ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ
Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΛΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΟΣ ΤΗΛΕΧΕΙΡΙΣΜΟΥ ΠΑΝΩ ΑΠΟ ΔΙΚΤΥΟ GSM

ΕΙΣΗΓΗΤΗΣ: ΜΙΑΟΥΛΑΚΗΣ ΑΝΔΡΕΑΣ
ΜΕΛΕΤΗ-ΑΝΑΠΤΥΞΗ: ΔΑΣΚΑΛΑΚΗΣ ΜΑΝΟΣ
ΗΡΑΚΛΕΙΟ 2011

ΠΕΡΙΛΗΨΗ

Με τους σημερινούς ρυθμούς ζωής αλλά και τη συνεχόμενη βελτίωση της τεχνολογίας, η ιδέα της εύκολης διαχείρισης των συσκευών που χρησιμοποιούμε έχει γίνει επιτακτική. Ειδικά όταν αυτή η ευκολία παρέχεται οποτεδήποτε ή οπουδήποτε εμείς την χρειαστούμε μας βοηθά να οργανωνόμαστε καλύτερα, μας κάνει περισσότερο αυτόνομους και αξιοποιούμε στο έπακρο τις δυνατότητες της συσκευής. Έτσι λοιπόν σκοπός αυτής της πτυχιακής εργασίας είναι η ανάπτυξη μιας εφαρμογής βασισμένης σε μία διάταξη μικροελεγκτή. Ποιό συγκεκριμένα αναπτύσσεται ένα σύστημα ελέγχου που θα δίνει την δυνατότητα στο χρήστη να ενεργοποιεί ή να απενεργοποιεί την συσκευή που συνδέεται με αυτό. Ιδιαίτερο χαρακτηριστικό της εφαρμογής αποτελεί η σύνδεση του μικροελεγκτή με ένα gsm modem και έτσι τη χρήση της κυψελοειδούς τηλεφωνίας gsm. Μπορεί λοιπόν ο χρήστης απομακρυσμένα να διαχειριστεί την συσκευή οπουδήποτε και αν βρίσκεται κάνοντας χρήση σύντομων γραπτών μηνυμάτων. Για την περάτωση της πτυχιακής εξετάστηκαν οι λειτουργίες του gsm δικτύου που αφορούν την αποστολή και τη λήψη γραπτών μηνυμάτων, το πλαίσιο της pdu ακολουθίας χαρακτήρων ενός γραπτού μηνύματος καθώς και οι αλγόριθμοι για την κωδικοποίηση και αποκωδικοποίηση που χρησιμοποιούνται.
Για την υλοποίηση χρησιμοποιήθηκαν ο μικροελεγκτής PIC18F2520 της Microchip Inc ο οποίος ενσωματώθηκε στην πλακέτα PIC-MT της εταιρίας Olimex Ltd. Καθώς και το gsm modem από την Wavecom Instruments. Ο προγραμματισμός έγινε στην γλώσσα προγραμματισμού c.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ:

Μικροελεγκτής, σύστημα ελέγχου,gsm modem, σύντομα γραπτά μηνύματα (sms)

ABSTRACT

At the current pace of life and the continuous improvement of technology, the idea of easy management of equipment we use has become imperative. Especially when this management is provided whenever or wherever we need it, it help us be better organized, make us more independent and utilize the full potential of the device. So the aim of this thesis is to develop an application based on a microcontroller device. Specifically is developing a monitoring system that allows the user to enable or disable the device connected to it. A special feature of the application is connection of the microcontroller with a gsm modem, so its possible to use gsm cellular phones. So the user can remotely manage the device just using short messages (sms). For the completion of the thesis examined the functions of gsm network on sending and receiving text messages, the pdu format of the sort message and the algorithms for encoding and decoding it.
For the implementation I used the PIC18F2520 microcontroller from Microchip Inc which was incorporated into the PCB PIC-Mt of the Olimex Ltd company. As well as a gsm modem from Wavecom Instruments. The programming was done in c programming language.

KEY WORDS:

system control, microcontroller, gsm modem sms (short message service)

ΑΔΕΙΑ ΧΡΗΣΗΣ

ΠΕΡΙΛΗΨΗ **2**
ΕΙΣΑΓΩΓΗ **7**
ΚΕΦΑΛΑΙΟ 1 - ΠΕΡΙΓΡΑΦΗ ΣΥΣΤΗΜΑΤΟΣ **8**
Γενική περιγραφή συστήματος 8
Περιγραφή υλικού συστήματος 9
Μικροελεγκτής 9
Πλακέτα μικροελεγκτή 11
Gsm Modem module 12
Διασύνδεση RS232 (σειριακή) 13
Περιγραφή εργαλείων λογισμικού 14
Αναπτυξιακό περιβάλον 14
Προγραμματιστής 15
ΚΕΦΑΛΑΙΟ 2 - GSM **16**
Εισαγωγή στην τηλεφωνία κυψέλης 16
Επικοινωνίες πολλαπλής πρόσβασης 19
Πολλαπλή πρόσβαση διαίρεσης συχνότητας 19
Πολλαπλή πρόσβαση διαίρεσης χρόνου 20
Πολλαπλή πρόσβαση διαίρεσης κώδικα 21
Σύντομη αναδρομή 22
Τεχνικές λεπτομέρειες του GSM 23
Η μονάδα ταυτότητας συνδρομητή (SIM) 26
Η ασφάλεια στο GSM 26
Υπηρεσίες συνδρομητή του GSM 26
Υπηρεσία Σύντομων Γραπτών Μηνυμάτων SMS 28
Κωδικοποίηση PDU SMS 30
Αποκωδικοποίηση PDU SMS 31
AT Εντολές 31
AT+CMGS 32
AT+CNMI 32
ATE 32
AT+CMGD 33
AT+CMGF 33
Μορφή αποστολής μηνύματος 33
ΚΕΦΑΛΑΙΟ 3- Μικροελεγκτές οικογένειας PIC **35**
Μικροελεγκτές και Μικροεπεξεργαστές 35
Αρχιτεκτονικές Harvard και Von-Neumann 35
Γενικά χαρακτηριστικά των μικροελεγκτών της οικογένειας PIC 36
Πυρήνας (Core) 36
Περιφερειακά (Peripherals) 36
Ειδικά Χαρακτηριστικά (Special Features) 37
Μικροελεγκτές "μεσαίας" (Mid-Range) οικογένειας 37
Αρχιτεκτονική του PIC 37
Ο πυρήνας του PIC 38
Κεντρική Μονάδα Επεξεργασίας (Central Processing Unit, CPU) 38
Ρολόι - Χρονισμοί - Κύκλος Εντολής 39
Μονάδα συνεχούς διοχέτευσης εντολών (Instruction Pipelining) 40
Αριθμητική Λογική Μονάδα (ALU) 40
Διακοπές (Interrupts) 42
Γενική Αρχιτεκτονική των Διακοπών 42

Τα περιφερειακά του PIC 43
Γενικής χρήσης μονάδες εισόδου-εξόδου I/O (Ports) 43
PORTB - Οι καταχωρητές PORTB και TRISB 44
Χρονιστές (Timers) 44
Timer 0 44
Διαιρέτης Μέτρησης (Prescaler) 46
Πομπός / Δέκτης Ασύγχρονης / Σύγχρονης Σειριακής Επικοινωνίας (USART) 46
Γεννήτρια Ρυθμού μετάδοσης (Baud Rate Generator, BRG) 46
Πομπός ασύγχρονης κατάστασης λειτουργίας της USART (USART Asynchronous Transmitter) 47
Δέκτης ασύγχρονης κατάστασης λειτουργίας της USART (USART Asynchronous Receiver) 48
ΚΕΦΑΛΑΙΟ 4 – ΠΑΡΑΘΕΣΗ ΚΩΔΙΚΑ 50
Αρχείο main.c 51
Κώδικας main.c 55
Αρχείο Defs.h 63
Κώδικας Defs.h 63
Αρχείο Lcd.c: 65
Κώδικας Lcd.c: 65
Αρχείο Serial.c 68
Κώδικας Serial.c 68
Αρχείο Pdu decode.c 70
Κώδικας Pdu decode.c 70
ΠΡΟΤΑΣΕΙΣ 78
ΒΙΒΛΙΟΓΡΑΦΙΑ 79
ΑΝΑΦΟΡΕΣ 79

Εικόνα 1: Ο μικροελεγκτής PIC της Microchip 9
Εικόνα 2: Λογικό διάγραμμα των interrupts 11
Εικόνα 3: Η πλακέτα PIC-MT της Olimex Ltd 11
Εικόνα 4: Σχεδιάγραμμα κυκλώματος 12
Εικόνα 5: Fastrack M1206B της εταιρίας Wavecom Instruments 13
Εικόνα 6: Βύσμα RS-232 13
Εικόνα 7: Αρχιτεκτονική σπονδυλικής στήλης (backbone) 17
Εικόνα 8: Φασματική κατανομή συστήματος διαίρεσης συχνότητας 20
Εικόνα 9: Έκταση της πληροφορίας κατά την άμεση ακολουθία 22
Εικόνα 10: Παγκόσμιος χάρτης κάλυψης 23
Εικόνα 11: Αρχιτεκτονική του PIC 38
Εικόνα 12: Παράδειγμα συνεχούς διοχέτευσης εντολών (pipelining) 40
Εικόνα 13: Αριθμητική Λογική Μονάδα (ALU) του PIC 41
Εικόνα 14: Βασική αρχιτεκτονική timer0 45
Εικόνα 15: Η αρχιτεκτονική του πομπού της USART 48
Εικόνα 16: Η αρχιτεκτονική του δέκτη της USART 49
Εικόνα 17: Κύλιση πινάκων κωδικοποιημένου μηνύματος 53
Εικόνα 18: Αποκωδικοποίηση μηνύματος 53
Εικόνα 19: Κύλιση πινάκων για κωδικοποίηση μηνύματος 54
Εικόνα 20: Κωδικοποίηση μηνύματος 55

ΕΙΣΑΓΩΓΗ

Η πτυχιακή αυτή εργασία πραγματεύεται την ανάπτυξη ενός συστήματος απομακρυσμένου ελέγχου με χρήση σύντομων γραπτών μηνυμάτων SMS. Επικεντρώθηκε στο σχεδιασμό του συστήματος και στην ανάπτυξη του υλικού (hardware) μέρους αλλά και στη δημιουργία του λογισμικού μέρους (software).
Το υλικό μέρος αφορά στην επιλογή του κατάλληλου ηλεκτρονικού εξοπλισμού που να ικανοποιούν τον σχεδιασμό του συστήματος και την σωστή μεταξύ τους σύνδεση. Όσον αφορά στο λογισμικό, έγινε η συγγραφή του απαραίτητου κώδικα για την ορθή λειτουργία του υλικού και τελικά την εκτέλεση των απαιτούμενων εντολών. Καθώς η τεχνολογία εξελίσσεται, τα συστήματα και οι μηχανές γίνονται όλο και πιο περίπλοκα και η ανάγκη για ποιοτικότερο και πιο αξιόπιστο έλεγχο είναι επιτακτική. Αυτή την ανάγκη καλύπτουν τα συστήματα ελέγχου τα οποία αποκωδικοποιούν τις εντολές του χρήστη και αναλαμβάνουν να τις «μεταφράσουν» στα συστήματα που ελέγχονται. Οι εντολές μπορεί να είναι από την ενεργοποίηση ή απενεργοποίηση μιας συσκευής μέχρι και την πλήρη αυτοματοποίηση διαδικασιών βελτιστοποιώντας την απόδοσή της. Αυτός είναι και ο στόχος αυτής της πτυχιακής, δηλαδή ο χρήστης να έχει τη δυνατότητα απομακρυσμένα να απενεργοποιήσει ή να θέσει σε λειτουργία μια συσκευή που θα είναι συνδεμένη στο σύστημα ελέγχου.
Για την λειτουργία και το χειρισμό του συστήματος ελέγχου από τον χρήστη δημιουργήθηκε ένα βασικό ρεπερτόριο εντολών μέσω του οποίου θα παρέχεται η δυνατότητα επικοινωνίας και ελέγχου με τις συσκευές. Παράλληλα αναπτύχθηκε η διαδικασία αποκωδικοποίησης των εντολών αυτών από την πλευρά του συστήματος. Θεωρήθηκε σαν καλύτερος τρόπος για αυτόν τον χειρισμό η χρήση δικτύου κινητής τηλεφωνίας καθώς προσφέρει σχεδόν πλήρη ελευθερία καθώς καλύπτουν σχεδόν το σύνολο της χώρας αλλά θεωρητικά υπάρχει και παγκόσμια εμβέλεια. Επίσης ένας ακόμη σοβαρός λόγος επιλογής του συγκεκριμένου τύπου δικτύου είναι η μη ανάγκη ύπαρξης συγκεκριμένου εξοπλισμού για τον έλεγχο του συστήματος από τη μεριά του χρήστη παρά το κινητό τηλέφωνο, μια συσκευή που σχεδόν όλοι πλέον διαθέτουν.

ΕΥΧΑΡΙΣΤΙΕΣ!

Η διπλωματική εργασία έγινε υπό την επίβλεψη του κυρίου Μιαουδάκη Ανδρέα, διδάκτορα της σχολής Ε.Π.Π. του ΤΕΙ Κρήτης. Θα ήθελα να τον ευχαριστήσω ιδιαίτερα για την πολύτιμη και αμέριστη βοήθεια που προσέφερε κατά τη διάρκεια ανάπτυξής της. Η υπομονή και η στήριξή του ήταν σημαντική και αδιάκοπη.
Η πτυχιακή είναι αφιερωμένη, σε κάποια που επίσης ήταν στήριγμα όλο αυτό τον καιρό. Ήταν δίπλα μου, αλλά στο τέλος εγκατέλειψε!

ΚΕΦΑΛΑΙΟ 1 - ΠΕΡΙΓΡΑΦΗ ΣΥΣΤΗΜΑΤΟΣ

Γενική περιγραφή συστήματος

Για την εργασία χρησιμοποιήθηκε ένας μικροελεγκτής καθώς και ένα modem. Σαν μικροελεγκτής ορίζεται μια υπολογιστική μονάδα ενοποιημένη σε ένα κομμάτι ολοκληρωμένου κυκλώματος. Βέβαια με τον όρο υπολογιστική μονάδα αναφερόμαστε σε βασικές αρχές όπως τη κεντρική μονάδα επεξεργασίας, την μνήμη καθώς και το σύστημα εισόδου και εξόδου της διάταξης. Σε αυτά προστίθενται βέβαια περιφερειακά, χρονιστές (timers), μετατροπείς τάσης (A/D converters) και άλλα, αυξάνοντας τις δυνατότητες του μικροελεγκτή και καθιστώντας τον ιδανικό για μια πλειάδα εφαρμογών. Εφαρμογές που δεν χρειάζονται μεγάλη επεξεργαστική ισχύ αλλά αξιόπιστη λειτουργία, μικρό κόστος παραγωγής και κατανάλωσης ενέργειας. Χαρακτηριστικά λοιπόν των μικροελεγκτών είναι το μικρό τους μέγεθος, το χαμηλό κόστος κατασκευής, η λειτουργία σε συχνότητες χρονισμού έως και μερικών δεκάδων MHz, κατανάλωση μικρών ποσοτήτων ενέργειας ενώ υπάρχει η δυνατότητα να εισέλθουν σε κατάσταση αναμονής ενός γεγονότος, και έτσι μειώνεται ακόμα περισσότερο η κατανάλωση. Έτσι τους συναντάμε σε οικιακές συσκευές ακόμη και σε αυτοκίνητα ή συστήματα αυτοματισμών στη βιομηχανία και την παραγωγή.
Η μονάδα του modem (modulator-demodulator) είναι μια διάταξη η οποία κωδικοποιεί ένα αναλογικό σήμα σε ψηφιακό και το ανάποδο. Μεταφέρεται έτσι ψηφιακή πληροφορία μέσω αναλογικού μέσου. Στη συγκεκριμένη εργασία χρησιμοποιήθηκε ένα gsm modem όπου το αναλογικό μέσο είναι ο αέρας και το δίκτυο κινητής τηλεφωνίας. Τα δίκτυα κινητής τηλεφωνίας λειτουργούν με τη διαδικασία packet switching, χωρίζουν δηλαδή τη σύνδεση σε μικρά χρονικά μέρη όπου κατά τη διάρκεια του κάθε μέρους αποστέλλεται ένα πακέτο πληροφορίας. Έτσι μπορούν να μεταφέρονται φωνή και δεδομένα, εναλλάξ με τόση μικρή διαφορά μεταξύ τους, που σε μας φαίνεται σαν να γίνεται ταυτόχρονα. Τα modem συνήθως κατηγοριοποιούνται με βάση τα πρωτόκολλα που χρησιμοποιούν και κυρίως με τις μέγιστες (θεωρητικές) ταχύτητες δεδομένων που μπορούν να επιτύχουν.
Ο ρόλος του μικροελεγκτή εκτός από της αποκωδικοποίησης των εντολών και της εκτέλεσης τους είναι και αυτός του κινητού τηλεφώνου! Εκείνος είναι υπεύθυνος κυρίως για την αποστολή και λήψη ενός σύντομου γραπτού μηνύματος. Με τον τρόπο αυτό το modem μπορεί να θεωρηθεί μία περιφερειακή μονάδα του μικροελεγκτή και ειδικότερα ένας πομπός – δέκτης με εμβέλεια θεωρητικά σε ολόκληρο τον κόσμο και πρόσβαση σε εκατομμύρια τερματικά! Για τον έλεγχο του συστήματος ο χρήστης δεν έχει παρά να συντάξει ένα σύντομο μήνυμα με τις εντολές ελέγχου και να το αποστείλει στο modem. Ο μικροελεγκτής στην συνέχεια θα αναλάβει την λήψη του μηνύματος θα προχωρήσει στην αποκωδικοποίηση του και θα παράγει τελικά τα απαραίτητα σήματα ελέγχου για να οδηγήσει κατάλληλα τις συσκευές. Η διαδικασία θα ολοκληρωθεί με την αποστολή ενός μηνύματος επιβεβαίωσης από τον μικροελεγκτή μέσω του κινητού τηλεφώνου βάσης προς το χρήστη ότι ο έλεγχος ολοκληρώθηκε με επιτυχία.

Περιγραφή υλικού συστήματος

Τα βασικά μέρη που απαρτίζουν το κύκλωμα της εφαρμογής είναι ο μικροελεγκτής με την πλακέτα του, το modem, οι τροφοδοσίες τους, τα καλώδια για τη μεταξύ τους σύνδεση και η συσκευή που θα συνδέσουμε στην οποία θα γίνεται ο έλεγχος.

Μικροελεγκτής

Η καρδιά του κυκλώματος είναι ο μικροελεγκτής PIC18F2520 της εταιρίας Microchip Inc [1]

Εικόνα 1: Ο μικροελεγκτής PIC της Microchip

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.
Η ονομασία PIC προέρχεται από τα αρχικά των λέξεων Programmable Intelligent Computer. Οι μικροελεγκτές PIC είναι εξαιρετικά διαδεδομένοι εξαιτίας του χαμηλού τους κόστους, της διαθεσιμότητας, του μικρού κόστους των αναπτυξιακών τους εργαλείων, της πληθώρας των δωρεάν σημειώσεων υποστήριξης καθώς και της δυνατότητας σειριακού προγραμματισμού και επαναπρογραμματισμού. Μερικά από τα χαρακτηριστικά τους είναι:

- Ξεχωριστός χώρος για κώδικα και δεδομένα.
- Ένας μικρός αριθμός εντολών συγκεκριμένου μήκους
- Οι περισσότερες εντολές εκτελούνται σε ένα κύκλο εκτέλεσης(4 κύκλοι ρολογιού)
- Καταχωρητής συσσώρευσης (Accumulator)
- Καταχωρητές θυρών, περιφερειακών και επεξεργαστή

Σε αντίθεση με τις περισσότερες μονάδες επεξεργασίας, δεν υπάρχει διάκριση μεταξύ χώρου «μνήμης» και «καταχωρητή» διότι η RAM εξυπηρετεί και τις δύο εργασίες μνήμης και καταχώρησης και συνήθως αναφέρεται ως αρχείο καταχώρησης. Περισσότερες λεπτομέριες για τους μικροελεγκτές σε επόμενο κεφάλαιο.

Επιλέχθηκε ο PIC18F2520 με βάση τις απαιτήσεις του συστήματος και χρησιμοποιείται το κέλυφος με τα 28 ποδαράκια (pins) (τύπος SPDIP)
Βασικά χαρακτηριστικά του είναι:

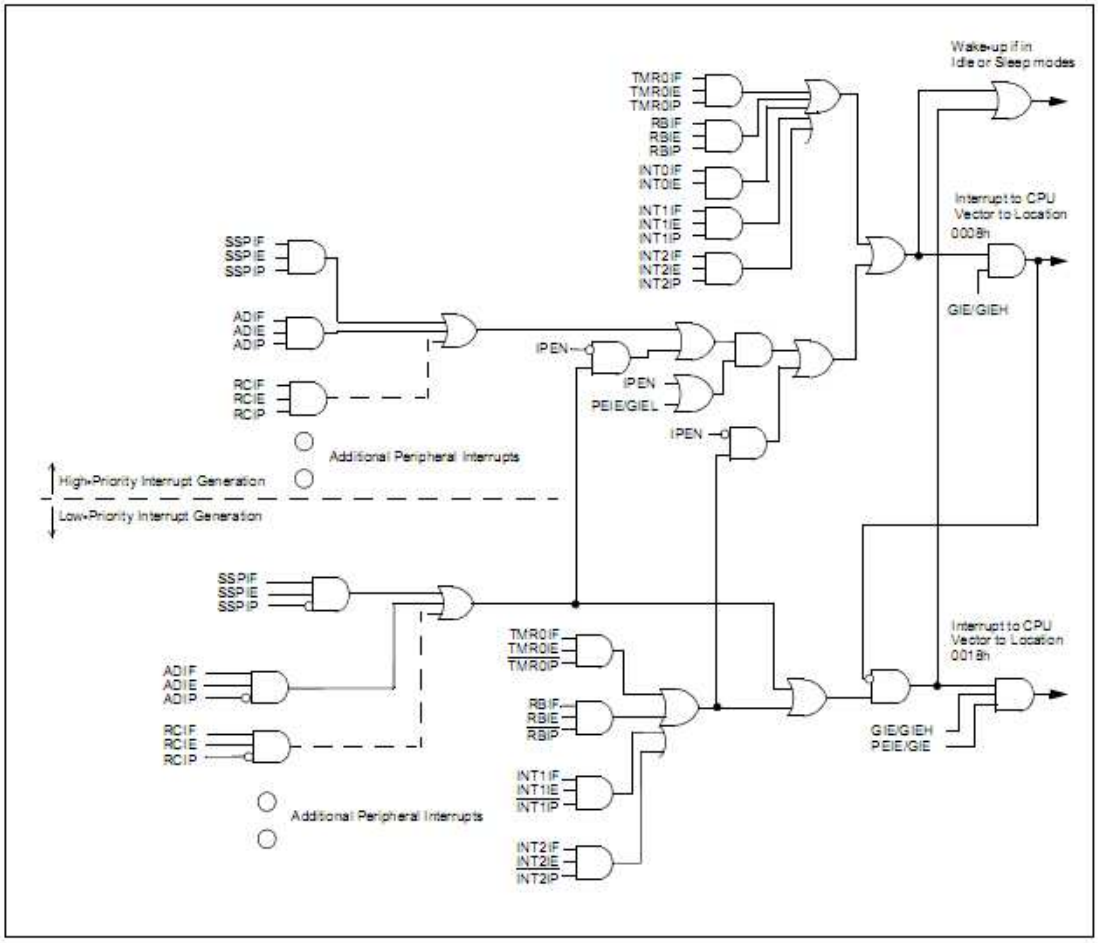
- Ο τύπος της μνήμης για το πρόγραμμα είναι τύπου Flash
- Η ταχύτητα του επεξεργαστή είναι 10 MIPS
- Η μνήμη προγραμματισμού είναι χωρητικότητας 32KB
- Η προσωρινή μνήμη (δεδομένων) είναι χωρητικότητας 1,536byte
- Ενώ η μνήμη EEPROM 256byte
- Διαθέτει ταλαντωτή ευέλικτο στη παραμετροποίηση

Επίσης διαθέτει πληθώρα περιφερειακών όπως

- Προγραμματιζόμενες πόρτες εισόδου – εξόδου
- Τέσσερις Χρονιστές (1x8 bit, 3x16 bit)
- Πομπό- δέκτη σύγχρονης σειριακής επικοινωνίας (USART)
- Μετατροπέα αναλογικού σε ψηφιακό σήμα (Anolog to Digital) και άλλα

Για την λειτουργία του μικροελεγκτή στο κύκλωμα, απαιτείται μόνο η πηγή τροφοδοσίας και ο ταλαντωτής για την παραγωγή του κατάλληλου σήματος χρονισμού. Το σύστημα χρονισμού που χρησιμοποιείται είναι κρυσταλλικού τύπου και παράγεται με την βοήθεια ενός κρυστάλλου των 4MHz.
Έχει γίνει παραμετροποίηση του χρόνου του κρυστάλλου με βάση τις απαιτήσεις της εφαρμογής και ειδικότερα για την ορθή μεταφορά δεδομένων μέσω της σειριακής από τον μικροελεγκτή προς το modem. Ακόμα ένα σημαντικό χαρακτηριστικό του συγκεκριμένου μικροελεγκτή είναι οι διακόπτες (interrupts) που διαθέτει. Εκτός από το πλήθος τους, μπορούν να διακριθούν σε υψηλής η χαμηλής προτεραιότητας. Υπάρχουν δέκα καταχωρητές(registers) για να χρησιμοποιήσουμε τα interrupts. Αυτοί είναι οι:
RCON
INTCON
INTCON2
INTCON3
PIR1, PIR2
PIE1, PIE2
IPR1, IPR2

Στην πτυχιακή γίνεται χρήση των RCON που αφορά τα interrupts που γίνονται από τον πομπό – δέκτη της σειριακής καθώς και κάποια bit από τον INTCON καταχωρητή που ενεργοποιεί ή απενεργοποιεί (ανάλογα αν θα θέσουμε στο bit το λογικό ένα ή μηδέν αντίστοιχα) όλα τα interrupts από όλες τις πηγές (general interrupt GIE).
Ακολουθεί το λογικό διάγραμμα των interrupts του μικροελεγκτή.



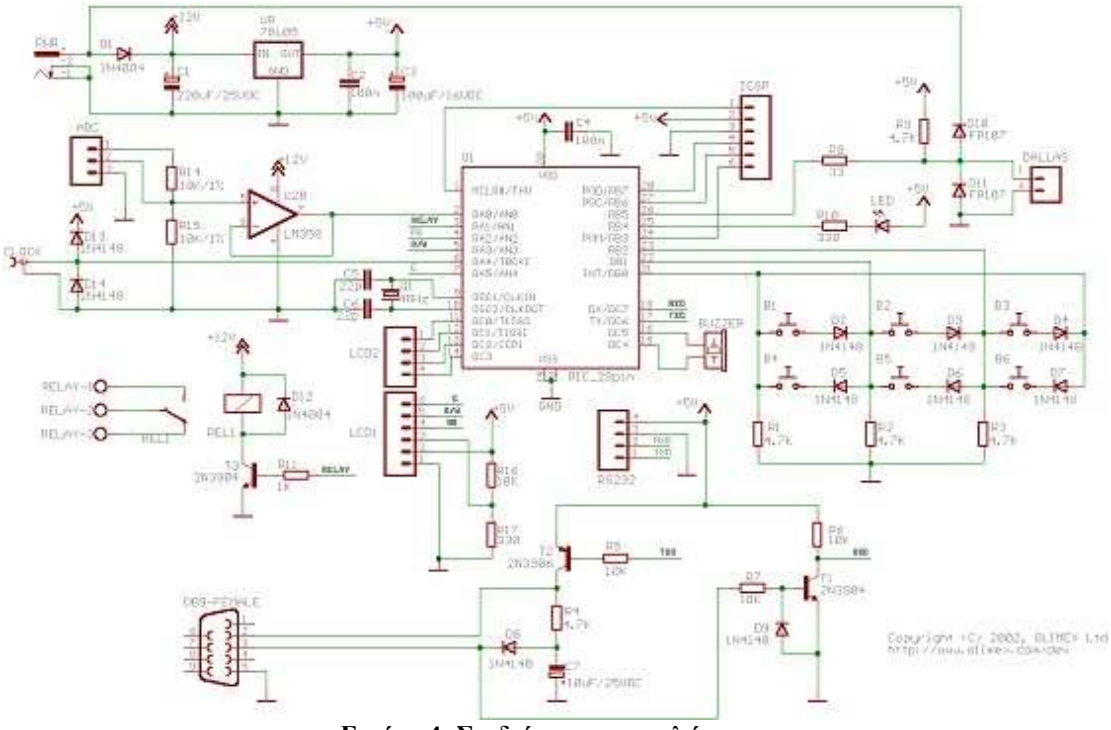
Εικόνα 2: Λογικό διάγραμμα των interrupts

Πλακέτα μικροελεγκτή

Ο μικοελεγκτής εγκαταστάθηκε σε μία πλακέτα συμβατή με τους PIC μικροελεγκτές από την εταιρία OLIMEX Ltd και συγκεκριμένα την PIC-MT [2].

Εικόνα 3: Η πλακέτα PIC-MT της Olimex Ltd

Η επιλογή της συγκεκριμένης πλακέτας ανάπτυξης έγινε, καθώς περιέχει όλα τα απαραίτητα περιφερειακά που θα χρειαστούμε για την περάτωση της πτυχιακής. Δηλαδή έχει «κολλημένη» πάνω της μία σειριακή θύρα, μία LCD οθόνη, κουμπιά χειρισμού, βομβητή, λυχνία LED καθώς και ένα relay με το οποίο θα ενεργοποιούνται οι συσκευές. Επίσης η πλακέτα υπήρχε ήδη διαθέσιμη καθώς είχε χρησιμοποιηθεί σε εργασία παλαιότερα οπότε δεν χρειάστηκε αγορά νέου υλικού. Ακολουθεί το σχεδιάγραμμα του κυκλώματος:



Εικόνα 4: Σχεδιάγραμμα κυκλώματος

Gsm Modem module

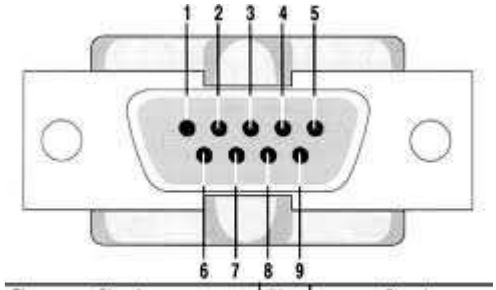
Για να είναι εφικτή η διαχείριση μέσω χρήσης της κινητής τηλεφωνίας χρειαζόμαστε και ένα Gsm Modem για τη λήψη και αποστολή των δεδομένων προς το δίκτυο. Χρησιμοποιούμε το Fastrack M1206B της εταιρίας Wavecom Instruments.

Εικόνα 5: Fastrack M1206B της εταιρίας Wavecom Instruments

Επίσης υπήρχε ήδη διαθέσιμο το συγκεκριμένο προϊόν οπότε δεν χρειάστηκε και πάλι προμήθεια νέου υλικού. Είναι ένα Dual Band GSM modem (λειτουργεί στις συχνότητες EGSM900/1800 MHz) και έχει σχεδιαστεί για εφαρμογές που υποστηρίζουν δεδομένα, fax, SMS και φωνή. Είναι πλήρως συμβατό με τις προδιαγραφές ETSI GSM Phase 2+ και GSM/GPRS class 10, ενώ μπορεί και υποστηρίζει χειρισμό μέσω AT εντολών (θα αναλυθούν παρακάτω). Ένα πολύ σημαντικό χαρακτηριστικό είναι ότι έχει ενσωματωμένα την κεραία, τη σειριακή θύρα καθώς και την υποδοχή της κάρτας sim (sim holder).

Διασύνδεση RS232 (σειριακή)

Για τη σύνδεση αυτών των δύο συσκευών μεταξύ τους αλλά και με τον υπολογιστή με τον οποίο προγραμματίστηκαν χρησιμοποιήθηκε ένα σειριακό καλώδιο RS232. Η πλακέτα και το modem διαθέτουν ένα βύσμα RS-232 (DB9 θηλυκής υποδοχής).



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

Εικόνα 6: Βύσμα RS-232

Στη σειριακή θύρα κάθε συσκευής υπάρχουν 3 ακροδέκτες για την αποστολή δεδομένων (TX), την λήψη (RX) δεδομένων καθώς και της γείωσης (GND). Έτσι συνδέοντας τους αντίστοιχους ακροδέκτες της κάθε συσκευής επιτυγχάνεται η σωστή σύνδεση.

Ο φορητός υπολογιστής που χρησιμοποιήθηκε δεν είχε ενσωματωμένη υποδοχή RS-232 οπότε χρησιμοποιήθηκε ένας μετατροπέας από USB σε RS-232.

Έτσι ο υπολογιστής είχε ένα βύσμα RS-232 (DB9 αρσενικής υποδοχής). Για τη σύνδεση και των τριών συσκευών κατασκευάστηκε ένα καλώδιο με ένα κεντρικό αρσενικό βύσμα που συνδέεται στο modem, ένα που συνδέεται στην πλακέτα και ένα θηλυκό βύσμα που συνδέεται στον φορητό υπολογιστή.

Περιγραφή εργαλείων λογισμικού

Αναπτυξιακό περιβάλον

Η εφαρμογή αναπτύχθηκε στη γλώσσα C καθώς είναι αρκετά σταθερή και ιδανική για εφαρμογές που υλοποιούνται από μικροελεγκτές.
Το αναπτυξιακό περιβάλλον που χρησιμοποιήθηκε ονομαζόμενο MPLAB [3] είναι της Microchip, σχεδιασμένο για PIC μικροελεγκτές και περιλαμβάνει έναν συμβολομεταφραστή (assembler), έναν διασυνδετή (linker), έναν εξομοιωτή λογισμικού (software simulator) και ένα εργαλείο αποσφαλμάτωσης (debugger). Πιο συγκεκριμένα το MPLAB Integrated Development Environment (IDE) είναι ένα σύνολο εργαλείων για την ανάπτυξη embedded εφαρμογών, την προσομοίωσή τους και τον εντοπισμό των σφαλμάτων τους. Προσφέρεται δωρεάν και για την πτυχιακή αυτή χρησιμοποιήθηκε η έκδοση MPLAB IDE v8.50.00. Παράλληλα χρέη μεταφραστή γλώσσας (compiler) εκτέλεσε ο Universal ToolSuite C Compiler της εταιρίας HI-TECH Software.
Τέλος μετά τη δοκιμή αρκετών προγραμμάτων προσομοίωσης σειριακής σύνδεσης, για τις δοκιμές στην επικοινωνία των συσκευών, γίνεται χρήση του δωρεάν εργαλείου Terminal v1.9b. Καθορίστηκαν οι ρυθμίσεις της σειριακής ως εξής:

- Bits per second: 9600
- Data bits: 8
- Parity: none
- Stop bits: 1
- Flow control: none

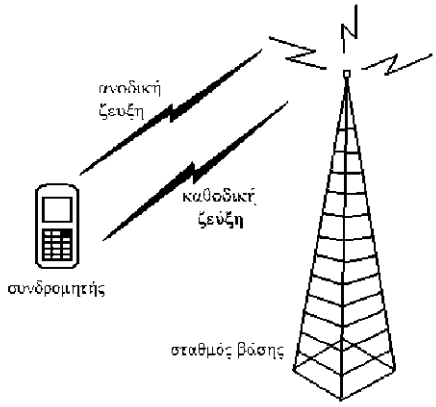
Προγραμματιστής

Για να μεταφέρουμε τον μεταγλωττισμένο κώδικα προγράμματος από το αναπτυξιακό περιβάλλον στον μικροελεγκτή απαιτείται μία διάταξη που ονομάζεται Προγραμματιστής. Οι μικροελεγκτές PIC έχουν τη δυνατότητα να προγραμματίζονται τοποθετημένοι πάνω στην πλακέτα της εφαρμογής υποστηρίζοντας την λειτουργία σειριακού προγραμματισμού ICSP (In Circuit Serial Programming). Ο σειριακός προγραμματισμός πραγματοποιείται χρησιμοποιώντας τρείς ακροδέκτες , δεδομένων PGD, χρονισμού PGC, τάσης προγραμματισμού VPP, και την τροφοδοσία της εφαρμογής. Στη συγκεκριμένη εφαρμογή χρησιμοποιήθηκε ο προγραμματιστής PIC-ICD2.5USB της εταιρίας Microchip Inc. Ένα προγραμματιστικό εργαλείο που δεν αναλαμβάνει απλά τη σειριακή μεταφορά του προγράμματος αλλά προσομοιώνει σε πραγματικό χρόνο τους PIC μικροελεγκτές και κάνει επαλήθευση του προγραμματισμού. Φυσικά συνεργάζεται με το πρόγραμμα MPLAB IDE.

ΚΕΦΑΛΑΙΟ 2 - GSM

Εισαγωγή στην τηλεφωνία κυψέλης

Τα συστήματα τηλεφωνίας κυψέλης είναι ασύρματα συστήματα τα οποία λειτουργούν με βάση την διανεμημένη μετάδοση πληροφοριών. Επομένως, αντί να υπάρχει μία μοναδική υπηρεσία μετάδοσης για πολλούς διαφορετικούς χρήστες γύρω από μια περιοχή κάλυψης, όπως στη ραδιοφωνική μετάδοση διαμόρφωσης συχνότητας (FM), η καλυπτόμενη περιοχή διαιρείται σε μικρότερες περιοχές γνωστές ως κελιά ή κυψέλες. Κάθε κυψέλη διαθέτει ένα σταθερό σύστημα πομποδέκτη γνωστό ως σταθμός βάσης.
Ο χρήστης ενός συστήματος κυψέλης επικοινωνεί με το σταθμό βάσης για να πραγματοποιήσει μία κλήση, φωνητική ή δεδομένων. Τότε ο σταθμός βάσης διανέμει την κλήση είτε προς το τερματικό σημείο ενός επίγειου δικτύου ή προς έναν άλλο χρήστη του δικτύου κυψέλης. Κανονικά, για τις φωνητικές κλήσεις, ο σταθμός βάσης διανέμει την κλήση σε ένα δίκτυο PSTN (Public Switched Telephony Network). Κάθε χρήστης του δικτύου ονομάζεται συνδρομητής. Η σχέση μεταξύ σταθμού βάσης και συνδρομητή φαίνεται στην διπλανή εικόνα. Η ζεύξη επικοινωνίας από τον σταθμό βάσης προς τον συνδρομητή ονομάζεται καθοδική, ενώ από τον συνδρομητή προς τον σταθμό βάσης ονομάζεται ανοδική ζεύξη.
Οι συνδρομητές έχουν τη δυνατότητα να κινούνται ή να μένουν σταθεροί. Αν ο συνδρομητής κινείται, το δίκτυο πρέπει να είναι ικανό να χειριστεί την περίπτωση κατά την οποία ο κινούμενος συνδρομητής, ή αλλιώς κινούμενος σταθμός, μετακινείται από μία κυψέλη σε άλλη. Για να διασφαλιστεί το γεγονός ότι η κλήση δεν θα τερματιστεί κατά τη μετακίνηση από μία κυψέλη σε άλλη, πληροφορίες σχετικές με τον κινούμενο σταθμό γίνονται γνωστές στους σταθμούς βάσης που εμπλέκονται στην περίπτωση. Γι' αυτόν και για άλλους λόγους, υπάρχει κάποια επικοινωνία στο δίκτυο η οποία συνδέει τους σταθμούς βάσης. Το δίκτυο αυτό ονομάζεται δίκτυο σπονδυλικής στήλης (backbone network).

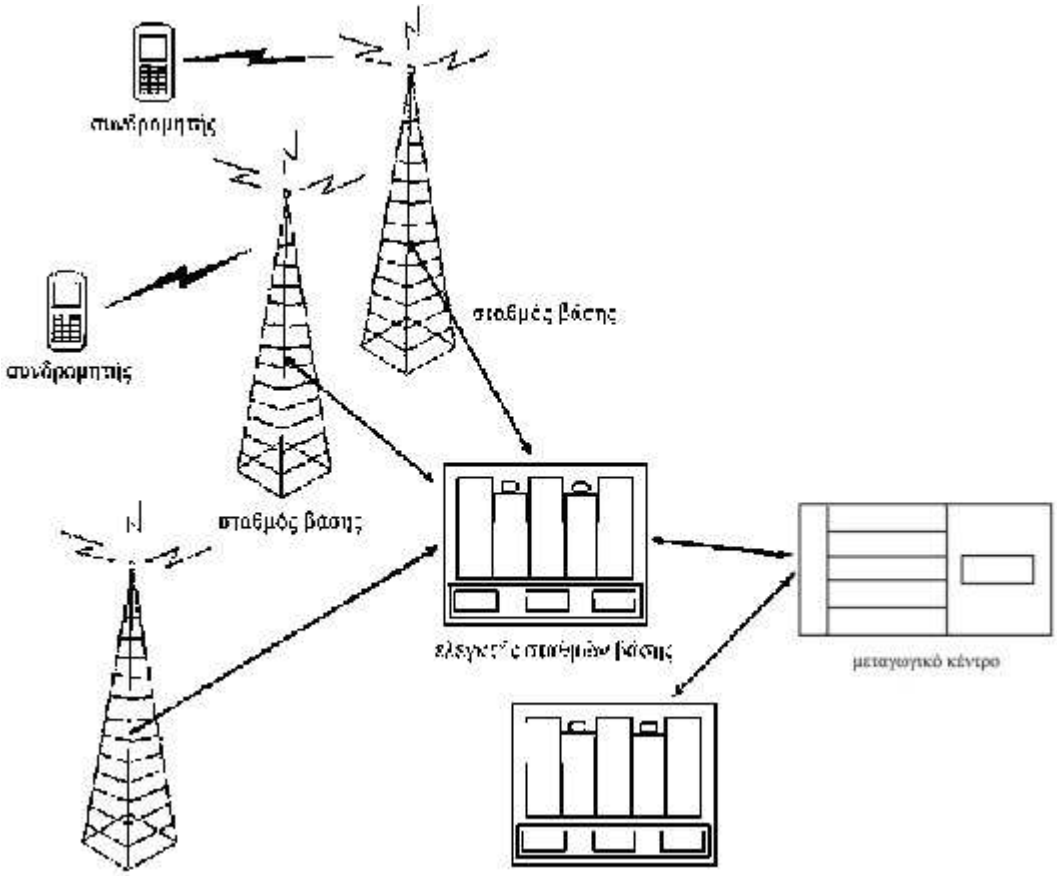


Ζεύξη συνδρομητή-σταθμού βάσης

Το δίκτυο backbone αποτελείται από ξεχωριστά συστήματα μεταξύ του δικτύου PSTN και του σταθμού βάσης. Ο σταθμός βάσης συνδέεται συνήθως με έναν ελεγκτή σταθμών βάσης (Base Station Controller, BSC), ο οποίος δικτυώνει ένα σύμπλεγμα (cluster) σταθμών βάσης για να διασφαλιστεί ότι η μεταφορά της κλήσης από τον ένα σταθμό στον άλλο πραγματοποιείται ισοδύναμα μέσα σε μία γεωγραφική περιοχή. Το σύμπλεγμα είναι μία ομάδα κυψελών η οποία χρησιμοποιεί το πλήρες σύνολο των διαθέσιμων τηλεφωνικών καναλιών σε ένα δίκτυο κυψέλης.

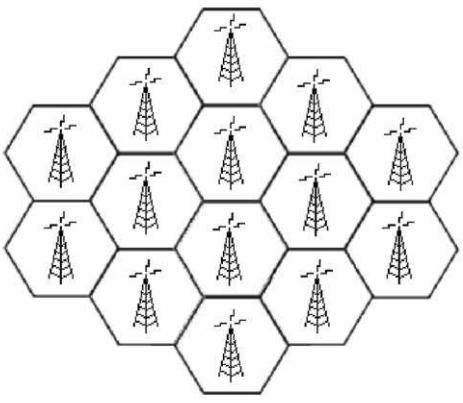
Ένας ή περισσότεροι ελεγκτές σταθμών βάσης συνδέονται συνήθως σε ένα μεταγωγικό κέντρο (Mobile Switching Center, MSC), το οποίο συνδέεται απ' ευθείας με το δίκτυο PSTN. Το μεταγωγικό κέντρο περιέχει πληροφορίες σχετικές με τον συνδρομητή που είναι δυνατό να χρησιμοποιηθούν για να δρομολογηθούν άλλες πληροφορίες προς αυτόν τον χρήστη κατά τη διάρκεια της κλήσης. Εκτός αυτού, ένας

καταχωρητής εντοπισμού έδρας (Home Location Register, HLR) μπορεί να είναι συνεγκατεστημένος με το μεταγωγικό κέντρο. Αυτή η μονάδα περιέχει συγκεκριμένες πληροφορίες οι οποίες χρησιμεύουν κυρίως στην πιστοποίηση του συνδρομητή κατά τη διάρκεια αρχικοποίησης της κλήσης.



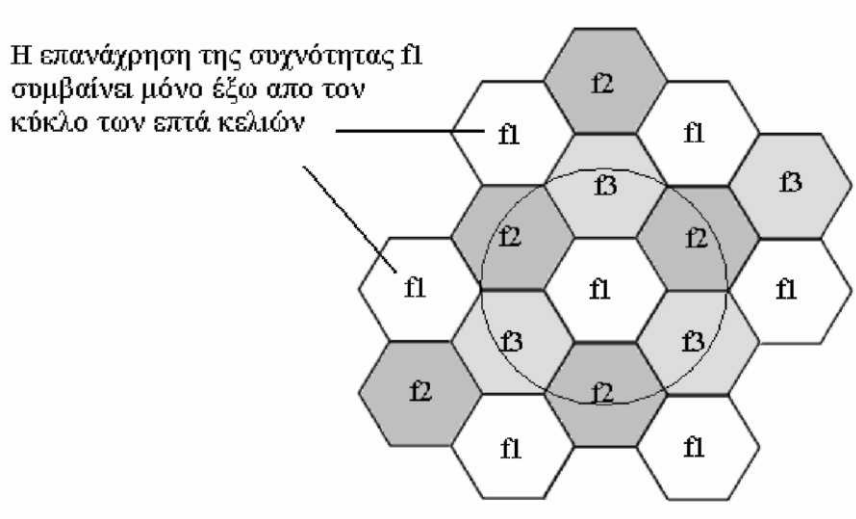
Εικόνα 7: Αρχιτεκτονική σπονδυλικής στήλης (backbone)

Ένα δίκτυο κυψέλης αποτελείται από πολλές κυψέλες σε μια συγκεκριμένη γεωγραφική διάταξη. Ένας σταθμός βάσης θα χρησιμοποιήσει διαφορετική συχνότητα (ή συχνότητες) επικοινωνίας από αυτή που χρησιμοποιούν οι σταθμοί βάσης των γειτονικών κυψελών. Αυτό αυξάνει τον παράγοντα επαναχρήσεως συχνότητας, ο οποίος αντιπροσωπεύει τον ελάχιστο αριθμό συχνοτήτων που απαιτούνται για ένα δεδομένο δίκτυο κυψέλης που εξασφαλίζει ότι η ομοσυχνοτική παρεμβολή βρίσκεται κάτω από ένα ανεκτό επίπεδο. Για λόγους ανάλυσης, τα κελιά αναπαρίστανται σε εξαγωνικό σχήμα (κυψέλης) με σκοπό να περιγραφεί ένα ιδανικό κυψελοειδές δίκτυο. Αυτός ο τύπος αναπαράστασης έχει ως αποτέλεσμα έναν παράγοντα επαναχρήσεως συχνότητας με τιμή 7, καθώς αυτός είναι ο ελάχιστος αριθμός των απαιτούμενων συχνοτήτων για να διασφαλιστεί ότι μεταξύ γειτονικών σταθμών βάσης δεν θα καταλαμβάνεται η ίδια συχνότητα.
Ο εξαγωνικός τύπος αναπαράστασης είναι επαρκής για την προκαταρκτική ανάλυση ενός δικτύου. Στην



Εξαγωνική αναπαράσταση κελιών

πραγματικότητα, διάφοροι παράγοντες όπως η μορφολογία του εδάφους, οι περιορισμοί στην παράταξη των σταθμών βάσης, κενά κάλυψης και υψηλής πυκνότητας (με την έννοια του αριθμού των συνδρομητών) περιοχές, αποτρέπουν την ομοιόμορφη σύνθεση του εξαγωνικού κελιού



Παράγοντας επανάχρησης συχνότητας με τιμή 7

Επικοινωνίες πολλαπλής πρόσβασης

Οι επικοινωνίες πολλαπλής πρόσβασης διαδραματίζουν καθοριστικό ρόλο ώστε να είναι εμπορικά βιώσιμο ένα σύστημα κυψέλης. Ο όρος πολλαπλή πρόσβαση αναφέρεται στο γεγονός κατά το οποίο οι χρήστες έχουν τη δυνατότητα να χρησιμοποιούν ταυτόχρονα το σύστημα κυψέλης. Τα ασύρματα συστήματα πολλαπλής πρόσβασης μπορούν να καταταχθούν σε τρείς κατηγορίες:

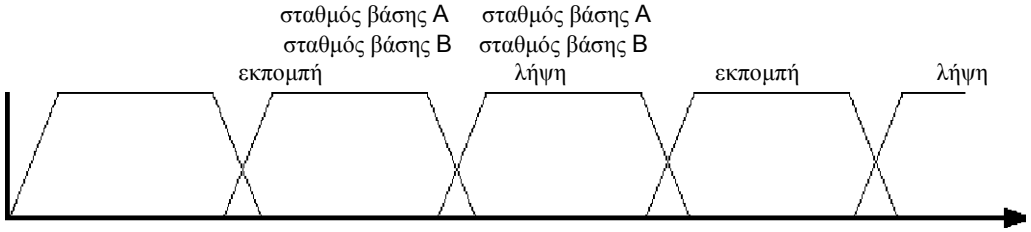
- Πολλαπλή πρόσβαση διαίρεσης συχνότητας Frequency Division Multiple Access (FDMA)
- Πολλαπλή πρόσβαση διαίρεσης χρόνου Time Division Multiple Access (TDMA)
- Πολλαπλή πρόσβαση διαίρεσης κώδικα Code Division Multiple Access (CDMA)

Δεν εννοείται μ' αυτό τον τρόπο ότι δεν υπάρχουν άλλα συστήματα πολλαπλής πρόσβασης στις επικοινωνίες κυψέλης. Παρόλα αυτά αυτές οι τρείς κατηγορίες περιγράφουν την εξέλιξη σχεδόν όλων των συστημάτων κυψέλης ανά τον κόσμο

Πολλαπλή πρόσβαση διαίρεσης συχνότητας

Τα συστήματα διαίρεσης συχνότητας αποτέλεσαν τα θεμέλια για τα πρώτα ευρέως αναπτυγμένα συστήματα κυψέλης στη Βόρεια Αμερική. Συγκεκριμένα, το Προοδευμένο Σύστημα Κινητής Τηλεφωνίας (Advanced Mobile Phone System, AMPS), αναπτύχθηκε αρχικά από τη εταιρεία AT & T και παρατάχθηκε στην Βόρειο Αμερική με μικρή παρουσία στην Πόλη του Μεξικό το 1981. Η πρώτη παράταξη του συστήματος στις Ηνωμένες Πολιτείες έγινε στην περιοχή του Σικάγο το 1983, σηματοδοτώντας το ξεκίνημα μίας εθνικού εύρους παρουσίασης των υπηρεσιών κυψέλης για το κοινό. Το σύστημα αυτό αναπτύχθηκε στη ζώνη των 800 MHz χρησιμοποιώντας απόσταση καναλιών στα 30 KHz η οποία ισχύει σ' αυτή τη ζώνη και σήμερα.

Το Ευρωπαϊκό Σύστημα Επικοινωνίας Ολικής Πρόσβασης (European Total Access Communication System, ETACS) αναπτύχθηκε στην Ευρώπη με τη μικρή διαφορά σε σχέση με το αμερικανικό AMPS στην απόσταση καναλιών που ήταν 25 KHz. Ομοίως το σύστημα N-AMPS (Narrowband AMPS, στενής ζώνης AMPS) αναπτύχθηκε από την Motorola ώστε να λειτουργεί με απόσταση καναλιών στα 10 KHz, αυξάνοντας την χωρητικότητα του συστήματος. Αυτά τα αρχικά συστήματα διαίρεσης συχνότητας αποκαλούνται ως συστήματα κυψέλης πρώτης γενιάς.
Τα συστήματα διαίρεσης συχνότητας γενικά λειτουργούν με κάθε σταθμό βάσης σε μια ομάδα κελιών να καταλαμβάνει ξεχωριστή συχνότητα και στις δύο ζώνες εκπομπής και λήψης (περιορισμένος παράγοντας επανάχρησης συχνότητας), με κάθε ζώνη να εξυπηρετεί έναν συνδρομητή. Ένα παράδειγμα φασματικής κατανομής για ένα σύστημα διαίρεσης συχνότητας δίνεται στην Εικόνα 8.



Εικόνα 8: Φασματική κατανομή συστήματος διαίρεσης συχνότητας

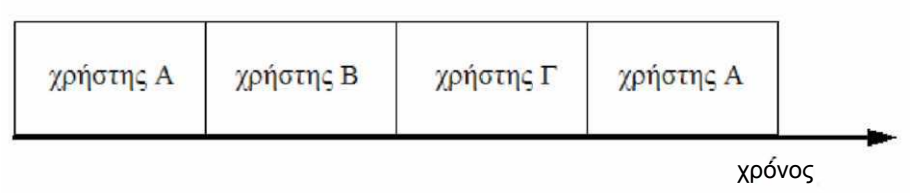
Αυτά τα πρώτης γενιάς συστήματα βασισμένα στη διαίρεση συχνότητας είναι πρωτίστως αναλογικά συστήματα, όπως το AMPS. Ωστόσο, αυτό δεν σημαίνει ότι δεν είναι δυνατό να μεταφερθεί ψηφιακή πληροφορία από τη φέρουσα ενός τέτοιου συστήματος. Είναι γεγονός ότι το AMPS μεταδίδει ομιλία σε αναλογική μορφή αλλά η πληροφορία ελέγχου μεταδίδεται σε ψηφιακή κατά τέτοιο τρόπο ώστε ο δέκτης να λαμβάνει είτε ομιλία ή πληροφορία ελέγχου μα ποτέ και τα δύο ταυτόχρονα.

Τα συστήματα διαίρεσης συχνότητας έμφυτους περιορισμούς εξ' αιτίας του γεγονότος ότι κάθε φασματικό κανάλι μπορεί να κατανεμηθεί σε έναν μόνο χρήστη. Συνεπώς, υποθέτοντας ότι ένας σταθμός βάσης έχει μία συχνότητα εκπομπής και μία συχνότητα λήψης, μπορεί να εξυπηρετήσει έναν χρήστη. Επιπρόσθετα, εξ' αιτίας του περιορισμένου παράγοντα επανάχρησης συχνότητας σε μια δεδομένη ανάπτυξη, η ενδοκαναλική παρεμβολή αποτελεί αντικείμενο για συζήτηση.

Πολλαπλή πρόσβαση διαίρεσης χρόνου

Στα πρώτα χρόνια της δεκαετίας του 1980, οι χειριστές κυψελωτής τηλεφωνίας και οι πωλητές ασύρματου εξοπλισμού ανά τον κόσμο αναγνώρισαν τους περιορισμούς χωρητικότητας των αναλογικών συστημάτων βασισμένων στη διαίρεση συχνότητας. Υπήρχε η ανησυχία ότι με την αύξηση της δημοτικότητας των υπηρεσιών κυψέλης, συστήματα όπως το AMPS δεν θα ήταν σε θέση να ανταποκριθούν αποτελεσματικά στις απαιτήσεις. Η ιδέα του AMPS συνελήφθη στα μέσα της δεκαετίας του 1960 και εκμεταλλεύτηκε την ύπαρξη της υψηλής τότε τεχνολογίας στη σχεδίαση κυκλωμάτων. Ωστόσο, δύο δεκαετίες αργότερα, ήταν δυνατή η ψηφιακή επεξεργασία σήματος πραγματικού χρόνου σε ολοκληρωμένα κυκλώματα.
Αυτό οδήγησε στην ανάπτυξη των πρώτων ψηφιακών συστημάτων βασισμένων στην πολλαπλή πρόσβαση διαίρεσης χρόνου. Στη Βόρεια Αμερική, υπήρχε η επιθυμία της διατήρησης της συμβατότητας μεταξύ του νέου συστήματος και της υπάρχουσας φασματικής κατανομής συχνοτήτων του AMPS. Ως αποτέλεσμα, στα τέλη της δεκαετίας του 1980, αναπτύχθηκε το D-AMPS (Digital AMPS) εφαρμόζοντας τη διαίρεση χρόνου με απόσταση καναλιών στα 30 KHz. Την ίδια περίοδο στην Ευρώπη, η Groupe Special Mobile (GSM) ανέπτυξε ένα ψηφιακό πρότυπο βασισμένο στη διαίρεση χρόνου με απόσταση καναλιών 200 KHz. Οι πρώτες υλοποιήσεις του GSM έγιναν το 1991 και του D-AMPS το 1992. Αυτά τα συστήματα ομαδοποιήθηκαν υπό τη γενική κατάταξη για τις πρώτες ψηφιακές τεχνολογίες κυψέλης γνωστή ως « δεύτερη γενιά ».

Τα συστήματα διαίρεσης χρόνου αξιοποιούν το φάσμα κατά παρόμοιο τρόπο με τα συστήματα διαίρεσης συχνότητας, με τον κάθε σταθμό βάσης σε μια ομάδα να καταλαμβάνει ξεχωριστή συχνότητα εκπομπής και λήψης. Ωστόσο, κάθε μια από τις δύο φασματικές ζώνες κατανέμεται επίσης στο χρόνο σε κάθε χρήστη κατά κυκλικό τρόπο. Ως παράδειγμα, η τριών θυρίδων διαίρεση χρόνου διαιρεί την εκπομπή σε τρείς αμετάβλητες χρονικές περιόδους (θυρίδες), κάθε μία με ισοδύναμη χρονική διάρκεια, με μια συγκεκριμένη θυρίδα να ορίζεται για εκπομπή σε (ή από, στην περίπτωση ανοδικής ζεύξης) έναν από τρείς πιθανούς χρήστες. Αυτός ο τύπος προσέγγισης απαιτεί εξαιρετικό συγχρονισμό μεταξύ του κινητού σταθμού και του σταθμού βάσης.



Εικ.2.6 πολλαπλή πρόσβαση διαίρεσης χρόνου τριών

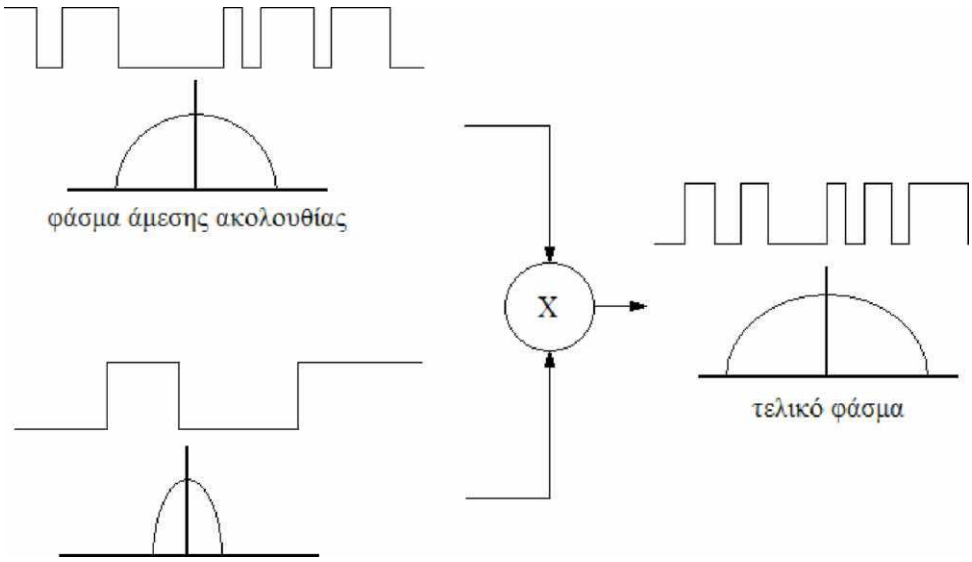
Ένας απλός κανόνας περιεκτικότητας είναι ότι ο αριθμός των θυρίδων σε ένα σύστημα διαίρεσης χρόνου είναι επίσης και ο αριθμός της αύξησης της χωρητικότητας συγκρινόμενης με αυτή του συστήματος διαίρεσης συχνότητας με πανομοιότυπο εύρος ζώνης. Ωστόσο, αυτό δεν επιβεβαιώνεται πάντα στα υλοποιημένα δίκτυα, εξ' αιτίας κυρίως των διαφορών της ψηφιακής επεξεργασίας στα συστήματα διαίρεσης χρόνου και της αναλογικής επεξεργασίας στα συστήματα διαίρεσης συχνότητας, λαμβάνοντας υπ' όψιν την ενδοκαναλική παρεμβολή και την μετρίαση των επιδράσεων του ασύρματου καναλιού.

Πολλαπλή πρόσβαση διαίρεσης κώδικα

Στα μέσα της δεκαετίας του 1980, αρκετοί ερευνητές θεώρησαν δυνατή την ύπαρξη μίας τεχνολογίας που θα χρησιμοποιείται για στρατιωτικές εφαρμογές καθώς επίσης και για τις επικοινωνίες κυψέλης. Αυτή η τεχνολογία, επικοινωνιών εκτεινόμενου φάσματος, η οποία ενέπλεκε τον μετασχηματισμό της πληροφορίας στενής ζώνης σε σήμα ευρείας ζώνης προς μετάδοση, θεωρήθηκε μέσο διευθυνσιοδότησης των περιορισμών της χωρητικότητας των συστημάτων διαίρεσης χρόνου (αποτέλεσμα του γεγονότος ότι ο αριθμός των χρηστών σε κάθε ξεχωριστή συχνότητα περιορίζεται από τον αριθμό των διαθέσιμων χρονικών θυρίδων).
Ένα σύστημα εκτεινόμενου φάσματος λειτουργεί μετασχηματίζοντας την στενής ζώνης πληροφορία ενός ξεχωριστού χρήστη σε πληροφορία ευρείας ζώνης με τη χρήση υψηλής συχνότητας κωδίκων, ο καθένας μοναδικός για αυτόν τον συγκεκριμένο χρήστη. Αναθέτοντας μοναδικούς κώδικες σε διαφορετικούς χρήστες, είναι δυνατή η πολλαπλή πρόσβαση, συγκεκριμένα η πολλαπλή πρόσβαση διαίρεσης κώδικα. Επιπλέον σε ένα σύστημα διαίρεσης κώδικα οι περιορισμοί στην επανάχρηση συχνότητας, που παρουσιάζουν τα συστήματα διαίρεσης συχνότητας και χρόνου, δεν είναι τόσο κρίσιμοι, καθώς πολλαπλοί κινητοί σταθμοί και σταθμοί βάσης μπορούν να κατέχουν τις ίδιες συχνότητες ταυτόχρονα.

Πρόσφατα, η πολλαπλή πρόσβαση διαίρεσης κώδικα σχημάτισε τη βάση για προηγμένα κυψελοειδή συστήματα ανά τον κόσμο. Αποτελώντας κομμάτι του ερευνητικού έργου της Διεθνούς Ένωσης Τηλεπικοινωνιών (International Telecommunications Union, ITU), τα συστήματα τρίτης γενιάς αναπτύχθηκαν για να βελτιώσουν τις ασύρματες υπηρεσίες πολυμέσων στους συνδρομητές.

Τα συστήματα πολλαπλής πρόσβασης διαίρεσης κώδικα αντιπροσωπεύονται από δύο τύπους: αναπήδησης συχνότητας και άμεσης ακολουθίας.
Η διαίρεση κώδικα που χρησιμοποιεί αναπήδηση συχνότητας εμπλέκει έναν χρήστη ο οποίος μεταδίδει μέσω πολλών συχνοτήτων συνεχώς στο χρόνο κατά ψευδοτυχαίο τρόπο. Η έννοια ψευδοτυχαίος σ' αυτή την περίπτωση αναφέρεται στο γεγονός ότι η ακολουθία των συχνοτήτων μετάδοσης είναι γνωστή στον πομπό και στον δέκτη, αλλά εμφανίζεται ως τυχαία σε κάθε άλλο δέκτη.
Στα αργής αναπήδησης συστήματα η αλλαγή συχνοτήτων γίνεται με μικρότερο ρυθμό από τον ρυθμό μετάδοσης της πληροφορίας ενώ στα γρήγορης αναπήδησης συμβαίνει το αντίθετο. Τα συστήματα αναπήδησης συχνότητας περιορίζονται από τον συνολικό αριθμό των διαθέσιμων συχνοτήτων προς μετάβαση. Αν δύο χρήστες μεταπηδήσουν στην ίδια συχνότητα ταυτόχρονα, θα παρεμβάλλουν ο ένας με τον άλλο.
Τα συστήματα άμεσης ακολουθίας λειτουργούν διαμορφώνοντας το σήμα πληροφορίας του χρήστη με μία ακολουθία γνωστή σε πομπό και δέκτη. Αυτή η ακολουθία παράγεται με ρυθμό κατά πολύ υψηλότερο από το σήμα του χρήστη, απλώνοντας κυριολεκτικά το εύρος ζώνης του σήματος του χρήστη. Η διαδικασία αυτή φαίνεται στην εικόνα 9. Όλα τα εμπορικά κυψελοειδή συστήματα διαίρεσης κώδικα χρησιμοποιούν την άμεση ακολουθία ανθιστάμενα στην τεχνολογία αναπήδησης συχνότητας.



Εικόνα 9: Έκταση της πληροφορίας κατά την άμεση ακολουθία

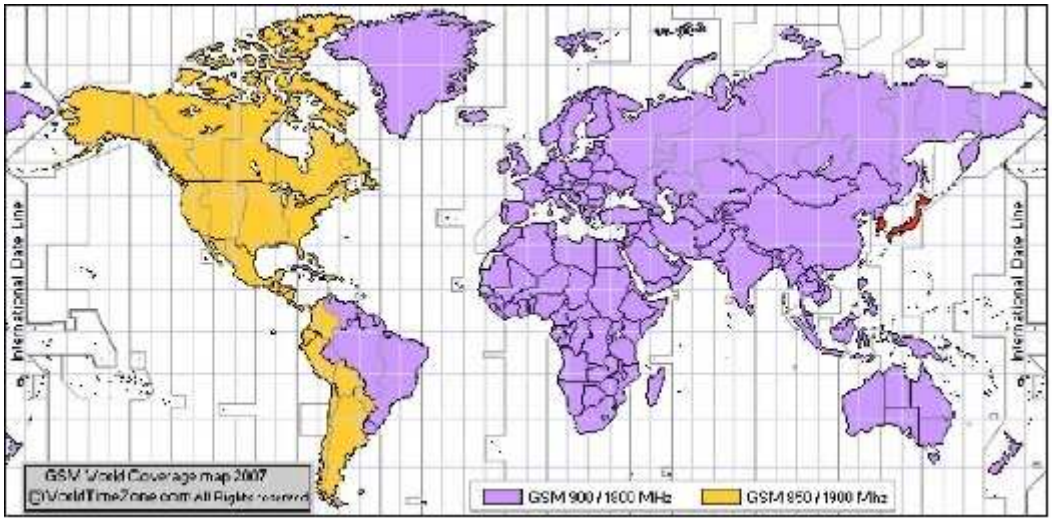
Σύντομη αναδρομή

Το 1982,το Ευρωπαϊκό Συμβούλιο Ταχυδρομικών και Τηλεπικοινωνιακών Διοικήσεων (European Conference of Postal and Telecommunications Administrations, CEPT) δημιούργησε την ομάδα Groupe Special Mobile (GSM) για να αναπτύξει ένα πρότυπο για ένα σύστημα κινητής τηλεφωνίας που θα μπορούσε να χρησιμοποιηθεί στην Ευρώπη. Το 1987 υπεγράφη το υπόμνημα κατανόησης από δεκατρείς χώρες για την ανάπτυξη ενός κοινού για την Ευρώπη κυψελοειδούς συστήματος τηλεφωνίας. Το 1989, την ευθύνη του GSM ανέλαβε το Ίδρυμα

Ευρωπαϊκών Τηλεπικοινωνιακών Προτύπων (European Telecommunications Standards Institute,ETSI) και το 1990 ανακοινώθηκε επίσημα για πρώτη φόρα το πρότυπο και τα χαρακτηριστικά του συστήματος. Τότε, η συντόμευση μετονομάστηκε από Group Special Mobile σε Global System for Mobile Communications GSM. Το πρώτο δίκτυο υλοποιήθηκε στη Φινλανδία από την Radiolinja με την συντήρηση της τεχνικής υποδομής σε συνεργασία με την Ericsson, ενώ στην Ελλάδα το 1993 από την WIND (τότε TELESTET).

Τεχνικές λεπτομέρειες του GSM

Το GSM είναι ένα κυψελοειδές δίκτυο, το οποίο σημαίνει ότι οι κινητοί σταθμοί (κινητά τηλέφωνα) συνδέονται σε αυτό αναζητώντας σταθμούς βάσης στην άμεση γειτνίαση. Το GSM λειτουργεί σε τέσσερις διαφορετικές ζώνες συχνοτήτων. Τα περισσότερα δίκτυα GSM λειτουργούν στις ζώνες των 900 MHz και 1800 MHz. Κάποιες χώρες στην Αμερική, συμπεριλαμβανόμενες και τις Ηνωμένες Πολιτείες και τον Καναδά, χρησιμοποιούν τις ζώνες των 850 MHz και 1900 MHz επειδή μέρος από τις ζώνες στα 900 και 1800 MHz έχουν ήδη κατανεμηθεί στα προηγούμενα αναλογικά συστήματα. Οι σπανιότερες ζώνες συχνοτήτων στα 400 και 450 MHz εκχωρήθηκαν σε κάποιες χώρες, κυρίως στη Σκανδιναβία, στις οποίες αυτές οι συχνότητες είχαν προηγουμένως χρησιμοποιηθεί για συστήματα πρώτης γενιάς.



Εικόνα 10: Παγκόσμιος χάρτης κάλυψης

Στη ζώνη των 900 MHz, η ζώνη συχνοτήτων ανοδικής ζεύξης είναι μεταξύ των 890 και 915 MHz και η ζώνη καθοδικής ζεύξης μεταξύ των 935 και 960 MHz. Το εύρος των 25 MHz υποδιαιρείται σε 124 φέρουσες συχνότητες καναλιών, το καθένα από

αυτά εκτεινόμενο στα 200 KHz. Χρησιμοποιείται η διαίρεση χρόνου για την παραγωγή οκτώ πλήρους ρυθμού η δεκάξι ημίσιου ρυθμού καναλιών ομιλίας για κάθε συχνοτικό κανάλι. Υπάρχουν οκτώ χρονοθυρίδες ομαδοποιημένες στο λεγόμενο πλαίσιο πολλαπλής πρόσβασης διαίρεσης χρόνου, TDMA frame. Ένα κανάλι ανταποκρίνεται στην επανεμφάνιση μιας θυρίδας ανά πλαίσιο και καθορίζεται από τη συχνότητα και τη θέση της θυρίδας μέσα στο πλαίσιο. Στο GSM υπάρχουν δύο τύποι καναλιών, τα κανάλια κίνησης και τα κανάλια ελέγχου.

- Τα κανάλια κίνησης μεταφέρουν πληροφορίες ομιλίας και δεδομένων. Τα κανάλια πλήρους ρυθμού (TCH/F) καθορίζονται χρησιμοποιώντας ένα σύνολο από 26 πλαίσια TDMA το οποίο ονομάζεται 26-multiframe. Το πολυπλαίσιο αυτό διαρκεί 120 ms. Μέσα σ' αυτή τη δομή, τα κανάλια της ανοδικής και της καθοδικής ζεύξης διαχωρίζονται από τρείς θυρίδες. Ως συνέπεια, οι κινητοί σταθμοί δεν χρειάζεται να εκπέμπουν και να λαμβάνουν ταυτόχρονα γεγονός που απλοποιεί το ηλεκτρονικό υλικό του συστήματος. Τα πλαίσια που αποτελούν το 26-πολυπλαίσιο έχουν διαφορετικές λειτουργίες και είναι: 24 πλαίσια για την κίνηση .1 πλαίσιο χρησιμοποιείται για το κανάλι συνεταιρικού ελέγχου και το τελευταίο πλαίσιο δεν χρησιμοποιείται. Αυτό το πλαίσιο επιτρέπει στον κινητό σταθμό άλλες λειτουργίες όπως η μέτρηση της ισχύος του σήματος γειτονικών κυψελών.

- Τα κανάλια ελέγχου μεταφέρουν πληροφορίες αναγνώρισης, πρόσβασης και συγχρονισμού του κινητού σταθμού με το δίκτυο, καθώς και σήματα που σχετίζονται με την επιλογή και διατήρηση του καναλιού κίνησης

Ο ρυθμός δεδομένων του καναλιού είναι 270,833 kbit/s και η διάρκεια του πλαισίου είναι 4,615 ms ενώ η διάρκεια της χρονοθυρίδας 0,577 ms.
Το E-GSM καθορίστηκε από την Ευρωπαϊκή Επιτροπή ΡάδιοΕπικοινωνιών στα τέλη της δεκαετίας του 1990 για να «αντικαταστήσει» το κλασικό GSM900 διατηρώντας βέβαια την δομή του αυξάνοντας όμως τις περιοχές συχνοτήτων από 880 έως 915 MHz ανοδικής ζεύξης και 925 έως 960 MHz καθοδικής ζεύξης. Έτσι επέτρεψε στα δίκτυα κινητής τηλεφωνίας να αυξήσουν τη χωρητικότητά τους και να καλύψουν την ανάγκη ζήτησης λόγω της αύξησης των χρηστών.
Στη ζώνη των 1800 MHz διατηρείται η δομή ενός GSM900 δικτύου αλλά χρησιμοποιούνται διαφορετικά ζεύγη συχνοτήτων, από τα 1710 έως τα 1785 MHz ανοδικής ζεύξης και από τα 1805 έως τα 1880 MHz καθοδικής ζεύξης. Οι περιοχές των 75 MHz υποδιαιρούνται η καθεμία σε 374+ (1 ελεύθερο) κανάλια και κάθε κανάλι έχει εύρος ζώνης 200KHz. Αυτή η αλλαγή στην ζώνη συχνοτήτων έγινε διότι οι ζώνες του GSM 900 στην Ευρώπη ήταν δεσμευμένες από άλλους παροχείς κινητής τηλεφωνίας. Όπως και στην χώρα μας σήμερα όλες οι εταιρείες κινητής τηλεφωνίας χρησιμοποιούν και τα δύο συστήματα (GSM900/1800) αυξάνοντας αισθητά τη χωρητικότητά στα δίκτυα τους.
Στο δίκτυο GSM υπάρχουν πέντε διαφορετικά μεγέθη κυψελών

- κυψέλη macro θεωρείται η κυψέλη στην οποία η κεραία του σταθμού βάσης είναι τοποθετημένη σε ιστό σε υψόμετρο μεγαλύτερο από το μέσο ύψος των κτιρίων.

- κυψέλη micro θεωρείται η κυψέλη στην οποία η κεραία του σταθμού βάσης βρίσκεται κάτω από το μέσο ύψος των κτιρίων και τυπικά χρησιμοποιείται

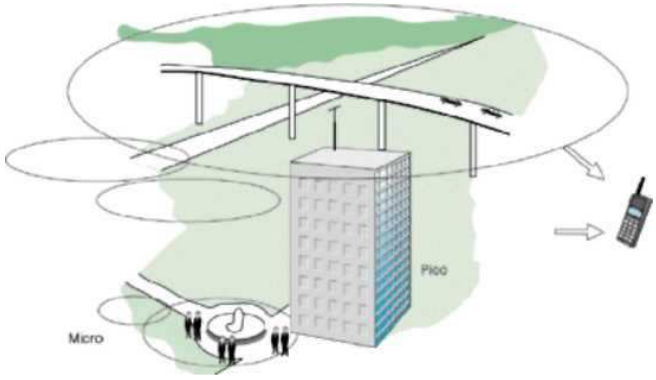
σε αστικές περιοχές.

- κυψέλη pico θεωρείται η κυψέλη της οποίας η διάμετρος κάλυψης είναι μερικές δεκάδες μέτρα και χρησιμοποιείται κατα κύριο λόγο σε εσωτερικούς χώρους

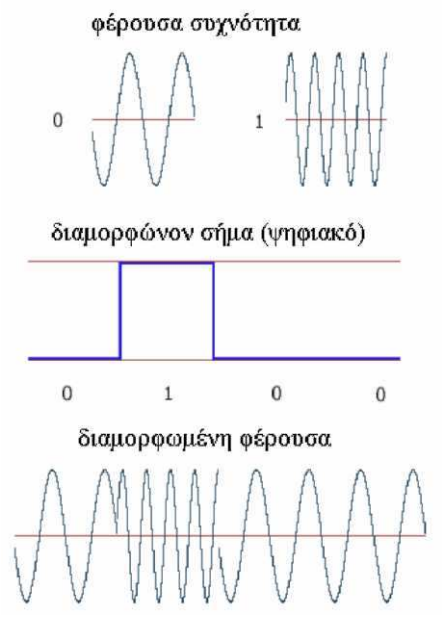
- κυψέλη femto σχεδιάστηκε για χρήση σε κατοικίες ή μικρούς επαγγελματικούς χώρους και συνδέονται με τον δικτυακό πάροχο μέσω ευρυζωνικής διαδικτυακής σύνδεσης.

- κυψέλη umbrella (ομπρέλα) χρησιμοποιείται για να καλύψει σκιασμένες περιοχές μικρότερων κυψελών και να γεμίσει κενά κάλυψης μεταξύ αυτών.

Η οριζόντια ακτίνα μιας κυψέλης ποικίλει εξαρτώμενη από το ύψος και το κέρδος της κεραίας καθώς και από τις συνθήκες διάδοσης, από λίγες εκατοντάδες μέτρα έως αρκετές δεκάδες χιλιόμετρα. Η μεγαλύτερη πρακτική απόσταση που υποστηρίζεται από τις προδιαγραφές του GSM είναι 35 χιλιόμετρα. Υπάρχουν επίσης υλοποιήσεις εκτεταμένης κυψέλης όπου η ακτίνα διπλασιάζεται ή και περισσότερο, εξαρτώμενη από το σύστημα της κεραίας και τη μορφολογία του εδάφους.



Η διαμόρφωση που χρησιμοποιείται για τη μετάδοση των πληροφοριών στο δίκτυο GSM είναι η GMSK. Στη διαμόρφωση GMSK, το σήμα που πρόκειται να διαμορφώσει τη φέρουσα κατα συχνότητα περνά από ένα Gaussian χαμηλοπερατό φίλτρο πριν τροφοδοτηθεί στον διαμορφωτή συχνότητας, γεγονός το οποίο μειώνει την παρεμβολή μεταξύ γειτονικών καναλιών



Εικ.2.13 διαμόρφωση

Η μονάδα ταυτότητας συνδρομητή (SIM)

Ένα από τα χαρακτηριστικά του συστήματος GSM είναι η Μονάδα Ταυτότητας Συνδρομητή γνωστή ως κάρτα SIM (SUBSCRIBER IDENTITY MODULE). Η SIM είναι μια «έξυπνη» κάρτα η οποία περιέχει τις πληροφορίες εγγραφής στο δίκτυο και τον τηλεφωνικό κατάλογο του συνδρομητή. Μια κάρτα SIM διαθέτει έναν μικροεπεξεργαστή, μια μνήμη ROM που χρησιμοποιείται για τις λειτουργίες του δικτύου (αναγνωριστικά, πιστοποίηση, κέντρο μηνυμάτων κτλ.) και μια μνήμη EPROM την οποία
χρησιμοποιεί ο χρήστης για τα προσωπικά του δεδομένα. Επιτρέπει έτσι στον συνδρομητή να διατηρεί τις προσωπικές του πληροφορίες κατά την αλλαγή τηλεφωνικής συσκευής. Εναλλακτικά, ο συνδρομητής έχει τη δυνατότητα να αλλάξει τον πάροχο των υπηρεσιών διατηρώντας την συσκευή απλά αλλάζοντας την SIM. Επίσης ο συνδρομητής έχει τη δυνατότητα να ασφαλίζει τα δεδομένα
που υπάρχουν στην κάρτα χρησιμοποιώντας παρεχόμενους με την κάρτα κωδικούς ασφαλείας.

Η ασφάλεια στο GSM

Το GSM σχεδιάστηκε με ένα μέτριο επίπεδο ασφαλείας. Το σύστημα πιστοποιεί την αυθεντικότητα του συνδρομητή χρησιμοποιώντας ένα προ-διαμοιρασμένο κλειδί (pre-shared key στην κρυπτογραφία) και την μέθοδο της έγκυρης απάντησης σε συγκεκριμένη ερώτηση (challenge-response). Οι επικοινωνίες μεταξύ του συνδρομητή και του σταθμού βάσης μπορούν να κρυπτογραφηθούν. Η ανάπτυξη του Συστήματος Παγκόσμιων Κινητών Τηλεπικοινωνιών (Universal Mobile Telecommunications System, UMTS) παρουσιάζει μια προαιρετική κάρτα, την USIM, η οποία χρησιμοποιεί ένα μακρύτερο κλειδί αυθεντικότητας αποδίδοντας μεγαλύτερη ασφάλεια. Το δίκτυο GSM χρησιμοποιεί αρκετούς αλγορίθμους ασφαλείας. Για να διασφαλιστεί η μυστικότητα των συνδιαλέξεων χρησιμοποιούνται τα κρυπτογραφήματα ροής A5/1 και A5/2. Το κρυπτογράφημα A5/1 αναπτύχθηκε πρώτο και είναι ένας δυνατότερος αλγόριθμος που χρησιμοποιείται στην Ευρώπη και στις Ηνωμένες Πολιτείες ενώ το κρυπτογράφημα A5/2 είναι ασθενέστερο και χρησιμοποιείται σε άλλες χώρες.

Υπηρεσίες συνδρομητή του GSM

Υπάρχουν δύο ειδών υπηρεσίες που προσφέρει στους συνδρομητές του το GSM

- οι υπηρεσίες τηλεφωνίας (αναφερόμενες και ως τηλεϋπηρεσίες) και

- οι υπηρεσίες δεδομένων (αναφερόμενες και ως υπηρεσίες κομιστή)

Οι υπηρεσίες τηλεφωνίας είναι κυρίως υπηρεσίες ομιλίας οι οποίες παρέχουν στον συνδρομητή τη δυνατότητα να επικοινωνεί με άλλους συνδρομητές. Οι υπηρεσίες δεδομένων παρέχουν την απαραίτητη χωρητικότητα ώστε να μεταδοθούν κατάλληλα σήματα δεδομένων μεταξύ δύο σημείων πρόσβασης δημιουργώντας διασύνδεση με το δίκτυο. Επιπρόσθετα της κανονικής τηλεφωνίας και των κλήσεων εκτάκτου ανάγκης το δίκτυο GSM παρέχει τις ακόλουθες υπηρεσίες.

- Διτονική πολλαπλή συχνότητα (dual-tone multifrequency, DTMF). Το DTMF είναι ένα σχέδιο τονικής σηματοδότησης που χρησιμοποείται συχνά για διάφορους σκοπούς ελέγχου διαμέσου του τηλεφωνικού δικτύου, όπως ο ασύρματος έλεγχος μιας μηχανής απάντησης.

- Υπηρεσία γραπτών μηνυμάτων (short message service, SMS). Στην υπηρεσία SMS, ένα μήνυμα αποτελούμενο το μέγιστο από 160 χαρακτήρες μπορεί να σταλεί από ή προς ένα κινητό σταθμό. Αν η συσκευή του συνδρομητή είναι απενεργοποιημένη ή βρίσκεται σε περιοχή εκτός κάλυψης, το μήνυμα αποθηκεύεται ώστε να σταλεί στον συνδρομητή αμέσως μόλις η συσκευή ενεργοποιηθεί ή επανέλθει σε καλυπτόμενη από το δίκτυο περιοχή. Αυτή η λειτουργία διασφαλίζει την παράδοση του μηνύματος.

- Ευρεία γνωστοποίηση από την κυψέλη (cell broadcast), που είναι μία διαφοροποίηση της υπηρεσίας SMS. Ένα μήνυμα με μέγιστο αριθμό 93 χαρακτήρων μπορεί να μεταδοθεί προς όλους τους συνδρομητές σε μια συγκεκριμένη γεωγραφική περιοχή. Τυπικές εφαρμογές αποτελούν οι προειδοποιήσεις κυκλοφοριακής συμφόρησης και οι αναφορές ατυχημάτων.

- Υπηρεσία φωνητικών μηνυμάτων (voice mail). Αυτή η υπηρεσία είναι ουσιαστικά μία μηχανή απάντησης εντός του δικτύου, η οποία ελέγχεται από τον συνδρομητή. Οι κλήσεις μπορούν να προωθηθούν στο φάκελο φωνητικών μηνυμάτων και ο συνδρομητής να τα ελέγξει χρησιμοποιώντας ως κλειδί πρόσβασης έναν κωδικό ασφαλείας.

Επίσης, το δίκτυο GSM υποστηρίζει ένα σύνολο από συμπληρωματικές υπηρεσίες που μπορούν να υποστηρίξουν και τις δύο βασικές υπηρεσίες, τηλεφωνίας και δεδομένων. Ένα μέρος αυτών ακολουθεί παρακάτω.

- Προώθηση κλήσεων (call forwarding). Αυτή η υπηρεσία παρέχει στον συνδρομητή τη δυνατότητα να προωθήσει τις εισερχόμενες κλήσεις σε έναν άλλο αριθμό εάν η καλούμενη κινητή μονάδα δεν είναι προσβάσιμη ή είναι απασχολημένη ή δεν απαντά.

- Φραγή κλήσεων (call barring). Αυτή η υπηρεσία επιτρέπει στον συνδρομητή να αποτρέψει εισερχόμενες ή εξερχόμενες κλήσεις.

- Κράτηση κλήσης (call hold). Αυτή η υπηρεσία επιτρέπει στον συνδρομητή να διακόψει μία τρέχουσα κλήση και στη συνέχεια να την επαναφέρει. Η υπηρεσία κράτησης κλήσης είναι διαθέσιμη μόνο στην κανονική τηλεφωνία.

- Αναμονή κλήσης (call waiting). Η υπηρεσία αναμονής κλήσης ειδοποιεί τον συνδρομητή για την ύπαρξη εισερχόμενης κλήσης κατα τη διάρκεια μιας συνδιάλεξης. Ο συνδρομητής μπορεί να απαντήσει, να απορρίψει ή να αγνοήσει την εισερχόμενη κλήση.

Υπηρεσία Σύντομων Γραπτών Μηνυμάτων SMS

Μία από τις πιο επιτυχημένες υπηρεσίες που αναφέρθηκαν πιο πάνω και προσέφεραν τα δίκτυα κινητής τηλεφωνίας είναι αυτή των σύντομων γραπτών μηνυμάτων (SMS, Short Message Service). Όπως και οι AT εντολές έτσι και τα sms καθορίζονται με βάση τα πρότυπα του Ευρωπαϊκού Οργανισμού ETSI (πρότυπα GSM). Είναι μία ασύμμετρη υπηρεσία και δεν χρειάζεται να υπάρχει δέσμευση φάσματος για τους κινητούς σταθμούς. Η αποστολή του μηνύματος από τον αποστολέα μπορεί να γίνει είτε είναι ο παραλήπτης ενεργός στο δίκτυο είτε όχι. Για αυτό και οι υπηρεσίες αποστολής και λήψης θεωρούνται ως ξεχωριστές υπηρεσίες. Βέβαια υπάρχει και η ανάγκη ενός κόμβου μεταξύ των δύο σταθμών. Έτσι ενώ ο αποστολέας στέλνει το μήνυμα που συντάσσει στον παραλήπτη αυτό φτάνει πρώτα στο κέντρο μεταγωγής μηνυμάτων της εταιρίας (SMS Center, SMSC) και από κει προωθείται στον πραγματικό παραλήπτη μαζί με κάποιες πρόσθετες πληροφορίες για την ορθή παράδοση του. Σύμφωνα με το πρότυπο του SMS και όπως προσδιορίστηκε από τον οργανισμό ETSI για το κάθε μήνυμα χρησιμοποιούνται 160 χαρακτήρες. Υπάρχουν δύο τρόποι αποστολής και λήψης των sms μηνυμάτων. Διαφέρει η δομή τους, η σύνταξη κάποιων εντολών, η υποστήριξη κάποιων χαρακτηριστικών, ακόμη και ευκολία χρήσης της κάθε μεθόδου.
Ο πρώτος είναι η αποστολή του μηνύματος σε μορφή κειμένου (text mode). Έχει πολύ απλή δομή. Τοποθετείται στην αρχή του μηνύματος ο αριθμός του παραλήπτη και ακολουθεί το μήνυμα. Για παράδειγμα για να στείλουμε στον αριθμό +1234567890 το μήνυμα «hellohello» θα έπρεπε να στείλουμε την εξής εντολή στο modem:

AT+CMGS="+1234567890"<CR>hellohello<Ctrl+z>

Ο δεύτερος τρόπος του pdu mode(Protocol Data Unit) στέλνει μια ακολουθία δεκαεξαδικών χαρακτήρων ακλουθώντας συγκεκριμένη δομή. Εσωκλείει διάφορες πληροφορίες όπως το νούμερο του αποστολέα, το κέντρο μεταγωγής σύντομων μηνυμάτων (SMS Center), την ώρα αποστολής, την ημερομηνία καθώς και άλλες πληροφορίες. Τα δεδομένα κάθε τμήματος είναι γραμμένα σε αλφαριθμητική μορφή που ονομάζεται hexadecimal-octets και semi decimal-octets (οι ορολογίες αυτές εισάγονται από τον οργανισμό ETSI) Για παράδειγμα για το παραπάνω μήνυμα η εντολή που πρέπει να σταλεί είναι:

AT+CMGS=32
<CR>0011000B916407281553F80000AA0AE8329BFD4697D9EC37
<Ctrl+z>

Η AT+CMGS εντολή στην αρχή είναι αυτή που πρέπει να σταλεί στο modem για την αποστολή ενός μηνύματος. Ακολουθεί το πλήθος των octets του μηνύματος και μετά

ένας ειδικός χαρακτήρας, ο χαρακτήρας επαναφοράς (carriage return η δεκαεξαδική του μορφή είναι 0x00D και είναι ο χαρακτήρας του enter).
Το σώμα αποστολής ενός pdu [4] περιλαμβάνει αναλυτικά:

Octet(s)	ΠΕΡΙΓΡΑΦΗ
00	Length of SMSC information. Here the length is 0, which means that the SMSC stored in the phone should be used. *Note: This octet is optional. On some phones this octet should be omitted! (Using the SMSC stored in phone is thus implicit)*
11	First octet of the SMS-SUBMIT message.
00	TP-Message-Reference. The "00" value here lets the phone set the message reference number itself.
0B	Address-Length. Length of phone number (11)
91	Type-of-Address. (91 indicates international format of the phone number).
6407281553F8	The phone number in semi octets (46708251358). The length of the phone number is odd (11), therefore a trailing F has been added, as if the phone number were "46708251358F". Using the unknown format (i.e. the Type-of-Address 81 instead of 91) would yield the phone number octet sequence 7080523185 (0708251358). Note that this has the length 10 (A), which is even.
00	TP-PID. Protocol identifier
00	TP-DCS. Data coding scheme.This message is coded according to the 7bit default alphabet. Having "04" instead of "00" here, would indicate that the TP-User-Data field of this message should be interpreted as 8bit rather than 7bit (used in e.g. smart messaging, OTA provisioning etc).
AA	TP-Validity-Period. "AA" means 4 days. *Note: This octet is optional, see bits 4 and 3 of the first octet*
0A	TP-User-Data-Length. Length of message. The TP-DCS field indicated 7-bit data, so the length here is the number of septets (10). If the TP-DCS field were set to 8-bit data or Unicode, the length would be the number of octets.
E8329BFD4697D9EC37	TP-User-Data. These octets represent the message "hellohello". How to do the transformation from 7bit septets into octets is shown here

Πίνακας 1: ΔΟΜΗ PDU ΑΠΟΣΤΟΛΗΣ

Παρόμοια είναι και η δομή κατά τη λήψη ενός pdu:

Octet(s)	ΠΕΡΙΓΡΑΦΗ
07	Length of the SMSC information (in this case 7 octets)
91	Type-of-address of the SMSC. (91 means international format of the phone number)
72 83 01 00 10 F5	Service center number(in decimal semi-octets). The length of the phone number is odd (11), so a trailing F has been added to form proper octets. The phone number of this service center is "+27381000015". See below.
04	First octet of this SMS-DELIVER message.
0B	Address-Length. Length of the sender number (0B hex = 11 dec)
C8	Type-of-address of the sender number
72 38 88 09 00 F1	Sender number (decimal semi-octets), with a trailing F
00	TP-PID. Protocol identifier.
00	TP-DCS Data coding scheme
99 30 92 51 61 95 80	TP-SCTS. Time stamp (semi-octets)
0A	TP-UDL. User data length, length of message. The TP-DCS field indicated 7-bit data, so the length here is the number of septets (10). If the TP-DCS field were set to indicate 8-bit data or Unicode, the length would be the number of octets (9).
E8329BFD4697D9EC37	TP-UD. Message "hellohello", 8-bit octets representing 7-bit data.

Πίνακας 2: ΔΟΜΗ PDU ΛΗΨΗΣ

Κωδικοποίηση PDU SMS

Όπως αναφέρθηκε στην αποστολή ενός sms κάθε χαρακτήρας απεικονίζεται από συνδυασμούς των 7 bit (GSM septets). Όταν γίνει η κωδικοποίηση [6][5] όμως θα προκύψουν χαρακτήρες των 8 bit (octets). Για χαρακτήρες γραμμάτων ή αριθμών η δεκαεξαδική τους τιμή συμπίπτει με την ASCII κωδικοποίηση τους όπως θα φανεί και στο παράδειγμα που θα ακολουθήσει. Η κωδικοποίηση κάθε χαρακτήρα στην ακολουθία δεν εξαρτάται μόνο από τον ίδιο αλλά είναι συνάρτηση και του επόμενου χαρακτήρα. Έτσι εξηγείται γιατί ίδιοι χαρακτήρες μέσα στο μήνυμα δεν παρουσιάζουν την ίδια κωδικοποίηση.
Για να δημιουργήσουμε την 8 bit (octet) μορφή ενός χαρακτήρα χρησιμοποιούμε το τελευταίο bit από τον επόμενο χαρακτήρα (λιγότερο σημαντικό bit) και το τοποθετούμε στην αρχή του (ως πιο σημαντικό bit).

Ο επόμενος χαρακτήρας βέβαια αφού του έχουμε ήδη αφαιρέσει το τελευταίο bit θα χρειαστεί από τον επόμενο 2 bit και ούτω καθεξής.
Έτσι λοιπόν στο παράδειγμα που ακολουθεί για την κωδικοποίηση της λέξης «hellohello» βλέπουμε με κόκκινο τα bit που μετακινούνται. Να σημειωθεί ότι για να δημιουργηθεί το όγδοο octet χρησιμοποιείται ολόκληρο το όγδοο septet μαζί με το bit του επόμενου septet. Αυτός είναι ο λόγος όπου από 10 septet δημιουργούνται 9 octet!

```
Alphabet ----Decimal------Septet -------Octet----------Hex
==========================================
h ----------------->104 --------------- >1101000 -------->11101000 ----------->E8
e ----------------->101 --------------- >1100101 -------->00110010------------>32
l ----------------->108 --------------- >1101100 -------->10011011 ----------->9B
l ----------------->108 --------------- >1101100 -------->11111101 ----------->FD
o------------------>111 --------------- >1101111 -------->01000110 ----------->46
h ----------------->104 --------------- >1101000 -------->10010111 ----------->97
e ----------------->101 --------------- >1100101--------->11011001 ----------->D9
l ----------------->108 --------------- >1101100--------- ---------- ----------
l ----------------->108 --------------- >1101100 --------->11101100 ----------->EC
o ----------------->111 --------------- >1101111---------->110111 ------------->37
```

Αυτή είναι η θεωρητική προσέγγιση του αλγόριθμου κωδικοποίησης. Σε επόμενο κεφάλαιο θα παρουσιαστεί αναλυτικά και ο κώδικας για την κωδικοποίηση στη γλώσσα προγραμματισμού C.

Αποκωδικοποίηση PDU SMS

Η διαδικασία της αποκωδικοποίησης [5] είναι ακριβώς η αντίστροφη.
Δημιουργούνται λοιπόν από δεκαεξαδικούς αριθμούς (8 bit) χαρακτήρες των 7 bit (septets) σύμφωνα πάλι με το GSM αλφάβητο.
Όταν λοιπόν κατά τη λειτουργία της εφαρμογής το modem λάβει κάποιο μήνυμα από τον κινητό σταθμό εκτελούνται κατά αναλογία τα αντίθετα βήματα που περιγράφηκαν στη διαδικασία της κωδικοποίησης και έτσι ανακτάται το αρχικό μήνυμα.
Όπως και με τον αλγόριθμο κωδικοποίησης έτσι θα αναλύσουμε στη συνέχεια της πτυχιακής και τον κώδικα σε C της αποκωδικοποίησης

AT Εντολές

Όπως γίνεται με όλες τις συσκευές modem, αλλά και τα κινητά, έτσι και σε αυτό που χρησιμοποιούμε στην εφαρμογή υπάρχει η δυνατότητα εκτέλεσης κάποιων λειτουργιών. Για παράδειγμα η δυνατότητα να διαβαστεί ένα γραπτό μήνυμα που υπάρχει σε κάποια θέση μνήμης, να συνταχθεί και να αποσταλεί κάποιο μήνυμα η διαχείριση του τηλεφωνικού καταλόγου, ή η μέτρηση της έντασης του λαμβανόμενου

σήματος είναι διαδικασίες που γίνονται εφικτές με τη χρήση AT εντολών (AT commands). Πολλές από τις εντολές που χρησιμοποιούνται στα ενσύρματα modem όπως ATD (κλήση), ATA (απάντηση), ATH (τερματισμός κλήσης), υποστηρίζονται και από τα modem GSM. Τα γράμματα AT, που προέρχονται από τη λέξη Attention που σημαίνει προσοχή, δηλώνουν στο modem ότι ακολουθεί το κύριο μέρος που πρέπει να εκτελεσθεί. Είναι εντολές με συγκεκριμένη δομή αναγνωρίσιμες και εκτελέσιμες από το modem που αφορούν τις περισσότερες από τις λειτουργίες τις συσκευής. Η σύνταξη γίνεται γράφοντας πρώτα το πρόθεμα AT ακολουθούμενο από το σύμβολο της πρόσθεσης (+) και το όνομα κάθε εντολής. Στη συνέχεια ακολουθεί το σύμβολο της ισότητας (=) καθώς και τα ορίσματα που τυχόν δέχεται η εντολή. Το τέλος της εντολής σηματοδοτείται από τον χαρακτήρα [CR] (Carriage Return) που ταυτόχρονα σημαίνει την έναρξη εκτέλεσης της εντολής. Ακολουθεί η μορφή και η λειτουργία κάθε εντολής που χρησιμοποιήθηκε στην εφαρμογή.

AT+CMGS

Η πιο συχνά χρησιμοποιούμενη στη συγκεκριμένη εφαρμογή εντολή με την οποία γίνεται η αποστολή ενός σύντομου γραπτού μηνύματος. Απαραίτητη προϋπόθεση είναι να έχουμε συντάξει το μήνυμά μας στη PDU μορφή του με το κωδικοποιημένο μήνυμα και τις απαραίτητες πληροφορίες που απαιτούνται για την μετάδοσή του.

Σύνταξη: + CMGS=<length><CR><pdu><CTRL-Z (/ESC)>

Η παράμετρος <length> είναι η τιμή η οποία δείχνει το πλήθος των octets της ακολουθίας.
Ο χαρακτήρας <CTRL-Z (/ESC)> στο τέλος είναι το αναγνωριστικό για την αποστολή (ή την ακύρωση αποστολής) του μηνύματος.

AT+CNMI

Με αυτή την εντολή επιλέγουμε τον τρόπο με τον οποίο το modem θα ενημερώνει το τερματικό με το οποίο συνδέεται (TE, Terminal Equipment) και άρα και τον μικροελεγκτή για την λήψη ενός νέου μηνύματος.

Σύνταξη: +CNMI=<index> (=1,2)

ATE

Η εντολή αυτή χρησιμοποιείται για την αποφυγή του φαινομένου της επιστροφής χαρακτήρων (echo) κατά την διαδικασία αποστολής εντολών προς το modem από τον μικροελεγκτή.

Σύνταξη: ate0

AT+CMGD

Η εντολή CMGD την χρησιμοποιείται για την διαγραφή σύντομων γραπτών μηνυμάτων.

Σύνταξη: +CMGD=<index>

Η παράμετρος <index> υποδεικνύει τη θέση του γραπτού μηνύματος που θα διαγραφεί.

AT+CMGF

Με την συγκεκριμένη εντολή υποδεικνύουμε στο modem ποια μορφή του sms θα χρησιμοποιήσει. Δηλαδή την text ή pdu μορφή για τα γραπτά μηνύματα.

Σύνταξη: +CMGF=<index> (=0)

Η παράμετρος <index> υποδεικνύει τη χρήση pdu μορφής αν είναι ίσο με 0 (είναι η προεπιλεγμένη μορφή) ή text μορφής αν έχει την τιμή 1.

Μορφή αποστολής μηνύματος

Η δομή ενός μηνύματος εντολής για τον έλεγχο του συστήματος θα πρέπει να έχει κάθε φορά την μορφή που παρουσιάζεται στο ακόλουθο σχήμα.

Για την χρήση του συστήματος πρέπει να ενημερωθεί ο χρήστης για την δομή του μηνύματος που πρέπει να αποστείλει στον σταθμό βάσης για να εκτελέσει την εντολή. Αν και είναι απλή έστω και ένα κενό παραπάνω να εισαχθεί στο μήνυμα θα λάβει απαντητικό μήνυμα λάθους
Όπως παρατηρούμε ένα μήνυμα εντολής ξεκινά πάντα με το τμήμα κωδικού. Ο κωδικός είναι ένας τετραψήφιος αριθμός που καθορίζεται κατά την διαδικασία προγραμματισμού και για την συγκεκριμένη εφαρμογή επιλέχτηκε το νούμερο «1234».
Το τμήμα του κωδικού ακολουθεί το τμήμα εντολής με την μεσολάβηση απαραίτητα ενός και μόνο χαρακτήρα κενού ο οποίος θα διαχωρίζει τα δύο τμήματα.
Ακολουθεί η λέξη relay (που αναφέρεται στο ρελέ που θα ενεργοποιήσει ή απενεργοποιήσει την συσκευή) και άλλος ένας χαρακτήρας κενού που χωρίζει τη λέξη relay με την εντολή on ή off, ανάλογα την κατάσταση που θέλουμε να έχει η συσκευή.

Δεν έχει σημασία αν τα γράμματα θα είναι κεφαλαία ή μικρά καθώς φροντίζει η εφαρμογή να τα μετατρέπει όλα σε κεφαλαία, ώστε να υπάρχουν λιγότερα λάθη. Πρέπει όμως να παραμείνει αυτή η δομή για την σωστή αναγνώριση της εντολής.

ΚΕΦΑΛΑΙΟ 3- Μικροελεγκτές οικογένειας PIC

Μικροελεγκτές και Μικροεπεξεργαστές

Είναι φανερό ότι οι μικροελεγκτές (microcontrollers) και οι μικροεπεξεργαστές (microprocessors) αποτελούν απαραίτητο μέρος των σύγχρονων τεχνολογικών εφαρμογών. Πιο συγκεκριμένα, ο μικροεπεξεργαστής αποτελεί το κεντρικό στοιχείο σε κάθε μικροϋπολογιστικό σύστημα. Ωστόσο, οι δύο αυτές συσκευές διαφέρουν μεταξύ τους σε αρκετά σημεία.
Μια βασική τους διαφορά, και η ποιο σημαντική, βρίσκεται στην λειτουργικότητα τους. Προκειμένου να λειτουργήσει ένας μικροεπεξεργαστής θα πρέπει να συνδεθεί και με άλλες συσκευές, όπως μνήμη (memory) ή συσκευή αποστολής και λήψης δεδομένων. Αυτό σημαίνει ότι ένας μικροεπεξεργαστής είναι η καρδιά του συστήματος. Αντιθέτως, ένας μικροελεγκτής σχεδιάζετε με τέτοιο τρόπο ώστε να περιέχει όλες τις παραπάνω συσκευές. Συνεπώς, δεν χρειάζονται άλλες συσκευές για την λειτουργία του, εφόσον όλα τα απαραίτητα περιφερειακά (peripherals) είναι ενσωματωμένα μέσα του. Με τον τρόπο αυτό, εξοικονομούμε χώρο και χρόνο, κατά την κατασκευή ενός μικροελεγκτή.

Αρχιτεκτονικές Harvard και Von-Neumann

Οι αρχιτεκτονικές Harvard και Von-Neumann αποτελούν τις δύο βασικές αρχιτεκτονικές των σύγχρονων μικροϋπολογιστικών συστημάτων. Οι μικροελεγκτές που είναι σχεδιασμένοι με βάση την αρχιτεκτονική Harvard καλούνται επίσης και μικροελεγκτές RISC (Reduced Instruction Set Computer) ενώ εκείνοι που χρησιμοποιούν την αρχιτεκτονική Von-Neumann καλούνται μικροελεγκτές CISI (Complex Instruction Set Computer).
Στην αρχιτεκτονική Harvard υπάρχει διαφορετικός δίαυλος για τη μεταφορά δεδομένων (data bus) και διαφορετικός δίαυλος για τη μεταφορά των εντολών (instruction bus). Η ύπαρξη δύο διαφορετικών μνημών, μνήμη δεδομένων (data memory) και μνήμη προγράμματος (program memory), καθιστά την αρχιτεκτονική Harvard πιο αποδοτική, αφού μπορεί να εκτελείται κάποια εντολή και παράλληλα να εγγράφεται ή να διαβάζεται η μνήμη. Με τον τρόπο αυτό επιτυγχάνεται η εκτέλεση της εντολής σε ένα μόνο χρόνο μηχανής.
Επίσης, η αρχιτεκτονική Harvard επιτρέπει οι εντολές να έχουν διαφορετικό μήκος σε δυαδικά ψηφία (binary digits, bit) από τα δεδομένα. Δίνεται η δυνατότητα να επιλέγεται, ανάλογα με το πλήθος των εντολών, το κατάλληλο μήκος της λέξης εντολής ώστε να επιτευχθεί η κωδικοποίηση της κάθε εντολής σε μία μόνο λέξη. Πετυχαίνεται με αυτό τον τρόπο να μειωθεί σημαντικά η ταχύτητα ανάκλησης (fetch) της κάθε εντολής. Είναι τέλος ενδεικτικό της αρχιτεκτονικής αυτής ο μειωμένος αριθμός εντολών καθώς και η εκτέλεση της κάθε εντολής σε ένα μόνο χρόνο Μηχανής.

Γενικά χαρακτηριστικά των μικροελεγκτών της οικογένειας PIC

Ο όρος PIC (Peripheral Interface Controller), όπως είναι το πλήρες όνομα τους, αναφέρεται στην οικογένεια 8-bit μικροελεγκτών της εταιρείας Microchip.
Η δομή τους στηρίζεται στην αρχιτεκτονική Harvard. Αυτό το χαρακτηριστικό τους σε συνδυασμό και με άλλα χαρακτηριστικά που αναφέρονται παρακάτω και που συναντώνται σε μικροελεγκτές RISC, τους μετατρέπουν σε συσκευές με αρκετά υψηλή επίδοση.
Χαρακτηριστικό των PIC είναι ότι για την εκτέλεση μίας εντολής χρειάζεται μόνο ένας κύκλος μηχανής (εκτός των εντολών που αλλάζουν την ροή του προγράμματος). Η ανάκληση μιας εντολής χρειάζεται επίσης μόνο ένα κύκλο μηχανής. Οι PIC διαθέτουν επιπλέον μια απλή μονάδα συνεχούς διοχέτευσης (pipeline) με την οποία πετυχαίνουν την εκτέλεση μιας εντολής ανά κύκλο μηχανής χωρίς να χρειάζεται ιδιαίτερα πολύπλοκη αρχιτεκτονική. Άλλο σημαντικό χαρακτηριστικό των PIC είναι ότι όλες οι εντολές επιτρέπεται να εκτελούνται σε οποιοδήποτε καταχωρητή (register) ακόμα και σε καταχωρητές ειδικού σκοπού. Παραδείγματος χάρη επιτρέπονται λογικές πράξεις με όρισμα το μετρητή προγράμματος (Program Counter, PC) ή τον καταχωρητή κατάστασης (STATUS register). Το γεγονός ότι δεν υπάρχουν ειδικές περιπτώσεις στη διαχείριση των εντολών σε συνδυασμό με τον μικρό αριθμό επιτρέπει την εύκολη και γρήγορη εκμάθηση τους.
Στη δομή ενός PIC διακρίνουμε τρία μέρη: Τον πυρήνα (Core), τα περιφερειακά (Peripherals) και τα ειδικά χαρακτηριστικά (Special Features).

Πυρήνας (Core)

Ο πυρήνας περιέχει όλες τις απαραίτητες συσκευές για την λειτουργία του μικροεπεξεργαστή, όπως τον ταλαντωτή (Oscillator), τα απαραίτητα κυκλώματα για τη σωστή εκκίνηση του μικροελεγκτή (Reset logic), την κεντρική μονάδα επεξεργασίας (Central Processing Unit, CPU), την αριθμητική μονάδα (Arithmetic Logic Unit, ALU), την μνήμη (Memory), τη λογική διακοπών (Interrupt operation) καθώς και το σύνολο των εντολών (Instruction Set).

Περιφερειακά (Peripherals)

Τα περιφερειακά είναι το πιο ενδιαφέρον κομμάτι ενός μικροελεγκτή αφού αποτελούν το σημαντικότερο στοιχείο για να αποφανθούμε εάν ο συγκεκριμένος μικροελεγκτής είναι κατάλληλος για την εφαρμογή μας. Εξάλλου, τα περιφερειακά είναι εκείνα που διαφοροποιούν τους μικροελεγκτές από τους μικροεπεξεργαστές. Οι μικροελεγκτές της οικογένειας PIC έρχονται με διαφορετικούς συνδυασμούς περιφερειακών ανάλογα με την συσκευή που επιλέγουμε. Τέτοια περιφερειακά είναι οι θύρες εισόδου, εξόδου (I/O Ports), οι χρονιστές (timers) και οι σειριακές θύρες (USART). Ο PIC που κατασκευάστηκε ενσωματώνει όλα τα προαναφερόμενα περιφερειακά.

Ειδικά Χαρακτηριστικά (Special Features)

Τα ειδικά χαρακτηριστικά είναι απαραίτητα για την εξυπηρέτηση των παρακάτω σκοπών :

- Ελάττωση του κόστους του συστήματος,
- Αύξηση της αξιοπιστίας του συστήματος και,
- Αύξηση της προσαρμοστικότητας σχεδιασμού (design flexibility).

Η Mid-Range οικογένεια περιλαμβάνει διάφορα τέτοια χαρακτηριστικά, όπως είναι ο Watchdog Timer και ο ενσωματωμένος σειριακός προγραμματιστής (In-Circuit Serial Programmer, ICSP)

Ανάλογα με το μήκος της εντολής που χρησιμοποιείται, διακρίνουμε τρεις υποοικογένειες μικροελεγκτών PIC:

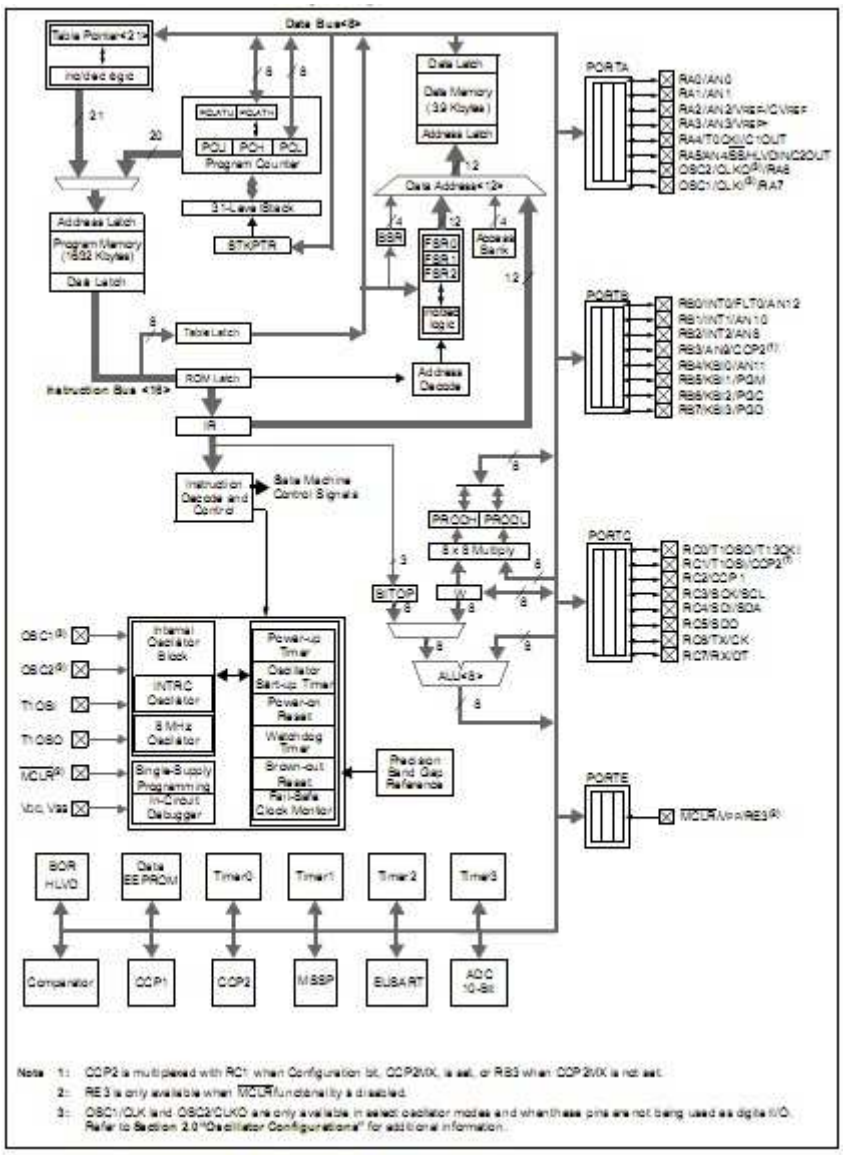
- Τη "βασική" (Base-Line) με μήκος λέξης εντολής των 12-bit.
- Τη "μεσαία" (Mid-Range) με μήκος λέξης εντολής των 14-bit.
- Την "προηγμένη" (High-End) με μήκος λέξης εντολής των 16-bit.

Μικροελεγκτές "μεσαίας" (Mid-Range) οικογένειας

Οι μικροελεγκτές της οικογένειας Mid-Range αναφέρονται επίσης και ως συσκευές της οικογένειας 16CXXX PICmicro MCU. Ο μικροελεγκτής PIC που σχεδιάστηκε στην διπλωματική ανήκει στη κατηγορία αυτή. Για το λόγο αυτό, είναι σημαντικό στο σημείο αυτό να περιγραφεί η αρχιτεκτονική αυτής της κατηγορίας, τα γενικά χαρακτηριστικά και τα ειδικά γνωρίσματα.
Όπως αναφέρθηκε και προηγουμένως, στην αρχιτεκτονική των PIC διακρίνουμε τρία κυρίως τμήματα, τον πυρήνα, τα περιφερειακά και κάποια επιπλέον τμήματα που βοηθούν στον γρηγορότερο σχεδιασμό και στην αποτελεσματικότερη λειτουργία της συσκευής (special features).
Συνεπώς, στην ενότητα αυτή, θα μελετηθούν αναλυτικότερα κάποια από αυτά.

Αρχιτεκτονική του PIC

Στο σχήμα που ακολουθεί, παρουσιάζεται η αρχιτεκτονική του PIC. Μπορούμε να διακρίνουμε την Αριθμητική Λογική Μονάδα, την Μνήμη Προγράμματος, την Μνήμη Δεδομένων, διάφορους καταχωρητές, καθώς και τα περιφερειακά του PIC, όπως είναι οι πόρτες I/O και οι Χρονιστές (Timers).



Εικόνα 11: Αρχιτεκτονική του PIC

Ο πυρήνας του PIC

Κεντρική Μονάδα Επεξεργασίας (Central Processing Unit, CPU)

Η Κεντρική Μονάδα Επεξεργασίας μπορεί να θεωρηθεί ως η "καρδιά" του PIC. Είναι υπεύθυνη για την σωστή μεταφορά της εντολής που πρόκειται να εκτελεστεί, για την αποκωδικοποίησή της (decoding), και για την εκτέλεσή της (executing).
Σε μερικές περιπτώσεις, η κεντρική μονάδα επεξεργασίας χρειάζεται να λειτουργήσει σε συνδυασμό με την Αριθμητική Λογική Μονάδα ώστε να συμπληρωθεί η εκτέλεση μιας εντολής (σε αριθμητικές και λογικές πράξεις).
Η CPU ελέγχει τον δίαυλο επικοινωνίας με την μνήμη προγράμματος, τον δίαυλο επικοινωνίας με την μνήμη δεδομένων και την πρόσβαση στη στοίβα (stack).
Οι βασικές λειτουργίες που εκτελεί μια CPU είναι:

- Διαβάζει εντολές από την μνήμη, τις αποκωδικοποιεί και τις εκτελεί
- Ελέγχει το όλο σύστημα παρέχοντας τα απαραίτητα προς αυτό σήματα. Έτσι, μεταφέρει δεδομένα από και προς την μνήμη καθώς επίσης από και προς τις μονάδες εισόδου / εξόδου
- Ανταποκρίνεται σε σήματα διακοπών και ελέγχου
- Διακλαδώνει την ομαλή ακολουθιακή ροή ενός προγράμματος σε άλλο σημείο, σε υπορουτίνα, επιστρέφει από υπορουτίνα και αποκρίνεται σε διακοπές από εξωτερικά σήματα ή από το πρόγραμμα.

Ρολόι - Χρονισμοί - Κύκλος Εντολής

Οι παλμοί που παράγονται από τον ταλαντωτή (OSC1) διαιρούνται εσωτερικά με το 4 για να δώσουν τέσσερις μη υπερκαλυπτόμενους παλμούς Q1, Q2, Q3, Q4. Οι παλμοί αυτοί χρησιμοποιούνται από τον πυρήνα για να συγχρονιστούν οι διάφορες λειτουργίες κατά τη διάρκεια ανάκλησης και εκτέλεσης μιας εντολής. Για παράδειγμα η ανάκληση της εντολής (fetch) ξεκινάει με τον Program counter να αυξάνει κατά την φάση Q1. Κατά την εκτέλεση η εντολή αποθηκεύεται στο καταχωρητή εντολών, Instruction Register (IR), κατά την διάρκεια της φάσης Q1. Η εντολή αποκωδικοποιείται και εκτελείται στις φάσεις Q2, Q3, Q4. Τα δεδομένα διαβάζονται από τη μνήμη κατά την φάση Q2, και γράφονται κατά την Q4. Κατά την φάση Q3 γίνεται η επεξεργασία των δεδομένων. Ακολουθεί μια σχηματική περιγραφή των παραπάνω :

Δραστηριότητα Q κύκλου (Fetch Instruction)

Q_1	Q_2	Q_3	Q_4
Ανάκληση Εντολής	Καμία Λειτουργ'α	Καμία Λειτουργ'α	Καμία Λειτουργία

Δραστηριότητα Q κύκλου (Execute Instruction

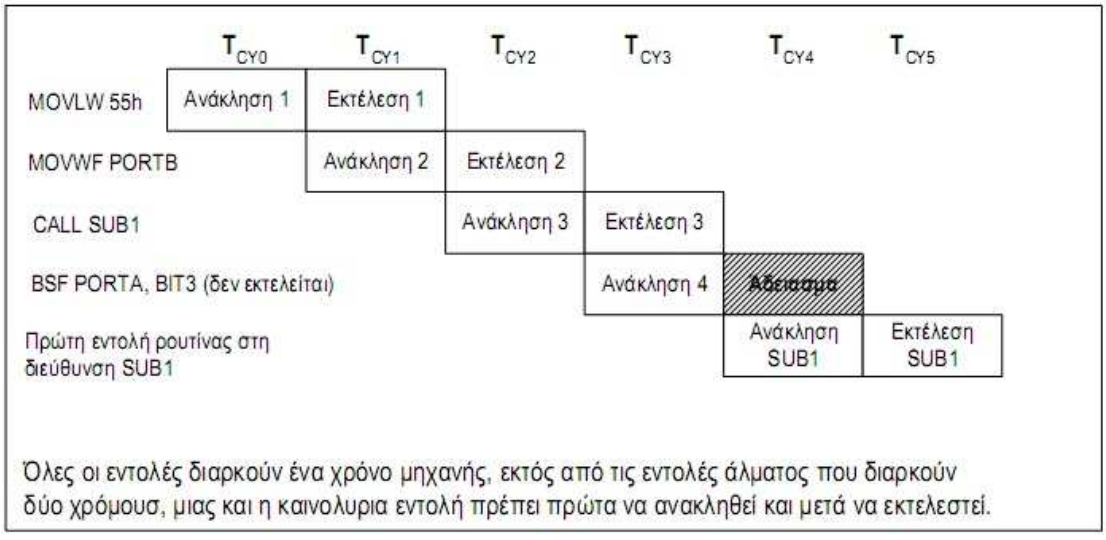
Q_1	Q_2	Q_3	Q_4
Αποκωδικο-ποιηση	Ανάγνωση Μνήμης	Επεξεργασί αΔεδομένω	Εγγραφή στο

Δραστηριότητα Q κύκλου (Q cycle activity)

Το σύνολο των τεσσάρων παλμών Q1 ως Q4 αποτελούν ένα κύκλο εντολής (ή κύκλο μηχανής). Μέσα σε ένα κύκλο εντολής εκτελείται μια οποιαδήποτε εντολή. Εάν υπολογίσουμε ότι για την εκτέλεση μιας εντολής χρειαζόμαστε ένα κύκλο μηχανής καθώς και έναν επιπλέον κύκλο μηχανής για την ανάκληση της εντολής έχουμε ότι συνολικά για την ανάκληση και την εκτέλεση μιας εντολής χρειαζόμαστε δύο κύκλους μηχανής. Όπως θα δούμε και στη συνέχεια με τη βοήθεια της μονάδας συνεχούς διοχέτευσης (pipeline) πετυχαίνουμε το σύνολο της ανάκλησης και εκτέλεσης μιας εντολής να φαίνεται ότι διαρκεί μόνο ένα κύκλο μηχανής. Συμπεραίνουμε λοιπόν ότι εάν έχουμε ένα PIC που δουλεύει με ένα κρύσταλλο των 4 MHz εκτελεί εντολές με ρυθμό 1 εκατομμύριο εντολές το δευτερόλεπτο, ή ότι η διάρκεια μιας εντολής είναι 1 μs (= 4 / 4MHz).

Μονάδα συνεχούς διοχέτευσης εντολών (Instruction Pipelining)

Ο κύκλος εντολής αποτελείται από τέσσερις Q φάσεις (Q1, Q2, Q3, Q4). Η αναζήτηση της εντολής διαρκεί ένα κύκλο εντολής και η εκτέλεση της άλλο ένα κύκλο εντολής. Εξαιτίας της μονάδας συνεχούς διοχέτευσης εντολών όμως, η εκτέλεση της εντολής διαρκεί ένα μόνο κύκλο εντολής αφού η ανάκληση της επόμενης εντολής γίνεται όσο εκτελείται η προηγούμενη της. Εξαίρεση αποτελούν όλες εκείνες οι εντολές που αλλάζουν τον μετρητή προγράμματος (PC) μιας και η επόμενη εντολή που περιμένει στην ουρά του pipeline δεν είναι αυτή που θα εκτελεστεί. Στην περίπτωση αυτή η εντολή διαρκεί δύο κύκλους εντολής. Στον πρώτο κύκλο γίνεται η εκτέλεση της εντολής δηλαδή η αλλαγή του μετρητή προγράμματος, και στον επόμενο γίνεται ανάκληση της σωστής εντολής και εκτέλεση μιας εντολής NOP (No OPeration) που ισοδυναμεί με άδειασμα του buffer της μονάδας συνεχούς διοχέτευσης.
Στη συνέχεια κολουθεί ένα παράδειγμα εκτέλεσης τμήματος ενός προγράμματος. Το σχήμα που ακολουθεί μας δείχνει σε ποιο κύκλο μηχανής ανακαλείται κάθε εντολή και σε ποιο κύκλο εντολής εκτελείται. Παρατηρούμε ότι μετά από την κλήση μιας υπορουτίνας με την εντολή CALL, η ουρά της συνεχούς διοχέτευσης (pipeline) αδειάζει και εκτελείται μια εντολή NOP. Ο επόμενος κύκλος ξεκινά με την ανάκληση της πρώτης εντολής της υπορουτίνας.

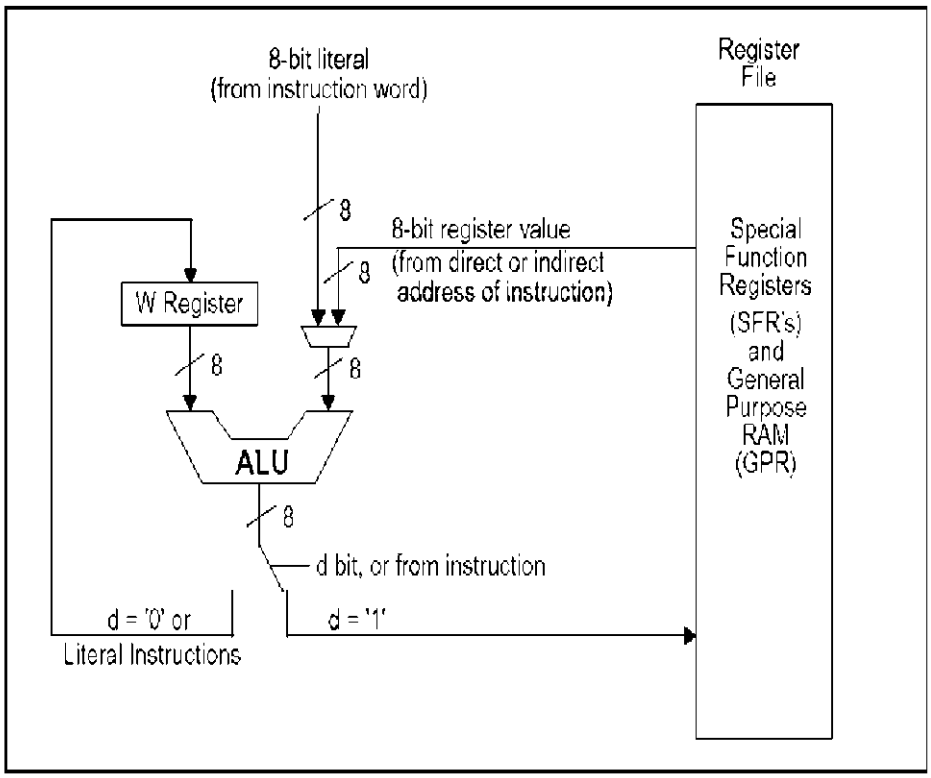


Εικόνα 12: Παράδειγμα συνεχούς διοχέτευσης εντολών (pipelining)

Αριθμητική Λογική Μονάδα (ALU)

Οι PIC της μεσαίας οικογένειας περιέχουν μια Αριθμητική Λογική Μονάδα των 8 bit. Η ALU είναι μιας γενικής χρήσης αριθμητική και λογική μονάδα. Είναι υπεύθυνη για αριθμητικές και λογικές πράξεις μεταξύ του δεδομένου στον καταχωρητή εργασίας (W register) και οποιουδήποτε άλλου καταχωρητή. Σε κάθε εντολή το αποτέλεσμα

μπορεί να μεταφερθεί είτε στον καταχωρητή ο οποίος συμμετέχει στην πράξη είτε στον W register (ανάλογα με το d bit του Instruction Register). Στο σχήμα που ακολουθεί παρουσιάζεται η ALU του PIC.



Εικόνα 13: Αριθμητική Λογική Μονάδα (ALU) του PIC

Η ALU έχει εύρος 8 bit και μπορεί να εκτελέσει πράξεις πρόσθεσης, αφαίρεσης, ολίσθησης και λογικές πράξεις. Όλες οι αριθμητικές λειτουργίες είναι της μορφής συμπληρώματος ως προς 2. Σε περίπτωση εντολής με δύο τελεστές (operands), ο ένας είναι ο καταχωρητής εργασίας W ενώ ο άλλος μπορεί να είναι είτε οποιοσδήποτε άλλος καταχωρητής, είτε κάποιο απ' ευθείας δεδομένο (literal). Σε εντολές με έναν τελεστή, αυτός μπορεί να είναι ή ο W register ή κάποιος άλλος καταχωρητής.
Ο καταχωρητής εργασίας W είναι ένας καταχωρητής εύρους 8 bit και ο οποίος δεν είναι διευθυνσιοδοτημένος στην μνήμη δεδομένων. Το αποτέλεσμα της ALU αποθηκεύεται στον W register.
Αναλόγως την εντολή που εκτελείται, η αριθμητική λογική μονάδα μπορεί να επηρεάσει την τιμή του Carry (C), του Digit Carry (DC) και του Zero (Z) που περιέχονται στα τρία λιγότερο σημαντικά ψηφία (Less Significant Bit, LSBs) του καταχωρητή κατάστασης STATUS.

Διακοπές (Interrupts)

Οι PIC υποστηρίζουν ένα μεγάλο αριθμό διακοπών. Οι διακοπές κατά πλειοψηφία παράγονται από τις περιφερειακές μονάδες. Υπάρχουν και περιφερειακές μονάδες οι οποίες μπορούν να παράγουν περισσότερες από μία διακοπές, όπως είναι ο πομπός / δέκτης σύγχρονης σειριακής επικοινωνίας (Universal Synchronous Asynchronous Receiver Transmitter, USART)
Οι σημαντικότερες διακοπές είναι:

- Εξωτερική διακοπή (external interrupt)
- Διακοπή από υπερχείλιση του χρονιστή timer0 (TMR0 Overflow Interrupt)
- Διακοπή όταν αλλάζει στάθμη ένας από τους ακροδέκτες PORTB (PORTB change interrupt)
- Διακοπή παράλληλης θύρας (Parallel Slave Port Interrupt)
- Διακοπές σειριακής θύρας (USART Interrupts)
- Διακοπή σειριακής λήψης (Receive Interrupt)
- Διακοπή σειριακής αποστολής (Transmit Interrupt)
- Διακοπή υπερχείλισης του χρονιστή 1 (Timer1 Overflow Interrupt)
- Διακοπή υπερχείλισης του χρονιστή 2 (Timer2 Overflow Interrupt)

Γενική Αρχιτεκτονική των Διακοπών

Η κάθε διακοπή ελέγχεται από δύο bit, το intnameIE (Interrupt Enable bit) και το intnameIF (Interrupt Flag Bit). Το intnameIE bit ενεργοποιεί την αντίστοιχη διακοπή ενώ το intnameIF σηματοδοτεί ότι υπάρχει διακοπή προς εξυπηρέτηση.
Το bit intnameIF γίνεται ένα (1) ανεξάρτητα, με το αν είναι ενεργοποιημένη η αντίστοιχη διακοπή ή όχι, δηλαδή αν το intnameIE είναι ένα (1) ή μηδέν (0). Το intnameIF δε γίνεται από μόνο του μηδέν μόλις εξυπηρετηθεί η διακοπή που προκάλεσε, και πρέπει να μηδενίζεται από το πρόγραμμα της ρουτίνας εξυπηρέτησης γιατί διαφορετικά θα προκληθεί νέα διακοπή. Το ίδιο bit μπορεί να χρησιμοποιηθεί χωρίς την χρήση διακοπών για να αναγνωρίζουμε το γεγονός που περιγράφει η αντίστοιχη διακοπή. Η διαδικασία αυτή γίνεται εξετάζοντας την τιμή του συγκεκριμένου bit κατά τακτά χρονικά διαστήματα (polling).
Τα διαφορά bit ελέγχου των διακοπών περιέχονται στους καταχωρητές: INTCON PIE1 PIR1

Ο INTCON είναι ο βασικότερος καταχωρητής που σχετίζεται με τις διακοπές και υλοποιείται σε κάθε συσκευή PIC αφού περιέχει τα bit ελέγχου των διακοπών. Στο παρακάτω σχήμα φαίνεται ο καταχωρητής INTCON.

Το GIE ενεργοποιεί τις διακοπές αν είναι ένα (1) ενώ τις απενεργοποιεί αν είναι μηδέν (0).
Το PEIE αντίστοιχα, ενεργοποιεί όλες τις διακοπές από τα περιφερειακά όταν είναι ένα (1), ενώ τις απενεργοποιεί όταν είναι μηδέν (0).
Το T0IE ενεργοποιεί την διακοπή του Timer 0.
Το INTE ενεργοποιεί την διακοπή του εξωτερικού ακροδέκτη INT.

Το RBIE ενεργοποιεί την διακοπή που συμβαίνει κάθε φορά που αλλάζει ένα από τα τέσσερα σημαντικότερα pins της PORTB (PORTB change interrupt) και εφόσον αυτά έχουν τεθεί ως είσοδοι.
Τα bit T0IF, INTF, RBIF αποτελούν τις αντίστοιχες σημαίες που δείχνουν πότε συμβαίνει κάποια από τις παραπάνω διακοπές.
Όταν συμβαίνει μια διακοπή, το bit GIE μηδενίζεται αυτόματα ώστε να μη μπορούν να προκληθούν και άλλες διακοπές. Η διεύθυνση επιστροφής που βρίσκεται στον PC προωθείται στη στοίβα και ο PC παίρνει την τιμή 0004h. Η διεύθυνση εκκίνησης της ρουτίνα εξυπηρέτησης των διακοπών, ή διάνυσμα διακοπών (Interrupt Vector), είναι κοινή για όλες τις διακοπές και είναι η διεύθυνση 0004h. Το αίτιο της διακοπής καθορίζεται από τις σημαίες των διακοπών intnameIF, που βρίσκονται στους καταχωρητές INTCON και PIR. Το πρόγραμμα επιστρέφοντας από μία ρουτίνα εξυπηρέτησης διακοπής με την εντολή RETFIE, θέτει ξανά το GIE με αποτέλεσμα να εκτελείται οποιαδήποτε διακοπή περιμένει (pending Interrupt). Κατά την εκκίνηση ή επανεκκίνηση του μικροελεγκτή το GIE μηδενίζεται και η διακοπές είναι απενεργοποιημένες

Τα περιφερειακά του PIC

Γενικής χρήσης μονάδες εισόδου-εξόδου I/O (Ports)

Οι γενικής χρήσης θύρες εισόδου-εξόδου, (I/O ports), είναι τα πιο απλά περιφερειακά. Οι θύρες αυτές είναι διπλής κατεύθυνσης και η κατεύθυνση του κάθε ακροδέκτη, δηλαδή το αν ένας ακροδέκτης λειτουργεί ως είσοδος ή ως έξοδος, ελέγχεται από τον καταχωρητή ελέγχου κατεύθυνσης που καλείται TRIS. Ο καταχωρητής TRISx ελέγχει αντίστοιχα τη διεύθυνση στην θύρα PORTx. Εάν κάποιο bit του καταχωρητή TRISx είναι μονάδα τότε ο αντίστοιχος ακροδέκτης της θύρας συμπεριφέρεται ως είσοδος, ενώ αν το bit είναι μηδέν ο ακροδέκτης συμπεριφέρεται ως έξοδος.
Ο καταχωρητής PORTx περιέχει τα δεδομένα εξόδου της θύρας. Όταν διαβάζουμε τα δεδομένα του καταχωρητή PORTx δε διαβάζουμε τον ίδιο τον καταχωρητή αλλά ότι εμφανίζεται στους ακροδέκτες της θύρας. Θα πρέπει να προσέχουμε με εντολές που διαβάζουν, αλλάζουν και στη συνέχεια γράφουν το τελικό αποτέλεσμα πίσω στον καταχωρητή όπως συμβαίνει με τις εντολές BSF και BCF.
Για οικονομία στο πλήθος των ακροδεκτών, οι είσοδοι και οι έξοδοι των περιφερειακών του PIC, όπως είναι ο A/D μετατροπέας, οι σειριακές θύρες κτλ., χρησιμοποιούν τους ίδιους ακροδέκτες με τις ψηφιακές θύρες. Το περιφερειακό αποφασίζει τον τρόπο που λειτουργεί ο ακροδέκτης που χρησιμοποιεί και μπορεί να παρακάμψει τη λειτουργικότητα του καταχωρητή TRIS. Άλλες φορές είναι απαραίτητο ο καταχωρητής TRIS να είναι σωστά ρυθμισμένος για να λειτουργήσει το περιφερειακό.

PORTB - Οι καταχωρητές PORTB και TRISB

Η θύρα Β είναι μια θύρα 8 δυαδικών ψηφίων, διπλής κατεύθυνσης. Η κατεύθυνση των ακροδεκτών, δηλαδή το αν ένας ακροδέκτης συμπεριφέρεται ως είσοδος ή ως έξοδος, καθορίζεται από τον καταχωρητή TRISB. Κάθε ακροδέκτης από την PORTB έχει ένας μικρής ισχύος pull-up διακόπτης. Μηδενίζοντας το bit RPBU μπορούμε να ενεργοποιήσουμε όλα τα pull-up. Το pull-up κομμάτι απενεργοποιείται αυτόματα όταν ο ακροδέκτης ρυθμίζεται ως έξοδος καθώς και μετά από ένα power-on reset. Το υψηλότερης αξίας ψηφίο της θύρας Β, δηλαδή οι ακροδέκτες RB7:RB4 παρέχουν μια λειτουργία κατά την οποία προκαλείται διακοπή μόλις αλλάξει η τιμή στην είσοδό τους. Οι είσοδοι RB7:RB4 συγκρίνονται κάθε φορά με τις παλιές τιμές που βρίσκονται στον καταχωρητή PORTB. Οποιαδήποτε αλλαγή σε κάποιο από τα τέσσερα bit προκαλεί διακοπή (RB Port Change Interrupt), θέτοντας και τη σημαία RBIF (INTCON). Η διακοπή αυτή μπορεί να επαναφέρει τη συσκευή από την κατάσταση SLEEP. Για να σταματήσουμε την παραγωγή της διακοπής θα πρέπει να διαβάσουμε ή να γράψουμε κατάλληλη τιμή στον καταχωρητή PORTB και στη συνέχεια να σβήσουμε τη σημαία RBIF. Με την παραπάνω διακοπή αλλά και με τη χρήση των εσωτερικών pull-up μπορούμε να χειριστούμε εύκολα ένα πληκτρολόγιο και να υλοποιήσουμε ακόμα πιο εύκολα την λειτουργία όπου η συσκευή θα επανέρχεται από την κατάσταση ύπνου μόλις πατηθεί κάποιο πλήκτρο

Χρονιστές (Timers)

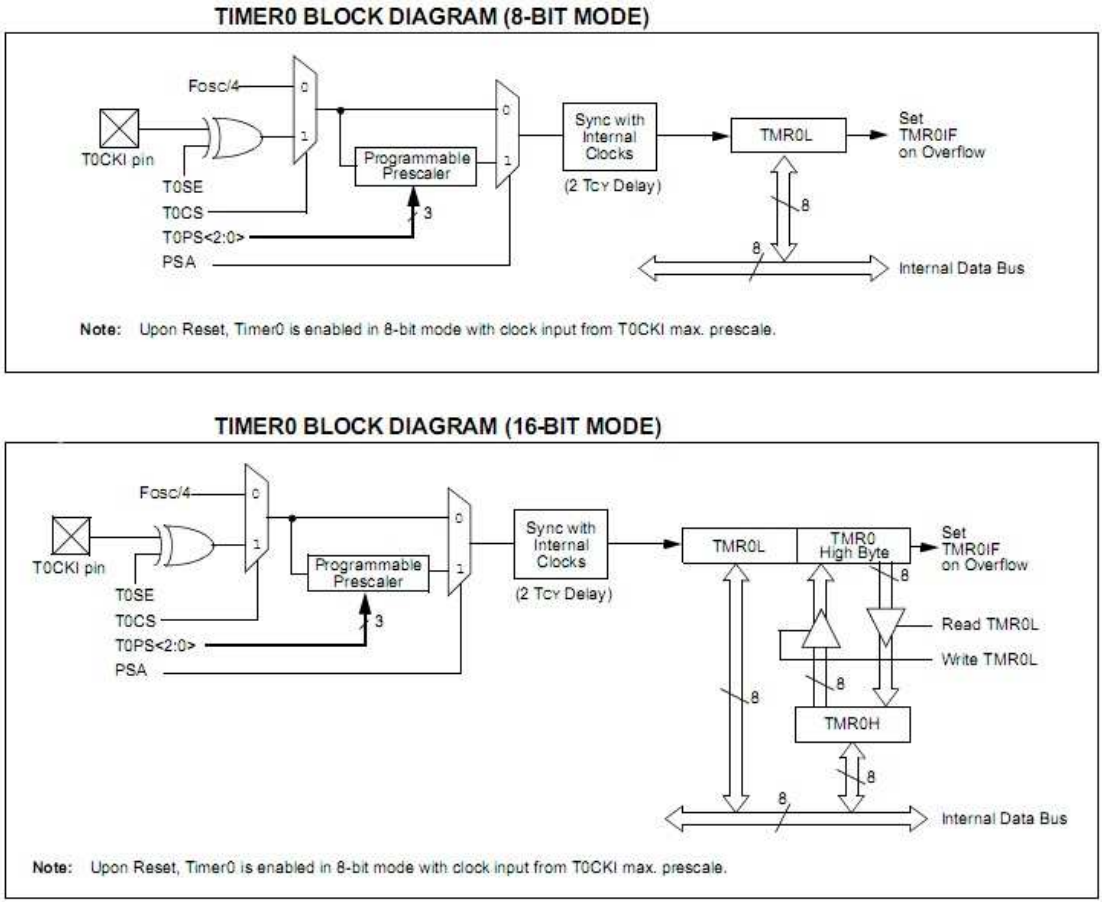
Οι χρονιστές (timers) είναι περιφερειακές συσκευές που αυξάνουν ή μειώνουν περιοδικά την τιμή ενός μετρητή σύμφωνα με τη συχνότητα ενός ρολογιού. Η οικογένεια PIC και ο 18F2520 υποστηρίζει τέσσερις διαφορετικούς τύπους χρονιστών: τον Timer0, τον Timer1, τον Timer2 και τον Timer3.
Οι χρονιστές αυτοί είναι αρκετά ευέλικτοι και παρέχουν δυνατότητες όπως διαίρεση της συχνότητας πριν ή μετά τον μετρητή (prescaling, postscaling), και λειτουργία ως μετρητή των παλμών που εμφανίζονται στην είσοδο ενός ακροδέκτη. Μπορούν επίσης να προκαλέσουν διακοπές σε τακτά χρονικά διαστήματα και παρέχουν και άλλα ιδιαίτερα χαρακτηριστικά όπως ο συγχρονισμός του σήματος εισόδου.

Timer 0

Παρακάμπτουμε τους υπόλοιπους χρονιστές και επικεντρώνουμε την ανάλυσή μας στον Timer0 τον οποίο χρησιμοποιούμε στην σχεδίαση μας. Σημειώνεται ότι και η λειτουργία των υπολοίπων εμφανίζει μεγάλες αναλογίες με εκείνη του Timer0.
Ο χρονιστής timer0 παρουσιάζει τα παρακάτω χαρακτηριστικά:

- Μετρητής των 8 ή 16 bit
- Είναι αναγνώσιμος (readable) και εγγράψιμος (writable)
- Επιλογέα κλίμακας χρονισμού (prescaler) που μπορεί να προγραμματιστεί μέσω λογισμικού (software)
- Επιλογή χρήσης εσωτερικού ή εξωτερικού ρολογιού
- Πρόκληση διακοπής (Interrupt) κατά την υπερχείλιση του μετρητή

Ακολουθεί το σχεδιάγραμμα αρχιτεκτονικής τοθ timer0



Εικόνα 14: Βασική αρχιτεκτονική timer0

Η λειτουργία του χρονιστή timer0 βασίζεται στην τιμή των ψηφίων που βρίσκονται στον καταχωρητή OPTION. Ο καταχωρητής αυτός είναι αναγνώσιμος και εγγράψιμος και περιέχει τα bit ελέγχου του timer0, του διαιρέτη μέτρησης (prescaler) και της εξωτερικής διακοπής (external INT Interrupt).
Ο χρονιστής timer0 μπορεί να λειτουργήσει είτε με το εσωτερικό ρολόι του PIC είτε με εξωτερικό ρολόι (external clock. Η επιλογή του ρολογιού λειτουργίας του χρονιστή γίνεται μέσω ενός πολυπλέκτη και με βάση την τιμή του bit TOCS. Αν το bit αυτό είναι 1 επιλέγεται το εξωτερικό ρολόι, διαφορετικά επιλέγεται το εσωτερικό ρολόι.
Η επιλογή χρήσης ή όχι του διαιρέτη μέτρησης (prescaler) γίνεται με παρόμοιο τρόπο (μέσω πολυπλέκτη) και το bit που καθορίζει την επιλογή αυτή είναι το PSA. Όταν το bit αυτό είναι 0 τότε ο prescaler χρησιμοποιείται από την μονάδα του χρονιστή ενώ όταν είναι 1 χρησιμοποιείται από τον Watchdog Timer.
Επομένως, ο μετρητής του χρονιστή μπορεί λειτουργεί είτε με το εσωτερικό ρολόι του PIC, είτε με εξωτερικό ρολόι, είτε με το ρολόι που δημιουργεί ο μετρητής του διαιρέτη μέτρησης (prescaler counter).
Τέλος, τα τρία λιγότερα σημαντικά ψηφία (LSBs) του καταχωρητή OPTION καθορίζουν τον ρυθμό λειτουργίας του διαιρέτη μέτρησης (prescaler).
Ο ρυθμός (prescaler rate) διαφέρει ανάλογα με το αν ο prescaler χρησιμοποιείται από τον timer0 ή από τον Watchdog Timer.

Διαιρέτης Μέτρησης (Prescaler)

Ο διαιρέτης μέτρησης (prescaler) αποτελεί την μονάδα που παράγει το ρολόι λειτουργίας του μετρητή του χρονιστή timerO (αν επιλεγεί). Ο prescaler δύναται να χρησιμοποιηθεί και από τον Watchdog Timer. Η επιλογή, όπως αναφέρθηκε και προηγουμένως, γίνεται μέσω του bit PSA. Όταν ο prescaler έχει ανατεθεί στον χρονιστή, μπορεί να παρέχει σε αυτόν διάφορους ρυθμούς μέτρησης.
Η λειτουργία του prescaler βασίζεται σε έναν μετρητή που αυξάνεται κατά 1 με κάθε παλμό του ρολογιού του PIC. Αναλόγως με την τιμή των τριών λιγότερο σημαντικών ψηφίων του καταχωρητή OPTION, επιλέγεται και το bit εκείνο του μετρητή που θα αποτελέσει το ρολόι εξόδου του prescaler. Με βάση αυτό το ρολόι λειτουργεί μετέπειτα ο χρονιστής timerO ή ο Watchdog Timer.

Πομπός / Δέκτης Ασύγχρονης / Σύγχρονης Σειριακής Επικοινωνίας (USART)

Ο πομπός / δέκτης Σύγχρονης / Ασύγχρονης σειριακής επικοινωνίας (Universal Sychronous Asynchronous Receiver Transmitter, USART) αποτελεί την μία από τις δύο μονάδες σειριακής εισόδου / εξόδου (I/O) του PIC. Η USART μπορεί να παραμετροποιηθεί σαν ένα σύστημα ασύγχρονης και ταυτόχρονης διπλής κατεύθυνσης (asynchronous full duplex) που μπορεί να επικοινωνήσει με περιφερειακές συσκευές, όπως ένα τερματικό (terminal) ή ένα προσωπικό υπολογιστή (personal computer) ή σαν ένα σύστημα σύγχρονης και μη ταυτόχρονης διπλής κατεύθυνσης (synchronous half duplex) που μπορεί να επικοινωνήσει με περιφερειακές συσκευές, όπως σειριακές EEPROMs ή ολοκληρωμένα κυκλώματα A/D (analog to digital) ή D/A (digital to analog).
Ο έλεγχος της USART γίνεται με την βοήθεια δύο καταχωρητών, του TXSTA και του RCSTA.

Γεννήτρια Ρυθμού μετάδοσης (Baud Rate Generator, BRG)

Η γεννήτρια ρυθμού μετάδοσης (BRG) υποστηρίζει και την σύγχρονη και την ασύγχρονη κατάσταση λειτουργίας της USART. Ουσιαστικά, πρόκειται για ένα ελεύθερης λειτουργίας (free running) μετρητή των 8 bit. Ο μετρητής αυτός ελέγχεται από τον καταχωρητή SPBRG (Baud Rate Generator Ragister). Συνεπώς, ο μετρητής αυτός λειτουργεί με το ρολόι του PIC και μετράει μέχρι να φτάσει την τιμή του καταχωρητή SPBRG. Στην ασύγχρονη κατάσταση λειτουργίας το bit BRGH (TXSTA) ελέγχει το ρυθμό μετάδοσης (baud rate) ενώ στην σύγχρονη κατάσταση, αγνοείται.

Καθένα από αυτά τα bit μπορεί να στέλνεται με βάση ένα από τα καθορισμένα ρολόγια που μπορεί να παράγει η γεννήτρα του ρυθμού μετάδοσης (Baud Rate Generator), όπως αναφέρθηκε προηγουμένως. Η USART στέλνει και λαμβάνει πρώτα το λιγότερο σημαντικό ψηφίο (LSB) της λέξης. Ο πομπός και ο δέκτης της USART

είναι ανεξάρτητα μεταξύ τους, ωστόσο χρησιμοποιούν την ίδια δομή και το ίδιο ρυθμό μετάδοσης. Το υλικό (hardware) της USART δεν υποστηρίζει έλεγχο ισοτιμίας (parity).

Πομπός ασύγχρονης κατάστασης λειτουργίας της USART (USART Asynchronous Transmitter)

Όπως παρατηρούμε, στην αρχιτεκτονική του πομπού της USART, ο πυρήνας του πομπού είναι ένας σειριακός καταχωρητής ολίσθησης (Transmit Shift Register, TSR). Ο καταχωρητής αυτός παίρνει τιμές από τον καταχωρητή TXREG (USART Transmit Data Register) ο οποίος με την σειρά του φορτώνεται με τιμή από τον χρήστη μέσω λογισμικού (software). Ο TSR δεν φορτώνεται παρά μόνο όταν αποσταλεί και το stop bit από την προηγούμενη φόρτωση. Από την στιγμή που αποστέλλεται και το stop bit, ο TSR παίρνει την καινούρια τιμή από τον TXREG, αν αυτή είναι διαθέσιμη. Τη στιγμή που ο TXREG δίνει την τιμή του στον TSR, θέτει και τη σημαία TXIF προκειμένου να γίνει διακοπή με βάση τα όσα αναφέρθησαν στην ενότητα των διακοπών. Αυτή η διακοπή μπορεί να ενεργοποιηθεί / απενεργοποιηθεί αν τεθεί (set) / καθαριστεί (clear) το αντίστοιχο bit ενεργοποίησης (enable bit) TXIF της συγκεκριμένης διακοπής. Το bit αυτό (TXIF) δεν μπορεί να γίνει μηδέν από το χρήστη αλλά μόνο από το υλικό (hardware). Αυτό γίνεται όταν ο TXREG φορτωθεί με καινούρια τιμή. Το bit TRMT (TXSTA) δείχνει την κατάσταση του καταχωρητή ολίσθησης (TSR). Το bit αυτό είναι μόνο αναγνώσιμο (readable) και γίνεται 1 όταν ο TSR έχει αποστείλει όλα του τα bit και είναι άδειος. Κάθε άλλη στιγμή το bit TRMT είναι 0. Το bit αυτό δεν μπορεί να προκαλέσει διακοπή και συνεπώς ο χρήστης θα πρέπει να το ελέγχει συνεχώς (polling) προκειμένου να διαπιστώσει πότε ο καταχωρητής TSR είναι άδειος.
Η ενεργοποίηση του πομπού της UASRT γίνεται θέτοντας 1 το bit TXEN (TXSTA). Η αποστολή δεν θα ξεκινήσει αν πρώτα δεν φορτωθεί με τιμή ο TXREG και η γεννήτρια του ρυθμού μετάδοσης δεν παράγει το ζητούμενο ρολόι. Αποστολή μπορεί να γίνει και αφού πάρει τιμή ο TXREG και ύστερα τεθεί 1 το bit TXEN.

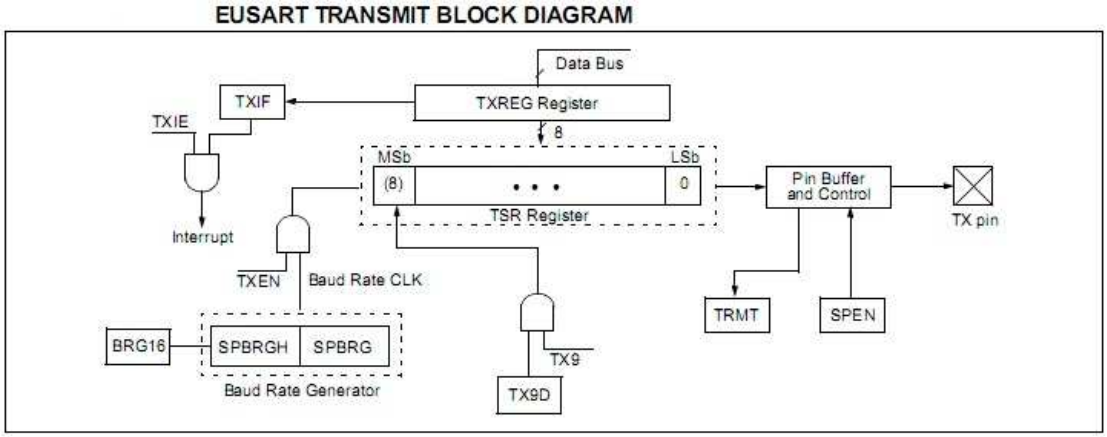
Τα βήματα που πρέπει να ακολουθήσει κανείς προκειμένου να πραγματοποιηθεί μια ασύγχρονη μετάδοση μέσω της USART είναι τα εξής :

- Αρχικοποίηση των ακροδεκτών σε εξόδου της σειριακής.
- Αρχικοποίηση του καταχωρητή SPBRG προκειμένου να παραχθεί ο κατάλληλος ρυθμός μετάδοσης.
- Αν θέλουμε υψηλής ταχύτητας (high speed) ρυθμό μετάδοσης πρέπει να τεθεί 1 και το bit BRGH.
- Ενεργοποίηση της ασύγχρονης κατάστασης λειτουργίας θέτοντας το bit SYNC .
- Αν είναι επιθυμητές οι διακοπές, τότε πρέπει να τεθούν τα bit TXIE για τις διακοπές κατά την αποστολή.
- Αν είναι επιθυμητή η αποστολή 9 bit δεδομένου και όχι 8, τότε πρέπει να τεθεί 1 το bit TX9 (στην παρούσα υλοποίηση θέτεται 0 αφού στέλνουμε 8 Bit).
- Ενεργοποίηση της αποστολής θέτοντας το bit TXEN, το οποίο θέτει και το bit

TXIF.

- Φόρτωση δεδομένου στον καταχωρητή TXREG και η αποστολή ξεκινάει.

Η αρχιτεκτονική του πομπού της USART φαίνεται παρακάτω:



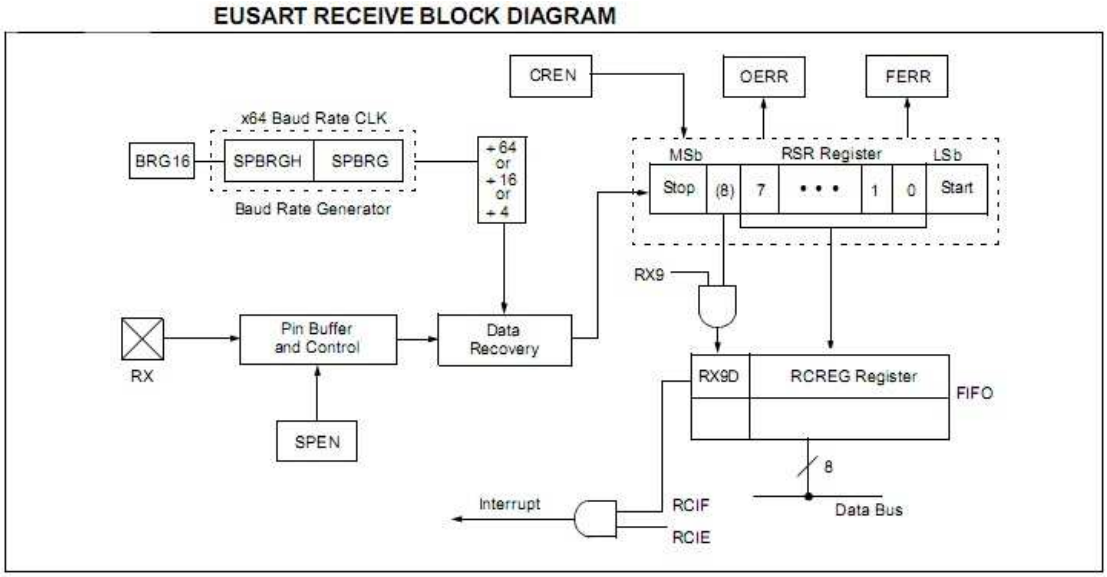
Εικόνα 15: Η αρχιτεκτονική του πομπού της USART

Δέκτης ασύγχρονης κατάστασης λειτουργίας της USART (USART Asynchronous Receiver)

Στην αρχιτεκτονική του πομπού της USART το δεδομένο λαμβάνεται στον ακροδέκτη RX/TX του PIC.
Εφόσον έχει επιλεγεί η ασύγχρονη κατάσταση λειτουργίας της USART μέσω του bit SYNC, η λήψη δεδομένων ενεργοποιείται θέτοντας το bit CREN. Ο πυρήνας της δέκτη είναι ο σειριακός καταχωρητής ολίσθησης λήψης (Receive Shift Register, RSR). Μόλις αναγνωριστεί και το stop bit στον ακροδέκτη RX/TX του PIC, το δεδομένο μεταφέρεται από τον RSR στον RCREG (USART Receive Data Register), αν αυτός είναι άδειος.
Όταν ολοκληρωθεί η μεταφορά τίθεται το bit RCIF. Το bit αυτό είναι μόνο αναγνώσιμο (readable) και μηδενίζεται μόνο μέσω υλικού (hardware) όταν ο καταχωρητής RCREG διαβαστεί και αδειάσει. Ο καταχωρητής αυτός μπορεί να είναι μια στοίβα 2 επιπέδων (two deep FIFO). Είναι δυνατόν 2 bytes να έχουνε ληφθεί και μεταφερθεί στην στοίβα και άλλο ένα byte να ολισθαίνει στον καταχωρητή RSR. Όταν κατά τη λήψη αναγνωριστεί το stop bit και η στοίβα είναι γεμάτη τότε το νέο δεδομένο χάνεται και τίθεται το bit OERR (Overrun Error Bit, OERR). Το bit αυτό πρέπει να μηδενιστεί από τον χρήστη μέσω λογισμικού (software) επαναθέτοντας (resetting) την λογική της λήψης (μηδενίζοντας και θέτοντας ξανά το bit CREN). Καθ 'όλη την διάρκεια που το OERR bit είναι 1, απαγορεύονται οι μεταφορές από τον RSR register στον RCREG. Σε περίπτωση που ενώ αναμένεται stop bit, αυτό δεν ληφθεί τότε τίθεται το bit FERR (Framing Error Bit. FERR). Το bit αυτό δεν προκαλεί διακοπή και συνεπώς ο χρήστης πρέπει να το ελέγχει συνεχώς (polling) για να διαπιστώσει αν έγινε κάποιο λάθος κατά την λήψη του δεδομένου.
Τα βήματα που πρέπει να ακολουθήσει κανείς προκειμένου να πραγματοποιηθεί μια ασύγχρονη λήψη μέσω της USART είναι τα εξής :

- Αρχικοποίηση των ακροδεκτών σε εισόδου της σειριακής.
- Αρχικοποίηση του καταχωρητή SPBRG προκειμένου να παραχθεί ο κατάλληλος ρυθμός λήψης.
- Αν θέλουμε υψηλής ταχύτητας (high speed) ρυθμό λήψης πρέπει να τεθεί 1 και το bit BRGH.
- Ενεργοποίηση της ασύγχρονης κατάστασης λειτουργίας θέτοντας το bit SYNC ίσο με 0.
- Αν είναι επιθυμητές οι διακοπές, τότε πρέπει να τεθούν τα bit RCIE στην τιμή 1.
- Αν είναι επιθυμητή η λήψη 9 bit δεδομένου και όχι 8, τότε πρέπει να τεθεί 1 το bit RX9.
- Ενεργοποίηση της αποστολής θέτοντας το bit CREN 1.
- Το bit RCIF τίθεται όταν ολοκληρωθεί η λήψη και η αντίστοιχη διακοπή θα γίνει αν και το bit RCIE είναι 1.
- Διάβασμα των 8 bit που ελήφθησαν από τον καταχωρητή RCREG.
- Αν έχει συμβεί κάποιο λάθος, πρέπει να καθαριστεί η αντίστοιχη σημαία λάθους μηδενίζοντας το bit CREN

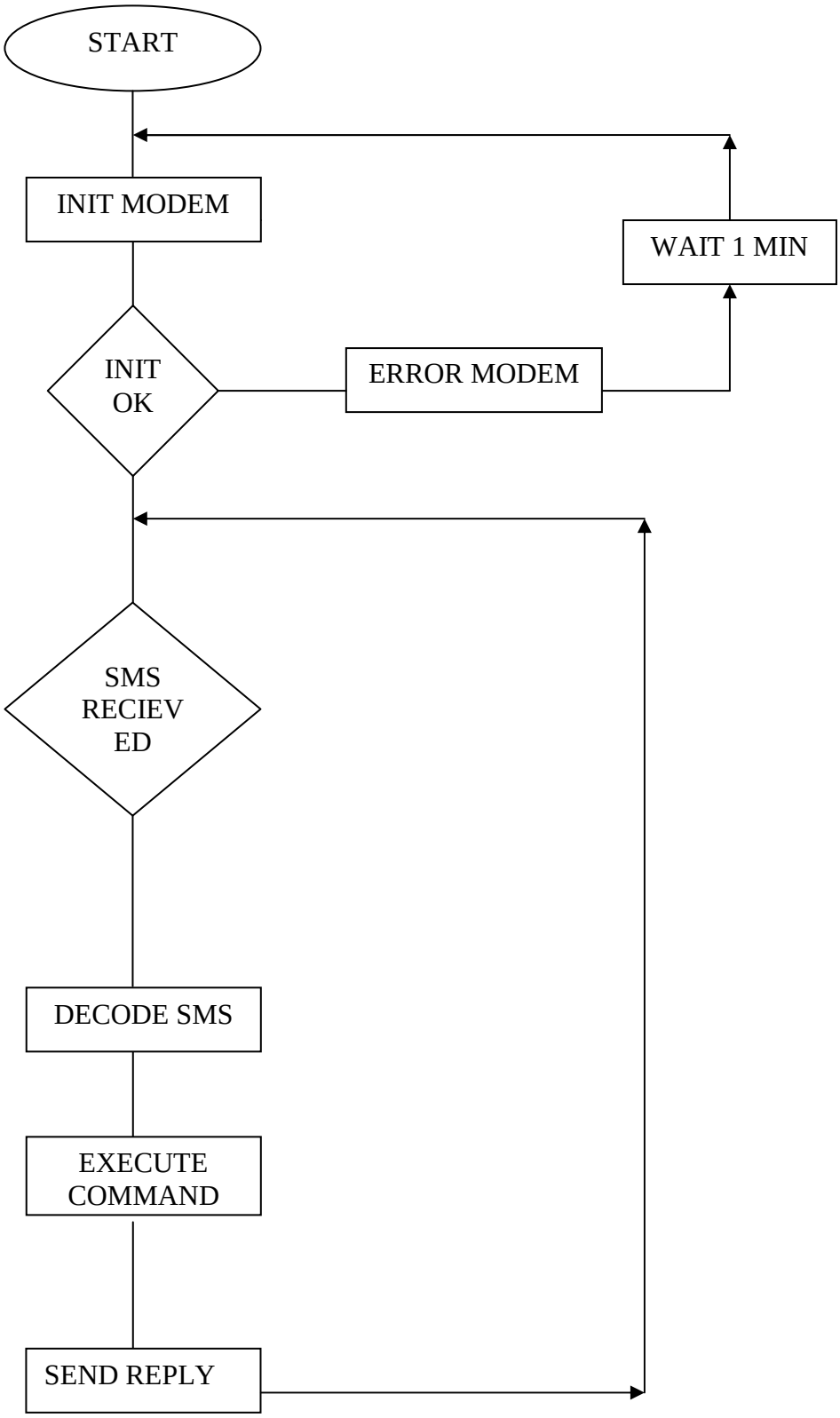
Η αρχιτεκτονική του δέκτη της USART φαίνεται παρακάτω:



Εικόνα 16: Η αρχιτεκτονική του δέκτη της USART

ΚΕΦΑΛΑΙΟ 4 – ΠΑΡΑΘΕΣΗ ΚΩΔΙΚΑ

Για τον ορθό σχεδιασμό και οργάνωση του κώδικα ακολουθήθηκε συγκεκριμένο διάγραμμα ροής. Με τη βοήθειά του προβλέφθηκαν κάποια πιθανά λάθη, ακόμη και ατέρμονες βρόγχοι. Το διάγραμμα ροής:



Όπως φαίνεται στο σχεδιάγραμμα η βασική ιδέα είναι να εκκινεί το σύστημα κάνοντας τις απαραίτητες αρχικοποιήσεις και ρυθμίσεις που χρειάζονται για την λειτουργία του.
Για παράδειγμα ρύθμιση του modem για την τροφοδότηση στη σειριακή των μηνυμάτων, η εισαγωγή του pin της κάρτας αν υπάρχει (στη παρούσα εφαρμογή δεν το έχουμε ενεργοποιήσει) η ρύθμιση χρήσης pdu δομής μηνυμάτων και άλλα.
Εάν αυτές δεν ολοκληρωθούν σωστά εμφανίζεται μήνυμα στην οθόνη που μας ενημερώνει ότι υπάρχει πρόβλημα με το modem. Σε αντίθετη περίπτωση το πρόγραμμα συνεχίζει να εκτελείται και βρίσκεται σε αναμονή για κάποιο νέο μήνυμα. Όταν αυτό έρθει αποκωδικοποιείται και ελέγχεται το περιεχόμενο για πιθανή εντολή. Σε περίπτωση που είναι κάποια από τις προκαθορισμένες εντολές εκτελεί την εντολή αυτή, στέλνει ένα απαντητικό μήνυμα επιβεβαίωσης και επανέρχεται στην κατάσταση αναμονής.
Αν η εντολή είναι άγνωστη αποστέλλεται μήνυμα που ενημερώνει για λάθος εντολή και μετά επανέρχεται πάλι σε κατάσταση αναμονής.
Ο κώδικας εκτός από το βασικό αρχείο έχει χωριστεί σε άλλα 4 αρχεία. Ένα που αφορά την οθόνη, ένα για τη σειριακή, ένα που εκτελεί την διαχωρισμό του αρχικού κειμένου από άλλες πληροφορίες των μηνυμάτων και τέλος το αρχείο με τις δηλώσεις των συναρτήσεων. Έτσι έχουμε τα lcd.c, serial.c, pdu decode.c, defs.h και φυσικά το main.c.

Αρχείο main.c

Το main.c αρχείο κώδικα περιλαμβάνει το κυρίως πρόγραμμα μαζί με μερικές ακόμα συναρτήσεις που αφορούν στις αρχικοποιήσεις του υλικού καθώς και η συνάρτηση για τα interrupts που ενεργοποιούνται κατά τη διάρκεια της εκτέλεσης του προγράμματος.
Πιο αναλυτικά το πρόγραμμα ξεκινά κάνοντας τις «κλήσεις» των συναρτήσεων που διαμορφώνουν τον μικροελεγκτή, αρχικοποιούν τις απαραίτητες μεταβλητές και γίνονται οι δηλώσεις των μεταβλητών που θα χρειαστούν.
Η εντολή _CONFIG είναι η απαραίτητη εντολή ρύθμισης του μικροελεγκτή. Με αυτήν στέλνουμε τα Bit που ενεργοποιούν δυνατότητες του μικροελεγκτή όπως το debug, τη χρήση του Watchdog Timer, το κλείδωμα του μικροελεγκτή ώστε να μην είναι δυνατή η αντιγραφή του κώδικα και άλλα.

Στη συνέχεια γίνονται οι δηλώσεις των μεταβλητών που θα χρησιμοποιηθούν καθώς και οι αρχικοποιήσεις σε όποιες χρειάζεται ή το «γέμισμα» σε κάποιους πίνακες.
Παρατηρούμε πως κάποιοι πίνακες αρχικοποιούνται με μεγάλα κομμάτια κωδικοποιημένου κειμένου. Αυτό συμβαίνει επειδή αν και έχει δημιουργηθεί ο κώδικας για την κωδικοποίηση κειμένου στη μορφή του pdu sms, δεν εκτελείται μιας και τα απαντητικά μηνύματα που στέλνονται είναι πάντα τα ίδια και τόσο μικρά που μόνο καθυστέρηση θα πρόσθεταν κατά την εκτέλεση του προγράμματος. Έτσι το μήνυμα έχει εκχωρείται σε δύο πίνακες. Ο πρώτος περιέχει την εντολή που πρέπει να σταλεί στο modem μαζί με το πρώτο μισό κομμάτι του κωδικοποιημένου μηνύματος, μέχρι το σημείο που πρέπει να υπάρχει το τηλέφωνο στο οποίο θα σταλεί το μήνυμα. Στο δεύτερο πίνακα υπάρχει το υπόλοιπο κομμάτι του μηνύματος, χωρίς βέβαια το τηλέφωνο του παραλήπτη, αφού είναι το μόνο που ίσως να αλλάζει κάθε φορά.
Μετά από αυτά εκτελείται το κεντρικό πρόγραμμα.

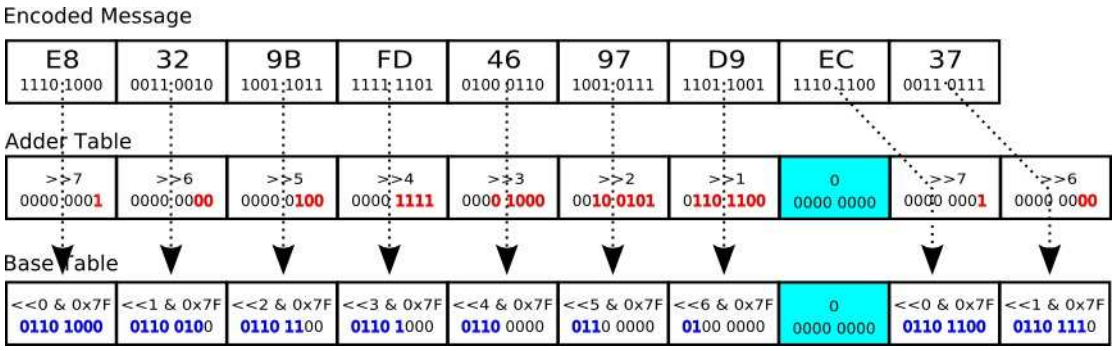
Ξεκινώντας το πρόγραμμα όπως είναι αναμενόμενο καλεί τις συναρτήσεις που ρυθμίζουν και αρχικοποιούν το υλικό μας.

Έτσι με την **InitUsart**() γίνονται οι ρυθμίσεις της σειριακής για τη σωστή μεταφορά των δεδομένων. Με την **InitPic**() ενεργοποιούμε τα interrupt του μικροελεγκτή. Ακολουθεί η **InitLCD**() με την οποία ρυθμίζεται η οθόνη και με τις εντολές LCDSendCmd(DISP_ON) και LCDSendCmd(CLR_DISP) την ενεργοποιούμε και σβήνουμε ότι μπορεί να υπάρχει. Με την **InitGsm**() αρχικοποιούμε και το modem και το θέτουμε σε αναμονή. Αποστέλλονται στη σειριακή κάποιες AT εντολές (οι οποίες επεξηγούνται σε προηγούμενο κεφάλαιο) για να λειτουργεί το modem κάτω από συγκεκριμένες και επιθυμητές συνθήκες. Αν σε αυτό το σημείο δεν εκτελεστεί σωστά το πρόγραμμα επιστρέφεται ένα μήνυμα λάθους. Τέλος η **InitTimer**() ρυθμίζει τον χρονοδιακόπτη κάνοντας μάλιστα χρήση της **writeTMR**(float timer) συνάρτησης η οποία καταχωρεί τη τιμή του χρόνου στον καταχωρητή.

Ακολούθως το πρόγραμμα μπαίνει σε έναν ατέρμων βρόγχο. Ελέγχεται συνέχεια αν στη σειριακή έχουν έρθει τα γράμματα C,M,T συνεχόμενα κάτι που υποδηλώνει ότι το modem έχει λάβει ένα νέο μήνυμα. Εάν κάτι τέτοιο γίνει, η μεταβλητή pdu_sms γίνεται true. Επίσης ελέγχει αν έρθει ο τελευταίος χαρακτήρας από το pdu sms δηλαδή ο χαρακτήρας επαναφοράς (carriage return η δεκαεξαδική του μορφή είναι 0x00D). Στη περίπτωση που έρθει στη σειριακή ο τελευταίος χαρακτήρας θα καλέσει την συνάρτηση που αποκωδικοποιεί τα sms με όρισμα το μήνυμα που μόλις έλαβε.

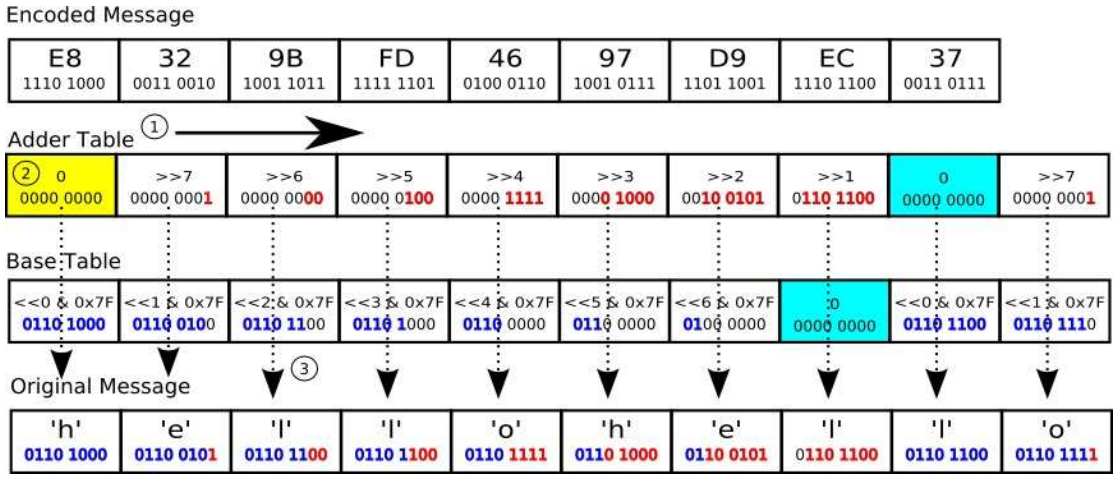
Αυτές είναι όλες οι εντολές που υπάρχουν στη κεντρική μας συνάρτηση.
Στη συνέχεια υπάρχει η συνάρτηση που είναι υπεύθυνη για τα interrupt η **myInterrupt**(void). Εκτελείται όταν λάβει κάτι η σειριακή και ελέγχει αν αυτός είναι το +, ο πρώτος χαρακτήρας που στέλνεται από το modem σε διάφορες περιπτώσεις μία εκ των οποίων και η λήψη νέου sms. Όταν αυτός ληφθεί ξεκινά η αποθήκευση των χαρακτήρων σε έναν πίνακα για να ελέγξουμε αν πρόκειται όντως για νέο sms. Αυτό όπως φαίνεται στον παρακάτω κώδικα είναι ένας έλεγχος που γίνεται στη κεντρική μας συνάρτηση. Αν όντως ήρθε νέο μήνυμα αποθηκεύονται στον πίνακα pdu_array όλοι οι χαρακτήρες μέχρι να ληφθεί και ο τελευταίος. Σε όλη αυτή τη διαδικασία έχουμε απενεργοποιήσει τα interrupt για να μην δημιουργηθεί πρόβλημα στη ροή του προγράμματος. Εδώ τελειώνει και η δουλειά της myInterrupt συνάρτησης. Ακολούθως όπως γράφηκε παραπάνω ο πίνακας με τους ληφθέντες χαρακτήρες γίνεται όρισμα για την συνάρτηση αποκωδικοποίησης **pdu_decode** η οποία χωρίζει τα δεδομένα του μηνύματος από όλες τις υπόλοιπες πληροφορίες για το μήνυμα και τις επιστρέφει στη main συνάρτηση.
Μετά την αποκωδικοποίηση του pdu sms λοιπόν ακολουθεί η αποκωδικοποίηση των δεδομένων του μηνύματος. Το αρχικό κείμενο του αποστολέα δηλαδή. Γίνεται με την συνάρτηση **data_decode** η οποία επίσης δέχεται σαν όρισμα έναν πίνακα που δημιούργησε η pdu_decode μόνο με τα χρήσιμα προς αποκωδικοποίηση δεδομένα, τον pdu_data. Όπως εξηγήθηκε και σε προηγούμενο κεφάλαιο πάνω από το δίκτυο GSM χρησιμοποιούμε 7-bits (septets) αντί 8-bits (octets) για την μετάδοση των χαρακτήρων. Έτσι το μοτίβο λογικής που ακολουθήθηκε για τη διαδικασία αποκωδικοποίησης ενός πίνακα δεδομένων κάνει χρήση δύο ακόμα πινάκων έστω τους Adder Table και Base Table. Και αυτό γιατί για κάθε οκτώ στήλες πρέπει να προσθέσουμε μία άδεια στήλη για να δημιουργήσουμε πάλι τα 8-bits (octets). Στον Adder Table κάνουμε ολίσθηση («shift») προς τα δεξιά τόσα bits όσα ο αύξον

αριθμός του κάθε χαρακτήρα (αν είναι μεγαλύτερος του 7 παίρνουμε το ακέραιο μέρος της διαίρεσης του με το 8). Αν ο χαρακτήρας αυτός είναι πολλαπλάσιο του οκτώ θέτουμε όλα τα bit μηδενικά. Στον Base Table κάνουμε ολίσθηση («shift») προς τα αριστερά bits ξεκινώντας από το 0 και αυξάνοντας όσο πάμε στους επόμενους χαρακτήρες (μέχρι τον 7ο) προσθέτοντας επίσης τον χαρακτήρα 0x7f. Ακολουθεί επεξηγηματικό σχεδιάγραμμα των δύο πινάκων που πρέπει να γίνουν.



Εικόνα 17: Κύλιση πινάκων κωδικοποιημένου μηνύματος

Αφού λοιπόν αυτοί δημιουργηθούν μετακινούμε όλα τα στοιχεία του Adder Table προς τα δεξιά και στην αρχή τοποθετούμε τον μηδενικό χαρακτήρα (0x00). Έπειτα προσθέτουμε τους δύο πίνακες εμφανίζοντας μας έτσι το αρχικό μήνυμα!



Εικόνα 18: Αποκωδικοποίηση μηνύματος

Για εξοικονόμηση μνήμης αυτή η διαδικασία έπρεπε να υλοποιηθεί στον κώδικα χωρίς τη δημιουργία των πινάκων κάθε φορά. Έτσι, τα adderByte και baseByte πλέον, διαμορφώνονταν εξαρχής και στο τέλος προσθέτονταν αποθηκεύοντας το τελικό αποτέλεσμα στον πίνακα data_decoded ο οποίος είχε τελικά το αρχικό μήνυμα. Ακολούθως κάθε γράμμα γίνεται capitalize δηλαδή κεφαλαία άσχετα αν ήταν ή όχι για καλύτερη ευελιξία του προγράμματος.

Έχοντας πλέον το αρχικό μήνυμα σε αναγνωρίσιμη μορφή διαχωρίζεται το τμήμα του κωδικού (password) από το υπόλοιπο τμήμα του κειμένου. Αν δεν είναι σωστός αυτός

ο κωδικός αποστέλλεται μήνυμα που ενημερώνει τον χρήστη ότι εισήγαγε λάθος κωδικό. Αν είναι σωστός εξετάζεται αν περιέχει κάποια από τις επιθυμητές εντολές. Οι επιλογές μας είναι τρείς. Να ανοίξει ή να κλείσει το relay του μικροελεγκτή ή να αγνοηθεί το μήνυμα ως άγνωστη εντολή. Ανάλογα λοιπόν αν η εντολή αφορά στο να ανοίξει το relay εκείνο ανοίγει, όπως και το led παραμένει αναμμένο συνεχόμενα και εμφανίζεται μήνυμα στην οθόνη του μικροελεγκτή ότι το relay άνοιξε. Αντίστοιχα αν η εντολή λέει να κλείσει εκείνο, καθώς και το led, παραμένουν κλειστά και εμφανίζεται στην οθόνη ότι το relay έκλεισε. Τέλος αν η εντολή δεν είναι κάποια από τις δύο παραπάνω αποστέλλεται ένα μήνυμα που ενημερώνει τον αποστολέα ότι η εντολή δεν αναγνωρίζεται.

Αν και δεν χρειάζεται στην παρούσα υλοποίηση του προγράμματος για λόγους πληρότητας αναπτύχθηκε και ο κώδικας για την κωδικοποίηση ενός μηνύματος. Αντίστοιχα με αυτόν της αποκωδικοποίησης για την ευκολότερη κατανόηση της λειτουργίας του κώδικα πρέπει να δημιουργήσουμε δύο ακόμα πίνακες εκτός του εκείνου του αρχικού κειμένου έστω τους Adder Table και Base Table. Αυτή τη φορά στον Adder Table πρέπει να κάνουμε ολίσθηση («shift») προς τα αριστερά bits ξεκινώντας από το 8 και φτάνοντας ως το 1. Ενώ στον Base Table «shift» προς τα δεξιά bits ξεκινώντας από το 0 μέχρι το 7.

Original Message

'h'	'e'	'l'	'l'	'o'	'h'	'e'	'l'	'l'	'o'
0110 1000	0110 0101	0110 1100	0110 1100	0110 1111	0110 1000	0110 0101	0110 1100	0110 1100	0110 1111

Adder Table

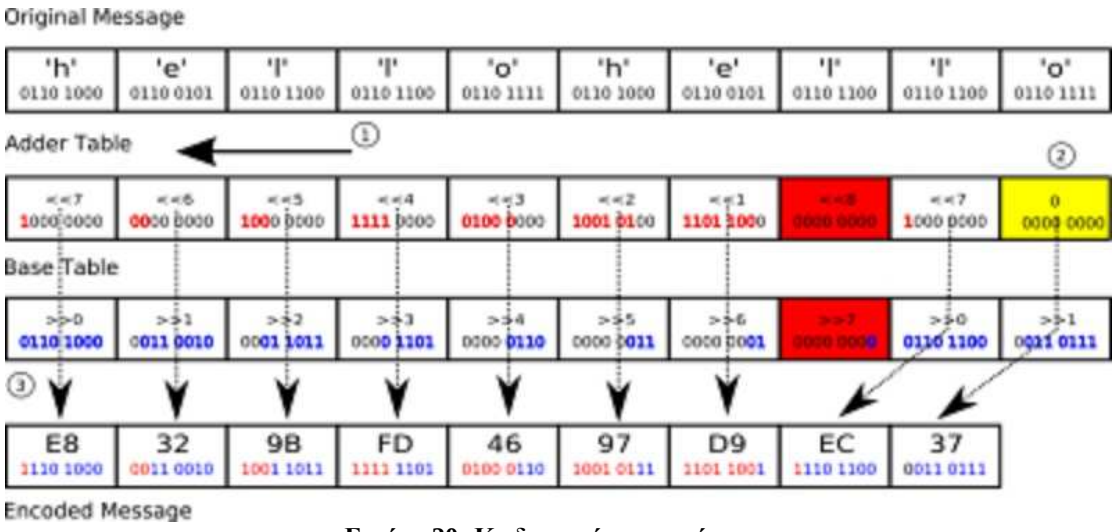
<<8	<<7	<<6	<<5	<<4	<<3	<<2	<<1	<<8	<<7
0000 0000	1000 0000	0000 0000	1000 0000	1111 0000	0100 0000	1001 0100	1101 1000	0000 0000	1000 0000

Base Table

>>0	>>1	>>2	>>3	>>4	>>5	>>6	>>7	>>0	>>1
0110 1000	0011 0010	0001 1011	0000 1101	0000 0110	0000 0011	0000 0001	0000 0000	0110 1100	0011 0111

Εικόνα 19: Κύλιση πινάκων για κωδικοποίηση μηνύματος

Έπειτα μετακινούμε τις στήλες του Adder Table προς τα αριστερά και προσθέτουμε στη τελευταία (κενή) στήλη τον μηδενικό χαρακτήρα (0x00). Πρέπει τέλος να προσθέσουμε τους δύο πίνακες εκτός από τις στήλες στις οποίες στον Adder Table έχουμε ολίσθηση κατά 8 bits ενώ στον Base Table κατά 7 bits.



Εικόνα 20: Κωδικοποίηση μηνύματος

Και εδώ για λόγους καλύτερης διαχείρισης μνήμης δεν δημιουργούνται οι δύο πίνακες. Αυτός είναι ένας τρόπος να δημιουργηθεί ένας πίνακας με κωδικοποιημένο το αρχικό μήνυμα, αν και στην συγκεκριμένη υλοποίηση αποστέλλονται ήδη κωδικοποιημένα μηνύματα για ταχύτερη ανταπόκριση της εφαρμογής.
Στον κώδικα υπάρχει η συνάρτηση **WaitResponse** με τη βοήθεια της οποίας δεν «κολλάει» η εφαρμογή. Χρησιμοποιείται όταν γίνεται χρήση εντολών που απαιτούν ίσως λίγο περισσότερο χρόνο για να εκτελεστούν ή είναι εύκολο να περιμένουν για κάποια απάντηση στη σειριακή θύρα από το modem. Δίνονται λοιπόν 2 ορίσματα, η αναμενόμενη απάντηση στη σειριακή και ο χρόνος για τον οποίο θα περιμένει αυτή την απάντηση ο μικροελεγκτής. Αν ικανοποιηθούν οι 2 συνθήκες η συνάρτηση επιστρέφει τον αριθμό 1. Σε αντίθετη περίπτωση τον αριθμό 0.

Κώδικας main.c

```
/*****************************************************
*****************************************************
*******
     Ανάπτυξη σε MPLAB IDE v8.50
     compiler: HI-TECH Universal ToolSuite C compiler
     Έγινε χρήση του PIC18F2520 με τον προγραμματιστή
ICD2.5 USB στη πλακέτα PIC-MT της Olimex LTD

     Το πρόγραμμα αποτελεί μέρος της διπλωματικής
εργασίας "Υλοποίηση συστήματος τηλεχειρισμού πάνω απο
δίκτυο GSM"
     που έγινε στο ΤΕΙ Κρήτης στο τμήμα Εφαρμοσμένης
Πληροφορικής και Πολυμέσων.
*****************************************************
*****************************************************
*****/

#include "defs.h"
```

```
__CONFIG(1, XT);
__CONFIG(2, PWRTDIS & BORV45 & WDTEN & WDTPS2K);
__CONFIG(3, PBDIGITAL & CCP2RB3);

/*
 * Enable Background Debug         = DEBUGEN/DIS
 * Enable Extended CPU             = XINSTEN/DIS
 * Enable Low Voltage Program      = LVPEN/DIS
 * Enable Stack Overflow Reset     = STVREN/DIS
 */
__CONFIG(4, DEBUGEN & XINSTDIS & LVPDIS & STVREN);



unsigned char ch;
unsigned char data_decoded[140];
unsigned char data_encoded[140];
char pdu_data[140];
char CRReceived=false;
char pdu_array[PINSIZE];
unsigned char sms_starts=false;
unsigned char plus_detected=false;
unsigned char pdu_sms=false;
unsigned char command[100];
const char
standar_msg[]={"AT+CMGS=25\n\r06910379010000001000C91"};
const char
relay_closed[]={"00000DF2323B9C078DD9EF79991C02"};
const char
relay_open[]={"00000DF2323B9C07A5E7A037BCEC06"};
const char
wrong_command[]={"00000D77F9DB7D068DDFED76D84D06"};
char sender_number[13];
const char password[4]={"1234"};
const char
wrong_pass[]={"00000D69B73DCC4E9341F0F07C1E02"};




void main( void ) {
     char tmp;
     const char *pos;
     char k,j=0;

     InitUsart();
     InitPic();
     InitLCD();
     InitTimer();
```

```
     LCDSendCmd(DISP_ON);
     LCDSendCmd(CLR_DISP);
     InitGsm();
     LCDSendCmd(CLR_DISP);
     LCDSendStr("Running");
     RELAY= RELAY_OFF;

     while(1) {


          CLRWDT();
          LED=!LED;
          //DelayMs(1000);

          if ((plus_detected==true) &&
(pdu_array[1]=='C') && (pdu_array[2]=='M') &&
(pdu_array[3]=='T')){
               pdu_sms=true;
          }

          if (CRReceived==true){
               pdu_decode(pdu_array);
               CRReceived=false;
          }

     }
}


 interrupt myInterrupt(void)       // interrupt service
routine at address 4
{
     static char i=0;
     char rec;
     GIE=0;
     if (RCIF)  //serial port receive interrupt?
     {


          if (RCREG=='+'){
               plus_detected=true;
               i=0;
          }
          if   (plus_detected==true)
               pdu_array[i]=RCREG;

          if ((pdu_sms==true)&&(RCREG==0x0A)){
               sms_starts=true;
               plus_detected=false;
               i=-1;
```

```
          }
          if ((sms_starts==true)&&(RCREG!=0x0A)){
               pdu_array[i]=RCREG;
               pdu_sms=false;
               if (RCREG==0x0D){
                    sms_starts=false;
                    CRReceived=true;
               }
          }
          i++;
          RCIF = 0;           // reset serial port
receive interrupt flag


     }


     GIE=1;
}


void InitPic (){

     GIE = 1;       // Set Global Interrupt Enable
     PEIE = 1;      // Set Peripheral Interrupt Enable
     TRISB4=0;
 }


void data_decode (char pdu_data[]){

     unsigned char i,j,k=0;
     unsigned char adderByte;
     unsigned char baseByte;
     char adderskip=0;
     char baseskip=0;
     char pass=0;


     while (k<=(strlen(pdu_data)+(strlen(pdu_data)/8))) {

          if (((k%(7+(baseskip*8)))!=0)|| k==0)
               baseByte=(pdu_data[k-
baseskip]<<(k%8))&0x7f;
     else{
               baseByte=0;
               baseskip++;
          }

          if   ((k%8!=0) || k==0)
```

```
                adderByte=(pdu_data[((k-adderskip)-
1)]>>(7-((k-1)%8)));
          else{
                adderByte=0x0;
                adderskip++;
                }


          if(k==0)
                adderByte=0;

          data_decoded[k]= (adderByte+ baseByte);
          data_decoded[k]= toupper(data_decoded[k]);
          k++;


     }
     data_decoded[k]='\0';

     for (i=0; i<sizeof(password); i++){
          if (data_decoded[i]==password[i])
                pass++;
     }

          //command isolation
     if (pass==sizeof(password)){
          for ((i=(sizeof(password))+1);
i<=sizeof(data_decoded); i++){  //+1 gia to keno sto sms
                command[j]=data_decoded[i];
                j++;
          }
          command[j]='\0';


          if (strcmp(command,"RELAY ON")==0){
                LED=LED_ON;
                RELAY=RELAY_ON;
                WriteString(standar_msg);
                WriteString(sender_number);
                WriteString(relay_open);
                WriteByte(0x1A);
                WriteByte(0x08);
                LCDSendCmd(CLR_DISP);
                LCDSendCmd(CUR_HOME);
                LCDSendStr("relay is open");
          }
          else if (strcmp(command,"RELAY OFF")==0){
                LED=LED_OFF;
                RELAY=RELAY_OFF;
                WriteString(standar_msg);
                WriteString(sender_number);
```

```
                WriteString(relay_closed);
                WriteByte(0x1A);
                WriteByte(0x08);
                LCDSendCmd(CLR_DISP);
                LCDSendCmd(CUR_HOME);
                LCDSendStr("relay closed!");
           }

           else{
                WriteString(standar_msg);
                WriteString(sender_number);
                WriteString(wrong_command);
                WriteByte(0x1A);
                WriteByte(0x08);

           }
      }
      else{
           WriteString(standar_msg);
           WriteString(sender_number);
           WriteString(wrong_pass);
           WriteByte(0x1A);
           WriteByte(0x08);

           }


      for (i=0; i<k+3; i++)
           data_decoded[i]='\0';
      for (i=0; i<PINSIZE; i++)
           pdu_array[i]='\0';
      for (i=0; i<PINSIZE; i++)
           pdu_data[i]='\0';
      for (i=0; i<k+3; i++)
           command[i]='\0';

}


void data_encode (char pdu_array[]){

// help:
http://embeddedfreak.wordpress.com/2008/10/09/encoding-
gsm-sms-septets/

unsigned char k=0;
unsigned char adderByte;
unsigned char baseByte;
char shift=0;
char jump=0;
```

```
     while (k<(strlen(pdu_array))){

          if (k==((strlen(pdu_array)-1)))
               adderByte=0;

          if (((k%7)!=0)|| k==0){
               adderByte=pdu_array[k+1]<<(7-((k%7)-
shift));
               baseByte=pdu_array[k]>>((k%7)-shift);
               data_encoded[k-shift]=adderByte+baseByte;
          }
          else{
               shift++;
               data_encoded[k]=0;
          }


          k++;

     }


}



void InitTimer(){
     T08BIT=0;
     T0CS=0;
     PSA=0;
     T0PS2=1;   //prescaler = 128
     T0PS1=1;
     T0PS0=0;


}

void InitGsm(){

     char x=0;
     char modem_ready=1;

     LCDSendCmd(CLR_DISP);

     printf("ate0:");
     AtCommand("ate0");
     x=WaitResponse("OK",1000);
     printf("%d",x);
     modem_ready=modem_ready && x;

     printf(" cnmi:");
```

```
     AtCommand("at+cnmi=1,2");
     x=WaitResponse("OK",7000);
     printf("%d",x);
     modem_ready=modem_ready && x;

     LCDSendCmd(CUR_HOME);
     LCDSendCmd(CUR_DOWN);


     printf("cmgf:");
     AtCommand("at+cmgf=0");
     x=WaitResponse("OK",3000);
     printf("%d",x);
     modem_ready=modem_ready && x;


     if (modem_ready)
          printf(" MODEM OK");
     else{
          printf(" MODEM ERROR");
          while(1);
     }

}

int WaitResponse(const char *respond, int time){

char resp[20];
char j;
char resp_detected=0;
GIE=0;
//ClocksToCount=time/0
writeTMR(65536-(int)(time/0.128));
TMR0ON=1;
j=0;
resp[0]=0;
while (!TMR0IF){

     if (RCIF){
          RCIF = 0;
          resp[j]=RCREG;
          j++;

     }
     resp[j]='\0';
     if (strstr(resp,respond)!=NULL){
          resp_detected=true;
          break;
     }
     else
          resp_detected=false;
```

```
          CLRWDT();
     }


     TMR0ON=0;
     TMR0IF=0;
//   resp[j]='\0';

     GIE=1;
     return resp_detected;
}


void writeTMR(unsigned int startValue){
     TMR0H= (unsigned char) (startValue >>8);
     TMR0L= (unsigned char) (startValue & 0xff);

}
```

Αρχείο Defs.h

Στο συγκεκριμένο αρχείο γίνονται όλοι οι ορισμοί σταθερών τιμών, δηλώνουμε ποια αρχεία βιβλιοθηκών να εισαχθούν, όπως επίσης δηλώνονται και όλες οι συναρτήσεις που χρησιμοποιούνται στο πρόγραμμα.

Κώδικας Defs.h

```
#include  <htc.h>
#include  <stdio.h>
#include  <string.h>
#include  <ctype.h>



#define        E              RA5
#define        RW             RA3
#define        RS             RA2

#define        CLR_DISP       0x01
#define        DISP_ON        0x0C
#define        DISP_OFF       0x08
#define        CUR_HOME       0x02
#define        CUR_OFF        0x0C
#define        CUR_ON_UNDER   0x0E
#define        CUR_ON_BLINK   0x0F
#define        CUR_LEFT       0x10
```

```
#define         CUR_RIGHT       0x14
#define         CUR_UP          0x80
#define         CUR_DOWN        0xC0
#define         ENTER           0xC0
#define         DD_RAM_ADDR     0x80
#define         DD_RAM_ADDR2    0xC0


#define         DELAY_1000us              498/4
     //498*2+3 = 999us

#define LED RB4
#define PINSIZE 200
#define LED_ON 0
#define LED_OFF 1

#define RELAY RA1   //ME 1 ANOIGEI

#define TRUE 1
#define true 1
#define FALSE (!TRUE)
#define false (!true)

#define RELAY_ON 1
#define RELAY_OFF 0

void Delay(unsigned int);
void E_Pulse(void);
void InitLCD(void);
void LCDSendChar(unsigned char c);
void putch(char i);
LCDSendCmd(unsigned char c);
void LCDSendStr(const char* str);
void InitUsart(void);
void  WriteByte(unsigned char byte);
unsigned char ReadByte(void);
void DelayMs(unsigned char);
void ReadString(char *pdu_array);
void InitPic ();
int pdu_get_timestamp(const char *buffer);
int pdu_decode(const char *buffer);
void data_decode (char pdu_array[]);
void data_encode (char pdu_array[]);
void InitGsm();
void  WriteString(const char* str);
int WaitResponse(const char *respond, int time);
void writeTMR(unsigned int timer);
void InitTimer();
void AtCommand (const char *str);
```

Αρχείο Lcd.c:

Ξεκινώντας με τις συναρτήσεις αρχικοποίησης στο αρχείο για την οθόνη έχουμε την **InitLCD** η οποία αρχικοποιεί την οθόνη του μικροελεγκτή.
Επίσης η **LCDSendChar** στέλνει έναν χαρακτήρα στην οθόνη ενώ με τη βοήθεια της, η **LCDSendStr** στέλνει μια ολόκληρη συμβολοσειρά. Υπάρχει ακόμα η **LCDSendCmd** η οποία στέλνει αν χρειαστεί τις απαραίτητες εντολές ελέγχου για την οθόνη. Τέλος με τη χρήση της **putch** παραπέμπουμε την printf να εμφανίζει τα ορίσματά της στην οθόνη του μικροελεγκτή και όχι στη σειριακή.

Κώδικας Lcd.c:

```
#include "defs.h"

unsigned  int dly;
unsigned  int i;
unsigned  char data;


//delay time = (cntr*2)+3 (in us for 20MHz)
//FULL OPTIMIZATION of C compilator(10 level, enable
post-pass optimisation)
void Delay(unsigned int cntr) {

     for (dly=0; dly!=cntr; dly++) {
          asm("nop");
          asm("nop");
          asm("nop");
     }
}

//delay time is ~ 1ms
void DelayMs (unsigned char e)
{
     while(--e) {
          Delay(DELAY_1000us);
      }

}

void E_Pulse(void) {
     E=1;
     asm("nop");
     asm("nop");
     asm("nop");
     E=0;
}
```

```
// Init LCD  after reset
void InitLCD(void) {
     ADCON1  = 0x06;              // Port as Digital IO
     TRISA2    = 0;           // RS pin as output

     TRISA3    = 0;           // RW pin as output

     TRISA5    = 0;           // E pin as output
     TRISC     = 0xF0;             // D4-D7 as output
     PCFG0     =1;
     PCFG1     =1;
     PCFG2     =1;
     PCFG3     =1;
     RS=0;
     RW=0;


     DelayMs(110);
     PORTC=0b00000011;
     E_Pulse();
     DelayMs(10);
     PORTC=0b00000011;
     E_Pulse();
     DelayMs(10);
     PORTC=0b00000011;
     E_Pulse();
     DelayMs(10);
     PORTC=0b00000010;
     E_Pulse();
}

// Send char to LCD
void LCDSendChar(unsigned char c) {

     DelayMs(2);
     //get upper nibble
     data = c & 0b11110000;
     //set D4-D7
     data = data >> 4;
     //send data to LCD
     PORTC = data;
     //set LCD to write
     RW=0;
     //set LCD to data mode
     RS=1;
     //toggle E for LCD
     E_Pulse();
     // get lower nibble
     data = c & 0b00001111;
     //send data to LCD
```

```
     PORTC = data;
     //set LCD to write
     RW=0;
     //set LCD to data mode
     RS=1;
     //toggle E for LCD
     E_Pulse();
}


void putch(char i){

     LCDSendChar(i);
     //WriteByte(i);
}

// Send command to LCD
LCDSendCmd(unsigned char c) {

     DelayMs(2);
     //get upper nibble
     data = c & 0b11110000;
     //set D4-D7
     data = data >> 4;
     //send data to LCD
     PORTC = data;
     //set LCD to write
     RW=0;
     //set LCD to data mode
     RS=0;
     //toggle E for LCD
     E_Pulse();
     // get lower nibble
     data = c & 0b00001111;
     PORTC = data;
     //set LCD to write
     RW=0;
     //set LCD to data mode
     RS=0;
     //toggle E for LCD
     E_Pulse();
}

void LCDSendStr(const char* str) {

     i=0;
     while(str[i]) {
          LCDSendChar(str[i]);
          i++;
     }
}
```

Αρχείο Serial.c

Στο αρχείο της σειριακής περιλαμβάνονται η **InitUsart** με την οποία αρχικοποιείται η σειριακή. Ακόμα υπάρχει η **ReadByte** με την οποία «διαβάζουμε» αν έχει έρθει κάτι στη θύρα της σειριακής. Βέβαια υπάρχει η **WriteByte** η οποία στέλνει έναν χαρακτήρα στη σειριακή ενώ χάριν σε αυτή με τη **WriteString** στέλνει μια ολόκληρη συμβολοσειρά.

Κώδικας Serial.c

```
#include "defs.h"


//__CONFIG(WDTDIS & LVPDIS & BORDIS & HS & PWRTEN) ;

#define LED RB4

char serial_string[100];
char j;

void InitUsart(void) {

     /*// TX Pin - output
     TRISC6 = 0;

     // RX Pin - input
     TRISC7 = 1;

     // RX Setting, 8bit, enable receive,
     RCSTA = 0x90;

     // TX Setting, 8bit, Asinchronius mode, High speed
     TXSTA = 0x24;

     // Set Baudrade - 9600 (from datasheet baudrade
table)
     SPBRG = 25;

*/

     PORTA = 0;
     PORTB = 0;
     PORTC = 0;
```

```
     BRGH = 1;      // high speed serial port mode
     SPBRG=25;      // Set 9600 baud for 4 MHz oscillator
     SYNC = 0;      // Clear SYNC bit ; Set_Async_Mode;
     SPEN = 1;      // Set serial port enable
     TX9 = 0;            // 8-bit transmissions
     TXEN = 1;      // Enable transmission
     RCIE = 1;      // Rx interrupts are desired
     RX9 = 0;            // 8-bit receptions
     CREN = 1;      // Enable reception
}


void  WriteByte(unsigned char byte) {

     // wait until register is empty
     while(!TXIF)
          CLRWDT();
     // transmite byte
     TXREG = byte;
}

unsigned char ReadByte(void) {

     // wait to receive character
     while(!RCIF);
     LED=!LED;
     // return received character
     return RCREG;
}

 void WriteString(const char *str) {
     const char *p;
     p=str;
     while(*p)
          WriteByte(*p++);

}

void AtCommand (const char *str){
     const char *p;
     p=str;
     while(*p)
          WriteByte(*p++);
     WriteByte('\n');
     WriteByte('\r');

}
```

Αρχείο Pdu decode.c

Το τρίτο αρχείο (pdu decode.c) αφορά στην αποκωδικοποίηση του εισερχόμενου μηνύματος. Ξεχωρίζει τα πεδία από τα οποία αποτελείται το SMS. Διαχωρίζει δηλαδή το νούμερο του αποστολέα, το κύριο σώμα του μηνύματος, την ώρα που στάλθηκε καθώς και άλλες λεπτομέρειες που περιγράφονται στη μορφή του pdu sms (σε προηγούμενο κεφάλαιο).
Δέχεται σαν όρισμα έναν πίνακα χαρακτήρων. Χρησιμοποιεί ακόμα τις συναρτήσεις **hexdigit2number** και **hex2number**. Με την πρώτη να εκφράζει έναν χαρακτήρα σε αριθμό και τη δεύτερη χρησιμοποιώντας την πρώτη σε βρόγχο επανάληψης να εκφράζει συμβολοσειρές σε αριθμούς. Ουσιαστικά η hexdigit2number παίρνει την hex τιμή ενός χαρακτήρα ή αριθμού (ανάμεσα στο εύρος τιμών A-F και 0-9) αφαιρεί την hex τιμή του πρώτου χαρακτήρα από αυτό το εύρος (A ή 0 αντίστοιχα) και έτσι εκφράζεται σαν καθαρός αριθμός. Για παράδειγμα η hex τιμή του 0 είναι 30 και του 4 34, οπότε όταν ο κώδικας συναντήσει το 34 αφαιρεί το 30 και επιστρέφει 4.

Κώδικας Pdu decode.c

```
//#include <stdlib.h>
//#include <stdio.h>
//#include <string.h>
//
#include "defs.h"
extern char pdu_data[160];
extern char sender_number[13];
/**
  * Converts hex digit to number, returns -1 on failure.
  */
int hexdigit2number(const char digit)
{

          if (digit >= '0' && digit <= '9') {
                return digit - '0';
        }
        if (digit >= 'a' && digit <= 'f') {
               return 0xa + digit - 'a';
        }
        if (digit >= 'A' && digit <= 'F') {
               return 0xa + digit - 'A';
        }
       // printf("[hexdigit2number] %c is not a
digit!\n", digit);
        return -1;


}
```

```
/**
 * Converts hex string of arbitrary length to number,
returns -1 on
 * failure.
 */

int hex2number(const char *buffer, const size_t len)
{
        int result = 0, tmp;
        size_t pos;

        for (pos = 0; pos < len; pos++) {
                result = result << 4;

                tmp = hexdigit2number(buffer[pos]);
                if (tmp < 0)
                   return tmp;
                result += tmp;
        }
        return result;
}

/**
  * Reads single number encoded in PDU.
 */
int pdu_get_number(const char *buffer, const int
semioctet)
 {
         int length;
        int type;
        char *out;
        int i;

       length = hex2number(buffer, 2);
        if (semioctet) {
               if (length % 2)
                       length++;
                length = length / 2 + 1;
       }
        //printf("Number length = %d\n", length);

       if (length == 0)
               return 2;

        type = hex2number(buffer + 2, 2);
       //printf("Number type = %d\n", type);

     /*  out = (char *)malloc((2 * length) + 1);
       if (out == NULL)
               return -1;
```

```
        memset(out, 0, 2 * length);
*/
        for (i = 0; i < (length - 1) * 2; i++) {
                 if (!isxdigit((int)buffer[4 + i])) {
                        //printf("Non hex digit in PDU
(%s)!\n", buffer + 4 + i);
                        //free(out);
                        return -1;
                }

                if (i % 2 == 0) {
                        out[i + 1] = buffer[4 + i];
                         if (out[i + 1] == 'F') {
                                out[i + 1] = '\0';
                       }
                 } else {
                         out[i - 1] = buffer[4 + i];
                }
                sender_number[i]=buffer[4 + i];
         }
           sender_number[i+1]='\0';
         //printf("Number = %s\n", out);
        // free(out);

        return (length * 2) + 2;
}

/**
 * Parses timestamp from PDU.
 */
int pdu_get_timestamp(const char *buffer)
{
         int i;

         for (i = 0; i < 14; i++) {
                 if (!isxdigit((int)buffer[i]))
                        return -1;
        }

         //printf("Date: %d-%d-%d %d:%d:%d TZ=%d\n");
           //    hex2number(buffer + 0, 2),
               // hex2number(buffer + 2, 2),
hex2number(buffer + 4, 2), hex2number(buffer + 6, 2),
hex2number(buffer + 8, 2), hex2number(buffer + 10, 2),
hex2number(buffer + 12, 2)
            // );

         return 14;
}

 /**
```

```
* Decodes textual PDU.
*/
int pdu_decode(const char *buffer)

{
        /* Values */
         int type;
         int mr;
         /* Status variables */
         int pos = 0, ret;
         /* Message content */
         int submit = 0, deliver = 0, report = 0;
         int vpf = 0;
        int rp = 0;
         int udh = 0;
         int pid = 0;
         int dcs = 0;
        int vp = 0;
         int udhl = 0;
         int ud;
         int i;

        /* SMSC number */
        ret = pdu_get_number(buffer + pos, 0);
         if (ret < 0)
                 return ret;
         pos += ret;

         /* Message type */
         type = hex2number(buffer + pos, 2);
        if (type < 0)
                 return type;
         pos += 2;
         //printf("Message type: %02x - ", type);

        switch (type & 0x3) {
                 case 0:
                         //printf("Deliver");
                         deliver = 1;
                         break;
                 case 1:
                         //printf("Submit");
                        submit = 1;
                         break;
                case 2:
                         //printf("Status report\n");
                         report = 1;
                        break;
                 case 3:
                         //printf("Reserved\n");
                         return -1;
```

```
        }
        if (submit || deliver) {
                if (type & (1 << 7)) {
                        rp = 1;
                }
                if (type & (1 << 6)) {
                       udh = 1;
                }
         }
         if (rp) {
                //printf(", Reply path set");
         }
         if (udh) {
                //printf(", UDH included");
         }
         if (submit) {
                switch (type & (0x3 << 3)) {
                        case 0:
                                //printf(", No VP");
                                break;
                        case 1:
                                //printf(", Reserved
VP!\n");
                                return -1;
                        case 2:
                               //printf(", Relative
VP");
                                vpf = 2;
                                break;
                        case 3:
                                //printf(", Absolute
VP");
                               vpf = 3;
                               break;
                }
         }
         //printf("\n");

         /* Message reference (for submit) */
         if (submit || report) {
                mr = hex2number(buffer + pos, 2);
                if (mr < 0)
                        return mr;
                pos += 2;
               //printf("MR = 0x%02X\n", mr);
         }

         /* Address (sender for deliver, receiver for
submit, recipient
          * for report) */
        ret = pdu_get_number(buffer + pos, 1);
```

```
         if (ret < 0)
                 return ret;
        pos += ret;

         if (report) {
                 /* Timestamp */
                 ret = pdu_get_timestamp(buffer + pos);
                 if (ret < 0)
                         return ret;
                 pos += ret;

                 /* SMSC timestamp */
                 ret = pdu_get_timestamp(buffer + pos);
                 if (ret < 0)
                         return ret;
                 pos += ret;
        }
     if (submit || deliver) {
                 /* PID */
                 pid = hex2number(buffer + pos, 2);
                if (pid < 0)
                         return pid;
                pos += 2;
                 //printf("PID = 0x%02X\n", pid);

                /* DCS */
                 dcs = hex2number(buffer + pos, 2);
                 if (dcs < 0)
                         return dcs;
                 pos += 2;
                 //printf("DCS = 0x%02X\n", dcs);
       }
         if (submit) {
                 /* Validity */
                 if (vpf == 2 || vpf == 0) {
                        vp = hex2number(buffer + pos, 2);
                        if (vp < 0)
                                return vp;
                         pos += 2;
                        //printf("VP = 0x%02X\n", vp);
                } else if (vpf == 3) {
                       ret = pdu_get_timestamp(buffer +
pos);
                         if (ret < 0)
                                 return ret;
                         pos += ret;
                 }
         }
         if (deliver) {
                 /* SMSC timestamp */
                 ret = pdu_get_timestamp(buffer + pos);
```

```
                if (ret < 0)
                        return ret;
                pos += ret;
        }
        if (submit || deliver) {
                /* UD */
               udhl = hex2number(buffer + pos, 2);
                if (udhl < 0)
                        return udhl;
                pos += 2;
                //printf("UDL = 0x%02X\n", udhl);
                /* GSM 03.40 section 9.2.3.10 (TP-Data-
Coding-Scheme) and GSM 03.38 section 4 */
                if ((((dcs & 0xC0) == 0) && ((dcs == 0)
|| ((dcs & 0x2C) == 0x00) || ((dcs & 0x2C) == 0x20))) ||
                    ((((dcs & 0xF0) == 0xC0) || ((dcs &
0xF0) == 0xD0)) && ((dcs & 4) != 4)) || (((dcs & 0xF0) ==
0xF0) && ((dcs & 8) != 8) && ((dcs & 4) == 0))) {
                        if ((udhl * 7) % 8 != 0) {
                                udhl = (udhl * 7) / 8;
                                udhl++;
                        } else {
                                udhl = (udhl * 7) / 8;
                        }
                        //printf("UDL[adjusted] =
0x%02X\n", udhl);
                }
                //printf("data = ");

                for (i = 0; i < udhl; i++) {
                        ud = hex2number(buffer + pos,
2);

                        if (ud < 0) {
                                //printf("\nData too
short!\n");
                                return ud;
                        }
                        pos += 2;
                        //printf("%02X", ud);
                              pdu_data[i]=ud;
                }


                //printf("\n");
                   data_decode(pdu_data);
         }

         if (buffer[pos] != '\r' && buffer[pos] != '\n'
&& buffer[pos] != '\0') {
```

```
                //printf("Did not reach end: %s\n",
buffer + pos);
                return 1;
        }

         return 0;
}
```

ΠΡΟΤΑΣΕΙΣ

Όσο θα προχωράει η τεχνολογία αλλά και ο ρυθμός ζωής υλοποιήσεις όπως της παρούσας πτυχιακής θα είναι βέβαιες σε αρκετές εφαρμογές.
Μπορούν βέβαια να αναπτυχθούν περεταίρω για να καλύπτουν άλλες, πιο εξειδικευμένες ανάγκες. Για την υλοποίηση της εφαρμογής λήφθηκε υπ' όψιν η λειτουργικότητα και η απλότητα της εφαρμογής καθώς και η φορητότητα της υπηρεσίας πάντα σε συνάρτηση με τον υπάρχον εξοπλισμό.
Θα μπορούσε λοιπόν να χρησιμοποιηθεί πιο σύνθετος εξοπλισμός για να έχουμε στη διάθεση μας περισσότερες δυνατότητες.
Θα μπορούσε για παράδειγμα να αναπτυχθεί ο κώδικας για την διαχείριση περισσότερων της μίας συσκευής. Έτσι με ένα μήνυμα θα μπορούσαμε να επιλέξουμε ποια ή ποιες συσκευές να ανοίξουμε ή να κλείσουμε.
Ακόμα για λόγους προσαρμοστικότητας νέων χρηστών δεν αξιοποιήθηκε η δυνατότητα να δέχεται το σύστημα τις εντολές μόνο από συγκεκριμένους αριθμούς. Έτσι δεν θα έχουν πρόσβαση στο σύστημα οι μη εγκεκριμένοι αριθμοί.
Επίσης θα μπορούσε να επιστρέφεται και η κατάσταση της συσκευής, δηλαδή αν είναι ενεργοποιημένη ή όχι. Αυτό προϋποθέτει να ενεργοποιείται ή να απενεργοποιείται η συσκευή μόνο από την διάταξη ή να υποστηρίζει την αποστολή της κατάστασης της.
Θα μπορούσε ακόμα να χρονοπρογραμματίζει πότε θα ενεργοποιηθεί ή απενεργοποιηθεί η συσκευή. Για παράδειγμα να αποστέλλω στη συσκευή να ενεργοποιηθεί για μισή ώρα, ή να κλείσει σε μία ώρα.
Τέλος για μια πιο ολοκληρωμένη εφαρμογή θα έπρεπε να αναπτυχθεί γραφικό περιβάλλον στο οποίο ο χρήστης θα μπορούσε να κάνει τις παραμετροποιήσεις που θα ήθελε (όπως και τις επιπλέον ρυθμίσεις που προαναφέρθηκαν). Όπως να δηλώνει τις εντολές για την ενεργοποίηση ή/και απενεργοποίηση της συσκευής, να ενεργοποιεί και να θέτει συγκεκριμένους αριθμούς ως εξουσιοδοτημένους αποστολείς των εντολών. Ακόμα και ποια θα είναι τα απαντητικά μηνύματα της συσκευής.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Myke Predko “Προγραμματίζοντας τον Μικροελεγκτή PIC”, Τζιόλα,1998
[2] Κ.Ζ.Πεκμεστζή, “Εργαστήριο Μικροϋπολογιστών”, 2003
[3] Microchip Web Page, http://www.microchip.com/
[4] HI-TECH Software, http://www.htsoft.com/
[5] ETSI Institute Web Page, http://www.etsi.org/
[6] Wavecom AT Command Reference
[7] Wavecom Web Page, http://www.wavecom.com.au/
[8] http://www.gsmfavorites.com/
[9] http://en.wikipedia.org/wiki/Main_Page

ΑΝΑΦΟΡΕΣ

[1] Microchip 18F2520: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010277
[2] Olimex PIC-MT: http://olimex.com/dev/index.html
[3] Microchip MPLAB: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002

[4] http://www.dreamfabric.com/sms/

[5] http://embeddedfreak.wordpress.com/2008/10/09/decoding-gsm-sms-septets/#more-197
[6] http://embeddedfreak.wordpress.com/2008/10/09/encoding-gsm-sms-septets/