

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ: ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΑΣ

ΕΡΓΑΣΤΗΡΙΟ: ΕΝΣΥΡΜΑΤΗΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΣ



Διπλωματική Εργασία

του φοιτητή του Τμήματος Ηλεκτρολόγων Μηχανικών και
Τεχνολογίας Υπολογιστών της Πολυτεχνικής Σχολής του
Πανεπιστημίου Πατρών

Δίκαρου Γεώργιου του Δημητρίου

Αριθμός Μητρώου: 6241

Θέμα

**«Αυτόματος διαχωρισμός και αναγνώριση ακουστικών συμβάντων μηχανής
αυτοκινήτου και υλοποίηση στον ADUC 7020»**

Επιβλέπων Καθηγητής

Επίκουρος Καθηγητής Ευάγγελος Δερματάς

Αριθμός Διπλωματικής Εργασίας:

Πάτρα, Φεβρουάριος 2013

ΠΙΣΤΟΠΟΙΗΣΗ

Πιστοποιείται ότι η Διπλωματική Εργασία με θέμα

«Αυτόματος διαχωρισμός και αναγνώριση ακουστικών συμβάντων μηχανής αυτοκινήτου και υλοποίηση στον ADUC 7020»

του φοιτητή του τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών

Δίκαρου Γεώργιου

Αριθμός Μητρώου: 6241

Παρουσιάστηκε δημόσια και εξετάστηκε στο Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών στις/...../.....

Ο Επιβλέπων

Ο Διευθυντής του Τομέα

Επίκουρος Καθηγητής

Καθηγητής

Ευάγγελος Δερματάς

Νικόλαος Φακωτάκης

Περίληψη

Στόχος της συγκεκριμένης διπλωματικής εργασίας ήταν η θεωρητική προσέγγιση και υλοποίηση ενός συστήματος ανίχνευσης βλαβών στα διάφορα μηχανικά μέρη που απαρτίζουν μια μηχανή αυτοκινήτου. Ένας πιεζοηλεκτρικός αισθητήρας μετατρέπει τις δονήσεις της μηχανής σε ηλεκτρικό σήμα, το οποίο στη συνέχεια ενισχύεται και φιλτράρεται κατάλληλα ώστε να τροφοδοτήσει την είσοδο ενός analog-to-digital μετατροπέα του μικροελεγκτή ADuC 7020 της Analog Devices, ο οποίος ενσωματώνεται στο ADuC-MT7020 development board της Olimex. Αφού γίνει αυτό, το σήμα μας επεξεργάζεται ψηφιακά με τη χρήση κατάλληλου αλγορίθμου που αναπτύχθηκε για τη συγκεκριμένη εφαρμογή. Στα κεφάλαια που ακολουθούν αναλύεται διαδοχικά κάθε στάδιο της διαδικασίας. Έτσι στο 1^ο κεφάλαιο παρουσιάζεται το φαινόμενο του πιεζοηλεκτρισμού και οι αρχές λειτουργίας που διέπουν τους πιεζοηλεκτρικούς μετατροπείς ενώ στο 2^ο περιγράφεται η διαδικασία της ψηφιοποίησης αναλογικών σημάτων. Στο 3^ο κεφάλαιο γίνεται η περιγραφή των μεθόδων που χρησιμοποιούνται ώστε να δημιουργήσουμε το φάσμα συχνοτήτων ενός σήματος ενώ στο 4^ο παρουσιάζονται τα σημαντικά πλεονεκτήματα που μας παρέχει η λεπτομερής ανάλυσή του μέσα από την ενδεικτική παρουσίαση των κυριότερων μηχανικών σφαλμάτων και τις μεταβολές που αυτά επιφέρουν στο φάσμα. Στο 5^ο κεφάλαιο γίνεται η περιγραφή των κυριότερων στοιχείων των μικροελεγκτών και των ενσωματωμένων συστημάτων ενώ παρουσιάζονται και τα χαρακτηριστικά του μικροελεγκτή και του development board που αναφέρθηκαν πιο πάνω για να ακολουθήσει στο 6^ο η παρουσίαση των κυριότερων περιφερειακών του συγκεκριμένου μικροελεγκτή με τη χρησιμοποίηση και κάποιων παραδειγμάτων ώστε να γίνει σαφέστερη η λειτουργία τους. Στο τελευταίο κεφάλαιο παρουσιάζονται οι τρεις υλοποιήσεις κώδικα που έγιναν στο προγραμματιστικό περιβάλλον μVision 4 της Keil με σκοπό την αναγνώριση των συνιστωσών που απαρτίζουν το σήμα μας και την ταυτοποίησή τους. Οι δύο πρώτες υλοποιήσεις, στις οποίες χρησιμοποιήθηκαν ένας απλός και ένας fixed-point FFT αντίστοιχα, ήταν αδύνατο να ανταποκριθούν στις real-time απαιτήσεις της εφαρμογής μας και για το λόγο αυτό οδηγηθήκαμε στην τρίτη και τελική υλοποίηση, στην οποία χρησιμοποιήθηκε ο αλγόριθμος Goertzel. Για λόγους απλοποίησης χρησιμοποιήθηκε ένα ημίτονο για την αναπαράσταση ενός σήματος με ιδιαίτερα πολύπλοκη κυματομορφή όπως αυτό που προκύπτει από τις δονήσεις μιας μηχανής αυτοκινήτου, εντούτοις επαρκεί ώστε να κατανοηθεί και να επιβεβαιωθεί η ορθή λειτουργία του κώδικά μας.

Abstract

The main target of this thesis was the theoretical approach and implementation of a system that recognizes the damages occurring at the machinery parts of a car's engine. A piezoelectric sensor transforms the vibrations of the engine into an analog electric signal, which is then amplified and filtered before it is applied to the analog-to-digital converter of Analog's ADuC 7020 microcontroller, incorporated in ADuC-MT7020 development board produced by Olimex. After that the signal is processed digitally through an appropriate algorithm developed for this application. Throughout the chapters that follow all the stages of this procedure are analyzed sequentially. At the first chapter the theory of piezoelectricity and the operating principles of piezoelectric sensors are presented, whereas at the second the procedure of digitizing analog signals is described. At chapter 3 the methods used in order to create a signal's spectrum are mentioned while at chapter 4 the benefits from a spectrum's proper analysis are highlighted through the indicative presentation of the most common machinery faults and how they alter it. At the fifth chapter a reference is made at the main characteristics of microcontrollers and embedded systems in general but also a more detailed description of the microcontroller and the development board mentioned above. ADuC 7020 microcontroller's main peripherals are presented at chapter 6 together with some examples that illustrate their function. The programming part of this thesis is included at the last chapter where the three code implementations developed in Keil's μ Vision 4 programming environment are presented. The first two implementations, within which a simple and a fixed-point FFT were used respectively, couldn't meet the real-time needs of our application resulting in a final third implementation using a Goertzel algorithm. Although we used a simple sinusoid to represent the signal produced by the vibrations of a car's engine characterized by a complex waveform, it is sufficient in comprehending and confirming the correct execution of the code we created.

Ευχαριστίες,

Θα ήθελα να ευχαριστήσω θερμά τον Αναπληρωτή Καθηγητή Κύριο Δερματά Ευάγγελο για την πολύτιμη βοήθεια και συνεισφορά του στη διεκπεραίωση της διπλωματικής μου εργασίας, τις σημαντικές γνώσεις που αποκόμισα στα πλαίσια της συνεργασίας μας κυρίως στον κλάδο των μικροελεγκτών και των ενσωματωμένων συστημάτων αλλά και τις ξεχωριστές συμβουλές του για μια επιτυχημένη επαγγελματική σταδιοδρομία.

Επίσης δεν θα μπορούσα να μην εκφράσω την ιδιαίτερη ευγνωμοσύνη μου στην οικογένειά μου και τους φίλους μου καθώς η παρουσία και η στάση τους ανέκαθεν αποτελούσε για μένα πηγή έμπνευσης ώστε να θέτω στόχους και να τους κατακτώ, με τελευταίο και κυρίαρχο την επιτυχή ολοκλήρωση των προπτυχιακών μου σπουδών.

Πίνακας περιεχομένων

ΚΕΦΑΛΑΙΟ 1 : ΠΙΕΖΟΗΛΕΚΤΡΙΣΜΟΣ

1.1 Πιεζοηλεκτρικό Φαινόμενο	11
1.2 Θεωρία Πιεζοηλεκτρισμού	12
1.2.1 Πόλωση	12
1.2.2 Μαθηματική Περιγραφή	13
1.2.3 Πιεζοηλεκτρικές σταθερές	15
1.3 Αισθητήρες - Ενεργοποιητές	16
1.3.1 Αρχές λειτουργίας πιεζοηλεκτρικών αισθητήρων	17
1.3.2 Εφαρμογές πιεζοηλεκτρικών αισθητήρων	19

ΚΕΦΑΛΑΙΟ 2: ΨΗΦΙΟΠΟΙΗΣΗ

2.1 Ψηφιακή Εποχή	21
2.2 Ψηφιοποιώντας αναλογικά σήματα	21
2.3 Υπερδειγματοληψία	26
2.4 Ανακτώντας το σήμα από τα δείγματά του	28

ΚΕΦΑΛΑΙΟ 3: ΜΕΘΟΔΟΙ ΜΕΤΑΒΑΣΗΣ ΣΤΟ ΠΕΔΙΟ ΤΗΣ ΣΥΧΝΟΤΗΤΑΣ

3.1 Γενικά	31
3.2 Διακριτός Μετασχηματισμός Fourier (DFT)	32
3.3 Γρήγορος Μετασχηματισμός Fourier (FFT)	37
3.4 Απαιτήσεις Συστήματος για τον Υπολογισμό του FFT	45
3.5 Παραθυροποίηση και Φασματική Διαρροή	47
3.6 Αλγόριθμος GOERTZEL	51

ΚΕΦΑΛΑΙΟ 4: ΑΝΑΛΥΣΗ ΔΟΝΗΣΕΩΝ

4.1 Η μείωση της δόνησης στα στοιχεία που την απαρτίζουν	57
4.1.1 Το πεδίο του χρόνου	57
4.1.2 Το πεδίο της συχνότητας	58
4.2 Χαρακτηριστικά δονήσεων των κυριότερων σφαλμάτων σε περιστρεφόμενο μηχανικό εξοπλισμό	59
4.2.1 Αζυγοσταθμία	60

4.2.2 Κακή ευθυγράμμιση.....	61
4.2.3 Μηχανική χαλαρότητα	63
4.2.4 Ελαττώματα οδοντωτών τροχών (γρανάζια).....	65
4.2.5 Ελαττώματα εδράνων κύλισης (ρουλεμάν)	67

ΚΕΦΑΛΑΙΟ 5: ΜΙΚΡΟΕΛΕΓΚΤΕΣ

5.1 Βασικές έννοιες μικροελεγκτών	72
5.1.1 Χρήση μικροελεγκτών	72
5.1.2 Χαρακτηριστικά	72
5.1.3 Δυνατότητες – Πλεονεκτήματα	74
5.2 Ενσωματωμένα Συστήματα.....	74
5.2.1 Γενικά.....	74
5.2.2 Παραδείγματα	75
5.2.3 Χαρακτηριστικά	76
5.2.4 Κυρίως αρχιτεκτονικές	77
5.3 Αρχιτεκτονική ARM	78
5.4 Μικροελεγκτές ARM7TDMI	79
5.5 Ο Μικροελεγκτής ADuC-702x	80
5.6 Η πλατφόρμα ανάπτυξης ADuC-MT7020 της Olimex	82
5.7 Το περιβάλλον ανάπτυξης εφαρμογών μVision της Keil.....	85

ΚΕΦΑΛΑΙΟ 6: ΚΥΡΙΑ ΠΕΡΙΦΕΡΕΙΑΚΑ ΤΟΥ ADUC 702x - ΠΑΡΑΔΕΙΓΜΑΤΑ

6.1 Flash/EE Memory - SRAM	88
6.2 ADC.....	89
6.3 Reset	91
6.4 DAC.....	91
6.5 UART	92
6.6 IRQ / FIQ.....	93
6.7 Timers.....	95
6.8 General Purpose Input-Output ports (GPIO).....	99
6.9 Παραδείγματα.....	101

ΚΕΦΑΛΑΙΟ 7: ΥΛΟΠΟΙΗΣΗ

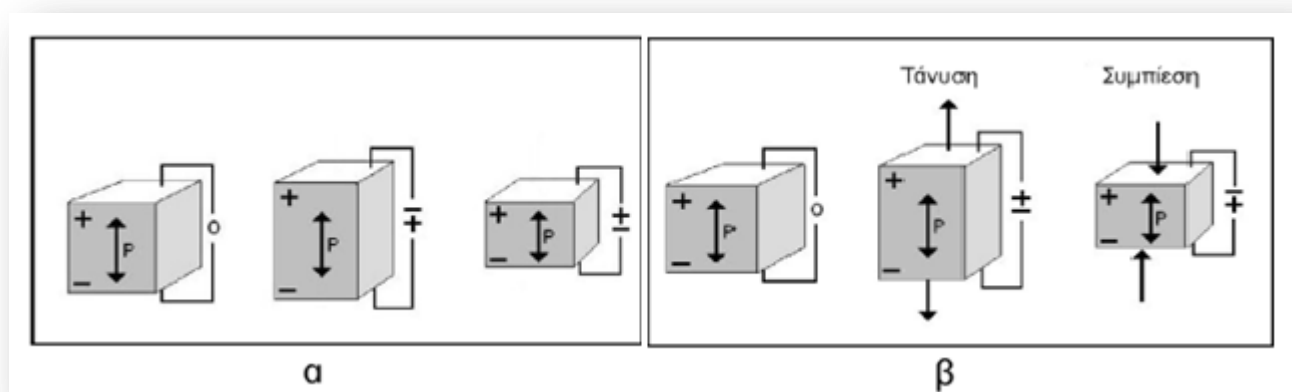
7.1 Υλοποίηση με απλό radix-2 DIT FFT	116
7.2 Υλοποίηση με fixed-point FFT	118
7.3 Υλοποίηση με αλγόριθμο Goertzel	119
Βιβλιογραφία	124
ΠΑΡΑΡΤΗΜΑ 1	126
ΠΑΡΑΡΤΗΜΑ 2	136
ΠΑΡΑΡΤΗΜΑ 3	151

ΚΕΦΑΛΑΙΟ 1^ο

ΠΙΕΖΟΗΛΕΚΤΡΙΣΜΟΣ

1.1 Πιεζοηλεκτρικό Φαινόμενο

Πιεζοηλεκτρικό ονομάζεται το φαινόμενο κατά το οποίο κάποια υλικά, κυρίως κρύσταλλοι και κεραμικά αναπτύσσουν ηλεκτρικό δυναμικό όταν στην επιφάνειά τους ασκείται κάποια μηχανική τάση. Η ανισοτροπία στην κρυσταλλική δομή του κεραμικού υλικού αποτελεί βασική προϋπόθεση για να υπάρξουν πιεζοηλεκτρικές ιδιότητες. Το ηλεκτρικό δυναμικό σε μερικές τάξεις κρυστάλλων μεταφράζεται ως ομοιόμορφη κατανομή του ηλεκτρικού φορτίου σε όλο το κρυσταλλικό πλέγμα. Εάν τα άκρα του υλικού στο οποίο εφαρμόζεται η δύναμη δεν είναι βραχυκυκλωμένα τότε το φορτίο που δημιουργείται προκαλεί μια τάση στα άκρα του. Εκτός από το πιεζοηλεκτρικό φαινόμενο, τη δημιουργία δηλαδή ηλεκτρικού φορτίου σε ένα υλικό λόγω της πίεσης που ασκείται σε αυτό (Σχ. 1.1β) υπάρχει και το αντίστροφο πιεζοηλεκτρικό φαινόμενο (Σχ.1.1α), κατά το οποίο παρατηρείται εμφάνιση δυνάμεων που επιδρούν πάνω στο υλικό λόγω παρουσίας ηλεκτρικού πεδίου [3].



Σχήμα 1.1 (α) Αντίστροφο πιεζοηλεκτρικό φαινόμενο (β) Ευθύ πιεζοηλεκτρικό φαινόμενο

Το φαινόμενο του πιεζοηλεκτρισμού παρατηρήθηκε πρώτη φορά από τα αδέρφια Curie το 1880. Η πειραματική τους διάταξη περιελάμβανε όργανα που μπορούσαν να μετρήσουν το ηλεκτρικό φορτίο που δημιουργούνταν σε ειδικά διαμορφωμένους κρυστάλλους έπειτα από την άσκηση δύναμης στην επιφάνειά τους. Το 1881 ο Gabriel Lippmann απέδειξε με τη χρήση μαθηματικών τύπων την ύπαρξη του αντίστροφου πιεζοηλεκτρικού φαινομένου. Την απόδειξη του Lippmann επιβεβαίωσαν λίγο αργότερα και πειραματικά οι αδερφοί Curie. Τα επόμενα χρόνια προσδιορίστηκαν οι είκοσι τάξεις κρυστάλλων οι οποίες εμφανίζουν πιεζοηλεκτρική συμπεριφορά.

Λίγο αργότερα το 1920 ανακαλύφθηκε ο φεροηλεκτρισμός, ο οποίος εκφράζει την παρουσία αυθόρμητης διπολικής ροπής. Εδώ πρέπει να τονιστεί πως το πιεζοηλεκτρικό και το φεροηλεκτρικό φαινόμενο είναι δύο διαφορετικά φαινόμενα. Όλα τα φεροηλεκτρικά υλικά είναι πιεζοηλεκτρικά αλλά το αντίθετο δεν ισχύει πάντα. Ένα φεροηλεκτρικό υλικό μετατρέπεται σε πιεζοηλεκτρικό όταν εφαρμοστεί στα άκρα του ισχυρό ηλεκτρικό πεδίο. Το πρώτο φεροηλεκτρικό κεραμικό υλικό ανακαλύφθηκε το 1943, του οποίου οι πιεζοηλεκτρικές ιδιότητες εντοπίστηκαν από τον S. Roberts το 1947. Το 1954 ο B. Jaffe ανακάλυψε το πιεζοηλεκτρικό κεραμικό υλικό γνωστό ως ζirkο-κυανικός μόλυβδος ή αλλιώς PZT, το οποίο αποτέλεσε και το κύριο συστατικό στην αντίστοιχη βιομηχανία της

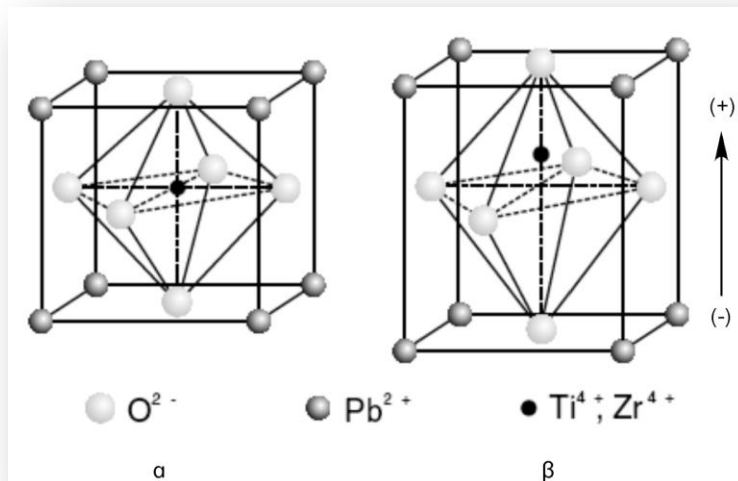
εποχής και είναι μέχρι και σήμερα το πιο ευρέως διαδεδομένο κεραμικό με πιεζοηλεκτρικές ιδιότητες.

1.2 Θεωρία Πιεζοηλεκτρισμού

Κάποιες από τις βασικές έννοιες που διέπουν τα πιεζοηλεκτρικά υλικά αναφέρονται εδώ με σκοπό την κατανόηση της λειτουργίας των πιεζοηλεκτρικών στοιχείων που χρησιμοποιούνται σε μεγάλο βαθμό στην μικρο/νανο-ρομποτική.

1.2.1 Πόλωση

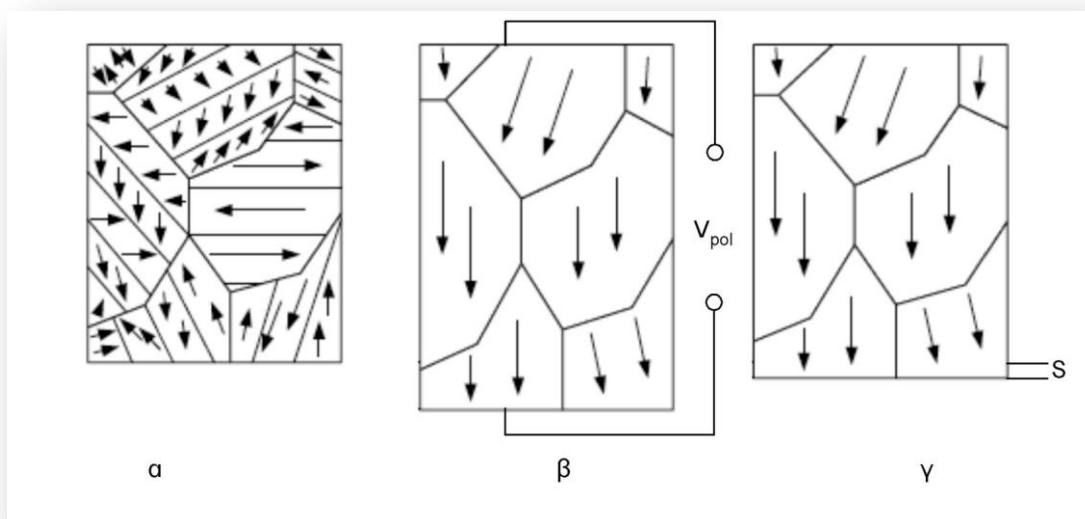
Όπως αναφέρθηκε και πιο πάνω ένα φεροηλεκτρικό υλικό δεν έχει εξ αρχής πιεζοηλεκτρικές ιδιότητες. Από μια τιμή θερμοκρασίας και πάνω, η οποία λέγεται θερμοκρασία Curie, ο κρύσταλλος του κεραμικού υλικού έχει απλή κυβική συμμετρία και δεν παρουσιάζει διπολική ροπή (Σχ. 1.2α). Σε θερμοκρασίες χαμηλότερες από τη θερμοκρασία Curie ο κρύσταλλος αποκτά τετραγωνική ή ρομβοειδή συμμετρία καθώς και διπολική ροπή (Σχ. 1.2β) [2]. Τα γειτονικά δίπολα σχηματίζουν περιοχές ομοιόμορφης τοπικής ευθυγράμμισης. Ο τυχαίος προσανατολισμός των κόκκων του υλικού και η ύπαρξη των περιοχών ευθύνεται για το ότι δεν υπάρχει καθαρή πόλωση (Σχ.1.3α).



Σχήμα 1.2 (α) Κρύσταλλος με απλή κυβική συμμετρία χωρίς διπολική ροπή (β) Κρύσταλλος με τετραγωνική συμμετρία και διπολική ροπή

Η εφαρμογή του ηλεκτρικού πεδίου στα άκρα του κάνει το κεραμικό υλικό να πολωθεί και να αποκτήσει πιεζοηλεκτρικές ιδιότητες. Πιο συγκεκριμένα με την εφαρμογή συνεχούς τάσης στο υλικό οι φεροηλεκτρικές περιοχές ευθυγραμμίζονται με τη φορά του ηλεκτρικού πεδίου κι έτσι δημιουργείται καθαρή πόλωση (Σχ.1.3β). Δεν ευθυγραμμίζονται όμως όλες οι περιοχές το ίδιο αλλά κάποιες μερικώς και άλλες καθόλου. Οι παράγοντες που καθορίζουν τον αριθμό των περιοχών

αυτών είναι η ένταση του ηλεκτρικού πεδίου, η θερμοκρασία του υλικού και ο χρόνος εφαρμογής του πεδίου. Η μόνιμη αυτή πόλωση που αποκτά το υλικό έχει σαν αποτέλεσμα και την μόνιμη επιμήκυνσή του (Σχ.1.3γ). Για να αλλάξουμε την πολικότητα πρέπει να εφαρμόσουμε ηλεκτρικό πεδίο αντίθετης πολικότητας είτε αυξάνοντας τη θερμοκρασία πάνω από το σημείο Curie είτε ασκώντας μηχανική τάση.



Σχήμα 1.3 (α) Μη πολωμένο φεροηλεκτρικό υλικό, (β) διαδικασία πόλωσης και (γ) μόνιμη πόλωση και επιμήκυνση

1.2.2 Μαθηματική Περιγραφή

Στο σημείο αυτό θα κάνουμε χρήση των καταστατικών εξισώσεων που διέπουν το φαινόμενο και θα γίνει μια σύντομη μαθηματική περιγραφή του φαινομένου.

Όσον αφορά τη μηχανική πλευρά ο νόμος που αξιοποιούμε είναι ο νόμος του Hooke

$$S = sT$$

όπου S είναι η παραμόρφωση, T η μηχανική τάση και s είναι η ελαστικότητα του υλικού. Επειδή στο φαινόμενο του πιεζοηλεκτρισμού λαμβάνουν χώρα και ηλεκτρικά φαινόμενα ισχύει και η καταστατική εξίσωση του ηλεκτρισμού, η οποία περιγράφει την κίνηση ενός ηλεκτρικού φορτίου όταν κάποιο διηλεκτρικό υποβάλλεται σε ηλεκτρικό πεδίο. Έτσι

$$D = \epsilon E$$

όπου το D εκφράζει την ηλεκτρική μετατόπιση, το ϵ είναι η ηλεκτρική διαπερατότητα του υλικού και E η ένταση του ηλεκτρικού πεδίου. Μετά από πράξεις καταλήγουμε στις γραμμικοποιημένες καταστατικές εξισώσεις του πιεζοηλεκτρισμού, οι οποίες δημοσιεύτηκαν αρχικά από την επιτροπή προτύπων της IEEE Ultrasonics, Ferroelectrics and Frequency Control Society το 1966 και αναθεωρήθηκαν το 1987.

$$S = s_E \cdot T + d \cdot E$$

$$D = d \cdot T + \epsilon_T \cdot E$$

όπου s_E είναι η μήτρα της ελαστικότητας του υλικού όταν υποβάλλεται σε σταθερό ηλεκτρικό πεδίο, ϵ_T είναι η ηλεκτρική διαπερατότητα μετρημένη υπό σταθερή τάση και d είναι η μήτρα των πιεζοηλεκτρικών σταθερών σε C/N.. Από τις παραπάνω εξισώσεις γίνεται αντιληπτό ότι η παραμόρφωση και η ηλεκτρική μετατόπιση του υλικού εξαρτώνται γραμμικά από τη μηχανική τάση και το ηλεκτρικό πεδίο στο οποίο υποβάλλονται. Σ' αυτές τις καταστατικές εξισώσεις δεν έχουν ληφθεί υπ' όψη οι μη γραμμικότητες των πιεζοηλεκτρικών (υστέρηση, ερπυσμός).

Οι καταστατικές εξισώσεις που παρατέθηκαν πιο πάνω μπορούν να πάρουν διάφορες μορφές με την εφαρμογή κατάλληλων μετασχηματισμών. Στον πίνακα 1.1 φαίνονται όλοι οι τρόποι αναπαράστασης των καταστατικών εξισώσεων. Στις εξισώσεις αυτές χρησιμοποιούνται οι ακόλουθοι συμβολισμοί:

c : στιβαρότητα του υλικού

e : πιεζοηλεκτρική σταθερά σύζευξης για τη μορφή (2) σε C/m²

g : πιεζοηλεκτρική σταθερά σύζευξης για τη μορφή (3) σε m²/C

q : πιεζοηλεκτρική σταθερά σύζευξης για τη μορφή (4) σε N/C

Πίνακας 1.1 Μορφές καταστατικών εξισώσεων

1)Μορφή παραμόρφωσης-φορτίου $S = s_E \cdot T + d \cdot E$ $D = d \cdot T + \epsilon_T \cdot E$	2)Μορφή μηχ. τάσης-φορτίου $T = c_E \cdot S - e \cdot E$ $D = e \cdot S + \epsilon_S \cdot E$
3)Μορφή παραμόρφωσης-ηλ. τάσης $S = s_D \cdot T + g \cdot D$ $E = -g \cdot T + \epsilon_T \cdot D$	4)Μορφή μηχ. τάσης-ηλ. τάσης $T = c_D \cdot S - q \cdot D$ $E = -q \cdot S + \epsilon_S \cdot D$

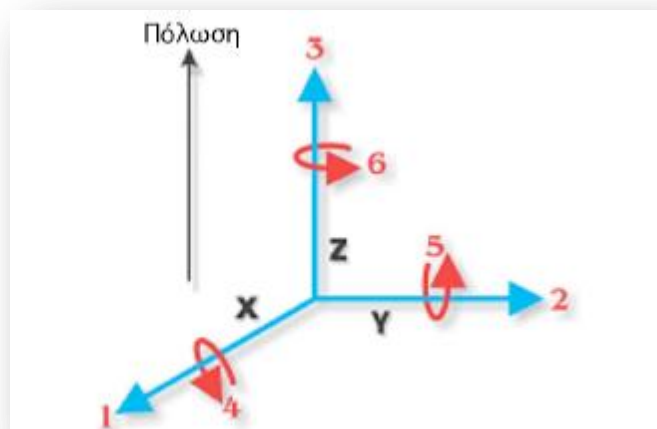
Για να προκύψουν οι πιο πάνω εξισώσεις απαιτείται η χρήση των κατάλληλων μετασχηματισμών οι οποίοι και παρουσιάζονται στον πίνακα 1.2.

Πίνακας 1.2 Μετασχηματισμοί για εναλλαγή της μορφής των καταστατικών εξισώσεων

Μορφή 1→ Μορφή 2 $c_E = s_E^{-1}$ $e = d \cdot s_E^{-1}$ $\varepsilon_S = \varepsilon_T - d \cdot s_E^{-1} \cdot d^T$	Μορφή 1→ Μορφή 3 $s_D = s_E - d \cdot \varepsilon_T^{-1} \cdot d^T$ $g = \varepsilon_T^{-1} \cdot d$
Μορφή 2→ Μορφή 4 $c_D = c_E - e \cdot \varepsilon_S^{-1} \cdot e^T$ $q = \varepsilon_S^{-1} \cdot e$	Μορφή 3→ Μορφή 4 $c_D = s_D^{-1}$ $q = g \cdot s_D^{-1}$ $\varepsilon_S^{-1} = \varepsilon_T^{-1} - g \cdot s_D^{-1} \cdot g^T$

1.2.3 Πιεζοηλεκτρικές σταθερές

Λόγω της ανισοτροπίας των πιεζοηλεκτρικών υλικών τα φαινόμενα που λαμβάνουν χώρα εξαρτώνται από τη διεύθυνση. Κάθε φυσική σταθερά που χρησιμοποιείται στις καταστατικές εξισώσεις έχει δύο δείκτες που υποδηλώνουν την κατεύθυνση κάποιων ποσοτήτων. Πιο συγκεκριμένα οι δείκτες 1,2,3 υποδηλώνουν κατεύθυνση X,Y,Z αντίστοιχα ενώ οι δείκτες 4,5 και 6 διάτμηση περί των αξόνων X,Y και Z αντίστοιχα.



Σχήμα 1.4 Ισοδυναμία δεικτών με το ορθοκανονικό σύστημα αξόνων

Πιεζοηλεκτρική σταθερά παραμόρφωσης ή φορτίου εξόδου d_{ij}

Όσον αφορά το πιεζοηλεκτρικό φαινόμενο, η σταθερά αυτή υποδηλώνει την πόλωση ανά μονάδα εφαρμοζόμενης μηχανικής τάσης. Ο πρώτος δείκτης συμβολίζει τη διεύθυνση της πόλωσης όταν το εφαρμοζόμενο ηλεκτρικό πεδίο είναι μηδέν και ο δεύτερος τη διεύθυνση της εφαρμοζόμενης δύναμης.

Όσον αφορά τώρα το αντίστροφο πιεζοηλεκτρικό φαινόμενο η σταθερά αυτή υποδηλώνει τη μηχανική παραμόρφωση ανά μονάδα εφαρμοζόμενου ηλεκτρικού πεδίου. Ο πρώτος δείκτης συμβολίζει τη διεύθυνση του εφαρμοζόμενου ηλεκτρικού πεδίου και ο δεύτερος τη διεύθυνση της παραμόρφωσης που προκύπτει.

Πιεζοηλεκτρική σταθερά ηλεκτρικής τάσης ή πεδίου εξόδου g_{ij}

Όταν εξετάζεται το ευθύ πιεζοηλεκτρικό φαινόμενο η σταθερά αυτή υποδηλώνει το ηλεκτρικό πεδίο που παράγεται ανά μονάδα εφαρμοζόμενης μηχανικής τάσης. Ο πρώτος δείκτης συμβολίζει τη διεύθυνση του ηλεκτρικού πεδίου που παράγεται και ο δεύτερος τη διεύθυνση της εφαρμοζόμενης δύναμης.

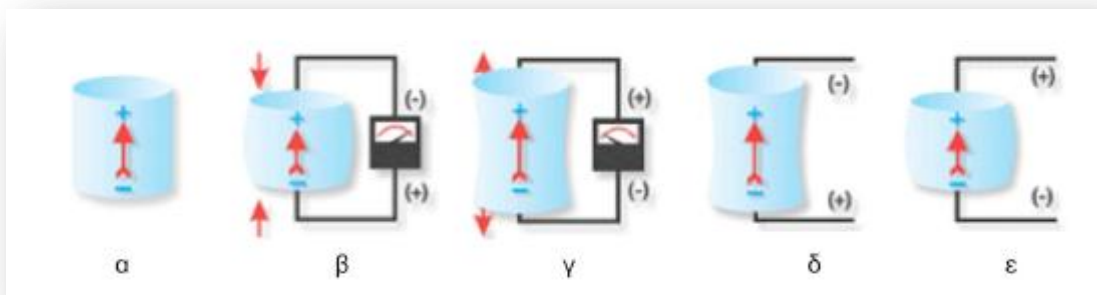
Κατά το αντίστροφο πιεζοηλεκτρικό φαινόμενο η σταθερά υποδηλώνει τη μηχανική παραμόρφωση ανά μονάδα εφαρμοζόμενης ηλεκτρικής μετατόπισης. Ο πρώτος δείκτης συμβολίζει τη διεύθυνση της εφαρμοζόμενης ηλεκτρικής μετατόπισης και ο δεύτερος τη διεύθυνση της παραμόρφωσης που προκύπτει.

Πιεζοηλεκτρική σταθερά ηλεκτρομηχανικής σύζευξης k_{ij}

Εκφράζει το ποσοστό μετατροπής της μηχανικής ενέργειας σε ηλεκτρική όταν εξετάζεται το ευθύ πιεζοηλεκτρικό φαινόμενο και το ποσοστό μετατροπής της ηλεκτρικής ενέργειας σε μηχανική όταν εξετάζεται το αντίστροφο πιεζοηλεκτρικό φαινόμενο. Ο πρώτος δείκτης συμβολίζει τη διεύθυνση της ηλεκτρικής ενέργειας που παράγεται ή εφαρμόζεται ενώ ο δεύτερος τη διεύθυνση της αντίστοιχης μηχανικής ενέργειας.

1.3 Αισθητήρες - Ενεργοποιητές

Κάθε διηλεκτρικό υλικό με πιεζοηλεκτρικές ιδιότητες μπορεί να χρησιμοποιηθεί είτε ως αισθητήρας (ευθύ πιεζοηλεκτρικό φαινόμενο) είτε ως ενεργοποιητής (αντίστροφο πιεζοηλεκτρικό φαινόμενο). Στο παρακάτω σχήμα φαίνεται ένα πολωμένο πιεζοηλεκτρικό υλικό.



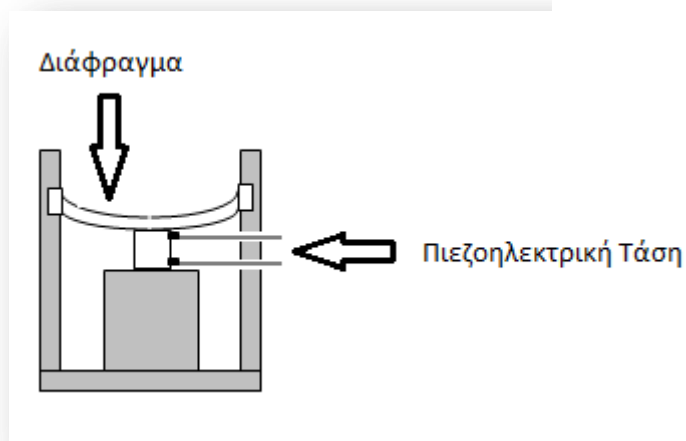
Σχήμα 1.5 (α) πιεζοηλεκτρικό σε θέση ισορροπίας, (β-γ) ευθύ πιεζοηλεκτρικό φαινόμενο και (δ-ε) αντίστροφο πιεζοηλεκτρικό φαινόμενο

Συμπίεση κατά τη διεύθυνση της πόλωσης ή τάνυση κάθετη στη διεύθυνση της πόλωσης δημιουργεί ηλεκτρική τάση ίδιας πολικότητας με την αρχική πόλωση (Σχ. 1.5β). Τάνυση κατά τη διεύθυνση της πόλωσης ή συμπίεση κάθετη στη διεύθυνση της πόλωσης δημιουργεί ηλεκτρική τάση αντίθετης πολικότητας με την αρχική πόλωση (Σχ.1.5γ). Έτσι ακριβώς λειτουργεί και ένας αισθητήρας ο οποίος παράγει κάθε φορά μια τάση ανάλογα με την τιμή της συμπίεσης ή τάνυσης που υφίσταται.

Στο σχήμα 1.5δ-ε φαίνεται το πώς λειτουργεί ένας ενεργοποιητής. Το πιεζοηλεκτρικό υλικό επιμηκύνεται στη διεύθυνση της πόλωσης και μειώνεται ταυτόχρονα και η διάμετρος του όταν εφαρμοστεί σε αυτό μια ηλεκτρική τάση ίδιας πολικότητας με την αρχική πόλωση. Αντίθετα αν αντιστραφεί η πολικότητα της εφαρμοζόμενης τάσης ο ενεργοποιητής συρρικνώνεται και η διάμετρος του αυξάνεται (Σχ. 1.5ε).

1.3.1 Αρχές λειτουργίας πιεζοηλεκτρικών αισθητήρων

Οι πιεζοηλεκτρικοί αισθητήρες είναι συνήθως κρύσταλλοι χαλαζία (quartz). Μπορούν να έχουν πολύ μικρό μέγεθος και είναι ανθεκτικοί σε υψηλές θερμοκρασίες. Η σταθερά k των κρυστάλλων είναι πολύ μεγάλη (δηλαδή οι κρύσταλλοι είναι «σκληροί»), με αποτέλεσμα η τάση V που παράγεται να έχει μικρές τιμές και να απαιτείται η ενίσχυσή της (χρήση ενισχυτή) ή η τοποθέτηση πολλών κρυστάλλων σε σειρά. Οι κρύσταλλοι τίθενται σε επαφή με ένα κινητό διάφραγμα, το οποίο δέχεται την πίεση, και συμπιέζονται από αυτό όπως φαίνεται στο Σχήμα 1.6.



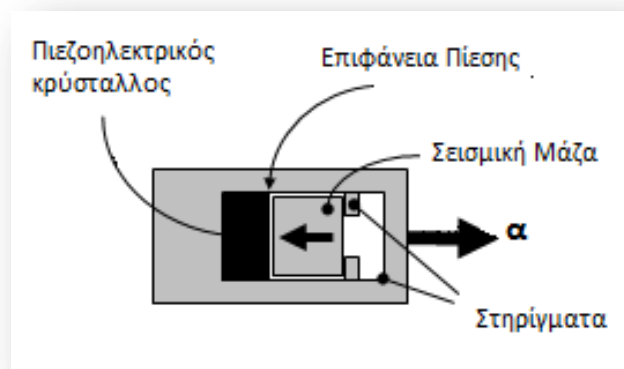
Σχήμα 1.6 Μορφή πιεζοηλεκτρικού αισθητήρα πίεσης

Οι πιεζοηλεκτρικοί κρύσταλλοι δε χρησιμοποιούνται μόνο για τη μέτρηση πιέσεων και δυνάμεων αλλά και επιταχύνσεων καθώς γνωρίζουμε ότι η δύναμη και η επιτάχυνση είναι μεγέθη ανάλογα, με βάση τη γνωστή σχέση:

$$F = m a$$

όπου **a** είναι η επιτάχυνση και **m** είναι η μάζα που δέχεται την επιτάχυνση.

Έτσι με τη βοήθεια των πιεζοηλεκτρικών επιταχυνσιομέτρων, όπως χαρακτηριστικά ονομάζονται, μπορούμε εκτός από επιταχύνσεις να ανιχνεύσουμε επιπρόσθετα δονήσεις και κρούσεις (shocks). Διαθέτουν μία μάζα *m* που τείνει να κινηθεί αντίθετα από την κατεύθυνση της επιτάχυνσης (λόγω της αδράνειας που εμφανίζει) και γι' αυτό ονομάζεται *σεισμική μάζα* (seismic mass). Η σεισμική αυτή μάζα τίθεται σε μόνιμη επαφή με έναν πιεζοηλεκτρικό κρύσταλλο, ο οποίος είναι ακλόνητα συνδεδεμένος με το περίβλημα του συστήματος (Σχήμα 1.7). Εάν το σύστημα υποστεί επιτάχυνση, η μάζα τείνει να κινηθεί αντίθετα προς την κατεύθυνση της επιτάχυνσης και πιέζει τον πιεζοηλεκτρικό κρύσταλλο, οπότε αυτός παράγει μία τάση ανάλογη της πίεσης (δύναμης) που έχει δεχθεί, και επομένως ανάλογη της επιτάχυνσης. Τα στηρίγματα του Σχήματος 1.7 αντικαθίστανται συχνά από μία βίδα, η οποία διαπερνά τον κρύσταλλο και τη σεισμική μάζα και τα κρατά σε επαφή.



Σχήμα 1.7 Αρχή λειτουργίας πιεζοηλεκτρικού επιταχυνσιόμετρου

1.3.2 Εφαρμογές πιεζοηλεκτρικών αισθητήρων

Συσκευές που περιλαμβάνουν πιεζοηλεκτρικούς αισθητήρες χρησιμοποιούνται ευρέως στις μέρες μας σε εργαστήρια, σε γραμμές βιομηχανικής παραγωγής και σαν πρωτότυπος εξοπλισμός. Χρησιμοποιούνται σε κάθε σχεδόν εφαρμογή στην οποία προϋποτίθενται οι ακριβείς μετρήσεις και η παρακολούθηση των αλλαγών διαφόρων μηχανικών μεγεθών, όπως η πίεση, η δύναμη και η επιτάχυνση. Συνοπτικά, αναφέρονται κάποιες εφαρμογές των πιεζοηλεκτρικών αισθητήρων στη σύγχρονη επιστήμη:

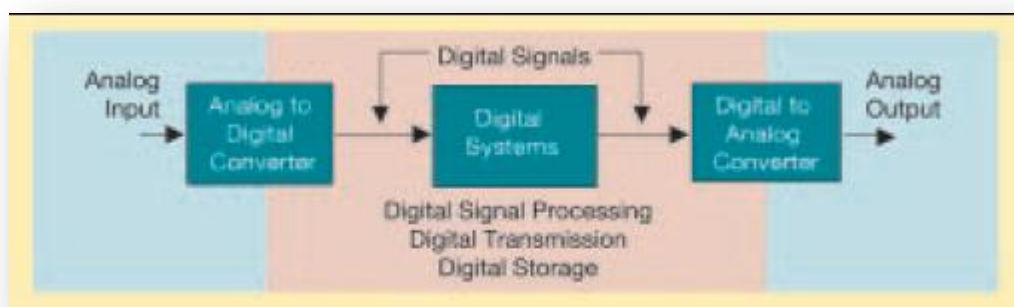
- Αεροδιαστημική έρευνα: τεχνολογία κατασκευής πυραύλων, συστήματα εκτίναξης επείγουσας ανάγκης, σχετικές δομικές κατασκευές
- Βαλλιστικές μελέτες: καύση, έκρηξη, εκπυρσοκρότηση και κατανομή ηχητικής πίεσης
- Βιοϊατρική: μετρήσεις δύναμης για ορθοπεδικό βηματισμό, εργονομία, νευρολογία, καρδιολογία και αποκατάσταση υγείας
- Έλεγχος μηχανών: καύση, εναλλαγή αερίων και ψεκασμός, διαγράμματα ενδείξεων
- Βιομηχανία και εργοστάσια: κοπή μετάλλων, αυτοματοποίηση διαδικασιών συναρμολόγησης, παρακολούθηση λειτουργίας μηχανών.

ΚΕΦΑΛΑΙΟ 2^ο

ΨΗΦΙΟΠΟΙΗΣΗ

2.1 Ψηφιακή Εποχή

Όσο η τεχνολογία εξελίσσεται όλο και περισσότερο καλούμαστε να διαχειριστούμε ψηφιακές πληροφορίες όπως, για παράδειγμα, ψηφιακές φωτογραφικές μηχανές, CD, DVD, ψηφιακό βίντεο, ψηφιακό ήχο και ψηφιακά κινητά τηλέφωνα, τα οποία είναι άρρηκτα συνδεδεμένα με τις αρχές της ψηφιοποίησης. Παρόλα αυτά οι περισσότερες από τις πληροφορίες που προέρχονται από φυσικές διεργασίες είναι αναλογικά σήματα. Η ανθρώπινη ομιλία, οι ήχοι που παράγονται από μουσικά όργανα, σήματα βίντεο από κάμερες, τα αποτελέσματα που προκύπτουν από όργανα μέτρησης, όπως σειсмоγράφοι ή θερμόμετρα, κλπ. είναι παραδείγματα αναλογικών σημάτων. Το αναλογικό σήμα χαρακτηρίζεται από το γεγονός ότι παρουσιάζει απόλυτη συνέχεια τόσο στο χρόνο όσο και στην τιμή του πλάτους του. Τα αναλογικά σήματα για να μπορούν να χρησιμοποιηθούν σε ψηφιακά συστήματα χρειάζεται να μετατραπούν σε ψηφιακά μέσω της διαδικασίας που παρουσιάζεται στο σχήμα 2.1. Η έξοδος ενός ψηφιακού συστήματος αποκτά και πάλι μέσω της αντίστροφης διαδικασίας αναλογική μορφή. Ο σκοπός όλου αυτού είναι να αξιοποιηθούν τα χαρακτηριστικά των ψηφιακών συστημάτων. Η επεξεργασία, αποθήκευση και μετάδοση σημάτων σε ψηφιακή μορφή είναι πολύ πιο εύκολη και αποδοτική από ό, τι στην φυσική αναλογική μορφή τους. Ωστόσο, εμείς δεν μπορούμε να αντιληφθούμε τα σήματα απευθείας στο ψηφιακό πεδίο. Αν προσπαθήσουμε να τροφοδοτήσουμε ένα ψηφιοποιημένο ηχητικό σήμα απευθείας σε ένα σύστημα ηχείων ή αντίστοιχα ένα σήμα ψηφιακού βίντεο στην είσοδο βίντεο μιας τηλεόρασης αυτό που θα ακούσουμε ή θα δούμε είναι απλά θόρυβος. Έτσι είναι απαραίτητο να μετατρέψουμε ξανά το ψηφιακά αποθηκευμένο και επεξεργασμένο σήμα στο αρχικό αναλογικό. Ένα βασικό ερώτημα που προκύπτει είναι αν μέσα από αυτές τις διαδικασίες μετατροπής αναλογικό σε ψηφιακό και ψηφιακό σε αναλογικό χάνονται πληροφορίες που αφορούν το αρχικό αναλογικό σήμα. Η απάντηση είναι ότι δεν χάνεται καθόλου πληροφορία αρκεί η μετατροπή να γίνει σωστά λαμβάνοντας υπόψη κάποια βασικά κριτήρια που ισχύουν κατά τη δειγματοληψία. Αν δεν ληφθούν υπόψη αυτά τα κριτήρια χάνουμε σχεδόν όλη την πληροφορία που περιέχεται στο αναλογικό σήμα.

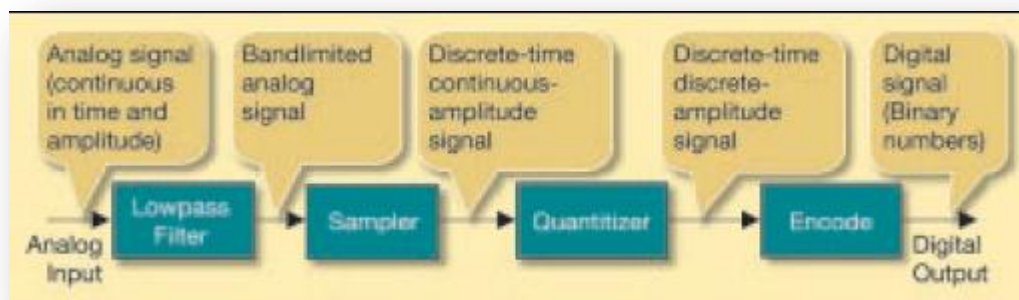


Σχήμα 2.1 Ψηφιακή επεξεργασία αναλογικού σήματος

2.2 Ψηφιοποιώντας αναλογικά σήματα

Ένα ψηφιακό σήμα έχει δύο βασικά χαρακτηριστικά. Πρώτον το σήμα αλλάζει σε τακτά χρονικά διαστήματα και επιπλέον το πλάτος του παίρνει συγκεκριμένες διακριτές τιμές σε αντίθεση με το

αναλογικό σήμα. Η μετατροπή ενός αναλογικού σήματος σε ψηφιακό περιλαμβάνει δύο στάδια. Αρχικά το σήμα μας γίνεται διακριτό στο πεδίο του χρόνου, το οποίο επιτυγχάνεται με τη δειγματοληψία και στη συνέχεια το πλάτος του αποκτά κβαντισμένες τιμές το οποίο προκύπτει μέσω μιας διαδικασίας που ονομάζεται κβαντοποίηση. Το σχήμα 2.2 παριστάνει τη σταδιακή μετατροπή ενός αναλογικού σήματος σε ψηφιακό από έναν analog-to-digital μετατροπέα [4]. Το σήμα διέρχεται διαδοχικά από ένα βαθυπερατό (low-pass) φίλτρο, η σημασία του οποίου θα γίνει σαφέστερη πιο κάτω, έναν δειγματολήπτη, έναν κβαντιστή και έναν κωδικοποιητή.



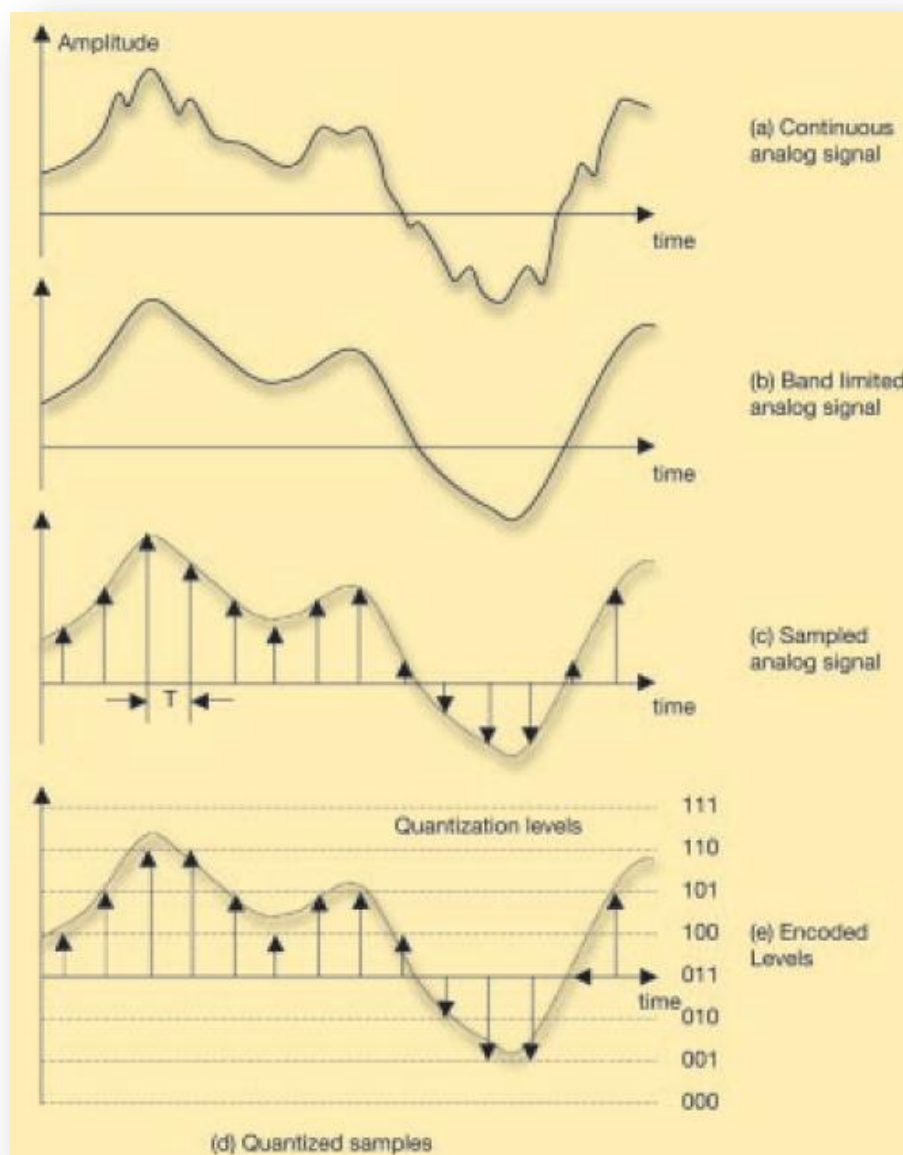
Σχήμα 2.2 Σταδιακή ψηφιοποίηση ενός αναλογικού σήματος από έναν analog-to-digital μετατροπέα (ADC).

Αν και το σήμα στην έξοδο του κβαντιστή θα μπορούσε να χαρακτηριστεί ψηφιακό παραμένει ακόμα ένα πολυεπίπεδο σήμα που αλλάζει σε διακριτά χρονικά διαστήματα. Σε καθένα από τα επίπεδα αυτά εκχωρείται μία δυαδική τιμή μέσω του κωδικοποιητή. Αυτή η δυαδική αναπαράσταση του σήματος που προκύπτει στην έξοδο του κωδικοποιητή αποτελεί την ψηφιακή του μορφή που συναντάται σε όλες τις εφαρμογές. Στο σχήμα 2.3 φαίνεται η μορφή της κυματομορφής που αποκτά κάθε φορά το σήμα στα διάφορα στάδια επεξεργασίας του που απεικονίζονται στο σχήμα 2.2. Η αναλογική είσοδος (a) περιέχει συνιστώσες υψηλών συχνοτήτων οι οποίες αποκόπτονται από το βαθυπερατό φίλτρο. Η ομαλοποιημένη έξοδος του φίλτρου (b) δειγματοληπτείται με περίοδο δειγματοληψίας T (c). Τα δείγματα στη συνέχεια αποκτούν κβαντισμένες τιμές (d) και κωδικοποιούνται (e). Για έναν κβαντιστή οκτώ επιπέδων, όπως φαίνεται και από το σχήμα απαιτείται ένας κωδικοποιητής τριών bit. Είναι φανερό ότι η κβαντοποίηση είναι μια προσεγγιστική διεργασία καθώς οι τιμές που αποδίδονται στα δείγματα προέρχονται από ένα καθορισμένο διακριτό σύνολο τιμών εισάγοντας έτσι ένα σφάλμα με αποτέλεσμα το αρχικό σήμα να μην μπορεί ποτέ να ανακτηθεί ακριβώς από τα δείγματά του. Ωστόσο το σφάλμα αυτό περιορίζεται όσο αυξάνονται τα επίπεδα κβαντοποίησης. Για παράδειγμα αν τα επίπεδα του κβαντιστή στο σχήμα 2.3 αυξηθούν σε 16, το σφάλμα υποδιπλασιάζεται και χρειαζόμαστε 4 bits για την κωδικοποίηση ($2^4=16$). Γενικά n bits μπορούν να κωδικοποιήσουν 2^n επίπεδα. Ο λόγος ισχύς σήματος προς θόρυβο που προκύπτει από την κβαντοποίηση δίνεται από τη σχέση:

$$(S/N) = 6n + 1.8 \text{ dB}$$

Από την εξίσωση αυτή γίνεται φανερό πως διπλασιάζοντας τα επίπεδα κβαντοποίησης και χρησιμοποιώντας ένα επιπλέον bit κατά την κωδικοποίηση βελτιώνεται η απόδοση κατά 6 dB. Τυπικές τιμές του n στην πράξη είναι 8 bits για ομιλία και βίντεο και 16-20 bits για μουσική. Από τη στιγμή που ο αριθμός n έχει επιλεγεί για μια εφαρμογή έχουμε αυτόματα προσδιορίσει και το επίπεδο παραμόρφωσης. Ότι και να κάνουμε ποτέ δεν μπορούμε να εξαλείψουμε τελείως την

παραμόρφωση από το σήμα μας. Δυστυχώς πάντα θα υπάρχει μία έστω και μικρή απώλεια στην ποιότητα του αρχικού αναλογικού σήματος.



Σχήμα 2.3 Μορφή κυματομορφής στα διάφορα στάδια ψηφιοποίησης

Η επόμενη βασική ερώτηση στην οποία καλούμαστε να απαντήσουμε είναι ποια θα πρέπει να είναι η συχνότητα δειγματοληψίας f_s η οποία είναι το αντίστροφο της περιόδου δειγματοληψίας T (σχήμα 2.3). Ο ρυθμός με τον οποίο ένα σήμα πρέπει να δειγματοληπτείται έτσι ώστε να προκύψει μια πλήρης και σαφή ανάκτησή του από τα δείγματά του προτάθηκε για πρώτη φορά από τον Nyquist. Η συχνότητα Nyquist, όπως ονομάζεται, εξαρτάται από την υψηλότερη συχνότητα η οποία είναι παρούσα στο σήμα και δίνεται από την εξίσωση:

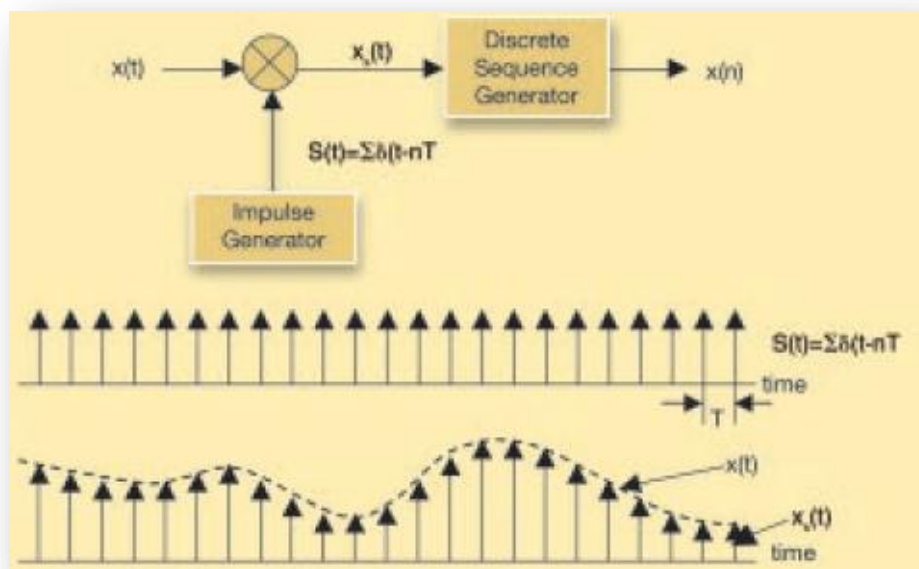
$$f_s \geq 2f_{\max}$$

Αν ένα σήμα δειγματοληπτείται με μια συχνότητα η οποία ικανοποιεί την πιο πάνω συνθήκη μπορούμε να ανακτήσουμε το σήμα χωρίς καμία απώλεια στην αρχική πληροφορία. Αντίθετα με την κβαντοποίηση η δειγματοληψία είναι μια διαδικασία που δεν εισάγει σφάλμα. Τώρα γίνεται πλέον πιο εύκολα αντιληπτό γιατί η παρουσία ενός βαθυπερατού φίλτρου πριν το δειγματολήπτη είναι απαραίτητη (σχήμα 2.2). Η μέγιστη συχνότητα στα αναλογικά σήματα είναι συνήθως υψηλότερη απ' ότι μπορούμε να εκτιμήσουμε. Για παράδειγμα σε τηλεφωνικές εφαρμογές συχνότητες μέχρι 3400 Hz θεωρούνται επαρκείς αν και τα σήματα ομιλίας περιέχουν και συχνότητες που είναι μεγαλύτερες από 3400 Hz. Συχνότητες μεγαλύτερες των 3.4 KHz μπορούν να αφαιρεθούν χωρίς να επηρεαστεί σημαντικά η ποιότητα του σήματος. Το βαθυπερατό φίλτρο κάνει ακριβώς αυτή τη δουλειά, αποκόπτει δηλαδή στη συγκεκριμένη περίπτωση συχνότητες μεγαλύτερες των 3.4 KHz κι έτσι εμείς πρέπει να επιλέξουμε σαν συχνότητα δειγματοληψίας μία συχνότητα μεγαλύτερη των 6.8 KHz ώστε να ικανοποιείται το θεώρημα Nyquist.

Μαθηματικά η δειγματοληψία είναι μια διαδικασία κατά την οποία το αναλογικό σήμα $x(t)$ πολλαπλασιάζεται με το διακριτό σήμα $s(t)$, το οποίο είναι ένα άθροισμα από κρουστικές μετατοπισμένες στο χρόνο, όπως φαίνεται στο σχήμα 2.4. Οι κρουστικές αυτές έχουν τιμή 1 σε χρονικά διαστήματα που απέχουν T μεταξύ τους ενώ οπουδήποτε αλλού έχουν τιμή 0. Έτσι το διακριτό σήμα που προκύπτει δίνεται από τη σχέση:

$$X_s(t) = x(t)s(t) = x(t) \sum_{n=-\infty}^{\infty} \delta(t - nT) = \sum_{n=-\infty}^{\infty} x(t)\delta(t - nT)$$

Από τη στιγμή που οι κρουστικές εμφανίζονται στα διαστήματα nT το γινόμενο $x(t)*s(t)$ έχει μη μηδενική τιμή μόνο στα διαστήματα nT και προφανώς αφού ως γνωστόν το πλάτος των κρουστικών είναι ίσο με 1 το γινόμενο είναι ίσο με την τιμή του $x(t)$ για $t=nT$ ή αλλιώς $x(nT)$.



Σχήμα 2.4 Ο δειγματολήπτης πολλαπλασιάζει το αναλογικό σήμα $x(t)$ με το διακριτό σήμα $s(t)$, το οποίο είναι ένα άθροισμα από κρουστικές μετατοπισμένες στο χρόνο.

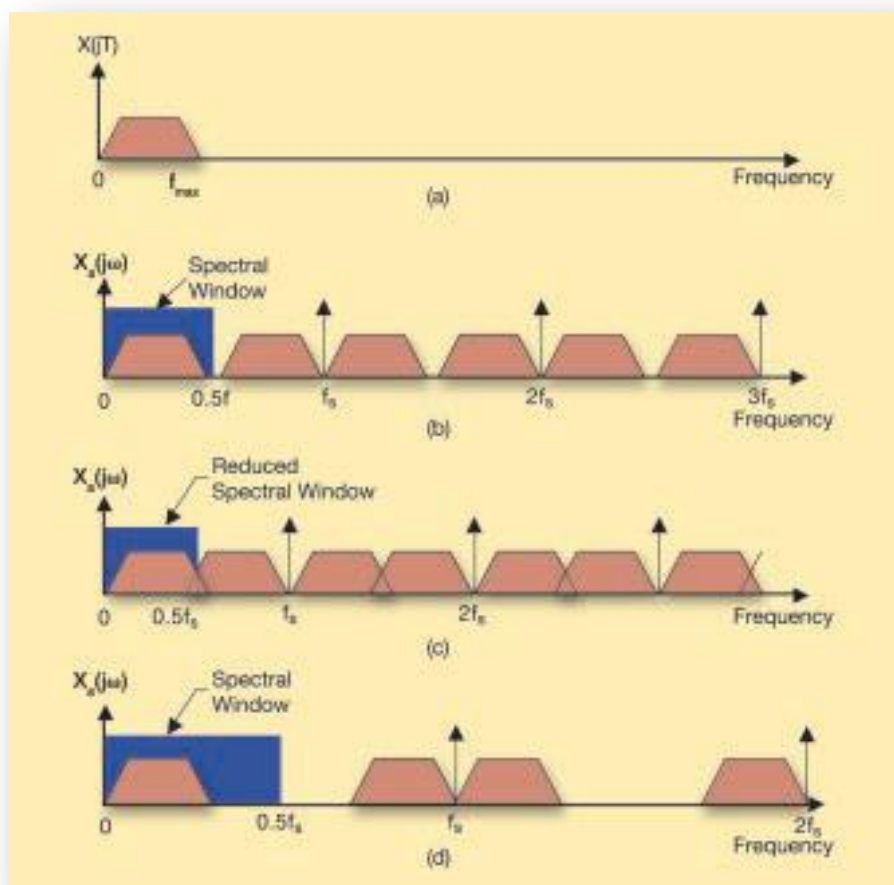
Το πεδίο της συχνότητας μας βοηθά να αντιληφθούμε τι συμβαίνει στην περίπτωση που η επιλεγμένη συχνότητα δειγματοληψίας δεν ικανοποιεί το κριτήριο του Nyquist. Στο σχήμα 2.5 μπορούμε να δούμε την αναπαράσταση ενός σήματος συνεχούς χρόνου στο πεδίο της συχνότητας αλλά και πως αυτό αναπαρίσταται όταν έχει υποστεί δειγματοληψία. Το εργαλείο μας για τη μετάβαση από το πεδίο του χρόνου στο πεδίο της συχνότητας είναι φυσικά ο μετασχηματισμός Fourier, ο οποίος παρουσιάζεται αναλυτικά στο επόμενο κεφάλαιο. Έτσι λοιπόν στην πρώτη περίπτωση εφαρμόζοντας μετασχηματισμό Fourier συνεχούς χρόνου μπορούμε να πάρουμε το πλάτος του σήματος $x(t)$ σε κάθε συχνότητα του φάσματος συχνοτήτων του (σχήμα 2.5(a)). Στο σχήμα 2.5(b) φαίνεται η αναπαράσταση του δειγματοληφθέντος σήματος όπως προκύπτει από τη συνέλιξη του μετασχηματισμού Fourier $X(j\omega)$ του $x(t)$ με τον $S(j\omega)$ του $s(t)$:

$$X_s(j\omega) = \frac{1}{2\pi} X(j\omega) * S(j\omega) = \frac{1}{T} \sum_{K=-\infty}^{\infty} X\{j(\omega - k\omega_s)\}$$

Προφανώς παρότι στο σχήμα 2.5(b) φαίνεται το πλάτος του σήματος σε συχνότητες από το 0 μέχρι $+\infty$ το φάσμα συχνοτήτων του δειγματοληφθέντος σήματος, όπως προκύπτει και από την παραπάνω εξίσωση, εκτείνεται από το $-\infty$ μέχρι το $+\infty$. Παρατηρώντας το σχήμα βλέπουμε ότι το φάσμα συχνοτήτων του σήματος εμφανίζεται εκατέρωθεν της συχνότητας δειγματοληψίας f_s και των αρμονικών της $2f_s$, $3f_s$, κ.ο.κ. Ουσιαστικά παρατηρείται μια διαμόρφωση πλάτους στη συχνότητα δειγματοληψίας και τις αρμονικές της από το αναλογικό σήμα.

Το φασματικό παράθυρο που περιλαμβάνει τις συχνότητες 0 έως $0.5f_s$ Hz είναι και αυτό που μας ενδιαφέρει όταν εξετάζουμε το φασματικό περιεχόμενο ενός σήματος για τον απλούστατο λόγο ότι οτιδήποτε συμβαίνει σ' αυτήν την περιοχή επαναλαμβάνεται σε όλο το φάσμα μέχρι το $+\infty$. Στην πραγματικότητα μπορούμε να δούμε μόνο ότι περικλείεται σ' αυτό το παράθυρο. Ο ψηφιακός μας κόσμος πρακτικά περιορίζεται σ' αυτό το εύρος συχνοτήτων. Εύκολα συμπεραίνουμε ότι αν το σήμα που πρόκειται να υποστεί δειγματοληψία περιέχει συχνότητες μεγαλύτερες από $0.5f_s$ τότε οι διαφορετικές μπάντες συχνοτήτων εκατέρωθεν της συχνότητας δειγματοληψίας και των αρμονικών της θα επικαλύπτονται, προκαλώντας μ' αυτόν τον τρόπο αλλοίωση του φασματικού περιεχομένου. Το φαινόμενο αυτό λέγεται αναδίπλωση και παρουσιάζεται στο σχήμα 2.5(c). Αυτό οφείλεται στο γεγονός ότι όλες οι συχνότητες που περιέχονται στο σήμα και υπερβαίνουν την τιμή $0.5f_s$ αντανakλώνται πίσω στην περιοχή που καλύπτεται από το φασματικό παράθυρο ($0 - 0.5f_s$). Λόγω αυτού ακριβώς του φαινομένου καθίσταται επομένως απαραίτητη η παρουσία ενός βαθυπερατού φίλτρου πριν το δειγματολήπτη με σκοπό να εξασφαλίσουμε ότι το σήμα που θα δειγματοληφθεί δεν θα περιέχει συχνότητες μεγαλύτερες από το μισό της συχνότητας δειγματοληψίας. Το φίλτρο αυτό πρέπει να διαθέτει πολύ απότομη χαρακτηριστική αποκοπής των συχνοτήτων που βρίσκονται κοντά στην τιμή $0.5 f_s$, το οποίο πρακτικά είναι πολύ δύσκολο. Αυτός είναι και ο λόγος που σχεδόν σ' όλες τις εφαρμογές επιλέγουμε ρυθμό δειγματοληψίας μεγαλύτερο από το διπλάσιο της μέγιστης συχνότητας του σήματος. Στο σχήμα 2.5(d) έχει επιλεγεί μία συχνότητα δειγματοληψίας πολύ μεγαλύτερη του ρυθμού δειγματοληψίας κατά Nyquist με αποτέλεσμα να υπάρχει ένα μεγάλο διάστημα μεταξύ των διαφορετικών μπαντών στις διάφορες αρμονικές που αποτελούν το φάσμα συχνοτήτων του σήματος. Αν και αυτό διευκολύνει το σχεδιασμό του φίλτρου μας, το οποίο μπορεί πλέον να έχει μια πιο αργή χαρακτηριστική κατά τη μετάβασή του σε συχνότητες μεγαλύτερες του μισού ρυθμού δειγματοληψίας, εν τούτοις προκύπτει η ανάγκη για μεγαλύτερη ταχύτητα λειτουργίας του κβαντιστή και του κωδικοποιητή. Μεγαλύτερη συχνότητα δειγματοληψίας σημαίνει αυτόματα

μικρότερη περίοδο δειγματοληψίας ($T_s=1/f_s$), η οποία είναι καθοριστική σε εφαρμογές ψηφιακής επεξεργασίας σήματος αφού καθορίζει την ταχύτητα λειτουργίας όλων των συσκευών που σχετίζονται με την εφαρμογή.

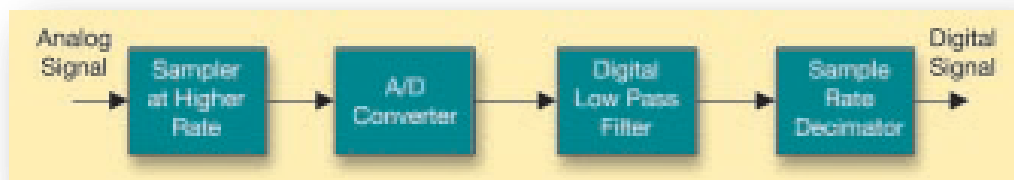


Σχήμα 2.5 (α) Φάσμα αναλογικού σήματος, (β) Φάσμα σήματος που έχει υποστεί δειγματοληψία με συχνότητα λίγο μεγαλύτερη της συχνότητας Nyquist, (γ) Φάσμα σήματος που έχει υποστεί δειγματοληψία με συχνότητα μικρότερη της συχνότητας Nyquist, (δ) Φάσμα σήματος που έχει υποστεί δειγματοληψία με συχνότητα πολύ μεγαλύτερη της συχνότητας Nyquist

2.3 Υπερδειγματοληψία

Σε κάποιες περιπτώσεις ανάλογα με τη φύση της εφαρμογής είναι δυνατό να αντικαταστήσουμε το αναλογικό βαθυπερατό φίλτρο που παρεμβάλλεται πριν τον δειγματολήπτη με ένα ψηφιακό φίλτρο που εφαρμόζεται στο ψηφιακό σήμα στην έξοδο του κωδικοποιητή [4]. Στην περίπτωση μας όπου χρησιμοποιείται ένας μικροελεγκτής αυτό το φίλτρο μπορεί να εφαρμοστεί στις ψηφιοποιημένες τιμές του σήματος που προκύπτουν μετά την επεξεργασία του από τον Analog-to-Digital Converter. Κατά κάποιο τρόπο αυτό αλλοιώνει τη θεμελιώδη σημασία του αναλογικού βαθυπερατού φίλτρου στην εξάλειψη του φαινομένου της αναδίπλωσης ωστόσο εφαρμόζεται κάποιες φορές. Προφανώς απ' τη στιγμή που δεν υπάρχει το αναλογικό φίλτρο ώστε να αποκόψει από το αναλογικό σήμα τις

συχνότητες που υπερβαίνουν το μισό της συχνότητας δειγματοληψίας καθίσταται απαραίτητο να επιλέξουμε έναν ρυθμό δειγματοληψίας πολύ μεγαλύτερο του ρυθμού Nyquist. Αφού το σήμα περάσει από το ψηφιακό φίλτρο μπορούμε να μειώσουμε το ρυθμό δειγματοληψίας του (σχήμα 2.6).

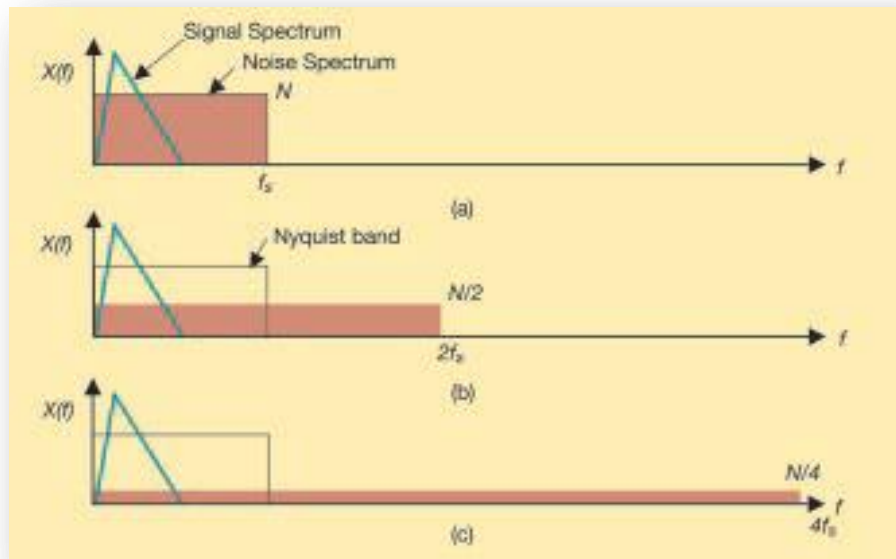


Σχήμα 2.6 Κατά την υπερδειγματοληψία είναι δυνατή η αντικατάσταση του αναλογικού low-pass φίλτρου με ένα αντίστοιχο ψηφιακό

Αυτή η τεχνική κάνει τη ζωή του σχεδιαστή πιο εύκολη αφού δεν απαιτείται πλέον η παρουσία ενός αναλογικού φίλτρου με πολύ απότομη χαρακτηριστική αποκοπής. Για παράδειγμα σε ένα σύστημα ψηφιακού ήχου FM καλής ποιότητας που έχει συχνότητα δειγματοληψίας 32KHz μια τυπική απαίτηση είναι η χαρακτηριστική απόσβεσης του φίλτρου να πέφτει 60-80 dB στο διάστημα 15KHz έως 16KHz, από το οποίο μπορούμε να συμπεράνουμε ότι ο σχεδιασμός ενός τέτοιου αναλογικού φίλτρου σίγουρα δεν είναι εύκολος. Όταν ένα σήμα δειγματοληπτείται με συχνότητα f_s το φάσμα του θορύβου που προκύπτει από την κβαντοποίηση καταλαμβάνει όλο το εύρος από 0- f_s . Όταν επιλέγουμε μεγαλύτερη συχνότητα δειγματοληψίας το φάσμα του θορύβου (εξαρτάται μόνο από τον αριθμό των bits που χρησιμοποιούνται) εκτείνεται μέχρι την νέα τιμή της f_s όπως φαίνεται στο σχήμα 2.7. Όπως προκύπτει από το σχήμα η πυκνότητα N του θορύβου για συχνότητα δειγματοληψίας f_s (α) μειώνεται στην τιμή $N/2$ για $f_s' = 2f_s$ (b) ενώ για $f_s'' = 4f_s$ γίνεται $N/4$ (c). Σ' όλες τις παραπάνω περιπτώσεις η συνολική ισχύς του θορύβου διατηρείται σταθερή. Ο λόγος ισχύς σήματος προς θόρυβο που αναφέρθηκε και πιο πάνω τροποποιείται στην περίπτωση που η συχνότητα δειγματοληψίας μπορεί να μεταβληθεί. Έτσι έχουμε:

$$(S / N)_{dB} = 6.02n + 1.8 + 10 \log(f_s / 2)$$

Από την εξίσωση αυτή εύκολα συμπεραίνουμε ότι κάθε φορά που ο ρυθμός δειγματοληψίας διπλασιάζεται ο λόγος ισχύς σήματος προς θόρυβο αυξάνεται κατά 3 dB.

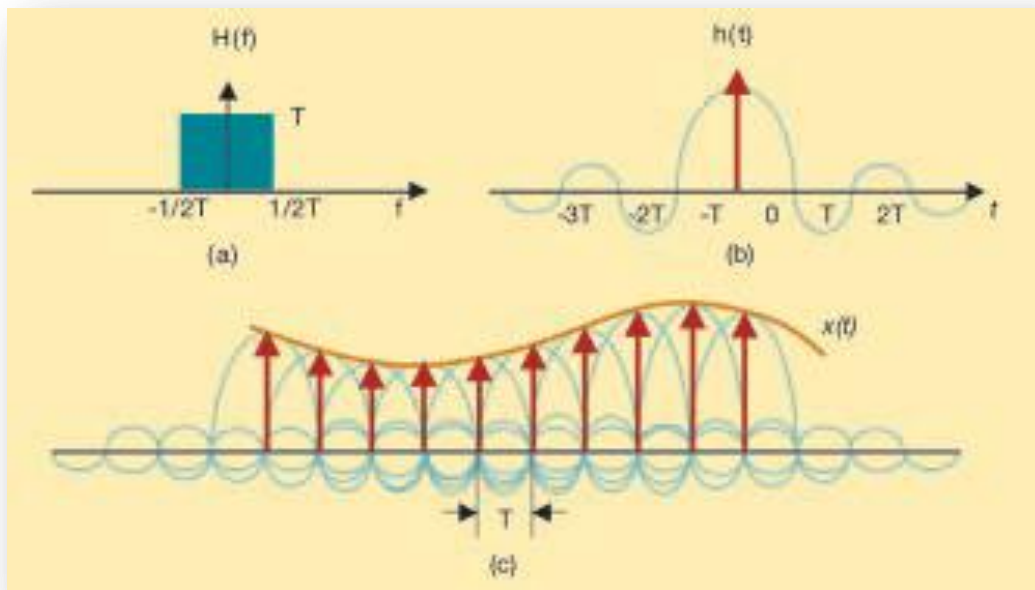


Σχήμα 2.7 (a) Ο θόρυβος που προκύπτει από την κβαντοποίηση εκτείνεται ομοιόμορφα με φασματική πυκνότητα N μέχρι την τιμή της συχνότητας δειγματοληψίας, (b) όταν ο ρυθμός δειγματοληψίας διπλασιάζεται η φασματική πυκνότητα του θορύβου μειώνεται στο μισό ($N/2$), (c) αντίστοιχα όταν η συχνότητα δειγματοληψίας γίνεται $4f_s$ η πυκνότητα του θορύβου μειώνεται σε $N/4$

2.4 Ανακτώντας το σήμα από τα δείγματά του

Η ανάκτηση ενός σήματος από τα δείγματά του γίνεται με την εφαρμογή ενός ακόμη βαθυπερατού φίλτρου [4]. Ένα ιδανικό βαθυπερατό φίλτρο με συχνότητα αποκοπής $0.5f_s = 1/2T$ (T είναι η περίοδος δειγματοληψίας) φαίνεται στο σχήμα 2.8 (a). Η κρουστική απόκριση του συγκεκριμένου φίλτρου έχει τη μορφή που φαίνεται στο σχήμα 2.8 (b). Η κρουστική απόκριση του φίλτρου είναι η έξοδος του για κρουστική είσοδο. Όπως φαίνεται και από το σχήμα όταν στο φίλτρο εφαρμόζεται σαν είσοδος μια κρουστική, η οποία είναι ένας πολύ στενός παλμός (ιδανικά μηδενικού φάρδους και απείρου πλάτους και ενέργειας) η έξοδος του αποκτά μια ημιτονοειδή μορφή, εμφανίζοντας ένα μεγάλο πλάτος για $t=0$ και στη συνέχεια ταλαντώνεται με όλο και μικρότερο πλάτος μέχρι το άπειρο. Η έξοδος που προκύπτει σ' αυτή την περίπτωση ορίζεται μαθηματικά σαν μία συνάρτηση sinc, η οποία εκφράζεται ως $\text{sinc}(x) = \sin(x)/x$. Η συνάρτηση αυτή έχει μηδενική τιμή στα διαστήματα nT , όπου n ένας οποιοσδήποτε θετικός ή αρνητικός ακέραιος. Πρέπει να τονιστεί πως ενώ από το σχήμα μας φαίνεται ότι η θέση της κρουστικής συμπίπτει με την τιμή 0 της εξόδου για την οποία έχουμε το μέγιστο πλάτος, στην πραγματικότητα δεν συμβαίνει κάτι τέτοιο αφού υπάρχει μια καθυστέρηση μεταξύ εισόδου και εξόδου. Η μέγιστη τιμή της κυματομορφής sinc είναι ανάλογη της ενέργειας της κρουστικής. Όπως έχει ήδη αναφερθεί το δειγματοληφθέν σήμα είναι μια σειρά από κρουστικές με ύψος ανάλογο της τιμής του σήματος τη στιγμή που δειγματοληπτείται. Η διέλευση αυτών των κρουστικών που αναπαριστούν τα διάφορα δείγματα μέσα από ένα τέτοιο φίλτρο έχει σαν αποτέλεσμα τη δημιουργία πολλών επικαλυπτόμενων τέτοιων σημάτων με ημιτονοειδή μορφή καθένα από τα οποία παρουσιάζει μέγιστο πλάτος για διαφορετικές τιμές του t . Η σύνθεση όλων

αυτών των sinc κυματομορφών έχει σαν αποτέλεσμα τελικά τη συγκρότηση του αρχικού αναλογικού σήματος $x(t)$.



Σχήμα 2.8 (a) Απόκριση συχνότητας ενός ιδανικού low-pass φίλτρου για ανακατασκευή σήματος, (b) κρουστική απόκριση του φίλτρου, (c) ανάκτηση σήματος μέσω της συνέλιξης των δειγμάτων με την κρουστική απόκριση του φίλτρου

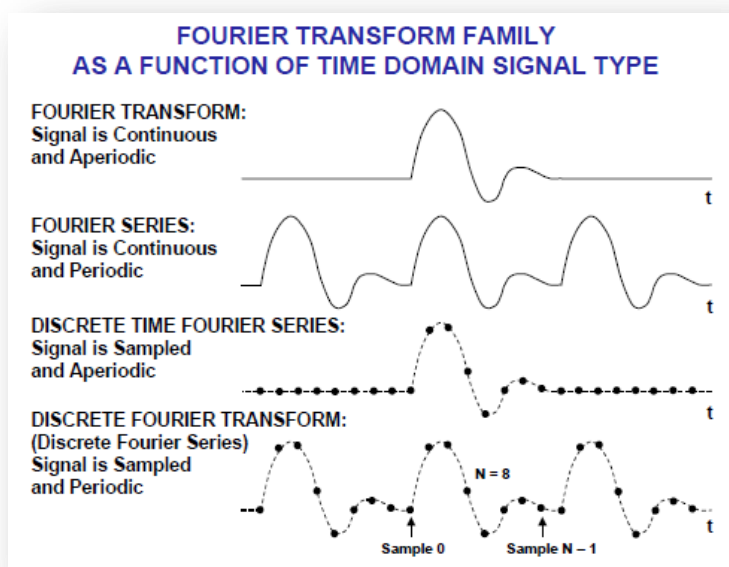
ΚΕΦΑΛΑΙΟ 3^ο

ΜΕΘΟΔΟΙ ΜΕΤΑΒΑΣΗΣ ΣΤΟ ΠΕΔΙΟ ΤΗΣ ΣΥΧΝΟΤΗΤΑΣ

3.1 Γενικά

Το 1807 ο Γάλλος μαθηματικός και φυσικός Jean Baptiste Fourier παρουσίασε μια επιστημονική εργασία στο Institut de France με αντικείμενο τη χρησιμοποίηση ημιτονοειδών σημάτων για την αναπαράσταση κατανομών θερμοκρασίας. Ο Fourier στα πλαίσια της εργασίας αυτής διατύπωσε το θεώρημα ότι κάθε συνεχές περιοδικό σήμα μπορεί να αναπαρασταθεί με την πρόσθεση κατάλληλα επιλεγμένων ημιτονοειδών κυμάτων. Παρά την αντίθεση του Joseph Louis Lagrange ο οποίος υποστήριξε ότι η προσέγγιση του Fourier δεν ισχύει για σήματα που παρουσιάζουν ασυνέχεια όσον αφορά την κλίση της κυματομορφής τους, όπως οι τετραγωνικοί παλμοί, η υπόθεση αυτή του Fourier αποτέλεσε τη βάση για την ψηφιακή επεξεργασία σήματος και χρησιμοποιείται ευρύτατα από επιστήμονες, μηχανικούς και μαθηματικούς. Ο μετασχηματισμός Fourier θα μπορούσαμε να πούμε απλά ότι επιτρέπει τη μετατροπή ενός σήματος στο πεδίο του χρόνου στο αντίστοιχό του στο πεδίο της συχνότητας. Αντίστροφα, αν η απόκριση συχνότητας ενός σήματος είναι γνωστή, ο αντίστροφος μετασχηματισμός Fourier χρησιμοποιείται για τον προσδιορισμό της αναπαράστασης του σήματος στο πεδίο του χρόνου. Εκτός από την ανάλυση συχνότητας οι μετασχηματισμοί αυτοί χρησιμοποιούνται στο σχεδιασμό φίλτρων αφού η απόκριση συχνότητας ενός φίλτρου μπορεί να προκύψει από το μετασχηματισμό Fourier της κρουστικής του απόκρισης. Αντίστοιχα πάλι, αν είναι γνωστή η απόκριση συχνότητάς του τότε η κρουστική απόκριση του φίλτρου μπορεί να υπολογιστεί από τον αντίστροφο μετασχηματισμό Fourier της απόκρισης συχνότητας. Μπορούμε να κατασκευάσουμε φίλτρα βασιζόμενοι στην κρουστική τους απόκριση αφού οι συντελεστές ενός ψηφιακού FIR φίλτρου και της κρουστικής του απόκρισης είναι ταυτόσημοι.

Η οικογένεια των μετασχηματισμών Fourier (Μετασχηματισμός Fourier, Σειρά Fourier, Διακριτού Χρόνου Μετασχηματισμός Fourier και Διακριτός Μετασχηματισμός Fourier) φαίνεται στο σχήμα 3.1. Αυτοί οι κοινώς αποδεκτοί ορισμοί έχουν εξελιχθεί (όχι απαραίτητα λογικά) με το πέρασμα των χρόνων και εξαρτώνται από το αν το σήμα είναι συνεχές-απεριοδικό, συνεχές-περιοδικό, διακριτό-απεριοδικό ή διακριτό-περιοδικό [9].



Σχήμα 3.1 Είδη μετασχηματισμών Fourier

3.2 Διακριτός Μετασχηματισμός Fourier (DFT)

Το μόνο μέλος αυτής της οικογένειας που σχετίζεται με την ψηφιακή επεξεργασία σήματος είναι ο Διακριτός Μετασχηματισμός Fourier (DFT), ο οποίος εφαρμόζεται σε ένα διακριτό σήμα στο πεδίο του χρόνου που είναι και περιοδικό [7] [9]. Το σήμα πρέπει να είναι περιοδικό ώστε να μπορεί να αναλυθεί σε ένα άθροισμα ημιτόνων. Ωστόσο μόνο ένας πεπερασμένος αριθμός δειγμάτων (N) είναι κατάλληλος για επεξεργασία μέσω του DFT. Το δίλημμα αυτό μπορεί να ξεπεραστεί με την τοποθέτηση άπειρων πακέτων των ίδιων N δειγμάτων το ένα μετά το άλλο, δημιουργώντας έτσι μια μαθηματική (όχι πραγματική) περιοδικότητα, όπως φαίνεται και από το σχήμα 3.1.

Η θεμελιώδης εξίσωση στην ανάλυση Fourier για την δημιουργία ενός N δειγμάτων DFT είναι η ακόλουθη:

$$X(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{-j \frac{2\pi nk}{N}} = \frac{1}{N} \sum_{n=0}^{N-1} x(n) \left[\cos \frac{2\pi nk}{N} - j \sin \frac{2\pi nk}{N} \right]$$

$0 \leq k \leq N-1$

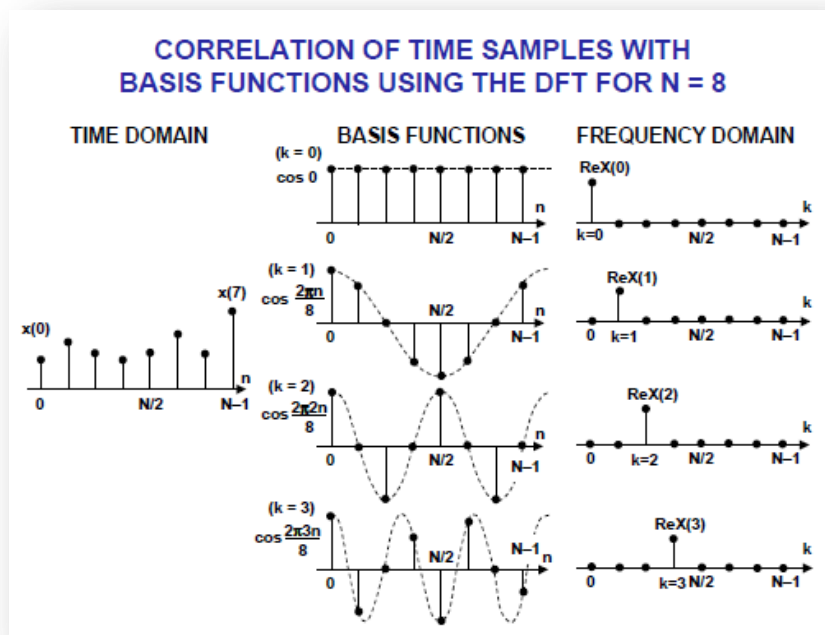
Το $X(k)$ αναπαριστά την τιμή του σήματος στο πεδίο της συχνότητας για $f=k$ (το k παίρνει τιμές από 0 έως $N-1$) όπως προκύπτει από τον DFT. Ο αριθμός N αντιπροσωπεύει τον αριθμό των δειγμάτων που επεξεργάζονται κάθε φορά από τον DFT. Η ποσότητα $x(n)$ αναπαριστά το n -οστό δείγμα στο πεδίο του χρόνου και το n παίρνει πάλι τιμές από 0 έως $N-1$. Το $x(n)$ μπορεί να είναι είτε πραγματικός είτε μιγαδικός. Είναι επίσης προφανές ότι εφαρμόζοντας την εξίσωση του Euler, που φαίνεται πιο κάτω, μπορούμε να εκφράσουμε τον DFT ενός σήματος με δύο διαφορετικούς τρόπους.

$$e^{j\theta} = \cos \theta + j \sin \theta$$

Το φάσμα συχνοτήτων που προκύπτει με την εφαρμογή του DFT είναι η αλληλοσυσχέτιση των N χρονικών δειγμάτων και των N ημιτονοειδών και συνημιτονοειδών κυμάτων. Αυτό φαίνεται καλύτερα στο σχήμα 3.2. Για τον υπολογισμό εδώ του πραγματικού μέρους των τεσσάρων πρώτων σημείων του φάσματος χρησιμοποιούνται τα αντίστοιχα συνημίτονα. Ομοίως για τον υπολογισμό του φανταστικού μέρους κάθε σημείου του φάσματος ακολουθείται η ίδια διαδικασία με τα αντίστοιχα ημίτονα.

Πιο αναλυτικά το πρώτο σημείο $X(0)$ είναι απλά το άθροισμα των N χρονικών δειγμάτων αφού $\cos(0)=1$. Ο όρος $1/N$ δεν εμφανίζεται αλλά θα πρέπει να ληφθεί υπ' όψη στο τελικό αποτέλεσμα. Εύκολα συμπεραίνουμε ότι το $X(0)$ είναι η μέση τιμή των χρονικών δειγμάτων ή απλά η DC συνιστώσα. Για τον υπολογισμό του πραγματικού μέρους του δεύτερου σημείου $\text{Re}X(1)$ πολλαπλασιάζουμε κάθε δείγμα με το αντίστοιχο σημείο στο συνημίτονο για $k=1$ το οποίο συμπληρώνει, όπως φαίνεται και από το σχήμα, έναν πλήρη κύκλο για N δείγματα και στη συνέχεια προσθέτουμε τους όρους που προκύπτουν. Ομοίως το $\text{Re}X(2)$ προκύπτει πολλαπλασιάζοντας κάθε δείγμα με το αντίστοιχο σημείο του συνημιτόνου για $k=2$ το οποίο συμπληρώνει δύο πλήρεις κύκλους για N δείγματα και προσθέτοντας πάλι τα αποτελέσματα. Αντίστοιχα υπολογίζεται και το $\text{Re}X(3)$. Με τον ίδιο τρόπο υπολογίζονται τα πραγματικά μέρη όλων των N σημείων του φάσματος. Η ίδια συλλογιστική εφαρμόζεται και για των υπολογισμό των φανταστικών μερών όλων των

σημείων του φάσματος με τη διαφορά ότι οι πολλαπλασιασμοί γίνονται με τα σημεία των αντίστοιχων ημιτόνων. Τα συνημίτονα και τα ημίτονα συχνά αναφέρονται ως βασικές συναρτήσεις.



Σχήμα 3.2 Συσχέτιση μεταξύ χρονικών δειγμάτων και βασικών συναρτήσεων

Ο αντίστροφος DFT(IDFT) χρησιμοποιείται για την ανάκτηση των χρονικών δειγμάτων του σήματος από το φασματικό του περιεχόμενο. Η διαδικασία που ακολουθούμε είναι παρόμοια με αυτήν που περιγράφηκε πιο πάνω για τον υπολογισμό του πραγματικού και του φανταστικού μέρους κάθε σημείου του φάσματος συχνοτήτων. Η εξίσωση του IDFT είναι η ακόλουθη:

$$x(n) = \sum_{k=0}^{N-1} X(k)e^{j2\pi nk/N} = \sum_{k=0}^{N-1} X(k)[\cos(2\pi nk/N) + j \sin(2\pi nk/N)]$$

Υπάρχουν δύο βασικοί τύποι DFT: ο real DFT και ο complex DFT. Ο πρώτος επεξεργάζεται πραγματικούς αριθμούς ενώ ο δεύτερος μιγαδικούς. Οι εξισώσεις που παρουσιάζονται στο σχήμα 3.3 αναφέρονται στον complex DFT όπου τόσο τα δείγματα στην είσοδο όσο και τα δείγματα στην έξοδο είναι μιγαδικοί. Από τη στιγμή που τα δείγματά μας στο πεδίο του χρόνου είναι πραγματικοί αριθμοί θέτουμε το φανταστικό μέρος κάθε δείγματος ίσο με το μηδέν. Τα δείγματα στην έξοδο του DFT, $X(k)$, ανεξάρτητα με τις τιμές που παίρνουν τα δείγματα στην είσοδο έχουν πάντα και πραγματικό και φανταστικό μέρος, μέσω των οποίων μπορούμε να υπολογίσουμε το πλάτος και τη φάση του καθενός από αυτά.

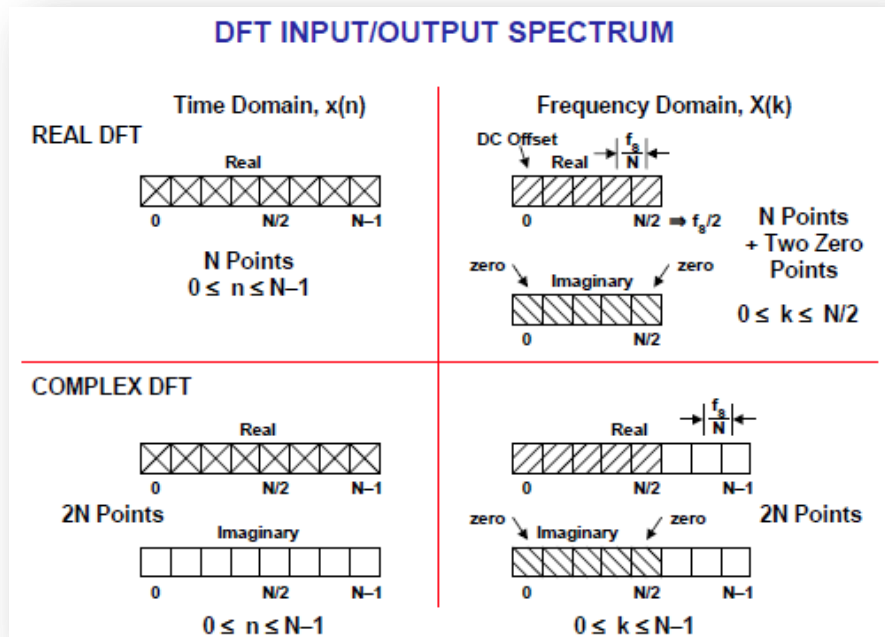
THE COMPLEX DFT

$$\begin{array}{lcl}
 \text{Frequency Domain} & \leftarrow \leftarrow & \text{DFT} & \leftarrow \leftarrow & \text{Time Domain} \\
 X(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N} & = & \frac{1}{N} \sum_{n=0}^{N-1} x(n) \left[\cos \frac{2\pi nk}{N} - j \sin \frac{2\pi nk}{N} \right] \\
 & & = \frac{1}{N} \sum_{n=0}^{N-1} x(n) W_N^{nk}, \quad 0 \leq k \leq N-1 \\
 & & \boxed{W_N = e^{-j2\pi/N}}
 \end{array}$$

$$\begin{array}{lcl}
 \text{Time Domain} & \leftarrow \leftarrow & \text{INVERSE DFT} & \leftarrow \leftarrow & \text{Frequency Domain} \\
 x(n) = \sum_{k=0}^{N-1} X(k) e^{j2\pi nk/N} & = & \sum_{k=0}^{N-1} X(k) \left[\cos \frac{2\pi nk}{N} + j \sin \frac{2\pi nk}{N} \right] \\
 & = & \sum_{k=0}^{N-1} X(k) W_N^{-nk}, \quad 0 \leq n \leq N-1
 \end{array}$$

Σχήμα 3.3 Εξισώσεις που χρησιμοποιούνται στον complex DFT

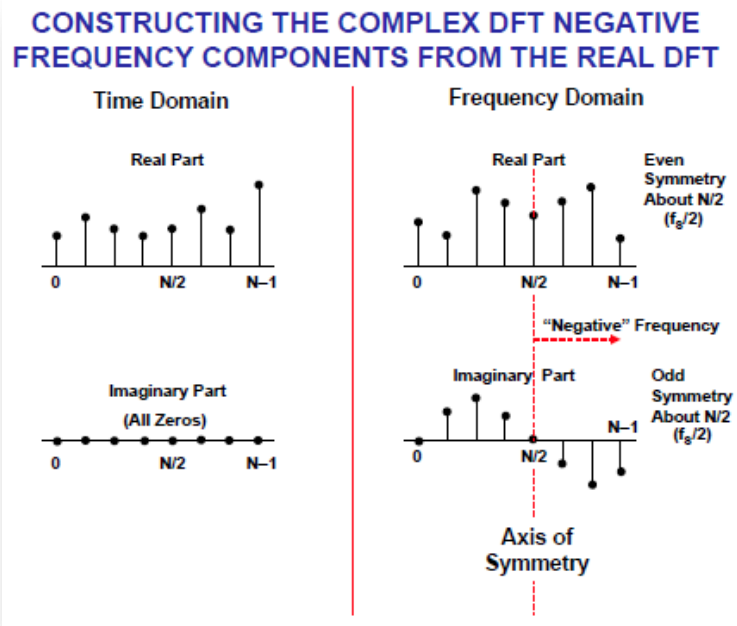
Ο real DFT, αν και κάπως απλούστερος, είναι ουσιαστικά μια απλοποίηση του complex DFT [7] [9]. Οι περισσότεροι κώδικες γρήγορου μετασχηματισμού Fourier (FFT), τον οποίο θα εξετάσουμε εκτενέστερα πιο κάτω, είναι γραμμένοι έχοντας σαν βάση τον complex DFT. Συνεπώς είναι πολύ σημαντική η κατανόηση του complex DFT και πως αυτός σχετίζεται με τον real. Για παράδειγμα αν ξέρουμε τις τιμές των σημείων του φάσματος ενός σήματος στο οποίο έχει εφαρμοστεί ο real DFT και θέλουμε να χρησιμοποιήσουμε τον complex IDFT για να υπολογίσουμε τις τιμές των δειγμάτων στο πεδίο του χρόνου είναι απαραίτητο να γνωρίζουμε πως από τον real DFT μπορεί να προκύψει ο complex. Στο σχήμα 3.4 φαίνονται οι είσοδοι και οι έξοδοι ενός real και ενός complex DFT. Παρατηρούμε ότι οι έξοδοι του real έχουν και πραγματικό και φανταστικό μέρος ενώ το k κυμαίνεται από 0 έως $N/2$. Τα φανταστικά μέρη $\text{Im}X(0)$ και $\text{Im}X(N/2)$ είναι πάντα 0 αφού $\sin(0)=0$ και $\sin(n\pi)=0$. Η τιμή $X(N/2)$ του φάσματος αντιστοιχεί σε συχνότητα που είναι ίση με το μισό της συχνότητας δειγματοληψίας f_s . Η απόσταση δύο διαδοχικών σημείων του άξονα συχνοτήτων είναι ίση με f_s/N .



Σχήμα 3.4 Είσοδοι και έξοδοι ενός real και ενός complex DFT

Για τον complex DFT τόσο οι τιμές που δέχεται σαν είσοδο όσο και αυτές που προκύπτουν μετά τον μετασχηματισμό έχουν και πραγματικά και φανταστικά μέρη. Οι γραμμοσκιασμένες περιοχές στο σχήμα είναι αυτές που είναι κοινές και για τους δύο μετασχηματισμούς.

Στο σχήμα 3.5 μπορούμε να διακρίνουμε τη σχέση που συνδέει τον real με τον complex DFT πιο λεπτομερώς. Στο φάσμα συχνοτήτων που προκύπτει από τον real DFT υπάρχουν μόνο τα σημεία 0 έως $N/2$ με το $\text{Im}X(0)$ και το $\text{Im}X(N/2)$ να είναι πάντα ίσα με το 0. Τα σημεία μεταξύ $N/2$ και $N-1$ περιέχουν τις αρνητικές συχνότητες στον complex DFT. Παρατηρούμε ότι το $\text{Re}X(N/2+1)$ έχει την ίδια τιμή με το $\text{Re}X(N/2-1)$. Ομοίως το $\text{Re}X(N/2+2)$ έχει την ίδια τιμή με το $\text{Re}X(N/2-2)$ κοκ. Επίσης όσον αφορά τα φανταστικά μέρη των σημείων του φάσματος ισχύει ότι και για τα πραγματικά με τη διαφορά ότι έχουν αντίθετα πρόσημα. Πιο συγκεκριμένα $\text{Im}X(N/2+1) = -\text{Im}X(N/2-1)$, $\text{Im}X(N/2+2) = -\text{Im}X(N/2-2)$ κοκ. Αξιοποιώντας αυτές τις σχέσεις είναι εύκολο γνωρίζοντας τα πραγματικά και τα φανταστικά μέρη των θετικών συχνοτήτων να δημιουργήσουμε αυτά των αρνητικών.



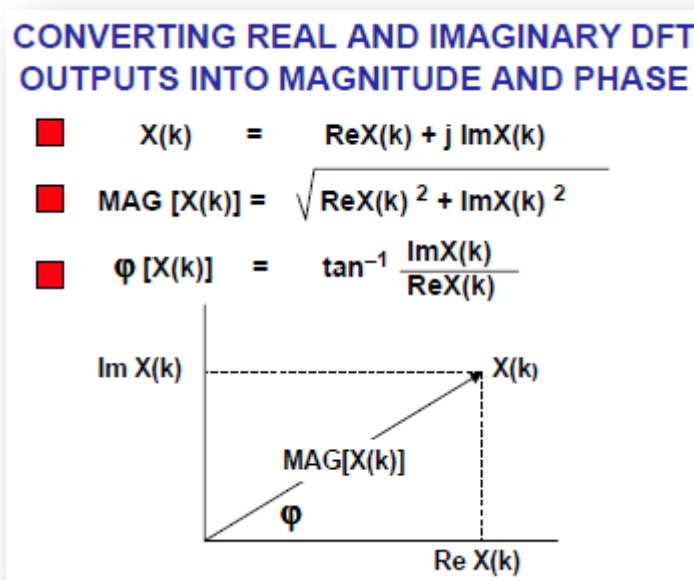
Σχήμα 3.5 Υπολογισμός των αρνητικών συχνοτήτων ενός complex DFT από τον real DFT

Οι εξισώσεις για τον real και τον complex DFT συνοψίζονται στο σχήμα 3.6. Πρέπει να επισημανθεί ότι οι εξισώσεις για τον complex DFT λειτουργούν περίπου με τον ίδιο τρόπο είτε χρησιμοποιώντας τον complex DFT είτε τον IDFT. Ο real DFT δεν χρησιμοποιεί μιγαδικούς και οι εξισώσεις για τον υπολογισμό του $X(k)$ με αυτές για τον υπολογισμό του $x(n)$ παρουσιάζουν διαφορές. Επίσης πριν την χρησιμοποίηση της εξίσωσης για τον υπολογισμό του $x(n)$ το $\text{Re}X(0)$ και το $\text{Re}X(N/2)$ πρέπει να διαιρεθεί με το 2.

COMPLEX AND REAL DFT EQUATIONS	
COMPLEX TRANSFORM	REAL TRANSFORM
$X(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N}$	$\text{Re}X(k) = \frac{2}{N} \sum_{n=0}^{N-1} x(n) \cos(2\pi nk/N)$
$x(n) = \sum_{k=0}^{N-1} X(k) e^{j2\pi nk/N}$	$\text{Im}X(k) = \frac{-2}{N} \sum_{n=0}^{N-1} x(n) \sin(2\pi nk/N)$
	$x(n) = \sum_{k=0}^{N/2} \left[\text{Re}X(k) \cos(2\pi nk/N) - \text{Im}X(k) \sin(2\pi nk/N) \right]$
Time Domain: $x(n)$ is complex, discrete, and periodic. n runs from 0 to $N-1$ Frequency Domain: $X(k)$ is complex, discrete, and periodic. k runs from 0 to $N-1$ $k = 0$ to $N/2$ are positive frequencies. $k = N/2$ to $N-1$ are negative frequencies	Time Domain: $x(n)$ is real, discrete, and periodic. n runs from 0 to $N-1$ Frequency domain: $\text{Re}X(k)$ is real, discrete, and periodic. $\text{Im}X(k)$ is real, discrete, and periodic. k runs from 0 to $N/2$ Before using $x(n)$ equation, $\text{Re}X(0)$ and $\text{Re}X(N/2)$ must be divided by two.

Σχήμα 3.6 Εξισώσεις για τον complex και τον real DFT

Το φάσμα συχνοτήτων που προκύπτει από τον DFT μπορεί να αναπαρασταθεί είτε σε πολικές (πλάτος και φάση) είτε σε καρτεσιανές συντεταγμένες (πραγματικό και φανταστικό μέρος) όπως φαίνεται και στο σχήμα 3.7. Η μετατροπή από τη μία μορφή στην άλλη γίνεται απ' ευθείας.



Σχήμα 3.7 Υπολογισμός πλάτους και φάσης από τις τιμές των εξόδων ενός DFT

3.3 Γρήγορος Μετασχηματισμός Fourier (FFT)

Για να κατανοήσουμε τη λειτουργία του FFT ας θεωρήσουμε πρώτα το ανάπτυγμα του DFT 8 σημείων που φαίνεται στο σχήμα 3.8 [6] [7] [9] [10] [11]. Για να απλοποιήσουμε το διάγραμμα ορίζουμε την ποσότητα W_N ως:

$$W_N = e^{-j2\pi/N}$$

Αυτό μας οδηγεί στον ορισμό των twiddle factors ως:

$$W_N^{nk} = e^{-j2\pi nk/N}$$

Οι twiddle factors είναι απλά οι ημιτονοειδείς και συνημιτονοειδείς βασικές συναρτήσεις γραμμένες σε πολική μορφή. Πρέπει να επισημανθεί ότι για τον υπολογισμό του DFT 8 σημείων που φαίνεται στο διάγραμμα χρειαζόμαστε 64 πολλαπλασιασμούς μιγαδικών. Γενικά ένας N σημείων DFT απαιτεί N^2 πολλαπλασιασμούς μιγαδικών. Ο αριθμός των πολλαπλασιασμών που απαιτούνται είναι σημαντικός μιας και ο χρόνος εκτέλεσης μιας πράξης πολλαπλασιασμού σε ένα σύστημα ψηφιακής

επεξεργασίας σήματος (DSP) είναι σχετικά μεγάλος. Στην πραγματικότητα ο συνολικός χρόνος που απαιτείται για τον υπολογισμό ενός DFT είναι ανάλογος του αριθμού των πολλαπλασιασμών που εμπεριέχονται σ' αυτόν.

THE 8-POINT DFT (N = 8)

$$X(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N} = \frac{1}{N} \sum_{n=0}^{N-1} x(n) W_N^{nk}$$

$W_N = e^{-j2\pi/N}$

$X(0) =$	$x(0)W_8^0 + x(1)W_8^0 + x(2)W_8^0 + x(3)W_8^0 + x(4)W_8^0 + x(5)W_8^0 + x(6)W_8^0 + x(7)W_8^0$
$X(1) =$	$x(0)W_8^0 + x(1)W_8^1 + x(2)W_8^2 + x(3)W_8^3 + x(4)W_8^4 + x(5)W_8^5 + x(6)W_8^6 + x(7)W_8^7$
$X(2) =$	$x(0)W_8^0 + x(1)W_8^2 + x(2)W_8^4 + x(3)W_8^6 + x(4)W_8^8 + x(5)W_8^{10} + x(6)W_8^{12} + x(7)W_8^{14}$
$X(3) =$	$x(0)W_8^0 + x(1)W_8^3 + x(2)W_8^6 + x(3)W_8^9 + x(4)W_8^{12} + x(5)W_8^{15} + x(6)W_8^{18} + x(7)W_8^{21}$
$X(4) =$	$x(0)W_8^0 + x(1)W_8^4 + x(2)W_8^8 + x(3)W_8^{12} + x(4)W_8^{16} + x(5)W_8^{20} + x(6)W_8^{24} + x(7)W_8^{28}$
$X(5) =$	$x(0)W_8^0 + x(1)W_8^5 + x(2)W_8^{10} + x(3)W_8^{15} + x(4)W_8^{20} + x(5)W_8^{25} + x(6)W_8^{30} + x(7)W_8^{35}$
$X(6) =$	$x(0)W_8^0 + x(1)W_8^6 + x(2)W_8^{12} + x(3)W_8^{18} + x(4)W_8^{24} + x(5)W_8^{30} + x(6)W_8^{36} + x(7)W_8^{42}$
$X(7) =$	$x(0)W_8^0 + x(1)W_8^7 + x(2)W_8^{14} + x(3)W_8^{21} + x(4)W_8^{28} + x(5)W_8^{35} + x(6)W_8^{42} + x(7)W_8^{49}$

N^2 Complex Multiplications
 $\frac{1}{N}$ Scaling Factor Omitted

Σχήμα 3.8 Ο DFT 8 σημείων

Ο FFT είναι απλά ένας αλγόριθμος που μπορεί να επισπεύσει τον υπολογισμό του DFT μειώνοντας τον αριθμό των απαιτούμενων πολλαπλασιασμών και προσθέσεων. Δημοσιεύτηκε από τους J.W. Cooley και J.W. Turkey τη δεκαετία του 60 αλλά ουσιαστικά η ιδέα ήταν ήδη γνωστή από τον Runge (1903) και τους Danielson και Lanczos (1942), όταν όμως δεν ήταν διαδεδομένη η χρήση των υπολογιστών και οι αριθμητικοί υπολογισμοί χωρίς τη βοήθειά τους μπορούσαν να διαρκέσουν ώρες. Επίσης είναι γνωστό ότι ο γερμανός μαθηματικός Karl Friedrich Gauss (1777-1855) είχε χρησιμοποιήσει τη μέθοδο πάνω από έναν αιώνα νωρίτερα.

Για να γίνουν πιο κατανοητές οι βασικές πτυχές του FFT, αρκεί να παρατηρήσουμε ότι το ανάπτυγμα του DFT στο σχήμα 3.8 μπορεί να απλοποιηθεί σε μεγάλο βαθμό αν εκμεταλλευτούμε τη συμμετρία και την περιοδικότητα των twiddle factors, όπως προκύπτει από το σχήμα 3.9. Αν οι εξισώσεις αναδιαταχθούν και παραγοντοποιηθούν αυτό που θα προκύψει είναι ο γρήγορος μετασχηματισμός Fourier (FFT), ο οποίος απαιτεί μόνο $(N/2) \cdot \log_2(N)$ μιγαδικούς πολλαπλασιασμούς. Η ταχύτητα εκτέλεσης των υπολογισμών που προσφέρει ο FFT σε σχέση με τον DFT αποκτά όλο και περισσότερη σημασία όσο αυξάνεται ο αριθμός των δειγμάτων N που δέχεται σαν είσοδο, όπως μπορούμε να διαπιστώσουμε και από το σχήμα 3.10. Παρ' όλα αυτά πρέπει να επισημανθεί ότι ο FFT υπολογίζει τις τιμές όλων των N σημείων του φάσματος. Στην περίπτωση δηλαδή που θέλουμε να υπολογίσουμε τις τιμές μερικών μόνο φασματικών σημείων ίσως ο DFT να είναι πιο αποδοτικός.

APPLYING THE PROPERTIES OF SYMMETRY AND PERIODICITY TO W_N^r FOR $N = 8$

Symmetry: $W_N^{r+N/2} = -W_N^r$, Periodicity: $W_N^{r+N} = W_N^r$

$N = 8$

$$W_8^4 = W_8^{0+4} = -W_8^0 = -1$$

$$W_8^5 = W_8^{1+4} = -W_8^1$$

$$W_8^6 = W_8^{2+4} = -W_8^2$$

$$W_8^7 = W_8^{3+4} = -W_8^3$$

$$W_8^8 = W_8^{0+8} = +W_8^0 = +1$$

$$W_8^9 = W_8^{1+8} = +W_8^1$$

$$W_8^{10} = W_8^{2+8} = +W_8^2$$

$$W_8^{11} = W_8^{3+8} = +W_8^3$$



Σχήμα 3.9 Συμμετρία και περιοδικότητα των twiddle factors για $N=8$

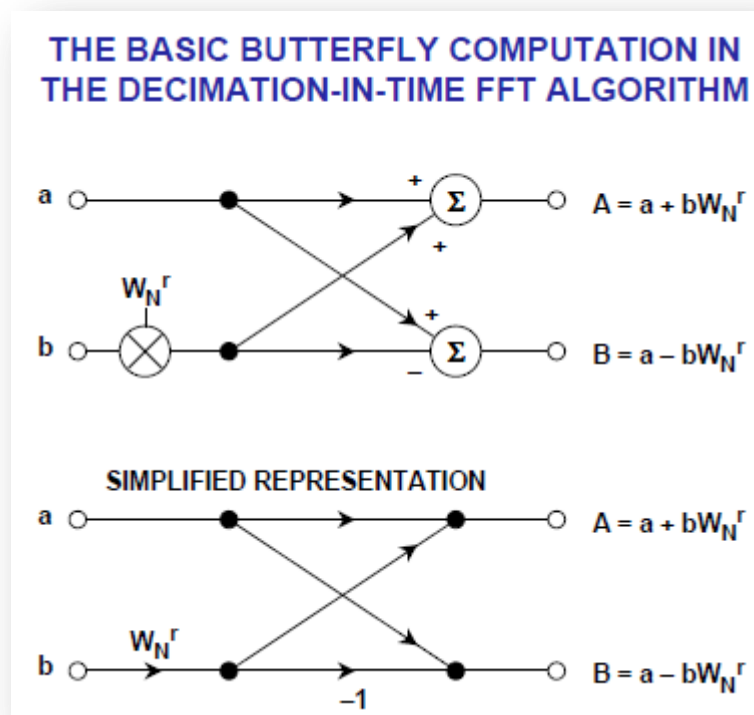
THE FAST FOURIER TRANSFORM (FFT) VS. THE DISCRETE FOURIER TRANSFORM (DFT)

- The FFT is Simply an Algorithm for Efficiently Calculating the DFT
- Computational Efficiency of an N-Point FFT:
 - ◆ DFT: N^2 Complex Multiplications
 - ◆ FFT: $(N/2) \log_2(N)$ Complex Multiplications

N	DFT Multiplications	FFT Multiplications	FFT Efficiency
256	65,536	1,024	64 : 1
512	262,144	2,304	114 : 1
1,024	1,048,576	5,120	205 : 1
2,048	4,194,304	11,264	372 : 1
4,096	16,777,216	24,576	683 : 1

Σχήμα 3.10 Σύγκριση FFT με τον DFT

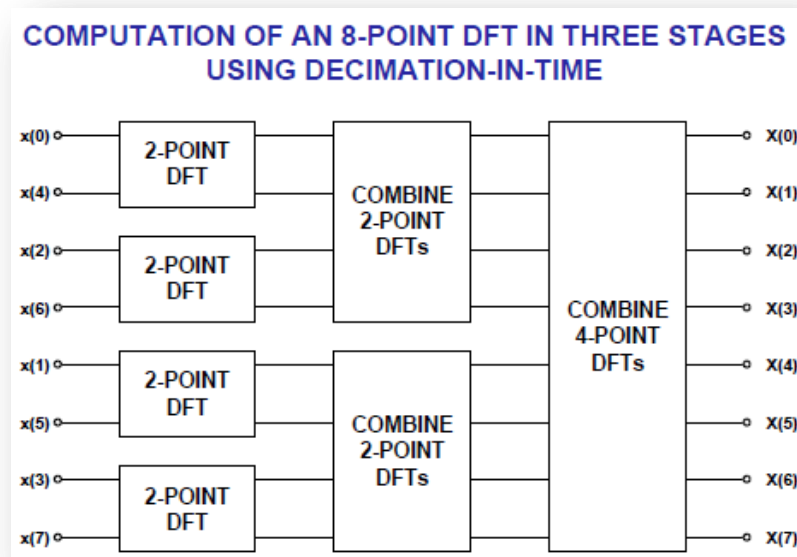
Ο radix-2 FFT αλγόριθμος χωρίζει ολόκληρο τον DFT σε ένα σύνολο από DFT 2 σημείων. Κάθε ένας από αυτούς τους DFT 2 σημείων αποτελείται από ένα συνδυασμό πράξεων μεταξύ των δύο εισόδων μέσω των οποίων προκύπτουν οι 2 έξοδοι. Η διαδικασία αυτή παρουσιάζεται στο σχήμα 3.11 και είναι ευρέως γνωστή ως μηχανισμός της πεταλούδας [6] [7] [9]. Στο σχήμα αυτό φαίνονται δύο διαφορετικές αναπαραστάσεις της πεταλούδας: στο πάνω διάγραμμα μπορούμε να διακρίνουμε τις πράξεις που εκτελούνται καθώς και τους ψηφιακούς πολλαπλασιαστές και αθροιστές. Στο απλοποιημένο διάγραμμα που βρίσκεται ακριβώς από κάτω η ποσότητα με την οποία πολλαπλασιάζεται η είσοδος τοποθετείται πάνω από ένα βέλος ενώ πρόσθεση δύο όρων εκτελείται κάθε φορά που δύο βέλη συγκλίνουν σε μια κουκίδα.



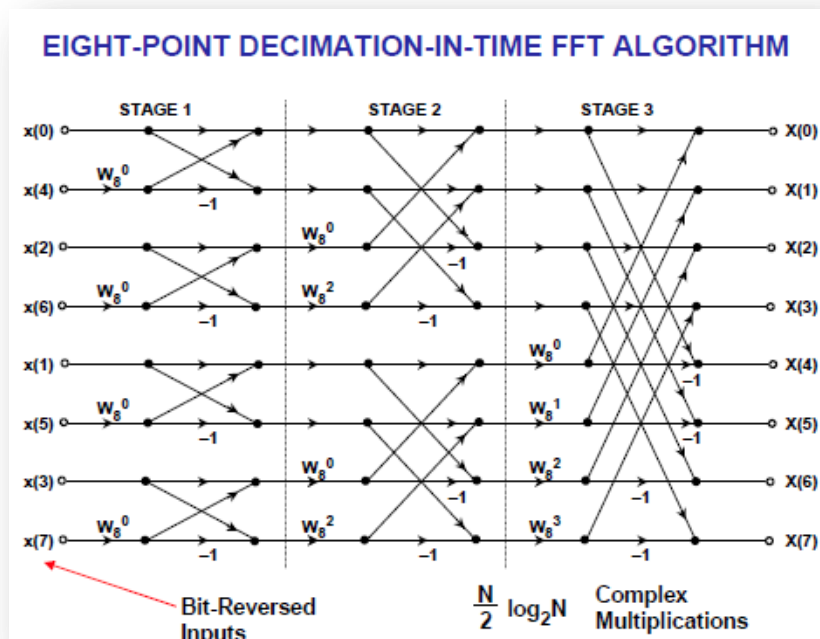
Σχήμα 3.11 Ο βασικός μηχανισμός της πεταλούδας για έναν radix-2 DIT FFT

Ο decimation-in-time (DIT) FFT αλγόριθμος 8 σημείων υπολογίζει τις τελικές εξόδους σε τρία στάδια, όπως μπορούμε να διαπιστώσουμε και από το σχήμα 3.12. Τα 8 χρονικά δείγματα που χρησιμοποιούνται σαν είσοδοι χωρίζονται αρχικά σε 4 δυάδες για κάθε μία από τις οποίες υπολογίζεται ο αντίστοιχος DFT. Οι 8 έξοδοι που προκύπτουν από αυτούς τους 4 μετασχηματισμούς συνδυάζονται εκ νέου μεταξύ τους και για τον κάθε συνδυασμό υπολογίζεται πάλι ο αντίστοιχος DFT. Τέλος οι έξοδοι που προκύπτουν από το δεύτερο στάδιο συνδυάζονται και πάλι και μέσω του DFT που εφαρμόζεται σε κάθε συνδυασμό προκύπτουν οι τελικές έξοδοι $X(k)$. Η συνολική διαδικασία παρουσιάζεται αναλυτικά στο σχήμα 3.13, όπου φαίνονται όλοι οι απαραίτητοι πολλαπλασιασμοί και προσθέσεις. Αξίζει να επισημάνουμε ότι ο βασικός DFT 2 σημείων (πεταλούδα) αποτελεί τη βάση για όλους τους υπολογισμούς. Ο υπολογισμός των εξόδων γίνεται σε τρία στάδια. Οι έξοδοι που προκύπτουν μετά την εκτέλεση των πράξεων που συμπεριλαμβάνονται σε κάθε στάδιο μπορούν να αποθηκευτούν στις θέσεις μνήμης που κρατούσαν την τιμή των εισόδων αφού βάσει της φύσης του αλγορίθμου δεν ξαναχρησιμοποιούνται. Έτσι οι τιμές των εξόδων που προκύπτουν από το πρώτο στάδιο μπορούν να αποθηκευτούν στις θέσεις μνήμης που

κρατούσαν την τιμή των χρονικών δειγμάτων $x(n)$. Αντίστοιχα οι τιμές των εξόδων που προκύπτουν από το πρώτο στάδιο μπορούν να αντικατασταθούν από αυτές που προκύπτουν από το δεύτερο κοκ. μέχρι να υπολογιστούν οι τιμές των τελικών εξόδων. Ένα τελευταίο χαρακτηριστικό του FFT το οποίο πρέπει να προσέξουμε ιδιαίτερα είναι ότι για να εκτελεστεί σωστά ο αλγόριθμος είναι απαραίτητη η κατάλληλη αναδιάταξη των χρονικών δειγμάτων $x(n)$ που χρησιμοποιούνται σαν είσοδοι, η οποία υλοποιείται από έναν άλλο αλγόριθμο ο οποίος είναι γνωστός ως bit-reversal αλγόριθμος.



Σχήμα 3.12 Τα 3 στάδια υπολογισμού ενός 8 σημείων DFT χρησιμοποιώντας έναν radix-2 DIT FFT



Σχήμα 3.13 radix-2 DIT FFT 8 σημείων

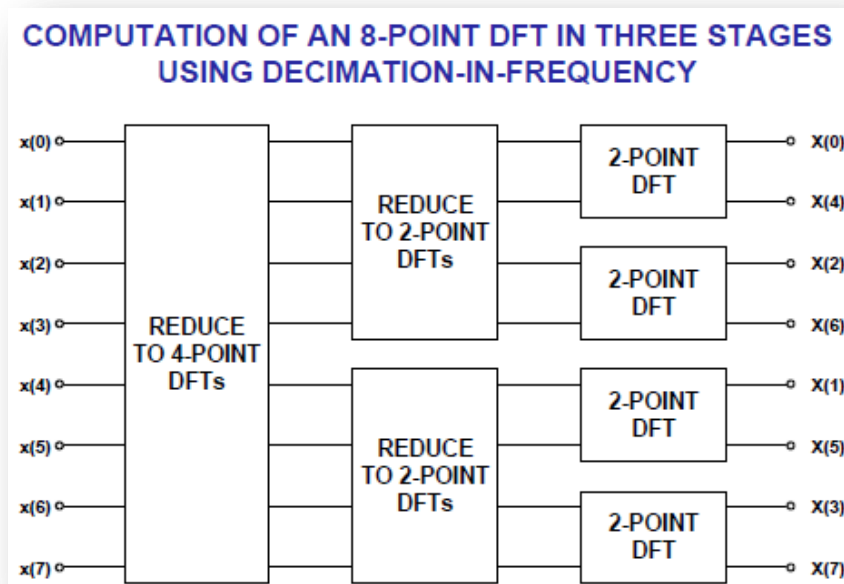
Ο bit-reversal αλγόριθμος [7] [9] [11] χρησιμοποιείται για να αλλάξει τη σειρά των δειγμάτων $x(n)$ με τον τρόπο που φαίνεται στο σχήμα 3.14. Η δεκαδική τιμή του δείκτη n μετατρέπεται στο δυαδικό της ισοδύναμο και στη συνέχεια τα δυαδικά ψηφία τοποθετούνται με την αντίστροφη σειρά (από τα δεξιά προς τα αριστερά). Η νέα δυαδική τιμή που προκύπτει μετατρέπεται στο αντίστοιχο δεκαδικό της ισοδύναμο. Έτσι πριν τον υπολογισμό του FFT ενός διακριτού σήματος που επεξεργάζεται σε πακέτα των 8 δειγμάτων αυτά πρέπει να τοποθετηθούν στη σειρά που φαίνεται στο σχήμα 3.14.

BIT REVERSAL EXAMPLE FOR $N = 8$

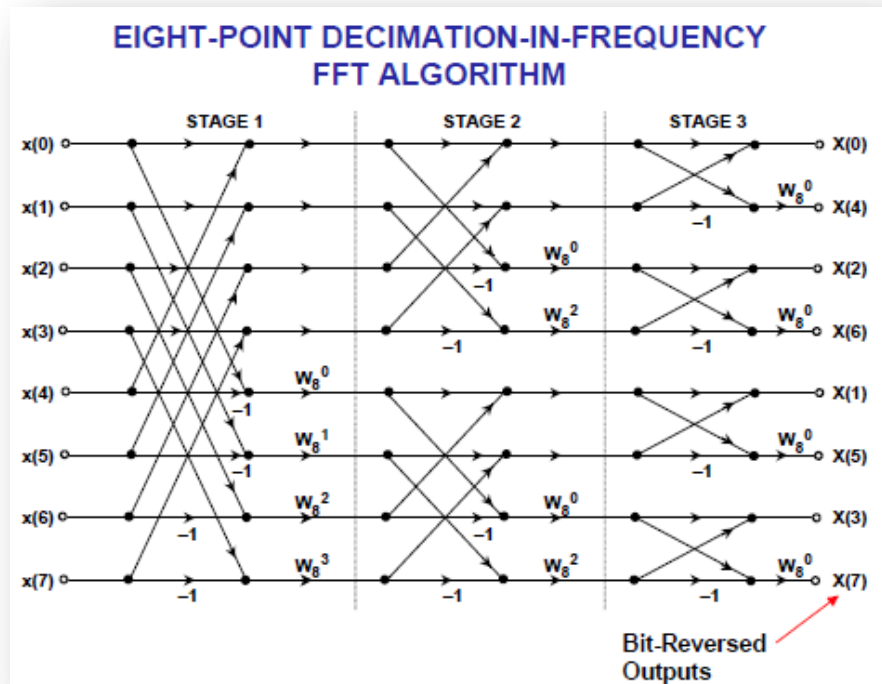
■ Decimal Number :	0	1	2	3	4	5	6	7
■ Binary Equivalent :	000	001	010	011	100	101	110	111
■ Bit-Reversed Binary :	000	100	010	110	001	101	011	111
■ Decimal Equivalent :	0	4	2	6	1	5	3	7

Σχήμα 3.14 bit reversal αλγόριθμος για $N=8$

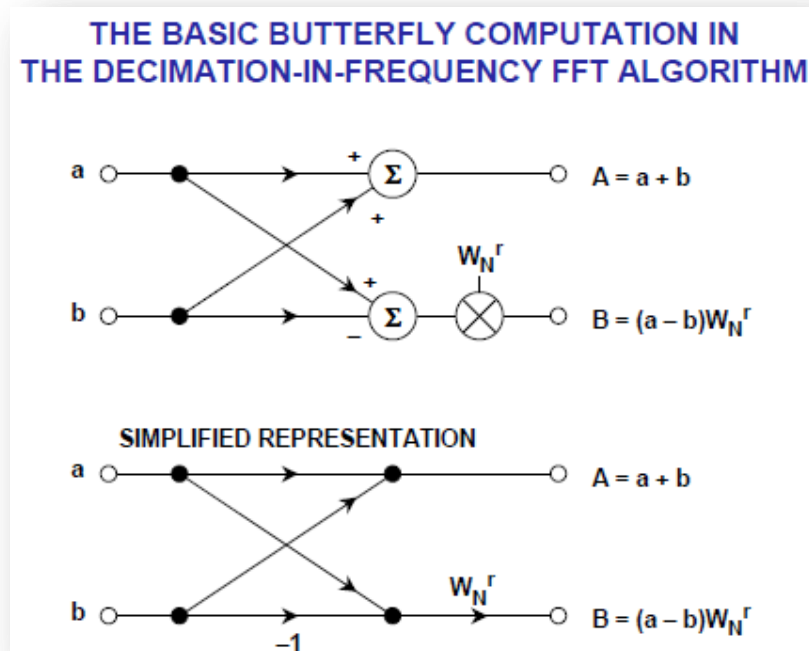
Ο τρόπος υπολογισμού του decimation-in-frequency (DIF) FFT παρουσιάζεται στα σχήματα 3.15 και 3.16 [6] [7] [9]. Αυτή η μέθοδος απαιτεί την εφαρμογή του bit-reversal αλγορίθμου στις τελικές εξόδους $X(k)$. Η πεταλούδα για τον DIF FFT διαφέρει από αυτήν που χρησιμοποιείται στον DIT FFT, όπως μπορούμε να διαπιστώσουμε και από το σχήμα 3.17.



Σχήμα 3.15 Τα 3 στάδια υπολογισμού ενός 8 σημείων DFT χρησιμοποιώντας έναν radix-2 DIF FFT



Σχήμα 3.16 radix-2 DIF FFT 8 σημείων



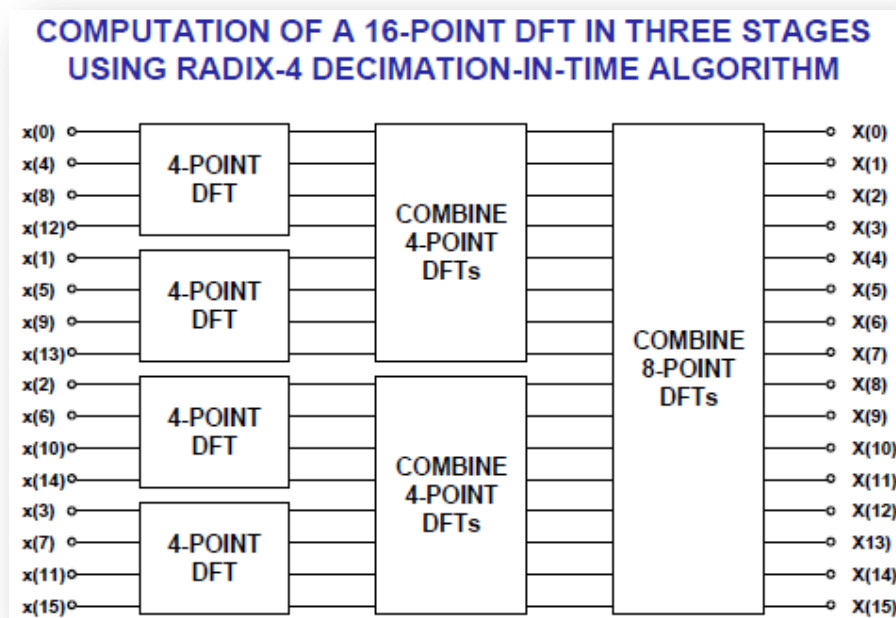
Σχήμα 3.17 Ο βασικός μηχανισμός της πεταλούδας για έναν radix-2 DIF FFT

Τόσο ο DIT FFT όσο και ο DIF FFT οδηγούν στα ίδια αποτελέσματα οπότε είναι καθαρά θέμα προτίμησης ποιον από τους δύο θα χρησιμοποιήσουμε. Ωστόσο οι περιορισμοί του συστήματος μπορεί να καθιστούν πιο κατάλληλη τη χρησιμοποίηση του ενός σε σχέση με τον άλλον.

Πρέπει να επισημανθεί εδώ ότι οι αλγόριθμοι που απαιτούνται για τον υπολογισμό του αντίστροφου FFT είναι σχεδόν ταυτόσημοι με αυτούς που απαιτούνται και για τον υπολογισμό του FFT, θεωρώντας ως δεδομένο ότι χρησιμοποιούνται complex FFTs. Μάλιστα μια μέθοδος για την επαλήθευση ενός complex FFT αλγορίθμου είναι αρχικά ο υπολογισμός του FFT των χρονικών δειγμάτων $x(n)$ (πραγματικοί αριθμοί) και στη συνέχεια αυτός του αντίστροφου FFT των συνιστωσών συχνότητας $X(k)$. Στο τέλος αυτής της διεργασίας πρέπει να προκύψουν οι τιμές των N αρχικών χρονικών δειγμάτων, οι οποίες πρέπει να περιλαμβάνονται στα πραγματικά μέρη των εξόδων $\text{Re}x(n)$, ενώ αντίστοιχα όλα τα φανταστικά μέρη των εξόδων $\text{Im}x(n)$ πρέπει να ισούνται με το μηδέν.

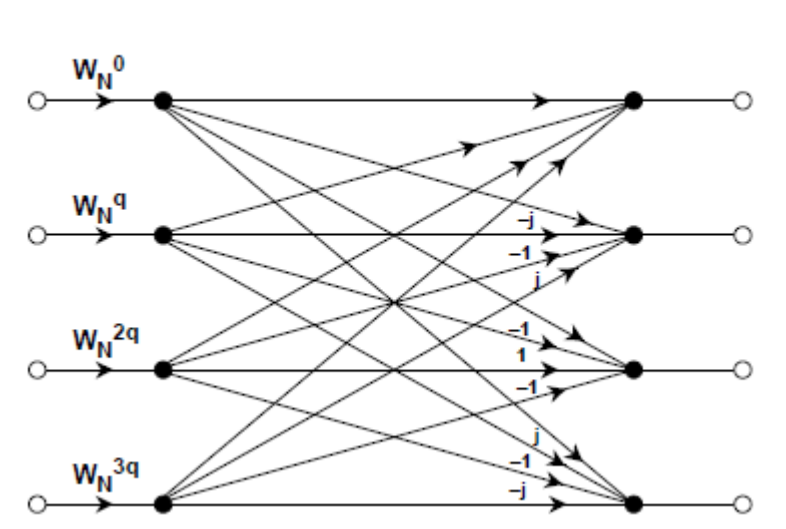
Οι FFT αλγόριθμοι που έχουν παρουσιαστεί μέχρι τώρα είναι radix-2 FFTs, που σημαίνει ότι οι υπολογισμοί έχουν σαν βάση την πεταλούδα 2 σημείων. Από αυτό εύκολα συμπεραίνουμε ότι ο αριθμός των δειγμάτων που υπόκεινται επεξεργασία πρέπει να είναι δύναμη του 2. Εάν ωστόσο ο αριθμός των δειγμάτων είναι και δύναμη του 4 ο FFT μπορεί να διαιρεθεί σε ένα σύνολο από DFTs 4 σημείων (σχ.3.18). Αυτός είναι ένας radix-4 FFT. Η πεταλούδα για έναν decimation-in-time radix-4 FFT φαίνεται στο σχήμα 3.19.

Ο radix-4 FFT απαιτεί λιγότερους μιγαδικούς πολλαπλασιασμούς αλλά περισσότερες προσθέσεις από τον radix-2 FFT για τον ίδιο αριθμό δειγμάτων. Το πόσο πιο αποτελεσματικός και γρήγορος είναι ο radix-4 FFT σε σχέση με τον radix-2 FFT εξαρτάται από το σύστημα ψηφιακής επεξεργασίας σήματος που χρησιμοποιείται. Για συστήματα με καλές σχετικά αρχιτεκτονικές ο αλγόριθμος αυτός μπορεί να είναι και 2 φορές πιο γρήγορος από τον radix-2.



Σχήμα 3.18 Τα 3 στάδια υπολογισμού ενός 16 σημείων DFT χρησιμοποιώντας έναν radix-4 DIT FFT

RADIX-4 FFT DECIMATION-IN-TIME BUTTERFLY



Σχήμα 3.19 Ο βασικός μηχανισμός της πεταλούδας για έναν radix-4 DIT FFT

3.4 Απαιτήσεις Συστήματος για τον Υπολογισμό του FFT

Σε γενικές γραμμές οι απαιτήσεις μνήμης για έναν FFT N σημείων είναι N θέσεις μνήμης για το πραγματικό μέρος των δειγμάτων, N θέσεις μνήμης για τα φανταστικά μέρη και N θέσεις μνήμης για τους twiddle factors [7] [9]. Επιπλέον χώρος στη μνήμη απαιτείται σε περίπτωση που γίνεται παραθυροποίηση των δεδομένων. Αφού εξασφαλίσουμε ότι οι απαιτήσεις μνήμης ικανοποιούνται το επόμενο κριτήριο που πρέπει να ικανοποιείται είναι η εκτέλεση των απαραίτητων υπολογισμών σε συγκεκριμένο χρόνο.

Μια άλλη παράμετρος που πρέπει να ληφθεί υπ' όψη είναι ανάλογα με την εφαρμογή η επιλογή ενός fixed-point ή ενός floating-point επεξεργαστή. Οι τιμές των εξόδων μιας πεταλούδας μπορεί να είναι μεγαλύτερες από τις τιμές των εισόδων της. Αυτή η αύξηση των τιμών μπορεί να αποτελέσει πρόβλημα για ένα σύστημα που μπορεί να επεξεργαστεί μέχρι ένα συγκεκριμένο αριθμό από bits. Για το λόγο αυτό τα δεδομένα πρέπει εξ' αρχής να αναχθούν σε κοινή κλίμακα αφήνοντας έναν ικανό αριθμό από bits ώστε να αποφευχθεί η υπερχείλιση. Εναλλακτικά τα δεδομένα μπορούν να ανάγονται σε κοινή κλίμακα σε κάθε στάδιο υπολογισμού του FFT. Σε ένα 16-bit fixed-point DSP μια λέξη 32-bit προκύπτει κάθε φορά μετά από έναν πολλαπλασιασμό. Είναι λοιπόν εύκολα κατανοητό γιατί η κλιμακοποίηση των δεδομένων καθίσταται αναγκαία. Αντίθετα σε ένα floating-point DSP δεν υπάρχει ανάγκη κλιμακοποίησης των δεδομένων κάτι που κάνει από τη μία τον κώδικα του FFT πιο απλό αλλά από την άλλη αυξάνεται σημαντικά ο χρόνος επεξεργασίας των float πλέον δεδομένων.

Απαιτήσεις Συστήματος σε Εφαρμογές FFT Πραγματικού Χρόνου

Αυτό που επιθυμούμε να επιτύχουμε όταν χρησιμοποιούμε τον FFT σε εφαρμογές πραγματικού χρόνου είναι ο χρόνος εκτέλεσης του αλγορίθμου να είναι μικρότερος ή ίσος του χρόνου που απαιτείται για τη συλλογή του επόμενου πακέτου δειγμάτων προς επεξεργασία. Η πρώτη μεταβλητή η τιμή της οποίας πρέπει να καθοριστεί είναι η συχνότητα δειγματοληψίας. Έτσι θεωρώντας γνωστό το εύρος ζώνης του σήματος εισόδου η απαιτούμενη συχνότητα δειγματοληψίας μπορεί να προκύψει αν πολλαπλασιάσουμε την τιμή του εύρους ζώνης με έναν παράγοντα μεταξύ 2 και 2.5 (μπορεί να απαιτείται και μεγαλύτερη συχνότητα δειγματοληψίας με σκοπό την καλύτερη λειτουργία του low-pass φίλτρου που προηγείται της ψηφιοποίησης). Το επόμενο βήμα είναι ο καθορισμός του αριθμού των δειγμάτων που θα επεξεργάζονται κάθε φορά από τον FFT έτσι ώστε να επιτευχθεί όσο το δυνατόν καλύτερη απεικόνιση του φάσματος συχνοτήτων. Το πόσο απέχουν μεταξύ τους δύο διαδοχικά σημεία του φάσματος συχνοτήτων καθορίζεται από το λόγο f_s/N όπου f_s η συχνότητα δειγματοληψίας και N ο αριθμός των δειγμάτων. Έτσι εύκολα συμπεραίνουμε ότι όσο μεγαλύτερη είναι η τιμή του N τόσο μειώνεται αυτή η απόσταση και συνεπώς αυξάνεται και ο αριθμός των συνιστωσών συχνότητας που απεικονίζονται στο φάσμα κάτι που μας οδηγεί σε πιο έγκυρα και ακριβή αποτελέσματα. Επίσης ο αριθμός των δειγμάτων N καθορίζει και την τιμή του θορύβου που εισάγεται κατά τον υπολογισμό του FFT. Τέλος ανάλογα με τις ανάγκες της εφαρμογής θα πρέπει να επιλέξουμε και το είδος του FFT που θα χρησιμοποιήσουμε. Έτσι μπορεί να κριθεί πιο κατάλληλη η χρησιμοποίηση ενός fixed-point FFT σε σχέση με έναν floating-point και ενός radix-4 σε σχέση με έναν radix-2.

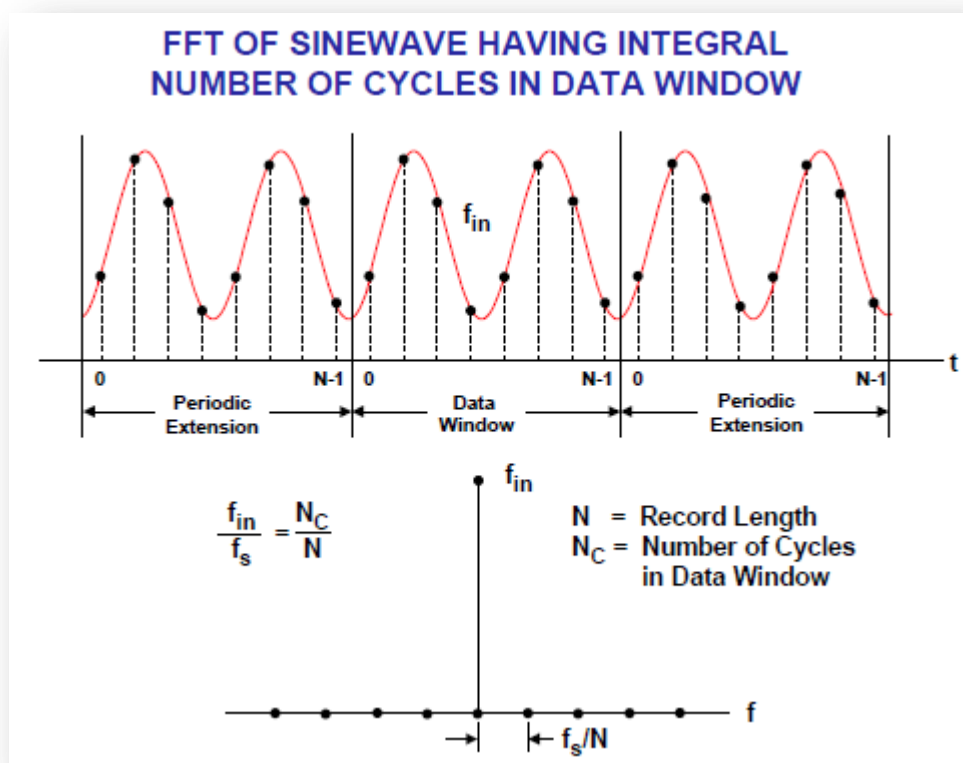
REAL TIME FFT CONSIDERATIONS

- Signal Bandwidth
- Sampling Frequency, f_s
- Number of Points in FFT, N
- Frequency Resolution = f_s / N
- Maximum Time to Calculate N-Point FFT = N / f_s
- Fixed-Point vs. Floating Point DSP
- Radix-2 vs. Radix-4 Execution Time
- FFT Processing Gain = $10 \log_{10}(N / 2)$
- Windowing Requirements

Σχήμα 3.20 Παράμετροι που πρέπει να ληφθούν υπόψη κατά τη χρησιμοποίηση ενός FFT

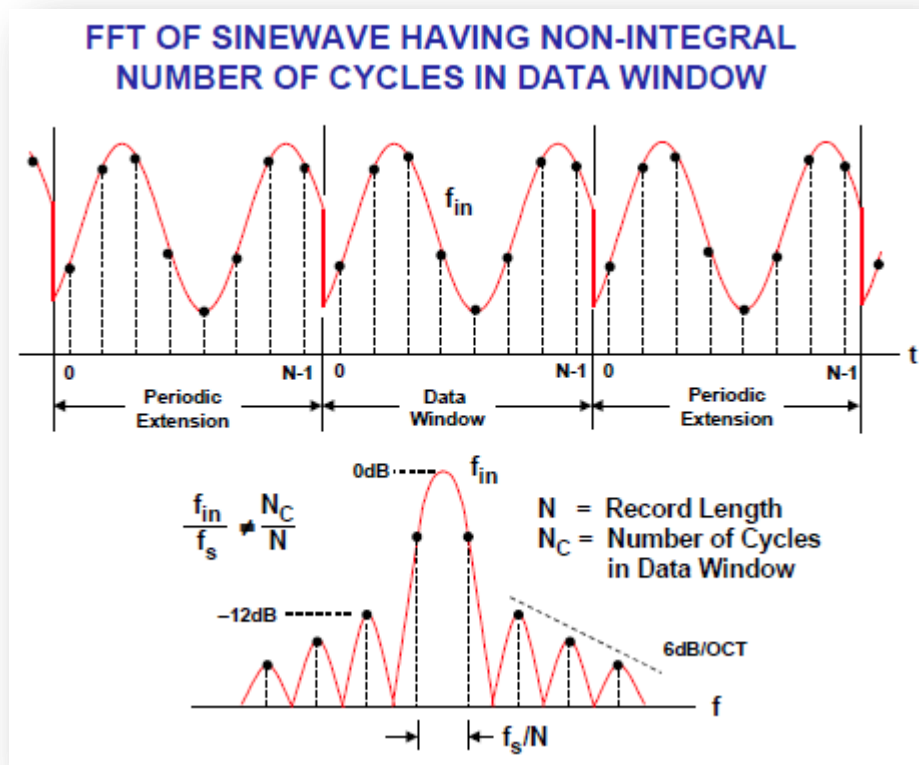
3.5 Παραθυροποίηση και Φασματική Διαρροή

Η φασματική διαρροή (spectral leakage) είναι ένα χαρακτηριστικό του FFT το οποίο μπορεί να ερμηνευτεί καλύτερα αν θεωρήσουμε την περίπτωση υπολογισμού του FFT ενός ημιτονοειδούς σήματος εισόδου N σημείων [5] [9] [10]. Θα εξετάσουμε δύο περιπτώσεις. Στο σχήμα 3.21 ο λόγος f_s/f_{in} είναι ένας ακέραιος αριθμός, πράγμα που σημαίνει ότι τη στιγμή που συλλέγεται και το N -ιοστό δείγμα το σήμα εισόδου συμπληρώνει έναν ακέραιο αριθμό κύκλων. Όπως έχει αναφερθεί και σε προηγούμενη παράγραφο ο DFT υποθέτει ότι ένας άπειρος αριθμός τέτοιων παραθύρων από N δείγματα τοποθετούνται το ένα δίπλα στο άλλο σχηματίζοντας έτσι μια περιοδική κυματομορφή, η οποία απεικονίζεται και στο σχήμα. Υπό αυτές τις συνθήκες η κυματομορφή δεν παρουσιάζει ασυνέχειες και η έξοδος που θα προκύψει ύστερα από τον υπολογισμό του FFT θα είναι μία συνιστώσα στο σημείο που θα έχει τιμή συχνότητας ίση με την τιμή συχνότητας του σήματος εισόδου ενώ οι συνιστώσες συχνότητας σε όλα τα υπόλοιπα σημεία του φάσματος θα είναι μηδέν.



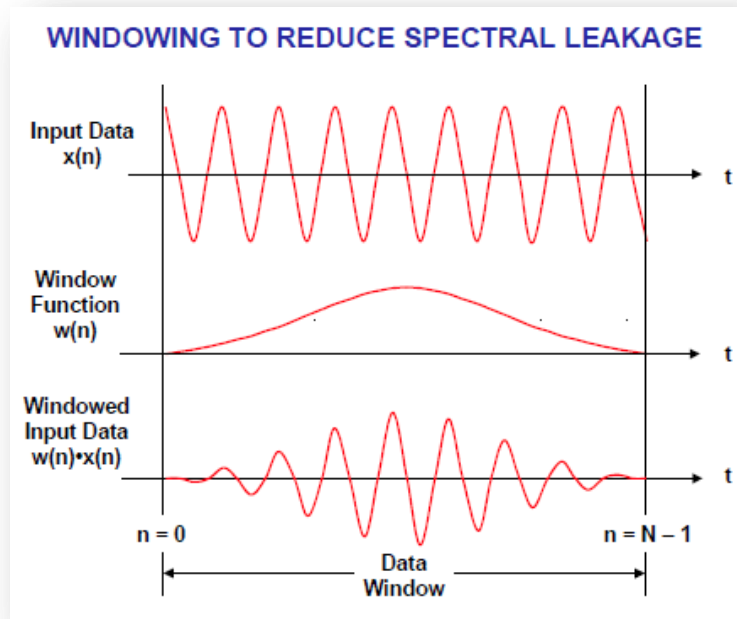
Σχήμα 3.21 FFT ενός ημιτονοειδούς σήματος που έχει συμπληρώσει έναν ακέραιο αριθμό κύκλων

Στο σχήμα 3.22 διακρίνουμε την περίπτωση όπου στο χρόνο που μεσολαβεί ώστε να συλλεχθούν τα N δείγματα το ημίτονο που χρησιμοποιείται σαν είσοδος δεν συμπληρώνει έναν ακέραιο αριθμό κύκλων. Οι ασυνέχειες που προκύπτουν στο τέλος του παραθύρου που “βλέπει” ο FFT έχουν σαν αποτέλεσμα την δημιουργία ενός φαινομένου που λέγεται leakage, το οποίο αποτυπώνεται στο φάσμα συχνοτήτων με την εμφάνιση συνιστωσών συχνότητας σε αρμονικές της συχνότητας του ημιτόνου που αποτελεί το σήμα εισόδου.



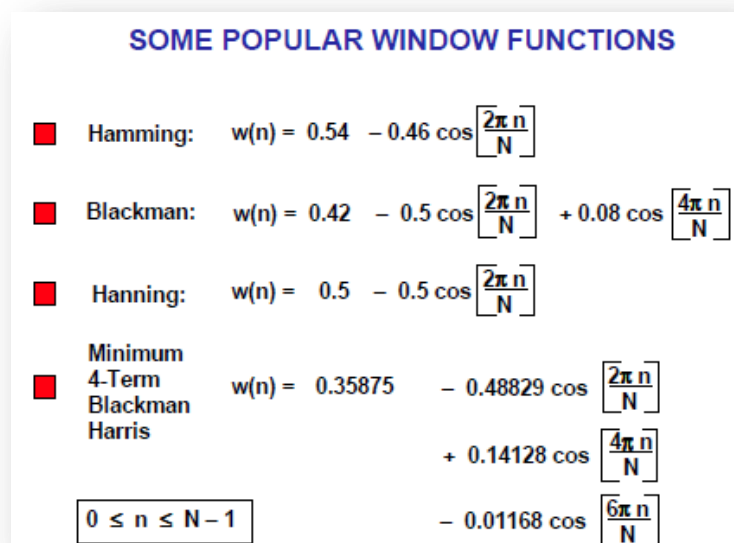
Σχήμα 3.21 FFT ενός ημιτονοειδούς σήματος που δεν έχει συμπληρώσει ακέραιο αριθμό κύκλων

Εκτός από τους πλευρικούς λοβούς ο κύριος λοβός του φάσματος διαχέεται και σε περισσότερες περιοχές μεταξύ των συχνοτήτων του φάσματος. Αυτή η διαδικασία είναι ισοδύναμη με τον πολλαπλασιασμό του ημιτόνου που χρησιμοποιείται σαν είσοδος με ένα τετραγωνικό παλμό, του οποίου η κρουστική απόκριση είναι η γνωστή συνάρτηση $\sin(x)/x$. Από το σχήμα μπορούμε να παρατηρήσουμε ότι ο πρώτος πλευρικός λοβός είναι μόνο 12dB χαμηλότερος από το πλάτος που αντιστοιχεί στην θεμελιώδη συχνότητα του σήματος και ότι το πλάτος κάθε λοβού μειώνεται με κλίση 6dB ανά οκτάβα. Αυτή η κατάσταση δεν ενδείκνυται για τις περισσότερες εφαρμογές ανάλυσης φάσματος συχνοτήτων. Από τη στιγμή που στις περισσότερες εφαρμογές υπολογίζουμε τον FFT για σήματα των οποίων οι συχνότητες που περιλαμβάνονται στο σήμα εισόδου δεν είναι γνωστές προκύπτει η ανάγκη να ελαχιστοποιηθούν όσο γίνεται αυτοί οι πλευρικοί λοβοί. Αυτό γίνεται εφαρμόζοντας ένα παράθυρο διαφορετικό από το τετραγωνικό. Τα χρονικά δείγματα που τροφοδοτούν την είσοδο του FFT πολλαπλασιάζονται με μια κατάλληλη παραθυρική συνάρτηση που μηδενίζει την τιμή του σήματος στις άκρες του παραθύρου όπως φαίνεται και από το σχήμα 3.23. Το ποσοστό εμφάνισης του κύριου λοβού στους υπόλοιπους και η κλίση με την οποία μειώνεται το πλάτος κάθε λοβού είναι και τα κριτήρια που καθορίζουν την επιλογή του κατάλληλου παραθύρου [8] [9].



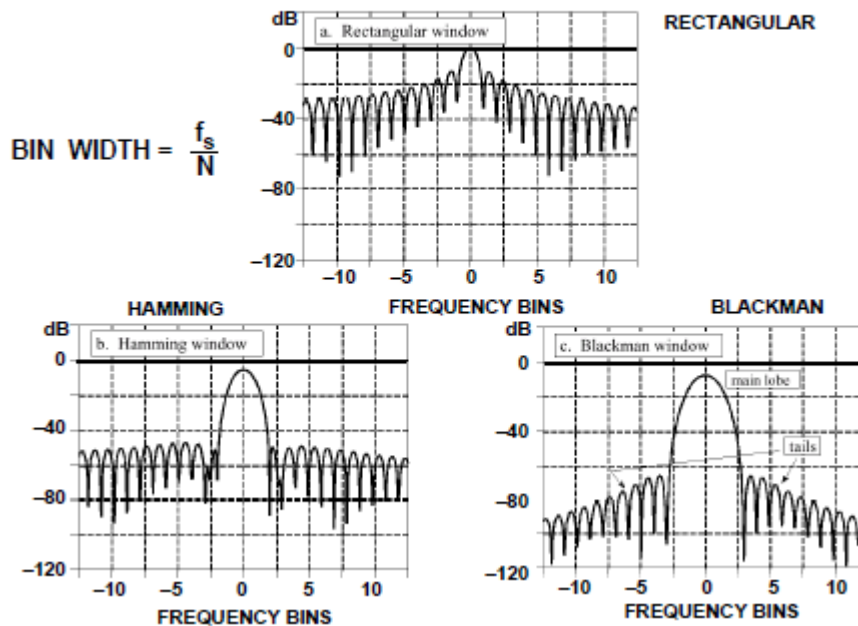
Σχήμα 3.23 Παραθυροποίηση

Στο σχήμα 3.24 παρουσιάζονται οι μαθηματικές συναρτήσεις που περιγράφουν τέσσερα γνωστά παράθυρα (Hamming, Blackman, Hanning, 4-term Blackman-Harris). Οι υπολογισμοί γίνονται απευθείας και οι συντελεστές του παραθύρου υπολογίζονται συνήθως πριν τη ρουτίνα του FFT και αποθηκεύονται στη μνήμη με σκοπό την μείωση όσο γίνεται του χρόνου επεξεργασίας των δεδομένων από τον FFT. Η απόκριση συχνότητας του τετραγωνικού, του Hamming και του Blackman παραθύρου απεικονίζεται στο σχήμα 3.25 ενώ στο σχήμα 3.26 φαίνεται για το κάθε παράθυρο πόσο ο κύριος λοβός εμφανίζεται στους υπόλοιπους, το πλάτος των πλευρικών λοβών και η κλίση με την οποία αυτό μειώνεται [8].



Σχήμα 3.24 Εξισώσεις των κυριότερων παραθυρικών συναρτήσεων

FREQUENCY RESPONSE OF RECTANGULAR, HAMMING, AND BLACKMAN WINDOWS FOR N = 256



Σχήμα 3.25 Απόκριση συχνότητας του τετραγωνικού, του Hamming και του Blackman παραθύρου για N=256

POPULAR WINDOWS AND FIGURES OF MERIT

WINDOW FUNCTION	3dB BW (Bins)	6dB BW (Bins)	HIGHEST SIDELobe (dB)	SIDELobe ROLLOFF (dB/Octave)
Rectangle	0.89	1.21	-12	6
Hamming	1.3	1.81	-43	6
Blackman	1.68	2.35	-58	18
Hanning	1.44	2.00	-32	18
Minimum 4-Term Blackman-Harris	1.90	2.72	-92	6

Σχήμα 3.26 Χαρακτηριστικά μεγέθη των κυριότερων παραθύρων

3.6 Αλγόριθμος GOERTZEL

Ο αλγόριθμος Goertzel όπως και ο FFT χρησιμοποιείται για την ανίχνευση των συχνοτήτων που απαρτίζουν ένα σήμα. Η διαφορά του σε σχέση με τον FFT είναι ότι ενώ ο πρώτος υπολογίζει τις συνιστώσες συχνότητας σε όλο το φάσμα συχνοτήτων με τον Goertzel μπορούμε να υπολογίσουμε τις συνιστώσες συχνότητας σε προκαθορισμένες τιμές συχνοτήτων. Εύκολα λοιπόν συμπεραίνουμε ότι η υλοποίηση του αλγορίθμου Goertzel έχει πολύ μικρότερες απαιτήσεις σε υπολογιστική ισχύ σε σχέση με αυτές του FFT και οδηγεί πολύ πιο γρήγορα σε αποτελέσματα.

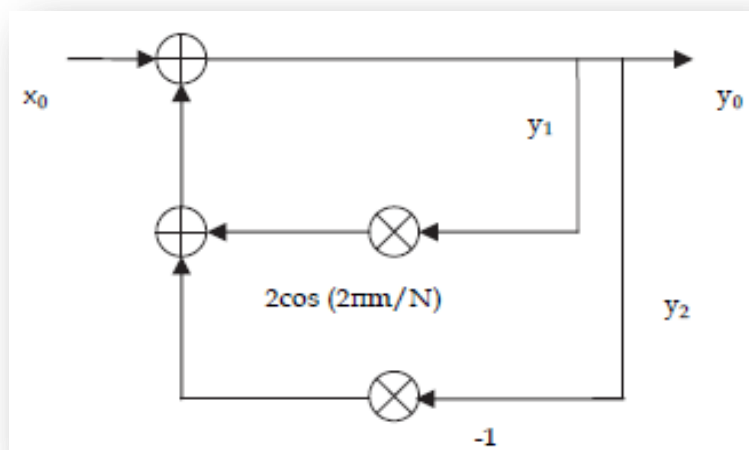
Πολλές εφαρμογές απαιτούν εντοπισμό συχνότητας, όπως:

- 1) DTMF (Dual Tone Multi-Frequency) αποκωδικοποίηση
- 2) Αποκωδικοποίηση κλήσεων
- 3) Μετρήσεις απόκρισης συχνότητας (στέλνοντας μια συχνότητα και διαβάζοντας ταυτόχρονα το αποτέλεσμα) - κάνοντάς το για ένα εύρος συχνοτήτων, η καμπύλη απόκρισης συχνοτήτων μπορεί να περιέχει πληροφορία. Π.χ. η καμπύλη απόκρισης συχνότητας μιας τηλεφωνικής γραμμής μας λέει, το αν κάποιο πηνίο φόρτισης είναι παρόν σε αυτήν την γραμμή.

Παρά το ότι ειδικά Integrated Circuits υπάρχουν για τις παραπάνω εφαρμογές, εκτελώντας αυτές τις λειτουργίες με λογισμικό κοστίζει λιγότερο. Δυστυχώς, πολλά ενσωματωμένα συστήματα δεν έχουν την ισχύ να αποδώσουν συνεχείς FFTs πραγματικού χρόνου. Εδώ είναι που χρησιμοποιείται ο αλγόριθμος Goertzel. Μέσα από την περιγραφή του βασικού Goertzel και του βελτιστοποιημένου Goertzel θα κατανοήσουμε την μεγάλη χρησιμότητα του.

Ο βασικός Goertzel μας δίνει πραγματικές και φανταστικές συνιστώσες συχνοτήτων όπως ο κανονικός DFT ή όπως ο FFT. Αν χρειαζόμαστε τη φάση και το πλάτος μπορούν να υπολογιστούν από το πραγματικό και το φανταστικό ζευγάρι. Ο βελτιστοποιημένος Goertzel είναι ακόμα πιο γρήγορος και απλούστερος από τον βασικό Goertzel, αλλά δεν δίνει τις πραγματικές και φανταστικές συνιστώσες της συχνότητας. Αντίθετα δίνει το σχετικό πλάτος στο τετράγωνο. Υπολογίζοντας την τετραγωνική ρίζα του αποτελέσματος παίρνουμε το σχετικό πλάτος αλλά δεν υπάρχει τρόπος να υπολογίσουμε και τη φάση.

Ο βασικός μετασχηματισμός Goertzel προέκυψε από τον διακριτό μετασχηματισμό Fourier (DFT). Ο αλγόριθμος παρουσιάστηκε από τον Gerald Goertzel (1920-2002) το 1958. Είναι μια ιδιαίτερα αποτελεσματική μέθοδος εντοπισμού μίας προκαθορισμένης συνιστώσας συχνότητας σε ένα πίνακα από δεδομένα που χρησιμοποιείται σαν είσοδος. Στο σχήμα 3.27 απεικονίζεται το διάγραμμα ροής ενός σήματος για τον βασικό αλγόριθμο Goertzel, όπου φαίνεται η επεξεργασία κάθε μεμονωμένου δείγματος [13].



Σχήμα 3.27 Υπολογισμός της εξόδου για κάθε δείγμα που λαμβάνεται σύμφωνα με τον βασικό αλγόριθμο Goertzel

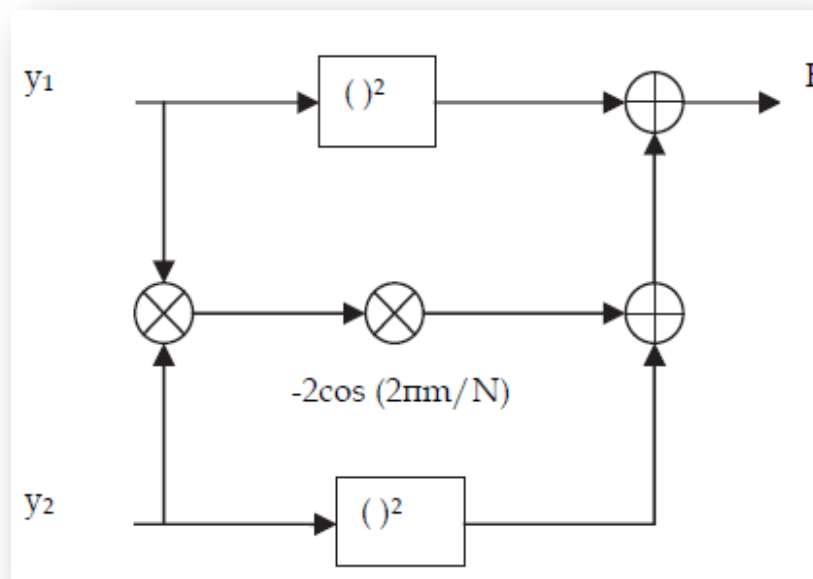
Σύμφωνα με το διάγραμμα αυτό μια έξοδος y_0 παράγεται κάθε φορά που επεξεργάζεται ένα δείγμα. Η έξοδος είναι ένας συνδυασμός που περιλαμβάνει την πρόσθεση του τελευταίου δείγματος (μετά την ψηφιοποίησή του από τον ADC μετατροπέα) με το γινόμενο της προηγούμενης εξόδου επί έναν συντελεστή μείον την τιμή που είχε η έξοδος δύο χρονικές στιγμές πριν. Έτσι η εξίσωση που περιγράφει το σχήμα 3.26 είναι:

$$y_0 = x_0 + y_1 \times 2 \cos\left(\frac{2\pi m}{N}\right) - y_2$$

όπου y_0 είναι η τιμή της εξόδου, x_0 είναι το πιο πρόσφατο δείγμα, y_1 είναι η τιμή της εξόδου μία χρονική στιγμή πριν, y_2 η τιμή της εξόδου δύο χρονικές στιγμές πριν και m είναι ένα σημείο πάνω στον άξονα συχνοτήτων του οποίου τη συνιστώσα επιθυμούμε να ανιχνεύσουμε. N είναι ο συνολικός αριθμός χρονικών δειγμάτων τα οποία υπόκεινται επεξεργασία και προφανώς και ο αριθμός των σημείων πάνω στον άξονα συχνοτήτων καθένα από τα οποία αντιστοιχεί σε μια τιμή συχνότητας. Είναι σημαντικό να επισημάνουμε εδώ ότι αντίθετα με το τι ισχύει στον FFT το N δεν χρειάζεται να είναι δύναμη του 2. Τα δείγματα που ψηφιοποιούνται επεξεργάζονται ένα προς ένα μέχρι να γίνει επεξεργασία και του N -ιστού δείγματος. Η συχνότητα που επιθυμούμε να ανιχνεύσουμε καθορίζεται από την τιμή του συντελεστή της παραπάνω εξίσωσης ο οποίος πολλαπλασιάζεται με την αμέσως προηγούμενη τιμή της εξόδου. Έτσι για κάθε συχνότητα που θέλουμε να ανιχνεύσουμε αντιστοιχεί ένας συγκεκριμένος συντελεστής και συνεπώς και μια διαφορετική εξίσωση. Οπότε ανάλογα με τον αριθμό των συχνοτήτων που αναζητάμε καθορίζεται και ο αριθμός των διαφορετικών εξισώσεων που πρέπει να εκτελούνται κάθε φορά που ένα καινούριο δείγμα προκύπτει στην έξοδο του ADC. Αφού γίνει επεξεργασία όλων των δειγμάτων μπορεί πλέον να υπολογιστεί και να επαληθευτεί η ενέργεια στις συχνότητες αυτές.

Ο αλγόριθμος Goertzel είναι ουσιαστικά η υλοποίηση ενός φίλτρου που υπολογίζει απευθείας την τιμή ενός συντελεστή του DFT. Παίζει το ρόλο ενός 2ης τάξης φίλτρου το οποίο μας δίνει την

ενέργεια που είναι παρούσα σε μια συγκεκριμένη συχνότητα. Ένα τέτοιο σύστημα υπολογισμού της φασματικής ενέργειας για μια συχνότητα φαίνεται στο σχήμα 3.28.



Σχήμα 3.28 Διάγραμμα ροής υπολογισμού της φασματικής ενέργειας για μια συχνότητα

Η εξίσωση που περιγράφει το πιο πάνω διάγραμμα είναι:

$$E_m = y_1^2 + y_2^2 - 2y_1y_2 \cos\left(\frac{2\pi m}{N}\right)$$

Ένα από τα πλεονεκτήματα του αλγόριθμου Goertzel είναι ότι ο υπολογισμός της φασματικής ενέργειας χρειάζεται να γίνει μόνο αφού ολοκληρωθεί η επεξεργασία όλων των δειγμάτων. Έτσι αφού υπολογιστεί η τελική τιμή της εξόδου μετά την επεξεργασία και του τελευταίου δείγματος η ενέργεια που περιέχεται σε μια συγκεκριμένη συχνότητα, όπως αυτή έχει ήδη καθοριστεί από τον συντελεστή, υπολογίζεται από την παραπάνω εξίσωση. Ο Goertzel σε σχέση με τον FFT έχει πολύ μικρότερες ανάγκες για αποθήκευση δεδομένων για το λόγο ότι ενώ ο δεύτερος απαιτεί να έχουν συλλεχθεί όλα τα δείγματα με τον Goertzel μπορούμε να επεξεργαζόμαστε συνεχόμενα τα δείγματα όπως αυτά παράγονται στην έξοδο του ADC.

Υλοποίηση του Αλγορίθμου GOERTZEL

Πριν περάσουμε στην παράθεση του κώδικα Goertzel ας εξετάσουμε την βήμα προς βήμα διαδικασία για την υλοποίησή του. Αρχικά αφού έχουμε αποφανθεί για τον αριθμό N των δειγμάτων που θα επεξεργαστούμε και έχοντας επιλέξει σαν συχνότητα δειγματοληψίας μια τιμή που καλύπτει

τις ανάγκες της εφαρμογής μας και ικανοποιεί το κριτήριο του Nyquist υπολογίζουμε τις παρακάτω σταθερές:

- $k=(\text{int})(0.5+(N*\text{target_freq})/\text{sample_rate}))$
- $w=(2*\pi/N)*k$
- $\text{cosine}=\cos w$
- $\text{sine}=\sin w$
- $\text{coeff}=2*\text{cosine}$

Στους πιο πάνω υπολογισμούς η μεταβλητή target_freq εκφράζει την τιμή της συχνότητας που θέλουμε να ανιχνεύσουμε [12]. Συνήθως επιλέγουμε μία τιμή η οποία να διαιρεί ακριβώς την τιμή της συχνότητας δειγματοληψίας που έχουμε επιλέξει. Η μεταβλητή coeff είναι αυτή που χρησιμοποιείται στην εξίσωση που μας δίνει την τιμή της εξόδου κάθε φορά που ολοκληρώνεται η επεξεργασία ενός δείγματος.

Για την επεξεργασία του κάθε δείγματος θα χρειαστούμε τρεις μεταβλητές, τις οποίες και ονομάζουμε Q_0, Q_1 και Q_2 . Σαν Q_0 συμβολίζουμε την πιο πρόσφατη τιμή της εξόδου, Q_1 την προηγούμενη τιμή της εξόδου και Q_2 την τιμή της εξόδου δύο χρονικές στιγμές πριν. Θεωρούμε ότι πριν αρχίσει η επεξεργασία των δειγμάτων τα Q_1 και Q_2 έχουν τιμή μηδέν. Για κάθε δείγμα υπολογίζουμε κάθε φορά τα Q_0, Q_1 και Q_2 ως εξής:

$$Q_0=(\text{coeff}*Q_1)-Q_2+\text{sample}$$

$$Q_2=Q_1$$

$$Q_1=Q_0$$

Αφού οι παραπάνω εντολές εκτελεστούν N φορές (N δείγματα) υπολογίζουμε το πραγματικό και τα φανταστικό μέρος για τη συχνότητα που μας ενδιαφέρει και από αυτά το πλάτος ως εξής:

$$\text{real}=(Q_1*\text{cosine})$$

$$\text{imag}=(Q_2*\text{sine})$$

$$\text{magnitude}=\sqrt{(\text{real}^2+\text{imag}^2)}$$

Υπάρχει και η δυνατότητα να υπολογίσουμε κατευθείαν το τετράγωνο του πλάτους που εκφράζει την ενέργεια στη συγκεκριμένη συχνότητα χωρίς να απαιτείται ο υπολογισμός του πραγματικού και του φανταστικού μέρους και η οποία δίνεται από την παρακάτω εξίσωση:

$$magnitude^2 = Q_1^2 + Q_2^2 - Q_1 * Q_2 * coeff$$

Πιο κάτω γίνεται η παρουσίαση του αλγορίθμου βασιζόμενη στη συλλογιστική που μόλις περιγράφηκε.

```
float goertzel(float target_freq, int size, const float x[]) {
    int i;

    float coeff;

    float Q0, Q1=0.0, Q2=0.0;

    k=(int)(0.5+((N*target_freq)/(sample_rate)));

    coeff =2.0 *cos(2.0 *π*k/N );

    for (i=0; i<size; i++){

        Q0 = x[i] +(coeff* Q1)- Q2;

        Q2= Q1;

        Q1 =Q0;

    }

    return (Q1* Q1) + (Q2 *Q2) -(Q1 * Q2*coeff);}
```

ΚΕΦΑΛΑΙΟ 4^ο

ΑΝΑΛΥΣΗ ΔΟΝΗΣΕΩΝ

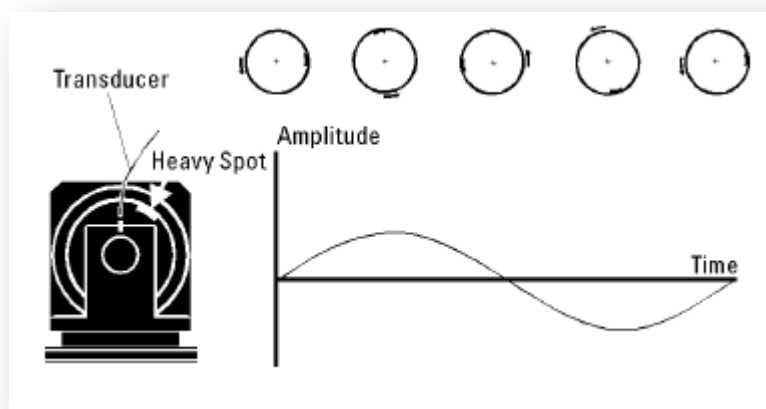
4.1 Η μείωση της δόνησης στα στοιχεία που την απαρτίζουν

Το σήμα που λαμβάνεται από έναν αισθητήρα μηχανικών δονήσεων είναι ένας πολύπλοκος συνδυασμός αποκρίσεων σε πολλαπλές εσωτερικές και εξωτερικές δυνάμεις. Το κλειδί για την αποτελεσματική ανάλυση αυτού του σήματος είναι να μειωθεί στα επιμέρους σήματα που το συγκροτούν, καθένα από τα οποία μπορεί στη συνέχεια να συσχετισθεί με την πηγή του.

Υπάρχουν δύο τρόποι αναπαράστασης ενός σήματος με σκοπό τον προσδιορισμό των συνιστωσών του: (1) Η αναπαράστασή του στο πεδίο του χρόνου όπου εκφράζεται το πλάτος του σήματος συναρτήσει του χρόνου και (2) αυτή στο πεδίο της συχνότητας όπου το πλάτος εκφράζεται συναρτήσει της συχνότητας. Ενώ το πεδίο του χρόνου βοηθάει στο να ερμηνεύσουμε καλύτερα τη φυσική πλευρά του φαινομένου της δόνησης, θα δούμε ότι η ανάλυση στο πεδίο των συχνοτήτων είναι ιδανική για τον προσδιορισμό των δομικών στοιχείων ενός σήματος όπως αυτό που προκαλείται από έναν μηχανικό κραδασμό.

4.1.1 Το πεδίο του χρόνου

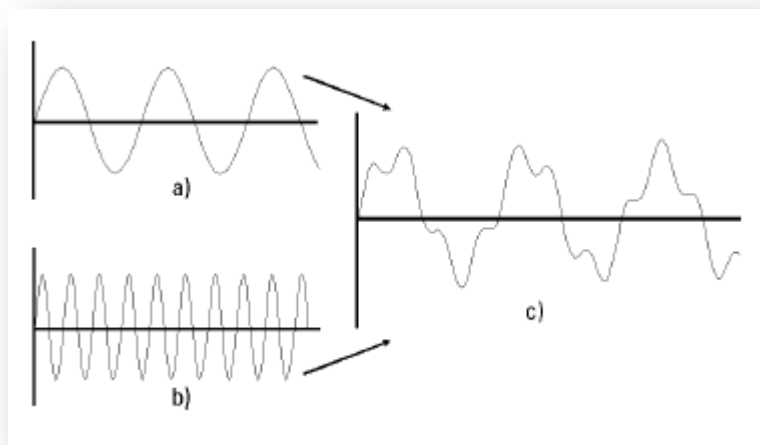
Ένας τρόπος για να εξετάσουμε καλύτερα μια δόνηση είναι να παρατηρήσουμε πώς το πλάτος της μεταβάλλεται με το χρόνο. Στο παρακάτω σχήμα φαίνεται πως μεταβάλλεται με το χρόνο ένα σήμα δόνησης που προκαλείται από την αζυγοσταθμία (το κέντρο μάζας του συστήματος δεν ταυτίζεται με το κέντρο περιστροφής) στο ρότορα μιας ηλεκτρικής μηχανής. Το πλάτος του σήματος είναι ανάλογο με το βαθμό αζυγοσταθμίας του ρότορα και την ταχύτητα περιστροφής του. Αυτό το σήμα είναι εύκολο να αναλυθεί γιατί χρησιμοποιήσαμε ένα εξιδανικευμένο παράδειγμα με μία μόνο πηγή δόνησης όμως στον πραγματικό κόσμο τα σήματα που προκύπτουν από δονήσεις είναι πολύ πιο περίπλοκα [14].



Σχήμα 4.1 Αναπαράσταση ενός σήματος δόνησης που προκύπτει λόγω αζυγοσταθμίας στον άξονα μιας μηχανής στο πεδίο του χρόνου

Όταν υπάρχουν περισσότερες από μία πηγές δόνησης η ανάλυση στο πεδίο του χρόνου γίνεται πιο δύσκολη. Η κατάσταση αυτή απεικονίζεται στο σχήμα 4.2, όπου δύο ημιτονοειδή σήματα

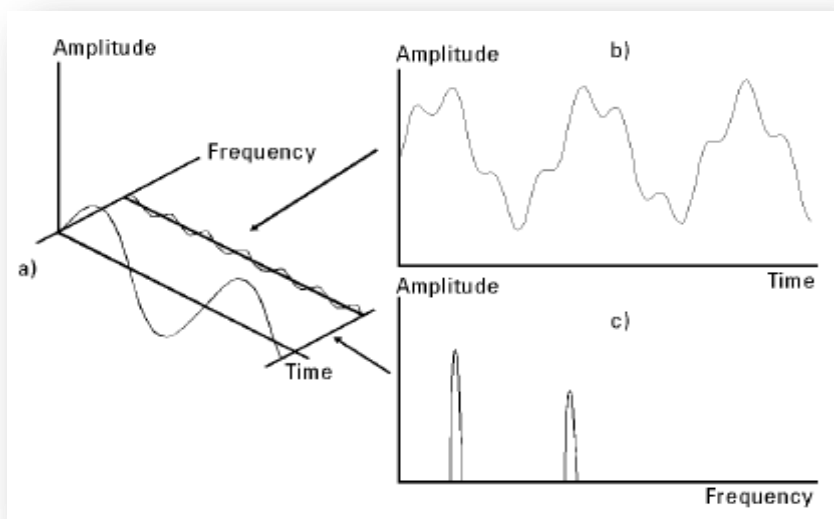
διαφορετικών συχνοτήτων συνθέτουν το τελικό σήμα. Το αποτέλεσμα αυτού του συνδυασμού είναι μια κυματομορφή η οποία έτσι όπως αποτυπώνεται στο πεδίο του χρόνου καθιστά πολύ δύσκολο των διαχωρισμό των ανεξάρτητων συνιστωσών της. Το πεδίο του χρόνου βοηθά στο να προσεγγίσουμε φυσικά το φαινόμενο. Ωστόσο, τα ανεξάρτητα δομικά στοιχεία ενός σύνθετου σήματος είναι δύσκολο να προσδιοριστούν. Μια πολύ καλύτερη επιλογή για τον διαχωρισμό των επιμέρους στοιχείων ενός σήματος είναι η ανάλυση στο πεδίο της συχνότητας.



Σχήμα 4.2 Σύνθετη κυματομορφή που προκύπτει από δύο ημιτονοειδή σήματα διαφορετικών συχνοτήτων

4.1.2 Το πεδίο της συχνότητας

Στο σχήμα 4.3(a) φαίνεται ένα τρισδιάστατο γράφημα του σήματος που χρησιμοποιήθηκε στο τελευταίο παράδειγμα. Οι δύο από τους τρεις άξονες είναι ο χρόνος και το πλάτος που είδαμε και προηγουμένως. Ο τρίτος άξονας είναι η συχνότητα, η οποία μας δίνει μια εικόνα για το πώς διαχωρίζονται τα σήματα που συναποτελούν την κυματομορφή ανάλογα με τη συχνότητά τους. Όταν βλέπουμε τη γραφική παράσταση κατά μήκος του άξονα της συχνότητας βλέπουμε την ίδια εικόνα που απεικονίζεται και στο σχήμα 4.2. Είναι η άθροιση των δύο ημιτονοειδών κυμάτων τα οποία δεν είναι πλέον εύκολα αναγνωρίσιμα (σχήμα 4.3(b)). Ωστόσο, αν δούμε τη γραφική παράσταση κατά μήκος του άξονα του χρόνου (σχήμα 4.3(c)), οι συνιστώσες της συχνότητας της κυματομορφής είναι εύκολα εμφανείς. Σε αυτό το σύστημα αξόνων, του πλάτους δηλαδή συναρτήσει της συχνότητας, κάθε συνιστώσα συχνότητας εμφανίζεται σαν μια κάθετη γραμμή. Το ύψος της αντιπροσωπεύει το πλάτος του αντίστοιχου σήματος και η θέση της τη συχνότητά του. Αυτή η αναπαράσταση του σήματος στο πεδίο των συχνοτήτων ονομάζεται φάσμα του σήματος [14] [15] [16].



Σχήμα 4.3 Αναπαράσταση του ίδιου σήματος στο πεδίο του χρόνου και στο πεδίο της συχνότητας

Η ισχύς του πεδίου συχνότητας έγκειται στο γεγονός ότι κάθε πραγματικό σήμα μπορεί να παραχθεί από την πρόσθεση ημιτονοειδών κυμάτων (θεώρημα Fourier). Έτσι, ενώ για να δείξουμε το πεδίο της συχνότητας χρησιμοποιήθηκε σαν παράδειγμα το άθροισμα δύο ημιτονοειδών κυμάτων, σύμφωνα με το πιο πάνω θεώρημα θα μπορούσαμε να διαχωρίσουμε κάθε σήμα που προκύπτει από ένα σύνολο δονήσεων που παράγονται σε μηχανικό εξοπλισμό στα στοιχειώδη ημιτονοειδή κύματα που το απαρτίζουν. Είναι σημαντικό να γίνει κατανοητό ότι το φάσμα συχνοτήτων ενός σήματος δόνησης ερμηνεύει πλήρως τη δόνηση - καμία πληροφορία δεν χάνεται καθώς περνάμε στο πεδίο των συχνοτήτων.

4.2 Χαρακτηριστικά δονήσεων των κυριότερων σφαλμάτων σε περιστρεφόμενο μηχανικό εξοπλισμό

Είδαμε πιο πάνω πως μπορεί να αναπαρασταθεί ένα σήμα στο πεδίο της συχνότητας και να αναλυθεί στα στοιχειώδη ημιτονοειδή σήματα που το απαρτίζουν. Θα προσπαθήσουμε τώρα να συσχετίσουμε τα στοιχειώδη αυτά σήματα με συγκεκριμένα μηχανικά χαρακτηριστικά και σφάλματα.

Κάθε μηχανικό ελάττωμα παράγει ένα μοναδικό σύνολο από συνιστώσες δόνησης που μπορούν να χρησιμοποιηθούν για την αναγνώρισή του. Προκειμένου να αναλύσουμε πως μέσα από το φάσμα συχνοτήτων ενός σήματος δόνησης μπορούμε να αναγνωρίσουμε τόσο τις πηγές δονήσεων όσο και το είδος των σφαλμάτων που εντοπίζονται σ' αυτές παρουσιάζουμε ενδεικτικά τους κυριότερους τύπους σφαλμάτων που εμφανίζονται στα βασικά εξαρτήματα ενός περιστρεφόμενου μηχανικού εξοπλισμού και πως αυτά επιδρούν στο φάσμα. Πρέπει να επισημανθεί ωστόσο ότι η συλλογιστική αυτή της αντιστοίχησης συνιστωσών δόνησης σε μηχανικά σφάλματα δεν είναι πάντα τόσο εύκολη λόγω κάποιων παραμέτρων, όπως η δυναμική των μηχανών, οι συνθήκες λειτουργίας (φορτίο,

θερμοκρασία), πολλαπλά σφάλματα και μεταβολές στην ταχύτητα λειτουργίας που επηρεάζουν τη δόνηση και δυσκολεύουν την εξαγωγή συμπερασμάτων. Η μετατροπή του φάσματος συχνοτήτων σε μια λεπτομερή αναφορά των δονήσεων που είναι παρούσες σε μηχανικό εξοπλισμό αποτελεί ένα από τα πιο σύνθετα κομμάτια της ανάλυσης δονήσεων.

4.2.1 Αζυγοσταθμία

Αζυγοσταθμία παρατηρείται σε κάποιο βαθμό σε όλες τις μηχανές και χαρακτηρίζεται από μια ημιτονοειδή δόνηση μια φορά ανά περιστροφή. Όταν εμφανίζεται μεγάλο πλάτος στη συχνότητα περιστροφής, ενώ πολλοί λόγοι μπορούν να συμβάλλουν σ' αυτό, σαν κύριο αίτιο προβάλλει η αζυγοσταθμία. Για το λόγο αυτό είναι απαραίτητο να επισημάνουμε εκείνα τα χαρακτηριστικά που τη διαχωρίζουν από τα υπόλοιπα σφάλματα.

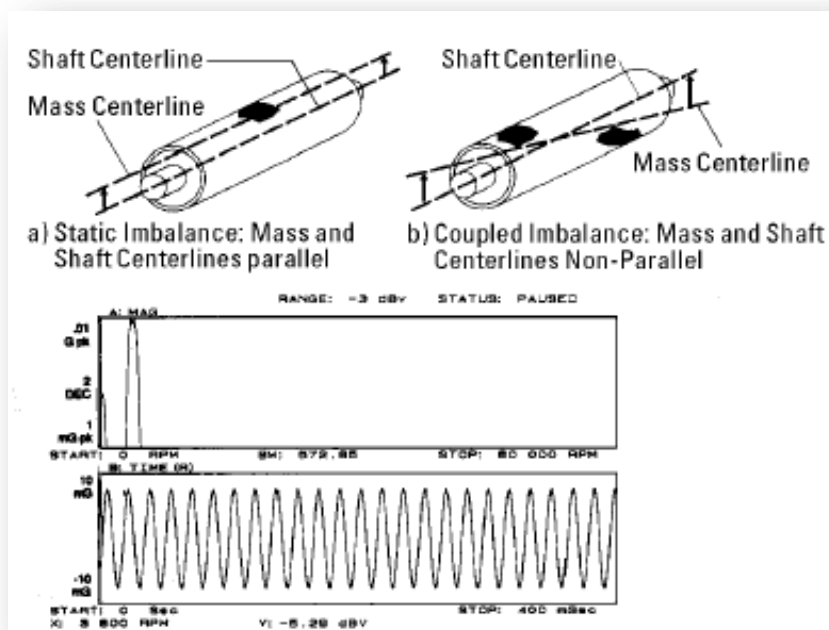
Κατάσταση αζυγοσταθμίας προκύπτει όταν το κέντρο μάζας ενός περιστρεφόμενου μηχανικού εξοπλισμού δεν συμπίπτει με το κέντρο περιστροφής. Διάφοροι μπορεί να είναι οι λόγοι που μπορούν να οδηγήσουν σε αυτό, όπως λανθασμένη συνδεσμολογία, το κέρδος ή η απώλεια υλικού κατά τη διάρκεια της λειτουργίας, η ανομοιομόρφη πυκνότητα υλικού, σταδιακή κάμψη του ρότορα και κάθε άλλο γεγονός που έχει επιπτώσεις στην περιστροφική μαζική διανομή. Όπως φαίνεται και στο σχήμα 4.5 αζυγοσταθμία μπορεί να υπάρχει σε ένα επίπεδο (στατική αζυγοσταθμία) ή σε περισσότερα (συνδυασμένη αζυγοσταθμία). Σε κάθε περίπτωση το αποτέλεσμα είναι ένα διάνυσμα που περιστρέφεται με τον άξονα παράγοντας την κλασσική μια φορά ανά περιστροφή χαρακτηριστική δόνησης [14].

Ιδιαίτερα χαρακτηριστικά αζυγοσταθμίας

Τα χαρακτηριστικά μιας δόνησης η οποία προέρχεται από αζυγοσταθμία είναι τα ακόλουθα:

- (1) Είναι ημιτονοειδής με συχνότητα μία φορά ανά περιστροφή (συχνότητα περιστροφής)
- (2) Είναι ένα περιστρεφόμενο διάνυσμα και
- (3) Το πλάτος της δόνησης αυξάνεται με την ταχύτητα.

Λαμβάνοντας υπόψη τα χαρακτηριστικά αυτά μπορούμε να διαχωρίσουμε την αζυγοσταθμία από άλλα σφάλματα τα οποία επιδρούν στο φάσμα με παρόμοιο τρόπο.



Σχήμα 4.5 Μορφές αζυγοσταθμίας και η επίδρασή τους στο φάσμα συχνοτήτων

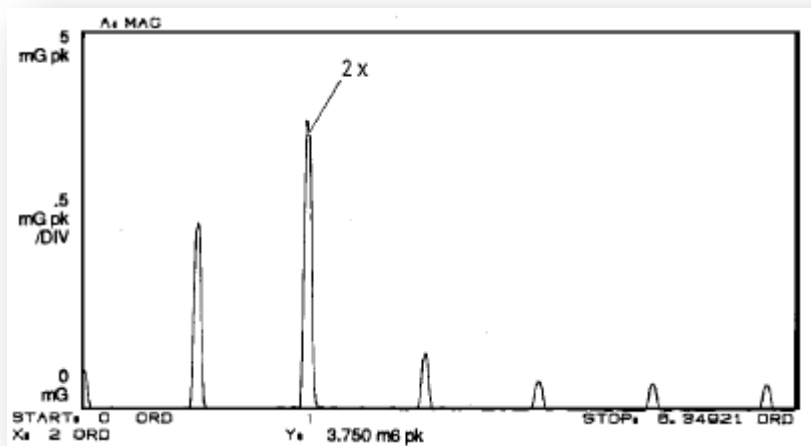
4.2.2 Κακή ευθυγράμμιση

Μία δόνηση που παράγεται εξαιτίας κακής ευθυγράμμισης συνήθως αποτυπώνεται στο φάσμα με ένα μεγάλο πλάτος σε συχνότητα διπλάσια της συχνότητας περιστροφής (2X). Η μελέτη της φάσης είναι επίσης ένα σημαντικό εργαλείο για τη σωστή διάκριση μεταξύ κακής ευθυγράμμισης και αζυγοσταθμίας [14].

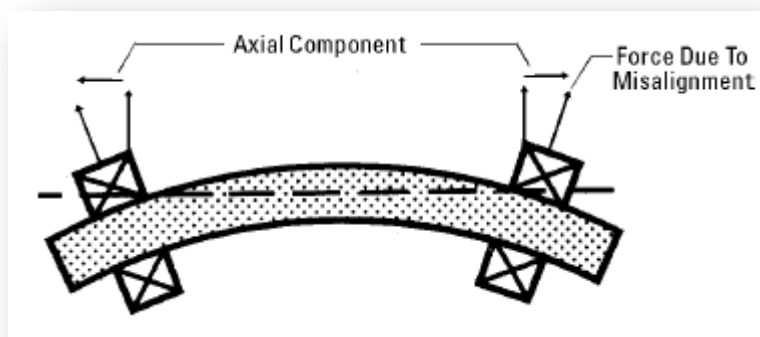
Η κακή ευθυγράμμιση παίρνει τις εξής βασικές μορφές: (1) φόρτιση από έναν λυγισμένο άξονα ή λανθασμένη τοποθέτηση ενός εδράνου κύλισης (ρουλεμάν), (2) αντιστάθμιση των νοητών ευθειών που περνάνε από τα κέντρα δύο ή περισσότερων συζευγμένων αξόνων και (3) γωνιακή ευθυγράμμιση. Η αξονική συνιστώσα της δύναμης που προκαλείται λόγω κακής ευθυγράμμισης φαίνεται στο σχήμα 4.7. Οι μηχανές είναι συνήθως πιο ευέλικτες στην αξονική διεύθυνση με αποτέλεσμα υψηλού επιπέδου αξονικές δονήσεις να συνοδεύουν συνήθως την κακή ευθυγράμμιση. Αυτές οι δονήσεις είναι και το κλειδί στον εντοπισμό του συγκεκριμένου σφάλματος.

Μια δεύτερη αρμονική μεγάλου πλάτους είναι επίσης ένα σύνηθες χαρακτηριστικό όταν δεν υπάρχει σωστή ευθυγράμμιση. Ο λόγος του πλάτους στη συχνότητα περιστροφής (1X) προς το πλάτος στη διπλάσια συχνότητα (2X) μπορεί να χρησιμοποιηθεί ως ένδειξη του μεγέθους της βλάβης. Η δεύτερη αρμονική προκαλείται από την ασυμμετρία τόσο της μηχανής και των στηριγμάτων της όσο και των συνδέσμων της όσον αφορά την ακαμψία που παρουσιάζουν. Αυτή η ασυμμετρία προκαλεί μια ημιτονοειδή μεταβολή στην απόκριση που θα μπορούσε να αναπαρασταθεί σαν ένα περιστρεφόμενο διάνυσμα αντίστασης. Η δόνηση που προκύπτει από την περιστροφική δύναμη και τα διανύσματα αντίστασης περιέχει μία συνιστώσα στη συχνότητα με τιμή διπλάσια της συχνότητας περιστροφής όπως φαίνεται και από το σχήμα 4.6. Η δόνηση που παράγεται λόγω κακής

ευθυγράμμισης περιέχει επίσης έναν μεγάλο αριθμό από αρμονικές με αποτέλεσμα το φάσμα που προκύπτει να μοιάζει πολύ με αυτό που προκύπτει όταν υπάρχει μηχανική χαλαρότητα. Το χαρακτηριστικό εκείνο που τα διαχωρίζει είναι η μεγάλη 2X συνιστώσα ειδικά στην αξονική διεύθυνση.



Σχήμα 4.6 Μία μεγάλης πλάτους συνιστώσα σε συχνότητα διπλάσια της συχνότητας περιστροφής (2x) είναι αποτέλεσμα κακής ευθυγράμμισης



Σχήμα 4.7 Ένας λυγισμένος ή κακός ευθυγραμμισμένος άξονας ευθύνεται για την ύπαρξη μιας έντονης αξονικής δόνησης

4.2.3 Μηχανική χαλαρότητα

Η χαλαρότητα του μηχανισμού, η οποία μπορεί να εμφανιστεί τόσο στην κάθετη όσο και σε οριζόντια κατεύθυνση μπορεί να εμφανιστεί με διάφορους τρόπους στο φάσμα. Μερικές φορές, η βασική 1x της συχνότητας περιστροφής του μηχανισμού εμφανίζεται με διεγερμένο πλάτος. Άλλες φορές εμφανίζεται σε πολλαπλάσια του μισού της συχνότητας περιστροφής του μηχανισμού (0.5x, 1.5x, 2.5x κτλ). Γενικά όμως σχεδόν σε όλες στις περιπτώσεις εμφανίζονται πολλαπλά πολλαπλάσια και του 0.5x και του 1x [16].

Χαλαρότητα στην κάθετη κατεύθυνση

Η χαλαρότητα στην κάθετη κατεύθυνση παράγει δονήσεις που εμφανίζονται και στων δύο ειδών τα πολλαπλάσια. Το παρακάτω σχήμα δίνει ένα παράδειγμα θέσεων αναμονής για αυτή την περίπτωση. Και πρακτικά δίνει πως παράγονται αυτές οι συχνότητες από κάτι που είναι χαλαρό.



Σχήμα 4.7 Μηχανική χαλαρότητα κατά την κάθετη κατεύθυνση

Τις περισσότερες φορές τα πολλαπλάσια του 0.5x παρουσιάζονται με το μισό του πλάτους από τα πολλαπλάσια του 1x. Παράγονται από τη προς τα πάνω μετατόπιση της μηχανής μέχρι αυτή να εμποδιστεί από τα στηρίγματά της. Η επαφή – κρούση με τα στηρίγματα κατά την άνοδο της μηχανής παράγουν τις δονήσεις στα πολλαπλάσια του 0.5x. Καθώς η μηχανή επιστρέφει στην αρχική της θέση συμβαίνει μεγαλύτερη κρούση η οποία αυξάνει τα πλάτη στα πολλαπλάσια της 1x.

Η διαφορά στο πλάτος μεταξύ αυτών των δύο ειδών αρμονικών οφείλεται στην επίδραση της βαρύτητας. Όταν η μηχανή φτάνει στο μέγιστο της ανύψωσής της η βαρύτητα αντιτίθεται στη

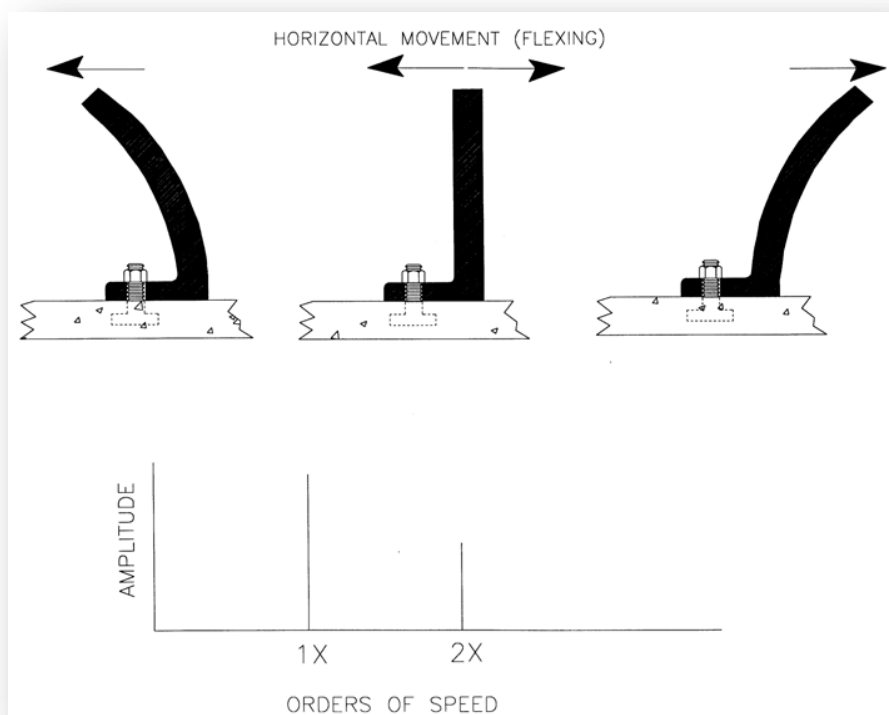
δύναμη ανόδου. Για αυτό η δύναμη που δέχεται το χαλαρό στήριγμα της μηχανής κατά την άνοδο είναι μειωμένη κατά το βάρος της μηχανής. Όταν η μηχανή πέφτει η δύναμη της βαρύτητας συμβάλλει με τη δύναμη που προκαλείται από τις έκκεντρες μάζες της μηχανής. Έτσι πρόκειται για πολύ μεγαλύτερη ενέργεια, άρα και μεγαλύτερο πλάτος.

Χαλαρότητα κατά την οριζόντια κατεύθυνση

Στο παρακάτω σχήμα παρουσιάζεται ο μηχανισμός της οριζόντιας μηχανικής χαλαρότητας. Σε αυτό το παράδειγμα τα στηρίγματα της μηχανής κάμπτονται κατά το οριζόντιο επίπεδο. Αντίθετα με την προηγούμενη περίπτωση η βαρύτητα δεν εμπλέκεται.

Η οριζόντια χαλαρότητα παράγει ένα συνδυασμό της πρώτης και της δεύτερης αρμονικής της ταχύτητας περιστροφής. Εφόσον η πηγή της ενέργειας είναι η περιστρεφόμενη άτρακτος της μηχανής, η περίοδος της ταλάντωσης θα είναι ίση με την περίοδο μίας πλήρους περιστροφής της ατράκτου. Κατά τη διάρκεια αυτής της περιστροφής τα στηρίγματα της μηχανής κάμπτονται στο μέγιστο της διαδρομής και από τις δύο πλευρές από την ουδέτερη θέση. Η διπλή αλλαγή στην κατεύθυνση καθώς το στήριγμα εκτρέπεται πρώτα στη μία πλευρά και μετά στη άλλη, δημιουργεί τη συχνότητα $2x$.

Εντούτοις αυτού του είδους οι συχνότητες δεν παράγονται σε οποιοσδήποτε συνθήκες χαλαρότητας. Για παράδειγμα χαλαρότητα που οφείλεται σε ρουλεμάν ή γρανάζια, δεν οδηγεί σε τέτοιου είδους πολλαπλές συχνότητες, αλλά διεγερμένα πλάτη στις συχνότητες αναμονής των ρουλεμάν και των γραναζιών.



Σχήμα 4.8 Μηχανική χαλαρότητα κατά την οριζόντια κατεύθυνση

4.2.4 Ελαττώματα οδοντωτών τροχών (γρανάζια)

Γενικά είναι εύκολο να αναγνωρίσουμε σφάλματα που προέρχονται από οδοντωτούς τροχούς σε ένα φάσμα συχνοτήτων αλλά πολλές φορές είναι δύσκολο να τα ερμηνεύσουμε πλήρως. Η δυσκολία έγκειται στους εξής δύο παράγοντες: (1) είναι συχνά δύσκολη η τοποθέτηση του αισθητήρα κοντά στην πηγή του σφάλματος και (2) οι διάφορες πηγές δόνησης που είναι παρούσες σε ένα σύνθετο σύστημα από γρανάζια έχουν σαν αποτέλεσμα τη δημιουργία πολλών και διαφορετικών συνιστωσών συχνότητας που δεν είναι πάντα εύκολο να διαχωριστούν. Επίσης είναι σημαντικό τα συγκεκριμένα ελαττώματα να εντοπίζονται έγκαιρα μέσα από συνεχή επίβλεψη του μηχανισμού μιας και σε προχωρημένο στάδιο είναι πολύ δύσκολο να αναλυθούν [14].

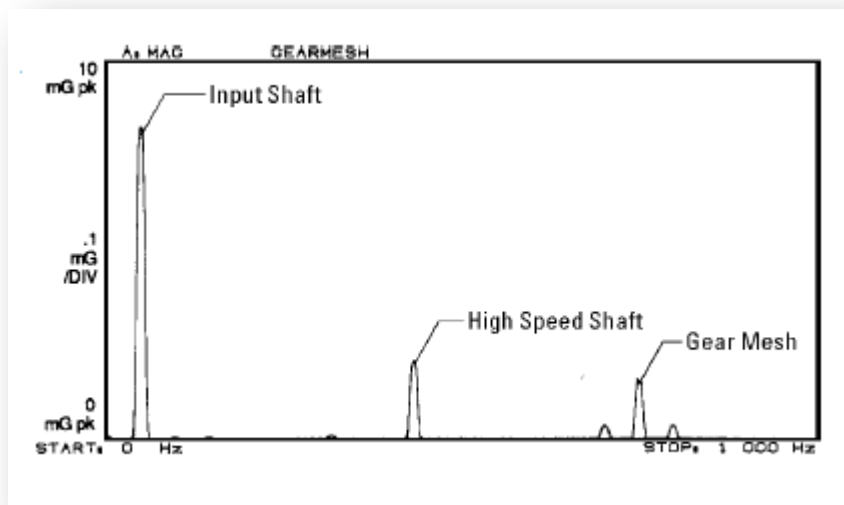
Χαρακτηριστικές συχνότητες

- **Συχνότητα σύμπλεξης (gear mesh frequency):** Είναι ένα μέγεθος που χαρακτηρίζει κάθε γρανάζι και ισούται με την τιμή της συχνότητας περιστροφής πολλαπλασιασμένη επί τον αριθμό των οδόντων του. Στο σχήμα 4.10 απεικονίζεται το φάσμα συχνοτήτων για ένα γρανάζι 15 οδόντων που περιστρέφεται με ταχύτητα 3000 rpm (50 Hz). Η συχνότητα σύμπλεξης ισούται με $15 \cdot 50 = 750$ Hz. Αυτή η συνιστώσα συχνότητας θα εμφανιστεί στο φάσμα ανεξάρτητα από το αν το γρανάζι είναι ελαττωματικό ή όχι. Συχνά εμφανίζονται μικρού πλάτους πλευρικές συνιστώσες γύρω από τη συχνότητα σύμπλεξης οι οποίες οφείλονται κυρίως σε μικρά ποσοστά εκκεντρότητας ή χαλαρότητας.

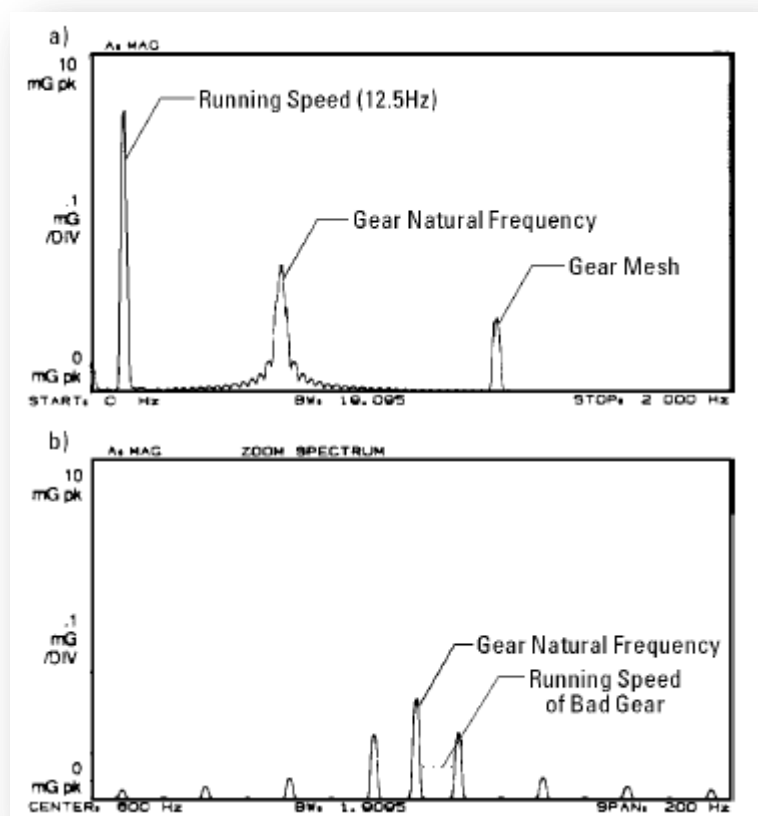
Το πλάτος που εμφανίζεται στη συχνότητα σύμπλεξης μπορεί να υποστεί μεγάλες μεταβολές ανάλογα με τις συνθήκες λειτουργίας, γεγονός το οποίο δεν μας επιτρέπει να θεωρούμε τη συχνότητα σύμπλεξης σαν αξιόπιστη ένδειξη της κατάστασης του εξοπλισμού. Από την άλλη πλευρά μεγάλου πλάτους πλευρικές συνιστώσες ή μεγάλες ποσότητες ενέργειας γύρω από τη συχνότητα σύμπλεξης ή τη φυσική συχνότητα που χαρακτηρίζει το γρανάζι είναι μια καλή ένδειξη ύπαρξης σφάλματος.

- **Φυσικές συχνότητες :** Οι κρουστικές που προκαλούνται από ελαττώματα σε γρανάζια που συναποτελούν ένα σύστημα συχνά διεγείρουν τις φυσικές συχνότητες ενός ή περισσότερων από αυτά. Συνήθως αυτό είναι και το κλειδί για τον εντοπισμό ενός σφάλματος από τη στιγμή που το πλάτος στη συχνότητα σύμπλεξης δεν αλλάζει πάντα. Στο φάσμα συχνοτήτων που απεικονίζεται στο σχήμα 4.10 η συχνότητα σύμπλεξης είναι 1272 Hz. Η φυσική συχνότητα που χαρακτηρίζει το γρανάζι εμφανίζεται στα 600 Hz. Αυτή πλαισιώνεται από πλευρικές συνιστώσες στην συχνότητα περιστροφής του ελαττωματικού γραναζιού. Η λεπτομέρεια αυτή φαίνεται καθαρά στο σχήμα 4.10(b) όπου έχει γίνει μεγέθυνση στην περιοχή του φάσματος όπου εμφανίζεται η φυσική συχνότητα.

- **Πλευρικές συχνότητες :** Σε ένα σύστημα που απαρτίζεται από έναν αριθμό γραναζιών μπορεί να παρατηρηθεί διαμόρφωση στα συνήθη σήματα δόνησης που προκύπτουν λόγω χαλαρότητας, εκκεντρότητας, μεγάλου φορτίου και δονήσεων που προκαλούνται από τα ελαττώματα. Οι πλευρικές συχνότητες που εμφανίζονται με αυτόν τον τρόπο είναι συχνά πολύτιμες για να προσδιορίσουμε ποιο από τα γρανάζια είναι ελαττωματικό. Στο σχήμα 4.10(b) για παράδειγμα, οι πλευρικές συνιστώσες γύρω από τη φυσική συχνότητα υποδηλώνουν ότι το ελαττωματικό γρανάζι περιστρέφεται με συχνότητα 12.5 Hz. Σε περίπτωση εκκεντρότητας η συχνότητα σύμπλεξης συνήθως πλαισιώνεται από πλευρικές συνιστώσες στη συχνότητα περιστροφής.



Σχήμα 4.9 Χαρακτηριστικό φάσμα ενός μη ελαττωματικού συστήματος γραναζιών



Σχήμα 4.10 Οι φυσικές συχνότητες που χαρακτηρίζουν ένα σύστημα γραναζιών διεγείρονται από τις κρουστικές που οφείλονται σε ελαττώματα σε αυτά

4.2.5 Ελαττώματα εδράνων κύλισης (ρουλεμάν)

Είναι αλήθεια πως τα περισσότερα προγράμματα ανάλυσης κραδασμών δημιουργούνται ώστε να μπορούν να ελέγχονται τα ρουλεμάν, τα οποία σπανίως βγαίνουν προβληματικά από τα εργοστάσια παραγωγής τους, αφού ο ποιοτικός έλεγχος σε αυτές τις εγκαταστάσεις είναι ο καλύτερος δυνατός. Ωστόσο, πολλές φορές σε μία κατασκευή μπορούμε να συναντήσουμε ρουλεμάν ελαττωματικά, για τους εξής λόγους:

- Λόγω έλλειψη λίπανσης
- Λόγω της χρήσης σε υψηλότερο από το προβλεπόμενο φορτίο
- Λόγω της κακής εργασίας τοποθέτησής τους στο σύστημα
- Λόγω της μόλυνσης, συμπεριλαμβανομένης και της υγρασίας
- Λόγω των τυχόν ατελειών που παρουσιάστηκαν μετά από την κατασκευή
- Λόγω της κακής συναρμογής τους με τους άξονες

Εάν η βλάβη δεν οφείλεται σε κάποια από τις προαναφερθείσες αιτίες, τότε πρέπει να εξεταστεί το σχέδιο. Όλοι οι κραδασμοί εμφανίζονται σε κάποια συγκεκριμένη συχνότητα, η γνώση της οποίας είναι χρήσιμη για την αντιμετώπιση του προβλήματος, κυρίως για τις βλάβες των ρουλεμάν [16]. Οι συχνότητες στις οποίες εμφανίζονται οι βλάβες σε ένα ρουλεμάν είναι :

$$BPFO = f_{outer} = \frac{n}{2} f_r \left(1 - \frac{BD}{PD} \cos \phi \right) (\alpha)$$

Συχνότητα διέλευσης
σφαιρών
στον εξωτερικό δακτύλιο
(BallPass Frequency Outer race)

$$BPFI = f_{inner} = \frac{n}{2} f_r \left(1 + \frac{BD}{PD} \cos \phi \right) (\beta)$$

Συχνότητα διέλευσης
σφαιρών
στον εσωτερικό δακτύλιο
(BallPass Frequency Inner)

$$BSF = f_{ball} = f_r \frac{PD}{BD} \left[1 - \left(1 + \frac{BD}{PD} \cos \phi \right)^2 \right] (\gamma)$$

Συχνότητα περιστροφής
σφαιρών
(Ball Spin Frequency)

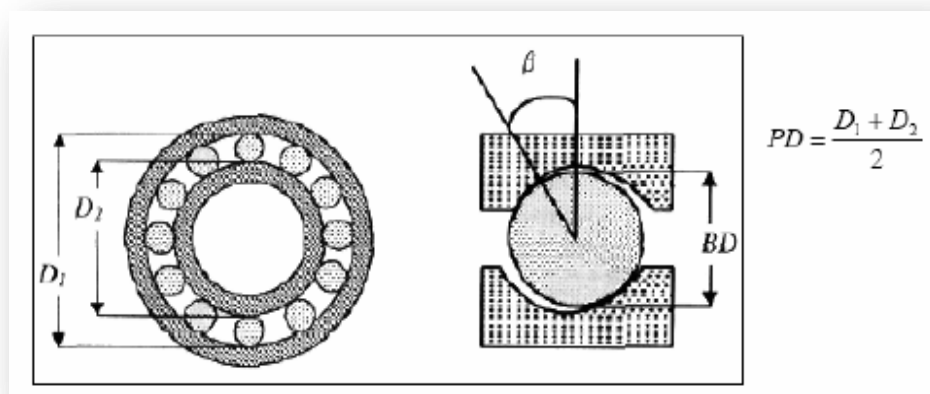
$$FTF = f_{cage} = \frac{1}{2} f_r \left(1 - \frac{BD}{PD} \cos \phi \right) (\delta)$$

Συχνότητα περιστροφής
κλωβοῦ
(Fundamental Train Frequency - Cage)

όπου :

n : αριθμός σφαιρών του ρουλεμάν

$f_r = f_s$: συχνότητα περιστροφής άξονα του ρουλεμάν

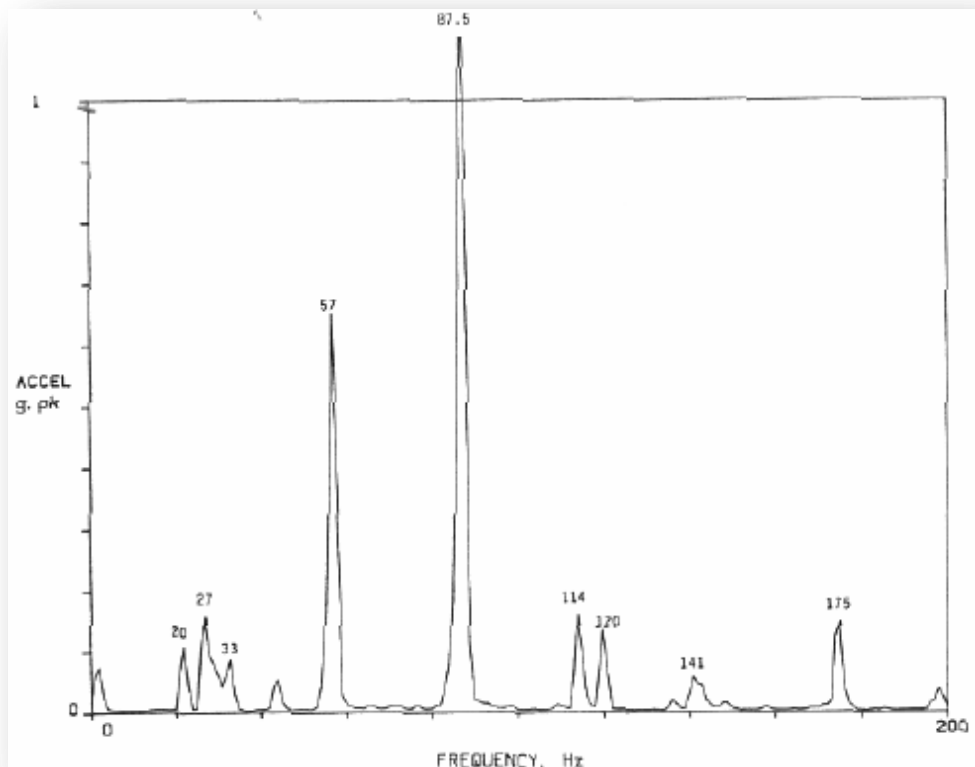


Σχήμα 4.11 Χαρακτηριστικά μεγέθη ρουλεμάν

Στο σχήμα 4.13 εικονίζεται ένα φάσμα αποδιαμορφωμένου σήματος χρονικού παραθύρου 200Hz ενός ρουλεμάν, το οποίο περιστρέφεται με 1715 RPM ($f_r = 28,58\text{Hz}$). Το επιταχυνσιόμετρο είναι τοποθετημένο σε κατακόρυφη διεύθυνση. Οι φέρουσες συχνότητες υπολογίζονται με χρήση των παραπάνω τύπων όπως αυτές ακολουθούν στο σχήμα 4.12.

<i>BPFO (Hz)</i>	<i>BPFI (Hz)</i>	<i>BSF (Hz)</i>	<i>FTF (Hz)</i>
87,37	141,27	57,25	10,92

Σχήμα 4.12 Αναμενόμενες συχνότητες που προκύπτουν με εφαρμογή των παραπάνω τύπων



Σχήμα 4.13 Φάσμα αποδιαμορφωμένου σήματος εύρους 200 Hz

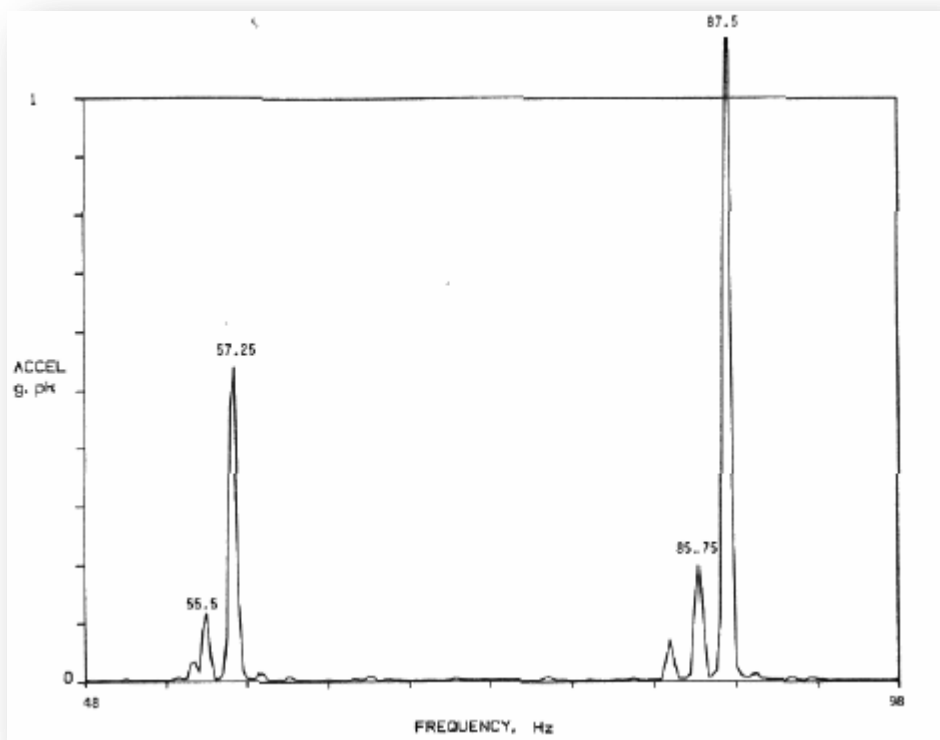
Παρατηρούμε κυρίως τρεις πιο υψηλές αιχμές στις συχνότητες των 87,5Hz, 57Hz και 27Hz. Η συχνότητα f_{cage} (κλωβού) στα 10,92 Hz δεν εμφανίζεται πουθενά. Από αυτό, μπορούμε να βγάλουμε το συμπέρασμα ότι δεν υπάρχει κάποιο πρόβλημα με τον κλωβό του ρουλεμάν. Οι συχνότητες στα 57 και 87,5 Hz, είναι πιθανώς οι συχνότητες BPFO και BSF, αλλά μπορούν να θεωρηθούν και αρμονικές της ταχύτητας:

$$fr = 28,58 \text{ Hz}$$

$$2* fr = 57,16 \text{ Hz}$$

$$3* fr = 85,74 \text{ Hz}$$

Για να δούμε τι ακριβώς συμβαίνει, κάνουμε μία μεγέθυνση στην εικόνα μας παίρνοντας τώρα ένα φάσμα σε χρονικό «παράθυρο» 48 – 98 Hz (σχήμα 4.14).



Σχήμα 4.14 Φάσμα αποδιαμορφωμένου σήματος εύρους 50Hz

Στο παραπάνω σχήμα βλέπουμε ότι η αιχμή στα 57 Hz του σχήματος 4.14 εμφανίζεται σε δύο διαφορετικές συχνότητες στα 55,5 και 57,25 Hz. Η συχνότητα των 57,25 Hz φαίνεται τώρα ότι είναι η BSF με αποτέλεσμα να πρέπει να διερευνήσουμε ακόμα καλύτερα την κατάσταση των σφαιρών των ρουλεμάν και με κάποιες άλλες μετρήσεις. Φυσικά, δεν αποκλείουμε και την περίπτωση η συχνότητα αυτή να είναι η δεύτερη ($2x$) αρμονική της συχνότητας περιστροφής. Για τη συχνότητα των 87,5 Hz, παρατηρούμε ότι έχει παραμείνει η αιχμή ακριβώς στην ίδια συχνότητα, αλλά, έχουμε και μία εμφανή αιχμή στα 85,75 Hz. Η τελευταία, είναι η τρίτη ($3x$) αρμονική της συχνότητας περιστροφής ενώ, αυτή στα 87,5 Hz είναι η BPFO. Τα παραπάνω, οδηγούν σε ανησυχία για την ευθυγράμμιση του συστήματος και την κατάσταση του εξωτερικού δακτυλίου του ρουλεμάν.

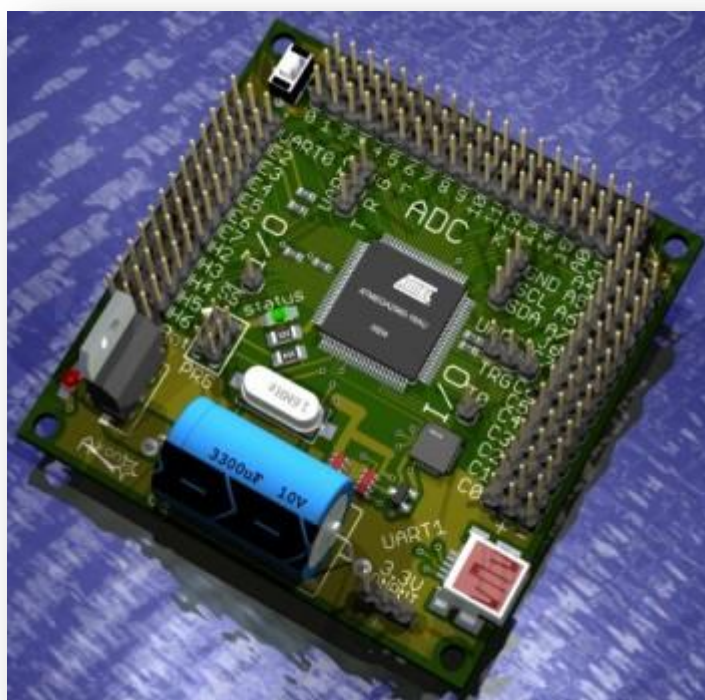
ΚΕΦΑΛΑΙΟ 5^ο

ΜΙΚΡΟΕΛΕΓΚΤΕΣ

5.1 Βασικές έννοιες μικροελεγκτών

5.1.1 Χρήση μικροελεγκτών

Οι μικροελεγκτές αποτελούν μέρος ενός εντυπωσιακού αριθμού προϊόντων τα οποία βρίσκονται γύρω μας. Το αυτοκίνητό μας, τα τηλεχειριστήριά μας, η τηλεόρασή μας, οι ψηφιακές κάμερες, τα κινητά τηλέφωνα, τα πλυντήριά μας είναι μερικά από αυτά. Στην ουσία δεν θα ήταν υπερβολή να πούμε ότι η χρήση μικροελεγκτών στις μέρες μας είναι καθολική και γενικά κάθε προϊόν το οποίο αλληλεπιδρά με ένα χρήστη περιλαμβάνει ένα μικροελεγκτή, ο οποίος παίζει το ρόλο του «εγκεφάλου» των ηλεκτρονικών κυκλωμάτων. Δεν είναι τυχαίο πλέον, ότι πολλές βιομηχανίες προσανατολίζονται σε εφαρμογές όπου αποτελούνται από ηλεκτρικά κυκλώματα τα οποία ελέγχονται από μικροελεγκτή σε αντίθεση με την πρακτική του παρελθόντος όπου χρησιμοποιούσαν ογκώδεις ηλεκτρονικούς υπολογιστές (PC) ή ηλεκτρονόμους και σύνθετη αλλά μόνιμη λογική. Η χρήση μικροελεγκτών ξεκίνησε πριν από περίπου τριάντα πέντε χρόνια. Το πρώτο μοντέρνο ενσωματωμένο σύστημα (embedded system) ήταν ο υπολογιστής του διαστημοπλοίου Apollo, ο οποίος αναπτύχθηκε από τον Charles Stark Draper στο Εργαστήριο Instrumentation Laboratory του MIT. Λόγω της ραγδαίας ανάπτυξης της ηλεκτρονικής έγινε δυνατή η βιομηχανική παραγωγή τους με μικρό κόστος.



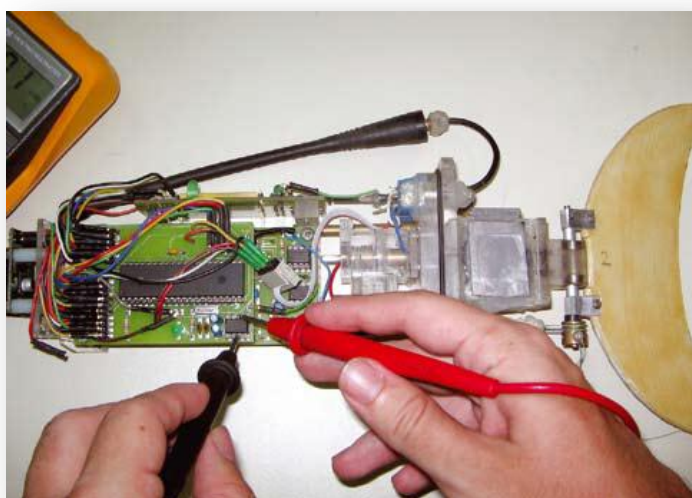
Σχήμα 5.1 Axon microcontroller

5.1.2 Χαρακτηριστικά

Ένας μικροελεγκτής είναι στην ουσία ένας υπολογιστής σε μια πιο συμπαγή μορφή. Έχει δηλαδή μία μονάδα μνήμης (CPU) στην οποία εκτελούνται τα προγράμματα, μία μνήμη στην οποία αποθηκεύονται και ανανεώνονται κατά την εκτέλεση του προγράμματος οι διάφορες μεταβλητές

καθώς και θύρες εισόδων – εξόδων (I/O ports) με τις οποίες μπορεί να επικοινωνήσει διαδραστικά και αμφίδρομα με τους χρήστες. Η κύρια διαφορά τους από τους σύγχρονους υπολογιστές έγκειται στο γεγονός ότι οι μικροελεγκτές έχουν περιορισμένη μνήμη (της τάξης μερικών Kbytes, τα οποία αρκούν για τις περισσότερες εφαρμογές, ακόμα και τις πιο απαιτητικές) ενώ δεν έχουν σκληρό δίσκο. Επιπλέον λειτουργούν με χαμηλή ισχύ (της τάξεως των 50mW σε σύγκριση με τα 50 W των ηλεκτρονικών υπολογιστών) και έχουν μικρότερη ταχύτητα επεξεργασίας δεδομένων, η οποία όμως σε μερικές περιπτώσεις φθάνει και τα 100 MIPS (Millions of Instructions Per Second: Εκατομμύρια εντολές το δευτερόλεπτο), ταχύτητα αρκετά ικανοποιητική για τις περισσότερες απαιτητικές εφαρμογές. Ο λόγος για τον οποίο οι μικροελεγκτές έχουν αντικαταστήσει τους ηλεκτρονικούς υπολογιστές είναι το αρκετά μικρό κόστος τους, το μεγάλο πεδίο περιβαλλοντικών συνθηκών στις οποίες μπορούν να αντεπεξέλθουν, πράγμα που τους καθιστά ιδανικούς για χρήση στο απαιτητικό βιομηχανικό περιβάλλον, καθώς και η απόκριση πραγματικού χρόνου (real time processing). Το τελευταίο χαρακτηριστικό εξασφαλίζει στους μικροελεγκτές τη δυνατότητα να αντιλαμβάνονται και να αποκρίνονται στον επιθυμητό για μας χρόνο με μεγάλη ακρίβεια. Μία από τις πρώτες εταιρείες, η οποία σχεδίασε μικροελεγκτές με πολλά περιφερειακά ενσωματωμένα σε ένα μόνο chip, ήταν η Intel με τη σειρά 8051 που χρησιμοποιείται ακόμα και σήμερα στο σχεδιασμό νέων προϊόντων. Άλλες εταιρείες που ακολούθησαν και τροφοδοτούν σήμερα την παγκόσμια αγορά με μικροελεγκτές είναι κυρίως οι: Atmel, Microchip, Motorola, Hitachi, Toshiba, AMD, Zilog, National Semiconductor, Philips/Sigmetics.

Για να μπορέσει κάποιος να αναπτύξει μία βιομηχανική εφαρμογή (π.χ. οδήγηση ενός βηματικού κινητήρα) δεν χρειάζονται πολλά. Κατ' αρχήν απαιτείται η αγορά μικροελεγκτών με τις επιθυμητές ιδιότητες σύνδεσης και επεξεργασίας καθώς και μερικών ηλεκτρονικών μικροεξαρτημάτων αμελητέου κόστους (πυκνωτές, μπαταρίες 9V, LED κ.λπ.). Απαιτείται επίσης ένα ολοκληρωμένο περιβάλλον (IDE και μεταγλωττιστής) για τον προγραμματισμό τους. Σήμερα υποστηρίζονται γλώσσες προγραμματισμού όπως η C, η Basic, η Pascal κλπ. Επιπλέον απαραίτητη είναι η ύπαρξη ενός προγραμματιστή (με τη μορφή ηλεκτρονικής πλακέτας), έτσι ώστε να μπορούμε να κατεβάσουμε στους μικροελεγκτές μας τα προγράμματα που φτιάχνουμε στο ολοκληρωμένο περιβάλλον. Τα ηλεκτρονικά κυκλώματα τα οποία θα φτιάχνουμε μπορούμε να τα υλοποιήσουμε είτε πάνω σε PCB (printed circuit boards) είτε σε breadboards.



Σχήμα 5.2 Ανάπτυξη ρομποτικού ψαριού ελεγχόμενου από μικροελεγκτή PIC

5.1.3 Δυνατότητες – Πλεονεκτήματα

Οι δυνατότητες των σύγχρονων μικροελεγκτών καλύπτουν το μεγαλύτερο μέρος των εφαρμογών της βιομηχανίας. Πιο αναλυτικά, οι περισσότεροι μικροελεγκτές έχουν δυνατότητες πολλαπλών αναλογικοψηφιακών μετατροπών (ADC είσοδοι) για λήψη μετρήσεων από όλων των ειδών αισθητήρων που υπάρχουν στην αγορά, έλεγχο κινητήρων (συνεχούς και εναλλασσόμενου ρεύματος, βηματικούς κ.λπ.) με χρήση διαμόρφωσης εύρους παλμού (Pulse Width Modulation PWM) και δυνατότητα προγραμματισμού κατευθυντών PID. Επιπλέον μπορούν να συνδεθούν με υπολογιστές μέσω σειριακής θύρας (RS-232), παράλληλης θύρας (IEEE 1284), καθώς και των βιομηχανικών πρωτοκόλλων RS-422 και RS-485. Επιπλέον, οι συνηθισμένοι μικροελεγκτές μπορούν να συνδεθούν με όλους τους αισθητήρες και υπολογιστές οι οποίοι είναι συμβατοί και όχι μόνο, με χρήση διαδεδομένων πρωτοκόλλων επικοινωνίας όπως τα I2C, CAN, SPI, RF κ.λπ. Εξειδικευμένοι μικροελεγκτές επιτρέπουν τη σύνδεσή τους μέσω Bluetooth, USB, ακόμα και σύνδεση με το διαδίκτυο μέσω Ethernet. Για ένα μέσο μικροελεγκτή, η ταχύτητά τους ξεκινά από το 1 MIPS και μπορεί να φτάσει και τα 100 MIPS, ταχύτητα αρκετά ικανοποιητική για απαιτητικές εφαρμογές όπως η μετάδοση βίντεο συνεχούς ροής (videostreaming), η επεξεργασία εικόνας (image processing) και η ψηφιακή επεξεργασία σήματος (DSP). Ένας μέσος μικροελεγκτής έχει σχεδίαση 8-bit, ενώ υπάρχουν και μικροελεγκτές με αρχιτεκτονική 32-bit, γεγονός που τους δίνει την ικανότητα να εκτελούν γρήγορα πράξεις σε πραγματικό χρόνο. Με λίγα λόγια οι σύγχρονοι μικροελεγκτές μπορούν επάξια να αντικαταστήσουν σε ένα μεγάλο ποσοστό εργασιών τους υπολογιστές, τις ακριβές κάρτες οδήγησης-συλλογής δεδομένων και τα PLC. Είναι σαφώς φθηνότεροι, επομένως εάν καταστραφούν δεν δημιουργούν πρόβλημα, καταλαμβάνουν μικρότερο όγκο, δεν απαιτούν εξειδικευμένες γνώσεις (όπως π.χ. προγραμματισμό σε διαγράμματα ladder στα PLC), αλλά μόνο γενικές γνώσεις προγραμματισμού. Επιπλέον με τη χρήση μικροελεγκτών δεν έχουμε κανένα περιορισμό όσον αφορά το είδος του προγράμματος, σε αντίθεση με τα PLC ή με ακριβές κάρτες οδήγησης που έχουν συγκεκριμένες προ-πληρωμένες δυνατότητες (π.χ. μορφές ελέγχου από τις οποίες ο χρήστης επιλέγει). Για την επίτευξη ελέγχου ή μετρήσεων σε πραγματικό χρόνο, οι ηλεκτρονικοί υπολογιστές πρέπει να τρέχουν λειτουργικά συστήματα πραγματικού χρόνου (όπως RT-Linux, QNX κ.ά.), ενώ οι μικροελεγκτές δεν απαιτούν επιπλέον λογισμικό. Τέλος, ένα άλλο πλεονέκτημα των μικροελεγκτών είναι η πολλή μικρή κατανάλωση ισχύος. Τόσο οι υπολογιστές όσο και τα PLC έχουν μικρή ενεργειακή αυτονομία σε σχέση με τους μικροελεγκτές. Έτσι για παράδειγμα ένα laptop μπορεί να εργαστεί για 2 ώρες χωρίς ρεύμα από το δίκτυο, ενώ ένας μέσος μικροελεγκτής μπορεί να εργάζεται για μέρες. Υπάρχουν δε και μικροελεγκτές χαμηλής κατανάλωσης οι οποίοι λειτουργούν με τάση κάτω των 2V!

5.2 Ενσωματωμένα Συστήματα

5.2.1 Γενικά

Ένα ενσωματωμένο σύστημα (embedded system), είναι ένα υπολογιστικό σύστημα ειδικού σκοπού, που είναι πολλές φορές ενσωματωμένο μέσα στην συσκευή που ελέγχει. Ένα ενσωματωμένο σύστημα έχει ειδικές απαιτήσεις και εκτελεί προκαθορισμένες λειτουργίες, σε αντίθεση με έναν γενικού σκοπού προσωπικό υπολογιστή.

5.2.2 Παραδείγματα

Μερικές από τις συσκευές στις οποίες χρησιμοποιούνται ενσωματωμένα συστήματα είναι οι παρακάτω:

- Συσκευές αυτόματης συναλλαγής (ATMs)
- Κινητά τηλέφωνα και τηλεφωνικοί μεταγωγείς
- Κινητά τηλέφωνα με επιπλέον δυνατότητες, προσωπικούς ψηφιακούς οδηγούς (PDAs) υποστήριξη Java
- Εξοπλισμός δικτύων υπολογιστών, όπως δρομολογητές, timeservers και τείχη προστασίας
- Εκτυπωτές υπολογιστών
- Πολυλειτουργικοί εκτυπωτές (MFPs)
- Αντιγραφικά
- Οδηγοί δίσκων (δισκέτες και σκληροί δίσκοι)
- Ελεγκτές μηχανών και ελεγκτές κατά του μπλοκαρίσματος των τροχών για τα φρένα των αυτοκινήτων
- Προϊόντα αυτοματισμού για το σπίτι, όπως θερμοστάτες, κλιματιστικά, ψεκαστήρες και συστήματα ασφαλείας
- Οικιακές συσκευές, όπως φούρνοι μικροκυμάτων, πλυντήρια, σύνθετα τηλεόρασης και μηχανές αναπαραγωγής / εγγραφής ψηφιακών δίσκων
- Συστήματα αδρανειακής καθοδήγησης, υλικό / λογισμικό ελέγχου πτήσης και άλλα ολοκληρωμένα συστήματα στην αεροπλοΐα και την πυραυλική τεχνολογία
- Ιατρικός εξοπλισμός
- Εξοπλισμός μετρήσεων, όπως παλμογράφοι, αναλυτές κυκλωμάτων και αναλυτές φάσματος
- Υπολογιστές τσέπης
- Πολυλειτουργικά ρολόγια χειρός
- Σταθερές και φορητές παιχνιδομηχανές
- Προσωπικοί ψηφιακοί οδηγοί (PDAs)
- Προγραμματιζόμενοι ελεγκτές (PLCs) για βιομηχανικό αυτοματισμό και παρακολούθηση



Σχήμα 5.3 Ευρύ φάσμα εφαρμογών των ενσωματωμένων συστημάτων

5.2.3 Χαρακτηριστικά

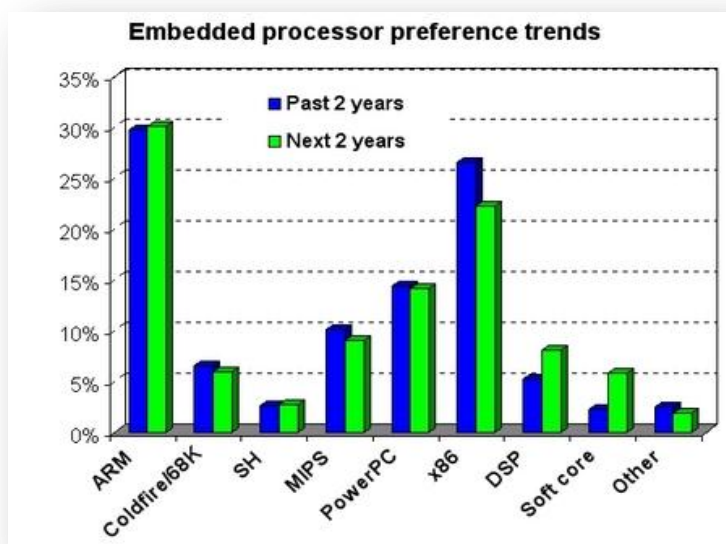
Τα ενσωματωμένα συστήματα συνήθως σχεδιάζονται ώστε να εκτελούν επιλεγμένες λειτουργίες με χαμηλό κόστος. Το σύστημα μπορεί να χρειαστεί να είναι πολύ γρήγορο για κάποιες λειτουργίες, αλλά για τις περισσότερες από τις υπόλοιπες πιθανότατα κάτι τέτοιο δεν θα χρειάζεται. Με αυτό το σκεπτικό, πολλά μέρη ενός ενσωματωμένου συστήματος θα έχουν μειωμένη απόδοση. Η βραδύτητα δεν σχετίζεται μονάχα με την ταχύτητα του ρολογιού. Όλη η αρχιτεκτονική ενός ενσωματωμένου συστήματος είναι συχνά σκόπιμα απλουστευμένη, με στόχο το μικρότερο κόστος, σε σχέση με το υλικό των υπολογιστών γενικού σκοπού. Για παράδειγμα, τα ενσωματωμένα συστήματα χρησιμοποιούν συχνά περιφερειακά που ελέγχονται από σειριακές διεπαφές, οι οποίες είναι δεκάδες έως εκατοντάδες φορές πιο αργές σε σύγκριση με τα περιφερειακά που χρησιμοποιούνται στους προσωπικούς υπολογιστές. Εφόσον πολλά ενσωματωμένα συστήματα παράγονται μαζικά κατά εκατομμύρια, η μείωση του κόστους είναι ένα κυρίαρχο ζήτημα.. Μερικά ενσωματωμένα συστήματα δεν απαιτούν μεγάλη υπολογιστική ισχύ ή πόρους και αυτό επιτρέπει την ελαχιστοποίηση του κόστους παραγωγής χρησιμοποιώντας έναν, σχετικά, αργό επεξεργαστή και μικρό μέγεθος μνήμης.

Με τον όρο *firmware*, εννοούμε το λογισμικό που είναι ενσωματωμένο σε μία συσκευή, για παράδειγμα σε μια μνήμη ROM ή Flash. Τα προγράμματα σε ένα ενσωματωμένο σύστημα συχνά εκτελούνται με περιορισμούς πραγματικού χρόνου και με περιορισμένους πόρους υλικού, όπως χωρίς σκληρό δίσκο, λειτουργικό σύστημα, πληκτρολόγιο ή οθόνη. Το πρόγραμμα μπορεί να μην διαθέτει κανένα σύστημα αρχείων. Αν υποστηρίζει μία διεπαφή χρήστη, αυτή μπορεί να αποτελείται από ένα πληκτρολόγιο και μία οθόνη υγρών κρυστάλλων. Τα ενσωματωμένα συστήματα τοποθετούνται σε μηχανήματα που αναμένεται να λειτουργούν διαρκώς και χωρίς σφάλματα. Με βάση τα παραπάνω το *firmware* αναπτύσσεται και δοκιμάζεται με μεγαλύτερη προσοχή από ότι το λογισμικό των προσωπικών υπολογιστών. Τα περισσότερα ενσωματωμένα συστήματα δεν χρησιμοποιούν μηχανικά μέρη, όπως οδηγούς δίσκων, διακόπτες ή κουμπιά, καθώς αυτά είναι αναξιόπιστα σε σχέση με τα σταθερά μέρη που χρησιμοποιούν τρανζίστορ, όπως μια μνήμη flash.

Τέλος πολλά ενσωματωμένα συστήματα μπορεί να τοποθετούνται εκεί όπου δεν είναι δυνατή η πρόσβαση από άνθρωπο και για το λόγο αυτό τα ενσωματωμένα συστήματα αυτά θα πρέπει να έχουν την δυνατότητα να μηδενίζονται και να αρχικοποιούνται ακόμα και σε περιπτώσεις καταστροφικών διακοπών στην λειτουργία τους. Αυτό συνήθως επιτυγχάνεται με ένα πρότυπο ηλεκτρονικό κομμάτι που ονομάζεται χρονόμετρο *watchdog*, το οποίο μηδενίζει τον υπολογιστή εάν το λογισμικό δεν μηδενίζει περιοδικά το χρονόμετρο.

5.2.4 Κυρίαρχες αρχιτεκτονικές

Υπάρχουν πολλές διαφορετικές αρχιτεκτονικές επεξεργαστών που χρησιμοποιούνται στα ενσωματωμένα συστήματα όπως οι ARM, MIPS, Coldfire/68k, PowerPC, X86, PIC, 8051 κ.α. Αυτό έρχεται σε αντίθεση με την αγορά των προσωπικών υπολογιστών, όπου ο ανταγωνισμός μεταξύ διαφορετικών αρχιτεκτονικών είναι περιορισμένος και κυρίως μεταξύ των Intel/AMD x86, και των Apple/Motorola/IBM PowerPC για τα Macintosh.



Σχήμα 5.4 Διαφορετικές αρχιτεκτονικές επεξεργαστών και ποσοστό χρησιμοποίησής τους σε ενσωματωμένα συστήματα

5.3 Αρχιτεκτονική ARM

Η ARM είναι η εταιρεία που κυριαρχεί στην κατασκευή και παροχή 32-bit embedded επεξεργαστών RISC αρχιτεκτονικής. Η Reduced Instruction Set Computer (RISC) Instruction Set Architecture (ISA) είναι η πιο ευρέως χρησιμοποιούμενη 32 bit ISA όσον αφορά τα παραγόμενα μεγέθη. Αρχικά η αρχιτεκτονική αυτή δημιουργήθηκε για τους προσωπικούς υπολογιστές στους οποίους τώρα επικρατεί η αρχιτεκτονική x86. Η απλότητα των ARM επεξεργαστών τους έκανε κατάλληλους για να χρησιμοποιηθούν σε χαμηλής κατανάλωσης εφαρμογές. Αυτό τους έκανε να επικρατήσουν στην αγορά κινητών και ενσωματωμένων συστημάτων ως χαμηλού κόστους και μεγέθους μικροεπεξεργαστές και μικροελεγκτές. Η RISC αρχιτεκτονική βασίζεται στη λογική ότι απλές εντολές σε αντίθεση με πιο πολύπλοκες, μπορούν να παρέχουν υψηλότερη απόδοση εφόσον αυτή η απλότητα οδηγεί στην πιο γρήγορη εκτέλεση της κάθε εντολής. Ιδιαίτερη προσοχή πρέπει να δοθεί στη λέξη reduced που φυσικά δεν αναφέρεται στο πλήθος των εντολών αλλά στο φόρτο που προκαλούν αυτές κατά την εκτέλεσή τους. Με άλλα λόγια η κάθε εντολή συνήθως εκτελείται σε έναν κύκλο ρολογιού.

Πίνακας 5.1 Οικογένεια επεξεργαστών ARM

Αρχιτεκτονική	Οικογένεια
ARMv1	ARM1
ARMv2	ARM2, ARM3
ARMv3	ARM6, ARM7
ARMv4	StrongARM, ARM7TDMI, ARM9TDMI
ARMv5	ARM7EJ, ARM9E, ARM10E, XScale
ARMv6	ARM11
ARMv7	Cortex

Η αρχιτεκτονική έχει εξελιχθεί και, αρχίζοντας από τη σειρά πυρήνων Cortex, ορίζονται τρία «προφίλ» ("profiles"):

- «Εφαρμογής» ("Application"): σειρά Cortex-A

- «Πραγματικού χρόνου» ("Real-time"): σειρά Cortex-R
- «Μικροελεγκτή» ("Microcontroller"): σειρά Cortex-M

Η αρχιτεκτονική ARM περιλαμβάνει τα εξής χαρακτηριστικά RISC:

- Αρχιτεκτονική φόρτωσης/αποθήκευσης (load/store).
- Δεν υποστηρίζει μη-ευθυγραμμισμένη (misaligned) πρόσβαση στη μνήμη (υποστηρίζεται στους πυρήνες ARMv6, με εξαιρέσεις όσον αφορά τις εντολές φόρτωσης/αποθήκευσης πολλών λέξεων).
- Ομοιόμορφο αρχείο 16 × 32-bit καταχωρητών.
- Σταθερό εύρος εντολών 32-bit για εύκολη αποκωδικοποίηση και διοχέτευση (pipelining), με μειονέκτημα την μικρότερη πυκνότητα κώδικα. Αργότερα, το σύνολο εντολών Thumb αύξησε την πυκνότητα του κώδικα.
- Συνήθως η εκτέλεση γίνεται σε έναν κύκλο.

Η απλή σχεδίαση, σε σχέση με τους επεξεργαστές Intel 80286 και Motorola 68020, που κυκλοφόρησαν την ίδια εποχή, διορθώθηκε με προσθήκη των εξής χαρακτηριστικών:

- Εκτέλεση υπό συνθήκη των περισσότερων εντολών, που μειώνει την επιβάρυνση διακλάδωσης (branch overhead) και αντιμετωπίζει την έλλειψη μηχανισμού πρόβλεψης διακλάδωσης (branch predictor).
- Οι αριθμητικές εντολές αλλάζουν τον κώδικα κατάστασης μόνο όταν χρειάζεται.
- Ολισθητής βαρελιού (barrel shifter) των 32-bit, που μπορεί να χρησιμοποιηθεί χωρίς επιβάρυνση στην ταχύτητα, στις περισσότερες εντολές αριθμητικής και υπολογισμού διευθύνσεων.
- Ισχυροί τρόποι σχηματισμού διευθύνσεων με δείκτες (indexed addressing modes).
- Ένας καταχωρητής συνδέσμου (link register) για γρήγορες κλήσεις συναρτήσεων-φύλλων (leaf function calls).
- Απλό αλλά γρήγορο υποσύστημα διακοπών 2 επιπέδων προτεραιοτήτων, με εναλλασσόμενους πίνακες καταχωρητών (switched register banks).

5.4 Μικροελεγκτές ARM7TDMI

Οι επεξεργαστές ARM7 είναι 32-bit cores RISC αρχιτεκτονικής (Reduced Instruction Set Computer). Διαθέτουν ένα 32-bit bus για εντολές και δεδομένα. Το μήκος των δεδομένων μπορεί να είναι 8/16/32 bits, ενώ το μήκος των εντολών είναι 32bits. Η εκτέλεση των περισσότερων εντολών απαιτεί ένα μόνο κύκλο ρολογιού. Οι επεξεργαστές ARM7 είναι επεξεργαστές χαμηλής κατανάλωσης, με μικρό κόστος και απόδοση έως και 130MIPS. Η οικογένεια των επεξεργαστών ARM7 ενσωματώνει 2 σύνολα εντολών, το σύνολο εντολών Thumb των 16bits και το σύνολο εντολών ARM των 32bits. Οι αρχιτεκτονικές επεξεργαστών παραδοσιακά όριζαν εντολές και δεδομένα ίδιου μεγέθους. Για αυτό το λόγο οι αρχιτεκτονικές των 32bits είχαν μεγαλύτερη απόδοση, διαχειριζόμενες δεδομένα των 32 bits και μπορούσαν να διευθυνσιοδοτήσουν μεγαλύτερο χώρο μνήμης από τις αντίστοιχες των 16bits. Από την άλλη οι αρχιτεκτονικές των 16bits αν και παρουσίαζαν σαφώς πιο συμπαγή μορφή κώδικα εμφάνιζαν τη μισή απόδοση. Με το σύνολο εντολών Thumb εισάγεται η δυνατότητα ύπαρξης εντολών των 16bits σε αρχιτεκτονική των 32bits. Με αυτόν τον τρόπο εξασφαλίζονται υψηλότερες επιδόσεις από τις αντίστοιχες της αρχιτεκτονικής των 16bits και συμπαγέστερος

κώδικας από τον αντίστοιχο της αρχιτεκτονικής των 32bits. Ο κώδικας Thumb έχει το 65% του μεγέθους του κώδικα ARM και παρέχει 60% μεγαλύτερη απόδοση από τον ARM όταν οι εντολές εκτελούνται από ένα σύστημα μνήμης των 16bits. Οι εντολές Thumb λειτουργούν με την κανονική διαμόρφωση των καταχωρητών για εντολές ARM, επιτρέποντας εναλλαγές μεταξύ ARM-Thumb και αντίστροφα. Κατά την εκτέλεση, οι εντολές Thumb αποσυμπίεζονται σε εντολές ARM σε πραγματικό χρόνο, χωρίς κόστος στην απόδοση του συστήματος. Οι επεξεργαστές ARM χειρίζονται τη μνήμη σε μια γραμμική συλλογή από bytes αριθμημένα σε αύξουσα σειρά ξεκινώντας από το μηδέν, ενώ υποστηρίζουν τη δομή little endian και τη δομή big endian για τον καθορισμό των most significant bits κάθε λέξης. Επίσης οι επεξεργαστές ARM7 περιέχουν 37 καταχωρητές, 6 εκ των οποίων είναι καταχωρητές κατάστασης, ενώ οι υπόλοιποι είναι γενικού σκοπού. Οι επεξεργαστές είναι σχεδιασμένοι για να ικανοποιούν τις απαιτήσεις μεγάλου πλήθους διαφορετικών εφαρμογών και υποστηρίζουν πολλά λειτουργικά συστήματα μεταξύ των οποίων Windows, Palm OS, Symbian OS και Linux. Οι επεξεργαστές που ανήκουν στην παραπάνω οικογένεια είναι ιδιαίτερα διαδεδομένοι σε ηλεκτρονικά προϊόντα καταναλωτών συμπεριλαμβανομένου PDAs, κινητά τηλέφωνα, iPods καθώς και άλλα ψηφιακά μέσα αναπαραγωγής ήχου, περιφερειακά H/Y όπως σκληροί δίσκοι και routers. Ο κώδικας που αναπτύσσεται για επεξεργαστές τύπου ARM7TDMI είναι απόλυτα συμβατός με όλους τους επεξεργαστές τύπου ARM7. Τα παραπάνω χαρακτηριστικά καθιστούν τους ARM7TDMI μια πολύ καλή επιλογή για τη δημιουργία ενσωματωμένων συστημάτων εξαιτίας του μικρού σχετικά μεγέθους, των επιδόσεων και της χαμηλής κατανάλωσης ενέργειας.

5.5 Ο Μικροελεγκτής ADuC-702x

Κατά τη διαδικασία ανάπτυξης μιας εφαρμογής που βασίζεται σε κάποια συγκεκριμένη προδιαγραφή ή ανάγκη υλοποίησης με χρήση μικροελεγκτών προσδιορίζεται πάντα από τον ίδιο τον σχεδιαστή της εφαρμογής. Ωστόσο είναι γεγονός ότι πάρα πολλές εφαρμογές μπορούν να υλοποιηθούν αποτελεσματικά με χρήση κάποιου μικροελεγκτή, ενώ σε αρκετές από αυτές τις περιπτώσεις μπορούμε να εκμεταλλευτούμε τα σημαντικά πλεονεκτήματα των μικροελεγκτών ADuC 702x (19/20/21/22/23/24/25/26/27/28/29). Ο μικροελεγκτής ADuC-702x διαθέτει έναν επεξεργαστή ARM7TDMI, αρχιτεκτονικής RISC. Τα γενικά χαρακτηριστικά του ADuC-702x είναι [18]:

- Analog-to-Digital-Converter (ADC) των 12 bits με 16 κανάλια και ικανότητα δειγματοληψίας 1MSPS (1 million samples per second). Μπορεί να λειτουργήσει με τρεις διαφορετικούς τρόπους:

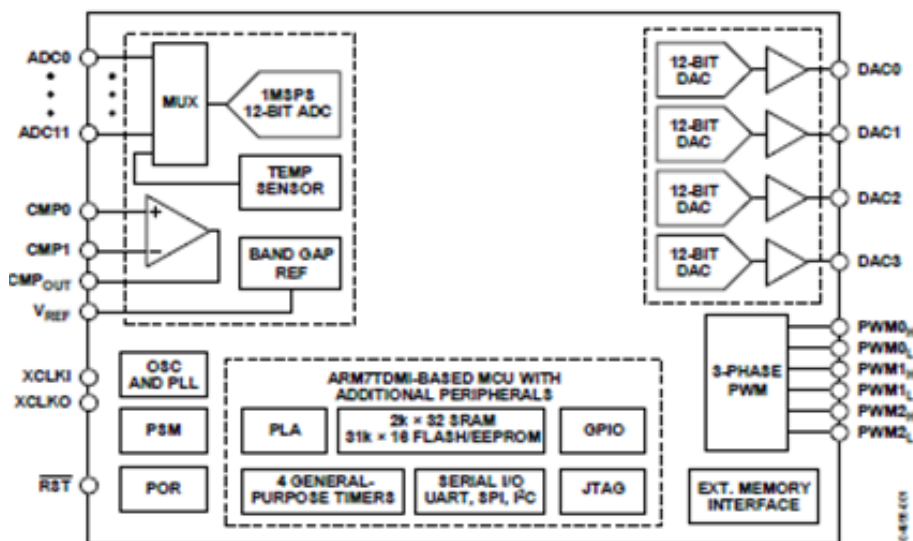
α) fully differential mode για μικρά ισορροπημένα σήματα

β) single-ended mode για κάθε single-ended σήμα

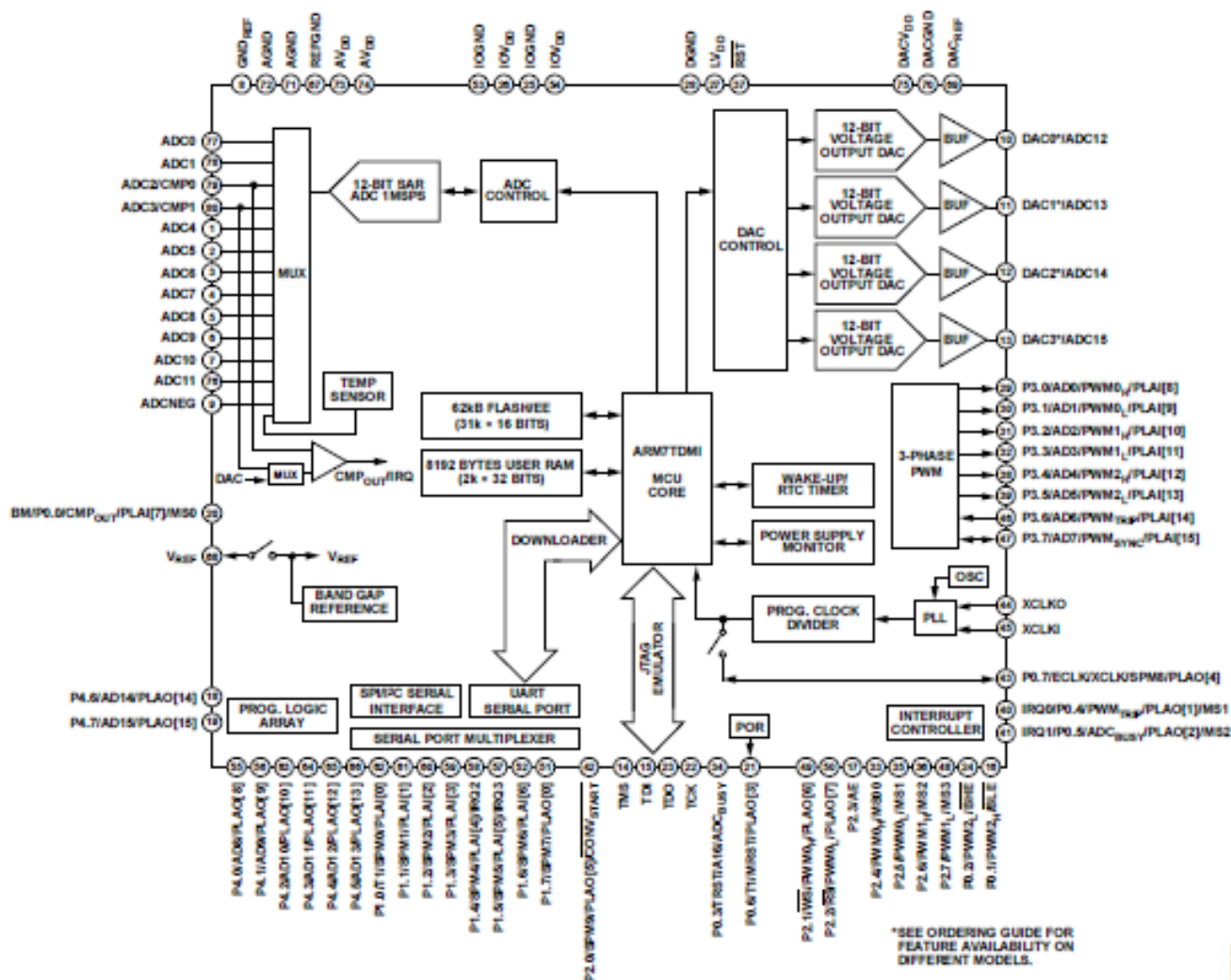
γ) pseudo-differential mode για κάθε single-ended σήμα αξιοποιώντας το πλεονέκτημα του rejection mode.

- Εύρος τάσης αναλογικών εισόδων 0-Vref.
- Digital-to-Analog-Converter (DAC) των 12 bits με 4 κανάλια
- On-chip κύκλωμα αναφοράς τάσης 2.5V που χρησιμοποιείται για τον ADC και τον DAC, ενώ η τάση αυτή εμφανίζεται και στο Vref pin.
- Αισθητήρας θερμοκρασίας υπό τη μορφή μίας τάσης εξόδου ανάλογης της απόλυτης θερμοκρασίας.
- Συγκριτής τάσης

- Θύρα JTAG για download και debug του κώδικα στον microcontroller
- Εσωτερικός ταλαντωτής στα 32.768kHz, εξωτερικός κρύσταλλος ρολογιού
- 45MHz phase-locked-loop
- Ενσωματωμένη (On-chip) μνήμη flash 64KB, με δυνατότητα Προγραμματισμού εντός του συστήματος, ως τμήμα προγράμματος, 8KB RAM για την αποθήκευση σταθερών τιμών και μεταβλητών
- Περιφερειακό ασύγχρονης σειριακής επικοινωνίας-Universal Asynchronous Receiver-Transmitter(UART)
- Δύο διαύλους διασύνδεσης ολοκληρωμένων κυκλωμάτων (Inter Integrated Circuit) για τη επέκταση των διασυνδέσεων με περιφερειακές μονάδες και τον δικτυακό έλεγχο διάφορων ολοκληρωμένων κυκλωμάτων.
- Μονάδα σειριακής επικοινωνίας (Serial Peripheral Interface) που αποτελεί μια πρότυπη μονάδα σύγχρονης σειριακής επικοινωνίας.
- Έναν διαμορφωτή εύρους Παλμών (Pulse Width Modulator)-16bits με τη βοήθεια του οποίου μπορούμε να λάβουμε αναλογικές τάσεις μέσα από κατάλληλα φίλτρα τύπου RC.
- Μια προγραμματιζόμενη λογική διάταξη (Programmable Logical Array), που περιλαμβάνει μια μήτρα συνδέσεων μεταξύ διάφορων λειτουργικών λογικών τμημάτων οδηγώντας σε πιο ολοκληρωμένες και λογικές λύσεις.
- Ικανότητα διασύνδεσης με εξωτερική μνήμη.
- 4 Χρονιστές γενικού σκοπού (TIMERS) για το χρονισμό ή τη σηματοδότηση διαφόρων γεγονότων ή για καταμέτρηση, χρονιστής wake-up για αφύπνιση του microcontroller όταν βρίσκεται σε sleep mode και watchdog timer για την αποφυγή της πιθανής κατάρρευσης του συστήματος.
- 40 θύρες γενικού σκοπού εισόδου/εξόδου (GPIO).
- Κύκλωμα τροφοδοσίας στα 3V.
- Περιφερειακό Power Supply Monitor που ελέγχει πιθανή πτώση τάσης κάτω από τα επιτρεπόμενα όρια.
- Κύκλωμα επανατοποθέτησης ή Reset που εξασφαλίζει ότι όλες οι εσωτερικές μονάδες και τα κυκλώματα ελέγχου θα ξεκινήσουν να λειτουργούν κατά τη διάρκεια της τροφοδοσίας από κάποια προκαθορισμένη κατάσταση, ενώ όλοι οι καταχωρητές θα βρίσκονται σε κατάλληλες αρχικές τιμές.



Σχήμα 5.5 Functional block diagram του ADuC 702x

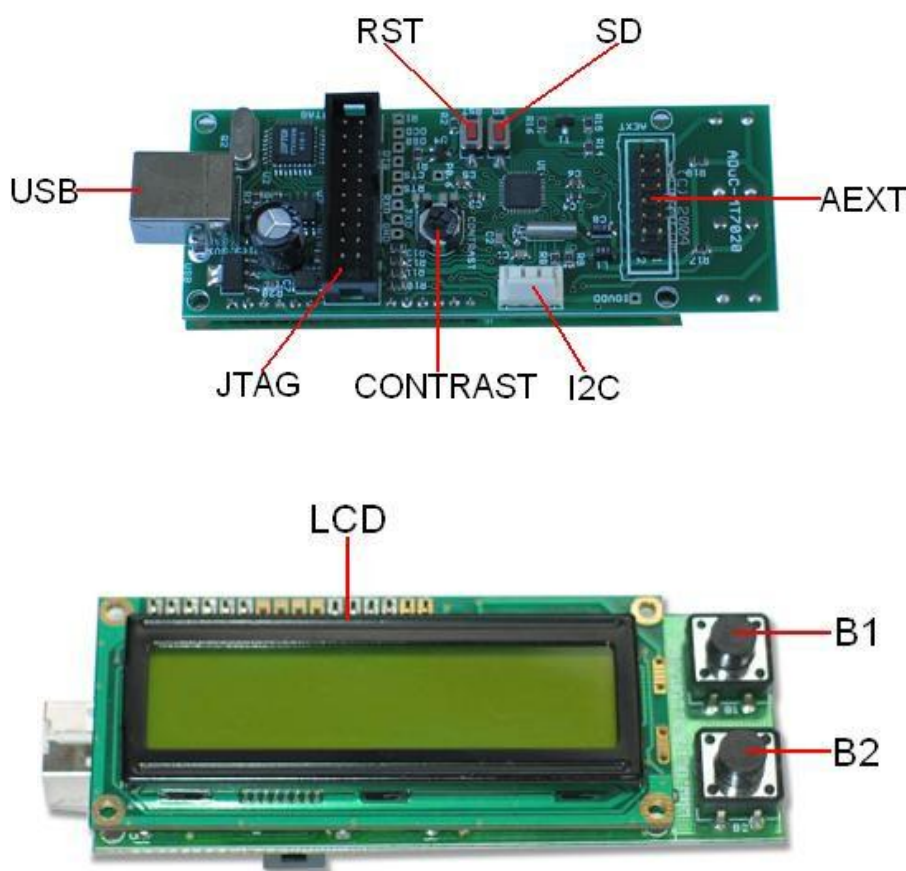


Σχήμα 5.6 Detailed block diagram του ADuC 702x

5.6 Η πλατφόρμα ανάπτυξης ADuC-MT7020 της Olimex

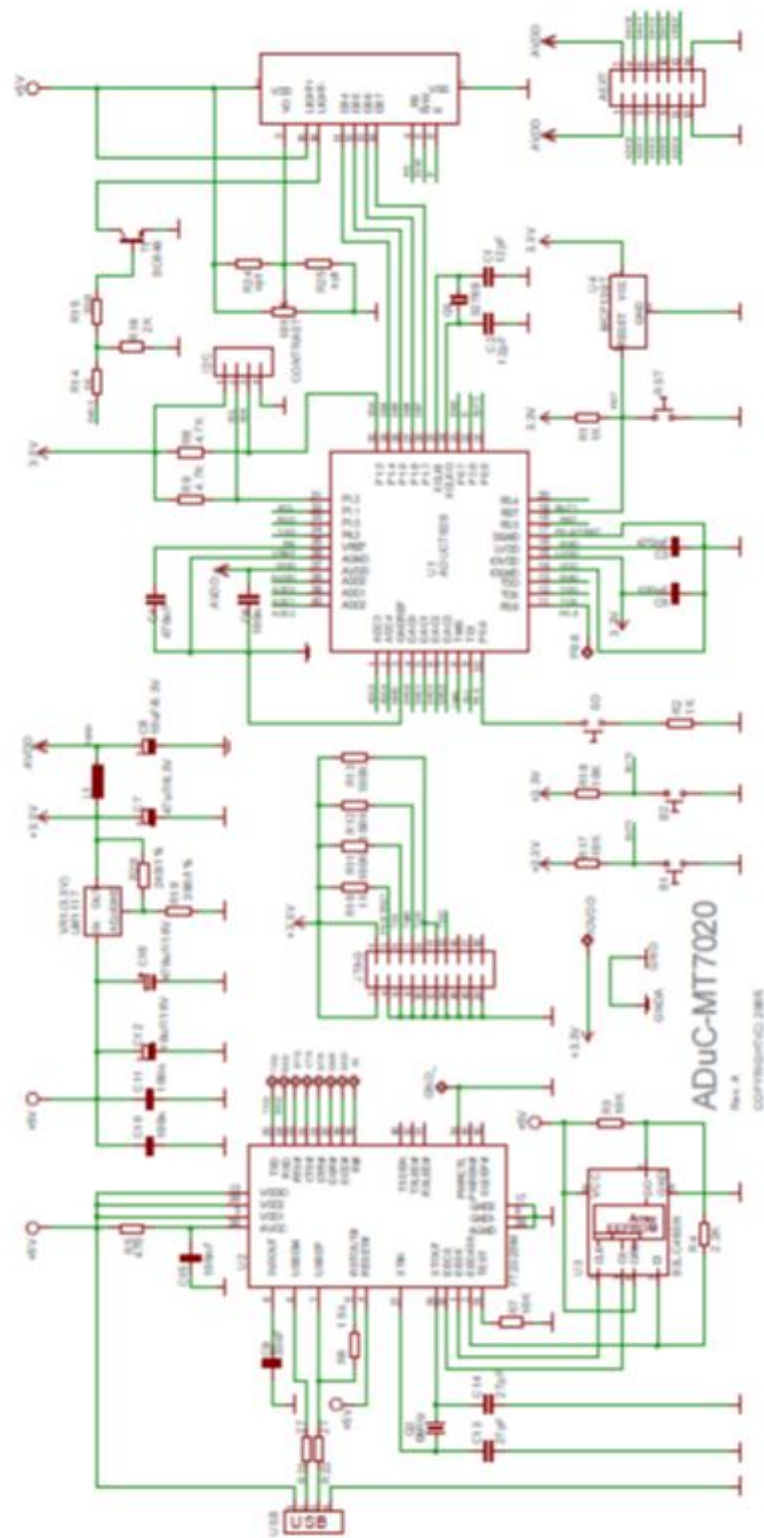
Η εταιρία Olimex μεταξύ άλλων προσφέρει επίσης και έναν αριθμό από αναπτυξιακά boards σε πολύ ανταγωνιστικές τιμές. Κάνοντας χρήση ενός αναπτυξιακού board μπορούμε να ελέγχουμε γρήγορα και αποτελεσματικά τα προγράμματα μας, πριν προχωρήσουμε στη σχεδίαση της κάρτας του τελικού προϊόντος. Όλες αυτές οι αναπτυξιακές πλατφόρμες διαθέτουν μια μονάδα επικοινωνίας με τον υπολογιστή μέσω της οποίας μπορούμε να φορτώσουμε το πρόγραμμα της εφαρμογής στον επεξεργαστή του board. Εφόσον όλοι οι μικροελεγκτές ADuC 702x προγραμματίζονται χωρίς την ανάγκη ειδικής συσκευής προγραμματισμού, το πρόγραμμα της εφαρμογής φορτώνεται απευθείας στη μνήμη του προγράμματος του ελεγκτή της κάρτας και στη συνέχεια ξεκινά αμέσως να εκτελείται. Το development board που χρησιμοποιήθηκε είναι το ADuC-MT7020 και τα βασικά του χαρακτηριστικά παραθέτονται παρακάτω [19]:

- MCU: ADuC 7020 - ARM7TDMI Core, 16/32-bit RISC architecture 62kB flash memory, 8KB SRAM, ADC με 5 κανάλια των 12 bits 1MSPS, DAC με 4 κανάλια των 12 bits ,bandgap voltage reference 2.5V, ενσωματωμένο αισθητήρα θερμοκρασίας, UARTs, 2x IIC, SPI, διαμορφωτή παλμών τριών φάσεων PWM, PLA προγραμματιζόμενης λογικής διάταξη, διασύνδεση εξωτερικής μνήμης, 2x32 bits timers, watchdog timer, I/O έως 5V, λειτουργία έως 45 MHz.
- Υποδοχή σύνδεσης ARM-JTAG 2x 10pins
- Μετατροπέα USB-RS232
- Σύνδεση I²C
- Reset κύκλωμα
- Πλήκτρο για serial download
- Δύο πλήκτρα B1 και B2 γενικής χρήσης
- Οθόνη LCD 16x2
- Κρύσταλλος 32.768kHz
- Πυκνωτή για το φιλτράρισμα της πηγής ισχύος.
- Κεφαλές προέκτασης για όλες τις θύρες του μικροελεγκτή.
- Πλήκτρα SD και Reset για εύκολη ενεργοποίηση του bootloader
- Δυνατότητα για τροφοδοσία μέσω θύρας USB, ώστε να υπάρχει εναλλακτική επιλογή για τροφοδοσία μέσω της θύρας ή μέσω του εξωτερικού τροφοδοτικού.
- Διαστάσεις 95x36 mm



Σχήμα 5.7 Το αναπτυξιακό board της Olimex ADuC-MT7020

SCHEMATIC



Σχήμα 5.7 Σχηματική αναπαράσταση του αναπτυξιακού board της Olimex ADuC-MT7020

5.7 Το περιβάλλον ανάπτυξης εφαρμογών μVision της Keil

Το μVision είναι μια πλατφόρμα ανάπτυξης που βασίζεται στο λειτουργικό σύστημα των Windows και αποτελεί σχεδιαστικό προϊόν της Keil [20]. Συνδυάζει έναν editor και έναν project manager, δίνοντας με αυτόν τον τρόπο τη δυνατότητα ανάπτυξης εφαρμογών για embedded systems και όχι μόνο ενώ ταυτόχρονα μέσω του debugger έχουμε την ικανότητα να κάνουμε πλήρη εξομοίωση ενός μικροελεγκτή με μηδενικό hardware στη διάθεσή μας. Το μVision εμπεριέχει όλα τα εργαλεία του C compiler, macro assembler, linker/locator, και HEX file generator. Το μVision επισπεύδει τη διαδικασία ανάπτυξης της ενσωματωμένης εφαρμογής παρέχοντας τα ακόλουθα:

- ✓ Πλήρη source code editor
- ✓ Αυτόματη μεταφορά των startup αρχείων βάση χαρακτηριστικών μιας ποικιλίας μικροελεγκτών. Τα συγκεκριμένα αρχεία εμπεριέχουν κώδικα προκειμένου να αρχικοποιήσουν και να εκκινήσουν τη CPU κατάλληλα για το συγκεκριμένο υλικό που χρησιμοποιούμε
- ✓ Project manager για τη δημιουργία και την ανάπτυξη projects,
- ✓ Δίνει τη δυνατότητα για assembling, compiling, και linking της

ενσωματωμένης εφαρμογής.

- ✓ Παράθυρα διαλόγου για τις ρυθμίσεις των αναπτυξιακών εργαλείων
- ✓ Debugger με high-speed CPU και εξομοιωτή περιφερειακών και

παρακολούθησης του προγράμματος σε real-time

- ✓ Δυνατότητα προγραμματισμού χρησιμοποιώντας flash memory για

κατέβασμα της εφαρμογής στη Flash ROM

- ✓ Links στα manuals των αναπτυξιακών εργαλείων, datasheets των

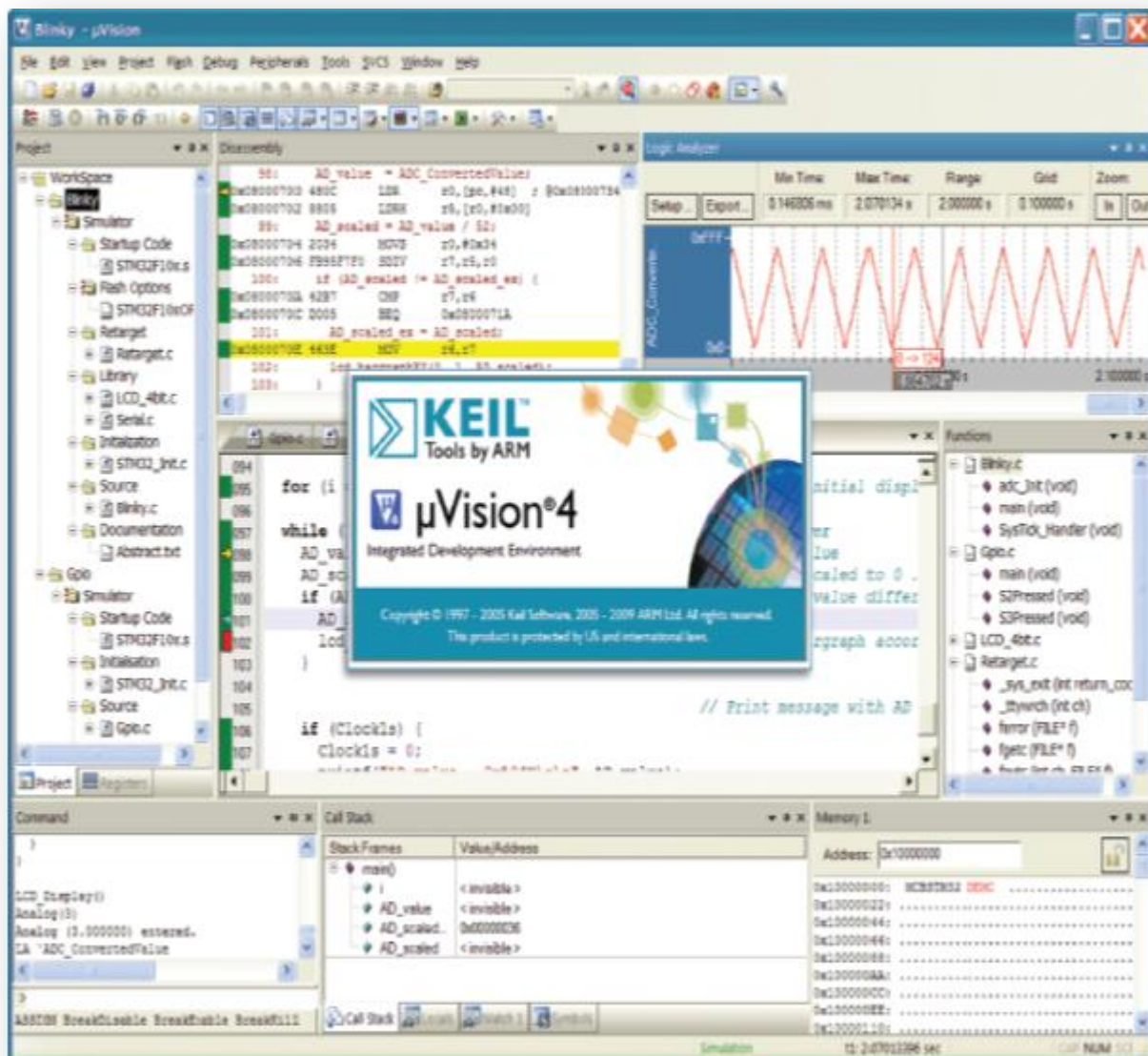
συσκευών και user's guides

- ✓ Παρακολούθηση κάθε στοιχείου που απαρτίζει τον μικροελεγκτή

- ✓ Εκτέλεση του προγράμματος step by step, συνεχή παρακολούθηση των

registers και των θέσεων μνήμης

Μέσω του μVision υποστηρίζεται το Build mode που κάνει compile τον κώδικα και το Debug mode που εξομοιώνει τον κώδικα.



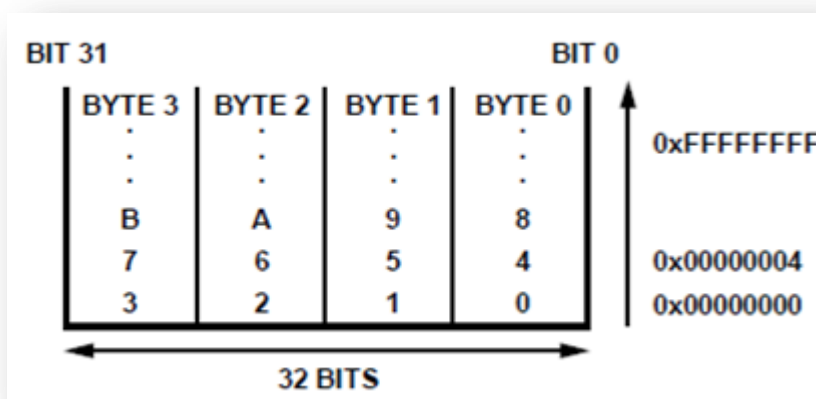
Σχήμα 5.8 Το περιβάλλον ανάπτυξης Keil μVision 4

ΚΕΦΑΛΑΙΟ 6^ο
ΚΥΡΙΑ ΠΕΡΙΦΕΡΕΙΑΚΑ ΤΟΥ ADUC
7020-ΠΑΡΑΔΕΙΓΜΑΤΑ

Στο σημείο αυτό αξίζει να γίνει μια εκτενέστερη περιγραφή σε κάποια από τα βασικότερα περιφερειακά του ADUC7020 microcontroller που αναφέρθηκαν στο προηγούμενο κεφάλαιο. Για την καλύτερη κατανόηση της χρήσης και αξιοποίησης των δυνατοτήτων του συγκεκριμένου μικροελεγκτή και των περιφερειακών του παραθέτουμε και κάποια παραδείγματα.

6.1 Flash/EE Memory - SRAM

Όπως προαναφέραμε, οι ADuC 7019/20/21/22/24/25/26/27/28/29 ενσωματώνουν 2 ξεχωριστά blocks μνήμης: 8kB SRAM και 64kB Flash/EE [18]. Από τα 64kB της μνήμης Flash/EE, τα 62 kB είναι διαθέσιμα στον χρήστη, ενώ τα υπόλοιπα 2kB είναι δεσμευμένα για factory-configured boot page. Ο επεξεργαστής ARM 7 «βλέπει» την μνήμη σαν μια γραμμική σειρά των 2^{32} byte. Η μνήμη είναι οργανωμένη σε μορφή little endian format, το οποίο σημαίνει ότι το λιγότερο σημαντικό byte είναι τοποθετημένο στην μικρότερη διεύθυνση byte address, ενώ το περισσότερο σημαντικό byte τοποθετείται στην υψηλότερη byte address, όπως φαίνεται και στο παρακάτω σχήμα:



Σχήμα 6.1 Little endian format οργάνωση μνήμης

Το πραγματικό εύρος της μνήμης Flash/EE είναι 16 bits, πράγμα το οποίο σημαίνει ότι σε ARM mode (όπου οι εντολές είναι 32-bits) απαιτούνται 2 προσπελάσεις στην μνήμη Flash/EE για κάθε εντολή. Έτσι λοιπόν, για βελτιστοποίηση της ταχύτητας προσπέλασης, προτιμάται η χρήση thumb mode. Συνεπώς, η μέγιστη ταχύτητα προσπέλασης για την μνήμη Flash/EE είναι τα 41.78 MHz σε thumb mode και τα 20.89 MHz σε ARM mode. 8 kB μνήμης SRAM είναι διαθέσιμα στον χρήστη, οργανωμένα σε $2 * 32$ bits, που είναι 2 words. Και εδώ, η συχνότητα είναι στα 41.78 MHz. Τέλος, έχουμε τους καταχωρητές MMR (Memory Mapped Registers), οι οποίοι παρέχουν μια διεπαφή μεταξύ της CPU και όλων των ενσωματωμένων (on-chip) περιφερειακών.

Οι καταχωρητές που σχετίζονται με την μνήμη Flash/EE είναι οι εξής : FEESTA, FEEMOD, FEECON, FEEDAT, FEEADR, FEESIGN, FEEPRO και FEEHIDE.

- Ο FEESTA είναι ένας καταχωρητής μόνο για ανάγνωση (read-only register), ο οποίος δείχνει την κατάσταση της flash control interface..
- Με τον καταχωρητή FEEMOD ορίζουμε τον τρόπο λειτουργίας της flash. ,
- Ο FEEDAT είναι ένας 16-bit καταχωρητής δεδομένων. .
- Ο FEEADR είναι ένας 16-bit καταχωρητής διεύθυνσης (address register).
- Ο FEESIGN είναι ένας 24-bit code signature.
- Οι καταχωρητές FEEPRO και FEEHIDE παρέχουν προστασία σε περίπτωση reset με την μόνη διαφορά ότι στον πρώτο απαιτείται κλειδί ασφαλείας, ενώ στον δεύτερο όχι. ,

6.2 ADC

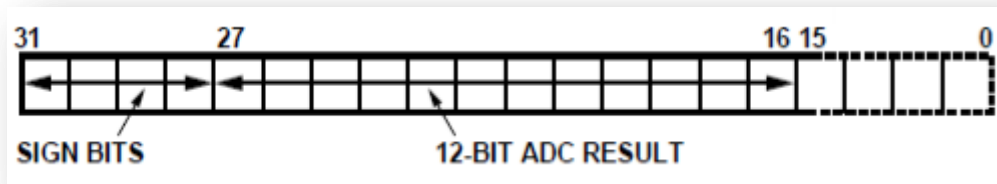
Όσον αφορά τον ADC ή αλλιώς τον αναλογικό σε ψηφιακό μετατροπέα των 12 bits, έχει 16 κανάλια και ικανότητα δειγματοληψίας έως και 1 MSPS, δηλαδή ένα εκατομμύριο δείγματα το δευτερόλεπτο [18]. Το εύρος τάσης του κυμαίνεται από 2.7V έως 3.6V. Ο εν λόγω μετατροπέας, μπορεί να λειτουργήσει σε τρία διαφορετικά modes (τρόποι λειτουργίας):

α) Fully Differential Mode για μικρά ισορροπημένα σήματα,

β) Single-Ended Mode για κάθε single-ended σήμα
και

γ) Pseudo-Differential Mode για κάθε single-ended σήμα, εκμεταλλευόμενος το πλεονέκτημα του rejection mode.

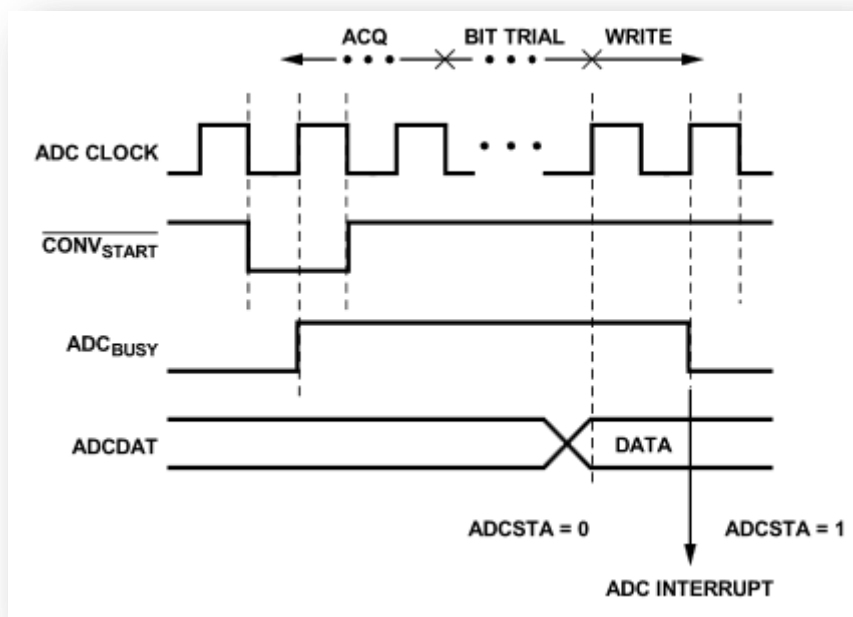
Ο ADC δέχεται εισόδους από την συσκευή που ελέγχει ο microcontroller ή εξωτερικά σήματα και παράγει ψηφιακά δεδομένα κατάλληλα για επεξεργασία. Το 12-bit αποτέλεσμα που προκύπτει, αποθηκεύεται προσωρινά στον ADC data register. Όπως φαίνεται και στο παρακάτω σχήμα, το 12-bit αποτέλεσμα αποθηκεύεται στα bits από 16 έως 27, ενώ τα πρώτα 4 bits αναφέρονται στο πρόσημο του αποτελέσματος (sign bits). Για αυτόν τον λόγο, όπως θα δούμε και αργότερα στον κώδικα του προγράμματός μας, όταν θέλουμε να αποθηκεύσουμε το αποτέλεσμα του ADC π.χ. σε έναν πίνακα , αρχικά το ολισθαίνουμε κατά 16 θέσεις.



Σχήμα 6.2 Τρόπος αποθήκευσης αποτελέσματος στον καταχωρητή ADCDAT

Ο προγραμματισμός του ADC γίνεται με την βοήθεια 8 καταχωρητών (Memory Mapped Registers): ADCCON , ADCCP, ADCCN, ADCSTA, ADCDAT, ADCRST, ADCGN, ADCOF.

- Ο ADCCON είναι ένας καταχωρητής ελέγχου, μέσω του οποίου ο προγραμματιστής ενεργοποιεί τα περιφερειακά του ADC, επιλέγει τον τρόπο λειτουργίας (Fully Differential Mode, Single-Ended Mode ή Pseudo Differential Mode) καθώς και τον τύπο της μετατροπής.
- Ο ADCCP είναι ένας καταχωρητής επιλογής θετικού καναλιού ADC
- Ο ADCCN είναι ένας καταχωρητής επιλογής αρνητικού καναλιού ADC.
- Ο ADCSTA είναι ένας καταχωρητής κατάστασης (status register), που υποδεικνύει αν το αποτέλεσμα της ADC μετατροπής είναι έτοιμο. Διαθέτει μόνο ένα bit, το ADCReady bit. Το bit αυτό ενεργοποιείται (παίρνει την τιμή 1) στο τέλος της ADC μετατροπής, δημιουργώντας ένα ADC interrupt (μια διακοπή). Όταν πραγματοποιείται μια μετατροπή, έχουμε την δυνατότητα να δούμε την κατάσταση του ADC μέσω του pin ADC_{BUSY}. Το pin αυτό έχει την τιμή 1 κατά την διάρκεια της μετατροπής και παίρνει την τιμή 0 όταν τελειώσει η μετατροπή.
- Στον καταχωρητή ADCDAT είναι αποθηκευμένο το 12-bit αποτέλεσμα της ADC μετατροπής.
- Με τον καταχωρητή ADCRST μπορούμε να κάνουμε reset σε όλους τους ADC καταχωρητές, απλά γράφοντας οποιαδήποτε τιμή στον καταχωρητή αυτό.
- Οι καταχωρητές ADCGN και ADCOF είναι 10-bit καταχωρητές gain calibration και offset calibration, αντίστοιχα.



Σχήμα 6.3 ADC timing

6.3 Reset

Υπάρχουν τέσσερις λόγοι για τους οποίους μπορεί να προκληθεί reset. Έτσι, ένα reset μπορεί να οφείλεται σε εξωτερικό παράγοντα, σε Power-on, στην λήξη του watchdog ή και σε θέματα λογισμικού. Οι καταχωρητές που σχετίζονται με το reset είναι: REMAP, RSTSTA και RSTCLR [18].

- Ο καταχωρητής RSTSTA δείχνει την αιτία για την οποία προκλήθηκε το τελευταίο reset
- Με τον καταχωρητή RSTCLR μπορούμε να «καθαρίσουμε» τον RSTSTA..

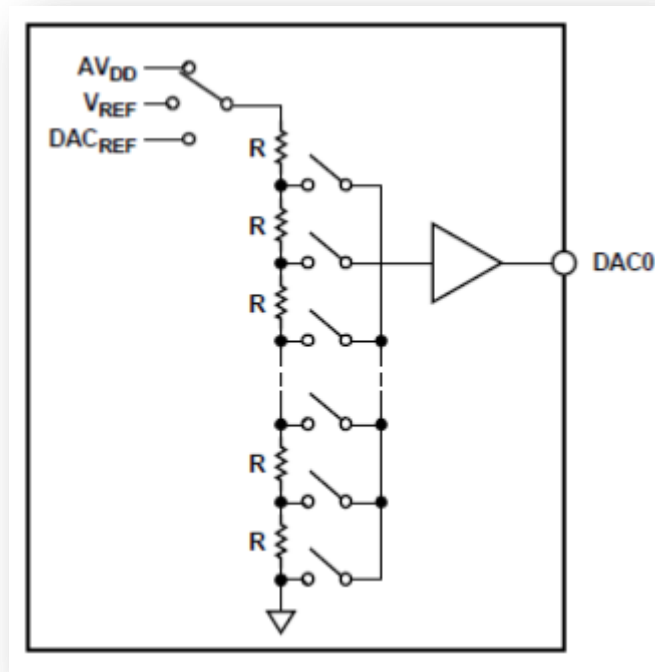
6.4 DAC

Ο μετατροπέας DAC είναι ένας digital-to-analog μετατροπέας και κάνει ουσιαστικά την αντίστροφη διαδικασία από τον ADC. Διαθέτει συνολικά 4 κανάλια, στα οποία παίρνουμε σαν έξοδο την αναλογική τάση ύστερα από κατάλληλη επεξεργασία σε ψηφιακά δεδομένα [18].

Κάθε μετατροπέας DAC μπορεί να έχει εύρος τάσης:

- 1) από 0 V έως V_{REF} (με εσωτερική τάση αναφοράς στα 2.5V),

- 2) από 0 V έως DAC_{REF} (όπου DAC είναι ισοδύναμη με την εξωτερική τάση αναφοράς),
- 3) από 0 V έως AV_{DD} .



Σχήμα 6.4 Δομή DAC

Κάθε ένας DAC μπορεί να προγραμματιστεί ανεξάρτητα από τους άλλους, μέσω δύο καταχωρητών, τους $DACxCON$ και $DACxDAT$, όπου το x μπορεί να πάρει τιμές από 1 έως 4, ανάλογα με το ποιος από τους DACs χρησιμοποιείται.

- Ο καταχωρητής $DACxCON$ είναι ένας καταχωρητής ελέγχου (control register), μέσω του οποίου καθορίζεται το εύρος τάσης λειτουργίας και ο τρόπος λειτουργίας του εκάστοτε μετατροπέα.
- Ο καταχωρητής $DACxDAT$ είναι καταχωρητής δεδομένων (data register), στον οποίο αποθηκεύεται προσωρινά το αποτέλεσμα της μετατροπής.

6.5 UART

Η UART είναι ένα περιφερειακό ασύγχρονης σειριακής επικοινωνίας (Universal Asynchronous Receiver-Transmitter). Πιο συγκεκριμένα, η UART αναπαριστά:

α) serial to parallel μετατροπές, σε χαρακτήρες δεδομένων που έχουν ληφθεί από μια περιφερειακή συσκευή ή modem

β) parallel to serial μετατροπές , σε χαρακτήρες δεδομένων που έχουν ληφθεί από την CPU.

Γενικά, η σειριακή επικοινωνία ακολουθεί ένα ασύγχρονο πρωτόκολλο, το οποίο υποστηρίζει ποικίλα word lengths, stop bits και parity generations, τα οποία καθορίζονται από τους καταχωρητές.

Η UART αποτελείται από 12 καταχωρητές: COMTX, COMRX, COMDIV0, COMIEN0, COMDIV1, COMIID0, COMCON0, COMCON1, COMSTA0, COMSTA1, COMSCR και COMDIV2 [18].

- Ο COMTX είναι ένας 8-bit καταχωρητής αποστολής δεδομένων (transmit register),
- Ο COMRX είναι ένας 8-bit καταχωρητής λήψης δεδομένων (receive register),
- Ο COMDIV0 είναι ένας low byte divisor latch,
- Ο COMIEN0 είναι ένας 8-bit καταχωρητής που χρησιμοποιείται για την ενεργοποίηση των διακοπών.
- Ο COMDIV1 είναι ένας high byte divisor latch
- Μέσω του καταχωρητή COMIID0, γίνεται identification της διακοπής (interrupt)
- Ο COMCON0 είναι καταχωρητής ελέγχου line control register. Με το 7^ο bit του καταχωρητή COMCON0, επιτρέπεται η πρόσβαση στα COMDIV0 και COMDIV1, προκειμένου αργότερα να μπορούμε να στείλουμε και να λάβουμε δεδομένα,
- Ο COMCON1 είναι καταχωρητής ελέγχου modem,
- Ο COMSTA0 είναι καταχωρητής κατάστασης line status register,
- Ο COMSTA1 είναι καταχωρητής κατάστασης modem,
- Ο COMSCR είναι ένας 8-bit καταχωρητής που χρησιμοποιείται για προσωρινή αποθήκευση. Μπορεί ακόμα να χρησιμοποιηθεί στην network addressable UART mode,
- Τέλος, ο COMDIV2 είναι ένας 16-bit fractional baud divide register.

6.6 IRQ / FIQ

Υπάρχουν 23 πηγές interrupt (διακοπών) στους microcontroller ADuC 7019/20/21/22/24/25/26/27/28/29, οι οποίες διακοπές ελέγχονται από τον interrupt controller. Οι περισσότερες από αυτές δημιουργούνται από τα ενσωματωμένα περιφερειακά, όπως οι ADC και UART. Υπάρχουν τέσσερις επιπρόσθετες πηγές interrupt, οι οποίες δημιουργούνται από εξωτερικά interrupt request pins, τα: IRQ0, IRQ1, IRQ2 και IRQ3 [18].

Table 158. IRQ/FIQ MMRs Bit Description

Bit	Description
0	All interrupts OR'ed (FIQ only)
1	SWI
2	Timer0
3	Timer1
4	Wake-up timer (Timer2)
5	Watchdog timer (Timer3)
6	Flash control
7	ADC channel
8	PLL lock
9	I2C0 slave
10	I2C0 master
11	I2C1 master
12	SPI slave
13	SPI master
14	UART
15	External IRQ0
16	Comparator
17	PSM
18	External IRQ1
19	PLA IRQ0
20	PLA IRQ1
21	External IRQ2
22	External IRQ3
23	PWM trip (IRQ only)/PWM sync (FIQ only)

Σχήμα 6.5 Αντιστοιχία bits των IRQ/FIQ registers με τις πηγές διακοπής

Ο μικροεπεξεργαστής ARM7TDMI CPU, διαχωρίζει τα interrupts σε δυο κατηγορίες, τις IRQ (normal interrupt request) και FIQ (fast interrupt request). Όλες οι διακοπές μπορούν να καθοριστούν ξεχωριστά. Ο έλεγχος και ο καθορισμός του συστήματος διακοπής γίνεται μέσω 9 καταχωρητών. Τέσσερις από αυτούς τους καταχωρητές αναφέρονται στην IRQ και άλλοι τέσσερις στην FIQ. Ένας επιπλέον καταχωρητής χρησιμοποιείται για να επιλέξει την πηγή του interrupt, καθορίζει δηλαδή το περιφερειακό εκείνο το οποίο μπορεί να προκαλέσει interrupt.

Το interrupt IRQ εξυπηρετεί διακοπές γενικού σκοπού που οφείλονται σε εσωτερικά και εξωτερικά γεγονότα. Οι τέσσερις 32-bit καταχωρητές που χρησιμοποιούνται στην IRQ είναι: IRQSTA, IRQSIG, IRQEN και IRQCLR.

- Ο καταχωρητής IRQSTA είναι ένας read-only register, ο οποίος μας παρέχει κάθε στιγμή την προσωρινή κατάσταση της πηγής που προκάλεσε το interrupt. Συνεπώς, όταν

έχει την τιμή 1, η πηγή ουσιαστικά δημιουργεί μια ενεργή αίτηση διακοπής στον μικροεπεξεργαστή ARM7TDMI.

- Ο καταχωρητής IRQSIG αναπαριστά την κατάσταση της κάθε πηγής interrupt IRQ. Εάν ένα περιφερειακό δημιουργήσει σήμα IRQ, ενεργοποιείται (παίρνει την τιμή 1) το αντίστοιχο bit του IRQSIG, διαφορετικά το αντίστοιχο bit έχει την τιμή 0.
- Ο καταχωρητής IRQEN παρέχει την τιμή της προσωρινά ενεργοποιημένης μάσκας διακοπών. Όταν κάποιο bit έχει την τιμή 1, αυτό σημαίνει ότι η αντίστοιχη πηγή αιτήματος έχει ενεργοποιηθεί ώστε να δημιουργήσει διακοπή IRQ. Όταν κάποιο bit έχει αντίστοιχα την τιμή 0, τότε η πηγή αιτήματος διακοπής είναι απενεργοποιημένη ή masked. Προκειμένου να απενεργοποιήσουμε μια ήδη ενεργοποιημένη πηγή interrupt, ο χρήστης πρέπει να ενεργοποιήσει το αντίστοιχο bit στον καταχωρητή IRQCLR.
- Ο καταχωρητής IRQCLR όπως προαναφέραμε, «καθαρίζει» τον καταχωρητή IRQEN έτσι ώστε να κάνει «mask» όπως αλλιώς λέμε την πηγή αιτήματος διακοπής.

Σε αυτό το σημείο πρέπει να τονίσουμε ότι προκειμένου να απενεργοποιήσουμε μια πηγή αιτήματος διακοπής απαιτείται να τοποθετήσουμε στο αντίστοιχο bit του IRQCLR την τιμή 1. «Καθαρίζοντας» τον καταχωρητή IRQEN, δεν συνεπάγεται την απενεργοποίηση της διακοπής.

Όσον αφορά τις διακοπές FIQ (fast interrupt request), αυτές λαμβάνουν χώρα προκειμένου να εξυπηρετήσουν την μεταφορά δεδομένων ή την επικοινωνία μεταξύ εργασιών των καναλιών με χαμηλή καθυστέρηση. Η διακοπή FIQ είναι πανομοιότυπη με την IRQ με την μόνη διαφορά ότι η πρώτη παρέχει διακοπή υψηλότερης προτεραιότητας (second-level interrupt, highest priority). Έτσι λοιπόν οι αντίστοιχοι τέσσερις 32-bit καταχωρητές της FIQ είναι οι : FIQSTA, FIQSIG, FIQEN, FIQCLR και FIQSTA.

Προσοχή και εδώ ισχύει ότι προκειμένου να απενεργοποιήσουμε μια πηγή αιτήματος διακοπής, απαιτείται να τοποθετήσουμε στο αντίστοιχο bit του FIQCLR την τιμή 1. «Καθαρίζοντας» τον καταχωρητή FIQEN, δεν συνεπάγεται την απενεργοποίηση της διακοπής.

6.7 Timers

Ο μικροελεγκτής ADuC702x διαθέτει 4 χρονιστές γενικού σκοπού : Timer0, Timer1, Timer2 ή wake-up timer και Timer3 ή watchdog timer [18]. Οι timers αυτοί λειτουργούν είτε σε free-running mode ή σε periodic mode. Στην πρώτη περίπτωση ο timer μετράει αντίστροφα από την αρχική τιμή που του έχουμε ορίσει ώσπου να φτάσει στο 0 και ξεκινάει ξανά από την μηδενική τιμή ή το αντίστροφο (μετράει από το 0 μέχρι την τιμή που του έχουμε ορίσει και στην συνέχεια ξεκινάει μετρώντας από την τιμή αυτή ως το 0). Κατά τον περιοδικό τρόπο λειτουργίας (periodic mode), ο timer ξεκινάει την αντίστροφη μέτρηση (ή μέτρηση προς τα πάνω) και όταν φτάσει στο 0 (ή στην maximum τιμή αντίστοιχα), ξαναφορτώνει την τιμή που του είχαμε ορίσει εξ αρχής στον load register. Στην normal mode, δημιουργείται ένα interrupt όταν ο μετρητής μηδενιστεί όταν έχει τεθεί σε αντίστροφη μέτρηση ή πάρει την maximum τιμή του όταν έχει τεθεί σε αύξουσα μέτρηση. Ένα IRQ «καθαρίζει» γράφοντας οποιαδήποτε τιμή στον αντίστοιχο καταχωρητή TxCLRl.

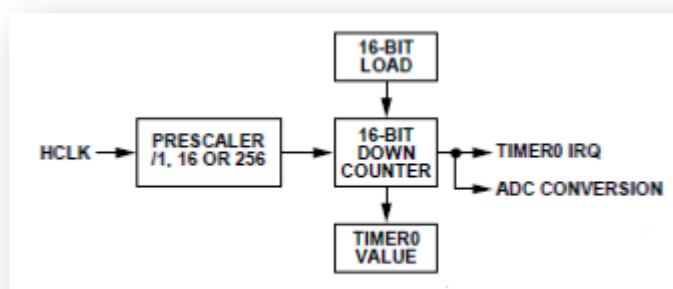
Το χρονικό διάστημα που μεσολαβεί μέχρι ο TIMER να μηδενιστεί ή αντίστοιχα να φτάσει στη μέγιστη τιμή υπολογίζεται ως εξής:

$$Interval = \frac{(TxD) \times Prescaler}{Source\ Clock}$$

όπου (TxD) η τιμή που βάζουμε στον TIMER.

Πρέπει να δώσουμε ιδιαίτερη προσοχή στο ότι το σήμα διακοπής μπορεί να χρειάζεται περισσότερο χρόνο να μηδενιστεί από τον χρόνο εκτέλεσης του κώδικα της ρουτίνας εξυπηρέτησης. Έτσι, πρέπει να επιβεβαιώσουμε πριν εγκαταλείψουμε την συνάρτηση ότι το σήμα διακοπής έχει καθαριστεί.

Όσον αφορά τον Timer0, πρόκειται για έναν 16-bit μετρητή γενικού σκοπού, αντίστροφης μέτρησης (count down) με προγραμματιζόμενο prescaler [18]. Συνεπώς, η συχνότητα του ρολογιού (HCLK-Core Clock Frequency) μπορεί να διαιρεθεί με 1, 16 ή 256.

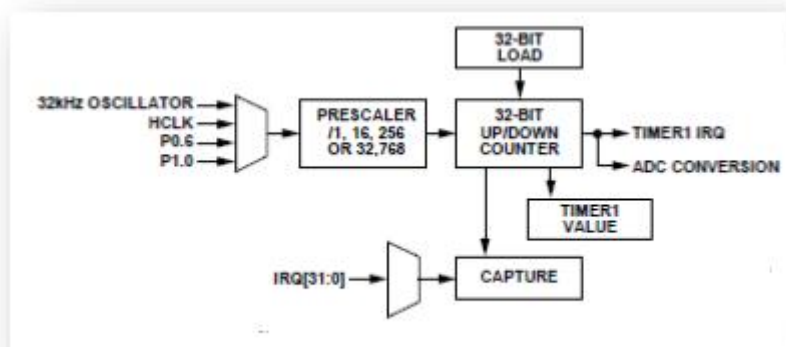


Σχήμα 6.6 Timer0 block diagram

Οι καταχωρητές του Timer0 είναι οι εξής (MMR): TOLD, TOVAL, TOCON και T0CLR1.

- Ο TOLD είναι ένας 16-bit load register, ο οποίος καθορίζει την συχνότητα του ρολογιού.
- Ο TOVAL είναι ένας 16-bit read-only register που αναπαριστά την προσωρινή κατάσταση του μετρητή.
- Με τον TOCON καθορίζουμε τον prescaler του Core Clock, τον τρόπο λειτουργίας του μετρητή (free-running ή periodic mode) και ενεργοποιούμε/απενεργοποιούμε τον μετρητή.
- Γράφοντας οποιαδήποτε τιμή στον 8-bit καταχωρητή T0CLR1, απενεργοποιούμε τα ενεργά interrupt του Timer0.

Ο Timer1, είναι ένας 32-bit μετρητής γενικού σκοπού [18]. Μπορεί να λειτουργήσει είτε σε φθίνουσα μέτρηση είτε σε αύξουσα (count down ή count up αντίστοιχα). Το source του Timer1 μπορεί να είναι ένας εξωτερικός κρύσταλλος 32kHz, η συχνότητα του ρολογιού ή και ένα εξωτερικό GPIO (P1.0 ή P0.6). Η μέγιστη συχνότητα του ρολογιού εισόδου είναι 44MHz και με prescaler μπορεί να διαιρεθεί με τους παράγοντες : 1, 16, 256 ή 32.768.

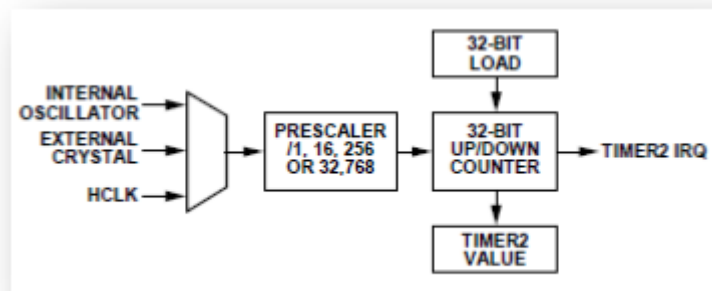


Σχήμα 6.7 Timer1 block diagram

Ο Timer1 διαθέτει τους εξής πέντε καταχωρητές (MMR): T1LD, T1VAL, T1CON, T1CLRI και T1CAP.

- Ο T1LD είναι ένας 32-bit load register, ο οποίος καθορίζει την συχνότητα του ρολογιού.
- Ο T1VAL είναι ένας 32-bit read-only register που αναπαριστά την προσωρινή κατάσταση του μετρητή.
- Με τον T1CON καθορίζουμε τον prescaler του Source Clock (1, 16, 256 ή 32.768) , αν ο Timer1 θα έχει τιμή δυαδική ή θα έχει προγραμματιστεί ώστε να μετράει σε Hr: min: sec (23 hours to 0 ή 255 hours to 0), τον τρόπο λειτουργίας του μετρητή (free-running ή periodic mode). Με τον καταχωρητή αυτόν, καθορίζουμε επίσης το source clock καθώς και την επιλογή του event.
- Γράφοντας οποιαδήποτε τιμή στον 8-bit καταχωρητή T1CLRI, απενεργοποιούμε τα ενεργά interrupt του Timer1.
- Ο T1CAP είναι ένας 32-bit καταχωρητής. Κρατάει την τιμή του T1VAL όταν συμβεί κάποιο event, το οποίο όμως event θα πρέπει να έχει καθοριστεί στον T1CON.

Ο Timer2 ή αλλιώς wake-up timer είναι ένας 32-bit μετρητής φθίνουσας ή αύξουσας μέτρησης, με προγραμματιζόμενο prescaler [18]. Η πηγή μπορεί να είναι ένας εξωτερικός κρύσταλλος 32kHz, η συχνότητα του ρολογιού ή ένας εσωτερικός ταλαντωτής 32kHz. Και εδώ το source clock μπορεί να διαιρεθεί με 1, 16, 256 ή 32.768.

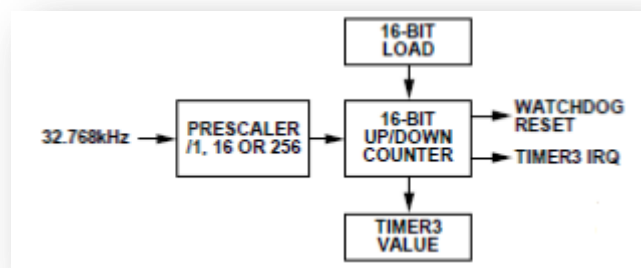


Σχήμα 6.8 Timer2 block diagram

Ο Timer2 διαθέτει τους εξής τέσσερις καταχωρητές (MMR): T2LD, T2VAL, T2CON και T2CLR1.

- Ο T2LD είναι ένας 32-bit load register, ο οποίος καθορίζει την συχνότητα του ρολογιού.
- Ο T2VAL είναι ένας 32-bit read-only register που αναπαριστά την προσωρινή κατάσταση του μετρητή.
- Με τον T2CON καθορίζουμε τον prescaler του Source Clock (1, 16, 256 ή 32.768) , αν ο Timer2 θα έχει τιμή δυαδική ή θα έχει προγραμματιστεί ώστε να μετράει σε Hr: min: sec (23 hours to 0 ή 255 hours to 0), τον τρόπο λειτουργίας του μετρητή (free-running ή periodic mode). Με τον καταχωρητή αυτόν, καθορίζουμε επίσης το source clock (external crystal, internal oscillator ή core clock).
- Γράφοντας οποιαδήποτε τιμή στον 8-bit καταχωρητή T2CLR1, απενεργοποιούμε τα ενεργά interrupt του Timer2.

Τέλος, ο Timer3 ή όπως αλλιώς ονομάζεται watchdog timer (χρονιστής επιτήρησης), είναι ένας χρονιστής ειδικού σκοπού [18]. Μπορεί να χρησιμοποιηθεί είτε σε normal mode είτε σε watchdog mode. Το source clock του Timer3 είναι ο εσωτερικός ταλαντωτής 32 KHz. Εκτός από την λειτουργία του σε normal mode, όπου λειτουργεί σαν ένας κοινός timer, ο χρονιστής αυτός χρησιμοποιείται συνήθως σε watchdog mode για την αποφυγή της πιθανής κατάρρευσης του συστήματος (crash). Η λειτουργία του χρονιστή επιτήρησης έχει ως εξής: από την στιγμή που θα οπλιστεί λειτουργεί ένας εσωτερικός μετρητής σε κάποιον συγκεκριμένο ρυθμό. Αν το πρόγραμμα χρήσης δεν μηδενίσει ή επαναθέσει τον μετρητή αυτό, τότε κάποια στιγμή θα επέλθει μηδενισμός του παραπάνω μετρητή και θα επανατοποθετηθεί ο μικροελεγκτής (reset). Έτσι όταν χρησιμοποιείται ο χρονιστής επιτήρησης, ο κώδικας του προγράμματος οφείλει να περιλαμβάνει εντολές ανανέωσης (reload) του watchdog timer, πληροφορώντας με αυτόν τον τρόπο ότι το σύστημα εργάζεται και δεν έχει καταρρεύσει. Η λογική αυτής της τεχνικής ελέγχου στηρίζεται στην υπόθεση ότι αν το πρόγραμμα χρήσης δεν ξαναφορτώσει τον Timer3, αυτό πιθανότατα σημαίνει ότι το πρόγραμμα έχει αποτύχει σε κάποια προσπάθειά του είτε λόγω πιθανής κατάρρευσης , είτε γενικότερα εξαιτίας κάποιας απρόβλεπτης συμπεριφοράς, οπότε είναι προτιμότερο να εκκινήσει διαδικασία επανατοποθέτησης.



Σχήμα 6.9 Timer3 block diagram

Και εδώ, έχουμε 4 καταχωρητές (MMR): T3LD, T3VAL, T3CON και T3CLR1.

- Ο T3LD είναι ένας 16-bit load register, ο οποίος καθορίζει την συχνότητα του ρολογιού.
- Ο T3VAL είναι ένας 16-bit read-only register που αναπαριστά την προσωρινή κατάσταση του μετρητή.
- Με τον T3CON καθορίζουμε τον prescaler του Source Clock (1, 16, 256 ή 32.768), τον τρόπο λειτουργίας του μετρητή (count-up ή count-down) και την ενεργοποίηση/απενεργοποίηση του Timer3.
- Γράφοντας οποιαδήποτε τιμή στον 8-bit καταχωρητή T3CLR1, απενεργοποιούμε τα ενεργά interrupt του Timer3 σε normal mode ή ανανεώνουμε τον χρόνο για την watchdog mode. Πρέπει να τονίσουμε ότι ο χρήστης οφείλει να ανανεώσει τον Timer3 ώστε να ανανεωθεί η περίοδος timeout period.

6.8 General Purpose Input-Output ports (GPIO)

Ο ADuC702x έχει 40 pin εισόδου/εξόδου γενικού σκοπού. Αυτά τα I/O pins υποστηρίζουν μια μέγιστη τάση εισόδου στα 5V. Γενικά, οι θύρες GPIOs έχουν πολλαπλές λειτουργίες με τη default λειτουργία να είναι αυτή της κατάστασης GPIO [18].

Όλες οι πύλες GPIO έχουν ένα εσωτερικό pull up αντιστάτη (περίπου των 100 kΩ) και η ικανότητα οδήγησης είναι της τάξης των 1.6 mA. Χρησιμοποιώντας τους καταχωρητές GPxPAR είναι δυνατό να ενεργοποιήσουμε ή να απενεργοποιήσουμε αυτούς τους αντιστάτες "pull-up" για τα ακόλουθα pins: P0.0, P0.4, P0.5, P0.6, P0.7 και τα 8 pins της θύρας 1.

Οι 40 GPIOs θύρες οργανώνονται σε 5 ομάδες, Port 0 μέχρι Port 4 (Port x). Κάθε port ελέγχεται από 4 ή 5 MMRs. Η τιμή της τάσης σε κάθε GPIO pin, μπορεί να διαβαστεί ανα πάσα κάθε στιγμή μέσω του καταχωρητή GPxDAT, ακόμα και όταν η θύρα ρυθμίζεται σε μια κατάσταση άλλη από την GPIO. Μέσω

των 32bit καταχωρητών GPxCON μπορεί να οριστεί ο τρόπος που λειτουργεί κάθε pin, συμβουλευόμενοι πάντα τους πίνακες που δίνονται από το manual του μικροελεγκτή, ενώ οι τιμές που αυτά παίρνουν τίθεται μέσω των GPxDAT και GPxSET καταχωρητών.

Table 78. GPIO Pin Function Descriptions

Port	Pin	Configuration			
		00	01	10	11
0	P0.0	GPIO	CMP	MS0	PLAI[7]
	P0.1	GPIO	PWM2 _H	$\overline{\text{BLE}}$	
	P0.2	GPIO	PWM2 _L	$\overline{\text{BHE}}$	
	P0.3	GPIO	TRST	A16	ADC _{BUSY}
	P0.4	GPIO/IRQ0	PWM _{TRIP}	MS1	PLAO[1]
	P0.5	GPIO/IRQ1	ADC _{BUSY}	MS2	PLAO[2]
	P0.6	GPIO/T1	MRST		PLAO[3]
	P0.7	GPIO	ECLK/XCLK ¹	SIN	PLAO[4]
1	P1.0	GPIO/T1	SIN	SCL0	PLAI[0]
	P1.1	GPIO	SOUT	SDA0	PLAI[1]
	P1.2	GPIO	RTS	SCL1	PLAI[2]
	P1.3	GPIO	CTS	SDA1	PLAI[3]
	P1.4	GPIO/IRQ2	RI	CLK	PLAI[4]
	P1.5	GPIO/IRQ3	DCD	MISO	PLAI[5]
	P1.6	GPIO	DSR	MOSI	PLAI[6]
	P1.7	GPIO	DTR	CSL	PLAO[0]
2	P2.0	GPIO	$\overline{\text{CONVSTART}}^2$	SOUT	PLAO[5]
	P2.1	GPIO	PWM0 _H	$\overline{\text{WS}}$	PLAO[6]
	P2.2	GPIO	PWM0 _L	$\overline{\text{RS}}$	PLAO[7]
	P2.3	GPIO		AE	
	P2.4	GPIO	PWM0 _H	MS0	
	P2.5	GPIO	PWM0 _L	MS1	
	P2.6	GPIO	PWM1 _H	MS2	
	P2.7	GPIO	PWM1 _L	MS3	
3	P3.0	GPIO	PWM0 _H	AD0	PLAI[8]
	P3.1	GPIO	PWM0 _L	AD1	PLAI[9]
	P3.2	GPIO	PWM1 _H	AD2	PLAI[10]
	P3.3	GPIO	PWM1 _L	AD3	PLAI[11]
	P3.4	GPIO	PWM2 _H	AD4	PLAI[12]
	P3.5	GPIO	PWM2 _L	AD5	PLAI[13]
	P3.6	GPIO	PWM _{TRIP}	AD6	PLAI[14]
	P3.7	GPIO	PWM _{SYNC}	AD7	PLAI[15]
4	P4.0	GPIO		AD8	PLAO[8]
	P4.1	GPIO		AD9	PLAO[9]
	P4.2	GPIO		AD10	PLAO[10]
	P4.3	GPIO		AD11	PLAO[11]
	P4.4	GPIO		AD12	PLAO[12]
	P4.5	GPIO		AD13	PLAO[13]
	P4.6	GPIO		AD14	PLAO[14]
	P4.7	GPIO		AD15	PLAO[15]

¹ When configured in Mode 1, P0.7 is ECLK by default, or core clock output. To configure it as a clock input, the MDCLK bits in PLLCON must be set to 11.

² The $\overline{\text{CONVSTART}}$ signal is active in all modes of P2.0.

Σχήμα 6.10 Λειτουργίες των pins κάθε GPIO θύρας

Άπαξ και γίνει η ρύθμιση των pin αυτών με τους παραπάνω καταχωρητές, τότε άλλοι καταχωρητές αναλαμβάνουν να επεξεργαστούν τα δεδομένα που εισέρχονται ή εξέρχονται μέσω αυτών των θυρών. Για κάθε λειτουργία υπάρχουν ξεχωριστοί καταχωρητές. Για παράδειγμα εάν μία θύρα οριστεί ως θύρα εισόδου και άλλη μία ως θύρα εξόδου ασύγχρονης σειριακής επικοινωνίας, τότε οι COM καταχωρητές της UART αναλαμβάνουν να επεξεργαστούν τα δεδομένα.

6.9 Παραδείγματα

Ακολουθούν παραδείγματα όπου γίνεται χρήση των περιφερειακών που περιγράφηκαν πιο πάνω.

Παράδειγμα 1

Παριστάνει 1024 συνεχόμενες μετατροπές αναλογικών τάσεων σε ψηφιακές. Η δειγματοληψία γίνεται στα κανάλια 0 και 1 εναλλάξ, τα αποτελέσματα αποθηκεύονται στην SRAM και στη συνέχεια αποστέλλονται μέσω της ασύγχρονης σειριακής μονάδας αποστολής δεδομένων (UART) με ρυθμό 9600bps.

```
#include <ADuC7026.h>

//δήλωση των συναρτήσεων κενού περιεχομένου που ορίζονται στο αρχείο
startup.s ως extern.

void Undef_Handler( void ) {}
void SWI_Handler(void) {}
void PAbt_Handler( void ) {}
void DAbt_Handler( void ) {}

void IRQ_Handler( void ) {}
void FIQ_Handler( void ) {}

void senddata(short); // συνάρτηση αποστολής δεδομένων μέσω UART

void ADCpoweron(int); /* συνάρτηση αναμονής μέχρι ο ADC να είναι fully
powered on */

char hex2ascii(char); /* συνάρτηση μετατροπής από binary σε hexadecimal
για απεικόνιση σε κάποιο display */

//κύριος κορμός του προγράμματος

int main (void) {
```

```

    unsigned short ADCDATA[1024]; /* ορισμός πίνακα τύπου short μη
    προσημασμένων δεδομένων μεγέθους 1024 στοιχείων */

    int i;

    ADCpoweron(20000); // power on ADC

    ADCCP = 0x00; //επιλογή θετικού καναλιού 0

    REFCON = 0x01; /* εσωτερική τάση αναφοράς στα 2.5V, 2.5V στο
    pin εξόδου Vref */

    GP0CON = 0x100000; /* προγραμματισμός του state του pin P0.5
    μέσω του αντίστοιχου register configuration, ADCbusy - P0.5, εκφράζουμε
    ότι ο ADC είναι απασχολημένος μέσω του αντίστοιχου bit */

    GP4DAT = 0x04000000; /* P4.2 κατασκευάζεται σαν έξοδος και τίθεται η
    τιμή εξόδου, ανάβοντας το led εξόδου */

    GP1CON = 0x011; /* κατασκευάζονται τα pins εξόδου P1.0 και P1.1 ως θύρες
    σειριακής εισόδου/εξόδου αντίστοιχα */

    //initialization UART at 9600bps

    COMCON0 = 0x80;
    COMDIV0 = 0x88;
    COMDIV1 = 0x00;
    COMCON0 = 0x07;

    ADCCON = 0x4E4; /* προγραμματισμός του ADC register  $f_{ADC}/2$ , acquisition
    time = 2 clocks => ADC Speed = 1MSPS,enable start conversion, enable
    ADCbusy pin, ADC power control, single-ended mode, continuous software
    conversion */

    while (1) // ατέρμονας βρόχος στους microcontrollers

    {

    // ξεκίνημα continuous conversion

        for (i=0; i <1024; i++)

        {

            while (!ADCSTA){} //όταν έχει τελειώσει η μετατροπή
            ADCSTA=1,άρα περίμενε όσο ADCSTA=0

```

```
        ADCDATA[i] = (ADCDAT>> 16); //ολίσθηση κατά 16 θέσεις
ώστε στο αντίστοιχο κελί του πίνακα να τοποθετηθεί το12-bit αποτέλεσμα
της μετατροπής που φυλάσσεται στον ADCDAT.
```

```
        if (ADCCP == 0) ADCCP= 1; // εναλλαγή καναλιών
```

```
        else ADCCP= 0;
```

```
    }
```

```
    GP4DAT ^= 0x00040000; //exclusive OR της τρέχουσας τιμής του GP4DAT
με την τιμή 00040000H, οδηγώντας σε flashing του P4.2
```

```
    for (i=0; i <1024; i++)        senddata (ADCDATA[i]);    // αποστολή
δεδομένων
```

```
    GP4DAT ^= 0x00040000; // flashing P4.2
```

```
    }
```

```
}
```

```
void senddata(short to_send)
```

```
{
```

```
    while(!(0x020==(COMSTA0 & 0x020))){}
```

```
        COMTX = 0x0A;        // output LF
```

```
    while(!(0x020==(COMSTA0 & 0x020))){}
```

```
        COMTX = 0x0D;        // output CR
```

```
    while(!(0x020==(COMSTA0 & 0x020))){}
```

```
        COMTX = hex2ascii ((to_send >> 8) & 0x0F);
```

```
    while(!(0x020==(COMSTA0 & 0x020))){}
```

```
        COMTX = hex2ascii ((to_send >> 4) & 0x0F);
```

```
    while(!(0x020==(COMSTA0 & 0x020))){}
```

```
        COMTX = hex2ascii (to_send & 0x0F);
```

```
}
```

```

char hex2ascii(char toconv)

{

    if (toconv<0x0A)

    {
        toconv += 0x30;
    }

    else

    {
        toconv += 0x37;
    }

    return (toconv);

}

void ADCpoweron (int time)

{

    ADCCON = 0x20;    // power-on the ADC
    while (time >=0)    // wait for ADC to be fully powered on
        time--;

}

```

Παράδειγμα 2

Παριστάνεται μια analog-to-digital conversion κάθε 100μs χρησιμοποιώντας τον timer 0, κάνοντας δειγματοληψία στα κανάλια 0 και 1 αντίστοιχα. Μετά το πέρας του μετρητή, συμβαίνει interrupt και η ρουτίνα εξυπηρέτησης αναλαμβάνει την αποστολή των δεδομένων μέσω της UART με ρυθμό 9600bps.

```

#include <ADuC7026.h> /*definitions για τον microcontroller δήλωση
συναρτήσεων κενού περιεχομένου*/

void Undef_Handler( void ) {}
void SWI_Handler(void) {}
void PAbt_Handler( void ) {}
void DAbt_Handler( void ) {}
void FIQ_Handler( void ) {}

```

```

void IRQ_Handler( void ) ; //ρουτίνα εξυπηρέτησης διακοπής

void senddata(short to_send); //αποστολή χαρακτήρων μέσω της UART
void ADCpoweron(int);
char hex2ascii(char toconv);
void delay(int);

int main (void) {

//ADC configuration

ADCpoweron(20000); // fully power on ADC

ADCCP = 0x00; //επιλογή θετικού καναλιού 0

ADCCON = 0x4E2; /* όταν ο timer0 μηδενιστεί γίνεται sampling στο κανάλι
και ξεκινά η ADC μετατροπή*/

/*ADC Configuration:  $f_{ADC}/2$ , acq. time = 2 clocks => ADC Speed
=1MSPS, τίθεται ο timer 0 ως γεγονός για την analog to digital μετατροπή,
single-ended mode, ADC normal mode, ενεργοποίηση του pin ADCbusy*/

REFCON = 0x01; /* σύνδεση της εσωτερικής τάσης αναφοράς 2.5V στο VREF
pin*/

/*κατασκευάζονται τα pins P1.0, P1.1 ως σειριακές θύρες εισόδου/εξόδου
αντίστοιχα για την αποστολή χαρακτήρων μέσω UART*/

GP1CON = 0x011;

//Initialization UART @9600 bps

COMCON0 = 0x80;

COMDIV0 =0x88;

COMDIV1 = 0x00;

COMCON0=0x07;

GP0CON = 0x10100000; /* enable ECLK output on P0.7, and ADCbusy on P0.5
*/

IRQEN = 0x80; /* τοποθετούμε μάσκα διακοπών, επιτρέποντας μόνο διακοπές
για τον timer 0*/

// configuration timer0

```

```

T0LD = 0x1000; //4096/40.96MHz = 100μ s για count- down μέτρηση

T0CON = 0xC0; // core clock, periodic mode, enable bit για timer 0

GP4DAT = 0x04000000; /* θέτει το pin P4.2 σαν έξοδο, και τοποθέτηση στα
δεδομένα εξόδου της τιμής 0, ανάβοντας το led εξόδου (ανάβει με 0)*/

while(1){} /* ατέρμων βρόχος από τον οποίο βγαίνει σε περίπτωση interrupt
και η ροή του προγράμματος μεταφέρεται στην ρουτίνα εξυπηρέτησης*/

return 0 ;

}

void IRQ_Handler( void )    {

GP4DAT ^= 0x00040000; /*γίνεται bitwise XOR της τρέχουσας κατάστασης του
data register GP4DAT και της 00040000hex προκαλώντας αλλαγή στην τιμή του
Pin 4.2 κάνοντας έτσι flashing στο led εξόδου*/

ADCCP ^= 1; // εναλλαγή καναλιών 0 και 1

senddata (ADCDAT >> 16); // αποστολή δεδομένων τα οποία είναι ολισθημένα
κατά 16 θέσεις ώστε να αποσταλεί το 12-bit αποτέλεσμα που φυλάσσεται στον
ADCDAT.

return ;
}

void senddata(short to_send) {

while(!(0x020==(COMSTA0 & 0x020))){}
    COMTX = 0x0A; // output LF
while(!(0x020==(COMSTA0 & 0x020))){}
    COMTX = 0x0D; //output CR
while(!(0x020==(COMSTA0 & 0x020))){}
    COMTX = hex2ascii ((to_send >> 8) & 0x0F);
while(!(0x020==(COMSTA0 & 0x020))){}
    COMTX= hex2ascii ((to_send >> 4) & 0x0F);
}

```

```

while(!(0x020==(COMSTA0 & 0x020))){}
    COMTX = hex2ascii (to_send & 0x0F);

}

char hex2ascii(char toconv)
{
    if (toconv<0x0A)
    {
        toconv += 0x30;
    }
    else
    {
        toconv += 0x37;
    }

    return (toconv);
}

void ADCpoweron (int time) {

    ADCCON = 0x20; //power-on the ADC

    while (time >=0) //αναμονή μέχρι ο ADC να είναι fully powered on
        time--;
}

```

Παράδειγμα 3

Απεικονίζεται στο κανάλι DAC1 μια ημιτονική κυματομορφή. Οι τιμές που παριστάνουν τα αντίστοιχα y της συνάρτησης $\sin x$ βρίσκονται αποθηκευμένες στον πίνακα. Τα ψηφιακά δεδομένα που χρειάζεται ο digital-to-analog converter ανανεώνονται με βάση το ρολόι του επεξεργαστή οπότε και θα έχουμε ένα νέο σημείο στην ημιτονική κυματομορφή που θέλουμε να αναπαραστήσουμε. Ο χρόνος στην ουσία που χρειάζεται ο επεξεργαστής για να εκτελέσει τις εντολές του βρόχου καθορίζει και το πότε θα εμφανιστεί ένα νέο σημείο.

```
#include<ADuC7026.h> /*definitions για τον microcontroller
```

```
//δήλωση των συναρτήσεων κενού περιεχομένου που ορίζονται στο αρχείο
startup.s ως extern
```

```
void Undef_Handler( void ) {}
void SWI_Handler(void) {}
void PAbt_Handler( void ) {}
void DAbt_Handler( void ) {}
void FIQ_Handler( void ) {}
void IRQ_Handler( void ) {} //ρουτίνα εξυπηρέτησης διακοπής
```

```
int main (void) {
```

```
/*πίνακας τιμών που εκφράζει συντεταγμένες σημείων της ημιτονικής
κυματομορφής και αποθηκεύεται στην flash memory, αφού η λέξη κλειδί const
δηλώνει ότι δεν μπορεί να τροποποιηθεί κατά την εκτέλεση του προγράμματος
και παίρνει αρχική τιμή τη στιγμή της δήλωσής της. */
```

```
const static unsigned short TableS[64] = {
```

```
0x07FF, 0x08C8, 0x098E, 0x0A51, 0x0B0F, 0x0BC4, 0x0C71, 0x0D12,0x0DA7,
0x0E2E, 0x0EA5, 0x0F0D, 0x0F63, 0x0FA6, 0x0FD7, 0x0FF5, 0x0FFF, 0x0FF5,
0x0FD7, 0x0FA6, 0x0F63, 0x0F0D, 0x0EA5, 0x0E2E, 0x0DA7, 0x0D12, 0x0C71,
0x0BC4, 0x0B0F, 0x0A51, 0x098E, 0x08C8, 0x07FF, 0x0736, 0x0670, 0x05AD,
0x04EF, 0x043A, 0x038D, 0x02EC, 0x0257, 0x01D0, 0x0159, 0x00F1, 0x009B,
0x0058, 0x0027, 0x0009, 0x0000, 0x0009, 0x0027, 0x0058, 0x009B, 0x00F1,
0x0159, 0x01D0, 0x0257, 0x02EC, 0x038D, 0x043A, 0x04EF, 0x05AD, 0x0670,
0x0736
```

```
};
```

```
int i = 0;
```

```
// προγραμματισμός και κατασκευή του DAC control register
```

```
DAC1CON= 0x13; /* εύρος 0-AVdd, κανονική λειτουργία του DAC
περιφερειακού, ανανέωση με βάση το ρολόι του πυρήνα*/
```

```
DAC1DAT = 0x08000000; /* στα bits 12-23 φυλάσσεται το 12-bit ψηφιακό σήμα
που θα αποτελέσει είσοδο στον DAC */
```

```
while(1){ // ατέρμονας βρόχος για τον microcontroller
```

```

DAC1DAT= ( TableS[i] << 16); /* ολίσθηση της ψηφιακής τιμής κατά
16 θέσεις ώστε η τιμή του TableS[i] να τοποθετηθεί στα bits 12-23 του
data register DAC1DAT */

    i++;

    i &= 0x03f; //ελέγχουμε αν είμαστε εντός των 64 θέσεων του
πίνακα

}

return (0);

}

```

Παράδειγμα 4

Χρησιμοποιώντας τον timer1 επιτυγχάνουμε διακοπές κάθε 50ms, οπότε και εναλλάσσεται η έξοδος της θύρας P4.2 όπου συνδέεται το led εξόδου προκαλώντας flashing.

```

#include<ADuC7026.h> //definitions για τον microcontroller

void Undef_Handler( void ) {} //συναρτήσεις κενού περιεχομένου

void SWI_Handler(void) {}
void PAbt_Handler( void ) {}
void DAbt_Handler( void ) {}
void FIQ_Handler( void ) {}
void IRQ_Handler( void ) ; //ρουτίνα εξυπηρέτησης διακοπής

int main (void) { //η εκτέλεση αρχίζει εδώ
T1LD = 0x20000; // ορίζουμε τη χρονική καθυστέρηση του timer1

T1CON = 0xC4; /*προγραμματισμός του control register του
timer1,sourceCLK/16, binary, περιοδικός τρόπος λειτουργίας*/

IRQEN = 0x08; // η μάσκα διακοπών επιτρέπει μόνο IRQ Timer1

GP4DAT = 0x04000000; /* θέτει το pin P4.2 σαν έξοδο, και τοποθέτηση στα
δεδομένα εξόδου της τιμής 0, ανάβοντας το led εξόδου (ανάβει με 0) */

while (1) /* ατέρμων βρόχος από τον οποίο βγαίνει σε περίπτωση interrupt
και η ροή του προγράμματος μεταφέρεται στην ρουτίνα εξυπηρέτησης*/

```

```

    {}
}

void IRQ_Handler(void) //συνάρτηση για την εξυπηρέτηση της διακοπής

{
    if ((IRQSTA & 0x08) != 0) /* γίνεται δυαδική AND μεταξύ των
bits του interrupt status register και της τιμής 0x08hex .Το αποτέλεσμα
γίνεται μονάδα μόνο σε περίπτωση Timer1 IRQ*/
    {

        GP4DAT ^= 0x00040000; /*γίνεται bitwise XOR της
τρέχουσας κατάστασης του data register GP4DAT και της 00040000hex
προκαλώντας αλλαγή στην τιμή του Pin 4.2 κάνοντας έτσι flashing στο led
εξόδου*/

        T1CLR1 = 0; /* καθαρίζουμε το σήμα διακοπής του Timer1
γράφοντας οποιαδήποτε τιμή στον T1CLR1*/
    }

return ;
}

```

Παράδειγμα 5

Το παρακάτω παράδειγμα χρησιμοποιεί δύο IRQs και μία FIQ. Ο timer1 προκαλεί μία IRQ που συμπληρώνει τη P4.2. Η εξωτερική διακοπή (XIRQ0) προκαλεί μια IRQ που επίσης κάνει flashing στο led που είναι συνδεδεμένο στη P4.2. Ο ADC προκαλεί μια FIQ που κάνει output το αποτέλεσμα του ADC0 στο κανάλι DAC1.

```

#include <ADuC7026.h> //definitions για τον microcontroller

//δήλωση των συναρτήσεων κενού περιεχομένου που ορίζονται στο αρχείο
startup ως extern.

void Undef_Handler( void ) {}
void SWI_Handler(void) {}
void PAbt_Handler( void ) {}
void DAbt_Handler( void ) {}
void FIQ_Handler( void ) ;
void IRQ_Handler( void ) ; //ρουτίνα εξυπηρέτησης διακοπής

```

```

void ADCpoweron(int); //συνάρτηση αρχικοποίησης

void delay(int); //συνάρτηση χρονοκαθυστέρησης

void DemoConfig(void); //συνάρτηση αρχικοποίησης περιφερειακών

long ADCconvert(void); //συνάρτηση μετατροπής analog-to-digital

int main (void) { //εδώ αρχίζει η εκτέλεση του προγράμματος

DemoConfig(); //set up ADC, DAC, & reference

GP0CON = 0x00100000; //ενεργοποίηση ADCbusy στο Pin 0.5

GP4DAT = 0x04000000; // κατασκευάζουμε το pin P 4.2 σαν έξοδο

GP3DAT= 0xff000000; /* κατασκευή της θύρας 3 ως έξοδος

//Timer 1 configuration

T1LD = 0x20000; /* φορτώνουμε στον counter register τη
χρονοκαθυστέρηση*/

T1CON = 0xC4; /* προγραμματίζουμε τον configuration register του timer1:
Enabled, Periodic, Binary and CLK/16 */

IRQEN = 0x00008000+0x00000008; /*φτιάχνουμε τη μάσκα διακοπών για timer1
και external interrupt αντίστοιχα*/

FIQEN = x00000080; //φτιάχνουμε τη μάσκα διακοπών fast interrupt

while(1) { //ατέρμων βρόχος. Σε περίπτωση IRQ ή FIQ, εκτελείται η
ρουτίνα διακοπής και μετά ο έλεγχος του προγράμματος μεταφέρεται στην
αμέσως επόμενη εντολή από το σημείο διακοπής.

GP3DAT ^= 0x00ff0000; // συμπληρώνει τη θύρα 3

ADCCON = 0x6E3; /*προγραμματίζουμε τον control register του ADC
περιφερειακού - single conversion */

delay(0x2000); /*χρονική καθυστέρηση για να είναι ορατό το
blinking στη θύρα P 3 */
}

```

```

return 0 ;

}

void delay (int length)

{
    while (length >=0) length--;
}

void DemoConfig(void) { //ADC, DAC and Reference configuration

    ADCpoweron(20000); //power on ADC

    ADCCP = 0x00; // conversion on ADC0

    DAC1 CON = 0x13; //AGND- A VDD range

    REFCON = x01; /*σύνδεση εσωτερικής τάσης αναφοράς 2.5V στο Vref
pin.*/

return;

}

void ADCpoweron(int time) {

    ADCCON = 0x20; //power- on the ADC
    while (time >=0) //αναμονή μέχρι να είναι fully powered on
        time--;
}

void IRQ_Handler(void) {

    if ((IRQSTA & 0x00000008) != 0) { /* ελέγχουμε για το αν
πρόκειται για Timer1 IRQ*/

        T1CLRI = 0; // καθαρίζουμε το σήμα διακοπής Timer IRQ

        GP4DAT ^= 0x00040000; //αλλάζει τιμή το pin εξόδου P4.2
    }
}

```

```

        if ((IRQSTA & 0x00008000) != 0) { /*ελέγχουμε για το αν πρόκειται
για external IRQ(push- button)*/
            GP4DAT ^= 0x00040000; // Complement P4.2

            while(GP0DAT & 0x00010){} /*αναμονή μέχρι η τάση XIRQ
να βρεθεί ξανά σε χαμηλό επίπεδο για να μην έχουμε συνεχόμενα
interrupts*/
        }

return ;

}

void FIQ_Handler(void) { // συνάρτηση εξυπηρέτησης fast interrupt

        if ((FIQSTA & 0x00000080) != 0) { /*ελέγχουμε για το αν
πρόκειται για ADC FIQ*/

            DAC1 DAT = ADCDAT; /*κάνει Output ADC0 Value στο κανάλι
DAC1*/
        }

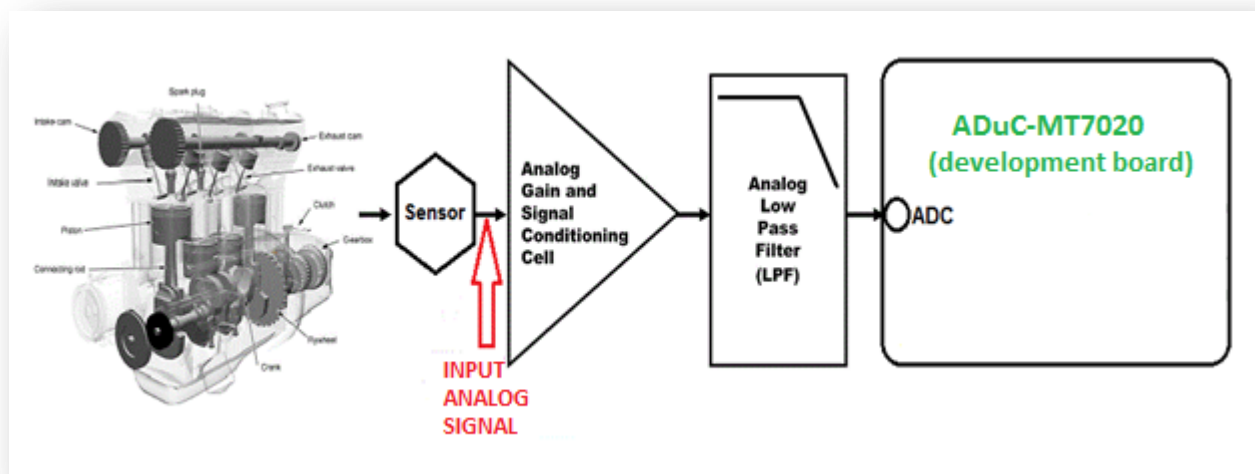
return ;

}

```

ΚΕΦΑΛΑΙΟ 7^ο

ΥΛΟΠΟΙΗΣΗ



Σχήμα 7.1 Τα διάφορα στάδια επεξεργασίας του σήματος που προκύπτει από τις δονήσεις των μηχανικών μερών μιας μηχανής αυτοκινήτου

Στο παραπάνω σχήμα απεικονίζονται επακριβώς τα διάφορα στάδια από τα οποία περνά το ηλεκτρικό σήμα που παράγεται στην έξοδο του πιεζοηλεκτρικού μας αισθητήρα και οφείλεται στις δονήσεις των μηχανικών μερών της μηχανής του αυτοκινήτου μέχρι αυτό να φτάσει στην είσοδο του Analog-to-Digital Converter του μικροελεγκτή μας. Έτσι λοιπόν μέχρι το σήμα να φτάσει στον μικροελεγκτή και να ψηφιοποιηθεί ακολουθεί μία διαδρομή στην οποία παρεμβάλλονται διαδοχικά:

- Πιεζοηλεκτρικός αισθητήρας
- Ενισχυτής σήματος
- Αναλογικό low-pass φίλτρο με συχνότητα αποκοπής 30KHz

Η συχνότητα αποκοπής του low-pass φίλτρου καθορίζεται από τη φύση της εφαρμογής μας και επειδή στη συγκεκριμένη εφαρμογή το σήμα μας προέρχεται από τις δονήσεις των μηχανικών εξαρτημάτων που απαρτίζουν μία μηχανή αυτοκινήτου αυτό μπορεί να περιέχει συχνότητες έως και 30 KHz. Χρησιμοποιώντας επομένως ένα φίλτρο με συχνότητα αποκοπής τα 30 KHz αναγκάζομαστε να επιλέξουμε σαν συχνότητα δειγματοληψίας του ADC μία τιμή τουλάχιστον διπλάσιά της (60 KHz) ώστε να ικανοποιείται το κριτήριο του Nyquist. Αφού γίνει δειγματοληψία του σήματος από τον ADC τα ψηφιοποιημένα πλέον δείγματά του επεξεργάζονται κατάλληλα ώστε να αναγνωρίσουμε τις συχνότητες που υπάρχουν στο σήμα και να τις αντιστοιχίσουμε στις διάφορες λειτουργίες των μηχανικών εξαρτημάτων και σε τυχόν σφάλματα σε αυτά. Στην προσπάθεια να φτιάξουμε ένα πρόγραμμα που να ανταποκρίνεται πλήρως στις ανάγκες για υψηλή ταχύτητα επεξεργασίας των δεδομένων που επιβάλλει η εφαρμογή μας αναγκαστήκαμε να δημιουργήσουμε και να απορρίψουμε δύο κώδικες μέχρι να φτάσουμε στην τρίτη και τελική υλοποίηση.

7.1 Υλοποίηση με απλό radix-2 DIT FFT

Κύριος σκοπός του προγράμματος μας είναι οι τιμές που ψηφιοποιούνται με τη βοήθεια του Analog-to-Digital Converter του μικροελεγκτή μας, αφού πολλαπλασιαστούν με ένα παράθυρο (Hamming) να υπολογίζεται το φάσμα συχνοτήτων τους μέσω του FFT και η ενέργεια σε κάθε συχνότητα. Οι τιμές αυτές αποτελούν δείγματα ενός σήματος ημιτόνου, το οποίο χρησιμοποιούμε για να αναπαραστήσουμε το σήμα που προκύπτει στην έξοδο του πιεζοηλεκτρικού μας αισθητήρα και προέρχεται από τις δονήσεις των μηχανικών μερών της μηχανής του αυτοκινήτου. Προφανώς η κυματομορφή ενός τέτοιου σήματος είναι πολύ πιο πολύπλοκη απλά στην προσπάθειά μας να γίνει πιο κατανοητή η διαδικασία επιλέξαμε ένα ημίτονο σαν σήμα εισόδου, χωρίς όμως αυτό να σημαίνει ότι ο κώδικάς μας δεν λειτουργεί ικανοποιητικά για οποιοδήποτε σήμα εισόδου. Αφού γίνει ο υπολογισμός της ενέργειας σε κάθε συχνότητα στη συνέχεια ακολουθούμε τη λογική της αναγνώρισης προτύπων. Αυτό δεν είναι τίποτα άλλο από την αναγνώριση (αν υπάρχουν) των βλαβών βάσει της διαδικασίας που περιγράφηκε σε προηγούμενο κεφάλαιο για την ανάλυση των ασφαμάτων σε περιστρεφόμενο μηχανικό εξοπλισμό.

Όπως εξηγήσαμε και πιο πάνω, η δόνηση κάθε εξαρτήματος που απαρτίζει έναν μηχανικό εξοπλισμό εκφράζεται με την παρουσία μιας συνιστώσας σε μια συγκεκριμένη τιμή συχνότητας του φάσματος, που αποτελεί κατά κάποιο τρόπο την ταυτότητά του. Έτσι γνωρίζοντας εκ των προτέρων τα μηχανικά μέρη της μηχανής μας και συνεπώς και που θα πρέπει να περιμένουμε την εμφάνιση συνιστωσών στο φάσμα συχνοτήτων και το πλάτος που θα πρέπει να έχουν αν δεν υπάρχει βλάβη, η παρουσία συνιστωσών σε διαφορετικά σημεία του φάσματος και η εμφάνιση τιμών πλάτους μεγαλύτερες των αναμενομένων υποδηλώνουν την παρουσία σφάλματος. Λαμβάνοντας υπ' όψη τις αλλαγές που προκαλεί κάθε σφάλμα στο φάσμα συχνοτήτων μπορούμε να το προσδιορίσουμε.

Αφού εξηγήσαμε σε γενικές γραμμές για ποιο λόγο και τι ακριβώς κάνει ο κώδικάς μας ας δούμε τώρα πως ακριβώς το πετυχαίνει αυτό μέσα από την ανάλυση των κύριων χαρακτηριστικών του.

- Η συχνότητα δειγματοληψίας καθορίζεται από την τιμή που βάζουμε στον timer, ο οποίος σε μια count-down μέτρηση κάθε φορά που μηδενίζεται, ο ADC κάνει μια analog-to-digital μετατροπή. Αυτό επιτυγχάνεται μέσω του καταχωρητή ADCCON, όπου προγραμματίζουμε τον ADC έτσι ώστε κάθε φορά που ο TIMER 0 στην περίπτωση μας μηδενίζεται να σηματοδοτείται μια καινούρια μετατροπή. Επίσης μέσω του καταχωρητή IRQEN ορίζουμε την διακοπή να την προκαλεί ο ADC, οπότε κάθε φορά που ο timer μηδενίζεται ο ADC προκαλεί μια διακοπή στο κύριο πρόγραμμα και εκτελεί τη μετατροπή.
- Θεωρούμε αρχικά 128 δείγματα. Τα δείγματα καθορίζουν το frequency resolution που θα έχουμε. Έτσι όσο μεγαλύτερος ο αριθμός των δειγμάτων τόσο λιγότερο θα απέχουν μεταξύ τους τα σημεία του άξονα συχνοτήτων και προφανώς τα αποτελέσματα που θα προκύπτουν θα ανταποκρίνονται πιο πολύ στην πραγματικότητα. Επίσης βασική προϋπόθεση του FFT είναι ο αριθμός των δειγμάτων να είναι δύναμη του 2 και το 128 ικανοποιεί αυτό το κριτήριο.
- Χρησιμοποιούμε έναν complex FFT, ο οποίος δέχεται σαν είσοδο μιγαδικούς αριθμούς, έναν πίνακα δηλαδή με το πραγματικό μέρος (REAL[]) και έναν με το φανταστικό (IMAG[]), αν και οι τιμές που επεξεργαζόμαστε είναι πραγματικοί αριθμοί. Γι' αυτό και στον κώδικα συμπληρώνουμε τον πίνακα IMAG[] με μηδενικά. Υπάρχουν αλγόριθμοι FFT που

επεξεργάζονται πραγματικές τιμές και στην προσπάθειά μας να αυξήσουμε την ταχύτητα επεξεργασίας των τιμών χρησιμοποιήσαμε και μία τέτοια υλοποίηση του FFT απλά η βελτίωση στην ταχύτητα δεν ήταν τόσο σημαντική ώστε να παραθέσουμε ξεχωριστά ένα πρόγραμμα που τον χρησιμοποιεί.

- Για την παραθυροποίηση των δεδομένων χρησιμοποιήσαμε το Hamming window, το οποίο εκφράζεται από την εξίσωση: $0,54-0,46\cos(2\pi x/N-1)$. Κάθε δείγμα που ψηφιοποιείται μέσω του ADC πολλαπλασιάζεται απευθείας με τον αντίστοιχο συντελεστή του παραθύρου.
- Χρησιμοποιείται ένας πίνακας 256 θέσεων για την αποθήκευση των αποτελεσμάτων που προκύπτουν από τον ADC. Η επεξεργασία των δεδομένων γίνεται ως εξής: όταν ψηφιοποιηθεί και το στοιχείο που βρίσκεται στη θέση 127 (128ο στοιχείο) ξεκινά η επεξεργασία των πρώτων αυτών 128 δειγμάτων από τον FFT και υπολογίζεται η ενέργειά τους. Παράλληλα για να μην χάνονται τιμές τα δείγματα που συλλέγονται σ' αυτό το χρονικό διάστημα γεμίζουν τις θέσεις 128-255 έτσι ώστε όταν συμπληρωθεί όλος ο πίνακας και έχει τελειώσει η επεξεργασία του πρώτου πακέτου να ξεκινήσει η επεξεργασία του δεύτερου. Με τη συμπλήρωση και των 256 θέσεων του πίνακα ο δείκτης μηδενίζεται και ο πίνακας συμπληρώνεται από την αρχή. Έτσι τα πρώτα 128 δείγματα των οποίων η επεξεργασία έτσι κι αλλιώς έχει τελειώσει αντικαθίστανται από τις νέες 128 τιμές που θα αποτελέσουν και το νέο πακέτο. Μ' αυτόν τον τρόπο συνεχίζεται η επεξεργασία κάθε πακέτου δειγμάτων που συλλέγεται. Στόχος της ταυτόχρονης αυτής επεξεργασίας και συλλογής των δειγμάτων είναι η ικανοποίηση των real-time απαιτήσεων της εφαρμογής μας.
- Με τη συνάρτηση startCpu γίνεται η αρχικοποίηση της CPU και το core clock προγραμματίζεται να λειτουργεί στη μέγιστη δυνατή συχνότητά του, τα 41,78 MHz.

Στο Παράρτημα 1 παρατίθεται η υλοποίηση η οποία συνοδεύεται από κάποια σχόλια για να γίνουν σαφέστερες οι εντολές που απαρτίζουν τον κώδικα.

Στην 1η υλοποίηση χρησιμοποιήθηκε για το πέρασμα από το πεδίο του χρόνου στο πεδίο της συχνότητας ένας απλός radix-2 DIT(decimation-in-time) FFT. Ο κύριος στόχος του κώδικά μας ήταν εξ αρχής να μπορεί να λειτουργεί σε real-time, να μην χάνει δηλαδή καθόλου τιμές ενώ επεξεργάζεται ένα συμπληρωμένο πακέτο δειγμάτων. Αυτό είναι που καθορίζει και το βαθμό λειτουργικότητάς του. Έτσι ενώ ο κώδικας που δημιουργήσαμε μπορεί να μην παρουσιάζει ούτε συντακτικά ούτε λογικά λάθη μπορεί όμως να μην ανταποκρίνεται στις real-time απαιτήσεις της εφαρμογής μας, γεγονός που τον χρήσει αυτόματα μη χρησιμοποιήσιμο. Η βασική προϋπόθεση που πρέπει να πληροί το πρόγραμμά μας είναι να λειτουργεί σύμφωνα με τον παραπάνω τρόπο στη συχνότητα δειγματοληψίας που καθορίζεται από τη φύση της εφαρμογής μας. Στη συγκεκριμένη εφαρμογή η συχνότητα με την οποία δειγματοληπτούμε δεν μπορεί να είναι μικρότερη από 60 KHz.

Έχοντας σαν συχνότητα δειγματοληψίας τα 60 KHz μπορούμε να υπολογίσουμε πόσο χρόνο χρειάζεται κάθε πακέτο δειγμάτων για να συλλεχθεί. Για κάθε δείγμα απαιτείται χρόνος $1/60000=16,7 \mu s$. Στην 1η υλοποίηση επεξεργαζόμαστε πακέτα των 128 δειγμάτων οπότε για τη συλλογή κάθε τέτοιου πακέτου απαιτείται χρόνος $128 \times 16,7=2,137 ms$. Εμείς εφόσον θέλουμε να επιτύχουμε παράλληλη επεξεργασία και συλλογή δειγμάτων εύκολα μπορούμε να συμπεράνουμε

ότι είναι απαραίτητο ο χρόνος επεξεργασίας του προηγούμενου πακέτου να είναι μικρότερος από το χρόνο συλλογής του επόμενου. Έτσι η επεξεργασία των 128 στην περίπτωση μας δειγμάτων από τον FFT και ο υπολογισμός της ενέργειας σε κάθε συχνότητα πρέπει να διαρκεί το πολύ 2,137 ms.

Ο απλός radix-2 DIT FFT που χρησιμοποιήσαμε, όπως προκύπτει εκτελώντας τον κώδικα στον simulator του Keil, χρειάζεται για να εκτελεστεί 73 ms, περίπου 35 φορές δηλαδή πιο αργά από την επιθυμητή ταχύτητα. Πιο συγκεκριμένα στον χρόνο που απαιτείται για τον υπολογισμό του FFT έχουν συλλεχθεί 35 νέα πακέτα δειγμάτων τα οποία δεν υπόκεινται επεξεργασία και χάνονται. Κάτι τέτοιο μας αναγκάζει για να μπορέσει να χρησιμοποιηθεί ο πιο πάνω κώδικας είτε να μειώσουμε τη συχνότητα δειγματοληψίας (συγκεκριμένα εδώ από 60KHz στο 1 KHz) είτε να απενεργοποιούμε τις διακοπές κάθε φορά που συλλέγεται ένα πακέτο 128 δειγμάτων. Στην πρώτη περίπτωση εφόσον το σήμα εισόδου περιλαμβάνει συχνότητες μέχρι και 30 KHz παραβαίνουμε το κριτήριο του Nyquist και άρα οδηγούμαστε σε λάθος αποτελέσματα ενώ στη δεύτερη μπορεί το κριτήριο να ικανοποιείται αλλά το πρόγραμμά σας δεν λειτουργεί σε real time. Λαμβάνοντας υπόψη τα παραπάνω γίνεται εύκολα αντιληπτό ότι ο πιο πάνω κώδικας δεν μπορεί να καλύψει τις ανάγκες της εφαρμογής μας και η χρήση του ενδείκνυται μόνο σε περιπτώσεις όπου γίνεται επεξεργασία σημάτων με συχνότητες μέχρι 500 Hz και επομένως καθίσταται αναγκαία η χρησιμοποίηση ενός άλλου πολύ πιο γρήγορου FFT.

7.2 Υλοποίηση με fixed-point FFT

Στην 2η υλοποίηση προτιμήθηκε η χρησιμοποίηση ενός fixed-point FFT αντί του απλού FFT που χρησιμοποιήθηκε στην πρώτη περίπτωση. Η κύρια διαφορά τους, η οποία και έχει άμεσο αντίκτυπο στο χρόνο επεξεργασίας των δεδομένων, έγκειται στο ότι ενώ ο πρώτος χρησιμοποιεί σαν είσοδο πίνακες τύπου float ο δεύτερος επεξεργάζεται δεδομένα τύπου int ή ακόμη και short. Με τον τρόπο αυτό χάνουμε κατά ένα μικρό ποσοστό σε ακρίβεια αλλά κερδίζουμε πάρα πολύ σε ταχύτητα. Επίσης όταν επεξεργαζόμαστε στοιχεία με τον απλό FFT πολλαπλασιάζουμε τα δεδομένα μας κανονικά με ημίτονα και συνημίτονα, ο υπολογισμός των οποίων ως γνωστόν είναι ιδιαίτερα χρονοβόρος. Αντίθετα ο fixed-point που χρησιμοποιούμε στην 2η υλοποίηση χρησιμοποιεί έναν πίνακα 1024 στοιχείων στον οποίο είναι αποθηκευμένες οι τιμές μιας περιόδου ενός ημιτόνου και οι οποίες είναι πολλαπλασιασμένες με την τιμή 32768 (1000000000000000 binary) και έχει κρατηθεί μόνο το ακέραιο μέρος τους. Γι' αυτό το λόγο όταν εκτελείται ο fixed-point FFT και οι τιμές που προκύπτουν από τη δειγματοληψία πολλαπλασιάζονται με μια τιμή ημιτόνου ή συνημιτόνου που υπάρχει στον πίνακα αυτόν καλείται μία συνάρτηση fix_mpy η οποία πολλαπλασιάζει τις δύο τιμές και στη συνέχεια εκτελεί shift στο αποτέλεσμα 15 θέσεις δεξιά διαιρώντας ουσιαστικά με την τιμή 32768 κρατώντας πάλι μόνο το ακέραιο μέρος του αποτελέσματος.

Οι τιμές που αποθηκεύονται στους πίνακες είναι ακέραιοι αριθμοί. Οι πίνακες δηλαδή που χρησιμοποιούνται από τον κώδικα είναι τύπου short ή int εκτός από τον πίνακα HamWin που περιέχει τους συντελεστές του παραθύρου ο οποίος είναι τύπου float. Κατά τον πολλαπλασιασμό κάθε δείγματος το οποίο είναι μια τιμή τύπου short με τον αντίστοιχο συντελεστή παραθύρου το οποίο είναι τύπου float προκύπτει ένα αποτέλεσμα τύπου short (ο ADCDATA είναι τύπου short) με την αυτόματη εξάλειψη του δεκαδικού μέρους. Με αυτόν τον τρόπο μπορεί να μειώνεται σε ένα μικρό βαθμό η ακρίβεια των αποτελεσμάτων αλλά αυξάνεται σημαντικά η ταχύτητα επεξεργασίας των δεδομένων.

Στο Παράρτημα 2 παραθέτουμε τον κώδικα ο οποίος είναι παρόμοιος με αυτόν της πρώτης υλοποίησης με τη διαφορά ότι χρησιμοποιείται ένας fixed-point FFT και ένα ημίτονο με ακέραιες τιμές από -32768 έως 32768 οι οποίες αντιπροσωπεύουν το -1.0 και το +1.0 αντίστοιχα.

Αφού κάναμε βελτιστοποίηση στον κώδικα μέσα από το Keil (optimization level 2) στη συνέχεια τον τρέξαμε πάλι χρησιμοποιώντας τον simulator του συγκεκριμένου περιβάλλοντος και διαπιστώσαμε ότι ο χρόνος που χρειάζεται ο fixed-point FFT για να επεξεργαστεί τα δεδομένα είναι 5,1ms και αν συμμεριστούμε ότι ο υπολογισμός στη συνέχεια της ενέργειας σε κάθε συχνότητα διαρκεί περίπου 0,2ms αυτό μας δίνει ένα συνολικό χρόνο επεξεργασίας 5,3ms. Όπως έχει ήδη επισημανθεί προκειμένου το πρόγραμμά μας να ανταποκρίνεται στις απαιτήσεις αυτής της εφαρμογής ο χρόνος αυτός δεν θα πρέπει να υπερβαίνει τα 2,137ms που είναι ο χρόνος που απαιτείται για να γίνει η δειγματοληψία ενός πακέτου 128 τιμών με συχνότητα δειγματοληψίας 60KHz. Ενώ λοιπόν καταφέραμε να αυξήσουμε σημαντικά την ταχύτητα επεξεργασίας των δεδομένων - από 73ms μειώθηκε σε μόλις 5,3ms - η βελτίωση αυτή δεν επαρκεί ώστε να μπορεί ο κώδικας που μόλις παραθέσαμε να χρησιμοποιηθεί στη συγκεκριμένη εφαρμογή. Προκύπτει επομένως για άλλη μια φορά η ανάγκη τροποποίησης του ώστε να μπορέσουμε να μειώσουμε κι άλλο το χρόνο επεξεργασίας των τιμών που ψηφιοποιούνται από τον ADC. Γνωρίζοντας όμως ότι ο fixed-point FFT αποτελεί την ταχύτερη υλοποίηση του συγκεκριμένου αλγορίθμου προκύπτει πλέον η ανάγκη εύρεσης κάποιου άλλου εργαλείου για το πέρασμα στο πεδίο της συχνότητας. Το εργαλείο αυτό δεν είναι άλλο από τον αλγόριθμο Goertzel, ο οποίος προβάλλει ως η μόνη και καλύτερη εναλλακτική λύση όταν ο FFT δεν μπορεί να καλύψει την ανάγκη για γρήγορη επεξεργασία δεδομένων που επιτάσσει μια εφαρμογή.

7.3 Υλοποίηση με αλγόριθμο Goertzel

Μετά και από τη χρησιμοποίηση του fixed-point FFT, η οποία δεν κατάφερε να μειώσει στον επιθυμητό βαθμό το χρόνο επεξεργασίας των δεδομένων η επιλογή πλέον του αλγορίθμου Goertzel για το πρόγραμμά μας ήταν κατά κάποιο τρόπο μονόδρομος. Ο αλγόριθμος Goertzel δεν μας δίνει ολόκληρο το φάσμα συχνοτήτων όπως γίνεται με τον FFT αλλά υπολογίζεται το πλάτος ή η ενέργεια σε συχνότητες οι οποίες είναι προκαθορισμένες. Όσον αφορά τη δική μας εφαρμογή εμείς έχοντας γνώση των εξαρτημάτων που απαρτίζουν τον μηχανικό μας εξοπλισμό είμαστε σε θέση να γνωρίζουμε σε ποιες συχνότητες εμφανίζονται συνιστώσες που έχουν να κάνουν με τη λειτουργία τους αλλά και σε ποιες συχνότητες εμφανίζονται συνιστώσες που οφείλονται αποκλειστικά σε σφάλματα. Έτσι μπορούμε μέσω του Goertzel να υπολογίσουμε την ενέργεια στις συχνότητες αυτές και να τις συγκρίνουμε με τις τιμές αναφοράς.

Εδώ να επισημάνουμε ότι πρώτα χρησιμοποιήθηκε αντί του FFT ο βελτιστοποιημένος Goertzel για την υλοποίηση του προγράμματος, ο οποίος έχει περιγραφεί αναλυτικά σε προηγούμενο κεφάλαιο. Ο συγκεκριμένος αλγόριθμος χρησιμοποιεί σαν είσοδο έναν συντελεστή τύπου float ($2\cos(2\pi f_{\text{target}} / f_s)$) για να καθοριστεί η συχνότητα στην οποία γίνεται ο υπολογισμός της ενέργειας. Για τον υπολογισμό όμως της ενέργειας σε μια τιμή συχνότητας απαιτείται η εκτέλεση N αριθμού εξισώσεων (το N συμβολίζει τον αριθμό των δειγμάτων που αποτελούν κάθε πακέτο) κάθε μία από τις οποίες περιλαμβάνει έναν πολλαπλασιασμό με τον προκαθορισμένο συντελεστή τύπου float ο οποίος αντιστοιχεί στη συγκεκριμένη τιμή συχνότητας. Αυτό σημαίνει ότι για να υπολογιστεί η ενέργεια σε μία μόνο συχνότητα απαιτούνται N τέτοιοι πολλαπλασιασμοί με μια float τιμή και αν υποθέσουμε ότι

k είναι ο αριθμός συχνοτήτων στις οποίες θέλουμε να υπολογίσουμε την ενέργεια χρειάζεται να εκτελεστούν $k \times N$ τέτοιοι πολλαπλασιασμοί με τους αντίστοιχους float συντελεστές. Όπως λοιπόν ήταν αναμενόμενο διαπιστώσαμε ότι ούτε μέσω του βελτιστοποιημένου Goertzel καταφέρνουμε να υπολογίσουμε τις ενέργειες σε έναν επαρκή αριθμό συχνοτήτων σε χρόνο μικρότερο απ' όσο απαιτεί η συλλογή ενός νέου πακέτου δειγμάτων. Άρα προέκυψε η ανάγκη να τροποποιήσουμε τον αλγόριθμο κατάλληλα ώστε να αυξήσουμε την ταχύτητα επεξεργασίας των δειγμάτων.

Το πρώτο που έπρεπε να γίνει ήταν να αντικαταστήσουμε τους συντελεστές τύπου float με άλλους οι τιμές των οποίων θα επεξεργάζονται πιο γρήγορα από τον μικροελεγκτή μας. Για το λόγο αυτό επιλέξαμε, επειδή αυτός ο συντελεστής παίρνει τιμές από -2 έως 2, να δημιουργήσουμε 17 τέτοιους συντελεστές οι οποίοι διαφέρουν μεταξύ τους κατά 0,25. Θεωρώντας λοιπόν πρώτο το -2, ο δεύτερος θα είναι ο -1,75, ο τρίτος ο -1,5 και μέχρι τον 17ο ο οποίος θα είναι το 2. Αυτό το κάναμε ώστε να μπορέσουμε να αξιοποιήσουμε την ταχύτητα με την οποία ένας επεξεργαστής, όπως είναι και αυτός του μικροελεγκτή μας, εκτελεί shift σε μια τιμή. Έτσι αν θέλουμε για παράδειγμα να πολλαπλασιάσουμε μια τιμή με το 2 ή το 0,5 αρκεί να κάνουμε shift στην τιμή αυτή μία θέση αριστερά ή δεξιά αντίστοιχα ενώ αν θέλουμε να πολλαπλασιάσουμε π.χ. με το 1,5 αρκεί να πολλαπλασιάσουμε πρώτα με το 2 (shift μία θέση αριστερά) και στη συνέχεια να αφαιρέσουμε το αποτέλεσμα του πολλαπλασιασμού της τιμής αυτής με το 0,5 (shift μία θέση δεξιά).

Επίσης στηριζόμενοι στο ότι ο αλγόριθμος Goertzel δεν απαιτεί τη συλλογή όλου του πακέτου δειγμάτων για να μπορεί να εκτελεστεί, αυτός καλείται κάθε φορά που παράγεται ένα ψηφιοποιημένο δείγμα στην έξοδο του ADC. Η τιμή του κάθε δείγματος που συλλέγεται χρησιμοποιείται στον υπολογισμό των 17 εξισώσεων που αντιστοιχούν στις 17 διαφορετικές τιμές συχνοτήτων στις οποίες θέλουμε να υπολογίσουμε την ενέργεια. Εδώ πρέπει να επισημανθεί ότι στον κώδικα που υλοποιήσαμε δεν υπολογίζουμε ακριβώς την ενέργεια σε κάθε συχνότητα αλλά την τετραγωνική της ρίζα κι αυτό γιατί θέλαμε να αποφύγουμε πιθανή υπερχείλιση στο αποτέλεσμα. Έτσι στην μεταβλητή eG προσθέτουμε κάθε φορά την τιμή v και όχι την v^2 . Επιπλέον αντί για 128 δείγματα που χρησιμοποιήσαμε στις προηγούμενες υλοποιήσεις στη συγκεκριμένη επιλέξαμε να χρησιμοποιήσουμε 64 αν και δεν ήταν απαραίτητο ο αριθμός αυτός να είναι δύναμη του 2 μιας και ο Goertzel σε αντίθεση με τον FFT δεν το επιβάλλει. Κάθε φορά που συλλέγεται και το 64ο δείγμα συγκρίνουμε τις τιμές των ενεργειών για τις 17 διαφορετικές τιμές συχνοτήτων με μια τιμή αναφοράς (threshold) και αν κάποια τιμή από αυτές την υπερβαίνει ανάβει ένα led σαν ένδειξη. Αυτό γίνεται με τη βοήθεια των θυρών GPIO (General Input-Output Ports). Στη συνέχεια μηδενίζουμε όλες τις μεταβλητές και εκτελούμε την διαδικασία από την αρχή, επεξεργαζόμενοι ουσιαστικά με αυτόν τον τρόπο πακέτα των 64 δειγμάτων. Η τιμή αναφοράς καθώς και οι τιμές των συχνοτήτων που καθορίζονται από τους 17 συντελεστές που ορίσαμε εξαρχής είναι καθαρά ενδεικτικές και χρησιμοποιούνται για να επιβεβαιώσουμε ότι ο κώδικάς μας λειτουργεί όπως πρέπει.

Στο Παράρτημα 3 παραθέτουμε τον κώδικα στον οποίο έχουμε συμπεριλάβει και κάποια σχόλια για να γίνει πιο κατανοητή η χρήση κάποιων μεταβλητών και εντολών.

Για να προσομοιώσουμε το σήμα που προκύπτει από τις δονήσεις των μηχανικών μερών της μηχανής χρησιμοποιήσαμε ένα ημίτονο με συχνότητα μία τιμή από τις target frequencies που ανιχνεύονται από τον αλγόριθμο Goertzel του προγράμματός μας. Αν και όπως έχει ήδη αναφερθεί ένα τέτοιο σήμα μπορεί να περιέχει συχνότητες μέχρι και 30 KHz εμείς επιλέξαμε σαν συχνότητα τα 6,9 KHz, χωρίς όμως αυτό να σημαίνει ότι δεν θα μπορούσαμε να επιλέξουμε μια μεγαλύτερη τιμή. Από τη στιγμή που έχουμε ορίσει μέσα στον κώδικα σαν συχνότητα δειγματοληψίας τα 60 KHz βάσει του θεωρήματος Nyquist μπορούμε να δειγματοληπτήσουμε επιτυχώς σήματα με συχνότητα έως και 30 KHz. Αν επιλέγαμε ένα ημίτονο με συχνότητα διαφορετική από τις τιμές συχνότητας που

ανιχνεύονται από τον αλγόριθμο Goertzel θα παρατηρούσαμε το φαινόμενο leakage μιας και θα εμφανίζονταν μεγάλες τιμές πλάτους στις target frequencies που είναι πιο κοντά στην τιμή συχνότητας που επιλέξαμε για το ημίτονό μας. Το Keil μvision4, που είναι το περιβάλλον ανάπτυξης στο οποίο δουλεύουμε, μας δίνει τη δυνατότητα καθώς τρέχουμε τον κώδικα της εφαρμογής μας να θέσουμε ως είσοδο σε διάφορα κανάλια συναρτήσεις που έχουμε υλοποιήσει. Έτσι λοιπόν δημιουργήσαμε ένα αρχείο μέσω της πλοήγησης debug →function editor→signal.ini., το οποίο αποτελεί ένα initial αρχείο όπως υποδηλώνει και η επέκταση του μέσα στο οποίο γράφουμε τον κώδικα που υλοποιεί μία ημιτονοειδή συνάρτηση με κάποια χαρακτηριστικά. Αυτά τα χαρακτηριστικά είναι η διάρκεια του συνολικού σήματος, η συχνότητα του ημιτόνου και το πλάτος του. Εμείς για τη συγκεκριμένη εφαρμογή ορίσαμε ένα ημίτονο με συχνότητα 6,9 KHz, εύρος τιμών από 0.2 έως 2.2 (προσθέσαμε μια DC συνιστώσα) και διάρκειας 100s. Αυτό το σήμα το θέτουμε σαν είσοδο στο κανάλι 0 του Analog-to Digital Converter του μικροελεγκτή μας.

```
SIGNAL void ADC0_Sine (void) {

    float volts;
    float frequency;
    float offset;
    float duration;
    float val;
    long i, end;

    volts      = 1.0;
    offset     = 1.2;
    frequency  = 6900;
    duration   = 100;

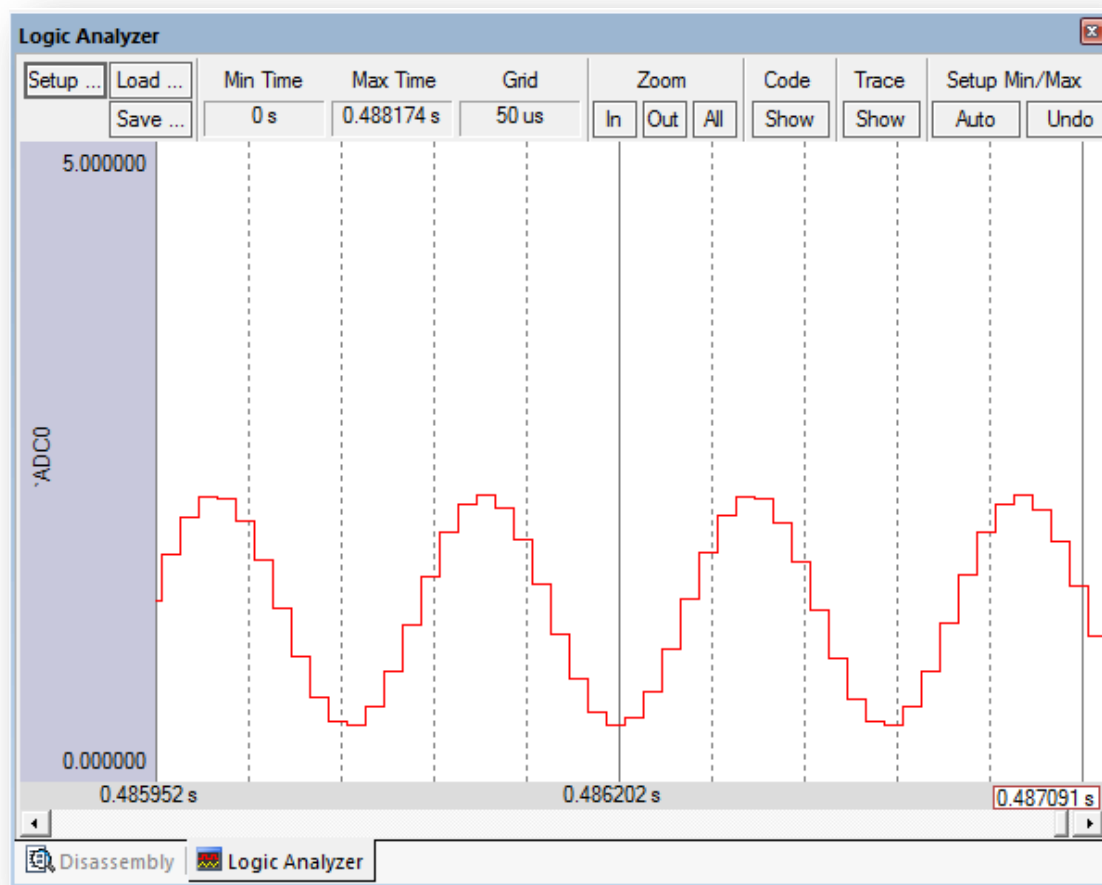
    printf ("Sine Wave Signal on AD Channel 0.\\n");

    end = (duration * 100000);
    for (i = 0 ; i < end; i++) {
        val = __sin (frequency * (((float) STATES) / CLOCK) * 2 * 3.1415926);
        ADC0 = (val * volts) +offset;
        swatch (0.00001);           // in 10 uSec steps
    }
}
```

Η σύνδεση του βασικού προγράμματος με αυτό μπορεί επίσης να γίνει πριν μπούμε σε debug mode μέσω της επιλογής Project→ Options for Target 7026→Debug. Εκεί, αφενός κλικάρουμε την επιλογή 'Simulator' που μας δίνει τη δυνατότητα να εξομοιώσουμε με μηδενικό υλικό και αφετέρου βρίσκουμε το .arxείο που έχουμε ήδη δημιουργήσει και το επιλέγουμε.

Ενώ πλέον είμαστε σε debug mode μπορούμε μέσω της επιλογής View→Analysis Window→ Logic Analyzer να παρατηρήσουμε τα σήματα που αναπαριστούν εισόδους, μεταβλητές και registers του βασικού μας προγράμματος. Εμείς μέσω του setup→ New(insert)→ ADC0 επιλέγουμε να εποπτεύσουμε την είσοδο ADC0 του μικροελεγκτή μας. Επίσης μέσω του command window δημιουργούμε στο toolbox ένα button για τη συνάρτηση με την εντολή: DEFINE BUTTON "ADC0

Sin","ADC0_Sine()". Στη συνέχεια κάνοντας κλικ στο κουμπί που δημιουργείται στο toolbox τροφοδοτούμε την είσοδο ADC0 με το ημίτονο που δημιουργήσαμε. Πιο κάτω παραθέτουμε το σήμα εισόδου όπως φαίνεται στο παράθυρο του Logic Analyzer.

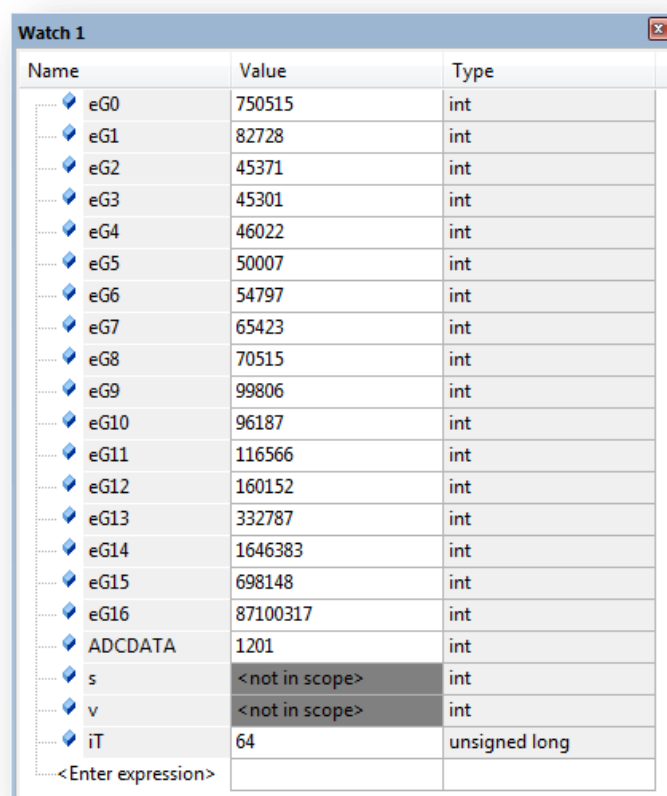


Σχήμα 7.2 Το ημιτονοειδές σήμα εισόδου συχνότητας 6.9KHz με το οποίο προσομοιώσαμε το σήμα που προκύπτει από τις δονήσεις της μηχανής όπως φαίνεται στο παράθυρο του Logic Analyzer

Είναι γνωστό ότι όλη η ενέργεια ενός σήματος ημιτόνου συμπεριλαμβάνεται σε μία μόνο τιμή συχνότητας που είναι και η βασική συχνότητά του. Έτσι το φάσμα συχνοτήτων ενός ημιτονοειδούς σήματος αναπαρίσται από μία συνιστώσα στην κύρια συχνότητά του. Το φάσμα συχνοτήτων ενός σήματος όπως αυτό που χρησιμοποιήσαμε συχνότητας 6,9 KHz, το οποίο είναι μετατοπισμένο στον άξονα y με τη χρήση μίας DC συνιστώσας αποτυπώνεται πλήρως με την παρουσία δύο συνιστωσών μία στην συχνότητα $f=0$ Hz και μία στην τιμή $f=6,9$ KHz. Συνεπώς αν θέλουμε να επιβεβαιώσουμε ότι το πρόγραμμά μας λειτουργεί σωστά θα πρέπει οι μεταβλητές eG14 ($f=6,9$ KHz) και eG16 ($f=0$) μετά την επεξεργασία κάθε πακέτου 64 δειγμάτων να λάβουν μία μεγάλη τιμή συγκριτικά με τις τιμές πλάτους στις υπόλοιπες τιμές συχνοτήτων.

Ενώ είμαστε πάλι σε debug mode μπορούμε να επιτηρήσουμε τις τιμές που λαμβάνει οποιαδήποτε μεταβλητή χρησιμοποιείται από τον κώδικά μας μέσω των Watch παραθύρων που μας προσφέρει το Keil. Αυτό το επιτυγχάνουμε αν επιλέξουμε διαδοχικά View→Watch Windows→Watch 1 or 2. Στο παράθυρο που ανοίγει κάνοντας διπλό κλικ στο “Enter expression” και εισάγοντας το όνομα μιας

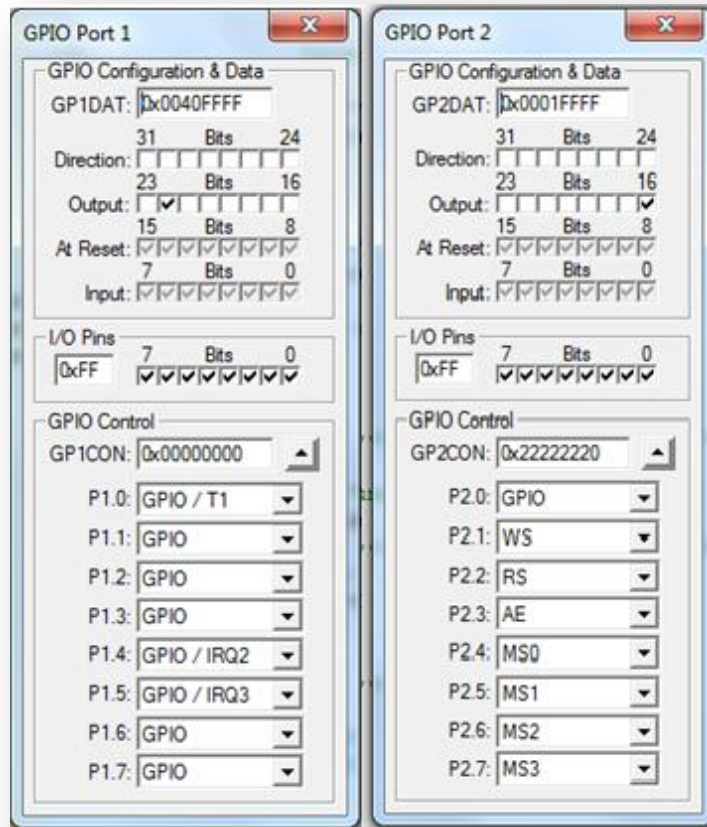
μεταβλητής που συμπεριλαμβάνεται στον κώδικά μας μπορούμε να βλέπουμε πως αυτή αλλάζει τιμές κατά τη διάρκεια εκτέλεσης του κώδικα. Πράγματι μετά από την επεξεργασία και των 64 δειγμάτων παρατηρούμε πως η μεταβλητή eG16 έχει λάβει μία πολύ μεγάλη τιμή, πράγμα που υποδεικνύει πολύ μεγάλο πλάτος στη συχνότητα $f=0$ Hz, ενώ η μεταβλητή eG14 έχει λάβει την αμέσως μεγαλύτερη τιμή, το οποίο με τη σειρά του υποδεικνύει μεγάλο πλάτος στη συχνότητα $f=6,9$ KHz.



Name	Value	Type
eG0	750515	int
eG1	82728	int
eG2	45371	int
eG3	45301	int
eG4	46022	int
eG5	50007	int
eG6	54797	int
eG7	65423	int
eG8	70515	int
eG9	99806	int
eG10	96187	int
eG11	116566	int
eG12	160152	int
eG13	332787	int
eG14	1646383	int
eG15	698148	int
eG16	87100317	int
ADCDATA	1201	int
s	<not in scope>	int
v	<not in scope>	int
iT	64	unsigned long
<Enter expression>		

Σχήμα 7.3 Το πλάτος σε κάθε μία από τις *target frequencies* όπως υπολογίζεται από τον αλγόριθμο Goertzel μετά την συλλογή και του 64ου δείγματος

Εφόσον έχουμε υπολογίσει το πλάτος σε κάθε συχνότητα με τις τιμές να είναι αυτές που φαίνονται πιο πάνω μένει να ελέγξουμε αν κάποια από αυτές είναι μεγαλύτερη από την τιμή αναφοράς που έχουμε ορίσει και αν στην περίπτωση αυτή ανάβει το αντίστοιχο led σαν ένδειξη βλάβης. Έτσι έχοντας θέσει σαν threshold την τιμή 1000000 παρατηρώντας τις παραπάνω τιμές θα πρέπει να ανάψει ένα led τόσο κατά τη σύγκριση της τιμής αυτής με την τιμή της μεταβλητής eG14 όσο και κατά τη σύγκρισή της με την τιμή της μεταβλητής eG16. Πιο συγκεκριμένα όπως προκύπτει από τον κώδικα θα πρέπει να γίνει 1 τόσο το pin6 της θύρας 1(bit22) όσο και το pin 0 της θύρας 2 (bit 16). Πράγματι...



Σχήμα 7.4 Το pin6 (bit 22) της θύρας 1 και το pin0 (bit16) της θύρας 2 γίνονται 1 αφού οι τιμές πλάτους στις συχνότητες 6.9KHz και 0KHz αντίστοιχα υπερβαίνουν την τιμή του threshold

Βιβλιογραφία

- [1] IEEE Standard on Piezoelectricity, ANSI/IEEE 176, 1987.
- [2] Piezoelectric Transducers for Vibration Control & Damping – Moheimani and Fleming
- [3] Piezoelectric and Acoustic Materials for Transducer Applications - Ahmad Safari, E. Koray. Akdoğan
- [4] Understanding the sampling process – R.N. Mutagi
- [5] Βασικές τεχνικές ψηφιακής επεξεργασίας σημάτων – Γεώργιος Β. Μουστακίδης, Καθηγητής, Πανεπιστήμιο Πατρών
- [6] Digital Signal Processing – Monson H. Hayes, Schaum's Outlines
- [7] The Scientist and Engineer's Guide to Digital Signal Processing - Steven W. Smith, Second Edition, 1999, California Technical Publishing
- [8] On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform - Fredrick J. Harris, Proc. IEEE, Vol. 66, No. 1, 1978 pp. 51-83.
- [9] Mixed Signal and DSP Design Techniques – Walt Kester
- [10] The FFT: Fundamentals and Concepts - R. W. Ramirez, Prentice-Hall, 1985.
- [11] Numerical Recipes in C, The Art of Scientific Computing - 2nd Edition, *William H. Press Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery*
- [12] "The Goertzel Algorithm", Kevin Banks, Embedded Systems Programming, August 8, 2002.
- [13] Determination of Spectral Parameters of Speech Signal by Goertzel Algorithm – Božo Tomas and Darko Zelenika
- [14] Effective Machinery Measurements using Dynamic Signal Analyzers – Hewlett Packard, Application Note 243-1
- [15] Machinery Analysis and Monitoring - Mitchell, John S, Second Edition, Penn Well Books, Tulsa, OK, 1993.
- [16] Practical Machinery Vibration Analysis and Predictive Maintenance - Scheffer & Girdhar
- [17] Programming Embedded Systems with C and GNU Development Tools – Michael Barr & Anthony Massa, 2nd Edition (2006)
- [18] Precision Analog Microcontroller, 12-Bit Analog I/O, ARM7TDMI MCU ADuC7019/20/21/22/24/25/26/27/28/29, Analog Devices
- [19] ADuC-MT7020 development board Users Manual, OLIMEX Ltd
- [20] Getting Started, Creating Applications with μVision 4 – Keil, Tools by ARM

ΠΑΡΑΡΤΗΜΑ 1

Κύριος σκοπός του προγράμματος μας είναι οι τιμές που ψηφιοποιούνται με τη βοήθεια του Analog-to-Digital Converter του μικροελεγκτή μας, αφού πολλαπλασιαστούν με ένα παράθυρο (Hamming) να υπολογίζεται το φάσμα συχνοτήτων τους μέσω του FFT και η ενέργεια σε κάθε συχνότητα. Οι τιμές αυτές αποτελούν δείγματα ενός σήματος ημιτόνου, το οποίο χρησιμοποιούμε για να αναπαραστήσουμε το σήμα που προκύπτει στην έξοδο του πιεζοηλεκτρικού μας αισθητήρα και προέρχεται από τις δονήσεις των μηχανικών μερών της μηχανής του αυτοκινήτου. Προφανώς η κυματομορφή ενός τέτοιου σήματος είναι πολύ πιο πολύπλοκη απλά στην προσπάθειά μας να γίνει πιο κατανοητή η διαδικασία επιλέξαμε ένα ημίτονο σαν σήμα εισόδου, χωρίς όμως αυτό να σημαίνει ότι ο κώδικάς μας δεν λειτουργεί ικανοποιητικά για οποιοδήποτε σήμα εισόδου. Αφού γίνει ο υπολογισμός της ενέργειας σε κάθε συχνότητα στη συνέχεια ακολουθούμε τη λογική της αναγνώρισης προτύπων. Αυτό δεν είναι τίποτα άλλο από την αναγνώριση (αν υπάρχουν) των βλαβών βάσει της διαδικασίας που περιγράφηκε σε προηγούμενο κεφάλαιο για την ανάλυση των σφαλμάτων σε περιστρεφόμενο μηχανικό εξοπλισμό.

Όπως εξηγήσαμε και πιο πάνω, η δόνηση κάθε εξαρτήματος που απαρτίζει έναν μηχανικό εξοπλισμό εκφράζεται με την παρουσία μιας συνιστώσας σε μια συγκεκριμένη τιμή συχνότητας του φάσματος, που αποτελεί κατά κάποιο τρόπο την ταυτότητά του. Έτσι γνωρίζοντας εκ των προτέρων τα μηχανικά μέρη της μηχανής μας και συνεπώς και που θα πρέπει να περιμένουμε την εμφάνιση συνιστωσών στο φάσμα συχνοτήτων και το πλάτος που θα πρέπει να έχουν αν δεν υπάρχει βλάβη, η παρουσία συνιστωσών σε διαφορετικά σημεία του φάσματος και η εμφάνιση τιμών πλάτους μεγαλύτερες των αναμενομένων υποδηλώνουν την παρουσία σφάλματος. Λαμβάνοντας υπ' όψη τις αλλαγές που προκαλεί κάθε σφάλμα στο φάσμα συχνοτήτων μπορούμε να το προσδιορίσουμε.

Αφού εξηγήσαμε σε γενικές γραμμές για ποιο λόγο και τι ακριβώς κάνει ο κώδικάς μας ας δούμε τώρα πως ακριβώς το πετυχαίνει αυτό μέσα από την ανάλυση των κύριων χαρακτηριστικών του.

- Η συχνότητα δειγματοληψίας καθορίζεται από την τιμή που βάζουμε στον timer, ο οποίος σε μια count-down μέτρηση κάθε φορά που μηδενίζεται, ο ADC κάνει μια analog-to-digital μετατροπή. Αυτό επιτυγχάνεται μέσω του καταχωρητή ADCCON, όπου προγραμματίζουμε τον ADC έτσι ώστε κάθε φορά που ο TIMER 0 στην περίπτωση μας μηδενίζεται να σηματοδοτείται μια καινούρια μετατροπή. Επίσης μέσω του καταχωρητή IRQEN ορίζουμε την διακοπή να την προκαλεί ο ADC, οπότε κάθε φορά που ο timer μηδενίζεται ο ADC προκαλεί μια διακοπή στο κύριο πρόγραμμα και εκτελεί τη μετατροπή.
- Θεωρούμε αρχικά 128 δείγματα. Τα δείγματα καθορίζουν το frequency resolution που θα έχουμε. Έτσι όσο μεγαλύτερος ο αριθμός των δειγμάτων τόσο λιγότερο θα απέχουν μεταξύ τους τα σημεία του άξονα συχνοτήτων και προφανώς τα αποτελέσματα που θα προκύπτουν θα ανταποκρίνονται πιο πολύ στην πραγματικότητα. Επίσης βασική προϋπόθεση του FFT είναι ο αριθμός των δειγμάτων να είναι δύναμη του 2 και το 128 ικανοποιεί αυτό το κριτήριο.
- Χρησιμοποιούμε έναν complex FFT, ο οποίος δέχεται σαν είσοδο μιγαδικούς αριθμούς, έναν πίνακα δηλαδή με το πραγματικό μέρος (REAL[]) και έναν με το φανταστικό (IMAG[]), αν και οι τιμές που επεξεργαζόμαστε είναι πραγματικοί αριθμοί. Γι' αυτό και στον κώδικα συμπληρώνουμε τον πίνακα IMAG[] με μηδενικά. Υπάρχουν αλγόριθμοι FFT που επεξεργάζονται πραγματικές τιμές και στην προσπάθειά μας να αυξήσουμε την ταχύτητα

επεξεργασίας των τιμών χρησιμοποιήσαμε και μία τέτοια υλοποίηση του FFT απλά η βελτίωση στην ταχύτητα δεν ήταν τόσο σημαντική ώστε να παραθέσουμε ξεχωριστά ένα πρόγραμμα που τον χρησιμοποιεί.

- Για την παραθυροποίηση των δεδομένων χρησιμοποιήσαμε το Hamming window, το οποίο εκφράζεται από την εξίσωση: $0,54-0,46\cos(2\pi x/N-1)$. Κάθε δείγμα που ψηφιοποιείται μέσω του ADC πολλαπλασιάζεται απευθείας με τον αντίστοιχο συντελεστή του παραθύρου.
- Χρησιμοποιείται ένας πίνακας 256 θέσεων για την αποθήκευση των αποτελεσμάτων που προκύπτουν από τον ADC. Η επεξεργασία των δεδομένων γίνεται ως εξής: όταν ψηφιοποιηθεί και το στοιχείο που βρίσκεται στη θέση 127 (128ο στοιχείο) ξεκινά η επεξεργασία των πρώτων αυτών 128 δειγμάτων από τον FFT και υπολογίζεται η ενέργειά τους. Παράλληλα για να μην χάνονται τιμές τα δείγματα που συλλέγονται σ' αυτό το χρονικό διάστημα γεμίζουν τις θέσεις 128-255 έτσι ώστε όταν συμπληρωθεί όλος ο πίνακας και έχει τελειώσει η επεξεργασία του πρώτου πακέτου να ξεκινήσει η επεξεργασία του δεύτερου. Με τη συμπλήρωση και των 256 θέσεων του πίνακα ο δείκτης μηδενίζεται και ο πίνακας συμπληρώνεται από την αρχή. Έτσι τα πρώτα 128 δείγματα των οποίων η επεξεργασία έτσι κι αλλιώς έχει τελειώσει αντικαθίστανται από τις νέες 128 τιμές που θα αποτελέσουν και το νέο πακέτο. Μ' αυτόν τον τρόπο συνεχίζεται η επεξεργασία κάθε πακέτου δειγμάτων που συλλέγεται. Στόχος της ταυτόχρονης αυτής επεξεργασίας και συλλογής των δειγμάτων είναι η ικανοποίηση των real-time απαιτήσεων της εφαρμογής μας.
- Με τη συνάρτηση startCpu γίνεται η αρχικοποίηση της CPU και το core clock προγραμματίζεται να λειτουργεί στη μέγιστη δυνατή συχνότητά του, τα 41,78 MHz.

```
#include <ADuC7026.h>                                // Include ADuC7026 Header File

#include <math.h>

#include <stdio.h>


#define pi 3.14159265

int N=128;

int M=7;


int d=0;

int flag1=0;                                           /* γίνεται 1 όταν έχει τελειώσει η δειγματοληψία και της
128ης τιμής */
```

```

int flag2=0;          /* γίνεται 1 όταν έχει τελειώσει η δειγματοληψία και της
256ης τιμής */

void ADCpoweron (int); //ADC power-on


static float HamWin[128]; /* πίνακας στον οποίο αποθηκεύονται οι συντελεστές του
Hamming window */

float E[128]; // πίνακας που περιέχει την ενέργεια σε κάθε συχνότητα

float REAL1[128]; /* REAL1 και REAL2 => πίνακες στους οποίους αποθηκεύεται το
πραγματικό μέρος - IMAG1 και IMAG2 => πίνακες στους οποίους αποθηκεύεται το
φανταστικό μέρος */

float IMAG1[128];

float REAL2[128];

float IMAG2[128];

static float ADCDATA[256]; /* πίνακας στον οποίο αποθηκεύονται οι τιμές της
δειγματοληψίας */


void delay (int);          // Συνάρτηση καθυστέρησης

void startCpu (void);      // Αρχικοποίηση της CPU

void HammingWindow (void); // Παράθυρο Hamming

void fft (int n, int m, float x[], float y[]);


int main (void) {

    int r;

    startCpu();

    HammingWindow();

    ADCpoweron(20000);

```

```

ADCCP = 0x00;           //επιλογή θετικού καναλιού 0

ADCCON = 0x6A2;         // start conversion on timer 0

// ADC Config: fADC/2, acq. time = 8 clocks


REFCON = 0x01;          // connect internal 2.5V reference to VREF pin

IRQEN = ADC_BIT;        // Enable ADC IRQ ( 0x80 )


// timer0 configuration

T0LD=0x2B3; /* τιμή στον timer ώστε να επιτύχουμε συχνότητα δειγματοληψίας
60 KHz */

T0CON = 0xC0;    //periodic mode


while(1)
{
    if (flag1)
    {
        flag1=0;
        for (r=0;r<128;r++)
        {

            REAL1[r]=(ADCDATA[r]); /*στον πίνακα REAL αποθηκεύονται
οι τιμές που προκύπτουν από τον ADC οι οποίες έχουν πολλαπλασιαστεί με τους
συντελεστές του παραθύρου */

            IMAG1[r]=0;           /* όλα τα στοιχεία του πίνακα
IMAG είναι 0 αφού το σήμα μας είναι πραγματικό */

        }
    }
}

```

```

        fft(N, M, REAL1, IMAG1); /* οι πίνακες REAL και IMAG μετά την
επεξεργασία τους με τον FFT περιέχουν το πραγματικό και το φανταστικό μέρος
αντίστοιχα του σήματος στο πεδίο της συχνότητας */

```

```

        for (r=0;r<128;r++)
        {
            E[r]= (REAL1[r]*REAL1[r]) + (IMAG1[r]*IMAG1[r]); /*0
πίνακας E περιέχει την ενέργεια του σήματος σε κάθε συχνότητα */

```

```

        }

```

```

    }

```

```

    else if (flag2)

```

```

    {

```

```

        flag2=0;

```

```

        for (r=128;r<256;r++)

```

```

        {

```

```

            REAL2[r-128]=(ADCDATA[r]);

```

```

            IMAG2[r-128]=0;

```

```

        }

```

```

    fft(N, M, REAL2, IMAG2);

```

```

    for (r=0;r<128;r++)

```

```

    {

```

```

        E[r]=(REAL2[r]*REAL2[r]) + (IMAG2[r]*IMAG2[r]);

```

```

    }

```

```

}

```

```

}

```

```

    return 0 ;

}

/*****
/*
/*      Interrupt Service Routine
/*
/*
*****/

__irq void IRQ_Handler()
{
    if (d<128)
    {
        ADCDATA[d] = (ADCDAT >> 16)*(HamWin[d]);
        d++;
    }
    else if (d<256)
    {
        ADCDATA[d]= (ADCDAT >> 16)*(HamWin[d-128]);
        d++;
    }
    if (d==128)
    {
        flag1=1;
    }
    if (d==256)
    {
        flag2=1;
        d=0;
    }
}

```

```

        }

        return ;
    }

void startCpu(void)
{
    POWKEY1=0x01;
    POWCON=0x00; //CPU στα 41,78MHz
    POWKEY2=0xF4;
    return ;
}

void HammingWindow (void) {
    int x;

    for (x=0;x<128;x++)
    {
        HamWin[x]=0.54-(0.46*cos((2*pi*x)/(N-1)));
    }
    return ;
}

void delay (int length)
{
    while (length >=0)
        length--;
}

```

```
}
```

```
void ADCpoweron(int time)
```

```
{
```

```
    ADCCON = 0x20;
```

```
    while (time >=0)    // wait for ADC to be fully powered on
```

```
        time--;
```

```
}
```

```
void fft(int n,int m,float x[],float y[])
```

```
{
```

```
int i,j,k,n1,n2;
```

```
float c,s,e,a,t1,t2;
```

```
j = 0;                                /* bit-reverse */
```

```
n2 = n/2;
```

```
for (i=1; i < n - 1; i++)
```

```
{
```

```
n1 = n2;
```

```
while ( j >= n1 )
```

```
{
```

```
j = j - n1;
```

```
n1 = n1/2;
```

```
}
```

```
j = j + n1;
```

```
if (i < j)
```

```
{
```

```
t1 = x[i];
```

```
x[i] = x[j];
```

```
x[j] = t1;
```

```

t1 = y[i];
y[i] = y[j];
y[j] = t1;
}
}

n1 = 0;                                     /* FFT */
n2 = 1;
for (i=0; i < m; i++)
{
n1 = n2;
n2 = n2 + n2;
e = -6.283185307179586/n2;
a = 0.0;
for (j=0; j < n1; j++)
{
c = cos(a);
s = sin(a);
a = a + e;
for (k=j; k < n; k=k+n2)
{
t1 = c*x[k+n1] - s*y[k+n1];
t2 = s*x[k+n1] + c*y[k+n1];
x[k+n1] = x[k] - t1;
y[k+n1] = y[k] - t2;
x[k] = x[k] + t1;
y[k] = y[k] + t2;
}
}

```

```

}
}
return;
}

```

Στην 1η υλοποίηση χρησιμοποιήθηκε για το πέρασμα από το πεδίο του χρόνου στο πεδίο της συχνότητας ένας απλός radix-2 DIT(decimation-in-time) FFT. Ο κύριος στόχος του κώδικά μας ήταν εξ αρχής να μπορεί να λειτουργεί σε real-time, να μην χάνει δηλαδή καθόλου τιμές ενώ επεξεργάζεται ένα συμπληρωμένο πακέτο δειγμάτων. Αυτό είναι που καθορίζει και το βαθμό λειτουργικότητάς του. Έτσι ενώ ο κώδικας που δημιουργήσαμε μπορεί να μην παρουσιάζει ούτε συντακτικά ούτε λογικά λάθη μπορεί όμως να μην ανταποκρίνεται στις real-time απαιτήσεις της εφαρμογής μας, γεγονός που τον χρήσει αυτόματα μη χρησιμοποιήσιμο. Η βασική προϋπόθεση που πρέπει να πληροί το πρόγραμμά μας είναι να λειτουργεί σύμφωνα με τον παραπάνω τρόπο στη συχνότητα δειγματοληψίας που καθορίζεται από τη φύση της εφαρμογής μας. Στη συγκεκριμένη εφαρμογή η συχνότητα με την οποία δειγματοληπτούμε δεν μπορεί να είναι μικρότερη από 60 KHz.

Έχοντας σαν συχνότητα δειγματοληψίας τα 60 KHz μπορούμε να υπολογίσουμε πόσο χρόνο χρειάζεται κάθε πακέτο δειγμάτων για να συλλεχθεί. Για κάθε δείγμα απαιτείται χρόνος $1/60000=16,7 \mu s$. Στην 1η υλοποίηση επεξεργαζόμαστε πακέτα των 128 δειγμάτων οπότε για τη συλλογή κάθε τέτοιου πακέτου απαιτείται χρόνος $128 \times 16,7=2,137 ms$. Εμείς εφόσον θέλουμε να επιτύχουμε παράλληλη επεξεργασία και συλλογή δειγμάτων εύκολα μπορούμε να συμπεράνουμε ότι είναι απαραίτητο ο χρόνος επεξεργασίας του προηγούμενου πακέτου να είναι μικρότερος από το χρόνο συλλογής του επόμενου. Έτσι η επεξεργασία των 128 στην περίπτωση μας δειγμάτων από τον FFT και ο υπολογισμός της ενέργειας σε κάθε συχνότητα πρέπει να διαρκεί το πολύ 2,137 ms.

Ο απλός radix-2 DIT FFT που χρησιμοποιήσαμε, όπως προκύπτει εκτελώντας τον κώδικα στον simulator του Keil, χρειάζεται για να εκτελεστεί 73 ms, περίπου 35 φορές δηλαδή πιο αργά από την επιθυμητή ταχύτητα. Πιο συγκεκριμένα στον χρόνο που απαιτείται για τον υπολογισμό του FFT έχουν συλλεχθεί 35 νέα πακέτα δειγμάτων τα οποία δεν υπόκεινται επεξεργασία και χάνονται. Κάτι τέτοιο μας αναγκάζει για να μπορέσει να χρησιμοποιηθεί ο πιο πάνω κώδικας είτε να μειώσουμε τη συχνότητα δειγματοληψίας (συγκεκριμένα εδώ από 60KHz στο 1 KHz) είτε να απενεργοποιούμε τις διακοπές κάθε φορά που συλλέγεται ένα πακέτο 128 δειγμάτων. Στην πρώτη περίπτωση εφόσον το σήμα εισόδου περιλαμβάνει συχνότητες μέχρι και 30 KHz παραβαίνουμε το κριτήριο του Nyquist και άρα οδηγούμαστε σε λάθος αποτελέσματα ενώ στη δεύτερη μπορεί το κριτήριο να ικανοποιείται αλλά το πρόγραμμά σας δεν λειτουργεί σε real time. Λαμβάνοντας υπόψη τα παραπάνω γίνεται εύκολα αντιληπτό ότι ο πιο πάνω κώδικας δεν μπορεί να καλύψει τις ανάγκες της εφαρμογής μας και η χρήση του ενδείκνυται μόνο σε περιπτώσεις όπου γίνεται επεξεργασία σημάτων με συχνότητες μέχρι 500 Hz και επομένως καθίσταται αναγκαία η χρησιμοποίηση ενός άλλου πολύ πιο γρήγορου FFT.

ΠΑΡΑΡΤΗΜΑ 2

Στην 2η υλοποίηση προτιμήθηκε η χρησιμοποίηση ενός fixed-point FFT αντί του απλού FFT που χρησιμοποιήθηκε στην πρώτη περίπτωση. Η κύρια διαφορά τους, η οποία και έχει άμεσο αντίκτυπο στο χρόνο επεξεργασίας των δεδομένων, έγκειται στο ότι ενώ ο πρώτος χρησιμοποιεί σαν είσοδο πίνακες τύπου float ο δεύτερος επεξεργάζεται δεδομένα τύπου int ή ακόμη και short. Με τον τρόπο αυτό χάνουμε κατά ένα μικρό ποσοστό σε ακρίβεια αλλά κερδίζουμε πάρα πολύ σε ταχύτητα. Επίσης όταν επεξεργαζόμαστε στοιχεία με τον απλό FFT πολλαπλασιάζουμε τα δεδομένα μας κανονικά με ημίτονα και συνημίτονα, ο υπολογισμός των οποίων ως γνωστόν είναι ιδιαίτερα χρονοβόρος. Αντίθετα ο fixed-point που χρησιμοποιούμε στην 2η υλοποίηση χρησιμοποιεί έναν πίνακα 1024 στοιχείων στον οποίο είναι αποθηκευμένες οι τιμές μιας περιόδου ενός ημιτόνου και οι οποίες είναι πολλαπλασιασμένες με την τιμή 32768 (1000000000000000 binary) και έχει κρατηθεί μόνο το ακέραιο μέρος τους. Γι' αυτό το λόγο όταν εκτελείται ο fixed-point FFT και οι τιμές που προκύπτουν από τη δειγματοληψία πολλαπλασιάζονται με μια τιμή ημιτόνου ή συνημιτόνου που υπάρχει στον πίνακα αυτόν καλείται μία συνάρτηση fix_mpy η οποία πολλαπλασιάζει τις δύο τιμές και στη συνέχεια εκτελεί shift στο αποτέλεσμα 15 θέσεις δεξιά διαιρώντας ουσιαστικά με την τιμή 32768 κρατώντας πάλι μόνο το ακέραιο μέρος του αποτελέσματος.

Οι τιμές που αποθηκεύονται στους πίνακες είναι ακέραιοι αριθμοί. Οι πίνακες δηλαδή που χρησιμοποιούνται από τον κώδικα είναι τύπου short ή int εκτός από τον πίνακα HamWin που περιέχει τους συντελεστές του παραθύρου ο οποίος είναι τύπου float. Κατά τον πολλαπλασιασμό κάθε δείγματος το οποίο είναι μια τιμή τύπου short με τον αντίστοιχο συντελεστή παραθύρου το οποίο είναι τύπου float προκύπτει ένα αποτέλεσμα τύπου short (ο ADCDATA είναι τύπου short) με την αυτόματη εξάλειψη του δεκαδικού μέρους. Με αυτόν τον τρόπο μπορεί να μειώνεται σε ένα μικρό βαθμό η ακρίβεια των αποτελεσμάτων αλλά αυξάνεται σημαντικά η ταχύτητα επεξεργασίας των δεδομένων.

```
#include <ADuC7026.h>                                // Include ADuC7026 Header File

#include <math.h>

#include <stdio.h>

#define M 7

#define FIX_MPY(DEST,A,B)          DEST = ((long)(A) * (long)(B))>>15

#define N          (1<<M)          //N=128

#define pi 3.14159265

#define N_WAVE          1024        /* dimension of Sinewave[] */

#define LOG2_N_WAVE          10      /* log2(N_WAVE) */
```

```

int d=0;

int flag1=0;          /* γίνεται 1 όταν έχει τελειώσει η δειγματοληψία και της
128ης τιμής */

int flag2=0;          /* γίνεται 1 όταν έχει τελειώσει η δειγματοληψία και της
256ης τιμής */

float HamWin[128];    /* πίνακας στον οποίο αποθηκεύονται οι συντελεστές του
Hamming window */

int E[128];           // πίνακας που περιέχει την ενέργεια σε κάθε συχνότητα

short REAL1[128];     /* REAL1 και REAL2 → πίνακες στους οποίους αποθηκεύεται το
πραγματικό μέρος - IMAG1 και IMAG2 → πίνακες στους οποίους αποθηκεύεται το
φανταστικό μέρος */

short IMAG1[128];

short REAL2[128];

short IMAG2[128];

short ADCDATA[256];   /* πίνακας στον οποίο αποθηκεύονται οι τιμές της
δειγματοληψίας */


void ADCpoweron(int); //ADC power-on

short fix_mpy(short a, short b); //fixed multiplication

void delay(int);       // Συνάρτηση καθυστέρησης

void startCpu(void);   // Αρχικοποίηση της CPU

void HammingWindow (void); //παράθυρο Hamming

void fix_fft(short fr[], short fi[], int m); //fixed point FFT


/* Ημίτονο με τις τιμές του οποίου γίνεται ο fixed multiplication με τις τιμές
των πινάκων REAL και IMAG μέσα στον FFT (οι τιμές είναι πολλαπλασιασμένες με τον
αριθμό 32768) */

short Sinewave[1024] = {
    0,      201,    402,    603,    804,    1005,    1206,    1406,
    1607,    1808,    2009,    2209,    2410,    2610,    2811,    3011,
    3211,    3411,    3611,    3811,    4011,    4210,    4409,    4608,

```

4807,	5006,	5205,	5403,	5601,	5799,	5997,	6195,
6392,	6589,	6786,	6982,	7179,	7375,	7571,	7766,
7961,	8156,	8351,	8545,	8739,	8932,	9126,	9319,
9511,	9703,	9895,	10087,	10278,	10469,	10659,	10849,
11038,	11227,	11416,	11604,	11792,	11980,	12166,	12353,
12539,	12724,	12909,	13094,	13278,	13462,	13645,	13827,
14009,	14191,	14372,	14552,	14732,	14911,	15090,	15268,
15446,	15623,	15799,	15975,	16150,	16325,	16499,	16672,
16845,	17017,	17189,	17360,	17530,	17699,	17868,	18036,
18204,	18371,	18537,	18702,	18867,	19031,	19194,	19357,
19519,	19680,	19840,	20000,	20159,	20317,	20474,	20631,
20787,	20942,	21096,	21249,	21402,	21554,	21705,	21855,
22004,	22153,	22301,	22448,	22594,	22739,	22883,	23027,
23169,	23311,	23452,	23592,	23731,	23869,	24006,	24143,
24278,	24413,	24546,	24679,	24811,	24942,	25072,	25201,
25329,	25456,	25582,	25707,	25831,	25954,	26077,	26198,
26318,	26437,	26556,	26673,	26789,	26905,	27019,	27132,
27244,	27355,	27466,	27575,	27683,	27790,	27896,	28001,
28105,	28208,	28309,	28410,	28510,	28608,	28706,	28802,
28897,	28992,	29085,	29177,	29268,	29358,	29446,	29534,
29621,	29706,	29790,	29873,	29955,	30036,	30116,	30195,
30272,	30349,	30424,	30498,	30571,	30643,	30713,	30783,
30851,	30918,	30984,	31049,	31113,	31175,	31236,	31297,
31356,	31413,	31470,	31525,	31580,	31633,	31684,	31735,
31785,	31833,	31880,	31926,	31970,	32014,	32056,	32097,
32137,	32176,	32213,	32249,	32284,	32318,	32350,	32382,
32412,	32441,	32468,	32495,	32520,	32544,	32567,	32588,
32609,	32628,	32646,	32662,	32678,	32692,	32705,	32717,

32727, 32736, 32744, 32751, 32757, 32761, 32764, 32766,
 32767,
 32766, 32764, 32761, 32757, 32751, 32744, 32736,
 32727, 32717, 32705, 32692, 32678, 32662, 32646, 32628,
 32609, 32588, 32567, 32544, 32520, 32495, 32468, 32441,
 32412, 32382, 32350, 32318, 32284, 32249, 32213, 32176,
 32137, 32097, 32056, 32014, 31970, 31926, 31880, 31833,
 31785, 31735, 31684, 31633, 31580, 31525, 31470, 31413,
 31356, 31297, 31236, 31175, 31113, 31049, 30984, 30918,
 30851, 30783, 30713, 30643, 30571, 30498, 30424, 30349,
 30272, 30195, 30116, 30036, 29955, 29873, 29790, 29706,
 29621, 29534, 29446, 29358, 29268, 29177, 29085, 28992,
 28897, 28802, 28706, 28608, 28510, 28410, 28309, 28208,
 28105, 28001, 27896, 27790, 27683, 27575, 27466, 27355,
 27244, 27132, 27019, 26905, 26789, 26673, 26556, 26437,
 26318, 26198, 26077, 25954, 25831, 25707, 25582, 25456,
 25329, 25201, 25072, 24942, 24811, 24679, 24546, 24413,
 24278, 24143, 24006, 23869, 23731, 23592, 23452, 23311,
 23169, 23027, 22883, 22739, 22594, 22448, 22301, 22153,
 22004, 21855, 21705, 21554, 21402, 21249, 21096, 20942,
 20787, 20631, 20474, 20317, 20159, 20000, 19840, 19680,
 19519, 19357, 19194, 19031, 18867, 18702, 18537, 18371,
 18204, 18036, 17868, 17699, 17530, 17360, 17189, 17017,
 16845, 16672, 16499, 16325, 16150, 15975, 15799, 15623,
 15446, 15268, 15090, 14911, 14732, 14552, 14372, 14191,
 14009, 13827, 13645, 13462, 13278, 13094, 12909, 12724,
 12539, 12353, 12166, 11980, 11792, 11604, 11416, 11227,
 11038, 10849, 10659, 10469, 10278, 10087, 9895, 9703,

9511, 9319, 9126, 8932, 8739, 8545, 8351, 8156,
 7961, 7766, 7571, 7375, 7179, 6982, 6786, 6589,
 6392, 6195, 5997, 5799, 5601, 5403, 5205, 5006,
 4807, 4608, 4409, 4210, 4011, 3811, 3611, 3411,
 3211, 3011, 2811, 2610, 2410, 2209, 2009, 1808,
 1607, 1406, 1206, 1005, 804, 603, 402, 201,
 0, -201, -402, -603, -804, -1005, -1206, -1406,
 -1607, -1808, -2009, -2209, -2410, -2610, -2811, -3011,
 -3211, -3411, -3611, -3811, -4011, -4210, -4409, -4608,
 -4807, -5006, -5205, -5403, -5601, -5799, -5997, -6195,
 -6392, -6589, -6786, -6982, -7179, -7375, -7571, -7766,
 -7961, -8156, -8351, -8545, -8739, -8932, -9126, -9319,
 -9511, -9703, -9895, -10087, -10278, -10469, -10659, -10849,
 -11038, -11227, -11416, -11604, -11792, -11980, -12166, -12353,
 -12539, -12724, -12909, -13094, -13278, -13462, -13645, -13827,
 -14009, -14191, -14372, -14552, -14732, -14911, -15090, -15268,
 -15446, -15623, -15799, -15975, -16150, -16325, -16499, -16672,
 -16845, -17017, -17189, -17360, -17530, -17699, -17868, -18036,
 -18204, -18371, -18537, -18702, -18867, -19031, -19194, -19357,
 -19519, -19680, -19840, -20000, -20159, -20317, -20474, -20631,
 -20787, -20942, -21096, -21249, -21402, -21554, -21705, -21855,
 -22004, -22153, -22301, -22448, -22594, -22739, -22883, -23027,
 -23169, -23311, -23452, -23592, -23731, -23869, -24006, -24143,
 -24278, -24413, -24546, -24679, -24811, -24942, -25072, -25201,
 -25329, -25456, -25582, -25707, -25831, -25954, -26077, -26198,
 -26318, -26437, -26556, -26673, -26789, -26905, -27019, -27132,
 -27244, -27355, -27466, -27575, -27683, -27790, -27896, -28001,
 -28105, -28208, -28309, -28410, -28510, -28608, -28706, -28802,

-28897, -28992, -29085, -29177, -29268, -29358, -29446, -29534,
 -29621, -29706, -29790, -29873, -29955, -30036, -30116, -30195,
 -30272, -30349, -30424, -30498, -30571, -30643, -30713, -30783,
 -30851, -30918, -30984, -31049, -31113, -31175, -31236, -31297,
 -31356, -31413, -31470, -31525, -31580, -31633, -31684, -31735,
 -31785, -31833, -31880, -31926, -31970, -32014, -32056, -32097,
 -32137, -32176, -32213, -32249, -32284, -32318, -32350, -32382,
 -32412, -32441, -32468, -32495, -32520, -32544, -32567, -32588,
 -32609, -32628, -32646, -32662, -32678, -32692, -32705, -32717,
 -32727, -32736, -32744, -32751, -32757, -32761, -32764, -32766,
 -32767, -32766, -32764, -32761, -32757, -32751, -32744, -32736,
 -32727, -32717, -32705, -32692, -32678, -32662, -32646, -32628,
 -32609, -32588, -32567, -32544, -32520, -32495, -32468, -32441,
 -32412, -32382, -32350, -32318, -32284, -32249, -32213, -32176,
 -32137, -32097, -32056, -32014, -31970, -31926, -31880, -31833,
 -31785, -31735, -31684, -31633, -31580, -31525, -31470, -31413,
 -31356, -31297, -31236, -31175, -31113, -31049, -30984, -30918,
 -30851, -30783, -30713, -30643, -30571, -30498, -30424, -30349,
 -30272, -30195, -30116, -30036, -29955, -29873, -29790, -29706,
 -29621, -29534, -29446, -29358, -29268, -29177, -29085, -28992,
 -28897, -28802, -28706, -28608, -28510, -28410, -28309, -28208,
 -28105, -28001, -27896, -27790, -27683, -27575, -27466, -27355,
 -27244, -27132, -27019, -26905, -26789, -26673, -26556, -26437,
 -26318, -26198, -26077, -25954, -25831, -25707, -25582, -25456,
 -25329, -25201, -25072, -24942, -24811, -24679, -24546, -24413,
 -24278, -24143, -24006, -23869, -23731, -23592, -23452, -23311,
 -23169, -23027, -22883, -22739, -22594, -22448, -22301, -22153,
 -22004, -21855, -21705, -21554, -21402, -21249, -21096, -20942,

```

-20787, -20631, -20474, -20317, -20159, -20000, -19840, -19680,
-19519, -19357, -19194, -19031, -18867, -18702, -18537, -18371,
-18204, -18036, -17868, -17699, -17530, -17360, -17189, -17017,
-16845, -16672, -16499, -16325, -16150, -15975, -15799, -15623,
-15446, -15268, -15090, -14911, -14732, -14552, -14372, -14191,
-14009, -13827, -13645, -13462, -13278, -13094, -12909, -12724,
-12539, -12353, -12166, -11980, -11792, -11604, -11416, -11227,
-11038, -10849, -10659, -10469, -10278, -10087, -9895, -9703,
-9511, -9319, -9126, -8932, -8739, -8545, -8351, -8156,
-7961, -7766, -7571, -7375, -7179, -6982, -6786, -6589,
-6392, -6195, -5997, -5799, -5601, -5403, -5205, -5006,
-4807, -4608, -4409, -4210, -4011, -3811, -3611, -3411,
-3211, -3011, -2811, -2610, -2410, -2209, -2009, -1808,
-1607, -1406, -1206, -1005, -804, -603, -402, -201,

```

```
};
```

```
int main (void) {
```

```
    int r;
```

```
    startCpu();
```

```
    HammingWindow();
```

```
    ADCpoweron(20000);
```

```
    ADCCP = 0x00;           //επιλογή θετικού καναλιού 0
```

```
    ADCCON = 0x6A2;         // start conversion on timer 0
```

```
    // ADC Config: fADC/2, acq. time = 8 clocks
```

```

REFCON = 0x01;    // connect internal 2.5V reference to VREF pin

IRQEN = ADC_BIT;    // Enable ADC IRQ ( 0x80 )

// timer0 configuration

T0LD=0x2B3; /* τιμή στον timer ώστε να επιτύχουμε συχνότητα δειγματοληψίας
60 KHz */

T0CON = 0xC0;    // periodic mode

while(1)
{
    if (flag1)
    {
        flag1=0;

        for (r=0;r<128;r++)
        {
            REAL1[r]=(ADCDATA[r]);    /*στον πίνακα REAL
αποθηκεύονται οι τιμές που προκύπτουν από τον ADC οι οποίες έχουν
πολλαπλασιαστεί με τους συντελεστές του παραθύρου */

            IMAG1[r]=0;                /* όλα τα στοιχεία του πίνακα
IMAG είναι 0 αφού το σήμα μας είναι πραγματικό */

/*Και οι 2 πίνακες είναι τύπου short αφού ο fixed point FFT που χρησιμοποιείται
επεξεργάζεται short πίνακες */

        }

        fix_fft(REAL1, IMAG1, M);    /* οι πίνακες REAL και IMAG μετά
την επεξεργασία τους με τον FFT περιέχουν το πραγματικό και το φανταστικό μέρος
αντίστοιχα του σήματος στο πεδίο της συχνότητας */

        for (r=0;r<128;r++)
        {

```

```

        E[r]=(REAL1[r]*REAL1[r]) + (IMAG1[r]*IMAG1[r]);
/*Ο πίνακας E περιέχει την ενέργεια του σήματος σε κάθε συχνότητα και είναι
τύπου int γιατί περιέχει το άθροισμα των τετραγώνων δύο short μεταβλητών οπότε
προκύπτει η ανάγκη για περισσότερα bytes */

```

```

    }

```

```

}

```

```

else if (flag2)

```

```

{

```

```

    flag2=0;

```

```

    for (r=128;r<256;r++)

```

```

    {

```

```

        REAL2[r-128]=(ADCDATA[r]);

```

```

        IMAG2[r-128]=0;

```

```

    }

```

```

    fix_fft(REAL2, IMAG2, M);

```

```

    for (r=0;r<128;r++)

```

```

    {

```

```

        E[r]=(REAL2[r]*REAL2[r]) + (IMAG2[r]*IMAG2[r]);

```

```

    }

```

```

}

```

```

    }
    return 0 ;
}

/*****
/*
/*      Interrupt Service Routine      */
/*
/*
*****/
__irq void IRQ_Handler()
{
    if (d<128)
    {
        ADCDATA[d] = (ADCDAT >> 16)*(HamWin[d]);
        d++;
    }
    else if (d<256)
    {
        ADCDATA[d] = (ADCDAT >> 16)*(HamWin[d-128]);
        d++;
    }
    if (d==128)
    {
        flag1=1;
    }
    if (d==256)
    {

```

```

        flag2=1;
        d=0;
    }

    return ;
}

```

```

void startCpu(void)
{
    POWKEY1=0x01;
    POWCON=0x00; //CPU στα 41,78MHz
    POWKEY2=0xF4;
    return ;
}

```

```

void HammingWindow (void) {
    int x;

    for (x=0;x<128;x++)
    {
        HamWin[x]=0.54-(0.46*cos((2*pi*x)/(N-1)));
    }
    return ;
}

```

```

void delay (int length)
{

```

```

while (length >=0)
length--;
}

void ADCpoweron (int time)
{
    ADCCON = 0x20;
    while (time >=0)    // wait for ADC to be fully powered on
        time--;
}

void fix_fft (short fr[], short fi[], int m)
{
    int mr,nn,i,j,l,k,istep, n;
    short qr,qi,tr,ti,wr,wi;

    n = 1<<m;

    mr = 0;
    nn = n - 1;

    /* decimation in time - re-order data/bit reversal */
    for (m=1; m<=nn; ++m)
    {
        l = n;
        do
        {
            l >>= 1;

```

```

    } while(mr+1 > nn);

    mr = (mr & (1-1)) + 1;

    if(mr <= m) continue;

    tr = fr[m];
    fr[m] = fr[mr];
    fr[mr] = tr;

}

l = 1;
k = LOG2_N_WAVE-1;
while(l < n) {
    istep = 1 << 1;
    for(m=0; m<l; ++m) {
        j = m << k;

        wr = Sinewave[j+N_WAVE/4]; //cos//
        wi = -Sinewave[j];

        wr >>= 1;
        wi >>= 1;

        for(i=m; i<n; i+=istep) {
            j = i + 1;
            tr = fix_mpy(wr,fr[j]) - fix_mpy(wi,fi[j]);
            ti = fix_mpy(wr,fi[j]) + fix_mpy(wi,fr[j]);
            qr = fr[i];

```

```

        qi = fi[i];
        qr >>= 1;
        qi >>= 1;

        fr[j] = qr - tr;
        fi[j] = qi - ti;
        fr[i] = qr + tr;
        fi[i] = qi + ti;

    }

}

    --k;
    l = istep;
}

return ;
}

```

*/*fix_mpy() - fixed-point multiplication*/*

```

short fix_mpy (short a, short b)
{
    FIX_MPY(a,a,b);
    return a;
}

```

Αφού κάναμε βελτιστοποίηση στον κώδικα μέσα από το Keil (optimization level 2) στη συνέχεια τον τρέξαμε πάλι χρησιμοποιώντας τον simulator του συγκεκριμένου περιβάλλοντος και διαπιστώσαμε ότι ο χρόνος που χρειάζεται ο fixed-point FFT για να επεξεργαστεί τα δεδομένα είναι 5,1ms και αν

συμμεριστούμε ότι ο υπολογισμός στη συνέχεια της ενέργειας σε κάθε συχνότητα διαρκεί περίπου 0,2ms αυτό μας δίνει ένα συνολικό χρόνο επεξεργασίας 5,3ms. Όπως έχει ήδη επισημανθεί προκειμένου το πρόγραμμά μας να ανταποκρίνεται στις απαιτήσεις αυτής της εφαρμογής ο χρόνος αυτός δεν θα πρέπει να υπερβαίνει τα 2,137ms που είναι ο χρόνος που απαιτείται για να γίνει η δειγματοληψία ενός πακέτου 128 τιμών με συχνότητα δειγματοληψίας 60KHz. Ενώ λοιπόν καταφέραμε να αυξήσουμε σημαντικά την ταχύτητα επεξεργασίας των δεδομένων - από 73ms μειώθηκε σε μόλις 5,3ms - η βελτίωση αυτή δεν επαρκεί ώστε να μπορεί ο κώδικας που μόλις παραθέσαμε να χρησιμοποιηθεί στη συγκεκριμένη εφαρμογή. Προκύπτει επομένως για άλλη μια φορά η ανάγκη τροποποίησης του ώστε να μπορέσουμε να μειώσουμε κι άλλο το χρόνο επεξεργασίας των τιμών που ψηφιοποιούνται από τον ADC. Γνωρίζοντας όμως ότι ο fixed-point FFT αποτελεί την ταχύτερη υλοποίηση του συγκεκριμένου αλγορίθμου προκύπτει πλέον η ανάγκη εύρεσης κάποιου άλλου εργαλείου για το πέρασμα στο πεδίο της συχνότητας. Το εργαλείο αυτό δεν είναι άλλο από τον αλγόριθμο Goertzel, ο οποίος προβάλλει ως η μόνη και καλύτερη εναλλακτική λύση όταν ο FFT δεν μπορεί να καλύψει την ανάγκη για γρήγορη επεξεργασία δεδομένων που επιτάσσει μια εφαρμογή.

ΠΑΡΑΡΤΗΜΑ 3

Μετά και από τη χρησιμοποίηση του fixed-point FFT, η οποία δεν κατάφερε να μειώσει στον επιθυμητό βαθμό το χρόνο επεξεργασίας των δεδομένων η επιλογή πλέον του αλγόριθμου Goertzel για το πρόγραμμά μας ήταν κατά κάποιο τρόπο μονόδρομος. Ο αλγόριθμος Goertzel δεν μας δίνει ολόκληρο το φάσμα συχνοτήτων όπως γίνεται με τον FFT αλλά υπολογίζεται το πλάτος ή η ενέργεια σε συχνότητες οι οποίες είναι προκαθορισμένες. Όσον αφορά τη δική μας εφαρμογή εμείς έχοντας γνώση των εξαρτημάτων που απαρτίζουν τον μηχανικό μας εξοπλισμό είμαστε σε θέση να γνωρίζουμε σε ποιες συχνότητες εμφανίζονται συνιστώσες που έχουν να κάνουν με τη λειτουργία τους αλλά και σε ποιες συχνότητες εμφανίζονται συνιστώσες που οφείλονται αποκλειστικά σε σφάλματα. Έτσι μπορούμε μέσω του Goertzel να υπολογίσουμε την ενέργεια στις συχνότητες αυτές και να τις συγκρίνουμε με τις τιμές αναφοράς.

Εδώ να επισημάνουμε ότι πρώτα χρησιμοποιήθηκε αντί του FFT ο βελτιστοποιημένος Goertzel για την υλοποίηση του προγράμματος, ο οποίος έχει περιγραφεί αναλυτικά σε προηγούμενο κεφάλαιο. Ο συγκεκριμένος αλγόριθμος χρησιμοποιεί σαν είσοδο έναν συντελεστή τύπου float ($2\cos(2\pi f_{\text{target}} / f_s)$) για να καθοριστεί η συχνότητα στην οποία γίνεται ο υπολογισμός της ενέργειας. Για τον υπολογισμό όμως της ενέργειας σε μια τιμή συχνότητας απαιτείται η εκτέλεση N αριθμού εξισώσεων (το N συμβολίζει τον αριθμό των δειγμάτων που αποτελούν κάθε πακέτο) κάθε μία από τις οποίες περιλαμβάνει έναν πολλαπλασιασμό με τον προκαθορισμένο συντελεστή τύπου float ο οποίος αντιστοιχεί στη συγκεκριμένη τιμή συχνότητας. Αυτό σημαίνει ότι για να υπολογιστεί η ενέργεια σε μία μόνο συχνότητα απαιτούνται N τέτοιοι πολλαπλασιασμοί με μια float τιμή και αν υποθέσουμε ότι k είναι ο αριθμός συχνοτήτων στις οποίες θέλουμε να υπολογίσουμε την ενέργεια χρειάζεται να εκτελεστούν $k \times N$ τέτοιοι πολλαπλασιασμοί με τους αντίστοιχους float συντελεστές. Όπως λοιπόν ήταν αναμενόμενο διαπιστώσαμε ότι ούτε μέσω του βελτιστοποιημένου Goertzel καταφέρνουμε να υπολογίσουμε τις ενέργειες σε έναν επαρκή αριθμό συχνοτήτων σε χρόνο μικρότερο απ' όσο απαιτεί η συλλογή ενός νέου πακέτου δειγμάτων. Άρα προέκυψε η ανάγκη να τροποποιήσουμε τον αλγόριθμο κατάλληλα ώστε να αυξήσουμε την ταχύτητα επεξεργασίας των δειγμάτων.

Το πρώτο που έπρεπε να γίνει ήταν να αντικαταστήσουμε τους συντελεστές τύπου float με άλλους οι τιμές των οποίων θα επεξεργάζονται πιο γρήγορα από τον μικροελεγκτή μας. Για το λόγο αυτό επιλέξαμε, επειδή αυτός ο συντελεστής παίρνει τιμές από -2 έως 2, να δημιουργήσουμε 17 τέτοιους συντελεστές οι οποίοι διαφέρουν μεταξύ τους κατά 0,25. Θεωρώντας λοιπόν πρώτο το -2, ο δεύτερος θα είναι ο -1,75, ο τρίτος ο -1,5 κοκ μέχρι τον 17ο ο οποίος θα είναι το 2. Αυτό το κάναμε ώστε να μπορέσουμε να αξιοποιήσουμε την ταχύτητα με την οποία ένας επεξεργαστής, όπως είναι και αυτός του μικροελεγκτή μας, εκτελεί shift σε μια τιμή. Έτσι αν θέλουμε για παράδειγμα να πολλαπλασιάσουμε μια τιμή με το 2 ή το 0,5 αρκεί να κάνουμε shift στην τιμή αυτή μία θέση αριστερά ή δεξιά αντίστοιχα ενώ αν θέλουμε να πολλαπλασιάσουμε π.χ. με το 1,5 αρκεί να πολλαπλασιάσουμε πρώτα με το 2 (shift μία θέση αριστερά) και στη συνέχεια να αφαιρέσουμε το αποτέλεσμα του πολλαπλασιασμού της τιμής αυτής με το 0,5 (shift μία θέση δεξιά).

Επίσης στηριζόμενοι στο ότι ο αλγόριθμος Goertzel δεν απαιτεί τη συλλογή όλου του πακέτου δειγμάτων για να μπορεί να εκτελεστεί, αυτός καλείται κάθε φορά που παράγεται ένα ψηφιοποιημένο δείγμα στην έξοδο του ADC. Η τιμή του κάθε δείγματος που συλλέγεται χρησιμοποιείται στον υπολογισμό των 17 εξισώσεων που αντιστοιχούν στις 17 διαφορετικές τιμές συχνοτήτων στις οποίες θέλουμε να υπολογίσουμε την ενέργεια. Εδώ πρέπει να επισημανθεί ότι στον κώδικα που υλοποιήσαμε δεν υπολογίζουμε ακριβώς την ενέργεια σε κάθε συχνότητα αλλά την τετραγωνική της ρίζα κι αυτό γιατί θέλαμε να αποφύγουμε πιθανή υπερχειλίση στο αποτέλεσμα.

Έτσι στην μεταβλητή eG προσθέτουμε κάθε φορά την τιμή v και όχι την v^2 . Επιπλέον αντί για 128 δείγματα που χρησιμοποιήσαμε στις προηγούμενες υλοποιήσεις στη συγκεκριμένη επιλέξαμε να χρησιμοποιήσουμε 64 αν και δεν ήταν απαραίτητο ο αριθμός αυτός να είναι δύναμη του 2 μιας και ο Goertzel σε αντίθεση με τον FFT δεν το επιβάλλει. Κάθε φορά που συλλέγεται και το 64ο δείγμα συγκρίνουμε τις τιμές των ενεργειών για τις 17 διαφορετικές τιμές συχνοτήτων με μια τιμή αναφοράς (threshold) και αν κάποια τιμή από αυτές την υπερβαίνει ανάβει ένα led σαν ένδειξη. Αυτό γίνεται με τη βοήθεια των θυρών GPIO (General Input-Output Ports). Στη συνέχεια μηδενίζουμε όλες τις μεταβλητές και εκτελούμε την διαδικασία από την αρχή, επεξεργαζόμενοι ουσιαστικά με αυτόν τον τρόπο πακέτα των 64 δειγμάτων. Η τιμή αναφοράς καθώς και οι τιμές των συχνοτήτων που καθορίζονται από τους 17 συντελεστές που ορίσαμε εξαρχής είναι καθαρά ενδεικτικές και χρησιμοποιούνται για να επιβεβαιώσουμε ότι ο κώδικάς μας λειτουργεί όπως πρέπει.

```
#include <ADuC7026.h>                                //Include ADuC7026 Header File

#include <math.h>

#include <stdio.h>

#include <stdlib.h>


#define BIT0 1
#define BIT1 (1<<1)
#define BIT2 (1<<2)
#define BIT3 (1<<3)
#define BIT4 (1<<4)
#define BIT5 (1<<5)
#define BIT6 (1<<6)
#define BIT7 (1<<7)
#define BIT8 (1<<8)
#define BIT9 (1<<9)
#define BIT10 (1<<10)
#define BIT11 (1<<11)
#define BIT12 (1<<12)
#define BIT13 (1<<13)
```

```

#define BIT14 (1<<14)
#define BIT15 (1<<15)
#define BIT16 (1<<16)
#define BIT17 (1<<17)
#define BIT18 (1<<18)
#define BIT19 (1<<19)
#define BIT20 (1<<20)
#define BIT21 (1<<21)
#define BIT22 (1<<22)
#define BIT23 (1<<23)
#define BIT24 (1<<24)
#define BIT25 (1<<25)
#define BIT26 (1<<26)
#define BIT27 (1<<27)
#define BIT28 (1<<28)
#define BIT29 (1<<29)
#define BIT30 (1<<30)
#define BIT31 (1<<31)

```

```

#define N 64 //N=64
#define PI 3.14159265
#define threshold 1000000 // τιμή αναφοράς

```

```

volatile int eG0, eG1, eG2, eG3, eG4, eG5, eG6, eG7, eG8, eG9, eG10, eG11,
eG12, eG13, eG14, eG15, eG16; /* μεταβλητές στις οποίες αποθηκεύεται η
τετραγωνική ρίζα της ενέργειας που αντιστοιχεί σε κάθε συχνότητα */

```

```

volatile int G0, G1, G2, G3, G4, G5, G6, G7, G8, G9, G10, G11, G12, G13, G14,
G15, G16; /* περιέχουν την τιμή της εξόδου ν μία χρονική στιγμή πριν */

```

```
volatile int  pG0, pG1, pG2, pG3, pG4, pG5, pG6, pG7, pG8, pG9, pG10, pG11,
pG12, pG13, pG14, pG15, pG16; /* περιέχουν την τιμή της εξόδου ν δύο χρονικές
στιγμές πριν */
```

```
volatile short NewData; // γίνεται 1 κάθε φορά που ψηφιοποιείται ένα δείγμα
```

```
volatile int ADCDATA; /* μεταβλητή στην οποία αποθηκεύεται το αποτέλεσμα της
analog-to-digital μετατροπής */
```

```
unsigned long int iT; /* μετράει πόσα δείγματα έχουν ψηφιοποιηθεί και κάθε
φορά που ψηφιοποιείται και το 64ο μηδενίζεται */
```

```
void goertzel(void); // τροποποιημένος Goertzel
```

```
void zeroEne( void ); /* καλείται κάθε φορά που ψηφιοποιείται και το 64ο
δείγμα και μηδενίζει όλες τις μεταβλητές που χρησιμοποιούνται στον αλγόριθμο
Goertzel καθώς και τον μετρητή iT */
```

```
// *****
```

```
void zeroEne( void )
```

```
{
```

```
    eG0 = 0 ;
```

```
    G0 = 0 ;
```

```
    pG0 =0;
```

```
    eG1 = 0 ;
```

```
    G1=0;
```

```
    pG1=0;
```

```
    eG2 = 0 ;
```

```
    G2=0;
```

```
    pG2=0;
```

```
    eG3 = 0 ;
```

```
    G3=0;
```

```
    pG3=0;
```

eG4 = 0 ;
G4=0;
pG4=0;
eG5 = 0 ;
G5=0;
pG5=0;
eG6 = 0 ;
G6=0;
pG6=0;
eG7 = 0 ;
G7=0;
pG7=0;
eG8 = 0 ;
G8=0;
pG8=0;
eG9 = 0 ;
G9=0;
pG9=0;
eG10 = 0 ;
G10=0;
pG10=0;
eG11 = 0 ;
G11=0;
pG11=0;
eG12 = 0 ;
G12=0;
pG12=0;
eG13 = 0 ;

```

        G13=0;

        pG13=0;

        eG14 = 0 ;

        G14=0;

        pG14=0;

        eG15 = 0 ;

        G15=0;

        pG15=0;

        eG16 = 0 ;

        G16=0;

        pG16=0;


GP0DAT=0xFF000000; // all pins of port 0 configured as output
GP1DAT=0xFF000000; // all pins of port 1 configured as output
GP2DAT=0x1000000; // pin 0 of port 2 configured as output

iT=0 ;
}


// *****

int main (void) {

    NewData=0;

    zeroEne( ) ;


    ADCCP = 0x00; //επιλογή θετικού καναλιού 0
    ADCCON = 0x6A2; // start conversion on timer 0

    // ADC Config: fADC/2, acq. time = 8 clocks

    REFCON = 0x01; // connect internal 2.5V reference to VREF pin

```

```
/* χρησιμοποιούμε fast interrupt request FIQ αντί για normal interrupt request  
IRQ */
```

```
    FIQEN = ADC_BIT;                // Enable ADC IRQ ( 0x80 )
```

```
    // timer0 configuration
```

```
    T0LD= 0x2B3; // ADC 60KHz
```

```
    T0CON=0xC0;    //periodic mode
```

```
while(1)
```

```
{
```

```
    if ( NewData )
```

```
    {
```

```
        NewData = 0 ;
```

```
        goertzel();
```

```
        iT++ ;
```

```
        if(iT == N)
```

```
        {
```

```
    // σύγκριση ενεργειών με την τιμή αναφοράς
```

```
        if (eG0 >= threshold)
```

```
        {
```

```
            GP0DAT = BIT16 ;
```

```
        }
```

```
        if (eG1 >= threshold)
```

```
        {
```

```
            GP0DAT = BIT17 ;
```

```
        }
```

```

if (eG2 >= threshold)
{
    GP0DAT = BIT18 ;
}

if (eG3 >= threshold)
{
    GP0DAT = BIT19 ;
}

if (eG4 >= threshold)
{
    GP0DAT = BIT20 ;
}

if (eG5 >= threshold)
{
    GP0DAT = BIT21 ;
}

if (eG6 >= threshold)
{
    GP0DAT = BIT22 ;
}

if (eG7 >= threshold)
{
    GP0DAT = BIT23 ;
}

if (eG8 >= threshold)
{
    GP1DAT = BIT16 ;
}

```

```

if (eG9 >= threshold)
{
    GP1DAT = BIT17 ;
}
if (eG10 >= threshold)
{
    GP1DAT = BIT18 ;
}
if (eG11 >= threshold)
{
    GP1DAT = BIT19 ;
}
if (eG12 >= threshold)
{
    GP1DAT = BIT20 ;
}
if (eG13 >= threshold)
{
    GP1DAT = BIT21 ;
}
if (eG14 >= threshold)
{
    GP1DAT = BIT22 ;
}
if (eG15 >= threshold)
{
    GP1DAT = BIT23 ;
}

```

```

        if (eG16 >= threshold)
        {
            GP2DAT = BIT16 ;
        }

        zeroEne() ;
    }
}
}

/*****

/*          Interrupt Service Routine          */
/*****

// ****

__irq void FIQ_Handler()
{
    ADCDATA = (ADCDAT >> 16);
    NewData = 1 ;
}

// ****

void goertzel() /* τροποποιημένος Goertzel- οι συχνότητες στις οποίες
υπολογίζεται κάθε φορά η ενέργεια φαίνονται πιο κάτω */
{

int s,v;

```

```

s = ADCDATA ;
v = s-(G0<<1)-pG0;
eG0 += abs(v) ;                               //ftarget=30KHz
pG0 = G0 ;
G0 = v ;

v = s-(G1<<1)+(G1>>2)-pG1;
eG1 += abs(v) ;                               //ftarget=25.185KHz
pG1 = G1 ;
G1 = v ;

v = s-G2-(G2>>1)-pG2;
eG2 += abs(v) ;                               //ftarget=23.11KHz
pG2 = G2 ;
G2 = v ;

v = s-G3-(G3>>2)-pG3;
eG3 += abs(v) ;                               //ftarget=21.458KHz
pG3 = G3 ;
G3 = v ;

v = s-G4-pG4;
eG4 += abs(v) ;                               //ftarget=20KHz
pG4 = G4 ;
G4 = v ;

v = s-G5+(G5>>2) - pG5;
eG5 += abs(v) ;                               //ftarget=18.68KHz

```

pG5 = G5 ;

G5 = v ;

v = s-(G6>>1) -pG6;

eG6 += abs(v) ; //ftarget=17.42KHz

pG6 = G6 ;

G6 = v ;

v = s-(G7>>2) - pG7;

eG7 += abs(v) ; //ftarget=16.2KHz

pG7 = G7 ;

G7 = v ;

v = s - pG8;

eG8 += abs(v) ; //ftarget=15KHz

pG8 = G8 ;

G8 = v ;

v = s+(G9>>2) - pG9;

eG9 += abs(v) ; //ftarget=13.81KHz

pG9 = G9 ;

G9 = v ;

v = s+(G10>>1) - pG10;

eG10 += abs(v) ; //ftarget=12.59KHz

pG10 = G10 ;

G10 = v ;

```

v = s+G11 - (G11>>2) - pG11;
eG11 += abs(v) ;                                //ftarget=11.335KHz
pG11 = G11 ;
G11 = v ;

v = s+G12 -pG12;
eG12 += abs(v) ;                                //ftarget=10KHz
pG12 = G12 ;
G12 = v ;

v = s+G13+(G13>>2) -pG13;
eG13 += abs(v) ;                                //ftarget=8.557KHz
pG13 = G13 ;
G13 = v ;

v = s+G14+(G14>>1) -pG14;
eG14 += abs(v) ;                                //ftarget=6.9KHz
pG14 = G14 ;
G14 = v ;

v = s+(G15<<1)-(G15>>2) -pG15;
eG15 += abs(v) ;                                //ftarget=4.828KHz
pG15 = G15 ;
G15 = v ;

v = s+(G16<<1) -pG16;
eG16 += abs(v) ;                                //ftarget=0KHz
pG16 = G16 ;

```

G16 = v ;

}

Για να προσομοιώσουμε το σήμα που προκύπτει από τις δονήσεις των μηχανικών μερών της μηχανής χρησιμοποιήσαμε ένα ημίτονο με συχνότητα μία τιμή από τις target frequencies που ανιχνεύονται από τον αλγόριθμο Goertzel του προγράμματός μας. Αν και όπως έχει ήδη αναφερθεί ένα τέτοιο σήμα μπορεί να περιέχει συχνότητες μέχρι και 30 KHz εμείς επιλέξαμε σαν συχνότητα τα 6,9 KHz, χωρίς όμως αυτό να σημαίνει ότι δεν θα μπορούσαμε να επιλέξουμε μια μεγαλύτερη τιμή. Από τη στιγμή που έχουμε ορίσει μέσα στον κώδικα σαν συχνότητα δειγματοληψίας τα 60 KHz βάσει του θεωρήματος Nyquist μπορούμε να δειγματοληπτήσουμε επιτυχώς σήματα με συχνότητα έως και 30 KHz. Αν επιλέγαμε ένα ημίτονο με συχνότητα διαφορετική από τις τιμές συχνότητας που ανιχνεύονται από τον αλγόριθμο Goertzel θα παρατηρούσαμε το φαινόμενο leakage μιας και θα εμφανίζονταν μεγάλες τιμές πλάτους στις target frequencies που είναι πιο κοντά στην τιμή συχνότητας που επιλέξαμε για το ημίτονό μας. Το Keil μvision4, που είναι το περιβάλλον ανάπτυξης στο οποίο δουλεύουμε, μας δίνει τη δυνατότητα καθώς τρέχουμε τον κώδικα της εφαρμογής μας να θέσουμε ως είσοδο σε διάφορα κανάλια συναρτήσεις που έχουμε υλοποιήσει. Έτσι λοιπόν δημιουργήσαμε ένα αρχείο μέσω της πλοήγησης debug →function editor→signal.ini., το οποίο αποτελεί ένα initial αρχείο όπως υποδηλώνει και η επέκταση του μέσα στο οποίο γράφουμε τον κώδικα που υλοποιεί μία ημιτονοειδή συνάρτηση με κάποια χαρακτηριστικά. Αυτά τα χαρακτηριστικά είναι η διάρκεια του συνολικού σήματος, η συχνότητα του ημιτόνου και το πλάτος του. Εμείς για τη συγκεκριμένη εφαρμογή ορίσαμε ένα ημίτονο με συχνότητα 6,9 KHz, εύρος τιμών από 0.2 έως 2.2 (προσθήσαμε μια DC συνιστώσα) και διάρκειας 100s. Αυτό το σήμα το θέτουμε σαν είσοδο στο κανάλι 0 του Analog-to Digital Converter του μικροελεγκτή μας.

```
SIGNAL void ADC0_Sine (void) {  
  
    float volts;  
    float frequency;  
    float offset;  
    float duration;  
    float val;  
    long i, end;  
  
    volts      = 1.0;  
    offset     = 1.2;  
    frequency = 6900;  
    duration  = 100;  
  
    printf ("Sine Wave Signal on AD Channel 0.\\n");  
  
    end = (duration * 100000);  
    for (i = 0 ; i < end; i++) {  
        val = __sin (frequency * (((float) STATES) / CLOCK) * 2 * 3.1415926);
```

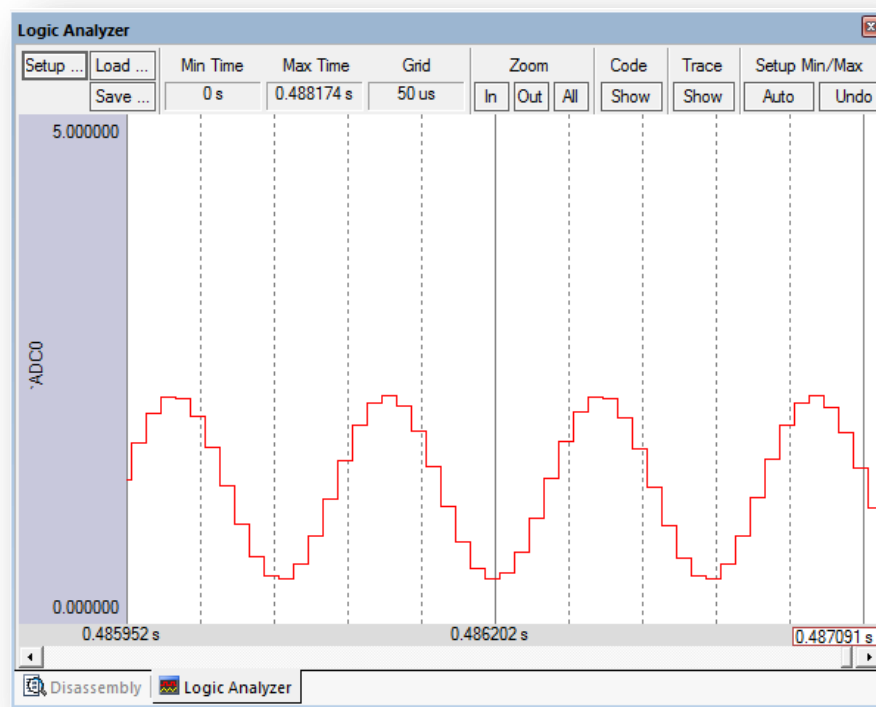
```

    ADC0 = (val * volts) +offset;
    swatch (0.00001);          // in 10 uSec steps
}
}

```

Η σύνδεση του βασικού προγράμματος με αυτό μπορεί επίσης να γίνει πριν μπούμε σε debug mode μέσω της επιλογής Project→ Options for Target 7026→Debug. Εκεί, αφενός κλικάρουμε την επιλογή 'Simulator' που μας δίνει τη δυνατότητα να εξομοιώσουμε με μηδενικό υλικό και αφετέρου βρίσκουμε το .αρχείο που έχουμε ήδη δημιουργήσει και το επιλέγουμε.

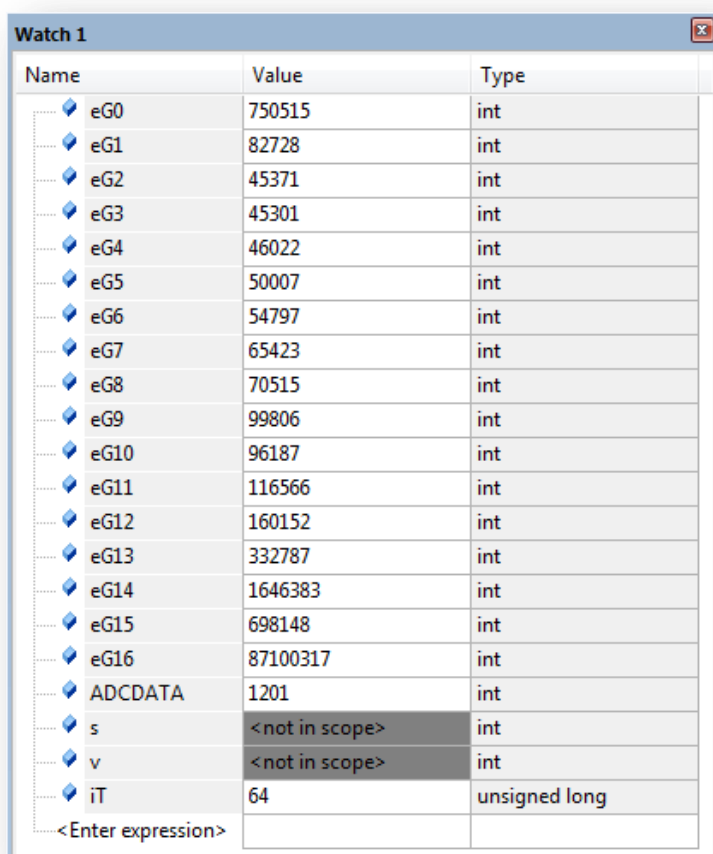
Ενώ πλέον είμαστε σε debug mode μπορούμε μέσω της επιλογής View→Analysis Window→ Logic Analyzer να παρατηρήσουμε τα σήματα που αναπαριστούν εισόδους, μεταβλητές και registers του βασικού μας προγράμματος. Εμείς μέσω του setup→ New(insert)→ ADC0 επιλέγουμε να εμποπτεύσουμε την είσοδο ADC0 του μικροελεγκτή μας. Επίσης μέσω του command window δημιουργούμε στο toolbox ένα button για τη συνάρτηση με την εντολή: DEFINE BUTTON "ADC0 Sin","ADC0_Sine()". Στη συνέχεια κάνοντας κλικ στο κουμπί που δημιουργείται στο toolbox τροφοδοτούμε την είσοδο ADC0 με το ημίτονο που δημιουργήσαμε. Πιο κάτω παραθέτουμε το σήμα εισόδου όπως φαίνεται στο παράθυρο του Logic Analyzer.



Είναι γνωστό ότι όλη η ενέργεια ενός σήματος ημιτόνου συμπεριλαμβάνεται σε μία μόνο τιμή συχνότητας που είναι και η βασική συχνότητά του. Έτσι το φάσμα συχνοτήτων ενός ημιτονοειδούς σήματος αναπαρίσταται από μία συνιστώσα στην κύρια συχνότητά του. Το φάσμα συχνοτήτων ενός σήματος όπως αυτό που χρησιμοποιήσαμε συχνότητας 6,9 KHz, το οποίο είναι μετατοπισμένο στον άξονα y με τη χρήση μίας DC συνιστώσας αποτυπώνεται πλήρως με την παρουσία δύο συνιστωσών μία στην συχνότητα $f=0$ Hz και μία στην τιμή $f=6,9$ KHz. Συνεπώς αν θέλουμε να

επιβεβαιώσουμε ότι το πρόγραμμά μας λειτουργεί σωστά θα πρέπει οι μεταβλητές eG14 ($f=6,9$ KHz) και eG16 ($f=0$) μετά την επεξεργασία κάθε πακέτου 64 δειγμάτων να λάβουν μία μεγάλη τιμή συγκριτικά με τις τιμές πλάτους στις υπόλοιπες τιμές συχνοτήτων.

Ενώ είμαστε πάλι σε debug mode μπορούμε να επιτηρήσουμε τις τιμές που λαμβάνει οποιαδήποτε μεταβλητή χρησιμοποιείται από τον κώδικά μας μέσω των Watch παραθύρων που μας προσφέρει το Keil. Αυτό το επιτυγχάνουμε αν επιλέξουμε διαδοχικά View→Watch Windows→Watch 1 or 2. Στο παράθυρο που ανοίγει κάνοντας διπλό κλικ στο “Enter expression” και εισάγοντας το όνομα μιας μεταβλητής που συμπεριλαμβάνεται στον κώδικά μας μπορούμε να βλέπουμε πως αυτή αλλάζει τιμές κατά τη διάρκεια εκτέλεσης του κώδικα. Πράγματι μετά από την επεξεργασία και των 64 δειγμάτων παρατηρούμε πως η μεταβλητή eG16 έχει λάβει μία πολύ μεγάλη τιμή, πράγμα που υποδεικνύει πολύ μεγάλο πλάτος στη συχνότητα $f=0$ Hz, ενώ η μεταβλητή eG14 έχει λάβει την αμέσως μεγαλύτερη τιμή, το οποίο με τη σειρά του υποδεικνύει μεγάλο πλάτος στη συχνότητα $f=6,9$ KHz.



Name	Value	Type
eG0	750515	int
eG1	82728	int
eG2	45371	int
eG3	45301	int
eG4	46022	int
eG5	50007	int
eG6	54797	int
eG7	65423	int
eG8	70515	int
eG9	99806	int
eG10	96187	int
eG11	116566	int
eG12	160152	int
eG13	332787	int
eG14	1646383	int
eG15	698148	int
eG16	87100317	int
ADCDATA	1201	int
s	<not in scope>	int
v	<not in scope>	int
iT	64	unsigned long
<Enter expression>		

Εφόσον έχουμε υπολογίσει το πλάτος σε κάθε συχνότητα με τις τιμές να είναι αυτές που φαίνονται πιο πάνω μένει να ελέγξουμε αν κάποια από αυτές είναι μεγαλύτερη από την τιμή αναφοράς που έχουμε ορίσει και αν στην περίπτωση αυτή ανάβει το αντίστοιχο led σαν ένδειξη βλάβης. Έτσι έχοντας θέσει σαν threshold την τιμή 1000000 παρατηρώντας τις παραπάνω τιμές θα πρέπει να ανάψει ένα led τόσο κατά τη σύγκριση της τιμής αυτής με την τιμή της μεταβλητής eG14 όσο και κατά τη σύγκρισή της με την τιμή της μεταβλητής eG16. Πιο συγκεκριμένα όπως προκύπτει από τον

κώδικα θα πρέπει να γίνει 1 τόσο το pin6 της θύρας 1(bit22) όσο και το pin 0 της θύρας 2 (bit 16).
Πράγματι...

