Πανεπιστήμιο Θεσσαλίας
Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών,
Τηλεπικοινωνιών & Δικτύων

Σχεδίαση VLSI 12-bit ΣΔ ADC πρώτης τάξης σε CMOS τεχνολογία 0.35μm

Διπλωματική Εργασία:

Τσιούγκος Δημήτριος

Μάρτιος 2007

Ευχαριστίες

Θα ήθελα αρχικά να ευχαριστήσω θερμά τον καθηγητη μου και βασικό επιβλέποντα της εργασίας κ. Παναγιωτη Δημητρόπουλο για τη συμπαράσταση και βοήθεια του ολο αυτό το διαστημα.

Θα ήθελα επίσης να ευχαριστήσω τον εταιρο καθηγητη μου κ. Γεώργιο Σταμούλη για τις καιριες υποδειξεις και συμβουλες του.

Ένα μεγάλο ευχαριστω στους κκ. Εμμανουήλ Κουτσουμπέλια και Δημήτριο Συρίβελη για τη βοήθειά τους επάνω σε θέματα μικροελεγκτών και στον προγραμματισμό τους.

Τέλος ευχαριστώ από καρδιας την οικογένειά μου, τους φίλους και συμφοιτητες μου, που άντεξαν και ηταν δίπλα μου στις πιο δυσκολες στιγμές μου.

Κεφάλαιο 1

Γενικά...

I. Περί μέτρων... ο λόγος

Η μέτρηση τόσο των φυσικών (πίεση, θερμοκρασία) όσο και των ηλεκτρικών φαινόμενων (τάση, ρεύμα) είναι αντικείμενο της τεχνολογίας αισθητήρων. Τα ηλεκτρονικά συστήματα μέτρησης γενικά υπερτερούν των παλαιών μηχανικών και ηλεκτρομηχανικών λύσεων για τους εξής απλούς λόγους :

1. Αποδεδειγμένη αξιοπιστία και αντοχή σε ακραία φαινόμενα
2. Ακρίβεια
3. Εύκολη ρύθμιση παραμέτρων (calibration)
4. Ασφάλεια

Το επόμενο βήμα, μετά τη μέτρηση, είναι η συλλογή των πληροφοριών και η επεξεργασία τους για την απομόνωση του θορύβου και άλλων παραμέτρων που επηρεάζουν άμεσα τα αποτελέσματα των μετρήσεων και συνάμα την εξαγωγή ακριβέστερων συμπερασμάτων.

II. Σκοπός – Στόχος

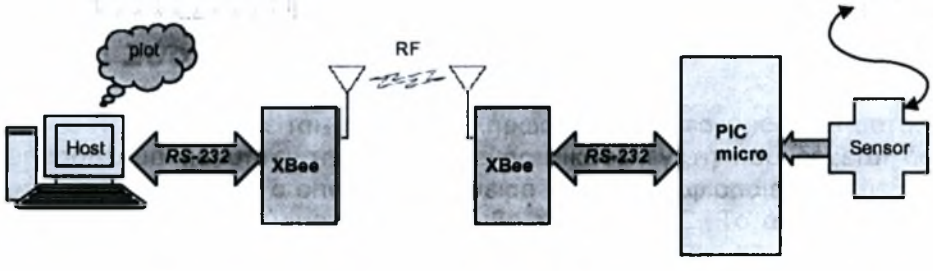
Κύριος στόχος της εργασίας αποτελεί η συλλογή αναλογικών δεδομένων που εξάγονται από έναν αισθητήρα, η ψηφιακή τους μετατροπή, επεξεργασία και αποστολή τους σε υπολογιστή για περαιτέρω ανάλυση και αναπαράσταση αποτελεσμάτων. Επειδή όμως η ανάγκη για συνεχείς μετρήσεις και καταγραφή δεδομένων, από διαφορετικά σημεία ταυτόχρονα, είναι μεγάλη, θα ήταν θεμιτό να εκμεταλλευτούμε την ασύρματη επικοινωνία και τις εφαρμογές της. Για την υλοποίηση του συστήματος αποφασίστηκε η χρήση ενός ολοκληρωμένου κυλωματος μικροελεγκτή που παρέχει υψηλής απόδοσης περιφερειακά όπως απλή συνδεσιμότητα (σειριακή επικοινωνία), A/D converter κ.α. καθώς και η χρήση ενός RF module για την ασύρματη μετάδοση των δεδομένων. Το χαμηλό κόστος των δυο υποσυστημάτων και η εξίσου χαμηλή κατανάλωση ισχύος τα κάνουν να υπερτερούν σε σχέση με αλλα modules. Το ολοκληρωμένο κύκλωμα μικροελεγκτή επιλέγεται να είναι τις οικογένειας μικροελεγκτών **PIC18**, και συγκεκριμένα το PIC18F4550 γιατι διαθέτει ADC, σειριακή και USB θύρα, δυνατότητα χρήσης εσωτερικου ή

εξωτερικού ρολογιού, I/O θύρες (ψηφιακές και αναλογικές) και η κατανάλωση ισχύος παραμένει σημαντικά χαμηλή.

II. Το Σύστημα γενικά

Το σύστημα αποτελείται από δυο πλήρως ανεξάρτητα υποσυστήματα. Το ένα υποσύστημα καλύπτει την πλευρά ενός μικροελεγκτή που δέχεται δεδομένα από τον αισθητήρα, ο οποίος με τη σειρά του, τα ψηφιοποιεί, τα επεξεργάζεται και τα μεταδίδει ασύρματα στον κόμβο του host PC. Το άλλο καλύπτει αυτή του host PC, όπου τα δεδομένα επεξεργάζονται και σχεδιάζονται για περαιτέρω ανάλυση και παρατήρηση.
Η ασύρματη επικοινωνία γίνεται μέσω δύο **RF modules (XBee)** τα οποία καλύπτουν τις ανάγκες του wireless πρωτοκόλλου **IEEE 802.15.4** και εκπέμπουν σε συχνότητα 2.4GHz, συνδυάζοντας χαμηλή κατανάλωση ισχύος και αξιοπιστία στη μετάδοση – λήψη δεδομένων (Πλήρης ανάλυση του συστήματος στο *Κεφάλαιο 3*). Παρακάτω δίνεται το σχεδιάγραμμα του συστήματος

Σχηματικό διάγραμμα Συστήματος



Το κεφάλαιο 2 καλύπτει εξολοκλήρου την αρχιτεκτονική του μικροελεγκτή καθώς και τα βασικά χαρακτηριστικά των περιφερειακών του που χρησιμοποιήθηκαν στην εφαρμογή.
Το κεφάλαιο 3 αναπτύσσει το σύστημα και περιγράφει τις τεχνικές λεπτομέρειες και τις λειτουργικότητες του κάθε εξαρτήματος ξεχωριστά.
Το κεφάλαιο 4 περιέχει τον κώδικα του μικροελεγκτή, του host PC και τον σχολιασμό του.
Τέλος το Κεφάλαιο 5 περιλαμβάνει τις μετρήσεις του συστήματος οι οποίες έγιναν με τη βοήθεια παλμογεννήτριας στο εργαστήριο.

Κεφάλαιο 2

Αρχιτεκτονική PIC18FXXXX και Περιφερειακά Στοιχεία

Το ***PIC18F4550*** ολοκληρωμένο κύκλωμα μικροελεγκτή ανήκει στην κατηγόρια των 8-bit **RISC** (Reduced Instruction Set Computer) **MCUs** και παρέχει ένα σύνολο ενσωματωμένων ηλεκτρονικών υποσυστημάτων υψηλής λειτουργικότητας που το καθιστούν αρκετά σημαντικό στις εφαρμογές αισθητήρων. Παρόλα αυτά, εξέχοντα χαρακτηριστικά των RISC μικροεπεξεργαστών παραλείπονται από την αρχιτεκτονική των PIC. Τα σημαντικότερα είναι :

1. Η αρχιτεκτονική δε βασίζεται στο κατά ***Von Neumann*** μοντέλο ***load/store***
2. Υπάρχει μοναδικός καταχώρησης (WREG) που δουλεύει συνεχώς σε σχέση με τους πραγματικούς RISC που αριθμός των καταχωρητων εργασίας είναι 32 ή και περισσότεροι.

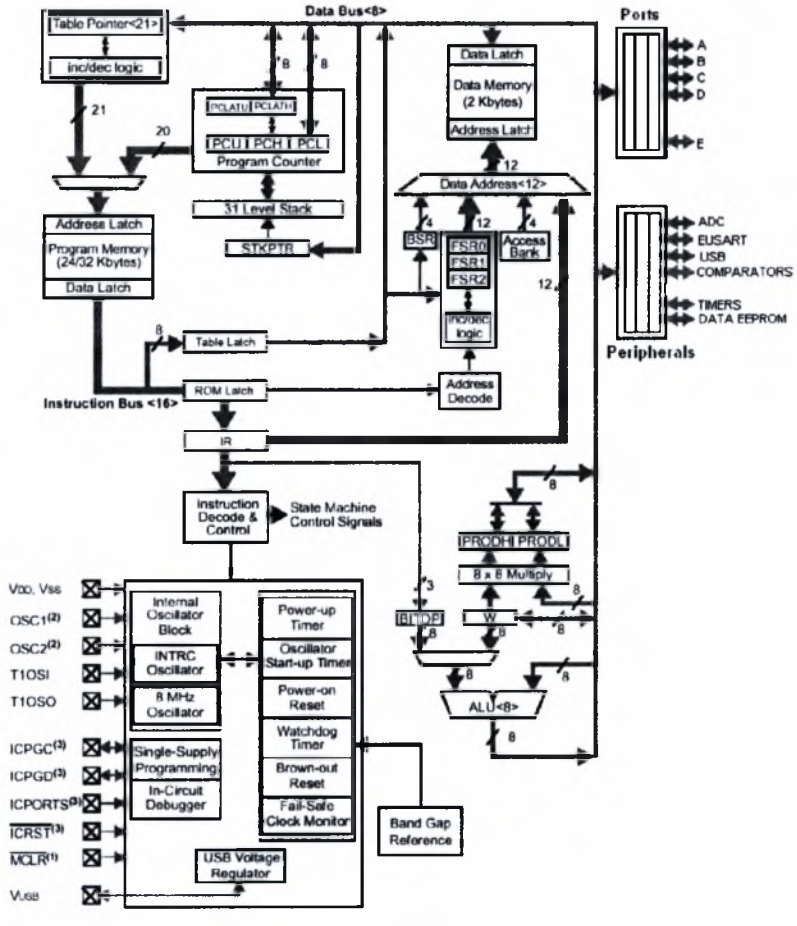
Μερικά από τα βασικά χαρακτηριστικά του **PIC18F4550** είναι που τον κάνουν ιδιαίτερα λειτουργικό:

- **Operating Frequency** DC– 48 MHz
- **Program Memory (Bytes)** 32768
- **Program Memory (Instructions)** 16384
- **Data Memory (Bytes)** 2048
- **Data EEPROM Memory (Bytes)** 256
- **Interrupt Sources** 20
- **I/O Ports** Ports A, B, C, D, E
- **Timers** 4
- **Serial Communications** Enhanced USART
- **10-bit Analog-to-Digital Module** 13 Input Channels
- **Comparators** 2
- **Resets** POR, BOR
- **Instruction Set** 75 Instructions, 83 with Extended instruction set

Το διάγραμμα οργάνωσης του μικροελεγκτή εικονίζεται παρακάτω. Για τη λειτουργία τόσο του ταλαντωτή του ολοκληρωμένου (ρολογιού) όσο και των περιφερειακών υποσυστημάτων, ο controller διαθέτει σε συγκεκριμένο σημείο

της μνήμης ειδικούς καταχωρητές ελέγχου (***configuration registers***) για κάθε ένα από τα περιφερειακά. Υπάρχει επίσης η δυνατότητα προστασίας των τμημάτων της "***Rom***" μνήμης(***Program memory blocks***, ***configuration bits***) καθώς και της "***Ram***" μέσω του ***linker script***. Όλα αυτά θα περιγράφουν αναλυτικά στις παρακάτω παραγράφους.

<u>Σχηματικό διάγραμμα Οργάνωσης μικροελεγκτή</u>



I. Δομή Ρολογιού/Ταλαντωτή

Ο μικροελεγκτής υποστηρίζει δώδεκα διαφορετικούς τρόπους τροφοδότησης του κυκλώματος με ρολόι και οι οποίοι αναφέρονται παρακάτω :

- **XT** Crystal/Resonator
- **XTPLL** Crystal/Resonator with PLL enabled
- **HS** High-Speed Crystal/Resonator
- **HSPLL** High-Speed Crystal/Resonator With PLL enabled

- **EC** External Clock with FOSC/4 output
- **ECIO** External Clock with I/O on RA6
- **ECPLL** External Clock with PLL enabled and FOSC/4 output on RA6
- **ECPIO** External Clock with PLL enabled, I/O on RA6

- **INTHS** Internal Oscillator used as Microcontroller clock source, HS Oscillator used as USB clock source
- **INTXT** Internal Oscillator used as Microcontroller clock source, XT Oscillator used as USB clock source
- **INTIO** Internal Oscillator used as Microcontroller clock source, EC Oscillator used as USB clock source, Digital I/O on RA6
- **INTCKO** Internal Oscillator used as Microcontroller clock source, EC Oscillator used as USB clock source,FOSC/4 output on RA6

Ιδιαίτερο ενδιαφέρον παρουσιάζει το γεγονός ότι οι τέσσερεις τελευταίοι τρόποι τροφοδότησης κάνουν χρήση δυο διαφορετικών ταλαντωτών ταυτόχρονα. Πιο αναλυτικά, το ρολόι του *CPU core* μπορεί να είναι ο εσωτερικός ταλαντωτής του microcontroller ενώ το περιφερειακό της *USB* να τροφοδοτείται από εξωτερικό ταλαντωτή (***INTIO***).

Για τη διανομή και τον έλεγχο του ρολογιού στο ολοκληρωμένο, υπεύθυνοι είναι τέσσερεις καταχωρητές :

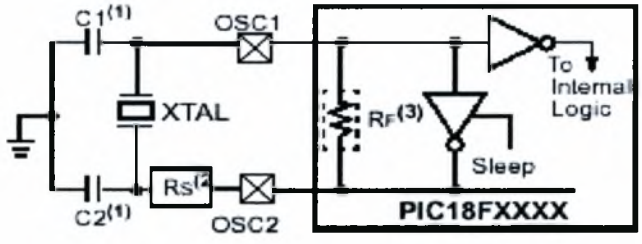
a. Οι καταχωρητές *CONFIG1L* and *CONFIG1H* διαλέγουν το oscillator mode (οι 12 παραπάνω τρόποι(CONFIG1H bits **FOSC3:0**)) και τον τρόπο τροφοδότησης της *USB*.

b. Ο *OSCCON* επιλέγει το clock mode το κυκλώματος (internal oscillator, primary clock, 'secondary' or timer clock, (OSCCONbits.**SCS1:0**))

c. Ο *OSCTUNE* κάνει αλλαγές στον εσωτερικό ταλαντωτή

Για την εφαρμογή μας, επιλέξαμε τον πρώτο τρόπο τροφοδότησης (**XT**) χωρίς τη χρήση PLL καθώς το περιφερειακό της USB δε χρησιμοποιείται από την εφαρμογή. Ένα απλό σχεδιάγραμμα του ρολογιού που χρησιμοποιήθηκε δίνεται παρακάτω.

Σχηματικό διάγραμμα κυκλωματος ταλαντωτή μικροελεγκτή

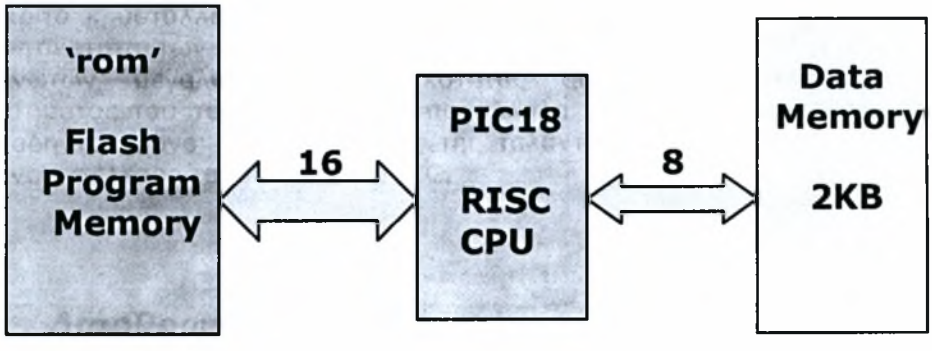


Όπως παρατηρούμε, το ενεργό κύκλωμα του ταλαντωτή βρίσκεται μέσα στον PIC ενώ εξωτερικά προσθέτουμε το παθητικο μερος το οποιο αποτελειται από διακριτο κρυσταλλο 4MHz και δυο πυκνωτες στα ακρα του κρυστάλλου. Η χωρητικοτητα των πυκνωτων είναι 30pF. Τελος να αναφέρθει ότι η χρηση πυκνωτων μεγαλυτερης χωρητικότητας αυξάνει την ευσταθεια του συστηματος του ταλαντωτη αλλα παραλληλα αυξανει και το start-up time, δηλαδη το χρονο σταθεροποίησης της ταλάντωσης και σαν αποτέλεσμα το χρόνο εκτέλεσης του κώδικα του PIC.

II. Διάρθρωση Μνήμης

Το κυριο χαρακτηριστικο και συναμα πλεονεκτημα των μικροελεγκτων είναι το προτυπο διαρθρωσης της μνήμης που χρησιμοποιουν. Γενικα οι μικροελεγκτες είναι προσανατολισμενοι στο ***Harvard Architecture style***, συμφωνα με το οποιο υπαρχουν δυο ξεχωριστα τμηματα μνήμης για τα δεδομενα(*data*) και για τις εντολής(*instructions*). Τα τμηματα αυτά χρησιμοποιούν διαφορετικούς διαύλους (busses) και έτσι μπορεί να επιτευχθεί ταυτόχρονη προσπέλαση στις δυο αυτές περιοχές της μνήμης. Ο δίαυλος για το program memory space εχει χωρητικότητα 16 bits ενώ για το data memory space το bus είναι 8 bits.

Harvard Architecture - Separate busses for "rom" and Data memory

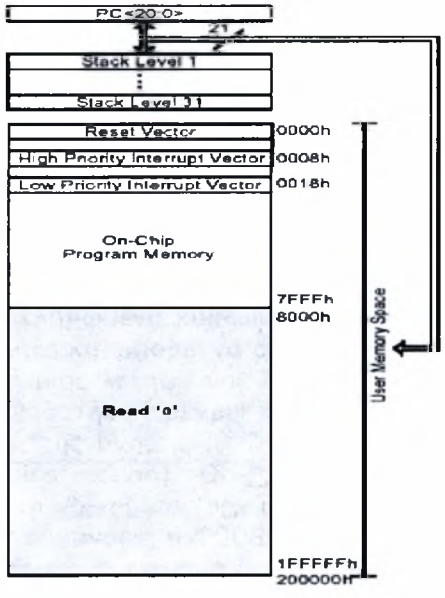


Βασικό μειονέκτημα του συγκεκριμένου προτύπου αποτελει η προσπελαση δεδομενων που είναι αποθηκευμένα στη μνήμη εντολών(***Program memory space***) αφού πρόκειται για αργή διαδικασια καθώς η διευθυνσιοδότηση δεν γίνεται άμεσα.

II.1 Οργάνωση μνήμης εντολών(Program memory)

Το ***PIC18F4550*** ολοκληρωμένο κύκλωμα μικροελεγκτή διαθέτει μνήμη flash χωρητικότητας 32Kbytes και μπορεί να αποθηκεύσει μέχρι 16,384 single-word εντολές (κάθε εντολή μιας λέξης είναι 2bytes). Αν και ο **PC** με πεδίο 21bits μπορεί να διευθυνσιοδοτήσει μέχρι και 2Mbytes διάστημα μνήμης, αν προσπελαστεί περιοχή της μνήμης με διεύθυνση μεγαλύτερη της 0x7FFF θα επιστρέψει '0' (Nop instruction). Οι ***PIC18*** μικροελεγκτές, διαθέτουν στη διεύθυνση 0x0000h το *Reset vector*, το σημείο που ο κώδικας αρχίζει να εκτελείται, και στις διευθύνσεις 0x0008h και 0x0018h τα *high interrupt* και *low interrupt vectors* αντίστοιχα. Απλό σχεδιάγραμμα της μνήμης παρουσιάζεται παρακάτω.

Σχηματικό διάγραμμα Οργάνωσης Μνήμης Εντολών μικροελεγκτή



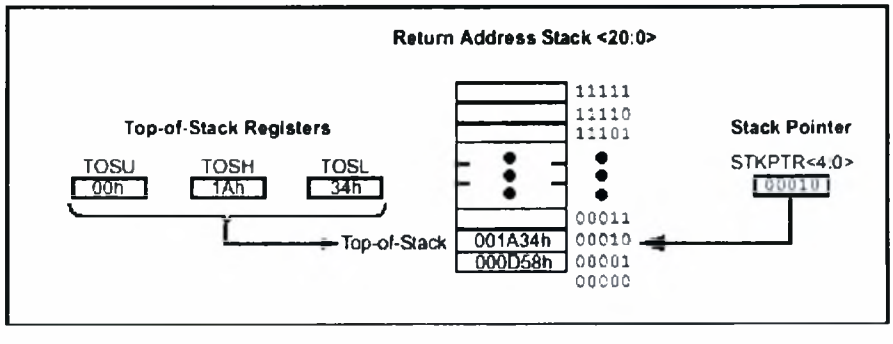
II.1.1 Program Counter

Ο ***PC*** καθορίζει τη διεύθυνση της επόμενης προς εκτέλεση εντολή. Αποτελείται από τρεις καταχωρητές, τους *PCU:PCH:PCL*. Ο *PCL* μπορεί να διαβαστεί και να γραφεί σε σχέση με τους δύο άλλους καταχωρητές των οποίων η προσπέλαση γίνεται μέσω των *PCLATU* και *PCLATH* καταχωρητών. Τα περιεχόμενα των *PCLATU* και *PCLATH* μεταφέρονται στους *PCU:PCH* όταν ο *PCL* γράφεται. Αντίστροφη μεταφορά δεδομένων γίνεται όταν ο *PCL* διαβάζεται. Το LSb του *PCL* είναι πάντα '0' και έτσι ο PC αυξάνεται πάντα κατά 2 (PC=PC+2). Οι εντολές CALL, RCALL και GOTO αποτελούν εντολές διακλάδωσης και γράφουν απ΄ευθείας στον PC. Έτσι τα περιεχόμενα των *PCLATU* και *PCLATH* δε μεταφέρονται στον ***PC***.

II.1.2 Return Address Stack

Η *address stack* επιτρεπει το συνδιασμο εως 31 κλησεων και διακοπών (interrupts) να συμβουν. Αποτελει ξεχωριστο τμημα τοσο από τη μνημη δεδομενων οσο και από τη μνημη εντολων και λειτουργει σα RAM 31 λεξεων από 21 bits η κάθε μια και έναν δείκτη (*STKPTR*). Όταν μια από τις CALL, RCALL εκτελειται ή μια διακοπή αναγνωρίζεται, τοτε ο ***PC*** μεταφέρεται στη στοίβα. Αν εκτελεστεί η CALL τότε ο STKPTR αυξάνεται και στη θέση που δείχνει αποθηκεύεται η τιμή του ***PC*** που υποδηλώνει τη νεα εντολή μετά την CALL. Στις εντολές τύπου RETURN, τα περιεχόμενα της θέσης που δείχνει ο STKPTR μεταφέρονται στον ***PC*** και ο STKPTR μειώνεται. Σε περίπτωση που συμβεί Reset στη συσκεύη ο STKPTR παιρνει την τιμή '00000'. Ο STKPTR διεχνει την κορυφή της στοίβας, η μπορεί να προσπελαστεί μέσω των καταχωρητών *TOS:TOSH:TOSL*. Το σχεδιαγραμμα παρουσιαζει τη λειτουργία της στοίβας.

H/W Return Address Stack and TOS Registers



II.1.3 Stack Pointer

Το πιο σημαντικό bit (STKFUL) του καταχωρητή παίρνει την τιμή 1 όταν γίνεται η 31η αποθήκευση στη στοίβα. Αν το STVREN configuration bit είναι 1 τοτε στην 31η εισαγωγή δεδομένων στη στοίβα, η τιμη PC+2 αποθηκεύεται και γίνεται Reset.
Αν STVREN=0 τοτε εχω αύξηση του STKPTR στην τιμή 31 και οποιαδήποτε νέο push δεν υπερκαλύπτει το 31η push. Το STKUNF γίνεται 1 όταν η στοίβα αδειάζει και γίνεται Reset αν STVREN=1.

STKPTR : Stack Pointer Register

STKFUL	STKUNF	-	SP4	SP3	SP2	SP1	SP0

Αξίζει να αναφερθεί ότι το σύνολο εντολών (*instruction set*) του μικροελεγκτή περιέχει τις εντολές PUSH και POP, οι οποίες μεταφέρουν τιμές του ***PC*** προς και από τη στοίβα (τιμές του TOS) χωρίς να επηρεάζουν την ομαλή εκτέλεση του κώδικα της εφαρμογής.

Λειτουργία των Push και Pop

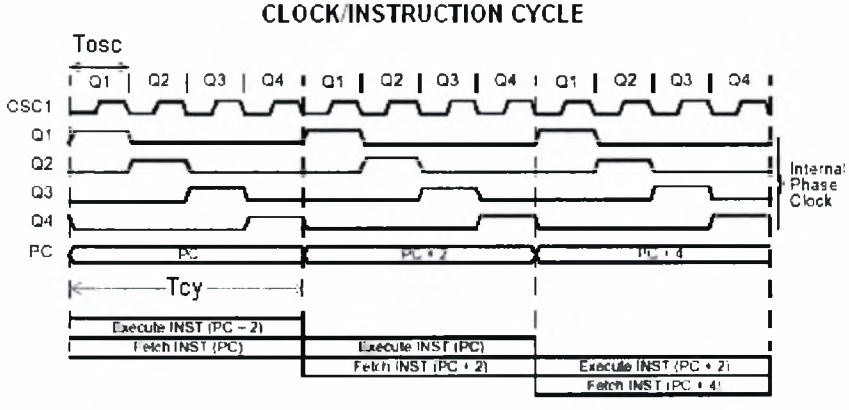
PUSH()	POP()
	STKPTR--;
STKPTR++;	TMP=MEM[STKPTR];
TOS = PC;	TOS = TMP;

II.1.4 Fast Register Stack

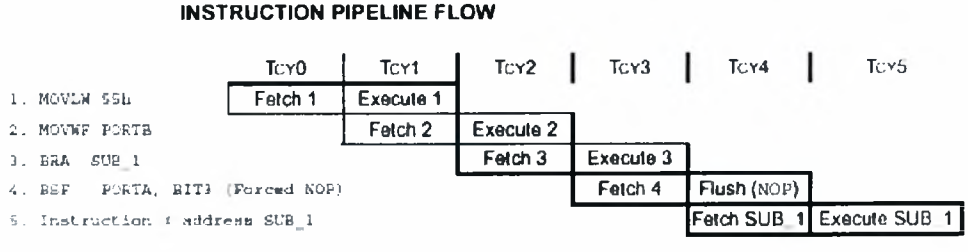
Ενα FRS χρησιμοποιείται για να σώζονται οι καταχωρητές WREG, BSR και STATUS όταν μια διακοπή (*interrupt*) συμβαίνει. Κάθε τέτοια στοίβα έχει βάθος μόνο ένα και όταν αναγνωριστεί κάποιο interrupt τότε οι τιμές των βασικών καταχωρητών φορτώνονται σε αυτές. Όταν εκτελείται η εντολή RETFIE τότε τα δεδομένα της κάθε στοίβας επιστρέφουν στους βασικούς καταχωρητές. Στην περίπτωση που τόσο οι υψηλής όσο και χαμηλής προτεραιότητας διακοπές έχουν ενεργοποιηθεί, οι FRS δεν εξασφαλίζουν τη σωστή επιστροφή των low priority interrupts. Και αυτό γιατί όταν μια υψηλή διακοπή συμβεί ενώ την ιδια στιγμή μια χαμηλή εξυπηρετείται, τότε οι τιμές της low διακοπής στο FRS γράφονται από πάνω από την high διακοπή.

II.1.5 Κύκλος εντολής του PIC

Το ρολόι του μικροελεγκτή διαιρείται εσωτερικά σε τέσσερεις μη επικαλυπτόμενες χρονικές διάρκειες Q1, Q2, Q3 και Q4. Ο ***PC*** (*Program Counter*) αυξάνεται κατά 2 και δείχνει την επόμενη προς εκτέλεση εντολή σε κάθε Q1. Κατά τη διάρκεια Q1 έως Q4, η εντολή έρχεται από τη μνήμη (fetched from *Program Memory*) και αποθηκεύεται στον ***IR*** register κατά τη φάση Q4. Με την έλευση της νέας χρονικής περιόδου Q1-Q4, η εντολή αποκωδικοποιείται και εκτελείται. Πιο συγκεκριμένα, για κάθε εντολή έχουμε δυο χρονικές φάσεις, αυτή της μεταφοράς από τη μνήμη (*fetch*) και αυτή της εκτέλεσης (*execute*) και συνολικά **δυο** κύκλους εντολής ($2T_{cy}$). Απο το παρακάτω διάγραμμα είναι φανερό ότι ισχύει: $T_{cy} = 4T_{osc}$



Κατά τη διάρκεια του κύκλου εκτέλεσης (*execute*), η προς εκτέλεση εντολή είναι αποθηκευμένη στον ***IR*** register στη φάση Q1. Στις Q2, Q3 και Q4, η εντολή αποκωδικοποιείται και εκτελείται. Η μνήμη δεδομένων (*Data memory*) διαβάζεται κατά την Q2(operand read) και γράφεται κατά την Q4 (write). Η αρχιτεκτονική του PIC όμως υποστηρίζει διπλό *pipelining* και έτσι η κάθε εντολή εκτελείται ουσιαστικά σε **ένα** κύκλο εντολών ($1T_{cy}$). Εντολές, όπως η GOTO, που αλλάζουν την υπάρχουσα τιμή του ***PC*** χρειάζονται 2 κύκλους για να ολοκληρώσουν την εκτέλεσή τους. Σχετικό διάγραμμα με την εκτέλεση εντολών ενός όσο και δυο κύκλων παρουσιάζεται παρακάτω.



Στο διάγραμμα, η εντολή 3 αποτελεί εντολή διακλάδωσης που αλλάζει την τιμή του PC ο οποίος μετά και την εκτέλεση της δείχνει σε διεύθυνση [PC+ &(SUB1)]. Η εντολή 4 αν και έρχεται στον IR προς εκτέλεση, "καίγεται" (*flushed*) και στον κύκλο εκτέλεσής της εκτελείται η Forced Nop (No operation). Αμέσως μετά, η εντολή που δείχνει ο PC (εντολή 5) εκτελείται. Άρα η εντολή 3 χρειάζεται δύο κύκλους για να ολοκληρωθεί.

II.1.6 Αποθήκευση Εντολών Στη Μνήμη

Οι εντολές αποθηκεύονται στη μνήμη σε μέγεθος δύο ή τέσσερα bytes. Το LS Byte αποθηκεύεται στη θέση μνήμης με άρτια διεύθυνση(LSb=0). Εντολές όπως οι CALL και GOTO έχουν ενσωματωμένες απόλυτες διευθύνσεις μνήμης, οι οποίες γράφονται στον PC. Η αποθήκευση της GOTO στη μνήμη (4 bytes) φαίνεται στο παρακάτω σχήμα.

Αποθηκευση εντολων στη μνήμη

INSTRUCTIONS IN PROGRAM MEMORY

			LSB = 1	LSB = 0	Word Address ↓
Program Memory Byte Locations →					000000h
					000002h
					000004h
					000006h
Instruction 1:	MOVLW	055h	0Fh	55h	000008h
Instruction 2:	GOTO	0006h	EFh	06h	00000Ah
			F0h	00h	00000Ch
Instruction 3:	MOVFF	123h, 456h	C1h	23h	00000Eh
			F4h	56h	000010h
					000012h
					000014h

Το Standard instruction set του PIC περιλαμβάνει τέσσερεις εντολές των τεσσάρων bytes: CALL, GOTO, MOVFF και LSFR. Το MS Byte είναι πάντα MSB=0xF, μια ειδική περίπτωση της Forced NOP για να εκτελείται σαν ξεχωριστή εντολή σε περίπτωση που τα δυο λιγότερα σημαντικά Bytes της εντολής παραληφθούν σε μια ακολουθία εντολών.

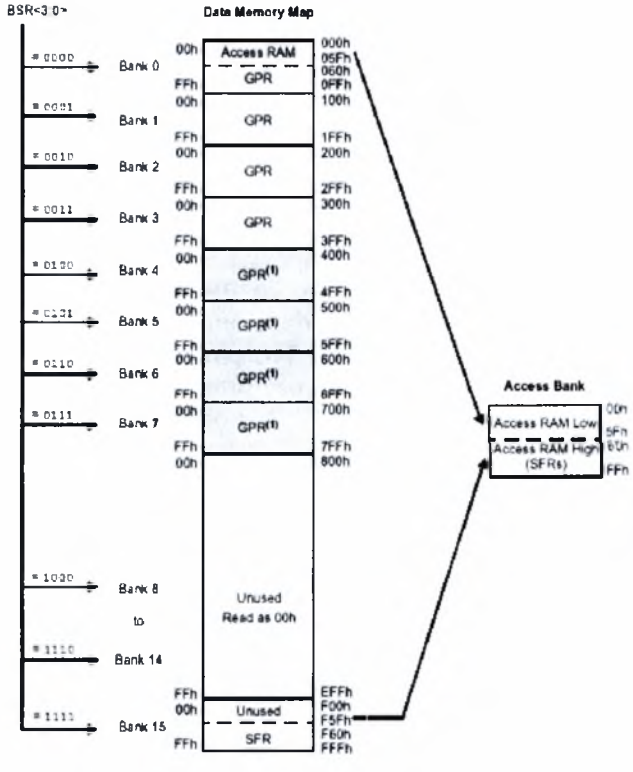
II.2 Οργάνωση μνήμης δεδομένων (Data memory)

Η μνήμη δεδομένων είναι σχεδιασμένη σα στατική μνήμη RAM. Αποτελείται από GPR(General Purpose Registers) και SFR(Special Function Registers) και η διεύθυνση για κάθε καταχώρηση είναι 12 bits, επιτρέποντας τη διευθυνσιοδοτηση έως και 4096 bytes. Το PIC18F4550 υλοποιεί μόνο 2048 bytes μνήμης δεδομένων, η οποία είναι χωρισμένη σε 16 τμήματα των 256 bytes (Banks). Οι καταχωρητές SFR ελέγχουν τον PIC και τα περιφερειακά του. Οι τρόποι διευθυνσιοδότησης του PIC είναι τρεις:

- **Άμεση** διευθυνσιοδοτηση
- **Έμμεση** διευθυνσιοδότηση
- **Διευθυνσιοδοτούμενη** διευθυνσιοδότηση

Για την προσπέλαση των πιο σημαντικών και χρησίμων καταχωρητών, ο μικροελεγκτής υλοποιεί την ***Access RAM,*** της οποίας η προσπέλαση κοστίζει μόνο ένα κύκλο εντολής(1Tcy). Απεικόνιση της RAM γίνεται στο παρακάτω σχήμα.

Μνήμη Δεδομενων (Data Memory)



Τα τμήματα μνήμης (Banks) 4 έως 7 χρησιμοποιούνται από το περιφερειακό της USB σαν ενδιάμεσο αποθηκευτικό χώρο (USB buffer). Μπορεί επίσης να χρησιμοποιηθεί σαν «κανονική» μνήμη με αλλαγή στον κώδικα του *Linker*.

II.2.1 Bank Select Register (BSR)

Το πεδίο της διεύθυνσης της μνήμης δεδομένων αποτελείται από 12 bits. Κάθε έγκυρη εντολή του instruction set του μικροελεγκτή, μπορεί να έχει ενσωματωμένη ολόκληρη τη διεύθυνση (12 bits) για το *data space* ή να περιέχει τα 8 λιγότερο σημαντικά bits της διεύθυνσης. Η πλήρης διεύθυνση συμπληρώνεται από τον *Bank Pointer* , που περιέχει τα 4 πιο σημαντικά bits της διεύθυνσης. Ο **BSR**(*BSR3:0*) καθορίζει το τμήμα της μνήμης (Memory Bank) που είναι αποθηκευμένα τα δεδομένα, ενώ τα 8 bits της εντολής λειτουργούν σαν **offset** που προστίθεται στη μικρότερη διεύθυνση του memory Bank που υποδεικνύεται από τον BSR. Αν και η επιλογή οποιουδήποτε εκ των 16 τμημάτων μνήμης είναι εφικτή, το γράψιμο (write) σε μη υλοποιημένες περιοχές αγνοείται, ενώ το διάβασμα (read) επιστρέφει '0'. Από ολόκληρο το instruction set, μόνο η MOVFF προσδιορίζει πλήρως την 12bit διεύθυνση του *data space*. Όλες οι υπόλοιπες χρησιμοποιούν το BSR ή την Access RAM για τον προσδιορισμό της διεύθυνσης.

II.2.2 Access RAM

Η Access RAM περιλαμβάνει 265 bytes του χώρου δεδομένων και αποτελείται από τα πρώτα 96 bytes του *Bank0* (0x000h – 0x05Fh) και τα τελευταία 160 bytes του *Bank15* (0xF60h – 0xFFFh). Το πρώτο τμήμα ονομάζεται **Access RAM** και αποτελείται από **GPRs** (General Purpose Registers) στους οποίους γίνεται η αποθήκευση των συχνότερα προσπελάσιμων δεδομένων (μεταβλητών και τιμών αυτών). Το δεύτερο τμήμα αποτελείται από τους **SFRs**(Special Function Registers), οι οποίοι είναι υπεύθυνοι για την επιθυμητή λειτουργία της συσκευής (CPU core Registers and peripherals). Η Access RAM χρησιμοποιείται από τις εντολές του μικροελεγκτή που εμπεριέχουν στο σώμα τους (opcode) το Access RAM bit ***a***. Αν ***a='1'*** τότε η εντολή κάνει χρήση του BSR και της 8bit διεύθυνσης που περιέχεται στο opcode. Αν ***a='0'*** τότε η εντολή κάνει χρήση της Access RAM και η αντίστοιχη τιμή του BSR αγνοείται. Το πλεονέκτημα της Access RAM είναι ότι επιτρέπει στην εντολή να προσπελάζει δεδομένα σε ένα μόνο κύκλο(1Tcy), χωρίς να ανανεώνει το BSR.

III. Διακοπές (Interrupts)

Το PIC18F4550 ολοκληρωμένο κύκλωμα ελεγκτή έχει πολλαπλές πηγές διακοπών καθώς και δυνατότητα προτεραιότητας διακοπών (interrupt priority). Η θέση του high priority vector βρίσκεται στη διεύθυνση 00008h ενώ αυτή του low priority vector στη διεύθυνση 000018h της μνήμης εντολών. Στην περίπτωση που συμβεί κάποια υψηλής προτεραιότητας διακοπή, προκαλείται αναστολή της εξυπηρέτησης οποιασδήποτε χαμηλής διακοπής. Κάθε interrupt έχει τρία bits που ελέγχουν την εφαρμογή της:

1. Ένα flag bit που προειδοποιεί ότι μια διακοπή συνέβη
2. Ένα Enable bit που επιτρέπει στο πρόγραμμα να αναστέλλει την κανονική του ροή και να μεταβαίνει στο αντίστοιχο interrupt vector όταν μια διακοπή συμβαίνει
3. Ένα Priority bit για την επιλογή υψηλής ή χαμηλής προτεραιότητας

Τρεις καταχωρητές είναι υπεύθυνοι για τον έλεγχο των διακοπών.

- **RCON Register**

R/W-0	R/W-1	U-0	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	SBOREN	-	- RI	-TO	- PD	POR	BOR

Όταν IPEN=1, τότε η προτεραιότητα διακοπών επιλεχθεί.

- **INTCON Register**

R/W-0	R/W-1	U-0	R/W-1	R-1	R-1	R/W-0	R/W-0
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF

Όταν GIEH=1 τότε ενεργοποιούνται όλες οι διακοπές που έχουν το enable bit ίσο με '1' και ανήκουν σε αυτές της υψηλής προτεραιότητας. Όταν GIEL=1 τότε ενεργοποιούνται όλες οι διακοπές που έχουν το enable bit ίσο με '1' και ανήκουν σε αυτές της χαμηλής προτεραιότητας.

- IPR register που περιέχει δυνατότητα προτεραιότητας στις διακοπές περιφερειακών. Για την παρούσα εφαρμογή, σημαντικός είναι ο IPR1 που θέτοντας το RCIP=1 ενεργοποιείται η προτεραιότητα διακοπής του data receive μέσω της σειριακής.

IV. Σειριακή Επικοινωνία (EUSART)

Το PIC18F4550 ολοκληρωμένο κύκλωμα διαθέτει ενσωματωμένη EUSART (**E**nhanced **U**niversal **S**ynchronous **A**synchronous **R**eceiver **T**ransmitter) και

υποστηρίζει full-duplex ασύγχρονη επικοινωνία καθώς μπορεί να συνδεθεί και να ανταλλάξει δεδομένα με εξωτερικές περιφερειακές συσκευές, όπως υπολογιστές ή και CRT. Για τη ρύθμιση της σειριακής είναι υπεύθυνοι τρεις καταχωρητές: TXSTA, RCSTA και BAUDCON.

IV.1 Baud Rate Generator (BRG)

Ο BRG είναι ένας 8-bit ή 16-bit generator που υποστηρίζει τόσο την ασύγχρονη όσο και σύγχρονη επικοινωνία του περιφερειακού της EUSART. Το ζεύγος καταχωρητών **SPBRGH:SPBRG** ελέγχουν την περίοδο ενός ελεύθερου timer.
Όταν το σύστημα δουλεύει ασύγχρονα, τα bits **BRGH** (TXSTA<2>) και **BRG16** (BAUDCON<3>) χρησιμοποιούνται για τον έλεγχο του baud rate. Το BRGH καθορίζει το ρυθμό ανταλλαγής δεδομένων σε low και high. Το BRG16, ανάλογα με την τιμή του(1 ή 0) συμπεριλαμβάνει ή όχι την τιμή του καταχωρητή SPBRGH στον υπολογισμό του baud rate. Ο τύπος για τη ρύθμιση του baud rate είναι:

BR = Fosc / (option* ([SPBRGH: SPBRG] +1))

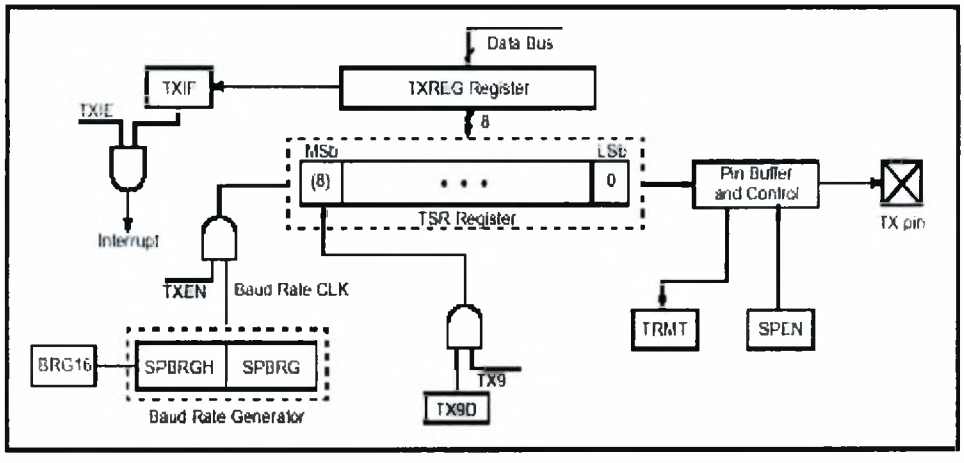
όπου option είναι ο αριθμός 64 ή 16 ανάλογα με την τιμή του BRGH (0 ή 1 αντίστοιχα). Για την παρούσα εφαρμογή, όπου το ρολόι λειτουργεί σε συχνότητα Fosc = 4 MHz, επιλέγεται τιμή για τον SPBRG=12 και το BD ρυθμίζεται στα 19.2Kbps. Η πραγματική τιμή του Baud Rate υπολογίζεται στα 19.231Kbps και έτσι το λάθος του πραγματικού με τον επιθυμητό ρυθμό μετάδοσης είναι 0.16%.

IV.2 Ασύγχρονη Μετάδοση (AT)

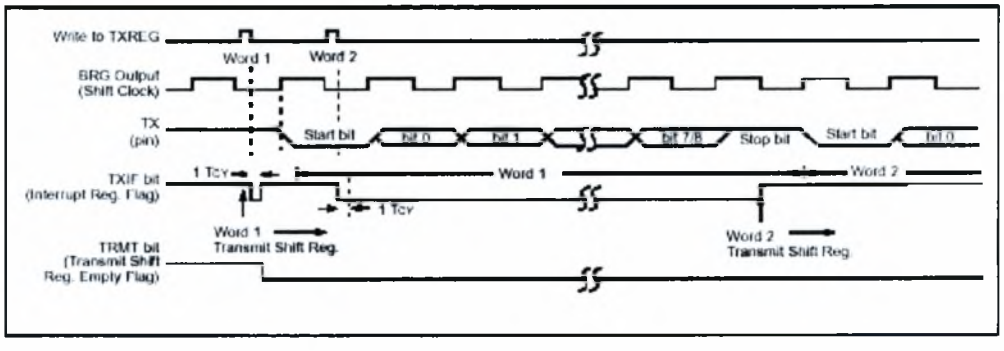
Η ασύγχρονη επικοινωνία επιτυγχάνεται θέτοντας το bit SYNC(<TXSTA>)=0. Η EUSART χρησιμοποιεί το standard NRZ (**N**on-**R**eturn-to-**Z**ero) και στέλνει 1 Start bit, 8 bits πληροφορίας και τέλος 1 Stop bit. Η σειριακή στέλνει και λαμβάνει το LSb πρώτα. Να τονιστεί ότι υπάρχει ανεξαρτησία μεταξύ της μετάδοσης και της λήψης δεδομένων. Η καρδιά του transmitter είναι ο TSR (Transmit Shift Register). Ο TSR λαμβάνει τα δεδομένα από τον TXREG. Ο TXREG φορτώνεται με δεδομένα από τον κώδικα της εφαρμογής. Ο TSR φορτώνεται από τον TXREG σε ένα μόνο κύκλο ρολογιού (1Tcy), όταν και το Stop bit των τελευταίων δεδομένων έχει ήδη μεταδοθεί. Όταν ο TXREG μείνει άδειος τότε το TXIF flag bit (PIR1<4>) γίνεται '1'. Αν το TXIE='1' τότε η διακοπή (interrupt) μετάδοσης ενεργοποιείται. Ακόμα και όταν το TXIE='0', το TXIF παίρνει την τιμή '1' και επανέρχεται στην αρχική του τιμή ('0') στον δεύτερο κύκλο (1Tcy) που ακολουθεί την εντολή φόρτωσης (load

instruction). Όταν ο TSR μείνει άδειος, τότε το bit TRMT (TXSTA<1>) γίνεται '1'.

Σχηματικό διάγραμμα υποσυστήματος μετάδοσης (Transmitter)



Ασύγχρονη Μετάδοση Δεδομένων

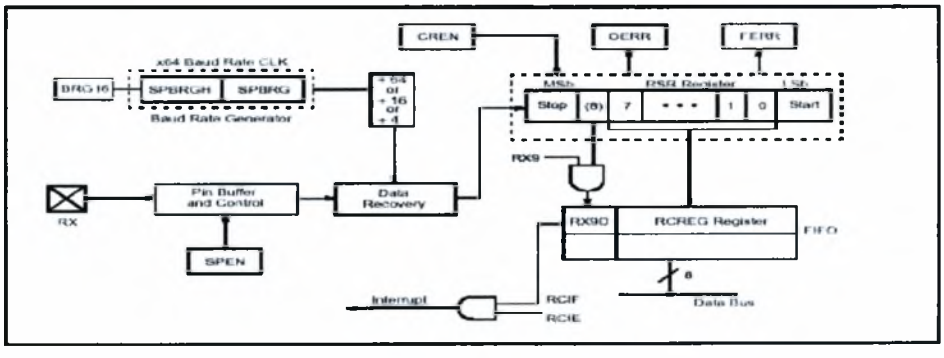


Όπως είναι φανερό, το TXIF bit παραμένει στο '0' αφού ο καταχωρητής TXREG παραμένει γεμάτος από δεδομένα μέχρι και το Stop bit της προηγούμενης λέξης δεδομένων μεταδοθεί.

IV.3 Ασύγχρονη Λήψη (AR)

Η αρχικοποίηση της λήψης δεδομένων είναι παρόμοια με αυτή της μετάδοσης. Αφού το baud rate ρυθμιστεί μέσω του SPBRG, το SYNC bit γίνεται '1'. Για την ενεργοποίηση της λήψης πρέπει το CREN='1'. Όταν η λήψη ολοκληρωθεί τότε το RCIF bit γίνεται '1'. Αν το RCIE='1' τότε μια διακοπή συμβαίνει. Τέλος τα δεδομένα διαβάζονται από τον RCREG και μέσω αυτού προωθούνται στο δίαυλο. Το σχηματικό διάγραμμα του υποσυστήματος λήψης φαίνεται παρακάτω.

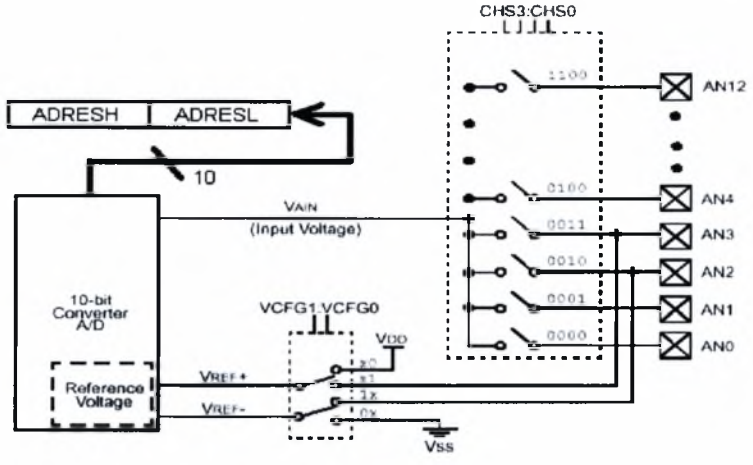
Σχηματικό διάγραμμα υποσυστήματος λήψης (Receiver)



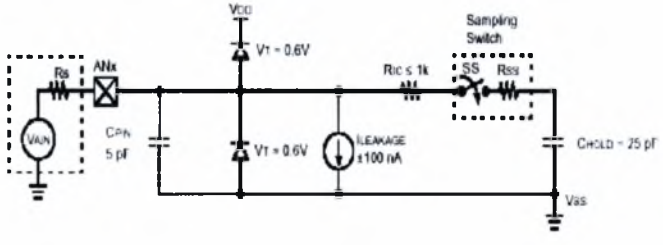
V. 10-Bit A/Ψ Μετατροπέας (ADC)

Το PIC18F4550 ολοκληρωμένο κύκλωμα μικροελεγκτή διαθέτει σαν περιφερειακό έναν 10-bit A/D SAR (Successive Approximation)
Converter με 13 εισόδους οι οποίες μπορούν να ρυθμιστούν σαν I/O Ports. Η λειτουργία του μετατροπέα ελέγχεται από τρεις καταχωρητές : ADCON0, ADCON1 και ADCON2. Το Vref του μετατροπέα μπορεί να ρυθμιστεί από τον κώδικα της εκάστοτε εφαρμογής μέσω των **VCFG1:VCFG0** (ADCON1<5, 4>). Η επιλογή των καναλιών εισόδου γίνεται στον ADCON0 (**CHS3:CHS0**). Ο A/D τίθεται σε λειτουργία με το GO/DONEbit='1'. Το αποτέλεσμα της μετατροπής του αναλογικού σήματος σε ψηφιακό αποθηκεύεται σε ένα ζεύγος καταχωρητών, τους **ADRESH:ADRESL**. Το σχηματικό διάγραμμα του ADC παρουσιάζεται παρακάτω.

Σχηματικό διάγραμμα υποσυστήματος A/D Converter



Το αναλογικό μοντέλο εισόδου το μετατροπέα δίνεται παρακάτω.



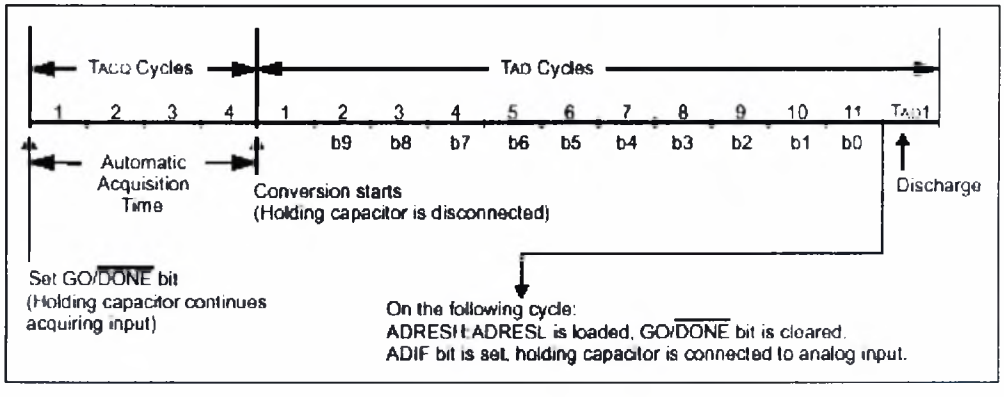
Για να πετύχει ο μετατροπέας τη μεγαλύτερη δυνατή ακρίβεια, θα πρέπει ο πυκνωτής να φορτιστεί πλήρως από το σήμα εισόδου. Οι αντιστάσεις Rs και Rss επηρεάζουν το χρόνο φόρτισης. Όταν το κανάλι εισόδου έχει επιλεχθεί, τότε απαιτείται ένα ελάχιστο χρονικό διάστημα πριν την έναρξη της μετατροπής (για τη φόρτιση του πυκνωτή). Αυτός ο χρόνος ονομάζεται Tacquisition και υπολογίζεται από τον τύπο : $\mathbf{T_{ACQ}} = T_{AMP} + T_C + T_{COFF}$
όπου T_{AMP} είναι το Settling time του ενισχυτή, T_C ο χρόνος φόρτισης του πυκνωτή και T_{COFF} ο παράγοντας θερμοκρασίας. Το $\mathbf{T_{ACQ}}$ ρυθμίζεται από τον καταχωρητή ADCON2 (**ACQT2: ACQT0**). Όταν το GO bit='1', τότε Ο μετατροπέας συνεχίζει να δειγματοληπτεί το κανάλι εισόδου για τον επιλεγμένο χρόνο $\mathbf{T_{ACQ}}$ και στη συνέχεια αρχίζει η πραγματική μετατροπή του σήματος σε ψηφιακό.

V.1 Επιλογή ρολογιού ADC

Ο χρόνος για τη μετατροπή ενός bit ορίζεται ως T_{AD} =1 / F_{AD}. Μία πλήρης μετατροπή απαιτεί 11 T_{AD} για τα 10 bits. Για την επίτευξη σωστών μετατροπών, το T_{AD} πρέπει να είναι όσο το δυνατόν μικρότερο, αλλά σίγουρα μεγαλύτερο από τη μικρότερη τιμή που ορίζει η τεχνολογία της συσκευής (minimum T_{AD}=1.4 usec).
Ο T_{AD} αποτελεί βασικό παράγοντα στη ρύθμιση του ανώτατου ορίου της συχνότητας στην οποία λειτουργεί ο μικροελεγκτής. Αν για παράδειγμα T_{AD}=2Tosc τότε η μέγιστη επιτρεπτή συχνότητα λειτουργίας είναι 1.43MHz.

V.2 Αναλογικές/ψηφιακές Μετατροπές

Ο χρόνος T_{ACQ} παίρνει τιμές από 2 έως 20 T_{AD}. Αν επιλεχθεί T_{ACQ} = 4T_{AD} τότε η αντίστοιχη μετατροπή παρουσιάζεται στο παρακάτω σχήμα.



Παρατηρούμε ότι η πραγματική μετατροπή αρχίζει από τη χρονική στιγμή που τελειώνει το T_{ACQ}. Όταν η μετατροπή ολοκληρωθεί, απαιτούνται ένα έως δύο T_{AD} για την αποφόρτιση του πυκνωτή.

Κεφάλαιο 3

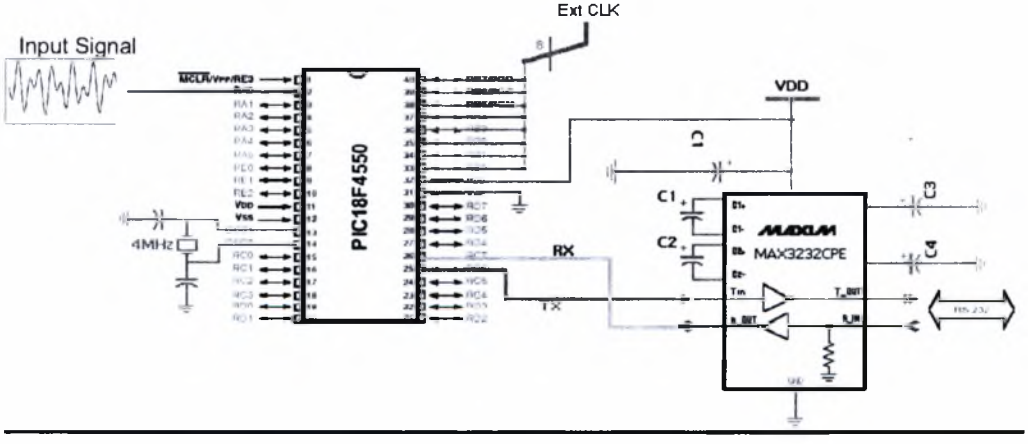
Περιγραφή Συστήματος

I. Υποσύστημα Μικροελεγκτή

I.1 Διανομή ρολογιού και RS-232 διασύνδεση

Πρώτο βήμα για την ορθή λειτουργία του συστήματος είναι η διανομή ρολογιού στον PIC. Το παρακάτω σχήμα παρουσιάζει την ενέργεια αυτή.

Σχηματικό διάγραμμα διανομης ρολογιού στον Μικροελεγκτή



Το σήμα εισόδου εισέρχεται στην RA0 η οποία αποτελεί και είσοδος για τον μετατροπέα A/D. Στις εισόδους OSC1, OSC2 συνδέεται κρύσταλλος ιδιοσυχνότητας 4MHz και δύο πυκνωτές 30pF ο καθένας. Το ενεργό κύκλωμα του ταλαντωτή βρίσκεται στο εσωτερικό του PIC. Το παραπάνω σχήμα παρουσιάζει και το ολοκληρωμένο κύκλωμα MAX3232CPE. Πρόκειται για έναν **voltage shifter** που δέχεται σαν είσοδο από τον μικροελεγκτή (τις εξόδους του TX και RX της EUSART) το λογικό '1' ή το '0' ηλεκτρικής στάθμης +5 και

0 Volt αντίστοιχα και σαν έξοδο βγάζει στάθμη -12 και +12 Volt αντίστοιχα. Το MAX3232 χρησιμοποιείται για τη σωστή λειτουργία της σειριακής θύρας. Για τη μετάδοση δεδομένων, η σειριακή χρησιμοποιεί το standard RS-232 που το λογικό '0' ορίζεται στα +12 Volt ενώ το λογικό '1' στα -12 Volt. Για τη λειτουργία του MAX προστίθενται 5 πυκνωτές χωρητικότητας 0.1μF. Το RS-232 interface χρησιμοποιείται από τον PIC για τη διασύνδεσή του με το RF module και την ασύρματη μετάδοση. Αξίζει να σημειωθεί πως η PORTB 8-bit θύρα του μικροελεγκτή έχει οριστεί σαν I/O digital port και αποτελεί είσοδο για το external clock το οποίο μας δίνει το timestamp κάθε φορά που ο buffer δεδομένων του PIC γεμίζει.

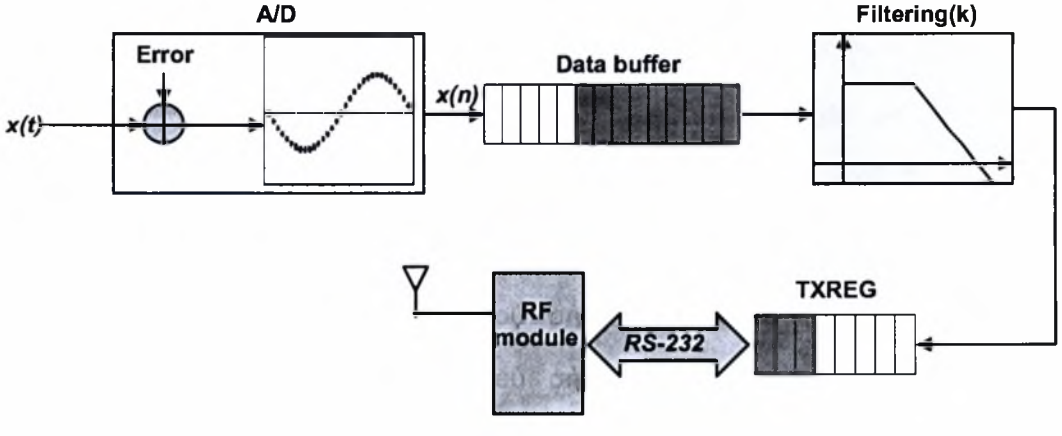
I.2 Λειτουργικότητα Μικροελεγκτή – Εσωτερική δομή

Το υποσύστημα του μικροελεγκτή είναι υπεύθυνο για :

1. Την μετατροπή του αναλογικού σήματος εισόδου (αισθητήρα) σε ψηφιακό
2. Την αποθήκευση του ψηφιακού σήματος σε περιοχή της μνήμης δεδομένων (**data buffer**)
3. Το φιλτράρισμα των δειγμάτων για την αποκοπή του θορύβου και
4. Την αποστολή των δεδομένων μέσω της σειριακής θύρας στο RF module για μετάδοση.

Το παρακάτω σχήμα αντικατοπτρίζει τη λειτουργικότητα του μικροελεγκτή.

Εσωτερική λειτουργικότητα Μικροελεγκτή



Η επικοινωνία του RF module με το μικροελεγκτή επιτυγχάνεται μέσω σειριακής θύρας (πρωτόκολλο RS-232) για τη μετάδοση των επεξεργασμένων από το φίλτρο ψηφιακών δεδομένων.

Ι.2.1 Λειτουργία A/D μετατροπέα και αποθήκευση στο buffer

Το ρολόι για τον A/D ορίζεται από τον κώδικα στο $F_{AD} = F_{OSC}/8 = 500KHz$.
Από την τιμή της συχνότητας του μετατροπέα καθορίζεται το $T_{AD} = 1/ F_{AD} = 2\mu s$. Η τιμή για το **Acquisition Time** ορίζεται στο $2T_{AD}$. Με βάση όσα έχουν αναφερθεί σχετικά με τις ρυθμίσεις το A/D και λαμβάνοντας υπόψη ότι Tosc = 0.25μs, κατά συνέπεια κάθε κύκλος εντολής υπολογίζεται στο 1μs, μια πλήρης μετατροπή και αποθήκευση της αντίστοιχης ψηφιακής τιμής στο data buffer κοστίζει 73μs. Το χρονικό αυτό διάστημα προκύπτει ως εξής :

- 30μs για μια πλήρη μετατροπή ($12\ T_{AD} + 2\ T_{AD} + 1\ T_{ADNEW} = 15T_{AD} = 30\mu s$)
- 43μs για την προσπέλαση της μνήμης (data buffer) και έλεγχο του βρόχου επανάληψης (Περισσότερα στο Κεφάλαιο 4).

Λαμβάνοντας υπόψη τα παραπάνω, η συχνότητα δειγματοληψίας του A/D Converter ορίζεται στα $\mathbf{F_S}$ = 1/73us = **13.698KHz**.
Η πληροφορία που περιέχει κάθε δείγμα είναι ένας αριθμός 10 bits. Από αυτό συμπεραίνουμε ότι ο A/D έχει 2^{10} (1024) επίπεδα τιμών καθένα από τα οποία έχει σχετικό «ύψος» στάθμης (V_{ref+} - V_{ref-}) /1024 Volt. Στη συγκεκριμένη εφαρμογή το V_{ref+} - V_{ref-} = 5 Volt, οπότε κάθε επίπεδο διαφέρει από το προηγούμενο και το επόμενο κατά 0.0048Volt.

Τη χρονική στιγμή που το ADCON0bits.GO γίνει '0', τα δεδομένα αποθηκεύονται στο ADRESH:ADRESL (10 bits) και είναι διαθέσιμα στον επόμενο κύκλο εντολής.
Τα δεδομένα αποθηκεύονται σε ένα χώρο μνήμης μεγέθους 1680 bytes. Επειδή κάθε δείγμα αποτελείται από πληροφορία των 10 bits, καταλαμβάνει μέγεθος απλού ακεραίου (**int**) και συνεπώς 2 bytes. Ο συνολικός αριθμός δειγμάτων που αποθηκεύεται στο data buffer και τελικά φιλτράρεται είναι 840 bytes. Ο buffer δεδομένων είναι στατικός και αμετάβλητος από παραμέτρους που δίνονται εξωτερικά από το χρήστη.
Ο συνολικός χρόνος που χρειάζεται ο data buffer για να γεμίσει δίνεται από τον γνωστό τύπο **T =N/Fs**. Γνωρίζοντας το μέγεθος του buffer καθώς και τη συχνότητα δειγματοληψίας έχουμε: **T = 61.32msec**. Ο χρόνος αυτός δε μεταβάλλεται (εκτός αν αλλάξει το Fs ή το μέγεθος του buffer).

Από το θεώρημα δειγματοληψίας του **Nyquist**, η μικρότερη συχνότητα δειγματοληψίας πρέπει να είναι τουλάχιστον ίση με το διπλάσιο της συχνότητας του σήματος που διεγματοληπτείται. Με άλλα λόγια πρέπει να ισχύει $\mathbf{F_S >= 2F_B}$.
Άρα η μέγιστη συχνότητα του σήματος που δειγματοληπτείται δε μπορεί να ξεπερνά τη συχνότητα $\mathbf{F_{Band} = 6.8KHz} = F_S/2$.

I.2.2 Φιλτράρισμα δεδομένων

Ο μικροελεγκτής, μετά και τη λήψη και των τελευταίων δειγμάτων από τον A/D, εφαρμόζει βαθυπερατό φίλτρο επί των αποθηκευμένων δεδομένων (δειγμάτων) με σκοπό την εξάλειψη του κβαντικού θορύβου που προκύπτει από τη μετατροπή του αναλογικού σήματος σε ψηφιακό αλλά και του θορύβου σήματος.

Το φίλτρο που υλοποιείται στο μικροελεγκτή αποτελεί ένα απλό **moving average FIR** (**F**inite **I**mpulse **R**esponse) βαθυπερατό φίλτρο και εκφράζεται με την παρακάτω ισότητα διαφορων στο πεδίο του χρόνου :

$$y(n) = \sum_{i=0}^{k-1} x(n-i)$$

Ο αντίστοιχος μετασχηματισμός **Z** του σήματος y(n), **Y(z)**, είναι ο ακόλουθος:

$$Y(z) = X(z)(1 + z^{-1} + z^{-2} + ... + z^{-(k-1)})$$

Η συνάρτηση μεταφοράς του συστήματος καθώς και η αντίστοιχη κλειστή μορφή δίνονται παρακάτω :

$$H(z) = \frac{Y(z)}{X(z)} = \frac{z^{k-1} + z^{k-2} + ... + z + 1}{z^{k-1}}$$

Η παράμετρος k, που δίνεται από το χρήστη, καθορίζει το βαθμό του φίλτρου και ανάλογα με την τιμή του, προκύπτουν τα αντίστοιχα Bode plots του μέτρου και της φάσης της συνάρτησης μεταφοράς ($H(e^{j\omega})$). Η αύξηση του k μειώνει τη ζώνη διέλευσης συχνοτήτων του βαθυπερατού, κρατώντας τη βασική μπάντα σήματος, και αποκόπτει τις το θόρυβο.

Για την εύρεση των δύο παραπάνω χαρακτηριστικών, πλάτους και φάσης, υπολογίζουμε το Διακριτό Μετασχηματισμό Fourier της H(z), $H(e^{j\omega})$, ο οποίος αποτελεί υποσύνολο του μετασχηματισμού **Z** ($z = e^{j\omega}$).

Η βασική συνθήκη για την ύπαρξη του DFT αποτελεί η σχέση

$$\sum_{n=-\infty}^{\infty} |h(n)| < \infty \quad \textbf{(1)}$$

Για την ύπαρξη ***BIBO ευστάθειας***, σημαντικό ρόλο παίζει η θέση των πόλων της συνάρτησης μεταφοράς του συστήματος, οι οποίοι πρέπει να βρίσκονται εντος του μοναδιαίου κύκλου. Απο τη συνάρτηση μεταφοράς, παρατηρείται ότι όλοι οι πόλοι βρίσκονται στο z=0 και άρα εντός του μοναδιαίου κύκλου. Αφού λοιπόν το σύστημα είναι BIBO ευσταθές και η σχέση (1) ισχύει, ο DFT υπάρχει και η αντίστοιχη συνάρτηση μεταφοράς παίρνει τη μορφή:

$$H(e^{j\omega}) = 1 + e^{-j\omega} + e^{-j2\omega} + \ldots + e^{-j\omega(k-1)} \Leftrightarrow$$

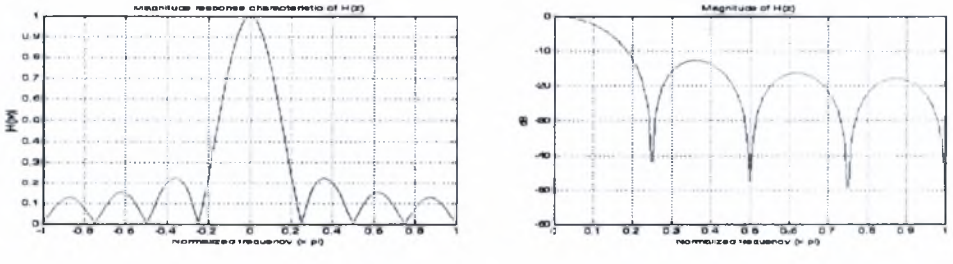
$$H(e^{j\omega}) = \frac{1 + e^{j\omega} + e^{j2\omega} + \ldots + e^{j\omega(k-1)}}{e^{j\omega(k-1)}}$$

Δύο απο τα βασικότερα χαρακτηριστικά του φίλτρου (εκτός της ύπαρξης των πόλων στο μηδέν) είναι:

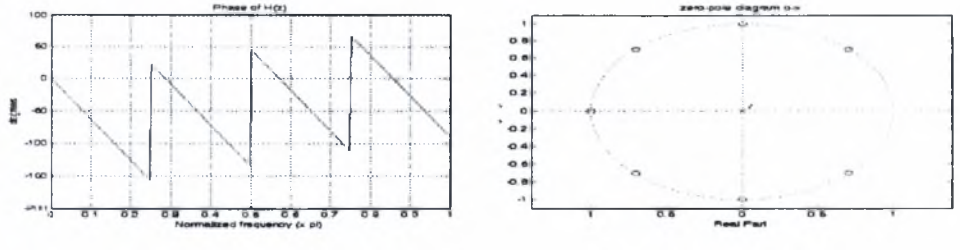
1. Το φίλτρο έχει τα μηδενικά του στις θέσεις $z = e^{j2\pi n/M}$, n=1,..,k-1
2. Η γραφική παράσταση του πλάτους της συνάρτησης μεταφοράς, $H(e^{j\omega})$, εμφανίζει μηδέν περιοδικά ως προς τη συχνότητα, στα σημεία $\omega_k = 2\pi k / M$

Οι χαρακτηριστικές του φίλτρου για μια τιμή του k=8 παρουσιάζονται στο παρακάτω σχήμα.

Χαρακτηριστικές Πλάτους – Φάσης για k=8



Zero-Pole διαγραμμά και διαγραμμά φάσης



I.2.3 Σειριακή επικοινωνία

Το υποσύστημα της σειριακής ρυθμίζεται σε Baud Rate = 19.2Kbps, θέτοντας την τιμή του SPBRG = 12. Ο υπολογισμός του Baud Rate γίνεται με τον τύπο **BR** = Fosc / (16* ([SPBRGH: SPBRG] +1)) = 4M /(16*(12+1)) = 19.231Kbps και το ποσοστό λάθους είναι 0.16%. Αλγοριθμικά, τα δεδομένα που φιλτράρονται κατά παράγοντα K, αποθηκεύονται σε μια μεταβλητή τύπου int και στέλνονται από τη σειριακή σε δυο φάσεις : πρώτα στέλνεται το Least significant Byte (LSB) και στη συνέχεια το Most significant byte (MSB). Ο χρόνος αποστολής ενός byte από τη σειριακή είναι σταθερός. Όταν ο data buffer σαρωθεί και όλα τα φιλτραρισμένα δείγματα σταλούν, η σειριακή στέλνει 2 bytes που εκφράζουν το time stamp της ολοκλήρωσης αποστολής δεδομένων.

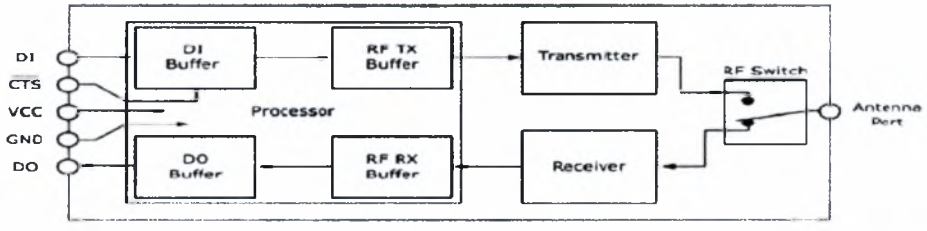
I.2.4 RF μετάδοση

Τα δεδομένα που φεύγουν από τη σειριακή στέλνονται στο RF module για την μετάδοσή τους. Το RF module αποτελεί το ένα εκ των δύο τμημάτων του RF bridge που υλοποιήθηκε. Πρόκειται για ένα σύστημα transmitter-receiver που ικανοποιεί τις προδιαγραφές του wireless **standard IEEE 802.15.4** και εκπέμπει σε συχνότητα 2.4GHz.
Το 802.15.4 πρωτόκολλο έχει σχεδιαστεί για χρήση σε εφαρμογές ασύρματων δικτύων αισθητήρων και εξασφαλίζει χαμηλό data rate (max 250Kbps) σε σχέση με άλλα πρωτόκολλα αλλά απλή συνδεσιμότητα και χαμηλή κατανάλωση ισχύος. Τέλος το πρωτόκολλο επιτρέπει point-to-point και point-to-multipoint επικοινωνία.
Το RF module λειτουργεί σε δυο modes :

- Transparent operation
- API operation

Για την παρούσα εφαρμογή, το transparent mode είναι ιδανικό καθώς επιτυγχάνεται λειτουργία των RF modules σαν ασύρματο σειριακό καλώδιο. Τα δεδομένα που φτάνουν στον **D**ata **I**nput buffer του module εκπέμπονται όταν ο χρόνος του **PO** (Packetized Timeout) λήξει ή όταν στον DI ξεπεραστεί το όριο των 100 χαρακτήρων. Σχεδιάγραμμα της δομής του RF module παρουσιάζεται παρακάτω.

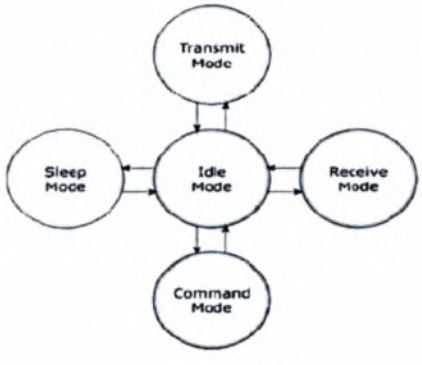
Δομη RF module και εσωτερικά blocks



I.2.5 Καταστάσεις RF module

Κάθε RF module διαθέτει ένα **Source address** και ένα **Destination address**, οι οποίες πρέπει να καθοριστούν πριν αρχίσει η μετάδοση μεταξύ των δύο κόμβων. Αυτό γίνεται μέσω **AT commands**. Με τα AT commands είναι δυνατή η αλλαγή των παραμέτρων του RF όπως το **MY** (source address) ή το **DL** (destination address Low). Για την είσοδο του module σε AT command mode στέλνεται αρχικά η ακολουθία **"+++"**. Αν το module απαντήσει OK τότε οι χαρακτήρες που δίνονται αντιμετωπίζονται σαν commands. Αν επιστρέψει ERROR τότε το module βρίσκεται σε data mode και οι εισερχόμενοι χαρακτήρες μεταδίδονται. Το RF μπορεί να βρεθεί σε μια από τις πέντε καταστάσεις, όπως δείχνει το σχήμα παρακάτω.

Πιθανές καταστάσεις RF module



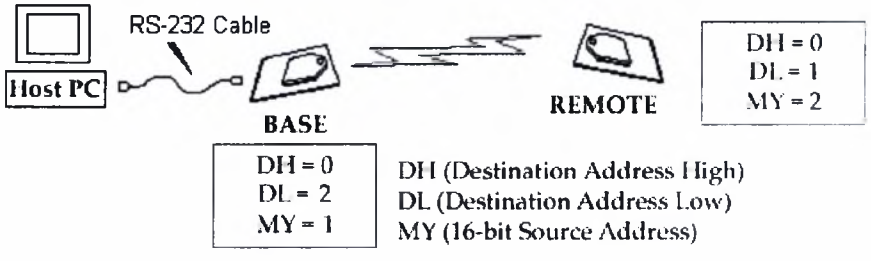
1. **Idle mode**: Κατάσταση στην οποία το module είναι ανενεργό
2. **Transmit-Receive mode (2)**: Κατάσταση στη οποία το module είναι ενεργό και μεταδίδει ή δέχεται RF πακέτα. Κάθε RF πακέτο περιέχει τις MY και DH:DL διευθύνσεις, το μήκος των οποίων είναι 16 ή 64 bits.
3. **Sleep mode**: Το module βρίσκεται σε κατάσταση πολύ χαμηλής κατανάλωσης ισχύος και παραμένει ανενεργό.
4. **AT command mode**: Ρύθμιση παραμέτρων.

Ο μικροελεγκτής, μπαίνοντας σε λειτουργία και εκτελώντας τον κώδικα εφαρμογής ρυθμίζει το δικό του RF module μέσω AT commands ανεξάρτητα από τα υπόλοιπα ενεργά τμήματα του συστήματος (host PC). Αυτή η δυνατότητα ανεξαρτησίας αποτελεί ισχυρό πλεονέκτημα για την εφαρμογή καθώς ο μικροελεγκτής μπορεί να μεταδώσει δεδομένα και σε άλλους κόμβους αποφασίζοντας ο ίδιος τον προορισμό αλλά και τη διάρκεια της μετάδοσης.

II. Υποσύστημα host PC

Το host PC αποτελεί το δεύτερο σκέλος του συστήματος και είναι συνδεδεμένο μέσω σειριακής με ένα RF module. Το host PC εκτελεί κώδικα MATLAB και ρυθμίζει την λειτουργία του RF με AT commands. Αρχικά δημιουργεί ένα αντικείμενο που αναφέρεται στη σειριακή θύρα επικοινωνίας (Serial Object) και θέτει το Baud Rate σε τιμή 19.2Kbps. Στη συνέχεια θέτει το RF module μέσω σειριακής σε AT command mode και καθορίζει το MY (Source address) και το DH:DL (destination address). Το RF επιστρέφει σε idle mode και η επικοινωνία είναι έτοιμη να αρχίσει. Το σχεδιάγραμμα ρυθμισμένου συστήματος για επικοινωνία παρουσιάζεται στο παρακάτω σχήμα.

Ρυθμισμένο σύστημα επικοινωνίας hostPC - MCU



Για την αποφυγή ατέρμονης εκτέλεσης του κώδικα του host όσο και την μετάδοση δεδομένων, το host PC στέλνει στο μικροελεγκτή ένα κωδικό της μορφής '***bxy*** ', με x και y τους χαρακτήρες από 1 έως 9, δηλώνοντας την επιθυμία για λήψη συγκεκριμένου αριθμού δεδομένων. Ο χαρακτήρας x αποτελεί τον παράγοντα k του φιλτραρίσματος που γίνεται στο μικροελεγκτή ενώ ο y υποδηλώνει τον αριθμό των δεδομένων σε bytes που ο host επιθυμεί να δεχτεί (840*y bytes). Όταν μικροελεγκτής συμπληρώσει τον αριθμό δεδομένων, σταματά τη μετάδοση και περιμένει για νέα αίτηση για δεδομένα. Και στις δύο πλευρές, τα συνδεδεμένα RF modules επιστρέφουν σε κατάσταση χαμηλής κατανάλωσης ισχύος (Sleep mode). Το host PC, αφού αποθηκεύσει τα δεδομένα σε αρχείο για περαιτέρω επεξεργασία, κλείνει τη σειριακή και καταστρέφει το serial object.

Κεφάλαιο 4

Υλοποίηση εφαρμογής

Στο παρόν κεφάλαιο παρουσιάζεται η ανάλυση του κώδικα των δύο υποσυστημάτων, του μικροελεγκτή και του host PC.

I. Υποσύστημα Μικροελεγκτή

Αρχείο **PIC.c**

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <string.h>
4. #include <math.h>
5. #include <adc.h>
6. #include <usart.h>
7. #include <delays.h>
8. #include <p18f4550.h>

// PIC18F4550 Configuration Settings
10. #pragma config FOSC = XT_XT
11. #pragma config FCMEM = OFF
12. #pragma config IESO = ON
13. #pragma config PWRT = OFF
14. #pragma config BOR = OFF
15. #pragma config VREGEN = OFF
16. #pragma config WDT = OFF
17. #pragma config MCLRE = OFF
18. #pragma config LPT1OSC = OFF
19. #pragma config PBADEN = OFF
20. #pragma config STVREN = ON
21. #pragma config LVP = OFF
22. #pragma config ICPRT = OFF
23. #pragma config XINST = OFF
24. #pragma config DEBUG = OFF
25. #pragma config CP0 = OFF
26. #pragma config CP1 = OFF
27. #pragma config CP2 = OFF
28. #pragma config CP3 = OFF
```

```
#pragma config CPB = OFF
#pragma config CPD = OFF
#pragma config WRT0 = OFF
#pragma config WRT1 = OFF
#pragma config WRT2 = OFF
#pragma config WRT3 = OFF
#pragma config WRTB = OFF

#define SIZE 1680

/*********Functions*********/
void main(void);
void init_adc_peripheral(void);
void init_USART_peripheral(unsigned char INT_mask);
void run_adc(void);
void simple_filter(unsigned char f);
void rcv_handler(void);
void delay(void);
void ZigBee_init(void);


/**********Global variables***********/
/*We declare these variables as volatile because
 they are accessed by both an ISR and mainline functions*/

volatile unsigned char filt;
volatile int enable = 0;


/*****************udata section*************/
#pragma udata buf_scn = 0x100
static unsigned char buffer[SIZE];
#pragma udata


/*****************High Priority Interrupt
section*************************/
#pragma code rcv_interrupt = 0x8

void rx_int(void){

_asm goto rcv_handler _endasm

}
#pragma code
```

```
/**********************ISR's
implementation*********************************/
74. #pragma interrupt rcv_handler

void rcv_handler(void){

        char rcv[3];

        getsUSART(&rcv[0],3);

        if(rcv[0] == 'b') {
                if( rcv[1]>'0' && rcv[1]<':' ){
                        enable=1;
                        filt = (rcv[1]-0x30);
                                times=(rcv[2]-0x30);
                }
        }
        PIR1bits.RCIF = 0;
}


/**********************Functions' implementation****************/
89. void init_adc_peripheral(void){

91.     TRISB=0xFF;
92.     TRISA = 0x01;

94.     OpenADC( ADC_FOSC_8 &
95.             ADC_RIGHT_JUST &
96.             ADC_2_TAD,
97.             ADC_CH0 &
98.             ADC_INT_OFF,
99.             14
        );

102.    ADCON1bits.VCFG1 = 0;
103.    ADCON1bits.VCFG0 = 0;

105.     return;
}

108. void init_USART_peripheral(unsigned char INT_mask){

110.    BAUDCONbits.BRG16 = 0;
111.    RCSTAbits.SPEN = 1;
112.    TRISCbits.TRISC6 = 0;
113.    TRISCbits.TRISC7 = 1;
```

```
115.  OpenUSART(USART_TX_INT_OFF &
116.                INT_mask &
117.                      USART_ASYNCH_MODE &
118.                 USART_EIGHT_BIT &
119.                 USART_CONT_RX &
120.                 USART_BRGH_HIGH,
121.                 12
      );

124.  return;
}


128. void ZigBee_init( void ){

130.  char a ='+';
131.  char src_adr[ ]="ATMY2";
132.  char dest_adr_L[ ]="ATDL1";
133.        char store_non_vol[ ]="ATWR";
134.  char exit_AT[ ]="ATCN";
135.  char resp1[5];
136.        char resp2[5];

138.        int i;

140.        Delay10KTCYx(150);
141.  delay();

143.        WriteUSART(a);
      while(BusyUSART());
      for(i=0;i<100;i++);
      WriteUSART(a);
      while(BusyUSART());
      for(i=0;i<100;i++);
      WriteUSART(a);
      while(BusyUSART());
151.        for(i=0;i<100;i++);

153.  getsUSART(resp1,2);
154.  delay();
155.        delay();
156.        delay();

158.        putsUSART(src_adr);
      while(BusyUSART());
      WriteUSART(0x0D);
      while(BusyUSART());
162.        delay();
```

```
164.  putsUSART(dest_adr_L);
165.  while(BusyUSART());
166.        WriteUSART(0x0D);
167.        while(BusyUSART());
168.        delay();

170.  putsUSART(store_non_vol);
      while(BusyUSART());
      WriteUSART(0x0D);
      while(BusyUSART());
      delay();

176.  putsUSART(exit_AT);
      while(BusyUSART());
      WriteUSART(0x0D);
      while(BusyUSART());
180.  delay();

182.  CloseUSART();

184.  return;
}


188. void delay(void){
189.  int i;
190.        for(i=0;i<200;i++){
            Delay1KTCYx(5);
      }
}


196. void run_adc( void ){

      int i = 0;

      while( i<SIZE ){
201.        ADCON0bits.GO = 1;
202.        while( ADCON0bits.GO );
203.        Nop();
204.        buffer[i] = ADRESL;
205.        i++;
206.        buffer[i] = ADRESH;
207.        i++;
      }
209.        return;
}
```

```
211. void simple_filter(unsigned char f){

        int i, p;
        int sum;
        unsigned char tmp, j;
        int count = 0;

218.    for(i=0; i<SIZE;  ){
            sum=0;
            i=count;

222.        for(j=0; j<f; j++){
                tmp=buffer[i];
                i++;
                p=buffer[i];
                p=p<<8;
                sum=sum+p+tmp;
                i++;
            }
230.        TXREG=(unsigned char)sum;
            tmp=sum>>8;
            TXREG=tmp;
            while(!TXSTAbits.TRMT);
234.        count=count+2;
        }

237.    TXREG=PORTB;
238.    Nop();
239.    TXREG=PORTB;
240.    while(!TXSTAbits.TRMT);

242.    return;
}


246. void main( void ){

        char counter;
        delay( );

        init_adc_peripheral( );
        init_USART_peripheral( USART_RX_INT_OFF );
        ZigBee_init( );
        delay();

        init_USART_peripheral( USART_RX_INT_ON );
```

```
 RCONbits.IPEN = 1;  /* Enable interrupt priority */
IPR1bits.RCIP = 1;  /* Make receive interrupt high priority */
INTCONbits.GIEH = 1; /* Enable all high priority interrupts */

counter=0;

while(1){

        if(enable){
                if(counter<times){
                        run_adc( );
                        simple_filter(filt);
                        counter++;
                }
                        else{
                        enable=0;
                        counter=0;
                        times=0;
                }
        }
}

}
```

Επεξήγηση Κώδικα

1-8. Ο κώδικας αρχίζει με τις δηλώσεις των βιβλιοθηκών συναρτήσεων που χρησιμοποιούνται από το πρόγραμμα.

10–35. Στο σημείο αυτό δηλώνονται τα #pragma directives. Ένα pragma directive χρησιμοποιείται για να διαχειριστεί:

- Τη μνήμη εντολών (**#pragma code, #pragma romdata**)
- Τη μνήμη δεδομένων (**#pragma idata, #pragma udata**)
- Τις διακοπές (**#pragma interrupt**)
- Configuration bits (**#pragma config**)

Ένα pragma directive αλλάζει την τρέχουσα περιοχή (section) μέσα στην οποία ο compiler δεσμεύει πληροφορίες του αντίστοιχου τύπου. Μια περιοχή (section) είναι ένα τμήμα της εφαρμογής που βρίσκεται σε ένα συγκεκριμένο μέρος της μνήμης και περιλαμβάνει εντολές ή δεδομένα. Σε αυτή την περίπτωση, η δήλωση **#pragma config** προσδιορίζει τα configuration bits του PIC (ξεχωριστά για κάθε PIC). Η πρώτη δήλωση #pragma config = XT_XT δηλώνει στον PIC ότι το ρολόι είναι εξωτερικό και προέρχεται από κρύσταλλο.

37. Δήλωση #define μεταβλητής (το μήκος του buffer σε bytes)

40-47. Δήλωση των υπογραφών των συναρτήσεων της εφαρμογής

54-56. Δήλωση δύο καθολικών μεταβλητών ως **volatile.** Το αναγνωριστικό volatile υποδηλώνει ότι οι μεταβλητές είναι προσπελάσιμες τόσο από τις κοινές συναρτήσεις του προγράμματος όσο και από τη ρουτίνα εξυπηρέτησης του interrupt handler (ISR).

59-61. Η δήλωση #pragma udata αλλάζει το τρέχον περιοχή (default section) του προγράμματος και δημιουργεί το buf_scn section στην απόλυτη διεύθυνση μνήμης 0x100 της μνήμης δεδομένων (data memory). Στη θέση αυτή θα τοποθετηθούν μη αρχικοποιημένα δεδομένα. Λεπτομερέστερα, ο χώρος αυτός της μνήμης καταλαμβάνει 1680 bytes και αποτελεί το **data buffer** αποθήκευσης των δειγμάτων του A/D. Επειδή το μέγεθος του buffer ξεπερνά κατά πολύ το μέγεθος ενός memory bank, είναι αναγκαία η αλλαγή του linker script για τη δημιουργία ενός τόσο μεγάλου αντικειμένου στο χώρο της μνήμης. Παρακάτω παρατίθενται το αρχικό και το τελικό αρχείο του linker script.

Αρχικό linker script

```
ACCESSBANK NAME=accessram START=0x0      END=0x5F
DATABANK   NAME=gpr0      START=0x60     END=0xFF
DATABANK   NAME=gpr1      START=0x100    END=0x1FF
DATABANK   NAME=gpr2      START=0x200    END=0x2FF
DATABANK   NAME=gpr3      START=0x300    END=0x3FF
DATABANK   NAME=usb4      START=0x400    END=0x4FF   PROTECTED
DATABANK   NAME=usb5      START=0x500    END=0x5FF   PROTECTED
DATABANK   NAME=usb6      START=0x600    END=0x6FF   PROTECTED
DATABANK  NAME=usb7     START=0x700      END=0x7FF   PROTECTED
ACCESSBANK NAME=accesssfr START=0xF60    END=0xFFF   PROTECTED

SECTION    NAME=CONFIG    ROM=config
STACK SIZE=0x100 RAM=gpr3
```

Τελικό linker script

```
ACCESSBANK NAME=accessram  START=0x0    END=0x5F
DATABANK   NAME=gpr0       START=0x60   END=0xFF
DATABANK   NAME=big        START=0x100  END=0x790  PROTECTED
DATABANK   NAME=gpr2       START=0x791  END=0x792
DATABANK   NAME=gpr3       START=0x793  END=0x794
DATABANK   NAME=usb4       START=0x795  END=0x796
DATABANK   NAME=usb5       START=0x797  END=0x798
DATABANK   NAME=usb6       START=0x799  END=0x7A0
DATABANK   NAME=usb7       START=0x7A1  END=0x7FF
ACCESSBANK NAME=accesssfr  START=0xF60  END=0xFFF

SECTION    NAME=CONFIG    ROM=config
SECTION    NAME=buf_scn   RAM=big

STACK SIZE=0x9F RAM=gpr0
```

Παρατηρούμε στην γραμμή 3 του τελικού αρχείου τη μεταβλητή big και το χώρο που καταλαμβάνει (0x790 – 0x100 =0x690h = 1680 bytes (decimal)). Η περιοχή ορίζεται ως PROTECTED. Τέλος να αναφερθεί ότι ο πίνακας που δηλώνεται στο buf_scn section έχει το αναγνωριστικό **static** υποδηλώνοντας ότι βρίσκεται στην περιοχή των καθολικών μεταβλητών του προγράμματος και η προσπέλαση γίνεται άμεσα (χωρίς τη χρήση κάποιου δείκτη).

65-69. Δηλώνεται ένα code section στη διεύθυνση 0x8h. Η διεύθυνση αυτή αποτελεί τη θέση του high interrupt vector. Όταν κάποια διακοπή συμβεί, ο PC μεταβαίνει σε αυτή τη διεύθυνση. Η ISR της διακοπής δεν τοποθετείται σε αυτή τη διεύθυνση. Αυτός είναι και ο λόγος που ο κώδικας που βρίσκεται στη θέση 0x08h είναι μια εντολή GOTO για μεταφορά του ελέγχου στο αντίστοιχο ISR.

74. Ένα **#pragma interrupt** δηλώνει μια συνάρτηση ως high priority ISR (**I**nterrupt **S**ervice **R**outine). Η παρούσα ISR εκτελείται όταν ο μικροελεγκτής λαμβάνει δεδομένα από τη σειριακή(USART Reception). Πιο αναλυτικά, περιμένει μια κωδικοποιημένη εντολή από τον host, ***bxy***, για να αρχίσει η δειγματοληψία του αναλογικού σήματος.

89-105. Ρουτίνα ρύθμισης των παραμέτρων του A/D. Κάθε I/O port διαθέτει ένα καταχωρητή που καθορίζει εάν το port pin αποτελεί είσοδο ή έξοδο. Για το PORTB, όλα τα pins είναι είσοδοι (είσοδοι για το external clock). Ο A/D έχει μόνο μια είσοδο, την RA0. Η συχνότητα του ρολογιού του μετατροπέα καθορίζεται στα 500KHz (ADC_FOSC_8), ενώ του Tacquisition ορίζεται στα 2μs. Το interrupt ολοκλήρωσης μετατροπής απενεργοποιείται και σαν Vref επιλέγεται το VDD-VSS = 5-0=5 Volt.

108 - 124. Αρχικοποίηση της USART. Ορίζεται το pin TX ως έξοδος και το RX ως είσοδος. Η λειτουργία της σειριακή ορίζεται ως ασύγχρονη (SYNC=0). Η διακοπή (transmission interrupt) αποστολής απενεργοποιείται ενώ η αντίστοιχη λήψης καθορίζεται από την παράμετρο INT_mask. Το μέγεθος κάθε πακέτου δεδομένων ορίζεται στο ένα byte και η τιμή του SPBRG είναι 12.

128-184. Αρχικοποίηση του RF module. Ο μικροελεγκτής στέλνει τη συμβολοσειρά '+++' και θέτει το RF σε AT command mode. Στη συνέχεια ορίζει τις source (MY) και destination (DL) address αποθηκεύοντάς τες στη μνήμη του module. Τέλος, το RF module επιστρέφει σε idle mode.

196-209. Ρουτίνα λειτουργίας του Converter. Μετατροπή του σήματος και αποθήκευση των δειγμάτων στη μνήμη. Ο χρόνος για μια πλήρη μετατροπή και αποθήκευση υπολογίζεται σε 73μs και η συχνότητα δειγματοληψίας σε Fs=1/73 μs=13.7 kHz.

211-242. Ρουτίνα υλοποίησης του ψηφιακού φίλτρου και αποστολή των επεξεργασμένων δεδομένων στη σειριακή. Ο data buffer σαρώνεται και

προστίθενται Κ δείγματα κάθε φορά. Επειδή κάθε δείγμα καταλαμβάνει χώρο ακεραίου (2 bytes) και γνωρίζονται ότι ο data buffer είναι τύπου **unsigned char** (δηλώθηκε έτσι για να κοστίζει λιγότερους κύκλους η προσπέλασή του σε σχέση με τον **int**), ο δείκτης θέσης στον data buffer προχωρά κατά δύο για να προσπελαστεί ένα δείγμα. Όταν η πρώτη Κ-αδα δειγμάτων προστεθεί, στέλνεται μέσω της σειριακής στο RF για μετάδοση. Η αποστολή του αποτελέσματος στη σειριακή γίνεται σε δυο φάσεις, καθώς το LSB αποστέλλεται πρώτο και το MSB δεύτερο. Για τον υπολογισμό της επόμενης Κ-αδας, ο δείκτης θέσης προχωρά κατά 2, αφήνοντας το πρώτο σε σειρά δείγμα και συμπεριλαμβάνοντας στον υπολογισμό το Κ+1 δείγμα.
Όταν η αποστολή και του τελευταίου αποτελέσματος ολοκληρωθεί, αποστέλλεται το Time Stamp ολοκλήρωσης αποστολής (2 bytes).

246. Η ρουτίνα της **main**. Αρχικά η main αρχικοποιεί τα περιφερειακά, πρώτα τον A/D και στη συνέχεια τη σειριακή. Αξίζει να σημειωθεί ότι η σειριακή κάνει **mask** τη διακοπή της λήψης (απενεργοποίηση). Αυτό γίνεται για την αρχικοποίηση του RF module καθώς ο PIC πρέπει να περιμένει τις κατάλληλες απαντήσεις απο το RF module (reply OK). Στη συνέχεια γίνεται **unmask** η διακοπή λήψης και το σύστημα είναι έτοιμο να δεχτεί τους κωδικοποιημένους χαρακτήρες από τον host. Στις γραμμές 257 έως και 260, θέτουμε τα bits που σχετίζονται με το interrupt reception. Θέτουμε:

- Το bit ενεργοποίησης προτεραιότητας διακοπής **(IPEN=1)**
- Το bit ενεργοποίησης της διακοπής λήψης (**interrupt reception**) σαν υψηλής προτεραιότητας διακοπή
- Το bit ενεργοποίησης όλων των υψηλής προτεραιότητας διακοπών

II. Υποσύστημα host PC

Ο κώδικας του host PC παρουσιάζεται παρακάτω

Αρχείο **ATcomm.m**

```
function T = ATcomm(src_adr,string)

%.....Serial port configuration.....
sobj =
serial('COM1','BaudRate',19200,'Databits',8,'Parity','none','Stopbits',1);
sobj.TimeOut = 10;
sobj.InputBufferSize = 30000;
sobj.ReadAsyncMode = 'continuous';
sobj.Terminator='CR';

%.....Connection with serial buffer.....
```

```
fd = fopen('serial_data.txt','w+');
if(fd == -1)
        fprintf('File not opened...\n\n');
        return;
end;

fopen(sobj);
fprintf('Connection established...\n');
%+++
fprintf(sobj,'%s','+++');
[xbee,c,mes]=fscanf(sobj,'%s');
fprintf('\nAT command mode Response : %s',xbee);

%ATDL <Enter>
fprintf(sobj,'%s\n','ATDL2');
[xbee,c,mes]=fscanf(sobj,'%s');
fprintf('\nSet destination address(DL)')
fprintf('\nResponse : %s',xbee);

%ASK NEW ATDL
fprintf(sobj,'%s\n','ATDL');
[xbee,c,mes]=fscanf(sobj,'%s');
fprintf('\nNew Destination address Low: %s',xbee);

%ATDH
fprintf(sobj,'%s\n','ATDH0');
[xbee,c,mes]=fscanf(sobj,'%s');
fprintf('\nSet DH : %s',xbee);

%Ask for new ATDH
fprintf(sobj,'%s\n','ATDH');
[xbee,c,mes]=fscanf(sobj,'%s');
fprintf('\nNew Destination address High: %s',xbee);

%Source address MY
fprintf(sobj,'%s\n',src_adr);
[xbee,c,mes]=fscanf(sobj,'%s');
fprintf('\nSet Source address');
fprintf('\nResponse : %s',xbee);

fprintf(sobj,'%s\n','ATMY');
[xbee,c,mes]=fscanf(sobj,'%s');
fprintf('\nSource address: %s',xbee);

%ATWR
fprintf(sobj,'%s\n','ATWR');
[xbee,c,mes]=fscanf(sobj,'%s');
fprintf('\nStore Response : %s',xbee);
```

```
%ATCN <Enter>
fprintf(sobj,'%s\n','ATCN');
[xbee,c,mes]=fscanf(sobj,'%s');
fprintf('\nExit Response : %s',xbee);

%*************Read data from PIC************
data_sample=840;
fprintf(sobj,'%s',string);

[A,count,msg] = fread(sobj, (string(3)-48)*(data_sample-(string(2)-48)+1),
'uint16');

fprintf('\nData read from serial (in bytes) : \n%d\n',2*count);
disp(msg);
fprintf(fd,'%d\n',A);
```

Ο παραπάνω κώδικας αρχικοποιεί το RF module με AT commands και λαμβάνει δεδομένα από τον μικροελεγκτή. Τα δεδομένα αυτά αποθηκεύονται σε αρχείο για επεξεργασία και εξαγωγή γραφικών παραστάσεων.
Το επόμενο αρχείο διαβάζει τα δεδομένα και σχεδιάζει τις γραφικές παραστάσεις.

Αρχείο **plotdata.m**

```
%Read PIC data from file
function T = plotdata(x_axis, y_axis, file_name, scale, filtering);

fd = fopen(file_name,'r');
if(fd == -1)
        fprintf('File not opened...\n\n');
        return;
end;

[a,cnt] = fscanf(fd, '%d', inf);
a = a';
MAX=max(a);
fclose(fd);

count = 0;
Ts = 0.000073;
Vref = 5;
T=zeros(1,cnt);
Voltage=zeros(1,cnt);

for n=1:1:cnt
        Voltage(1,n) = ((Vref*a(1,n))/(1023*filtering));
        T(1,n)=count*scale*Ts;
```

```
        count = count+1;
end;

fprintf('Data read from file = %d \n',cnt);
fprintf('Maximum value is %d\n',MAX);
plot(T,Voltage,'r');
axis([x_axis, y_axis, -1, 6]);
xlabel('time (sec)');
ylabel('Voltage (Volt)');
grid;
```

Κεφάλαιο 5

Μετρήσεις – Αποτελέσματα

Σε αυτό το κεφάλαιο παρατίθενται τα αποτελέσματα των μετρήσεων και οι γραφικές παραστάσεις των FIR ψηφιακών φίλτρων. Η τιμή του k, όπως προαναφέρθηκε, καθορίζει το βαθμό του φίλτρου και αποκόπτει το θόρυβο του σήματος εισόδου (*generator noise*).

I. Κβαντικός θόρυβος

Το περιφερειακό A/D του μικροελεγκτή, για να πετύχει τη μέγιστη δυνατή ακρίβεια των αποτελεσμάτων του, θεωρεί ως μέγιστο λάθος– απόκλιση- το ½ LSb. Το maximum επιτρεπτό λάθος, $e_q(n)$, βρίσκεται στο κλειστό διάστημα $\left[-\frac{\Delta}{2},\frac{\Delta}{2}\right]$, όπου Δ είναι το quantization step size. Αφού η μέγιστη αριθμητική τιμή που εξέρχεται του μετατροπέα είναι ο αριθμός 1023, ενώ η ελάχιστη το μηδέν, ο αριθμός των quantization levels 1024. Άρα κάθε επίπεδο διαφέρει από το προηγούμενο και το επόμενο κατά $\frac{V_{ref}}{1024-1}$ =5/1023=0.0049 Volt.
Παίρνοντας σα δεδομένο ότι ο θόρυβος είναι ομοιόμορφα κατανεμημένος στο $\left[-\frac{\Delta}{2},\frac{\Delta}{2}\right]$, όπως δείχνει και το παρακάτω σχήμα, μπορούμε να υπολογίσουμε το **SQNR** (**S**ignal to **N**oise **Q**uantization **R**atio) που ισούται με

$$SQNR = 10\log\frac{P_x}{P_n}$$

όπου P_x είναι η ισχύς του σήματος εισόδου και το P_n αποτελεί την ισχύ του θορύβου. Ο υπολογισμός του P_n παρουσιάζεται παρακάτω.

$$P_n = \frac{1}{\Delta}\int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} e^2 p(e)de = \frac{\Delta^2}{12}$$

Με δεδομένο το Δ που υπολογίστηκε παραπάνω η ισχύς θορύβου καθορίζεται στην τιμή των P_n = 1.9907.

Το σήμα εισόδου είναι της μορφής $x_a(t) = A\cos\omega_0 t$ και η αντίστοιχη ισχύς υπολογίζεται απο τον τύπο

$$P_x = \frac{1}{T_p}\int_0^{T_p}(A\cos\omega_0 t)^2\,dt = \frac{A^2}{2}.$$

Το Α αποτελεί το μέτρο του ημιτόνου εισόδου και καθορίζεται στα 5 Volt. Άρα η συνολική ισχύς του σήματος εισόδου υπολογίζεται στα P_x=12.5 Volt. Ο συνολικός SNQR υπολογίζεται σε 67.74 dB.

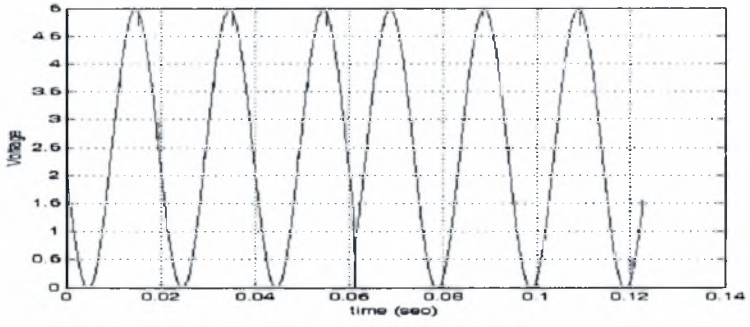
II. Αποκρίσεις φίλτρων

Στην ενότητα αυτή, παρουσιάζονται οι αποκρίσεις πλάτους και φάσης, καθώς και η χρονική απόκριση του συστήματος. Η τιμή του πλάτους εισόδου ορίζεται στα 5 Volt και οι συχνότητες του σήματος στα 50 και 1KHz αντίστοιχα. Οι συχνότητες επιλέγονται σε αυτές τις τιμές ώστε να γίνονται ευδιάκριτες οι ζώνες διέλευσης και αποκοπής συχνοτήτων των ψηφιακών φίλτρων. Αρχικά παρουσιάζονται οι αποκρίσεις στα 50Hz και στη συνέχεια οι αντίστοιχες στο 1KHz.

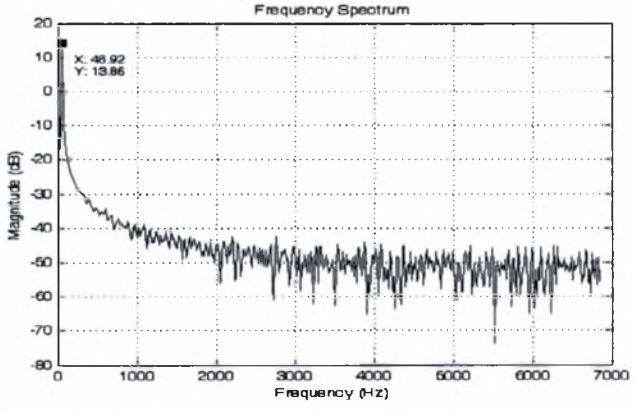
II.1 Φίλτρο μηδενικού βαθμού (k=1, 50Hz)

Στην περίπτωση που το k=1 τότε ουσιαστικά δεν εφαρμόζεται κάποιο φιλτράρισμα και τα δεδομένα μεταφέρονται στη σειριακή θύρα για μετάδοση.
Η χρονική απόκριση καθώς και η απόκριση πλάτους και φάσης εικονίζονται παρακάτω.

Χρονική απόκριση σήματος



Μετασηματισμός Fourier και φασματικό περιεχόμενο

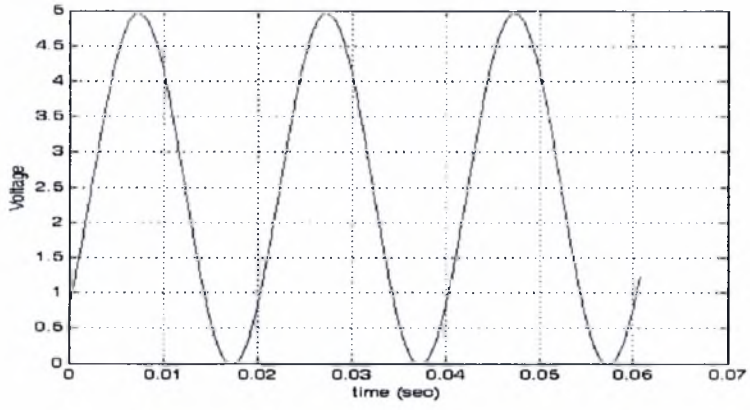


Στο προηγούμενο σχήμα, διακρίνουμε εύκολα τη βασική αρμονική του σήματος εισόδου στη συχνότητα F=50Hz, στην οποία η τιμή του μέτρου είναι στα 13.86 dB, δηλαδή περίπου 5 Volt. Οι μικρές ευδιάκριτες γραμμές αποτελούν το θόρυβο του σήματος εισόδου, που θα αποκοπεί με αύξηση του βαθμού (αύξηση παράγοντα k) του βαθυπερατού φίλτρου.

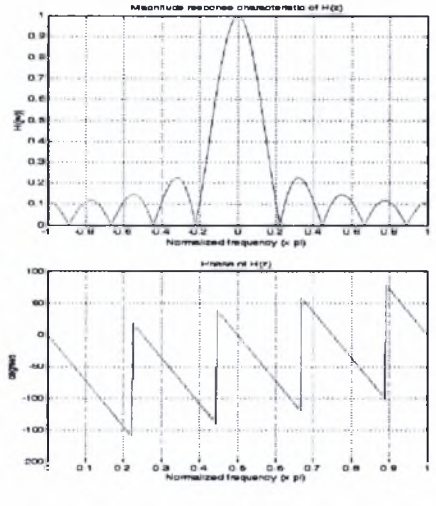
II.2 Φίλτρο <u>ογδόου</u> βαθμού (k=9, 50Hz)

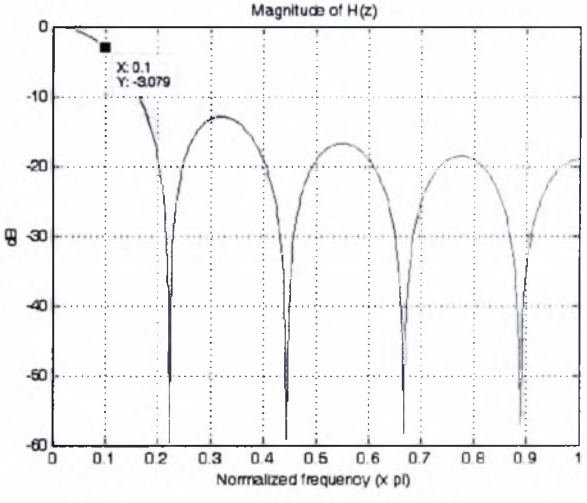
Παρακάτω δίνονται η χρονική απόκριση του σήματος και τα διαγράμματα πλάτους, φάσης και το zero-pole διάγραμμα με τις αντίστοιχες θέσεις των πόλων και των μηδενικών.

Χρονική απόκριση σήματος



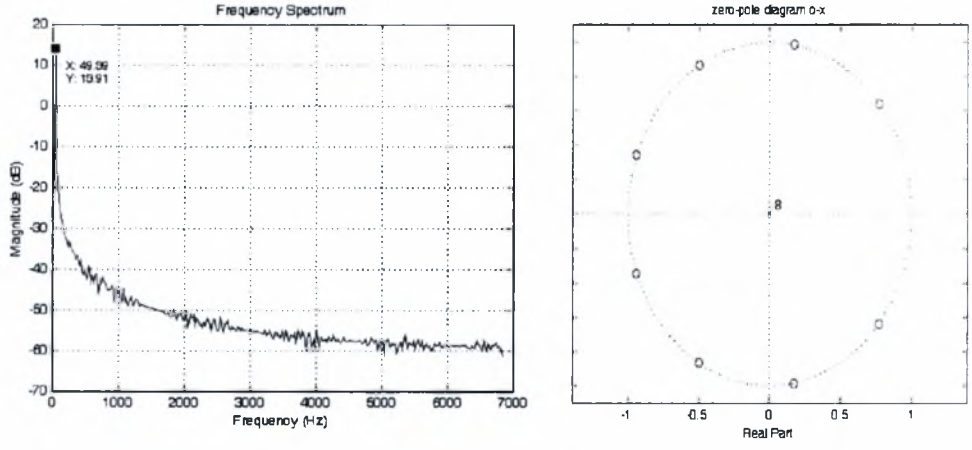
Απόκριση πλάτους και φάσης





Στο σχεδιάγραμμα απόκρισης του μέτρου του σήματος εισόδου, παρατηρείται η συρρίκνωση της ζώνης διέλευσης και η τιμή -3dB (θέση πόλων) παρουσιάζεται στη συχνότητα f=0.1. Η βασική συχνότητα του σήματος εισόδου f=50/Fs=0.035, περνά ελεύθερα απο τη ζώνη διέλευσης χωρίς μείωση του πλάτους.

Διάγραμμα Zero-pole και φασματικό περιεχόμενο

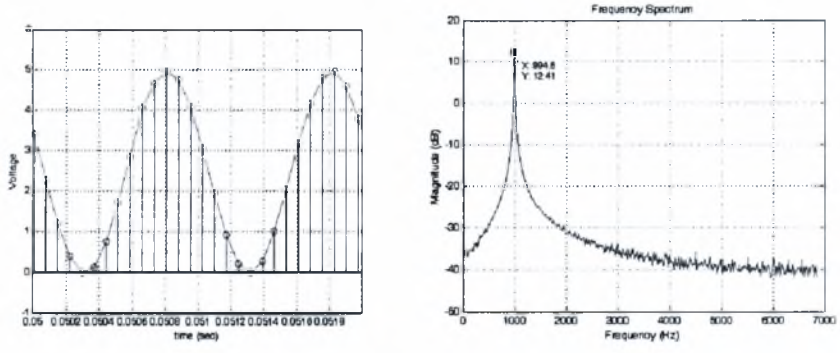


Παρατηρώντας προσεκτικά το φασματικό περιεχόμενο του παραπάνω σχήματος, διακρίνεται μια αρκετά μεγάλη μείωση του θορύβου, χωρίς όμως την παραμικρή παραμόρφωση της τιμής του πλάτους σήματος εισόδου στη βασική συχνοτική περιοχή των f=50Hz. Άξιο παρατήρησης είναι επίσης οι θέσεις των τεσσάρων ζευγών μιγαδικών μηδενικών που μηδενίζουν την τιμή του πλάτους περιοδικά (διάγραμμα απόκρισης πλάτους) και έχουν σαν αποτέλεσμα τη δημιουργία των λοβών.

II.3 Φίλτρο μηδενικού βαθμού (k=1, 1KHz)

Η αύξηση της συχνότητας του σήματος εισόδου, έχοντας σα δεδομένο ότι η συχνότητα δειγματοληψίας βρίσκεται στην τιμή των Fs=**13,698 kHz**, ο αριθμός των δειγμάτων ανά περίοδο μειώνεται. Στην προκειμένη περίπτωση ο αριθμός των δειγμάτων υπολογίζεται στα **13.6 Samples/period**. Στα επόμενα σχεδιαγράμματα παρατίθενται η χρονική απόκριση του σήματος και το φασματικό περιεχόμενο.

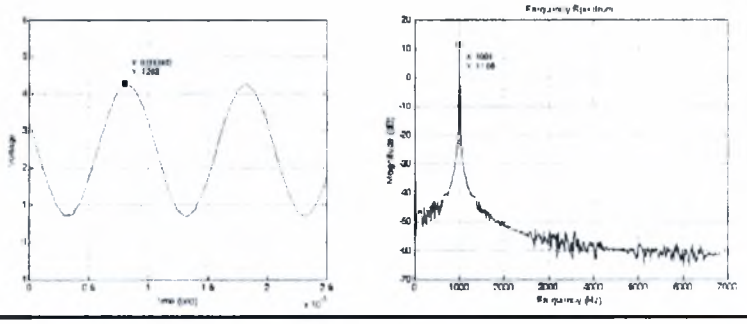
Διαγράμματα Χρονικής απόκρισης και Φασματικού περιεχομένου



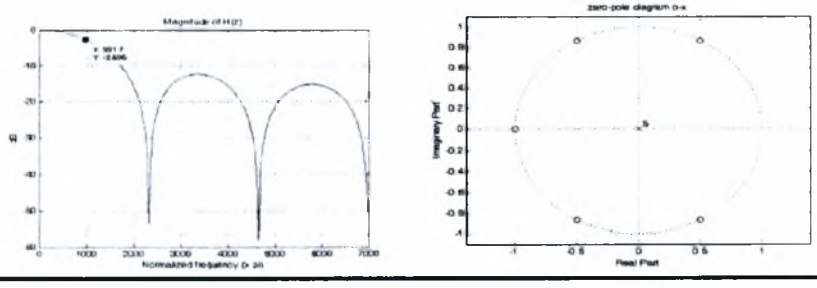
Στο διάγραμμα φασματικού περιεχομένου διακρίνεται η βασική αρμονική και η τιμή του μέτρου της τάσης σε dB.

Η μικρή αύξηση του του παράγοντα K του φίλτρου έχει ως αποτέλεσμα τη συρρίκνωση της ζώνης διέλευσης η οποία και αποκόπτει τη βασική συχνοτική περιοχή. Στα παρακάτω σχεδιαγράμματα, γίνεται εμφανής η μείωση του πλάτους του σήματος.

Χρονική απόκριση και Φασματικό περιεχόμενο



Απόκριση Πλάτους και Zero-Pole διάγραμμα



III. Επίλογος

Όπως μπορεί εύκολα να παρατηρηθεί από τα προαναφερθέντα, σκοπός της εργασίας ήταν να σχεδιαστεί ένα απλό ηλεκτρονικό σύστημα για την καταγραφή και μέτρηση αναλογικών σημάτων για την εξαγωγή χρήσιμων συμπερασμάτων και υπολογισμών. Οι εφαρμογές αισθητήρων καθιστούν επιτακτική την ανάγκη για σχεδιασμά και υλοποίηση ασύρματων συστημάτων πολύ χαμηλής κατανάλωσης ισχύος (χρήση απλής μπαταρίας) που εξασφαλίζουν ασφαλή μετάδοση στα δεδομένα, χαμηλό κόστος κατασκευής και απλή συνδεσιμότητα (interface). Η τεχνολογία των νέων ολοκληρωμένων κυκλωμάτων μικροελεγκτών συμβάλλουν σημαντικά προς την κατεύθυνση αυτή καθώς διαθέτουν υψηλής απόδοσης και λειτουργικότητας περιφερειακά (**rfPIC, dsPIC**) και μπορούν να ικανοποιήσουν και τις πιο απαιτητικές ανάγκες μιας εφαρμογής. Η απλή σειριακή διασύνδεση, η επιλογή εξωτερικού ή εσωτερικού ταλαντωτή, η ταχύτητα των κυκλωμάτων με την αύξηση της συχνότητας, οι A/D converters πολλών καναλιών, τα πρωτόκολλα επικοινωνίας που υποστηρίζουν και το μικρό τους μέγεθος είναι μερικά από τα βασικά πλεονεκτήματά τους για τη σχεδίαση τέτοιων συστημάτων. Άξιο αναφοράς είναι και η ανάπτυξη μεταγλωττιστών για τη συγγραφή *'έξυπνου'* και αποδοτικότερου κώδικα για την εκάστοτε εφαρμογή. Τέλος, ας ελπίσουμε ότι ο τομέας αυτός της τεχνολογίας θα μπορέσει να αναπτυχθεί ακόμη περισσότερο και οι εφαρμογές του θα κάνουν πιο βολική και ασφαλή τη ζωή των ανθρώπων.

Βιβλιογραφία

1. *John G. Proakis, Dimitris G. Manolakis, "Digital signal processing: principles, algorithms", and applications*, 3rd ed, Prentice_Hall, 1996
2. *A. Openheim, A. Wilsky, "Signals and Systems"*, 2nd ed. Prentice Hall 1996.
3. *A. Openheim, R. Schafer*, "*Discrete-Time Signal Processing*", 2nd ed. Prentice Hall 1999.
4. *Randy Yates, "Fixed-Point Arithmetic: An Introduction"*, 2001
5. *"PIC18F2455/2550/4455/4550, 28/40/44-Pin High-Performance, Enhanced Flash USB Microcontrollers with NanoWatt Technology Data sheet"*, © 2004 Microchip Technology Inc.
6. *Rodger RicheyYet, "Another Clock Featuring the PIC16C924"*, AN649, © 1997 Microchip Technology Inc.
7. *Mike Garbutt "Asynchronous Communications with the PICmicro® USART"*, AN774, © 2003 Microchip Technology Inc.
8. *Steve Bible and Jim Lepkowsk, "Interfacing the TC77 Thermal Sensor to a PICmicro® Microcontroller"*, AN913, © 2004 Microchip Technology Inc.

9. *Rawin Rojvanit, "Migrating Applications to USB from RS-232 UART with Minimal Impact on PC Software",* AN956, © 2004 Microchip Technology Inc.
10. *"Mplab® C18 C COMPILER user's guide",* (DS51288J), © 2004 Microchip Technology Inc.
11. *"MPLAB® C18 C COMPILER LIBRARIES",* (DS51297F), © 2004 Microchip Technology Inc.
12. *Frank J. Testa,"IEEE 754 Compliant Floating Point Routines",* © 2004, Microchip Technology Inc.
13. *"Introduction to UtilityMetering Tutorial",* (DS39757A), © 2004 Microchip Technology Inc.
14. *"802.15.4 and ZigBee™",* © 2006 MaxStream Inc
15. *"XBee™/XBee-PRO™ OEM RF Modules, Product Manual",* © 2006 MaxStream Inc
16. *David Flowers, Kim Otten, Nilesh Rajbharti, Yifeng Yang "Microchip Stack for the ZigBee Protocol",* AN965, © 2007 Microchip Technology Inc.

Δικτυακοί τόποι

17. www.microchip.com
18. forum.microchip.com
19. www.maxstream.net
20. www.embbeded.com
21. www.analogplanet.com
22. www.palowireless.com/zigbee
23. www.networkworld.com
24. www.microcontroller.com
25. www.mikroe.com/en/
26. www.mrrobot.com/
27. www.piclist.com/techref/piclist
28. www.mathtools.net/MATLAB/Test_and_Measurement/
29. www.camiresearch.com/Data_Com_Basics/RS232_standard/
30. www.maxim-ic.com
31. www.rsgreece.com