



ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ Σ.Τ.Ε.Φ.
ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ

<< ΠΑΡΑΓΩΓΗ ΑΝΑΛΟΓΙΚΩΝ ΣΗΜΑΤΩΝ ΜΕΣΩ ΘΥΡΑΣ USB >>



Πτυχιακή Εργασία: Καλαμάρης Δημήτριος (Α.Μ. 06/2274)

Επίβλεψη: Κιζήρογλου Μιχάλης

Θεσσαλονίκη Νοέμβριος 2013

*Αφιερωμένη στην οικογένεια μου, που με στήριξαν καθόλη
την διάρκεια των σπουδών μου.*

*Επίσης θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή μου κ. Κυζήρογλου Μιχάλη,
που μου εμπιστεύτηκε την παρούσα διπλωματική εργασία και για τις
πολύτιμες συμβουλές του*

ΕΙΣΑΓΩΓΗ

Σε αυτή την διπλωματική εργασία δημιουργήσαμε ένα αναλογικό σήμα μέσω μιας θύρας usb. Για την μελέτη ,την κατανόηση και πραγματοποίηση της πτυχιακής χρειάστηκε να ανατρέξουμε σχεδόν σε όλες τις εμπειρίες κ γνώσεις που λάβαμε από τις σπουδές στο Τμήμα Αυτοματισμού .Χρειάστηκε να προμηθευτούμε ένα module με ενσωματωμένη θύρα usb το UM245R και ένα DAC7611.Πρωτος στόχος μας ήταν να μάθουμε να κατανοήσουμε την λειτουργιά του UM245R .Στη συνέχεια ,γράψαμε τον κώδικα στο περιβάλλον τις Visual C++. Στον προγραμματισμό του UM25R χρησιμοποιήσαμε της βιβλιοθήκες του ώστε να μπορέσουμε να συνδεθεί με το λειτουργικό τον Windows 7.Δημιουργησαμε ένα τριγωνικό σήμα μέχρι τα12bit. Από τις 8 εξόδους του UM245R χρησιμοποιήσαμε τις 3 πρώτες ώστε ι μια ν στέλνει τα δεδομένα στον converter η δεύτερη να στέλνει τους παλμούς του ρολογιού και η τρίτη να μας δείχνει ποτέ φορτώνετε το σήμα μας. όλα αυτά συμφώνα με το διάγραμμα χρονισμού του DAC. Έπειτα, προσθέσαμε τον DAC7611 όπου θα μετατρέψει τα ψηφιακά δεδομένα σε αναλογικό σήμα ώστε να μπορέσουμε να το απεικονίσουμε στον παλμογράφο.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Κεφάλαιο 1: ΘΕΩΡΗΤΙΚΗ ΕΙΣΑΓΩΓΗ

Κεφάλαιο 2: UM245R-DAC7611

Κεφάλαιο 3: ΠΕΡΙΓΡΑΦΗ ΚΥΚΛΩΜΑΤΟΣ

Κεφάλαιο 4 :ΠΕΡΙΓΡΑΦΗ ΚΩΔΙΚΑ

Βιβλιογραφία

1. ΘΕΩΡΗΤΙΚΗ ΕΙΣΑΓΩΓΗ

Βασικές τεχνολογίες εισόδου/εξόδου

Τα κυκλώματα εισόδου/εξόδου (I/O) των υπολογιστικών συστημάτων μάς εξασφαλίζουν τη δυνατότητα της επικοινωνίας με άλλους υπολογιστές και περιφερειακές συσκευές. Ο έλεγχος εξωτερικών οργάνων, καθώς και η εισαγωγή δεδομένων από το περιβάλλον προς το υπολογιστικό κύκλωμα, επιτυγχάνεται μέσω των θυρών επικοινωνίας, με τη χρήση κατάλληλων τεχνικών.

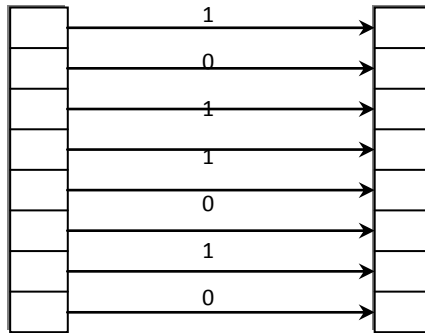
Στα επόμενα, με τον όρο *υπολογιστικό σύστημα* αναφερόμαστε κυρίως σε προσωπικούς υπολογιστές (PC), αλλά και σε μικροελεγκτές, που επίσης διαθέτουν κυκλώματα εισόδου/εξόδου (I/O) και μπορούν να χρησιμοποιήσουν τις ίδιες τεχνικές επικοινωνίας. Επίσης ο όρος μπορεί να αναφέρεται σε «έξυπνες συσκευές», ικανές να διασυνδεθούν και να επικοινωνήσουν με άλλες συσκευές σε ένα υπολογιστικό περιβάλλον. Για το λόγο αυτό οι βασικές αρχές θα παρουσιαστούν γενικά.

Παράλληλη Επικοινωνία

Στην παράλληλη επικοινωνία κάθε ψηφιολέξη που εισάγεται προς επεξεργασία ή που παράγεται ως αποτέλεσμα σε ένα υπολογιστικό σύστημα εμφανίζεται στον λεγόμενο *διάδρομο δεδομένων*: κάθε bit εμφανίζεται σε μια γραμμή του διαδρόμου και όλα τα bits της ψηφιολέξης εμφανίζονται ταυτόχρονα. Η εισαγωγή ή εξαγωγή της ψηφιολέξης από και προς το εξωτερικό περιβάλλον γίνεται με τη βοήθεια των διαύλων εισόδου/εξόδου (I/O) και με την υποστήριξη κατάλληλων κυκλωμάτων, που ονομάζονται *θύρες εισόδου/εξόδου (I/O ports)*.

Ο πιο άμεσος τρόπος για τη μετάδοση δεδομένων είναι η *παράλληλη επικοινωνία*. Στην παράλληλη επικοινωνία όλα τα bits μιας λέξης δεδομένων μεταδίδονται ταυτόχρονα προς τον αποδέκτη. Το υπολογιστικό σύστημα φορτώνει το προς μετάδοση δεδομένο σε έναν καταχωρητή, που ενέχει θέση μνήμης με

Καταχωρητής



Σχήμα 1.1 Παράλληλη μεταφορά οκτώ bits

συγκεκριμένη διεύθυνση στον χάρτη της μνήμης του συστήματος. Τα κυκλώματα εξόδου μεταβιβάζουν το περιεχόμενο του καταχωρητή μέσω ενός καλωδίου πολλαπλών συρμάτων στο κύκλωμα λήψης. Η ιδέα αυτή εικονίζεται στο Σχήμα 1.1.

Τα πλεονεκτήματα της παράλληλης μεταφοράς δεδομένων εντοπίζονται κυρίως στην ταχύτητα μετάδοσης. Είναι προφανές ότι με τον τρόπο αυτόν η μετάδοση γίνεται πολύ γρήγορα, αφού όλα τα bits μιας ψηφιολέξης μεταφέρονται ταυτόχρονα από τον πομπό στον αποδέκτη. Ουσιαστικά, ο ρυθμός μετάδοσης περιορίζεται από την ταχύτητα με την οποία λειτουργούν τα κυκλώματα εισόδου/εξόδου του πομπού και του δέκτη.

Τα μειονεκτήματα της παράλληλης μετάδοσης είναι το κόστος και η δυσκολία της εγκατάστασης των καλωδίων, λόγω του σημαντικού αριθμού τους σε κάθε επικοινωνιακή ζεύξη. Εξάλλου, σε μεγάλες αποστάσεις το σήμα εξασθενεί και συχνά παρατηρείται αλληλεπίδραση των σημάτων που μεταδίδονται στις παράλληλες γραμμές ενός πολλαπλού καλωδίου.

Για τους παραπάνω λόγους η παράλληλη επικοινωνία χρησιμοποιείται κυρίως στη διασύνδεση συσκευών σε μικρές αποστάσεις, όταν υπάρχει ανάγκη για υψηλούς ρυθμούς μετάδοσης της πληροφορίας. Έτσι, συσκευές όπως οι εκτυπωτές και οι οπτικοί σαρωτές, που απαιτούν ταχεία μεταφορά σημαντικής ποσότητας πληροφορίας, διασυνδέονται μέσω παράλληλης επικοινωνίας με τον ηλεκτρονικό υπολογιστή.

Σημειώνεται ότι οι περισσότεροι σύγχρονοι υπολογιστές διαθέτουν διάδρομο δεδομένων (data bus) εύρους 32 bits. Εντούτοις, η παράλληλη μεταφορά των δεδομένων προς εξωτερικές συσκευές γίνεται με ταυτόχρονη μετάδοση μόνον οκτώ bits κάθε φορά.

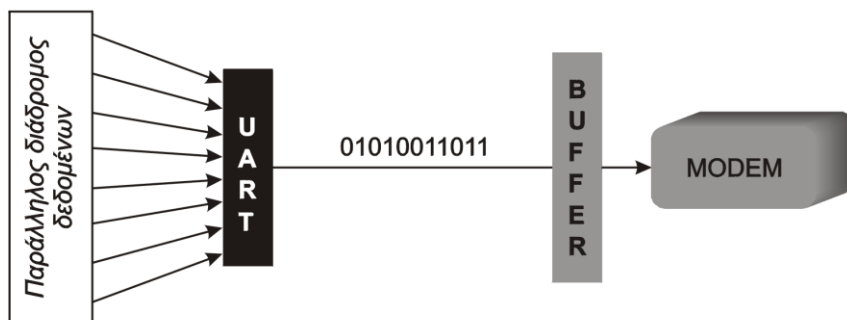
Τα πιο γνωστά πρωτόκολλα παράλληλης επικοινωνίας είναι το πρωτόκολλο CENTRONICS, για την τυπική παράλληλη θύρα ενός υπολογιστή

PC και το πρωτόκολλο IEEE 488 ή GPIB, το οποίο χρησιμοποιείται σε πολλά επιστημονικά όργανα για τη δημιουργία αυτοματοποιημένων συστημάτων μετρήσεων.

Σειριακή Επικοινωνία

Ένας τρόπος μετάδοσης της πληροφορίας, ειδικά σε σημαντικές αποστάσεις, είναι η σειριακή επικοινωνία. Με τον τρόπο αυτό τα bits της πληροφορίας μεταδίδονται ένα κάθε φορά, στη σειρά, μέσα από έναν αγωγό μεταφοράς των δεδομένων. Στην απλούστερη περίπτωση τέτοιας επικοινωνίας χρειαζόμαστε τρεις συνολικά αγωγούς, έναν για την αποστολή δεδομένων, έναν για τη λήψη και έναν που θα βρίσκεται στο δυναμικό αναφοράς των μεταδιδόμενων σημάτων.

Είναι προφανές ότι για να αποσταλούν με σειριακό τρόπο κάποια δεδομένα μέσω μίας θύρας επικοινωνίας ενός ηλεκτρονικού υπολογιστή, πρέπει πρώτα να μετατραπούν από τη παράλληλη μορφή, με την οποία εμφανίζονται στο διάδρομο δεδομένων, σε σειριακή μορφή. Τη λειτουργία αυτή αναλαμβάνει ένα κύκλωμα που ονομάζεται *UART* (*Universal Asynchronous Receiver/Transmitter*), το οποίο υπάρχει σε ολοκληρωμένη μορφή επάνω στη μητρική πλακέτα ή στις μονάδες ελέγχου των περιφερειακών συσκευών ενός υπολογιστή. Η λειτουργία του κυκλώματος αυτού στηρίζεται στη λειτουργία του καταχωρητή ολίσθησης, ο οποίος



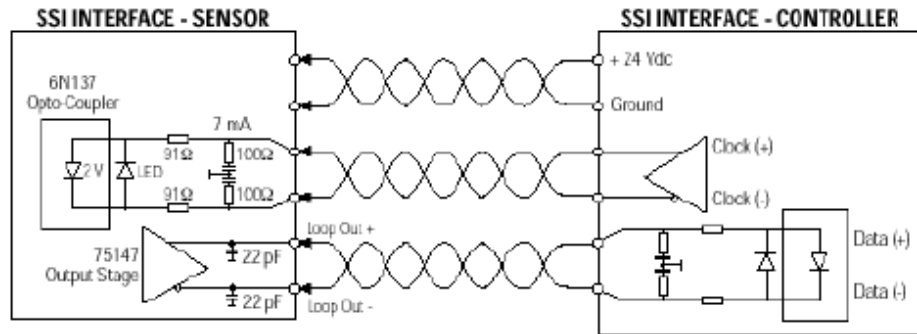
Σχήμα 1.2 Σειριακή μετάδοση πληροφορίας από τον υπολογιστή προς εξωτερική συσκευή

αφού λάβει κάποια δεδομένα και τα καταχωρήσει στα flip-flops που διαθέτει, ολισθαίνει τα bits της ψηφιολέξης που έχει καταχωρήσει ένα-ένα προς τα δεξιά ή προς τα αριστερά.

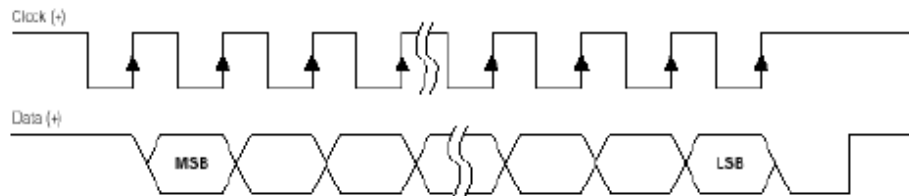
Τα κυριότερο πλεονέκτημα της σειριακής επικοινωνίας είναι ο μικρός αριθμός καλωδίων διασύνδεσης που απαιτείται, σε σχέση με την παράλληλη επικοινωνία. Αυτό κάνει την εγκατάσταση φθηνότερη όταν οι αποστάσεις είναι μεγάλες. Επιπλέον, τα πρωτόκολλα επικοινωνίας που χρησιμοποιούνται στη σειριακή επικοινωνία επιτρέπουν μεγάλες στάθμες σημάτων σε σχέση με τα πρωτόκολλα της παράλληλης επικοινωνίας, οπότε οι απώλειες του σήματος δημιουργούν μικρότερο πρόβλημα και η μετάδοση σε μεγάλη απόσταση είναι εφικτή. Εξάλλου, με την σειριακή επικοινωνία είναι πολύ ευκολότερη η ασύρματη μετάδοση, ειδικά μέσω διατάξεων υπέρυθρης ακτινοβολίας, που είναι πολύ διαδεδομένες. Τέλος, η σειριακή μετάδοση είναι πιο κατάλληλη για χρήση με μικροελεγκτές. Ο λόγος είναι ότι οι διάφορες διατάξεις, όπως μετατροπείς A/D, μνήμες, υπολογιστές κ.λπ. καταλαμβάνουν πολύ λιγότερους ακροδέκτες του μικροελεγκτή όταν επικοινωνούν σειριακά με αυτόν, παρά όταν επικοινωνούν παράλληλα. Εξάλλου, μερικά συστήματα μικροελεγκτών έχουν ενσωματωμένες θύρες σειριακής διασύνδεσης με το εξωτερικό περιβάλλον, κάτι που καθιστά απλή τη σειριακή διασύνδεσή τους με περιφερειακές συσκευές. Για της βιομηχανίες έχει αναπτυχθεί το SSI (Serial Synchronous Interface). Το SSI είναι μια ευρέως χρησιμοποιούμενη σειριακή διεπαφή μεταξύ των αισθητήρων και των ελεγκτών σε βιομηχανικά συστήματα αυτοματισμού. Το SSI χρησιμοποιεί μια σειρά παλμών ρολογιού από έναν ελεγκτή για να ξεκινήσει μία έξοδο από τον αισθητήρα. Τα δεδομένα ενημερώνεται συνεχώς από τον αισθητήρα και περνούν στον καταχωρητή ολίσθησης. Μεταξύ των παλμών του ρολογιού υπάρχει μια ελάχιστη καθυστέρηση των 25 χιλιοστών του δευτερολέπτου, κατά την οποία τα καινούρια δεδομένα μεταφέρονται στον καταχωρητή. Τα δεδομένα μετατοπίζεται όταν ο αισθητήρας λάβει παλμο ρολογιού από το controller. όταν το ρολόι είναι high και η ελάχιστη χρονοκαθυστέρηση παρέλθει νέα δεδομένα είναι διαθέσιμα για να διαβάσετε.

Το διάγραμμα απεικονίζει τη λειτουργία της θέσης με SSI. Η θέση του μαγνήτη τοποθετημένο σε μια μηχανή με ακρίβεια προσδιορίζεται με μέθοδο μαγνητοσυστολής. Η τιμή μετατόπισης παρέχεται σε ένα Binary 24 ή 25-bit ή GRAY κώδικα. Η ροή δεδομένων μεταδόθηκαν σε έναν ελεγκτή μέσω της διεπαφής SSI σε απόσταση έως 240m με baudrate των 100kbps.

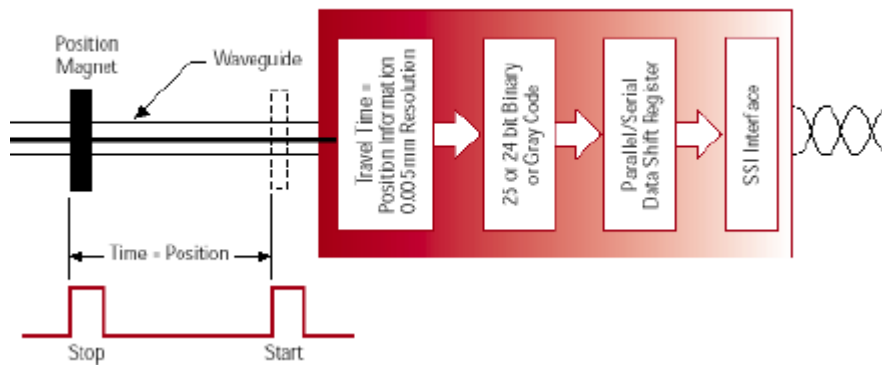
SSI Block Diagram



SSI Timing Diagram



<<Serial Synchronous Interface>>



<<Serial Synchronous interface application>>

Θύρα USB

Όποιος αποκτά μια εξοικείωση με την επικοινωνία μέσω της παράλληλης και της σειριακής θύρας, θεωρεί ότι είναι φυσικό κάποια στιγμή να μεταβεί στη χρήση τη θύρας USB. Είναι φανερό ότι η αγορά έχει αποδεχτεί πλήρως τη νέα αυτή θύρα επικοινωνίας, που προφανώς υπερέχει σε ευελιξία και σε τεχνικά χαρακτηριστικά, όπως η ταχύτητα μεταφοράς δεδομένων, ο μέγιστος αριθμός

των υποστηριζόμενων περιφερειακών, η διαχείριση της ηλεκτρικής ενέργειας κ. ά.

Ο Γενικός Σειριακός Δίαυλος (Universal Serial Bus) γεννήθηκε μέσα από την ανάγκη να συνδέσουμε διαφόρων ειδών περιφερειακά, με έναν τύπο συνδέσμου, κατά το δυνατό μικρό σε μέγεθος, καταργώντας την ποικιλία των καλωδίων και των μετασχηματιστών τροφοδοσίας, που συναντάμε σχεδόν σε κάθε υπολογιστικό σύστημα.

Από φυσική άποψη, ο σύνδεσμος USB αποτελείται από τέσσερις επαφές από τις οποίες ξεκινούν δύο ζεύγη καλωδίων. Το ένα ζεύγος καλωδίων χρησιμοποιείται από το πρωτόκολλο για την σειριακή εκπομπή διαφορικών σημάτων ανάμεσα στον υπολογιστή (host ή ξενιστή) και στην περιφερειακή συσκευή. Το δεύτερο ζεύγος χρησιμεύει για την παροχή τροφοδοσίας στα περιφερειακά. Ο σύνδεσμος από τη μεριά του ξενιστή (σύνδεσμος A) είναι διαφορετικός από το σύνδεσμο προς τη μεριά των περιφερειακών συσκευών (σύνδεσμος B) και έτσι είναι αδύνατο να συνδεθεί το καλώδιο ανάποδα.

Το USB έχει ριζικά εκσυγχρονίσει τη διαδικασία αυτή . Οι χρήστες μπορούν να προσθέτουν και αφαιρούν περιφερειακά όπως εκτυπωτές , φωτογραφικές μηχανές , σαρωτές , και ένα ευρύ φάσμα των ανθρώπινων Συσκευές διασύνδεσης χρήστη (HIDs) χωρίς επανεκκίνηση . Συσκευές είναι " hot swappable " . Οι οδηγοί συσκευών είναι ακόμη απαραίτητοι, αλλά το μόνο που χρειάζεται είναι να εγκατασταθούν μια φορά . Και οι εξειδικευμένες εσωτερικές κάρτες είναι περιττες όταν μια συσκευή συνδέεται μέσω της θύρας USB . Το USB έχει επίσης εξαιρετική ικανότητα επέκτασης . Όταν συνδυάζεται με USB hubs, επαρκείς πόρους για PC , και με την κατάλληλη καλωδίωση , μια ενιαία θύρα USB μπορεί να διαχειριστεί μέχρι και 127 συσκευές . (Συγκριτικά , ένα πρότυπο DB9 σύνδεση ελέγχει μόνο ένα κομμάτι του εξοπλισμού .) Έτσι, δεν υπάρχει καμία αμφιβολία ότι η USB έχει αποδειχθεί ότι είναι πολύ χρήσιμο και δεν έχει έρθει εδώ για να μείνει . Αλλά για να πάρει τα μέγιστα από τον εξοπλισμό που χρησιμοποιούν παλαιότερα πρωτόκολλα , και για να κρατήσει το λειτουργικό εξοπλισμό , με πλήρη λειτουργικότητα σε όλη την ωφέλιμη διάρκεια ζωής του , οι παλαιότερες συσκευές πρέπει να είναι σε θέση να επικοινωνούν με τα συστήματα που χρησιμοποιούν USB . Που μπορεί να γίνει με αρκετά εύκολα με USB σε σειριακή μετατροπείς - με την προϋπόθεση ότι έχετε επιλέξει το σωστό μετατροπέα .

Έχουν κατασκευαστεί σειρές από βιομηχανικές υποδοχές USB (π.χ. IP65/IP67 ODVA). Οι σύνδεσμοι είναι ιδανικές για όλες τις βιομηχανικές εφαρμογές USB όπου απαιτείται μια μεγάλη δυνατότητα ασφάλειας . Οι συσκευές έχουν σχεδιαστεί για τη διασύνδεση με μικροελεγκτές για

μηχατρονικές συνδέσεις και I / O interfaces στη βιομηχανική τεχνολογία αυτοματισμού , μηχανολογία (π.χ. πάνελ σύνδεσης) και μηχανολογικών εγκαταστάσεων . Τυπικές εφαρμογές περιλαμβάνουν βιομηχανικές λύσεις PC , ελέγχου και αυτοματισμού, inverters και ρομποτικές εφαρμογές , καθώς και συστημάτων fieldbus .

ETHERNET

Το Ethernet είναι μακράν η πιο διαδεδομένη τεχνολογία LAN σήμερα , συνδέοντας περισσότερα από το 85 τοις εκατό στον κόσμο των LAN συνδεδεμένων υπολογιστές και σταθμούς εργασίας . Το Ethernet αναφέρεται στην οικογένεια των τεχνολογιών δικτύωσης υπολογιστών που καλύπτονται από το πρότυπο IEEE 802.3 , και μπορεί να τρέξει πάνω και από οπτικές ίνες και από καλώδια στρίβο-ζευγαριού (twisted-pair cables) . Με τα χρόνια , το Ethernet έχει σταθερά εξελιχθεί για να παράσχει πρόσθετη απόδοση και ευφυΐα στα δίκτυα . Αυτή η συνεχής βελτίωση έχει κάνει το Ethernet μια εξαιρετική λύση για βιομηχανικές εφαρμογές . Σήμερα , η τεχνολογία αυτή μπορεί να παρέχει τέσσερις ρυθμούς δεδομένων .

- 10BASE - T Ethernet προσφέρει απόδοση έως και 10 Mbps πάνω από χάλκινο καλώδιο συνεστραμμένου ζεύγους .

- Το Fast Ethernet αύξανε της προδιαγραφές (100 Mbps) ταχύτητας του 10BASE - T Ethernet ενώ διατηρώντας πολλές από τις τεχνικές προδιαγραφές του Ethernet . Οι ομοιότητες αυτές επιτρέπουν στους οργανισμούς να χρησιμοποιούν 10BASE -T εφαρμογές και τα εργαλεία διαχείρισης του δικτύου σε γρήγορα δίκτυα Ethernet.

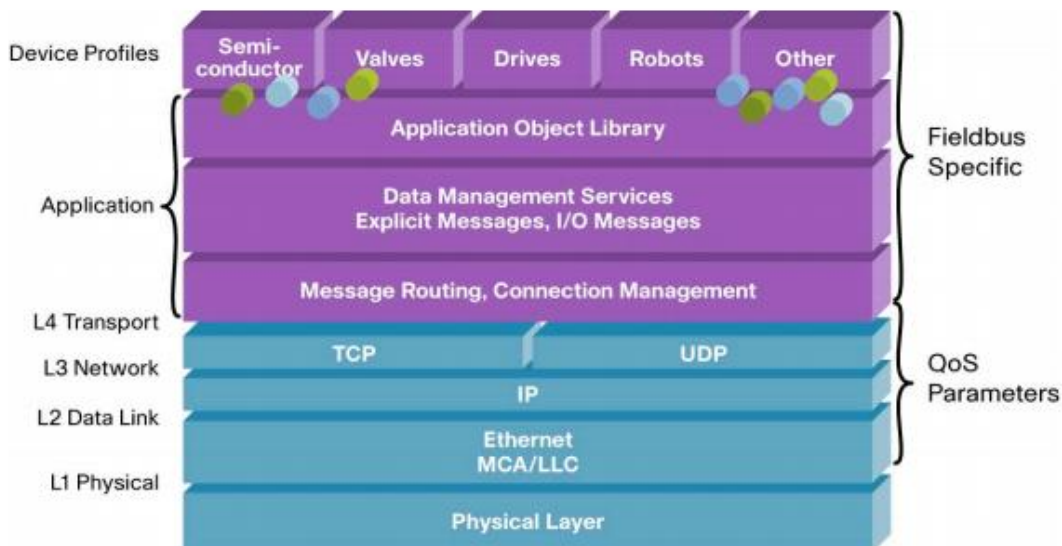
- Gigabit Ethernet επεκτείνει το πρωτόκολλο Ethernet ακόμη περισσότερο , αυξάνοντας την ταχύτητα δέκα φορές πάνω από Fast Ethernet σε 1000 Mbps ή 1 Gbps . Επειδή βασίζεται στο σημερινό πρότυπο Ethernet και είναι συμβατό με την εγκατεστημένη βάση του Ethernet και Fast Ethernet switches και routers , οι διαχειριστές του δικτύου μπορεί να υποστηρίξουν Gigabit Ethernet χωρίς να χρειάζεται να επανακατάρτιθουν ή να μάθουν μια νέα τεχνολογία

- 10 Gigabit Ethernet , που επικυρώθηκε ως πρότυπο τον Ιούνιο του 2002 , αποτελεί μία ακόμη ταχύτερη έκδοση του Ethernet . Επειδή το 10 Gigabit Ethernet είναι ένα είδος Ethernet , που μπορεί να στηρίξει ευφυείς Ethernet -based υπηρεσίες δικτύου , μπορεί να λειτουργήσει με τις υπάρχουσες αρχιτεκτονικές , και την ελάχιστη προσπάθεια εκμάθησης των χρηστών . Η υψηλή ταχύτητα μετάδοσης

δεδομένων των 10 Gbps προσθετεί μια καλή λύση για την παροχή υψηλού εύρους ζώνης σε δίκτυα ευρείας περιοχής και των μητροπολιτικών δικτύων (MAN).

Αναγνωρίζοντας ότι το Ethernet είναι η κορυφαία λύση δικτύωσης, πολλές βιομηχανικές οργανώσεις εισήγαγαν την παραδοσιακή fieldbus αρχιτεκτονική για το Industrial Ethernet. Το Industrial Ethernet εφαρμόζει τα πρότυπα που έχουν αναπτυχθεί για τα δεδομένα επικοινωνία με τα δίκτυα ελέγχου παραγωγής (Σχήμα 3). Χρησιμοποιώντας IEEE standards-based, οι βιομηχανίες μπορούν να μεταφέρουν το σύνολο ή μέρος των εργασιών στο εργοστάσιο τους σε ένα περιβάλλον Ethernet με τον ρυθμό που επιθυμούν.

Figure 3. Using Intelligent Ethernet for Automation Control



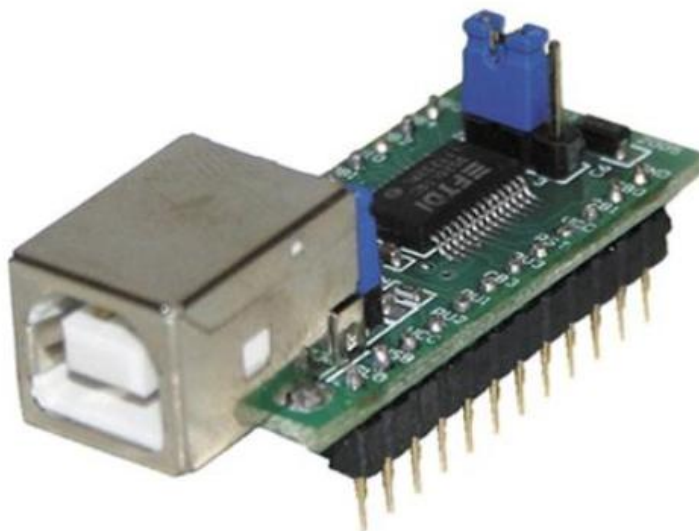
Για παράδειγμα, το πρωτόκολλο Common Industrial (CIP) έχει εφαρμογές που βασίζονται στο Ethernet και στο IP protocol suite (Ethernet / IP), στο DeviceNet και στο ControlNet (μεταξύ άλλων). Οι περισσότεροι ελεγκτές (με τις κατάλληλες συνδέσεις δικτύου) μπορεί να μεταφέρουν δεδομένα από το ένα δίκτυο στα άλλα, αξιοποιώντας τις υφιστάμενες εγκαταστάσεις, αλλά εκμεταλλευόμενοι το Ethernet. Η fieldbus δομή δεδομένων εφαρμόζεται σε στρώματα 5, 6, και 7 του μοντέλου αναφοράς OSI από τα Ethernet, IP, και TCP / UDP στο επίπεδο μεταφοράς (Layer 4). Το πλεονέκτημα του Industrial Ethernet είναι ότι οι βιομηχανίες και οι συσκευές μπορούν να συνεχίσουν να χρησιμοποιούν τα παραδοσιακά εργαλεία τους και τις εφαρμογές που τρέχουν σε μια πολύ πιο αποτελεσματική υποδομή δικτύωσης. Το Industrial Ethernet δεν δίνει μόνο την γρήγορη επικοινωνία μεταξύ των κατασκευαστικών συσκευών, αλλά και δίνει στους χρήστες τη καλύτερη

συνδεσιμότητα και τη διαφάνεια, επιτρέποντας στους χρήστες να συνδεθούν με τις συσκευές που θέλουν χωρίς να απαιτείται ξεχωριστές πύλες.

2.UM245R-DAC7611

2.1.UM245R

Το UM245R είναι ένα module που της εταιρίας FTDI (Future Technology Devices International Ltd) το οποίο χρησιμοποιεί το chip FT245R. Το FT245R είναι ένα usb με παράλληλο FIFO interface. Επιπλέον διαθέτει τις λειτουργίες σύγχρονη και ασύγχρονη bit bang interface. Από USB σε παράλληλο σχεδιάστηκε χρησιμοποιώντας το FT245R έχει απλοποιηθεί περαιτέρω με την πλήρη ενσωμάτωση του εξωτερικού EEPROM , κύκλωμα ρολογιού και USB αντιστάσεις επάνω στη συσκευή. Η UM245R παρέχεται σε ένα PCB το οποίο έχει σχεδιαστεί για να συνδέσετε σε ένα πρότυπο 15,0 χιλιοστών (0,6 ") σε πλάτος 24 pin DIP . Όλα τα εξαρτήματα που χρησιμοποιούνται , συμπεριλαμβανομένης της FT245RL είναι Pb - free (RoHS συμβατό) .



2.1.1 The Bit Bang Mode-Asynchronous

Η Bit mode Bang είναι μια ειδική λειτουργία της συσκευής FT232R και FT245R που αλλάζει τις 8 γραμμές I/O σε ένα 8bit αμφίδρομο δίαυλο δεδομένων. Υπάρχουν τρεις τύποι των bit mode Bang για την FT232R: ασύγχρονης Bang Bit, το οποίο είναι το ίδιο με το FTDI BM και C chip-style Bang Bit mode, με την προσθήκη της «ανάγνωσης» και «εγγραφής» στην περίπτωση του FT232R. Στη «Σύγχρονη Bit Bang» τα δεδομένα μόνο διαβάζονται. Αυτή είναι η ίδια λειτουργία με την σύγχρονη λειτουργία του FT2232 Bang Bit. Η CBUS Bit Bang λειτουργία, μια νέα 4-bit έκδοση του Bit λειτουργία Bang είναι διαθέσιμα στο FT232R CBUS καρφίτσες αλλά όχι για το FT245R.

Η ασύγχρονη και σύγχρονη λειτουργία Bang Bit ενεργοποιείτε από τις εντολές του driver, ενώ CBUS Bit Bang πρέπει να συσταθεί στη συσκευή EEPROM πριν να μπορεί να ενεργοποιηθεί με την εντολή του driver.

Εμείς χρησιμοποιήσαμε στον προγραμματισμό του UM245r την ασύγχρονη λειτουργία. Στην Ασύγχρονη λειτουργία Bit Bang είναι το ίδιο με BM-style λειτουργία Bang Bit. Οποιαδήποτε δεδομένα γράφονται στη συσκευή κατά τον συνήθη τρόπο θα αυτοσυγχρονίζονται στα pins τα οποία έχουν διαμορφωθεί ως έξοδοι. Κάθε pin μπορεί να καθοριστεί αυτοτελώς ως είσοδος ή έξοδος. Ο ρυθμός ότι τα δεδομένα συγχρονίζονται ελέγχεται από την Baud-rate γεννήτρια. Για αλλάξουν τα δεδομένα πρέπει να εγγραφούν νέα και το ρολόι της Baud-rate πρέπει να «τερματίσει». Αν δεν εγγράφουν νέα δεδομένα στη συσκευή, τα pins θα κρατήσουν την τελευταία τιμή του γράφτηκε.

Όταν η Ασύγχρονη Bang Bit mode είναι ενεργοποιημένη οι γραμμές I/O διαμορφώνονται ως εξής :

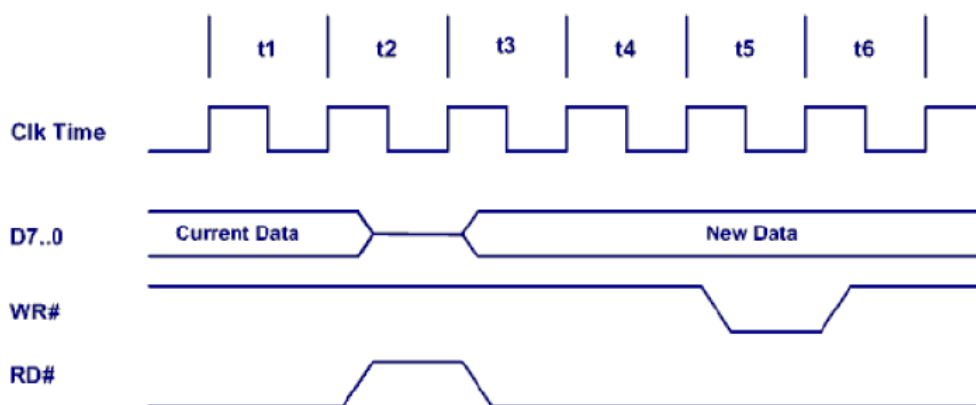
FT232RL/FT245RL Pin Number	FT232RQ/FT245RQ Pin Number	Signal	Type	Description
1	30	D0	Input/Output	Bit Bang data Bus bit 0
5	2	D1	Input/Output	Bit Bang data Bus bit 1
3	32	D2	Input/Output	Bit Bang data Bus bit 2
11	8	D3	Input/Output	Bit Bang data Bus bit 3
2	31	D4	Input/Output	Bit Bang data Bus bit 4
9	6	D5	Input/Output	Bit Bang data Bus bit 5
10	7	D6	Input/Output	Bit Bang data Bus bit 6
6	3	D7	Input/Output	Bit Bang data Bus bit 7

Μερικές D2XX εντολές του οδηγού είναι αναγκαίες για τη χρήση Ασύγχρονη Bang Bit. Οι πιο σημαντικές αναφέρονται παρακάτω:

D2XX Function	Description
FT_SetBitMode	Asynchronous Bit Bang mode is enabled using the FT_SetBitMode command. A value of 0x01 will enable it and a value of 0x00 will reset the device mode. (see note 1)
FT_SetBaudRate	The rate of data transfer can be controlled by using the FT_SetBaudRate command. The maximum Baud rate is 3MBaud, but to allow time for the data to be setup and held around the WR# strobe the Baud rate should be less than 1MBaud. The clock for the Asynchronous Bit Bang mode is actually 16 times the Baud rate. A value of 9600 Baud would transfer the data at $(9600 \times 16) = 153600$ bytes per second, or 1 every 6.5 μ S.
FT_Write	Data can be written to the device in Asynchronous Bit Bang mode using the FT_Write command. If multiple bytes are written to the device the values on the pins will change at the rate set by FT_SetBaudRate
FT_GetBitMode	FT_GetBitMode returns the instantaneous value of the pins. A single byte will be returned containing the current values of the pins, both those which are inputs and those which are outputs.
FT_Read	FT_Read will return a buffer of values which have been sampled from the pins at the rate set by FT_SetBaudRate. If the read buffers have filled, data will be lost. Each byte returned contains the values of the pins, both those which are inputs and those which are outputs.

2.1.2 The Bit Bang Mode-Synchronous

Με την σύγχρονη λειτουργία Bit Bang, τα δεδομένα θα σταλούν μόνο αν υπάρχει χώρος στη συσκευή για τα δεδομένα που πρέπει να διαβαστούν από τα pins. Επίσης στην ασύγχρονη λειτουργία θα διαβαστούν τα δεδομένα που βρίσκονται στα pins, προτού σταλεί το byte θα μεταφερθεί. Συνεπώς, υπάρχει 1 byte πίσω από την έξοδο και έτσι θα διαβάσετε τις εισόδους για το byte που μόλις έστειλε, και ένα άλλο byte πρέπει να σταλεί.



<<το διάγραμμα χρονισμού >>

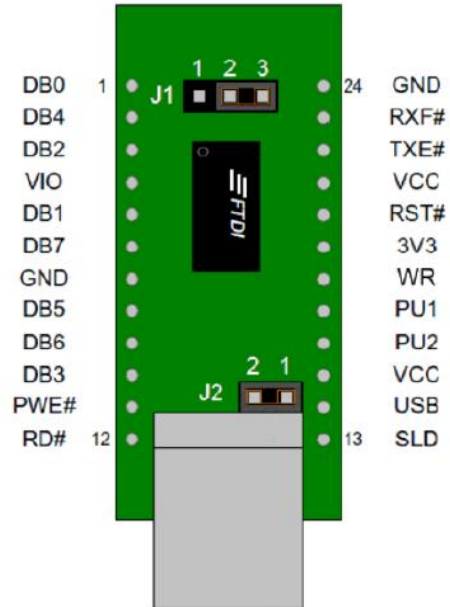
Όταν η σύγχρονη Bang Bit mode είναι ενεργοποιημένη οι γραμμές I/O διαμορφώνονται ως εξής :

FT232RL/FT245RL Pin Number	FT232RQ/FT245RQ Pin Number	Signal	Type	Description
1	30	D0	Input/Output	Bit Bang data Bus bit 0
5	2	D1	Input/Output	Bit Bang data Bus bit 1
3	32	D2	Input/Output	Bit Bang data Bus bit 2
11	8	D3	Input/Output	Bit Bang data Bus bit 3
2	31	D4	Input/Output	Bit Bang data Bus bit 4
9	6	D5	Input/Output	Bit Bang data Bus bit 5
10	7	D6	Input/Output	Bit Bang data Bus bit 6
6	3	D7	Input/Output	Bit Bang data Bus bit 7

Μερικές D2XX εντολές του οδηγού είναι αναγκαίες για τη χρήση Ασύγχρονη Bang Bit. Οι πιο σημαντικές αναφέρονται παρακάτω:

D2XX Function	Description
FT_SetBitMode	Asynchronous Bit Bang mode is enabled using the FT_SetBitMode command. A value of 0x01 will enable it and a value of 0x00 will reset the device mode. (see note 1)
FT_SetBaudRate	The rate of data transfer can be controlled by using the FT_SetBaudRate command. The maximum Baud rate is 3MBaud, but to allow time for the data to be setup and held around the WR# strobe the Baud rate should be less than 1MBaud. The clock for the Asynchronous Bit Bang mode is actually 16 times the Baud rate. A value of 9600 Baud would transfer the data at $(9600 \times 16) = 153600$ bytes per second, or 1 every 6.5 μ S.
FT_Write	Data can be written to the device in Asynchronous Bit Bang mode using the FT_Write command. If multiple bytes are written to the device the values on the pins will change at the rate set by FT_SetBaudRate
FT_GetBitMode	FT_GetBitMode returns the instantaneous value of the pins. A single byte will be returned containing the current values of the pins, both those which are inputs and those which are outputs.
FT_Read	FT_Read will return a buffer of values which have been sampled from the pins at the rate set by FT_SetBaudRate. If the read buffers have filled, data will be lost. Each byte returned contains the values of the pins, both those which are inputs and those which are outputs.

2.1.3 Περιγραφή UM245R



Περιγραφή θυρών UM245R

Pin No.	Name	Type	Description
1	DB0	I/O	FIFO Data Bus Bit 0*
2	DB4	I/O	FIFO Data Bus Bit 4*
3	DB2	I/O	FIFO Data Bus Bit 2*
4	VIO	PWR	+1.8V to +5.25V supply to the FIFO Interface and Control group pins (1...3, 5, 6, 9...14, 22, 23). In USB bus powered designs connect to 3V3OUT to drive out at 3.3V levels (connect jumper J1 pins 1 and 2 together), or connect to VCC to drive at 5V CMOS level (connect jumper J1 pins 2 and 3 together). This pin can also be supplied with an external 1.8V - 5.0V supply in order to drive at different levels. It should be noted that in this case this supply should originate from the same source as the supply to Vcc. This means that in bus powered designs a regulator which is supplied by the 5V on the USB bus should be used.
5	DB1	I/O	FIFO Data Bus Bit 1*
6	DB7	I/O	FIFO Data Bus Bit 7*
7, 24	GND	PWR	Module ground supply pins
8	DB5	I/O	FIFO Data Bus Bit 5*
9	DB6	I/O	FIFO Data Bus Bit 6*
10	DB3	I/O	FIFO Data Bus Bit 3*
11	PWE#	I/O	Goes low after the device is configured by USB, then high during USB suspend. Can be used to control power to external logic P-Channel logic level MOSFET switch. Enable the interface pull-down option when using the PWREN# pin in this way.
12	RD#	I/O	Enables the current FIFO data byte on D0...D7 when low. Fetches the next FIFO data byte (if available) from the receive FIFO buffer when RD# goes from high to low. See Section 4.4 for timing diagram.*
13	SLD	GND	USB Cable shield.
14	USB	Output	5V Power output USB port. For a low power USB bus powered design, up to 100mA can be sourced from the 5V supply on the USB bus. A maximum of 500mA can be sourced from the USB bus in a high power USB bus powered design.
15, 21	VCC	PWR or Output	These two pins are internally connected on the module PCB. To power the module from the 5V supply on USB bus, connect jumper J2 pins 1 and 2 together (this is the module default configuration). In this case these pins would have the same description as pin 14. To use the UM245R module in a self powered configuration, ensure that jumper J2 pins 1 and 2 are not connected together, and apply an external 3.3V to 5.25V supply to one or both of these pins.
17	PU1	Control	Pull up resistor pin connection 2. Connect to pin 17 (RST#) in a self powered configuration.
16	PU2	Control	Pull up resistor pin connection 1. Connect to pin 14 (USB) in a self powered configuration.
19	3V3	Output	3.3V output from integrated LDO regulator. This pin is decoupled to ground on the module PCB with a 100nF capacitor. The prime purpose of this pin is to provide the internal 3.3V supply to the USB transceiver cell and the internal 1.5kΩ pull up resistor on USBDP. Up to 50mA can be drawn from this pin to power external logic if required. This pin can also be used to supply the FT245RL's VCCIO pin by connecting this pin to pin 4 (VIO), or by connecting together pins 1 and 2 on jumper J1.
20	RST#	Input	Can be used by an external device to reset the FT245R. If not required can be left unconnected, or pulled up to VCCIO.
18	WR	I/O	Writes the data byte on the D0...D7 pins into the transmit FIFO buffer when WR goes from high to low. See Section 4.4 for timing diagram.*

Pin No.	Name	Type	Description
22	TXE#	I/O	When high, do not write data into the FIFO. When low, data can be written into the FIFO by strobing WR high, then low. During reset this signal pin is tri-state, but pulled up to VCCIO via an internal 200kΩ resistor. See Section 4.4 for timing diagram.
23	RXF#	I/O	When high, do not read data from the FIFO. When low, there is data available in the FIFO which can be read by strobing RD# low, then high again. During reset this signal pin is tri-state, but pulled up to VCCIO via an internal 200kΩ resistor. See Section 4.4 for timing diagram. If the Remote Wakeup option is enabled in the internal EEPROM, during USB suspend mode (PWREN# = 1) RXF# becomes an input which can be used to wake up the USB host from suspend mode. Strobing the pin low will cause the device to request a resume on the USB bus.

Pin No.	Name	Type	Description
1	3V3	Output	3.3V output from integrated LDO regulator. This pin is decoupled to ground on the module PCB with a 100nF capacitor. The prime purpose of this pin is to provide the internal 3.3V supply to the USB transceiver cell and the internal 1.5kΩ pull up resistor on USBDP. Up to 50mA can be drawn from this pin to power external logic if required. This pin can also be used to supply the FT245RL's VCCIO pin by connecting this pin to pin 4 (VIO), or by connecting together pins 1 and 2 on jumper J1.
2	VIO	PWR	+1.8V to +5.25V supply to the FIFO Interface and control pins (1...3, 5, 6, 9...14, 22, 23). In USB bus powered designs connect to 3V3 to drive out at 3.3V levels (connect jumper J1 pins 1 and 2 together), or connect to VCC to drive out at 5V CMOS level (connect jumper J1 pins 2 and 3 together). This pin can also be supplied with an external 1.8V - 2.8V supply in order to drive out at lower levels. It should be noted that in this case this supply should originate from the same source as the supply to Vcc. This means that in bus powered designs a regulator which is supplied by the 5V on the USB bus should be used.
3	VCC	PWR	VCC Output. This will be 5V from the USB bus if pins 1 and 2 on jumper J2 are connected. Alternatively, if the module is in a self powered configuration, the supply to the VCC module pins (15 and 21) will be brought out to this jumper pin. Connect this jumper J1 pin 2 in order to supply the device IO pins from the supply to VCCIO.

Pin No.	Name	Type	Description
1	USB	PWR	5V Power output USB port. For a low power USB bus powered design, up to 100mA can be sourced from the 5V supply on the USB bus. A maximum of 500mA can be sourced from the USB bus in a high power USB bus powered design with the use of an external power switch (See Section 7.3).
2	VCC	PWR or Output	Board supply input. Connect to jumper J2 pin 1 in order to supply the board from the USB bus. This pin is internally connected to the VCC DIP pins. Remove the jumper connector in a self powered design.

Διαγράμματα Διεπαφής ελέγχου χρονικών κύκλων

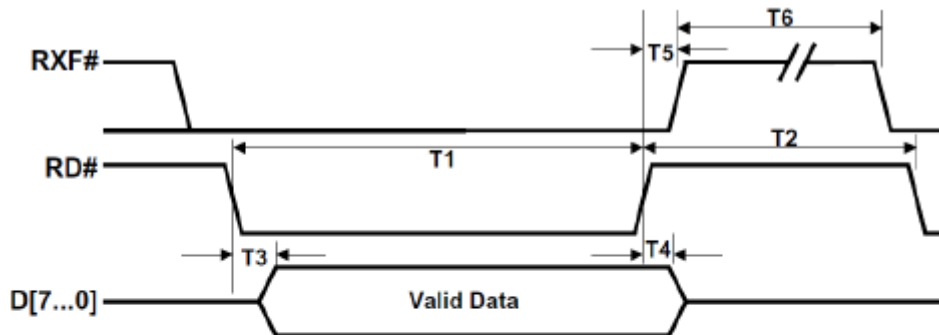


Figure 4.2 FIFO Read Cycle

Time	Description	Min	Max	Unit
T1	RD# Active Pulse Width	50		ns
T2	RD# to RD# Pre-Charge Time	50 + T6		ns
T3	RD# Active to Valid Data*	20	50	ns
T4	Valid Data Hold Time from RD# Inactive*	0		ns
T5	RD# Inactive to RXF#	0	25	ns
T6	RXF# Inactive After RD# Cycle	80		ns

Table 4.4 FIFO Read Cycle Timings

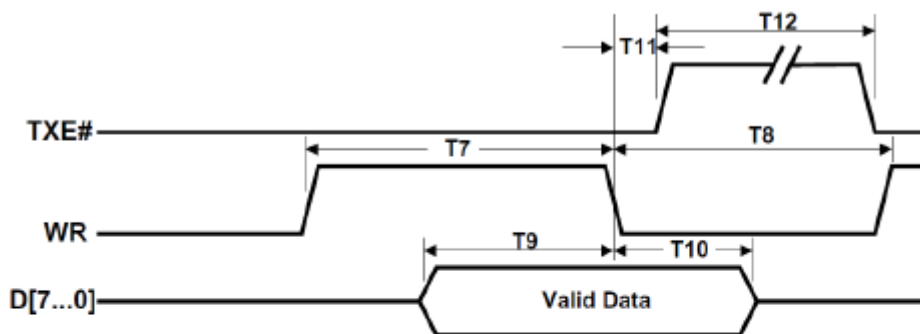


Figure 4.3 FIFO Write Cycle

Time	Description	Min	Max	Unit
T7	WR Active Pulse Width	50		ns
T8	WR to RD Pre-Charge Time	50		ns
T9	Data Setup Time before WR Inactive	20		ns
T10	Data Hold Time from RD Inactive	0		ns
T11	WR Inactive to TXE#	0	25	ns
T12	TXE# Inactive After WR Cycle	80		ns

2.2 DAC7611

Περιγραφή DAC7611

Το DAC7611 είναι ένας 12-bit D to A μετατροπέας(DAC).Απαιτεί 5V τροφοδοσία και περιέχει έναν shift καταχωρητή εισόδου, ένα latch, 2.435V τιμή αναφοράς, DAC, έναν ενισχυτή παράγωγης υψηλής ταχύτητας rail to rail. Για ένα παραγοντικό βήμα μεγέθους η έξοδος θα γίνει σε 1 LSB στα 7ms. Η συσκευή καταναλώνει 2.5mW (0.5mA σε 5V). Το σύγχρονο σειριακό interface είναι συμβατό με ευρεία ποικιλία των DSPs και μικροελεγκτών. Το ρολόι(CLK), σειριακά δεδομένα εισόδου (SDI), και το LD περιλαμβάνει τη σειριακή θύρα. Επιπλέον, δύο ακόμα pins παρέχουν ένα CS λειτουργίας και μια ασύγχρονη clear (CLR) εισόδου. από την εφαρμογή. Η είσοδος CLR μπορεί να χρησιμοποιηθεί για να εξασφαλιστεί ότι η έξοδος του DAC7611 είναι 0V για power-up ή όπως απαιτείται από την εφαρμογή. Τέλος, το DAC7611 είναι διαθέσιμο σε μια 8-SOIC μολύβδου ή 8-pin πλαστική συσκευασία DIP .

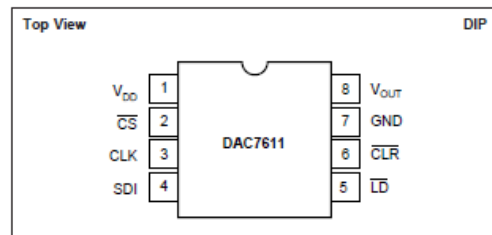
Χαρακτηριστικά

- LOW POWER: 2.5mW
- FAST SETTLING: 7ms to 1 LSB
- 1mV LSB WITH 4.095V FULL-SCALE RANGE
- COMPLETE WITH REFERENCE
- 12-BIT LINEARITY AND MONOTONICITY OVER INDUSTRIAL TEMP RANGE
- ASYNCHRONOUS RESET TO 0V
- 3-WIRE INTERFACE: Up to 20MHz Clock
- ALTERNATE SOURCE TO DAC8512

ΕΦΑΡΜΟΓΕΣ

- ΕΛΕΓΧΟΣ ΔΙΕΡΓΑΣΙΑΣ
- ΑΠΟΚΤΗΣΗ ΔΕΔΟΜΕΝΩΝ
- ΕΛΕΓΧΟΣ ΜΕ ΣΕΡΒΟΜΗΧΑΝΙΣΜΟ ΚΛΕΙΣΤΟΝ ΒΡΟΧΩΝ
- ΠΕΡΙΦΕΡΙΑΚΟ ΥΠΟΛΟΓΙΣΤΗ
- ΦΟΡΗΤΗ ΕΝΟΡΓΑΝΩΣΗ

ΔΙΑΜΟΡΦΩΣΗ ΑΚΡΟΔΕΚΤΩΝ

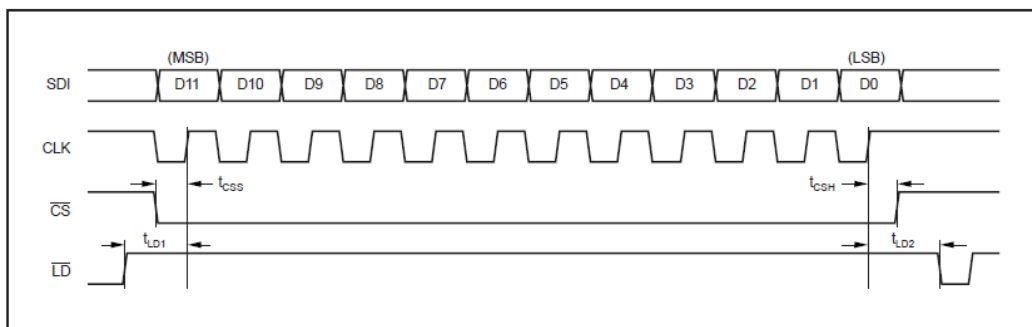


ΠΕΡΙΓΡΑΦΗ ΑΚΡΟΔΕΚΤΩΝ

PIN	LABEL	DESCRIPTION
1	V_{DD}	Power Supply
2	\overline{CS}	Chip Select (active LOW).
3	CLK	Synchronous Clock for the Serial Data Input.
4	SDI	Serial Data Input. Data is clocked into the internal serial register on the rising edge of CLK.
5	\overline{LD}	Loads the Internal DAC Register. NOTE: The DAC register is a transparent latch and is transparent when \overline{LD} is LOW (regardless of the state of \overline{CS} or CLK).
6	\overline{CLR}	Asynchronous Input to Clear the DAC Register. When \overline{CLR} is strobed LOW, the DAC register is set to 000_H and the output voltage to $0V$.
7	GND	Ground
8	V_{OUT}	Voltage Output. Fixed output voltage range of approximately $0V$ to $4.095V$ ($1mV/LSB$). The internal reference maintains this output range over time, temperature, and power supply variations (within the values defined in the specifications section).

ΔΙΑΓΡΑΜΜΑΤΑ ΧΡΟΝΙΣΜΟΥ

TIMING DIAGRAMS



3.ΠΕΡΙΓΡΑΦΗ ΚΥΚΛΩΜΑΤΟΣ

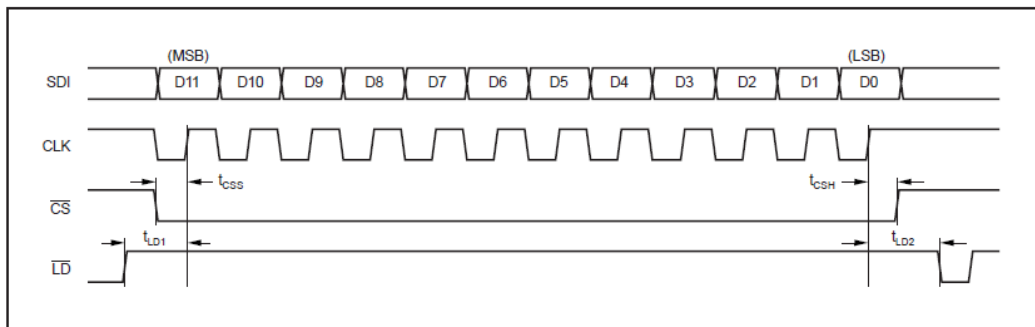
Το πρώτο μέρος του κυκλώματος μας είναι ο υπολογιστής όπου με μια θύρα usb το συνδέουμε με το UM245R. Από το UM245R χρησιμοποιήσαμε τις πρώτες 3 εξόδους(D0,D1,D2). Από αυτές τις 3 εξόδους συνδέουμε τον DAC7611 στους ακροδέκτες D0->SDI, D1->CLK και D2->LD. Από τον DAC από τον ακροδέκτη Vout στον παλμογράφο.

<φωτο>

4.ΠΕΡΙΓΡΑΦΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

Ο κώδικας γράφτηκε στο περιβάλλον της Visual C++ . Χρησιμοποιήσαμε το Software Application Development D2XX Programmer's Guide που το προμηθευτήκαμε από το www.ftdichip.com .Η εταιρία χορηγεί τους drivers και της βιβλιοθήκες που χρειάστηκαν. Αφού κάνουμε connect το UM245R με τον υπολογιστή καταφέραμε να δημιουργήσουμε ένα κώδικα σύμφωνα με το παρακάτω διάγραμμα χρονισμού του DAC7611

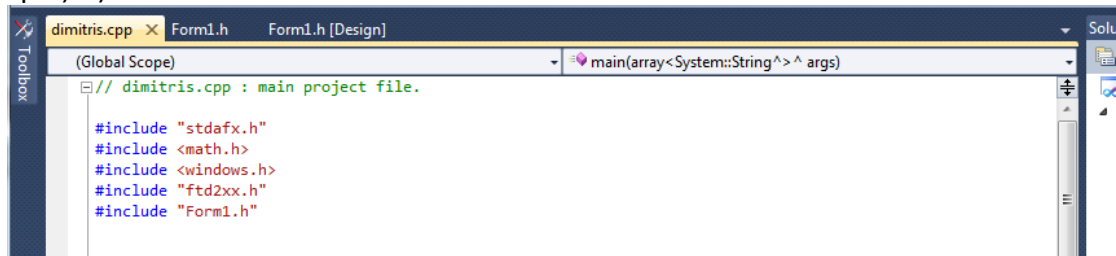
TIMING DIAGRAMS



1.Διάγραμμα χρονισμού DAC7611

4.1. ΒΙΒΛΙΟΘΗΚΕΣ ΥΠΟΣΤΗΡΗΞΕΙΣ

Η πρώτη και βασικότερη βιβλιοθήκη είναι η «**ftd2xx.h**» όπου αντιστοιχεί στις εντολές του UM245R. Στην συνέχεια προσθέτουμε την `<windows.h>` για πραγματοποιήσουμε Windows API και την `<math.h>` για μαθηματικές πράξεις.

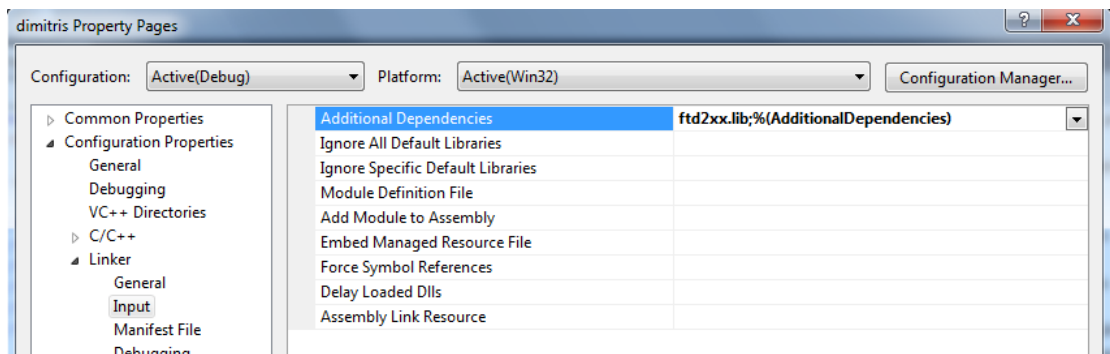


```

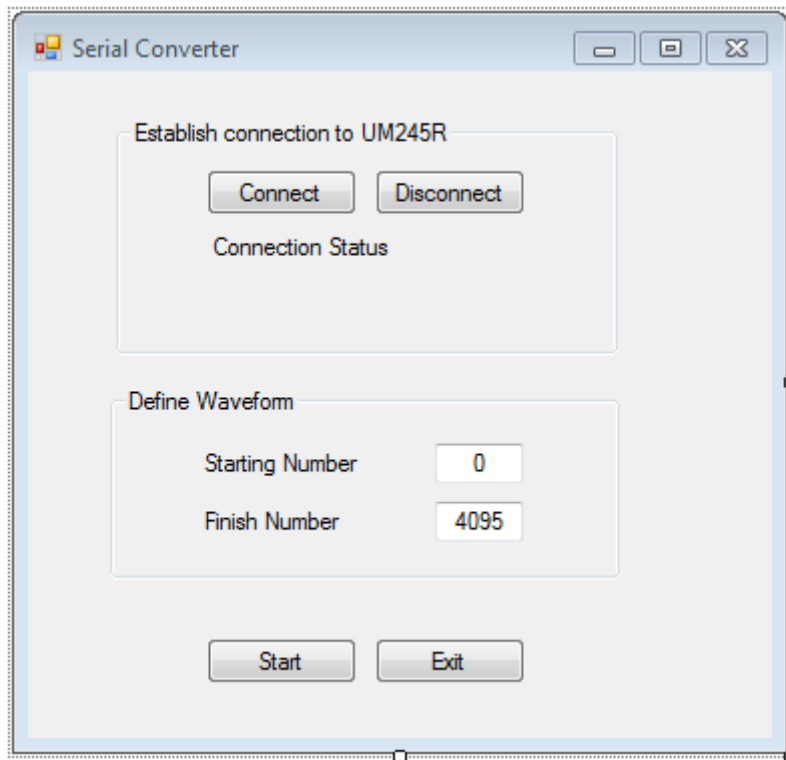
dimitris.cpp  Form1.h  Form1.h [Design]
(Global Scope)  main(array<System::String^> ^ args)
dimitris.cpp : main project file.

#include "stdafx.h"
#include <math.h>
#include <windows.h>
#include "ftd2xx.h"
#include "Form1.h"
  
```

Στην συνέχεια ακολουθούμε το «μονοπάτι» project<properties<Linker<input<additional Dependencies και προσθέτουμε το ftd2xx.lib



4.2 FORM DESING



Αυτή είναι η η Form Desing της εφαρμογής. Στο πλαίσιο «Establish connection to UM245R» χρειαστήκαμε 2 button για να απεικονίσουμε την σύνδεση ή μη του UM245R και 2 label ώστε να απεικονίζεται αν συνδέθηκε ή όχι το module. Μετά στο πλαίσιο «Define Waveform» προσθέσαμε δυο textbox για να ορίσουμε την αρχική και την τελική τιμή του τριγωνικού σήματος μας που θα στείλουμε στον DAC. Τέλος ένα button για “Start” για να αρχίζει να στέλνει τα bit και ένα button για “Exit” για να τερματίζουμε την εφαρμογή.

4.3.ΚΩΔΙΚΑΣ

```
#pragma once
#include <time.h> // θα μας χρειαστεί στη συνέχεια

namespace dimitris {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    FT_HANDLE ftHandle;
    FT_STATUS ftStatus;
    DWORD BytesWritten;

    int bitArray[12]; //
```

είναι μεταβλητές που απαιτούνται να γραφτούν στο πρόγραμμα και ο πίνακας bitArray []

```
/// <summary>
/// Summary for Form1
/// </summary>
public ref class Form1 : public System::Windows::Forms::Form
{
public:
    Form1(void)
    {
        InitializeComponent();
        //
        //TODO: Add the constructor code here
        //
    }

protected:
    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    ~Form1()
    {
        if (components)
        {
            delete components;
        }
    }
}
```

```

private: System::Windows::Forms::Button^ ConnectBtn;
private: System::Windows::Forms::Label^ ConnectLbl;
private: System::Windows::Forms::Button^ StartBtn;

private: System::Windows::Forms::TextBox^ textBox1;
private: System::Windows::Forms::TextBox^ textBox2;
private: System::Windows::Forms::Label^ StartLbl;
private: System::Windows::Forms::Label^ FinishLbl;
private: System::Windows::Forms::GroupBox^ groupBox1;
private: System::Windows::Forms::Button^ ExitBtn;
private: System::Windows::Forms::Button^ DisconnectBtn;
private: System::Windows::Forms::Label^ BitModeLbl;
private: System::Windows::Forms::Label^ BaudRateLbl;
private: System::Windows::Forms::GroupBox^ groupBox2;
protected:

private:
    /// <summary>
    /// Required designer variable.
    /// </summary>
    System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    void InitializeComponent(void)
    {
        this->ConnectBtn = (gcnew System::Windows::Forms::Button());
        this->ConnectLbl = (gcnew System::Windows::Forms::Label());
        this->StartBtn = (gcnew System::Windows::Forms::Button());
        this->textBox1 = (gcnew System::Windows::Forms::TextBox());
        this->textBox2 = (gcnew System::Windows::Forms::TextBox());
        this->StartLbl = (gcnew System::Windows::Forms::Label());
        this->FinishLbl = (gcnew System::Windows::Forms::Label());
        this->groupBox1=(gcnewSystem::Windows::Forms::GroupBox());
        this->ExitBtn = (gcnew System::Windows::Forms::Button());
        this->DisconnectBtn=(gcnew System::Windows::Forms::Button());
        this->BitModeLbl = (gcnew System::Windows::Forms::Label());
        this->BaudRateLbl = (gcnew System::Windows::Forms::Label());
        this->groupBox2 = (gcnew System::Windows::Forms::GroupBox());
        this->groupBox1->SuspendLayout();
        this->groupBox2->SuspendLayout();
        this->SuspendLayout();
        //
        // ConnectBtn
        //
        this->ConnectBtn->Location = System::Drawing::Point(45, 25);
        this->ConnectBtn->Name = L"ConnectBtn";
        this->ConnectBtn->Size = System::Drawing::Size(75, 23);
        this->ConnectBtn->TabIndex = 0;
        this->ConnectBtn->Text = L"Connect";
        this->ConnectBtn->UseVisualStyleBackColor = true;
        this->ConnectBtn->Click +=
gcnewSystem::EventHandler(this,&Form1::ConnectBtn_Click);
        //
        // ConnectLbl

```

```

        //
        this->ConnectLbl->AutoSize = true;
        this->ConnectLbl->Location = System::Drawing::Point(46, 57);
        this->ConnectLbl->Name = L"ConnectLbl";
        this->ConnectLbl->Size = System::Drawing::Size(94, 13);
        this->ConnectLbl->TabIndex = 1;
        this->ConnectLbl->Text = L"Connection Status";
        this->ConnectLbl->Click += gcnew System::EventHandler(this,
&Form1::label1_Click);
        //
        // StartBtn
        //
        this->StartBtn->Location = System::Drawing::Point(89, 283);
        this->StartBtn->Name = L"StartBtn";
        this->StartBtn->Size = System::Drawing::Size(75, 23);
        this->StartBtn->TabIndex = 2;
        this->StartBtn->Text = L"Start";
        this->StartBtn->UseVisualStyleBackColor = true;
        this->StartBtn->Click += gcnew System::EventHandler(this,
&Form1::StartBtn_Click);
        //
        // textBox1
        //
        this->textBox1->Location = System::Drawing::Point(162, 28);
        this->textBox1->Name = L"textBox1";
        this->textBox1->Size = System::Drawing::Size(44, 20);
        this->textBox1->TabIndex = 4;
        this->textBox1->Text = L"0";
        this->textBox1->TextAlign =
System::Windows::Forms::HorizontalAlignment::Center;
        this->textBox1->TextChanged += gcnew System::EventHandler(this,
&Form1::textBox1_TextChanged);
        //
        // textBox2
        //
        this->textBox2->Location = System::Drawing::Point(162, 57);
        this->textBox2->Name = L"textBox2";
        this->textBox2->Size = System::Drawing::Size(44, 20);
        this->textBox2->TabIndex = 5;
        this->textBox2->Text = L"4095";
        this->textBox2->TextAlign =
System::Windows::Forms::HorizontalAlignment::Center;
        this->textBox2->TextChanged += gcnew System::EventHandler(this,
&Form1::textBox2_TextChanged);
        //
        // StartLbl
        //
        this->StartLbl->AutoSize = true;
        this->StartLbl->Location = System::Drawing::Point(44, 31);
        this->StartLbl->Name = L"StartLbl";
        this->StartLbl->Size = System::Drawing::Size(83, 13);
        this->StartLbl->TabIndex = 6;
        this->StartLbl->Text = L"Starting Number";
        //
        // FinishLbl
        //
        this->FinishLbl->AutoSize = true;
        this->FinishLbl->Location = System::Drawing::Point(44, 60)
        this->FinishLbl->Name = L"FinishLbl";
        this->FinishLbl->Size = System::Drawing::Size(74, 13);
        this->FinishLbl->TabIndex = 7;
        this->FinishLbl->Text = L"Finish Number";

```

```

        //
        // groupBox1
        //
this->groupBox1->Controls->Add(this->FinishLbl);
this->groupBox1->Controls->Add(this->StartLbl);
this->groupBox1->Controls->Add(this->textBox2);
this->groupBox1->Controls->Add(this->textBox1);
this->groupBox1->Location = System::Drawing::Point(41, 158);
this->groupBox1->Name = L"groupBox1";
this->groupBox1->Size = System::Drawing::Size(254, 96);
this->groupBox1->TabIndex = 8;
this->groupBox1->TabStop = false;
this->groupBox1->Text = L"Define Waveform";
        //
        // ExitBtn
        //
this->ExitBtn->Location = System::Drawing::Point(173, 283)
this->ExitBtn->Name = L"ExitBtn";
this->ExitBtn->Size = System::Drawing::Size(75, 23);
this->ExitBtn->TabIndex = 9;
this->ExitBtn->Text = L"Exit";
this->ExitBtn->UseVisualStyleBackColor = true;
this->ExitBtn->Click += gcnew System::EventHandler(this,
&Form1::ExitBtn_Click);
        //
        // DisconnectBtn
        //
this->DisconnectBtn->Location = System::Drawing::Point(129, 25);
this->DisconnectBtn->Name = L"DisconnectBtn";
this->DisconnectBtn->Size = System::Drawing::Size(75, 23);
this->DisconnectBtn->TabIndex = 10;
this->DisconnectBtn->Text = L"Disconnect";
this->DisconnectBtn->UseVisualStyleBackColor = true;
this->DisconnectBtn->Click += gcnew System::EventHandler(this,
&Form1::DisconnectBtn_Click);
        //
        // BitModeLbl
        //
this->BitModeLbl->AutoSize = true;
this->BitModeLbl->Location = System::Drawing::Point(46, 75);
this->BitModeLbl->Name = L"BitModeLbl";
this->BitModeLbl->Size = System::Drawing::Size(0, 13);
this->BitModeLbl->TabIndex = 11;
        //
        // BaudRateLbl
        //
this->BaudRateLbl->AutoSize = true;
this->BaudRateLbl->Location = System::Drawing::Point(46, 92)
this->BaudRateLbl->Name = L"BaudRateLbl";
this->BaudRateLbl->Size = System::Drawing::Size(0, 13);
this->BaudRateLbl->TabIndex = 12;
        //
        // groupBox2
        //
this->groupBox2->Controls->Add(this->BaudRateLbl);
this->groupBox2->Controls->Add(this->BitModeLbl);
this->groupBox2->Controls->Add(this->DisconnectBtn);
this->groupBox2->Controls->Add(this->ConnectLbl);
this->groupBox2->Controls->Add(this->ConnectBtn);
this->groupBox2->Location = System::Drawing::Point(44, 24);
this->groupBox2->Name = L"groupBox2";
this->groupBox2->Size = System::Drawing::Size(250, 118);

```

```

this->groupBox2->TabIndex = 13;
this->groupBox2->TabStop = false;
this->groupBox2->Text = L"Establish connection to UM245R";
    //
    // Form1
    //
this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
this->ClientSize = System::Drawing::Size(371, 333);
this->Controls->Add(this->groupBox2);
this->Controls->Add(this->ExitBtn);
this->Controls->Add(this->groupBox1);
this->Controls->Add(this->StartBtn);
this->Name = L"Form1";
this->Text = L"Serial Converter - Δημήτρης Καλαμάρης";
this->Load += gcnew System::EventHandler(this,
&Form1::Form1_Load);
this->groupBox1->ResumeLayout(false);
this->groupBox1->PerformLayout();
this->groupBox2->ResumeLayout(false);
this->groupBox2->PerformLayout();
this->ResumeLayout(false);
}
#pragma endregion

//Οι μεταβλητές μου
private:
    int temp;
    int temp2;
    int StartNumber;
    int FinishNumber;
    int bitToSend;

private: System::Void Form1_Load(System::Object^ sender,
System::EventArgs^ e) {
}
private: System::Void label1_Click(System::Object^ sender,
System::EventArgs^ e) {
}
private: System::Void ConnectBtn_Click(System::Object^ sender,
System::EventArgs^ e)
{

//Κάνουμε connect
ftStatus = FT_Open(0,&ftHandle);
if (ftStatus == FT_OK)
{this->ConnectLbl->Text = L"UM245R connected";}
else
{this->ConnectLbl->Text = L"UM245R not connected";}
}

```

Ορισμός συνάρτησης

FT_STATUS FT_Open (int *iDevice*, FT_HANDLE **ftHandle*)

Ορίσματα

iDevice Δείκτης της συσκευής για να ανοίξει. Δείκτης στο 0

ftHandle Δείκτης σε μια μεταβλητή τύπου FT_HANDLE όπου η λαβή θα πρέπει να αποθηκεύεται.

Αυτή η λαβή θα πρέπει να χρησιμοποιηθεί για την πρόσβαση της συσκευής.

```
//Θέτουμε το UM245r σε λειτουργία BitMode
ftStatus = FT_SetBitMode(ftHandle, 255, 1);

if (ftStatus == FT_OK)
{this->BitModeLbl->Text = L"BitMode set";}
else
{this->BitModeLbl->Text = L"BitMode failed to set";}
```

Ορισμός συνάρτησης

FT_STATUS FT_SetBitmode (FT_HANDLE *ftHandle*, UCHAR *ucMask*, UCHAR *ucMode*)

Η εντολή FT_SetBitMode ενεργοποιεί διαφορετικούς τρόπους χρήσης του driver. Το *m_DeviceHandle* είναι η λαβή,

Με την τιμή &H0 ρυθμίζουμε την *ucMask* η οποία ρυθμίζει ποιά bits είναι για είσοδο και ποια για έξοδο. Η τιμή 0 καθορίζει το αντίστοιχο pin σε είσοδο ενώ η τιμή 1 καθορίζει το αντίστοιχο pin σε έξοδο.

Με την τιμή &H1 ρυθμίζουμε το *ucMode* δηλαδή το είδος λειτουργίας του driver όπου 1 είναι η λειτουργία σε Asynchronous Bit Bang mode.

Οι παρακάτω τιμές δηλώνουν και την χρήση λειτουργίας του Driver, επίσης ο παρακάτω κατάλογος παρουσιάζει τις λειτουργίες χρήσης σε ποιούς driver υποστηρίζονται

0x0 = Reset

0x1 = Asynchronous Bit Bang

0x2 = MPSSE (FT2232, FT2232H, FT4232H and FT232H devices only)

0x4 = Synchronous Bit Bang (FT232R, FT245R, FT2232, FT2232H, FT4232H and FT232H devices only)

0x8 = MCU Host Bus Emulation Mode (FT2232, FT2232H, FT4232H and FT232H devices only)

0x10 = Fast Opto-Isolated Serial Mode (FT2232, FT2232H, FT4232H and FT232H devices only)

0x20 = CBUS Bit Bang Mode (FT232R and FT232H devices only)

0x40 = Single Channel Synchronous 245 FIFO Mode (FT2232H and FT232H devices only)


```

//Θέτουμε το BaudRate
ftStatus = FT_SetBaudRate(ftHandle, 9600);

if (ftStatus == FT_OK)
{this->BaudRateLbl->Text = L"Baudrate set";}
else
{this->BaudRateLbl->Text = L"BaudRate failed to set";}
//Βάζουμε τους ακροδέκτες του UM245R στο Low
}

```

Αυτή η εντολή λειτουργίας ρυθμίζει την ταχύτητα επικοινωνίας (baudrate) μεταξύ της θύρας και της συσκευής η συγκεκριμένη ταχύτητα είναι η ταχύτητα όπως δίνεται απο το αρχείο D2XX_Programmer's_Guide(FT_000071).

```

private: System::Void textBox1_TextChanged(System::Object^ sender,
System::EventArgs^ e)
{
//Καθορίζουμε τις τιμές που μπορεί να πάρει το textBox1 (0-4095)
if (Int32::TryParse(textBox1->Text, temp))
StartNumber = Convert::ToInt32(textBox1->Text);
else
textBox1->Text = Convert::ToString(StartNumber);

if (StartNumber<0 || StartNumber>4095)
{StartNumber=0;
textBox1->Text = "0";}
}

private: System::Void textBox2_TextChanged(System::Object^ sender,
System::EventArgs^ e)
{
//Καθορίζουμε τις τιμές που μπορεί να πάρει το textBox2 (0-4095)
if (Int32::TryParse(textBox1->Text, temp))
FinishNumber = Convert::ToInt32(textBox2->Text);
else
textBox2->Text = Convert::ToString(FinishNumber);

if (FinishNumber<0 || FinishNumber>4095)
{FinishNumber=4095;
textBox2->Text = "4095";}
}

private: System::Void ExitBtn_Click(System::Object^ sender, System::EventArgs^
e) {
// Κουμπί Exit
Application::Exit();
}

```

```

private: System::Void StartBtn_Click(System::Object^ sender,
System::EventArgs^ e) {
//Όταν πατηθεί το κουμπί Start ξεκινάει η όλη διαδικασία

StartNumber = Convert::ToInt32(this->textBox1->Text);
FinishNumber = Convert::ToInt32(this->textBox2->Text);

if (StartNumber <= FinishNumber)
{

/* Για την άνοδο των αριθμών */

for (int i=StartNumber; i< (FinishNumber+1); i++) //Για κάθε αριθμό στο
διάστημα [StartNumber:FinishNumber]
{
/*Μετατροπή του τρέχοντα αριθμού i από δεκαδικό σε δυαδικό και τοποθέτηση των
δυαδικών του ψηφίων στον πίνακα bitArray[*]/

temp2 = i;

for (int y=0;y<12;y++)
{
bitArray[y] = 0;

if (temp2 >0)
{
bitArray[y] = temp2 %2;

temp2 = temp2 /2;
}
}
}
}

```

```

/* Αποστολή των bit του bitArray στο UM245R με τις κατάλληλες εντολές*/

for (int y=11;y>-1;y--) //Αποστέλλουμε από το μεγαλύτερης σημασίας bit προς
το χαμηλότερης
{
bitToSend = bitArray[y];

//Κάνουμε το CLK και το SDI στο Low και το LD high (στέλνουμε το byte 00000100
= 4)

unsigned int var1 = 4;

ftStatus = FT_Write(ftHandle, &var1, 1, &BytesWritten);

//Δίνουμε στο SDI την τιμή που πρέπει να πάρει σύμφωνα με το bitArray
if (bitToSend == 1)
{
var1 = 6;
ftStatus = FT_Write(ftHandle, &var1, 1, &BytesWritten);
}

Sleep(50);

//Κάνουμε το CLK High

var1 = var1 ^ 1;
ftStatus = FT_Write(ftHandle, &var1, 1, &BytesWritten);

Sleep(50);

/*
//Κάνουμε το CLK Low
var1 = var1 ^ 1;
ftStatus = FT_Write(ftHandle, &var1, 1, &BytesWritten);
*/

}
}

/* Για την κάθοδο των αριθμών */
for (int i=FinishNumber-1; i> (StartNumber-1); i--) //Για κάθε αριθμό στο
διάστημα [StartNumber:FinishNumber]
{
/*Μετατροπή του τρέχοντα αριθμού i από δεκαδικό σε δυαδικό και τοποθέτηση των
δυαδικών του
ψηφίων στον πίνακα bitArray[]*/

temp2 = i;

```

```

for (int y=0;y<12;y++)
{
bitArray[y] = 0;

if (temp2 >0)
{
bitArray[y] = temp2 %2;

temp2 = temp2 /2;
}

}

/* Αποστολή των bit του bitArray στο UM245R με τις κατάλληλες εντολές*/

for (int y=11;y>-1;y--) //Αποστέλλουμε από το μεγαλύτερης σημασίας bit προς
το χαμηλότερης
{
bitToSend = bitArray[y];

//Κάνουμε το CLK και το SDI στο Low (στέλνουμε το byte 0000100 = 4)
unsigned int var1 = 4;

ftStatus = FT_Write(ftHandle, &var1, 1, &BytesWritten);

//Δίνουμε στο SDI την τιμή που πρέπει να πάρει σύμφωνα με το bitArray
if (bitToSend == 1)
{
var1 = 6;
ftStatus = FT_Write(ftHandle, &var1, 1, &BytesWritten);
}

Sleep(50);

//Κάνουμε το CLK High

var1 = var1 ^ 1;
ftStatus = FT_Write(ftHandle, &var1, 1, &BytesWritten);

Sleep(50);

/*
//Κάνουμε το CLK Low
var1 = var1 ^ 1;
ftStatus = FT_Write(ftHandle, &var1, 1, &BytesWritten);
*/

}

}

} //End if

else{Application::Exit();}

```

```
}  
private: System::Void DisconnectBtn_Click(System::Object^ sender,  
System::EventArgs^ e) {  
  
    int var2 = 0;  
  
    ftStatus = FT_Write(ftHandle, &var2, 1, &BytesWritten);  
  
    FT_Close(ftHandle);  
    this->ConnectLbl->Text = L"UM245R disconnected";  
    this->BitModeLbl->Text = L"";  
    this->BaudRateLbl->Text = L"";  
}  
  
};  
}
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] www.ftdichip.com

[2] datasheet DAC7611

[3] “Elements of industrial Automation” at ELDE

[4] Industrial Ethernet: A Control Engineer’s Guide of Cisco

[5] www.teiser.gr/icd/staff/kalomiros/Syllogi