ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Απομακρυσμένος Έλεγχος μέσω Σύντομων Γραπτών Μηνυμάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεράσιμος Γ. Φαληρέας

Επιβλέπων : Κιαμάλ Ζ. Πεκμεστζή
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2004

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Απομακρυσμένος Έλεγχος μέσω Σύντομων Γραπτών Μηνυμάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεράσιμος Γ. Φαληρέας

Επιβλέπων : Κιαμάλ Ζ. Πεκμεστζή
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 7η Ιουλίου 2004.

.....	.....	.....
Κιαμάλ Πεκμεστζή	Τσανάκας Παναγιώτης	Κοζύρης Νεκτάριος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2004

.....

Γεράσιμος Γ. Φαληρέας

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Σκοπός αυτής της διπλωματικής εργασίας αποτελεί η ανάπτυξη μιας εφαρμογής που θα βασίζεται σε μια διάταξη ενός μικροελεγκτή. Ποίο συγκεκριμένα αναπτύσσεται ένα σύστημα ελέγχου μέσο του οποίου κάποιος χρήστης θα έχει την δυνατότητα να απενεργοποιήσει ή να θέσει σε λειτουργία τις συσκευές που συνδέονται με αυτό. Ιδιαίτερο χαρακτηριστικό της εφαρμογής αποτελεί η σύνδεση του μικροελεγκτή με ένα κινητό τηλέφωνο *GSM* ώστε να δίνεται η δυνατότητα στους χρήστες να έχουν το έλεγχο του συστήματος από απόσταση μέσο της αποστολής σύντομων γραπτών μηνυμάτων *SMS*. Για την ανάπτυξη της εφαρμογής εξετάστηκαν οι επιμέρους λειτουργίες ενός κινητού τηλεφώνου που αφορούν τη αποστολή και λήψη γραπτού μηνύματος, το πλαίσιο μιας *PDU* συμβολοακολουθίας χαρακτήρων που αναπαριστά ένα γραπτό μήνυμα καθώς και οι αλγόριθμοι κωδικοποίησης και αποκωδικοποίησης που χρησιμοποιούνται σύμφωνα πάντα με τα πρότυπα του Ευρωπαϊκού Οργανισμού *ETSI (European Telecommunication Institute)*.

Για την υλοποίηση της εφαρμογής χρησιμοποιείται ένας μικροελεγκτής *PIC16F877* και ένα κινητό τηλέφωνο *Ericsson T28s*, ενώ ανάπτυξη του κώδικα για το προγραμματισμό του μικροελεγκτή έγινε στην γλώσσα προγραμματισμού *C*.

Λέξεις Κλειδιά

Σύστημα Ελέγχου, Μικροελεγκτής, Κινητό Τηλέφωνο, Σύντομο Γραπτό Μήνυμα.

Abstract

The objective of this thesis is the development of an application that is based on a microcontroller unit and a mobile phone. We develop a control system that gives users the capability to remotely activate and deactivate other devices. The most important characteristic of this application is the communication line between the microcontroller and the GSM mobile phone. In this way users will have the capability of control over the device simply by sending an SMS. For the development of this application we examined those operations of the mobile phone that concern transmission, reception and handling of written messages, we examined a PDU frame (Protocol Description Unit) which consists of a string of characters that represent a written message as well as the algorithms of coding and decoding according to the standards of ETSI Institute (European Telecommunication Institute).

In this application we use the *PIC16F877* microcontroller and the *Ericsson T28s* mobile telephone, while the programming of the microcontroller was implemented in *C*-Programming Language

Key Words

Control System, Microcontroller, Mobile Phone, SMS (Short Message Service)

Πρόλογος

Η εργασία που ακολουθεί αφορά την ανάπτυξη και λειτουργία ενός συστήματος ελέγχου βασιζόμενο σε μια διάταξη μικροελεγκτή και μια συσκευή κινητής τηλεφωνίας GSM. Στόχος της εργασίας αυτής αποτελεί η διασύνδεση και επικοινωνία των παραπάνω διατάξεων με σκοπώ την δημιουργία ενός νέου συστήματος με δυνατότητες και χαρακτηριστικά που καθορίζονται από τις απαιτήσεις και ιδιότητες της εφαρμογής. Η αναφορά που παρουσιάζεται στην συνέχεια αποτελείται από πέντε κεφάλαια το περιεχόμενο των οποίων παρουσιάζεται εν συντομία στην συνέχεια.

Στο Κεφάλαιο 1 γίνεται μια σύντομη περιγραφή της διάταξη ενός μικροελεγκτή, των συστημάτων ελέγχου καθώς και του συστήματος που αναπτύσσεται κατά την εφαρμογή.

Στο Κεφάλαιο 2 γίνεται μια αναλυτική περιγραφή του συστήματος ελέγχου. Παρουσιάζονται τα επιμέρους τμήματα που το αποτελούν, ο τρόπος διασύνδεσης τους καθώς και η διαδικασία χειρισμού του συστήματος μέσο των εντολών και των κανόνων λειτουργίας που θεσπίστηκαν προκειμένου οι χρήστες του να είναι σε θέση να πραγματοποιήσουν ένα επιτυχή χειρισμό ελέγχου.

Το Κεφάλαιο 3 αφιερώνεται εξολοκλήρου στην συσκευή κινητής τηλεφωνίας GSM αναφέροντας τις διαδικασίες, τις εντολές και του αλγορίθμους που χρησιμοποιούνται κατά την διάρκεια λειτουργίας του συστήματος.

Στο Κεφάλαιο 4 περιγράφονται οι μικροελεγκτές της οικογένειας PIC. Παρουσιάζεται αναλυτικά η δομή τους, ο τρόπος λειτουργίας τους καθώς και το πλήθος των περιφερικών που διαθέτουν. Επίσης παρατίθεται το ρεπερτόριο των εντολών με τις οποίες γίνεται ο προγραμματισμός τους.

Τέλος το Κεφάλαιο 5 είναι αφιερωμένο στο λογισμικό της εφαρμογής με την παράθεση των διαγραμμάτων ελέγχου, τον κώδικα του προγράμματος και των κατάλληλων σχόλιων που απαιτούνται για την επαρκή επεξήγηση και κατανόηση της λειτουργίας του προγράμματος.

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε στον εργαστήριο Μικροϋπολογιστών και Ψηφιακών Συστημάτων του Εθνικού Μετσόβιου Πολυτεχνείου. Με την ολοκλήρωση της θα ήθελα να ευχαριστήσω ιδιαίτερα τον υποψήφιο διδάκτορα της σχολής Η.Μ. και Μ.Υ. κ. Κώστα Γκότση για την πολύτιμη και αμέριστη βοήθεια που προσέφερε κατά την διάρκεια ανάπτυξης της καθώς και τον συμφοιτητή μου Δημητριάδη Βασίλειο για τις χρήσιμες συμβουλές και υποδείξεις του

Περιεχόμενα

ΠΕΡΙΕΧΟΜΕΝΑ ..... **- 11 -**

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ ..... **- 13 -**

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ ..... **- 15 -**

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ ..... **- 15 -**

ΚΕΦΑΛΑΙΟ 1 ..... **- 17 -**

ΕΙΣΑΓΩΓΗ ..... **- 17 -**

1.1 ΟΡΙΣΜΟΣ ΜΙΚΡΟΕΛΕΓΚΤΗ ..... - 17 -
1.2 ΟΡΙΣΜΟΣ ΣΥΣΤΗΜΑΤΟΣ ΕΛΕΓΧΟΥ ..... - 19 -
1.3 ΣΥΝΤΟΜΗ ΠΕΡΙΓΡΑΦΗ ΣΥΣΤΗΜΑΤΟΣ ..... - 20 -

ΚΕΦΑΛΑΙΟ 2 ..... **- 23 -**

ΓΕΝΙΚΗ ΠΕΡΙΓΡΑΦΗ ΣΥΣΤΗΜΑΤΟΣ ..... **- 23 -**

2.1 ΓΕΝΙΚΗ ΠΕΡΙΓΡΑΦΗ ..... - 23 -
2.2 ΠΕΡΙΓΡΑΦΗ ΚΥΚΛΩΜΑΤΟΣ ..... - 24 -
2.2.1 Μικροελεγκτής ..... *- 25 -*
2.2.2 Συσκευή Κινητού Τηλεφώνου ..... *- 27 -*
2.2.3 Διασύνδεση RS 232 ..... *- 30 -*
2.2.4 Ασύγχρονη Σειριακή Επικοινωνία ..... *- 31 -*
2.2.5 Παράθεση Κυκλώματος / Λίστα Υλικών ..... *- 33 -*
2.3 ΧΕΙΡΙΣΜΟΣ ΣΥΣΤΗΜΑΤΟΣ ..... - 36 -
2.4 ΠΡΟΫΠΟΘΕΣΕΙΣ ΧΡΗΣΗΣ ΣΥΣΤΗΜΑΤΟΣ ..... - 37 -
2.5 ΠΑΡΑΘΕΣΗ ΕΝΤΟΛΩΝ ΧΕΙΡΙΣΜΟΥ ..... - 38 -
2.5.1 Εντολή Ενεργοποίησης ..... *- 39 -*
2.5.2 Εντολή Απενεργοποίησης ..... *- 40 -*
2.6 ΜΟΡΦΗ ΑΠΟΣΤΟΛΗΣ ΜΗΝΥΜΑΤΟΣ / ΠΑΡΑΘΕΣΗ ΚΑΝΟΝΩΝ ΣΥΝΤΑΞΗΣ ..... - 42 -
2.6.1 Μορφή Αποστολής Μηνύματος ..... *- 42 -*
2.6.2 Κανόνες Σύνταξης ..... *- 43 -*

ΚΕΦΑΛΑΙΟ 3 ..... **- 46 -**

ΣΥΣΚΕΥΗ ΚΙΝΗΤΟΥ ΤΗΛΕΦΩΝΟΥ ..... **- 46 -**

3.1 ΚΡΙΤΗΡΙΑ ΕΠΙΛΟΓΗΣ ΣΥΣΚΕΥΗΣ ..... - 46 -
3.2 ΑΝΑΦΟΡΑ AT ΕΝΤΟΛΩΝ ..... - 47 -
3.2.1 AT+CPMS ..... *- 48 -*
3.2.2 AT+CMGR ..... *- 49 -*
3.2.3 AT+CMGD ..... *- 51 -*
3.2.4 AT+CMGS ..... *- 52 -*
3.2.5 AT+CNMI ..... *- 53 -*
3.2.6 ATE ..... *- 54 -*
3.3 Η ΧΡΗΣΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ HYPERTERMINAL ..... - 55 -
3.4 ΣΥΝΤΟΜΟ ΜΗΝΥΜΑ ..... - 56 -
3.4.1 Υπηρεσία Σύντομων Γραπτών Μηνυμάτων SMS ..... *- 57 -*
3.4.2 PDU Ακολουθία κατά την λήψη μηνύματος ..... *- 58 -*
3.4.3 PDU Ακολουθία κατά την αποστολή μηνύματος ..... *- 60 -*

3.5 Αλγοριθμος Κωδικοποιησης ..... - 62 -
3.6 Αλγοριθμος Αποκωδικοποιησης ..... - 63 -

ΚΕΦΑΛΑΙΟ 4 ..... - 65 -

ΜΙΚΡΟΕΛΕΓΚΤΕΣ ΟΙΚΟΓΕΝΕΙΑΣ PIC ..... - 65 -

4.1 Μικροελεγκτες και Μικροεπεξεργαστες ..... - 65 -
4.2 Αρχιτεκτονικες Harvard και Von-Neumann ..... - 65 -
4.3 Γενικα χαρακτηριστικα των μικροελεγκτων της οικογενειας PIC ..... - 66 -
4.4 Μικροελεγκτες "μεσαιας" (Mid-Range) οικογενειας ..... - 68 -
4.4.1 Αρχιτεκτονική του PIC ..... - 70 -
4.4.2. Ο πυρήνας του PIC ..... - 71 -
4.4.3. Τα περιφερειακά του PIC ..... - 107 -

ΚΕΦΑΛΑΙΟ 5 ..... - 123 -

ΛΟΓΙΣΜΙΚΟ ..... - 123 -

5.1 Περιγραφη Λογισμικου ..... - 123 -
5.2 Λειτουργια του Προγραμματος ..... - 124 -
5.3 Δομη Προγραμματος ..... - 126 -
5.3.1 Αρχείο main.c ..... - 126 -
Α. Ρουτίνα Εξυπηρέτησης Διακοπής (Interrupt Service Routine) ..... - 126 -
Β. Διαδικασία Αρχικοποίησης ..... - 129 -
Γ. Διαδικασία Εξυπηρέτησης Σύντομου Μηνύματος (SMS) ..... - 130 -
Δ. Διαδικασία Εξυπηρέτησης Σήματος Ενεργοποίησης ..... - 132 -
5.3.2 Διασύνδεσης Σειριακής Επικοινωνίας ..... - 132 -
5.3.3 Συναρτήσεις Καθυστέρησης ..... - 134 -
5.4 Παραθεση Κωδικα ..... - 134 -
5.4.1 Αρχείο main.c ..... - 134 -
5.4.2 Αρχείο main.h ..... - 153 -
5.4.3 Αρχείο sci.c ..... - 155 -
5.4.4 Αρχείο sci.h ..... - 157 -
5.4.5 Αρχείο delay.c ..... - 158 -
5.4.6 Αρχείο delay.h ..... - 159 -
5.5 Χαρτης Μνημης ..... - 160 -
5.6 Προγραμματισμος Μικροελεγκτη ..... - 161 -

ΠΑΡΑΡΤΗΜΑ ..... - 162 -

Καθορισμενη GSM αλφαβητο 7-bits ..... - 162 -
Λεξιλογιο ..... - 164 -

ΒΙΒΛΙΟΓΡΑΦΙΑ ..... - 166 -

Κατάλογος Σχημάτων

Σχημα 1.1 Βασικο δομικο διαγραμμα μικροελεγκτη ..... - 18 -
Σχημα 1.2 Λειτουργια συστηματος ελεγχου ..... - 19 -
Σχημα 1.3 Ενισχυση σηματων ελεγχου ..... - 20 -
Σχημα 2.1 Σχηματικη λειτουργια του συστηματος ελεγχου της εφαρμογης - 23 -
Σχημα 2.2 Block διαγραμμα κυκλωματος εφαρμογης ..... - 24 -
Σχημα 2.3 Συστημα Τροφοδοσιας / Χρονισμου μικροελεγκτη ..... - 26 -
Σχημα 2.4 Παρασταση ακροδεκτων τηλεφωνου ..... - 28 -
Σχημα 2.5 Σχηματικη περιγραφη συνδεσης του τηλεφωνου στο κυκλωμα ..... - 30 -
Σχημα 2.6 Ασυγχρονη σειριακη ροη δεδομενων ..... - 31 -
Σχημα 2.7 Κυκλωμα εφαρμογης ..... - 33 -
Σχημα 2.8 Το τμημα του μικροελεγκτη στο κυκλωμα ..... - 34 -
Σχημα 2.9 RS 232 INTERFACE ..... - 35 -
Σχημα 2.10 Το τμημα συσκευων στο κυκλωμα ..... - 35 -
Σχημα 2.11 Δομη μηνυματος ..... - 42 -
Σχημα 3.1 Αρχιτεκτονικη δικτυου SMS ..... - 58 -
Σχημα 3.2 Αλγοριθμος κωδικοποιησης ..... - 62 -
Σχημα 3.3 Αλγοριθμος αποκωδικοποιησης ..... - 64 -
Σχημα 4.1 Αρχιτεκτονικες Harvard και Von Neumann ..... - 66 -
Σχημα 4.2 Αρχιτεκτονικη PIC της οικογενειας 16CXXX ..... - 70 -
Σχημα 4.3 Δραστηριοτητα Q κυκλου (Q cycle activity) ..... - 72 -
Σχημα 4.4 Παραδειγμα συνεχους διοχετευσης εντολων (pipelining) ..... - 73 -
Σχημα 4.5 Αριθμητικη Λογικη Μοναδα (ALU) του PIC ..... - 74 -
Σχημα 4.6 Χαρτης της μνημης προγραμματος (program memory map) ..... - 76 -
Σχημα 4.7 Χαρτης της μνημης δεδομενων (Data Memory) ..... - 78 -
Σχημα 4.8 Άμεση προσπελαση μνημης (Direct Addressing) ..... - 79 -
Σχημα 4.9 Έμμεση προσπελαση μνημης (Indirect Addressing) ..... - 80 -
Σχημα 4.10 Επιλογη του τροπου προσπελασης της μνημης ..... - 81 -
Σχημα 4.11 Τροπος λειτουργιας της στοιβας (stack) ..... - 82 -
Σχημα 4.12 Μετρητης Προγραμματος PC ..... - 83 -
Σχημα 4.13 Απευθειας αναφορα στα 8 χαμηλης αξιας bit του μετρητη προγραμματος ..... - 83 -
Σχημα 4.14 Αλλαγη περιεχομενου του μετρητη για την εντολη GOTO ..... - 84 -
Σχημα 4.15 Αλλαγη περιεχομενου του μετρητη προγραμματος ..... - 85 -
κατα την εκτελεση της εντολης CALL ..... - 85 -
Σχημα 4.16 Αλλαγη περιεχομενου του μετρητη προγραμματος σε περιπτωση επιστροφης απο υποροutινα ..... - 85 -
Σχημα 4.17 Τα bit του καταχωρητη καταστασης (STATUS) ..... - 86 -
Σχημα 4.19 Λογικη των διακοπων (Interrupts) ..... - 90 -
Σχημα 4.20 Κατηγοριες Εντολων του PIC ..... - 92 -
Σχημα 4.21 Αρχιτεκτονικη μοναδας εισοδου – εξοδου (I/O) του PIC ..... - 109 -
Σχημα 4.22 Ακροδεκτες RB7:RB4 μοναδας εισοδου/ εξοδου (I/O) PORT B - 110 -
Σχημα 4.23 Αρχιτεκτονικη (block diagram) του timer0 ..... - 111 -
Σχημα 4.24 Τα bit του καταχωρητη OPTION ..... - 112 -
Σχημα 4.25 Λειτουργια του timer0 χωρις την χρηση του prescaler ..... - 114 -
Σχημα 4.26 Λειτουργια του timer0 με την χρηση του prescaler ..... - 114 -
Σχημα 4.27 Ο καταχωρητης TXSTA ..... - 115 -

Σχημα 4.28 Ο καταχωρητησ RCSTA ..... - 115 -
Σχημα 4.28 Ο καταχωρητησ RCSTA ..... - 115 -
Σχημα 4.29 Ρυθμοι μεταδοσησ για ασυγχρονη κατασταση λειτουργιασ ..... - 116 -
Σχημα 4.30 Αρχιτεκτονικη του πομπου τησ USART ..... - 118 -
Σχημα 4.31 Ασυγχρονη αποστολη μεσω USART ..... - 119 -
Σχημα 4.32 Αρχιτεκτονικη του δεκτη τησ USART ..... - 120 -
Σχημα 4.33 Ασυγχρονη ληψη μεσω USART ..... - 122 -
Σχημα 5.1 Ατερμων βροχοσ ελεγχου ..... - 124 -
Σχημα 5.2 Αποθηκευση δεδομενων PDU ακολουθιασ ..... - 128 -
Σχημα 5.3 Αποθηκευση ειδοποιησησ νεου SMS ..... - 128 -
Σχημα 5.4 Block διαγραμμα τησ διαδικασιασ εξυπηρετησησ SMS ..... - 131 -
Σχημα 5.5 Block διαγραμμα τησ διαδικασιασ εξυπηρετησησ σηματοσ ενεργοποιησησ ..... - 132 -

Κατάλογος Πινάκων

ΠΙΝΑΚΑΣ 2.1 ΕΠΕΞΗΓΗΣΗ ΑΚΡΟΔΕΚΤΩΝ ΤΗΛΕΦΩΝΟΥ ..... - 28 -
ΠΙΝΑΚΑΣ 2.2 ΛΙΣΤΑ ΕΞΑΡΤΗΜΑΤΩΝ ΚΥΚΛΩΜΑΤΟΣ ..... - 36 -
ΠΙΝΑΚΑΣ 3.1 ΕΠΕΞΗΓΗΣΗ ΤΜΗΜΑΤΩΝ PDU ΑΚΟΛΟΥΘΙΑΣ ΜΗΝΥΜΑΤΟΣ ΛΗΨΗΣ ..... - 60 -
ΠΙΝΑΚΑΣ 3.2 ΕΠΕΞΗΓΗΣΗ ΤΜΗΜΑΤΩΝ ΑΚΟΛΟΥΘΙΑΣ ΓΙΑ ΤΗΝ ΑΠΟΣΤΟΛΗ ΜΗΝΥΜΑΤΟΣ - 61 -
ΠΙΝΑΚΑΣ 4.1 ΆΜΕΣΗ ΚΑΙ ΕΜΜΕΣΗ ΠΡΟΣΠΕΛΑΣΗ ΜΝΗΜΗΣ ..... - 77 -
ΠΙΝΑΚΑΣ 4.2 ΣΥΜΒΑΣΕΙΣ ΠΕΡΙΓΡΑΦΗΣ ΕΝΤΟΛΩΝ (INSTRUCTION DESCRIPTION CONVENTIONS) ..... - 93 -
ΠΙΝΑΚΑΣ 4.3 ΡΥΘΜΟΣ ΛΕΙΤΟΥΡΓΙΑΣ ΤΟΥ ΔΙΑΙΡΕΤΗ ΜΕΤΡΗΣΗΣ (PRESCALER) ΜΕ ΒΑΣΗ ΤΗΝ ΤΙΜΗ ΤΩΝ BIT PS2:PS0 ΤΟΥ OPTION REGISTER ..... - 113 -
ΠΙΝΑΚΑΣ 4.4 ΣΥΝΑΡΤΗΣΕΙΣ ΥΠΟΛΟΓΙΣΜΟΥ ΤΟΥ ΡΥΘΜΟΥ ΜΕΤΑΔΟΣΗΣ(BAUD RATE) ΤΗΣ USART ..... - 116 -
ΠΙΝΑΚΑΣ 5.1 ΛΕΙΤΟΥΡΓΙΑ ΣΥΝΑΡΤΗΣΗΣ GET_HEX() ..... - 151 -
ΠΙΝΑΚΑΣ 5.2 ΧΑΡΤΗΣ ΜΝΗΜΗΣ ΜΙΚΡΟΕΛΕΓΚΤΗ ..... - 160 -

Κατάλογος Εικόνων

ΕΙΚΟΝΑ 2.1 ΜΙΚΡΟΕΛΕΓΚΤΗΣ PIC16F877 ..... - 25 -
ΕΙΚΟΝΑ 2.2 ΤΟ ΚΙΝΗΤΟ ΤΗΛΕΦΩΝΟ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ERICSSON T28S ..... - 27 -
ΕΙΚΟΝΑ 2.3 ΣΥΝΔΕΣΗ ΤΗΛΕΦΩΝΟΥ ΜΕ ΤΟ ΚΑΛΩΔΙΟ ΔΕΔΟΜΕΝΩΝ ..... - 29 -
ΕΙΚΟΝΑ 2.4 RS232 DB 9 PIN ASSIGNMENT ..... - 29 -

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

1.1 Ορισμός Μικροελεγκτή

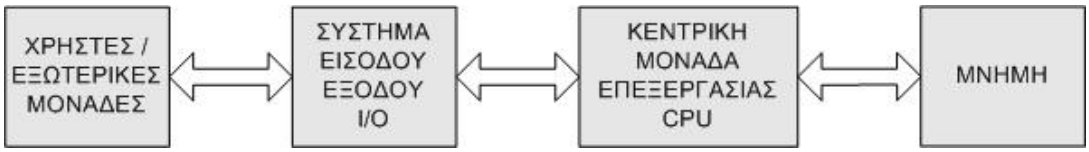
Η διπλωματική αυτή εργασία πραγματεύεται την δημιουργία ενός συστήματος ελεγχόμενου μέσο σύντομων γραπτών μηνυμάτων SMS. Βασικός στόχος μέσα από την επιλογή αυτής της διπλωματικής εργασίας ήταν η σχεδίαση και η ανάπτυξη μιας εφαρμογής η οποία θα βασίζεται σε έναν μικροελεγκτή. Η ανάπτυξη μιας εφαρμογής αυτού του είδους απαιτεί την επιμέρους ανάπτυξη δύο συνιστωσών. Οι συνιστώσες αυτές αποτελούν το υλικό (Hardware) και το λογισμικό (Software) μέρος της εφαρμογής.

Το υλικό μέρος περιλαμβάνει την σχεδίαση του κυκλώματος, την επιλογή των ολοκληρωμένων κυκλωμάτων και των ηλεκτρονικών εξαρτημάτων που απαιτούνται καθώς και την κατασκευή αυτού πάνω σε ένα κομμάτι πλακέτας. Στόχος κατά την ανάπτυξη του υλικού μέρους μια εφαρμογής αποτελεί η αξιόπιστη και συνεχής λειτουργία της, η δυνατότητα προσθήκης ή τροποποίησης τμημάτων της καθώς και η προστασία των επιμέρους κυκλωμάτων που την αποτελούν.

Το λογισμικό μέρος μίας εφαρμογής αφορά την ανάπτυξη και τη δημιουργία κώδικα για τον προγραμματισμό και την λειτουργία του μικροελεγκτή. Αποτελεί ουσιαστικά το κύριο κομμάτι της εφαρμογής αφού καθορίζει και διαμορφώνει την συμπεριφορά του μικροελεγκτή ανάλογα με τις λειτουργίες και τις διαδικασίες που θέλουμε αυτός να εκτελεί. Η συγγραφή του κώδικα του προγράμματος μπορεί να γίνει σε γλώσσα Assembly ή σε γλώσσα υψηλότερου επιπέδου ανάλογα με την εφαρμογή και τις απατήσεις του χρήστη. Ο προγραμματισμός της μνήμης του μικροελεγκτή με το πρόγραμμα της εφαρμογής αποτελεί μια εύκολη και γρήγορη διαδικασία η οποία μπορεί να επαναληφθεί αρκετές φορές.

Πριν προχωρήσουμε στην παρουσίαση της συγκεκριμένης εφαρμογής κρίνεται σκόπιμο στο σημείο αυτό να δώσουμε έναν αρχικό ορισμό και μια σύντομη περιγραφή της διάταξης που αποκαλούμε «μικροελεγκτή». Μπορούμε λοιπόν να

ορίσουμε έναν μικροελεγκτή ως μία υπολογιστική μονάδα ενοποιημένη μέσα σε ένα κομμάτι ολοκληρωμένου κυκλώματος (IC). Χρησιμοποιώντας τον όρο υπολογιστική μονάδα αναφερόμαστε εμμέσως στις βασικές δομικές μονάδες, της κεντρικής μονάδα επεξεργασίας (CPU), της μνήμης και του συστήματος εισόδου/ εξόδου (I/O) που συνιστούν ένα υπολογιστικό σύστημα. Η δομή ενός μικροελεγκτή παρουσιάζεται στο διάγραμμα του σχήματος 1.1 και όπως παρατηρούμε είναι ανάλογη με το βασικό δομικό διάγραμμα μιας υπολογιστικής μονάδας.



Σχήμα 1.1 *Βασικό δομικό διάγραμμα μικροελεγκτή*

Ωστόσο οι μονάδες που περιέχει ένας μικροελεγκτής δεν περιορίζονται μόνο στα βασικά δομικά του στοιχεία. Μια μεγάλη ποικιλία από περιφερειακά, χρονιστές (timers), μετατροπείς τάσης (A/D converters), απαριθμητές (counters) και άλλα συμπληρώνουν και συνθέτουν μαζί μια ευέλικτη και ισχυρή υπολογιστική μονάδα. Με κατάλληλο προγραμματισμό οι μικροελεγκτές μπορούν να χρησιμοποιηθούν σε μια πλειάδα εφαρμογών προσφέροντας συνεχή και αξιόπιστη λειτουργία.

Βασικά χαρακτηριστικά των μικροελεγκτών αποτελούν το μικρό τους μέγεθος, η μικρή κατανάλωση ισχύος και το χαμηλό τους κόστος. Για την λειτουργία τους δεν απαιτείται παρά η εφαρμογή της τάσης τροφοδοσίας και ενός εξωτερικού σήματος χρονισμού για να αρχίσει ο μικροελεγκτής να εκτελεί τις εντολές του προγράμματος με τις οποίες είναι προγραμματισμένος. Οι μικροελεγκτές χρησιμοποιούνται συνήθως σε εργασίες όπου η υπολογιστική ισχύς δεν αποτελεί κύριο χαρακτηριστικό της εφαρμογής. Το γεγονός αυτό όμως δεν εμποδίζει να χρησιμοποιούνται σε μια πληθώρα εφαρμογών και εργασιών που ξεκίνα από την χρήση τους σε οικιακές ηλεκτρονικές συσκευές και φτάνει μέχρι τα αυτοκίνητα και τις διατάξεις αυτοματισμών στην βιομηχανία και την παραγωγή.

Ολοκληρώνοντας αυτή την σύντομη αναφορά στους μικροελεγκτές θα πρέπει να αναφέρουμε επιπλέον και ορισμένες πληροφορίες που αφορούν την οικονομική τους διάσταση στην παγκόσμια αγορά των ημιαγωγών. Οι πωλήσεις των μικροελεγκτών

σε ετήσια βάση ξεπερνούν τα 3 δισεκατομμύρια ολοκληρωμένα κυκλώματα και υπολογίζεται ότι είναι περισσότερες από το μισό των πωλήσεων που επιτυγχάνουν οι μικροεπεξεργαστές στο αντίστοιχο χρονικό διάστημα. Το γεγονός αυτό αλλά και η απλότητα τους σε σχέση με άλλες διατάξεις ημιαγωγών (όπως οι επεξεργαστές) καθιστούν το κόστος τους ιδιαίτερα χαμηλό και προσιτό. Η σχέση μεταξύ πωλήσεων και κόστους στους μικροελεγκτές διαμορφώνει την οικονομική τους απόδοση στην παγκόσμια αγορά των ημιαγωγών σε ποσοστό που ανέρχεται στο 15% των συνολικών κερδών.

1.2 Ορισμός Συστήματος Ελέγχου

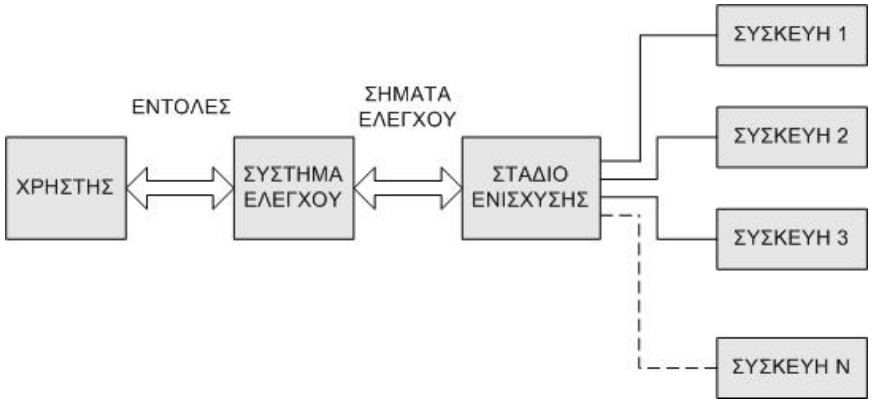
Η ανάγκη για την δημιουργία και ανάπτυξη των συστημάτων ελέγχου ήλθε σαν απόρροια της εξέλιξης συστημάτων και μηχανών. Καθώς η απαίτηση για ποίο δύσκολες καις εξεζητημένες εργασίες μεγάλωνε δημιουργήθηκε η ανάγκη χειρισμού και ελέγχου των συστημάτων με ένα ποίο αποδοτικό και επισφαλή τρόπο. Την ανάγκη αυτή έρχεται να καλύψει με την λειτουργία του το σύστημα ελέγχου το οποίο μεσολαβεί μεταξύ του χρήστη και του συστήματος στο οποίο θα επιτευχθεί ο έλεγχος.

Σχήμα 1.2 *Λειτουργία συστήματος ελέγχου*

Ο χρήστης είναι πλέον σε θέση να στέλνει τις εντολές του μέσο ενός συστήματος ελέγχου και αυτό στην συνέχεια αναλαμβάνει να τις αποκωδικοποιήσει και να οδηγήσει το σύστημα δημιουργώντας τα κατάλληλα σήματα ελέγχου. Οι λειτουργίες που μπορεί κάποιος να επιτελέσει μέσο ενός συστήματος ελέγχου ποικίλουν και εξαρτώνται από την εργασία και το είδος του ελέγχου που θέλουμε να επιτευχθεί. Ξεκινώντας από μια απλή ενεργοποίηση ή απενεργοποίηση μίας συσκευής ο έλεγχος μπορεί να φτάσει μέχρι και την πλήρη αυτοματοποίηση διαδικασιών, βελτιστοποιώντας την απόδοση και την λειτουργία τους.

1.3 Σύντομη περιγραφή Συστήματος

Το σύστημα ελέγχου που αναπτύχθηκε στα πλαίσια αυτής της διπλωματικής εργασίας λειτουργεί ανάλογα με μια συστοιχία διακοπτών στην οποία συνδέονται οι συσκευές που θα πραγματοποιηθεί ο έλεγχος. Με βάση το γεγονός αυτό οι χρήστες έχουν δυνατότητα να απενεργοποιήσουν ή να θέσουν σε λειτουργία τις συσκευές που βρίσκονται συνδεμένες με αυτό. Ο τρόπος με τον οποίο γίνεται κάτι τέτοιο παρουσιάζεται στο σχήμα 1.3 και επιτυγχάνεται μέσο της δημιουργίας ηλεκτρικών ψηφιακών σημάτων τα οποία αφού περάσουν από το στάδιο ενίσχυσης είναι σε θέση να λειτουργήσουν ως τάση τροφοδοσίας για τις συσκευές.



Σχήμα 1.3 *Ενίσχυση σημάτων ελέγχου*

Για την λειτουργία και το χειρισμό του συστήματος ελέγχου από τον χρήστη δημιουργήθηκε ένα βασικό ρεπερτόριο εντολών μέσο του οποίου θα παρέχεται η δυνατότητα ελέγχου και επικοινωνίας με τις συσκευές. Παράλληλα αναπτύχθηκε η διαδικασία αποκωδικοποίησης των εντολών αυτών από την πλευρά του συστήματος όπως και η διαδικασία δημιουργίας των κατάλληλων σημάτων που απαιτούνται για τον έλεγχο των συσκευών.

Ένα σημείο το οποίο αποτέλεσε και την μεγαλύτερη πρόκληση κατά την διάρκεια ανάπτυξης του συστήματος ήταν να δώσουμε στους χρήστες την δυνατότητα έλεγχου από απόσταση. Το χαρακτηριστικό αυτό, ιδιαίτερα σημαντικό, θα έδινε μεγάλη ευελιξία και ταυτόχρονα μεγάλη ευκολία κατά την χρήση του συστήματος αφού δεν θα είναι πλέον απαραίτητη η παρουσία του χρήστη στις συσκευές και κατ' επέκταση στο σύστημα ελέγχου. Η επίτευξη ωστόσο κάτι τέτοιου προϋποθέτει την παρουσία δύο στοιχείων. Το πρώτο αφορά την ύπαρξη ενός τερματικού μέσο του οποίου θα

γίνεται έμμεσα ο έλεγχος, ενώ το δεύτερο αποτελεί το δίκτυο με το οποίο θα γίνει εφικτή η επικοινωνία μεταξύ τερματικού και συστήματος ελέγχου. Μια από τις δυνατότητες που υπήρχαν για την επιλογή των τερματικών και της σύνδεσης ήταν η χρήση υπολογιστών οι οποίοι θα συνδέονται με το σύστημα ελέγχου μέσο ενός ενσύρματου τοπικού δικτύου (LAN) ή μέσο του διαδικτύου (internet). Η επιλογή αυτή ωστόσο κρίθηκε μη ικανοποιητική αφού δεν εξυπηρετούσε τόσο από την πλευρά των τερματικών όσο και από την πλευρά του δικτύου. Η επιλογή ενός υπολογιστή στην θέση του τερματικού προϋποθέτει την γνώση και την εμπειρία χειρισμού από το σύνολο των χρηστών ενώ παράλληλα περιορίζει όσους δεν έχουν άμεση πρόσβαση σε αυτόν. Όσο αναφορά το δίκτυο η σύνδεση του συστήματος ελέγχου σε αυτό θα απαιτούσε επιπλέον εξοπλισμό και υποδομή ενώ το γεγονός ότι είναι ενσύρματο περιορίζει σημαντικά την πρόσβαση και την ευελιξία των χρηστών. Λόγο των προβλημάτων που αναφέρθηκαν η επιλογή δικτύου και τερματικού θα έπρεπε να γίνει ικανοποιώντας ορισμένες προϋποθέσεις. Αρχικά η επιλογή τερματικού θα έπρεπε να είναι τέτοια ώστε να εξυπηρετεί τη πλειονότητα των χρηστών τόσο από την πλευρά πρόσβασης σε αυτό όσο και από την πλευρά γνώσεων και χειρισμού κατά την λειτουργία του. Επίσης όσο αφορά την επιλογή του δικτύου θα ήταν από όλες τις απόψεις προτιμότερο να χρησιμοποιηθεί ένα ασύρματο δίκτυο επικοινωνίας γεγονός το οποίο θα έδινε ακόμα μεγαλύτερη ελευθερία και περισσότερες δυνατότητες χειρισμού στους χρήστες. Με βάση τα στοιχεία που αναφέρθηκαν για την επιλογή του τερματικού κρίθηκε ως καλύτερη επιλογή η χρήση ενός κινητού τηλεφώνου GSM ενώ για το δικτύου η χρήση του δικτύου κινητής τηλεφωνίας.

Τα κινητά τηλέφωνα αποτελούν στις μέρες μια από τις ποιο συνηθισμένες και ευρύτερα διαδεδομένες ηλεκτρονικές συσκευές. Το μεγαλύτερο μέρος του πληθυσμού κάνει χρήση του δικτύου κινητής τηλεφωνίας για επικοινωνία, ενημέρωση ή ψυχαγωγία. Μέσο της συσκευής του κινητού τηλεφώνου οποιοσδήποτε έχει την δυνατότητα να συνομιλήσει με άλλους χρήστες, να ανταλλάξει μηνύματα κειμένου (SMS), ήχου, εικόνας και γενικά κάθε είδους δεδομένα τα οποία μπορούν να κωδικοποιηθούν και να αποσταλούν μέσο του δικτύου. Τα δίκτυα κινητής τηλεφωνίας επίσης αποτελούν σήμερα ένα από τα ποίο σύγχρονα και ευρύτερα διαδεδομένα δίκτυα που καλύπτουν το μεγαλύτερο μέρος της χώρας στην οποία εγκαθίστανται.

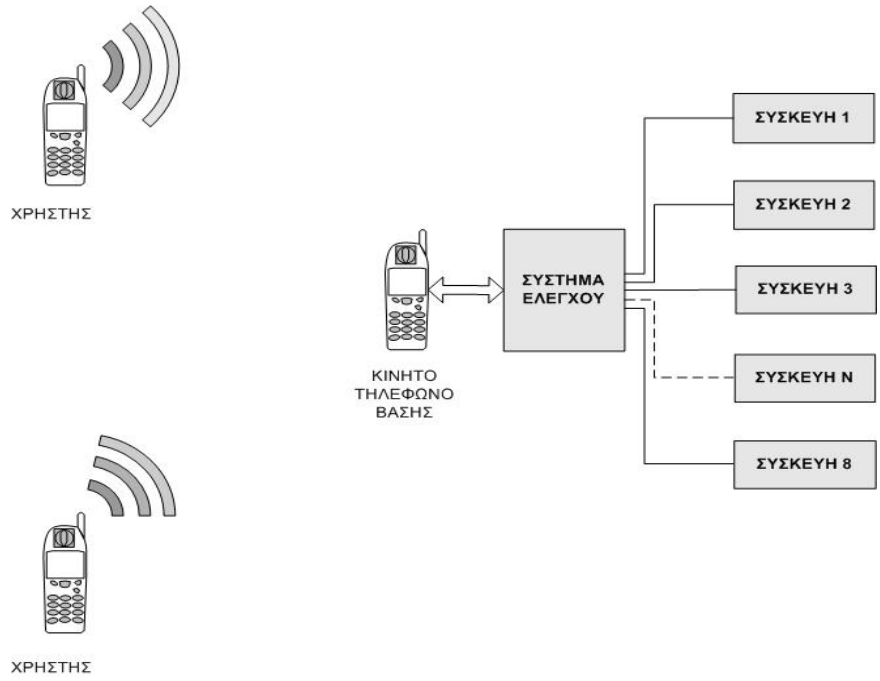
Ο τρόπος με τον οποίο θα γίνει ο έλεγχος του συστήματος μέσο του κινητού τηλεφώνου είναι χρησιμοποιώντας την υπηρεσία σύντομων γραπτών μηνυμάτων SMS. Η υπηρεσία αυτή επιτρέπει την σύνταξη ενός γραπτού μηνύματος μέχρι 140 χαρακτήρες και την αποστολή στο κινητό τηλέφωνο ενός χρήστη προσδιοριζόμενο από τον αριθμό του κινητού του τηλεφώνου. Η χρησιμοποίηση του κινητού τηλεφώνου στην εφαρμογή που αναπτύσσουμε γίνεται μέσο της σύνδεσης του με τον μικροελεγκτή του συστήματος. Τη συσκευή αυτή θα την αποκαλούμε «κινητό τηλέφωνο βάσης» ώστε να την ξεχωρίζουμε από τις συσκευές που χρησιμοποιούν οι χρήστες για τον έλεγχο του συστήματος. Ο μικροελεγκτής εκτός από τις διαδικασίες για την αποκωδικοποίηση των εντολών και την παραγωγή των κατάλληλων σημάτων ελέγχου αναλαμβάνει επίσης και την λειτουργία του κινητού τηλεφώνου και ειδικότερα εκείνες τις διαδικασίες που αφορούν την αποστολή και λήψη ενός σύντομου μηνύματος. Με τον τρόπο αυτό το κινητό τηλέφωνο γίνεται μια περιφερική μονάδα του μικροελεγκτή και ειδικότερα ένα πομπός - δέκτης με εμβέλεια θεωρητικά σε ολόκληρο τον κόσμο και πρόσβαση σε εκατομμύρια τερματικά. Για τον έλεγχο του συστήματος οι χρήστες δεν έχουν παρά να συντάξουν ένα σύντομο μήνυμα με τις εντολές ελέγχου και να το αποστείλουν στο κινητό τηλέφωνο βάσης. Ο μικροελεγκτής στην συνέχεια θα αναλάβει την λήψη του μηνύματος από τον κινητό τηλέφωνο, 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 μέσο του κινητού τηλεφώνου βάσης προς το χρήστη ότι ο έλεγχος ολοκληρώθηκε με επιτυχία.

ΚΕΦΑΛΑΙΟ 2

ΓΕΝΙΚΗ ΠΕΡΙΓΡΑΦΗ ΣΥΣΤΗΜΑΤΟΣ

2.1 Γενική περιγραφή

Το σύστημα ελέγχου που αναπτύσσεται αποτελείται ουσιαστικά από ένα μικροελεγκτή και μια συσκευή κινητής τηλεφωνίας GSM. Στην διάταξη αυτή θα υπάρχει η δυνατότητα σύνδεσης μέχρι και οκτώ συσκευών τις οποίες θα μπορούμε να απενεργοποιήσουμε ή να θέσουμε αντίστοιχα σε λειτουργία. Ο χειρισμός του συστήματος ελέγχου και κατεπέκτασιν των συσκευών μπορεί να γίνει μεμονωμένα από πολλούς χρήστες μέσο της αποστολής σύντομων μηνυμάτων SMS από το κινητό τους τηλέφωνο.



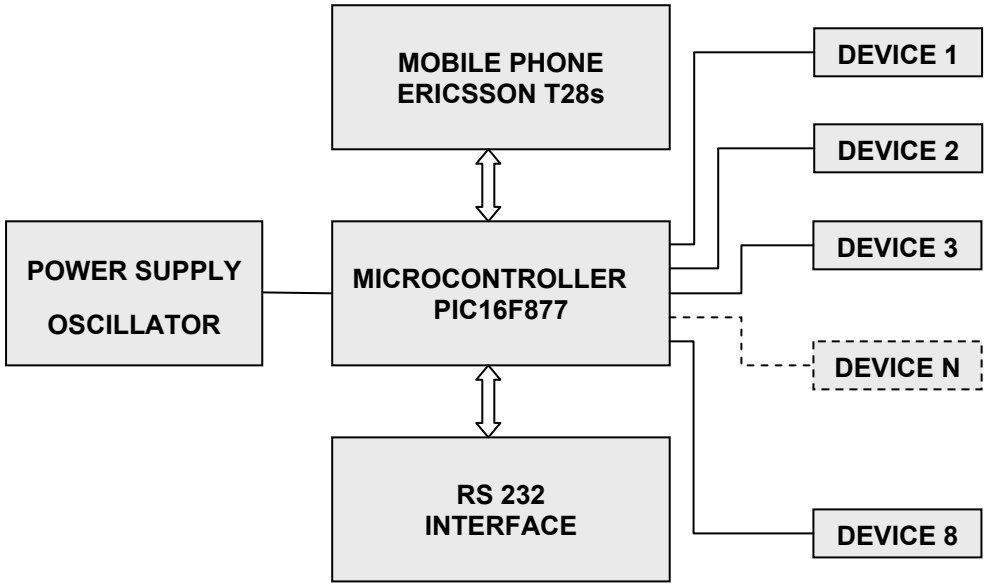
Σχήμα 2.1 *Σχηματική λειτουργία του συστήματος ελέγχου της εφαρμογής*

Η συσκευή κινητής τηλεφωνίας που συνδέεται με τον μικροελεγκτή θα την αποκαλούμε από εδώ και στο εξής «κινητό τηλέφωνο βάσης» για να την ξεχωρίζουμε με τον τρόπο αυτό από τις συσκευές των κινητών τηλεφώνων που χρησιμοποιούν οι

χρήστες για τον έλεγχο του συστήματος. Για να ελέγξει λοιπόν ένας χρήστης τις συσκευές του συστήματος θα πρέπει να στήλη γραπτό μήνυμα στο κινητό τηλέφωνο βάσης στο οποίο θα περιγράφονται κωδικοποιημένα οι συσκευές που θέλει να ελέγξει καθώς και το είδος το ελέγχου (ενεργοποίηση, απενεργοποίηση) που θέλει να εκτελέσει. Στο σχήμα 2.1 παρουσιάζεται σχηματικά η λειτουργία του συστήματος της εφαρμογής.

2.2 Περιγραφή Κυκλώματος

Τα βασικά μέρη που απαρτίζουν το κύκλωμα της εφαρμογής είναι ένας μικροελεγκτής, η την πηγή τροφοδοσίας, το σύστημα χρονισμού του (ταλαντωτής), ένα κινητό τηλέφωνο GSM (κινητό τηλέφωνο βάσης), η διασύνδεση RS 232 για την επικοινωνία του μικροελεγκτή με το κινητό τηλέφωνο καθώς και οι συσκευές στις οποίες θα γίνει ο έλεγχος. Ο τρόπος διασύνδεσης των επιμέρους αυτών τμημάτων παρουσιάζεται στο block διάγραμμα του σχήματος 2.2 .



Σχήμα 2.2 *Block διάγραμμα κυκλώματος εφαρμογής*

Όπως παρατηρούμε και από το διάγραμμα καρδία του συστήματος ελέγχου αποτελεί ο μικροελεγκτής ο οποίος συνδέεται απευθείας με το κινητό τηλέφωνο ενώ

παράλληλα είναι αυτός που δημιουργεί τα κατάλληλα σήματα ελέγχου που θα οδηγήσουν στην συνέχεια τις συσκευές.

2.2.1 Μικροελεγκτής

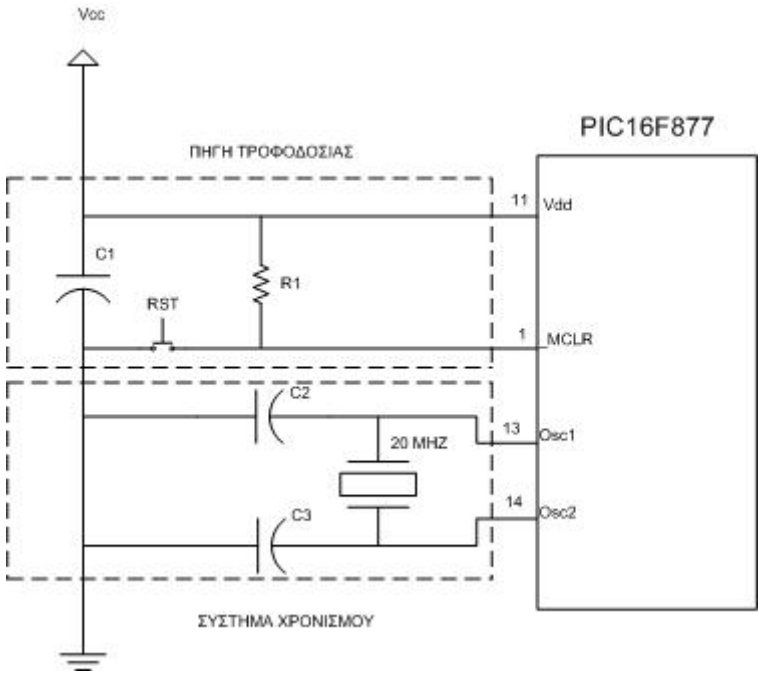
Καρδία του κυκλώματος της εφαρμογής αποτελεί ένας μικροελεγκτής *PIC16F877* της εταιρίας Microchip Inc [5] (εικόνα 2.1).

Εικόνα 2.1 *Μικροελεγκτής PIC16F877*

Ο μικροελεγκτής αυτός ανήκει στην κατηγορία μικροελεγκτών μεσαίας τάξης (Mid-range PICs) με μήκος λέξης εντολής 14-bit.
Βασικά του χαρακτηριστικά αποτελούν:

- Η υψηλής απόδοσης κεντρική μονάδα επεξεργασίας (Central Processing Unit, CPU) τύπου RISC
- Το ρεπερτόριο εντολών 35 λέξεων
- Η 8K x 14 words μνήμη προγράμματος (Program Memory, FLASH)
- Η 368 x 8 bytes μνήμη δεδομένων (Data Memory, RAM)
- Η 256 x 8 bytes μνήμη δεμένων EEPROM
- Η δυνατότητα πολλαπλών διακοπών
- Η πληθώρα περιφερειακών που διαθέτει όπως
    - προγραμματιζόμενες πόρτες εισόδου-εξόδου
    - χρονιστές
    - ο πομπός-δέκτης σύγχρονης σειριακής επικοινωνίας (USART),
    - ο μετατροπέας αναλογικού σε ψηφιακό (Analog 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 χρησιμοποιούμε είναι κρυσταλλικού τύπου και παράγεται με την βοήθεια ενός κρυστάλλου των 20MHz. Η σύνδεση του με τον μικροελεγκτή παρουσιάζεται στο σχήμα 2.3 και όπως παρατηρούμε αποτελείται από τον κρύσταλλο και τους πυκνωτές C2, C3. Η διάταξη αυτή έχει την δυνατότητα να παρέχει σήμα χρονισμού υψηλής ταχύτητας (HS) και ακρίβειας που φτάνει σε ποσοστό το 0,002%. Η επιλογή του κυκλώματος χρονισμού αυτού του τύπου έγινε με βάση τις απαιτήσεις της εφαρμογής και ειδικότερα λόγο της μετάδοσης σειριακών δεδομένων μεταξύ του μικροελεγκτή και του κινητού τηλεφώνου.



Σχήμα 2.3 *Σύστημα Τροφοδοσίας / Χρονισμού μικροελεγκτή*

Η σύνδεση και η επικοινωνία του μικροελεγκτή με το κινητό τηλέφωνο γίνεται μέσο του Πομπού / Δέκτη Ασύγχρονης / Σύγχρονης Σειριακής Επικοινωνίας (USART, Universal Synchronous Asynchronous Receiver Transmitter). Μέσο του προγράμματος εφαρμογής αρχικοποιούμε κατάλληλα την *USART* ώστε πετύχουμε μια ασύγχρονη και ταυτόχρονα διπλής κατεύθυνσης (asynchronous full duplex) επικοινωνία με το κινητό τηλέφωνο βάσης. Για την σύνδεση του μικροελεγκτή με το κύκλωμα απαιτείται η χρήση δύο Pins (25, 26) του μικροελεγκτή εκ των οποίων το

πρώτο είναι για την αποστολή (TX) και το δεύτερο για τη λήψη (RX).των σημάτων επικοινωνίας.

Για την παραγωγή των σημάτων ελέγχου που θα οδηγούν τις συσκευές έχει αφιερωθεί μια από τις θύρες γενικής χρήσης του μικροελεγκτή (PORT D). Η Θύρα αυτή αφού διαμορφωθεί ως ψηφιακή θύρα εξόδου με τους οκτώ ακροδέκτες που διαθέτει μας δίνει την δυνατότητα παραγωγής οκτώ σημάτων ελέγχου. Κάθε ακροδέκτης μπορεί μεμονωμένα και ανάλογα με τις εντολές του χρήστη να διαμορφωθεί σε επίπεδο τάσης 0 ή +5 Volt (λογικό1 ή 0 αντίστοιχα). Τα σήματα αυτά ωστόσο δεν είναι σε θέση από μόνα τους να τροφοδοτήσουν και να οδηγήσουν κάποια από τις συσκευές του συστήματος πρώτον διότι δεν έχουν την απαραίτητη ισχύ και δεύτερον γιατί υπάρχει κίνδυνος να «τραβήξουν» ρεύμα από το ολοκληρωμένο του μικροελεγκτή γεγονός που μπορούσε να αποβεί επιζήμιο για την λειτουργία του. Για τον λόγο αυτό επιβάλλετε πριν την χρήση τους η μεσολάβηση ενός σταδίου ενίσχυσης μέσο του οποίου θα αποκτήσουν την κατάλληλη ισχύ που απαιτείται προκειμένου να χρησιμοποιηθούν για την τροφοδότηση των συσκευών.

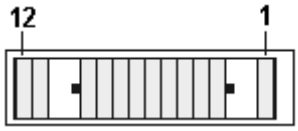
2.2.2 Συσκευή Κινητού Τηλεφώνου

Το κινητό τηλέφωνο που χρησιμοποιείται στο σύστημα ελέγχου της εφαρμογής το αποκαλούμε όπως αναφέρθηκέ και πρωτύτερα κινητό τηλέφωνο βάσης. Σκοπός της συσκευής αυτής στην λειτουργία του συστήματος αποτελεί η λήψη και η αποστολή σύντομων γραπτών μηνυμάτων SMS από και προς τους χρήστες.

Εικόνα 2.2 *Το κινητό τηλέφωνο της εφαρμογής Ericsson T28s*

Το κινητό τηλέφωνο βάσης που επιλέχτηκε είναι μια εμπορική συσκευή κινητής τηλεφωνίας GSM της εταιρίας *Sony Ericsson Mobile Communications* [6] με την ονομασία *Ericsson T28s* (εικόνα 2.2).

Ο χειρισμός των επιμέρους λειτουργιών του τηλεφώνου βάσης που λαμβάνουν χώρα κατά την διάρκεια λειτουργίας του συστήματος γίνεται από τον μικροελεγκτή μέσο της αποστολής AT εντολών (AT commands). Οι εντολές αυτές είναι παρόμοιες με τις εντολές που δέχονται οι συσκευές Modem και μέσο αυτών μπορούν να εκτελεστούν και να ελεγχθούν πλήθος διαδικασιών και λειτουργιών που αφορούν τη συσκευή. Η σύνδεση του στο κύκλωμα γίνεται εκμεταλλευόμενοι την δυνατότητα που διαθέτει να συνδέεται με άλλα περιφερειακά όπως Modem και ηλεκτρονικούς υπολογιστές. Αν παρατηρήσουμε στο κάτω μέρος της συσκευής θα δούμε τις υποδοχές σύνδεσης δίπλα σε αυτές που αφορούν την φόρτιση της μπαταρίας λειτουργίας. Στο σχήμα 2.4 καθώς και στον πίνακα 2.1 που ακολουθεί παρατείθεται η επεξήγηση και η λειτουργία κάθε ακροδέκτη (Pin) σύνδεσης.



Σχήμα 2.4 *Παράσταση ακροδεκτών τηλεφώνου*

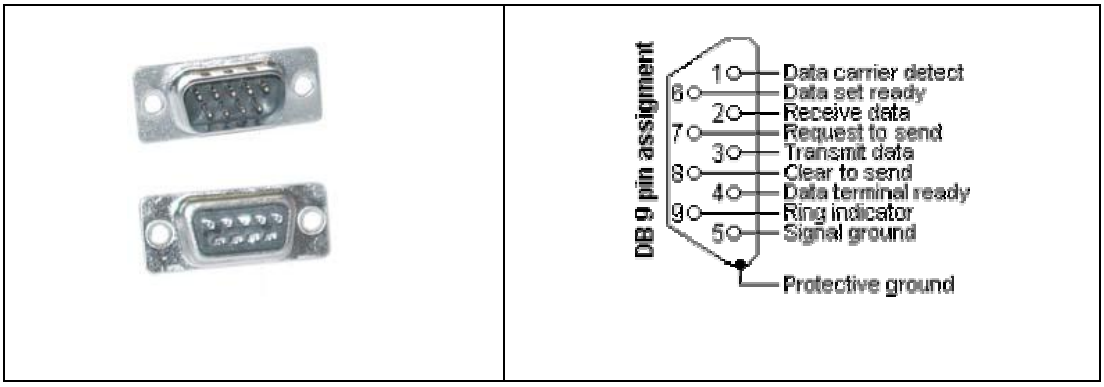
Pin	ΟΝΟΜΑ	ΛΕΙΤΟΥΡΓΙΑ	in/out
1	In DC	DC in for battery charging	in/out
2	DATA IN / RX	Data received	in
3	GND	Ground digital	
4	DATA OUT / TX	Data transmit	out
5	+ 5 Volt	+5V output Limited	out
6	Test	Switch phone off and provide +5V.	Test
7	MUTE	Mute 0-Normal, 1-In Call	
8	Internal/external	Portable handsfree In	
9	GND	Ground Analog	
10	Accessory	Related to Mic/Speak	
11	BF in	BF input	in
12	BF out	BF output	out

Πίνακας 2.1 *Επεξήγηση ακροδεκτών τηλεφώνου*

Όπως μπορούμε να δούμε το κινητό τηλέφωνο διαθέτει, παρόμοια με τον μικροελεγκτή, ακροδέκτες για την αποστολή TX και λήψη RX σειριακών δεδομένων και είναι αυτοί που θα χρησιμοποιηθούν στην συνέχεια για την επικοινωνία με τον μικροελεγκτή.

Εικόνα 2.3 *Σύνδεση τηλεφώνου με το καλώδιο δεδομένων*

Για την επίτευξη της σύνδεση του τηλεφώνου βάσης μέσο των ακροδεκτών αποστολής TX και λήψης RX χρησιμοποιείται το ειδικό καλώδιο σύνδεσης (data cable), που προορίζεται για την διασύνδεση του με άλλα περιφερειακά μέσο της σειριακής θύρας δεδομένων. Όπως φαίνεται και στην εικόνα 2.3 στο ένα άκρο του καλωδίου αυτού βρίσκεται το βύσμα σύνδεσης με το κινητό τηλέφωνο ενώ το άλλο άκρο καταλήγει σε ένα βύσμα τύπου RS-232 (DB9 female connector).



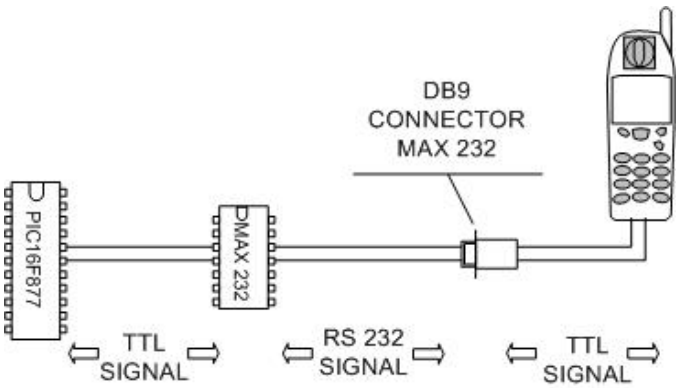
Εικόνα 2.4 *RS232 DB 9 pin assignment*

Για την σύνδεση του καλωδίου με τον μικροελεγκτή χρησιμοποιείται όπως φαίνεται και στην εικόνα 2.4 ένα αρσενικό βύσμα RS-232D (B9 male connector) το οποίο ενώνεται αντίστοιχα με το βύσμα του καλωδίου σύνδεσης. Με τον τρόπο αυτό

ενώνονται οι ακροδέκτες αποστολής TX, λήψης RX και γείωσης GND με τους αντίστοιχους ακροδέκτες του μικροελεγκτή ώστε να επιτευχθεί μια σταθερή επικοινωνία μεταξύ τους όπως φαίνεται και στο διάγραμμα του κυκλώματος.

2.2.3 Διασύνδεση RS 232

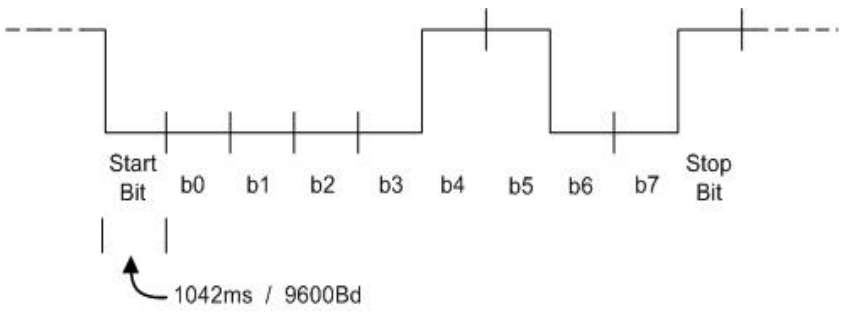
Τα σήματα που αναπτύσσονται τόσο στους ακροδέκτες του μικροελεγκτή όσο και του κινητού τηλεφώνου βάσης είναι ψηφιακά σήματα τεχνολογίας TTL μεταξύ 0 και +5 Volt. Το γεγονός αυτό σημαίνει ότι πρόκειται για συμβατά μεταξύ τους σήματα τα οποία θα μπορούσαν να συνδέσουν άμεσα τις δύο συσκευές χωρίς την μεσολάβηση κάποιου μετατροπέα σήματος. Για την σύνδεση ωστόσο του τηλεφώνου βάσης στο κύκλωμα χρησιμοποιείται για πρακτικούς λόγους το συριακό καλώδιο σύνδεσης με υπολογιστή (data cable) το οποίο περιέχει ένα ολοκληρωμένο κύκλωμα μετατροπής (MAX232) σημάτων TTL του τηλεφώνου σε σήματα τύπου RS-232. Τα σήματα αυτά, με τα οποία είναι συμβατή η συριακή θύρα του υπολογιστή, αναπαριστούν τα λογικά επίπεδα 1 και 0 με τάσεις που κυμαίνονται από +12 Volt ως -12 Volt σε αντίθεση δηλαδή με τα σήματα τεχνολογίας TTL που χρησιμοποιούνται τάσης από +5 Volt ως -5 Volt. Με βάση το γεγονός αυτό για μετατρέψουμε και πάλι το σήμα από την μια άκρη του καλωδίου σύνδεσης σε σήμα τύπου TTL χρησιμοποιούμε ένα επιπλέον ολοκληρωμένο κύκλωμα μετατροπής (MAX 232) το οποίο μεσολαβή πριν την σύνδεση με τον μικροελεγκτή. Η παραπάνω διαδικασία παρουσιάζεται στο σχήμα 2.5 που ακολουθεί.



Σχήμα 2.5 *Σχηματική περιγραφή σύνδεσης του τηλεφώνου στο κύκλωμα*

2.2.4 Ασύγχρονη Σειριακή Επικοινωνία

Η επικοινωνία του κινητού τηλεφώνου με τον μικροελεγκτή πραγματοποιείται μέσο της αποστολής και λήψης σειριακών δεδομένων με ασύγχρονο τρόπο. Σύμφωνα με αυτή την διαδικασία επικοινωνίας δεν απαιτείται η παρουσία σήματος χρονισμού, μεταξύ πομπού και δέκτη, ώστε να πραγματοποιηθεί η διακίνηση των δεδομένων. Η μορφή που παρουσιάζει η εκπομπή σειριακών δεδομένων παρουσιάζεται στο σχήμα 2.6 . Στο παράδειγμά αυτό φαίνεται η αποστολή ενός byte δεδομένων (0x30 - 0b00110000) το οποίο παριστά στην ASCII κωδικοποίηση τον χαρακτήρα ‘0’ (μηδέν). Όπως μπορούμε να παρατηρήσουμε εκτός από τα bit δεδομένων χρησιμοποιούνται επιπρόσθετα δυο ακόμα bit. Αυτά είναι το bit εκκίνησης (Start bit) και το bit τερματισμού (Stop bit) των οποίων η παρουσία αντικαθιστά το σήμα χρονισμού. Η έναρξη της αποστολής δεδομένων επισημάνεται στο δέκτη με την πτώση τάσης της γραμμής σε χαμηλό δυναμικό για το χρονικό διάστημα ενός bit. Στην συνέχεια ακολουθεί η αποστολή των bit δεδομένων ξεκινώντας με το bit χαμηλότερης τάξης του χαρακτήρα αποστολής. Τα δεδομένα αυτά ανάλογα με την κωδικοποίηση δεδομένων που χρησιμοποιείται μπορεί να περιέχουν 8 ή 9 bit. Για την εφαρμογή επιλέγεται η αποστολή 8 bit δεδομένων. Η αποστολή ολοκληρώνεται με την επιστροφή της γραμμής σε υψηλό δυναμικό για χρόνο ενός bit γεγονός το οποίο υποδεικνύει την ύπαρξη bit τερματισμού (Stop bit).



Σχήμα 2.6 *Ασύγχρονη σειριακή ροή δεδομένων*

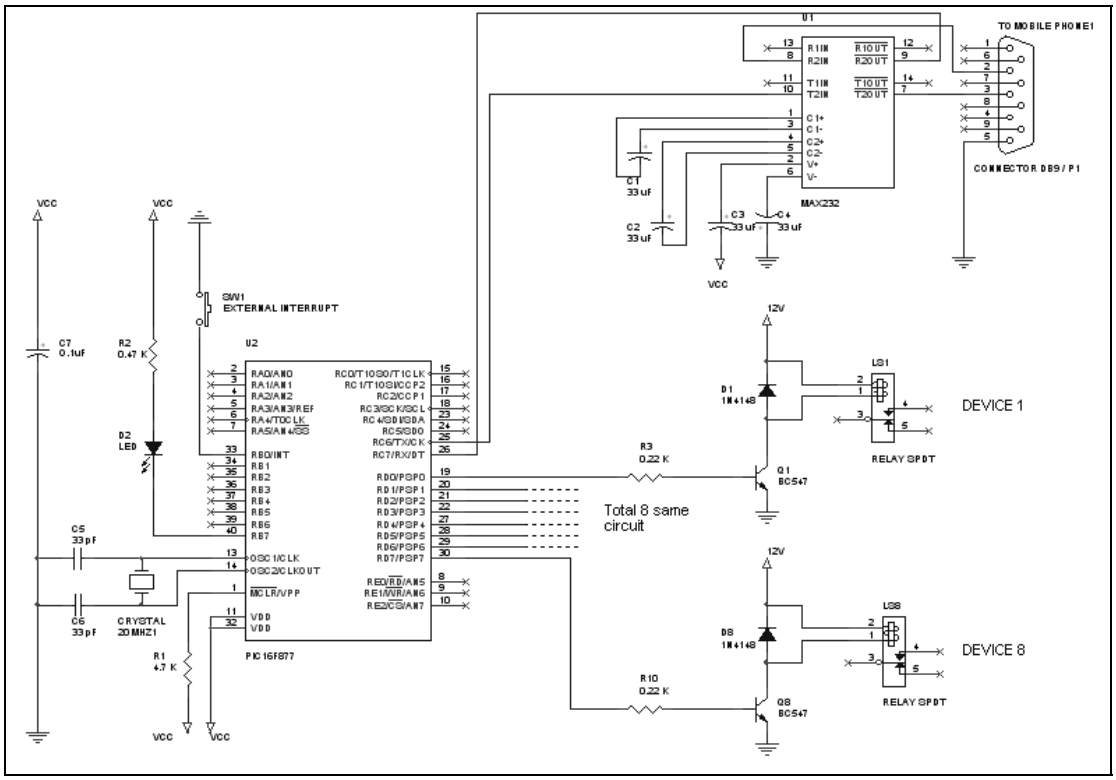
Στο σημείο αυτό θα πρέπει να επισημάνουμε τη δυνατότητα αποστολής bit ισοτιμίας (parity bit), μετά την αποστολή των bit δεδομένων, που χρησιμοποιείται για

τον έλεγχο σφαλμάτων κατά την μετάδοση. Το bit ισοτιμίας δεν χρησιμοποιείται στη δική μας εφαρμογή.

Ένα ακόμα μέγεθος το οποίο περιλαμβάνει η ασύγχρονη συριακή αποστολή δεδομένων είναι ο ρυθμός μετάδοσης (Baud Rate). Ο ρυθμός αυτός ισούται με τον αντίστροφο του χρόνου διάρκειας ενός bit. Για παράδειγμα αν ο χρόνος διάρκειας ενός bit είναι 1042 ms ο ρυθμός μετάδοσης δεδομένων είναι 1/ 1042 ms ή 9600 Bd. Για την επιλογή του ρυθμού μετάδοσης δεδομένων στον μικροελεγκτή θα πρέπει να ληφθεί υπόψη η συχνότητα χρονισμού και να αρχικοποιηθούν κατάλληλα οι καταχωρητές που σχετίζονται με την συριακή μετάδοση. Η διαδικασία αυτή περιγράφεται αναλυτικά στο κεφάλαιο 4 στην παράγραφο που αφορά τον πομπό / δέκτη σειριακής μετάδοσης δεδομένων ενώ η υλοποίηση της γίνεται στο τμήμα κώδικα που περιλαμβάνεται στο αρχείο “sci.c “.

2.2.5 Παράθεση Κυκλώματος / Λίστα Υλικών

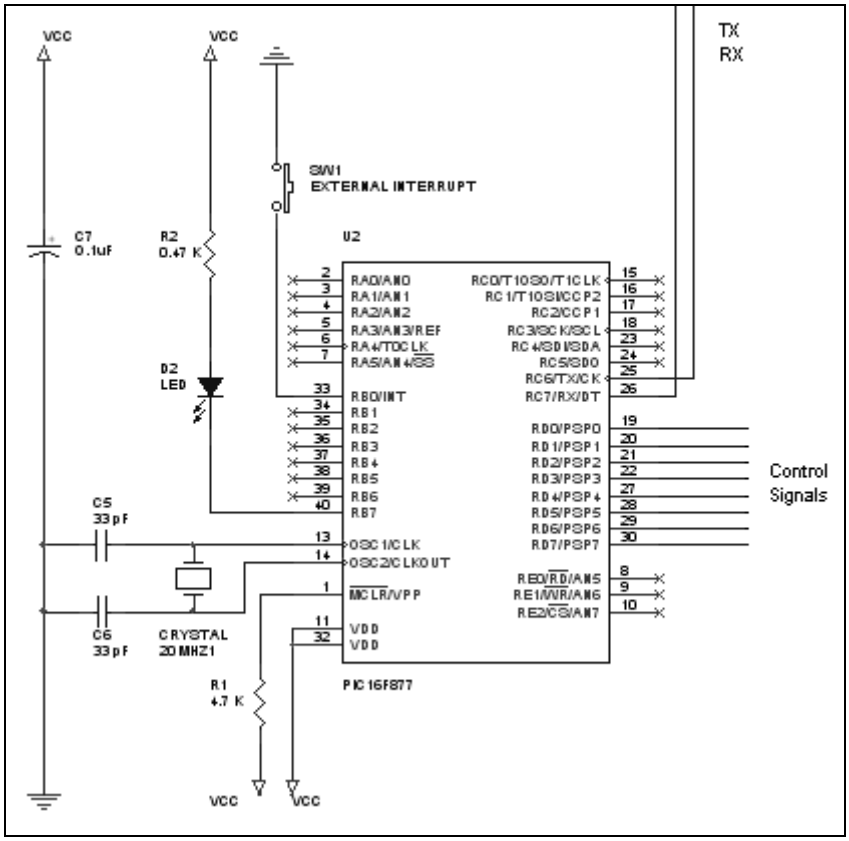
Η σύνδεση των επιμέρους τμημάτων του κυκλώματος καθώς και το σύνολο ηλεκτρονικών εξαρτημάτων που το αποτελούν παρουσιάζονται στο διάγραμμα του κυκλώματος που ακολουθεί.



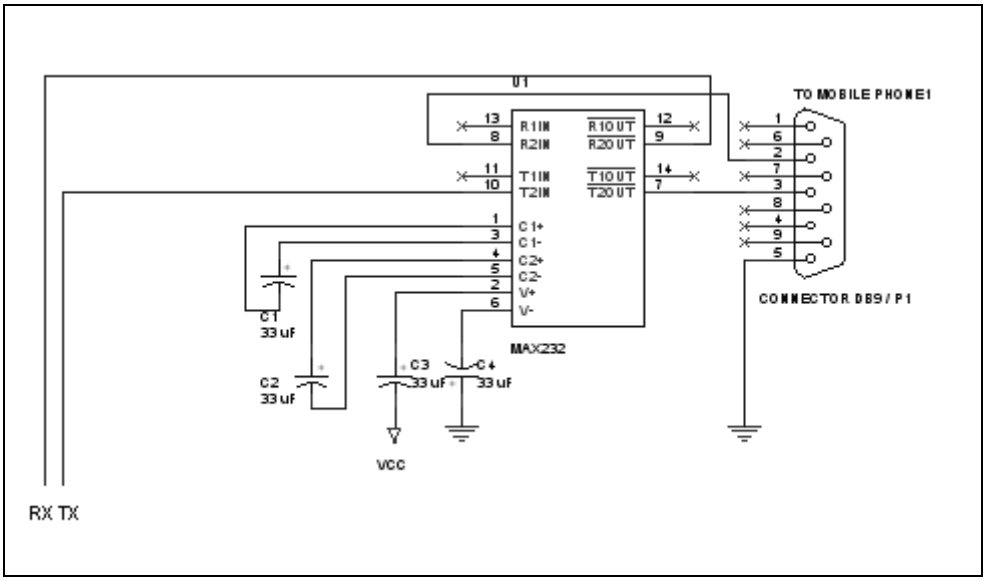
Σχήμα 2.7 *Κύκλωμα εφαρμογής*

Όπως μπορούμε να δούμε και από το διάγραμμα, το κύκλωμα της εφαρμογής αποτελούν τρία επιμέρους τμήματα. Τα τμήματα αυτά είναι:

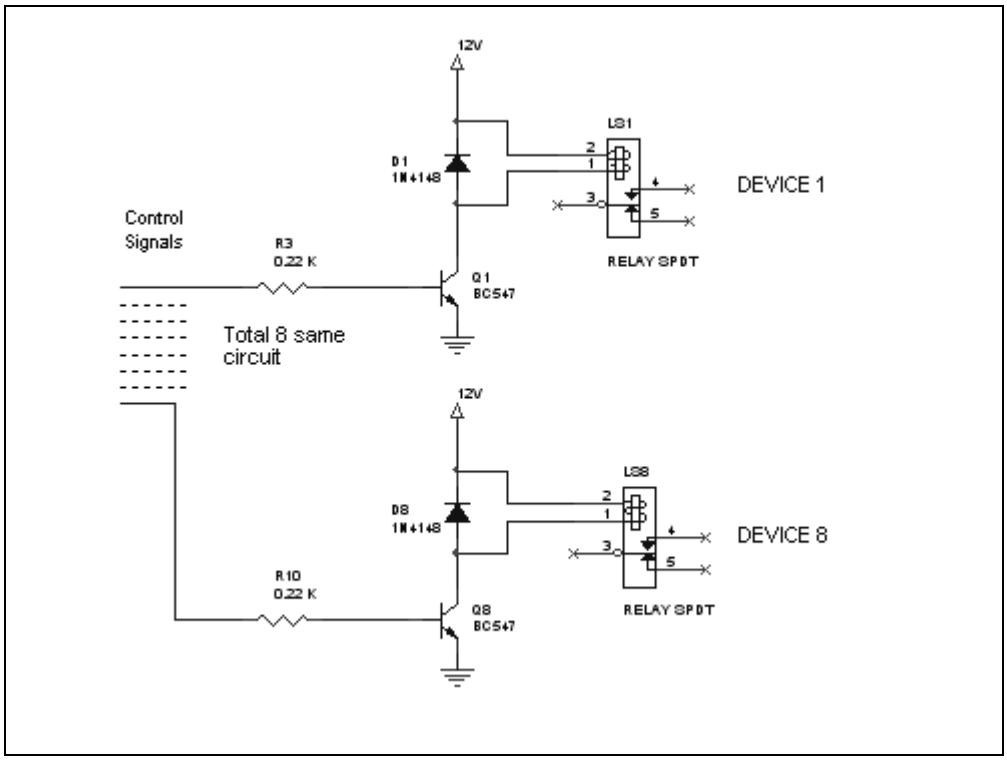
- Ο μικροελεγκτής, με την τροφοδοσία και το κύκλωμα χρονισμού.
- Το ολοκληρωμένο MAX232, για την σύνδεση με το κινητό τηλέφωνο.
- Και το τμήμα οδήγησης των συσκευών



Σχήμα 2.8 *Το τμήμα του μικροελεγκτή στο κύκλωμα*



***Σχήμα** 2.9 RS 232 INTERFACE*



***Σχήμα 2.10** Το τμήμα συσκευών στο κύκλωμα*

Α/Α	ΠΟΣΟΤΗΤΑ	ΤΥΠΟΣ	ΑΝΑΦΟΡΑ	ΜΕΓΕΘΟΣ
1	4	CP	C1, C2, C3, C4	33μF 63V
2	2	C	C5, C6	33pF
3	1	CP	C7	47μF 50V
4	1	R	R1	4,7KΩ
5	1	R	R2	0,47KΩ
6	8	R	R3, R4, R5, R6 R7, R8, R9, R10	0.22KΩ
7	8	T	Q1, Q2, Q3, Q4 Q5, Q6, Q7, Q8	BC547
8	8	DIODE	D1,D2, D3, D4 D5, D6, D7, D8	1N4148
9	8	LS	LS1, LS2, LS3, LS4 LS5, LS6, LS7, LS8	RELAY SPDT
10	1	PHOTODIODE	D9	LED
11	1	IC	U1	PIC16F877
12	1	IC	U2	MAX232
13	1	XTAL	Y1	20MHZ
14	1	CONNECTOR	P1	CONNECTOR DB9
15	1	SWITCH	SW1	SWITCH

Πίνακας 2.2 *Λίστα εξαρτημάτων κυκλώματος*

2.3 Χειρισμός Συστήματος

Κατά την σχεδίαση και ανάπτυξη του συστήματος ελέγχου δόθηκε ιδιαίτερη έμφαση στο τμήμα που αφορά την χρήση και τον χειρισμό του συστήματος. Βασικός στόχος από την αρχή ήταν η αποφυγή δύσκολων και εξειδικευμένων χειρισμών και η όσο το δυνατό απλοποίηση και συντόμευση της διαδικασίας. Το γεγονός αυτό επηρέασε τόσο το υλικό μέρος της εφαρμογής, με την επιλογή του καταλληλότερου τρόπου επικοινωνίας του χρήστη με το σύστημα, όσο και στο λογισμικό κομμάτι του οποίου πολυπλοκότητα και η έκταση αυξήθηκαν ως αντιστάθμισμα της ευκολίας και της απλότητας που θα έπρεπε να δοθεί στον χρήστη. Σε κάθε περίπτωση και για κάθε

επιλογή που θα γίνονταν στο υλικό ή λογισμικό μέρος αυτή θα έπρεπε να γίνει σύμφωνα με τα παρακάτω κριτήρια:

- Συντομία και απλότητα κατά την διαδικασία χειρισμού του συστήματος
- Μικρή πιθανότητα λάθους κατά την χρήση
- Δυνατότητα απάντησης – επιβεβαίωσης από το σύστημα
- Δυνατότητα πρόσβασης στο σύστημα από πολλούς χρήστες
- Ασφάλεια συστήματος. Εκχώρηση πρόσβασης μόνο σε συγκεκριμένους χρήστες

Με βάση όσα αναφέρθηκαν και εκτιμώντας όλες τις δυνατές επιλογές αποφασίστηκε ότι καλύτερος τρόπος χειρισμού και επικοινωνίας του συστήματος ελέγχου ήταν μέσο της αποστολής σύντομων γραπτών μηνυμάτων SMS από τα κινητά τηλέφωνα των χρηστών. Ο τρόπος αυτός ικανοποιούσε αρκετά από τα κριτήρια που τέθηκαν δίνοντας παράλληλα μεγάλη ευελιξία και ευκολία στους χρήστες. Την επιλογή αυτή ωστόσο θα έπρεπε να ακολουθήσει η οριοθέτηση εντολών και κανόνων σύνταξης του γραπτού μηνύματος με τέτοιο τρόπο ώστε να ικανοποιούνται και στην συνέχεια τα κριτήρια και οι απαιτήσεις που τέθηκαν εξ αρχής.

2.4 Προϋποθέσεις Χρήσης Συστήματος

Για την καλύτερη δυνατή εξυπηρέτηση των χρηστών αναπτύχθηκε ένα στοιχειώδες ρεπερτόριο εντολών μέσο του οποίου θα γίνεται χρήση του συστήματος. Για να μπορέσει κάποιος χρήστης να χρησιμοποιήσει το σύστημα ελέγχου θα πρέπει να γνωρίζει:

1. Τον κωδικό χρήση του συστήματος.

   Ο κωδικός αυτός είναι ένας τριψήφιος αριθμός ο οποίος έχει καθοριστεί κατά την ανάπτυξη του λογισμικού και αποτελεί ένα μέτρο ασφάλειας ώστε να επιτρέπεται η χρήση του συστήματος μόνο σε ορισμένους χρήστες. Αν κατά την διαδικασία αποστολή γραπτού μηνύματος ο κωδικός πρόσβασης δεν είναι έγκυρος το σύστημα δεν θα εκτελέσει την εντολή ενώ θα αφήσει ανεπηρέαστη την τρέχουσα κατάσταση λειτουργίας των συσκευών.

2. Τις εντολές χειρισμού συστήματος καθώς και τους κανόνες σύνταξης που τις διέπουν.

   Η αποστολή μια εντολής μέσο γραπτού μηνύματος προϋποθέτει την ορθή σύνταξη της βάση των κανόνων ώστε να είναι δυνατή η αποκωδικοποίηση της από το σύστημα. Αποστολή μηνύματος που περιέχει λάθος εντολές ή λάθη κατά την σύνταξη του αφήνει την κατάσταση των συσκευών ανέπαφη ενώ προκαλεί την αποστολή, από το σύστημα προς των χρήστη, γραπτού μηνύματος με την λέξη "Error", για να τον ενημέρωση ότι υπήρξε λάθος κατά διαδικασία ελέγχου των συσκευών. Αν σε αντίθετη περίπτωση όλα γίνουν με βάση τους κανόνες σύνταξης το σύστημα θα αλλάξει την κατάσταση των συσκευών βάση των εντολών του χρήστη ενώ θα αποστείλει σε αυτόν μήνυμα με την λέξη "OK" για να τον ενημερώσει ότι η απόπειρα χειρισμού των συσκευών ήταν επιτυχής.

3. Το αριθμό του κινητού τηλεφώνου βάσης, απαραίτητο στοιχείο για την αποστολή του γραπτού μηνύματος.

2.5 Παράθεση Εντολών Χειρισμού

Στην συνέχεια της αναφοράς παραθέτουμε το ρεπερτόριο των εντολών που χρησιμοποιούνται για τον χειρισμό του συστήματος. Οι εντολές που ορίζονται είναι :

- Η εντολή ενεργοποίησης ON
- Η εντολή απενεργοποίησης OFF
- Η εντολή ανάγνωσης της τρέχουσας κατάστασης του συστήματος STATUS

Επίσης εκτός από τον ορισμό, την περιγραφή και την αναφορά των ορισμάτων τους δίνονται επιπλέον και ορισμένα παραδείγματα για την καλύτερη και ευκολότερη κατανόηση τους.

2.5.1 Εντολή Ενεργοποίησης

ON *Ενεργοποίηση Συσκευής*

Περιγραφή: Θέτει σε λειτουργία τις συσκευές που έχει ως όρισμα. Αν η συσκευή είναι απενεργοποιημένη τίθεται σε λειτουργία. Αν είναι ήδη σε λειτουργία παραμένει στην κατάσταση αυτή.

Σύνταξη: **ON** < Όρισμα >

Παράμετροι:	< Όρισμα >	**1-8**	Κάθε ένας από τους αριθμούς αυτούς αντιστοιχεί σε μια από τις οκτώ θέσεις ελέγχου του συστήματος στις οποίες συνδέονται οι αντίστοιχες συσκευές.
		ALL	Όρισμα που χρησιμοποιείται όταν θέλουμε να συμπεριλάβουμε το σύνολο των συσκευών.
		FOS **FUN** **THERMO**	Το όνομα κάποιας συσκευής. Ο ορισμός και η αντιστοίχηση με την θέση ελέγχου γίνονται στο πρόγραμμα της εφαρμογής.
Παράδειγμα:	ON 2 3 FOS 8		Ενεργοποιεί τις συσκευές που βρίσκονται στις θέσεις ελέγχου δύο τρία και οκτώ καθώς και την συσκευή που αντιστοιχεί στην λέξη FOS.
	ON ALL		Ενεργοποιεί όλες τις συσκευές που συνδεμένες στο σύστημα ελέγχου.
Επιστρέφει:	OK / ERROR		Αποστολή γραπτού μηνύματος για να επιβεβαιωθεί ότι η εντολή εκτελέστηκε κανονικά ή ότι παρουσιάστηκε πρόβλημα.

2.5.2 Εντολή Απενεργοποίησης

OFF *Απενεργοποίηση Συσκευής*

Περιγραφή: Απενεργοποιεί τις συσκευές που έχει ως όρισμα. Αν η συσκευή βρίσκεται σε λειτουργία απενεργοποιείτε. Στην περίπτωση που είναι ήδη απενεργοποιημένη παραμένει στην κατάσταση αυτή.

Σύνταξη: **OFF** < Όρισμα >

Παράμετροι:	< Όρισμα >	**1-8**	Κάθε ένας από τους αριθμούς αυτούς αντιστοιχεί σε μια από τις οκτώ θέσεις ελέγχου του συστήματος στις οποίες συνδέονται οι αντίστοιχες συσκευές.
		ALL	Λέξη που χρησιμοποιείται όταν θέλουμε να συμπεριλάβουμε το σύνολο των συσκευών.
		FOS **FUN** **THERMO**	Το όνομα κάποιας συσκευής. Ο ορισμός και η αντιστοίχηση με την θέση ελέγχου γίνεται στο πρόγραμμα της εφαρμογής.
Παράδειγμα:	OFF 6 5 THERMO		Απενεργοποιεί τις συσκευές που βρίσκονται στις θέσεις ελέγχου πέντε και έξι καθώς και την συσκευή που αντιστοιχεί στην λέξη THERMO.
	OFF ALL		Απενεργοποιεί όλες τις συσκευές που είναι συνδεμένες στο σύστημα ελέγχου.
Επιστρέφει:	OK / ERROR		Αποστολή γραπτού μηνύματος για να επιβεβαιωθεί ότι η εντολή εκτελέστηκε κανονικά ή ότι παρουσιάστηκε πρόβλημα.

2.5.3 Εντολή Ανάγνωσης Τρέχουσας Κατάστασης

STATUS *Εντολή ανάγνωσης της τρέχουσα κατάστασης του συστήματος*

Περιγραφή: Αποστέλλει μέσο σύντομού γραπτού μηνύματος την τρέχουσα κατάσταση του συστήματος ελέγχου, αναγράφοντας ποίες από τις συσκευές βρίσκονται σε λειτουργία. Αν καμία από τις συσκευές δεν λειτουργεί αποστέλλεται ανάλογο μήνυμα.

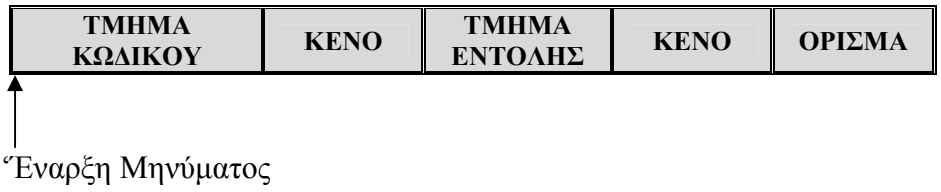
Σύνταξη:	**STATUS**	Για την σύνταξη της εντολής αρκεί μόνο το όνομα της.
Παράμετροι:	<ΚΑΜΙΑ>	Δεν δέχεται καμία παράμετρο.
Παράδειγμα:	**STATUS**	
Συσκευές σε λειτουργία :	1 2 5 6 8	Οι συσκευές που βρίσκονται εκείνη την στιγμή ενεργοποιημένες.
Επιστρέφει μήνυμα:	"Active 1 2 5 6 8"	
Συσκευές σε λειτουργία :	<ΚΑΜΙΑ>	Καμιά από τις συσκευές του συστήματος δεν βρίσκεται εκείνη την στιγμή σε λειτουργία.
Επιστρέφει μήνυμα:	"All devices are closed"	

2.6 Μορφή Αποστολής Μηνύματος / Παράθεση Κανόνων Σύνταξης

Η αποστολή μιας εντολής μέσο γραπτού μηνύματος για να γίνει αποδεκτή από το σύστημα ελέγχου θα πρέπει να έχει καθορισμένη μορφή και να ικανοποιεί παράλληλα ορισμένους κανόνες σύνταξης. Στην συνέχεια αυτής της παραγράφου παρουσιάζονται οι δύο αυτές προϋποθέσεις με βάση τις οποίες θα πρέπει να γίνεται η σύνταξη των μηνυμάτων από τους χρήστες.

2.6.1 Μορφή Αποστολής Μηνύματος

Η δομή ενός μηνύματος εντολής για τον έλεγχο του συστήματος θα πρέπει να έχει κάθε φορά την μορφή που παρουσιάζεται στο σχήμα 2.11 .



Σχήμα 2.11 *Δομή μηνύματος*

Όπως παρατηρούμε ένα μήνυμα εντολής ξεκινά πάντα με το τμήμα κωδικού. Ο κωδικός είναι ένα τριψήφιος αριθμός που καθορίζεται κατά την διαδικασία προγραμματισμού και για την συγκεκριμένη εφαρμογή επιλέχτηκε το νούμερο «123». Το τμήμα του κωδικού ακολουθεί το τμήμα εντολής με την μεσολάβηση απαραίτητα ενός και μόνο χαρακτήρα κενού ο οποίος θα διαχωρίζει τα δύο τμήματα. Στο τμήμα εντολής περιέχετε μια από τις τρεις εντολές ελέγχου του συστήματος (ON, OFF, STATUS). Το τμήμα εντολής ακολουθεί επίσης ένας και μόνο χαρακτήρας κενού για να το διαχωρίσει από το τμήμα ορίσματος που πιθανός να ακολουθεί στην συνέχεια (εντολές ON, OFF). Στην περίπτωση που υπάρχουν περισσότερα του ενός ορίσματα αυτά θα πρέπει επίσης να διαχωρίζονται μεταξύ με ένα και μόνο χαρακτήρα κενού.

2.6.2 Κανόνες Σύνταξης

Εκτός από την έγκυρη δομή που θα πρέπει να έχει ένα μήνυμα θα πρέπει παράλληλα να ικανοποιεί και ορισμένους κανόνες σύνταξης. Οι κανόνες αυτοί παρουσιάζονται στη συνέχεια καθώς επίσης και ορισμένα παραδείγματα για την καλύτερη κατανόηση τους.

Αν κατά την αποκωδικοποίηση ενός μηνύματος δεν ικανοποιείται κάποιος από τους κανόνες σύνταξης τότε το σύστημα πηγαίνει σε κατάσταση λάθους η οποία θα αφήσει ανεπηρέαστη την τρέχουσα κατάσταση των συσκευών ενώ θα προκαλέσει παράλληλα και την αποστολή γραπτού μηνύματος προς το χρήστη με την λέξη "ERROR" για να τον ενημερώσει ότι η απόπειρα ελέγχου του συστήματος που επιχείρησε περιείχε κάποιο σφάλμα.

- Αρχικά θα πρέπει μεταξύ των διαφορετικών τμημάτων του μηνύματος εντολής αλλά και των ορισμάτων που αυτό περιλαμβάνει να μεσολαβεί απαραίτητα ένας χαρακτήρας κενού. Στην περίπτωση που κάποιο τμήμα ή όρισμα δεν διαχωρίζεται από το άλλο ή στην περίπτωση που αφεθούν περισσότεροι του ενός χαρακτήρες κενού τότε το σύστημα αναγνωρίζει το μήνυμα ως εσφαλμένο και εκτελεί τις ανάλογες διαδικασίες.

Παράδειγμα	123ON 2 3 4	Λάθος σύνταξη διότι το τμήμα του κωδικού ενώνεται με αυτό της εντολής.
	123 ON 1 2 34 5	Λάθος σύνταξη διότι τα ορίσματα θα πρέπει να ξεχωρίζουν μεταξύ τους με ένα κενό χαραχτήρα.
	123 OFF 7 6__5 4	Λάθος σύνταξη διότι μεταξύ των ορισμάτων 6 και 5 υπάρχουν περισσότεροι του ενός χαρακτήρες κενού.

- Η σειρά των ορισμάτων μιας εντολής δεν έχει σημασία.

Παράδειγμα 123 ON 1 2 3 4 5 6 Ορθή σύνταξη.

123 ON 6 5 4 3 2 1 Ορθή σύνταξη εντολής ισοδύναμη με την προηγούμενη.

❖ Αν κάποια από τις εντολές ενεργοποίησης (ON) ή απενεργοποίησης (OFF) γραφεί μόνη της χωρίς όρισμα προκαλείται σφάλμα.

Παράδειγμα 123 OFF 1 2 ON Λάθος σύνταξη διότι η εντολή ON δεν έχει όρισμα.

❖ Κατά την αποστολή ενός μηνύματος ο χρήστης μπορεί να χρησιμοποιήσει μεμονωμένα ή ταυτόχρονα τις εντολές ενεργοποίησης (ON) ή απενεργοποίησης (OFF) χωρίς να έχει σημασία η σειρά η χρησιμοποιήσεις τους .

Παράδειγμα 123 ON 2 3 4 Ορθή σύνταξη.

123 OFF ALL Ορθή σύνταξη.

123 ON FOS OFF 8 Ορθή σύνταξη ανεξάρτητος της σειρά των εντολών.

123 OFF 8 ON 6 7 Ορθή σύνταξη ανεξάρτητος της σειρά των εντολών.

❖ Η εντολή STATUS θα πρέπει πάντα να αποστέλλεται μόνη της.

Παράδειγμα 123 STATUS Ορθή σύνταξη της εντολή STATUS. Απαιτείται μόνο ο κωδικός και η εντολή.

123_STATUS Λάθος σύνταξη διότι μεταξύ κωδικού και εντολής υπάρχουν περισσότεροι του ενός χαρακτήρες κενού.

123 ON 6 STATUS Λάθος σύνταξη διότι η εντολή STATUS περιέχεται μαζί με άλλη εντολή.

Ολοκληρώνοντας την παράθεση των κανόνων σύνταξης και της δομής που απαραίτητα θα πρέπει να έχει ένα γραπτό μήνυμα εντολής, για την λειτουργία του συστήματος απαιτείται μόνο η σύνδεση του με τις συσκευές και την πηγή τροφοδοσίας. Αφού το σύστημα τεθεί σε λειτουργία ο μικροελεγκτής ξεκίνα την εκτέλεση του κώδικα με το οποίο είναι προγραμματισμένος αρχικοποιώντας τις συσκευές σε κατάσταση μη λειτουργίας και μπαίνοντας στην συνέχεια σε έναν βρόγχο ελέγχου για λήψη ενός νέου μηνύματος από το κινητό τηλέφωνο βάσης. Αν τα μηνύματα που λάβει ο μικροελεγκτής είναι σωστά εκτελεί τις κατάλληλες εντολές στέλνει μήνυμα επιβεβαίωσης και αναμένει την αποστολή του επόμενου μηνύματος. Αν το μήνυμα που λάβει είναι περιέχει σφάλμα τότε αφήνει τις συσκευές ως έχουν στέλνει μήνυμα σφάλματος προς τον χρήστη και επιστρέφει ξανά στην κατάσταση έλεγχου για την άφιξη ενός νέου μηνύματος εντολής. Σε περίπτωση που η λειτουργία του συστήματος διακοπή η τρέχουσα κατάσταση των συσκευών χάνεται και αρχικοποιείτε ξανά όταν αυτό επαναλειτουργήσει.

ΚΕΦΑΛΑΙΟ 3

ΣΥΣΚΕΥΗ ΚΙΝΗΤΟΥ ΤΗΛΕΦΩΝΟΥ

3.1 Κριτήρια επιλογής συσκευής

Η επιλογή της συσκευής κινητού τηλεφώνου για την υλοποίηση της εφαρμογής έγινε βάση συγκεκριμένων κριτηρίων τα οποία θα έπρεπε να πληρούνται προκειμένου να επιτευχθεί ο έλεγχος του συστήματος. Πρώτο και βασικό κριτήριο αποτελεί η δυνατότητα έμμεσης πρόσβασης στις επιμέρους λειτουργίες του κινητού τηλεφώνου και ειδικότερα σε εκείνες που αφορούν την διαχείριση των σύντομων γραπτών μηνυμάτων (αποθήκευση, λήψη, αποστολή). Δεύτερο κριτήριο αποτελούν οι δυνατότητες και ο τρόπος διασύνδεσης του κινητού τηλεφώνου με περιφερειακές συσκευές. Με βάση τον τρόπο επικοινωνίας του μικροελεγκτή ο οποίος υποστηρίζει τη σειριακή μετάδοση δεδομένων μέσο της περιφερειακής μονάδας *USART*, η επιλογή του τηλεφώνου θα έπρεπε να είναι τέτοια ώστε να υποστηρίζεται ο παραπάνω τρόπος για να είναι εφικτή μια σταθερή και αξιόπιστη επικοινωνία μεταξύ των δύο συσκευών. Τέλος θα πρέπει αναφερθούν και οι περιορισμοί που προήρθαν κατά την εύρεση των κατάλληλων περιφερειακών σύνδεσης των κινητών τηλεφώνων (όπως καλώδια και βύσματα σύνδεσης) είτε γιατί οι εταιρίες κατασκευής τους δε τα υποστήριζαν πλέον, είτε γιατί αυτά που είδη κυκλοφορούσαν (συνήθως από τρίτους κατασκευαστές, αμφίβολης ποιότητας και προέλευσης) δεν εξασφάλιζαν αξιόπιστη και σωστή λειτουργία.

Σύμφωνα με αυτά που προαναφέρθηκαν η συσκευή που τελικά επιλεγεί για την ανάπτυξη της εφαρμογής είναι το κινητό τηλέφωνο *T28s* της εταιρίας *Sony Ericsson Mobile Communications* [6]. Η συσκευή αυτή ικανοποιεί τα παραπάνω κριτήρια αφού ο χρήστης μπορεί να έχει πρόσβαση στις λειτουργίες τις μέσο ενός ρεπερτορίου εντολών που υποστηρίζει (AT commands), ενώ υπάρχει δυνατότητα σύνδεσης με ηλεκτρονικό υπολογιστή μέσο συριακού καλωδίου.

3.2 Αναφορά AT εντολών

Οι συσκευές κινητής τηλεφωνίας GSM, κατά αναλογία με τις συσκευές Modem, μπορούν να δεχτούν ένα σύνολο εντολών με στις οποίες ο χρήστης είναι σε θέση να εκτελεί επιμέρους και μεμονωμένες λειτουργίες. Για παράδειγμα θα μπορούσε κάποιος να διαβάσει ένα γραπτό μήνυμα που βρίσκεται αποθηκευμένο σε κάποια θέση μνήμης του τηλεφώνου ή να στήλη κάποιο αφού προηγουμένως το συντάξει και ορίσει τον παραλήπτη του. Πέρα όμως από τις διαδικασίες διαχείρισης των γραπτών μηνυμάτων υπάρχουν και εντολές με τις οποίες μπορούν να πραγματοποιηθούν και άλλου είδους λειτουργίες όπως π.χ. η διαχείριση του τηλεφωνικού καταλόγου, η αυξομείωση της ένταση του ήχου σε μια εισερχόμενη κλίση, η δυνατότητα να μάθουμε πληροφορίες για το δίκτυο κινητής τηλεφωνίας που χρησιμοποιούμε ή και ακόμα να μετρήσουμε την ένταση του λαμβανόμενου σήματος από την συσκευή. Όλες αυτές οι επιμέρους διεργασίες που σε συνδυασμό μας παρέχουν το σύνολο των υπηρεσιών που κάθε χρήστης κινητού τηλεφώνου γνωρίζει και χρησιμοποιεί μπορούν να γίνου μέσο των AT εντολών (AT commands).

Κατά την διάρκεια ανάπτυξης της εφαρμογής με σκοπώ την επίτευξη της επικοινωνίας μεταξύ του κινητού τηλεφώνου και του μικροελεγκτή κάναμε χρήση των AT εντολών. Οι εντολές αυτές στέλνονται από τον μικροελεγκτή στο κινητό τηλέφωνο βάσης ώστε να εκτελούνται συγκεκριμένες λειτουργίες. Η σύνταξη των AT εντολών γίνεται γράφοντας πρώτα το πρόθεμα AT και προσθέτοντας στην συνέχεια το όνομα κάθε εντολής. Μεταξύ του προθέματος και ονόματος περιλαμβάνεται πάντα το σύμβολο της πρόσθεσης (+). Στην συνεχεία ακολουθεί το σύμβολο της ισότητας (=) καθώς και τα ορίσματα που τυχόν δέχεται κάποια εντολή. Το τέλος της εντολής σηματοδοτείται από το χαρακτήρα [CR] (Carriage Return) που σημαίνει ταυτόχρονα και την έναρξη εκτέλεσης της από το κινητό τηλέφωνο.

Στην συνέχεια παραθέτουμε την μορφή και την λειτουργία κάθε εντολής που χρησιμοποιήσαμε σε αυτή την εφαρμογή όπως αυτή αναφαίρετε στο εγχειρίδιο εντολών της εταιρίας *Sony Ericsson Mobile Communications* [4] για την συγκεκριμένη συσκευή τηλεφώνου *Ericsson T28s*.

3.2.1 AT+CPMS

Η πρώτη εντολή που χρησιμοποιείται στην εφαρμογή είναι η εντολή AT+CPMS μέσο τη οποίας δηλώνεται η θέση μνήμης που θα αποθηκεύεται το λαμβανόμενο γραπτό μήνυμα στο κινητό τηλέφωνο βάσης. Στην καθορισμένη λειτουργία του το κινητό τηλέφωνο αποθηκεύει και στην συνέχεια ανακτά κάθε γραπτό μήνυμα από την μνήμη που βρίσκεται στην κάρτα SIM. Για να αλλάξουμε την θέση αποθήκευσης των μηνυμάτων και να ορίσουμε νέα αυτήν του κινητού τηλεφώνου χρησιμοποιούμε την εντολή CPMS.

AT+CPMS Επιλογή μνήμης για την αποθήκευση γραπτών μηνυμάτων SMS

Περιγραφή: Καθορισμός των θέσεων μνήμης <mem1>, <mem2>, <mem3> που θα χρησιμοποιηθούν για διάβασμα, το γράψιμο και την αποθήκευση των γραπτών μηνυμάτων.

Σύνταξη: **+CPMS** =<mem1>,[<mem2>],[<mem3>]

Παράμετροι:		
	<mem1>	Μνήμη από την οποία τα μηνύματα θα διαβάζονται και θα σβήνονται. **"ME"** Μνήμη αποθήκευσης του τηλεφώνου. **"SM"** Μνήμη αποθήκευσης της κάρτας SIM.
	<mem2>	Μνήμη από την οποία θα γίνονται οι λειτουργίες γραψίματος και αποστολής. **"ME"** Μνήμη αποθήκευσης του τηλεφώνου. **"SM"** Μνήμη αποθήκευσης της κάρτας SIM. Προκαθορισμένη θέση η **"SM".**
	<mem3>	Μνήμη στην οποία θα αποθηκεύονται τα λαμβανόμενα SMS . **"ME"** Μνήμη αποθήκευσης του τηλεφώνου.

Επιστρέφει:	**+CPMS:**<used1>,<total1>,<used2>,<total2>,<used3>,<total3>	
όπου	<used1>,<used2>,<used3>	Συνολικός αριθμός μηνυμάτων που περιέχονται στις μνήμες <mem1>, <mem2> και <mem3> αντίστοιχα.
	<total1>,<total2>,<total3>	Συνολικός αριθμός θέσεων μηνυμάτων στις μνήμες <mem1>, <mem2> και <mem3> αντίστοιχα

Παράδειγμα: AT+CPMS=”SM”,”SM”

Επιστρέφει: +CPMS: 3,20,3,20

OK

3.2.2 AT+CMGR

Η εντολή AT+CMGR χρησιμοποιείται για το διάβασμα ενός μηνύματος από την καθορισμένη θέση αποθήκευσης του. Την εντολή αυτή στέλνει ο μικροελεγκτής στο κινητό τηλέφωνο μετά την λήψη ενός γραπτού μηνύματος. Το κινητό αποκρίνεται αποστέλλοντας το μήνυμα στην PDU μορφή του.

AT+CMGR Εντολή ανάγνωσης γραπτού μηνύματος

Περιγραφή: Επιστρέφει το μήνυμα το οποίο βρίσκεται στη θέση <index> μέσα στην προκαθορισμένη μνήμη αποθήκευσης γραπτών μηνυμάτων <mem1>. Μαζί με τα δεδομένα που επιστρέφονται σε <pdu> μορφή καθορίζεται και η τρέχουσα κατάσταση του μηνύματος.

Σύνταξη:	**+CMGR =**<index>		
Παράμετροι:	<index>		Η θέση ενός γραπτού μηνύματος μέσα στην προκαθορισμένη μνήμη αποθήκευσης γραπτών μηνυμάτων.
Επιστρέφει:	<status>	**0**	Γραπτό μήνυμα που έχει ληφθεί και δεν έχει διαβαστεί.
		1	Γραπτό μήνυμα που έχει ληφθεί και έχει διαβαστεί.
		2	Αποθηκευμένο μήνυμα που δεν έχει σταλεί.
		3	Αποθηκευμένο μήνυμα που έχει σταλεί.
		16	Προσχέδιο μηνύματος.
	<length>		Μήκος δεδομένων του μηνύματος
	<pdu>		Τα δεδομένα του μηνύματος σε δεκαεξαδική μορφή κωδικοποίησης.
Παράδειγμα:	AT+CMGR=2		Εντολή ανάγνωσης γραπτού μηνύματος που βρίσκεται στην δεύτερη θέση της μνήμης <mem1>.
Επιστρέφει:	+CMGR: 0, ,68 <64 byte pdu>		
	OK		

3.2.3 AT+CMGD

Η εντολή CMGD χρησιμοποιείται για τη διαγραφή ενός μηνύματος από την επιλεγόμενη θέση μνήμης. Η εντολή εκτελείται μετά την αποκωδικοποίηση και επεξεργασία ενός μηνύματος ώστε να ελευθερωθεί η πρώτη θέση μνήμης στην οποία θα γραφτεί το επόμενο γραπτό μήνυμα που θα αποσταλεί.

AT+CMGD Εντολή διαγραφής γραπτού μηνύματος

Περιγραφή: Διαγράφει το γραπτό μήνυμα που βρίσκεται στην θέση <index> στην μνήμη αποθήκευσης μηνυμάτων <mem1>.

Σύνταξη: **+CMGD =**<index>

Παράμετροι:	<index>	Η θέση ενός γραπτού μηνύματος μέσα στην προκαθορισμένη μνήμη αποθήκευσης γραπτών μηνυμάτων.
	AT+CMGD=2	Διαγραφή του μηνύματος από την θέση <index>2 μέσα στη μνήμη αποθήκευσης μηνυμάτων.
Παράδειγμα:	AT+CMGD=2	Διαγραφή του μηνύματος από την θέση <index>2 μέσα στη μνήμη αποθήκευσης μηνυμάτων.

Επιστρέφει: OK

3.2.4 AT+CMGS

Η εντολή CMGS χρησιμοποιείται για την αποστολή ενός σύντομου μηνύματος από το κινητό τηλέφωνο προ το δίκτυο. Απαραίτητη προϋπόθεση είναι να έχουμε προηγουμένως δημιουργήσει την ακολουθία του μηνύματος στην PDU μορφή με το κωδικοποιημένο μήνυμα και τις απαραίτητες πληροφορίες που απαιτούνται για την μετάδοση του.

AT+CMGS Εντολή μετάδοσης ενός σύντομου μηνύματος SMS

Περιγραφή: Στέλνει ένα σύντομο μήνυμα στο δίκτυο κινητής τηλεφωνίας. Στην περίπτωση επιτυχούς αποστολής επιστρέφεται ο αριθμός αναφοράς του μηνύματος. Η αποστολή μπορεί να ματαιωθεί με την αποστολή του χαρακτήρα *ESC*.

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

Παράμετροι:	<length>	Τιμή η οποία κατά την αποστολή ενός γραπτού μηνύματος σε PDU μορφή (+CMGF=0), δείχνει το πλήθος των octets της ακολουθίας.
Επιστρέφει:	<mr>	Αριθμός αναφοράς μηνύματος

Παράδειγμα: AT+CMGS=35<CR><35 byte pdu><CTRL-Z>

+CMGS: 13

OK

3.2.5 AT+CNMI

Με την εντολή CNMI επιλέγουμε το τρόπο με τον οποίο το κινητό τηλέφωνο θα ενημερώνει το τερματικό με το οποίο συνδέεται (TE, Terminal Equipment) για την λήψη ενός νέου μηνύματος. Στην συνέχεια το τερματικό δηλ. ο μικροελεγκτής αναλαμβάνει να στείλει τις κατάλληλες εντολές για το διάβασμα και την περαιτέρω επεξεργασία και αποκωδικοποίηση του μηνύματος.

AT+CNMI Ειδοποίηση νέου μηνύματος στο TE

Περιγραφή: Επιλέγεται η διαδικασία μέσο της οποίας θα υποδεικνύεται η άφιξη ενός νέου μηνύματος στο TE.

Σύνταξη: **+CNMI**= [<mode>[,<mt>[,<bm>[,<ds>[,<bfr>]]]]]

Παράμετροι:

<mode>	**3**	Μνήμη από την οποία τα μηνύματα θα διαβάζονται και θα σβήνονται **"ME"** Μνήμη αποθήκευσης του τηλεφώνου **"SM"** Μνήμη αποθήκευσης της κάρτας SIM
<mt>	**0**	Δεν αποστέλλεται ειδοποίηση προς το τερματικό σύνδεσης **TE** για την για την αποστολή ενός νέου SMS Εκ των προτέρων επιλογή =**0**
	1	Εάν ένα νέο SMS ληφθεί από το κινητό τηλέφωνο αποθηκεύεται στην επιλεγμένη μνήμη **ME .** Ειδοποίηση για την θέση αποθήκευσης αποστέλλεται στο τερματικό σύνδεσης **TE** με την μορφή +CMTI:<mem>,<index>
	3	Τα νέα SMS δρομολογούνται απευθείας στο **TE** με την μορφή

			+CMT: <length><CR><LF><pdu> Εκ των προτέρων επιλογή =**0**
	<bm>	**0**	Αποθήκευση μηνύματος στο "BM". Καμία CBM ειδοποίηση δεν αποστέλλεται καμία ειδοποίηση προς το **TE**.
		2	

Επιστρέφει: **+CPMS:**<used1>,<total1>,<used2>,<total2>,<used3>,<total3>

όπου	<used1>,<used2>,<used3>	Συνολικός αριθμός μηνυμάτων που περιέχονται στις μνήμες <mem1>, <mem2> και <mem3> αντίστοιχα
	<total1>,<total2>,<total3>	Συνολικός αριθμός θέσεων μηνυμάτων στις μνήμες <mem1>, <mem2> και <mem3> αντίστοιχα

Παράδειγμα: AT+CNMI=3,1,2,0

Επιστέφει: OK

3.2.6 ATE

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

ATE Εντολή για την ήχο

Περιγραφή: Ενεργοποιεί ή απενεργοποιεί την ηχώ κατά την διαδικασία σύνταξη των εντολών.

Σύνταξη: **E=**[<value>]

Παράμετροι:	<value>	**0**	Απενεργοποίηση του φαινομένου.
		1	Ενεργοποίηση του φαινομένου.
			Προκαθορισμένη λειτουργία Ενεργοποίηση του φαινομένου

ATE=1

Παράδειγμα: ATE=1

Επιστέφει: OK

3.3 Η χρήση του προγράμματος HyperTerminal

Πριν την διαδικασία ανάπτυξης της εφαρμογής και ειδικότερα πριν ξεκινήσει η διαδικασία προγραμματισμού του μικροελεγκτή κρίθηκε σκόπιμο η αποστολή προς το κινητό τηλέφωνο των εντολών που αναφέρθηκαν, με σκοπώ να εξομοιώσουμε την διαδικασία που θα αναλάμβανε να εκτελέσει στην συνέχεια ο μικροελεγκτής. Για τον λόγο αυτό θα έπρεπε να συνδέσουμε το κινητό τηλέφωνο της εφαρμογής με ένα τερματικό και μέσο μιας γραμμής εντολών να αποστείλουμε τις εντολές AT προς αυτό. Το τερματικό που χρησιμοποιήθηκε ήταν ένας ηλεκτρονικός υπολογιστής ενώ η σύνδεση του με το κινητό τηλέφωνο έγινε μέσο του καλωδίου σύνδεσης στην συριακή θύρα του υπολογιστή. Για την πρόσβαση στη συριακή θύρα κάναμε χρήση του προγράμματος *HyperTerminal* αφού πρώτα καθορίσαμε τις ρυθμίσεις της θύρας δεδομένων (Port Settings) ως εξής:

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

Για την σύνταξη μιας εντολής AT δεν έχουμε παρά να την πληκτρολογήσουμε και να πατήσουμε το πλήκτρο “Enter” (αποστολή χαρακτήρα [CR]) ώστε να αρχίσει η διαδικασία εκτέλεσης της από το κινητό τηλέφωνο.

Αναλυτικά η διαδικασία που λαμβάνει χώρα είναι η εξής. Με το πάτημα ενός πλήκτρου από το πληκτρολόγιο του υπολογιστή αποστέλλεται μέσο της συριακής θύρας μια αλληλουχία bit τα οποία αντιπροσωπεύουν τον ASCII κωδικό του χαρακτήρα που στάλθηκε. Στην συνέχεια και εφόσον είναι ενεργοποιημένη η ηχώ από το κινητό τηλέφωνο (ATE=1) ο χαρακτήρας αυτός επιστρέφεται πίσω στο υπολογιστή τον οποίο λαμβάνει ο δέκτης της συριακής θύρας και τον εκτυπώνει στην

οθόνη. Η ίδια διαδικασία επαναλαμβάνεται για κάθε χαρακτήρα μέχρι το τέλος σύνταξης της εντολής και την αποστολή του χαρακτήρα [CR] (πάτημα πλήκτρου "Enter"). Αν η εντολή που στάλθηκε είναι σωστή το κινητό τηλέφωνο εκτελεί τις κατάλληλες διεργασίες και στέλνει ανάλογο μήνυμα προς τον υπολογιστή. Σε διαφορετική περίπτωση αν ανιχνευθεί λάθος τότε αποστέλλεται κατάλληλο μήνυμα.

Για παράδειγμα αν στην γραμμή εντολών του προγράμματος πληκτρολογήσουμε την εντολή *AT* το τηλέφωνο ανταποκρίνεται με την μήνυμα "*OK"* που σημαίνει ότι η σύνδεση και η επικοινωνία έχει επιτευχθεί και ότι όλα βαίνουν καλώς. Στην συνέχεια μπορούμε να εκτελέσουμε οποιαδήποτε εντολή από το ρεπερτόριο των AT εντολών που διαθέτει η συσκευή δίνοντας ιδιαίτερη έμφαση σε εκείνες που χρησιμοποιούνται για την υλοποίηση της εφαρμογής εξομοιώνοντας με αυτό τον τρόπο την λειτουργία του συστήματος.

3.4 Σύντομο Μήνυμα

Μια από τις βασικές λειτουργίες που εκτελούνται κατά την διάρκεια λειτουργίας του συστήματος είναι εκείνη που αφορά την αποκωδικοποίηση ενός σύντομου μηνύματος. Η μορφή του μηνύματος την οποία λαμβάνει ο μικροελεγκτής από το κινητό τηλέφωνο βάσης διαφέρει κατά πολύ από την μορφή του μηνύματος που απέστειλε ο χρήστης προς το σύστημα. Για τον λόγο αυτό κρίθηκε απαραίτητο η παρουσίαση και η μελέτη όλων παραμέτρων και των διαδικασιών που λαμβάνουν χώρα κατά την μετάδοση ενός σύντομου μηνύματος μέσο της υπηρεσίας SMS του δικτύου κινητής τηλεφωνίας. Επίσης παρουσιάζεται αναλυτικά και εξηγούνται ιδιαίτερα όλα τα τμήματα που συνθέτουν και αποτελούν την PDU μορφή ενός σύντομου μηνύματος τόσο κατά την διαδικασία αποστολής όσο και κατά την διαδικασία λήψης.

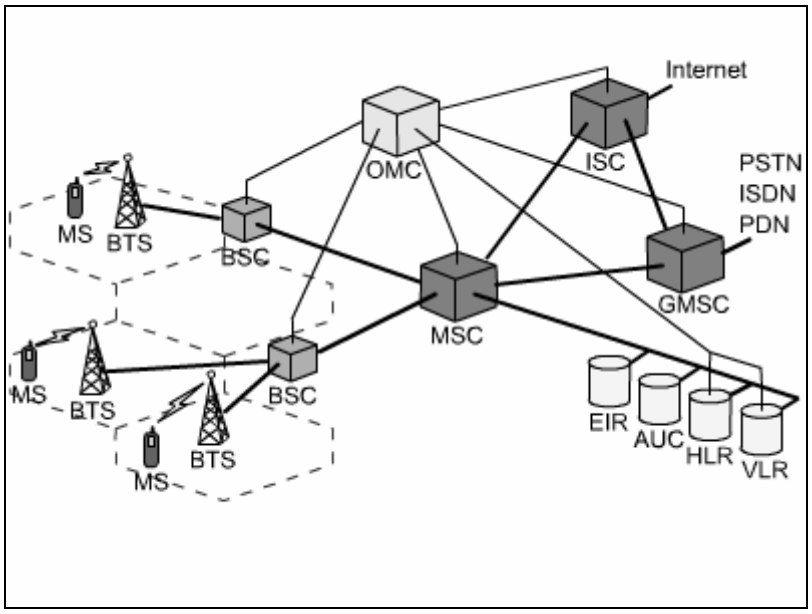
3.4.1 Υπηρεσία Σύντομων Γραπτών Μηνυμάτων SMS

Η υπηρεσία σύντομων γραπτών μηνυμάτων *(SMS, Short Message Service)* όπως και οι AT εντολές καθορίζεται με βάση τα πρότυπα του Ευρωπαϊκού Οργανισμού *ETSI* (πρότυπα *GSM 03.40* και *03.38*). Μέσο αυτής παρέχεται η δυνατότητα στους χρήστες να ανταλλάσσουν αλφαριθμητικά μηνύματα μέχρι και 140 χαρακτήρες όταν κάθε χαρακτήρας αναπαρίσταται από μια λέξη μήκους 7-bits σύμφωνα με την καθορισμένη GSM αλφάβητο (*7-bits GSM default alphabet)*. Η υπηρεσία αυτή δεν απαιτεί εγκατάσταση σύνδεσης από άκρη σε άκρη και κατά συνέπεια η μετάδοση σύντομων μηνυμάτων μπορεί να λάβει χώρα ακόμη και όταν το κινητό τερματικό δεν βρίσκεται σε πλήρη επικοινωνία. Εξάλλου θα πρέπει να αναφέρουμε ότι για την αποστολή σύντομου μηνύματος δεν απαιτείται η δέσμευση φάσματος όπως στην υπηρεσία μεταφοράς φωνής ή δεδομένων αλλά η υπηρεσία αυτή χρησιμοποιεί μόνο τους πόρους του συστήματος σηματοδοσίας.

Υπάρχουν δύο τύποι παροχής υπηρεσιών SMS: *η εκπομπή σε κυψέλη* και *η εκπομπή σημείου προς σημείο*. Στην πρώτη περίπτωση έχουμε αποστολή μηνύματος σε όλα τα ενεργοποιημένα κινητά τερματικά μιας κυψέλης. Η υπηρεσία αυτή χρησιμοποιείται συνήθως για να την μετάδοση μηνυμάτων που αφορούν συνθήκες κυκλοφορίας, πρόβλεψης καιρού κλπ. Στην υπηρεσία σημείου προς σημείο τα μηνύματα μπορούν να μεταδοθούν από το ένα κινητό τερματικό προς το άλλο και αντίστροφα και είναι η υπηρεσία αυτή που χρησιμοποιείται κατά την εφαρμογή.

Η υπηρεσία SMS αποτελεί μια ασύμμετρη υπηρεσία. Τόσο η μετάδοση σύντομων μηνυμάτων από το κινητό τερματικό όσο και μετάδοση προς το κινητό τερματικό θεωρούνται διαφορετικές μεταξύ τους υπηρεσίες. Στην μετάδοση ενός σύντομου μηνύματος μεταξύ δύο τερματικών μεσολαβεί πάντα κάποιο κέντρο μεταγωγής συντόμων μηνυμάτων (SMS Center, SMSC). Έτσι για την αποστολή ενός μηνύματος ο χρήστης αφού πληκτρολογήσει το περιεχόμενο του, θα πρέπει στην συνέχεια να ορίσει τον τελικό προορισμό αποστολής και με κατάλληλη εντολή του κινητού τερματικού να ξεκινήσει την μετάδοση. Το μήνυμα στην συνέχεια φτάνει σε ένα κέντρο SMSC το οποίο στην συνέχεια θα δρομολογηθεί στον πραγματικό του προορισμό. Ο παραλήπτης του μηνύματος προσδιορίζεται από τον αριθμό κλίσης του και εκτός των δεδομένων επισυνάπτονται επιπλέον πληροφορίες που εξυπηρετούν

την διαδικασία παράδοσης του. Τα κυριότερα στοιχεία της αρχιτεκτονικής ενός δικτύου SMS παρουσιάζονται στο σχήμα 3.1



Σχήμα 3.1 *Αρχιτεκτονική δικτύου SMS*

3.4.2 PDU Ακολουθία κατά την λήψη μηνύματος

Η αποστολή και λήψη ενός σύντομου μηνύματος για τον χρήστη δεν είναι τίποτα άλλο παρά η σύνταξη ή το διάβασμα ενός γραπτού κειμένου στην οθόνη ενός κινητού τηλεφώνου. Για το δίκτυο κινητής τηλεφωνίας όμως και κατεπέκτασιν για ένα κινητό τηλέφωνο GSM ένα γραπτό μήνυμα περιέχει πολλά περισσότερα και σε διαφορετικές κωδικοποιήσεις στοιχεία και δεδομένα τα οποία είναι απαραίτητα για την αποστολή και αναπαραγωγή του. Η αποστολή και η λήψη ενός σύντομου μηνύματος μπορεί να γίνει σε μορφή κειμένου ή σε μορφή PDU. Η μορφή κειμένου (text mode) δεν είναι διαθέσιμη σε όλες τις συσκευές κινητής τηλεφωνίας GSM γεγονός το οποίο ισχύει και για την συσκευή της εφαρμογής.

Ένα σύντομο μήνυμα το οποίο βρίσκεται σε PDU μορφή αποτελείται από μια ακολουθία δεκαεξαδικών χαρακτήρων. Η ακολουθία αυτή εκτός των δεδομένων που περιέχονται σε κωδικοποιημένη μορφή περιλαμβάνει και επιπλέον πληροφορίες που αφορούν τον αποστολέα, το κέντρο μεταγωγής σύντομων μηνυμάτων (SMS Center), την ώρα αποστολής, την ημερομηνία καθώς και άλλες πληροφορίες. Η ακολουθία

που παρουσιάζεται στην συνεχεία αποτελεί την PDU μορφή ενός γραπτού μηνύματος στο οποίο περιέχεται η λέξη “hello” όπως το λάβαμε στο κινητό τηλέφωνο *Ericsson T28s* της εφαρμογής και το διαβάσαμε με την βοήθεια του προγράμματος *HyperTerminal* αφού προηγουμένως εκτελέσαμε τις κατάλληλες αρχικοποιήσεις για την μνήμη αποθήκευσης του μηνύματος με την εντολής AT+CPMS.

AT+CMGR=1
0691037901000040C9103969743200500004030912110820805E8329BFD06
OK

Η ακολουθία αυτή όπως θα δήξουμε στη συνέχεια αποτελείται από τμήματα σε κάθε ένα εκ των οποίων περιέχονται συγκεκριμένες πληροφορίες και δεδομένα με διαφορετική κωδικοποίηση. 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*). Στον πίνακας 3.1 που ακολουθεί εξηγούνται τα επιμέρους τμήματα του μηνύματος.

Octet(s)	Πεδίο	Format	In this example
06	Length of the SMSC information	Hex-Octet	
91	Type of address of SMSC	Hex-Octet	International Format
03 79 01 00 00	SMSC Number	Decimal semi-octet	
04	First octet of this SMS-DELIVER message	Hex-Octet	
0C	Length of the sender address	Hex-Octet	12 (decimal)
91	Type of address of the sender number	Hex-Octet	
03 96 97 43 20 05	Sender Number	Decimal semi-octet	30 6979340250
00	TP-PID	Hex-Octet	
00	TP-DCS	Hex-Octet	
40 30 91 21 10 82 08	TP-SCTS	Decimal semi-octet	19-03-04
05	TP-UDL	Hex-Octet	5 characters
E8 32 9B FD 06	TP-UD	8-bit octets representing 7-bit data	“hello”

Πίνακας 3.1 *Επεξήγηση τμημάτων PDU ακολουθίας μηνύματος λήψης*

3.4.3 PDU Ακολουθία κατά την αποστολή μηνύματος

Για την αποστολή ενός σύντομου μηνύματος SMS θα πρέπει πρώτα να συντάξουμε το μήνυμα σε PDU μορφή. Η διαδικασία αυτή περιλαμβάνει αρχικά την κωδικοποίηση του μηνύματος και στην συνέχεια την συμπλήρωση της ακολουθίας με τις απαραίτητες πληροφορίες που απαιτούνται για την αποστολή του. Η παρακάτω ακολουθία αποτελεί την μορφή που θα πρέπει να φτιάξουμε για την αποστολή του μηνύματος “hello”. Στον πίνακας 3.2 που ακολουθεί στην συνέχεια εξηγούνται τα επιμέρους τμήματα της ακολουθίας.

0011000C91039697432005000AA05C8329BFD06

Octet(s)	Πεδίο	Format	
00	Length of SMSC information		
11	First octet of the SMS-SUBMIT message		
00	TP-Message-Reference		
0C	Address-Length. Length of phone number (11)		
91			
03 96 97 43 20 05		Decimal semi-octet	
00	TP-PID		
00	TP-DCS		
AA	TP-Valid Period "AA" means 4 days		
05	TP-User-Data-Length		05 (decimal) The 5 characters of the word“hello”
E8 32 9B FD 06	TP-User-Data		“hello”

Πίνακας 3.2 *Επεξήγηση τμημάτων ακολουθίας για την αποστολή μηνύματος*

Για την μετάδοση του μηνύματος χρησιμοποιούμε την εντολή AT+CMGS όπως φαίνεται στο παρακάτω παράδειγμα σύμφωνα πάντα με τους κανόνες σύνταξης της εντολής.

AT+CMGS=19
> 0011000C91039697432005000AA05E8329BFD06<CTRL-Z>

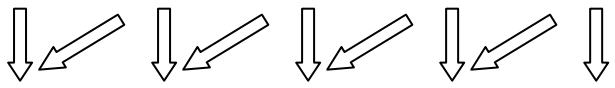
Η μετάδοση του μηνύματος ξεκινά μετά την αποστολή του χαρακτήρα 1A (η ASCII τιμή του πλήκτρου <CTRL-Z>). Για ακύρωση της μετάδοσης πρέπει να πατήσουμε το πλήκτρο ESC .

3.5 Αλγόριθμος Κωδικοποίησης

Για την αποστολή ενός σύντομου μηνύματος απαιτείται αρχικά η δημιουργία μιας αλφαριθμητικής ακολουθίας η οποία αποτελεί την PDU μορφή του. Η ακολουθία αυτή περιέχει τα δεδομένα που θέλουμε να αποστείλουμε στην κωδικοποιημένη τους μορφή, όπως απαιτείται για την επιτυχή μετάδοση τους. Ο αλγόριθμος κωδικοποίησης ενός μηνύματος καθορίζεται από τον οργανισμό *ETSI* σύμφωνα με το πρότυπο *GSM 03.38*. Η μετατροπή των δεδομένων γίνεται με βάση την καθορισμένη *GSM* αλφάβητο (*7-bits GSM default alphabet)* όπου για την αναπαράσταση κάθε χαρακτήρα απαιτούνται 7 bits (*septets)*. Αντίθετα από την διαδικασία κωδικοποίησης προκύπτουν δεκαεξαδικοί αριθμοί που για την αναπαράσταση τους απαιτούνται 8 bit ανά χαρακτήρα δεδομένου (octes).

Ο αλγόριθμος κωδικοποίησης θα περιγραφεί στην συνέχεια με την βοήθεια ενός παραδείγματος. Για την κωδικοποίηση της λέξης “hello” θα πρέπει αρχικά να εκφράσουμε κάθε χαρακτήρα του μηνύματος στην δεκαεξαδική του μορφή σύμφωνα με την καθορισμένη *GSM* αλφάβητο. Για χαρακτήρες γραμμάτων ή αριθμών η δεκαεξαδική τους τιμή συμπίπτει με την ASCII κωδικοποίηση τους όπως φαίνεται και στο παράδειγμα. Η κωδικοποίηση κάθε χαρακτήρα στην ακολουθία δεν εξαρτάτε μόνο από τον ίδιο αλλά είναι συνάρτηση και του επόμενου χαρακτήρα του μηνύματος που ακολουθεί.

Character	h	E	l	l	o
Hex	68	65	6C	6C	6F
Septets	1101000	1100101	1101100	1101100	1101111



Octets	11101000	00110010	10011011	11111101	00000110
Hex	E8	32	9B	FD	06

Σχήμα 3.2 *Αλγόριθμος κωδικοποίησης*

Για τον λόγο αυτό ίδιοι χαρακτήρες μέσα στο ίδιο μήνυμα δεν παρουσιάζουν την ίδια κωδικοποίηση. Όπως φαίνεται στο σχήμα 3.2 για την δημιουργία του πρώτου octet (C8h) χρειαστήκαμε τα εφτά MSB bits του πρώτου septet (68h χαρακτήρας 'h') και το LSB bit (1) από τον δεύτερο septet (65h, χαρακτήρας 'e') του μηνύματος. Το bit αυτό τοποθετείτε στην αρχή του πρώτου octet και στην συνέχεια έπονται τα εφτά MSB bits (1001000) του πρώτου septet. Για την δημιουργία του δεύτερου octet (32) χρειαζόμαστε τα έξι MSB bits του δεύτερου septet (65 χαρακτήρας 'e') και τα δύο LSB (00) bit από το τρίτο septet (6C, χαρακτήρας 'l'). Τοποθετώντας τα δύο LSB bit στην αρχή και προσθέτοντας τα έξι MSB bits προκύπτει το δεύτερο octet (32). Εφαρμόζοντας την διαδικασία αυτή κατά τρόπο ανάλογο και στους επόμενους χαρακτήρες του μηνύματος θα προκύψει η κωδικοποίηση (packing) της λέξης "hello" E8329BFD06 όπως αυτή φαίνεται και στην PDU μορφή του μηνύματος.

Στο σημείο αυτό θα πρέπει να επισημάνουμε ότι κατά την εφαρμογή της διαδικασίας κωδικοποίησης το τελευταίο octet του μηνύματος προκύπτει από septet του τελευταίου χαρακτήρα καθώς και του «μηδενικού» septet που εμείς θεωρούμε ότι ακολουθεί το τελευταίο septet του μηνύματος. Επίσης ένα άλλο σημείο που χρήζει προσοχής είναι ότι το όγδοο στην σειρά octet προκύπτει μόνο από το όγδοο septet του μηνύματος ενώ δεν εξαρτάται από την μορφή του ένατου septet που μπορεί να ακολουθεί. Το γεγονός αυτό αποτελεί την αιτία για τη οποία από οκτώ χαρακτήρες μηνύματος (από οκτώ septets δηλ) προκύπτουν επτά και όχι οκτώ octet. Η διαδικασία που περιγράφτηκε επαναλαμβάνεται για κάθε οκτάδα septet μέχρι το τέλος του μηνύματος.

3.6 Αλγόριθμος Αποκωδικοποίησης

Ο αλγόριθμος αποκωδικοποίησης αποτελεί την αντίστροφη διαδικασία της κωδικοποίησης. Στην περίπτωση αυτή μετατρέπονται δεκαεξαδικοί αριθμοί (octes) σε χαρακτήρες που αναπαρίστανται από επτά bit (septets) σύμφωνα και πάλι με την GSM αλφάβητο. Για την υλοποίηση του αλγορίθμου αυτού εκτελούνται κατά αναλογία τα αντίθετα βήματα που περιγράφτηκαν στην διαδικασία κωδικοποίησης. Στο σχήμα 3.3 παρουσιάζεται γραφικά μια εφαρμογή του αλγορίθμου αυτού.

Hex	E8	32	9B	FD	06
Octets	11101000	00110010	10011011	11111101	00000110

Septets	1001000	1100101	1101100	1101100	1101111
Character	h	e	l	l	o

Σχήμα 3.3 *Αλγόριθμος αποκωδικοποίησης*

Κατά την λειτουργία της εφαρμογής ο αλγόριθμος αυτός εκτελείται την στιγμή που ο μικροελεγκτής παραλάβει ένα νέο μήνυμα από το κινητό τηλέφωνο βάσης, σε PDU μορφή, οπότε και θα πρέπει να το ανακτήσει στην αρχική μορφή που απέστειλε ο χρήστης.

ΚΕΦΑΛΑΙΟ 4

ΜΙΚΡΟΕΛΕΓΚΤΕΣ ΟΙΚΟΓΕΝΕΙΑΣ PIC

4.1 Μικροελεγκτές και Μικροεπεξεργαστές

Είναι φανερό ότι οι μικροελεγκτές (microcontrollers) και οι μικροεπεξεργαστές (microprocessors) αποτελούν απαραίτητο μέρος των σύγχρονων τεχνολογικών εφαρμογών. Πιο συγκεκριμένα, ο μικροεπεξεργαστής (μΕ) αποτελεί το κεντρικό στοιχείο σε κάθε μικροϋπολογιστικό σύστημα. Ωστόσο, οι δύο αυτές συσκευές διαφέρουν μεταξύ τους σε αρκετά σημεία.

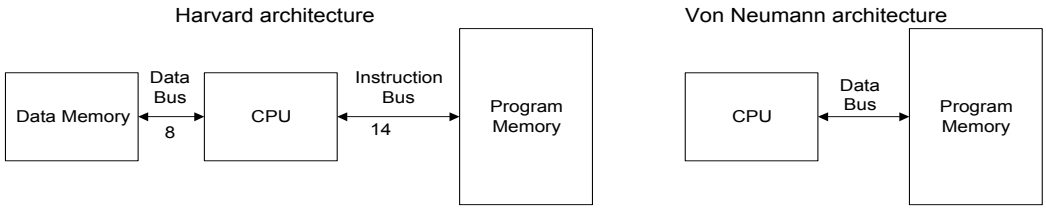
Μια βασική τους διαφορά, και η ποιο σημαντική, βρίσκεται στην λειτουργικότητά τους. Προκειμένου να λειτουργήσει ένας μικροεπεξεργαστής θα πρέπει να συνδεθεί και με άλλες συσκευές, όπως μνήμη (memory) ή συσκευή αποστολής και λήψης δεδομένων. Αυτό σημαίνει ότι ένας μικροεπεξεργαστής είναι η καρδιά του 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.

4.2 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 (**R**educed **I**nstruction **S**et **C**omputer) ενώ εκείνοι που χρησιμοποιούν την αρχιτεκτονική Von-Neumann καλούνται μικροελεγκτές CISI (**C**omplex **I**nstruction **S**et **C**omputer).

Όπως εύκολα μπορεί κανείς να παρατηρήσει και από το σχήμα 4.1, 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 (**b**inary dig**its**, 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.



Σχήμα 4.1 *Αρχιτεκτονικές Harvard και Von Neumann*

4.3 Γενικά χαρακτηριστικά των μικροελεγκτών της οικογένειας PIC

Ο όρος PIC (**P**eripheral **I**nterface **C**ontroller), 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[1] (**I**n-**C**ircuit **S**erial **P**rogrammer, 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.

4.4 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 θέματα του Pic που αφορούν:

- **την αρχιτεκτονική του**

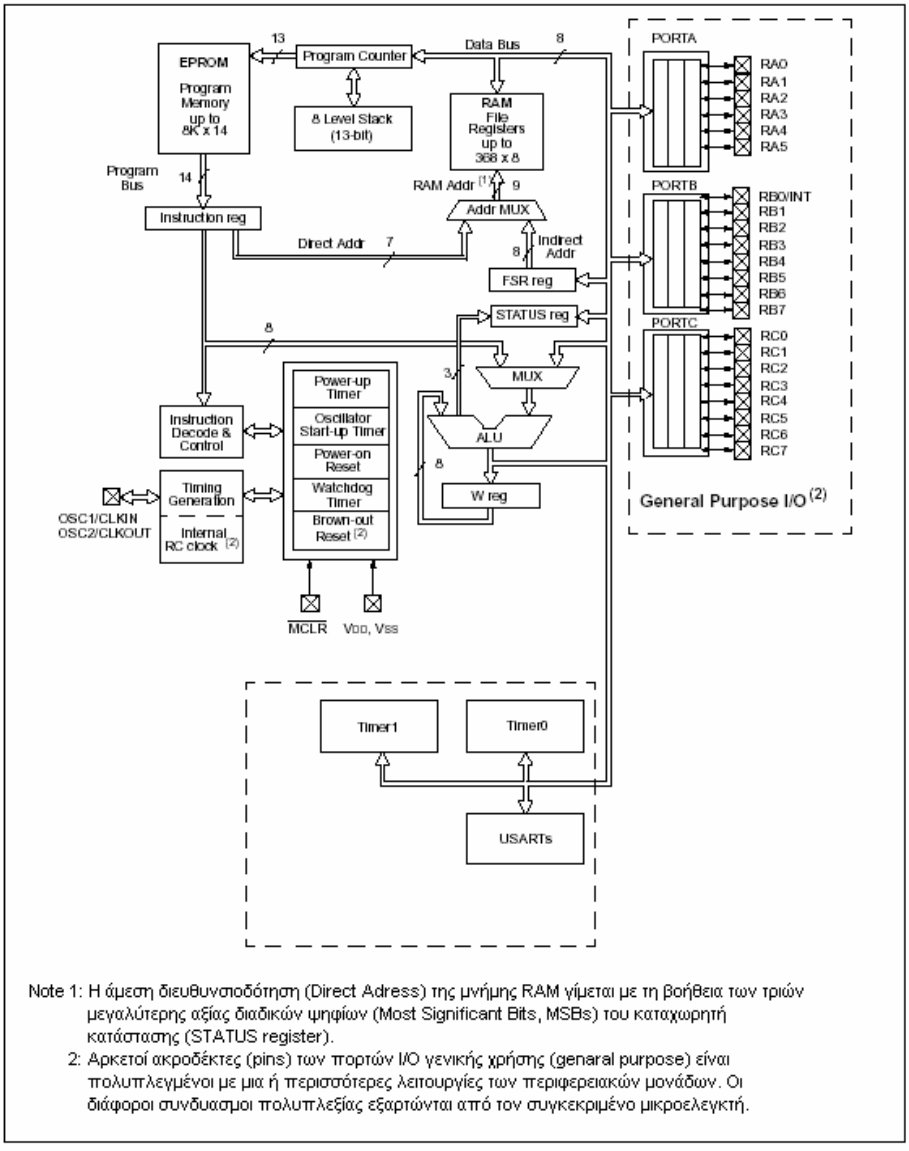
- **τον πυρήνα του :**
  - Κεντρική Μονάδα Επεξεργασίας (**C**entral **P**rocessing **U**nit, CPU)
  - Ρολόι - Χρονισμοί - Κύκλος Εντολής
  - Μονάδα συνεχούς διοχέτευσης εντολών (Instruction Pipelining)
  - Αριθμητική Λογική Μονάδα (**A**rithmetic **L**ogic **U**nit, ALU)
  - Μνήμη προγράμματος ROM (Program Memory)
  - Μνήμη δεδομένων RAM (Data Memory)
  - Στοίβα (Stack)
  - Καταχωρητές: μετρητής προγράμματος (**P**rogram **C**ounter, PC)
    καταχωρητής εργασίας (**W**orking Register, W)
    καταχωρητής κατάστασης (STATUS register)
  - Διακοπές (Interrupts)
  - Εντολές (Instruction Set)

- **τα περιφερειακά του :**
  - Γενικής χρήσης I/O πόρτες (PortA, PortB, PortC)
  - Χρονιστές (Timer0, Timer1)
  - Πομπός / δέκτης σύγχρονης σειριακής επικοινωνίας (**U**niversal **S**ynchronous **A**synchronous **R**eceiver **T**ransmitter, USART)

- **τα ειδικά χαρακτηριστικά του :**
  - Watchdog Timer
  - Ενσωματωμένος σειριακός προγραμματιστής (**I**n-**C**ircuit **S**erial **P**rogrammer, ICSP)

4.4.1 Αρχιτεκτονική του PIC

Στο σχήμα 4.2 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).



Note 1: Η άμεση διευθυνσοδότηση (Direct Adress) της μνήμης RAM γίμεται με τη βοήθεια των τριών μεγαλύτερης αξίας διαδικών ψηφίων (Most Significant Bits, MSBs) του καταχωρητή κατάστασης (STATUS register).

2: Αρκετοί ακροδέκτες (pins) των πορτών I/O γενικής χρήσης (genaral purpose) είναι πολυπλεγμένοι με μια ή περισσότερες λειτουργίες των περιφερειακών μονάδων. Οι διάφοροι συνδυασμοι πολυπλεξίας εξαρτώνται από τον συγκεκριμένο μικροελεγκτή.

Σχήμα 4.2 *Αρχιτεκτονική PIC της οικογένειας 16CXXX*

4.4.2. Ο πυρήνας του PIC

Α. 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).

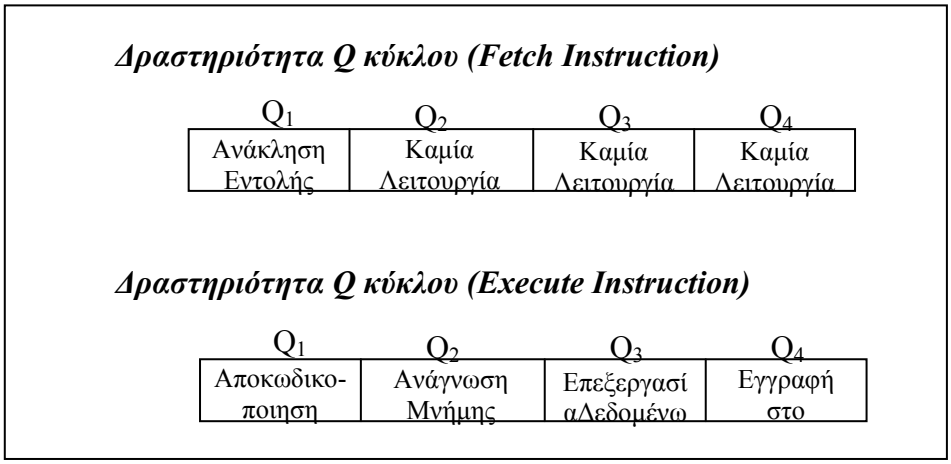
Οι βασικές λειτουργίες που εκτελεί μια CPU είναι :

- 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.

Β. Ρολόι - Χρονισμοί - Κύκλος Εντολής

Οι παλμοί που παράγονται από τον ταλαντωτή (OSC1) διαιρούνται εσωτερικά με το 4 για να δώσουν τέσσερις μη υπερκαλυπτόμενους παλμούς Q_1, Q_2, Q_3, Q_4. 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 Q_1. Κατά την εκτέλεση η εντολή αποθηκεύεται στο καταχωρητή εντολών, Instruction Register (IR), κατά την διάρκεια της φάσης Q_1. Η εντολή αποκωδικοποιείται και εκτελείται στις φάσεις Q_2, Q_3, Q_4. Τα δεδομένα διαβάζονται από τη μνήμη κατά την φάση Q_2, και γράφονται κατά την Q_4. Κατά την φάση Q_3

γίνεται η επεξεργασία των δεδομένων. Το σχήμα 4.3 παρουσιάζει μια σχηματική περιγραφή των παραπάνω :



Σχήμα 4.3 *Δραστηριότητα Q κύκλου (Q cycle activity)*

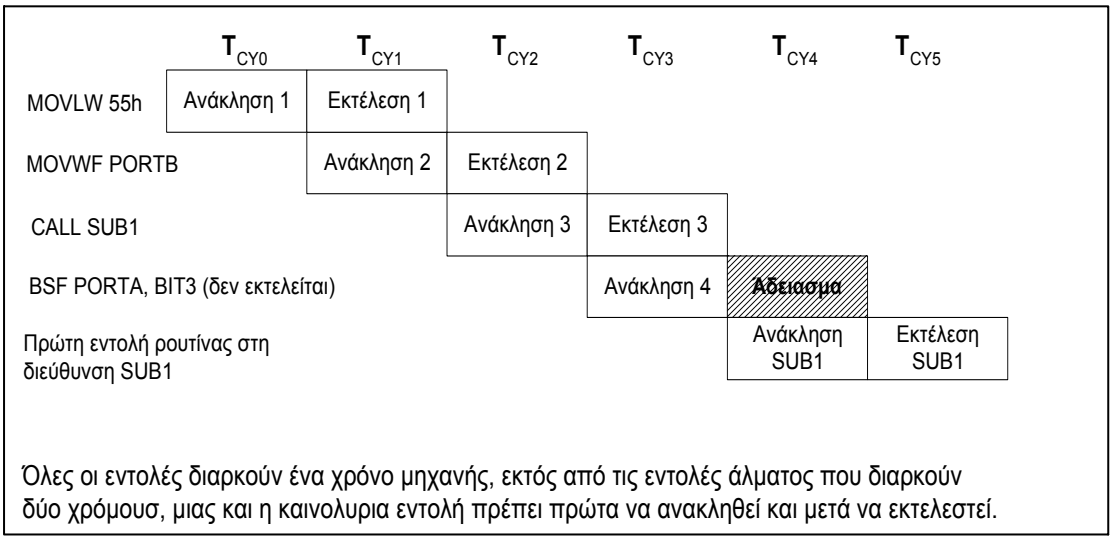
Το σύνολο των τεσσάρων παλμών Q_1 ως Q_4 αποτελούν ένα κύκλο εντολής (ή κύκλο μηχανής). Μέσα σε ένα κύκλο εντολής εκτελείται μια οποιαδήποτε εντολή. Εάν υπολογίσουμε ότι για την εκτέλεση μιας εντολής χρειαζόμαστε ένα κύκλο μηχανής καθώς και έναν επιπλέον κύκλο μηχανής για την ανάκληση της εντολής έχουμε 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).

Γ. Μονάδα συνεχούς διοχέτευσης εντολών (Instruction Pipelining)

Ο κύκλος εντολής αποτελείται από τέσσερις Q φάσεις (Q_1, Q_2, Q_3, Q_4). 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 (**N**o OPeration) που ισοδυναμεί με άδειασμα του buffer της μονάδας συνεχούς διοχέτευσης.

Στη συνέχεια (σχήμα 4.4) ακολουθεί 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.

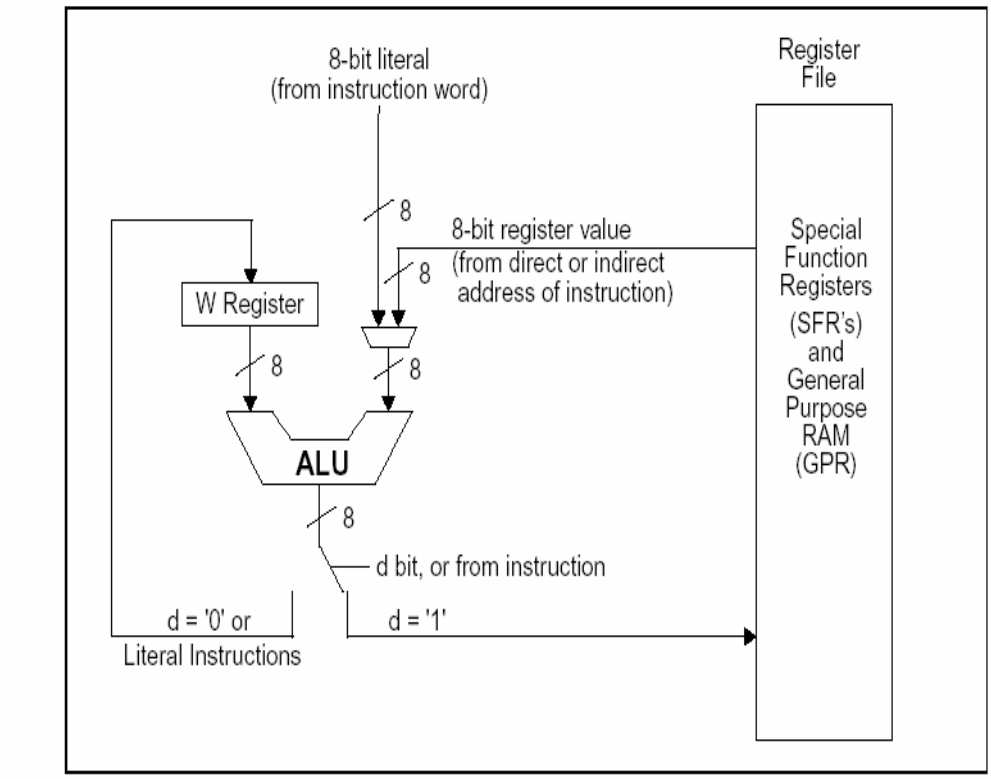


Σχήμα 4.4 *Παράδειγμα συνεχούς διοχέτευσης εντολών (pipelining)*

Δ. 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 4.5 που ακολουθεί παρουσιάζεται η ALU του PIC.



Σχήμα 4.5 *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 (**L**ess **S**ignificant **Bit**, LSBs) του καταχωρητή κατάστασης STATUS.

Ε. Μνήμη Προγράμματος ROM (Program Memory)

Η οικογένεια PIC16XXX διαθέτει μετρητή προγράμματος των 13-bit και μπορεί συνεπώς να διαχειρίζεται μνήμη προγράμματος ίση με 2^{13}=8K θέσεις. Το εύρος του διαύλου επικοινωνίας με την μνήμη προγράμματος είναι 14 bit. Από τη στιγμή που όλες εντολές είναι μιας λέξης (single word), μια συσκευή με 8K x 14 bit εντολής, διαθέτει χώρο για 8K εντολές.

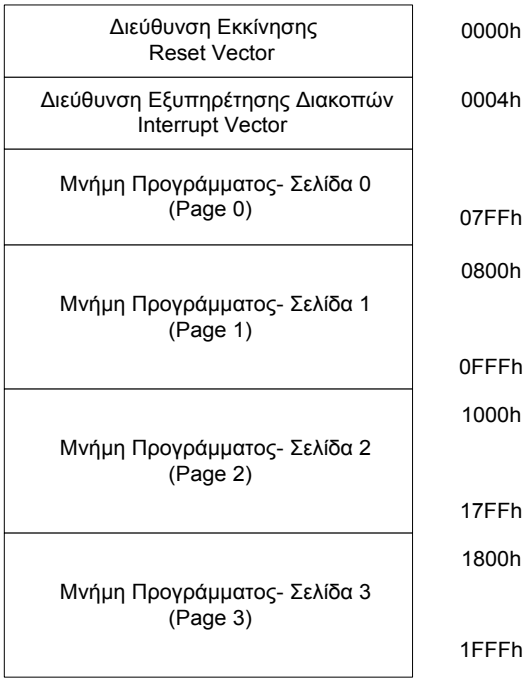
Η μνήμη εντολών διαιρείται σε τέσσερις σελίδες (pages) 2K λέξεων η κάθε μια, που αντιστοιχεί στις επόμενες διευθύνσεις :

- Page 1 : 0h - 7FFh
- Page 2 : 800h – FFFh
- Page 3 : 1000h - 17FFh
- Page 4 : 1800h - 1FFFh

Στο παρακάτω σχήμα (σχήμα 4.6) φαίνεται ο χάρτης της μνήμης προγράμματος. Για να μπορούμε να εκτελέσουμε άλματα πρέπει να βρούμε δύο bit για να προσδιορίσουμε την σελίδα που θα κάνουμε άλμα. Τα δύο αυτά bit περιέχονται στον καταχωρητή PCLATH (PCLATH<4:3>). Πριν από μια εντολή CALL ή GOTO θα πρέπει να είμαστε σίγουροι ότι τα δύο bit που δείχνουν τον αριθμό της σελίδας στην οποία εκτελείται το άλμα είναι τα επιθυμητά. Αντίθετα όταν γυρνάμε από μια υπορουτίνα με μια εντολή RET, δεν χρειάζεται οποιοσδήποτε χειρισμός αυτών των δύο bit αφού και τα 13 bit του PC αποκαθίστανται από την στοίβα.

Για να γίνει άλμα μεταξύ διαφορετικών σελίδων, όπως είδαμε, θα πρέπει να αλλάξουμε τα υψηλής αξίας bit του μετρητή προγράμματος μέσω του καταχωρητή PCLATH. Στην περίπτωση που εκτελούνται διαδοχικά οι εντολές χωρίς την αλλαγή της ροής του προγράμματος τότε το πρόγραμμα θα συνεχίσει κανονικά σε μια νέα σελίδα ανεξάρτητα της τιμής του καταχωρητή PCLATH.

Κατά την εκκίνηση (reset) ο μετρητής προγράμματος (PC) αρχικοποιείται στην διεύθυνση 0h και ο PCLATH επίσης στην τιμή 0. Συνεπώς η αρχή για κάθε πρόγραμμα είναι η διεύθυνση 0 της σελίδας 0. Η διεύθυνση αυτή ονομάζεται άνυσμα διεύθυνσης επανεκκίνησης (Reset Vector Address).



Σχήμα 4.6 *Χάρτης της μνήμης προγράμματος (program memory map)*

Όταν συμβαίνει κάποια διακοπή (Interrupt),ο μετρητής προγράμματος PC, παίρνει την τιμή 0004h που είναι η διεύθυνση της επόμενης εντολής που θα εκτελεσθεί. Η διεύθυνση αυτή καλείται άνυσμα διεύθυνσης διακοπής (Interrupt Vector Address). Ο καταχωρητής PCLATH δεν αλλάζει. Γι' αυτό μόλις μπούμε μέσα στην ρουτίνα εξυπηρέτησης της διακοπής (Interrupt Service Routine) θα πρέπει να αλλάζουμε τη διεύθυνσή του κατάλληλα, εάν αυτό επιθυμούμε, και να σώζουμε τα δεδομένα του ώστε να μην επηρεαστεί το κυρίως πρόγραμμα μετά την εξυπηρέτηση της διακοπής. Για την επιστροφή από τη διακοπή δεν χρειάζεται ο καταχωρητής PCLATH μια και ολόκληρος ο μετρητής προγράμματος αποθηκεύεται στη στοίβα κατά την είσοδο μας στη ρουτίνα εξυπηρέτησης της διακοπής.

Η μνήμη προγράμματος περιέχει επίσης και ένα αριθμό από πληροφορίες για την ακριβέστερη ρύθμιση της κάθε συσκευής (Calibration Values). Οι τιμές αυτές προγραμματίζονται από την Microchip, κατά την τελική δοκιμή της συσκευής. Οι διευθύνσεις στις οποίες σώζονται οι τιμές αυτές θα πρέπει να σώζονται εάν πρόκειται να σβήσουμε την μνήμη προγράμματος και να αποκαθίστανται στη συνέχεια.

ΣΤ. Μνήμη Δεδομένων RAM (Data Memory)

Η μνήμη δεδομένων περιέχει τους ειδικού τύπου καταχωρητές (**S**pecial **F**unction **R**egisters, SFRs) και τους γενικού τύπου καταχωρητές (**G**eneral **P**urpose **R**egisters, GPRs). Χωρίζεται σε τέσσερα το πολύ τμήματα (banks) των 128 bytes. Τα bit που ελέγχουν σε ποιο τμήμα (bank) αναφερόμαστε, είναι τα 3 MSB ψηφία του καταχωρητή STATUS (STATUS<7:5>) και η μνήμη μπορεί να προσπελαστεί είτε άμεσα είτε έμμεσα.

Κάθε τμήμα περιέχει καταχωρητές γενικού σκοπού (GPRs) και καταχωρητές ειδικού σκοπού (SFRs). Η σχηματική παρουσίαση της μνήμη του PIC ακολουθεί την δομή του σχήματος ? που παρουσιάζεται στην επόμενη σελίδα.

Οι πρώτες θέσεις κάθε τμήματος είναι δεσμευμένες για τους καταχωρητές ειδικού σκοπού (SFR). Το υπόλοιπο κάθε τμήματος αποτελείται από τους καταχωρητές γενικού σκοπού (GPR). Όλοι οι καταχωρητές είναι υλοποιημένοι ως στατική μνήμη. Καταχωρητές ειδικού σκοπού που χρησιμοποιούνται συχνά αλλά και ένα τμήμα από 16 GPRs μπορεί να είναι προσβάσιμοι από παραπάνω από ένα τμήματα.

Όπως αναφέρθηκε και προηγουμένως, η προσπέλαση της μνήμης μπορεί να είναι άμεση (direct addressing) ή έμμεση (indirect addressing). Η επιλογή του τμήματος της μνήμης, ανάλογα με το αν η προσπέλαση της είναι άμεση ή έμμεση γίνεται με βάση τον παρακάτω πίνακα (Πίνακας 4.1).

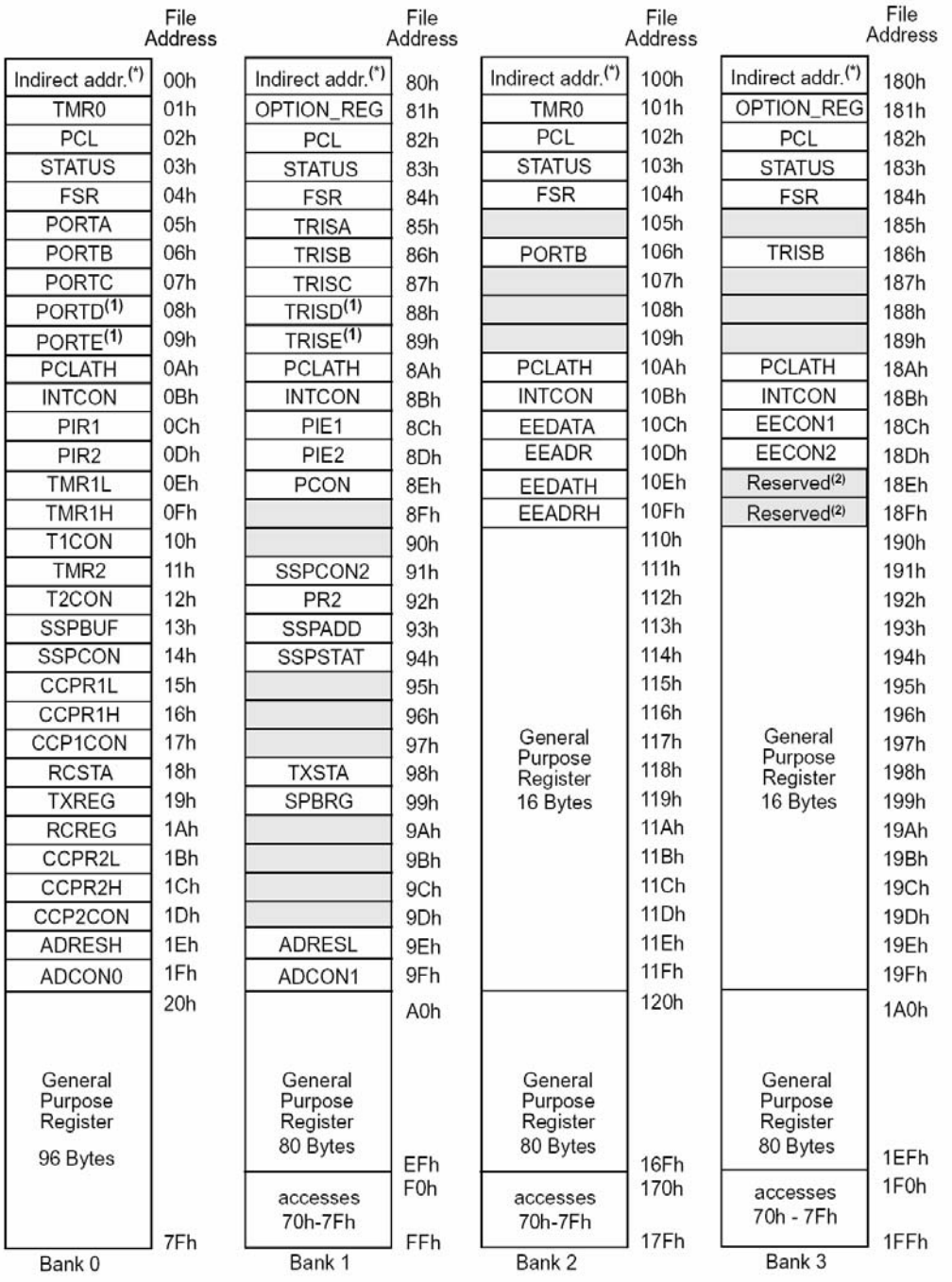
Accessed Bank	Direct (RP1:RP0)	Indirect (IRP)
0	0 0	0
1	0 1	
2	1 0	1
3	1 1	

Πίνακας 4.1 *Άμεση και έμμεση προσπέλαση μνήμης*

Άμεση Διευθυνσιοδότηση (Direct Addressing)

Άμεση διευθυνσιοδότηση (direct addressing) έχουμε όταν το όρισμα της εκτελούμενης εντολής είναι η διεύθυνση του καταχωρητή στον οποίο αναφερόμαστε.

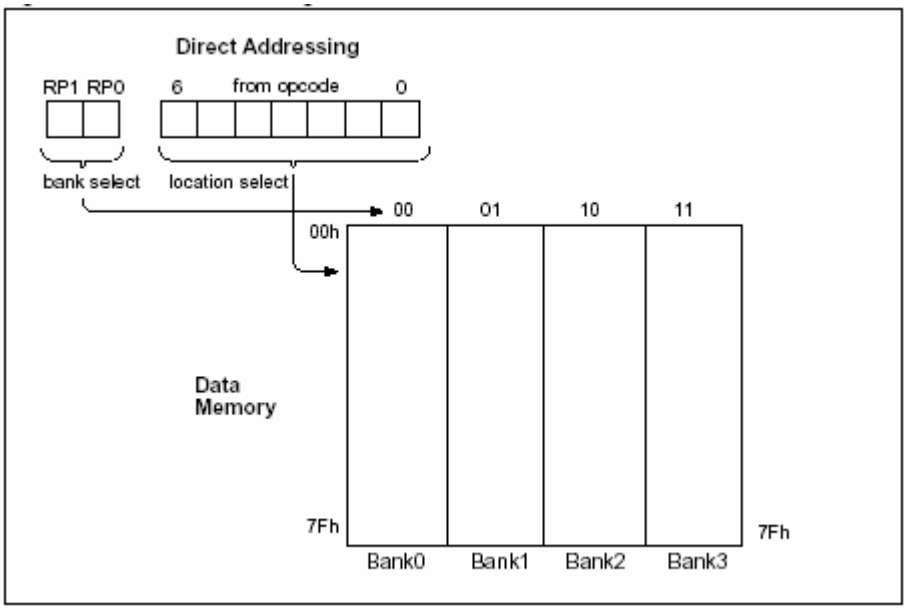
Επειδή το όρισμα αυτό είναι το πολύ 7-bit, μπορούμε να προσπελάσουμε μόνο ένα καταχωρητή από κάθε τμήμα κάθε φορά.



Σχήμα 4.7 *Χάρτης της μνήμης δεδομένων (Data Memory)*

Για να αλλάξουμε μεταξύ των διαφορετικών τμημάτων της μνήμης δεδομένων θα πρέπει να δίνουμε κατάλληλες τιμές στα bit RP0,RP1 που βρίσκονται στο καταχωρητή STATUS (STATUS <6:5>) σύμφωνα με τον πίνακα 4.1.

Μια σχηματική παρουσίαση όλων των παραπάνω φαίνεται στο σχήμα 4.8.

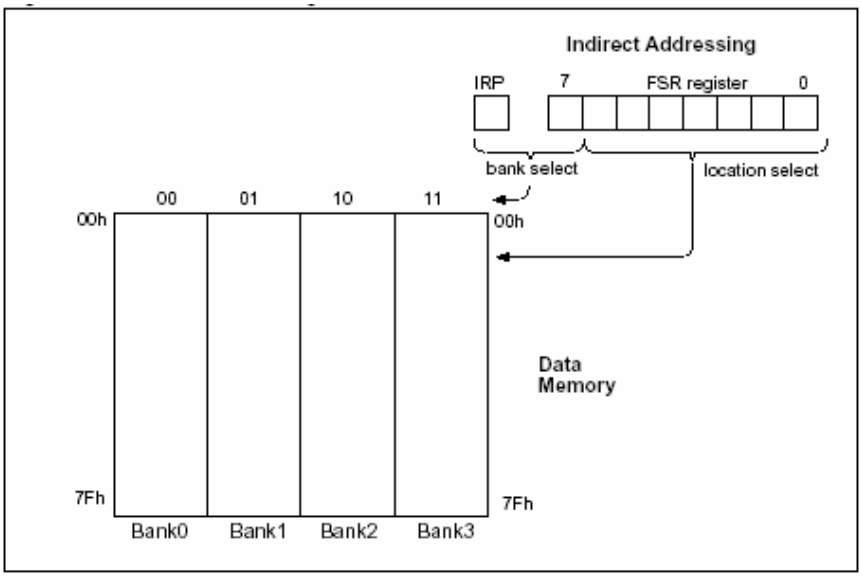


Σχήμα 4.8 *Άμεση προσπέλαση μνήμης (Direct Addressing)*

Έμμεση Διευθυνσιοδότηση (Indirect Addressing)

Με αυτόν τον τρόπο, το όρισμα της διεύθυνσης στην οποία απευθύνεται η εντολή δεν είναι σταθερό, αλλά ένας ειδικός καταχωρητής δείχνει τη διεύθυνση που θέλουμε να προσπελάσουμε. Με αυτό το τρόπο μπορούμε να κάνουμε εύκολη υλοποίηση της δομής ενός πίνακα δεδομένων .

Η έμμεση διευθυνσιοδότηση είναι δυνατή χρησιμοποιώντας τον καταχωρητή INDF σε συνεργασία με τον FSR. Κάθε εντολή που χρησιμοποιεί τον καταχωρητή INDF στην ουσία κάνει προσπέλαση στον καταχωρητή που δείχνει ο FSR. Επειδή ο FSR έχει μέγεθος 8 bit το ένατο bit της διεύθυνσης προέρχεται από το bit 7 του καταχωρητή STATUS (IRP) (Σχήμα 4.9). Εάν προσπαθήσουμε να διαβάσουμε τον ίδιο τον INDF έμμεσα, (για FSR=0) θα πάρουμε ως αποτέλεσμα την τιμή 0 ενώ εάν προσπαθήσουμε να γράψουμε δεν έχουμε καμία ενέργεια (αν και τα bit κατάστασης μπορεί να μεταβληθούν)



Σχήμα 4.9 *Έμμεση προσπέλαση μνήμης (Indirect Addressing)*

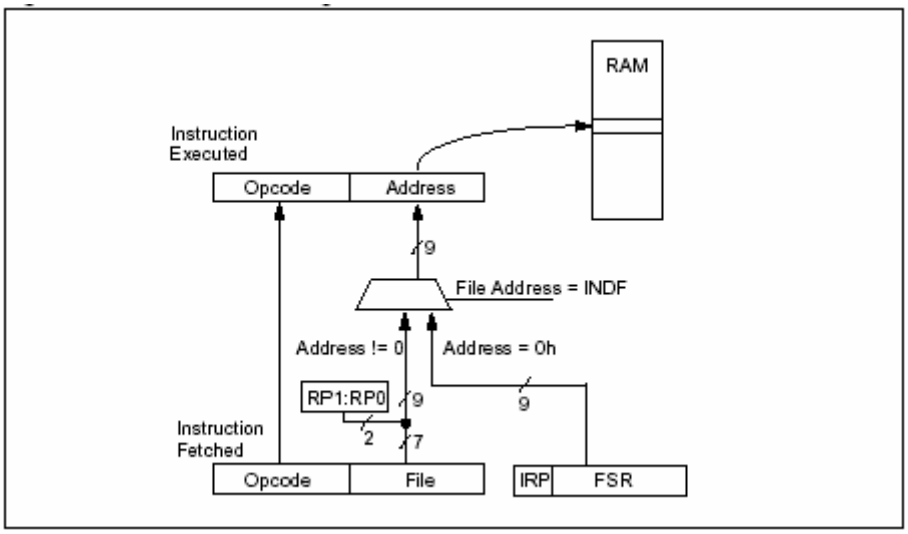
Ένα γενικότερο σχήμα σχετικά με τον τρόπο προσπέλασης της μνήμης, φαίνεται παρακάτω. Μπορούμε να παρατηρήσουμε πως σε περίπτωση άμεσης προσπέλασης, τα 9 bit της διεύθυνσης της μνήμης καθορίζονται από τον καταχωρητή της εντολής και από τα bit RP0 και RP1 του STATUS register, ενώ σε περίπτωση έμμεσης προσπέλασης, η ζητούμενη διεύθυνση προκύπτει από τα 8 bit του FSR και το bit IRP του STATUS register. Ένας πολυπλέκτης καθορίζει με βάση την διεύθυνση που δίνεται στο καταχωρητή εντολών (**I**nstruction **R**egister, IR), για το αν πρόκειται για άμεση ή έμμεση προσπέλαση μνήμης.

Καταχωρητές γενικής χρήσης (General Purpose Registers)

Οι γενικού σκοπού καταχωρητές είναι το σύνολο της διαθέσιμης μνήμης δεδομένων του χρήστη με συνολική χωρητικότητα 368 bytes. Όπως φαίνεται στο σχήμα 4.7 , οι καταχωρητές αυτοί βρίσκονται στο τέλος κάθε τμήματος. Οι καταχωρητές γενικής χρήσης δεν αρχικοποιούνται στην περίπτωση Power-on Reset και κρατούν τις τιμές τους στην περίπτωση όλων των άλλων resets. Οι καταχωρητές είναι προσβάσιμοι είτε άμεσα είτε έμμεσα μέσω του καταχωρητή File Select Register (FSR).

Σε μερικές συσκευές, ένα μέρος των καταχωρητών αυτών είναι το ίδιο σε όλα τα τμήματα και συνεπώς μπορούμε να έχουμε πρόσβαση σε αυτούς ανεξαρτήτως του

τμήματος (bonk). Εγγραφή σε αυτούς σημαίνει εγγραφή και στην ίδια διεύθυνσης σε όλα τα υπόλοιπα τμήματα.



Σχήμα 4.10 *Επιλογή του τρόπου προσπέλασης της μνήμης*

Καταχωρητές ειδικού σκοπού (Special Function Registers)

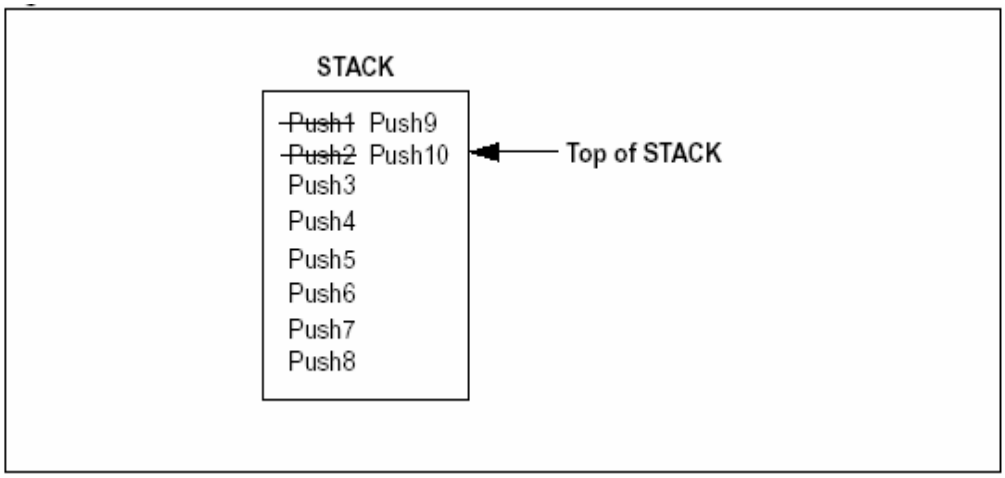
Οι καταχωρητές ειδικού σκοπού μπορούν να διαχωριστούν σε δύο κατηγορίες. Σε αυτούς που χρησιμοποιούνται για τον έλεγχο της κατάστασης και τη ρύθμιση της λειτουργίας των περιφερειακών μονάδων και σε εκείνους που χρησιμοποιούνται από τον πυρήνα της συσκευής και την Κεντρική Μονάδα Επεξεργασίας. Μερικοί καταχωρητές ειδικού τύπου αρχικοποιούνται στην περίπτωση ενός Power-on Reset ενώ άλλοι παραμένουν άθικτοι.

Ζ. Στοίβα (Stack)

Η στοίβα (stack) έχει διαστάσεις 8x13bit επιτρέποντας να συμβούν διαδοχικά μέχρι 8 κλήσεις προγράμματος (CALL) ή διακοπών. Η στοίβα δεν ανήκει ούτε στη μνήμη προγράμματος ούτε στη μνήμη δεδομένων. Το πρόγραμμα δεν μπορεί να έχει άμεση πρόσβαση στη στοίβα ούτε για εγγραφή ούτε για ανάγνωση. Κατά την εκτέλεση μιας εντολής κλήσης υπορουτίνας (CALL) ή τη στιγμή που συμβαίνει μια διακοπή, η τιμή του PC προωθείται αυτόματα στην στοίβα (PUSH). Μόλις το

πρόγραμμα συναντήσει μια εντολή επιστροφής από υπορουτίνα ή διακοπή (RETURN, RETLW, RETFIE) η παλιά τιμή του PC ανακαλείται από τη στοίβα και αποκαθίσταται. Ο καταχωρητής PCLATH δεν αλλάζει ενώ η τιμή του PC προωθείται προς τη στοίβα ή ανακαλείται από αυτή.

Η στοίβα είναι κυκλική. Αυτό σημαίνει ότι μόλις αποθηκευθεί η όγδοη λέξη, η επόμενη λέξη θα γραφτεί πάνω από την πρώτη κ.ο.κ. (Σχήμα 4.11). Αυτό βέβαια θα έχει καταστροφικές συνέπειες για τη ροή του προγράμματος για αυτό θα πρέπει να διασφαλίζεται ότι ο μέγιστος αριθμός των συνεχόμενων κλήσεων σε υπορουτίνες συνυπολογίζοντας και την περίπτωση κάποιας διακοπής, δεν θα πρέπει να ξεπερνά τις οκτώ.



Σχήμα 4.11 *Τρόπος λειτουργίας της στοίβας (stack)*

Η. Καταχωρητές PC και STATUS

*Μετρητής Προγράμματος (**P**rogram **C**ounter, PC)*

Ο μετρητής προγράμματος (PC) δείχνει την εντολή που θα ανακληθεί από την μνήμη εντολών και θα εκτελεσθεί στον επόμενο κύκλο εντολής. Ο μετρητής προγράμματος έχει μήκος 13-bit συνεπώς μπορούμε να διαχειριζόμαστε μέχρι 2^{13}=8k μνήμης προγράμματος.

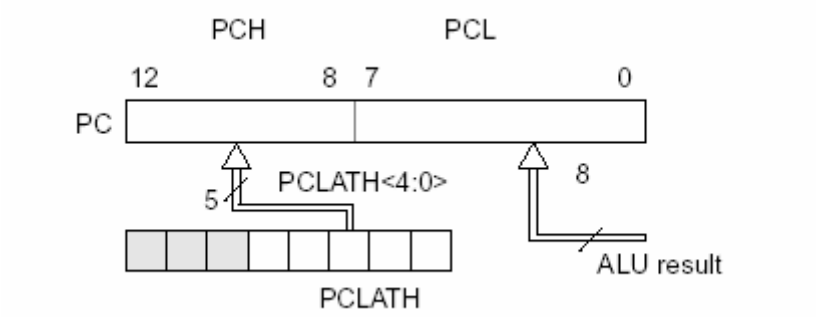
Ο καταχωρητής PC αποτελείται από δύο τμήματα. Τα χαμηλής αξίας 8 bit βρίσκονται στον καταχωρητή PCL και έχουμε άμεση πρόσβαση σε αυτά για εγγραφή και ανάγνωση.

Σχήμα 4.12 *Μετρητής Προγράμματος PC*

Τα υψηλής αξίας δυαδικά ψηφία περιέχονται στον καταχωρητή PCH αλλά δεν είναι προσβάσιμα για ανάγνωση παρά μόνο για εγγραφή. Οι αλλαγές στον PCH γίνονται έμμεσα μέσω του καταχωρητή PCLATH (Program Counter Latch High). Χρησιμοποιούνται μόνο τα 5 πρώτα bit του PCLATH, ενώ υπόλοιπα 3 bit του παραμένουν πάντα μηδενικά.

Εξετάζουμε τώρα τους διαφορετικούς τρόπους με τους οποίους αλλάζει το περιεχόμενο του μετρητή προγράμματος PC:

- Στην περίπτωση του επόμενου σχήματος (σχήμα 4.13) έχουμε απευθείας αναφορά με προορισμό τον καταχωρητή PCL.

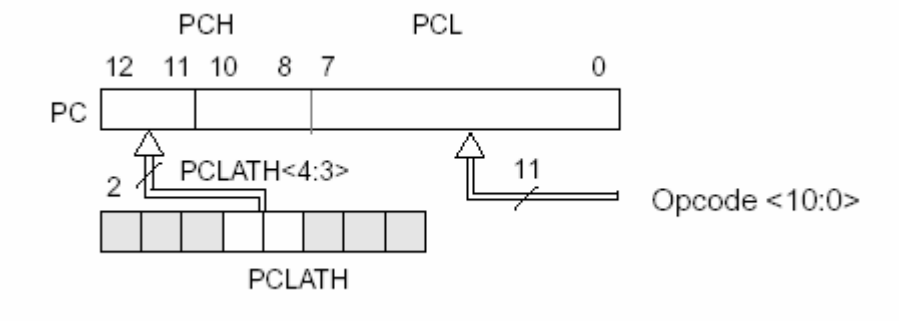


Σχήμα 4.13 *Απευθείας αναφορά στα 8 χαμηλής αξίας bit του μετρητή προγράμματος*

Το όρισμα που έρχεται από την αριθμητική και λογική μονάδα, ALU, τοποθετείται απευθείας στο κομμάτι PCL του μετρητή προγράμματος ενώ τα υπόλοιπα 5 bit του μετρητή προγράμματος, δηλαδή το τμήμα PCH, φορτώνονται από τον καταχωρητή PCLATH. Για να εκτελέσουμε, για παράδειγμα, μια εντολή άλματος με μεταβλητό όρισμα διεύθυνσης αρκεί να προσθέτουμε την επιθυμητή τιμή (offset)

στον καταχωρητή PCL. Πρέπει να προσέξουμε ότι το μέγεθος του PCL είναι 256 bytes και ότι μετά την πρόσθεση ο PCH θα πάρει την τιμή του PCLATH.

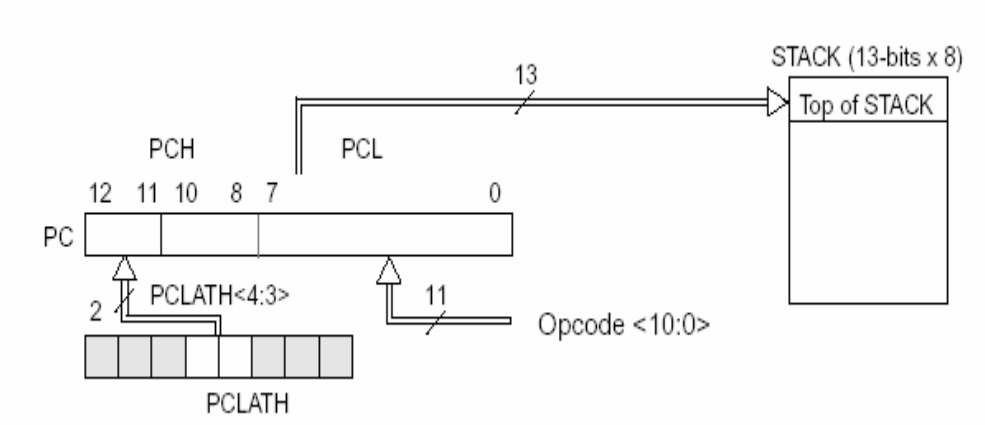
- Το επόμενο σχήμα (σχήμα 4.14) μας εξηγεί την περίπτωση όπου έχουμε μια εντολή άλματος, GOTO.



Σχήμα 4.14 *Αλλαγή περιεχομένου του μετρητή για την εντολή GOTO*

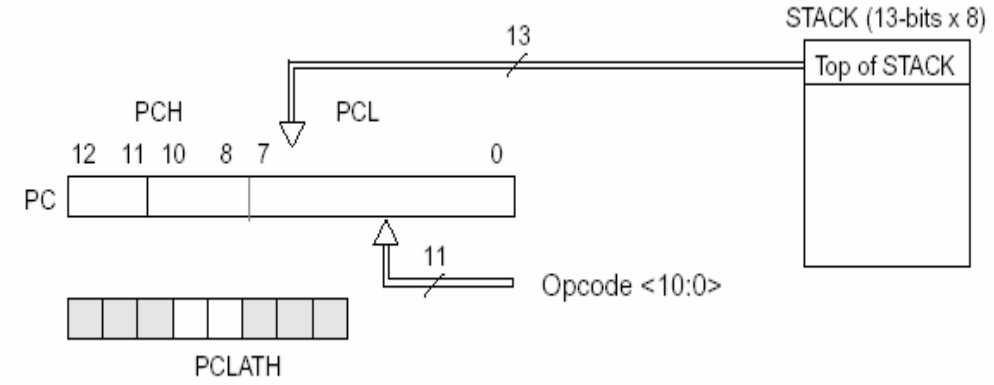
Το όρισμα της διεύθυνσης στην εντολή άλματος GOTO, έχει μήκος 11 δυαδικά ψηφία, τα οποία φορτώνονται απευθείας στα 11 μικρότερης αξίας δυαδικά ψηφία, <10:0>, του μετρητή προγράμματος PC. Τα υπόλοιπα 2 bit προέρχονται από τον καταχωρητή PCLATH.

- Η εντολή κλήσης υπορουτίνας, CALL λειτουργεί όπως ακριβώς και η εντολή άλματος GOTO μόνο που, επιπλέον, ολόκληρη η τιμή του καταχωρητή PC, σώζεται στην στοίβα.



Σχήμα 4.15 *Αλλαγή περιεχομένου του μετρητή προγράμματος κατά την εκτέλεση της εντολής CALL*

- Τέλος στην περίπτωση όπου έχουμε επιστροφή από υπορουτίνα ή διακοπή, ανακαλείται ολόκληρη η τιμή του μετρητή προγράμματος PC από την στοίβα όπου είχε αποθηκευθεί προσωρινά οπότε δεν χρειάζεται να φροντίσουμε να χειριστούμε τα bit του PCLATH για να επιστρέψουμε στην σωστή διεύθυνση.

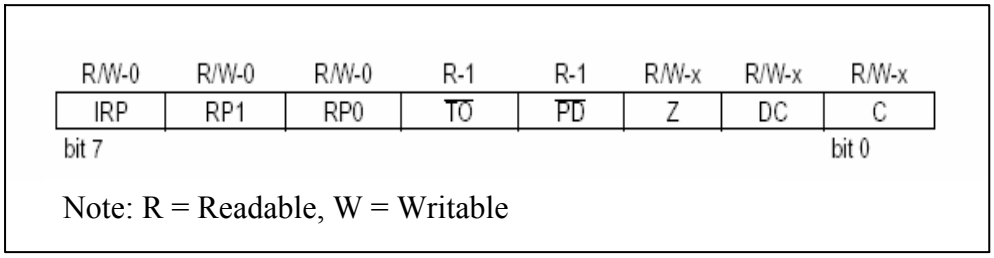


Σχήμα 4.16 *Αλλαγή περιεχομένου του μετρητή προγράμματος σε περίπτωση επιστροφής από υπορουτίνα*

Καταχωρητής κατάστασης (Status Register)

Ο καταχωρητής κατάστασης Status περιέχει τις σημαίες (flags), που μας δίνουν πληροφορίες για το αποτέλεσμα που προκύπτει από την Αριθμητική Λογική Μονάδα, καθώς επίσης και τα bit εκείνα που είναι απαραίτητα για την επιλογή του κατάλληλου

τμήματος της μνήμης κατά την προσπέλασή της. Επιπρόσθετα, τα bit 3 και 4 του Status register δεν μπορούν να εγγραφούν από τον χρήστη, παρά μόνο να διαβαστούν. Συνεπώς, η εγγραφή στη μνήμη με προορισμό τον Status register μπορεί να μην έχει το αποτέλεσμα που αναμένεται. Ειδικότερα, τα bit που αποτελούν τον καταχωρητή εργασίας φαίνονται στο παρακάτω σχήμα (Σχήμα 4.17).



Σχήμα 4.17 *Τα bit του καταχωρητή κατάστασης (STATUS)*

Επεξήγηση των bit του Status Register:

- Bit 7 : **IRP:** Bit για την επιλογή του τμήματος μνήμης (για έμμεση προσπέλαση)
  1 = Bank 2, 3 (100h - 1FFh)
  0 = Bank 0, 1 (00h - FFh)

- Bit 6:5 : **RP1:RP0**: Bit για την επιλογή του τμήματος μνήμης (για άμεση προσπέλαση)
  11 = Bank 3 (180h - 1FFh)
  10 = Bank 2 (100h - 17Fh)
  01 = Bank 1 (80h - FFh)
  00 = Bank 0 (00h - 7Fh)

- Bit 4 : **TO**: Time-out bit
  1 = στο άνοιγμα ή κατά τις εντολές CLRWDT και SLEEP
  0 = Όταν ο μετρητής Watchdog Timer υπερχειλίσει (Overflow)

- Bit 3 : **PD**: Power-down bit

  1 = στο άνοιγμα ή κατά την εντολή CLRWDT

  0 = κατά την εκτέλεση της εντολής SLEEP

- Bit 2 : **Z**: Bit μηδενισμού

  1 = το αποτέλεσμα της αριθμητικής ή λογικής πράξης είναι 0

  0 = το αποτέλεσμα της αριθμητικής ή λογικής πράξης δεν είναι 0

- Bit 1 : **DC**: Bit ενδιάμεσου κρατουμένου / δανεισμού

  1 = Δημιουργήθηκε κρατούμενο από το 4^o στο 5^o bit.

  0 = Δεν δημιουργήθηκε κρατούμενο από το 4^o στο 5^o bit.

- Bit 0 : **C**: Bit κρατουμένου / δανεισμού

  1 = Δημιουργήθηκε κρατούμενο από το 8^o στο 9^o bit.

  0 = Δεν δημιουργήθηκε κρατούμενο από το 8^o στο 9^o bit.

Η σημαία κρατουμένου (C), γίνεται ένα (1) εάν το αποτέλεσμα της πρόσθεσης είχε κρατούμενο στο 8^o bit, διαφορετικά είναι μηδέν. Στην περίπτωση της αφαίρεσης εάν το αποτέλεσμα που προκύπτει είναι θετικό ή μηδέν, δηλαδή δεν έχουμε την ανάγκη δανεισμού από το 9^o bit, τότε η σημαία κρατουμένου είναι ένα (1). Εάν έχουμε δανεισμό, δηλαδή εάν ο αφαιρετέος είναι μεγαλύτερος από το μειωτέο, τότε η σημαία κρατουμένου γίνεται μηδέν.

Η σημαία ενδιαμέσου κρατουμένου (DC), γίνεται ένα εάν από το αποτέλεσμα της πρόσθεσης προκύπτει κρατούμενο από το 4ο στο 5ο bit, ενώ και πάλι στην περίπτωση της αφαίρεσης γίνεται μηδέν στην περίπτωση που έχουμε ανάγκη δανεισμού από το 5ο bit.

Η σημαία μηδενισμού (Z) μας δείχνει εάν το αποτέλεσμα μιας εντολής είναι μηδενικό. Ιδιαίτερη προσοχή πρέπει να δοθεί στο ότι κάποιες εντολές που αλλάζουν τα περιεχόμενα ενός καταχωρητή δεν επηρεάζουν απαραίτητα τη σημαία αυτή.

Ιδιαίτερη προσοχή απαιτείται στην περίπτωση εκτέλεσης εντολών με προορισμό τον καταχωρητή STATUS, όταν αυτές επηρεάζουν τις σημαίες. Για παράδειγμα η εντολή clrf STATUS που μηδενίζει τον καταχωρητή έχει ως αποτέλεσμα μετά την εκτέλεση της ο καταχωρητής STATUS να περιέχει την δυαδική τιμή ’00000100’

αφού μόλις μηδενιστεί ο καταχωρητής στη συνέχεια τίθεται η σημαία μηδενισμού (Z=1) που βρίσκεται στον ίδιο.

Θ. 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 (**U**niversal **S**ynchronous **A**synchronous **R**eceiver **T**ransmitter, USART)

Οι σημαντικότερες διακοπές είναι:

- Εξωτερική διακοπή στον ακροδέκτη PB0/INT (external interrupt)
- Διακοπή από υπερχείλιση του χρονιστή timer0 (TMR0 Overflow Interrupt)
- Διακοπή όταν αλλάζει στάθμη ένας από τους ακροδέκτες PORTB<7:4> (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)

Στις επόμενες παραγράφους θα γίνει αναφορά σε διακοπές οι οποίες σχετίζονται με περιφερειακά τα οποία χρησιμοποιούμε στη δική μας σχεδίαση.

Γενική Αρχιτεκτονική των Διακοπών

Η κάθε διακοπή ελέγχεται από δύο bit, το *intname*IE (Interrupt Enable bit) και το *intname*IF (Interrupt Flag Bit). Το *intname*IE bit ενεργοποιεί την αντίστοιχη διακοπή ενώ το *intname*IF σηματοδοτεί ότι υπάρχει διακοπή προς εξυπηρέτηση.

Το bit *intname*IF γίνεται ένα (1) ανεξάρτητα, με το αν είναι ενεργοποιημένη η αντίστοιχη διακοπή ή όχι, δηλαδή αν το *intname*IE είναι ένα (1) ή μηδέν (0). Το *intname*IF 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). Η λογική των διακοπών φαίνεται στο σχήμα ?.

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.

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GIE	PEIE (3)	T0IE	INTE (2)	RBIE (1, 2)	T0IF	INTF (2)	RBIF (1,2)
bit 7							bit 0

Note: R = Readable , W = Writable

Σχήμα 4.18 *Καταχωρητής ελέγχου διακοπών (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).

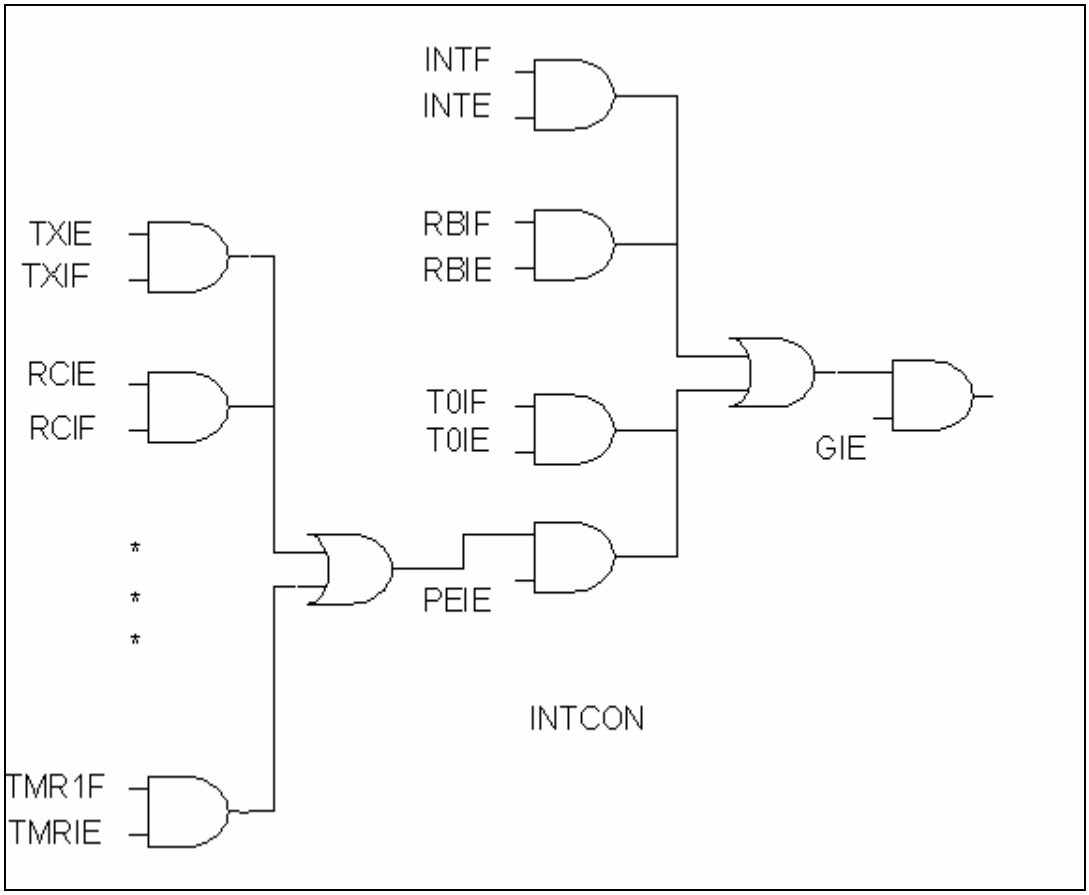
Το **T0IE** ενεργοποιεί την διακοπή του Timer 0.

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 *intname*IF, 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.



Σχήμα 4.19 *Λογική των διακοπών (Interrupts)*

Το bit PEIE (Peripheral Interrupt Enable bit), όπως φαίνεται από το σχήμα 4.19, ενεργοποιεί όλες εκείνες τις διακοπές που τα δύο bit ελέγχου τους *intname*IE και *intname*IF δεν περιέχονται στον καταχωρητή INTCON, αλλά στον καταχωρητή PIR1. Για παράδειγμα για να προκληθεί μια διακοπή από την *USART* θα πρέπει εκτός από το αντίστοιχο bit ελέγχου TXIE ή RCIE να είναι ένα (1) και τα bit PEIE και GIE.

Οι καταχωρητές PIE & PIR

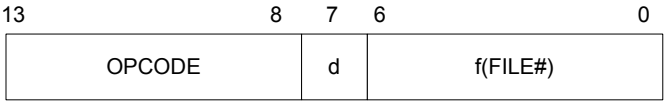
Ανάλογα με τα περιφερειακά που περιέχονται σε ένα μικροελεγκτή της οικογένειας PIC μπορούμε να έχουμε ένα ή και παραπάνω καταχωρητές ενεργοποίησης διακοπών **PIE** και ελέγχου διακοπών **PIR**. Κάθε bit του καταχωρητή **PIE** σχετίζεται με την ενεργοποίηση μιας διακοπής ενός περιφερειακού ενώ ο καταχωρητής **PIR** μας δείχνει ποιες διακοπές περιμένουν να εξυπηρετηθούν. Με άλλα λόγια ο **PIE** περιέχει τις σημαίες ενεργοποίησης των διακοπών *intname*IE, ενώ ο **PIR** τις σημαίες ένδειξης διακοπής *intname*IF

I. Εντολές (Instruction Set)

Γενική μορφή εντολών

Στην οικογένεια των PIC διακρίνουμε κυρίως 4 κατηγορίες εντολών. Το μέγεθος του κωδικού μιας εντολής κυμαίνεται από 3-bit έως 6-bit. Το σύνολο των εντολών στην περίπτωση των PIC της συγκεκριμένης κατηγορίας είναι 35. Το κομμάτι του κωδικού εντολής (Opcode), περιέχει τον κωδικό της εντολής ενώ τα υπόλοιπα κομμάτια περιέχουν το σύνολο των πληροφοριών για την εκτέλεση της εντολής. Στο επόμενο σχήμα (σχήμα 4.20) βλέπουμε τις κατηγορίες εντολών:

❖ Εντολές επεξεργασίας byte (byte-oriented):

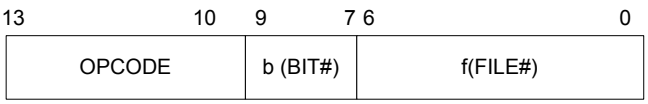


d=0 εάν ο αποδέκτης των δεδομένων είναι ο καταχωρητής W

d=1 εάν ο αποδέκτης των δεδομένων είναι ο καταχωρητής f

f=ο καταχωρητής από τον οποίο παίρνουμε ή και στέλνουμε δεδομένα

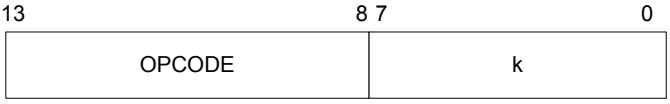
❖ Εντολές επεξεργασίας δυαδικών ψηφίων (bit-oriented):



b=σε πιο bit αναφερόμαστε

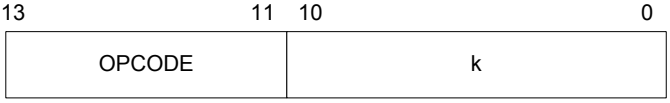
f=ο καταχωρητής από τον οποίο παίρνουμε ή και στέλνουμε δεδομένα

❖ Υπόλοιπες εντολές:



k=απευθείας (άμεσο) δεδομένο (literal)

❖ Εντολές άλματος (CALL, GOTO):



k=απευθείας (άμεσο) δεδομένο (literal)

Σχήμα 4.20 *Κατηγορίες Εντολών του PIC*

Οι εντολές

Θα μιλήσουμε στη συνέχεια ξεχωριστά για κάθε μια από τις εντολές και την λειτουργία που επιτελεί.

Στον πίνακα 4.2 παρουσιάζονται κάποιες συμβάσεις που χρησιμοποιούνται στην περιγραφή των εντολών.

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register (0 to 7)
k	Literal field, constant data or label (may be either an 8-bit or an 11-bit value)
x	Don't care (0 or 1) The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f.
dest	Destination either the W register or the specified register file location
label	Label name
TOS	Top of Stack
PC	Program Counter
PCLATH	Program Counter High Latch
GIE	Global Interrupt Enable bit
WDT	Watchdog Timer
$\overline{TO}$	Time-out bit
$\overline{PD}$	Power-down bit
[ ]	Optional
( )	Contents
→	Assigned to
< >	Register bit field
∈	In the set of
italics	User defined term (font is courier)

Πίνακας 4.2 *Συμβάσεις περιγραφής εντολών (Instruction Description Conventions)*

ADDLW : Πρόσθεση απευθείας δεδομένου (literal) με τον καταχωρητή εργασίας W

Σύνταξη: [ετικέττα] ADDLW k

Τελεστές: $0 \leq k \leq 255$

Λειτουργία: (W) + k → W

Ενημέρωση σημαιών: C,DC,Z

Κωδικοποίηση: 11 – 111x – kkkk - kkkk

Περιγραφή: Στο περιεχόμενο του καταχωρητή W, προστίθεται η 8-bit τιμή k και το αποτέλεσμα αποθηκεύεται πάλι στον καταχωρητή W.

ADDWF : Πρόσθεση του καταχωρητή f με τον καταχωρητή εργασίας W

Σύνταξη: [ετικέττα] ADDWF f,d

Τελεστές: $0 \leq f \leq 127$, $d \in [0, 1]$

Λειτουργία: (W) + f → destination

Ενημέρωση σημαιών: C, DC, Z

Κωδικοποίηση: 00 – 0111 – dfff - ffff

Περιγραφή: Το περιεχόμενο του καταχωρητή W αθροίζεται με το περιεχόμενο του καταχωρητή με διεύθυνση f (0≤f≤127) του παρόντος τμήματος (bank) και το αποτέλεσμα αποθηκεύεται είτε στον καταχωρητή W, αν το όρισμα d είναι μηδέν είτε στον καταχωρητή f αν το όρισμα είναι ένα.

ANDLW : Λογικό AND απευθείας δεδομένου με τον καταχωρητή εργασίας W

Σύνταξη: [ετικέττα] ANDLW k

Τελεστές: $0 \leq k \leq 255$

Λειτουργία: (W).AND. (k) → W

Ενημέρωση σημαιών: Z

Κωδικοποίηση: 11 – 1001 – kkkk - kkkk

Περιγραφή: Κάθε bit του καταχωρητή W γίνεται λογικό AND με κάθε bit από την 8-bit τιμή k και το αποτέλεσμα αποθηκεύεται στον καταχωρητή W.

ANDWF : Λογικό AND του καταχωρητή f με τον καταχωρητή εργασίας W

Σύνταξη: [ετικέττα] ANDWF f,d

Τελεστές: $0 \leq f \leq 127$, $d \in [0, 1]$

Λειτουργία: (W).AND. (f) → destination

Ενημέρωση σημαιών: Z

Κωδικοποίηση: 00 – 0101 – dfff - ffff

Περιγραφή: Κάθε bit του καταχωρητή W γίνεται λογικό AND με κάθε bit του καταχωρητή με διεύθυνση f (0≤f≤127) του παρόντος τμήματος (bank) και το αποτέλεσμα αποθηκεύεται είτε στον καταχωρητή W αν το όρισμα d είναι μηδέν είτε στον καταχωρητή f αν το όρισμα είναι ένα.

BCF : Μηδενισμός ενός bit του καταχωρητή f

Σύνταξη: [ετικέττα] BCF f, b

Τελεστές: $0 \leq f \leq 127$, $0 \leq b \leq 7$

Λειτουργία: 0 → f<b>

Ενημέρωση σημαιών: Καμία

Κωδικοποίηση: 01 – 00bb – bfff - ffff

Περιγραφή: Η εντολή αυτή μηδενίζει το bit με αριθμό b ($0 \leq b \leq 7$) του καταχωρητή f ($0 \leq f \leq 127$) του παρόντος τμήματος (bank). Ο μηδενισμός δεν γίνεται απευθείας αλλά ο πυρήνας διαβάζει πρώτα όλο το περιεχόμενο του καταχωρητή, μηδενίζει το συγκεκριμένο bit και στη συνέχεια ξαναγράφει τη νέα τιμή στον καταχωρητή (read-write-modify).

BSF : Θέση ένα bit του καταχωρητή f

Σύνταξη: [ετικέττα] BSF f,b

Τελεστές: $0 \leq f \leq 127$, $0 \leq b \leq 7$

Λειτουργία: 1 → f<b>

Ενημέρωση σημαιών: Καμία

Κωδικοποίηση: 01 – 01bb – bfff - ffff

Περιγραφή: Η εντολή αυτή θέτει το bit με αριθμό b ($0 \leq b \leq 7$), του καταχωρητή f ($0 \leq f \leq 127$) του παρόντος τμήματος (bank). Η διαδικασία όπως και στην περίπτωση που μηδενίζουμε ένα bit δεν γίνεται απευθείας αλλά ο πυρήνας διαβάζει πρώτα όλο το περιεχόμενο του καταχωρητή, θέτει το συγκεκριμένο bit και στη συνέχεια ξαναγράφει τη νέα τιμή στον καταχωρητή (read-write-modify).

BTFSC : Έλεγχος ενός bit του καταχωρητή f, και όχι εκτέλεση της επόμενης εντολής στην περίπτωση που το bit είναι μηδέν

Σύνταξη: [ετικέττα] BTFSC f,b

Τελεστές: $0 \leq f \leq 127$, $0 \leq b \leq 7$

Λειτουργία: skip if f<b> = 0

Ενημέρωση σημαιών: Καμία

Κωδικοποίηση: 01 – 10bb – bfff - ffff

Περιγραφή: Η εντολή αυτή ελέγχει το bit με αριθμό b (0≤b≤7) του καταχωρητή f (0≤f≤127) του παρόντος τμήματος (bank). Στην περίπτωση που το bit αυτό είναι μηδενικό τότε δεν εκτελείται η επόμενη εντολή αλλά η μεθεπόμενη. Επειδή σε αυτή τη περίπτωση έχουμε άλμα (παράκαμψη μιας εντολής) η εντολή διαρκεί δύο κύκλους. Εάν το προς έλεγχο bit είναι ένα τότε δεν γίνεται καμία αλλαγή στην ροή του προγράμματος και η εντολή διαρκεί έναν κύκλο.

BTFSS : Έλεγχος ενός bit του καταχωρητή f και όχι εκτέλεση της επόμενης εντολής στην περίπτωση που το bit είναι ένα

Σύνταξη: [ετικέττα] BTFSS f,b

Τελεστές: $0 \le f \le 127$, $0 \le b \le 7$

Λειτουργία: skip if f<b> = 1

Ενημέρωση σημαιών: Καμία

Κωδικοποίηση: 01 – 11bb – bfff - ffff

Περιγραφή: Η εντολή αυτή ελέγχει το bit με αριθμό b (0≤b≤7) του καταχωρητή f (0≤f≤127) του παρόντος τμήματος (bank). Στην περίπτωση που το bit αυτό είναι ένα τότε δεν εκτελείται η επόμενη εντολή αλλά η μεθεπόμενη. Επειδή σε αυτή τη περίπτωση έχουμε άλμα η εντολή διαρκεί δύο κύκλους. Εάν το προς έλεγχο bit είναι μηδέν τότε δεν γίνεται καμία αλλαγή στην ροή του προγράμματος και η εντολή διαρκεί έναν κύκλο.

CALL : Κλήση υπορουτίνας

Σύνταξη: [ετικέττα] CALL k

Τελεστές: $0 \le k \le 2047$

Λειτουργία: (PC) + 1 → TOS, k → PC<10:0>, (PCLATH<4:3>) → PC<12:11>

Ενημέρωση σημαιών: Καμία

Κωδικοποίηση: 10 – 0kkk – kkkk - kkkk

Περιγραφή: Με την εντολή CALL γίνεται κλήση σε υπορουτίνα με διεύθυνση k. Όπως έχουμε δει το όρισμα k έχει μήκος 11-bit, συνεπώς μπορεί να έχει οποιαδήποτε τιμή από 0 έως 2047(2k), δηλαδή μπορεί να είναι οποιαδήποτε διεύθυνση μέσα σε μια σελίδα (page). Η επιλογή της σελίδας γίνεται από τα bit <4:3> του καταχωρητή PCLATH που φορτώνονται αυτόματα στα υψηλότερης αξίας bit<12:11> του μετρητή προγράμματος PC κατά την εκτέλεση της εντολής. Εάν πρόκειται να κάνουμε άλμα σε κάποια καινούργια σελίδα θα πρέπει πάντα πριν από την εντολή CALL να έχουμε βάλει την κατάλληλη τιμή στον καταχωρητή PCLATH. Και τα 13-bit της διεύθυνσης επιστροφής που είναι η διεύθυνση της επόμενης εντολής μετά την εντολή CALL, προωθούνται στην στοίβα. Συνεπώς δεν χρειάζεται καμία απολύτως αλλαγή του καταχωρητή PCLATH κατά την επιστροφή από την υπορουτίνα.

CLRF :Μηδενισμός του καταχωρητή f

Σύνταξη: [ετικέττα] CLRF f

Τελεστές: $0 \leq f \leq 127$

Λειτουργία: 00h → f , 1 → Z

Ενημέρωση σημαιών: Z=1

Κωδικοποίηση: 00 – 0001 – 1fff - ffff

Περιγραφή: Το περιεχόμενο του καταχωρητή f ($0 \leq f \leq 127$) του παρόντος τμήματος (bank) γίνεται μηδέν και η σημαία Z γίνεται ένα.

CLRW : Μηδενισμός του καταχωρητή W

Σύνταξη: [ετικέττα] CLRW

Τελεστές: Κανένας

Λειτουργία: 00h → W , 1 → Z

Ενημέρωση σημαιών: Z=1

Κωδικοποίηση: 00 – 0001 – 0xxx - xxxx

Περιγραφή: Το περιεχόμενο του καταχωρητή W γίνεται μηδέν και η σημαία Z γίνεται ένα.

CLRWDT : Αρχικοποίηση του WDT (Watch Dog Timer)

Σύνταξη: [ετικέττα] CLRWDT

Τελεστές: Κανένας

Λειτουργία: 00h → WDT , 0 → WDT prescaler count , 1 → $\overline{TO}$, 1 → $\overline{PD}$

Ενημέρωση σημαιών: $\overline{TO}$, $\overline{PD}$

Κωδικοποίηση: 00 – 0000 – 0110 - 0100

Περιγραφή: Το περιεχόμενο του καταχωρητή WDT και του prescaler του WDT γίνονται μηδέν. Οι σημαίες $\overline{TO}$, $\overline{PD}$ γίνονται ένα.

COMF : Συμπλήρωμα του καταχωρητή f ως πρός ένα

Σύνταξη: [ετικέττα] COMF f,d

Τελεστές: $0 \le f \le 127$, $d \in [0, 1]$

Λειτουργία: ($\overline{f}$) → destination

Ενημέρωση σημαιών: Z

Κωδικοποίηση: 00 – 1001 – dfff - ffff

Περιγραφή: Το περιεχόμενο του καταχωρητή f ($0 \le f \le 127$) του παρόντος τμήματος (bank) συμπληρώνεται ως προς ένα. Το αποτέλεσμα επιστρέφει στον καταχωρητή αν το d είναι ένα (1) διαφορετικά αν η τιμή του d είναι μηδέν (0) το αποτέλεσμα αποθηκεύεται στον καταχωρητή W.

DECF : Μείωση του καταχωρητή f

Σύνταξη: [ετικέττα] DECF f,d

Τελεστές: $0 \le f \le 127$, $d \in [0, 1]$

Λειτουργία: (f) - 1 → destination

Ενημέρωση σημαιών: Z

Κωδικοποίηση: 00 – 0011 – dfff - ffff

Περιγραφή: Το περιεχόμενο του καταχωρητή καταχωρητή f (0≤f≤127) του παρόντος τμήματος (bank) μειώνεται κατά ένα. Το αποτέλεσμα αποθηκεύεται είτε στον καταχωρητή f εάν d=1, είτε στον καταχωρητή W εάν d=0.

DECFSZ : Μείωση του καταχωρητή f και στην περίπτωση που το αποτέλεσμα είναι μηδέν όχι εκτέλεση της επόμενης εντολής

Σύνταξη: [ετικέττα] DECFSZ f,d

Τελεστές: $0 \leq f \leq 127$, $d \in [0, 1]$

Λειτουργία: (f) - 1 → destination , skip if result = 0

Ενημέρωση σημαιών: Καμία

Κωδικοποίηση: 00 – 1011 – dfff - ffff

Περιγραφή: Το περιεχόμενο του καταχωρητή f (0≤f≤127) του παρόντος τμήματος (bank) μειώνεται κατά ένα. Το αποτέλεσμα αποθηκεύεται είτε στον καταχωρητή f εάν d=1, είτε στον καταχωρητή W εάν d=0. Στην περίπτωση που το αποτέλεσμα είναι μηδενικό δεν εκτελείται η επόμενη εντολή. Προφανώς στην περίπτωση του άλματος η εντολή διαρκεί δύο κύκλους. Ιδιαίτερη προσοχή χρειάζεται το γεγονός ότι δεν ενημερώνεται η σημαία μηδενισμού Z ακόμα και στην περίπτωση μηδενισμού.

GOTO :Άλμα χωρίς συνθήκη

Σύνταξη: [ετικέττα] GOTO k

Τελεστές: $0 \leq k$ 2047

Λειτουργία: k → PC<10:0>, (PCLATH<4:3>) → PC<12:11>

Ενημέρωση σημαιών: Καμία

Κωδικοποίηση: 10 – 1kkk – kkkk - kkkk

Περιγραφή: Με την εντολή GOTO γίνεται άλμα στη διεύθυνση k. Το όρισμα k έχει μήκος 11-bit, συνεπώς μπορεί να έχει οποιαδήποτε τιμή από 0 έως 2047(2k), δηλαδή μπορεί να είναι οποιαδήποτε διεύθυνση μέσα σε μια σελίδα (page). Η επιλογή της σελίδας γίνεται από τα bit <4:3> του καταχωρητή PCLATH που φορτώνονται αυτόματα στα υψηλότερης αξίας bit<12:11> του μετρητή προγράμματος PC κατά την εκτέλεση της

εντολής. Αν πρόκειται να κάνουμε άλμα σε κάποια καινούργια σελίδα θα πρέπει πάντα πριν από την εντολή GOTO να έχουμε βάλει την κατάλληλη τιμή στον καταχωρητή PCLATH.

INCF : Αύξηση του καταχωρητή f

Σύνταξη: [ετικέττα] INCF f,d

Τελεστές: $0 \leq f \leq 127$, $d \in [0, 1]$

Λειτουργία: (f) + 1 → destination

Ενημέρωση σημαιών: Z

Κωδικοποίηση: 00 – 1010 – dfff - ffff

Περιγραφή: Το περιεχόμενο του καταχωρητή f ($0 \leq f \leq 127$) του παρόντος τμήματος (bank) αυξάνεται κατά ένα. Το αποτέλεσμα αποθηκεύεται είτε στον καταχωρητή f αν d=1, είτε στον καταχωρητή W αν d=0.

INCFSZ : Αύξηση του καταχωρητή f και στην περίπτωση που το αποτέλεσμα είναι μηδέν όχι εκτέλεση της επόμενης εντολής

Σύνταξη: [ετικέττα] INCFSZ f,d

Τελεστές: $0 \leq f \leq 127$, $d \in [0, 1]$

Λειτουργία: (f) + 1 → destination , skip if result = 0

Ενημέρωση σημαιών: Καμία

Κωδικοποίηση: 00 – 1111 – dfff - ffff

Περιγραφή: Το περιεχόμενο του καταχωρητή f ($0 \leq f \leq 127$) του παρόντος τμήματος (bank) αυξάνεται κατά ένα. Το αποτέλεσμα αποθηκεύεται είτε στον καταχωρητή f αν d=1, είτε στον καταχωρητή W αν d=0. Στην περίπτωση που το αποτέλεσμα είναι μηδέν δεν εκτελείται η επόμενη εντολή. Στην περίπτωση του άλματος η εντολή διαρκεί δύο κύκλους. Ιδιαίτερη προσοχή χρειάζεται το γεγονός ότι δεν ενημερώνεται η σημαία μηδενισμού Z ακόμα και στην περίπτωση μηδενισμού.

IORLW : Λογικό OR απευθείας δεδομένου με τον καταχωρητή εργασίας W

Σύνταξη: [ετικέττα] IORLW k

Τελεστές: $0 \leq k \leq 255$

Λειτουργία: (W).OR.k → W

Ενημέρωση σημαιών: Z

Κωδικοποίηση: 11 – 1000 – kkkk - kkkk

Περιγραφή: Κάθε bit του καταχωρητή W γίνεται λογικό OR με κάθε bit από την 8-bit τιμή k και το αποτέλεσμα αποθηκεύεται στον καταχωρητή W.

IORWF : Λογικό OR του καταχωρητή f με τον καταχωρητή εργασίας W

Σύνταξη: [ετικέττα] IORWF f,d

Τελεστές: $0 \leq f \leq 127$, $d \in [0, 1]$

Λειτουργία: (W).OR.f → W

Ενημέρωση σημαιών: Z

Κωδικοποίηση: 00 – 0100 – dfff - ffff

Περιγραφή: Κάθε bit του καταχωρητή W γίνεται λογικό OR με κάθε bit του καταχωρητή με διεύθυνση f ($0 \leq f \leq 127$) του παρόντος τμήματος (bank) και το αποτέλεσμα αποθηκεύεται είτε στον καταχωρητή W αν το όρισμα d είναι μηδέν είτε στον καταχωρητή f αν το όρισμα είναι ένα.

MOVLW : Μεταφορά του απευθείας δεδομένου στον καταχωρητή εργασίας W

Σύνταξη: [ετικέττα] MOVLW k

Τελεστές: $0 \leq k \leq 255$

Λειτουργία: k → W

Ενημέρωση σημαιών: Καμία

Κωδικοποίηση: 11 – 00xx – kkkk - kkkk

Περιγραφή: Ο καταχωρητή W παίρνει την 8-bit τιμή k. Ενημέρωση σημαιών δεν γίνεται.

MOVF : Μεταφορά του περιεχομένου του καταχωρητή f

Σύνταξη: [ετικέττα] MOVF f,d

Τελεστές: $0 \leq f \leq 127$, $d \in [0, 1]$

Λειτουργία: f → destination

Ενημέρωση σημαιών: Z

Κωδικοποίηση: 00 – 1000 – dfff - ffff

Περιγραφή: Το περιεχόμενο του καταχωρητή με διεύθυνση f (0≤f≤127) του παρόντος τμήματος (bank) μεταφέρεται είτε στον καταχωρητή W αν το όρισμα d είναι μηδέν είτε στον καταχωρητή f αν το όρισμα είναι ένα. Η επιστροφή του περιεχόμενου στον ίδιο καταχωρητή χρησιμοποιείται για να εξετάσουμε αν το περιεχόμενο του είναι μηδενικό αφού ενημερώνεται η σημαία Z.

MOVWF : Μεταφορά του περιεχομένου του καταχωρητή W στον καταχωρητή f

Σύνταξη: [ετικέττα] MOVWF f

Τελεστές: $0 \leq f \leq 127$

Λειτουργία: (W) → f

Ενημέρωση σημαιών: Καμία

Κωδικοποίηση: 00 – 0000 – 1fff - ffff

Περιγραφή: Το περιεχόμενο του καταχωρητή W μεταφέρεται στον καταχωρητή με διεύθυνση f (0≤f≤127) του παρόντος τμήματος (bank).

NOP :Αργία

Σύνταξη: [ετικέττα] NOP

Τελεστές: Κανένας

Λειτουργία: Καμία

Ενημέρωση σημαιών: Καμία

Κωδικοποίηση: 00 – 0000 – 0xx0 – 0000

Περιγραφή: Ο μικροελεγκτής δεν κάνει τίποτα σε αυτό τον κύκλο.

RETFIE : Επιστροφή από ρουτίνα εξυπηρέτησης διακοπών

Σύνταξη: [ετικέττα] RETFIE

Τελεστές: Κανένας

Λειτουργία: TOS → PC , 1 → GIE

Ενημέρωση σημαιών: Καμία

Κωδικοποίηση: 00 – 0000 – 0000 – 1001

Περιγραφή: Κατά την εκτέλεση της εντολής RETFIE, ανακαλείται από τη στοίβα ολόκληρη η διεύθυνση στο σημείο που είχε διακοπεί το πρόγραμμα και αποθηκεύεται στον μετρητή προγράμματος PC. Επιπλέον ενεργοποιούνται οι διακοπές θέτοντας το bit GIE (INTCON<7>). Η εντολή αυτή καθώς και όλες οι εντολές επιστροφής διαρκούν δύο κύκλους αφού αλλάζουν τα δεδομένα του μετρητή προγράμματος.

RETLW : Επιστροφή από ρουτίνα με ανάθεση άμεσου δεδομένου στον W

Σύνταξη: [ετικέττα] RETLW k

Τελεστές: $0 \leq k \leq 255$

Λειτουργία: k → W , TOS → PC

Ενημέρωση σημαιών: Καμία

Κωδικοποίηση: 11 – 01xx – kkkk – kkkk

Περιγραφή: Κατά την εκτέλεση της εντολής RETLW ανακαλείται από τη στοίβα ολόκληρη η διεύθυνση επιστροφής και αποθηκεύεται στον μετρητή προγράμματος PC. Παράλληλα ο καταχωρητής W παίρνει την τιμή k. Με την βοήθεια της εντολής αυτής υλοποιούνται οι πίνακες με σταθερές στη μνήμη προγράμματος και αυτό γιατί με την αρχιτεκτονική Harvard είναι αδύνατο να έχουμε άμεση πρόσβαση στη μνήμη προγράμματος.

RETURN : Επιστροφή από ρουτίνα

Σύνταξη: [ετικέττα] RETURN

Τελεστές: Κανένας

Λειτουργία: TOS → PC

Ενημέρωση σημαιών: Καμία

Κωδικοποίηση: 00 – 0000 – 0000 – 1000

Περιγραφή: Κατά την εκτέλεση της εντολής RETURN ανακαλείται από τη στοίβα ολόκληρη η διεύθυνση επιστροφής και αποθηκεύεται στον μετρητή προγράμματος PC.

RLF : Αριστερή Ολίσθηση του περιεχομένου του καταχωρητή f μέσω του κρατουμένου.

Σύνταξη: [ετικέττα] RLF f,d

Τελεστές: $0 \leq f \leq 127$, $d \in [0, 1]$

Λειτουργία: Βλέπε την περιγραφή

Ενημέρωση σημαιών: C

Κωδικοποίηση: 00 – 1101 – dfff – ffff

Περιγραφή: Το περιεχόμενο του καταχωρητή με διεύθυνση f ($0 \leq f \leq 127$) του παρόντος τμήματος (bank) ολισθαίνετε μια θέση αριστερά μέσω του κρατουμένου, ενώ η παλιά τιμή του κρατουμένου ανατροφοδοτείται στην αρχή. Το αποτέλεσμα επιστρέφει στον καταχωρητή αν το d είναι ένα διαφορετικά αν είναι μηδέν το αποτέλεσμα αποθηκεύεται στον καταχωρητή W.

RRF : Δεξιά Ολίσθηση του περιεχομένου του καταχωρητή f μέσω του κρατουμένου.

Σύνταξη: [ετικέττα] RRF f,d

Τελεστές: $0 \leq f \leq 127$, $d \in [0, 1]$

Λειτουργία: Βλέπε την περιγραφή

Ενημέρωση σημαιών: C

Κωδικοποίηση: 00 – 1100 – dfff – ffff

Περιγραφή: Το περιεχόμενο του καταχωρητή με διεύθυνση f ($0 \leq f \leq 127$) του παρόντος τμήματος (bank) ολισθαίνεται μια θέση δεξιά μέσω του κρατουμένου, ενώ η παλιά τιμή του κρατουμένου ανατροφοδοτείται στο υψηλότερο αξίας δυαδικό ψηφίο του καταχωρητή f.. Το αποτέλεσμα επιστρέφει στον καταχωρητή αν το d είναι ένα

διαφορετικά αν η τιμή του d είναι μηδέν το αποτέλεσμα αποθηκεύεται στον καταχωρητή W.

SLEEP : Κατάσταση "ύπνου"

Σύνταξη: [ετικέττα] SLEEP

Τελεστές: Κανένας

Λειτουργία: 00h → WDT , 0 → WDT prescaler count , 1 → $\overline{TO}$, 0 → $\overline{PD}$

Ενημέρωση σημαιών: $\overline{TO}$, $\overline{PD}$

Κωδικοποίηση: 00 – 0000 – 0110 – 0011

Περιγραφή: Το bit που σηματοδοτεί την κατάσταση χαμηλής κατανάλωσης ισχύος ($\overline{PD}$) μηδενίζεται, ενώ το bit που σηματοδοτεί τη λήξη χρόνου ($\overline{TO}$) τίθεται στο ένα. Ο καταχωρητής WDT και ο prescaler του μηδενίζονται. Ο μικροελεγκτής πέφτει σε κατάσταση «ύπνου» με τον ταλαντωτή να σταματά.

SUBLW : Αφαίρεση άμεσου δεδομένου με τον καταχωρητή εργασίας W

Σύνταξη: [ετικέττα] SUBLW k

Τελεστές: $0 \leq k \leq 255$

Λειτουργία: k – (W) → W

Ενημέρωση σημαιών: C, DC, Z

Κωδικοποίηση: 11 – 110x – kkkk – kkkk

Περιγραφή: Από το απευθείας δεδομένο k αφαιρείται το περιεχόμενο του καταχωρητή W και το αποτέλεσμα αποθηκεύεται στον καταχωρητή W.

SUBWF : Αφαίρεση του καταχωρητή εργασίας W από τον καταχωρητή f

Σύνταξη: [ετικέττα] SUBWF f,d

Τελεστές: $0 \leq f \leq 127$, $d \in [0, 1]$

Λειτουργία: f – (W) → destination

Ενημέρωση σημαιών: C, DC, Z

Κωδικοποίηση: 00 – 0010 – dfff – ffff

Περιγραφή: Από το περιεχόμενο του καταχωρητή με διεύθυνση f (0≤f≤127) του παρόντος τμήματος (bank) αφαιρείται το περιεχόμενο του καταχωρητή W και το αποτέλεσμα αποθηκεύεται είτε στον καταχωρητή W αν το όρισμα d είναι μηδέν είτε στον καταχωρητή f αν το όρισμα είναι ένα.

SWAPF : Αλλαγή θέσης ψηφίων του καταχωρητή f

Σύνταξη: [ετικέττα] SWAPF f,d

Τελεστές: $0 \leq f \leq 127$, $d \in [0, 1]$

Λειτουργία: (f<3:0>) → destination<7:4> , (f<7:4>) → destination<3:0>

Ενημέρωση σημαιών: Καμία

Κωδικοποίηση: 00 – 1110 – dfff – ffff

Περιγραφή: Τα υψηλότερης αξίας 4 bit του καταχωρητή με διεύθυνση f (0≤f≤127) του παρόντος τμήματος (bank) ανταλλάσσουν θέση με τα τέσσερα χαμηλότερης αξίας δυαδικά ψηφία του καταχωρητή. Το αποτέλεσμα αποθηκεύεται στον καταχωρητή W αν το όρισμα d είναι μηδέν ή στον καταχωρητή f αν το όρισμα είναι ένα.

XORLW : Λογικό XOR απευθείας δεδομένου με τον καταχωρητή εργασίας W

Σύνταξη: [ετικέττα] XORLW k

Τελεστές: $0 \leq k \leq 255$

Λειτουργία: (W).XOR.k → W

Ενημέρωση σημαιών: Z

Κωδικοποίηση: 11 – 1010 – kkkk – kkkk

Περιγραφή: Κάθε bit του καταχωρητή W γίνεται λογικό XOR με κάθε bit από την 8-bit τιμή k και το αποτέλεσμα αποθηκεύεται στον καταχωρητή W.

XORWF : Λογικό XOR του καταχωρητή f με τον καταχωρητή εργασίας W

Σύνταξη: [ετικέττα] XORWF f,d

Τελεστές: $0 \leq f \leq 127$, $d \in [0, 1]$

Λειτουργία: (W).XOR.(f) → W

Ενημέρωση σημαιών: Z

Κωδικοποίηση: 00 – 0110 – dfff – ffff

Περιγραφή: Κάθε bit του καταχωρητή W γίνεται λογικό XOR με κάθε bit του καταχωρητή με διεύθυνση f (0≤f≤127) του παρόντος τμήματος (bank) και το αποτέλεσμα αποθηκεύεται είτε στον καταχωρητή W αν το όρισμα d είναι μηδέν είτε στον καταχωρητή f αν το όρισμα είναι ένα.

4.4.3. Τα περιφερειακά του PIC

Α. 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 κατευθύνσεις 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[2] 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. Το σχήμα ? φαίνεται η γενική αρχιτεκτονική ενός ακροδέκτη μιας θύρας.

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

2

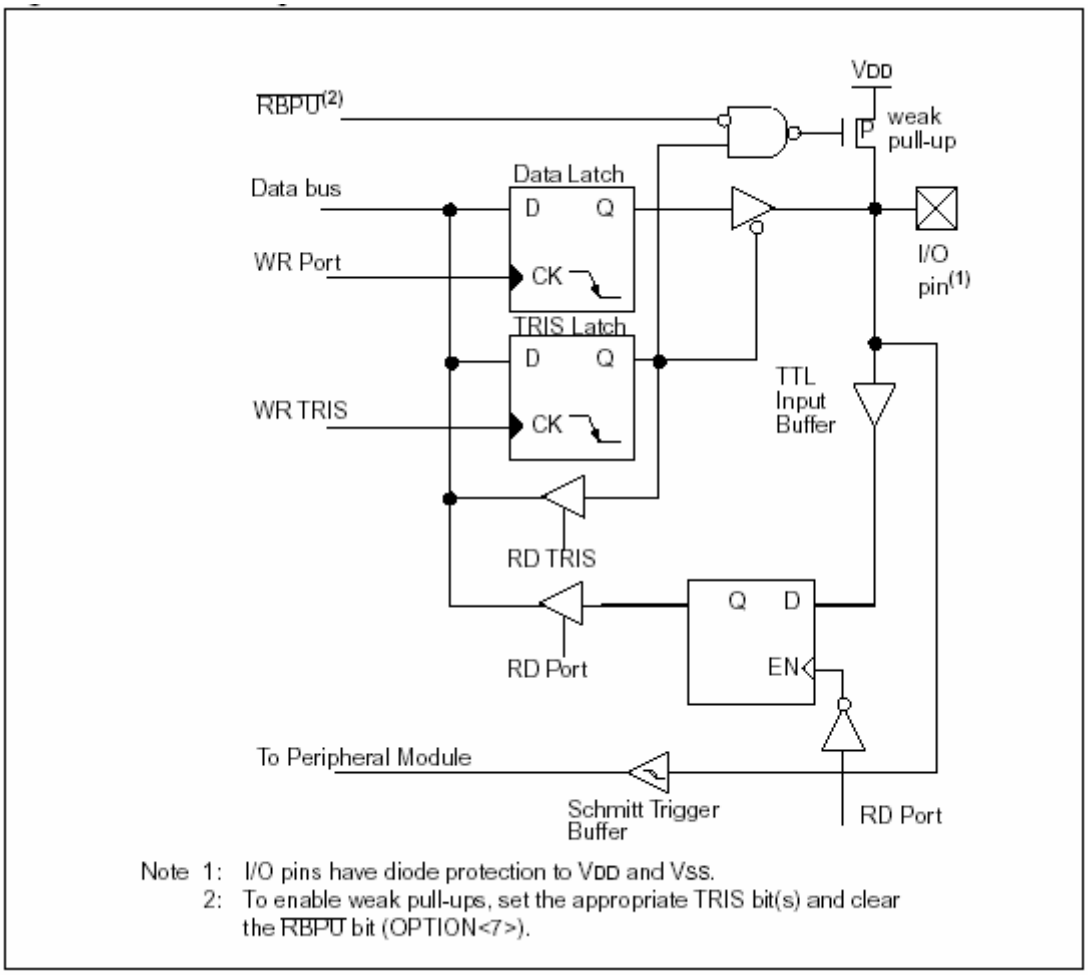
απαραίτητο ο καταχωρητής TRIS να είναι σωστά ρυθμισμένος για να λειτουργήσει το περιφερειακό.

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 διακόπτη. Μηδενίζοντας το bit $\overline{\text{RPBU}}$ (OPTION<7>) 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<0>). Η γενική αρχιτεκτονική των ακροδεκτών αυτών φαίνεται στο σχήμα 4.22.

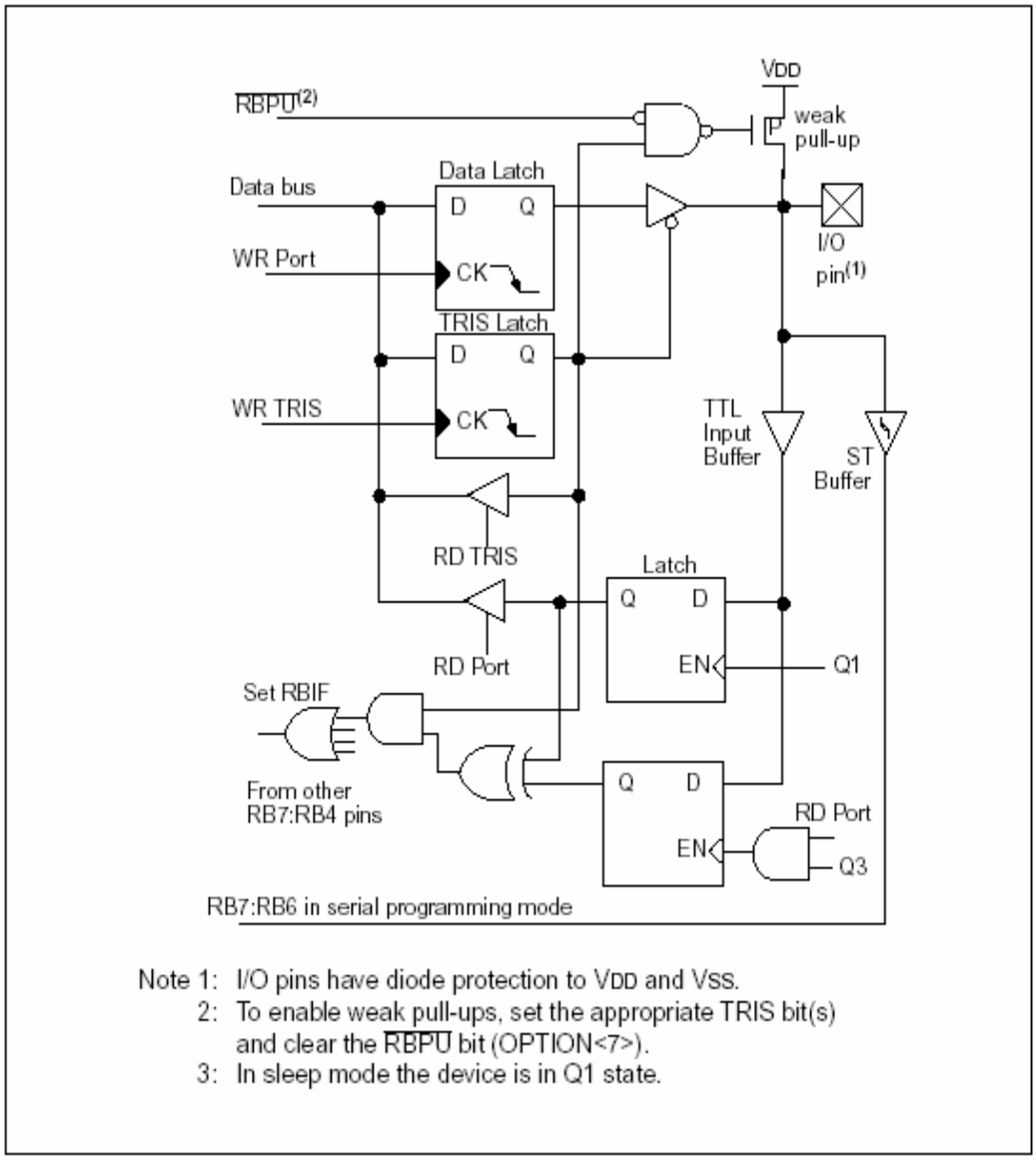
Η διακοπή αυτή μπορεί να επαναφέρει τη συσκευή από την κατάσταση SLEEP. Για να σταματήσουμε την παραγωγή της διακοπής θα πρέπει να διαβάσουμε ή να γράψουμε κατάλληλη τιμή στον καταχωρητή PORTB και στη συνέχεια να σβήσουμε τη σημαία RBIF. Με την παραπάνω διακοπή αλλά και με τη χρήση των εσωτερικών pull-up 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.



Σχήμα 4.21 *Αρχιτεκτονική μονάδας εισόδου – εξόδου (I/O) του PIC*

Διαδοχικές εντολές σε μια θύρα

Ένα τελευταίο θέμα που θα θίξουμε έχει να κάνει με την ταχύτητα της θύρας και το πόσο γρήγορα αυτή ανταποκρίνεται. Η πραγματική εγγραφή σε μια θύρα συμβαίνει στο τέλος ενός κύκλου εντολής ενώ όσο αναφορά το διάβασμα τα δεδομένα πρέπει να είναι έτοιμα στην αρχή του κύκλου εντολής. Γι' αυτό δεν θα πρέπει να διαβάζουμε την κατάσταση της θύρας απευθείας μετά από μια εγγραφή, γιατί μπορεί αυτή να μην έχει σταθεροποιηθεί και έτσι να διαβάσουμε την προηγούμενη κατάσταση αντί της νέας. Το πόσο γρήγορα ανταποκρίνεται η θύρα εξαρτάται από το φορτίο που έχει να οδηγήσει. Όσο πιο μεγάλο το φορτίο (μεγάλη χωρητικότητα) τόσο μεγαλύτερος γίνεται ο χρόνος ανόδου και καθόδου του σήματος στην θύρα.



Σχήμα 4.22 *Ακροδέκτες RB7:RB4 μονάδας εισόδου/ εξόδου (I/O) PORT B*

Β. 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 16F877 υποστηρίζει τρεις διαφορετικούς τύπους χρονιστών: τον Timer0, τον Timer1 και τον Timer2.

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.

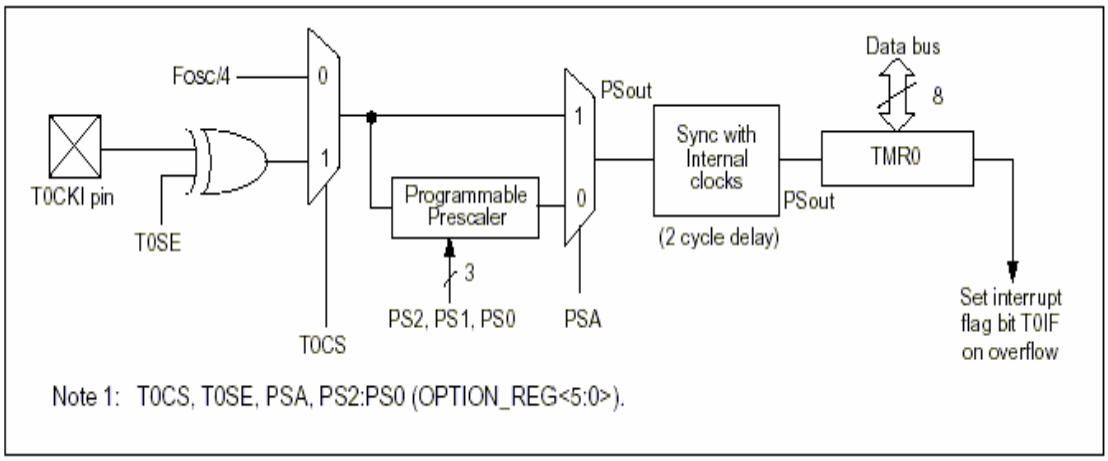
Timer 0

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:

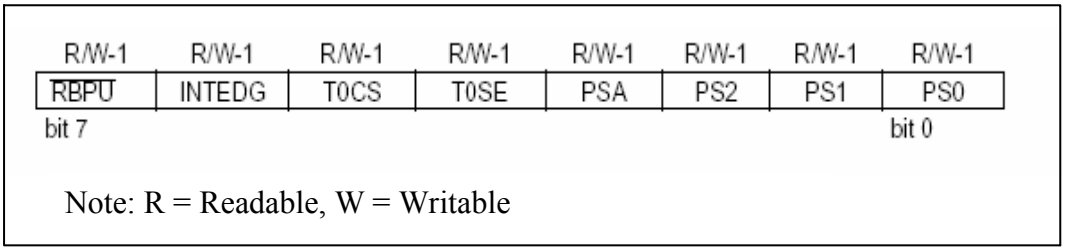
- Μετρητή των 8 bit
- Είναι αναγνώσιμος (readable) και εγγράψιμος (writable)
- Επιλογέα κλίμακας χρονισμού (prescaler) που μπορεί να προγραμματιστεί μέσω λογισμικού (software)
- Επιλογή χρήσης εσωτερικού ή εξωτερικού ρολογιού
- Πρόκληση διακοπής (Interrupt) κατά την υπερχείλιση του μετρητή από FFh σε 00h

Η βασική αρχιτεκτονική του timer0 φαίνεται στο σχήμα 4.23.



Σχήμα 4.23 *Αρχιτεκτονική (block diagram) του 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). Πιο συγκεκριμένα, τα bit του καταχωρητή OPTION φαίνονται στο παρακάτω σχήμα (Σχήμα 4.24).



Σχήμα 4.24 *Τα bit του καταχωρητή OPTION*

Όπως μπορεί κανείς να παρατηρήσει και από το σχήμα 4.23 ο χρονιστής timer0 μπορεί να λειτουργήσει είτε με το εσωτερικό ρολόι του PIC είτε με εξωτερικό ρολόι (external clock) μέσω του ακροδέκτη (pin) TOCK1 του PIC. Η επιλογή του ρολογιού λειτουργίας του χρονιστή γίνεται μέσω ενός πολυπλέκτη και μα βάση την τιμή του bit TOCS (OPTION<5>). 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 (OPTION<3>). 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 (PS2:PS0) καθορίζουν τον ρυθμό λειτουργίας του διαιρέτη μέτρησης (prescaler) με βάση τον παρακάτω πίνακα (Πίνακας 4.3). Όπως φαίνεται, ο ρυθμός (prescaler rate) διαφέρει ανάλογα με το αν ο prescaler χρησιμοποιείται από τον timer0 ή από τον Watchdog Timer.

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

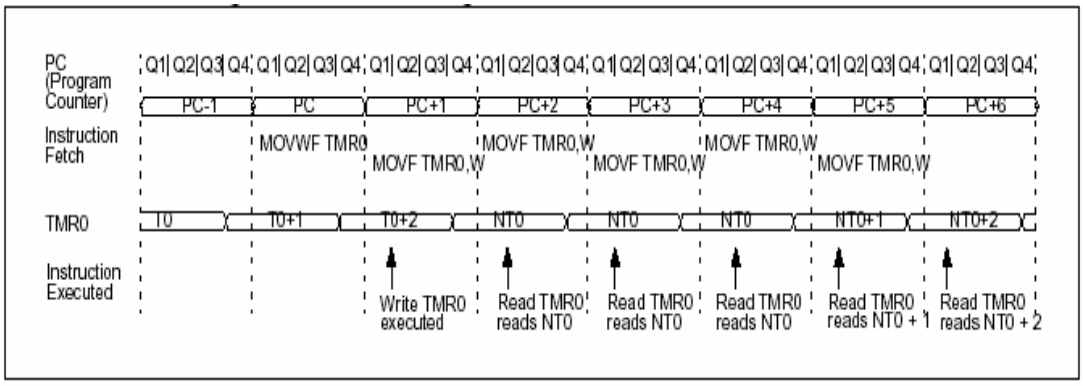
Πίνακας 4.3 *Ρυθμός λειτουργίας του διαιρέτη μέτρησης (prescaler) με βάση την τιμή των bit PS2:PS0 του OPTION register*

Διαιρέτης Μέτρησης (Prescaler)

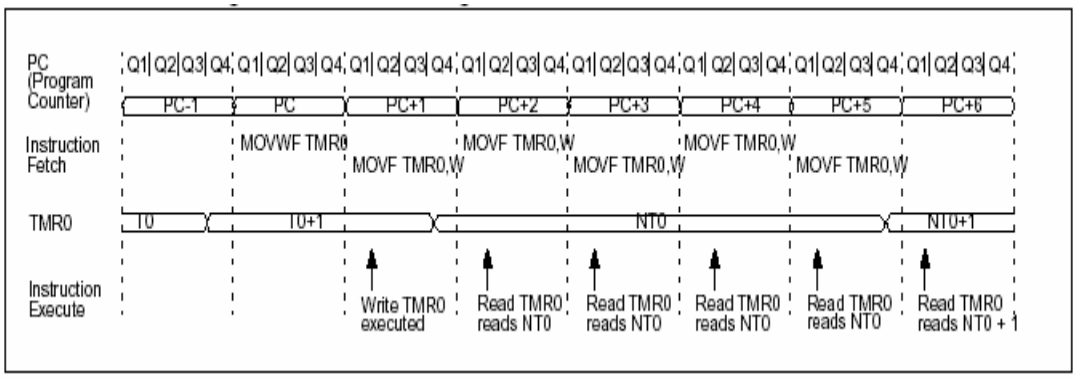
Ο διαιρέτης μέτρησης (prescaler) αποτελεί την μονάδα που παράγει το ρολόι λειτουργίας του μετρητή του χρονιστή timer0 (αν επιλεγεί). Ο prescaler δύναται να χρησιμοποιηθεί και από τον Watchdog Timer. Η επιλογή, όπως αναφέρθηκε και προηγουμένως, γίνεται μέσω του bit PSA (OPTION<3>). Όταν ο prescaler έχει ανατεθεί στον χρονιστή, μπορεί να παρέχει σε αυτόν διάφορους ρυθμούς μέτρησης όπως φαίνεται και στον πίνακα 4.3.

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, PS2:PS0, επιλέγεται και το bit εκείνο του μετρητή που θα αποτελέσει το ρολόι εξόδου του prescaler. Με βάση αυτό το ρολόι λειτουργεί μετέπειτα ο χρονιστής timer0 ή ο Watchdog Timer.

Στα παρακάτω σχήματα παρουσιάζονται δύο παραδείγματα από την λειτουργία του μετρητή. Στο σχήμα 4.25 δεν χρησιμοποιείται ο prescaler, ενώ στο σχήμα 4.26 γίνεται χρήση του prescaler με ρυθμό 1:2, που σημαίνει ότι το ρολόι που παράγεται από τον prescaler έχει την μισή περίοδο (διπλάσια συχνότητα) από το γενικό ρολόι του PIC.



Σχήμα 4.25 *Λειτουργία του timer0 χωρίς την χρήση του prescaler*

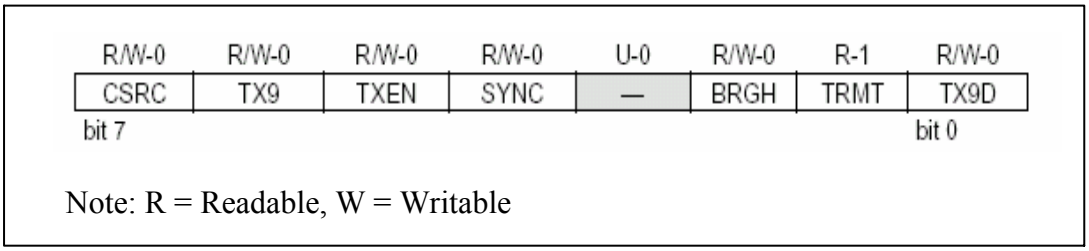


Σχήμα 4.26 *Λειτουργία του timer0 με την χρήση του prescaler*

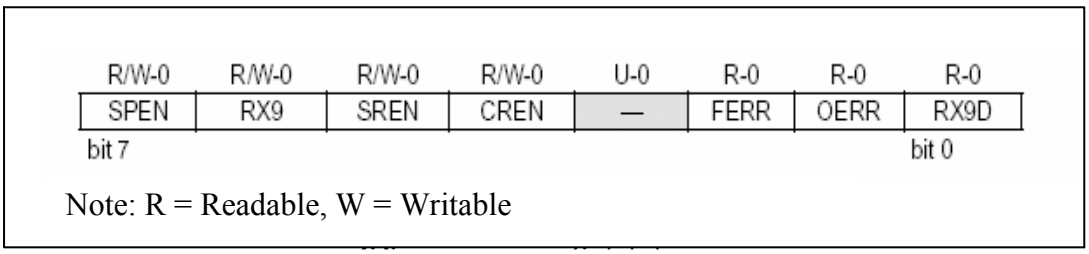
Γ. Πομπός / Δέκτης Ασύγχρονης / Σύγχρονης Σειριακής Επικοινωνίας (USART)

Ο πομπός / δέκτης Σύγχρονης / Ασύγχρονης σειριακής επικοινωνίας (**U**niversal **S**ychronous **A**synchronous **R**eceiver **T**ransmitter, 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 (98h) και του RCSTA (18h). Στο σχήμα 4.27 παρουσιάζεται ο καταχωρητής TXSTA, ενώ στο σχήμα 4.28 ο καταχωρητής RCSTA.



Σχήμα 4.27 *Ο καταχωρητής TXSTA*



Σχήμα 4.28 *Ο καταχωρητής RCSTA*

Η λειτουργία των bit των παραπάνω καταχωρητών εξηγείται λεπτομερειακά παρακάτω.

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 (99h) (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<2>) ελέγχει το ρυθμό μετάδοσης (baud rate) ενώ στην σύγχρονη κατάσταση, αγνοείται.

Οι τύποι που χρησιμοποιούνται για τον υπολογισμό του ρυθμού μετάδοσης της *USART*, για τις δύο καταστάσεις λειτουργίας και σε συνάρτηση με το εσωτερικό

ρολόι του PIC (Fosc) και το περιεχόμενο X του καταχωρητή SPBRG φαίνονται στο παρακάτω πίνακα (Πίνακας 4.4).

SYNC	BRGH = 0 (Low Speed)	BRGH = 1 (High Speed)
0	(Asynchronous) Baud Rate = Fosc/(64(X+1))	Baud Rate= Fosc/(16(X+1))
1	(Synchronous) Baud Rate = Fosc/(4(X+1))	NA

Πίνακας 4.4 *Συναρτήσεις υπολογισμού του ρυθμού μετάδοσης(Baud Rate) της USART*

Στη συνέχεια, αν είναι επιθυμητό, μπορεί να υπολογιστεί και το σφάλμα στον υπολογισμό του ρυθμού μετάδοσης, εφαρμόζοντας τους ίδιους τύπους αλλά γνωρίζοντας πλέον την τιμή του X και ζητώντας την νέα τιμή του baud rate.

Στο παρακάτω σχήμα (σχήμα 4.29) φαίνονται οι τιμές του ρυθμού μετάδοσης της *USART* για διάφορες τυχαίες συχνότητες λειτουργίας του PIC και οι αντίστοιχες τιμές του καταχωρητή SPBRG. Πρόκειται για ασύγχρονη λειτουργία της *USART* και με επιλεγμένο 1 το bit BRGH (TXSTA<2>) (High Speed). Με παρόμοιο τρόπο προκύπτουν και οι αντίστοιχες τιμές του ρυθμού μετάδοσης για άλλες συχνότητες λειτουργίας του PIC και για την σύγχρονη κατάσταση.

BAUD RATE (Kbps)	Fosc = 20 MHz		SPBRG value (decimal)	16 MHz		SPBRG value (decimal)	10 MHz		SPBRG value (decimal)	7.15909 MHz		SPBRG value (decimal)
	KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR	
0.3	NA	-	-	NA	-	-	NA	-	-	NA	-	-
1.2	1.221	+1.73	255	1.202	+0.16	207	1.202	+0.16	129	1.203	+0.23	92
2.4	2.404	+0.16	129	2.404	+0.16	103	2.404	+0.16	64	2.380	-0.83	46
9.6	9.469	-1.36	32	9.615	+0.16	25	9.766	+1.73	15	9.322	-2.90	11
19.2	19.53	+1.73	15	19.23	+0.16	12	19.53	+1.73	7	18.64	-2.90	5
76.8	78.13	+1.73	3	83.33	+8.51	2	78.13	+1.73	1	NA	-	-
96	104.2	+8.51	2	NA	-	-	NA	-	-	NA	-	-
300	312.5	+4.17	0	NA	-	-	NA	-	-	NA	-	-
500	NA	-	-	NA	-	-	NA	-	-	NA	-	-
HIGH	312.5	-	0	250	-	0	156.3	-	0	111.9	-	0
LOW	1.221	-	255	0.977	-	255	0.6104	-	255	0.437	-	255

Σχήμα 4.29 *Ρυθμοί μετάδοσης για ασύγχρονη κατάσταση λειτουργίας*

Ασύγχρονη Κατάσταση Λειτουργίας της USART (USART Asynchronous Mode)

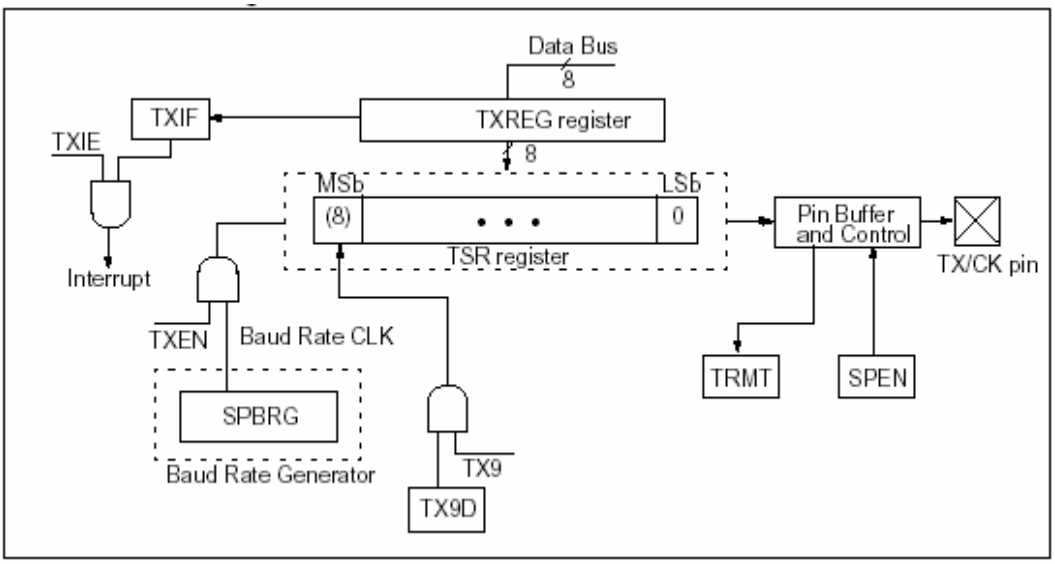
Στην κατάσταση αυτή, η *USART* χρησιμοποιεί τη καθορισμένη διάταξη μη επιστροφής στο μηδέν (**N**on**R**eturn-to-**Z**ero, NRZ), δηλαδή 1 bit έναρξης (start bit), 8 ή 9 bit για το δεδομένο (data bit) και 1 bit τέλους (stop bit). Καθένα από αυτά τα bit μπορεί να στέλνεται με βάση ένα από τα καθορισμένα 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)

Η αρχιτεκτονική του πομπού της *USART* φαίνεται στο σχήμα 4.30. Όπως παρατηρούμε, ο πυρήνας του πομπού είναι ένας σειριακός καταχωρητής ολίσθησης (**T**ransmit **S**hift **R**egister, TSR). Ο καταχωρητής αυτός παίρνει τιμές από τον καταχωρητή TXREG (19h) (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 (PIR1<4>) 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<1>) 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<5>). 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.



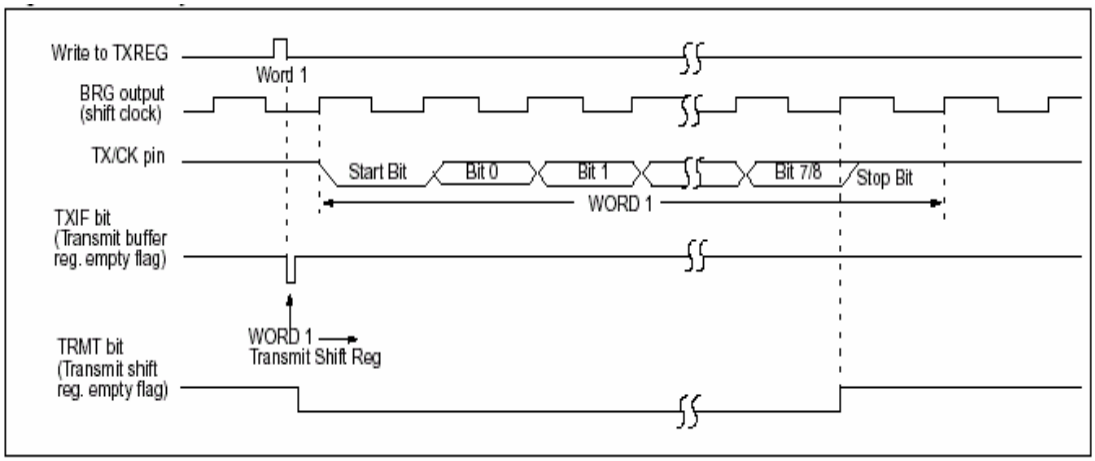
Σχήμα 4.30 *Αρχιτεκτονική του πομπού της USART*

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

- ⇨ 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 (TXSTA<4>).
- ⇨ Αν είναι επιθυμητές οι διακοπές, τότε πρέπει να τεθούν τα bit TXIE (PIE1<4>), GIE (INTCON<7>) και PEIE (INTCON<6>).
- ⇨ Αν είναι επιθυμητή η αποστολή 9 bit δεδομένου και όχι 8, τότε πρέπει να τεθεί 1 το bit TX9 (TXSTA<6>).

- ⇨ Ενεργοποίηση της αποστολής θέτοντας το bit TXEN (TXSTA<5>), το οποίο θέτει και το bit TXIF (PIR1<4>)
- ⇨ Αν είναι επιθυμητή η αποστολή 9 bit, τότε το 9^{o} bit πρέπει να φορτωθεί στο bit TX9D (TXSTA<0>).
- ⇨ Φόρτωση δεδομένου στον καταχωρητή TXREG και η αποστολή ξεκινάει.

Στο σχήμα 4.31 δίνεται ένα παράδειγμα την ασύγχρονης κατάστασης λειτουργίας του πομπού της USART

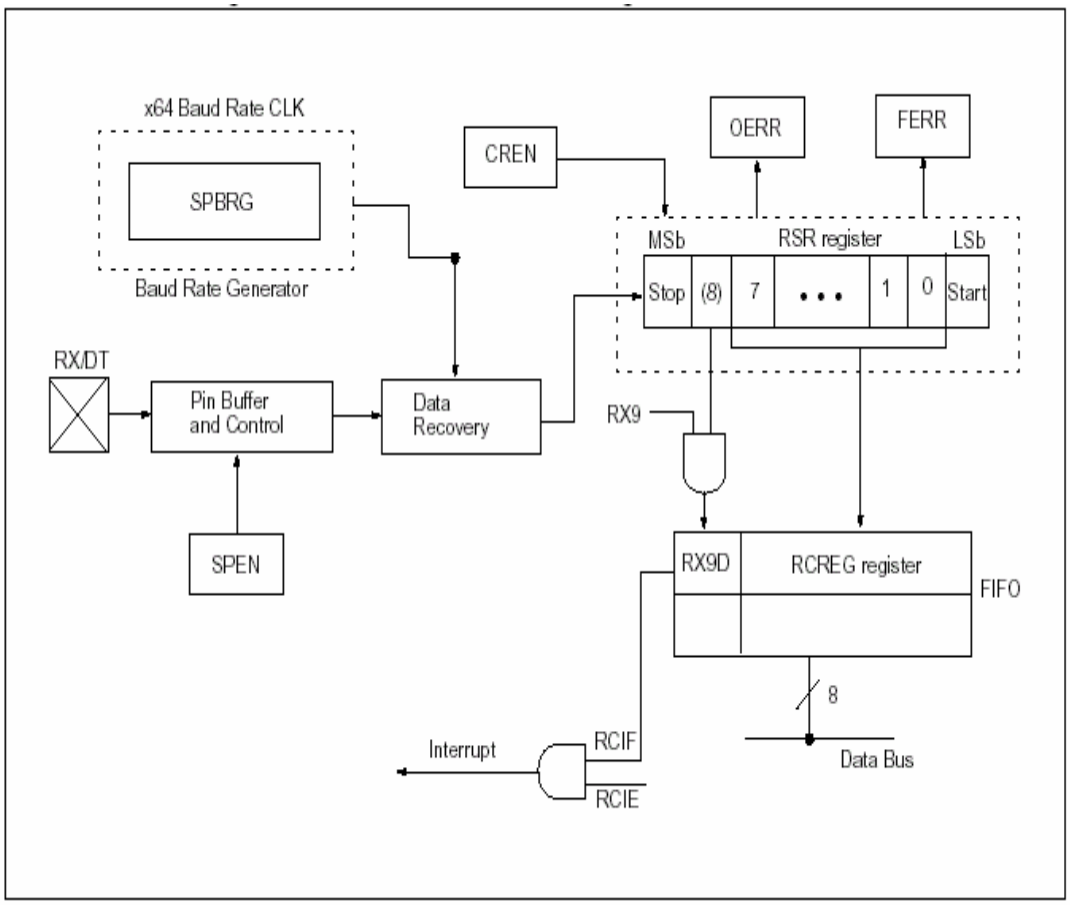


Σχήμα 4.31 *Ασύγχρονη αποστολή μέσω USART*

Δέκτης ασύγχρονης κατάστασης λειτουργίας της USART (USART Asynchronous Receiver)

Η αρχιτεκτονική του πομπού της USART φαίνεται στο σχήμα 4.32. 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 (RCSTA<4>).Ο πυρήνας της δέκτη είναι ο σειριακός καταχωρητής ολίσθησης λήψης (**R**eceive **S**hift **R**egister, RSR). Μόλις αναγνωριστεί και το stop bit στον ακροδέκτη RX/TX του PIC, το δεδομένο μεταφέρεται από τον RSR στον RCREG (1Ah) (USART Receive Data Register), αν αυτός είναι άδειος.



Σχήμα 4.32 *Αρχιτεκτονική του δέκτη της USART*

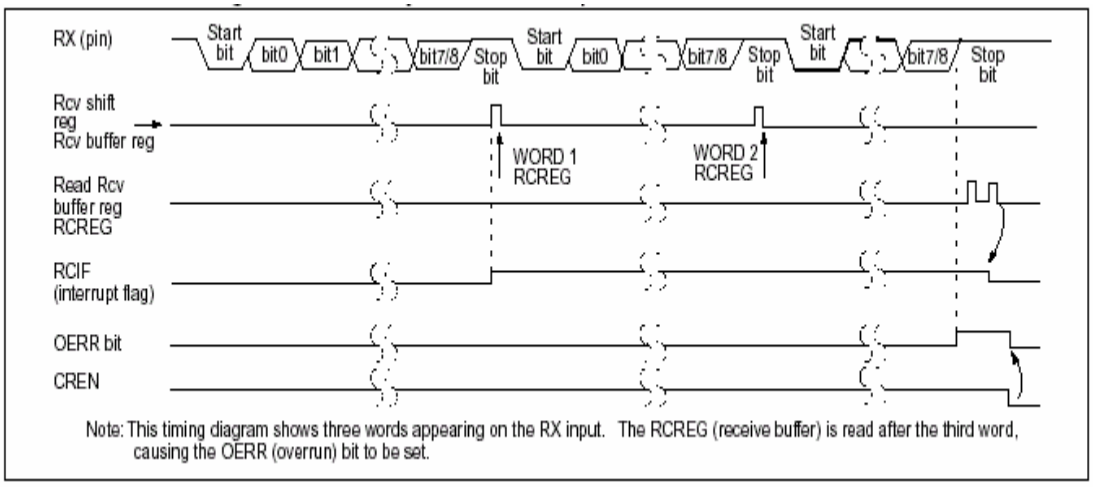
Όταν ολοκληρωθεί η μεταφορά τίθεται το bit RCIF (PIR1<5>). Το bit αυτό είναι μόνο αναγνώσιμο (readable) και μηδενίζεται μόνο μέσω υλικού (hardware) όταν ο καταχωρητής RCREG διαβαστεί και αδειάσει. Ο καταχωρητής αυτός μπορεί να είναι μια στοίβα 2 επιπέδων (two deep FIFO). Είναι δυνατόν 2 bytes να έχουνε 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 (RCSTA<1>) (**O**verrun **Err**or 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 (RCSTA<2>) (**F**raming **Err**or 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 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 (TXSTA<4>).
- ⇨ Αν είναι επιθυμητές οι διακοπές, τότε πρέπει να τεθούν τα bit RCIE (PIE1<5>), GIE (INTCON<7>) και PEIE (INTCON<6>).
- ⇨ Αν είναι επιθυμητή η λήψη 9 bit δεδομένου και όχι 8, τότε πρέπει να τεθεί 1 το bit RX9 (RCSTA<6>).
- ⇨ Ενεργοποίηση της αποστολής θέτοντας το bit CREN (RCSTA<4>)
- ⇨ 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.

Στο παρακάτω σχήμα φαίνεται ένα παράδειγμα της ασύγχρονης λειτουργίας του δέκτη της USART.



Σχήμα 4.33 *Ασύγχρονη λήψη μέσω USART*

ΚΕΦΑΛΑΙΟ 5

ΛΟΓΙΣΜΙΚΟ

5.1 Περιγραφή Λογισμικού

Κατά την διάρκεια ανάπτυξης της εφαρμογής το τμήμα που αποτέλεσε τη μεγαλύτερη δυσκολία αλλά ταυτόχρονα και την μεγαλύτερη πρόκληση ήταν η διαδικασία ανάπτυξης του λογισμικού (software). Το λογισμικό της εφαρμογής αφορά αποκλειστικά τον μικροελεγκτή και είναι το πρόγραμμα που θα εκτελείται κάθε φορά που αυτός τίθεται σε λειτουργία. Κατά την συγγραφή του προγράμματος δόθηκε ιδιαίτερη έμφαση στην απλούστευση και ελαχιστοποίηση του κώδικα, στην εξοικονόμηση πόρων του μικροελεγκτή (θέσεις μνήμης), καθώς και στην βελτιστοποίηση όλων εκείνων των λειτουργιών και διαδικασιών που πρέπει να εκτελεστούν προκειμένου να επιτευχθεί σωστή και αξιόπιστη λειτουργία του συστήματος. Επίσης έγινε προσπάθεια ο κώδικας του προγράμματος να είναι ευανάγνωστος και εύκολα τροποποιήσιμος στην περίπτωση που χρειαστεί να αλλάξουν δεδομένα ή ονόματα μεταβλητών σύμφωνα με τις επιθυμίες και τις απαιτήσεις του χρήστη.

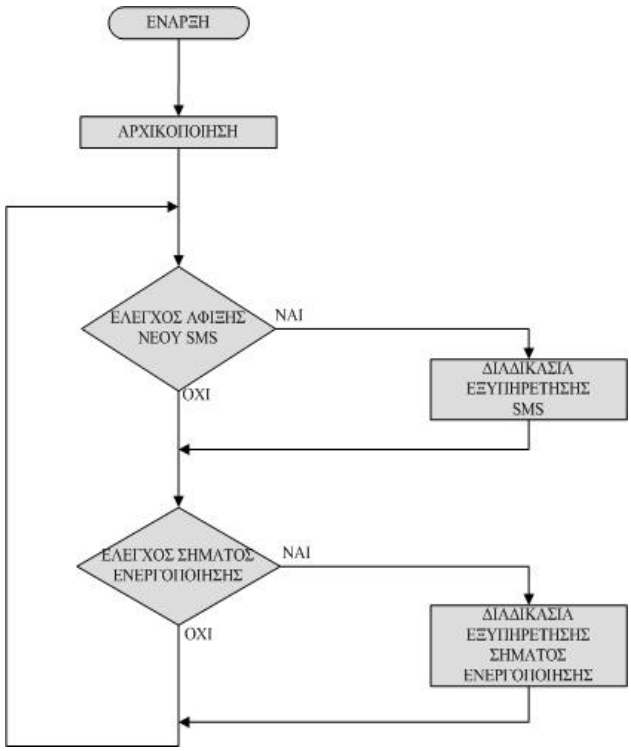
Για την συγγραφή του κώδικα του προγράμματος επιλέχτηκε η χρήση της γλώσσας *C* ως γλώσσα προγραμματισμού υψηλού επιπέδου αντί του προγραμματισμού σε γλώσσα *Assembly* κάνοντας άμεσα χρήση του ρεπερτορίου εντολών του μικροελεγκτή. Ο λόγος για την επιλογή αυτή προήρθε από τις δυνατότητες που μας δίνει η χρήση μια τέτοιας γλώσσας όσο αναφορά τον έλεγχο ροής του προγράμματος, την υποστήριξη σύνθετων δομών όπως πίνακες, δείκτες και συμβολοσειρές καθώς και την χρησιμοποίηση ρουτινών ή συναρτήσεων που υπάρχουν στις βιβλιοθήκες της γλώσσας και οι οποίες χρησιμοποιούνται άμεσα κατά την εφαρμογή.

Η μετάφραση του προγράμματος της εφαρμογής έγινε με τον *compiler PICC (έκδοση 7.87PL2)* που αναπτύσσεται από την εταιρία λογισμικού για μικροελεγκτές

HI-TECH Software [6]. Ο μεταφραστής αυτός αποτελεί μια υλοποίηση της γλώσσας προγραμματισμού *C* ειδικά για τους μικροελεγκτές *PIC* της εταιρίας *Microchip Technology Inc*. Η ανάπτυξη του λογισμικού έγινε στο περιβάλλον *εργασίας MPLAB IDE (έκδοση 6.40.00)* το οποίο αποτελεί ένα πλήρες αναπτυξιακό περιβάλλον για τους ελεγκτές *PIC* και δίνεται ελεύθερα από την εταιρία *Microchip Technology Inc*. Στο αναπτυξιακό αυτό πακέτο περιλαμβάνονται ένας επεξεργαστής κειμένου *(Text Editor)*, ένας εξομοιωτής *(MPLAB SIM)*, ένα εργαλείο υποστήριξης ελέγχου έκδοσης εφαρμογών καθώς και το πρόγραμμα επικοινωνίας του συστήματος με το προγραμματιστή του μικροελεγκτή *(PIC STAR PLUS)*.

5.2 Λειτουργία του Προγράμματος

Η κύρια λειτουργία του προγράμματος της εφαρμογής παρουσιάζεται στο block διάγραμμα του σχήματος 5.1.



Σχήμα 5.1 *Ατέρμων βρόγχος ελέγχου*

Όπως παρατηρούμε στο διάγραμμα μετά την έναρξη εκτέλεσης του προγράμματος ακολουθεί στο στάδιο αρχικοποίησης και καθορισμού *(configuration)* των περιφερειακών του μικροελεγκτή και του τηλεφώνου. Στην συνέχεια ο έλεγχος περνά μέσα σε ένα ατέρμονα βρόγχο στον οποίο εκτελούνται διαδοχικά δύο έλεγχοι. Ο πρώτος έλεγχος αφορά την άφιξη ενός νέου σύντομου γραπτού μηνύματος. Στην περίπτωση που ένα νέο μήνυμα έχει αφιχθεί στο κινητό τηλέφωνο βάσης τότε ο έλεγχος περνά στην διαδικασία αποκωδικοποίησης και εκτέλεσης του. Μετά την ολοκλήρωση της διαδικασίας ή στην περίπτωση που δεν υπάρχει ένα νέο μήνυμα η εκτέλεση συνεχίζεται με τον δεύτερο έλεγχο στον οποίο εξετάζεται αν υπάρχει σήμα ενεργοποίησης. Στην περίπτωση που υπάρχει ένα τέτοιο σήμα ο έλεγχος περνά στην διαδικασία εξυπηρέτησης σήματος ενεργοποίησης η οποία αφού ολοκληρωθεί στέλνει τον έλεγχο στην αρχή του βρόγχου. Η βρόγχος αυτός επαναλαμβάνεται συνέχεια έκτος αν το σύστημα έλεγχου (ο μικροελεγκτής) επανατοποθετηθεί με αποτέλεσμα να χαθεί η τρέχουσα κατάσταση λειτουργίας και να ξεκινήσει η εκτέλεση από την αρχή.

Επειδή η εφαρμογή διαχειρίζεται τυχαία γεγονότα (όπως τη λήψη σύντομων μηνυμάτων SMS) το πρόγραμμα σχεδιάστηκε έτσι ώστε να είναι «οδηγούμενο» από διακοπές. Αυτό σημαίνει ότι σημαντικές λειτουργίες εκτελούνται την στιγμή εμφάνισης κάποιας διακοπής. Οι διακοπές που λαμβάνουν χώρα κατά την εκτέλεση του προγράμματος είναι δύο:

1. Η διακοπή σειριακής θύρας *(USART Interrupts)* την στιγμή της λήψη ενός *byte.*
2. Η διακοπή κατά την αλλαγή επιπέδου στάθμης *(rising)* στον ακροδέκτη *RB0/INT* της θύρας *PORTB*.

Όποτε συμβεί κάποια από τις παραπάνω διακοπές ενημερώνονται από την ρουτίνα εξυπηρέτησης διακοπής αντίστοιχες μεταβλητές δείχνοντας με αυτό τον τρόπο την ύπαρξη του ανάλογου γεγονότος. Στην συνέχεια ο έλεγχος του προγράμματος περνά στις διαδικασίες εξυπηρέτησης των γεγονότων αυτών μέχρι την στιγμή που αυτές ολοκληρωθούν για επιστρέψει και πάλι στον κεντρικό βρόγχο ελέγχου.

5.3 Δομή Προγράμματος

Με σκοπώ την καλύτερη παράθεση του προγράμματος ο κώδικας έχει χωριστεί σε αρχεία ανάλογα με την λειτουργία που επιτελεί το κάθε ένα. Όλο το πρόγραμμα αποτελείται από έξι αρχεία με τα εξής ονόματα:

- main.c / main.h
- sci.c / sci.h
- delay.c / delay.h

Για το διαχωρισμό του τμήματος δηλώσεων από το τμήμα εντολών κάθε αρχείο χωρίζεται σε δύο τμήματα. Τα αρχεία με την προέκταση .c, "*filename.c"* αποτελούν τα αρχεία με τον κώδικα και τις εντολές του προγράμματος στην γλώσσα *C*. Τα αρχεία με την προέκταση .h, "*filename.h" (header files)* είναι τα αρχεία στα οποία γίνονται οι ορισμοί και οι δηλώσεις μεταβλητών ή συναρτήσεων.

5.3.1 Αρχείο main.c

Το αρχείο *main.c* αποτελεί το κεντρικό αρχείο του προγράμματος στο οποίο περιλαμβάνονται όλα τα υπόλοιπα αρχεία κώδικα μέσο της εντολής *#include* που αποτελεί οδηγία του προ-επεξεργαστή της *C*. Το αρχείο αυτό περιλαμβάνει τα δομικά μέρη του προγράμματος όπως την ρουτίνα εξυπηρέτησης διακοπής *(isr, interrupt service routine)*, την αρχικοποίηση του συστήματος, τον κύριο βρόχο ελέγχου καθώς και τις διαδικασίες εξυπηρέτησης των γεγονότων.

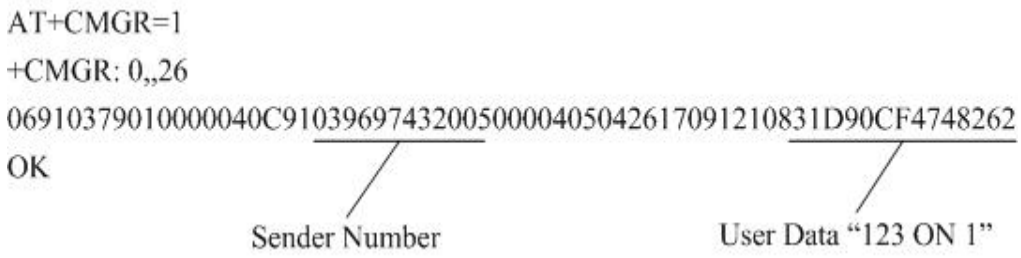
Το κάθε τμήμα αποτελεί ένα ξεχωριστό κομμάτι του οποίου η χρησιμότητα και η λειτουργία παρουσιάζονται στην συνέχεια της αναφοράς.

Α. Ρουτίνα Εξυπηρέτησης Διακοπής (Interrupt Service Routine)

Η ρουτίνα εξυπηρέτησης διακοπής περιλαμβάνει το σύνολο των εντολών που θα εκτελούνται όποτε συμβεί κάποια από τις διακοπές του συστήματος. Οι διακοπές που μπορούν να συμβούν κατά την διάρκεια εκτέλεσης του προγράμματος προέρχονται από την συριακή θύρα, κατά την λήψη ενός δεδομένου και από τον ακροδέκτη

RB0/INT της θύρας *B (PORT B)*. Διακοπή λόγο λήψης δεδομένων έχουμε όταν το κινητό τηλέφωνο βάσης επιχειρήσει να αποστείλει μια ακολουθία χαρακτήρων προς τον μικροελεγκτή. Το γεγονός αυτό μπορεί να συμβεί σε δύο περιπτώσεις. Πρώτον όταν ο μικροελεγκτής στείλει μια εντολή προς το κινητό και στην συνέχεια αυτό απαντήσει και δεύτερον όταν το κινητό τηλέφωνο στείλει μια ειδική ακολουθία χαρακτήρων προς τον μικροελεγκτή για να τον ενημερώσει για την λήψη ενός νέου σύντομου μηνύματος. Σε κάθε περίπτωση όποτε το κινητό τηλέφωνο αποστείλει δεδομένα τότε προκαλούνται απανωτές διακοπές από τον δέκτη της σειριακής θύρας (μία για κάθε χαρακτήρα αποστολής) με τους χαρακτήρες να αποθηκεύονται σε κατάλληλους πίνακες *(buf[], TEL[])* ώστε να χρησιμοποιηθούν στην συνέχεια για την περαιτέρω επεξεργασία τους. Ο λόγος για τον οποίο κρίθηκε αναγκαία η ύπαρξη αυτού του είδους της διακοπής προέρχεται από την δυσκολία συγχρονισμού του κινητού τηλεφώνου βάσης και του μικροελεγκτή κατά την λήψη δεδομένων. Για να διαβάσουμε δεδομένα από το δέκτη της *USART* θα πρέπει να αναθέσουμε το περιεχόμενο του καταχωρητή *RCREG* σε κάποια μεταβλητή ώστε αυτός να ελευθερωθεί και να είναι έτοιμος για την λήψη του επόμενου χαρακτήρα. Αν και ο καταχωρητής αυτός αποτελεί μια στοίβα 2 επιπέδων *(2 bit FIFO)* και είναι σε θέση να αποθηκεύσει 2 χαρακτήρες *(2 byte)* στην περίπτωση που ένας νέος χαρακτήρας προσέρθει ενώ οι προηγούμενοι δε έχουν διαβαστεί τότε ο χαρακτήρας αυτός χάνεται και προκαλείται σφάλμα κατά την μετάδοση των δεδομένων. Επιπλέον ένα ακόμα πρόβλημα που θα προέκυπτε αν δεν γινόταν χρήση της συγκεκριμένης διακοπής είναι ότι θα έπρεπε να ήμασταν αναγκασμένοι να ελέγχουμε πότε θα ολοκληρωθεί η διαδικασία της μετάδοσης των δεδομένων ώστε να προχωρήσουμε στη συνέχεια στην εκτέλεση του προγράμματος.

Κατά την διάρκεια εκτέλεσης της ρουτίνας εξυπηρέτησης διακοπής που προέρχεται από τον δέκτη της συριακής θύρας τα δεδομένα που αποθηκεύονται είναι το τμήμα δεδομένων *(User Data)* και το τμήμα του αριθμού τηλεφώνου *(Sender Number)* μιας PDU ακολουθίας ενός μηνύματος όπως φαίνεται στο σχήμα 5.2. Το μήνυμα αυτό αποστέλλεται από το κινητό τηλέφωνο στον μικροελεγκτή ύστερα από την αποστολή της AT εντολής *AT+CMGR=1* (είναι το μήνυμα που υπάρχει στην πρώτη θέση μνήμης του κινητού τηλεφώνου) .



Σχήμα 5.2 *Αποθήκευση δεδομένων PDU ακολουθίας*

Επιπλέον εκτός αυτών που αναφέρθηκαν αποθηκεύεται και η ειδοποίηση του κινητού τηλεφώνου (μεταβλητή *taf*) για την ύπαρξη ενός νέου μηνύματος γεγονός που θα χρησιμοποιηθεί αργότερα κατά την διαδικασία ελέγχου. Η ειδοποίηση αυτή είναι αποτέλεσμα της αρχικοποίησης του κινητού τηλεφώνου έτσι όπως έγινε μέσο της εντολής *AT+CMNI* και έχει την μορφή του σχήματος 5.3. Για να επιβεβαιώσουμε πως πρόκειται για ειδοποίηση νέου μηνύματος αρκεί να αποθηκεύσουμε τον έβδομο χαρακτήρα της ακολουθίας και να ελέγξουμε στην συνέχεια (πρώτο στάδιο ελέγχου) αν πρόκειται για τον χαρακτήρα «Τ».

Σχήμα 5.3 *Αποθήκευση ειδοποίησης νέου SMS*

Η δεύτερη διακοπή που μπορεί να συμβεί κατά την διάρκεια εκτέλεσης προέρχεται από την πόρτα εισόδου *B (PORT B)* και αφορά την αλλαγή στάθμης στο ακροδέκτη *RB7*. Κάθε φορά που ο ακροδέκτης αυτός αλλάζει επίπεδο τάσης προκαλείται διακοπή και ενημερώνονται σχετικά οι κατάλληλες μεταβλητές οι οποίες θα χρησιμοποιηθούν στην συνέχεια κατά την διαδικασία ελέγχου.

Ο διαχωρισμός και η επιλογή της συνάρτησης που θα εκτελεστεί την στιγμή που ο έλεγχος πηγαίνει στην ρουτίνα εξυπηρέτησης διακοπής γίνεται μέσο των *bit* ενεργοποίησης *(intnameIE, Interrupt Enable bit)* και εξυπηρέτησης *(intnameIF,*

Interrupt Flag Bit) διακοπής. Έτσι ανάλογα με το ποίες από τι σημαίες αυτές είναι ενεργοποιημένες εκτελούνται και οι αντίστοιχες εντολές της διακοπής.

Β. Διαδικασία Αρχικοποίησης

Η εκτέλεση του προγράμματος ξεκινά με την διαδικασία αρχικοποίησης η οποία περιλαμβάνει τόσο τον μικροελεγκτή όσο και το κινητό τηλέφωνο βάσης. Η αρχικοποίηση του μικροελεγκτή γίνεται μέσο της συνάρτηση *device_config(void)* και αφορά τα περιφερειακά και τις διακοπές που θα χρησιμοποιηθούν κατά τη εφαρμογή. Ποίο συγκεκριμένα καθορίζονται :

- Η πόρτα *D (PORTD)* ως έξοδος.

  Θα χρησιμοποιηθεί για την παραγωγή των σημάτων ελέγχου.
- Ο ακροδέκτης *RB7* της πόρτας *B (PORTB)* ως είσοδος.

  Θα χρησιμοποιηθεί για την εξωτερική διακοπή του σήματος ενεργοποίησης.
- Ο Πομπός / Δέκτης Ασύγχρονης / Σύγχρονης Σειριακής Επικοινωνίας *(USART)* για ασύγχρονη λειτουργία με ρυθμό μετάδοσης 9600Bd και μεταφορά οκτώ bit δεδομένων.
- Οι κατάλληλοι καταχωρητές ώστε να ενεργοποιηθούν οι διακοπές που αφορούν την λήψη δεδομένων καθώς και την αλλαγή επιπέδου τάσης στον ακροδέκτη *RB7*.

Η αρχικοποίηση του κινητού τηλεφώνου βάσης περιλαμβάνει :

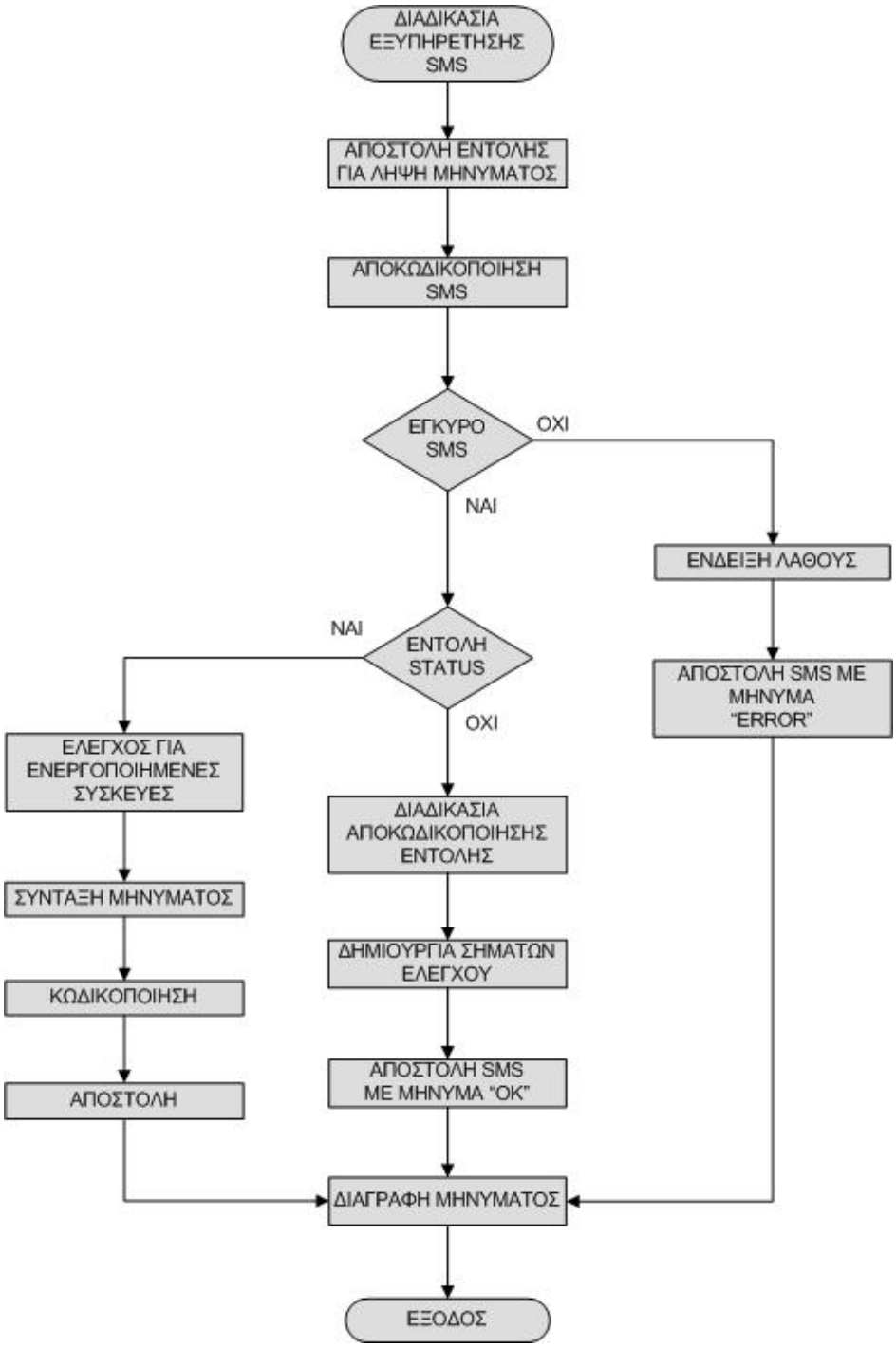
- Το καθορισμό του τρόπου ειδοποίησης του τερματικού σύνδεσης (TE) κατά την άφιξη ενός νέου συντόμου γραπτού μηνύματος (SMS) μέσο της εντολής *AT+CNMI* .
- Τον καθορισμό της μνήμης για την αποθήκευση ενός νεοεισερχόμενου μηνύματος μέσο της εντολής *AT+CPMS.*

Μετά την διαδικασία αρχικοποίησης ο έλεγχος του προγράμματος περνά σε έναν ατέρμονα βρόχο ο οποίος αποτελεί και το κύριο μέρος του προγράμματος της εφαρμογής. Ο βρόγχος αυτός περιλαμβάνει την διεξαγωγή δύο ελέγχων που αφορούν την λήψη ενός νέου μηνύματος και την ύπαρξη σήματος ενεργοποίησης. Όσο δεν συμβαίνει κανένα από αυτά τα δύο γεγονότα και η ρουτίνα εξυπηρέτησης διακοπής δεν ενημερώνει τις αντίστοιχες μεταβλητές η διαδικασία ελέγχου εξακολουθεί να

εκτελείται μέχρι την στιγμή που κάποιο από τα δύο ενδεχόμενα συμβεί. Στην περίπτωση αυτή ο έλεγχος του προγράμματος περνά στην διαδικασία εξυπηρέτησης του κατάλληλου γεγονότος και αφού ολοκληρωθεί επιστρέφει και πάλι στην διαδικασία ελέγχου.

Γ. Διαδικασία Εξυπηρέτησης Σύντομου Μηνύματος (SMS)

Η διαδικασία εξυπηρέτησης σύντομου μηνύματος παρουσιάζεται στο block διάγραμμα του σχήματος 5.4. Όπως παρατηρούμε η διαδικασία ξεκινά με τη λήψη του μηνύματος από τον μικροελεγκτή με την αποστολή της εντολής *AT+CMGR*. Μετά την λήψη ακολουθεί η αποκωδικοποίηση του τμήματος δεδομένων της PDU ακολουθίας του μηνύματος η οποία χρησιμοποιεί τον αλγόριθμο αποκωδικοποίησης που περιγράφτηκε στο κεφάλαιο 3. Αφού ολοκληρωθεί η αποκωδικοποίηση και αποκατασταθεί η αρχική μορφή του μηνύματος που έστειλε ο χρήστης ο έλεγχος περνά στο στάδιο ελέγχου και εγκυρότητας του μηνύματος ώστε να διασφαλιστεί ότι γράφτηκε σύμφωνα με τους κανόνες σύνταξης. Το στάδιο αυτό περιλαμβάνει πέντε επιμέρους ελέγχους και στην περίπτωση που τουλάχιστον ένας δεν ικανοποιείται τίθεται η σημαία σφάλματος *(ERROR_FLAG)* ώστε να εκτελεστεί στην συνέχεια η διαδικασία αποστολής μηνύματος με την ένδειξη λάθους. Αν το μήνυμα περάσει όλα τα στάδια ελέγχου η εκτέλεση του προγράμματος συνεχίζεται με την αναγνώριση της εντολής και των ορισμάτων που πιθανόν αυτή έχει για την δημιουργία στην συνέχεια των κατάλληλων σημάτων ελέγχου. Η διαδικασία ολοκληρώνεται με την αποστολή της λέξης *"OK"* για την ενημέρωση του χρήστη ότι ο έλεγχος του συστήματος έλιξε με επιτυχία. Στην περίπτωση που η εντολή του χρήστη είναι η εντολή *STATUS* τότε γίνεται έλεγχος για της ενεργοποιημένες συσκευές, συντάσσεται το κατάλληλο μήνυμα και αφού κωδικοποιηθεί κατάλληλα τότε αποστέλλεται στον χρήστη. Στο τέλος της διαδικασίας διαγράφεται το μήνυμα που λάβαμε από την πρώτη θέση μνήμης του κινητού τηλεφώνου βάσης ώστε να ελευθερωθεί η θέση για το επόμενο μήνυμα.



Σχήμα 5.4 *Block διάγραμμα της διαδικασίας εξυπηρέτησης SMS*

Δ. Διαδικασία Εξυπηρέτησης Σήματος Ενεργοποίησης

Η διαδικασία εξυπηρέτησης του σήματος ενεργοποίησης αποτελεί το σύνολο εντολών που εκτελούνται όταν υπάρξει μια εξωτερική διακοπή από τον ακροδέκτη *PB0/INT*. Ό έλεγχος του προγράμματος περνά στο σημείο αυτό αφού πρώτα η ρουτίνα εξυπηρέτησης διακοπής θέσει κατάλληλα την σημαία *ALARM* ώστε να επιτραπεί η είσοδος στην διαδικασία εξυπηρέτησης σήματος μέσα στο βρόχο ελέγχου. Όταν συμβεί αυτό το σύστημα θα αποστείλει κατάλληλο μήνυμα σε προεπιλεγμένο αριθμό κινητού τηλεφώνου ώστε να ενημερώσει τον χρήστη για την ύπαρξη του σήματος και κατεπέκτασιν του γεγονός που έχουμε ορίσει να αντιπροσωπεύει αυτό. Η όλη διαδικασία παρουσιάζεται στο block διάγραμμα του σχήματος 5.5 .

Σχήμα 5.5 *Block διάγραμμα της διαδικασίας εξυπηρέτησης σήματος ενεργοποίησης*

5.3.2 Διασύνδεσης Σειριακής Επικοινωνίας

Στα αρχεία *sci.h / sci.c* γίνονται οι δηλώσεις και ο ορισμός συναρτήσεων για το τμήμα Διασύνδεσης Σειριακής Επικοινωνίας *(SCI, Synchronous Communications*

Interface) καθώς επίσης και ότι έχει σχέση για την μετάδοση μεταξύ του μικροελεγκτή και του κινητού τηλεφώνου βάσης.

Οι συναρτήσεις που ορίζονται είναι:

1. Η συνάρτηση *sci_Init(unsigned long int, unsigned char)* μέσο της οποίας αρχικοποιείται ο Πομπός / Δέκτης Ασύγχρονης / Σύγχρονης Σειριακής Επικοινωνίας *(USART)* για ασύγχρονη λειτουργία με ρυθμό μετάδοσης 9600Bd και μεταφορά οκτώ *bit* δεδομένων. Επιπλέον καθορίζεται και η διακοπή κατά την λήψη δεδομένων στον δέκτη της σειριακής μονάδας.
2. Η συνάρτηση *sci_PutByte(unsigned char)*

   Συνάρτηση για την αποστολή ενός *byte* από τον πομπό της σειριακής μονάδας του μικροελεγκτή.
3. Η συνάρτηση *sci_GetByte(void)*

   Συνάρτηση για την λήψη δεδομένων από δέκτη της σειριακής μονάδα. Καλείται στην ρουτίνα εξυπηρέτησης διακοπής μετά την λήψη ενός δεδομένου από τον μικροελεγκτή.
4. Η συνάρτηση *sci_CheckOERR(void)*

   Συνάρτηση η οποία μηδενίζει και θέτει ξανά το *bit CREN* στην περίπτωση που προκληθεί λάθος και έχει ενεργοποιηθεί η σημαία *OERR (Overrun Error Bit)* κατά την διάρκεια λήψης των δεδομένων. Με την επανατοποθέτηση του bit *CREN* άρετε η απαγόρευση της λήψης δεδομένων και αποκαθίσταται η επικοινωνία του μικροελεγκτή με το κινητό τηλέφωνο.
5. Η συνάρτηση *sendAT(const unsigned char *ch)*

   Η συνάρτηση αυτή χρησιμοποιείται για την αποστολή μια αλληλουχίας χαρακτήρων από τον μικροελεγκτή προς το κινητό τηλέφωνο η οποία αντιπροσωπεύει μια AT εντολή. Το τέλος κάθε ακολουθίας σημειώνεται με την αποστολή του ASCII χαρακτήρα 0x0D που αποτελεί τον χαρακτήρα έναρξης εκτέλεσης της εντολής ([CR],Carriage Return) από το κινητό τηλέφωνο βάσης.
6. Η συνάρτηση *ATE(void)*

   Με την συνάρτηση αυτή στέλνουμε στο κινητό τηλέφωνο την AT εντολή *ATE 0* για απενεργοποιήσουμε την εκ’ νέου αποστολή των χαρακτήρων που λαμβάνει το κινητό τηλέφωνο. Η συνάρτηση αυτή καλείται πριν την αποστολή της AT εντολής *AT+CMGR=1* για το διάβασμα ενός μηνύματος αφού στην περίπτωση αυτή οι διακοπές είναι ενεργοποιημένες με αποτέλεσμα

εκτός του μηνύματος να αποθηκεύονται στην αρχή οι χαρακτήρες της εντολής διαβάσματος (*AT+CMGR=1*).

5.3.3 Συναρτήσεις Καθυστέρησης

Τα αρχεία *delay.h / delay.c* περιλαμβάνουν την δήλωση και τον ορισμό των συναρτήσεων με τις οποίες θα επιτυγχάνεται καθυστέρηση κατά την διάρκεια εκτέλεσης της ροής του προγράμματος. Ο ορισμός των συναρτήσεων γίνεται με βάση την συχνότητα του κρυστάλλου της εφαρμογής αφού είναι ο παράγοντας που καθορίζει την ταχύτητα με τη οποία θα εκτελούνται οι εντολές από τον μικροελεγκτή.

5.4 Παράθεση Κώδικα

5.4.1 Αρχείο main.c

```
/*
*********************************************************************
Όνομα αρχείου: main.c
*********************************************************************
Περιλαμβάνονται τα αρχεία: "main.h","sci.h", "sci.c", "delay.h",
"delay.c"
*********************************************************************
Μεταφραστής γλώσσας (compiler): PICC έκδοση 7.87PL2 της εταιρίας HI-
TECH Software
*********************************************************************
Αναπτυξιακό περιβάλλον: MPLAB IDE έκδοση 6.40.00 της εταιρίας
Microchip Technology Inc
*********************************************************************
Προγραμματιστής: PIC STAR PLUS) της εταιρίας Microchip Technology Inc
*********************************************************************
Τύπος μικροελεγκτή: PIC16F877 της εταιρίας Microchip Technology Inc
*********************************************************************
Συσκευή κινητού τηλεφώνου: T28s της εταιρίας Sony Ericsson Mobile
Communications AB
*********************************************************************
Σχόλια: Το πρόγραμμα που ακολουθεί αποτελεί το λογισμικό τμήμα της
διπλωματικής εργασίας με τίτλο "Απομακρυσμένος Έλεγχος Μέσω Σύντομων
Γραπτών Μηνυμάτων (SMS)" που έγινε στο Εργαστήριο Μικροϋπολογιστών
και Ψηφιακών Συστημάτων του τμήματος Ηλεκτρολόγων Μηχανικών του
Εθνικού Μετσόβιου Πολυτεχνείου.
*********************************************************************
*/

#include<pic.h>
#include<stdio.h>
#include<string.h>
__CONFIG(0x3F32);

#include "delay.h"
```

```
#include "delay.c"
#include "sci.h"
#include "sci.c"
#include "main10.h"

#define SMS_REPLAY 0

void device_config(void);
unsigned char get_Hex(void);
unsigned char get_Sept(unsigned char hex);
void change(int k);
void sendPDU(const unsigned char *ch);
void DectoHex(void);
void packing(void);
void SEND_SMSSTATUS(void);
void Send_OK(void);
void Send_ERROR(void);


/* interrupt service routine */
static void interrupt
isr(void)
{
      /* serial service interrupt */
      if(RCIF&&RCIE){

         data=sci_GetByte();

         if(addr>67){ //Αποθήκευση μηνύματος
               buf[addr-68]=data;
               w++;
         }
      else if(addr>35 && addr<48) //Αποθήκευση αριθμού τηλεφώνου
              TEL[addr-36]=data;
      else if(addr==7)
               taf=data; //Αποθήκευση ειδοποίησης νέου μηνύματος

      addr++;
      sci_CheckOERR(); //Έλεγχος λάθους κατά την μετάδοση
         }


   /* RB0/INT change interrupt */
   if(INTF&&INTE){

         bit_set(STATE,SWITCH_ON);

         INTF=0; //Flag Enable interrupt
   }

} //end of isr

void main(void)
{
   PORTD=0xFF; //Τα σήματα ελέγχου όλα κλειστά
   RB7=1;      //LED λειτουργίας – Ανοικτό

   device_config();

   for(i1=0;i1<80;i1++)
   buf[i1]=0x30;
```

```
/* 4 sec delay */
for(timer=16;timer;timer--)
      DelayMs(250);

sendAT(CNMI); //Καθορισμός ειδοποίησης νέου μηνύματος
DelayMs(250);

sendAT(CPMS); //Καθορισμός μνήμης για αποθήκευση μηνύματος
DelayMs(250);

PEIE=1; //Enable Peripheral interrupt
INTE=1; //Enable RB0/INT interrupt
GIE=1;  //Enable General interrupt

RB7=0;  //LED λειτουργίας – Κλειστό


  /* Ατέρμον βρόχος ελέγχου */
  while(1){

        /* Διαδικασία εξυπηρέτησης SMS */
        if(taf=='T'){

              device_config();

              for(timer=16;timer;timer--)
                    DelayMs(250);

              for(i1=0;i1<80;i1++)
                    buf[i1]=0x30;

              ATE(); //Disable echo - GIE=0

              PEIE=1;GIE=1;
              addr=0;

              /****** Ανάγνωση Μηνύματος  ******/
              sendAT(CMGR);

              for(timer=20;timer;timer--) //2 sec
                    DelayMs(250);

              GIE=0;PEIE=0; //Απενεργοποίηση διακοπών

              /******** Αποκωδικοποίηση Μηνύματος ********/
              q=0;
              count1=0;

              n_Sept=((w/14)*8) + ((w%14)/2);

              for(count=0;count<n_Sept;count++){
                    if(!bit_test(STATE,FLAG)){
                          Hex=get_Hex();
                          q++;
                    }
                    else
                          bit_clear(STATE,FLAG);

                    data=get_Sept(Hex);
                    buf[count1]=data;
```

```
                        count1++;
                  }
            /* Από το σημείο αυτό στον πίνακα buf[] περιέχεται το
            μήνυμα που έστειλε ο χρήστης στην αρχική του μορφή */

                  buf[count1-3]='\0';
                  TEL[12]='\0';

                  FLAG_ON=0;FLAG_OFF=0;

                  /****** Έλεγχος εγκυρότητας μηνύματος ******/

             /* Ο έλεγχος γίνεται σε ένα ατέρμον βρόχο ο οποίος
             διασχίζεται μόνο μια φορά. Στην περίπτωση που
             τουλάχιστον ένα σημείο ελέγχου δεν ικανοποιείται τίθεται
             η σημαία λάθους (ERROR_FLAG=1;) και ο έλεγχος φεύγει από
             τον βρόχο αφού δεν απαιτείται η εξέταση των υπολοίπων
             σημείων ελέγχου. Αν ο έλεγχος φτάσει στο τέλος του
             βρόχου το μήνυμα είναι έγκυρο και ακολουθεί έξοδος από
             αυτόν
             */


            while(1)
            {

/*Check point 1*/
//Έλεγχος για την αποφυγή μηνύματος μεγάλου μήκους
                  if(strlen(buf)>MAX_LENGHT){
                        ERROR_FLAG=1;
                  }
                  if(ERROR_FLAG)
                        break;

                  mikos1=strlen(buf);

/*Check point 2*/
//Έλεγχος κωδικού μηνύματος
                  if(strncmp(buf,"123 ",4)){
                        ERROR_FLAG=1;
                  }
                  if(ERROR_FLAG)
                        break;

/*Check point 3*/
//Για την αποφυγή μηνυμάτων με την μορφή "123_"
                  if(strcmp(buf,"123") && mikos1==4 ){
                        ERROR_FLAG=1;
                  }
                  if(ERROR_FLAG)
                        break;


/*Check point 4*/
/*Για να αποφύγουμε μηνύματα που περιέχουν περισσότερους από δύο
συνεχόμενους κενούς χαρακτήρες π.χ."123_ON_8____4_5".*/
                  i=0;
                        while(buf[i]!='\0'){
                              if(buf[i]==' ' && buf[i+1]==' '){
                                    ERROR_FLAG=1;
```

```
                                    break;
                              }
                              i++;
                        }
                        if(ERROR_FLAG)
                              break;


/*Check point 5*/
//Για να αποφύγουμε την ύπαρξη δύο η περισσοτέρων ON ή OFF
                        pch1 = strstr (buf,"ON");
                        i=0;
                        if(pch1!=NULL){
                              do{
                                    pch1 = strstr (pch1+1,"ON");
                                    i++;
                                    if(i>1){
                                          ERROR_FLAG=1;
                                          break;
                                    }
                              }while(pch1!=NULL);
                        }
                        if(ERROR_FLAG)
                              break;

                        pch1 = strstr (buf,"OFF");
                        i=0;
                        if(pch1!=NULL){
                              do{
                                    pch1 = strstr (pch1+1,"OFF");
                                    i++;
                                    if(i>1){
                                          ERROR_FLAG=1;
                                          break;
                                    }
                              }while(pch1!=NULL);
                        }
                        if(ERROR_FLAG)
                              break;

//Για να αποφύγουμε την ύπαρξη "ON OFF" ή "OFF ON"
                        for(i=0;i<6;i++){
                              if(strcmp(buf,ONOFF[i])==0){
                                    ERROR_FLAG=1;
                                    break;
                              }
                        }
                        if(ERROR_FLAG)
                              break;

                        break; /* Τέλος του βρόχου
                                  Έγκυρο μήνυμα
                                  Έξοδος από τον βρόχο
                                */

                  }//end of while(1) for check points

                  //Έλεγχος για την   εντολή STATUS.
                  if(!strcmp(buf,"123 STATUS"))
                        STATUS_FLAG=1;
```

```
        /* Εδώ καλείται η συνάρτηση strtok(buf," ")
           για πρώτη φορά */

           pch=strtok(buf," ");

           pch=strtok(NULL," ");

           while (pch!=NULL && ERROR_FLAG!=1)
           {
                 if(strlen(pch)==1){
                       if(FLAG_ON==FLAG_OFF){
                             ERROR_FLAG=1;
                             break;
                       }
                       /* Έλεγχος ορισμάτων από 1-8 */
                       if(strpbrk(pch,key)!=NULL){
                       switch(*pch){
                             case '1':
                                   change(1);
                                   break;
                             case '2':
                                   change(2);
                                   break;
                             case '3':
                                   change(3);
                                   break;
                             case '4':
                                   change(4);
                                   break;
                             case '5':
                                   change(5);
                                   break;
                             case '6':
                                   change(6);
                                   break;
                             case '7':
                                   change(7);
                                   break;
                             case '8':
                                   change(8);
                                   break;
                             }
                             pch=strtok(NULL," ");
                                   continue;
                       }
                       else{
                             ERROR_FLAG=1;
                             break;
                       }
                 }// end of if(strlen(pch)==1){

     /* Έλεγχος λέξεων από τον πίνακα *words[] */
        for(i=0;i<7;i++){
              if(strcmp (pch,words[i])==0)
              {
                    switch(i){
                    case (0): //Για την λέξη "ON"
                          FLAG_ON=1;
                          FLAG_OFF=0;
                          break;
                    case (1): //Για την λέξη "OFF"
```

```
                        FLAG_OFF=1;
                        FLAG_ON=0;
                        break;
                  case (2): //Για την λέξη "STATUS"
                        if(STATUS_FLAG==1)
                        { //Έλεγχος αν όλες οι συσκευές είναι
                          //ανενεργές

                        if(!mPORTD){

                  /** Αποστολή SMS "All devices are closed" **/
                              strcpy (buf,startSMS);
                              strcat (buf,TEL);
                              strcat (buf,SMSplus);
                              strcat (buf,STATUSOFF_MSG);

                              buf[70]='\0';

                              sendAT(CMGSSTATUS0FF);



                              for(timer=6;timer;timer--)
                                   DelayMs(250);

                              sendPDU(buf);


                              for(timer=16;timer;timer--)
                                   DelayMs(250);
                  /***************************************/

                              break;
                        }

            /* Κάποιες από τις συσκευές βρίσκονται σε λειτουργία.
               Αποστολή κατάλληλου μηνύματος. */

                              SEND_SMSSTATUS();
                                    break;
                        }
                        else

/* Αν ο έλεγχος του προγράμματος φτάσει σε αυτό το σημείο σημαίνει
ότι έχουμε  την λέξη STATUS αλλά όχι στην σωστή μορφή δηλ. 123
STATUS", οπότε τίθεται η σημαία λάθους. */

                        ERROR_FLAG1=1;
                              break;
                        case (3): //Για την λέξη "ALL"
                              if(FLAG_ON==FLAG_OFF){
                                    ERROR_FLAG1=1;
                                    break;
                              }
                              if((FLAG_ON==0)&&(FLAG_OFF==1)){
                                          for(n=1;n<9;n++)
                                          change(n);
                              }
                              if((FLAG_ON==1)&&(FLAG_OFF==0)){
                                          for(n=1;n<9;n++)
                                          change(n);
```

```
                        }

                              break;
                        case (4):  //Για την λέξη "FOS"
                              if(FLAG_ON==FLAG_OFF){
                                    ERROR_FLAG1=1;
                                    break;
                                    }
                                    change(SWH1);
                                    break;
                        case (5): //Για την λέξη "THERMO"
                              if(FLAG_ON==FLAG_OFF){
                                    ERROR_FLAG1=1;
                                    break;
                                    }
                                    change(SWH2);
                                    break;
                        case (6):  //Για την λέξη "FUN"
                              if(FLAG_ON==FLAG_OFF){
                                    ERROR_FLAG1=1;
                                    break;
                                    }
                                    change(SWH3);
                                    break;
                              }
                              pch=strtok(NULL," ");
                              ERROR_FLAG=0;
                              break;
                        }
                        else
                              ERROR_FLAG=1;
                  }//end of for(i=0;i<5;i++)
            }//end of while (pch!=NULL && ERROR_FLAG!=1)

            GIE=0;PEIE=0;
            ATE();

            sendAT(CMGD);  //Διαγραφή πρώτου SMS

            for(timer=8;timer;timer--) //2 sec delay
                  DelayMs(250);

            if(STATUS_FLAG==0)
            {
                  if(ERROR_FLAG||ERROR_FLAG1)
                        {
                        /* Error indication */
                        for(count=3;count;count--)
                        {
                              RB7=1; // closed
                              for(timer=4;timer;timer--)
                                    DelayMs(250);
                              RB7=0; // open
                              for(timer=4;timer;timer--)
                                    DelayMs(250);
                        }

                        mPORTD=ex_mPORTD;

                        /*** Output ***/
                        PORTD=~mPORTD;
```

```
                              ERROR_FLAG=0;ERROR_FLAG1=0;

#if SMS_REPLAY==1
                              /* Αποστολή SMS με τη λέξη "ERROR" */

                              Send_ERROR();

                              /**********************************/
#endif

                              }
                        else{
                              ex_mPORTD=mPORTD;

                              /*** Output ***/
                              PORTD=~mPORTD;

#if SMS_REPLAY==1

                              /* Αποστολή SMS με τη λέξη "OK" */

                              Send_OK();

                              /*********************************/
#endif
                        }
                  }

            /************************************************/

                  ATE();
                  GIE=0;PEIE=0;

                  sendAT(CMGD); //Διαγραφή πρώτου SMS
                  DelayMs(250);
                  DelayMs(250);

                  sendAT(CNMI); //Καθορισμός ειδοποίησης
                  DelayMs(250);

                  sendAT(CPMS);
                  DelayMs(250); //Καθορισμός μνήμης

                  /* Αρχικοποίηση μεταβλητών - Σημαιών */
                  addr=0;
                  taf=0;
                  shift=0;
                  okt=0;
                  w=0;

                  ERROR_FLAG=0;
                  ERROR_FLAG1=0;
                  STATUS_FLAG=0;
                  bit_clear(STATE,FLAG);

                  PEIE=1;INTE=1;GIE=1;
        }//end of if(data1[7]=='T'){
```

```
        /************ ALARM SIGNAL Check point *************/

            if(bit_test(STATE,SWITCH_ON))
            {
                  GIE=0;PEIE=0; //Απενεργοποίηση διακοπών.

                  /* Αποστολή SMS "ALARM SIGNAL DETECTED" */

                  for(i1=50;i1<75;i1++)
                        buf[i1]=0x30;

                  strcpy (buf,startSMS);
                  strcat (buf,ALARM_NUMBER);
                  strcat (buf,SMSplus);
                  strcat (buf,ALARM_MSG);

                  buf[68]='\0';

                  sendAT(CMGSALARM);

                  for(timer=4;timer;timer--) //1 sec delay
                        DelayMs(250);

                  sendPDU(buf);

                  for(timer=16;timer;timer--) //4 sec delay
                        DelayMs(250);

                  sci_PutByte(0x1B);  //ESC
                  DelayMs(250);

      /*******************************************/

                  ATE();

                  sendAT(CNMI);
                  DelayMs(250);

                  sendAT(CPMS);
                  DelayMs(250);

                  addr=0;
                  taf=0;

                  bit_clear(STATE,SWITCH_ON);

                  PEIE=1;INTE=1;GIE=1; //Ενεργοποίηση διακοπών
            }

      }//end of while(1) main Loop

}//end of main

/***************** Τέλος Προγράμματος ********************/
```

```
/*************** Ορισμός Συναρτήσεων ********************/

/* Συνάρτηση αρχικοποίησης */

void device_config(void)
{
  /* USART SETUP */
     sci_Init(9600,SCI_EIGHT);

  /* PERIPHERAL SETUP */

  /* RB0/INT */
    TRISB0=1;  //RB0 Είσοδος
    INTEDG=1;  //RISING
    INTE=1;    //Ενεργοποίηση διακοπής RB0/INT

    TRISB7=0;  //Για το LED ένδειξης λειτουργίας

    TRISD=0x00;//Η PORTD καθορίζεται ως έξοδος

    TRISC7=1;  //RC7 RC6 Είσοδοι για τους ακροδέκτες Rx Tx
    TRISC6=1;
}


unsigned char get_Hex(void)
{

      for(j=0;j<2;j++){
            switch(buf[(2*q)+j]){
            case '0':
                  temp[j]=0x00;
                  break;
            case '1':
                  temp[j]=0x01;
                  break;
            case '2':
                  temp[j]=0x02;
                  break;
            case '3':
                  temp[j]=0x03;
                  break;
            case '4':
                  temp[j]=0x04;
                  break;
            case '5':
                  temp[j]=0x05;
                  break;
            case '6':
                  temp[j]=0x06;
                  break;
            case '7':
                  temp[j]=0x07;
                  break;
            case '8':
                  temp[j]=0x08;
                  break;
            case '9':
                  temp[j]=0x09;
                  break;
            case 'A':
```

```
                  temp[j]=0x0A;
                  break;
            case 'B':
                  temp[j]=0x0B;
                  break;
            case 'C':
                  temp[j]=0x0C;
                  break;
            case 'D':
                  temp[j]=0x0D;
                  break;
            case 'E':
                  temp[j]=0x0E;
                  break;
            case 'F':
                  temp[j]=0x0F;
                  break;
            }
            if(j==0)
                  temp[j]=temp[j]<<4;
      }
    Hex=temp[0]|temp[1];
    return Hex;
}


/* Συνάρτηση αποκωδικοποίησης */

unsigned char get_Sept(unsigned char hex)
{
      temp[0]=ex_Hex;
      temp[0]>>=(8-shift);
      temp[1]=Hex;
      temp[1]<<=shift;
      Sept=temp[0]|temp[1];
      Sept=bit_clear(Sept,7);
      ex_Hex=Hex;
      shift++;
      if(shift==7)
            bit_set(STATE,FLAG);
      if(shift==8){
            shift=0;
            okt++;
      }
      return Sept;
}

/* Συνάρτηση καθορισμού σημάτων στην πόρτα ελέγχου */

void change(int k)
{
      if(FLAG_ON)
            bit_set(mPORTD,k-1);
      if(FLAG_OFF)
            bit_clear(mPORTD,k-1);

}



/* Συνάρτηση αποστολής ακολουθίας PDU */
```

```
void sendPDU(const unsigned char *ch)
{
      while(*ch!='\0')
      {
            sci_PutByte(*ch);
            *ch++;
      }
      sci_PutByte(0x1A);   //Ctrl/z
}

/* Συνάρτηση μετατροπής δεκαδικών σε δεκαεξαδικών */

void DectoHex(void)
{
      switch(temp[0])
      {
      case (0):
            temp[0]='0';
            break;
      case (1):
            temp[0]='1';
            break;
      case (2):
            temp[0]='2';
            break;
      case (3):
            temp[0]='3';
            break;
      case (4):
            temp[0]='4';
            break;
      case (5):
            temp[0]='5';
            break;
      case (6):
            temp[0]='6';
            break;
      case (7):
            temp[0]='7';
            break;
      case (8):
            temp[0]='8';
            break;
      case (9):
            temp[0]='9';
            break;
      case (10):
            temp[0]='A';
            break;
      case (11):
            temp[0]='B';
            break;
      case (12):
            temp[0]='C';
            break;
      case (13):
            temp[0]='D';
            break;
      case (14):
            temp[0]='E';
```

```
            break;
      case (15):
            temp[0]='F';
            break;

      }
}

 /* Συνάρτηση κωδικοποίησης */

 void packing(void)
      {
            do
            {
                  temp[0]=buf[i];
                  temp[0]>>=shift;
                  temp[1]=buf[i+1];
                  temp[1]<<=(7-shift);
                  Pak[okt]=temp[0]|temp[1];
                  shift++;
                  if(shift==7){
                     shift=0;
                     i++;
                  }
                  i++;okt++;
            }while(buf[i]!='\0');


            do
            {
                  for(j=0;j<2;j++)
                  {
                        temp[0]=Pak[q];
                        if(j==0)
                              temp[0]>>=4;
                        else
                              temp[0]&=0x0F;

                        switch(temp[0]){
                        case (0):
                              buf[(2*q)+j]='0';
                              break;
                        case (1):
                              buf[(2*q)+j]='1';
                              break;
                        case (2):
                              buf[(2*q)+j]='2';
                              break;
                        case (3):
                              buf[(2*q)+j]='3';
                              break;
                        case (4):
                              buf[(2*q)+j]='4';
                              break;
                        case (5):
                              buf[(2*q)+j]='5';
                              break;
                        case (6):
                              buf[(2*q)+j]='6';
                              break;
                        case (7):
```

```
                              buf[(2*q)+j]='7';
                              break;
                        case (8):
                              buf[(2*q)+j]='8';
                              break;
                        case (9):
                              buf[(2*q)+j]='9';
                              break;
                        case (10):
                              buf[(2*q)+j]='A';
                              break;
                        case (11):
                              buf[(2*q)+j]='B';
                              break;
                        case (12):
                              buf[(2*q)+j]='C';
                              break;
                        case (13):
                              buf[(2*q)+j]='D';
                              break;
                        case (14):
                              buf[(2*q)+j]='E';
                              break;
                        case (15):
                              buf[(2*q)+j]='F';
                              break;
                        }//end of case
                  }//end of for(j=0;j<2;j++){
                  q++;
            }while(q!=2*okt);

            buf[q]='\0';

      }

/* Συνάρτηση αποστολής μηνύματος για την εντολή STATUS */

 void SEND_SMSSTATUS(void)
 {
    /*Σύνταξη κατάλληλου μηνύματος */
      strcpy (buf,"Active");

      for(n=0;n<9;n++)
      {
            if(bit_test(mPORTD,n))
                  strcat (buf,translate[n]);
      }

      /******** Αποστολή SMS "Active 1 2 3 ......." ********/


      lenght=strlen(buf);  //Υπολογισμός μήκους μηνύματος
                           //TP-User-Data-Length
      temp[0]=lenght/16;
      DectoHex();
      mikos[0]=temp[0];

      temp[0]=lenght%16;
      DectoHex();
      mikos[1]=temp[0];
```

```
mikos[2]='\0';

i=0;
shift=0;
okt=0;
q=0;

packing();  //Κωδικοποίησης μηνύματος

i=0;
while((Pak[i] = buf[i]) != '\0')
       i++;

strcpy (buf,startSMS);
strcat (buf,TEL);
strcat (buf,SMSplus);
strcat (buf,mikos);
strcat (buf,Pak);

/** Υπολογισμός μήκους για την εντολή "AT+CMGS=.." **/

lenght=(strlen(buf)/2)-1; //Το μήκος δε πρέπει να ξεπερνά
                          // το 100
temp[0]=lenght/10;
temp[0]=temp[0]|0x30;
temp[0]=(char)temp[0];
mikos[0]=temp[0];

temp[0]=lenght%10;
temp[0]=temp[0]|0x30;
temp[0]=(char)temp[0];
mikos[1]=temp[0];

mikos[2]='\0';

/** Επισύναψη του μήκους στην εντολή "AT+CMGS=.." **/

sci_PutByte('A');
sci_PutByte('T');
sci_PutByte('+');
sci_PutByte('C');
sci_PutByte('M');
sci_PutByte('G');
sci_PutByte('S');
sci_PutByte('=');
sci_PutByte(mikos[0]);
sci_PutByte(mikos[1]);
sci_PutByte(0x0D);

for(timer=6;timer;timer--) //1.5 sec delay
     DelayMs(250);

sendPDU(buf);

for(timer=24;timer;timer--) //6 sec delay
     DelayMs(250);

sci_PutByte(0x1B);  //ESC
DelayMs(250);

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

```
 }

void Send_OK(void)
{
      for(i1=0;i1<45;i1++)
            buf[i1]=0x30;

      strcpy (buf,startSMS);
      strcat (buf,TEL);
      strcat (buf,SMSplus);
      strcat (buf,OK_MSG);

      buf[34]='\0';

      sendAT(CMGSOK);

      for(timer=4;timer;timer--)
            DelayMs(250);

      sendPDU(buf);

      for(timer=16;timer;timer--)
            DelayMs(250);

      sci_PutByte(0x1B);  //ESC
      DelayMs(250);

}


void Send_ERROR(void)
{
      for(i1=0;i1<45;i1++)
            buf[i1]=0x30;

      strcpy (buf,startSMS);
      strcat (buf,TEL);
      strcat (buf,SMSplus);
      strcat (buf,ERROR_MSG);

      buf[40]='\0';

      sendAT(CMGSERROR);

      for(timer=4;timer;timer--)
            DelayMs(250);

      sendPDU(buf);

      for(timer=16;timer;timer--)
            DelayMs(250);

      sci_PutByte(0x1B);  //ESC
      DelayMs(250);

}
```

Στο αρχείο main.c ορίζονται 10 συναρτήσεις η κάθε μια με καθορισμένη λειτουργία. Οι συναρτήσεις αυτές είναι:

1. Συνάρτηση *void device_config(void)*

   Η συνάρτηση καλείται στην αρχή και σκοπώ έχει την αρχικοποίηση του συστήματος.

2. Συνάρτηση *unsigned char get_Hex(void)*

   Η συνάρτηση αυτή καλείται κατά την διαδικασία αποκωδικοποίησης του μηνύματος. Η λειτουργία της είναι να διαβάζει από δύο διαδοχικές θέσεις μνήμης τους ASCII χαρακτήρες μιας PDU ακολουθίας και να επιστρέφει ένα byte με την δεκαεξαδική κωδικοποίηση της κάθε μιας. Η διαδικασία απεικονίζεται στον πίνακα 5.1

ΧΑΡΑΚΤΗΡΑΣ	**ASCII**	**Hex Κωδικοποίηση**			**Byte που επιστρέφεται**
0	00110000	0	0	0000	0 C 0000 1100
C	01000011	12	C	1100	
F	01000110	15	F	1111	F 4 1111 0100
4	00110100	4	4	0100	

Πίνακας 5.1 *Λειτουργία συνάρτησης get_Hex()*

3. Συνάρτηση *char get_Sept(unsigned char hex)*

   Η συνάρτηση αυτή υλοποιεί το αλγόριθμο αποκωδικοποίησης που περιγράφτηκε στο κεφάλαιο 3. Δέχεται ως όρισμα ένα byte από την συνάρτηση *get_Hex(void)* ενώ επιστέφει ένα χαρακτήρα από το μήνυμα που στάλθηκε στο κινητό τηλέφωνο βάσης.

4. Συνάρτηση *change(int k)*

   Η συνάρτηση αυτή χρησιμοποιείται για την παραγωγή των σημάτων ελέγχου από την πόρτα εξόδου D. Δέχεται ως όρισμα ένα αριθμό (από το 1-8) που αντιστοιχεί σε κάποια από τις συσκευές και ανάλογα την εντολή που υπάρχει (με το ποία σημαία είναι ενεργοποιημένη *FLAG_ON, FLAG_OFF*) θέτει ή απενεργοποιεί τον αντίστοιχο ακροδέκτη της πόρτας.

5. Συνάρτηση *sendPDU(const unsigned char *ch)*

Η συνάρτηση αυτή χρησιμοποιείται για την αποστολή μιας PDU ακολουθίας από τον μικροελεγκτή προς τον κινητό τηλέφωνο. Καλείται στην περίπτωση που θέλουμε να στείλουμε ένα σύντομο μήνυμα SMS και ειδικότερα μετά την αποστολή της εντολής *AT+CMGS*.

6. Συνάρτηση *DectoHex(void)*

   Η συνάρτηση αυτή μετατρέπει αριθμούς από την δεκαδική στην δεκαεξαδική τους μορφή. Χρησιμοποιείται για την επισύναψη του μήκους ενός μηνύματος στην PDU ακολουθία κατά τη διαδικασία αποστολής μηνύματος μετά από μια εντολή STATUS.

7. Συνάρτηση *packing(void)*

   Η συνάρτηση αυτή υλοποιεί τον αλγόριθμο κωδικοποίησης για την κατάλληλη μετατροπή των δεδομένων και την επισύναψη τους στην PDU ακολουθία του μηνύματος που θέλουμε να αποστείλουμε.

8. Συνάρτηση *SEND_SMSSTATUS(void)*

   Η συνάρτηση αυτή καλείται μετά από μια εντολή STATUS στην περίπτωση που κάποιες από συσκευές είναι ενεργοποιημένες και απαιτείται η αποστολή ενός σύντομου μηνύματος για την ενημέρωση του χρήστη. Κατά την εκτέλεση της συνάρτησης αρχικά συντάσσεται το μήνυμα που πρέπει να αποσταλεί, ακολουθεί η κωδικοποίηση και η δημιουργία της PDU ακολουθίας. Η συνάρτηση ολοκληρώνεται με την αποστολή των εντολών και της ακολουθίας που θα προκαλέσουν την αποστολή από τον κινητό τηλέφωνο.

9. Συνάρτηση Send_OK(void)

   Η συνάρτηση αυτή καλείται μετά από ένα επιτυχή χειρισμό του συστήματος και προκαλεί την αποστολή ενός γραπτού μηνύματος με την λέξη “OK” για να ενημερώσει κατάλληλα τον χρήστη.

10. Συνάρτηση Send_ERROR(void)

    Η συνάρτηση αυτή καλείται μετά από ένα ανεπιτυχή χειρισμό του συστήματος και προκαλεί αντίστοιχα την αποστολή μηνύματος με την λέξή “ERROR” για ενημερώσει τον χρήστη ότι κατά τη απόπειρα ελέγχου παρουσιάστηκε σφάλμα.

5.4.2 Αρχείο main.h

```
/* Όνομα αρχείου: main.h
   Περιγραφή: Περιέχει τις δηλώσεις των μεταβλητών του αρχείο
              main.c
*/

/************ Μακροεντολές *********************/
#define bit_test(var,bit)  ((var) & (1 << (bit)))
#define bit_set(var,bit)  ((var) |= (1 << (bit)))
#define  bit_clear(var,bit)  ((var) &= ~(1 << (bit)))

/*********Byte για την αποθήκευση σημαιών **************/
unsigned char STATE=0x00;
#define FLAG 0      // =1 Flag was set
#define SWITCH_ON 1 // =1 Switch was set

/************* Μέγιστο μήκος μηνύματος ******************/
#define MAX_LENGHT 35

/*************** Καθορισμός λέξεων **********************/
#define WORD1 "FOS"
#define WORD2 "THERMO"
#define WORD3 "FAN"

/*************** Καθορισμός συσκευών *****************/
#define SWH1 8  //Από 1-8
#define SWH2 7
#define SWH3 6

/* Καθορισμός αριθμού τηλεφώνου για την αποστολή του μηνύματος ALARM
*/
#define ALARM_NUMBER "039697432005"


/****************** Καθορισμός AT εντολών  ************************/
const unsigned char CPMS[]="AT+CPMS=\"ME\",\"ME\",\"ME\""; /*
Καθορισμός αποθήκευσης
                                                          μνήμης
του SMS */
const unsigned char CMGR[]="AT+CMGR=1";   //Διάβασμα του πρώτου SMS
const unsigned char CMGD[]="AT+CMGD=1";   //Διαγραφή του πρώτου SMS
const unsigned char CNMI[]="AT+CNMI=3,1,2,0";//Καθορισμός ειδοποίησης
                                              νέου SMS


#asm
      psect   eedata,delta=2,abs,ovrld
      org     2100h
    db      0,0x0d,0x56,0x43,0x14,0x55,0x4b,0
#endasm

unsigned char data;
int addr=0;


bank1 unsigned char buf[80];
bank2 unsigned char TEL[13];
bank3 unsigned char Pak[80];
```

```
bank2 int timer; //Για την καθυστέρηση

/*Για την συνάρτηση get_Hex()*/
unsigned char Hex,temp[2];
bank2 int q,j,a1;

/*Για την συνάρτηση get_Sept(unsigned char hex);*/
unsigned char Sept,ex_Hex;
bank2 int shift=0;


unsigned char data;
bank2 int count1;
bank2 int count;
bank2 int okt=0;
bank2 int w=0;

/*************************/
char * pch;
const char * pch1;
const char *words[]={"ON","OFF","STATUS","ALL",WORD1,WORD2,WORD3};
const char *ONOFF[]={"123 ON OFF","123 OFF ON","123 ON","123
OFF","123 ON ","123 OFF "};
const char *translate[9]={" 1"," 2"," 3"," 4"," 5"," 6"," 7"," 8"};
const char key[]="12345678";
bank2 int ERROR_FLAG=0;
bank2 int FLAG_ON=0;
bank2 int FLAG_OFF=0;
bank2 int ERROR_FLAG1=0;  //Για τα ορίσματα FOS THERMO FUN
bank2 int STATUS_FLAG=0;

bank2 int i,k,n;
unsigned char mPORTD=0x00;
unsigned char ex_mPORTD=0x00;


bank2 int i1;

unsigned char taf=0;

bank2 int n_Sept;

bank2 int mikos1;
bank2 unsigned char mikos[3];
bank2 int lenght;

/****************** Καθορισμός μηνυμάτων **********************/
const unsigned char startSMS[]="0011000C91";
const unsigned char SMSplus[]="0000AA";

/* "OK" */
const unsigned char CMGSOK[]="AT+CMGS=16";
const unsigned char OK_MSG[]="02CF25";

/* "ERROR" */
const unsigned char CMGSERROR[]="AT+CMGS=19";
const unsigned char ERROR_MSG[]="0545A9F42905";

/* "All devices are closed" */
const unsigned char CMGSSTATUS0FF[]="AT+CMGS=34";
```

```
const unsigned char
STATUSOFF_MSG[]="1641361B442EDBD3E3F21C1496974163F67B5E2603";

/* "ALARM SIGNAL DETECTED" */
const unsigned char CMGSALARM[]="AT+CMGS=33";
const unsigned char
ALARM_MSG[]="15416650DA044D93476790092216A9C521B54804";
```

5.4.3 Αρχείο sci.c

```
/*
**********************************************************************
Όνομα αρχείου: sci.c
**********************************************************************
Περιλαμβάνονται τα αρχεία: "pic.h","sci.h"
**********************************************************************
Σχόλια: Αποτελεί μέρος του προγράμματος main.c .Γίνονται ο ορισμός συναρτήσεων
που αφορούν τον Πομπός / Δέκτη Ασύγχρονης / Σύγχρονης Σειριακής Επικοινωνίας
(USART).
**********************************************************************
*/

#include <pic.h>
#include "sci.h"

/* Συνάρτηση με την οποία γίνεται αρχικοποίηση της σειριακής */

unsigned char
sci_Init(unsigned long int baud, unsigned char ninebits)
{
     int X;
     unsigned long tmp;

     /* Υπολογισμός του baud rate */
     /* για ασύγχρονη λειτουργία */
     tmp = 16UL * baud;
     X = (int)(FOSC/tmp) - 1;
     if((X>255) || (X<0))
     {
       tmp = 64UL * baud;
       X = (int)(FOSC/tmp) - 1;
       if((X>255) || (X<0))
       {
             return 1;   /* panic - baud rate unobtainable */
       }
       else
             BRGH = 0;   /* low baud rate */
     }
     else
     BRGH = 1;     /* high baud rate */
     SPBRG = X;    /* set the baud rate */

     SYNC = 0;     /* Ασύγχρονη */
     SPEN = 1;     /* enable serial port pins */
     CREN = 1;     /* Ενεργοποίηση λήψης */
     SREN = 0;     /* no effect */
     TXIE = 0;     /* Απενεργοποίηση tx διακοπών */
```

```
    RCIE = 1;     /* Ενεργοποίηση rx διακοπών */
    TX9  = ninebits?1:0;      /* 8- ή 9-bit αποστολής */
    RX9  = ninebits?1:0;      /* 8- ή 9-bit λήψης */
    TXEN = 1;     /* Ενεργοποίηση πομπού  */

    return 0;
}

/* Συνάρτηση για την αποστολή byte */

void
sci_PutByte(unsigned char byte)
{
    while(!TXIF)  /* set when register is empty */
      continue;
    TXREG = byte;

    return;
}

/* Συνάρτηση για την λήψη byte */

unsigned char
sci_GetByte(void)
{
    while(!RCIF)  /* set when register is not empty */
      continue;

    return RCREG; /* RXD9 and FERR are gone now */
}

unsigned char
sci_CheckOERR(void)
{
    if(OERR)      /* re-enable after overrun error */
    {
      CREN = 0;
      CREN = 1;
      return 1;
    }

    return 0;
}

#define sci_PutNinth(bitnine) (TX9D = bitnine?1:0;)

unsigned char
sci_GetNinth(void)
{
    while(!RCIF)
      continue;

    return RX9D;  /* RCIF is not cleared until RCREG is read */
}

unsigned char
sci_GetFERR(void)
{
    while(!RCIF)
      continue;
```

```
    return FERR;  /* RCIF is not cleared until RCREG is read */
}

/* Συνάρτηση για την αποστολή AT εντολών */

void sendAT(const unsigned char *ch)
{
    while(*ch!='\0')
       {
    sci_PutByte(*ch);
    *ch++;
    }
       sci_PutByte(0x0D);   //enter
}

/* Συνάρτηση για την αποστολή της AT εντολής "ATE 0" */

void ATE(void)     /*close echo*/
{
    GIE=0;
    sci_PutByte('A');
    sci_PutByte('T');
    sci_PutByte('E');
    sci_PutByte('0');
    sci_PutByte(0x0D);
    DelayMs(100);
    GIE=1;
}
```

5.4.4 Αρχείο sci.h

```
/*
*********************************************************************
Όνομα αρχείου: sci.h
*********************************************************************
Σχόλια: Αποτελεί μέρος του προγράμματος sci.c .Γίνονται  οι δηλώσεις συναρτήσεων
και σταθερών που αφορούν τον Πομπός / Δέκτη Ασύγχρονης / Σύγχρονης Σειριακής
Επικοινωνίας (USART) και ορίζονται στο αρχείο sci.c
*********************************************************************
*/

#define FOSC      (20000000L)
#define SCI_EIGHT (0)
#define SCI_NINE  (1)

unsigned char     sci_Init(unsigned long int, unsigned char);
void        sci_PutByte(unsigned char);
unsigned char     sci_GetByte(void);
void        sci_PutNinth(unsigned char);
unsigned char     sci_GetNinth(void);
unsigned char     sci_GetFERR(void);
unsigned char     sci_CheckOERR(void);
void sendAT(const unsigned char *ch);
void ATE(void);
```

5.4.5 Αρχείο delay.c

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

Όνομα αρχείου: delay.c

```
**********************************************************************
```

Σχόλια: Αποτελεί μέρος του προγράμματος main.c .Γίνεται ο ορισμός των συναρτήσεων με τις οποίες θα προκαλείται καθυστέρηση και την διάρκεια εκτέλεσης ροής του προγράμματος

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

#include    "delay.h"

/* Συνάρτηση για καθυστέρηση milliseconds */

void
DelayMs(unsigned char cnt)
{
#if   XTAL_FREQ <= 2MHZ
     do {
       DelayUs(996);
     } while(--cnt);
#endif

#if    XTAL_FREQ > 2MHZ
     unsigned char i;
     do {
       i = 4;
       do {
             DelayUs(250);
       } while(--i);
     } while(--cnt);
#endif
}
```

5.4.6 Αρχείο delay.h

```
/*
*********************************************************************
Όνομα αρχείου: delay.h
*********************************************************************
```

Σχόλια: Αποτελεί μέρος του προγράμματος delay.c Γίνονται οι δηλώσεις συναρτήσεων και σταθερών για την δημιουργία των συναρτήσεων καθυστέρησης.
Ορίζονται οι συναρτήσεις :

DelayUs(x) Για καθυστέρηση σε microseconds
DelayMs(x) Για καθυστέρηση σε milliseconds

Το x δεν πρέπει να ξεπερνά το 255

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

#ifndef     XTAL_FREQ
#define     XTAL_FREQ   4MHZ        /* Συχνότητα κρυστάλλου σε MHz */
#endif

#define     MHZ   *1000L            /* Αριθμός kHz σε ένα MHz */
#define     KHZ   *1

#if   XTAL_FREQ >= 12MHZ

/* Συνάρτηση για καθυστέρηση microseconds */

#define     DelayUs(x)  { unsigned char _dcnt; \
                  _dcnt = (x)*((XTAL_FREQ)/(12MHZ)); \
                  while(--_dcnt != 0) \
                       continue; }
#else

#define     DelayUs(x)  { unsigned char _dcnt; \
                  _dcnt = (x)/((12MHZ)/(XTAL_FREQ))|1; \
                  while(--_dcnt != 0) \
                        continue; }
#endif

extern void DelayMs(unsigned char);
```

5.5 Χάρτης Μνήμης

Για την μετάφραση (compiling) του προγράμματος της εφαρμογής χρησιμοποιήθηκε ο μεταφραστής *PICC* της εταιρίας *HI-TECH (έκδοση 7.87PL2).* Μια από τις δυνατότητες αυτού του μεταφραστή είναι η παράθεση του χάρτη μνήμης του μικροελεγκτή που προγραμματίζεται ώστε να γνωρίζουμε κάθε στιγμή την κατανομή και την επάρκεια των πόρων του συστήματος. Ο χάρτης μνήμης του μικροελεγκτή της εφαρμογής για το πρόγραμμα που παρουσιάστηκε φαίνεται στο πίνακα 5.2 .

```
Memory Usage Map:

Program ROM   $0000 - $0134  $0135 (   309) words
Program ROM   $013A - $07FF  $06C6 (  1734) words
Program ROM   $0803 - $10F8  $08F6 (  2294) words
Program ROM   $13AE - $17FF  $0452 (  1106) words
                             $1543 (  5443) words total Program ROM

Bank 0 RAM    $0020 - $007F  $0060 (    96) bytes total Bank 0 RAM
Bank 1 RAM    $00A0 - $00EF  $0050 (    80) bytes total Bank 1 RAM
Bank 2 RAM    $0110 - $0149  $003A (    58) bytes total Bank 2 RAM
Bank 3 RAM    $0190 - $01DF  $0050 (    80) bytes total Bank 3 RAM
Config Data   $0000 - $2107  $2108 (  8456) words
Config Data   $2007 - $2007  $0001 (     1) words
                             $2109 (  8457) words total Config Data

Program statistics:

Total ROM used     5443 words (66.4%)
Total RAM used      314 bytes (85.3%)
```

***Πίνακας** 5.2 Χάρτης μνήμης μικροελεγκτή*

Όπως μπορούμε να δούμε έχουμε χρησιμοποιήσει το 66% της μνήμης προγράμματος *ROM (Program Memory)* και το 85% της μνήμης δεδομένων *RAM (Data Memory).* Κατά την διαδικασία δέσμευσης μνήμης έπρεπε να δώσουμε ιδιαίτερη προσοχή στην μνήμη δεδομένων πρώτον διότι οι απαιτήσεις του προγράμματος σε μεταβλητές ήταν μεγάλες και δεύτερον γιατί θα έπρεπε να κατανεμηθούν στα διαφορά τμήματα (Bank) ανάλογα με τη λειτουργία τους. Για την εξοικονόμηση θέσεων μνήμης δεδομένων χρησιμοποιήσαμε την μνήμη προγράμματος για την αποθήκευση μεταβλητών το περιεχόμενο των οποίων μένει σταθερό σε όλη την διάρκεια εκτέλεσης του προγράμματος. Οι μεταβλητές αυτές δηλώνονται με το προσδιοριστικό *const* και

αποθηκεύονται στην μνήμη προγράμματος σε χώρο που έχει αρχικά δεσμευθεί. Για την κατανομή των μεταβλητών στα διάφορα τμήματα έπρεπε εκ των προτέρων να καθοριστεί η λειτουργία τους και στην συνέχεια να δηλωθούν τα τμήματα στα οποία θα βρίσκονται ανάλογα με τις απαιτήσεις που ορίζονται από τον μεταφραστή (compiler). Έτσι στο Bank 0 έχουν αποθηκευτή όλες οι μεταβλητές δείκτη, στο Bank 1 έχει οριστεί ο πίνακας buf [] 80 θέσεων στον οποίο θα αποθηκεύεται το μήνυμα, στο Bank 2 έχουν οριστεί μεταβλητές γενικού σκοπού ενώ στο Bank 3 ορίζεται ένας επιπλέον πίνακας 80 θέσεων ο Pak[] ο οποίος χρησιμοποιείται για την προσωρινή αποθήκευση δεδομένων κατά την διαδικασία σύνταξης και αποστολής ενός σύντομου μηνύματος.

5.6 Προγραμματισμός Μικροελεγκτή

Ο προγραμματισμός του μικροελεγκτή γίνεται μέσο του προγραμματιστή *PIC STAR PLUS*. Η συσκευή αυτή δίνεται από την εταιρία *Microchip Technology Inc* και συνδέεται με τον ηλεκτρονικό υπολογιστή μέσο της συριακής θύρας. Το λογισμικό λειτουργίας του προγραμματιστή περιέχεται στο λογισμικό ανάπτυξης *MPLAB IDE* και μέσου αυτού γίνεται η φόρτωση με τα αρχεία αντικειμενικού κώδικα (.HEX) που προήρθαν από την επιτυχή μετάφραση του προγράμματος της εφαρμογής, στην μνήμη προγράμματος του μικροελεγκτή. Κατά τον προγραμματισμό ο μικροελεγκτής αποσπάται από την πλακέτα του συστήματος και τοποθετείται στον προγραμματιστή μέσα σε ειδική εγκοπή ώστε οι ακροδέκτες του να εφάπτονται με αυτούς του προγραμματιστή. Στην συνέχεια μέσο του λογισμικού ανάπτυξης *MPLAB IDE* φορτώνουμε τα αρχεία αντικειμενικού κώδικα (.HEX) της εφαρμογής στην μνήμη προγράμματος του μικροελεγκτή. Αφού η διαδικασία ολοκληρωθεί με επιτυχία τοποθετούμε τον μικροελεγκτή στην πλακέτα του συστήματος και τον θέτουμε σε λειτουργία ώστε να ξεκινήσει η εκτέλεση του προγράμματος.

Παράρτημα

Καθορισμένη GSM αλφάβητο 7-bits

#	character	ASCII Code	bits	#	character	ASCII Code	Bits
0	@	64	01000000	38	&	38	00100110
1	£	163	10100011	39	'	39	00100111
2	$	36	00100100	40	(	40	00101000
3	¥	165	10100101	41	)	41	00101001
4	θ	952	1110111000	42	*	42	00101010
5	ι	953	1110111001	43	+	43	00101011
6	ω	969	1111001001	44	,	44	00101100
7	μ	956	1110111100	45	-	45	00101101
8	ς	962	1111000010	46	.	46	00101110
9	Η	919	1110010111	47	/	47	00101111
10		10	00001010	48	0	48	00110000
11	Ψ	936	1110101000	49	1	49	00110001
12	ψ	968	1111001000	50	2	50	00110010
13		13	00001101	51	3	51	00110011
14	Ε	917	1110010101	52	4	52	00110100
15	ε	949	1110110101	53	5	53	00110101
16	Δ	916	1110010100	54	6	54	00110110
17	_	95	01011111	55	7	55	00110111
18	Φ	934	1110100110	56	8	56	00111000
19	Γ	915	1110010011	57	9	57	00111001
20	Λ	923	1110011011	58	:	58	00111010
21	Ω	937	1110101001	59	;	59	00111011
22	Π	928	1110100000	60	<	60	00111100
23	Ψ	936	1110101000	61	=	61	00111101
24	Σ	931	1110100011	62	>	62	00111110
25	Θ	920	1110011000	63	?	63	00111111
26	Ξ	926	1110011110	64	¨	901	1110000101
27	€	8364	10000010101100	65	A	65	01000001
28	Z	918	1110010110	66	B	66	01000010
29	ζ	950	1110110110	67	C	67	01000011
30	ί	943	1110101111	68	D	68	01000100
31	Ι	921	1110011001	69	E	69	01000101
32		32	00100000	70	F	70	01000110
33	!	33	00100001	71	G	71	01000111
34	"	34	00100010	72	H	72	01001000
35	#	35	00100011	73	I	73	01001001
36	¤	164	10100100	74	J	74	01001010
37	%	37	00100101	75	K	75	01001011

#	character	ASCII Code	bits	#	character	ASCII Code	bits
76	L	76	01001100	105	i	105	01101001
77	M	77	01001101	106	j	106	01101010
78	N	78	01001110	107	k	107	01101011
79	O	79	01001111	108	l	108	01101100
80	P	80	01010000	109	m	109	01101101
81	Q	81	01010001	110	n	110	01101110
82	R	82	01010010	111	o	111	01101111
83	S	83	01010011	112	p	112	01110000
84	T	84	01010100	113	q	113	01110001
85	U	85	01010101	114	r	114	01110010
86	V	86	01010110	115	s	115	01110011
87	W	87	01010111	116	t	116	01110100
88	X	88	01011000	110	n	110	01101110
89	Y	89	01011001	111	o	111	01101111
90	Z	90	01011010	112	p	112	01110000
91	Δ	916	1110010100	113	q	113	01110001
92	Φ	934	1110100110	114	r	114	01110010
93	Ρ	929	1110100001	115	s	115	01110011
94	ά	940	1110101100	117	u	117	01110101
95	§	167	10100111	118	v	118	01110110
96	Ώ	911	1110001111	119	w	119	01110111
97	a	97	01100001	120	x	120	01111000
98	b	98	01100010	121	y	121	01111001
99	c	99	01100011	122	z	122	01111010
100	d	100	01100100	123	δ	948	1110110100
101	e	101	01100101	124	φ	966	1111000110
102	f	102	01100110	125	ρ	961	1111000001
103	g	103	01100111	126	ό	972	1111001100
104	h	104	01101000	127	ΰ	944	1110110000

Λεξιλόγιο

AUC	AUthentication Center
BSC	Base Station Controller
BTS	Base Transceiver Station
CPU	Central Processing Unit
EIR	Equipment Identity Register
ESTI	European Telecommunication Institute
GSM	Global System for Mobile communications
HLR	Home Location Register
ISC	International Switching Center
ISDN	Integrated Service Digital Network
ISR	Interrupt Service Routine
MS	Mobile Station
MSC	Mobile Switching Center
MT	Mobile Terminal
OERR	Overrun Error Bit
OMC	Operation and Maintenance Center
PDN	Public Data Network
PDU	Protocol Description Unit
PSTN	Public Switched Telephone Network
SCI	Synchronous Communications Interface
SIM	Subscriber Identity Module
SMS	Short Message Service
SMSC	SMS Center
TE	Terminal Equipment
TP PID.	Protocol Identifier
TP SCTS.	Time Stamp
TP UD	User Data
TP-DCS	Data Coding Scheme
TP-SCTS	Service Centre Time Stamp
TP-UDL	User Data Length

VLR	Visited Location Register

Βιβλιογραφία

[1] Κ.Ζ.Πεκμεστζή, "Συστήματα Μικροϋπολογιστών" Συμμέτρια,1995

[2] Κ.Ζ.Πεκμεστζή, "Εργαστήριο Μικροϋπολογιστών", 2003

[3] Myke Predko "Προγραμματίζοντας τον Μικροελεγκτή PIC", Τζιόλα,1998

[4] T28 AT Command Reference

[5] Microchip Web Page, *http://www.microchip.com/*

[6] Sony Ericsson Web Page, *http://www.sonyericsson.com/*

[6] HI-TECH Software, *http://www.htsoft.com/*

[7] ETSI Institute Web Page, *http://www.etsi.org/*