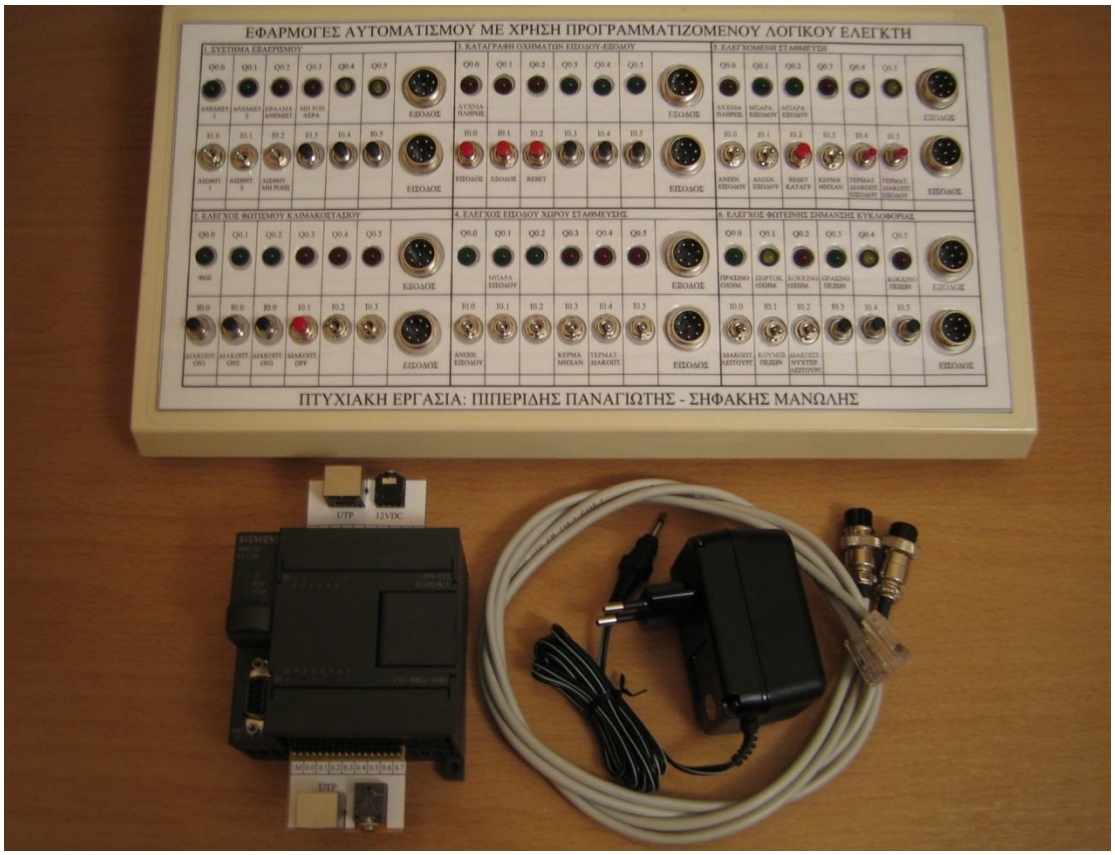
ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΕΦΑΡΜΟΓΕΣ ΑΥΤΟΜΑΤΙΣΜΟΥ ΜΕ ΧΡΗΣΗ ΠΡΟΓΡΑΜΜΑΤΙΖΟΜΕΝΟΥ ΛΟΓΙΚΟΥ ΕΛΕΓΚΤΗ



ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:ΦΡΑΓΚΙΑΔΑΚΗΣ ΝΙΚΟΛΑΟΣ

ΣΠΟΥΔΑΣΤΕΣ: ΠΙΠΕΡΙΔΗΣ ΠΑΝΑΓΙΩΤΗΣ
ΣΗΦΑΚΗΣ ΕΜΜΑΝΟΥΗΛ

ΠΕΡΙΛΗΨΗ

Η πτυχιακή αυτή εργασία εκπονήθηκε, έχοντας κατά νου ένα διπλό στόχο. Καταρχήν την παρουσίαση και την εκτενή ανάλυση των σύγχρονων τεχνολογιών όπως είναι οι προγραμματιζόμενοι λογικοί ελεγκτές και διαμέσου αυτής να καταλάβει ο αναγνώστης την αξία και την αναγκαιότητα των προγραμματιζόμενων λογικών ελεγκτών, τα πλεονεκτήματα τους και την έκταση εφαρμογών τους.

Επίσης, στα πλαίσια του θεωρητικού μέρους της εργασίας, παρουσιάζονται οι γλώσσες προγραμματισμού των προγραμματιζόμενων λογικών ελεγκτών με έμφαση στις γλώσσες STL και Ladder.

Επιπλέον, στα πλαίσια του πειραματικού κομματιού της εργασίας παρουσιάζονται κάποιες προτεινόμενες εφαρμογές αυτοματισμού με την επίλυση τους, έχοντας ως στόχο την παρότρυνση του αναγνώστη να ασχοληθεί με τους προγραμματιζόμενους λογικούς ελεγκτές μιας και όπως φαίνεται, η χρήση τους και ο προγραμματισμός τους πλέον γίνεται σε ένα πολύ φιλικό προς τον χρήστη περιβάλλον.

ABSTRACT

This diploma thesis was worked out, aiming a double mark in mind. Firstly the presentation and the extensive analysis of modern technologies such as programmable logic controllers and through this, we aim the reader to understand the worth and the need of programmable logic controllers, the advantages and the area of applications they cover.

Also, in the frames of theoretical part of work the programming languages of programmable logic controllers are presented with emphasis in STL language and Ladder language.

Moreover, in the frames of experimental part of work some proposed applications of automation are presented with their settlements, aiming the encouragement of the reader to deal with programming programmable logic controllers after their usage is more quite simple.

ΕΥΧΑΡΙΣΤΙΕΣ

Στο σημείο αυτό, δε θα μπορούσαμε να μην ευχαριστήσουμε τον Κο Φραγκιαδάκη Νικόλαο εισηγητή της πτυχιακής αυτής εργασίας για την πολύτιμη βοήθεια που μας προσέφερε κατά την διάρκεια εκπόνησης της παρούσας πτυχιακής εργασίας, καθώς και ότι μας εμπιστεύτηκε την παρούσα εργασία και μας έδωσε την ευκαιρία να έρθουμε σε επαφή με τις νέες τεχνολογίες και να προσπαθήσουμε να τις κατανοήσουμε όσο το δυνατό καλύτερα.

ΧΑΝΙΑ 2009

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ

Η εργασία αυτή αποτελεί την απαραίτητη εργασία που πρέπει να εκπονηθεί καλούμενη ‘‘πτυχιακή εργασία’’ ώστε να καταστεί κάποιος πτυχιούχος.

Η πτυχιακή αυτή εργασία έχει ως στόχο την εμβάθυνση μας σε ένα συγκεκριμένο αντικείμενο που έχουμε διδαχθεί κατά την διάρκεια της φοίτησης μας στο τμήμα ηλεκτρονικής του Τ.Ε.Ι. Χανίων.

Η εκπόνηση της ξεκίνησε τον Οκτώβριο του 2008 και ολοκληρώθηκε τον Απρίλιο του 2009.

Έτσι, η παρούσα πτυχιακή εργασία που φέρει ως τίτλο ‘‘Εφαρμογές αυτοματισμού με χρήση προγραμματιζόμενου λογικού ελεγκτή’’ αποτελείται από δύο κυρίως μέρη. Στο πρώτο μέρος αναπτύσσεται η υπάρχουσα θεωρία των προγραμματιζόμενων λογικών ελεγκτών. Πιο αναλυτικά παρουσιάζεται η δομή, η αρχή λειτουργίας, ο τρόπος προγραμματισμού και τα πλεονεκτήματα των προγραμματιζόμενων λογικών ελεγκτών. Ιδιαίτερη βαρύτητα δίνεται στις τεχνικές προγραμματισμού και στις γλώσσες προγραμματισμού με εκτενέστερη αναφορά στην γλώσσα STL. Στο δεύτερο και πειραματικό μέρος παρουσιάζονται έξι προτεινόμενες εφαρμογές αυτοματισμού μαζί με την υλοποίηση τους. Επίσης πρέπει να αναφερθεί ότι οι συγκεκριμένες εφαρμογές έχουν υλοποιηθεί στην γλώσσα STL και αφορούν το PLC S7-200 της SIEMENS.

ΕΙΣΑΓΩΓΗ

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 για 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 (όπως αυτά διασώθηκαν με το πρωτότυπο κείμενο ή σε μεταφράσεις) 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. Για τους ελεγκτές προγραμματιζόμενης λογικής PLC, η χρονιά με την ιστορική σημασία ήταν το 1958. Τότε η εμπορική ονομασία Simatic που είχε ήδη γίνει συνώνυμη με τα PLC και ήταν η σειρά που κυριαρχούσε στην παγκόσμια αγορά, έγινε καταχωρημένο εμπορικό σήμα. 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 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. Η εξέλιξη των αυτοματισμών και οι προγραμματιζόμενοι λογικοί ελεγκτές (PLC).

Η εξέλιξη των αυτοματισμών, όπως ήταν φυσικό, ακολούθησε την πορεία εξέλιξης της τεχνολογίας. Οι πρώτοι αυτοματισμοί ήταν καθαρά μηχανικοί, όλοι οι έλεγχοι δηλαδή καθοριζόταν από την κίνηση γραναζιών και μοχλών. Το μεγάλο άλμα στους αυτοματισμούς έγινε με τη χρήση του ηλεκτρισμού. Το κύριο εξάρτημα των ηλεκτρολογικών αυτοματισμών είναι ο ηλεκτρονόμος.

Μετά τον δεύτερο παγκόσμιο πόλεμο αρχίζει η ηλεκτρονική εποχή. Ήδη από τις αρχές του 20^{ου} αιώνα έχουμε τις πρώτες ηλεκτρονικές συσκευές, το ραδιόφωνο, αργότερα την τηλεόραση, τους ασύρματους και τα ραντάρ. Το κύριο εξάρτημα αυτών των συσκευών ήταν η ηλεκτρονική λυχνία. Η ανακάλυψη του τρανζίστορ το 1950 ήταν η αρχή της ηλεκτρονικής επανάστασης των ημιαγωγών. Το θαυματουργό αυτό στοιχείο αντικατέστησε την ακριβή, ογκώδη και ενεργειοβόρα ηλεκτρονική λυχνία και έκανε τις ηλεκτρονικές συσκευές μικρότερες, εύκολες στην κατασκευή και απείρως πιο φθηνές.

Το 1945 κατασκευάστηκε ο πρώτος ηλεκτρονικός υπολογιστής, ο ENIAC, ο οποίος χρησιμοποιούσε λυχνίες. Ο ENIAC δεν θύμιζε σε τίποτα τους σημερινούς υπολογιστές, ήταν ένα ολόκληρο εργοστάσιο το οποίο έλυνε μαθηματικές εξισώσεις. Μετά το 1950 και με τη χρήση των τρανζίστορ έχουμε τους πρώτους πραγματικούς υπολογιστές, οι οποίοι χρησιμοποιούνται κυρίως στο θέμα της μηχανογράφησης, δηλαδή στην αποθήκευση και διαχείριση μεγάλων αρχείων δεδομένων.

Από τη δεκαετία του ΄60, ήδη οι μηχανικοί άρχισαν να σκέφτονται τρόπους για να αξιοποιήσουν τις καταπληκτικές δυνατότητες των υπολογιστών στη βιομηχανία. Από τις πρώτες εφαρμογές των υπολογιστών στη βιομηχανία ήταν οι αυτόματες εργαλειομηχανές (τόρνοι, φρέζες κ.λπ.), οι οποίες μέχρι τότε χρησιμοποιούσαν κυρίως μηχανολογικούς και λιγότερο ηλεκτρολογικούς αυτοματισμούς.

Η επιτυχημένη αυτή εφαρμογή οδήγησε τους μηχανικούς να αρχίσουν να σκέφτονται την αντικατάσταση όλων των αυτοματισμών ενός εργοστασίου από ένα υπολογιστή. Μέχρι όμως την δεκαετία του 80 αυτό ήταν αδύνατο, διότι ο υπολογιστής ήταν μια πανάκριβη και δύσκολη στην χρήση της συσκευή.

Η επανάσταση της πληροφορικής ξεκινά το 1975 με την κατασκευή του πρώτου μικροϋπολογιστή. Πολλά από όλα όσα σήμερα θεωρούμε αυτονόητα δημιουργήθηκαν μετά το 1980. Η τεχνολογία άλλαξε πορεία, αλλάζοντας πορεία σε όλους τους τομείς της καθημερινής ζωής. Ο μικροϋπολογιστής τρύπωσε παντού, σε οποιοδήποτε τομέα, σε οποιαδήποτε εφαρμογή.

Η βιομηχανία μέχρι και τη δεκαετία του ΄80 μπορούμε να πούμε ότι χρησιμοποιούσε ελάχιστα τα ηλεκτρονικά. Το 90% και πλέον των αυτοματισμών καταλάμβαναν οι αυτοματισμοί με ηλεκτρονόμους. Τα ηλεκτρονικά χρησιμοποιούνταν κυρίως για κάποιες ευφυείς εργασίες, και οι πλακέτες αυτές τοποθετούνταν μέσα στους πίνακες των ηλεκτρονόμων.

Στις αρχές της δεκαετίας του ΄80 οι εταιρείες παραγωγής ηλεκτρολογικού υλικού εμφανίζουν στους μηχανικούς και τεχνικούς της βιομηχανίας ένα νέο προϊόν αυτοματισμού, το οποίο ονόμασαν PLC. Η πλήρης ονομασίας αυτής της νέας συσκευής είναι Programmable Logic Controller (Προγραμματιζόμενος Λογικός Ελεγκτής). Οι εταιρείες δεν χρησιμοποίησαν αρχικά στην αγορά την πλήρη ονομασία, μιλώντας απλά για PLC, πράγμα που ίσως έγινε έντεχνα για να μην τρομάξουν το τεχνικό κατεστημένο της Βιομηχανίας.

Το PLC δεν είναι τίποτα άλλο παρά ένας μικροϋπολογιστής, κατάλληλα προσαρμοσμένος ώστε να χρησιμοποιείται για τη λειτουργία αυτοματισμών. Τα PLC προορίζονταν να αντικαταστήσουν τον κλασικό πίνακα αυτοματισμού με τους ηλεκτρονόμους. Όπως γίνεται εύκολα κατανοητό, μιλάμε για μια τεράστια αλλαγή στον τρόπο που μέχρι τότε δούλευε η βιομηχανία, δηλαδή έπρεπε να περάσει κατευθείαν από τους ηλεκτρονόμους στους υπολογιστές. Εδώ ήταν που οι εταιρείες παραγωγής PLC έπαιξαν ένα σπουδαίο παιχνίδι μάρκετινγκ. Προσάρμοσαν τον τρόπο χρήσης του PLC στον τρόπο που δούλευε μέχρι τότε η βιομηχανία, δηλαδή:

Έντεχνα απέφυγαν να χρησιμοποιήσουν λέξεις που θα τρόμαζαν το τεχνικό κατεστημένο της βιομηχανίας, όπως για παράδειγμα υπολογιστής, προγραμματισμός κ.λπ. Ακόμη και το όνομα του νέου προϊόντος απέφυγαν να το χρησιμοποιήσουν ολοκληρωμένο και προτιμούσαν να αναφέρουν τη συσκευή ως PLC.

Προσπάθησαν να μην αλλάξουν τον μέχρι τότε τρόπο εργασίας στον τομέα των αυτοματισμών. Δεν άλλαξαν δηλαδή τίποτα σε σχέση με τον σχεδιασμό ενός αυτοματισμού. Απλά είπαν, στους τεχνικούς: αυτό το σχέδιο αντί να το δώσετε στον ηλεκτρολόγο για να το κατασκευάσει, θα το φτιάξετε με τον τρόπο που θα σας δείξουμε και στην ουσία τους μάθαιναν προγραμματισμό. Οι πρώτες γλώσσες προγραμματισμού δεν έκαναν τίποτα παραπάνω από το να αντιγράφουν με πλήκτρα σε μία ειδική συσκευή προγραμματισμού το σχέδιο του ηλεκτρολογικού αυτοματισμού.

Με τον τρόπο αυτό η είσοδος του PLC στην βιομηχανία υπήρξε επιτυχής και ομαλή. Σήμερα, ο κλασικός αυτοματισμός με ηλεκτρονόμους τείνει να εκλείψει. Όλες οι καινούργιες εγκαταστάσεις χρησιμοποιούν PLC. Μετά από λίγα χρόνια ελάχιστες εγκαταστάσεις θα χρησιμοποιούν πίνακες κλασικού αυτοματισμού.

Σήμερα, τα PLC έχουν εξελιχτεί πάρα πολύ σε σχέση με τα πρώτα μοντέλα της δεκαετίας του 80. Και βέβαια το προσωπικό της βιομηχανίας έχει εκπαιδευτεί κατάλληλα στον χειρισμό και προγραμματισμό τους. Σήμερα ένας ηλεκτρολόγος πρέπει να γνωρίζει στοιχειώδη πράγματα από τα ηλεκτρονικά και τις βασικές αρχές των υπολογιστών, αλλιώς θα είναι πολύ δύσκολο να διαβάσει και να καταλάβει ακόμη και το πιο απλό εγχειρίδιο ενός PLC.

Η χρήση των PLC παρέχει πάρα πολλά πλεονεκτήματα σε σχέση με τον κλασικό αυτοματισμό. Η καθολική όμως γενίκευση της χρήσης τους δεν οφείλεται μόνο στα πλεονεκτήματα που παρέχουν στον τελικό χρήστη.

Η χρήση των PLC σε σχέση με τον κλασικό αυτοματισμό συμφέρει πρώτιστα στις εταιρείες που παράγουν είδη αυτοματισμού. Φανταστείτε μόνο πόσο κοστίζει σε μια εταιρεία παραγωγής ηλεκτρολογικού εξοπλισμού η παραγωγή ενός τεράστιου αριθμού βοηθητικών ηλεκτρονόμων και ενός μεγάλου αριθμού χρονικών και απαριθμητών. Σε αντίθεση με τα υλικά αυτά αυτοματισμού, όσον αφορά τα PLC η εταιρεία τι παράγει; Η απάντηση είναι: Μια και μοναδική συσκευή!

Η ψηφιοποίηση σε όλους τους τομείς (και όχι μόνο στον τομέα των αυτοματισμών) οδηγεί σε τρομακτική μείωση του κόστους παραγωγής των αντίστοιχων συσκευών. Γι αυτό μην απορείτε που οι τιμές στα ηλεκτρονικά πέφτουν. Να είστε σίγουροι ότι τα κέρδη των εταιρειών ανεβαίνουν.

Πάντως θα μπορούσαμε να πούμε ότι, ενώ σε όλους τους τομείς της παραγωγής περάσαμε από τις ηλεκτρολογικές συσκευές, στις ηλεκτρονικές με λυχνίες, μετά στις ηλεκτρονικές με ημιαγωγούς (τρανζίστορ) και τέλος φθάσαμε στις

συσκευές με μικροϋπολογιστές, στις ψηφιακές συσκευές δηλαδή, στον τομέα των αυτοματισμών περάσαμε σχεδόν κατευθείαν από τους ηλεκτρολογικούς αυτοματισμούς στους αυτοματισμούς με PLC. Αν θέλουμε να αναζητήσουμε την αιτία γι αυτό, θα λέγαμε ότι μάλλον δεν προλάβαμε. Οι εξελίξεις στην ηλεκτρονική ήταν ραγδαίες, ενώ αντίθετα η βιομηχανική τεχνολογία αλλάζει με πολύ πιο αργούς ρυθμούς.

2. Τι είναι ο προγραμματιζόμενος λογικός ελεγκτής.

Ο Προγραμματιζόμενος Λογικός Ελεγκτής (PLC) είναι μια ηλεκτρονική διάταξη η οποία από την άποψη της λειτουργίας της θα μπορούσε να προσομοιωθεί με ένα πίνακα αυτοματισμού. Έχει δηλαδή εισόδους και εξόδους που συνδέονται με τα στοιχεία μιας εγκατάστασης και βέβαια ένα αλγόριθμο που καθορίζει ότι κάποιος συνδυασμός εισόδων παράγει ένα αποτέλεσμα στις εξόδους. Αντί για την κατασκευή ενός πίνακα με πολύπλοκες συνδεσμολογίες μεταξύ των παραπάνω υλικών, που έχουμε στον κλασικό αυτοματισμό, με την χρήση του PLC η λειτουργία του αυτοματισμού προγραμματίζεται μέσω μιας ειδικής συσκευής (προγραμματιστή) ή μέσω ενός ηλεκτρονικού υπολογιστή με τη βοήθεια ειδικού λογισμικού.

Ανήκουν στην ευρύτερη κατηγορία των ψηφιακών υπολογιστικών συστημάτων. Χρησιμοποιούνται για να ελέγχουν μηχανές και διεργασίες όπου απαιτείται να γίνονται αυτόματες λειτουργίες: κατ’ εξοχήν στη βιομηχανία αλλά και σε κτιριακές εγκαταστάσεις, στη ναυτιλία, σε μεγάλα έργα του δημοσίου ή ιδιωτικού τομέα (σήραγγες, σταθμούς παραγωγής ενέργειας, ορυχεία, βιολογικούς καθαρισμούς), στον έλεγχο κυκλοφορίας οχημάτων, στον φωτισμό αεροδρομίων, σε συστήματα ανελκυστήρων και δεκάδες άλλους τομείς εφαρμογών.

2.1 Η δομή ενός προγραμματιζόμενου λογικού ελεγκτή.

Το PLC είναι ένας μικροϋπολογιστής ειδικού τύπου. Επομένως το υλικό του μέρος (Hardware) θα μοιάζει με το αντίστοιχο των Η/Υ. Έτσι ένα PLC αποτελείται από τα εξής μέρη:

α) Μονάδα εισόδων

β) Μονάδα εξόδων

γ) Κεντρική μονάδα επεξεργασίας (Central Processing Unit - CPU)

δ) Μνήμες

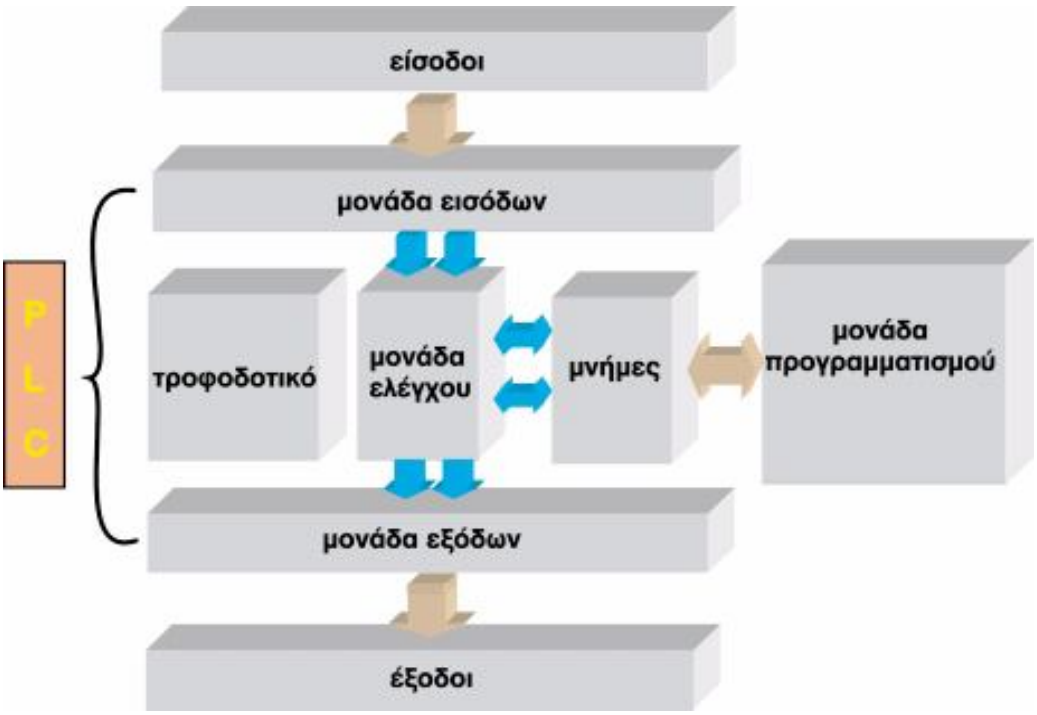
ε) Μονάδα τροφοδοσίας

στ) Πλαίσια τοποθέτησης-επέκτασης

ζ) Διάφορες άλλες βοηθητικές μονάδες

η) Θύρα επικοινωνίας

Στο παρακάτω σχήμα απεικονίζεται η δομή ενός προγραμματιζόμενου λογικού ελεγκτή.



Σχήμα 2.1: Δομή ενός Προγραμματιζόμενου Λογικού Ελεγκτή

Ας δούμε όμως πιο αναλυτικά τις μονάδες από τις ποίες αποτελείται ένα PLC. Πιο συγκεκριμένα έχουμε:

α) Μονάδα εισόδων. Υπάρχουν δύο βασικοί τύποι τέτοιων μονάδων:

• Ψηφιακές (ON-OFF), στις οποίες η είσοδος μπορεί να αναγνωρίζει δύο μόνο τιμές τάσης (υψηλή - χαμηλή). Η τάση αυτή μπορεί να δημιουργείται είτε από το τροφοδοτικό του PLC, είτε από δικό μας εξωτερικό τροφοδοτικό. Η τιμή της στα περισσότερα PLC είναι 24 V_{DC}.

Στις ψηφιακές εισόδους του PLC μπορούμε να συνδέσουμε διαφόρων ειδών εξαρτήματα και υλικά (που ανήκουν στην κατηγορία των αισθητηρίων "sensors")όπως μπουτόν, επαφές ρελέ, διακόπτες, τερματοδιακόπτες, διακόπτες προσέγγισης (proximity switch διαφόρων τύπων - χωρητικούς, επαγωγικούς κλπ.), φωτοκύτταρα και πλήθος ακόμα εξαρτήματα.

• Αναλογικές, σε αυτή την περίπτωση έχουμε το δεύτερο "είδος" εισόδων ενός PLC διαφορετικό_από αυτό των ψηφιακών εισόδων. Οι αναλογικές είσοδοι του PLC "αντιλαμβάνονται" ("ανιχνεύουν", "αναγνωρίζουν") όχι δύο διακριτές καταστάσεις όπως στην_περίπτωση των ψηφιακών εισόδων – αλλά μια κατάσταση που συνεχώς μεταβάλλεται.

Ένα κλασσικό παράδειγμα είναι η μέτρηση στάθμης ενός υγρού υλικού_σε μια δεξαμενή. Η μεταβαλλόμενη στάθμη του υγρού "μεταφράζεται" από το αισθητήριο σε ένα αντίστοιχα μεταβαλλόμενο ηλεκτρικό σήμα που κυμαίνεται σε μία τυποποιημένη κλίμακα έντασης ρεύματος (π.χ. 4 έως 20 mA) ή τάσης ρεύματος (π.χ. 0-10 V) Η - ειδικού τύπου - αναλογική είσοδος του PLC (διαφορετική στην κατασκευή όπως ήδη είπαμε από την ψηφιακή) "αντιλαμβάνεται" τις διαφοροποιήσεις (αυξομειώσεις του ηλεκτρικού σήματος-ρεύματος από π.χ. 4 έως20 mA ή τάσης π.χ. 0-10 V) και τις "μεταφράζει" σε μεταβολές (αυξομειώσεις) του φυσικού φαινομένου, δηλαδή της στάθμης του υγρού.

β) Μονάδα εξόδων. Και οι μονάδες εξόδου διακρίνονται σε 2 κατηγορίες:

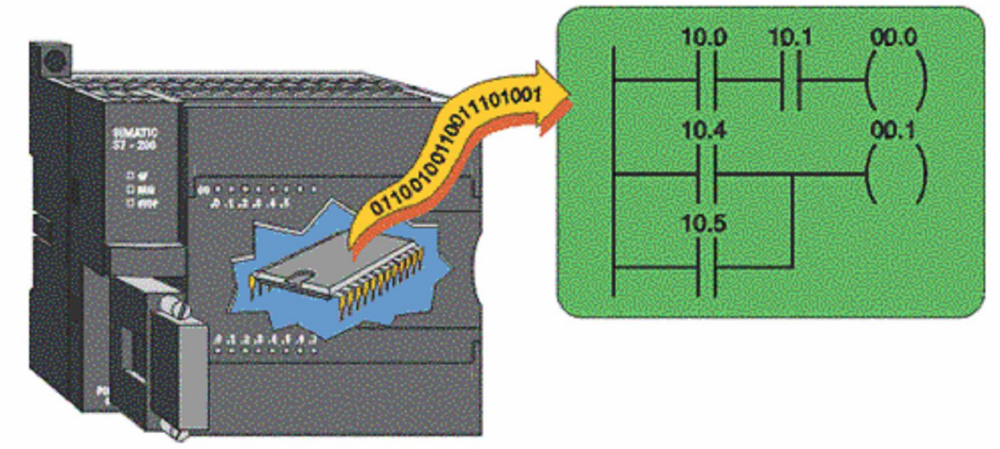
• Ψηφιακές, οι ψηφιακές έξοδοι μπορούν να έχουν κατάσταση ON ή OFF. Σε αυτές συνδέονται και ενεργοποιούνται ή απενεργοποιούνται τα φορτία. Η σύνδεση των φορτίων με τις εξόδους γίνεται είτε απ' ευθείας ή (το πιο συνηθισμένο) μέσω διατάξεων ενεργοποίησης όπως ρελέ κλπ.

• Αναλογικές, η κατάσταση μιας αναλογικής εξόδου μεταβάλλεται συνεχώς. Για παράδειγμα μια αναλογική έξοδος μπορεί να παρέχει ηλεκτρικό σήμα του οποίου η τάση μεταβάλλεται από 0 έως 10 V και το οποίο οδηγεί ένα αναλογικό όργανο μέτρησης π.χ. θερμοκρασίας, ταχύτητας ή βάρους. Ακόμα μέσω ενός ηλεκτροπνευματικού μετατροπέα το μεταβαλλόμενο ηλεκτρικό σήμα μιας αναλογικής εξόδου μπορεί τελικά να ελέγχει μια βαλβίδα αέρος κ.α.

Ένα PLC περιλαμβάνει έναν καθορισμένο μέγιστο αριθμό μονάδων εισόδων και εξόδων που εξαρτάται από τις δυνατότητες της CPU. Τον αριθμό αυτό τον καθορίζει ο εκάστοτε κατασκευαστής.

γ) Κεντρική μονάδα επεξεργασίας (Central Processing Unit - CPU).

Η Κεντρική Μονάδα Επεξεργασίας του PLC είναι ένα ψηφιακό κύκλωμα, ένας μικροεπεξεργαστής συγκεκριμένα (microprocessor) που αποτελεί τον "εγκέφαλο" του PLC. 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



Σχήμα 2.2: Η CPU του PLC

δ) Μνήμες. Διακρίνουμε τις εξής:

• Μνήμη προγράμματος (τύπου RAM - Random Access Memory - Μνήμη Τυχαίας Προσπέλασης). Εδώ αποθηκεύεται το πρόγραμμα που αναπτύσσουμε. Το ότι είναι μνήμη τύπου RAM, επιτρέπει γρήγορες αλλαγές στο πρόγραμμα. Συνδέεται με μπαταρία (διάρκειας περίπου ενός χρόνου), ώστε να διατηρεί το περιεχόμενό της ακόμη και όταν το PLC αποσυνδεθεί από την τροφοδοσία.

• Μνήμη συστήματος (συνήθως τύπου ROM - Read Only Memory – Μνήμη Μόνο για Ανάγνωσης ή PROM -Programmable Read Only Memory - Προγραμματιζόμενη Μνήμη Μόνο για Ανάγνωση). Είναι η μνήμη στην οποία βρίσκεται αποθηκευμένο (από τον κατασκευαστή) το λογισμικό ανάπτυξης (κέλυφος) του PLC.

•Προαιρετική μνήμη EEPROM (Electicaly Erasable Programmable Read Only Memory). Σε αυτή μπορεί να αποθηκευτεί το πρόγραμμα αφού πάρει την τελική του μορφή απελευθερώνοντας έτσι τη μνήμη RAM. Επίσης μπορεί να χρησιμοποιηθεί για φύλαξη ή μεταφορά σε διάφορες (παραπάνω από μία) συσκευές PLC.

ε) Μονάδα τροφοδοσίας.

Η μονάδα τροφοδοσίας ενός PLC έχει σκοπό να δημιουργήσει από την τάση του δικτύου τροφοδοσίας τις απαραίτητες εσωτερικές τάσεις, που απαιτούνται για την τροφοδοσία των ηλεκτρονικών στοιχείων (τρανζίστορ, ολοκληρωμένα κυκλώματα κ.λπ.) του PLC.

στ) Πλαίσια τοποθέτησης-επέκτασης.

Πρόκειται για ειδικές βάσεις-ράγες στις οποίες τοποθετούνται οι βαθμίδες για το σχηματισμό ενός PLC. Στη βάση αυτή είναι ενσωματωμένο σύστημα αγωγών για την επικοινωνία των διαφόρων μονάδων με την Κεντρική Μονάδα Επεξεργασίας (CPU).

ζ) Βοηθητικές μονάδες.

Πρόκειται για συσκευές που δεν είναι απαραίτητες για τη λειτουργία του PLC, σίγουρα όμως δίνουν καλύτερη εποπτεία και έλεγχο του αυτοματισμού. Οι κυριότερες είναι:

-Μονάδα προσομοίωσης. Είναι μία σειρά από διακόπτες με τους οποίους μπορούμε να κάνουμε εργαστηριακό έλεγχο του αυτοματισμού.

- Modem. Είναι συσκευές με τις ποίες μπορούμε να διαβιβάσουμε πληροφορίες και να δώσουμε εντολές μέσω τηλεφωνικής γραμμής.

- Οθόνες (monitors) για έγχρωμες απεικονίσεις μιμικών διαγραμμάτων υψηλής ακρίβειας.

- Εκτυπωτές όλων των τύπων.

η) Θύρα επικοινωνίας.

Η ανταλλαγή πληροφοριών μεταξύ χρήστη και συσκευής μπορεί να γίνει είτε παράλληλα είτε σειριακά. Τα PLC χρησιμοποιούν συνήθως σειριακή ανταλλαγή πληροφοριών με θύρα RS 232C.

2.2 Συστήματα Αρίθμησης – Δεκαδικό & Δυαδικό.

Τα PLC όπως ήδη αναφέραμε είναι ένα ψηφιακό υπολογιστικό σύστημα. Όλα τα συστήματα αυτού του τύπου και το PLC διαχειρίζονται και αποθηκεύουν πληροφορίες με τη μορφή ΔΥΟ διαφορετικών καταστάσεων : ON και OFF (ή αλλιώς με τη μορφή του "λογικού" 1 και 0). Οι πληροφορίες δηλαδή αποτελούνται από "δυαδικά ψηφία" (στα Αγγλικά Binary Digits ή συντομογραφικά: **Bits**). Τα Bits σαν στοιχεία πληροφορίας χρησιμοποιούνται είτε αυτόνομα (αναπαριστώντας όπως είπαμε τις καταστάσεις ON ή OFF) ή χρησιμοποιούνται σε συνδυασμό με συγκεκριμένη κωδικοποίηση αναπαριστώντας αριθμούς.

Στα μαθηματικά υπάρχουν διάφορα συστήματα αρίθμησης. Στην τεχνολογία του αυτοματισμού και των PLC χρησιμοποιούνται κατά βάση δύο: το δεκαδικό(αυτό που έχει καθιερωθεί παγκοσμίως να χρησιμοποιούμε παντού, στην καθημερινή μας ζωή) και το δυαδικό. Όλα τα συστήματα αρίθμησης έχουν 3 κοινά χαρακτηριστικά: τα "ψηφία" (digits), τη "βάση" (base) και το "βάρος" (weight).

Το δεκαδικό σύστημα βασίζεται στον αριθμό 10 και έχει τα ακόλουθα χαρακτηριστικά:

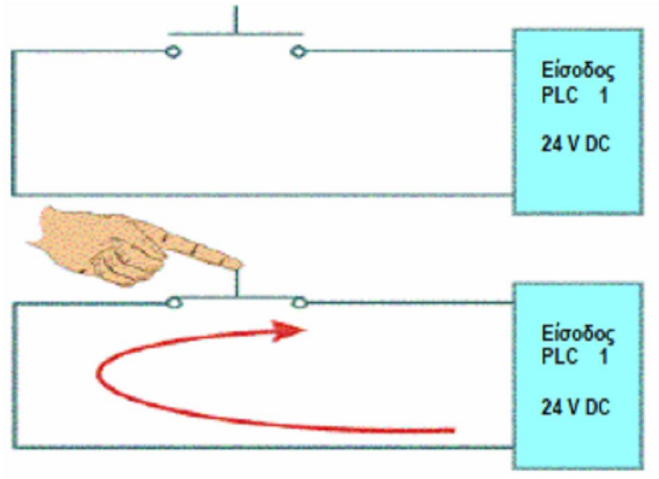
10 ψηφία: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Βάση: 10
Βάρη: 1, 10, 100, 1000, … ("δυνάμεις" ή πολλαπλάσια του 10)

Το δυαδικό σύστημα είναι αυτό που χρησιμοποιείται από τα PLC και έχει τα ακόλουθα χαρακτηριστικά:

2 ψηφία: 0, 1
Βάση: 2
Βάρη: 1, 2, 4, 8, 16, 32 ("δυνάμεις" ή πολλαπλάσια του 2)

2.3 Η λογική των ψηφιακών υπολογιστικών συστημάτων όπως τα PLC(Λογικό 0 – Λογικό 1).

Μία συσκευή PLC μπορεί να "αντιληφθεί" και να επεξεργασθεί πληροφορίες είτε σε ψηφιακή μορφή (0-1 ή ON-OFF): π.χ. "πατημένο" ή "μη πατημένο" μπουτόν, τερματοδιακόπτης, κλειστή ή ανοιχτή επαφή ή σε αναλογική μορφή: πληροφορία από σύστημα μέτρησης στάθμης, θερμοκρασίας, πίεσης, βάρους (ζυγιστικό) κλπ.



Σχήμα 2.3:Αντίληψη λογικού 0 – λογικού 1από το PLC

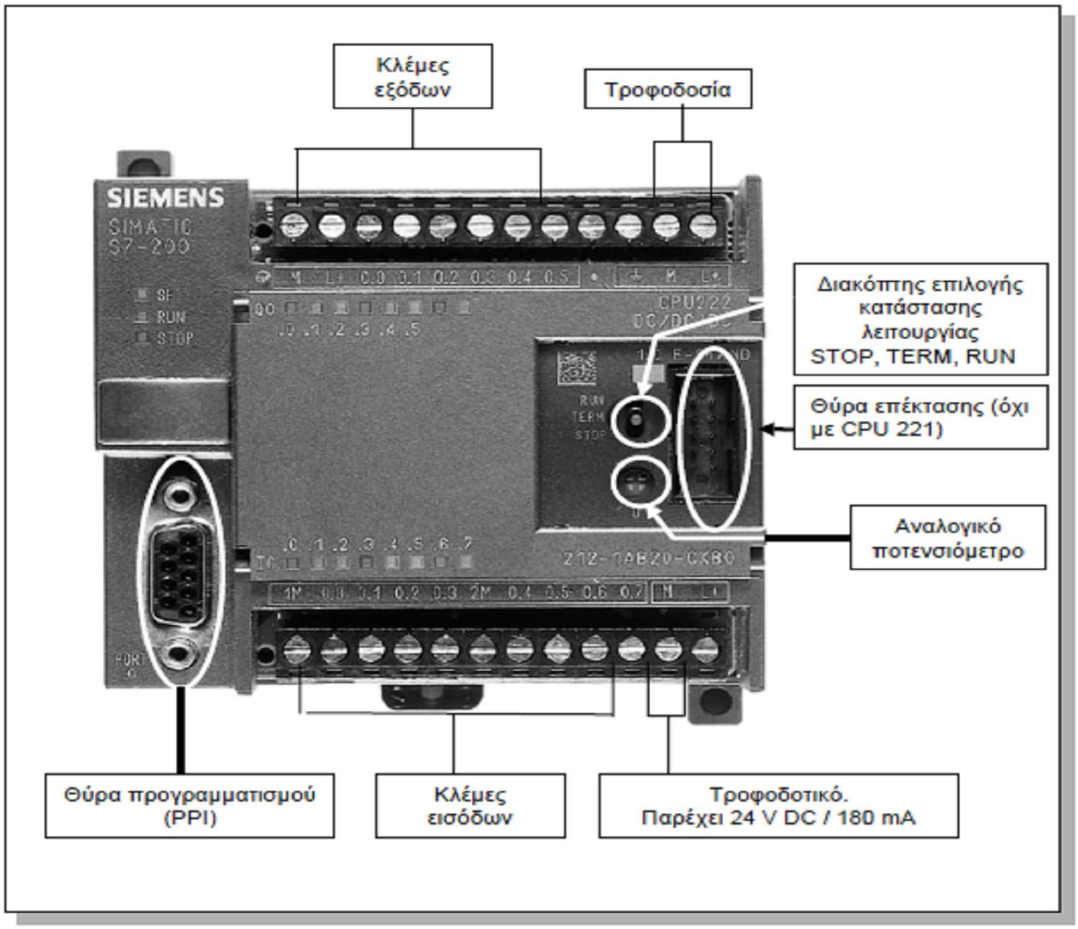
Παρ' όλα αυτά η "καρδιά" ή ο "εγκέφαλος" του PLC, ή αλλιώς η CPU που είναι ένα ολοκληρωμένο ψηφιακό κύκλωμα (microchip) "αντιλαμβάνεται" **ΜΟΝΟ** ψηφιακά δυαδικά σήματα "ON" – "OFF" ή λογικά 1 και 0. "Αντιλαμβάνεται" δηλαδή ΜΟΝΟ την ύπαρξη της κατάστασης "ON", του λογικού 1 ή στην πράξη της διέλευσης - ροής ηλεκτρικού σήματος που σχετίζεται με μία "κλειστή επαφή", με ένα "κλειστό διακόπτη" ή το ακριβώς αντίθετο: της κατάστασης "OFF" ή του λογικού 0 ή στην πράξη της μη διέλευσης ρεύματος (μη ροής ηλεκτρικού σήματος) που σχετίζεται με μία "ανοιχτή επαφή", με ένα "ανοικτό διακόπτη". Εννοείται ότι στα ψηφιακά microchip κυκλώματα δεν υπάρχουν διακόπτες και επαφές όπως στα ηλεκτρολογικά κυκλώματα (με ρελέ κλπ.) αλλά εκατοντάδες ή χιλιάδες κυκλώματα ημιαγωγών

(transistor κλπ.) σε μικροσκοπική μορφή. Ωστόσο η αρχή λειτουργίας παραμένει η ίδια και βασίζεται στη διέλευση ή μη ηλεκτρικού σήματος(ρεύματος) που μεταφράζεται σε κατάσταση “ON” ή “OFF”, σε κατάσταση δηλαδή λογικού 1 ή 0.

2.4 ΕΝΝΟΙΕΣ BIT , BYTE , WORD , DOUBLE WORD.

Το bit είναι η μικρότερη πληροφοριακή μονάδα και όπως έχουμε ήδη πει μπορεί να έχει μόνο δύο τιμές ΄0΄και ΄1΄. Μια ομάδα από 8 συνεχόμενα bit ορίζεται ως byte. Δύο συνεχόμενα byte ορίζουν μια word και δύο συνεχόμενα word ορίζουν μια double word.

Πιο συγκεκριμένα σ΄αυτή την πτυχιακή εργασία θα ασχοληθούμε με το PLC S7-200 της SIEMENS.Η δομή αυτού φαίνεται στο παρακάτω σχήμα.



Σχήμα 2.4: Η δομή ενός S7-200

Όπως παρατηρούμε στο παραπάνω σχήμα το συγκεκριμένο PLC ανήκει στην κατηγορία των compact (συμπαγούς μορφής) PLC.Το S7-200 με το χαρακτηριστικό κυβοειδές σχήμα έχουν ενσωματωμένο το υποσύστημα τροφοδοσίας και επίσης ενσωματωμένες εισόδους και εξόδους.

2.5 Αρχή λειτουργίας ενός προγραμματιζόμενου λογικού ελεγκτή.

Ας υποθέσουμε ότι ένα PLC βρίσκεται σε κατάσταση λειτουργίας του αυτοματισμού (RUN). Τα βήματα που ακολουθεί κατά τη λειτουργία του είναι τα εξής:

Βήμα 1ο. Στην αρχή ο μικροεπεξεργαστής διαβάζει τις εισόδους. Αυτό σημαίνει ότι για κάθε είσοδο ελέγχει αν έχει υψηλή τάση (λογικό 1) ή χαμηλή τάση (λογικό 0). Η τιμή 0 ή 1 για κάθε είσοδο αποθηκεύεται σε μια ειδική περιοχή μνήμης η οποία ονομάζεται εικόνα εισόδων. Την εικόνα εισόδων μπορείτε να τη φανταστείτε σαν ένα πίνακα, όπου ο μικροεπεξεργαστής σημειώνει τις τιμές, που διάβασε. Π.χ. είσοδος I1=1, I2=0, I3=0 και συνεχίζει.

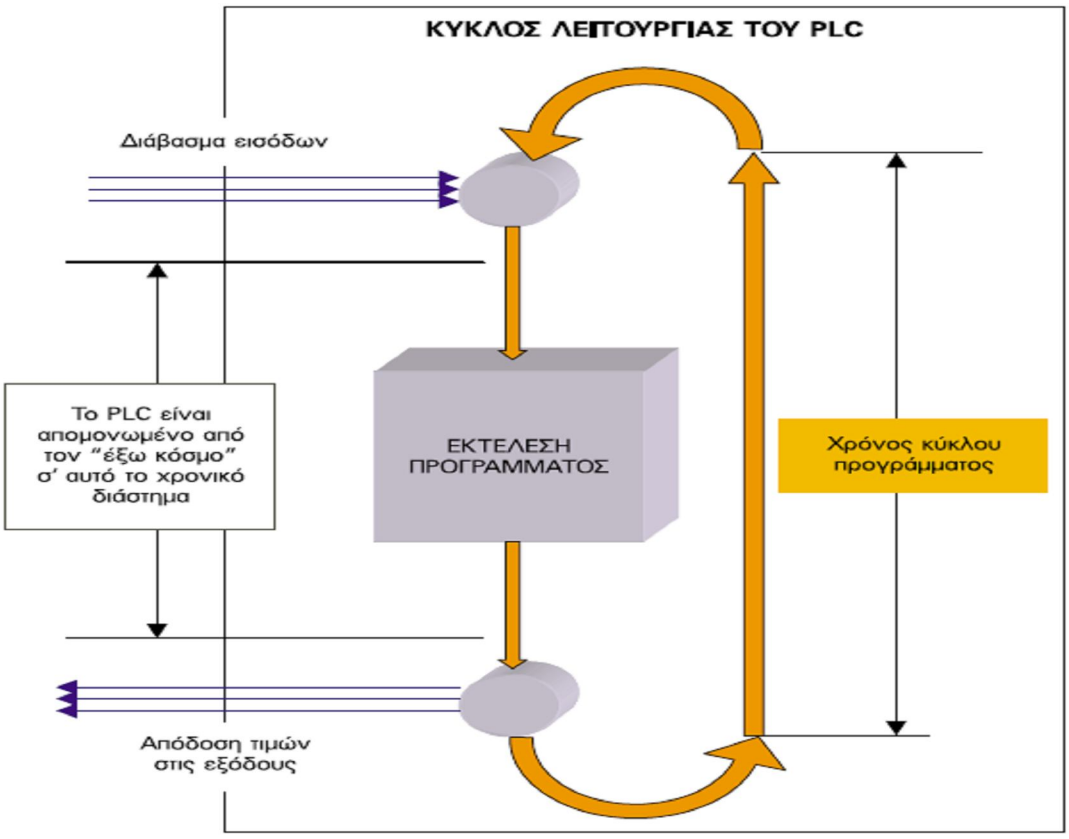
Βήμα 2ο. Στη συνέχεια ο μικροεπεξεργαστής χρησιμοποιώντας σαν δεδομένα τις τιμές των εισόδων, που διάβασε, εκτελεί τις εντολές του προγράμματος, το οποίο λειτουργεί τον αυτοματισμό. Το πρόγραμμα αυτό στην ουσία περιέχει μια σειρά από λογικές πράξεις.

Η εκτέλεση του προγράμματος θα δώσει αποτελέσματα για τις εξόδους. Τα αποτελέσματα αυτά αποθηκεύονται στην ειδική περιοχή της μνήμης που ονομάζεται εικόνα εξόδων. Όπως η εικόνα εισόδων, η εικόνα εξόδων περιέχει την τιμή (0 ή 1) για κάθε έξοδο, π.χ. Q1=1, Q2=1, Q3=0. Σημειώνουμε ότι οι τιμές αυτές προκύπτουν από την εκτέλεση των λογικών πράξεων του προγράμματος.

Βήμα 3ο. Στη συνέχεια ο μικροεπεξεργαστής αποδίδει τις τιμές της εικόνας εξόδων στις εξόδους. Αυτό σημαίνει ότι θα δοθεί υψηλή τάση σε όποια έξοδο έχει 1 και θα δοθεί χαμηλή τάση σε όποια έξοδο έχει 0.

Με τη συμπλήρωση του 3ου βήματος συμπληρώνεται ένας πλήρης κύκλος λειτουργίας και η διαδικασία ξαναρχίζει από την αρχή. Ο κύκλος λειτουργίας εκτελείται συνεχώς όσο το PLC βρίσκετε σε κατάσταση RUN. Δηλαδή ένα PLC εκτελεί συνεχώς τα βήματα του κύκλου λειτουργίας.

Ο χρόνος που χρειάζεται για να εκτελέσει το PLC ένα πλήρη κύκλο λειτουργίας ονομάζεται χρόνος κύκλου και εξαρτάται από τη ταχύτητα του μικροεπεξεργαστή του PLC, αλλά και από τον αριθμό και το είδος των εντολών του προγράμματος. Δηλαδή στο ίδιο PLC για ένα μεγαλύτερο πρόγραμμα έχουμε μεγαλύτερο χρόνο κύκλου. Ο χρόνος κύκλου αποτελεί και ένα μέτρο σύγκρισης μεταξύ των PLC. Για να μπορούν να συγκριθούν τα PLC ως προς τη ταχύτητα εκτέλεσης ενός προγράμματος, ορίζουμε το μέσο χρόνο κύκλου, σαν το χρόνο κύκλου ενός προγράμματος που περιλαμβάνει 1 Kbyte δυαδικές εντολές. Πάντως στη χειρότερη περίπτωση και σε ένα αργό PLC, ο χρόνος κύκλου δεν ξεπερνά μερικές εκατοντάδες χιλιοστά του δευτερολέπτου. Στο παρακάτω σχήμα απεικονίζεται σχηματικά ο κύκλος λειτουργίας ενός PLC.



Σχήμα 2.5: Κύκλος λειτουργίας PLC.

2.6 Οι προγραμματιζόμενοι λογικοί ελεγκτές της αγοράς.

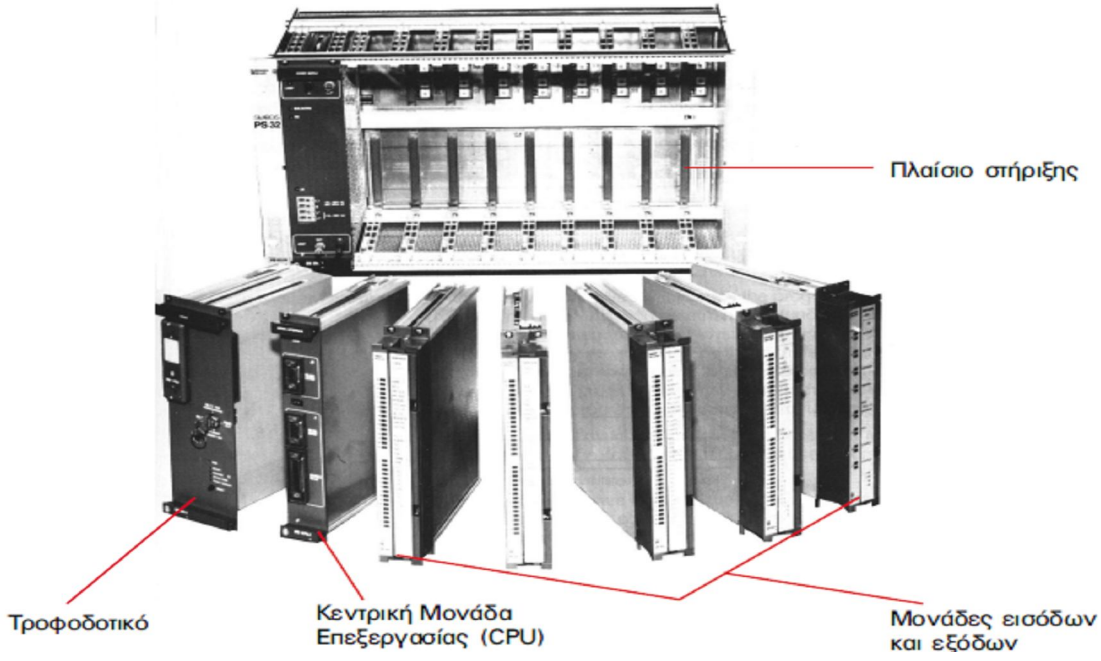
Τα PLC χωρίζονται σε δύο κατηγορίες (ανεξάρτητες των εταιρειών). Τα compact (συμπαγούς μορφής) και τα modular (δομοστοιχειωτής δομής).

α) Compact PLC

Όλες οι εταιρείες διαθέτουν PLC, στα οποία όλες οι μονάδες τους (τροφοδοσίας, κεντρική μονάδα και μονάδες εισόδων - εξόδων) είναι ενσωματωμένες σε μια συσκευή. Σ αυτού του είδους τα PLC είσοδοι και έξοδοι είναι συνήθως μέχρι 20 και όλες οι είσοδοι (ή έξοδοι) έχουν τα ίδια τεχνικά χαρακτηριστικά.

β) Modular PLC

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.



Σχήμα 2.6: Modular PLC. Αποτελείται από ανεξάρτητες μονάδες οι οποίες προσαρμόζονται στο πλαίσιο στήριξης.

Σχήμα 2.7: Compact PLC. Περιλαμβάνει τροφοδοτικό, κεντρική μονάδα επεξεργασίας, εισόδους και εξόδους, όλα ενσωματωμένα σε μια ενιαία συσκευή.

2.7 Τα πλεονεκτήματα των προγραμματιζόμενων λογικών ελεγκτών.

Τα πρώτα μεγάλα πλεονεκτήματα των PLC αφορούν τους κατασκευαστές εξοπλισμού αυτοματισμών και πινάκων αυτοματισμού:

• Το κόστος κατασκευής ενός PLC είναι σημαντικά μικρότερο από το κόστος παραγωγής ενός μεγάλου αριθμού βοηθητικών ηλεκτρονόμων, χρονικών και απαριθμητών.

• Ο χρόνος κατασκευής του αυτοματισμού είναι μηδαμινός σε σχέση με την κατασκευή ενός κλασικού πίνακα αυτοματισμού.

Υπάρχουν όμως πολλά πλεονεκτήματα που έχουν σχέση με τον τελικό χρήστη, τις βιομηχανίες δηλαδή που εφαρμόζουν τους αυτοματισμούς, και είναι αυτά που μας ενδιαφέρουν περισσότερο.

• Τα PLC ελαχιστοποιούν το κόστος συντήρησης του πίνακα αυτοματισμού. Το κόστος αυτό αναλύεται ως εξής: Συχνότητα βλαβών, χρόνος εντοπισμού μιας βλάβης και αποκατάστασής της. Δηλαδή, όταν υπάρχει μια βλάβη στον πίνακα μιας εγκατάστασης κλασικού αυτοματισμού, υπάρχει καθυστέρηση στην παραγωγή μέχρι να εντοπιστεί η βλάβη. Αφού εντοπιστεί, πρέπει να έχουμε διαθέσιμο το κατάλληλο ανταλλακτικό στην αποθήκη, γιατί διαφορετικά θα υπάρξει σημαντική καθυστέρηση,

αφού θα χρειαστεί να γίνει η σχετική παραγγελία και η προμήθεια. Στον αυτοματισμό με PLC δεν υπάρχει ουσιαστικό θέμα βλάβης εσωτερικά στον πίνακα της εγκατάστασης. Θα πείτε, δεν χαλά το PLC; Αυτό συμβαίνει σπάνια και οι εγγυήσεις είναι πάρα πολύ μεγάλες.

• Τα PLC είναι ευέλικτα στην τροποποίηση της λειτουργίας του αυτοματισμού. Δηλαδή, αν υποθέσουμε ότι θέλουμε να κάνουμε μια αλλαγή στον αυτοματισμό, αυτή μπορεί να γίνει μέσα σε λίγα λεπτά, αρκεί μόνο να αλλάξουμε το πρόγραμμα. Σε ένα πίνακα κλασικού αυτοματισμού, τέτοιου είδους αλλαγές είναι πράγμα πολύ δύσκολο και χρονοβόρο.

• Ο αυτοματισμός με PLC επεκτείνεται πολύ εύκολα. Αυτό γίνεται είτε απλά αλλάζοντας το πρόγραμμα, είτε με την τοποθέτηση νέων μονάδων εισόδων και εξόδων. Κάθε επέκταση στον κλασικό αυτοματισμό είναι πολύ δύσκολη.

• Ο αυτοματισμός με PLC μας παρέχει καταπληκτικές δυνατότητες. Μπορούμε να δημιουργούμε πολύ εύκολα πολύπλοκες και έξυπνες επεξεργασίες, οι οποίες στον κλασικό αυτοματισμό είναι εξαιρετικά δύσκολο να υλοποιηθούν.

• Σε μια εγκατάσταση, που χρησιμοποιεί αυτοματισμούς με PLC, σήμερα παρέχονται δυνατότητες σύνδεσης με κεντρικό ηλεκτρονικό υπολογιστή, σύνδεσης με το σύστημα αποθήκης, λογιστηρίου κλπ.

• Το PLC καταλαμβάνει ελάχιστο χώρο σε σχέση με τον αντίστοιχο πίνακα κλασικού αυτοματισμού.

Βλέπουμε ότι από τη χρήση των PLC προκύπτουν μόνο πλεονεκτήματα. Υπάρχουν άραγε μειονεκτήματα; Θα μπορούσαμε ίσως να θεωρήσουμε μειονέκτημα την έλλειψη επαρκούς ενημέρωσης των τεχνικών όλων των βαθμίδων, ειδικά στην Ελλάδα, πράγμα το οποίο δυσκολεύει και δημιουργεί προβλήματα στην εφαρμογή των PLC.

3. Προγραμματισμός ενός προγραμματιζόμενου λογικού ελεγκτή.

Το βασικότερο κομμάτι σε ένα σύστημα αυτοματισμού με PLC δεν είναι το υλικό μέρος αλλά το λογισμικό, δηλαδή το πρόγραμμα που υλοποιεί τον επιθυμητό αυτοματισμό. Το πρόγραμμα αναπτύσσεται σε μια γλώσσα προγραμματισμού. Δυστυχώς στα PLC δεν υπήρξε τυποποίηση σε κανέναν τομέα, λόγω του ανταγωνισμού των εταιρειών, ούτε βέβαια στο θέμα των γλωσσών προγραμματισμού. Δηλαδή δεν υπάρχουν γλώσσες προγραμματισμού για PLC που να ισχύουν ανεξάρτητα από εταιρεία, όπως για παράδειγμα συμβαίνει στον προγραμματισμό των ηλεκτρονικών υπολογιστών. Παρ όλα αυτά οι γλώσσες των PLC των διαφόρων εταιρειών μοιάζουν πολύ μεταξύ τους, έτσι που να μπορούμε να μιλάμε σήμερα για μια τυποποίηση της αγοράς.

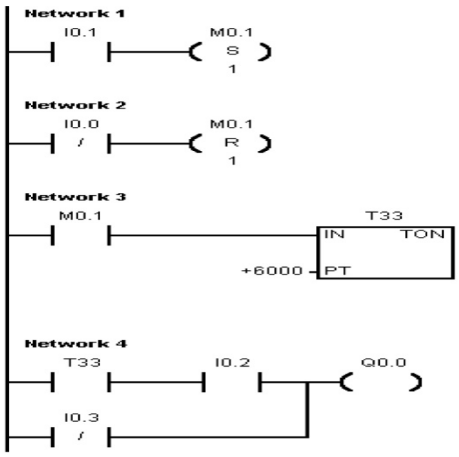
3.1 Γλώσσες προγραμματισμού των προγραμματιζόμενων λογικών ελεγκτών.

Τρεις είναι σήμερα οι κυριότερες κατηγορίες γλωσσών προγραμματισμού για PLC, τις οποίες συναντούμε με μικρές διαφορές στα PLC όλων των εταιρειών:

1. Γλώσσα LADDER ή γλώσσα ηλεκτρολογικών γραφικών.

Είναι η πρώτη γλώσσα που αναπτύχθηκε ιστορικά. Η γλώσσα Ladder στην ουσία επιτρέπει τη μεταφορά του ηλεκτρολογικού σχεδίου, μέσω της συσκευής προγραμματισμού στο PLC. Με τη γλώσσα αυτή η εκπαίδευση των τεχνικών, που ήταν συνηθισμένοι στον κλασικό αυτοματισμό, γινόταν εύκολα και γρήγορα, αφού δεν άλλαζε ουσιαστικά την εργασία σχεδιασμού του αυτοματισμού. Η γλώσσα LADDER χρησιμοποιεί όχι την Ευρωπαϊκή προτυποποίηση στο σχεδιασμό των ηλεκτρικών επαφών, αλλά την Αμερικάνικη. Αυτό ίσως οφείλεται στο γεγονός ότι τα

πρώτα PLC αναπτύχθηκαν στην Αμερική. Όμως στη συνέχεια ο τρόπος αυτός σχεδιασμού βόλεψε και έτσι διατηρήθηκε και από τις Ευρωπαϊκές εταιρείες, με αποτέλεσμα σήμερα να είναι πλέον καθιερωμένος.



Σχήμα 3.1: Πρόγραμμα σε γλώσσα LADDER

2. Γλώσσα STL (Statement List) ή γλώσσα λογικών εντολών.

Η γλώσσα αυτή αναπτύχθηκε σχεδόν ταυτόχρονα με τη γλώσσα LADDER, αν και οι εταιρείες έδειξαν στην αρχή δισταγμό στο να την προωθήσουν, φοβούμενες μην τρομάξουν το τεχνικό κατεστημένο της βιομηχανίας. Η γλώσσα αυτή δημιουργεί λίστα προγράμματος με εντολές, οι οποίες αντιστοιχούν στις λογικές πύλες (AND, OR, NOT κ.λπ.).

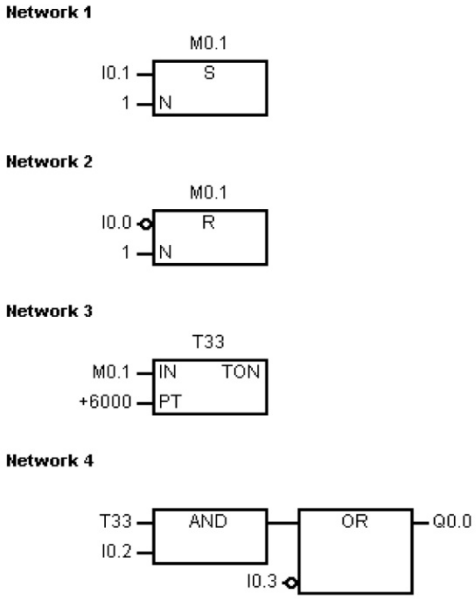
Στην αρχή η γλώσσα λίστα εντολών ήταν πολύ φτωχή και περιοριζόταν μόνο στις βασικές λογικές εντολές, οι οποίες αντιστοιχούσαν αμέσως στις γραφικές εντολές της γλώσσας LADDER. Σήμερα οι γλώσσες αυτές έχουν εξελιχθεί πάρα πολύ και συναντά κανείς σε αυτές στοιχεία από τις γλώσσες των υπολογιστών και κυρίως των γλωσσών Assembly. Ο προγραμματισμός σε λίστα εντολών απαιτεί από τον ηλεκτρολόγο να έχει έστω στοιχειώδεις γνώσεις προγραμματισμού.

```
NETWORK 1
LD      I0.1
S       M0.1, 1

NETWORK 2
LDN     I0.0
R       M0.1, 1

NETWORK 3
LD      M0.1
TON     T33, +6000

NETWORK 4
LD      T33
A       I0.2
ON      I0.3
=       Q0.0
```

Σχήμα 3.2: Πρόγραμμα σε γλώσσα λογικών εντολών(STL)

3. Γλώσσα FBD (Function block diagram) λογικών γραφικών ή λογικού διαγράμματος.

Η γλώσσα αυτή είναι επίσης γραφική, αλλά αντί του ηλεκτρολογικού σχεδίου του αυτοματισμού, χρησιμοποιεί το αντίστοιχο λογικό κύκλωμα. Η γλώσσα αυτή είναι νεότερη και δεν χρησιμοποιείται από όλες τις εταιρείες.



Σχήμα 3.3: Πρόγραμμα σε γλώσσα λογικών γραφικών(FBD)

3.2 Συσκευές προγραμματισμού των προγραμματιζόμενων λογικών ελεγκτών.

Αφού συντάξουμε το πρόγραμμα στο χαρτί σε οποιαδήποτε γλώσσα προγραμματισμού, πρέπει να το εισάγουμε στο PLC. Αυτό συνήθως γίνεται μέσω μιας συσκευής προγραμματισμού, ενός προγραμματιστή, που συνδέεται με το PLC. Ορισμένα μικρά PLC προγραμματίζονται με τη βοήθεια ενός αριθμού πλήκτρων που είναι ενσωματωμένα επάνω στη συσκευή του PLC και δε χρειάζονται συσκευή προγραμματισμού.

Μια συσκευή προγραμματισμού μπορεί να είναι μιας από τις παρακάτω μορφές:

1) Ειδικός προγραμματιστής χειρός.

Κάθε PLC συνοδεύεται από μια ειδική συσκευή προγραμματιστή, η οποία είναι συνήθως χειρός, δηλαδή φορητή. Αυτές οι συσκευές προγραμματισμού διαθέτουν μια μικρή οθόνη υγρών κρυστάλλων και τυποποιημένα πλήκτρα προγραμματισμού. Συνήθως οι ειδικοί προγραμματιστές μπορούν να προγραμματίσουν τα PLC μόνο σε γλώσσα λίστα εντολών. Υπάρχουν όμως και προγραμματιστές με τους οποίους μπορούμε να προγραμματίσουμε και σε κάποια από τις γραφικές γλώσσες. Για να προγραμματίσουμε το PLC πρέπει να το συνδέσουμε με τον προγραμματιστή. Η σύνδεση πραγματοποιείται μέσω της ειδικής θύρας που υπάρχει στην κεντρική μονάδα επεξεργασίας του PLC. Αφού πληκτρολογήσουμε το πρόγραμμα, το μεταφέρουμε στη μνήμη του PLC. Όταν ολοκληρώσουμε τη διαδικασία αυτή, ο προγραμματιστής μπορεί να αποσυνδεθεί. Ο τρόπος χειρισμού του προγραμματιστή είναι τελείως ειδικός για κάθε PLC. Οι προγραμματιστές των διαφόρων εταιριών δεν μοιάζουν πολύ μεταξύ τους και αυτό είναι μια δυσκολία στην εκμάθηση του προγραμματισμού ενός νέου PLC.

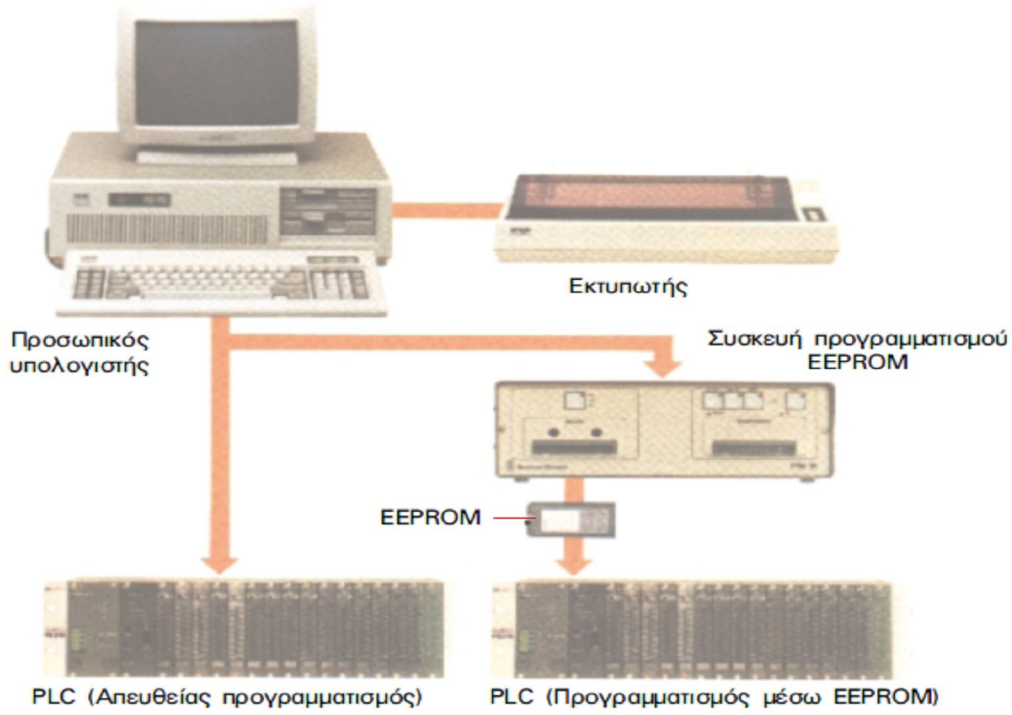
Σχήμα 3.4: Προγραμματιστής χειρός PLC.

Οι προγραμματιστές χειρός σήμερα διαθέτουν και άλλες δυνατότητες, όπως για παράδειγμα:
Μπορούν να συνδεθούν με εκτυπωτή, για να εκτυπώσουμε το πρόγραμμα.
Μπορούν να συνδεθούν με προσωπικό υπολογιστή (PC) με όσα πλεονεκτήματα μπορεί αυτό να έχει, π.χ. μπορούμε να αποθηκεύσουμε σε δισκέτα το πρόγραμμα, να κάνουμε εκτύπωση του προγράμματος κ.λπ.
Μπορούν να συνδεθούν με ειδική συσκευή προγραμματισμού EEPROM, με την οποία μπορούμε να θέτουμε το πρόγραμμα σε πλακέτες EEPROM.
Ακόμη με τον προγραμματιστή μπορούμε να ελέγχουμε την λειτουργία του προγράμματος αυτοματισμού και να κάνουμε ανίχνευση βλαβών. Αυτή η δυνατότητα είναι ίσως το σημαντικότερο πλεονέκτημα του προγραμματιστή χειρός, γιατί μπορεί να μεταφερθεί σε οποιαδήποτε εγκατάσταση PLC, να συνδεθεί στο PLC και να ψάξουμε για βλάβες στη λειτουργία του αυτοματισμού.

2) Προσωπικός υπολογιστής (PC) και χρήση ειδικού λογισμικού.

Ο πιο εύκολος τρόπος προγραμματισμού ενός PLC σήμερα είναι μέσω ενός προσωπικού υπολογιστή (PC). Με την χρήση ειδικού λογισμικού, το οποίο δίνεται από την εταιρεία, το PC μετατρέπεται σε προγραμματιστή. Για τη σύνδεση του PC με το PLC ή με την συσκευή προγραμματισμού EEPROM χρειάζεται ειδική κάρτα σύνδεσης (interface), η οποία τοποθετείται στο PC. Ο προγραμματισμός μέσω PC είναι πολύ ευκολότερος από τον προγραμματισμό με τον ειδικό προγραμματιστή χειρός, ειδικά για κάποιον που είναι εξοικειωμένος με την χρήση του PC. Ο

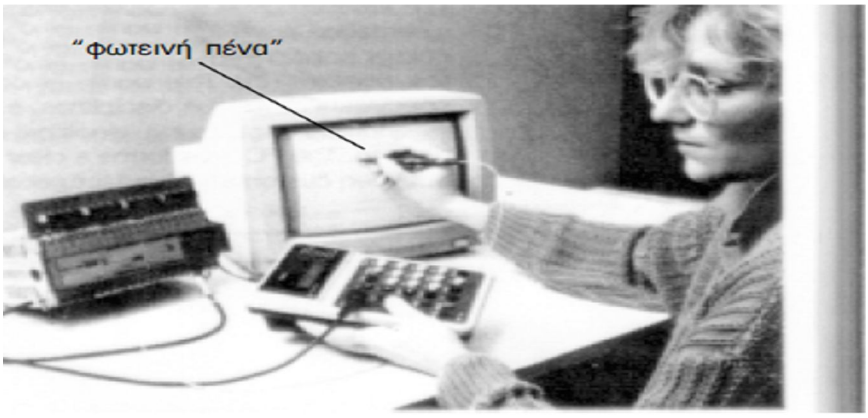
προγραμματισμός στις γραφικές γλώσσες γίνεται με τρόπο ιδανικό στην οθόνη του PC. Τα υπόλοιπα πλεονεκτήματα νομίζουμε είναι προφανή, δηλαδή:
Μπορούμε να αποθηκεύουμε και να αρχειοθετούμε τα προγράμματά μας.
Μπορούμε να τυπώνουμε τα προγράμματα.



Σχήμα 3.5: Προγραμματισμός PLC με την βοήθεια προσωπικού υπολογιστή.

3) Ειδικές συσκευές προγραμματισμού.

Εκτός από τις δύο μορφές που προαναφέραμε, υπάρχουν κάποιες ειδικές συσκευές με τις οποίες ο προγραμματισμός, κυρίως στις γραφικές γλώσσες γίνεται πολύ εύκολα. Μία τέτοια συσκευή είναι η φωτεινή πένα (light pen). Πρόκειται για μια συσκευή η οποία περιλαμβάνει μια οθόνη, επάνω στην οποία σχεδιάζουμε με μια ειδική φωτεινή πένα.



Σχήμα 3.6: Προγραμματισμός PLC με φωτεινή πένα (Light Pen).

4. Ανάπτυξη προγράμματος σε προγραμματιζόμενο λογικό ελεγκτή.

Στις ενότητες που ακολουθούν θα δούμε, πως προγραμματίζουμε ένα PLC. Το πρόβλημα που υπάρχει σχετικά με τον προγραμματισμό των PLC είναι αυτό στο οποίο έχουμε ήδη αναφερθεί, δηλαδή το γεγονός ότι οι γλώσσες προγραμματισμού των PLC δεν είναι τυποποιημένες, αλλά διαφέρουν από εταιρεία σε εταιρεία. Διαφέρουν ακόμη και μεταξύ των μοντέλων της ίδιας εταιρείας. Βέβαια η λογική όλων των γλωσσών προγραμματισμού σε όλα τα PLC είναι ίδια. Αλλά και οι εντολές προγραμματισμού στις διάφορες γλώσσες μοιάζουν μεταξύ τους σε ένα σημαντικό ποσοστό. Έτσι, όποιος μάθει να χρησιμοποιεί πολύ καλά τις γλώσσες προγραμματισμού ενός μοντέλου PLC, αρκετά εύκολα μαθαίνει τις γλώσσες προγραμματισμού ενός άλλου μοντέλου, εντοπίζοντας πολύ γρήγορα τις διαφορές.

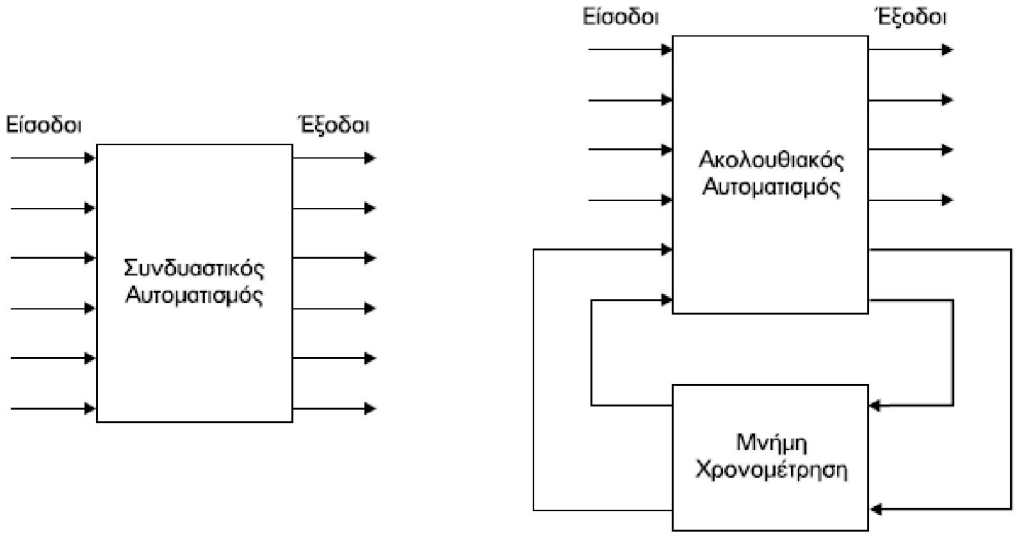
Θα παρουσιάσουμε τον προγραμματισμό των PLC σε δύο ενότητες. Στην πρώτη ενότητα θα δούμε πως προγραμματίζουμε σε ένα PLC συνδυαστικούς αυτοματισμούς και στη δεύτερη ενότητα πως προγραμματίζουμε ακολουθιακούς αυτοματισμούς. Αυτό το κάνουμε, γιατί οι βασικές διαφορές στον προγραμματισμό των PLC εμφανίζονται, όταν έχουμε χρήση χρονικών, απαριθμητών και των λοιπών ειδικών συναρτήσεων των ακολουθιακών αυτοματισμών.

•Συνδυαστικοί αυτοματισμοί

Συνδυαστικό αυτοματισμό ονομάζουμε τον αυτοματισμό εκείνο στον οποίον οι έξοδοι εξαρτώνται μόνο από τις εισόδους. Αυτό σημαίνει ότι οι κινητήρες, βαλβίδες και οι λοιποί αποδέκτες του αυτοματισμού λαμβάνουν εντολές μόνο από τους αισθητήρες και τους διακόπτες εισόδου και δεν εξαρτώνται από το χρόνο ή από προηγούμενες καταστάσεις των εξόδων.

•Ακολουθιακοί αυτοματισμοί

Ακολουθιακό αυτοματισμό ονομάζουμε τον αυτοματισμό εκείνο, στον οποίο οι έξοδοι εξαρτώνται όχι μόνο από τις εισόδους, αλλά και από το χρόνο ή και από προηγούμενες καταστάσεις των εξόδων.
Σχηματικά οι δύο κατηγορίες αυτοματισμών φαίνονται στο παρακάτω σχήμα:



Σχήμα 4.1:Σχηματική αναπαράσταση συνδυαστικού και ακολουθιακού αυτοματισμού.

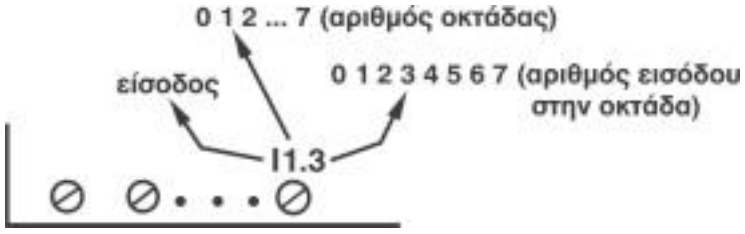
4.1 Προγραμματιστικά χαρακτηριστικά και ονοματολογία των στοιχείων ενός προγραμματιζόμενου λογικού ελεγκτή.

Όταν ξεκινάμε να μελετάμε, πως θα προγραμματίσουμε ένα PLC, πρέπει να γνωρίζουμε:

- Πόσες εισόδους έχει, πως τις ονομάζουμε και πως τις αναγνωρίζουμε.

Οι είσοδοι σχεδόν σε όλα τα PLC χαρακτηρίζονται με το γράμμα I (Input). Το κάθε γράμμα συνοδεύεται από δύο αριθμούς που χωρίζονται από μια τελεία (π.χ. είσοδος I1.3).

Αν χρησιμοποιείτε συσκευή με 16 εισόδους θα δείτε ότι η αρίθμηση δεν είναι από I1 έως I16.Αντίθετα χωρίζονται σε δύο ομάδες των οκτώ και ο πρώτος αριθμός μετά το γράμμα μας δίνει τον αριθμό της οκτάδας. Κάθε οκτάδα αντιστοιχεί σε ένα byte. Επίσης παρατηρήστε ότι η αρίθμηση ξεκινά από το μηδέν. Έτσι, π.χ. η είσοδος I1.3 αντιστοιχεί στην τέταρτη είσοδο της δεύτερης ομάδας εισόδων, δηλαδή στην 8+4=12 είσοδο, όπως φαίνεται στο σχήμα.



Σχήμα 4.2: Συμβολισμός εισόδων

- Πόσες εξόδους έχει, πως τις ονομάζουμε και πως τις αναγνωρίζουμε.

Τα ίδια, που ισχύουν για τις εισόδους, ισχύουν και για τις εξόδους. Το γράμμα με το οποίο χαρακτηρίζονται οι έξοδοι στα διάφορα PLC είναι συνήθως το Q ή το O (Output). Για τους αριθμούς, που ακολουθούν το γράμμα, ισχύει ότι και για τις εισόδους.

- Πόσες βοηθητικές μνήμες έχει και πως τις ονομάζουμε.

Στα διάφορα PLC θα τις συναντήσουμε με το όνομα Markers ή Flags. Πρόκειται για θέσεις μνήμης, στις οποίες αποθηκεύονται ενδιάμεσες λογικές καταστάσεις και πληροφορίες. Όπως ισχύει για τις εισόδους και τις εξόδους, χαρακτηρίζονται με ένα γράμμα ακολουθούμενο από έναν αριθμό ή δύο αριθμούς, που χωρίζονται με τελεία. Το γράμμα στα διάφορα PLC είναι το M (Marker) ή το F (Flag). Ο αριθμός τους συνήθως είναι πάνω από 1000 και είναι οργανωμένες σε οκτάδες.

- Τις ειδικές συναρτήσεις του PLC.

Πρέπει να γνωρίζουμε ποιες είναι, πως ονομάζονται, πως τις χειρίζεται το συγκεκριμένο PLC και πόσες από την κάθε μία διαθέτει. Οι ειδικές συναρτήσεις κατά σειρά σπουδαιότητας είναι:

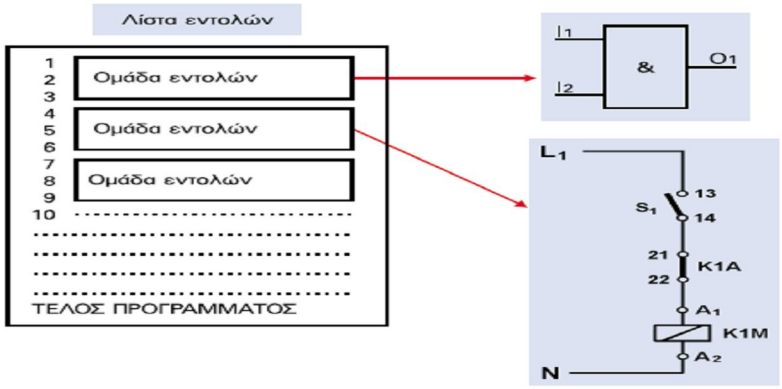
•Τα χρονικά.
•Οι απαριθμητές.
•Οι συγκριτές.
•Οι γεννήτριες παλμοσειρών.
•Ο μετρητής πραγματικού χρόνου.

Όλα τα παραπάνω στοιχεία λέμε ότι αποτελούν το προγραμματιστικό μοντέλο ενός PLC. Για να ξεκινήσουμε τον προγραμματισμό πρέπει να γνωρίζουμε το προγραμματιστικό μοντέλο του συγκεκριμένου PLC, που διαθέτουμε.

4.2 Ανάπτυξη προγράμματος σε γλώσσα λίστα εντολών(STL).

- Μορφή του προγράμματος.

Το πρόγραμμα αποτελείται από μια σειρά εντολών. Κάθε εντολή αποτελεί μια γραμμή προγράμματος. Οι εντολές κατανέμονται σε ομάδες εντολών. Κάθε ομάδα εντολών αντιστοιχεί σε μία λογική πύλη, ή αλλιώς σε ένα κλάδο του ηλεκτρολογικού κυκλώματος αυτοματισμού.



Σχήμα 4.3: Μορφή προγράμματος στη γλώσσα λίστα εντολών.

- Μορφή εντολής.

Κάθε εντολή του προγράμματος αποτελείται από δύο μέρη. Το πρώτο μέρος καθορίζει την ενέργεια την οποία θα εκτελέσει το PLC, δηλαδή χαρακτηρίζει την ίδια την εντολή. Το δεύτερο μέρος καθορίζει την παράμετρο, δηλαδή καθορίζει σε ποια είσοδο, έξοδο, βοηθητική μνήμη κ.λπ. αναφέρεται η ενέργεια της εντολής.

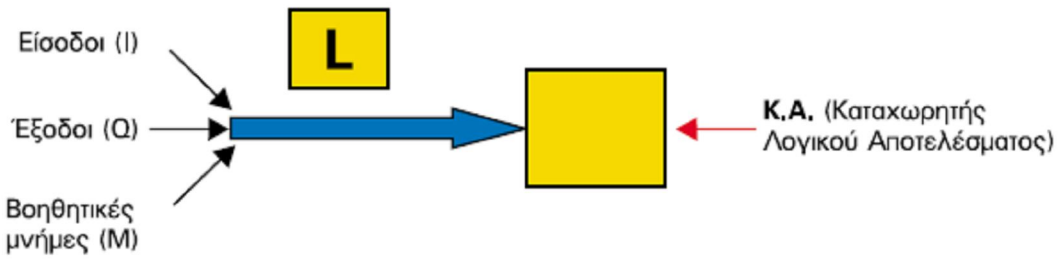
Σχήμα 4.4: Μορφή εντολής στη γλώσσα λίστα εντολών.

4.2.1 Βασικές εντολές προγραμματισμού στη γλώσσα λίστα εντολών.

Παρουσίαση εντολών.

• Η εντολή L (Load)

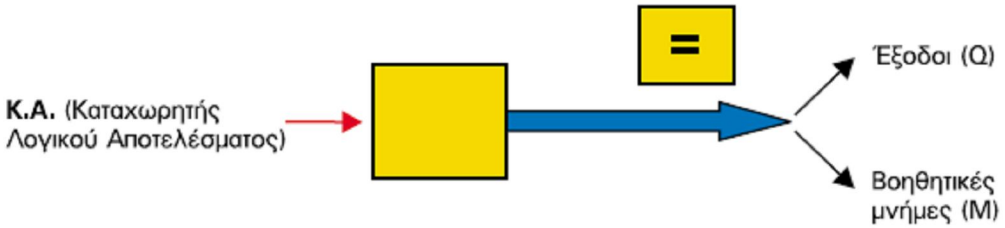
Το πρόγραμμα το οποίο αντιστοιχεί σε μια πύλη λογικού κυκλώματος (ή κλάδο ηλεκτρολογικού κυκλώματος αυτοματισμού) ξεκινά με την εντολή L (Load, φόρτωσε). Το PLC με την εντολή Load διαβάζει τη λογική κατάσταση (0 ή 1) μιας εισόδου, εξόδου, βοηθητικής μνήμης, χρονικού κ.λπ., και τη φορτώνει σε ένα καταχωρητή (μια ειδική θέση μνήμης) τον οποίο θα ονομάζουμε Καταχωρητή Λογικού Αποτελέσματος (Κ.Α.).
Η εντολή L μπορεί να αναφέρεται σε εισόδους, εξόδους, βοηθητικές μνήμες, χρονικά, κ.λπ. Π.χ. L I 0.1, L Q 0.2, L M 0.1.



Σχήμα 4.5: Σχηματική παράσταση της εντολής Load.

• Η εντολή = (ίσον)

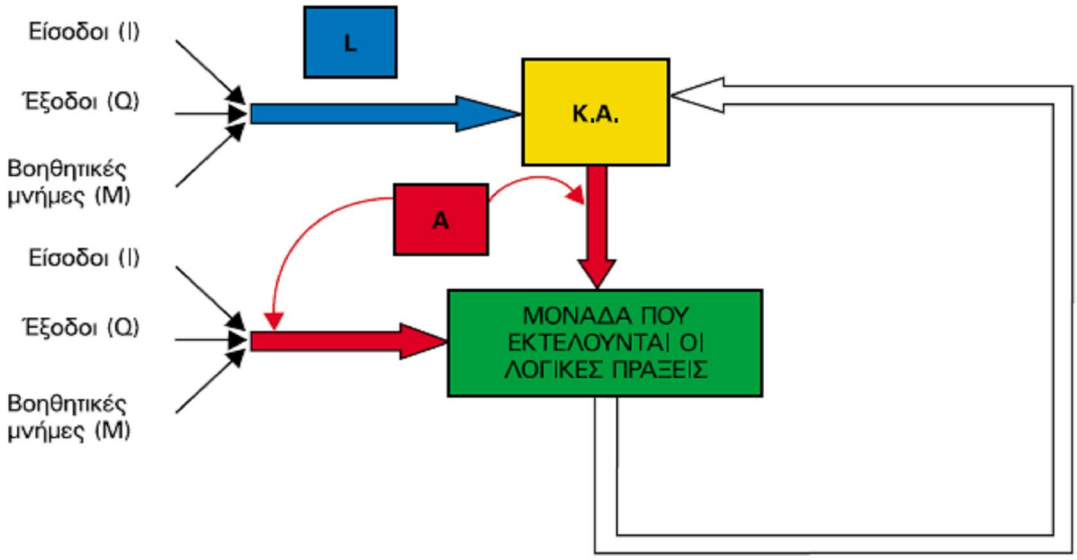
Το πρόγραμμα που αντιστοιχεί σε μια πύλη λογικού κυκλώματος (ή κλάδο ηλεκτρολογικού κυκλώματος αυτοματισμού) καταλήγει πάντα με την εντολή = (ίσον). Η εντολή αναφέρεται σε εξόδους και βοηθητικές μνήμες. Το PLC με την εντολή = μεταφέρει στις εξόδους ή στις βοηθητικές μνήμες το περιεχόμενο του καταχωρητή λογικού αποτελέσματος. Π.χ. = Q 1.2, = M 0.1



Σχήμα 4.6: Σχηματική παράσταση της εντολής =

• Η εντολή A (AND)

Η εντολή A υπαγορεύει στο PLC να εκτελέσει τη λογική πράξη AND. Η εντολή AND αναφέρεται σε εισόδους, εξόδους, βοηθητικές μνήμες, χρονικά κ.λπ. Η λογική πράξη γίνεται μεταξύ της λογικής κατάστασης της εισόδου, εξόδου, βοηθητικής μνήμης, χρονικού, κ.λπ. το οποίο αναφέρεται στην εντολή και του περιεχομένου του Κ.Α. Το αποτέλεσμα επιστρέφει στον Κ.Α. Προσέξτε το παρακάτω σχήμα, το οποίο αποδίδει πολύ καλά τον τρόπο που το PLC εκτελεί τη λογική πράξη AND.



Σχήμα 4.7: Σχηματική παράσταση της εντολής AND.

Παράδειγμα: Έστω ότι έχουμε δύο διακόπτες S1 και S2 (επαφές σε ηρεμία ανοικτές), συνδεδεμένους σε σειρά και θέλουμε να ανάβει η λυχνία L1 όταν είναι κλειστοί και οι δύο διακόπτες ταυτόχρονα.

Όπως έχουμε αναφέρει η ύπαρξη τάσης σε μια είσοδο-έξοδο του PLC μεταφράζεται σε σήμα ″1″. Έτσι όταν ρωτάμε για την ύπαρξη σήματος ″1″ η εντολή στο πρόγραμμα είναι η A I0.0. Για να φορτώσουμε την εντολή στην είσοδο που έχουμε επιλέξει χρησιμοποιούμε την εντολή LD I0.0. Έτσι όπως θα δούμε και παρακάτω οι εντολές LD I0.0 και LD I0.1 φορτώνουν τις εντολές (A) στις εισόδους I0.0 και I0.1. Η εντολή A I0.0 ρωτάει αν και στην είσοδο I0.0 έχουμε σήμα ″1″ και η εντολή A I0.1 ρωτάει αν και στην είσοδο I0.1 έχουμε σήμα ″1″. Αν ισχύουν οι δύο αυτές συνθήκες τότε ενεργοποιείται η έξοδος Q0.0 που έχουμε ορίσει με την εντολή = Q0.0 που αντιστοιχεί στην λυχνία L1. Παρακάτω ακολουθεί το πρόγραμμα και ο πίνακας που μας δείχνει τις πιθανές καταστάσεις και αντίστοιχα πώς συμπεριφέρεται η έξοδος.

```
LD   I0.0
LD   I0.1
A    I0.0
A    I0.1
=    Q0.0
END
```

ή

```
LD   I0.0
A    I0.1
=    Q0.0
END
```

I0.0	I0.1	Q0.0
0	0	0
0	1	0
1	0	0
1	1	1

Όπως παρατηρούμε η μοναδική περίπτωση που ανάβει η λυχνία εξόδου είναι όταν και οι δύο διακόπτες είναι κλειστοί δηλαδή έχουν σήμα ″1″.Σε κάθε άλλη περίπτωση η έξοδος δεν ενεργοποιείται.

• Η εντολή AN (AND NOT)

Η εντολή AN υπαγορεύει στο PLC να εκτελέσει τη λογική πράξη AND NOT, η οποία είναι στην ουσία το συμπλήρωμα της εντολής AND. Η εντολή AND NOT αναφέρεται σε εισόδους, εξόδους, βοηθητικές μνήμες, χρονικά κ.λπ. Η λογική πράξη γίνεται μεταξύ της λογικής κατάστασης της εισόδου, εξόδου, βοηθητικής μνήμης, χρονικού, κ.λπ. το οποίο αναφέρεται στην εντολή και του περιεχομένου του Κ.Α. Το αποτέλεσμα επιστρέφει στον Κ.Α.

Παράδειγμα: Έστω ότι έχουμε δύο διακόπτες S1 και S2 (επαφές σε ηρεμία ανοικτές), συνδεδεμένους σε σειρά και θέλουμε να ανάβει η λυχνία L1 όταν είναι ανοιχτοί και οι δύο διακόπτες ταυτόχρονα.

Σε αυτήν την περίπτωση η ερώτηση για σήμα μηδέν σε μια είσοδο-έξοδο του PLC γίνεται με την εντολή AN (AND NOT). Για να φορτώσουμε την εντολή αυτή στην είσοδο που έχουμε επιλέξει χρησιμοποιούμε την εντολή LDN (Load Not). Έτσι στο παρακάτω παράδειγμα οι εντολές LDN I0.0 και LDN I0.1 φορτώνουν τις εντολές AN I0.0 και AN I0.1 αντίστοιχα στους διακόπτες που έχουμε επιλέξει. Έτσι λοιπόν η εντολή AN I0.0 ρωτάει αν και στην είσοδο I0.0 έχουμε σήμα ″0″ και η εντολή AN I0.1 ρωτάει αν και στην είσοδο I0.1 έχουμε σήμα ″0″. Αν ισχύουν οι δύο αυτές συνθήκες τότε ενεργοποιείται η έξοδος Q0.0 που έχουμε ορίσει με την εντολή = Q0.0 που αντιστοιχεί στην λυχνία L1. Παρακάτω ακολουθεί το πρόγραμμα και ο πίνακας που μας δείχνει τις πιθανές καταστάσεις και αντίστοιχα πώς συμπεριφέρεται η έξοδος.

```
LDN   I0.0
LDN   I0.1
AN    I0.0
AN    I0.1
=     Q0.0
END
```

ή

```
LDN   I0.0
AN    I0.1
=     Q0.0
END
```

I0.0	I0.1	Q0.0
0	0	1
0	1	0
1	0	0
1	1	0

Όπως παρατηρούμε η μοναδική περίπτωση που ανάβει η λυχνία εξόδου είναι όταν και οι δύο διακόπτες είναι ανοιχτοί δηλαδή έχουν σήμα ″0″.Σε κάθε άλλη περίπτωση η έξοδος δεν ενεργοποιείται.

• Η εντολή Ο (OR)

Η εντολή OR υπαγορεύει στο PLC να εκτελέσει τη λογική πράξη OR. Η εντολή OR αναφέρεται σε εισόδους, εξόδους, βοηθητικές μνήμες, χρονικά, κ.λπ. Εκτελείται με ανάλογο τρόπο, με αυτόν που εκτελείται η εντολή AND.

Παράδειγμα: Έστω ότι έχουμε δύο διακόπτες S1 και S2 (επαφές σε ηρεμία ανοικτές), και θέλουμε να ανάβει η λυχνία L1 όταν πατηθεί τουλάχιστον ένας ή και οι δύο διακόπτες ταυτόχρονα.

Σε αυτήν την περίπτωση η ερώτηση για σήμα ″1″ σε μια είσοδο-έξοδο του PLC γίνεται με την εντολή Ο (OR). Για να φορτώσουμε την εντολή αυτή στην είσοδο που έχουμε επιλέξει χρησιμοποιούμε την εντολή LD (Load). Έτσι στο παρακάτω παράδειγμα οι εντολές LD I0.0 και LD I0.1 φορτώνουν τις εντολές Ο I0.0 και Ο I0.1 αντίστοιχα στους διακόπτες που έχουμε επιλέξει. Έτσι λοιπόν η εντολή Ο I0.0 ρωτάει είτε στην είσοδο I0.0 έχουμε σήμα ″1″ και η εντολή Ο I0.1 ρωτάει είτε στην είσοδο I0.1 έχουμε σήμα ″1″. Δηλαδή αν ένας από τους δύο διακόπτες είναι κλειστός ή και οι δύο είναι κλειστοί τότε ενεργοποιείται η έξοδος Q0.0 που έχουμε ορίσει με την εντολή = Q0.0 που αντιστοιχεί στην λυχνία L1. Παρακάτω ακολουθεί το πρόγραμμα και ο πίνακας που μας δείχνει τις πιθανές καταστάσεις και αντίστοιχα πώς συμπεριφέρεται η έξοδος.

```
LD   I0.0
LD   I0.1
O    I0.0
O    I0.1
=    Q0.0
END
```

ή

```
LD   I0.0
O    I0.1
=    Q0.0
END
```

I0.0	I0.1	Q0.0
0	0	0
0	1	1
1	0	1
1	1	1

Όπως παρατηρούμε όταν οι δύο διακόπτες είναι ανοιχτοί έχουμε κλειστή την λυχνία στην έξοδο. Σε οποιοδήποτε άλλο συνδυασμό των διακοπτών η λυχνία εξόδου θα παραμένει ανοιχτή.

• Η εντολή ON (OR NOT)

Η εντολή OR NOT υπαγορεύει στο PLC να εκτελέσει τη λογική πράξη OR NOT,η οποία είναι το συμπλήρωμα της εντολής OR. Η εντολή OR NOT αναφέρεται σε εισόδους, εξόδους, βοηθητικές μνήμες, χρονικά, κ.λπ. Εκτελείται με ανάλογο τρόπο, με αυτόν που εκτελείται η εντολή AND.

Παράδειγμα: Έστω ότι έχουμε δύο διακόπτες S1 και S2 (επαφές σε ηρεμία ανοικτές), και θέλουμε να ανάβει η λυχνία L1 είτε όταν και οι δύο διακόπτες είναι ανοιχτοί είτε όταν ένας από τους δύο είναι ανοιχτός.

Σε αυτήν την περίπτωση η ερώτηση για σήμα ″0″ σε μια είσοδο-έξοδο του PLC γίνεται με την εντολή ON (OR NOT). Για να φορτώσουμε την εντολή αυτή στην είσοδο που έχουμε επιλέξει χρησιμοποιούμε την εντολή LDN (Load Not). Έτσι στο παρακάτω παράδειγμα οι εντολές LDN I0.0 και LDN I0.1 φορτώνουν τις εντολές ON

I0.0 και ON I0.1 αντίστοιχα στους διακόπτες που έχουμε επιλέξει. Έτσι λοιπόν η εντολή ON I0.0 ρωτάει <u>είτε</u> στην είσοδο I0.0 έχουμε σήμα ″0″ και η εντολή ON I0.1 ρωτάει <u>είτε</u> στην είσοδο I0.1 έχουμε σήμα ″0″. Αν ισχύουν οι δύο αυτές συνθήκες τότε ενεργοποιείται η έξοδος Q0.0 που έχουμε ορίσει με την εντολή = Q0.0 που αντιστοιχεί στην λυχνία L1. Παρακάτω ακολουθεί το πρόγραμμα και ο πίνακας που μας δείχνει τις πιθανές καταστάσεις και αντίστοιχα πώς συμπεριφέρεται η έξοδος.

```
LDN   I0.0
LDN   I0.1
ON    I0.0
ON    I0.1
=     Q0.0
END
```

ή

```
LDN   I0.0
ON    I0.1
=     Q0.0
END
```

I0.0	I0.1	Q0.0
0	0	1
0	1	1
1	0	1
1	1	0

Όπως παρατηρούμε όταν οι δύο διακόπτες είναι κλειστοί έχουμε κλειστή την λυχνία στην έξοδο. Σε οποιοδήποτε άλλο συνδυασμό των διακοπτών η λυχνία εξόδου θα παραμένει ανοιχτή.

•<u>Η εντολή N (NOT)</u>

Η εντολή NOT κάνει αλλαγή στην κατάσταση των εισόδων ή των εξόδων. Δηλαδή αν το σήμα μιας ή παραπάνω εισόδων-εξόδων είναι ″1″ τότε το κάνει ″0″ και αντίστροφα. Έτσι λοιπόν στο παράδειγμα που ακολουθεί αν βάλουμε ένα NOT πριν

την εντολή εξόδου τότε θα έχουμε τα αντίστροφα αποτελέσματα σύμφωνα με τον πίνακα που ακολουθεί.

```
LD    I0.0
LD    I0.1
A     I0.0
A     I0.1
NOT
=     Q0.0
END
```

ή

```
LD    I0.0
A     I0.1
NOT
=     Q0.0
END
```

I0.0	I0.1	Q0.0
0	0	1
0	1	1
1	0	1
1	1	0

Σημείωση: Από τα παραπάνω παραδείγματα παρατηρούμε ότι για την φόρτωση των εντολών A,O χρησιμοποιούμε την εντολή Load (LD) ενώ για την φόρτωση των εντολών AN,ON χρησιμοποιούμε την εντολή Load not(LDN).

4.3 Ανάπτυξη προγράμματος σε γλώσσα LADDER (LAD).

4.3.1 Γενικά.

Η γλώσσα Ladder (LAD) είναι γλώσσα που χρησιμοποιεί τα ηλεκτρολογικά γραφικά. Το πρόγραμμα σε γλώσσα LADDER μοιάζει με το ηλεκτρολογικό σχέδιο του αυτοματισμού.

Οι ιδιαιτερότητες που έχουμε να αντιμετωπίσουμε στη γλώσσα αυτή είναι:

Χρησιμοποιούνται σύμβολα από την Αμερικάνικη τυποποίηση και όχι από την Ευρωπαϊκή με την οποία είμαστε εξοικειωμένοι.

Το σχέδιο-πρόγραμμα είναι τυποποιημένο, δεν έχουμε δηλαδή την ελευθερία που έχουμε κατά τη σχεδίαση. Για παράδειγμα σε κάθε κλάδο μπορούμε να έχουμε περιορισμένο αριθμό στοιχείων προγράμματος (διακόπτες και επαφές). Επίσης, δεν μπορούμε να κάνουμε οποιασδήποτε μορφής διακλάδωση.

Άρα η δουλειά που έχει να κάνει ο προγραμματιστής στη γλώσσα LADDER είναι να προσαρμόσει το σχέδιο του αυτοματισμού, στα δεδομένα που απαιτεί η γλώσσα.

4.3.2 Δομή προγράμματος στη γλώσσα LADDER.

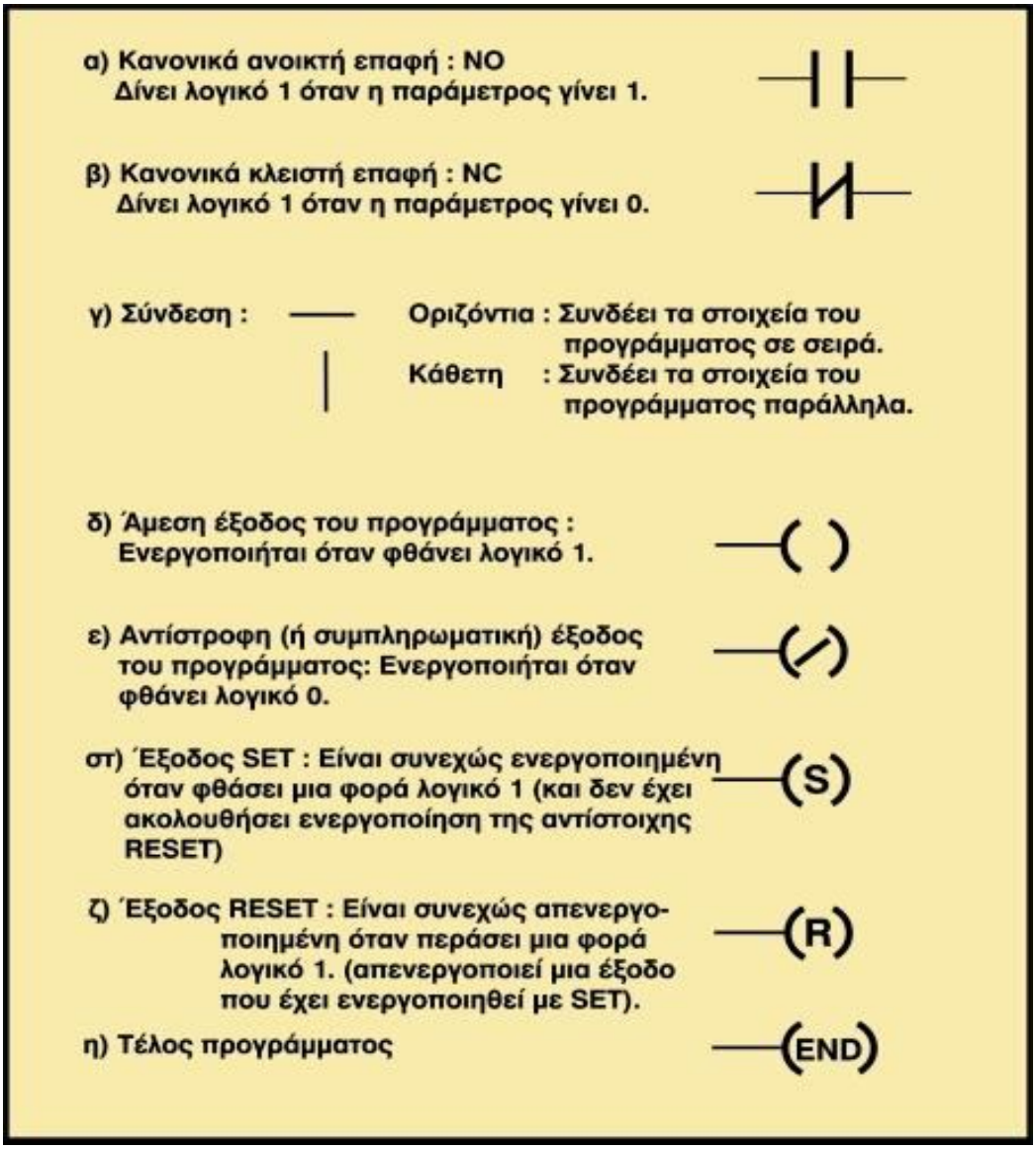
Το πρόγραμμα σε γλώσσα LADDER είναι ένα σχέδιο, ένα διάγραμμα επαφών, δηλαδή σχεδόν το ηλεκτρολογικό σχέδιο του αυτοματισμού.

Το διάγραμμα επαφών της γλώσσας LADDER σχεδιάζεται όχι κατακόρυφα αλλά οριζόντια. Δηλαδή σε ένα πρόγραμμα LADDER έχουμε δύο παράλληλες κατακόρυφες γραμμές (μπάρες), η αριστερή γραμμή παριστάνει τη μπάρα τροφοδοσίας με το υψηλό δυναμικό (+) και η δεξιά γραμμή τη μπάρα τροφοδοσίας με το χαμηλό δυναμικό (-).

Μεταξύ των δύο γραμμών σχεδιάζουμε οριζόντια τους κλάδους του κυκλώματος.

Κάθε κλάδος του διαγράμματος Ladder, που ξεκινά από την αριστερή μπάρα και καταλήγει στη δεξιά μπάρα, αποτελεί μια γραμμή προγράμματος, η οποία αντιστοιχεί στην ομάδα εντολών της γλώσσας λίστα εντολών.

Τα σύμβολα που χρησιμοποιούνται είναι σύμβολα από την Αμερικανική τυποποίηση του ηλεκτρολογικού σχεδίου (ANSI). Τα βασικά σύμβολα δίνονται στο παρακάτω σχήμα. Δίπλα σε κάθε σύμβολο γράφεται το στοιχείο (η παράμετρος) στο οποίο αναφέρεται το σύμβολο.



Σχήμα 4.8: Βασικά στοιχεία προγράμματος σε γλώσσα Ladder για τον προγραμματισμό PLC.

Μερικές σημαντικές παρατηρήσεις για την κατάστρωση ενός προγράμματος είναι:

Οι επαφές (contacts) μπορεί να αντιστοιχούν (να έχουν ονομασία) σε:

- Εισόδους, με πρώτο σύμβολο I.
- Εσωτερικά στοιχεία μνήμης (markers), με πρώτο σύμβολο M.
- Εξόδους, με πρώτο σύμβολο Q

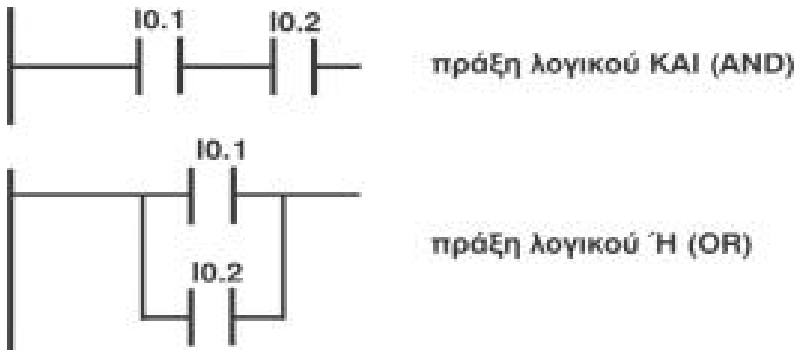
Οι έξοδοι (coils) του προγράμματος μπορεί να αντιστοιχούν (να έχουν ονομασία) σε:

- Εξόδους, με πρώτο σύμβολο Q.
- Εσωτερικά στοιχεία μνήμης, με πρώτο σύμβολο M.

Η έξοδος είναι πάντα το τελευταίο στοιχείο σε μια γραμμή στοιχείων προγράμματος. Στο πρόγραμμα δεν μπορεί δύο διαφορετικές γραμμές στοιχείων προγράμματος να καταλήγουν σε εξόδους (coils) με την ίδια ονομασία.

Σε ένα πρόγραμμα PLC αποφεύγουμε να δίνουμε αρχικές τιμές, αφού λόγω της κυκλικής εκτέλεσης του προγράμματος, σε περίπτωση λανθασμένου σχεδιασμού, αυτές θα επανέρχονται συνεχώς (συνήθως υπάρχουν ειδικές θέσεις μνήμης που διαβάζονται στην εκτέλεση μόνο του πρώτου κύκλου).Τα προγράμματα τελειώνουν πάντα με την εντολή END.

Κατά τη σχεδίαση ενός προγράμματος σε γλώσσα Ladder τα στοιχεία - εντολές συνδεσμολογούνται με βάση το λογικό ΚΑΙ (π.χ. όταν κλείσω το διακόπτη S1 ΚΑΙ το διακόπτη S2) ή το λογικό Η (π.χ. όταν κλείσω το διακόπτη S1 Ή το διακόπτη S2). Στο παρακάτω σχήμα βλέπουμε πώς γράφεται το πρόγραμμα που υλοποιεί τη λογική πράξη ΚΑΙ (AND) και τη λογική πράξη Ή (OR), όπου ο διακόπτης S1 αντιστοιχεί στην επαφή I0.1 και ο S2 στην I0.2 (όταν οι διακόπτες S1 και S2 έχουν μία επαφή NO).



Σχήμα 4.9: Πράξεις λογικού ΚΑΙ , λογικού Ή στη γλώσσα Ladder

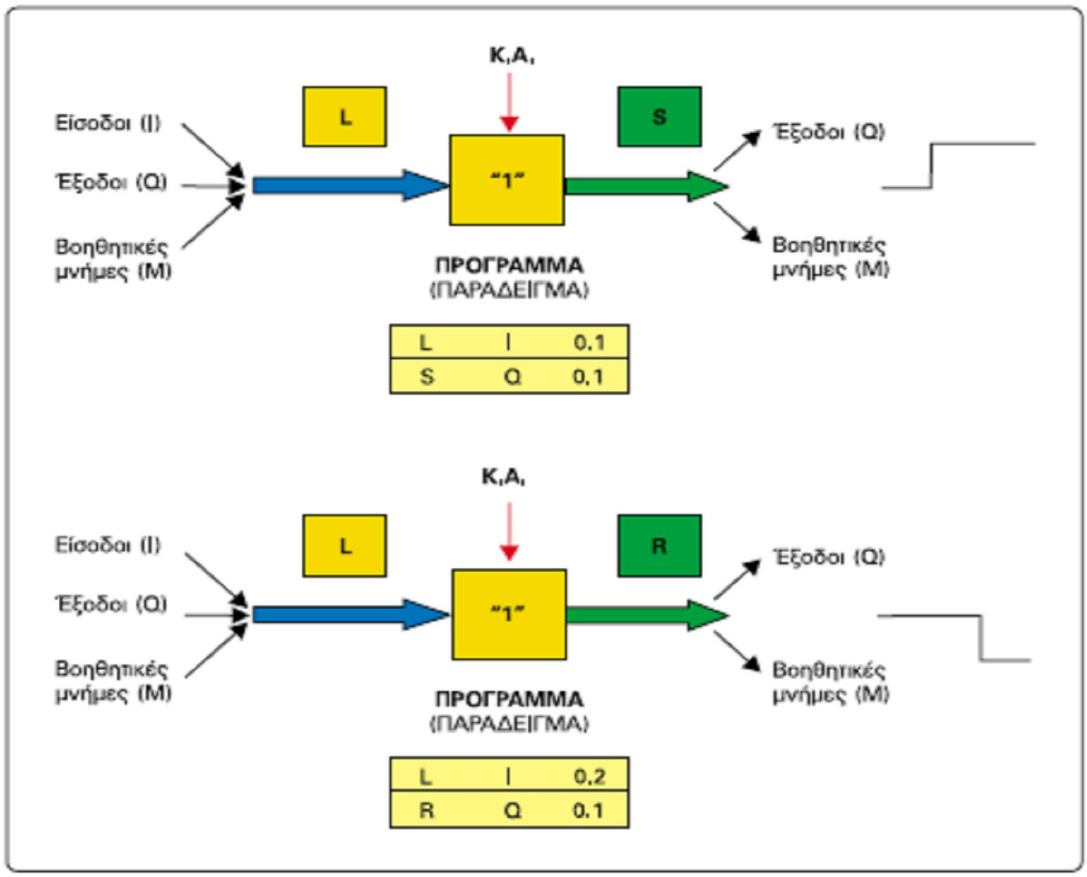
Το αποτέλεσμα μιας λογικής πρότασης είναι να ενεργοποιείται μία έξοδος, που είναι και το τελευταίο στοιχείο της. Στο παρακάτω σχήμα φαίνεται το κύκλωμα που υλοποιεί την πρόταση όταν κλείσει ο διακόπτης S1 ΚΑΙ δεν κλείσει ο διακόπτης S2 ΤΟΤΕ ενεργοποιείται η ενδεικτική λυχνία. Οι διακόπτες S1 και S2 (με μία NO επαφή ο καθένας) αντιστοιχούν στις εισόδους I0.1 και I0.2 αντίστοιχα, ενώ η λυχνία συνδέεται στην έξοδο Q0.0.

Σχήμα 4.10: Ενεργοποίηση εξόδου

Στο σχήμα 4.13 προσέξτε ότι η έκφραση όταν κλείσει ο διακόπτης S1... υλοποιείται στο πρόγραμμα με μία ανοικτή επαφή, ενώ η έκφραση ... ΚΑΙ δεν κλείσει ο διακόπτης S2... υλοποιείται με μία κλειστή επαφή (εφόσον οι S1 και S2 έχουν μία επαφή NO).

4.4 Οι εντολές S (SET) και R (RESET).

Σε όλα τα PLC υπάρχουν οι εντολές S (SET) και R (RESET), οι οποίες αντιστοιχούν στο S-R flip - flop. Αναφέρονται όπως και η εντολή (=) σε εξόδους και βοηθητικές μνήμες. Η εντολή SET θέτει την έξοδο ή τη βοηθητική μνήμη, στην οποία αναφέρεται σε κατάσταση 1, όταν στον Κ.Α. υπάρχει λογική τιμή 1 κατά την εκτέλεση της εντολής. Η έξοδος ή η βοηθητική μνήμη διατηρεί την κατάσταση 1 έστω και αν σε επόμενη εκτέλεση της εντολής SET στον Κ.Α. υπάρχει λογική τιμή 0. Η εντολή RESET θέτει την έξοδο ή τη βοηθητική μνήμη στην οποία αναφέρεται σε κατάσταση 0 όταν στον Κ.Α. υπάρχει λογική τιμή 1 κατά την εκτέλεση της εντολής.



Σχήμα 4.11: Σχηματική παράσταση των εντολών SET και RESET.

Για να γίνει ξεκάθαρη η διαφορά της εντολής SET από την εντολή (=) δίνουμε το παρακάτω παράδειγμα:

Εντολή =

L	I	0.1
=	Q	0.1

Εντολές S, R.

L	I	0.1
S	Q	0.1
L	I	0.2
R	Q	0.1

Στο πρόγραμμα που χρησιμοποιούμε την εντολή (=), η έξοδος Q 0.1 είναι σε κατάσταση 1, όσο η είσοδος I 0.1 είναι σε κατάσταση 1. Μόλις η είσοδος I 0.1 αποκτήσει κατάσταση 0 τότε και η έξοδος Q 0.1 αποκτά κατάσταση 0.

Στο πρόγραμμα που χρησιμοποιούμε την εντολή SET, μόλις η είσοδος I 0.1 αποκτήσει κατάσταση 1, η έξοδος Q 0.1 αποκτά κατάσταση 1.

Αλλά η έξοδος παραμένει σε κατάσταση 1 ακόμη και όταν η είσοδος I 0.1 επανέλθει σε κατάσταση 0. Για να επανέλθει η έξοδος Q 0.1 σε κατάσταση 0 πρέπει να ενεργοποιηθεί η εντολή RESET, δηλαδή πρέπει η είσοδος I 0.2 να αποκτήσει κατάσταση 1.

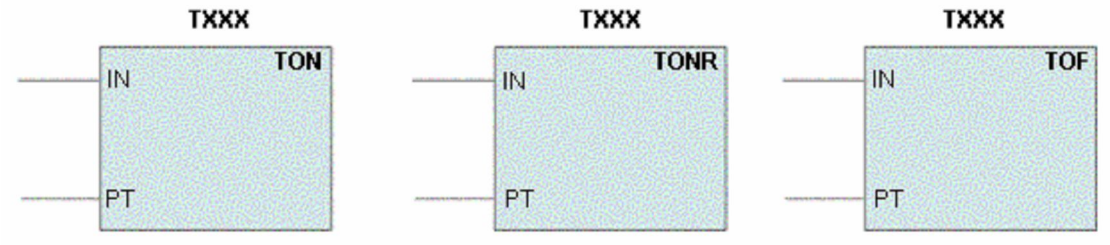
4.5 Ανάπτυξη προγραμμάτων με χρονικές λειτουργίες.

Τα περισσότερα PLC διαθέτουν σημαντικές ευκολίες όσον αφορά στον προγραμματισμό χρονικών λειτουργιών με χρησιμοποίηση των χρονικών λειτουργιών που διαθέτουν. Το κακό είναι ότι στον προγραμματισμό των χρονικών λειτουργιών υπάρχουν μεγάλες διαφορές μεταξύ των PLC της αγοράς, αντίθετα με ότι συμβαίνει με τις εντολές που είδαμε μέχρι τώρα, όπου οι διαφορές είναι ελάχιστες έως και ανύπαρκτες. Εμείς θα προσπαθήσουμε να παρουσιάσουμε τον προγραμματισμό των χρονικών λειτουργιών με τέτοιο τρόπο, ώστε να μπορεί κάποιος πολύ εύκολα να καταλάβει πως χειρίζεται τα χρονικά το κάθε PLC με το οποίο θα ασχοληθεί.

4.5.1 Χρονικά

Τα χρονικά είναι προγραμματιστικές δομές που υλοποιούν και επιτηρούν χρονικά συνδεδεμένες διαδικασίες. Οι εντολές των χρονικών επιτρέπουν στο πρόγραμμα μας να εκτελούν λειτουργίες, όπως χρόνος αναμονής, χρόνος επιτήρησης, δημιουργία παλμοσειρών και μέτρηση χρόνου. Υπάρχουν διαφόρων τύπων χρονικά στα PLC. Στο S7-200 συγκεκριμένα, υπάρχουν χρονικά 3 τύπων. Τα χρονικά αναπαρίστανται στη γλώσσα Ladder με ορθογώνια. Όταν ένα χρονικό δεχθεί το σήμα έναυσης (IN) ξεκινά η μέτρηση του χρόνου και η μετρούμενη τιμή συγκρίνεται συνεχώς με την προκαθορισμένη τιμή (PT). Όσο διάστημα η μετρούμενη τιμή είναι μικρότερη από την προκαθορισμένη τιμή, η έξοδος του χρονικού είναι OFF (λογικό 0). Όταν η μετρούμενη τιμή γίνει μεγαλύτερη από την προκαθορισμένη τιμή, τότε η έξοδος του χρονικού αλλάζει και γίνεται ON (λογικό 1). Οι τρεις τύποι χρονικού του S7-200 είναι: καθυστέρησης έλξης (On Delay TON),καθυστέρησης έλξης με αυτοσυγκράτηση (Retentive On Delay TONR) και καθυστέρησης πτώσης (Off Delay TOF). Οι αναλύσεις χρόνου για τα χρονικά είναι: ανάλυση 1 millisecond, 10

millisecond και 100 millisecond με μέγιστη τιμή μέτρησης 32,767 second, 327,67 second και 3.276,7 second αντίστοιχα. Για μεγαλύτερους χρόνους χρησιμοποιούμε επιπλέον πρόγραμμα.



Σχήμα 4.12:Σχηματική αναπαράσταση χρονικών στη γλώσσα LADDER.

4.5.2 Ανάλυση χρονικών

Τα χρονικά μετρούν χρονικά διαστήματα. Η ανάλυση (βάση χρόνου) καθορίζει το ποσό του χρόνου, που αντιστοιχεί σε ένα χρονικό διάστημα. Ο χρόνος που μετράει το χρονικό είναι λοιπόν το γινόμενο της ανάλυσης επί τον αριθμό των διαστημάτων. Παρακάτω βλέπουμε τον κάθε τύπο χρονικού πιο αναλυτικά.

- Χρονικό καθυστερημένης έλξης (On Delay Timer)

Αυτό το χρονικό μετράει χρόνο όσο η είσοδος είναι "1" (αυτό συμβαίνει για ένα χρονικό διάστημα). Έτσι όταν η είσοδος του χρονικού μηδενιστεί για κάποιο λόγο τότε μηδενίζεται και ο χρόνος του χρονικού. Σε νέα ενεργοποίηση της εισόδου ο χρόνος αρχίζει να μετρά από το μηδέν. Όταν ο χρόνος που έχει μετρήσει το χρονικό είναι μεγαλύτερος ή ίσος από την προκαθορισμένη τιμή το BIT του χρονικού ενεργοποιείται. Στον παρακάτω πίνακα βλέπουμε τις χρονικές τιμές που μπορεί να πάρει ο TON.

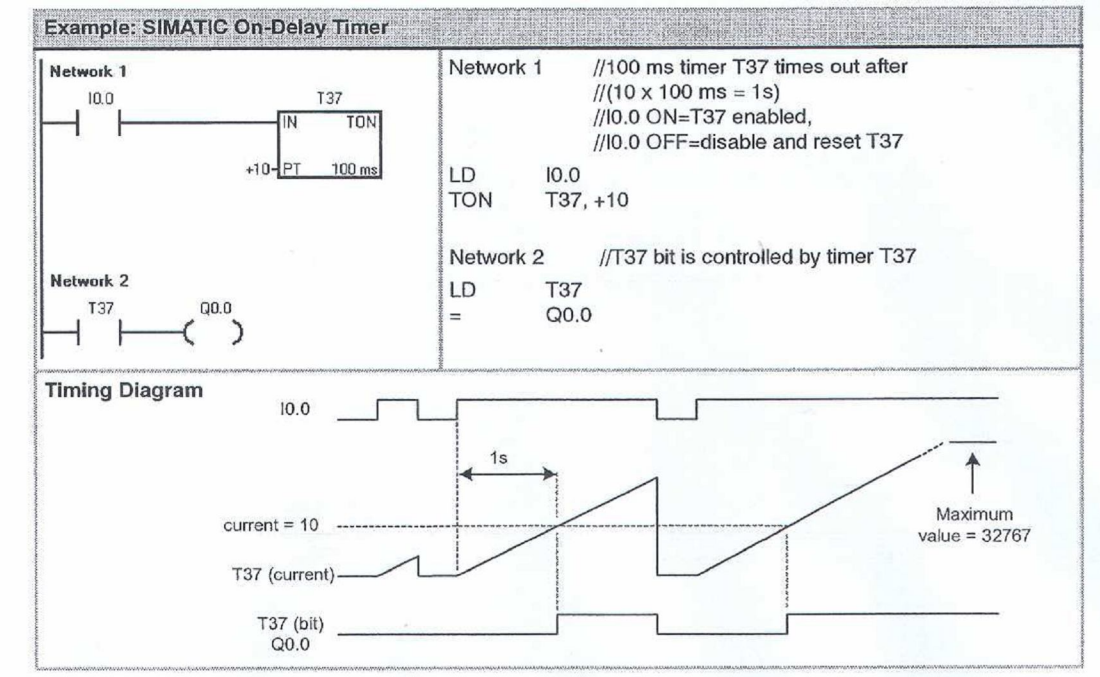
Τύπος χρονικού	Τιμές σε ms	Max τιμή σε sec	Αριθμός Timer
TON	1ms	32,767 sec	T32,T96
	10ms	327,67 sec	T33-T36,T97-T100
	100ms	3276,7 sec	T37-T63,T101-T255

Για παράδειγμα η εντολή TON T32,+50 μας δηλώνει ότι ο timer TON μετράει για 50*1ms=50ms.

Ομοίως η εντολή TON T35,+60 μας δηλώνει ότι ο timer TON μετράει για 60*10ms=600ms=0,6sec.

Ομοίως η εντολή TON T60,+80 μας δηλώνει ότι ο timer TON μετράει για 80*100ms=8000ms=8sec.

Στο παρακάτω σχήμα φαίνεται το χρονικό διάγραμμα του χρονικού TON T37,+10 δηλαδή όπως προείπαμε 10*100ms=1000ms=1sec.



Σχήμα 4.13:Παραδείγματα χρονικού καθυστερημένης έλξης (On Delay Timer)

Παράδειγμα: Με το πάτημα του διακόπτη S1 θέλουμε να ανάβει η λυχνία εξόδου L1 και να παραμένει ανοικτή για 2 sec.

```
LD      I0.0
S       Q0.0, 1
TON     T37, +20
LD      T37
R       Q0.0, 1
END
```

Επεξήγηση: Στο πρόγραμμα αυτό πατώντας τον διακόπτη I0.0 κάνουμε SET στην έξοδο Q0.0 με την εντολή S Q0.0,1 δηλαδή ανάβουμε την λυχνία. Ταυτόχρονα αρχίζει να μετράει ο timer με την εντολή TON T37,+20 για 20*100ms=2sec και όταν φτάσει την τιμή αυτή γίνεται reset στην έξοδο με την εντολή R Q0.0,1.

- Χρονικό καθυστερημένης έλξης με αυτοσυγκράτηση (Retentive On Delay Timer)

Το χρονικό αυτό ενεργοποιείται εφόσον συμβεί μεταβολή από "0" σε "1". Για όση ώρα έχουμε κατάσταση "1" το χρονικό θα μετράει ωσότου φτάσει την τελική του τιμή που εμείς του έχουμε δώσει μέσα στο πρόγραμμα μας. Αν υπάρχει κατάσταση "1" περισσότερη ώρα από τον χρόνο που έχουμε ορίσει στο χρονικό να μετρήσει τότε αυτό αφού πάρει την τελική του τιμή θα παραμείνει σε αυτήν μέχρι την αλλαγή κατάστασης. Αν έχουμε αλλαγή κατάστασης από "1" σε "0" πριν το χρονικό φτάσει στην τελική του τιμή τότε κρατάει την τιμή που έχει εκείνη την χρονική στιγμή της αλλαγής της κατάστασης μέχρι την επόμενη αλλαγή δηλαδή από "0" σε "1" όπου και συνεχίζει να μετράει από την τιμή αυτή ως την τελική του τιμή. Από τα παραπάνω βγάζουμε το συμπέρασμα ότι το χρονικό αυτό δεν μηδενίζεται όσες αλλαγές καταστάσεων και αν γίνουν. Για να καταφέρουμε τον μηδενισμό του χρησιμοποιούμε την εντολή Reset και μόνο αυτήν. Στον παρακάτω πίνακα βλέπουμε τις χρονικές τιμές που μπορεί να πάρει ο TONR.

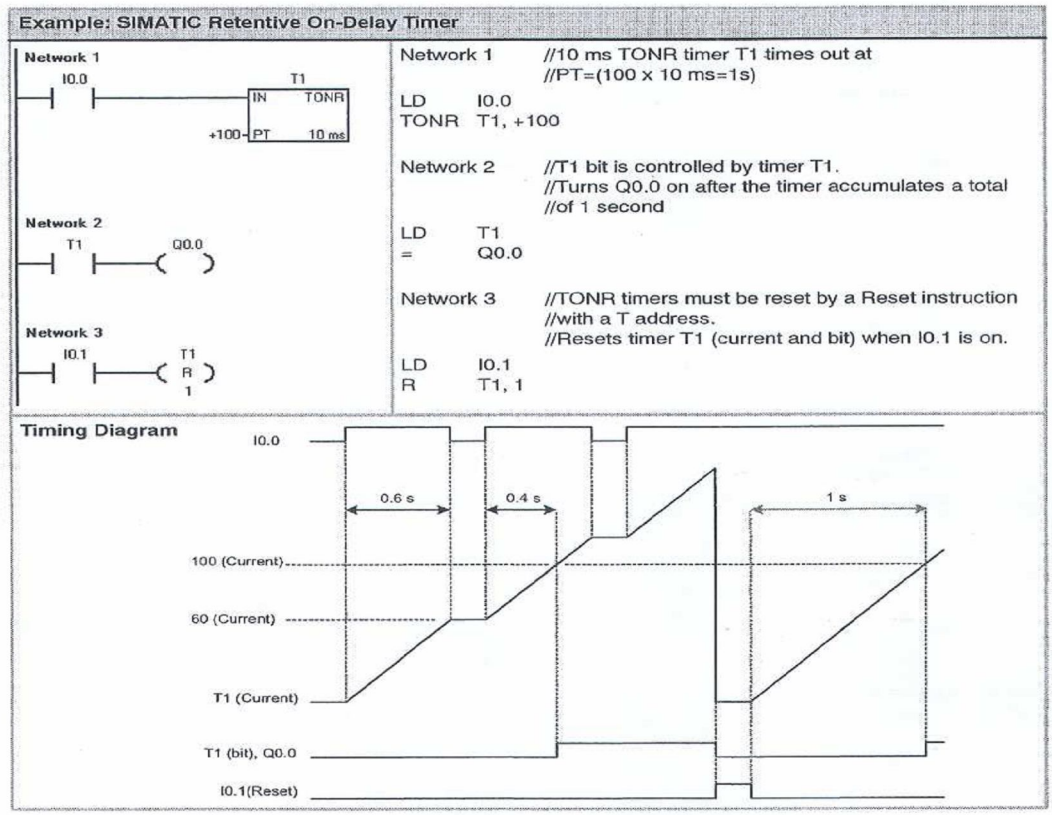
Τύπος χρονικού	Τιμές σε ms	Max τιμή σε sec	Αριθμός Timer
TONR	1ms	32,767 sec	T0,T64
	10ms	327,67 sec	T1-T4,T65-T68
	100ms	3276,7 sec	T5-T31,T69-T95

Για παράδειγμα η εντολή TONR T0,+10 μας δηλώνει ότι ο timer TONR μετράει για 10*1ms=10ms.

Ομοίως η εντολή TONR T3,+40 μας δηλώνει ότι ο timer TONR μετράει για 40*10ms=400ms=0,4sec.

Ομοίως η εντολή TONR T69,+70 μας δηλώνει ότι ο timer TONR μετράει για 70*100ms=7000ms=7sec.

Στο παρακάτω σχήμα φαίνεται το χρονικό διάγραμμα του χρονικού TONR T1,+100 δηλαδή όπως προείπαμε 100*10ms=1000ms=1sec.



Σχήμα 4.14:Παράδειγμα χρονικού καθυστερημένης έλξης με αυτοσυγκράτηση (Retentive On Delay Timer)

Παράδειγμα: Με το πάτημα του διακόπτη S1 θέλουμε να ανάβει η λυχνία εξόδου L1 και να παραμένει ανοικτή. Θέλουμε η λυχνία να κλείσει όταν ο συνολικός χρόνος που ο διακόπτης είναι κλειστός (έχει σήμα "1") να είναι 5 sec ανεξάρτητα αν ανοίξουμε τον διακόπτη για κάποια χρονικά διαστήματα πριν την συμπλήρωση του χρόνου αυτού. Επίσης να δοθεί δυνατότητα επανάληψης του προγράμματος για όσες φορές επιθυμούμε.

```
LD      I0.0
S       Q0.0, 1
TONR    T5, +50
LD      T5
R       Q0.0, 1
LD      I0.1
R       T5, 1
END
```

Επεξήγηση: Στο πρόγραμμα αυτό πατώντας τον διακόπτη I0.0 κάνουμε SET στην έξοδο Q0.0 με την εντολή S Q0.0,1 δηλαδή ανάβουμε την λυχνία. Όταν ανοίξουμε τον διακόπτη αρχίζει να μετράει ο timer με την εντολή TONR T5,+50 για 50*100ms=5 sec. Αν πριν την συμπλήρωση του χρόνου ο διακόπτης ανοίξει τότε το χρονικό κρατάει την τιμή που έχει εκείνη την χρονική στιγμή μέχρι να ξανακλείσει όπου και συνεχίζει την μέτρηση από αυτήν την τιμή μέχρι τα 5 sec. Για να επαναλάβουμε την μέτρηση πρέπει να κάνουμε reset στο χρονικό με κάποιον διακόπτη που έχουμε ορίσει. Το reset αυτό γίνεται με την εντολή R T5,1.

- Χρονικό καθυστερημένης πτώσης (Off Delay Timer)

Αυτό το χρονικό χρησιμοποιείται για να καθυστερήσει την απενεργοποίηση μιας εξόδου για ένα συγκεκριμένο χρονικό διάστημα από την απενεργοποίηση της εισόδου. Μόλις λοιπόν η είσοδος απενεργοποιηθεί τότε το χρονικό ξεκινάει την μέτρηση του για το χρονικό διάστημα που του έχουμε προσδώσει. Αν η είσοδος παραμένει απενεργοποιημένη για περισσότερο χρονικό διάστημα από την διάρκεια του χρονικού τότε αυτό θα σταματήσει την μέτρηση και θα παραμείνει σταθερό στην

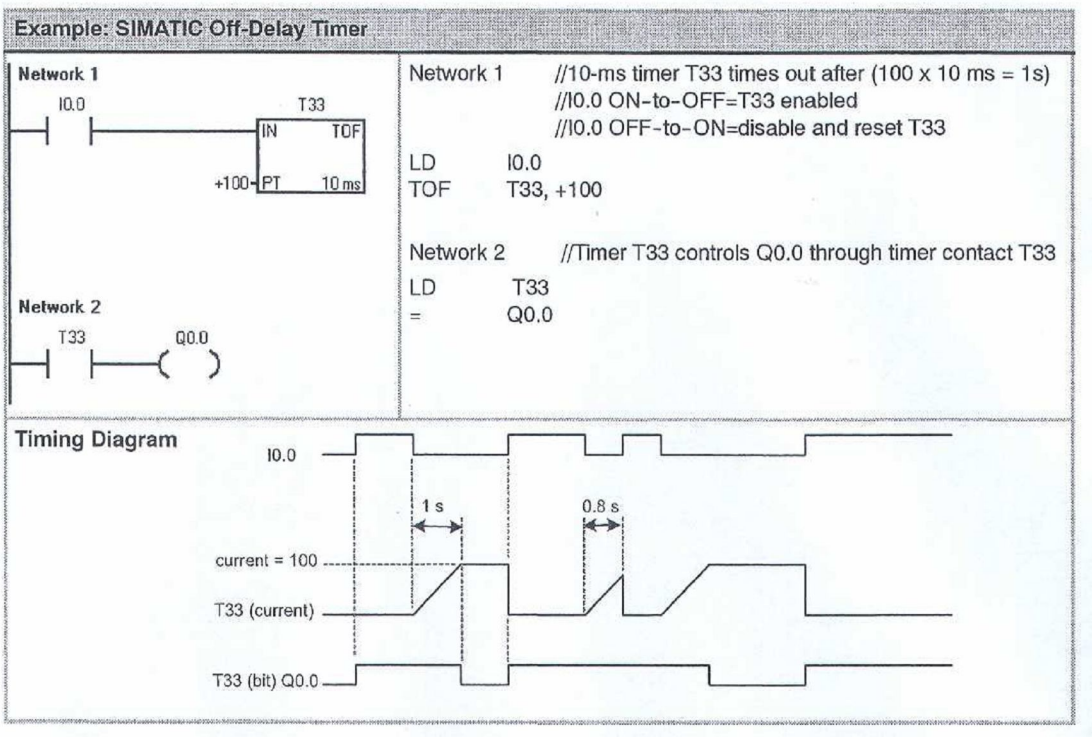
τελική του τιμή ώσπου να ενεργοποιήσουμε την είσοδο και πάλι όπου το χρονικό μηδενίζεται. Με την επόμενη ενεργοποίηση της εισόδου ξεκινάει η ίδια διαδικασία. Στον παρακάτω πίνακα βλέπουμε τις χρονικές τιμές που μπορεί να πάρει ο TOF.

Τύπος χρονικού	Τιμές σε ms	Max τιμή σε sec	Αριθμός Timer
TOF	1ms	32,767 sec	T32,T96
	10ms	327,67 sec	T33-T36,T97-T100
	100ms	3276,7 sec	T37-T63,T101-T255

Για παράδειγμα η εντολή TOF T32,+20 μας δηλώνει ότι ο timer TON μετράει για 20*1ms=20ms.

Ομοίως η εντολή TOF T35,+30 μας δηλώνει ότι ο timer TON μετράει για 30*10ms=300ms=0,3sec.

Ομοίως η εντολή TOF T60,+50 μας δηλώνει ότι ο timer TON μετράει για 50*100ms=5000ms=5sec. Στο παρακάτω σχήμα φαίνεται το χρονικό διάγραμμα του χρονικού TOF T33,+100 δηλαδή όπως προείπαμε 100*10ms=1000ms=1sec.



Σχήμα 4.15:Παράδειγμα χρονικού καθυστερημένης πτώσης (Off Delay Timer)

Παράδειγμα: Με το πάτημα του διακόπτη S1 θέλουμε να ανάβει η λυχνία εξόδου L1 και να παραμένει ανοικτή. Όταν κλείσουμε τον διακόπτη θέλουμε η λυχνία να κλείσει μετά από 5 sec.

```
LD      I0.0
A       I0.0
S       Q0.0, 1
A       I0.0
TOF     T33, +500
LDN     T33
R       Q0.0, 1
END
```

Επεξήγηση: Στο πρόγραμμα αυτό πατώντας τον διακόπτη I0.0 κάνουμε SET στην έξοδο Q0.0 με την εντολή S Q0.0,1 δηλαδή ανάβουμε την λυχνία. Όταν ανοίξουμε τον διακόπτη αρχίζει να μετράει ο timer με την εντολή TOF T33,+500 για 500*10ms=5 sec και όταν φτάσει την τιμή αυτή γίνεται reset στην έξοδο με την εντολή R Q0.0,1

Παράδειγμα: Να δημιουργήσετε ένα πρόγραμμα στο οποίο η έξοδος Q0.0 να δουλεύει σαν clock με διάρκεια παλμού 2 sec.

```
LD      I1.0
A       I1.0
AN      T38
TON     T37, +20
LD      T37
=       Q0.0
A       T37
TON     T38, +20
END
```

Επεξήγηση: Στο πρόγραμμα αυτό με το πάτημα του διακόπτη I0.0 έχουμε σβήσιμο της εξόδου για 2 sec με την εντολή TON T37, +20 και στη συνέχεια άναμμα της εξόδου για 2 sec με την εντολή TON T38, +20.

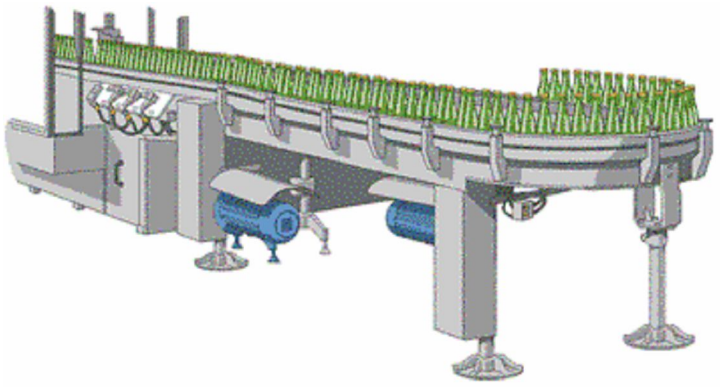
4.6 Απαριθμητές (Counters)

Οι απαριθμητές μας δίνουν την δυνατότητα να εκτελούμε λειτουργίες απαρίθμησης μέσα στην CPU. Οι απαριθμητές μετρούν προς τα πάνω ή προς τα κάτω. Για να γίνει μια μέτρηση η CPU πρέπει να αντιληφθεί αλλαγή στην κατάσταση του σήματος σε μια είσοδο.

Οι απαριθμητές στα PLC εκτελούν λειτουργία αντίστοιχη με αυτή των μηχανικών απαριθμητών. Η μετρούμενη τιμή συγκρίνεται με μία προκαθορισμένη τιμή. Συνήθως οι απαριθμητές λειτουργούν με δύο διαφορετικές λογικές:

- Μετρούν μέχρι την προκαθορισμένη τιμή και προκαλούν ένα γεγονός
- Προκαλούν ένα γεγονός μέχρι η μέτρηση να φτάσει την προκαθορισμένη τιμή.

Μία μηχανή (γραμμή παραγωγής) εμφιάλωσης για παράδειγμα μπορεί να χρησιμοποιεί έναν απαριθμητή για να μετρά 6άδες μπουκαλιών που μετά θα μπουν μαζί σε μία συσκευασία. Σε αυτήν την περίπτωση ο απαριθμητής λειτουργεί με την πρώτη από τις δύο λογικές που περιγράφηκαν παραπάνω.



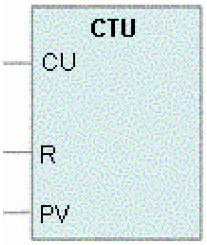
Σχήμα 4.16:Παράδειγμα χρήσης απαριθμητή σε γραμμή παραγωγής.

4.6.1 Κατηγορίες απαριθμητών

Υπάρχουν τρείς κατηγορίες απαριθμητών:

• Counter up (CTU).

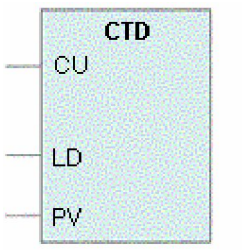
Σε αυτόν τον μετρητή αυξάνεται η τιμή του κατά ένα κάθε φορά που η είσοδος CU μεταβαίνει από το μηδέν στο ένα. Όταν η τιμή του μετρητή γίνει ίση ή μεγαλύτερη από την προκαθορισμένη τιμή PV το bit του απαριθμητή γίνεται ένα. Όταν ο μετρητής φθάσει την επιθυμητή τιμή σταματάει να απαριθμεί. Ο απαριθμητής απενεργοποιείται-μηδενίζεται, όταν η είσοδος του reset μεταβεί από το μηδέν στο ένα. Παρακάτω απεικονίζεται ο CTU απαριθμητής στην γλώσσα Ladder.



Σχήμα 4.17: Ο CTU απαριθμητής.

• Counter down (CTD).

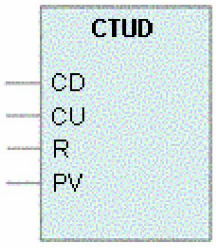
Σε αυτόν τον μετρητή μειώνεται η τιμή του κατά ένα κάθε φορά που η είσοδος CU μεταβαίνει από το μηδέν στο ένα. Όταν η τιμή του μετρητή γίνει ίση με το μηδέν το bit του απαριθμητή γίνεται ένα. Όταν ο μετρητής φθάσει την τιμή μηδέν σταματάει να απαριθμεί. Ο απαριθμητής απενεργοποιείται και η τρέχουσα τιμή του τίθεται ίση με την προκαθορισμένη τιμή PV όταν η είσοδος του reset μεταβεί από το μηδέν στο ένα. Παρακάτω απεικονίζεται ο CTD απαριθμητής στην γλώσσα Ladder.



Σχήμα 4.18: Ο CTD απαριθμητής.

• Counter up/down (CTUD).

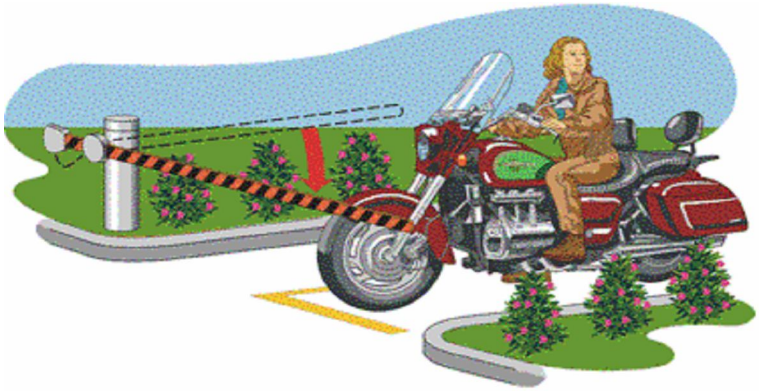
Αυτός ο μετρητής απαριθμεί προς τα πάνω, όταν η είσοδος CU μεταβαίνει από το μηδέν στο ένα και απαριθμεί προς τα κάτω, όταν η είσοδος του CD μεταβαίνει από το μηδέν στο ένα. Όταν η τιμή του μετρητή είναι μεγαλύτερη ή ίση από την προκαθορισμένη τιμή, ενεργοποιείται το bit του απαριθμητή. Ο απαριθμητής παύει να απαριθμεί όταν φτάσει την προκαθορισμένη τιμή PV.Ο απαριθμητής απενεργοποιείται όταν ενεργοποιείται η είσοδος reset. Παρακάτω απεικονίζεται ο CTUD απαριθμητής στην γλώσσα Ladder.



Σχήμα 4.19: Ο CTUD απαριθμητής.

4.6.2 Παράδειγμα χρήσης απαριθμητών στην πράξη.

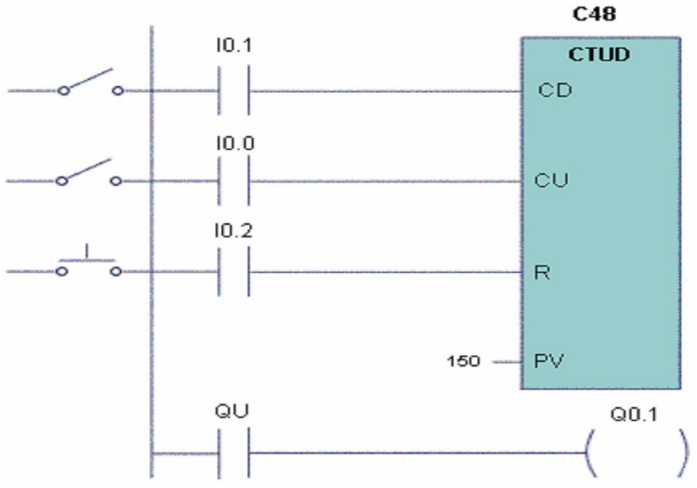
Ένας απαριθμητής μπορεί να χρησιμοποιηθεί για να ελέγχεται ο αριθμός των οχημάτων σε ένα χώρο στάθμευσης. Όταν τα οχήματα μπαίνουν στο χώρο από την είσοδο, ο απαριθμητής μετρά προς τα πάνω. Όταν βγαίνουν από την έξοδο ο απαριθμητής μετρά προς τα κάτω. Όταν η μετρούμενη τιμή φτάσει την προκαθορισμένη τιμή που αντιστοιχεί στο μέγιστο αριθμό οχημάτων που χωρά ο χώρος στάθμευσης τότε ενεργοποιείται μια ενδεικτική λυχνία στην είσοδο του parking που μας ειδοποιεί ότι ο χώρος στάθμευσης είναι γεμάτος.



Σχήμα 4.20: Παράδειγμα χρήσης απαριθμητών σε χώρο στάθμευσης οχημάτων.

Για να υλοποιηθεί η εφαρμογή χρησιμοποιείται ένας απαριθμητής αύξησης/μείωσης, συγκεκριμένα ο up/down counter 48. Ένας διακόπτης που έχει προσαρμοσθεί στην είσοδο του χώρου στάθμευσης συνδέεται στην είσοδο I0.0. Ένας άλλος διακόπτης, προσαρμοσμένος στην έξοδο, συνδέεται στην είσοδο I0.1. Ο χώρος στάθμευσης έχει χώρο για 150 οχήματα. Αυτή είναι και η τιμή που δίνεται στον απαριθμητή ως προκαθορισμένη τιμή μέτρησης PV.

Στην έξοδο Q0.1 συνδέεται η ενδεικτική λυχνία “πληρότητας” του χώρου στάθμευσης. Αν η μέτρηση στον απαριθμητή φθάσει το 150 η έξοδος Q0.1 ενεργοποιείται και ανάβει η ενδεικτική λυχνία. Αν ένα αυτοκίνητο βγει από το χώρο στάθμευσης, η μέτρηση γίνεται 149 και η ενδεικτική λυχνία σβήνει. Στο παρακάτω σχήμα φαίνεται η υλοποίηση του εν λόγω προγράμματος σε γλώσσα Ladder.



Σχήμα 4.21:Υλοποίηση προγράμματος σε γλώσσα Ladder.

Παράδειγμα: Σε μια διαδικασία ελέγχου θέλουμε να γίνονται τα εξής: Όταν περάσουν 5 αυτοκίνητα από την είσοδο του παρκινγκ να ανάβει μια λυχνία η οποία θα φωτοβολεί για 3 sec. Η διαδικασία να επαναλαμβάνεται συνέχεια.

```
LD      I0.0
LD      I0.1
CTU     C34, +5
LD      C34
=       Q0.0
TON     T37, +30
LD      T37
R       Q0.0, 1
END
```

Επεξήγηση: Στο πρόγραμμα αυτό με τον διακόπτη I0.0 μεταβάλλουμε τον απαριθμητή ενώ με τον I0.1 μηδενίζουμε τον απαριθμητή. Όταν θα περάσουν 5 οχήματα ανάβει η λυχνία Q0.0.Ταυτόχρονα ενεργοποιείται ο timer για 3 δευτερόλεπτα. Με την εντολή R Q0.0, 1 κάνουμε reset στην έξοδο.

Παράδειγμα: Να φτιαχτεί ένα πρόγραμμα που όταν ένας διακόπτης πατηθεί δύο φορές να ενεργοποιείται ένα clock με κατάσταση "1" που διαρκεί 3 sec και κατάσταση "0" που διαρκεί 1 sec.

```
LD      I0.0
LD      I0.1
CTU     C1, +2
LD      C1
AN      T37
TON     T38, +10
LD      T38
=       Q0.0
A       T38
TON     T37, +30
END
```

Επεξήγηση: Στο πρόγραμμα αυτό όταν οι διακόπτες I0.0 και I0.1 πατηθούν συνολικά δύο φορές έχουμε σβήσιμο του LED για 1 sec με την εντολή TON T38, +10 και στη συνέχεια άναμμα του LED για 3 sec με την εντολή TON T37, +30.

4.7 Συγκρίσεις

Οι λειτουργίες σύγκρισης συγκρίνουν δύο ψηφιακές τιμές αριθμητικές ή λογικές. Το αποτέλεσμα της σύγκρισης επηρεάζει το RLO και με αυτό μπορούμε να πάρουμε κάποιες αποφάσεις όπως για παράδειγμα όταν η στάθμη μιας δεξαμενής φτάσει το 90% του μέγιστου επιτρεπτού να ανάβει μια προειδοποιητική λυχνία. Οι συγκρίσεις γίνονται στην CPU και εκτελούνται ανεξάρτητα από την ικανοποίηση ή μη ορισμένων συνθηκών ή λογικών μανδαλώσεων. Ανάλογα με το είδος των αριθμών που πρόκειται να συγκρίνουμε (Integer,Real,Byte) υπάρχει και η αντίστοιχη εντολή.

• ΣΥΓΚΡΙΣΗ ΑΚΕΡΑΙΩΝ

Οι εντολές σύγκρισης ακεραίων χρησιμοποιούνται για την σύγκριση δύο ακεραίων τιμών έστω INT 1 και INT 2 και περιλαμβάνουν τις παρακάτω πράξεις:

•Εντολή ισότητας: LDW= INT 1,INT 2
•Εντολή διάφορου: LDW<> INT 1,INT 2
•Εντολή μικρότερου από: LDW< INT 1,INT 2
•Εντολή μεγαλύτερου από: LDW> INT 1,INT 2
•Εντολή μικρότερου ή ίσου: LDW<= INT 1,INT 2
•Εντολή μεγαλύτερου ή ίσου: LDW>= INT 1,INT 2

Παράδειγμα: Να γίνει πρόγραμμα που να συγκρίνει δύο ακεραίους.

```
LDW>=    +12, +5
=        Q0.0
END
```

Με την εκτέλεση του προγράμματος θα γίνει η παραπάνω σύγκριση στην CPU και αν αληθεύει θα ενεργοποιηθεί η έξοδος Q0.0.

• ΣΥΓΚΡΙΣΗ ΠΡΑΓΜΑΤΙΚΩΝ

Οι εντολές σύγκρισης πραγματικών χρησιμοποιούνται για την σύγκριση δύο πραγματικών τιμών έστω INT 1 και INT 2 και περιλαμβάνουν τις παρακάτω πράξεις:

•Εντολή ισότητας: LDR= INT 1,INT 2
•Εντολή διάφορου: LDR<> INT 1,INT 2
•Εντολή μικρότερου από: LDR< INT 1,INT 2
•Εντολή μεγαλύτερου από: LDR> INT 1,INT 2
•Εντολή μικρότερου ή ίσου: LDR<= INT 1,INT 2
•Εντολή μεγαλύτερου ή ίσου: LDR>= INT 1,INT 2

Παράδειγμα: Να γίνει πρόγραμμα που να συγκρίνει δύο πραγματικούς.

```
LDR<>       17.8, 12.7
=           Q0.0
END
```

Με την εκτέλεση του προγράμματος θα γίνει η παραπάνω σύγκριση στην CPU και αν αληθεύει θα ενεργοποιηθεί η έξοδος Q0.0.

• ΣΥΓΚΡΙΣΗ BYTE

Οι εντολές σύγκρισης byte χρησιμοποιούνται για την σύγκριση δύο τιμών έστω INT1 και INT 2 και περιλαμβάνουν τις παρακάτω πράξεις:

•Εντολή ισότητας:	LDB=	INT 1,INT 2
•Εντολή διάφορου:	LDB<>	INT 1,INT 2
•Εντολή μικρότερου από:	LDB<	INT 1,INT 2
•Εντολή μεγαλύτερου από:	LDB>	INT 1,INT 2
•Εντολή μικρότερου ή ίσου:	LDB<=	INT 1,INT 2
•Εντολή μεγαλύτερου ή ίσου:	LDB>=	INT 1,INT 2

4.8 Αριθμητικές πράξεις.

Οι αριθμητικές πράξεις εφαρμόζονται πάνω σε δύο ψηφιακές τιμές που βρίσκονται η μια στον Accumulator 1 και η άλλη στον Accumulator 2. Οι πράξεις εκτελούνται ανεξάρτητα από την τιμή του RLO δηλαδή άσχετα από την ικανοποίηση ή μη ορισμένων συνθηκών ή λογικών μανδαλώσεων.

•ΠΡΟΣΘΕΣΗ ΚΑΙ ΑΦΑΙΡΕΣΗ ΑΚΕΡΑΙΩΝ.

Στην περίπτωση πρόσθεσης ή αφαίρεσης ακεραίων αριθμών οι πράξεις αυτές θα γίνουν μόνο μεταξύ των δεξιών τμημάτων των δύο Accumulators επειδή όπως είπαμε και παραπάνω οι πράξεις αυτές γίνονται μόνο μεταξύ των περιεχομένων δύο Accumulators. Για να κάνουμε λοιπόν πρόσθεση ή αφαίρεση ακεραίων αριθμών πρέπει να φορτώσουμε κάθε αριθμό σε κάποιον Accumulator.

Η σύνταξη μιας εντολής πρόσθεσης δύο ακεραίων αριθμών έχει ως εξής:

+I IN1, OUT

Η σύνταξη μιας εντολής αφαίρεσης δύο ακεραίων αριθμών έχει ως εξής:

−I IN1, OUT

Τα IN1 και OUT συμβολίζουν δύο ακεραίους αριθμούς και τα +I, -I την πράξη της πρόσθεσης και της αφαίρεσης αντίστοιχα. Μετά την εκτέλεση κάποιας από τις δύο παραπάνω πράξεις ο IN1 παραμένει αμετάβλητος ενώ στον OUT θα αποθηκευτεί το αποτέλεσμα της αριθμητικής πράξης που εκτελέστηκε. Δηλαδή ισχύουν οι δύο παρακάτω ισότητες:

IN1+OUT = OUT
OUT−IN1 = OUT

Παράδειγμα: Να γίνει ένα πρόγραμμα το οποίο θα προσθέτει δύο ακεραίους αριθμούς και ένα άλλο το οποίο θα τους αφαιρεί.

```
LD      I0.0
A       I0.0
MOVW    +4, AC1
MOVW    +7, AC2
+I      AC2, AC1
END
```

```
LD      I0.0
A       I0.0
MOVW    +5, AC1
MOVW    +8, AC2
−I      AC2, AC1
END
```

Στα δύο παραπάνω προγράμματα χρησιμοποιούμε τον διακόπτη I0.0 για να αποθηκεύσουμε τους δύο ακεραίους αριθμούς στους δύο Accumulators και στην συνέχεια να εκτελεστούν οι αριθμητικές πράξεις. Όπως είπαμε και παραπάνω το περιεχόμενο του AC2 δεν θα αλλάξει μετά από τις πράξεις όμως το περιεχόμενο του AC1 θα αλλάξει και θα περιλαμβάνει το αποτέλεσμα της πράξης που γίνεται δηλαδή το άθροισμα των AC1 και AC2 για την πρόσθεση και το υπόλοιπο AC1 και AC2 για την αφαίρεση αντίστοιχα.

•ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ ΚΑΙ ΔΙΑΙΡΕΣΗ ΑΚΕΡΑΙΩΝ.

Για να κάνουμε λοιπόν πολλαπλασιασμό και διαίρεση κατά όμοιο τρόπο με την πρόσθεση και την αφαίρεση ακεραίων αριθμών πρέπει να φορτώσουμε κάθε αριθμό σε κάποιον Accumulator.

Η σύνταξη μιας εντολής πολλαπλασιασμού δύο ακεραίων αριθμών έχει ως εξής:

***I IN1, OUT**

Η σύνταξη μιας εντολής διαίρεσης δύο ακεραίων αριθμών έχει ως εξής:

/I IN1, OUT

Τα IN1 και OUT συμβολίζουν δύο ακεραίους αριθμούς και τα *I, /I την πράξη του πολλαπλασιασμού και της διαίρεσης αντίστοιχα. Μετά την εκτέλεση κάποιας από τις δύο παραπάνω πράξεις ο IN1 παραμένει αμετάβλητος ενώ στον OUT θα αποθηκευτεί το αποτέλεσμα της αριθμητικής πράξης που εκτελέστηκε. Δηλαδή ισχύουν οι δύο παρακάτω ισότητες:

IN1*OUT = OUT
OUT/IN1 = OUT

Παράδειγμα: Να γίνει ένα πρόγραμμα το οποίο θα πολλαπλασιάζει δύο ακεραίους αριθμούς και ένα άλλο το οποίο θα τους διαιρεί.

```
LD      I0.0
A       I0.0
MOVW    +7, AC1
*I      2, AC1
END
```

```
LD      I0.0
A       I0.0
MOVW    +9, AC1
/I      AC1, +3
END
```

Στα δύο παραπάνω προγράμματα χρησιμοποιούμε τον διακόπτη I0.0 για να αποθηκεύσουμε τον ακέραιο αριθμό στον AC1 και στην συνέχεια να εκτελεστούν οι αριθμητικές πράξεις. Μπορούμε να χρησιμοποιήσουμε και άλλον ένα AC αποθηκεύοντας σε αυτόν τον αριθμό που θέλουμε να πολλαπλασιάσουμε ή να διαιρέσουμε όπως στην πρόσθεση αφαίρεση.

•ΠΡΟΣΘΕΣΗ ΚΑΙ ΑΦΑΙΡΕΣΗ ΠΡΑΓΜΑΤΙΚΩΝ

Στην περίπτωση πρόσθεσης ή αφαίρεσης πραγματικών αριθμών οι πράξεις αυτές θα γίνουν μόνο μεταξύ των δεξιών τμημάτων των δύο Accumulators επειδή όπως είπαμε και παραπάνω οι πράξεις αυτές γίνονται μόνο μεταξύ των περιεχομένων δύο Accumulators. Για να κάνουμε λοιπόν πρόσθεση ή αφαίρεση πραγματικών αριθμών πρέπει να φορτώσουμε κάθε αριθμό σε κάποιον Accumulator.

Η σύνταξη μιας εντολής πρόσθεσης δύο πραγματικών αριθμών έχει ως εξής:

+R IN1, OUT

Η σύνταξη μιας εντολής αφαίρεσης δύο πραγματικών αριθμών έχει ως εξής:

−R IN1, OUT

Τα IN1 και OUT συμβολίζουν δύο πραγματικών αριθμούς και τα +R, -R την πράξη της πρόσθεσης και της αφαίρεσης αντίστοιχα. Μετά την εκτέλεση κάποιας από τις δύο παραπάνω πράξεις ο IN1 παραμένει αμετάβλητος ενώ στον OUT θα αποθηκευτεί το αποτέλεσμα της αριθμητικής πράξης που εκτελέστηκε. Δηλαδή ισχύουν οι δύο παρακάτω ισότητες:

IN1+OUT = OUT
OUT−IN1 = OUT

Παράδειγμα: Να γίνει ένα πρόγραμμα το οποίο θα προσθέτει δύο πραγματικούς αριθμούς και ένα άλλο το οποίο θα τους αφαιρεί.

```
LD      I0.0
A       I0.0
MOVR    4.7, AC1
MOVR    7.2, AC2
+R      AC2, AC1
END
```

```
LD      I0.0
A       I0.0
MOVR    3.5, AC1
MOVR    5.8, AC2
−R      AC2, AC1
END
```

Στα δύο παραπάνω προγράμματα χρησιμοποιούμε τον διακόπτη I0.0 για να αποθηκεύσουμε τους δύο πραγματικούς αριθμούς στους δύο Accumulators και στην συνέχεια να εκτελεστούν οι αριθμητικές πράξεις. Όπως είπαμε και παραπάνω το περιεχόμενο του AC2 δεν θα αλλάξει μετά από τις πράξεις όμως το περιεχόμενο του AC1 θα αλλάξει και θα περιλαμβάνει το αποτέλεσμα της πράξης που γίνεται δηλαδή το άθροισμα των AC1 και AC2 για την πρόσθεση και το υπόλοιπο AC1 και AC2 για την αφαίρεση αντίστοιχα.

•ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ ΚΑΙ ΔΙΑΙΡΕΣΗ ΠΡΑΓΜΑΤΙΚΩΝ.

Για να κάνουμε λοιπόν πολλαπλασιασμό και διαίρεση κατά όμοιο τρόπο με την πρόσθεση και την αφαίρεση πραγματικών αριθμών πρέπει να φορτώσουμε κάθε αριθμό σε κάποιον Accumulator.

Η σύνταξη μιας εντολής πολλαπλασιασμού δύο πραγματικών αριθμών έχει ως εξής:

***R IN1, OUT**

Η σύνταξη μιας εντολής διαίρεσης δύο πραγματικών αριθμών έχει ως εξής:

/R IN1, OUT

Τα IN1 και OUT συμβολίζουν δύο πραγματικούς αριθμούς και τα *R, /R την πράξη του πολλαπλασιασμού και της διαίρεσης αντίστοιχα. Μετά την εκτέλεση κάποιας από τις δύο παραπάνω πράξεις ο IN1 παραμένει αμετάβλητος ενώ στον OUT θα αποθηκευτεί το αποτέλεσμα της αριθμητικής πράξης που εκτελέστηκε. Δηλαδή ισχύουν οι δύο παρακάτω ισότητες:

IN1*OUT = OUT
OUT/IN1 = OUT

Παράδειγμα: Να γίνει ένα πρόγραμμα το οποίο θα πολλαπλασιάζει δύο πραγματικούς αριθμούς και ένα άλλο το οποίο θα τους διαιρεί.

```
LD      I0.0
A       I0.0
MOVR    7.0, AC1
*R      2.3, AC1
END
```

```
LD      I0.0
A       I0.0
MOVR    9.1, AC1
/R      AC1, +3.5
END
```

Στα δύο παραπάνω προγράμματα χρησιμοποιούμε τον διακόπτη I0.0 για να αποθηκεύσουμε τον πραγματικό αριθμό στον AC1 και στην συνέχεια να εκτελεστούν οι αριθμητικές πράξεις. Μπορούμε να χρησιμοποιήσουμε και άλλον ένα AC αποθηκεύοντας σε αυτόν τον αριθμό που θέλουμε να πολλαπλασιάσουμε ή να διαιρέσουμε όπως στην πρόσθεση αφαίρεση.

Παράδειγμα: Έστω ότι η είσοδος S1 απαριθμεί προϊόντα τύπου Α που περνάνε από μπροστά της αλλά και μια είσοδος S2 η οποία απαριθμεί τον ίδιο τύπο προϊόντων. Στην διαδικασία αυτή θέλουμε να ανάβει η ενδεικτική λυχνία L1 αν κατά την διάρκεια της έχει περάσει ο ίδιος αριθμός προϊόντων από τις δύο εισόδους , η ενδεικτική λυχνία L2 αν έχει περάσει περισσότερος αριθμός προϊόντων από την S1 και η ενδεικτική λυχνία L3 αν έχει περάσει περισσότερος αριθμός προϊόντων από την S2.

```
LD      I0.0
LD      I0.1
CTU     C1, +0
LD      I0.2
LD      I0.3
CTU     C2, +0
LDW=    C1, C2
=       Q0.0
LDW>    C1, C2
=       Q0.1
LDW<    C1, C2
=       Q0.2
END
```

Επεξήγηση: Στο πρόγραμμα αυτό οι διακόπτες I0.0 και I0.1 αποτελούν την είσοδο S1 ενώ οι διακόπτες I0.2 και I0.3 αποτελούν την είσοδο S2. Συμπεραίνουμε λοιπόν ότι ο μετρητής C1 είναι ο μετρητής εισόδου S1 και ο C2 ο μετρητής εισόδου S2.Αν C1=C2 τότε ενεργοποιείται η λυχνία L1, αν C1>C2 τότε ενεργοποιείται η λυχνία L2 και αν C1<C2 τότε ενεργοποιείται η λυχνία L3.

Παράδειγμα: Σε μια διαδικασία ελέγχου η οποία χρονομετρείται για 10 sec ζητάμε μετά το πέρας 3 sec να γίνεται ενεργοποίηση του relay K1 και μετά το πέρας 6 sec να γίνεται ενεργοποίηση και του relay K2. Τα δύο αυτά relay να παραμένουν ενεργοποιημένα μέχρι την επανάληψη της διαδικασίας αυτής.

```
LD      I0.0
A       I0.0
TON     T37, +100
LDW=    T37, +30
S       Q0.0, 1
LDW=    T37, +60
S       Q0.1, 1
END
```

Επεξήγηση: Στο πρόγραμμα αυτό πατώντας τον διακόπτη I0.0 ξεκινά να μετράει ο timer για 10 δευτερόλεπτα. Όταν αυτός γίνει ίσος με 3 δευτερόλεπτα ενεργοποιείται η έξοδος Q0.0 με την εντολή S Q0.0, 1 ενώ όταν γίνει ίσος με 6 δευτερόλεπτα ενεργοποιείται η έξοδος Q0.1.

4.9 Βοηθητικές μνήμες.

Πολλές φορές κατά την εκτέλεση ενός προγράμματος καλούμαστε να επαναλάβουμε τμήματα του κώδικα για να εκτελέσουμε κάποιες διαδικασίες. Ένας τρόπος είναι να γράψουμε τον επαναλαμβανόμενο κώδικα τόσες φορές όσες χρειαζόμαστε. Αυτό όμως στοιχίζει σε χρόνο και σε μνήμη προγράμματος.

Η ενδεδειγμένη λύση είναι η χρήση βοηθητικών. Καταγράφεται μια φορά η λογική, αποθηκεύεται σε ένα βοηθητικό και το βοηθητικό αυτό το χρησιμοποιούμε όσες φορές και σε όποιο σημείο του προγράμματος μας θέλουμε.

Τα βοηθητικά παίζουν το ρόλο των βοηθητικών ρελέ στον κλασσικό αυτοματισμό. Τα βοηθητικά ρελέ είναι πολυάριθμες, φθηνές "έξοδοι'' του προγραμματιζόμενου λογικού ελεγκτή που δεν εμφανίζονται στην έξοδο του και δεν

έχουμε πρόσβαση σε αυτές. Συμβολίζονται με το γράμμα Μ π.χ. Μ0.0, και χρησιμοποιούνται για την αποθήκευση ενδιάμεσων αποτελεσμάτων.

4.10 Διακοπές κύκλου προγράμματος.

Όπως αναφέραμε και πιο πριν, οι προγραμματιζόμενοι λογικοί ελεγκτές εκτελούν το πρόγραμμα τους κυκλικά. Ο χρόνος κάθε κύκλου εξαρτάται από το μέγεθος του προγράμματος, τον αριθμό εισόδων και εξόδων και τον όγκο των επικοινωνιών που πρέπει να υλοποιηθούν. Ο κύκλος ξεκινά με έλεγχο της κατάστασης των εισόδων. Στη συνέχεια εκτελείται το πρόγραμμα με βάση τη λογική που περιέχει και την κατάσταση των εισόδων. Όταν ολοκληρωθεί η εκτέλεση του προγράμματος η CPU εκτελεί ορισμένες εσωτερικές διαγνωστικές λειτουργίες και πραγματοποιεί τις επικοινωνίες. Στο τέλος, ενημερώνονται οι καταστάσεις των εξόδων και ο κύκλος ξεκινά από την αρχή.

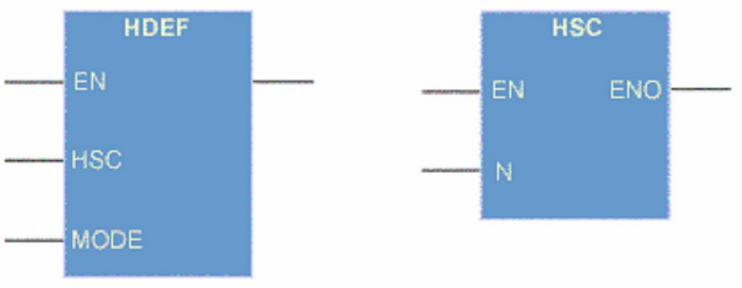
Υπάρχουν όμως κάποιες περιπτώσεις που ο προγραμματιζόμενος λογικός ελεγκτής πρέπει να αντιδράσει άμεσα σε κάποια γεγονότα που συμβαίνουν πριν να ολοκληρωθεί ο κύκλος του PLC. Αυτό γίνεται με τις λεγόμενες “διακοπές” του προγράμματος (interrupts) και με εντολές ειδικού τύπου (high speed instructions).

4.11 Γρήγοροι απαριθμητές.

Οι γρήγοροι απαριθμητές είναι σε θέση να μετρήσουν ψηφιακά σήματα που αλλάζουν κατάσταση (0-1) πολύ γρήγορα (παλμούς). Οι γρήγοροι απαριθμητές του S7-200 μπορούν να μετρήσουν παλμούς μέγιστης συχνότητας από 10 έως 50 KHz ανάλογα με το μοντέλο, τον τρόπο λειτουργίας κ.α. Στη γλώσσα Ladder οι γρήγοροι απαριθμητές αναπαρίστανται με ορθογώνια.

Οι γρήγοροι απαριθμητές έχουν δώδεκα διαφορετικούς τρόπους λειτουργίας. Μαζί με την εντολή του γρήγορου απαριθμητή υπάρχει πάντα και μια συνοδευτική εντολή που στη Ladder αναπαρίσταται επίσης με ορθογώνιο, ονομάζεται “definition

box” και αυτό που κάνει είναι να καθορίζει τον τρόπο λειτουργίας του γρήγορου απαριθμητή.



Σχήμα 4.22:Αναπαράσταση γρήγορου απαριθμητή σε γλώσσα Ladder.

5. Εισαγωγή στο προγραμματιστικό περιβάλλον του PLC.

5.1 Γενικά.

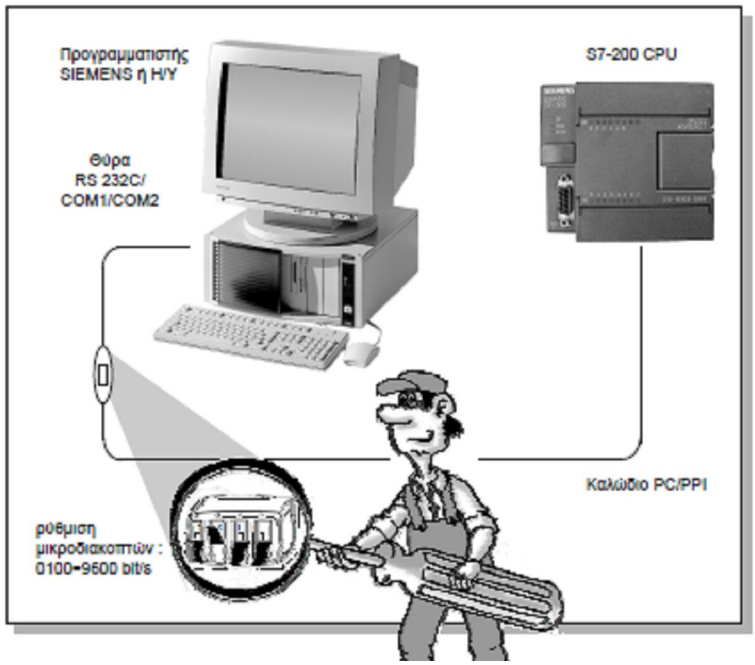
Αφού εγκαταστήσουμε το απαραίτητο λογισμικό το οποίο δίνεται με την αγορά κάθε PLC σε μορφή CD είμαστε έτοιμοι να κάνουμε δοκιμές, να γράψουμε τα πρώτα μας προγράμματα και γενικότερα να δούμε τον πώς επικοινωνεί το PLC με τον Η/Υ.

5.2 Ρύθμιση του ρυθμού μετάδοσης δεδομένων.

Το καλώδιο PC/PPI συνδέει τον Η/Υ με το PLC S7-200. Στον Η/Υ σας θα πρέπει να χρησιμοποιήσετε τη σειριακή πόρτα που έχει 9 ακίδες (pin) ή αυτή που έχει 25-pin αλλά με μετατροπέα 25pin-9pin*, π.χ. την COM 2.

Το S7-200 στέλνει και δέχεται δεδομένα με ρυθμό 9600 bit/s. Με τη βοήθεια του παρακάτω σχήματος ρυθμίστε το ρυθμό μετάδοσης δεδομένων στο καλώδιο PC/PPI. Στη συνέχεια συνδέστε τον Η/Υ σας με το PLC όπως φαίνεται στο σχήμα.

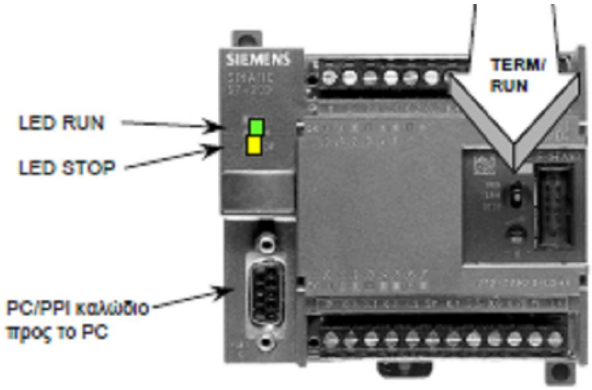
Χρησιμοποιήστε την ίδια τροφοδοσία για το PLC και τον Η/Υ για να αποφύγετε πιθανές διαφορές τάσεως. Τροφοδοτήστε το PLC με τάση. Η ενδεικτική λυχνία STOP ή RUN ανάβει.



Σχήμα 5.1:Ρύθμιση του ρυθμού μετάδοσης δεδομένων.

5.3 Πρώτη δοκιμή λειτουργίας

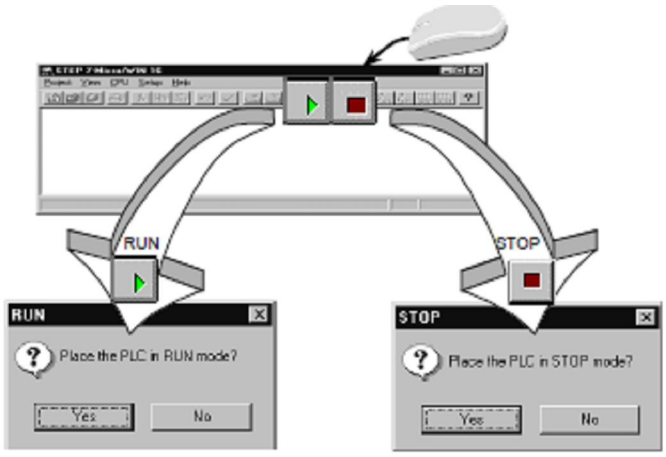
Τοποθετείστε τον επιλογικό διακόπτη που βρίσκεται κάτω από το κάλυμμα της πάνω πλευράς του PLC στη θέση TERM.



Σχήμα 5.2:Δοκιμή λειτουργίας

Σημείωση: Μόνο με το διακόπτη στη θέση **TERM** μπορείτε να επιλέξετε την κατάσταση λειτουργίας (RUN ή STOP) από τον Η/Υ.

Από τον Η/Υ βάλτε το S7 200 σε κατάσταση RUN και μετά πάλι σε STOP.



Σχήμα 5.3:Δοκιμή λειτουργίας του PLC από το πρόγραμμα

Η πράσινη ενδεικτική λυχνία RUN ανάβει όταν το PLC βρίσκεται σε κατάσταση RUN. Η κίτρινη λυχνία STOP ανάβει όταν το PLC βρίσκεται σε κατάσταση STOP. Αν μπορείτε να αλλάξετε τις καταστάσεις του PLC από τον Η/Υ σας, τότε η σύνδεση μεταξύ Η/Υ και PLC που έχετε κάνει είναι σωστή.

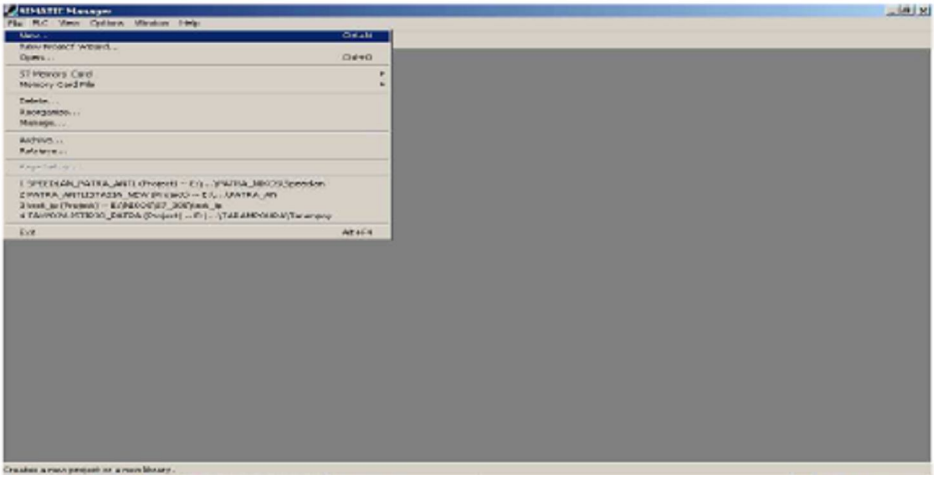
Τα παραπάνω αποτελούν τα αρχικά βήματα για την σύνδεση του PLC με τον Η/Υ. Παρακάτω θα αναλύσουμε εκτενέστερα το προγραμματιστικό περιβάλλον του PLC.

5.4 Ανοίγοντας τον "Simatic Manager"

Ο Simatic manager αποτελεί το κύριο εργαλείο στο STEP 7 αφού είναι αυτός που διαχειρίζεται τα project. Πατώντας διπλό κλικ στο εικονίδιο που υπάρχει στο desktop του Η/Υ εκκινεί το πρόγραμμα. Ένας άλλος τρόπος εκκίνησης του προγράμματος είναι από την επιλογή Start › Simatic › Step 7.

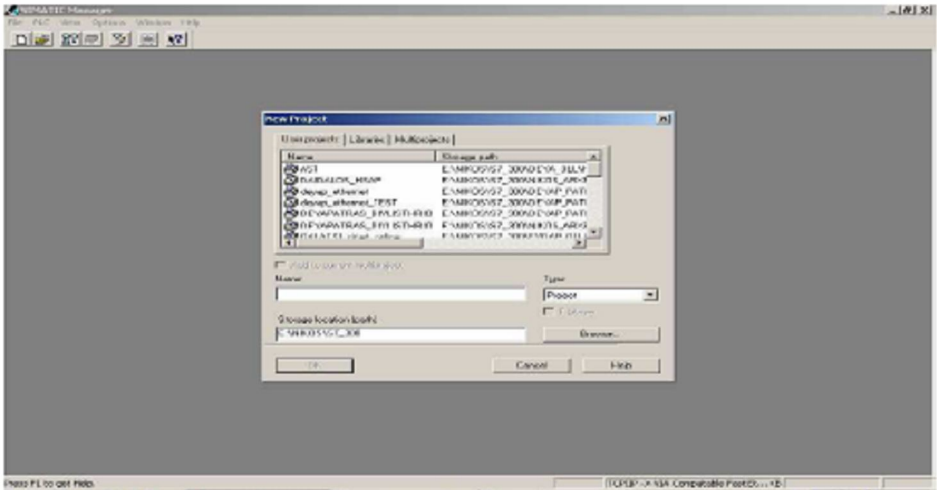
5.5 Δημιουργία Project.

Στο περιβάλλον του Simatic manager επιλέγουμε File › New όπως φαίνεται στο παρακάτω σχήμα:



Σχήμα 5.4:Δημιουργία νέου αρχείου

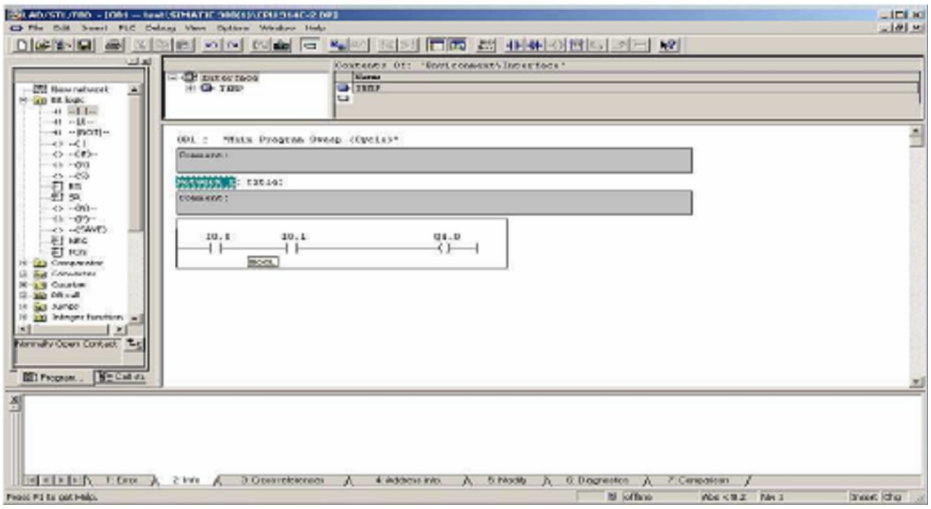
Έπειτα εμφανίζεται το παρακάτω παράθυρο δημιουργίας νέας εφαρμογής στο οποίο δίνουμε όνομα στο project που πρόκειται να φτιάξουμε και που θα αποθηκευτεί στον σκληρό δίσκο του Η/Υ.



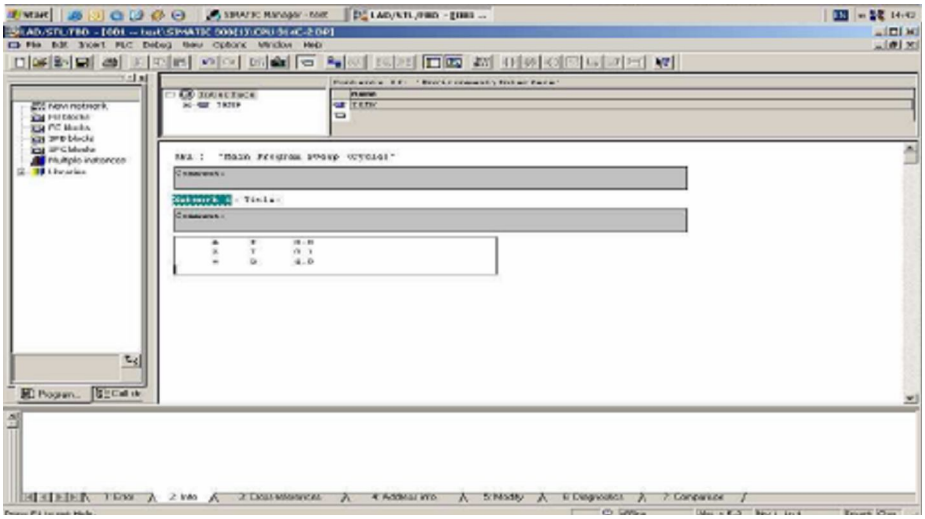
Σχήμα 5.5:Αποθήκευση αρχείου

5.6 Επιλέγοντας γλώσσα προγραμματισμού.

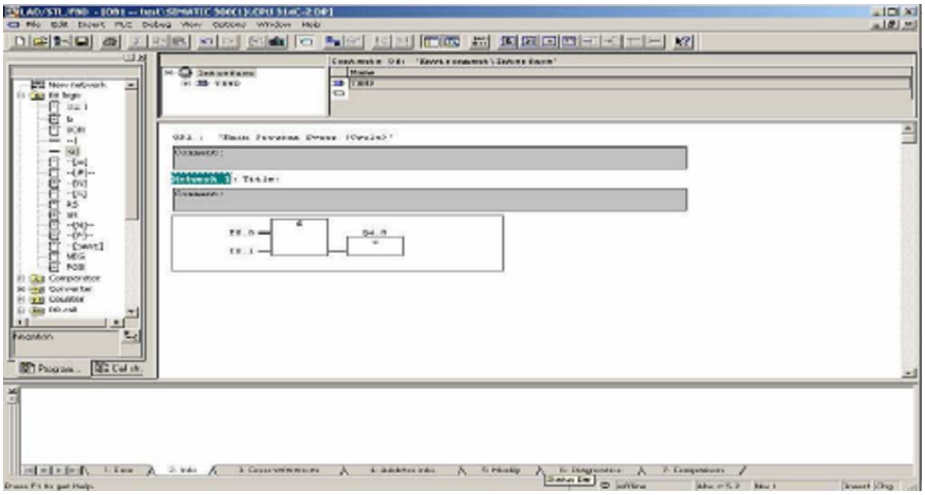
Όπως έχουμε ήδη αναφέρει τρεις είναι οι γλώσσες προγραμματισμού που χρησιμοποιούνται για την υλοποίηση μιας διαδικασίας. Έτσι και στο STEP 7 σύμφωνα με την προτίμηση μας και τις γνώσεις μας μπορούμε να επιλέξουμε ανάμεσα στην γλώσσα Ladder, STL ή FBD. Στα παρακάτω σχήματα φαίνεται το περιβάλλον της καθεμιάς γλώσσας στο STEP 7.



Σχήμα 5.6:Το περιβάλλον της γλώσσας Ladder

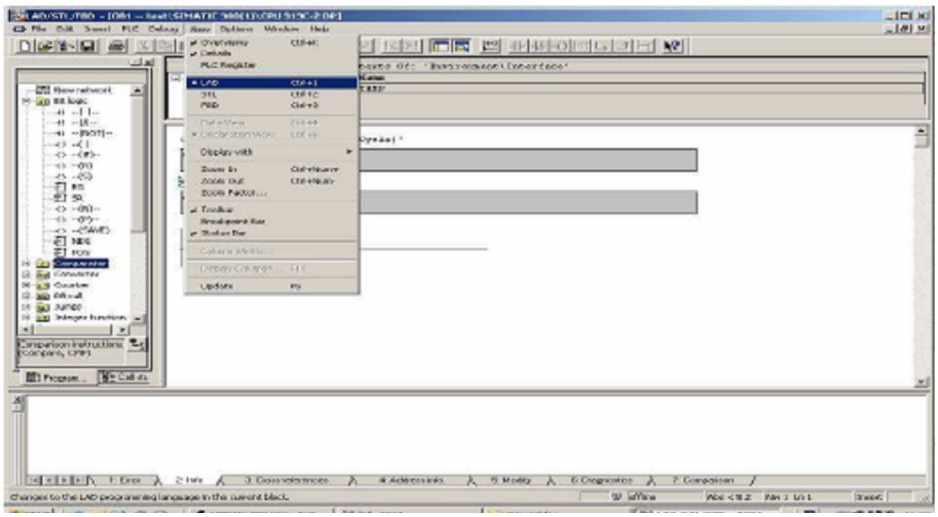


Σχήμα 5.7: Το περιβάλλον της γλώσσας STL



Σχήμα 5.8: Το περιβάλλον της γλώσσας FBD

Για να επιλέξουμε την γλώσσα προγραμματισμού που θέλουμε να γράψουμε τον κώδικα μας επιλέγουμε από το πρόγραμμα View › LAD / STL / FBD όπως φαίνεται στο παρακάτω σχήμα:

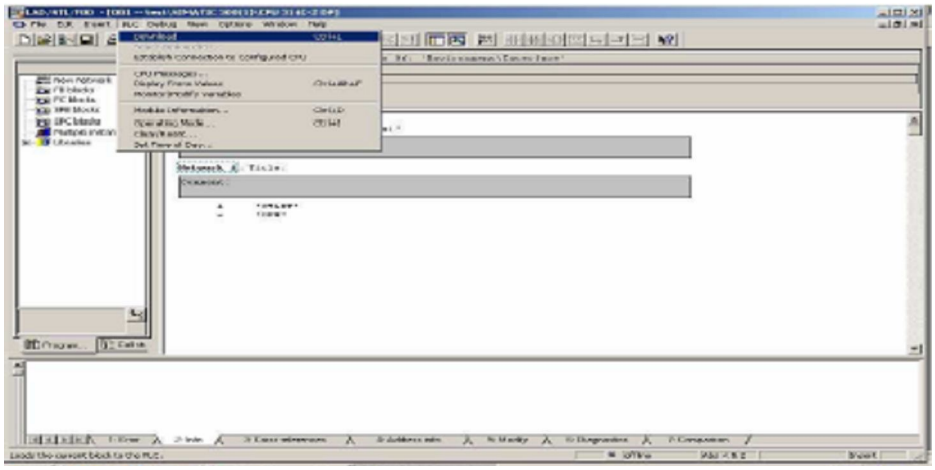


Σχήμα 5.9:Επιλογή γλώσσας προγραμματισμού

Αφού ολοκληρώσουμε τον κώδικα μας το αποθηκεύουμε στον σκληρό δίσκο του Η/Υ επιλέγοντας File › Save. Αν υπάρχουν λάθη στην σύνταξη του κώδικα το πρόγραμμα δεν μας αφήνει να το σώσουμε μέχρι να τα διορθώσουμε.

5.7 Μεταφορά προγράμματος στο PLC.

Ο κώδικας που έχουμε γράψει μέχρι τώρα βρίσκεται στον σκληρό δίσκο του Η/Υ. Το επόμενο βήμα είναι να μεταφέρουμε το πρόγραμμα στο PLC. Αυτό γίνεται από τον LAD / STL / FBD Editor επιλέγοντας PLC › Download όπως φαίνεται στο παρακάτω σχήμα:



Σχήμα 5.10:Μεταφορά προγράμματος στο PLC

ΠΑΡΑΡΤΗΜΑ: ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ

Σε αυτό το κομμάτι της πτυχιακής εργασίας παρουσιάζονται έξι προτεινόμενες εφαρμογές αυτοματισμού μαζί με τις λύσεις τους. Επίσης πρέπει να αναφερθεί ότι οι εφαρμογές αυτές έχουν προσομοιωθεί με την βοήθεια των πλακετών που κατασκευάστηκαν και αποτελούν και αυτές μέρος της πτυχιακής εργασίας.

ΕΦΑΡΜΟΓΗ 1:
ΣΥΣΤΗΜΑ ΕΞΑΕΡΙΣΜΟΥ

Το σύστημα παρέχει φρέσκο αέρα σ' ένα χώρο είτε εξάγει τον πολυκαιρισμένο αέρα.

Έχουμε δύο ανεμιστήρες (Q0.0) και (Q0.1) οι οποίοι εποπτεύονται από αισθητήρια ροής (I0.0) και (I0.1). Ο πρώτος ανεμιστήρας (Q0.0) πρέπει να ενεργοποιείται μόνο εφόσον η αξιόπιστη λειτουργία του δεύτερου (Q0.1) επιβεβαιώνεται από το αισθητήριο ροής (I0.1).

Μια προειδοποιητική λυχνία (Q0.2) δείχνει αν κάποιος από τους ανεμιστήρες παύει να λειτουργεί.

Εάν μετά από 10sec δεν ανιχνεύεται ροή αέρα (I0.2) το σύστημα απενεργοποιείται και επιδεικνύεται σφάλμα (Q0.3).

ΕΦΑΡΜΟΓΗ 1:

ΣΥΣΤΗΜΑ ΕΞΑΕΡΙΣΜΟΥ

```
Network 1                  λειτουργία ανεμιστήρα

LD    I0.1                 // επιβεβαίωση λειτουργίας 2ου ανεμιστήρα
A     I0.0                 // και επιβεβαίωση λειτουργίας 1ου ανεμιστήρα
=     Q0.0                 // λειτουργία 1ου ανεμιστήρα
=     Q0.1                 // λειτουργία 2ου ανεμιστήρα

Network 2                  σφάλμα λειτουργίας

LDN   I0.1                 // όταν έχω παύση λειτουργίας του 2ου
ON    I0.0                 // ή παύση λειτουργίας του 1ου
=     Q0.2                 // έχω ένδειξη σφάλματος λειτουργίας

Network 3                  μη ροή αέρα

LD    I0.2                 // ενεργοποίηση αισθητηρίου μη ροής αέρα
TON   T37, 100             // ξεκίνημα μετρητή για 10 sec
LD    T37                  // όταν περάσουν τα 10 sec
R     Q0.0, 1              // απενεργοποίηση 1ου ανεμιστήρα
R     Q0.1, 1              // απενεργοποίηση 2ου ανεμιστήρα
R     Q0.2, 1              // απενεργοποίηση ένδειξης σφάλματος
S     Q0.3, 1              // ένδειξη σφάλματος λειτουργίας συστήματος
LDN   I0.2                 // όταν υπάρχει ένδειξη ροής αέρα
R     Q0.3, 1              // απενεργοποίηση ένδειξης σφάλματος
END                        // τέλος
```

ΕΦΑΡΜΟΓΗ 2:
ΕΛΕΓΧΟΣ ΦΩΤΙΣΜΟΥ ΚΛΙΜΑΚΟΣΤΑΣΙΟΥ

Το πρόγραμμα αυτό ελέγχει το φωτισμό σ' ένα κλιμακοστάσιο. Τα ON buttons που βρίσκονται στους ορόφους είναι όλα συνδεμένα στην είσοδο (I0.0) του plc. Όταν κάποιο ON button πατηθεί το φως ανάβει (Q0.0) για 10sec. Αν κατά τη διάρκεια που το φως είναι αναμμένο, πατηθεί ξανά κάποιο ON button, ο χρόνος ανανεώνεται. Έτσι το φως δε σβήνει παρά μόνον αφού έχουν περάσει 10sec από την τελευταία φορά που κάποιο ON button πατηθεί. Υπάρχει γενικός διακόπτης OFF (I0.1) που σβήνει το φως.

ΕΦΑΡΜΟΓΗ 2:
ΕΛΕΓΧΟΣ ΦΩΤΙΣΜΟΥ ΚΛΙΜΑΚΟΣΤΑΣΙΟΥ

```
Network 1                    φωτισμός κλιμακοστασίου

LD      I0.0                 // πατάμε το on button
R       T37, 1               // μηδενισμός του timer
S       Q0.0, 1              // άναμμα λάμπας
LD      SM0.0                // το bit είναι πάντα on
TON     T37, 100             // ο timer ξεκινά να μετράει
LD      T37                  // όταν περάσουν 10 sec
R       Q0.0, 1              // σβήνει η λάμπα
LD      I0.1                 // πάτημα διακόπτη
R       Q0.0, 1              // σβήσιμο λάμπας
END                          // τέλος
```

ΕΦΑΡΜΟΓΗ 3:
ΕΛΕΓΧΟΜΕΝΗ ΣΤΑΘΜΕΥΣΗ

Υπάρχουν μια πύλη εισόδου οχημάτων (I0.0) καθώς και μία πύλη εξόδου (I0.1). Τα αμάξια που εισέρχονται στο χώρο καταγράφονται και όταν φτάσουν τα 10 που είναι ο μέγιστος αριθμός χωρητικότητας οχημάτων ανάβει ενδεικτική λυχνία (Q0.0). Ο μέγιστος αριθμός εξαρτάται και από τα οχήματα που φεύγουν απ' το χώρο στάθμευσης και καταγράφονται από την πύλη εξόδου. Υπάρχει κουμπί (I0.2) που μηδενίζει τον απαριθμητή.

ΕΦΑΡΜΟΓΗ 3:
ΕΛΕΓΧΟΜΕΝΗ ΣΤΑΘΜΕΥΣΗ

```
Network 1                 μετρητής

LD      I0.0              // είσοδος οχημάτων
LD      I0.1              // έξοδος οχημάτων
LD      I0.2              // κουμπί μηδενισμού
CTUD    C48, +10          // μετρητής

Network 2                 ενεργοποίηση λυχνίας

LD    C 48                // όταν υπάρχουν 10 οχήματα στον χώρο
=     Q0.0                // ενεργοποίηση λυχνίας
END                       // τέλος
```

ΕΦΑΡΜΟΓΗ 4:
ΕΛΕΓΧΟΣ ΕΙΣΟΔΟΥ ΧΩΡΟΥ ΣΤΑΘΜΕΥΣΗΣ

Η είσοδος ενός χώρου στάθμευσης αυτοκινήτων ελέγχεται αυτόματα μέσω μιας μπάρας που ανοίγει (Q0.1) κάθε φορά που ένα αυτοκίνητο εντοπιστεί από το φωτοκύτταρο (I0.0) και τοποθετηθεί κέρμα στον κερματοδέκτη (I0.3). Όταν ανοίγει τελείως η μπάρα ενεργοποιείται ένας τερματικός διακόπτης (I0.4) που σταματά το μοτέρ της. Μετά από 15sec από την ενεργοποίηση του (I0.4) η μπάρα θέλουμε να κλείνει αυτόματα.

ΕΦΑΡΜΟΓΗ 4:

ΕΛΕΓΧΟΣ ΕΙΣΟΔΟΥ ΧΩΡΟΥ ΣΤΑΘΜΕΥΣΗΣ

Network 1 άνοιγμα μπάρας

```
LD    I0.0     // ανίχνευση αυτοκινήτου από φωτοκύτταρο
A     I0.3     // και τοποθέτηση κέρματος στο μηχάνημα
=     Q0.1     // τότε έχω άνοιγμα μπάρας
=     M0.0     // αποθήκευση αποτελέσματος
```

Network 2 τερματικός διακόπτης

```
LD    I0.4     // τερματικός διακόπτης ανοιχτός
A     M0.0     // και παραπάνω αποτέλεσμα
TON   T37, 150 // έχουμε εκκίνηση μετρητή για 15 sec
LD    T37      // μετά από 15 sec
R     Q0.1, 1  // κλείνει η μπάρα
=     M0.1     // αποθήκευση μνήμης
```

Network 3 μπάρα κλειστή

```
LDN   I0.0     // αν δεν ανιχνευτεί αυτοκίνητο από φωτοκύτταρο
ON    I0.3     // ή δεν υπάρχει κέρμα στο μηχάνημα
R     Q0.1, 1  // τότε η μπάρα κατεβασμένη
END            // τέλος
```

ΕΦΑΡΜΟΓΗ 5:
ΕΛΕΓΧΟΣ ΕΙΣΟΔΟΥ ΕΞΟΔΟΥ ΠΕΡΙΟΡΙΣΜΕΝΟΥ ΑΡΙΘΜΟΥ

Ο έλεγχος των οχημάτων γίνεται όπως παραπάνω (εφαρμογή 4) με την προϋπόθεση ότι υπάρχει κενή θέση για να ανοίξει η μπάρα εισόδου. Ο μέγιστος αριθμός θέσεων στάθμευσης οχημάτων είναι δέκα και υπάρχει ενδεικτική λυχνία πλήρωσης (Q0.0).

Υπάρχει φωτοκύτταρο (I0.1) που ανιχνεύει όχημα στην έξοδο και σηκώνει την μπάρα εξόδου (Q0.2). Μόλις η μπάρα ανοίξει ενεργοποιείται ο τερματικός διακόπτης εξόδου (I0.5) και μετά από 10 sec η μπάρα κλείνει αυτόματα. Τέλος έχουμε κουμπί (I0.2) για το μηδενισμό του μετρητή οχημάτων.

ΕΦΑΡΜΟΓΗ 5:
ΕΛΕΓΧΟΣ ΕΙΣΟΔΟΥ ΕΞΟΔΟΥ ΠΕΡΙΟΡΙΣΜΕΝΟΥ ΑΡΙΘΜΟΥ

```
Network 1                  λάμπα πλήρωσης

LD      I0.0               // φωτοκύτταρο εισόδου
LD      I0.1               // φωτοκύτταρο εξόδου
LD      I0.2               // μηδενισμός μετρητή
CTUD    C48, +10           // μετρητής
LD      C48                // όταν φτάσει στα 10 οχήματα
=       Q0.0               // άναμμα λάμπας πλήρωσης
=       M0.0               // αποθήκευση

Network 2                  άνοιγμα μπάρας εισόδου

LD      I0.3               // τοποθέτηση κέρματος στο μηχάνημα
A       I0.0               // και ανίχνευση από φωτοκύτταρο
AN      M0.0               // και υπάρχει κενή θέση
S       Q0.1, 1            // τότε άνοιγμα μπάρας εισόδου
=       M0.1               // αποθήκευση
```

```
Network 3                 κλείσιμο μπάρας εισόδου

LD      I0.4              // τερματικός διακόπτης εισόδου
A       M0.1              // και μπάρα εισόδου ανοιχτή
TON     T37, 150          // τότε εκκίνηση μετρητή για 15 sec
LD      T37               // μετά από 15 sec
R       Q0.1, 1           // κλείσιμο μπάρας εισόδου

Network 4                 μπάρα κλειστή

LDN     I0.0              // αν δεν ανιχνευθεί αυτοκίνητο
ON      I0.3              // ή δεν υπάρχει κέρμα
R       Q0.1, 1           // τότε η μπάρα είναι κλειστή

Network 5                 άνοιγμα μπάρας εισόδου

LD      I0.1              // ανίχνευση αυτοκινήτου στην έξοδο
=       Q0.2              // άνοιγμα μπάρας εξόδου
=       M0.2              // αποθήκευση

Network 6                 κλείσιμο μπάρας εξόδου

LD      I0.5              // τερματικός διακόπτης εξόδου
A       M0.2              // και μπάρα ανοιχτή
TON     T38, 100          // εκκίνηση μετρητή για 10 sec
LD      T38               // μετά από 10 sec
R       Q0.2, 1           // κλείσιμο μπάρας εξόδου
END                       // τέλος
```

ΕΦΑΡΜΟΓΗ 6:
ΕΛΕΓΧΟΣ ΦΩΤΕΙΝΗΣ ΣΗΜΑΝΣΗΣ ΚΥΚΛΟΦΟΡΙΑΣ

Φωτεινός σηματοδότης ελέγχει την κυκλοφορία των οχημάτων και των πεζών προς μια κατεύθυνση σε διασταύρωση. Το σύστημα τίθεται σε λειτουργία πατώντας τον διακόπτη (I0.0).

Οι φωτεινοί σηματοδότες (Q0.0) πράσινο, (Q0.1) πορτοκαλί, (Q0.2) κόκκινο ελέγχουν την κυκλοφορία των οχημάτων, ενώ οι φωτεινοί σηματοδότες (Q0.3) πράσινο και (Q0.4) κόκκινο ελέγχουν την κυκλοφορία των πεζών.

Για τα οχήματα το (Q0.0) ανάβει για 20 sec, το (Q0.1) για 5 sec, το (Q0.2) για 15 sec και για τους πεζούς το (Q0.3) ανάβει για 10 sec, το (Q0.5) για 25 sec.

Το φανάρι έχει κουμπί πεζών (I0.2) όπου εάν πατηθεί ανάβει μόνο το κόκκινο οχημάτων και το πράσινο πεζών για 10 sec και μετά το σύστημα συνεχίζει κανονικά.

Τέλος το όλο σύστημα τίθεται σε νυχτερινή λειτουργία πατώντας τον διακόπτη (I0.3) κατά την οποία αναβοσβήνει το πορτοκαλί φανάρι των οδηγών μόνο.

ΕΦΑΡΜΟΓΗ 6:

ΕΛΕΓΧΟΣ ΦΩΤΕΙΝΗΣ ΣΗΜΑΝΣΗΣ ΚΥΚΛΟΦΟΡΙΑΣ

```
Network 1                    παύση συστήματος

LDN    I0.0                  // κλείσιμο γενικού διακόπτη
S      M0.0, 1               // παύση λειτουργίας πράσινου οχημάτων
S      M0.1, 1               // παύση λειτουργίας πορτοκαλί οχημάτων
S      M0.2, 1               // παύση λειτουργίας κόκκινου οχημάτων
S      M0.3, 1               // παύση λειτουργίας πράσινου πεζών
S      M0.4, 1               // παύση λειτουργίας κόκκινου πεζών

Network 2                    φανάρι πράσινων οχημάτων

LDN    M0.0                  // επανάληψη μνήμης
TON    T37, +400             // για χρόνο διπλάσιο από 20 sec
LDW>=  T37, +200             // αφού τιμή μεγαλύτερη από 20 sec
=      Q0.0                  // άναμμα φαναριού
LD     T37                   // μέχρι να περάσει ο χρόνος
=      M0.0                  // αποθήκευση μνήμης

Network 3                    φανάρι πορτοκαλί οχημάτων

LDN    M0.1                  // επανάληψη μνήμης
TON    T38, +100             // για χρόνο διπλάσιο από 5 sec
LDW>=  T38, +50              // αφού τιμή μεγαλύτερη από 5 sec
=      Q0.1                  // άναμμα φαναριού
LD     T38                   // μέχρι να περάσει ο χρόνος
=      M0.1                  // αποθήκευση μνήμης
```

Network 4 φανάρι κόκκινο οχημάτων

```
LDN     M0.2          // επανάληψη μνήμης
TON     T39, +300     // για χρόνο διπλάσιο από 15 sec
LDW>= T39, +150       // αφού τιμή μεγαλύτερη από 15 sec
=       Q0.2          // άναμμα φαναριού
LD      T39           // μέχρι να περάσει ο χρόνος
=       M0.2          // αποθήκευση μνήμης
```

Network 5 φανάρι πράσινο πεζών

```
LDN     M0.3          // επανάληψη μνήμης
TON     T40, +200     // για χρόνο διπλάσιο από 10 sec
LDW>= T40, +100       // αφού τιμή μεγαλύτερη από 10 sec
=       Q0.3          // άναμμα φαναριού
LD      T40           // μέχρι να περάσει ο χρόνος
=       M0.3          // αποθήκευση μνήμης
```

Network 6 φανάρι κόκκινο πεζών

```
LDN     M0.4          // επανάληψη μνήμης
TON     T41, +500     // για χρόνο διπλάσιο από 25 sec
LDW>= T41, +250       // αφού τιμή μεγαλύτερη από 25 sec
=       Q0.5          // άναμμα φαναριού
LD      T41           // μέχρι να περάσει ο χρόνος
=       M0.4          // αποθήκευση μνήμης
```

Network 7 παύση νυχτερινής λειτουργίας

```
LDN   I0.2            // κλείσιμο διακόπτη νυχτερινής λειτουργίας
S     M0.6, 1         // παύση νυχτερινής λειτουργίας
```

```
Network 8                    νυχτερινή λειτουργία

LDN     M0.6                 // επανάληψη μνήμης
R       Q0.0, 1              // σβήσε πράσινο οχημάτων
R       Q0.2, 1              // σβήσε κόκκινο οχημάτων
R       Q0.3, 1              // σβήσε πράσινο πεζών
R       Q0.5, 1              // σβήσε κόκκινο πεζών
TON     T43, +20             // για χρόνο διπλάσιο από 1 sec
LDW>= T43, +10               // αφού τιμή μεγαλύτερη από 1 sec
=       Q0.1                 // άναμμα πορτοκαλί
LD      T43                  // μέχρι να περάσει ο χρόνος
=       M0.6                 // αποθήκευση μνήμης

Network 9                    κουμπί πεζών

LD      I0.1                 // πάτημα κουμπιού πεζών
AN      I0.2                 // και διακόπτης νυχτερινής λειτουργίας κλειστός
TOF     T44, +100            // εκκίνηση μετρητή για 10 sec
LD      T44                  // για 10 sec
R       Q0.0, 1              // σβήσε πράσινο οχημάτων
R       Q0.1, 1              // σβήσε πορτοκαλί οχημάτων
R       Q0.5, 1              // σβήσε κόκκινο πεζών
S       Q0.2, 1              // άναψε κόκκινο οχημάτων
S       Q0.3, 1              // άναψε πράσινο πεζών
END                          // τέλος
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Ν.Μαραντίδη: Αυτοματισμός με SIMATIC S7,ΣΗΜΕΝΣ ΑΕ 2000
2. Ν.Α Πανταζής: Προγραμματιζόμενοι λογικοί ελεγκτές, Εκδόσεις "ΙΩΝ",1992
3. Ν.Α Πανταζής: Αυτοματισμοί με PLC, Εκδόσεις Α.Σταμούλης,1998
4. Ν.Ζούλης, Π.Καφφετζάκης, Γ.Σούλτης: Συστήματα αυτοματισμών Β΄Τόμος,2000
5. Δ.Βέντζας, Ν.Γλώσσας, Α.Νικολόπουλος: Εργαστήριο αυτοματισμών και συστημάτων αυτομάτου ελέγχου, 2001

ΗΛΕΚΤΡΟΝΙΚΑ ΕΓΧΕΙΡΙΔΙΑ

1. SIEMENS: S7-200 Programmable controller system manual.
2. SIEMENS: Simatic Step 7 Documentation.
3. SIEMENS: 1 hour primer
4. SIEMENS: 2 hour primer
5. SIEMENS: Βιομηχανικοί αυτοματισμοί με PLC.