ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΧΟΛΗ Σ.Τ.ΕΦ.

ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ

<<ΠΑΡΑΓΩΓΗ ΑΝΑΛΟΓΙΚΩΝ ΣΗΜΑΤΩΝ ΜΕΣΩ ΘΥΡΑΣ USB>>

Πτυχιακή Εργασία:Κατσικίνης Αντώνιος (Α.Μ. 07/2307)

ΟΛΟ ΤΟ ΠΡΩΤΟ ΜΕΡΟΣ ΤΗΣ ΠΤΥΧΙΑΚΗΣ , ΕΓΙΝΕ ΣΕ ΣΥΝΕΡΓΑΣΙΑ ΜΕ ΤΟΝ ΚΑΛΑΜΑΡΗ ΔΗΜΗΤΡΙΟ.

Επίβλεψη:Κιζήρογλου Μιχαήλ

Θεσσαλονίκη Νοέμβριος 2014

Αφιερωμένη στην οικογένεια μου, που με στήριξαν καθόλη

την διάρκεια των σπουδών μου.

Επίσης θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή μου κ. Κιζήρογλου 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 και γνώσεις που λάβαμε από τις σπουδές στο Τμήμα Αυτοματισμού .

ΠΡΩΤΟ ΜΕΡΟΣ

Χρειάστηκε να προμηθευτούμε ένα moduleμε ενσωματωμένη θύρα USBτο UM245Rκαι ένα DAC7611. Πρώτος στόχος μας ήταν να μάθουμε και να κατανοήσουμε την λειτουργιά του UM245R, όπου η πτυχιακή του συναδέλφου Φιλίππου Ονησιφόρου μας βοήθησε αρκετά. Στη συνεχεία ,γράψαμε τον κώδικα στο περιβάλλον τις VisualC++. Στον προγραμματισμό του UM25R χρησιμοποιήσαμε της βιβλιοθήκες του ώστε να μπορέσουμε να συνδεθεί με το λειτουργικό των Windows 7. Δημιουργήσαμε ένα τριγωνικό σήμα μέχρι τα 12bit. Από τις 8 εξόδους του UM245R χρησιμοποιήσαμε τις 3 πρώτες ώστε η μια να 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.

ΔΕΥΤΕΡΟ ΜΕΡΟΣ

Αφού προμηθεύτηκα την πλακέτα MSP430G2553 της TEXAS INSTUMENTS προσπάθησα να επαναλάβω το παραπάνω project. Το περιβάλλον εργασίας που χρησιμοποιήθηκε είναι το Code Composer Studio. Το επιθυμητό αποτέλεσμα το παίρνουμε πάλι απ’ τον DAC7611p αφού του στείλουμε τις επιθυμητές πληροφορίες απ’ τον MSP.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Κεφάλαιο 1: ΘΕΩΡΗΤΙΚΗ ΕΙΣΑΓΩΓΗ

Κεφάλαιο 2:UM245R-DAC7611

Κεφάλαιο 3: ΠΕΡΙΓΡΑΦΗ ΚΥΚΛΩΜΑΤΟΣ

Κεφάλαιο 4 :ΠΕΡΙΓΡΑΦΗ ΚΩΔΙΚΑ

Κεφάλαιο 5 : ΧΡΟΝΟΙ ΛΕΙΤΟΥΡΓΙΑΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

Κεφάλαιο 6: MSP430G2553. ΠΡΟΓΡΑΜΜΑ ΚΑΙ ΑΝΑΛΥΣΗ

Επίλογος

Βιβλιογραφία

1. ΘΕΩΡΗΤΙΚΗ ΕΙΣΑΓΩΓΗ

Βασικές τεχνολογίες εισόδου/εξόδου

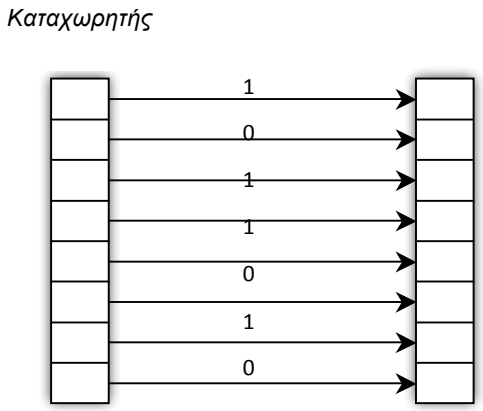
Τα κυκλώματα εισόδου/εξόδου (I/O) των υπολογιστικών συστημάτων μάς εξασφαλίζουν τη δυνατότητα της επικοινωνίας με άλλους υπολογιστές και περιφερειακές συσκευές. Ο έλεγχος εξωτερικών οργάνων, καθώς και η εισαγωγή δεδομένων από το περιβάλλον προς το υπολογιστικό κύκλωμα, επιτυγχάνεται μέσω των θυρών επικοινωνίας, 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 συνδεθούν 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 byte 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 του byte εμφανίζονται ταυτόχρονα. Η εισαγωγή ή εξαγωγή του byte 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



Σχήμα 1.1Παράλληλη μεταφορά οκτώ bits[5]

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 (databus) εύρους 32 bits. Εντούτοις, η παράλληλη μεταφορά των δεδομένων προς εξωτερικές συσκευές γίνεται με ταυτόχρονη μετάδοση μόνον οκτώ bits κάθε φορά.

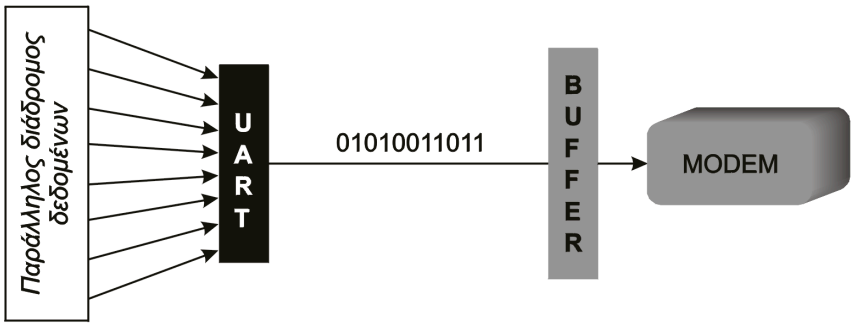
Τα πιο γνωστά πρωτόκολλα παράλληλης επικοινωνίας είναι το πρωτόκολλο CENTRONICS, για την τυπική παράλληλη θύρα ενός υπολογιστή

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

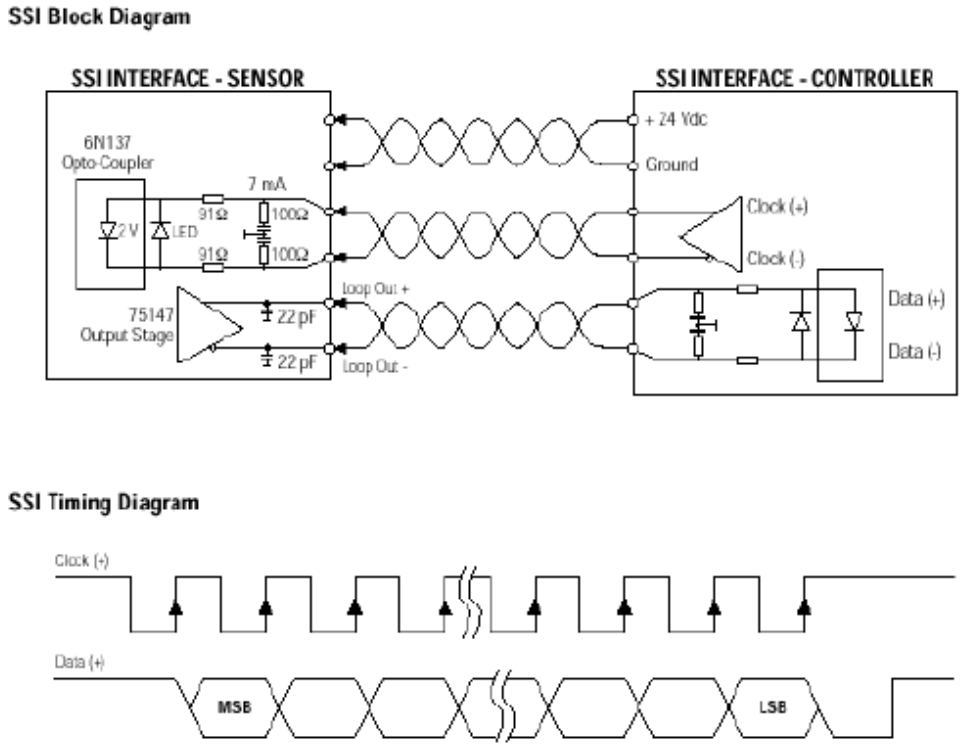


Σχήμα 1.2 Σειριακή μετάδοση πληροφορίας από τον υπολογιστή προς εξωτερική συσκευή[5]

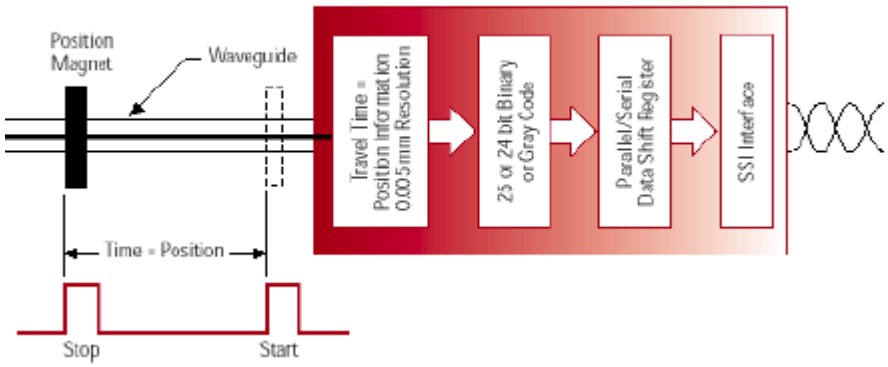
αφού λάβει κάποια δεδομένα και τα καταχωρήσει στα flip-flops που διαθέτει, ολισθαίνει τα bits της ψηφιολέξης που έχει καταχωρήσει ένα-ένα προς τα δεξιά ή προς τα αριστερά.

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 αναπτυχτεί 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 όποια τα καινούρια δεδομένα μεταφέρονται στον καταχωρητή. Τα δεδομένα μετατοπίζονται όταν ο αισθητήρας λάβει παλμό 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.



Σχημα1.3 Serial Synchronous Interface[3]



Σχήμα1.4 Serial Synchronous interface application[3]

ΘύραUSB

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 USB (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 περιττές 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 ιδανικοί 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% 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-paircables). Μετά από 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 αυξάνει 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.

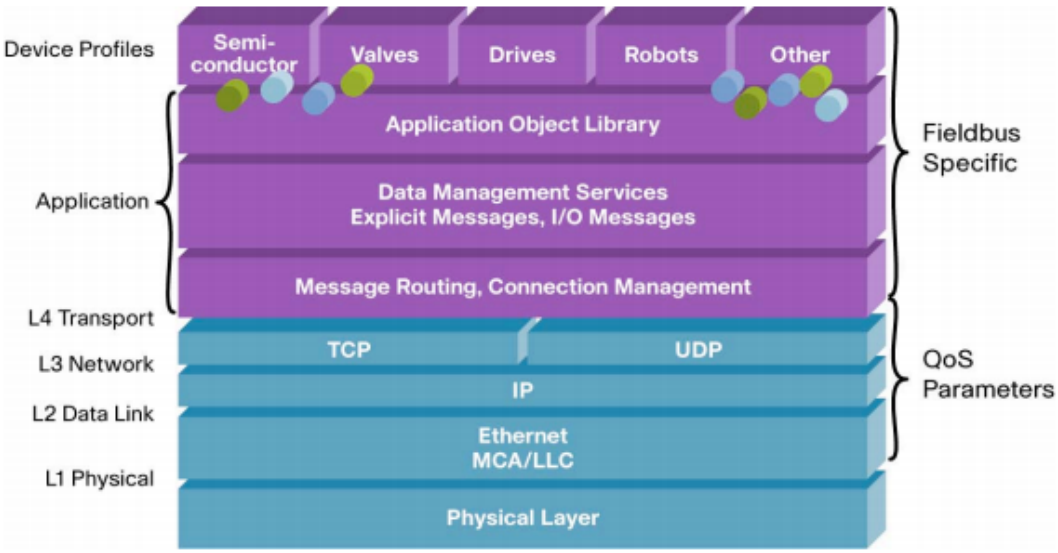
● Gigabit Ethernet επεκτείνει το πρωτόκολλο Ethernet ακόμη περισσότερο , αυξάνοντας την ταχύτητα δέκα φορές πάνω από Fast Ethernet σε 1000 Mbps ή 1 Gbps. Επειδή βασίζεται στο σημερινό πρότυπο Ethernet και είναι συμβατό με την εγκατεστημένη βάση του Ethernet και Fast Ethernet switches και routers , οι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



Σχήμα 1.4 Using intelligent Ethernet for Automation Control[4]

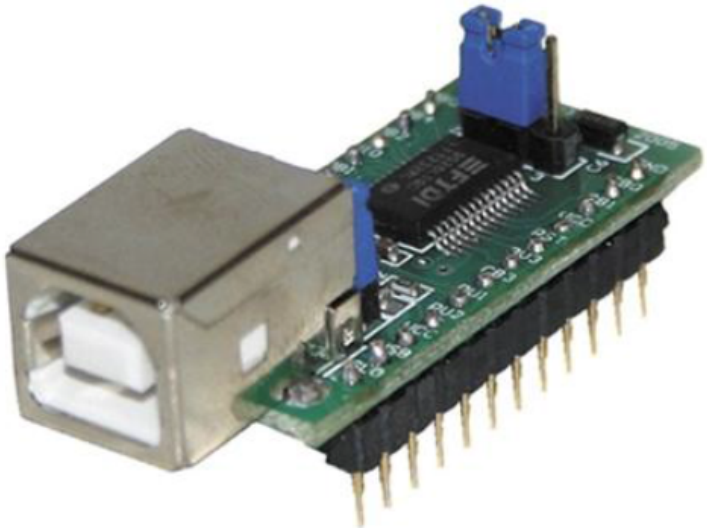
Για παράδειγμα, το πρωτόκολλο Common Industrial (CIP) έχει εφαρμογές που βασίζονται στο Ethernet και στο IP protocol suite (Ethernet / IP), στο Device Net και στο Control Net (μεταξύ άλλων). Οι περισσότεροι ελεγκτές (με τις κατάλληλες συνδέσεις δικτύου) μπορεί να μεταφέρουν δεδομένα από το ένα μέρος δικτύου 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 την καλύτερη

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

2.UM245R-DAC7611

2.1.UM245R

Το UM245R είναι ένα module της εταιρίας FTDI (Future Technology Devices International Ltd) το οποίο χρησιμοποίει το chip FT245R. Το FT245R είναι ένα USB με παράλληλο FIFO interface. Επιπλέον διαθέτει τις λειτουργίες σύγχρονο και ασύγχρονο bitbang interface. Από USB σε παράλληλο σχεδιάστηκε χρησιμοποιώντας το FT245R και 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 συνδέσει 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).



Σχήμα 2.1 UM245R[1]

2.1.1 The BitBang Mode - Asynchronous

Η Bitmode 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 bitmodeBang για την FT232R: ασύγχρονης BangBit, το οποίο είναι το

ίδιο με το FTDIBM και Cchip – style BangBit mode, με την προσθήκη της «ανάγνωσης» και «εγγραφής» στην περίπτωση του FT232R. Στη «Σύγχρονη BitBang» τα δεδομένα μόνο διαβάζονται. Αυτή είναι η ίδια λειτουργιά με την σύγχρονη λειτουργία του FT2232 BangBit. Η CBUS BitBang λειτουργία, μια νέα 4-bit έκδοση του Bit λειτουργία Bang είναι διαθέσιμα στο FT232R CBUS αλλά όχι για το FT245R.

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

Εμείς χρησιμοποιήσαμε στον προγραμματισμό του UM245r την ασύγχρονη λειτουργιά. Στην Ασύγχρονη λειτουργία Bit Bang είναι το ίδιο με BM-style λειτουργία BangBit. Οποιαδήποτε δεδομένα γράφονται στη συσκευή κατά τον συνήθη τρόπο θα αυτοσυχρονίζονται 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 Bitmode 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

Σχήμα 2.2 Πινάκας εισόδων/εξόδων[1]

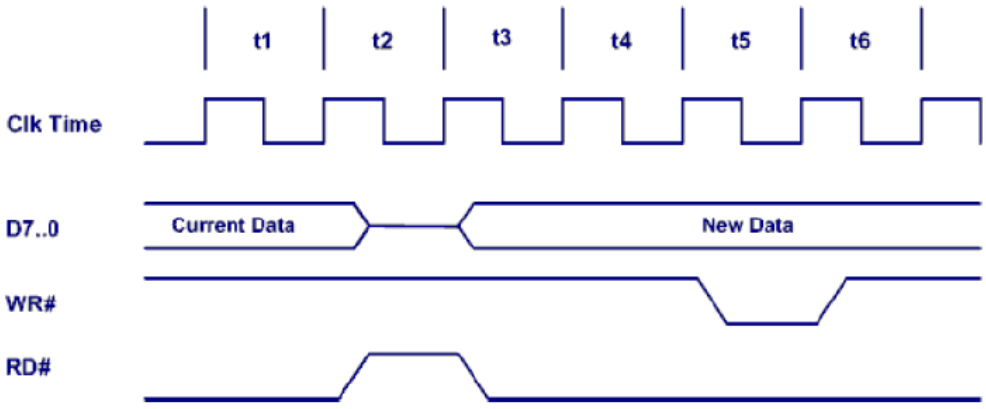
Μερικές D2XX εντολές του οδηγού είναι αναγκαίες για τη χρήση Ασύγχρονη Bang Bit. Οι πιο σημαντικές αναφέρονται παρακάτω:

2.1.2 TheBitBangMode-Synchronous

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 thedata 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 (9600x16) = 153600 bytes per second, or 1 every 6.5 µS.
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.

Σχήμα 2.3 Βασικές εντολές

Με την σύγχρονη λειτουργία BitBang, τα δεδομένα θα σταλούν μόνο αν υπάρχει χώρος στη συσκευή για τα δεδομένα που πρέπει να διαβαστούν από τα pins.Επίσης στην ασύγχρονη λειτουργιά θα διαβαστούν τα δεδομένα που βρίσκονται στα pins, προτούσταλεί 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.



Σχήμα 2.4 Το διάγραμμα χρονισμού [1]

Όταν η σύγχρονη Bang Bitmode 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

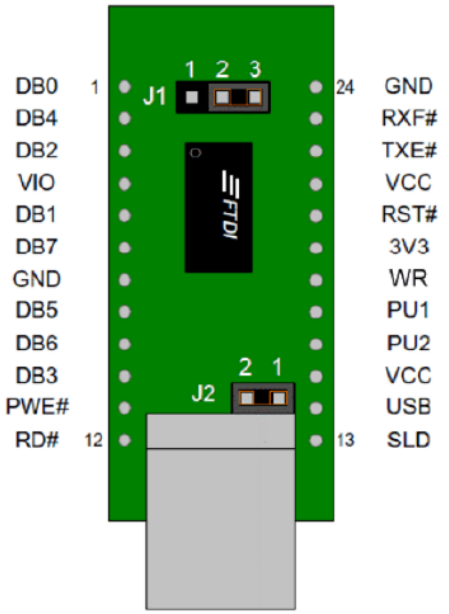
Σχήμα 2.5 Πινάκας εισόδων/εξόδων[1]

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 thedata 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 (9600x16) = 153600 bytes per second, or 1 every 6.5 µS.
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.

Σχήμα 2.6Βασικές εντολές[1]

2.1.3ΠεριγραφήUM245R



Σχήμα 2.7 Κάτοψη UM245R[1]

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. Fetched 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.*

Σχήμα 2.8[1]

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.

Σχήμα 2.9[1]

Διαγράμματα Διεπαφής ελέγχου χρονικών κύκλων

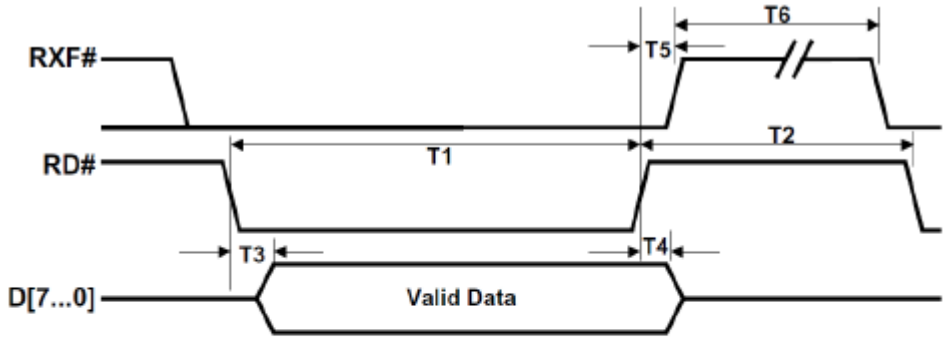


Figure 4.2 FIFO Read Cycle

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

Σχήμα 2.10 Διάγραμμα χρονισμού για «Διάβασμα»[1]

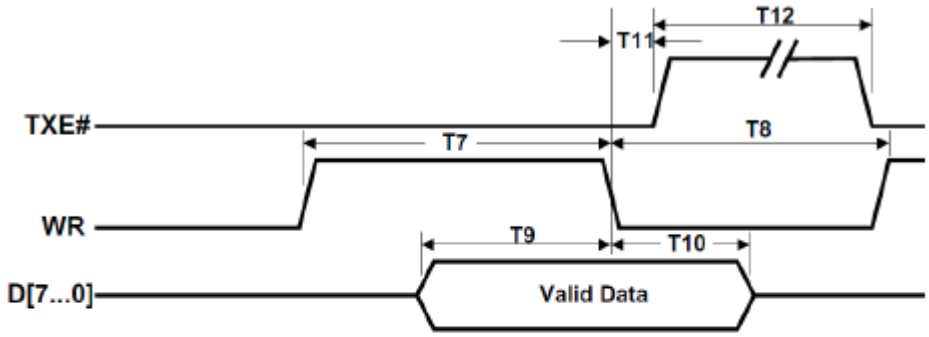


Figure 4.3 FIFO Write Cycle

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

Σχήμα 2.11 Διάγραμμα χρονισμού για «Εγγραφή»[1]

2.2 DAC7611

Περιγραφή DAC7611

Το DAC7611 είναι ένας 12-bit DtoA μετατροπέας (DAC). Απαιτεί 5V τροφοδοσία και περιέχει έναν shift καταχωρητή εισόδου, ένα latch, 2.435V τιμή αναφοράς, DAC, έναν ενισχυτή παράγωγης υψηλής ταχύτητας railtorail. 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 όπως απαιτείται από την εφαρμογή. Τέλος,το DAC7611 είναι διαθέσιμο σε μια 8-SOIC μολύβδου ή 8-pin πλαστική συσκευασία DIP .

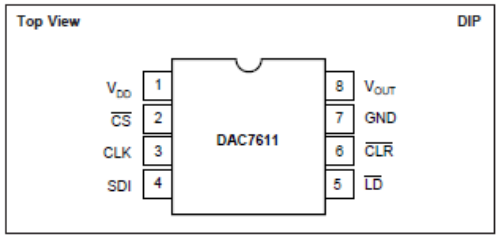
Χαρακτηριστικά

- 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 ΚΛΕΙΣΤΩΝ ΒΡΟΧΩΝ
- ΠΕΡΙΦΕΡΙΑΚΟ ΥΠΟΛΟΓΙΣΤΗ
- ΦΟΡΗΤΗ ΕΝΟΡΓΑΝΩΣΗ

ΔΙΑΜΟΡΦΩΣΗ ΑΚΡΟΔΕΚΤΩΝ



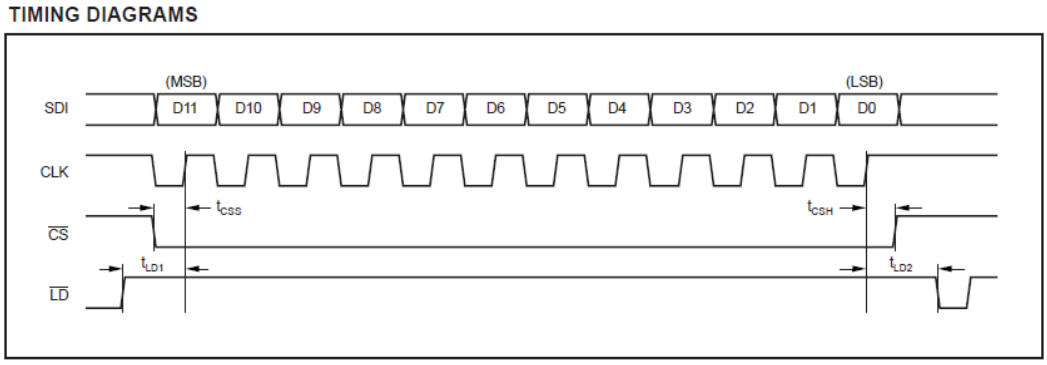
Σχήμα 2.12 Κάτοψη DAC7611[2]

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 strobbed 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).

Σχήμα 2.13 Περιγραφή ακροδεκτώνDAC7611[2]

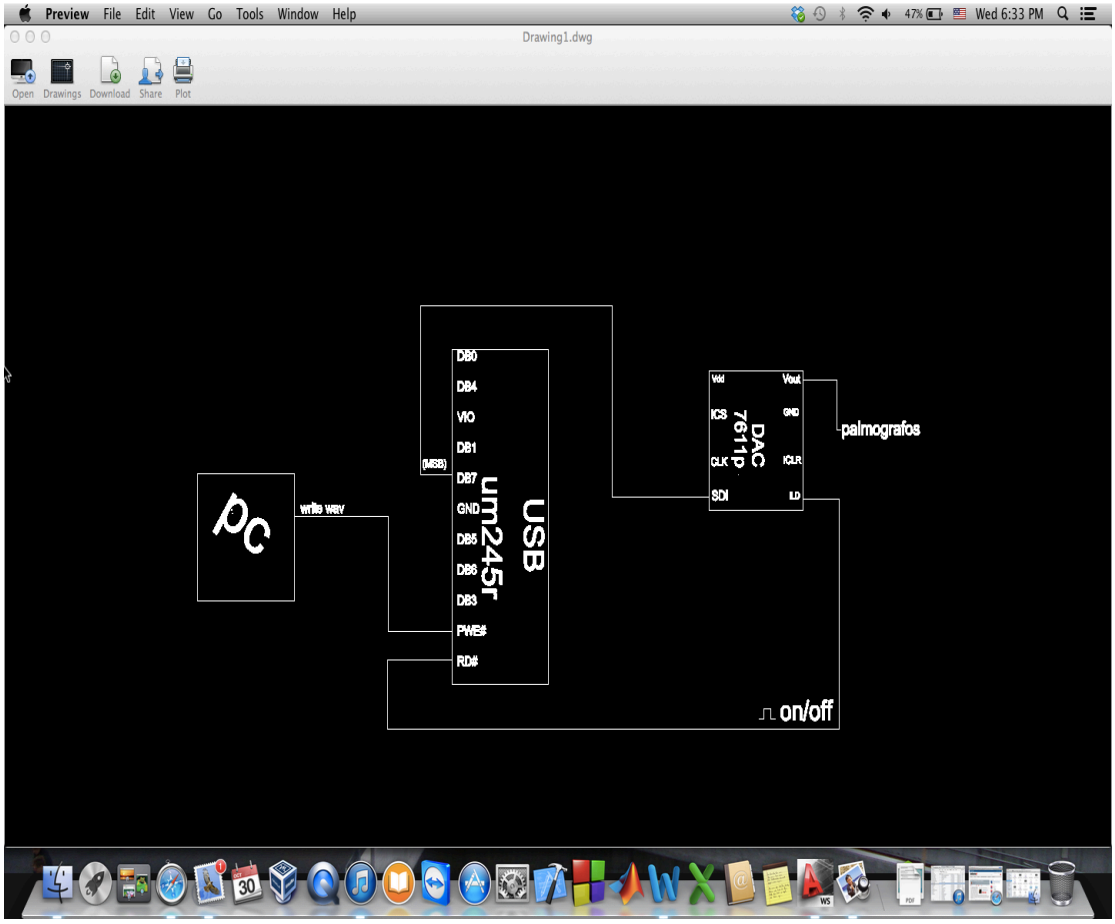
ΔΙΑΓΡΑΜΜΑΤΑ ΧΡΟΝΙΣΜΟΥ



Σχήμα 2.14 Διάγραμμα χρονισμού DAC7611[2]

3.ΠΕΡΙΓΡΑΦΗ ΚΥΚΛΩΜΑΤΟΣ

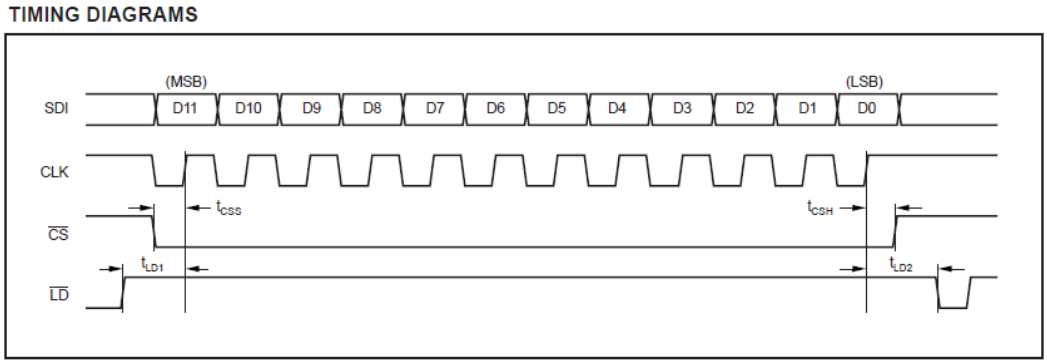
Το πρώτο μέρος του κυκλώματος μας είναι ο υπολογιστής όπου με μια θύρα USB το συνδέουμε με το UM245R. Από το UM245R χρησιμοποιήσαμε της πρώτες 3 εξόδους(D0,D1,D2). Από αυτές τις 3 εξόδους συνδέουμε τον DAC7611 στους ακροδέκτες D0->SDI, D1->CLK καιD2->LD. Από τον DAC από τον ακροδέκτη V_{out}στον παλμογράφο



Σχήμα 3.1. πλάνο εργασίας

4.ΠΕΡΙΓΡΑΦΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

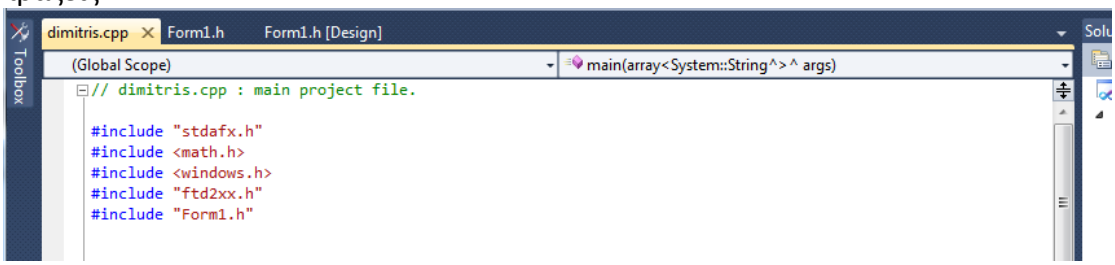
Ο κώδικας γράφτηκε στο περιβάλλον της VisualC++ . Χρησιμοποιήσαμε το Software Application Development D2XX Programmer's Guide που το προμηθευτήκαμε από το www.ftdichip.com. Η εταιρία χορήγει τους drivers και της βιβλιοθήκες που χρειάστηκαν. Αφού κάνουμε connect το UM245R με τον υπολογιστή καταφέραμε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



Σχήμα 4.1.Διάγραμμα χρονισμού DAC7611[2]

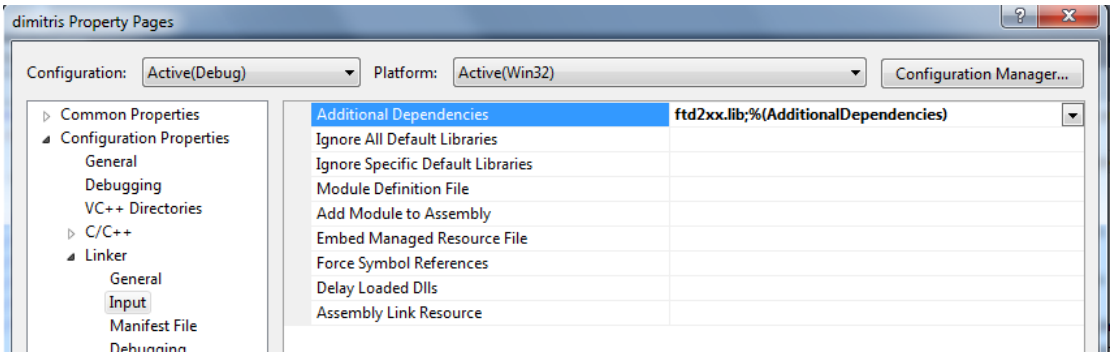
4.1. ΒΙΒΛΙΟΘΗΚΕΣ ΥΠΟΣΤΗΡΙΞΗΣ

Η πρώτη και βασικότερη βιβλιοθήκη είναι η «ftd2xx.h»οπου αντιστοιχεί στις εντολές του UM245R. Στην συνεχεία προσθέτουμε την <windows.h > για να πραγματοποιήσουμε WindowsAPI και την <math.h> για μαθηματικές πράξεις.



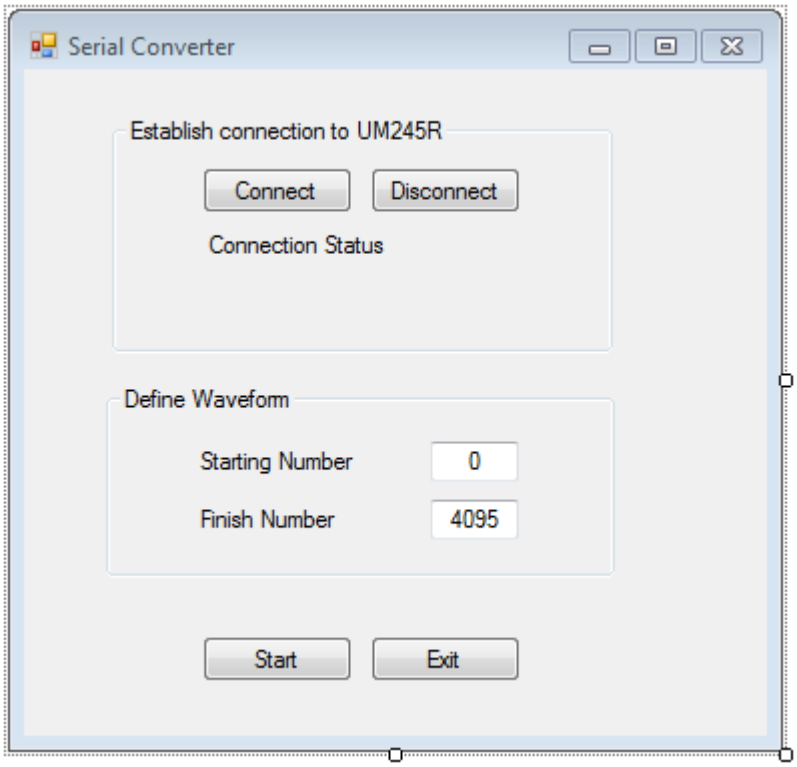
Σχήμα 4.2

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



Σχήμα 4.3

4.2 FORM DESIGN



Σχήμα 4.4 FormDesign

Αυτή είναι η Form Design 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 Wave form» 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

```
#pragmaonce
#include<time.h> // θα μας χρειαστεί στη συνεχεία

namespacedimitris {

	usingnamespace System;
	usingnamespace System::ComponentModel;
	usingnamespace System::Collections;
	usingnamespace System::Windows::Forms;
	usingnamespace System::Data;
	usingnamespace 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>
	publicrefclass 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;

#pragmaregion Windows Form Designer generated code
               /// <summary>
               /// Required method for Designer support - do not modify
               /// the contents of this method with the code editor.
               /// </summary>
               voidInitializeComponent(void)
               {
               this->ConnectBtn =(gcnewSystem::Windows::Forms::Button());
               this->ConnectLbl = (gcnewSystem::Windows::Forms::Label());
               this->StartBtn = (gcnew System::Windows::Forms::Button());
               this->textBox1 = (gcnewSystem::Windows::Forms::TextBox());
               this->textBox2 = (gcnewSystem::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=(gcnewSystem::Windows::Forms::Button());
        this->BitModeLbl = (gcnewSystem::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 - ΔημήτρηςΚαλαμάρης";
            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);

            }
#pragmaendregion

      //Οι μεταβλητές μου
      private:
            int temp;
            int temp2;
            intStartNumber;
            intFinishNumber;
            intbitToSend;




      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 λειτουργεια 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 ποια bits είναι για είσοδοι και ποια για έξοδοι. 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 από 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 (inti=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 Lowκαι το LDhigh (στέλνουμε το byte 00000100 =
4)

unsignedint 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 (inti=FinishNumber-1; i> (StartNumber-1); i--) //Γιακάθεαριθμόστοδιάστημα
[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 00000100 = 4)

unsignedint 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"";
}

};
}
```

5.ΧΡΟΝΟΙ ΛΕΙΤΟΥΡΓΙΑΣ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ

Αδυναμία επίλυσης στο λειτουργικό σύστημα το οποίο εξαρχής επιλέξαμε

Για την πραγματοποίηση της πτυχιακής μας εργασίας επιλέξαμε το περιβάλλον Microsoft Visual Studio 2010 και πιο συγκεκριμένα το Visual C++ 2010. Η εντολή Sleep(50) εκτελεί «καθυστέρηση» για 50 ms. Αυτός ο χρόνος μπορεί να μειωθεί ως και τα 4-10 ms (ανάλογα το σύστημα και την έκδοση του Microsoft Visual Studio που διαθέτουμε). Μικρότερο χρόνο στο περιβάλλον της Microsoft ήταν αδύνατο να πετύχουμε.

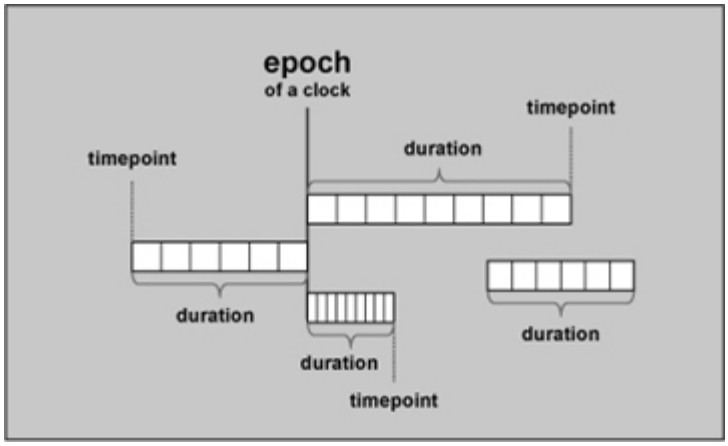
Τη λύση ήρθε να δώσει το περιβάλλον της Microsoft Visual 2013, καθώς εγκαταστάθηκαν σ' αυτές και πιο συγκεκριμένα στο compiler C++11 , η βιβλιοθήκη ChronoLibrary, η ThreadLibrary καθώς και άλλες πολλές βιβλιοθήκες.

Εφόσον επιλέγαμε τη χρησιμοποίηση περιβάλλοντος σε Unix/POSIX, τότε η δυνατότητα των nanosecondήταν και είναι διαθέσιμη, στην Microsoft όμως έπρεπε να έρθει το 2013 για να επιτευχθούν αυτοί οι χρόνοι.

ChronoLibrary

Η βιβλιοθήκη αυτή δημιουργήθηκε , ώστε να δίνει τη δυνατότητα χρησιμοποίησης timers και clocks που πιθανόν να διαφέρουν σε διαφορετικά συστήματα , όπως επίσης να δίνει τη δυνατότητα της βελτίωσης τους με την πάροδο του χρόνου.

Στη συγκεκριμένη βιβλιοθήκη 3 είναι οι χρήσιμοι παράγοντες : Duration , Timepoint και Clock.



Σχήμα 5.1. χρονικά διαστήματα , «εποχές» του clock

Duration (διάρκεια) είναι το γινόμενο του αριθμού που μετρήθηκε επί τον αριθμό των sec <a combination of a value representing the number of ticks and a fraction representing the unit in seconds>. Για παράδειγμα

```
std::chrono::duration<int>twentySeconds(20);
std::chrono::duration<double,std::ratio<60>>halfAMinute(0.5);
std::chrono::duration<long,std::ratio<1,1000>>oneMillisecond(1);
```

Η βιβλιοθήκη λοιπόν παρέχει τους ακόλουθους ορισμούς τύπου

```
namespace std
{
      namespacechrono
       {
            typedef   duration<signed   int-type   >=   64   bits,nano>
      nanoseconds;
            typedef   duration<signed   int-type   >=   55   bits,micro>
      microseconds;
            typedef   duration<signed   int-type   >=   45   bits,milli>
      milliseconds;
            typedef    duration<signed    int-type    >=    35    bits>
      seconds;
            typedef  duration<signed  int-type  >=  29  bits,ratio<60>>
      minutes;
            typedef duration<signed int-type >= 23 bits,ratio<3600>>
      hours;
      }
}
```

παρέχει όλες τις αριθμητικές πράξεις μεταξύ των durations…

Το CLOCK απ’ την άλλη είναι αυτό που καθορίζει μια epoch και μια περίοδο. Το interface του Clock είναι λοιπόν μια λειτουργία για την ακριβή απόδοση ενός αντικειμένου την τρέχουσα τιμή.

Το TIMEPOINT αντιπροσωπεύει ένα συγκεκριμένο σημείο στο χρόνο συνδέοντας το Duration στο Clock που μας δίνεται.

```
namespace std {
namespacechrono {
template<typename Clock,
typename Duration = typename Clock::duration>
classtime_point;
   }
}
```

Οι εντολές Sleep_for(), Sleep_until() , try_lock_for(), wait_for() είναι διαθέσιμες απ' την βιβλιοθήκη this_thread
Πχ this_thread::sleep_for(chrono::seconds(10));

Η οποία βιβλιοθήκη this_thread() είναι διαθέσιμη στους χρήστες απ' την έκδοση του compilerC++11 και μετά.

ΛΥΣΗ

Η λύση λοιπόν στο πρόβλημα που αντιμετωπίσαμε είναι η χρησιμοποίηση της έκδοσης MicrosoftVisualStudio 2013

Χρησιμοποιώντας τις εξής βιβλιοθήκες και εντολές

#include<chrono>

#include<thread>

…

std::this_thread::sleeo_for(std::chrono::nanoseconds(1));

Όπου στην περίπτωση του 1 nanosecond θα μπορούσαμε να είχαμε βάλει οποιαδήποτε τιμή (οποιοδήποτε χρόνο) από 1 nanosecondκαι πάνω. Η τιμή που θα τοποθετήσουμε μπορεί να είναι sleep περισσότερο από τη δεδομένη περίοδο.

Πχ αν τοποθετούσαμε τα 40 nanosecondυπάρχει μια περιοχή «λάθους» στα +/- 2 ns.

6. MSP430G2553 και DAC7611p
Η διαδικασία, το πρόγραμμα και η ανάλυση του.

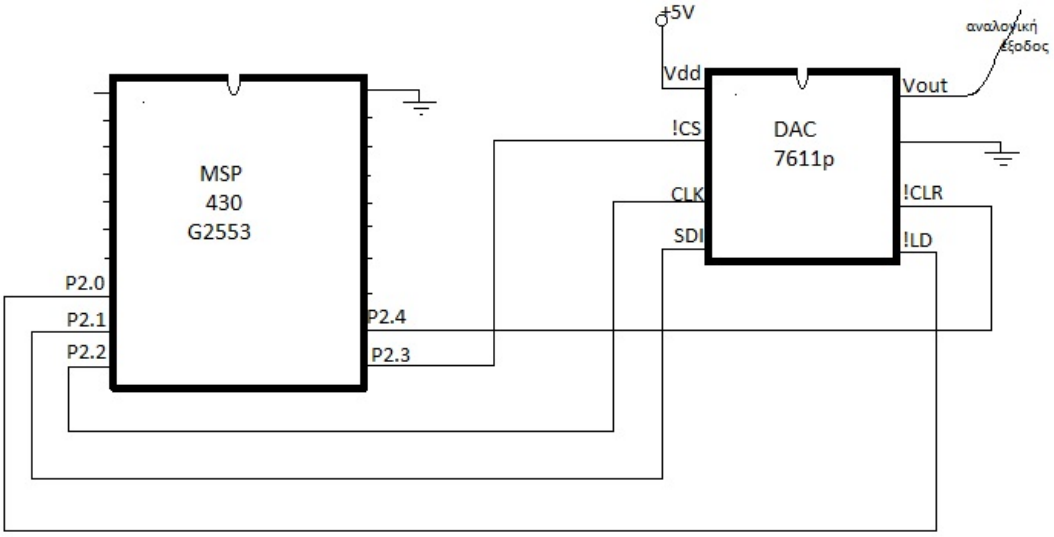
Γνωρίζαμε εξ αρχής ότι ο μικροελεγκτής MSP430G2553 έχει λειτουργία ADC10, ενώ εμείς ζητάμε DAC, να μετατρέψουμε το ψηφιακό σήμα που παίρνουμε μέσω USB απ' τον Η/Υ σε ένα αναλογικό σήμα. Επομένως τον εξ αρχής γνωρίζαμε ότι η είναι απαραίτητη η χρησιμοποίηση του DAC7611p (ο οποίος είχε χρησιμοποιηθεί και στον πρώτο τρόπο επίλυσης, με τον UMR).

Εφόσον μου δόθηκε ο MSP430 επέλεξα το περιβάλλον εργασίας (IDE – Integrated Development Environment) στο οποίο θα υλοποιούσα το project. Είχα τη δυνατότητα να επιλέξω ανάμεσα στον CodeComposerStudio CCS, στον IAR Embedded Workbench Kickstart και τον Open Source GCC for MSP. Επέλεξα το πρώτο μια και είναι το πιο διαδεδομένο και με μια πολύ καλή διαδικτυακή κοινότητα, την http://e2e.ti.com/support/microcontrollers/msp430.

Προτού χρησιμοποιήσω για πρώτη φορά το λειτουργικό CCS ενημερώθηκα απ' το κανάλι της εταιρίας στο youtube. Στην πορεία, για την εκμάθηση του λειτουργικού υπάρχουν πάνω από 80 παραδείγματα, τα οποία τα βρίσκεις στην εισαγωγή (TI Resource Explorer) του CCS και μπορείς να τα χρησιμοποιήσεις , να τα αναπαράγεις και να πειραματιστείς πάνω σε αυτά, μαθαίνοντας έτσι τις εντολές (commands) και το σκεπτικό με πολλά σχόλια σε κάθε παράδειγμα.

Έπειτα, διαβάζοντας το datasheet του DAC7611p, χρησιμοποίησα τα Timing Diagrams έτσι ώστε αντιληφθώ ποια θα πρέπει να είναι η πορεία – αντίστοιχα σε ποιο χρονικό διάστημα να ενεργοποιήσω το LD (load), το CS (cheap select), το CLK (clock), το CLR (clear), αλλά και ΠΟΤΕ πρέπει να περάσω το σήμα μου SDI (Serial Data Input).

Στην πορεία έπρεπε να επιλέξω πόσο είναι το χρονικό διάστημα στο οποίο θέλω να κάνει σε κάθε κατάσταση καθυστέρηση – delay, το αν θέλω τα δεδομένα μου να περνάν αριστερόστροφα ή δεξιόστροφα, πχ. Shift Left Data -> value=value>>1;
Καθώς και άλλες μικρές λεπτομέρειες.



Σχήμα 6.1. το πλάνο

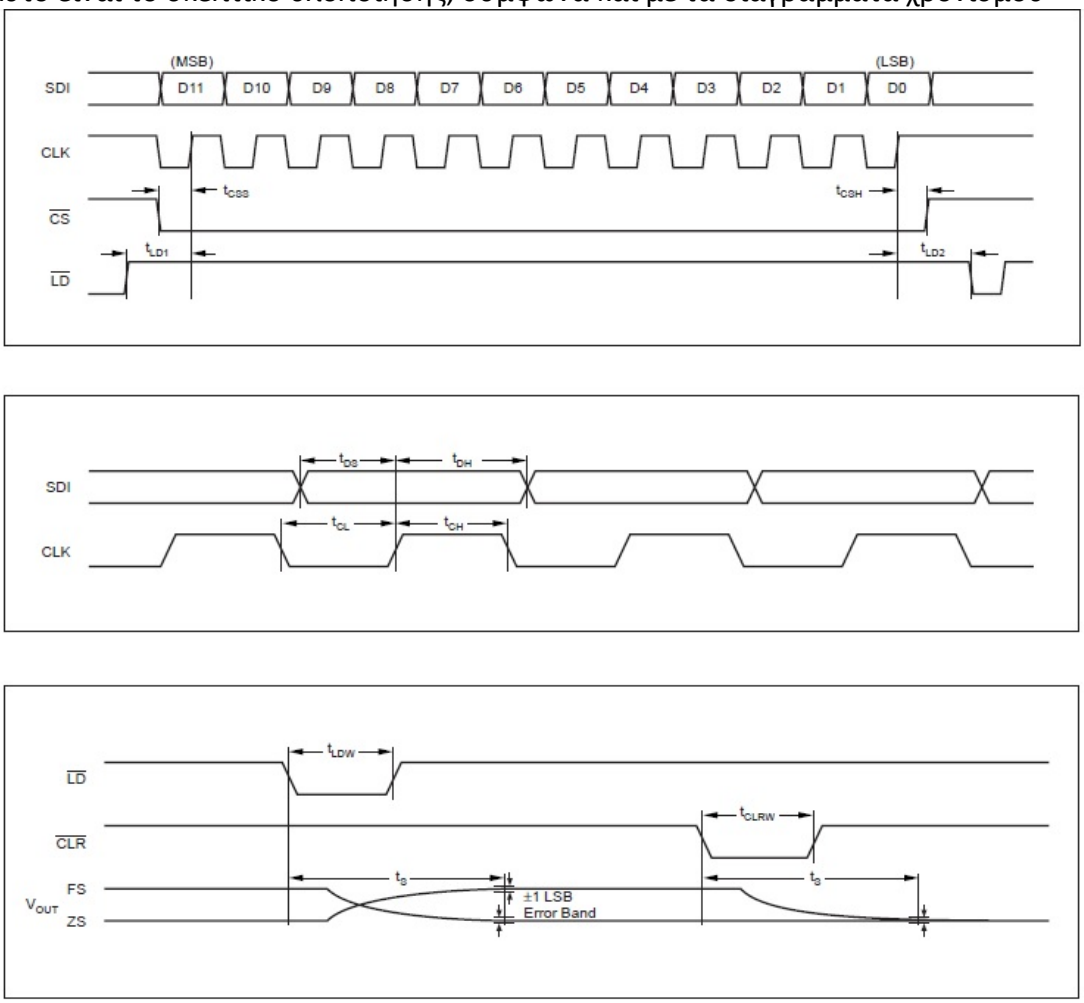
Ας δούμε τώρα το σκεπτικό υλοποίησης

```
CLR_HI;
CS_HI;
SDI_HI;
CLK_HI;
LD_LO;
Delay
LD_HI;
Delay
CS_LO;
Delay

Long int value= « Ηεπιθυμητή τιμή σεhexadecimal»
For (j=0 ; j<12 ; j++)
{
        CLK_LO;
        If ( LSB=1)  // or MSB
                SDI_HI;
        Else
                SDI_LO;
        Delay
        CLK_HI;
        Shift data -value- left
}
```

CLR_LO;
Delay
CLR_HI;

Αυτό είναι το σκεπτικό υλοποίησης, σύμφωνα και με τα διαγράμματα χρονισμού



Σχήμα 6.2. timing diagrams of DAC7611p

Το **πρόγραμμα** μου αυτούσιο (η τελική του μορφή)

```
#include <msp430.h>

#define CLR_LO (P2OUT&=~BIT4)
#define CLR_HI (P2OUT|=BIT4)

#define CS_LO (P2OUT&=~BIT3)
#define CS_HI (P2OUT|=BIT3)

#define SDI_LO (P2OUT&=~BIT1)
#define SDI_HI (P2OUT|=BIT1)

#define CLK_LO (P2OUT&=~BIT2)
#define CLK_HI (P2OUT|=BIT2)

#define LD_LO (P2OUT&=~BIT0)
#define LD_HI (P2OUT|=BIT0)
```

```
void SPI_send(short int data)
{
  int j;
  for(j=0;j<12;j++)
  {
    CLK_LO;
    if(data & 0x800)   // MSB
      SDI_HI;
    else
      SDI_LO;

    //__delay_cycles(0.01); // 1000 clock cycles delay = 1ms@1kHz
    CLK_HI;
    //__delay_cycles(0.01);
    data <<= 1;
  }
}

int main(void)
{

       WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer

       P1DIR = 0; //input Port1

       P2DIR = BIT0 | BIT1 | BIT2 | BIT3 | BIT4; //Out bit 0..4

       P2OUT = 0xff;

       CLR_HI;
       CS_HI;
       SDI_HI;
       CLK_HI;
       LD_LO;

       //__delay_cycles(0.05); // 1000 clock cycles delay = 1ms@1kHz

       LD_HI;

       //__delay_cycles(0.05); // 1000 clock cycles delay = 1ms@1kHz


       //SDI <-322 hex  τυχαία τιμή που επέλεξα για εκκίνηση
       short int data=0x322;
       short int anti;
       short int a=0xFFF;
       anti = a-data;
       CS_LO;
       int j;

       for(j=0; j<anti; j+=256)
       {
              CS_LO;
           //__delay_cycles(0.005); // 1000 clock cycles delay = 1ms@1kHz

              data = data + 0x100;
              SPI_send(data);
```

```
            CS_HI;

            //__delay_cycles(0.0005);
            LD_LO;
            //__delay_cycles(0.0005);
            LD_HI;
      }

      while(1)
      {
            for(j=0; j<a; j+=256)
            {
                  CS_LO;
                //__delay_cycles(0.005);

                  data = data - 0x100;
                  SPI_send(data);

                  CS_HI;

                  //__delay_cycles(0.0005);
                  LD_LO;
                  //__delay_cycles(0.0005);
                  LD_HI;
            }
            __delay_cycles(20000); // 1000 clock cycles delay = 1ms@1MHz


            for(j=0; j<a; j+=256)
            {
                  CS_LO;
                //__delay_cycles(0.005);

                  data = data + 0x100;
                  SPI_send(data);

                  CS_HI;

                  //__delay_cycles(0.0005);
                  LD_LO;
                  //__delay_cycles(0.0005);
                  LD_HI;
            }

      }

}
```

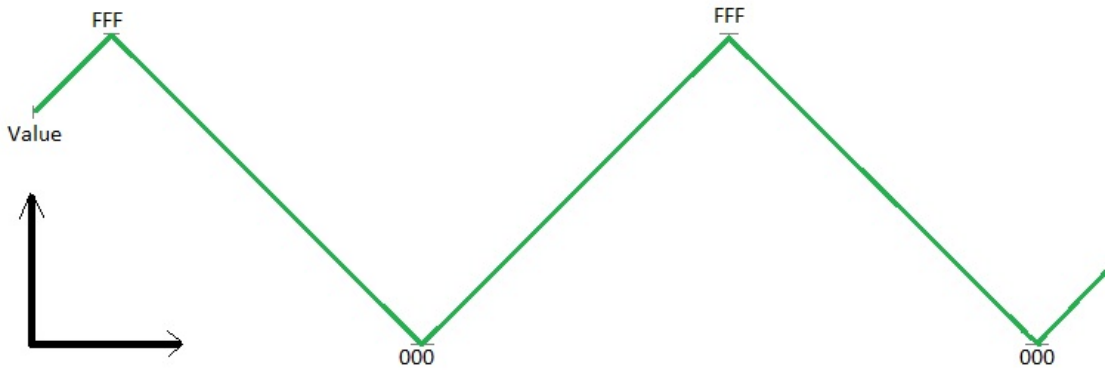
//έβαλα όλα τα delay – καθυστερήσεις σε σχόλιο επειδή ενώ αρχικά επιθυμούσα να έχω καθυστερήσεις και να είναι εμφανές στα led του κυκλώματος μου η λειτουργία του κυκλώματος, τα αφαίρεσα για να πετύχω καλύτερους χρόνους
Καθώς και να μπορέσω να δω την έξοδο του κυκλώματος στον παλμογράφο
//

Εδώ να εξηγήσω ότι ξεκίνησα το πρόγραμμα θέτοντας τις εξόδους του MSP430,
P2.0 -> LD
P2.1 -> SDI

P2.2 -> CLK
P2.3 ->CS
P2.4 ->CLR

Έτσι ώστε στη διάρκεια του προγραμματισμού του μικροελεγκτή μου να είναι πιο ευκατανόητο και πιο ευκολοδιάβαστο το πρόγραμμα μου.
Σε αυτό το σημείο θέλω να εξηγήσω κάποιες απ' τις εντολές που χρησιμοποίησα.
Η εντολή `P2OUT&=~BIT4` σημαίνει P2.4=0
Ενώ `P2OUT|=BIT4` σημαίνει P2.4=1.
Με την εντολή `P1DIR = 0;` δηλώνουμε ότι η Port1 είναι δηλωμένη ως Input.
Ενώ με αυτήν `P2DIR = BIT0 | BIT1 | BIT2 | BIT3 | BIT4` δηλώνουμε ότι οι P2.0 έως P2.4 είναι έξοδοι. Αυτές θα στείλουμε ως είσοδο στον DAC μας.
Ενώ την συγκεκριμένη loopa `for (i = 100000; i>0; i--);` Ή διαφορετικά αυτή την εντολή `__delay_cycles(1);`
χρησιμοποιούμε για καθυστέρηση delay. Αλλάζοντας την τιμή του αλλάζουμε την καθυστέρηση.

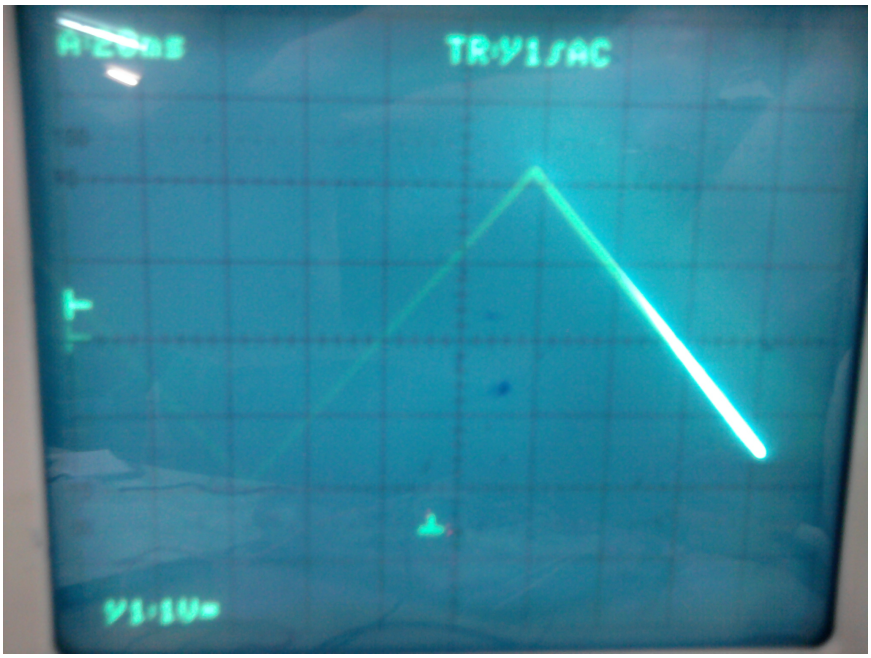
Η έξοδος του προγράμματος είναι

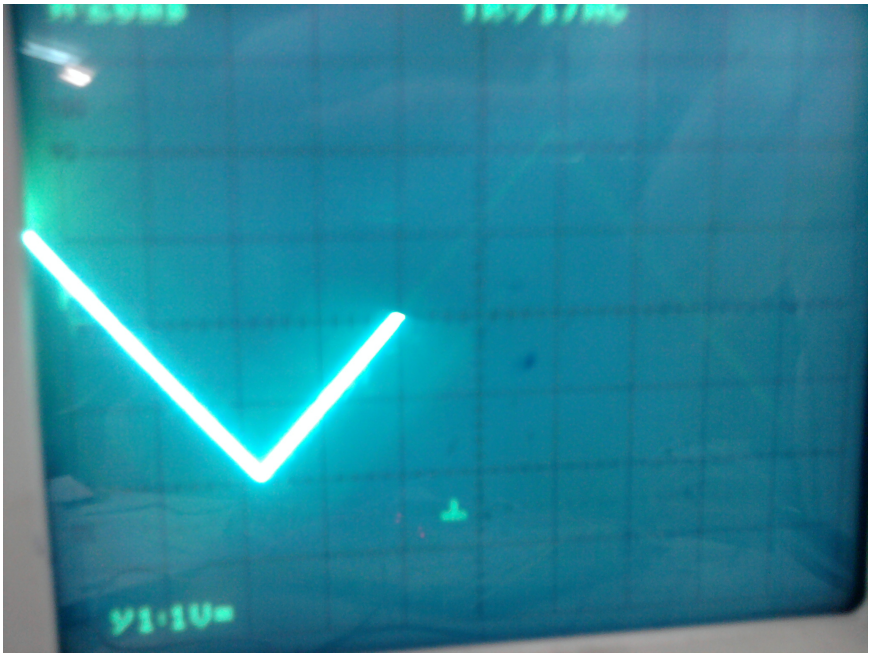


Σχήμα 6.3. η έξοδος απ' τον DAC7611p

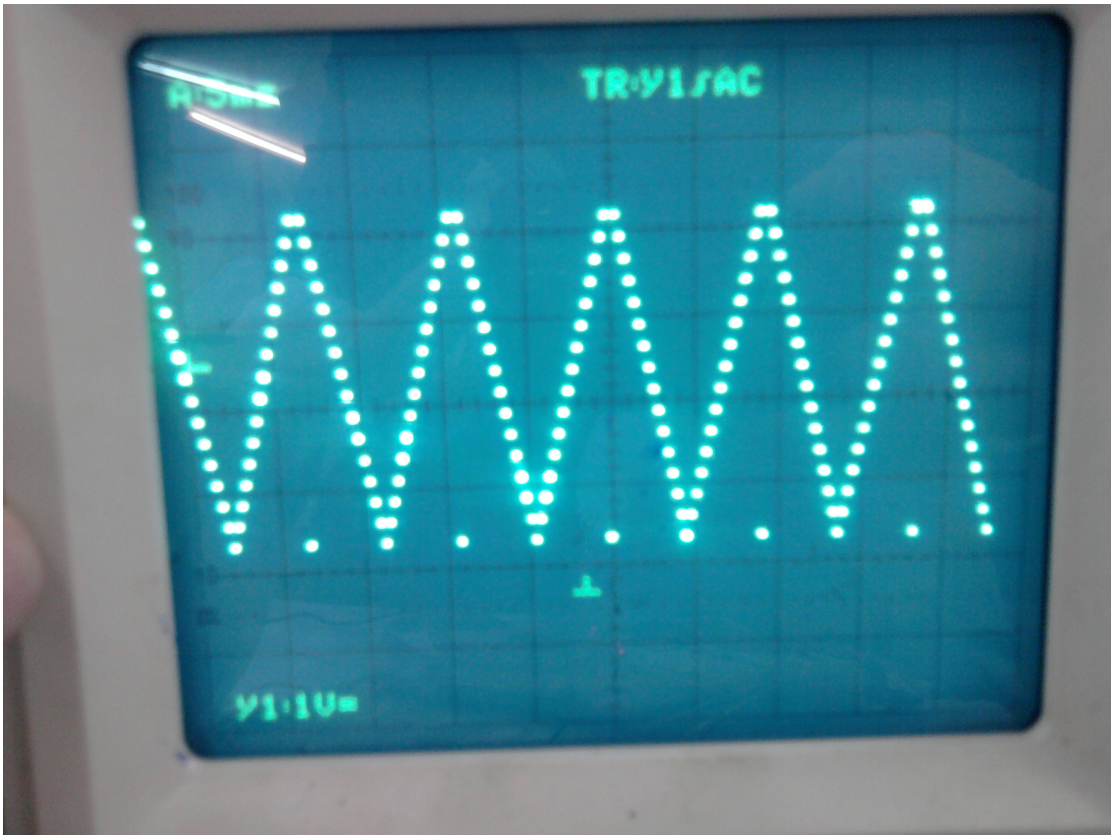
Ένα τριγωνικό αναλογικό σήμα

Από δω και στο εξής μπορούμε να ασχοληθούμε με τον έλεγχο της συχνότητας. Να μεταβάλλουμε την συχνότητα του σήματος.

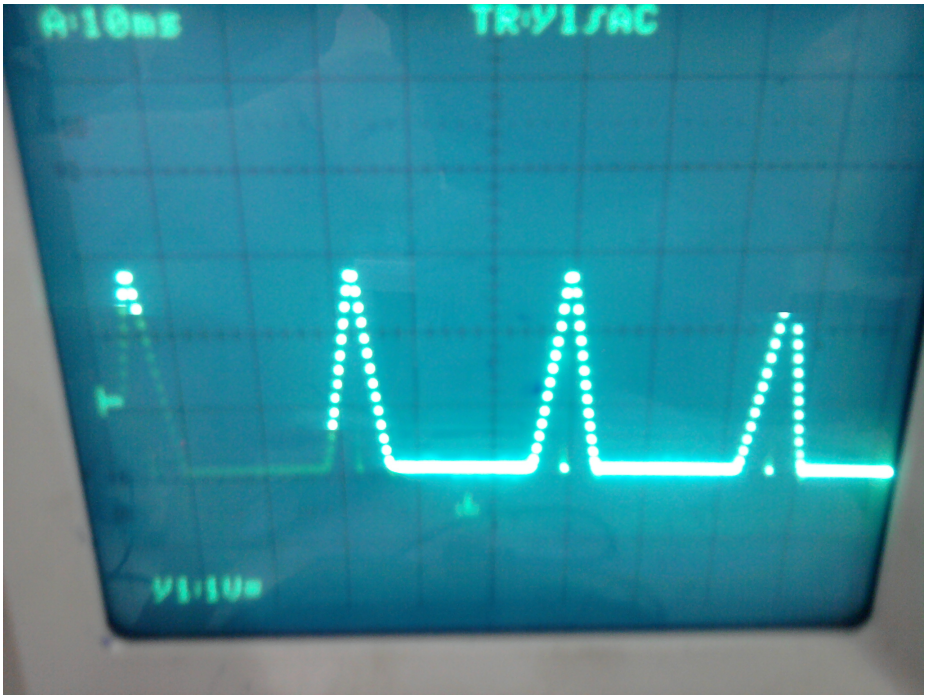


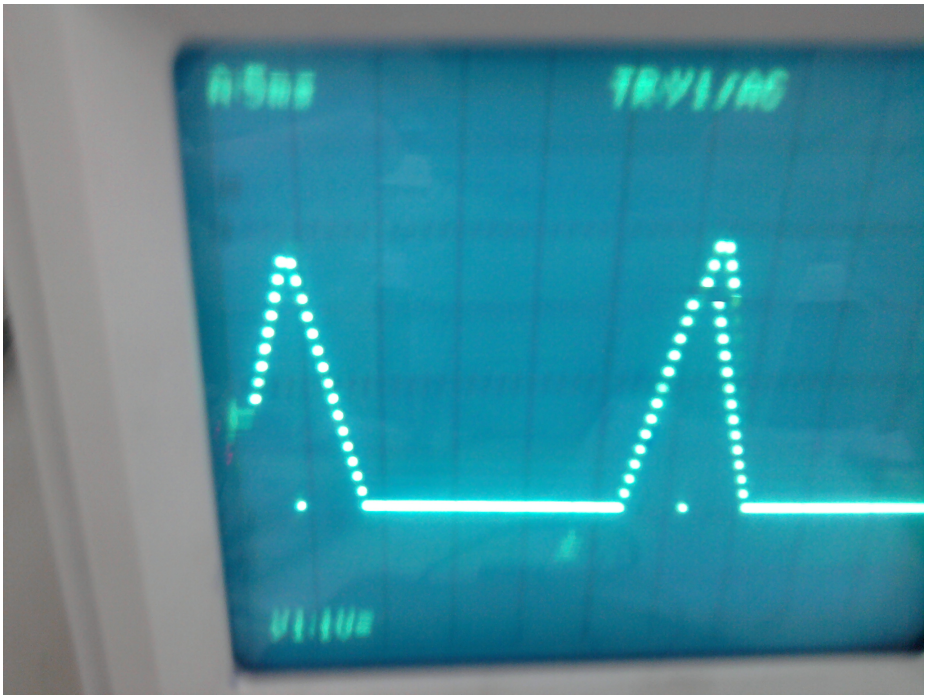


Αλλάζοντας το βήμα μας από 1 σε 100, παίρνουμε το σήμα μας σε αυτή τη μορφή



Στο παρακάτω παράδειγμα, έχω χρησιμοποιήσει μετά από κάθε ανεβοκατέβασμα κάποια καθυστέρηση – delay – ενώ θα μπορούσα να είχα περάσει το οποιοδήποτε σήμα ήθελα – δεδομένα (πληροφορία)
Πλέον μπορώ να ελέγχω το αναλογικό μου σήμα





Να αναφέρουμε ότι η περίοδος του τριγωνικού μας σήματος είναι T = 10ms, η συχνότητα f = 100 Hz,
Με βήμα 100 λοιπόν, βλέπουμε 16 φορές δεδομένα στην άνοδο του σήματος και άλλες 16 στην κάθοδο, επομένως 32 αλλαγές σε 10 ms επομένως
η συχνότητα του σήματος μας είναι στα 100 *32 Hz = 3.2 kHz.

Ενώ σε περίπτωση που ήθελα να περάσω μόνο 1 register στον μικροελεγκτή μου , το πρόγραμμα είναι το εξής

```
#include <msp430.h>

#define CLR_LO (P2OUT&=~BIT4)
#define CLR_HI (P2OUT|=BIT4)

#define CS_LO (P2OUT&=~BIT3)
#define CS_HI (P2OUT|=BIT3)

#define SDI_LO (P2OUT&=~BIT1)
#define SDI_HI (P2OUT|=BIT1)

#define CLK_LO (P2OUT&=~BIT2)
#define CLK_HI (P2OUT|=BIT2)

#define LD_LO (P2OUT&=~BIT0)
#define LD_HI (P2OUT|=BIT0)

int main(void)
{

	WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer

	P1DIR = 0; //input Port1
```

```
	P2DIR = BIT0 | BIT1 | BIT2 | BIT3 | BIT4 ; //input P2.4-P2.5, Out
bit 0..4

	P2OUT = 0xff;

	CLR_HI;
	CS_HI;
	SDI_HI;
	CLK_HI;
	LD_LO;

	__delay_cycles(500000); // 1000 clock cycles delay = 1ms@1MHz
	LD_HI;

	__delay_cycles(500000); // 1000 clock cycles delay = 1ms@1MHz


	//SDI <- 970 hex
	long int value=0x970;
	CS_LO;
	int j;

	for(j=0; j<12; j++)
	{
		CLK_LO;
		if(value & 0x800) //if(value & 1)
			SDI_HI;
		else
			SDI_LO;
		__delay_cycles(500000); // 1000 clock cycles delay = 1ms@1MHz
		CLK_HI;
		__delay_cycles(500000); // 1000 clock cycles delay = 1ms@1MHz
		value <<= 1;		//shift left data
	}
	SDI_HI;
	CS_HI;

	__delay_cycles(800000); // 1000 clock cycles delay = 1ms@1MHz
	LD_LO;

	__delay_cycles(600000); // 1000 clock cycles delay = 1ms@1MHz
	LD_HI;

	__delay_cycles(6000000); // 1000 clock cycles delay = 1ms@1MHz
	CLR_LO;
}
```

Εδώ με το που περαστεί η 1 εγγραφή (register) σύμφωνα και με το διάγραμμα χρονισμού ενεργοποιείται η εντολή CLR_LO; και μηδενίζεται η έξοδος μου.

ΕΠΙΛΟΓΟΣ

Το δεύτερο μέρος της εργασίας, δηλαδή η δημιουργία του αναλογικού σήματος μέσω θύρας USB, αλλά αυτή τη φορά χρησιμοποιώντας άλλον microcontroller, τον MSP430, σκοπό είχε να αποδείξει ποιος ο καλύτερος τρόπος εκπόνησης της ίδιας εργασίας (του ίδιου θέματος). Αποδείχτηκε ότι ο MSP430 με την δικιά του σουίτα προγραμματισμού (CodeComposerStudio) μπορεί πιο εύκολα να πραγματοποιήσει το ίδιο έργο με τον UMR (ο οποίος χρησιμοποίησε την σουίτα VisualC++ για τον προγραμματισμό του). Εκεί ήταν απαραίτητο να βρεθούν όλες οι βιβλιοθήκες επικοινωνίας , καθώς και φόρμα (με button, οθόνη, τερματικούς…) Ενώ στη σουίτα CCS που καλύπτει όλα τα devices της MSP δεν είχα να ασχοληθώ με αυτά, γιατί τα παρείχε εξαρχής.

Καταλήγοντας, ασχοληθήκαμε με την παραγωγή αναλογικού σήματος μέσω θύρας USB,διότι μας ενδιέφερε η επικοινωνία μεταξύ Η/Υ -μικροελεγκτή και εν συνεχεία χρησιμοποιήθηκε και ψηφιακός μετατροπέας. Η κατασκευή μας αυτή αποτελεί ένα πρότυπο ή ένα αρχικό στάδιο για την δημιουργία μιας εξωτερικής κάρτας ήχου.

Άλλες εφαρμογές αυτοματισμού υπαρκτές οι οποίες «πατάνε» πάνω στην συγκεκριμένη εργασία που εκπόνησα

- Motor Control
- Display Ηλεκτρονικών
- Ψηφιακό Ποτενσιόμετρο
- Αποκωδικοποίηση video
- Σύστημα Διανομής Δεδομένων
- Λογισμικό εκπομπής

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] www.ftdichip.com

[2] datasheet DAC7611

[3] “Elements of industrial Automation” at ELDE

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

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

[6] http://www.informit.com/articles/article.aspx?p=1881386&seqNum=2

[7]http://stackoverflow.com/questions/11063978/c-how-to-sleep-for-a-nanosecond

[8] http://msdn.microsoft.com

[9] datasheet MSP430g2553

[10] http://e2e.ti.com/support/microcontrollers/msp430/
TEXAS INSTRUMENTSforum

[11]http://e2e.ti.com/support/microcontrollers/msp430/f/166/p/379668/1342252.aspx#1342252

[12] http://stackoverflow.com/questions/6647783/check-value-of-least-significant-bit-lsb-and-most-significant-bit-msb-in-c-c

[13]http://e2e.ti.com/support/microcontrollers/msp430/f/166/t/372790.aspx