό των ηλεκτρικών επαφών, αλλά την Αμερικάνικη. Αυτό ίσως οφείλεται στο γεγονός ότι τα πρώτα PLC αναπτύχθηκαν στην Αμερική. Όμως στη συνέχεια αυτός ο σχεδιασμός “βόλεψε” και έτσι διατηρήθηκε και από τις Ευρωπαϊκές εταιρίες, με αποτέλεσμα σήμερα να είναι καθιερωμένος.



Σχήμα 3.3.1.1 *Λογική LADDER*

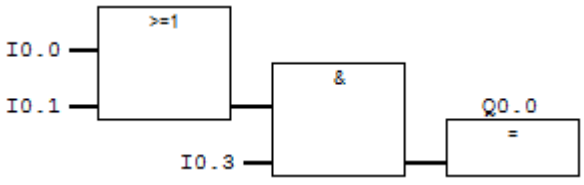
3.3.1.2Γλώσσα λογικών εντολών (STL – Statement List)

STL (Statement List ή γλώσσα λογικών εντολών). Η δεύτερη γλώσσα προγραμματισμού είναι η Statement List (STL) που αναπτύχθηκε σχεδόν ταυτόχρονα με τη LADDER, αν και οι εταιρίες έδειξαν στην αρχή δισταγμό φοβούμενες μην τρομάξουν το τεχνικό κατεστημένο της βιομηχανίας. Η γλώσσα αυτή δημιουργεί λίστα προγράμματος με εντολές, που αντιστοιχούν στις λογικές πύλες (AND, OR NOT κτλ). Στην αρχή η γλώσσα αυτή ήταν πολύ φτωχή και περιοριζόταν μόνο στις βασικές Boolean εντολές. Στη συνέχεια οι γλώσσες αυτές αναπτύχθηκαν πολύ και συναντά κανείς σε αυτές στοιχεία από τις γλώσσες των υπολογιστών και κυρίως των γλωσσών Assembly. Ο προγραμματισμός σε αυτή τη γλώσσα απαιτεί από το χρήστη να έχει στοιχειώδεις γνώσεις προγραμματισμού.

Σχήμα 3.3.1.2 *Statement List*

3.3.1.3Διάγραμμα λογικών πυλών (FBD – Function Block Diagram)

Η τρίτη γλώσσα είναι η Function Block Diagram η οποία χρησιμοποιεί και αυτή γραφικά, αλλά αντί του ηλεκτρολογικού σχεδίου του αυτοματισμού χρησιμοποιεί το αντίστοιχο λογικό. Η γλώσσα αυτή είναι νεότερη και δεν χρησιμοποιείται από όλες τις εταιρίες.



Σχήμα 3.3.1.2 *FBD*

3.3.1.4 Πλεονεκτήματα και μειονεκτήματα λίστας εντολών (STL) σε σχέση με το σχέδιο επαφών (LAD) και το διάγραμμα λογικών πυλών (FBD)

Πλεονεκτήματα

- Έχει τις μεγαλύτερες δυνατότητες, γιατί υπάρχουν εντολές, οι οποίες δεν είναι δυνατόν να παρασταθούν γραφικά.
- Γνωρίζουμε με απόλυτη ακρίβεια τη σειρά, με την οποία ο μικροεπεξεργαστής επεξεργάζεται το πρόγραμμα (τη μία εντολή ύστερα από την άλλη).
- Καταλαμβάνει μικρότερο χώρο στη μνήμη για την αποθήκευση του προγράμματος.
- Είναι πολύ προσιτή στην χρήση σε όποιον έχει ασχοληθεί ήδη με προγραμματισμό κάθε είδους.
- Μπορούν να χρησιμοποιηθούν μικροί, φτηνοί, φορητοί προγραμματιστές χειρός (ενώ αντίθετα για τη «σχεδίαση» μιας γραφικής μορφής απαιτείται οθόνη, αν θέλουμε να έχουμε εποπτεία).
- Ο χειρισμός κατά την πληκτρολόγηση του προγράμματος είναι πολύ απλούστερος. Αντίθετα, για την πληκτρολόγηση ενός στοιχείου στις γραφικές μορφές, π.χ. μιας επαφής, πρέπει ο δείκτης (cursor) στην οθόνη να βρίσκεται στη σωστή θέση.
- Αν σαν βάση για τον προγραμματισμό χρησιμοποιηθεί ένα κλασσικό συνδεσμολογικό σχέδιο με ρελέ ή ένα λογικό διάγραμμα (flow-chart), τότε η «μετάφραση» τους σε λίστα εντολών είναι το ίδιο εύκολη με την «μετάφραση» τους σε σχέδιο επαφών ή λογικό διάγραμμα αντίστοιχα (αν όχι ευκολότερη πολλές φορές).
- Πρέπει να τονιστεί, ότι ένα ηλεκτρολογικό συνδεσμολογικό σχέδιο, πολύ σπάνια μπορεί να προγραμματιστεί όπως είναι, χωρίς μετατροπές, σε σχέδιο επαφών.

Μειονεκτήματα

- Ένα πρόγραμμα γραμμένο σε λίστα εντολών δεν έχει την ίδια εποπτεία «με μια ματιά», την οποία έχουν οι γραφικές μορφές. Με τις δυνατότητες όμως σχολιασμού προγράμματος, που παρέχουν οι σύγχρονες συσκευές προγραμματισμού, το μειονέκτημα αυτό παύει να είναι ιδιαίτερα σημαντικό.
- Η παρακολούθηση του αυτοματισμού σε λειτουργία (πάνω σε μια συσκευή προγραμματισμού οθόνης συνδεδεμένη στον ελεγκτή) είναι απλούστερη και πιο εποπτική, αν το πρόγραμμα είναι γραμμένο σε κάποια από τις δύο γραφικές μορφές.

3.3.2 Στάδια δημιουργίας ενός Project στο Simatic Manager

Στο Simatic S7 όλες οι απαιτήσεις σε Hardware και Software μιας διαδικασίας αυτοματισμού διαχειρίζονται μέσα από ένα project.

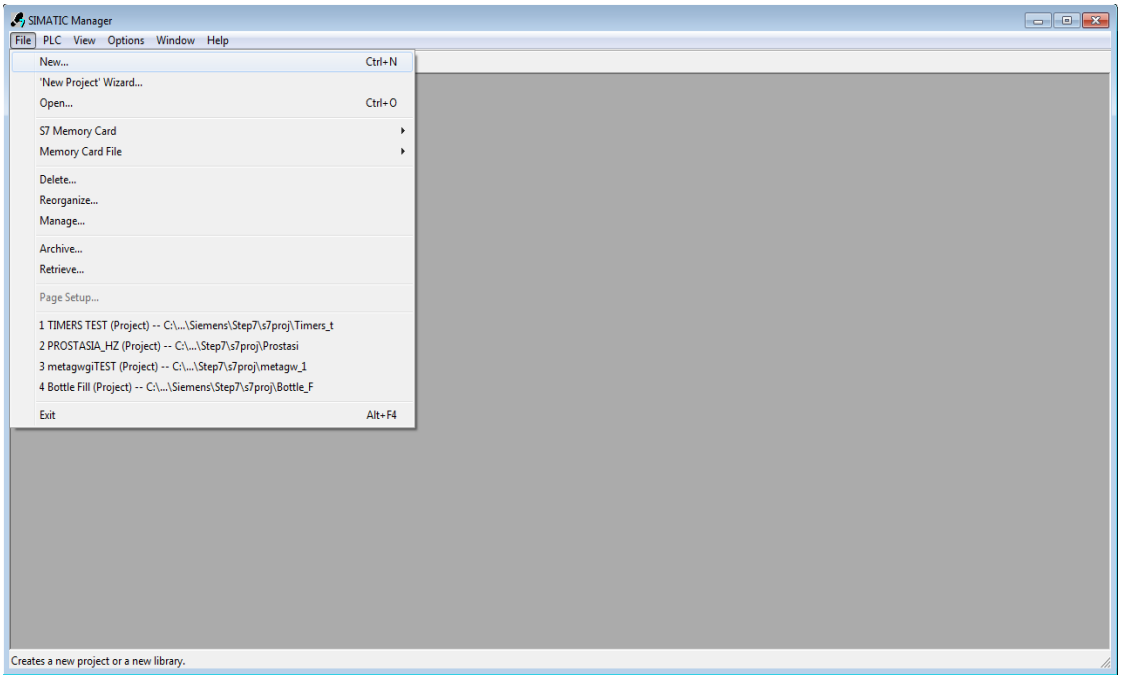
Ένα project περιλαμβάνει το απαραίτητο Hardware (με τη διαμόρφωση του υλικού), το δίκτυο (με τη διαμόρφωση του), όλα τα προγράμματα καθώς και ολόκληρη τη διαχείριση δεδομένων για μια λύση αυτοματισμού. Το Simatic Manager είναι το κύριο εργαλείο στο Step 7 αφού είναι αυτό που διαχειρίζεται τα project.

Σχήμα 3.3.2 *Simatic Manager*

Ένα STEP 7 project μπορεί να δημιουργηθεί με διαφορετική σειρά, ειτε ρυθμίζοντας αρχικά το hardware (hardware configuration) είτε στο τέλος αφού έχει αναπτυχθεί το πρόγραμμα.

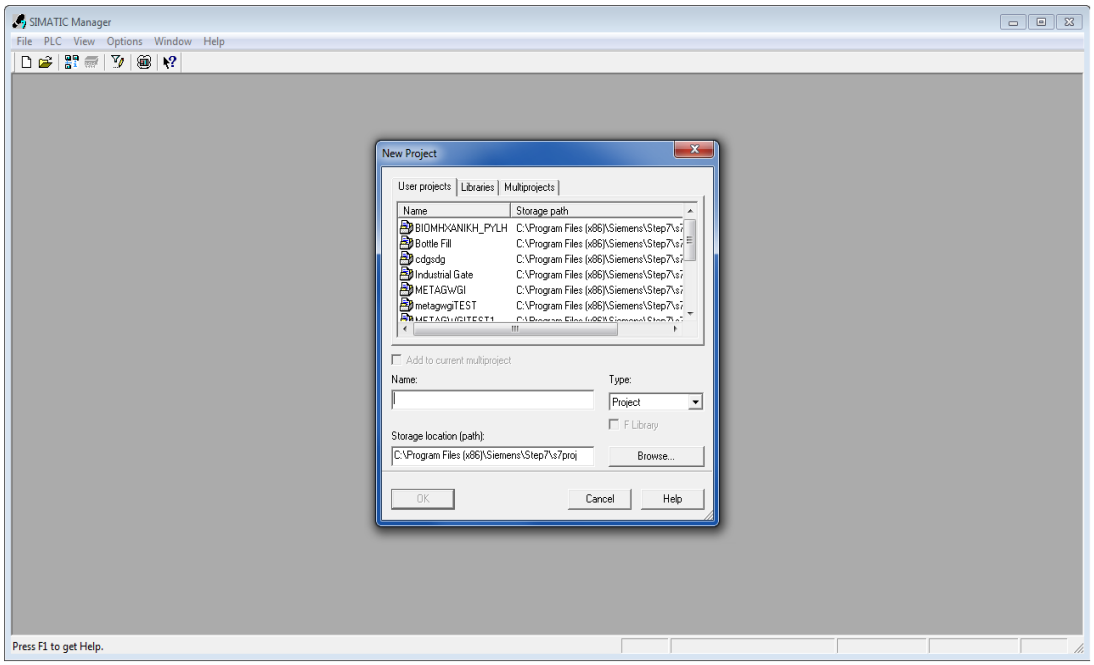
Για τη δημιουργία ενός πολύπλοκου προγράμματος με πολλές εισόδους και εξόδους, συνιστάται πρώτα να γίνεται το “hardware configuration”. Το πλεονέκτημα αυτής της ενέργειας είναι ότι στη διαδικασία του “hardware configuration” το STEP 7 εμφανίζει τις πιθανές διευθύνσεις εισόδων/εξόδων που μπορούν να χρησιμοποιηθούν κατά τον προγραμματισμό. Στην άλλη περίπτωση θα πρέπει να γίνουν οι δηλώσεις των εισόδων/εξόδων αυθαίρετα από τον χρήστη με αποτέλεσμα να μην είναι δυνατή η κλήση τους από το STEP 7.

Στο περιβάλλον του Simatic Manager επιλέγουμε **File > New** όπως φαίνεται στο παρακάτω σχήμα :



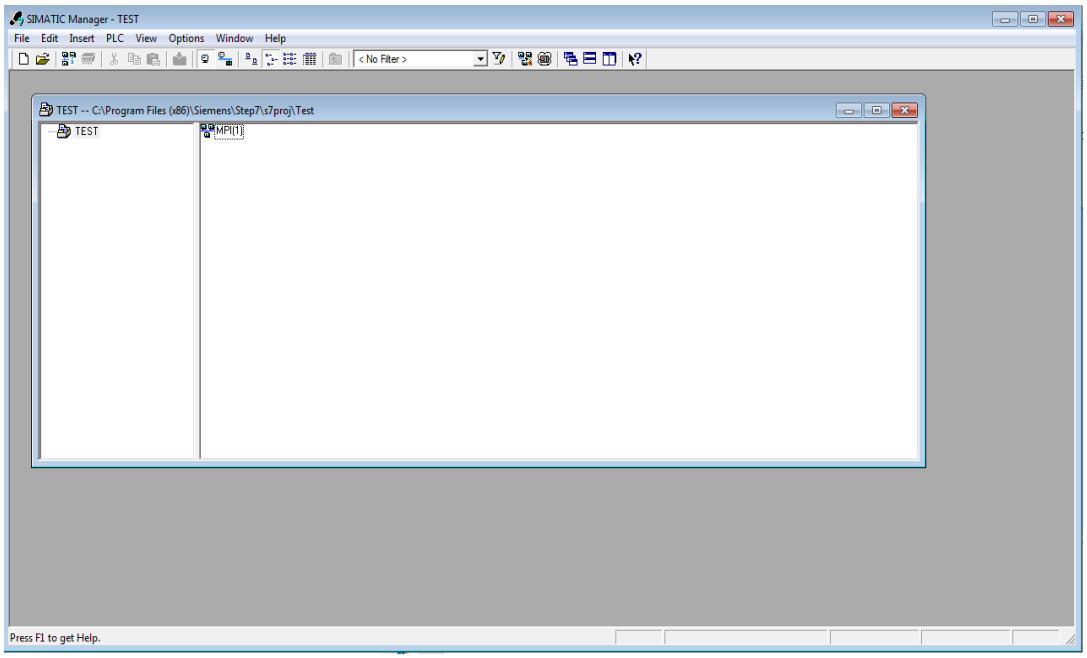
Σχήμα 3.3.2(Α) *Δημιουργία Project*

οπότε και εμφανίζεται το παρακάτω παράθυρο δημιουργίας νέας εφαρμογής στο οποίο ονομάζουμε το project και τον κατάλογο στον οποίο θα αποθηκευτεί στο σκληρό δίσκο :



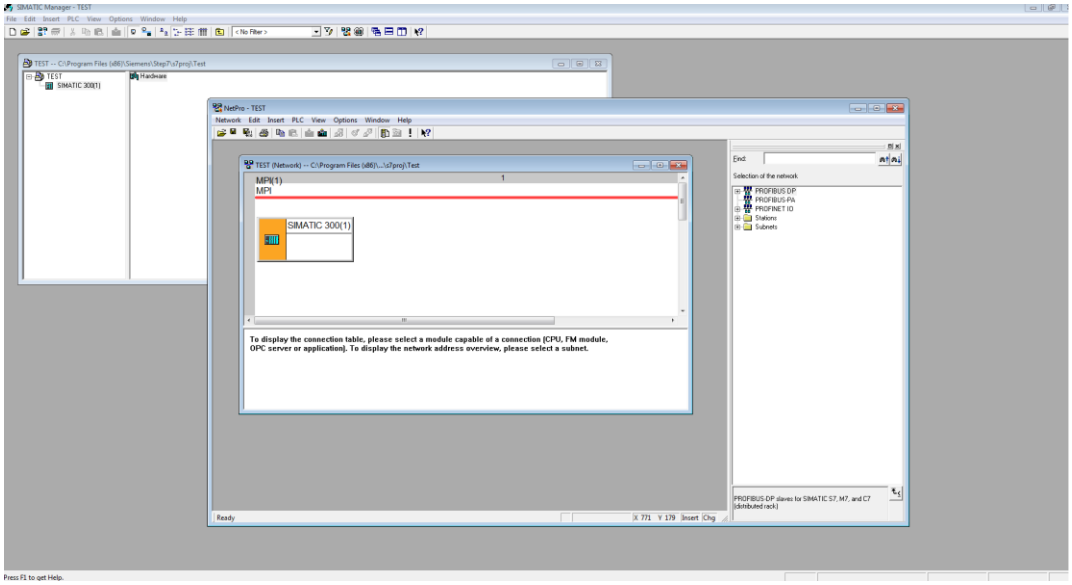
Σχήμα 3.3.2(B) *Ονομα Project*

Μετά την δημιουργία του project



Σχήμα 3.3.2(C) *Εισαγωγή Σταθμός*

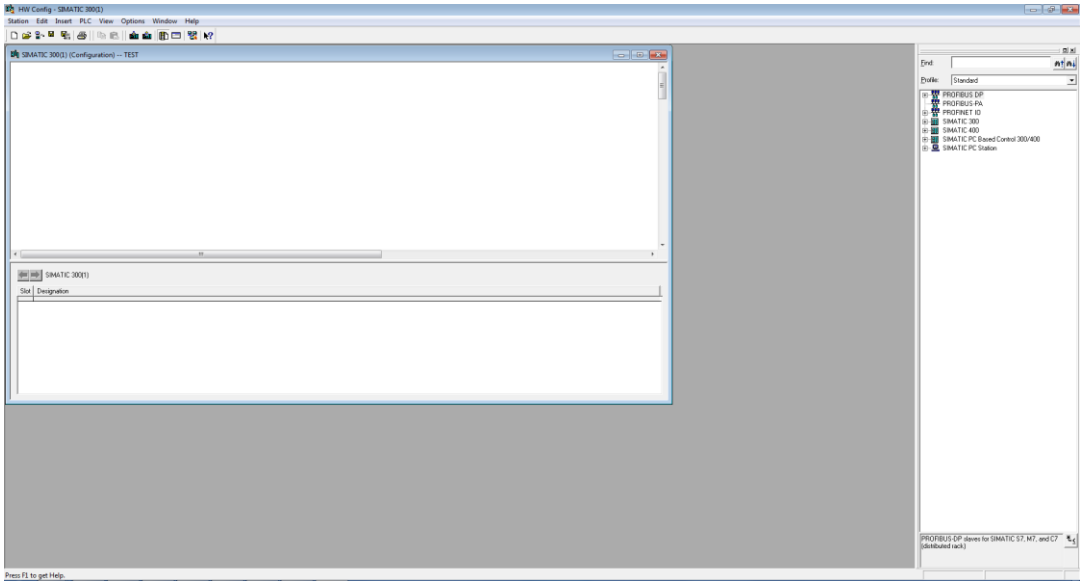
θα δηλώσουμε όλα τα PLC που υπάρχουν στην εγκατάστασή μας. Στο παράδειγμα μας θα φτιάξουμε έναν σταθμό S7-300. Αυτό γίνεται επιλέγοντας το project (κάνοντας κλικ στο όνομα του) και κατόπιν : **Insert New Object > SIMATIC 300 Station**



Σχήμα 3.3.2(D) *Σταθμός S7-300*

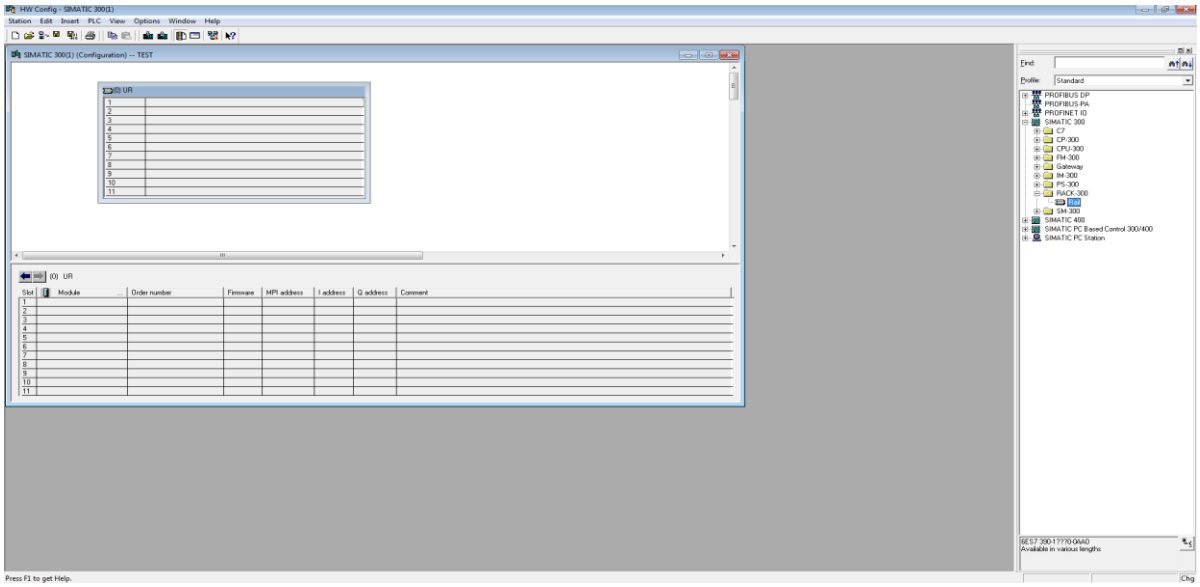
3.3.2.1 Ορισμός υλικού HardWare Config

Επιλέγοντας το όνομα του σταθμού εμφανίζεται ένα εικονίδιο με την ένδειξη Hardware. Με διπλό κλικ στο εικονίδιο αυτό καλούμε το πρόγραμμα HW Configurations. Το παράθυρο που εμφανίζεται αποτελείται από 3 περιοχές – τη σύντομη περιγραφή (πάνω αριστερά) , την αναλυτική (με κωδικό και διευθύνσεις, κάτω αριστερά) και τον κατάλογο του Hardware.



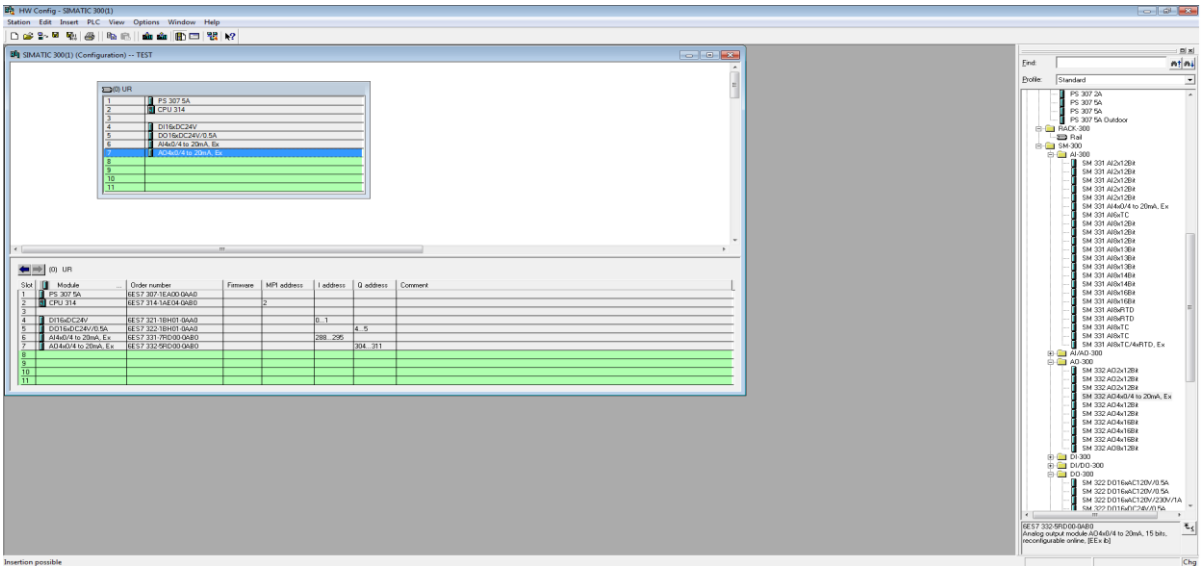
Σχήμα 3.3.2(E) *Hardware Configuration*

Από τον κατάλογο του Hardware επιλέγουμε πρώτα το rack οπού κατόπιν θα τοποθετήσουμε τις κάρτες (Simatic 300 – Rack 300 – Rail)



Σχήμα 3.3.2(F) *Rack*

Οι θέσεις πάνω στο rack είναι τυποποιημένες ,δηλαδή στην θέση 1 έχουμε το τροφοδοτικό , θέση 2 τη CPU, θέση 3 τη κάρτα διασύνδεσης αν υπάρχει και άλλο rack και θέσεις από 4 έως 11 κάρτες SM, FM και CP. Στο παρακάτω σχήμα φαίνεται το hardware ενός σταθμού PLC με το τροφοδοτικό του, την CPU και τις κάρτες ψηφιακών και αναλογικών εισόδων και εξόδων.



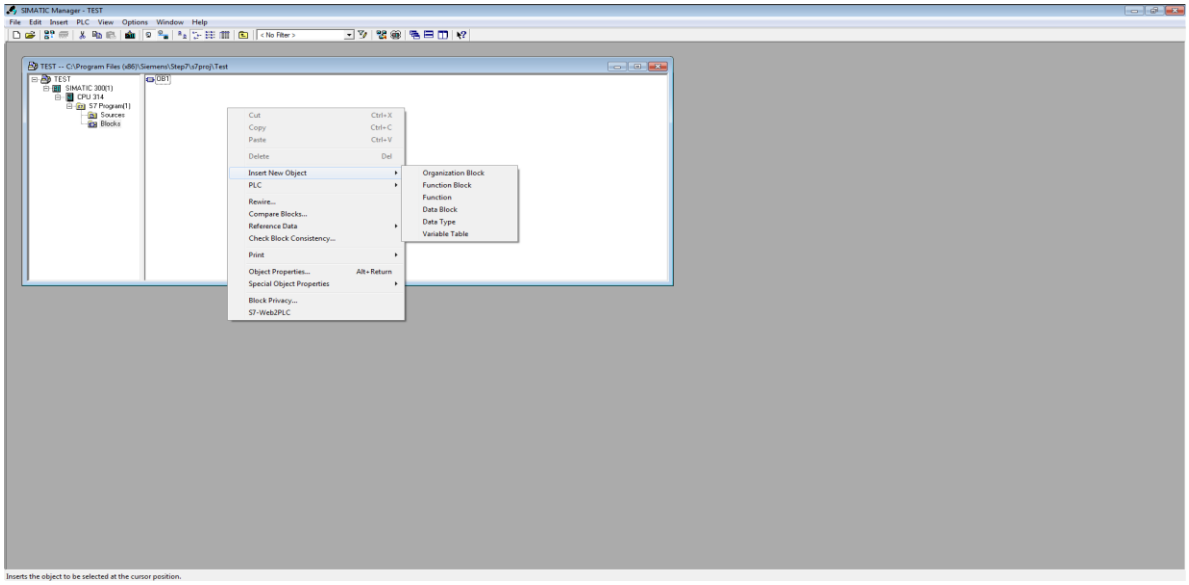
Σχήμα 3.3.2(G) *Ολοκληρωμένο Hardware*

Αφού έχουμε τελείωσει με την δημιουργία του Hardware , η επόμενη ενέργεια που πρέπει να κάνουμε είναι ο έλεγχος για τη σωστή τοποθέτηση και παραμετροποίηση του Hardware . Αυτό γίνεται επιλέγοντας **Station > Consistency Check**.

Εάν είναι όλα εντάξει αποθηκεύουμε τα δεδομένα στο σκληρό δίσκο επιλέγοντας **Station > Save and Compile** .Επόμενη κίνηση είναι να «κατεβάσουμε» (Download) το Hardware στο PLC .

3.3.2.2Δημιουργία block στο S7

Μετά τον ορισμό του Hardware είμαστε σε θέση να γράψουμε το πρόγραμμα για την υλοποίηση του αυτοματισμού. Στο project που έχουμε δημιουργήσει επιλέγουμε το εικονίδιο Blocks όπως φαίνεται στην παρακάτω εικόνα. Εδώ υπάρχει ήδη το OB1 που είναι ένα ειδικό block το οποίο εκτελείται κυκλικά και είναι το μόνο που ελέγχει άμεσα ο επεξεργαστής του PLC. Εάν θέλουμε να δημιουργήσουμε και άλλα block για την ευκολότερη επεξεργασία του προγράμματος μας επιλέγουμε Blocks δεξί κλικ με το ποντίκι **Insert New Object** όπως φαίνεται στην παρακάτω εικόνα .



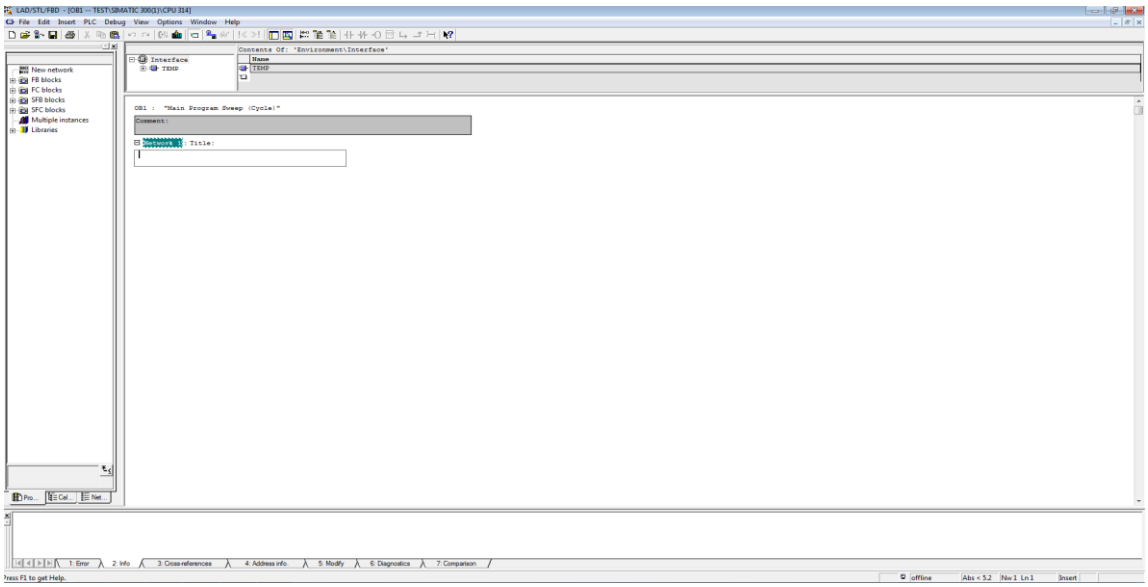
Σχήμα 3.3.2(Η) *Εισαγωγή Block*

Εδώ έχουμε να επιλέξουμε μια από τις προτεινόμενες επιλογές:

- Organizations Block (OB)
- Function (FC)
- Function Block (FB)
- Data Block (DB)
- Data Type (UDT)
- Variable Table (VAT)

3.3.2.3Δημιουργία κώδικα στο S7

Η προσθήκη κώδικα στο OB1, στα FB και στις FC γίνεται κάνοντας διπλό κλικ πάνω τους, οπότε καλείται από το σύστημα η εφαρμογή LAD/STL/FBD η οποία αποτελεί έναν συντάκτη εντολών (editor).



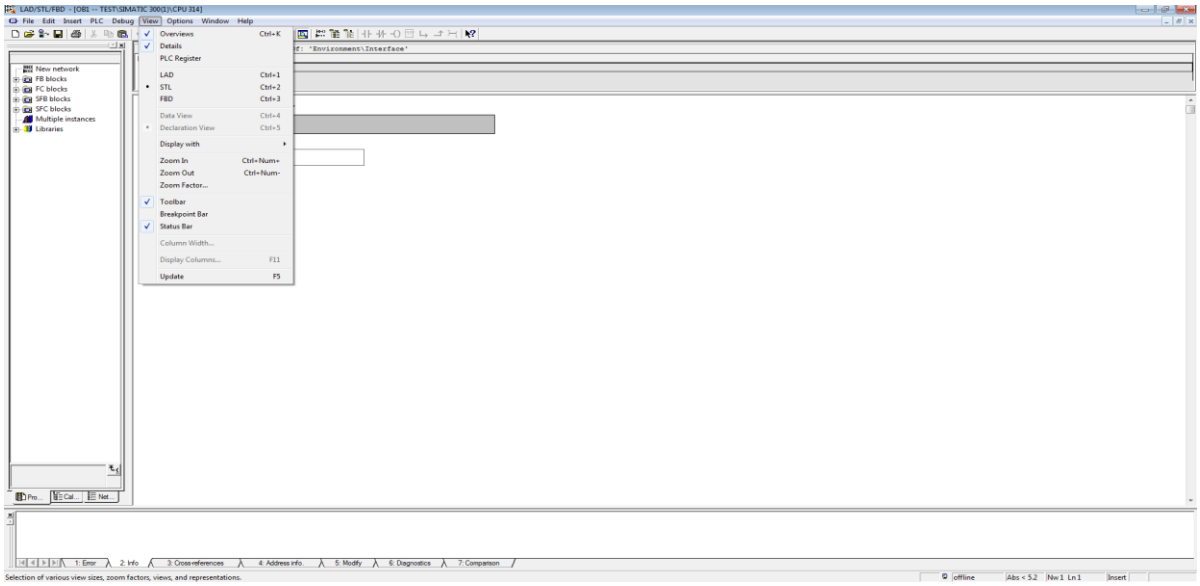
Σχήμα 3.3.2.3 *Συντάκτης εντολών*

Η συγκεκριμένη εφαρμογή είναι εξαιρετικά «δυνατή», αφού εκτός των άλλων έχει τη δυνατότητα σύνδεσης και on-line παρακολούθησης του προγράμματος, όπως αυτό εκτελείται στο PLC .

Αυτό γίνεται επιλέγοντας **Debug > Monitor** ή επιλέγοντας το εικονίδιο με τα γυαλλάκια από το Toolbar.

Για τη δημιουργία κώδικα υπάρχουν οι τρεις γλώσσες προγραμματισμού όπου αναφέραμε παραπάνω, που μπορούν να χρησιμοποιηθούν σύμφωνα με τις προτιμήσεις και τη γνώση μας.

Επιλέγοντας τώρα τη γλώσσα προγραμματισμού με την οποία θέλουμε να γράψουμε των κώδικα , αυτό γίνεται επιλέγοντας **View > LAD / STL / FBD** όπως φαίνεται στην παρακάτω εικόνα:

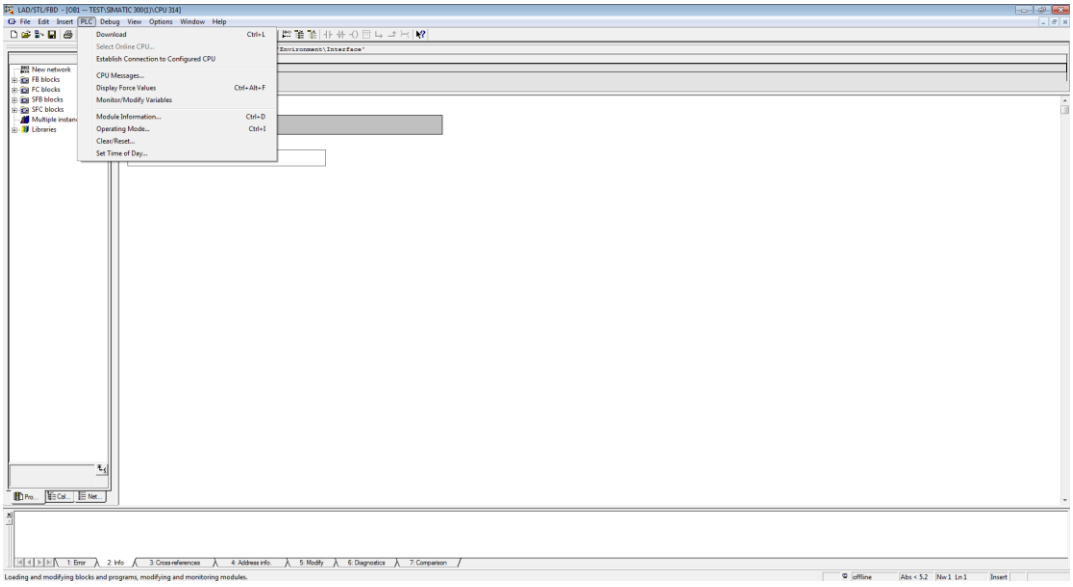


Σχήμα 3.3.2.3(Α) *Επιλογή γλώσσας προγραμματισμού*

Μετά την δημιουργία του προγράμματος μας το αποθηκεύουμε στο σκληρό δίσκο επιλέγοντας **File > Save** Αν υπάρχουν τυχόν λάθη στη σύνταξη των εντολών του προγράμματος μας το STEP 7 δεν μας αφήνει να το αποθηκεύουμε.

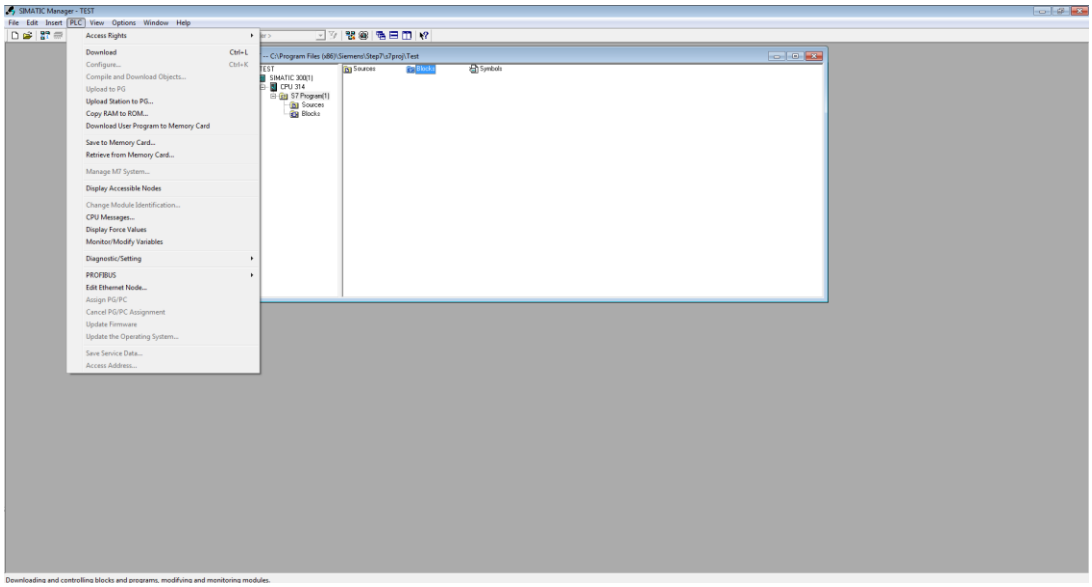
3.3.2.4Μεταφορά του προγράμματος στο PLC

Η δημιουργία των Blocks και ο κώδικας εντολών που έχουμε γράψει μέχρι τώρα είναι στον υπολογιστή μας , το επόμενο βήμα είναι τα μεταφέρουμε στο PLC. Αυτό επιτυγχάνεται είτε από τον LAD/STL/FBD Editor επιλέγοντας **PLC > Download**



Σχήμα 3.3.2.4 *Μεταφορά προγράμματος*

Είτε από το Simatic Manager και εφόσον έχουμε επιλέξει τον κατάλογο Blocks ξανά με την εντολή **PLC > Download.**



Σχήμα 3.3.2.4(Α) *Μεταφορά προγράμματος*

Επιλέγοντας το δεύτερο τρόπο μεταφέρουμε στο PLC όλα Block που έχουμε δημιουργήσει μέσα στο project , ενώ με την πρώτη επιλογή το συγκεκριμένο Block που έχουμε ανοιχτό.

4Ο ΚΕΦΑΛΑΙΟ

"ΠΑΡΑΔΕΙΓΜΑΤΑ ΕΦΑΡΜΟΓΩΝ ΜΕ SIMATIC S7"

4.1 Εφαρμογή 1η

ΣΥΣΤΗΜΑ ΠΛΗΡΩΣΗΣ ΚΑΙ ΑΠΑΡΙΘΜΙΣΗΣ ΜΠΟΥΚΑΛΙΩΝ

4.1.1 Παρουσίαση απαιτήσεων εφαρμογής

Σε μια βιομηχανική μονάδα κρασιού λειτουργεί 1 μηχανή παραγωγής, για την αύξηση της παραγωγής πρέπει να εγκατασταθεί ένα σύστημα πλήρωσης, για την μεταφορά και το γέμισμα των μπουκαλιών.

Το σύστημα θα αποτελείται απο 1 ταινιόδρομο Α που κινείται απο τον κινητήρα Μ1. Στην αρχή στην μέση και στο τέλος του ταινιόδρομου θα υπάρχουν φωτοκύτταρα S1,S2 και S3. Επίσης στην μέση του ταινιόδρομου θα υπάρχει σύστημα πλήρωσης που όταν θα ανιχνεύεται μπουκάλι απο το αντίστοιχο φωτοκύτταρο τότε ο ταινιόδρομος θα σταματάει και θα ξεκινάει η διαδικασία της πλήρωσης. Όταν αυτή ολοκληρωθεί τότε ο ταινιόδρομος θα ξεκινάει και παλι και η διαδικασία θα επαναλαμβάνεται.

Οι συνθήκες κίνησης λειτουργίας και πλήρωσης θα πρέπει να είναι οι ακόλουθες:

1. Το σύστημα θα πρέπει να έχει ένα γενικό START/STOP για την ενεργοποίηση και απενεργοποίηση του συστήματος.
2. Να υπάρχει ενδεικτική λυχνία για την κατάσταση λειτουργίας της εγκατάστασης
3. Να υπάρχει δυνατότητα επιλογής τρόπου λειτουργίας της εγκατάστασης, δηλαδή ειτε αυτόματο είτε χειροκίνητο, ο οποίος θα γίνεται αποδεκτός με το πάτημα BUTTON.
4. Σε περίπτωση χειροκίνητης λειτουργίας θα υπάρχουν δύο BUTTON ένα για την κίνηση της μεταφορικής ταινίας προς τα εμπρός και ένα για την κίνηση της προς τα πίσω.
5. Θα πρέπει επίσης να γίνεται καταγραφή άδειων και γεμάτων μπουκαλιών.

Ζητούνται

a) Να συνταχθούν οι πίνακες εισόδων – εξόδων

b) Να συνταχθεί πρόγραμμα P.L.C σε γλώσσα LADDER στο περιβάλλον Simatic Manager S7

4.1.2 Πίνακες εισόδων / εξόδων

Πίνακας εισόδων

α/α	**ΠΕΡΙΓΡΑΦΗ**	**ΔΙΕΥΘΥΝΣΗ ΕΙΣΟΔΩΝ**	**ΤΥΠΟΣ**
1.	Start Εγκατάστασης	I0.0	BOOL
2.	Stop Εγκατάστασης	I0.1	BOOL
3.	Επιλογή λειτουργίας	I0.2	BOOL
4.	Επιβεβαίωση τρόπου λειτουργίας	I0.3	BOOL
5.	Button κίνησης προς τα μπροστά	I0.4	BOOL
6.	Button κίνησης προς τα πίσω	I0.5	BOOL
7.	Φωτοκύτταρο πλήρωσης	I0.6	BOOL
8.	Φωτοκύτταρο ανίχνευσης μπουκαλιού στην είσοδο	I1.0	BOOL
9	Φωτοκύτταρο ανίχνευσης μπουκαλιού στην έξοδο	I1.1	BOOL

Πίνακας Εξόδων

α/α	**ΠΕΡΙΓΡΑΦΗ**	**ΔΙΕΥΘΥΝΣΗ ΕΞΟΔΩΝ**	**ΤΥΠΟΣ**
1.	Κατάσταση εγκατάστασης	Q0.0	BOOL
2.	Ενδεικτική Λυχνία	Q0.1	BOOL
3.	Χειροκίνητος τρόπος λειτουργίας	Q0.2	BOOL
4.	Αυτόματος τρόπος λειτουργίας	Q0.3	BOOL
5.	Κίνηση ταινίας προς τα μπροστά	Q0.4	BOOL
6.	Κίνηση ταινίας προς τα πίσω	Q0.5	BOOL
7.	Πλήρωση	Q1.0	BOOL

4.1.3 Πρόγραμμα σε Ladder

OB1 - <offline>
""
Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 11/12/2017 05:15:04 PM
Interface: 02/15/1996 04:51:12 PM
Lengths (block/logic/data): 00128 00016 00022

Name	Data Type	Address	Comment
TEMP		0.0	
OB1_EV_CLASS	Byte	0.0	Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
OB1_SCAN_1	Byte	1.0	1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
OB1_PRIORITY	Byte	2.0	Priority of OB Execution
OB1_OB_NUMBR	Byte	3.0	1 (Organization block 1, OB1)
OB1_RESERVED_1	Byte	4.0	Reserved for system
OB1_RESERVED_2	Byte	5.0	Reserved for system
OB1_PREV_CYCLE	Int	6.0	Cycle time of previous OB1 scan (milliseconds)
OB1_MIN_CYCLE	Int	8.0	Minimum cycle time of OB1 (milliseconds)
OB1_MAX_CYCLE	Int	10.0	Maximum cycle time of OB1 (milliseconds)
OB1_DATE_TIME	Date_And_Time	12.0	Date and time OB1 started

Block: OB1 "Main Program Sweep (Cycle)"

Network: 1

```
CALL  FC    1
```

FC1 - <offline>

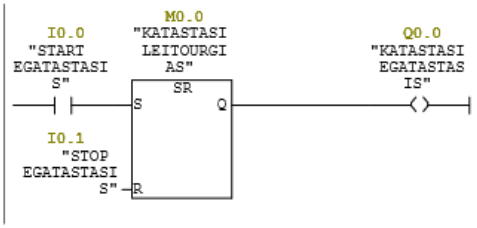
""

Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 11/19/2017 12:46:54 PM
Interface: 11/12/2017 12:46:41 PM
Lengths (block/logic/data): 00388 00260 00000

Name	Data Type	Address	Comment
IN		0.0	
OUT		0.0	
IN_OUT		0.0	
TEMP		0.0	
RETURN		0.0	
RET_VAL		0.0	

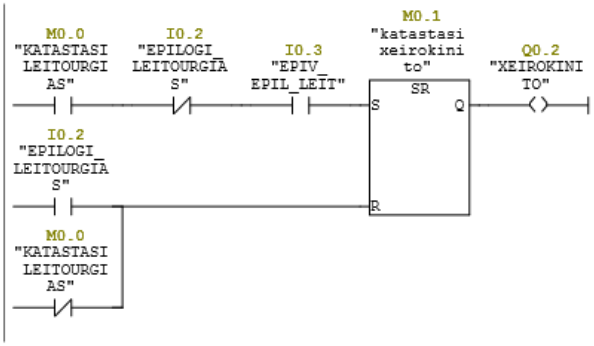
Block: FC1

Network: 1 KATASTASI EGATASTASIS

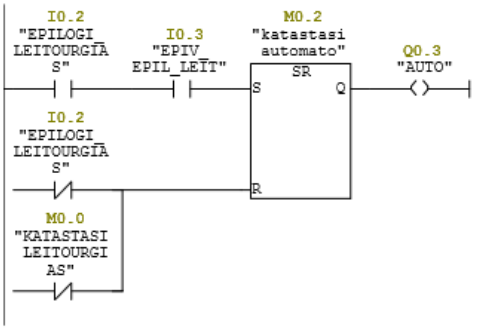


Network: 2 ENDEIKTIKH LUXNIA EGKATASTASIS

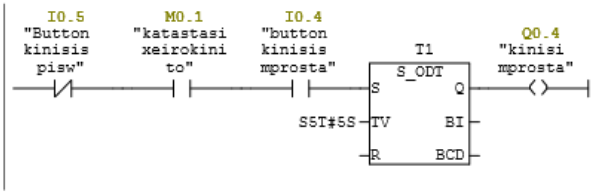
Network: 3 XEIROKINITO



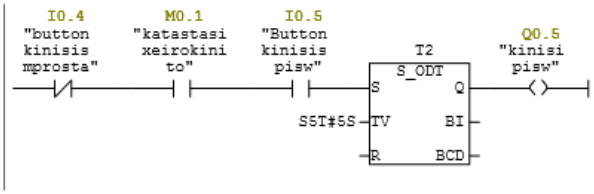
Network: 4 AYTOMATO



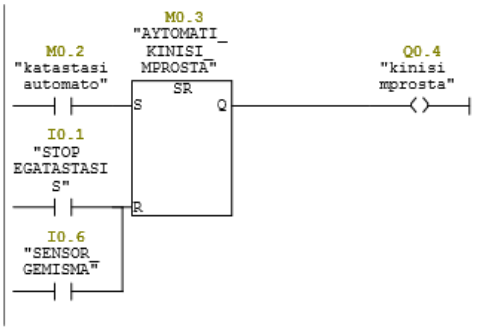
Network: 5 kinisi metaforikis tainias pros ta mprosta sto xeirokinito



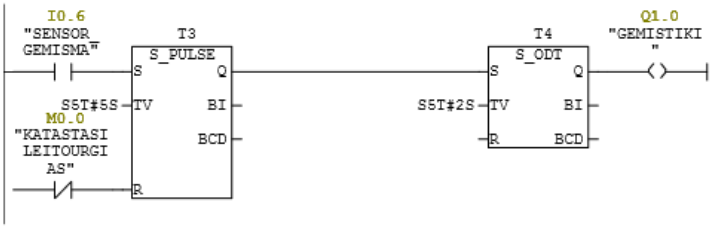
Network: 6 kinisi metaforikis tainias pros ta pisw sto xeirokinito



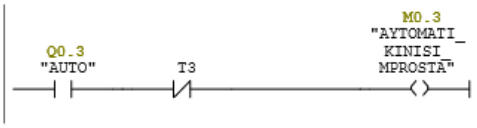
Network: 7 AYTOMATH LEITOURGIA



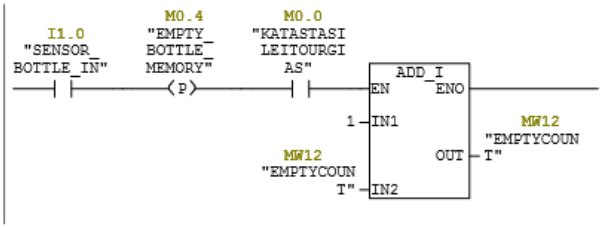
Network: 8 PLIRWSI



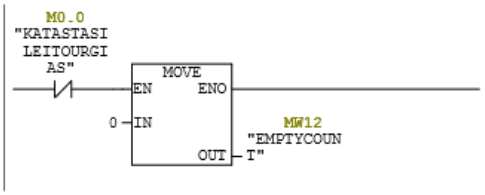
Network: 9 EKKINISI METAFORIKIS TAINIAS META TO GEMISMA



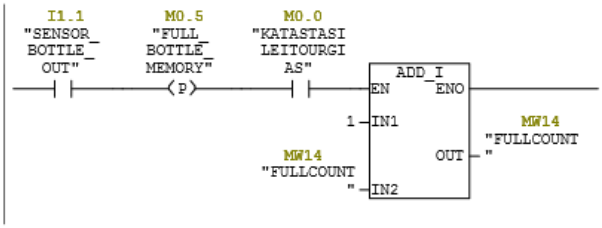
Network: 10 KATAGRAFI ADEIWN MPOYKALIWN



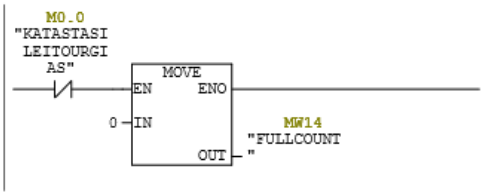
Network: 11 MIDENISMOS COUNTER ADEIWN MPOUKALIWN



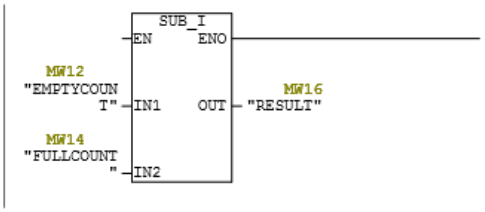
Network: 12 KATAGRAFI GEMATWN MPOUKALIWN



Network: 13 MIDENISMOS GEMATWN MPOUKALIWN



Network: 14 APOTELESMA



4.2 Εφαρμογή 2η

ΧΩΡΟΣ ΣΤΑΘΜΕΥΣΗΣ ΑΥΤΟΚΙΝΗΤΩΝ

4.2.1 Παρουσίαση απαιτήσεων εφαρμογής

Σε ενα κλειστό χώρο στάθμευσης «50» αυτοκινήτων πρέπει να εγκατασταθεί αυτόματο σύστημα καταμέτρησης εισαρχομένων και εξαρχομένων οχημάτων για την καλύτερη εξυπηρέτηση των πελατών. Το σύστημα θα αποτελείται απο δύο μπάρες μια εισόδου και μια εξόδου οι οποίες θα κινούνται απο τους κινητήρες M1 και M2 αντίστοιχα. Πριν και μετα από κάθε μπάρα θα υπάρχουν τα φωτοκύτταρα S1, S2, S3 και S4.

Οι συνθήκες λειτουργίας του χώρου στάθμευσης θα πρέπει να είναι οι εξής:

1. Για την ενεργοποίηση και απενεργοποίηση του συστήματος θα υπάρχει ένα START και ενα STOP.
2. Να υπάρχει σηματοδότηση τόσο για την είσοδο όσο και την έξοδο των οχημάτων
3. Για την καλύτερη λειτουργία θα πρέπει να καταμετρώνται τα αυτοκίνητα που εισέρχονται και εξέρχονται από τον χώρο.
4. Για να αποκλειστεί το ενδεχόμενο συνωστισμού αυτοκινήτων στην είσοδο του χώρου να υπάρχει αντίστοιχη ένδειξη που θα υποδεικνύει την πληρότητα του χώρου.

Ζητούνται

a) Να συνταχθούν οι πίνακες εισόδων – εξόδων

b) Να συνταχθεί πρόγραμμα P.L.C σε γλώσσα LADDER στο περιβάλλον Simatic Manager S7

4.2.2 Πίνακες εισόδων / εξόδων

4.2.2.1Πίνακας εισόδων

Πίνακας 4.1.2.1

α/α	ΠΕΡΙΓΡΑΦΗ	ΔΙΕΥΘΥΝΣΗ ΕΙΣΟΔΩΝ	ΤΥΠΟΣ
1.	Λειτουργία Πάρκινγκ	I0.0	BOOL
2.	Είσοδος αυτοκινήτων	I0.1	BOOL
3.	Άνω τερματικό μπάρας εισόδου	I0.2	BOOL
4.	Φωτοκύτταρο εισόδου μετά την μπάρα	I0.3	BOOL
5.	Κάτω τερματικό μπάρας εισόδου	I0.4	BOOL
6.	Έξοδος αυτοκινήτων	I1.0	BOOL
7.	Άνω τερματικό μπάρας εξόδου	I1.1	BOOL
8.	Φωτοκύτταρο εξόδου μετά την μπάρα	I1.2	BOOL
9.	Κάτω τερματικό μπάρας εξόδου	I1.3	BOOL

4.2.2.2Πίνακας Εξόδων

Πίνακας 4.2.2.2

α/α	**ΠΕΡΙΓΡΑΦΗ**	**ΔΙΕΥΘΥΝΣΗ ΕΞΟΔΩΝ**	**ΤΥΠΟΣ**
1.	Λειτουργία Πάρκινγκ	I0.0	BOOL
2.	Είσοδος αυτοκινήτων	I0.1	BOOL
3.	Άνω τερματικό μπάρας εισόδου	I0.2	BOOL
4.	Φωτοκύτταρο εισόδου μετά την μπάρα	I0.3	BOOL
5.	Κάτω τερματικό μπάρας εισόδου	I0.4	BOOL
6.	Έξοδος αυτοκινήτων	I1.0	BOOL
7.	Άνω τερματικό μπάρας εξόδου	I1.1	BOOL
8.	Φωτοκύτταρο εξόδου μετά την μπάρα	I1.2	BOOL
9.	Κάτω τερματικό μπάρας εξόδου	I1.3	BOOL

4.2.3 Πρόγραμμα σε LADDER

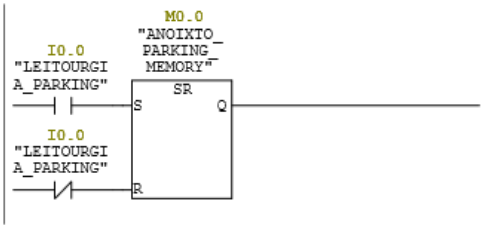
OB1 - <offline>
""
Name: Family:
Author: Version: 0.1
Block version: 2
Time stamp Code: 11/26/2017 10:13:09 AM
Interface: 02/15/1996 04:51:12 PM
Lengths (block/logic/data): 00144 00028 00022

Object properties:
S7_language

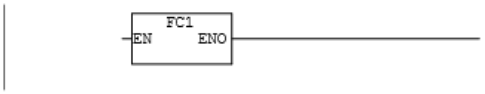
Name	Data Type	Address	Comment
TEMP		0.0	
OB1_EV_CLASS	Byte	0.0	Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
OB1_SCAN_1	Byte	1.0	1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
OB1_PRIORITY	Byte	2.0	Priority of OB Execution
OB1_OB_NUMBR	Byte	3.0	1 (Organization block 1, OB1)
OB1_RESERVED_1	Byte	4.0	Reserved for system
OB1_RESERVED_2	Byte	5.0	Reserved for system
OB1_PREV_CYCLE	Int	6.0	Cycle time of previous OB1 scan (milliseconds)
OB1_MIN_CYCLE	Int	8.0	Minimum cycle time of OB1 (milliseconds)
OB1_MAX_CYCLE	Int	10.0	Maximum cycle time of OB1 (milliseconds)
OB1_DATE_TIME	Date_And_Time	12.0	Date and time OB1 started

Block: OB1 "Main Program Sweep (Cycle)"

Network: 1



Network: 2 CALL FUCTION1



FC1 - <offline>

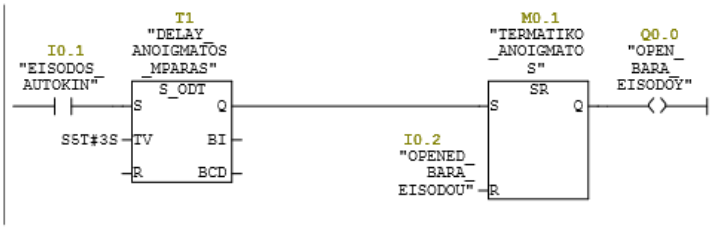
""

Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 11/26/2017 10:12:04 AM
Interface: 11/05/2017 12:26:50 PM
Lengths (block/logic/data): 00396 00270 00002

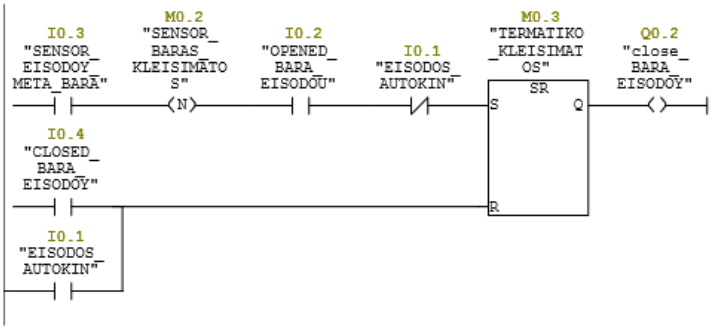
Name	Data Type	Address	Comment
IN		0.0	
OUT		0.0	
IN_OUT		0.0	
TEMP		0.0	
RETURN		0.0	
RET_VAL		0.0	

Block: FC1

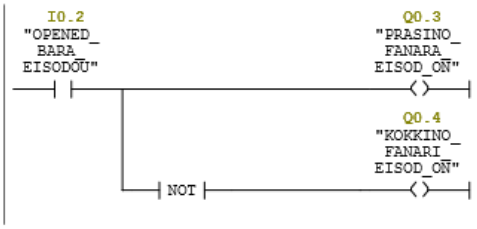
Network: 1 ANOIGMA_BARAS_EISODOY



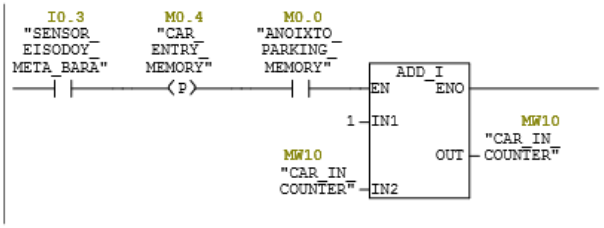
Network: 2 KLEISIMO MPARAS EISODOU



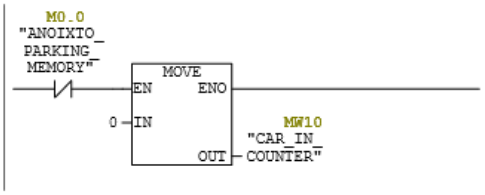
Network: 3 FWTEINI SIMATODOTISI EISODOU



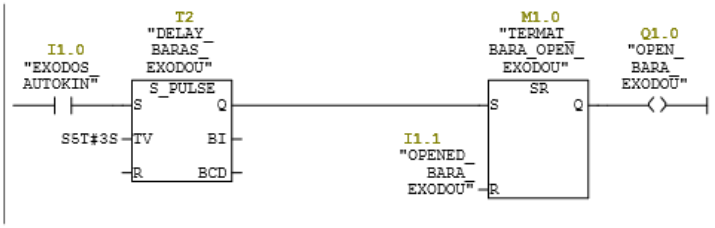
Network: 4 OXIMATA POU EISILTHAN STO PARKING



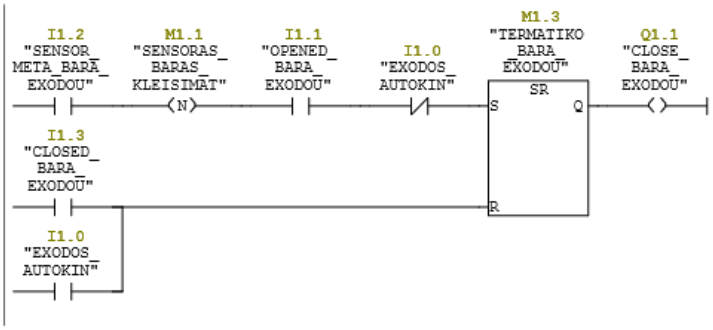
Network: 5 MIDENISMOS AUTOKINITWN POU EISILTHAN



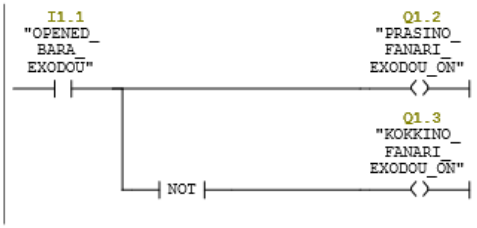
Network: 6 ANOIGMA MPARAS EXODOU



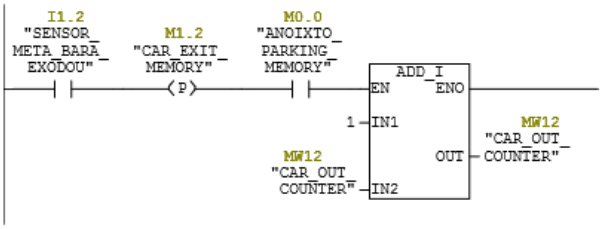
Network: 7 KLEISIMO MPARAS EXODOU



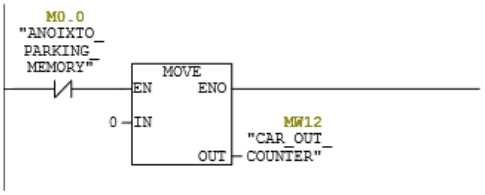
Network: 8 FWTEINI SIMATODOTISI EXODOU



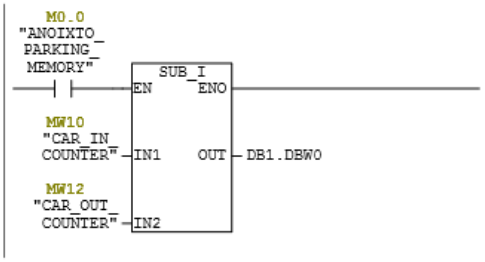
Network: 9 OXIMATA POU EKSILTHAN APO TO PARKING



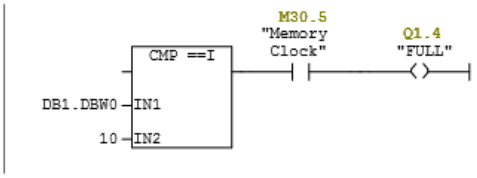
Network: 10 MIDENISMOS OXIMATOS POU EKSILTHAN APO TO PARKING



Network: 11 ARITHMOS OXIMATWN POU BRISKONTAI MESA STO PARKING



Network: 12 FWTEINI ENDEIKSI PARKING FULL



4.3 Εφαρμογή 3η

ΒΙΟΜΗΧΑΝΙΚΗ ΠΥΛΗ

4.3.1 Παρουσίαση απαιτήσεων εφαρμογής

Σε μια βιομηχανία για την ασφαλέστερη είσοδο και έξοδο οχημάτων θα εγκατασταθεί σύστημα για τον αυτόματο έλεγχο της πύλης. Το σύστημα θα αποτελείται από την πύλη, η οποία θα ελέγχεται μέσω κινητήρα Μ1, καθώς επίσης και απο μια δοκό ασφαλείας.

Οι συνθήκες λειτουργίας της πύλης θα πρέπει να είναι οι ακόλουθες:

1. Θα υπάρχουν δύο διακόπτες με τους οποίους θα ελέγχεται η λειτουργία της πύλης
2. Κατά την διάρκεια κίνησης της πύλης θα πρέπει να υπάρχει αντίστοιχη ενδεικτική λυχνία
3. Θα υπάρχει ένα STOP με το οποίο η πύλη θα σταματά σε οποιοδήποτε σημείο και αν βρίσκεται.
4. Για την αποφυγή λάθους οι διακόπτες που ανοίγουν και κλείνουν την πύλη θα ενεργοποιούνται μόνο όταν αυτή δεν κινείται προς την αντίθετη κατεύθυνση.
5. Για να αποτρεπεί τυχόν τραυματισμός και παγίδευση ή καταστροφή αντικειμένων η κίνηση της πύλης θα σταματά άταν η δοκός πίεσης ασφαλείας ενεργοποιηθεί.

Ζητούνται

a) Να συνταχθούν οι πίνακες εισόδων – εξόδων
b) Να συνταχθεί πρόγραμμα P.L.C σε γλώσσα LADDER στο περιβάλλον Simatic Manager S7

4.3.2 Πίνακες εισόδων εξόδων

4.3.2.1Πίνακας Εισόδων

Πίνακας 4.3.2.1

α/α	ΠΕΡΙΓΡΑΦΗ	ΔΙΕΥΘΥΝΣΗ ΕΙΣΟΔΩΝ	ΤΥΠΟΣ
1.	Διακόπτης ανοίγματος πύλης	I0.0	BOOL
2.	Διακόπτης κλεισίματος πύλης	I0.1	BOOL
3.	Button Stop	I0.2	BOOL
4.	Τερματικό ανοίγματος πυλης	I0.3	BOOL
5.	Τερματικό κλεισίματος πυλης	I0.4	BOOL
6.	Δοκός ασφαλείας	I0.5	BOOL

4.3.2.2Πίνακας Εξόδων

Πίνακας 4.3.2.2

α/α	ΠΕΡΙΓΡΑΦΗ	ΔΙΕΥΘΥΝΣΗ ΕΞΟΔΩΝ	ΤΥΠΟΣ
1.	Άνοιγμα πύλης	Q0.0	BOOL
2.	Κλείσιμο πύλης	Q0.1	BOOL
3.	Ενδεικτική λυχνία	Q0.2	BOOL

4.3.3 Πρόγραμμα σε LADDER

OB1 - <offline>

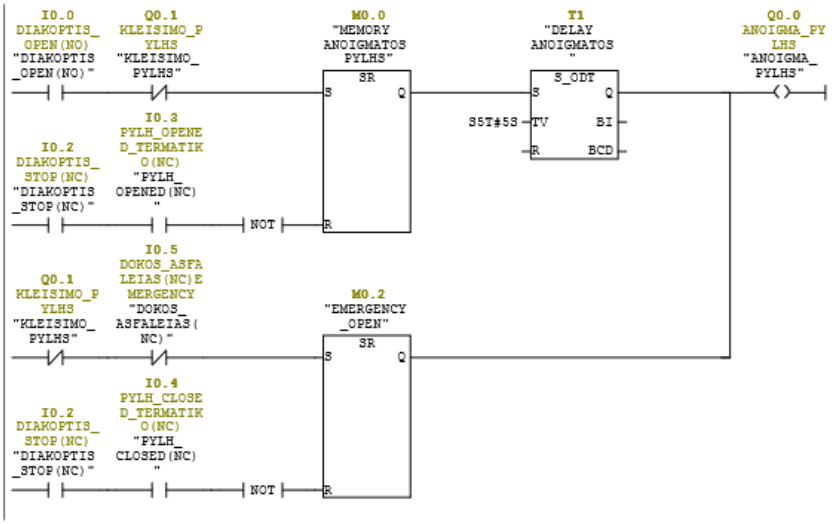
""

Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 03/19/2018 09:19:07 PM
Interface: 02/15/1996 04:51:12 PM
Lengths (block/logic/data): 00230 00114 00020

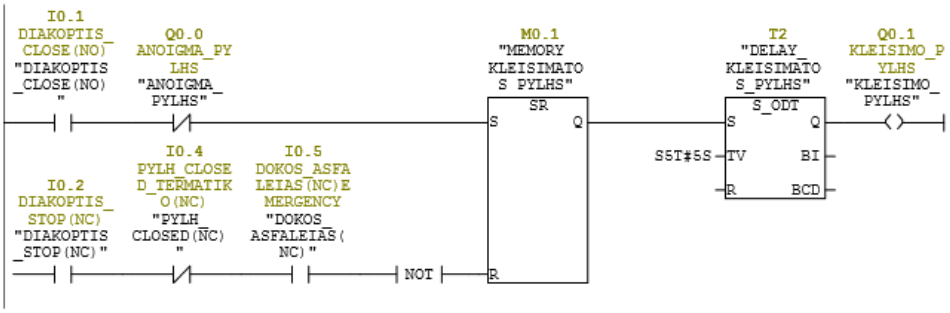
Name	Data Type	Address	Comment
TEMP		0.0	
OB1_EV_CLASS	Byte	0.0	Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
OB1_SCAN_1	Byte	1.0	1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
OB1_PRIORITY	Byte	2.0	Priority of OB Execution
OB1_OB_NUMBR	Byte	3.0	1 (Organization block 1, OB1)
OB1_RESERVED_1	Byte	4.0	Reserved for system
OB1_RESERVED_2	Byte	5.0	Reserved for system
OB1_PREV_CYCLE	Int	6.0	Cycle time of previous OB1 scan (milliseconds)
OB1_MIN_CYCLE	Int	8.0	Minimum cycle time of OB1 (milliseconds)
OB1_MAX_CYCLE	Int	10.0	Maximum cycle time of OB1 (milliseconds)
OB1_DATE_TIME	Date_And_Time	12.0	Date and time OB1 started

Block: OB1 "Main Program Sweep (Cycle)"

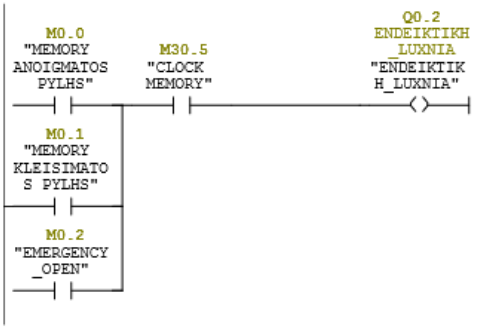
Network: 1 ANOIGMA PYLHS



Network: 2 KLEISIMO PYLHS



Network: 3 ENDEIKTIKH LUXNIA ANOIGMATOS/KLEISIMATOS PYLHS



4.4 Εφαρμογή 4η

ΣΥΣΤΗΜΑ ΠΑΡΑΓΩΓΗΣ ΕΝΕΡΓΕΙΑΣ

4.4.1 Παρουσίαση απαιτήσεων εφαρμογής

Σε ενα εργοστάσιο παραγωγής ενέργειας πρόκειται να εγκατασταθούν δύο γεννήτριες για την καλύτερη λειτουργία του εργοστασίου τόσο σε περίπτωση βλάβης όσο και σε περίπτωση συντήρησης.

Τα χαρακτηρηστικά των δύο γεννητριών είναι :

- Γεννήτρια 1η : 3Φ, 650kVA, 50Hz
- Γεννήτρια 2η 3Φ, 250kVA, 50H

Η εκκίνηση κάποιας εκ των δύο γεννητριών θα γίνεται όταν η παροχή ηλεκτρικού ρεύματος βρίσκεται εκτός λειτουργίας. Για το ποια γεννήτρια θα ξεκινήσει εξαρτάται από το εάν ο αντιδραστήρας βρίσκεται σε λειτουργία ή όχι.

Οι συνθήκες λειτουργίας του συστήματος αυτού θα πρέπει να είναι οι ακόλουθες:

1. Πριν από την εκκίνηση κάθε γεννήτριας θα πρέπει να γίνεται έλεγχος για μηχανικό σφάλμα (θερμοκρασία, στάθμη, πίεση λαδιού-λιπαντικού, καυσίμου κλπ)
2. Θα πρέπει να γίνεται και έλεγχος για ηλεκτρικό σφάλμα (υπερφόρτιση, υπέρταση, σφάλμα συχνότητας κλπ)
3. Για κάθε τύπο σφάλματος θα πρέπει να υπάρχει αντίστοιχη ενδεικτική λυχνία και ηχητικό σήμα (ALARM)
4. Για να γίνει ξανά εκκίνηση κάποιας γεννήτριας θα πρέπει να έχουν γίνει RESET τα ALARM

Ζητούνται

a) Να συνταχθούν οι πίνακες εισόδων – εξόδων

b) Να συνταχθεί πρόγραμμα P.L.C σε γλώσσα LADDER στο περιβάλλον Simatic Manager S7

4.4.2 Πίνακες εισόδων / εξόδων

4.4.2.1Πίνακας Εισόδων

Πίνακας 4.4.2.1

α/α	**ΠΕΡΙΓΡΑΦΗ**	**ΔΙΕΥΘΥΝΣΗ ΕΙΣΟΔΩΝ**	**ΤΥΠΟΣ**
1.	Παροχή ΔΕΗ	I0.0	BOOL
2.	Αντιδραστήρας	I0.1	BOOL
3.	Χαμηλή στάθμη λαδιού	I0.2	BOOL
4.	Χαμηλή στάθμη νερού	I0.3	BOOL
5.	Χαμηλή στάθμη καυσίμου	I0.4	BOOL
6.	Υψηλη Θερμοκρασία	I0.5	BOOL
7.	Σφάλμα ρουλεμάν	I0.6	BOOL
8.	Υπέρταση	I0.7	BOOL
9.	Υπερένταση	I1.0	BOOL
10.	Υψηλή ταχύτητα	I1.1	BOOL
11.	Αντίστροφη ισχύς	I1.2	BOOL
12.	Χαμηλή ή υψηλή πίεση	I1.3	BOOL
I3.	Επαναφορά συναγερμών	I1.4	BOOL
14.	Επιτυχημένη λειτουργία	I1.5	BOOL

4.4.2.2Πίνακας Εξόδων

Πίνακας 4.4.2.2

α/α	**ΠΕΡΙΓΡΑΦΗ**	**ΔΙΕΥΘΥΝΣΗ ΕΞΟΔΩΝ**	**ΤΥΠΟΣ**
1.	1η Γεννήτρια	Q0.0	BOOL
2.	2η Γεννήτρια	Q0.1	BOOL
3.	Εκκίνηση γεννήτριας	Q0.2	BOOL
4.	Έλεγχος συναγερμών	Q0.3	BOOL
5.	Μηχανικό σφάλμα	Q0.4	BOOL
6.	Ηλεκτρικό σφάλμα	Q0.5	BOOL
7.	Κλείσιμο φρένου	Q1.0	BOOL
8.	Γεννήτρια σε λειτουργία	Q1.1	BOOL
9.	Συγχρονισμός	Q1.2	BOOL
10.	Έλεγχος γεννήτριας	Q1.3	BOOL
11.	Ανέβασμα στροφών γεννήτριας	Q1.4	BOOL

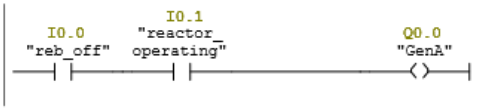
4.4.3 Πρόγραμμα σε LADDER

OB1 - <offline>
""
Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 12/17/2017 02:28:01 PM
Interface: 02/15/1996 04:51:12 PM
Lengths (block/logic/data): 00326 00186 00020

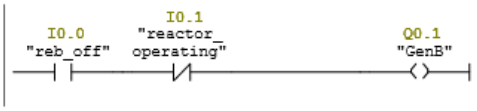
Name	Data Type	Address	Comment
TEMP		0.0	
OB1_EV_CLASS	Byte	0.0	Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
OB1_SCAN_1	Byte	1.0	1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
OB1_PRIORITY	Byte	2.0	Priority of OB Execution
OB1_OB_NUMBR	Byte	3.0	1 (Organization block 1, OB1)
OB1_RESERVED_1	Byte	4.0	Reserved for system
OB1_RESERVED_2	Byte	5.0	Reserved for system
OB1_PREV_CYCLE	Int	6.0	Cycle time of previous OB1 scan (milliseconds)
OB1_MIN_CYCLE	Int	8.0	Minimum cycle time of OB1 (milliseconds)
OB1_MAX_CYCLE	Int	10.0	Maximum cycle time of OB1 (milliseconds)
OB1_DATE_TIME	Date_And_Time	12.0	Date and time OB1 started

Block: OB1 "Main Program Sweep (Cycle)"

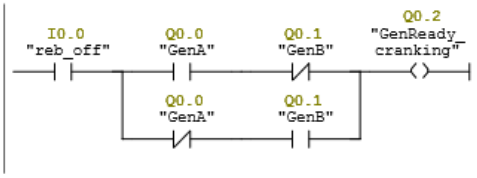
Network: 1



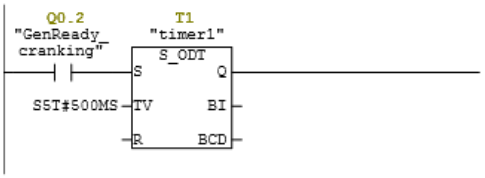
Network: 2



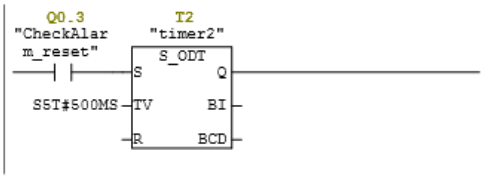
Network: 3



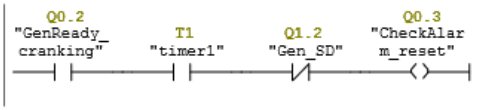
Network: 4



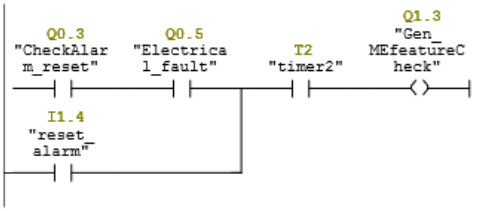
Network: 5



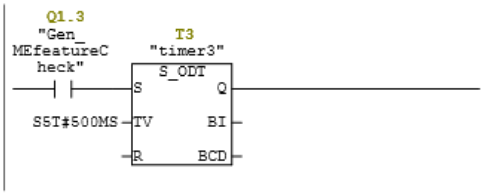
Network: 6



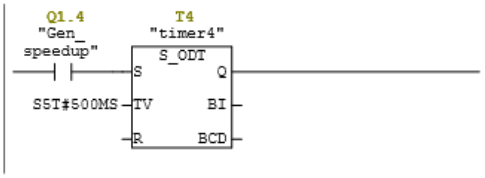
Network: 7



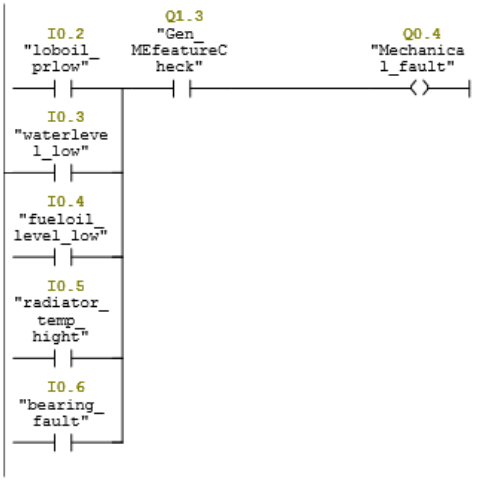
Network: 8



Network: 9

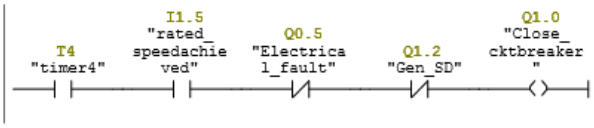


Network: 10

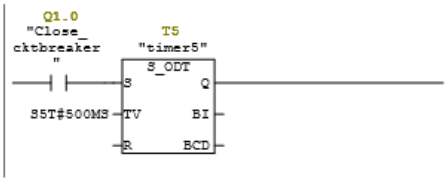


Network: 11

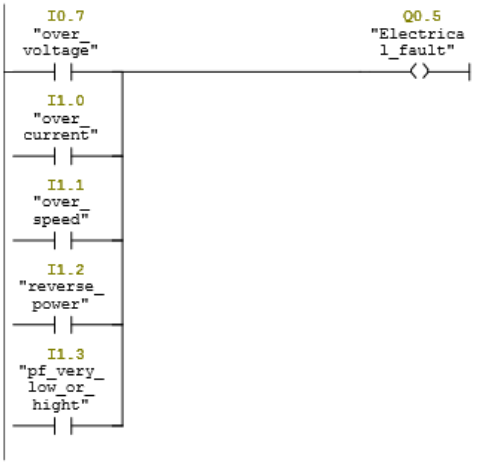
Network: 12



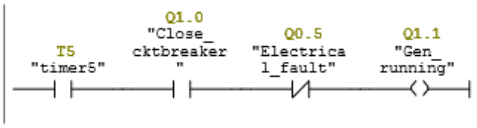
Network: 13



Network: 14



Network: 15



4.5 Εφαρμογή 5η

ΑΥΤΟΜΑΤΗ ΜΕΤΑΓΩΓΗ

4.5.1 Παρουσίαση απαιτήσεων εφαρμογής

Σε μια βιομηχανική μονάδα λειτουργούν κάποια κρίσιμα φορτία, γι'αυτό το λόγο θα εγκατασταθεί σύστημα αυτόματης μεταγωγής προκειμένου να διασφαλίσουμε την τροφοδοσία των φορτίων σε περιπτώσεις σφάλματος στη μία γραμμή.
Το σύστημα αποτελείται από τα εξής :

1. τον διακόπτη της κανονικής πηγής (ΔΕΗ) εξοπλισμένο με μοτέρ τηλεχειρισμού, βοηθητική επαφή 1OF (επαφή θέσης) και βοηθητική επαφή σφάλματος SD (επαφή τριπ)
2. τον διακόπτη της πηγής αντικατάστασης (Η/Ζ) εξοπλισμένο με μοτέρ τηλεχειρισμού, βοηθητική επαφή 1OF (επαφή θέσης) και βοηθητική επαφή σφάλματος SD (επαφή τριπ)
3. Πλάτη με μηχανική μανδάλωση
4. Τέλος θα υπάρχει ένδειξη που θα υποδυκνύει τον τρόπο λειτουργίας.

Ζητούνται

a) Να συνταχθούν οι πίνακες εισόδων – εξόδων

b) Να σχεδιαστεί πρόγραμμα P.L.C σε γλώσσα LADDER στο περιβάλλον Simatic Manager S7

4.5.2 Πίνακες εισόδων / εξόδων

4.5.2.1Πίνακας Εισόδων

Πίνακας 4.5.2.1

α/α	**ΠΕΡΙΓΡΑΦΗ**	**ΔΙΕΥΘΥΝΣΗ ΕΙΣΟΔΩΝ**	**ΤΥΠΟΣ**
1.	Παρουσία ΔΕΗ	I0.0	BOOL
2.	Διακόπτης ΔΕΗ σε κατάσταση ON	I0.1	BOOL
3.	Διακόπτης ΔΕΗ σε κατάσταση TRIP	I0.2	BOOL
4.	Παρουσία Η/Ζ	I1.0	BOOL
5.	Διακόπτης Η/Ζ σε κατάσταση ON	I1.1	BOOL
6.	Διακόπτης Η/Ζ σε κατάσταση TRIP	I1.2	BOOL

4.5.2.2Πίνακας Εξόδων

Πίνακας 4.5.2.2

α/α	**ΠΕΡΙΓΡΑΦΗ**	**ΔΙΕΥΘΥΝΣΗ ΕΞΟΔΩΝ**	**ΤΥΠΟΣ**
1.	Διακόπτης ΔΕΗ μη οπλισμένος	Q0.0	BOOL
2.	Διακόπτης ΔΕΗ οπλισμένος	Q0.1	BOOL
3.	Ένδεικτική λυχνία ΔΕΗ	Q0.2	BOOL
4.	Διακόπτης Η/Ζ μη οπλισμένος	Q1.0	BOOL
5.	Διακόπτης Η/Ζ οπλισμένος	Q1.1	BOOL
6.	Ένδεικτική λυχνία Η/Ζ	Q1.2	BOOL

4.5.3 Πρόγραμμα σε LADDER

OB1 - <offline>

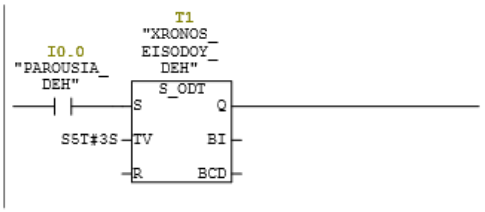
""

Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 03/18/2018 04:40:49 PM
Interface: 02/15/1996 04:51:12 PM
Lengths (block/logic/data): 00284 00146 00020

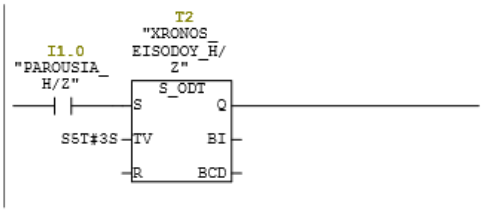
Name	Data Type	Address	Comment
TEMP		0.0	
OB1_EV_CLASS	Byte	0.0	Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
OB1_SCAN_1	Byte	1.0	1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
OB1_PRIORITY	Byte	2.0	Priority of OB Execution
OB1_OB_NUMBR	Byte	3.0	1 (Organization block 1, OB1)
OB1_RESERVED_1	Byte	4.0	Reserved for system
OB1_RESERVED_2	Byte	5.0	Reserved for system
OB1_PREV_CYCLE	Int	6.0	Cycle time of previous OB1 scan (milliseconds)
OB1_MIN_CYCLE	Int	8.0	Minimum cycle time of OB1 (milliseconds)
OB1_MAX_CYCLE	Int	10.0	Maximum cycle time of OB1 (milliseconds)
OB1_DATE_TIME	Date_And_Time	12.0	Date and time OB1 started

Block: OB1 "Main Program Sweep (Cycle)"

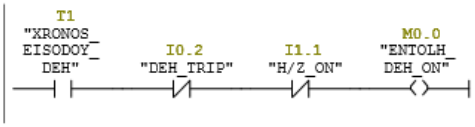
Network: 1 PAROUSIA_DEH



Network: 2 PAROUSIA_H/Z

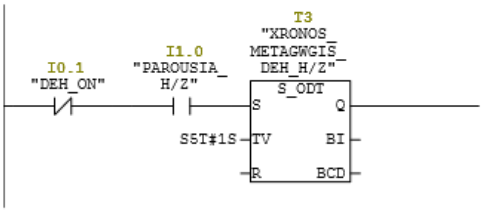


Network: 3 ENTOLH DEH_ON

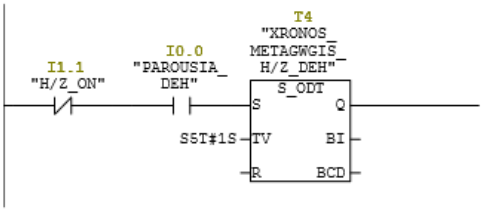


Network: 4 ENTOLH H/Z_ON

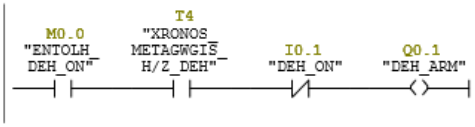
Network: 5 XRONOS_METAGWGHS_DEH_H/Z



Network: 6 XRONOS_METAGWGHS_H/Z_DEH

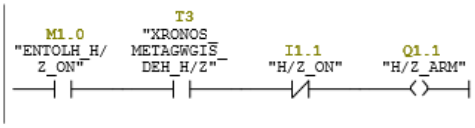


Network: 7 KLEISIMO DEH



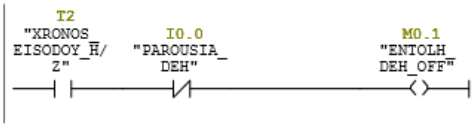
Network: 8 ENDEIKSI DEH_ARM

Network: 9 KLEISIMO H/Z



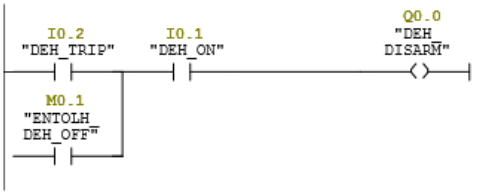
Network: 10 ENDEIKSI H/Z_ARM

Network: 11 ENTOLH_DEH_OFF

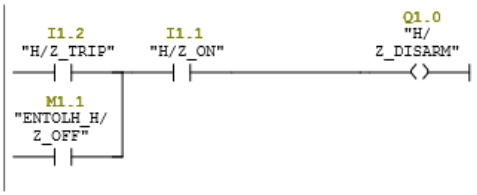


Network: 12 ENTOLH_H/Z_OFF

Network: 13 ANOIGMA_DEH



Network: 14 ANOIGMA_H/Z



4.6 Εφαρμογή 6η

ΠΡΟΣΤΑΣΙΑ ΜΕΤΑΓΩΓΗΣ

4.6.1 Παρουσίαση απαιτήσεων εφαρμογής

Σε μια βιομηχανία είναι εγκατεστημένο ένα σύστημα μεταγωγής ώστε να μην σταματάει η παραγωγή της σε περίπτωση σφάλματος της κύριας γραμμής τροφοδοσίας. Για τον έλεγχο και την προστασία του Η/Ζ πρόκειται να εγκατασταθεί ένα σύστημα προστασίας για να αποφευχθεί η καταστροφή του Η/Ζ απο κάποιο σφάλμα. Οι συνθήκες λειτουργίας του συστήματος είναι οι παρακάτω:

1. Θα υπάρχει περιστροφικός διακόπτης τύπου 0-1-2 όπου θα επιλέγεται ο τρόπος λειτουργίας (αυτόματο, χειροκίνητο, δοκιμή)
2. Πριν την εκκίνηση του Η/Ζ θα πρέπει να γίνεται έλεγχος του καυσίμου
3. Σε περίπτωση επαναφοράς της κανονικής πηγής τροφοδοσίας το Η/Ζ να σταματάει ομαλά ώστε να ψύχεται
4. Θα υπάρχει φαροσειρήνα (ALARM) σε περίπτωση αποτυχίας εκκίνησης του Η/Ζ

Ζητούνται

a) Να συνταχθούν οι πίνακες εισόδων – εξόδων

b) Να σχεδιαστεί πρόγραμμα P.L.C σε γλώσσα LADDER στο περιβάλλον Simatic Manager S7

4.6.2 Πίνακες εισόδων / εξόδων

4.6.2.1Πίνακας Εισόδων

Πίνακας 4.6.2.1

α/α	**ΠΕΡΙΓΡΑΦΗ**	**ΔΙΕΥΘΥΝΣΗ ΕΙΣΟΔΩΝ**	**ΤΥΠΟΣ**
1.	Λειτουργία αυτόματο	I0.0	BOOL
2.	Λειτουργία δοκιμής	I0.1	BOOL
3.	Λειτουργία χειροκίνητο	I0.2	BOOL
4.	ΔΕΗ εντος επιτρεπόμενων ορίων	I0.3	BOOL
5.	Λειρουγία Η/Ζ	I0.4	BOOL
6.	Βλάβη	I0.5	BOOL

4.6.2.2Πίνακας Εξόδων

Πίνακας 4.6.2.2

α/α	**ΠΕΡΙΓΡΑΦΗ**	**ΔΙΕΥΘΥΝΣΗ ΕΞΟΔΩΝ**	**ΤΥΠΟΣ**
1.	Βαλβίδα καυσίμου	Q0.0	BOOL
2.	Μίζα	Q0.1	BOOL
3.	Alarm	Q0.2	BOOL
4.	Αποτυχία εκκίνησης	Q0.3	BOOL

4.6.3 Πρόγραμμα σε LADDER

OB1 - <offline>

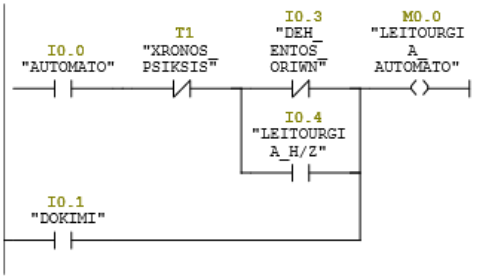
""

Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 01/21/2018 05:43:26 PM
Interface: 02/15/1996 04:51:12 PM
Lengths (block/logic/data): 00322 00186 00020

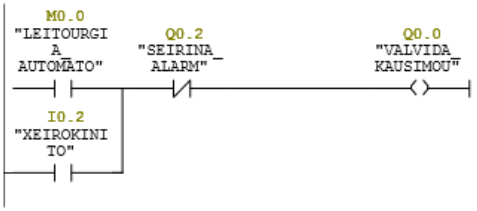
Name	Data Type	Address	Comment
TEMP		0.0	
OB1_EV_CLASS	Byte	0.0	Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
OB1_SCAN_1	Byte	1.0	1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
OB1_PRIORITY	Byte	2.0	Priority of OB Execution
OB1_OB_NUMBR	Byte	3.0	1 (Organization block 1, OB1)
OB1_RESERVED_1	Byte	4.0	Reserved for system
OB1_RESERVED_2	Byte	5.0	Reserved for system
OB1_PREV_CYCLE	Int	6.0	Cycle time of previous OB1 scan (milliseconds)
OB1_MIN_CYCLE	Int	8.0	Minimum cycle time of OB1 (milliseconds)
OB1_MAX_CYCLE	Int	10.0	Maximum cycle time of OB1 (milliseconds)
OB1_DATE_TIME	Date_And_Time	12.0	Date and time OB1 started

Block: OB1 "Main Program Sweep (Cycle)"

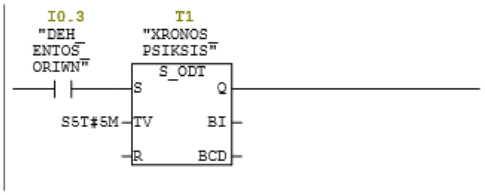
Network: 1 EPILOGI TROPOU LEITOURGIAS



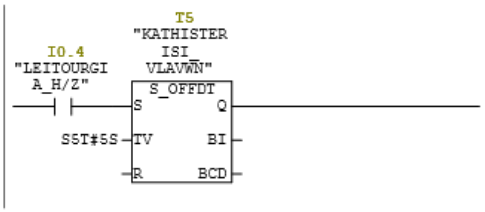
Network: 2 VALVIDA KAUSIMOU



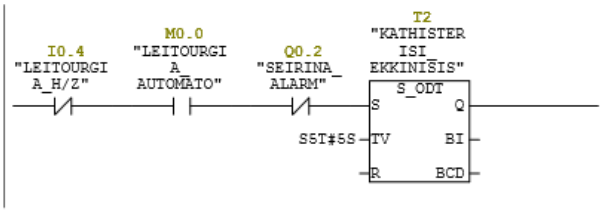
Network: 3 XRONOS PSIKSIS H/Z AFOU EPANELTHEI I DEH



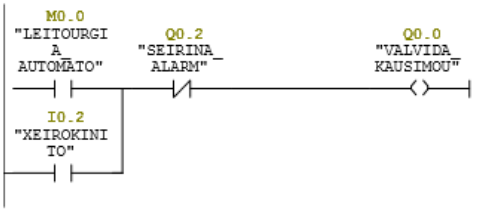
Network: 4 KATHISTERISI VLAVWN AMA TO H.Z SVISEI APOTOMA



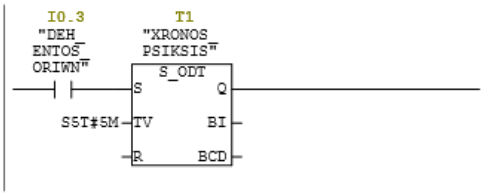
Network: 5 KATHISTERISI EKKINISIS



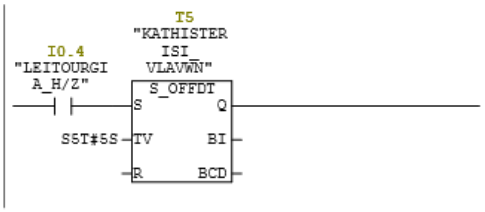
Network: 2 VALVIDA KAUSIMOU



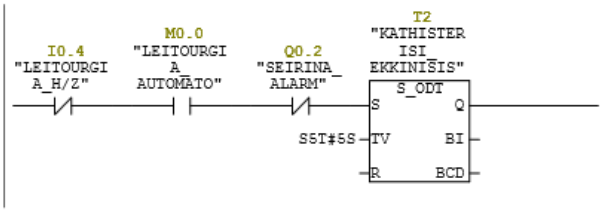
Network: 3 XRONOS PSIKSIS H/Z AFOU EPANELTHEI I DEH



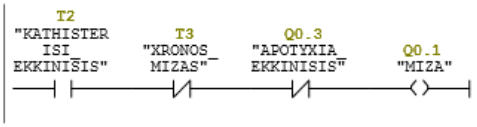
Network: 4 KATHISTERISI VLAVWN AMA TO H.Z SVISEI APOTOMA



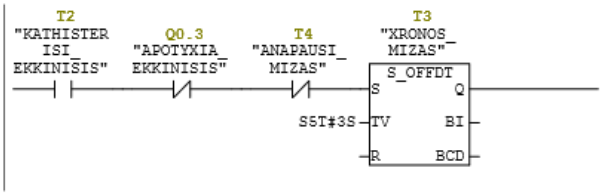
Network: 5 KATHISTERISI EKKINISIS



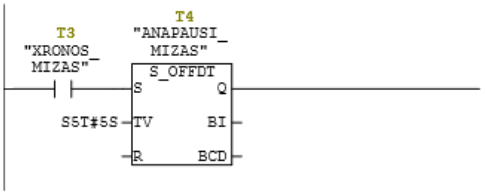
Network: 6 MIZA



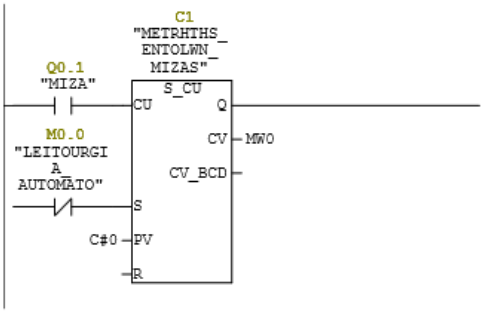
Network: 7 XRONOS MIZAS



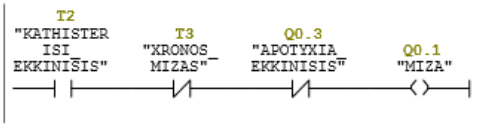
Network: 8 ANAPAUSI MIZAS



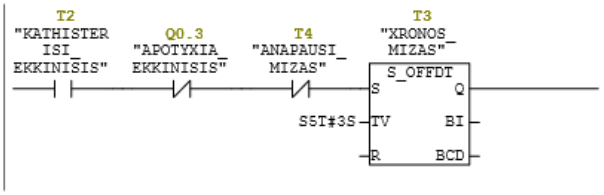
Network: 9 METRITIS APOTUXIMENWN PROSPATHEIWN EKKINISIS



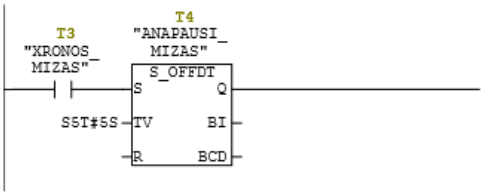
Network: 6 MIZA



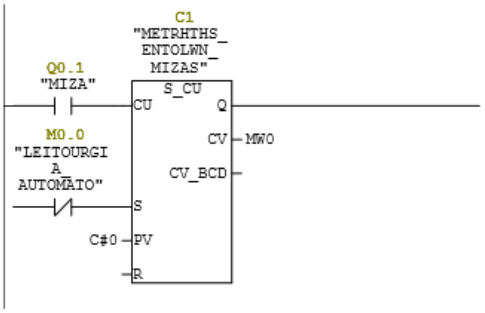
Network: 7 XRONOS MIZAS



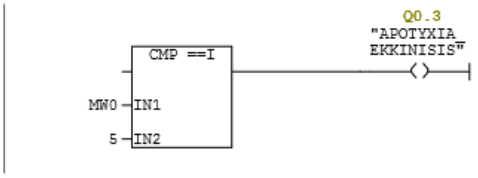
Network: 8 ANAPAUSI MIZAS



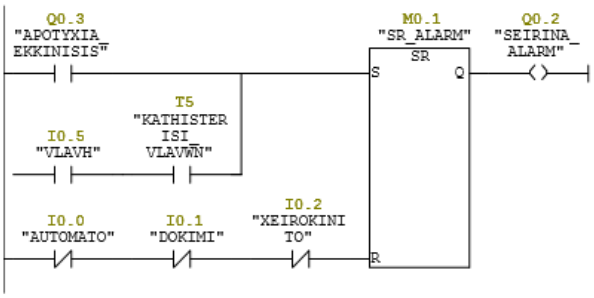
Network: 9 METRITIS APOTUXIMENWN PROSPATHEIWN EKKINISIS



Network: 10 AN APOTUXIMENES PROSPATHEIES EINAI 5 TOTE SFALMA EKKINISIS



Network: 11 SEIRINA SFALMATOS



ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] N.Mohan, T.M. Undeland, "Power Electronics, Converters, Applications and Design", John Wiley & Sons, 1995.

[2] I. Khan, J. Tapson, I. de Vriew, "Frequency control of a current-fed inverter for induction heating", IEEE Proceedings on Industrial Electronics, Vol. 1, No.1, 2000, pp.343-346.

[3] Fairchild semiconductors, "Induction Heating System Topology Review", Applications notes, July 2000.

[4] Ζούμης Ν., Σάλτης Γ., Καφφετζάκης Π., [2000], Συστήματα Αυτοματισμών Β Τόμος Ηλεκτρολογικού Τομέα ΤΕΕ, Παιδαγωγικό Ινστιτούτο ΥΠΕΠΘ.

[5] Μπερέτας Ι., [2008], Αυτοματισμός με χρήση PLC, Εκδόσεις Α. Τζιόλα Ε.

[6] Κρανάς Γ., Δασκαλόπουλος Ε., (2001), Βιομηχανικοί Αυτοματισμοί και Προγραμματιζόμενοι Λογικοί Ελεγκτές (PLC), Εκδόσεις Ίων

[7] SIEMENS SIMATIC, Working with STEP 7 V5.1, Getting Started A5E00069681-03, Edition 08/2000

[8] SIEMENS SIMATIC, Standard Software for S7-300 and S7-400 PID Control, User Manual, C79000-G7076-C516-01, 05/1996

[9] SIEMENS SIMATIC, Ladder Logic (LAD) for S7-300 and S7-400 Programming, Manual, C79000-G7076-C504-02, 0 5/1996

[10] SIEMENS SIMATIC, S7-300 Programmable Controller Quick Start Primer, C79000-G7076-C500-01, 05/1996

[11] SIEMENS SIMATIC, Function Block Diagram (FBD) for S7-300 and S7-400 Programming, Reference Manual, A5E00261409-01, Edition 01/2004

[12] SIEMENS SIMATIC, S7-300 Getting Started for First Time Users, A5E01094750-01, 04/2007

[13] SIEMENS SIMATIC, System Software for S7-300/400 System and Standard Functions, Reference Manual, A5E00261410-01, Edition 01/2004

[14] SIEMENS SIMATIC, Programming with STEP 7, Manual A5E00706944-01, Edition 03/2006

ΑΙΓΑΛΕΩ

ΦΕΒΡΟΥΑΡΙΟΣ 2018