ΤΕΙ ΚΕΝΤΡΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

ΙΩΑΝΝΗΣ ΚΑΛΟΜΟΙΡΟΣ

Εισαγωγή στους μικροελεγκτές PIC

Σημειώσεις για το μάθημα "Ενσωματωμένα Συστήματα"

ΣΕΡΡΕΣ 2015

1. Εισαγωγή στα ενσωματωμένα συστήματα

1.1 Τι είναι τα ενσωματωμένα συστήματα

Ενσωματωμένα συστήματα είναι μικρά σε μέγεθος υπολογιστικά συστήματα, που ο άμεσος ρόλος τους δεν είναι η επεξεργασία δεδομένων. Εξυπηρετούν αυτοματισμούς και βρίσκονται πλέον σε κάθε τύπο ηλεκτρονικής συσκευής καθημερινής χρήσης, όπως κάμερες, οικιακές συσκευές, συσκευές γραφείου. Απαντώνται σε συστήματα μετρήσεων, σε παιχνίδια, σε βιομηχανικούς αυτοματισμούς, σε αυτοκίνητα και όλα τα μέσα μεταφοράς, σε ηλεκτρονικά όργανα και τηλεπικοινωνιακά συστήματα. Τα ενσωματωμένα συστήματα αποτελούν την ταχύτερα αναπτυσσόμενη βιομηχανία και αλλάζουν την καθημερινότητα, καθώς διεισδύουν παντού. Αποτελούν τη βάση της τεχνολογίας που αναφέρεται ως διαδίκτυο πραγμάτων (Internet of Things).

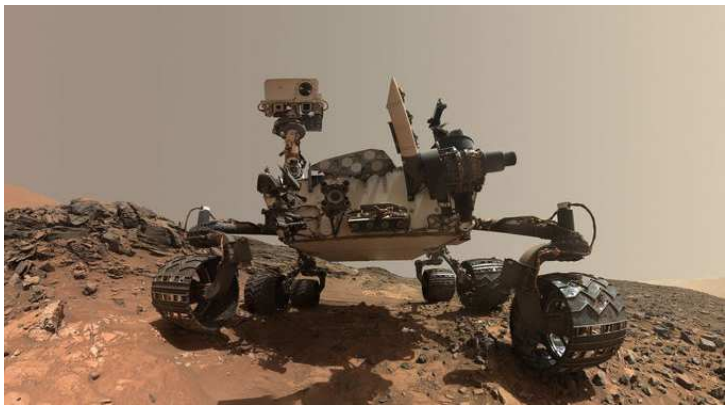
Anti-lock brakes
Auto-focus cameras
Automatic teller machines
Automatic toll systems
Automatic transmission
Avionic systems
Battery chargers
Camcorders
Cell phones
Cell-phone base stations
Cordless phones
Cruise control
Curbside check-in systems
Digital cameras
Disk drives
Electronic card readers
Electronic instruments
Electronic toys/games
Factory control
Fax machines
Fingerprint identifiers
Home security systems
Life-support systems
Medical testing systems
Modems
MPEG decoders
Network cards
Network switches/routers
On-board navigation
Pagers
Photocopiers
Point-of-sale systems
Portable video games
Printers
Satellite phones
Scanners
Smart ovens/dishwashers
Speech recognizers
Stereo systems
Teleconferencing systems
Televisions
Temperature controllers
Theft tracking systems
TV set-top boxes
VCR's, DVD players
Video game consoles
Video phones
Washers and dryers

Εικόνα 1.1 Συσκευές που περιέχουν ενσωματωμένα συστήματα

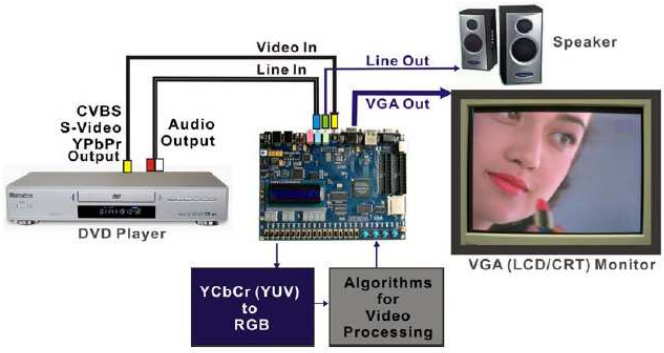
Τα ενσωματωμένα συστήματα αλληλεπιδρούν με τον πραγματικό κόσμο, καθώς συλλέγουν πληροφορίες και ανταποκρίνονται σ' αυτές. Για το όγο αυτό, πολλές φορές λειτουργούν ωα συστήματα πραγματικού χρόνου, καθώς είναι κρίσιμο να ανταποκρίνονται και να παράγουν τις εξόδους τους σε καθορισμένα χρονικά διαστήματα.

Παρακάτω δίνονται εικόνες συσκευών που η λειτουργία τους στηρίζεται σε ενσωματωμένη υπολογιστική νοημοσύνη. Μια μεγάλη κατηγορία τέτοιων συστημάτων είναι οι διάφοροι ρομποτικοί αυτοματισμοί, όπως οχήματα αυτόνομης πλοήγησης. Επίσης πολύ σημαντικά ενσωματωμένα συστήματα ειναι οι διάφορες πολυμεσικές

εφαρμογές, δηλαδή συστήματα επεξεργασίας εικόνας και ήχου, που απαντώνται σε συσκευές καταναλωτικών ηλεκτρονικών (mp3 players, αποκωδικοποιητές, τηλεοράσεις κλπ.)



Εικόνα 1.2 Το εξερευνητικό ρόβερ "Curiosity" στον Άρη



Εικόνα 1.3 Συστήματα επεξεργασίας σήματος

Τα ενσωματωμένα συστήματα στηρίζουν τη λειτουργία τους σε αισθητήρες, με τους οποίους συλλέγουν πληροφορία από τον περιβάλλον. Τέτοιοι αισθητήρες μετρούν φυσικά μεγέθη όπως θερμοκρασία, πίεση, υγρασία, ταχύτητα, επιτάχυνση. Σε πολλές περιπτώσεις ειναι σύνθετοι αισθητήρες, όπως κάμερες ή υπερηχητικοί αισθητήρες απόστασης. Η τεχνολογία επιτρέπει πλέον τη δημιουργία αισθητήρων με τη μορφή σύνθετων ηλεκτρομηχανικών συστημάτων (MEMS-micro-electro-mechanical systems) που αναπτύσσονται σε τσιπ πυριτίου και ενσωματώνουν και κυκλώματα μετατροπής του αναλογικού σήματος σε ψηφιακό ή και μονάδες επεξεργασίας. Τέτοια συστήματα ονομάζονται «έξυπνοι αισθητήρες».

Εικόνα 1.4 Αισθητήρας επιτάχυνσης τύπου MEMS.

Τα ενσωματωμένα συστήματα αποτελούν πλέον σημαντικό μέρος των εκπαιδευτικών προγραμμάτων που είναι προσανατολισμένα στην τεχνολογία. Η προσέγγισή τους γίνεται μέσω εκπαιδευτικών μικροελεγκτών, όπως ο Arduino και με εκπαιδευτικά ρομποτικά αναπτύγματα, όπως VEX, Tetrix, Lego Mindstorms κλπ.

Εικόνα 1.5 Η εκπαιδευτική διάσταση των ενσωματωμένων συστημάτων

Κατά τη σχεδίαση ενσωματωμένων συστημάτων λαμβάνονται υπόψη κάποιες «μετρικές», οι οποίες πρέπει να βελτιστοποιούνται. Αυτές αναφέρονται στα επόμενα.

1.2 Μετρικές σχεδίασης ενσωματωμένων συστημάτων

Το ζητούμενο είναι να σχεδιάσουμε ένα σύστημα σύμφωνα με τις προδιαγραφές. Το σύστημα πρέπει να υλοποιεί την επιθυμητή λειτουργικότητα. Όμως, πρέπει να βελτιστοποιεί και κάποιες σχεδιαστικές μετρικές. Οι σχεδιαστικές μετρικές είναι μετρήσιμα χαρακτηριστικά του συστήματος. Αν δεν βελτιστοποιηθούν, το προϊόν καταδικάζεται σε αποτυχία:

- ✓ NRE cost (Non-Recurrent Engineering cost): το κόστος της σχεδίασης και ανάπτυξης του συστήματος
- ✓ Κόστος Μονάδας (Unit cost): το κόστος της κατασκευής μιας μονάδας μείον το κόστος NRE
- ✓ Μέγεθος (Size): το μέγεθος κάθε μονάδας
- ✓ Απόδοση (performance): ο χρόνος εκτέλεσης της μονάδας έργου (throughput)
- ✓ Ισχύς: Το ποσό της ισχύος που δαπανά το σύστημα
- ✓ Ευελιξία: πόσο εύκολα το σύστημα μπορεί να αλλάξει λειτουργικότητα
- ✓ Χρόνος πρωτοτυποποίησης (time to prototype)
- ✓ Χρόνος πρόσβασης στην αγορά (time-to-market)

Οι παραπάνω μετρικές συχνά είναι αντικρουόμενες. Έτσι, η αύξηση της απόδοσης, συχνά αυξάνει το μέγεθος του συστήματος, ενώ η μείωση της δαπανώμενης ισχύος αυξάνει το κόστος.

1.3 Τεχνολογίες ενσωματωμένων συστημάτων

Υπάρχουν τρεις τεχνολογίες-κλειδιά που αφορούν στα ενσωματωμένα συστήματα:

- ✓ Τεχνολογία επεξεργαστών
- ✓ Τεχνολογία Ολοκληρωμένων κυκλωμάτων
- ✓ Τεχνολογία σχεδίασης

Οι τεχνολογίες αυτές αλληλοεπικαλύπτονται: έαν ολοκληρωμένο κύκλωμα μπορεί να περιέχει επεξεργαστή της μιας ή της άλλης τεχνολογίας και να είναι σχεδιασμένο με την μια ή την άλλη τεχνολογία σχεδίασης.

Θα εξετάσουμε με συντομία τις παραπάνω τεχνολογίες:

1.3.1 Τεχνολογίες επεξεργαστών

α. *Επεξεργαστές γενικού σκοπού*: Προγραμματιζόμενες διατάξεις ικανές να ανταποκριθούν στις ανάγκες πολλών εφαρμογών. Έχουν **μεγάλη RAM** (εσωτερική ή εξωτερική), και συνήθως στηρίζονται σε **αρχιτεκτονική Von-Neumann** και ένα γενικό διάδρομο επεξεργασίας (data path). Παραδείγματα: Pentium, 8088, 68000…

β. *Εξειδικευμένοι επεξεργαστές* (application-specific). Αποτελούν κατηγορία προγραμματιζόμενων επεξεργαστών που είναι βελτιστοποιημένοι για συγκεκριμένη κατηγορία εφαρμογών. Έχουν περιορισμένη **RAM** πάνω στο τσιπ, συνήθως υλοποιούν αρχιτεκτονική **Harvard**, η επεξεργασία τους είναι ειδικού σκοπού και έχουν ενσωματωμένα πολλά **περιφερειακά κυκλώματα**. Επίσης, έχουν **μνήμη προγράμματος**. Παραδείγματα: microcontrollers, DSP processors

γ. *Επεξεργαστές μοναδικού σκοπού* (single purpose): Ψηφιακά κυκλώματα σχεδιασμένα να εκτελούν μόνον μία διεργασία. Περιλαμβάνουν μόνον τα απαραίτητα κυκλώματα που χρειάζονται ώστε να εκτελέσουν ένα μοναδικό πρόγραμμα. Δεν έχουν μνήμη προγράμματος, αλλά σε ορισμένες περιπτώσεις χρειάζονται λίγη RAM. Παραδείγματα: επιταχυντές υλικού, όπως video compressors, ψηφιακά φίλτρα, επεξεργαστές μηχανικής όρασης. Οι περιφερειακοί ελεγκτές (κυκλώματα επικοινωνίας με θύρες I/O και με ICs) θεωρούνται επίσης επεξεργαστές μοναδικού σκοπού.

Οι επεξεργαστές γενικού σκοπού και εξειδικευμένοι επεξεργαστές είναι προγραμματιζόμενοι, άρα για να λειτουργήσουν πρέπει να γράψουμε κώδικα γι’

αυτούς. Από αυτή την άποψη αντιπροσωπεύουν το μέρος του συστήματος που ταυτίζεται με το ΛΟΓΙΣΜΙΚΟ.

Στους επεξεργαστές μοναδικού σκοπού, το κύκλωμα «είναι» το πρόγραμμα, άρα δεν γράφουμε κώδικα γι' αυτούς. Αντιπροσωπεύουν το ΥΛΙΚΟ μέρος του συστήματος.

Στα ενσωματωμένα συστήματα, το υλικό και το λογισμικό σχεδιάζονται παράλληλα, κάτι που αναφέρεται ως συνσχεδιασμός υλικού-λογισμικού (hardware-software co-design).

1.3.2 Τεχνολογίες ολοκληρωμένων κυκλωμάτων

α. Πλήρως προσαρμοσμένα (Full custom/VLSI). Όλα τα στρώματα του κυκλώματος σχεδιάζονται από την αρχή, από τα τραζίστορ μέχρι τις διασυνδέσεις.

Β. Μερικώς προσαρμοσμένα (Semi-custom ASIC). Τα χαμηλότερα στρώματα είναι μερικώς σχεδιασμένα. Η σχεδίαση γίνεται με τοποθέτηση και διασύνδεση προσχεδιασμένων βαθμίδων.

Γ. Διαμορφούμενες διατάξεις PLD (CPLDs/FPGAs). Όλα τα στρώματα υπάρχουν εξαρχής μέσα στο τσιπ. Η σχεδίαση γίνεται με τη διαμόρφωση των πινάκων αναφοράς (LUT) και των προγραμματιζόμενων διασυνδέσεων.

1.3.3 Τεχνολογίες σχεδίασης

Για τη δημιουργία της επιθυμητής λειτουργικότητας έχουν αναπτυχθεί διάφορα εργαλεία σχεδίασης.
Για το λογισμικό διαθέτουμε ολοκληρωμένα περιβάλλοντα για ανάπτυξη λογισμικού, όπως assemblers, μεταγλωττιστές, προσομοιωτές και προγραμματιστές
Για το υλικό μέρος διαθέτουμε εργαλεία σχεδίασης τύπου CAD και γλώσσες περιγραφής υλικού. Τα εργαλεία αυτά επιτρέπουν τη σχεδίαση σε διάφορα επίπεδα:

- στο επίπεδο του ολοκληρωμένου συστήματος (system level)
- Στο επίπεδο της συμπεριφοράς του κυκλώματος (behavioral level)
- Στο δομικό επίπεδο της μεταφοράς σημάτων ανάμεσα σε πύλες και καταχωρητές (Register-transfer level).

2. Εισαγωγή στους μικροελεγκτές

2.1 Τι είναι ένας Μικροελεγκτής

Ένας μικροελεγκτής είναι ένα μικρό *υπολογιστικό κύκλωμα*, σχεδιασμένο σε ένα και μόνο ολοκληρωμένο κύκλωμα υψηλής κλίμακας ολοκλήρωσης. Όπως κάθε υπολογιστικό κύκλωμα, περιέχει κεντρική μονάδα επεξεργασίας, έναν αριθμό καταχωρητών, κυκλώματα μνήμης και κυκλώματα ελέγχου περιφερειακών συσκευών. Κάθε μικροελεγκτής είναι λοιπόν ικανός να ανταλλάξει σήματα με το εξωτερικό περιβάλλον, να εκτελέσει πράξεις ανάμεσα σε μεταβλητές και να καταχωρήσει κάποιες τιμές στη μνήμη RAM που διαθέτει.

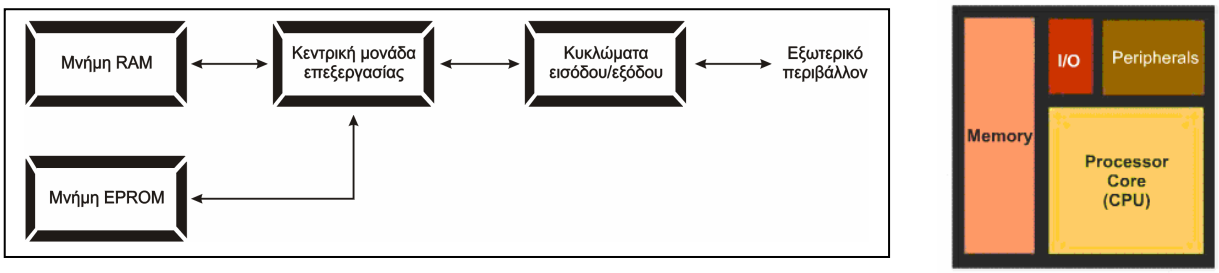
Κάθε μικροελεγκτής περιέχει μέσα σε ένα και μοναδικό ολοκληρωμένο κύκλωμα τα παρακάτω στοιχεία:

- έναν αριθμό από καταχωρητές ειδικού σκοπού (συσσωρευτή, καταχωρητή κατάστασης, μετρητή προγράμματος, καταχωρητή εντολών, καταχωρητή δείκτη).
- εσωτερικούς χρονιστές - απαριθμητές.
- αριθμητική και λογική μονάδα (ALU).
- μονάδα αποκωδικοποίησης εντολών.

Βασικά στοιχεία ενός μικροελεγκτή αποτελούν:

- η μνήμη προγράμματος (ROM ή EPROM) και
- η μνήμη καταχωρητών / μεταβλητών (RAM).

Στο Σχ. 2.1 φαίνεται η βασική δομή ενός μικροελεγκτή.



Σχήμα 2.1 Βασική δομή ενός μικροελεγκτή

Στους μικροελεγκτές διακρίνουμε επίσης

- τα κυκλώματα χρονισμού και ελέγχου

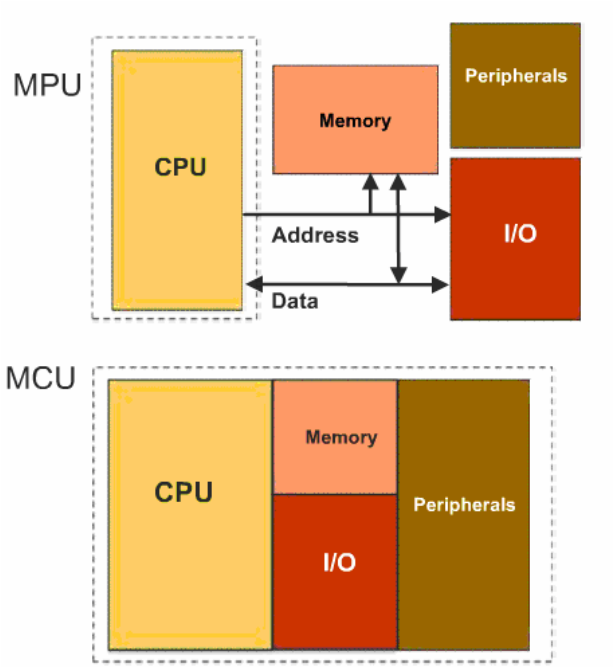
Τέλος, βασικά μέρη ενός μικροελεγκτή είναι

- παράλληλες θύρες εισόδου/εξόδου
- άλλα περιφερειακά κυκλώματα (UART, A/D μετατροπείς κλπ.)

Μέσα από τις θύρες I/O ένας μικροελεγκτής μπορεί να δέχεται σήματα εισόδου με τη μορφή λογικών ψηφιακών καταστάσεων, χαρακτήρες ή bytes δεδομένων με την τεχνική της ασύγχρονης ή της σύγχρονης σειριακής επικοινωνίας, σήματα διακοπών, ή σε ορισμένες περιπτώσεις και αναλογικά σήματα, τα οποία στη συνέχεια μετατρέπονται σε ψηφιακά. Επίσης μπορεί να αποστέλλει σήματα σε άλλες συσκευές μέσα από θύρες εξόδου, να οδηγεί ηλεκτρονόμους, διόδους LED και άλλα κατάλληλα κυκλώματα, που συνήθως περιλαμβάνονται σε κάθε μορφής αυτοματισμό.

Οι μικροελεγκτές χαρακτηρίζονται από ένα περιορισμένο ρεπερτόριο εντολών, οι οποίες μπορούν να γραφούν σε συμβολική μορφή (*assembly*), με τη βοήθεια μνημονικών ονομάτων. Στους μικροελεγκτές PIC μεσαίας τάξης (midrange), το μήκος της εντολής σε γλώσσα μηχανής είναι 14 bits, τα οποία καταχωρούνται στη μνήμη προγράμματος, τύπου EEPROM. Για τα εργαλεία που χρησιμοποιούνται για τον σκοπό αυτό θα μιλήσουμε σε επόμενη παράγραφο.

Σε τι διαφέρει ένας μικροελεγκτής από έναν συνηθισμένο μικροεπεξεργαστή; Ο μικροελεγκτής είναι ένα μικρό *αυτόνομο* υπολογιστικό σύστημα, προγραμματισμένο να



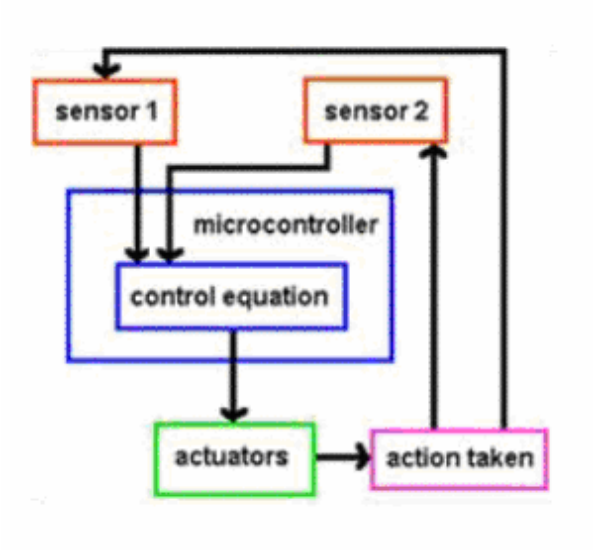
Σχ. 2.2 Διαφορές ανάμεσα σε σύστημα μικροεπεξεργαστή (MPU) και μικροελεγκτή (MCU).

εκτελεί μία συγκεκριμένη λογική ακολουθία εντολών, οι οποίες έχουν καταχωρηθεί στην προγραμματιζόμενη μόνιμη μνήμη του. Κάθε φορά που θα επανεκκινείται ο μικροελεγκτής, θα εκτελεί την ίδια λογική. Θα ανακαλεί τα δεδομένα, θα τα επεξεργάζεται και με βάση τα αποτελέσματα της επεξεργασίας θα ελέγχει το περιβάλλον του. Πρόκειται, δηλαδή, για σύστημα **ειδικού σκοπού**, *αφιερωμένο* (*dedicated*) στον έλεγχο και την εξυπηρέτηση ενός συγκεκριμένου αυτοματισμού.

Αντίθετα, ένας μικροεπεξεργαστής μετά την εκκίνησή του δεν είναι από μόνος του σε θέση να εκτελέσει κάποια λογική ακολουθία. Αν και μπορεί να συνδεθεί με μνήμες RAM και ROM, αυτές αποτελούν ξεχωριστές μονάδες, που συνήθως δεν ολοκληρώνονται μέσα στον ίδιο τον μικροεπεξεργαστή. Οι διαφορές αυτές φαίνονται στο Σχ. 2.2.

Το παρακάτω σχήμα 2.3 δείχνει σχηματικά το βρόχο ελέγχου που εκτελεί ένας μικροελεγκτής, σε μια τυπική ενσωματωμένη εφαρμογή. Ο μικροελεγκτής λειτουργεί ως ψηφιακός ελεγκτής συλλέγοντας δεδομένα από τους αισθητήρες και ενεργοποιώντας εξωτερικά κυκλώματα, όπως κινητήρες, ηλεκτρονόμους, κυκλώματα ισχύος κλπ. Αντίστοιχα με όσα περιγράψαμε στην παράγραφο 1.2.1, ο μικροελεγκτής λαμβάνει ως είσοδο την τρέχουσα κατάσταση του συστήματος που ελέγχει, συνήθως με τη μορφή ενός αναλογικού σήματος, που προέρχεται από έναν ή περισσότερους αισθητήρες. Το σήμα αυτό ψηφιοποιείται, με τη βοήθεια ενός μετατροπέα αναλογικού σήματος σε ψηφιακό και συγκρίνεται με μια τιμή αναφοράς, που περιγράφει την επιθυμητή κατάσταση του συστήματος. Με βάση την απόκλιση της τρέχουσας κατάστασης από την επιθυμητή κατάσταση αναφοράς, λαμβάνεται μια απόφαση και παράγεται ένα σήμα «οδήγησης», που επιδρά πάνω στο ελεγχόμενο σύστημα και μεταβάλλει την κατάστασή του. Για το σκοπό αυτό, ο μικροελεγκτής διαθέτει κατάλληλες θύρες εισόδου/εξόδου, με τις οποίες διασυνδέεται με τον εξωτερικό κόσμο. Επίσης διαθέτει περιφερειακά, όπως κυκλώματα παραγωγής παλμών μεταβαλλόμενου εύρους (PWM) για οδήγηση κινητήρων, μετατροπής αναλογικού σήματος σε ψηφιακό (ADC), θύρες επικοινωνίας κ.ά. Ο ψηφιακός ελεγκτής είναι συνήθως σύστημα πραγματικού χρόνου.

Μια άλλη κατηγορία ενσωματωμένων συστημάτων είναι οι επεξεργαστές ψηφιακού σήματος (Digital Signal Processors ή DSP). Αυτά τα κυκλώματα είναι αφιερωμένα στην γρήγορη επεξεργασία σημάτων, συνήθως ήχου και εικόνας. Ο σκοπός τους είναι να επιτελέσουν μέσα σε συγκεκριμένη χρονική προθεσμία τις μαθηματικές πράξεις που απαιτούνται από τις ανάγκες επεξεργασίας του σήματος. Τέτοιες πράξεις είναι συχνά πολλαπλασιασμοί και αθροίσεις, όπως αυτές της σχέσης (1.1), η οποία περιγράφει ένα τυπικό ψηφιακό φίλτρο. Για το σκοπό αυτό, οι DSP επεξεργαστές διαθέτουν κατάλληλη αρχιτεκτονική και αριθμητικές μονάδες ώστε να εκτελούν με ταχύτητα πολλαπλασιασμούς και συσσωρεύσεις (Multiply Accumulate ή MAC). Εκτός από το κατάλληλο data path οι DSP επεξεργαστές διαθέτουν και περιφερειακά κυκλώματα για διακίνηση δεδομένων μεγάλου όγκου και με την κατάλληλη ψηφιακή κωδικοποίηση, σύμφωνα με πρωτόκολλα που χρησιμοποιούνται για δεδομένα ήχου και



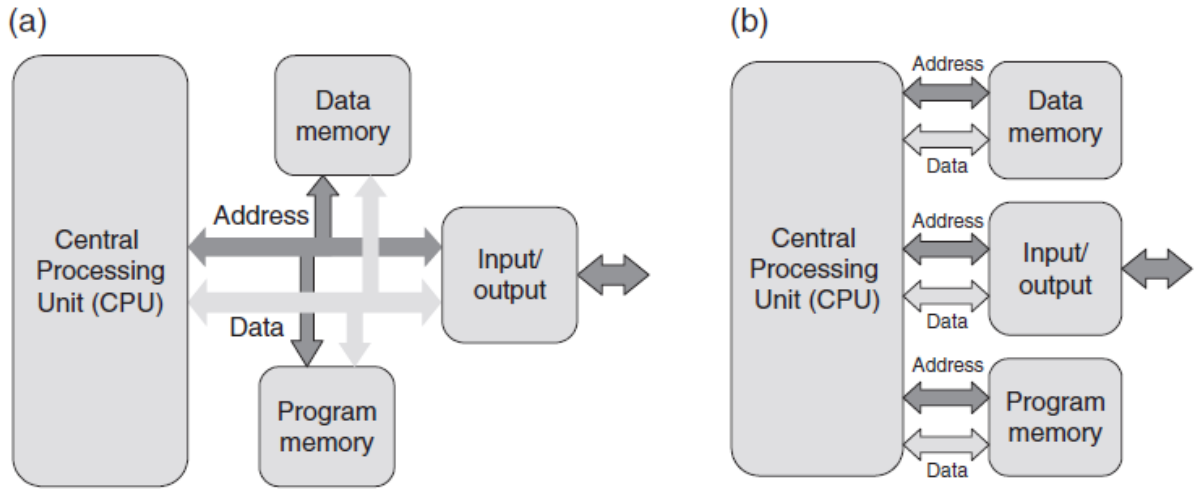
Σχ. 2.3 Σχηματική παράσταση του βρόχου ελέγχου που εκτελεί ένας μικροελεγκτής

βίντεο. Οι DSP επεξεργαστές ανήκουν στην κατηγορία των συστημάτων επεξεργασίας σήματος και πολυμέσων, που εξετάσαμε στην παράγραφο 1.2.4

2.2 Αρχιτεκτονική Harvard

Βασικό χαρακτηριστικό της αρχιτεκτονικής πολλών μικροελεγκτών, που δεν απαντάται στους συνηθισμένους μικροεπεξεργαστές, είναι ότι έχουν διαφορετικό **διάδρομο για τις εντολές** (*instructions bus*) και διαφορετικό **διάδρομο δεδομένων** (*data bus*), για όλα τα υπόλοιπα δεδομένα και αποτελέσματα της επεξεργασίας. Η αρχιτεκτονική αυτή αναφέρεται και ως αρχιτεκτονική Harvard. Βλέπε σχετικά και το Σχήμα 2.4.

Ας σημειωθεί εδώ ότι οι συνηθισμένοι μικροϋπολογιστές είναι δομημένοι σύμφωνα με τη λεγόμενη *αρχιτεκτονική von Neumann*. Σε αυτούς, το πρώτο βήμα για την εκτέλεση μιας λογικής ακολουθίας είναι να καταχωρηθούν στη μνήμη RAM οι εντολές του προγράμματος, μέσω μίας περιφερειακής μονάδας (πληκτρολόγιο ή μαγνητική μνήμη). Το πρόγραμμα, δηλαδή, καταλαμβάνει ένα μέρος στην ίδια μνήμη που διατίθεται επίσης για τα δεδομένα και τα αποτελέσματα της επεξεργασίας. Όταν διακοπεί η τροφοδοσία, τα δεδομένα της μνήμης RAM χάνονται, μαζί με το πρόγραμμα. Την επόμενη φορά ο μικροπεξεργαστής μπορεί να φορτώσει στη μνήμη RAM και να επεξεργαστεί ένα διαφορετικό πρόγραμμα. Οι διάφορες μονάδες του υπολογιστικού συστήματος επικοινωνούν παράλληλα μέσω των ίδιων διαδρόμων δεδομένων, διευθύνσεων και ελέγχου.

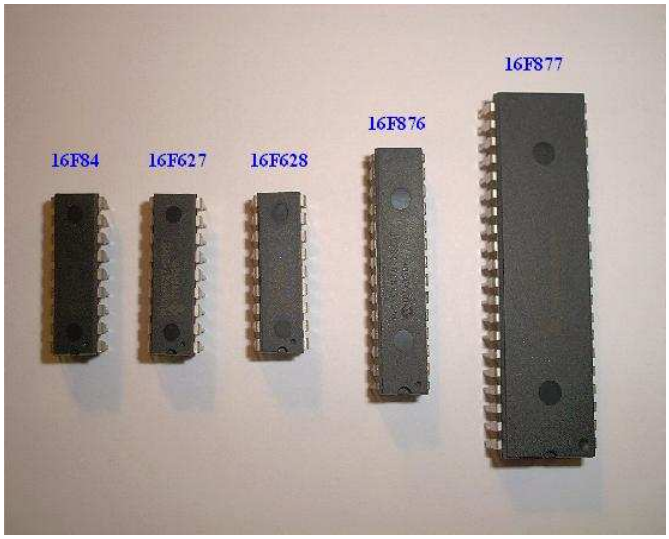


Σχήμα 2.4: Παρουσίαση των δυο τύπων αρχιτεκτονικής α) Ενιαία μνήμη προγράμματος και δεδομένων (αρχιτεκτονική Von-Neumann) β) Ξεχωριστή μνήμη προγράμματος και δεδομένων (αρχιτεκτονική Harvard).

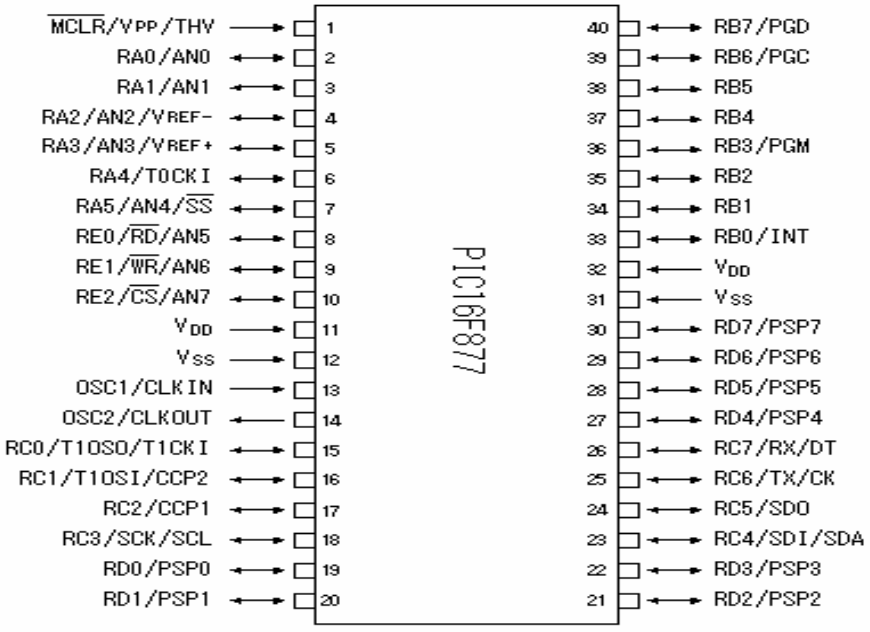
2.3 Οι μικροελεγκτές PIC – Ο PIC 16F877

Στα επόμενα θα αναφερθούμε ειδικά στους μικροελεγκτές PIC της εταιρείας Microchip. Πρόκειται για μία οικογένεια κυκλωμάτων που καλύπτουν ένα εύρος δυνατοτήτων και κόστους. Ειδικότερα, θα μελετήσουμε τους μικροελεγκτές PIC16F87x, που ανήκουν στη μεσαία κατηγορία της οικογένειας. Ένα χαρακτηριστικό που κάνει έναν μικροελεγκτή, όπως τον PIC16F877, ιδιαίτερα ελκυστικό και κατάλληλο για απλές εφαρμογές είναι η ιδιότητα της μνήμης του προγράμματος να διαγράφεται και να επαναπρογραμματίζεται μέσω ηλεκτρικών σημάτων. Με άλλα λόγια, η μνήμη προγράμματος είναι *ηλεκτρικά διαγραφόμενη και προγραμματιζόμενη* (*electrically erasable – programmable*) και γι' αυτό ονομάζεται ***Flash EEPROM*** (η λέξη *flash* δηλώνει ότι η μνήμη λειτουργεί με υψηλή ταχύτητα κατά τη διαγραφή και τον προγραμματισμό). Δεν απαιτείται, δηλαδή, κάποιο ακριβό μηχάνημα διαγραφής των περιεχομένων της μνήμης με τη βοήθεια υπεριώδους ακτινοβολίας, όπως συμβαίνει με άλλες μνήμες. Έτσι, το ίδιο ολοκληρωμένο κύκλωμα μπορεί να χρησιμοποιηθεί για την ανάπτυξη του πρωτότυπου κυκλώματος και του πιλοτικού προγράμματος, καθώς και για την τελική υλοποίηση του κυκλώματος που θα τεθεί σε χρήση ή και θα παραχθεί μαζικά. Η διαγραφή και ο προγραμματισμός του ολοκληρωμένου μπορούν να γίνουν χιλιάδες φορές.

Επιπλέον, ο μικροελεγκτής PIC16F877 έχει τη δυνατότητα να προγραμματίζεται μέσα από εξαιρετικά απλούς **προγραμματιστές**, δηλαδή κυκλώματα που από τη μια μεριά συνδέονται στη σειριακή ή στη θύρα USB ενός προσωπικού υπολογιστή και από την άλλη συνδέονται σε ορισμένους από τους ακροδέκτες (pins) του μικροελεγκτή,



(α)

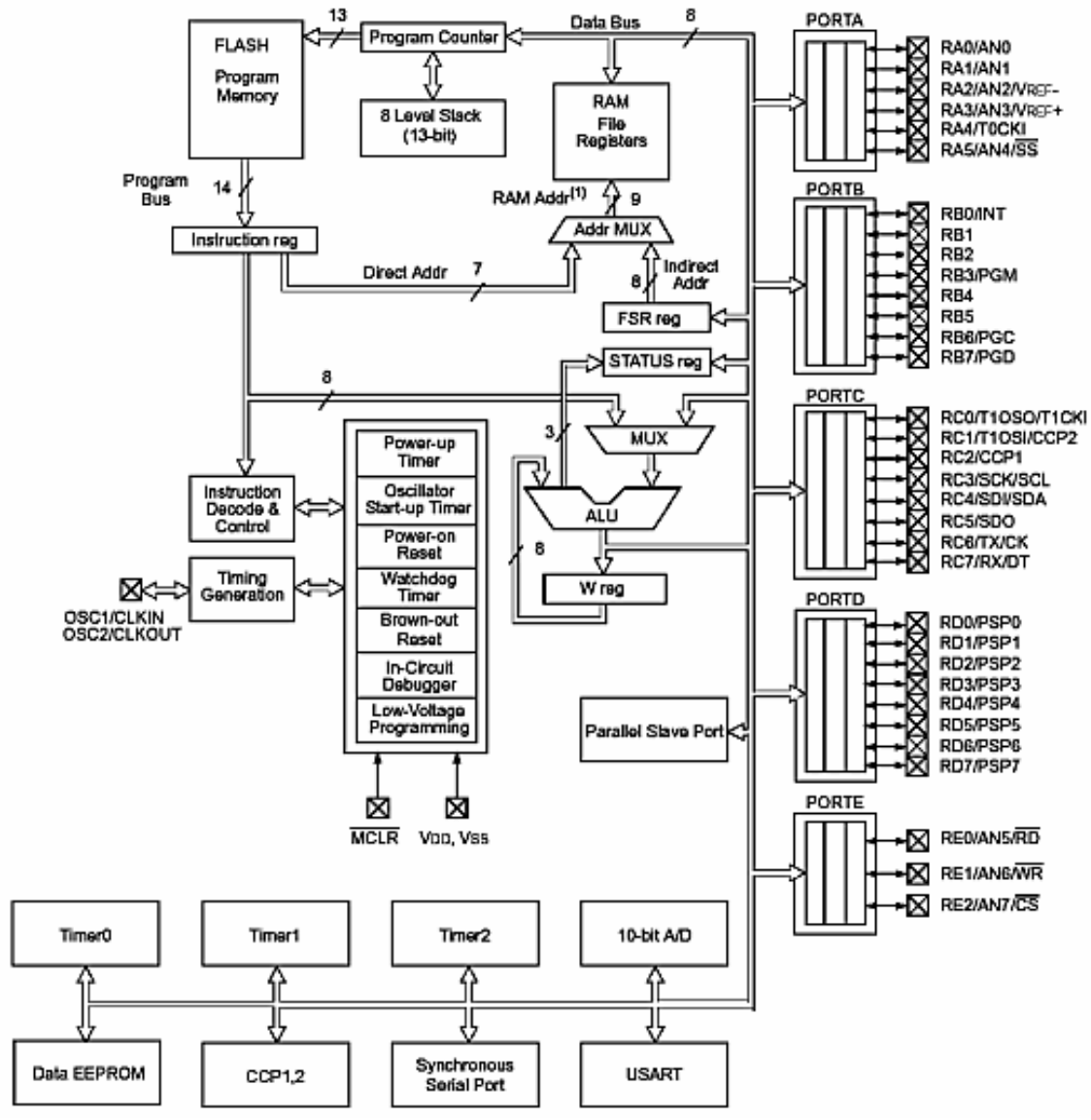


(β)

Σχήμα 2.5 α) **Η** οικογένεια τον μικροελεγκτών PIC 16fXXX και β) Διάγραμμα ακροδεκτών του μικροελεγκτή PIC16F877

προκειμένου να προγραμματίσουν τη μνήμη EEPROM. Η εταιρία Microchip διαθέτει λογισμικό για την ανάπτυξη εφαρμογών καθώς και όλες τις απαραίτητες πληροφορίες για τους μικροελεγκτές PIC, στην ιστοσελίδα της στο Διαδίκτυο, στη διεύθυνση http://www.microchip.com.

Στο Σχήμα 2.5 (β) παρουσιάζεται η αντιστοιχία των 40 συνολικά ακροδεκτών του μικροελεγκτή. Το κύκλωμα διατίθεται σε πλαστική συσκευασία DIP (*Dual In-line Package – διάταξης ακροδεκτών σε δύο σειρές*), είναι τεχνολογίας CMOS και έχει χαμηλό κόστος.



Σχήμα 2.6 Γενικό διάγραμμα της αρχιτεκτονικής του μικροελεγκτή PIC16F877

2.4 Αρχιτεκτονική του Μικροελεγκτή PIC16F877

Όπως όλοι οι μικροελεγκτές PIC, το μοντέλο 16F877 ενσωματώνει την *αρχιτεκτονική Harvard*, της οποίας όπως είδαμε, βασικό χαρακτηριστικό είναι ο διαφορετικός διάδρομος εντολών και δεδομένων και η συνακόλουθη διαφορά των μνημών προγράμματος και καταχωρητών/μεταβλητών. Η αρχιτεκτονική των μικροελεγκτών PIC υπακούσει στους κανόνες της λεγόμενης αρχιτεκτονικής RISC. Ο διάδρομος εντολών έχει εύρος 14 bits. Κάθε εντολή αποτελείται από μία λέξη 14 bits και συνεπώς εκτελείται σε έναν και μοναδικό κύκλο εκτέλεσης, κάτι που αποτελεί βασικό χαρακτηριστικό της αρχιτεκτονικής RISC.

Ο διάδρομος δεδομένων έχει εύρος 8 bits. Ο μικροελεγκτής μπορεί να προγραμματιστεί με 35 συνολικά εντολές και με τη βοήθεια δεκαεπτά καταχωρητών ειδικού σκοπού, που αντιστοιχούν σε συγκεκριμένες θέσεις της μνήμης RAM. Κάθε εντολή εκτελείται σε τέσσερις συνολικά κύκλους του εξωτερικού ωρολογιακού σήματος, που συνιστούν έναν **κύκλο εκτέλεσης** ή **κύκλο εντολής** (*αποκωδικοποίηση εντολής*, *ανάγνωση τελεστέου καταχωρητή*, *επεξεργασία δεδομένων* και *εγγραφή αποτελεσμάτων στον τελικό προορισμό*). Έτσι, εάν ο εξωτερικός ταλαντωτής είναι ένας κρύσταλλος με ιδιοσυχνότητα 4MHz, τότε κάθε εντολή θα εκτελείται σε χρόνο 1 μs (= $^{1}/_{1MHz}$).

Ο μικροελεγκτής PIC16F877 διαθέτει 368 bytes μνήμης RAM, όπου ο χρήστης μπορεί να ορίσει μεταβλητές και να αποθηκεύσει δεδομένα. Επίσης έχει τριαντατρείς ακροδέκτες εισόδου/εξόδου, μοιρασμένους σε πέντε θύρες, που ονομάζονται **PORTA** (6 ακροδέκτες) και **PORTB** (8 ακροδέκτες), **PORTC** (8 ακροδέκτες), **PORTD** (8 ακροδέκτες) και **PORTE** (3 ακροδέκτες), αντίστοιχα. Διαθέτει τρεις χρονιστές, που λειτουργούν και ως απαριθμητές (**Timer0, Timer1, Timer2**) κι έναν ακόμη, εσωτερικά ταλαντούμενο χρονιστή, που επιτηρεί και επαναφέρει τον μικροελεγκτή, σε περίπτωση που βρεθεί εκτός ελέγχου. Ο τελευταίος χρονιστής ονομάζεται **WDT** ή *Watchdog Timer*. Παρακάτω θα αναφερθούμε με κάποια λεπτομέρεια στον χρονιστή **Timer0** (8-bits).

Η μνήμη προγράμματος (EEPROM) του μικροελεγκτή 16F877 έχει χωρητικότητα 2K ή 2048 bytes. Εκτός από τη μνήμη αυτή, ο 16F877 διαθέτει άλλα 64 bytes μνήμης EEPROM, τα οποία προορίζονται για τη μόνιμη αποθήκευση δεδομένων.

Τα βασικά αυτά χαρακτηριστικά της αρχιτεκτονικής του μικροελεγκτή PIC16F877 φαίνονται στο διάγραμμα βαθμίδων του Σχήματος 2.6. Το σχεδιάγραμμα αυτό αναφέρεται σε όλη την οικογένεια των μικροελεγκτών PIC16F87x, ενώ δηλώνονται και οι μεταξύ τους διαφορές.

Ας δούμε μερικά αρχιτεκτονικά στοιχεία με μεγαλύτερη λεπτομέρεια.

2.5 Οργάνωση της Μνήμης RAM

Ένα βασικό θέμα που πρέπει να καταλάβει κανείς, προκειμένου να εργαστεί με κάποιον μικροελεγκτή είναι η δομή της μνήμης RAM. Στον μικροελεγκτή PIC16F877 η μνήμη RAM είναι ένας πίνακας τεσσάρων σελίδων, 128 θέσεων η κάθε μια. Η πρώτη

σελίδα ξεκινά από τη διεύθυνση 0x000 και φτάνει στη διεύθυνση 0x07F (0 έως 127 στο δεκαδικό). Η πρόσβαση στις θέσεις αυτές και η αλλαγή του περιεχομένου τους γίνεται αποκλειστικά μέσω εντολών. Δηλαδή, η μνήμη RAM δεν «φορτώνεται» από περιφερειακές διατάξεις, όπως συμβαίνει στους «κανονικούς» υπολογιστές.

Η εταιρία Microchip ονομάζει τις θέσεις μνήμης «*αρχεία καταχωρητών*» (*files* ή *registers*) και όταν αναφέρεται σ' αυτές, στις διάφορες εντολές της συμβολικής γλώσσας, χρησιμοποιεί το γράμμα **f**. Κάθε θέση μνήμης αποτελείται από 8 bits.

Αφού η μνήμη δεδομένων του PIC16F877 αποτελείται από τέσσερα τμήματα (banks 0-3), με μέγεθος 128 Bytes το καθένα, η συνολική της χωρητικότητα είναι 512 Bytes. Το κάθε τμήμα (σελίδα) μνήμης αποτελείται τόσο από καταχωρητές γενικού όσο και ειδικού σκοπού.

Οι πρώτες θέσεις μνήμης αντιστοιχούν σε εσωτερικούς καταχωρητές, που ρυθμίζουν εσωτερικές λειτουργίες του μικροελεγκτή (Καταχωρητές ειδικού σκοπού - *Special Function Registers*). Μερικοί από τους καταχωρητές ειδικού σκοπού χρησιμοποιούνται για τον έλεγχο του πυρήνα του PIC ενώ άλλοι για τον έλεγχο των περιφερειακών του. Η τιμή τους μπορεί να αλλάξει μέσω εντολών, αλλά ο καθένας καταλαμβάνει συγκεκριμένη θέση στον πίνακα μνήμης RAM. Στις πρώτες αυτές θέσεις δεν μπορούμε να αποθηκεύσουμε δικές μας μεταβλητές.

Οι επόμενες θέσεις (368 συνολικά στον PIC16F877), αντιστοιχούν στην περιοχή *Καταχωρητών Γενικού Σκοπού* (*General Purpose Registers* ή *GPR*). Αυτή είναι η περιοχή όπου μπορούμε να δηλώσουμε και να αποθηκεύσουμε τις δικές μας μεταβλητές.

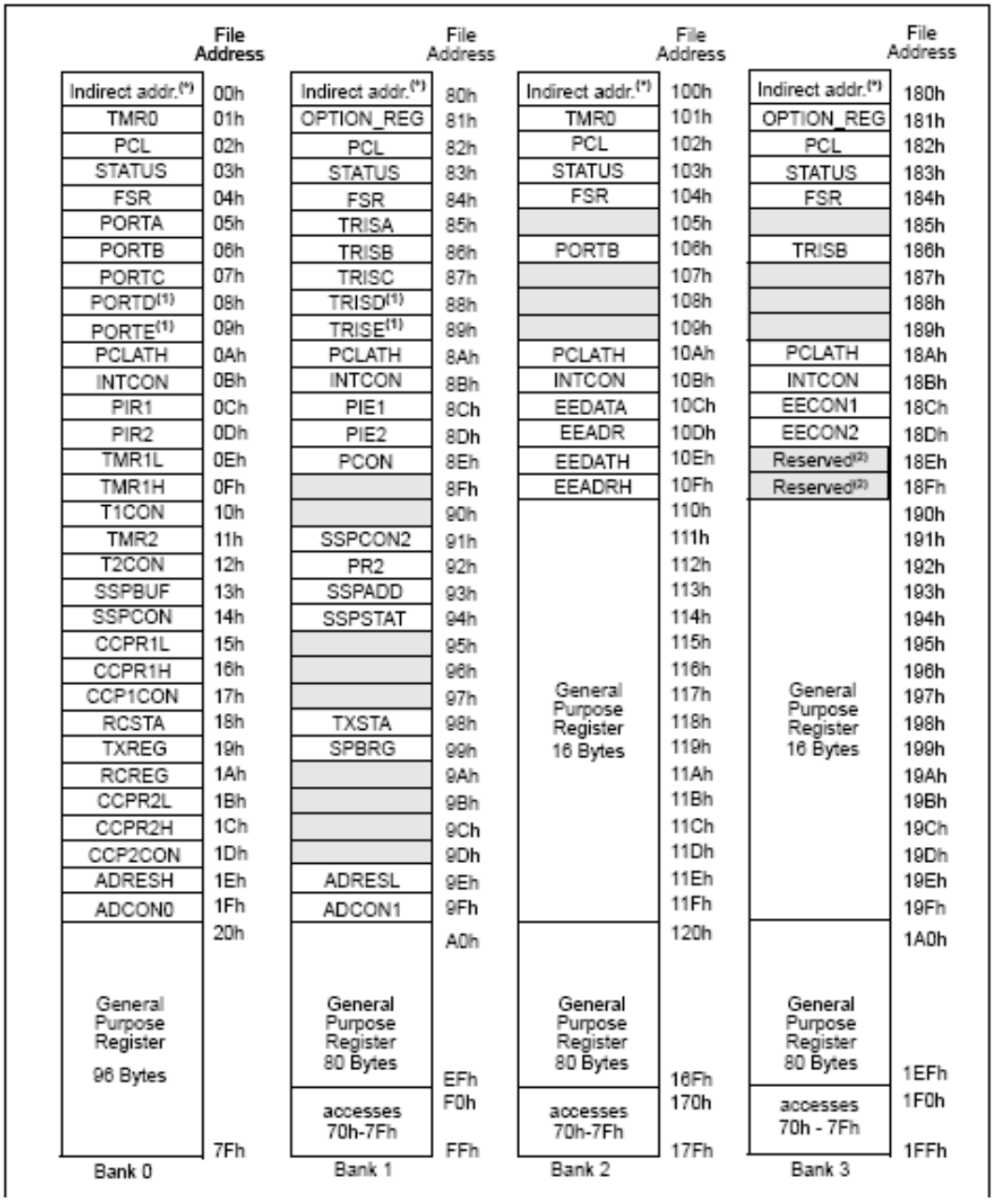
Η διάταξη των θέσεων μνήμης παρουσιάζεται στο Σχήμα 2.7. Το σχήμα αυτό ονομάζεται και *χάρτης της μνήμης*. Στο σχήμα αυτό φαίνεται ένα χαρακτηριστικό της μνήμης που χρειάζεται κάποια προσοχή. Όπως σημειώσαμε, η μνήμη αποτελείται από τέσσερις σελίδες, με έναν αριθμό θέσεων η κάθε μία, οι οποίες είναι χρήσιμες για τον χρήστη. Ορισμένοι από τους καταχωρητές ειδικού σκοπού, όπως ο STATUS και ο FSR χαρτογραφούνται σε περισσότερες από μία σελίδες, και συνεπώς μπορούμε να απευθυνθούμε σ' αυτούς είτε βρισκόμαστε στη μία είτε στην άλλη σελίδα μνήμης. Όμως, για τους περισσότερους από τους καταχωρητές ειδικού σκοπού, που βρίσκονται στις πρώτες διευθύνσεις, έχει σημασία σε ποια σελίδα μνήμης βρισκόμαστε. Έτσι, οι καταχωρητές **OPTION**, **TRISA**, **TRISB**, **TRISC, TRISD, TRISE,** χαρτογραφούνται ***μόνον*** σε ορισμένες σελίδες μνήμης και συνεπώς μπορούμε να τους προσπελάσουμε μόνον μέσα από τις αντίστοιχες διευθύνσεις. Στις αντίστοιχες θέσεις στις άλλες σελίδες υπάρχουν άλλοι ειδικοί καταχωρητές.

Κατά την εκκίνηση, ο μικροελεγκτής βλέπει εξ' ορισμού τη μηδενική σελίδα μνήμης (bank0). Εάν χρειαστεί να προσπελάσουμε καταχωρητές που βρίσκονται στη σελίδα μνήμης 1 (bank1), θα πρέπει να το ορίσουμε αυτό με την εντολή:

BSF STATUS, RP0

η οποία θέτει σε λογικό ένα το bit RP0 του καταχωρητή κατάστασης (*STATUS Register*). Αυτό είναι το bit 5 του καταχωρητή **STATUS**. Το λεπτομερές νόημα της εντολής **BSF f,b**

θα το παρουσιάσουμε σε επόμενη παράγραφο. Μετά την προσπέλαση της σελίδας μνήμης 1 (bank1) οφείλουμε να επιστρέψουμε ξανά στην αρχική σελίδα, όπου βρίσκεται και η πλειονότητα των θέσεων μνήμης που μας χρειάζονται. Ας σημειωθεί εδώ, ότι ο διάδρομος διευθύνσεων (*address bus*) της μνήμης RAM έχει εύρος 7 bits (βλέπε και Σχήμα 2.4). Από τα 14 bits του κωδικού των εντολών, τα επτά προορίζονται για τη δήλωση της διεύθυνσης των καταχωρητών. Άρα, συνολικά μπορούμε να προσπελάσουμε 128 καταχωρητές (2^7), μέσω του διαδρόμου διευθύνσεων της μνήμης



Σχήμα 2.7 Χάρτης της μνήμης RAM του μικροελεγκτή PIC16F877

RAM. Αυτός είναι ο λόγος που, για να προσπελάσουμε τη δεύτερη σελίδα μνήμης RAM (bank1), δηλαδή διευθύνσεις καταχωρητών μεγαλύτερες του 128 (δεκαεξαδικό 0x07F), χρειάζεται να θέσουμε σε λογικό ένα το bit RP0 του καταχωρητή **STATUS**. Δηλαδή, αυτό λειτουργεί ως το περισσότερο σημαντικό bit μιας διεύθυνσης εύρους 8 bits.

2.6 Οι εντολές των μικροελεγκτών PIC.

Όπως αναφέρθηκε, οι μικροελεγκτές PIC διαθέτουν ένα περιορισμένο σύνολο από 35 εντολές, που η κάθε μια αποτελείται από 14 bits. Επειδή και ο διάδρομος εντολών έχει μήκος 14 bits κάθε εντολή μπορεί να εκτελεστεί σε έναν μοναδικό κύκλο εκτέλεσης.
Ο πίνακας 2.1 παραθέτει τις 35 εντολές που αποτελούν τη γλώσσα Assembly των μικροελεγκτών PIC.

Οι εντολές των ελεγκτών PIC μπορούν να χωριστούν σε τέσσερις κατηγορίες.

1. Αριθμητικές εντολές
2. Ελέγχου εκτέλεσης
3. Ελέγχου του μικροεπεξεργαστή
4. Χειρισμού των bit των καταχωρητών

Η πρώτη κατηγορία αποτελείται από τις «αριθμητικές» εντολές, που περιλαμβάνουν τη πράξη της πρόσθεσης και της αφαίρεσης ανάμεσα στα περιεχόμενα καταχωρητών, καθώς και πράξεις αύξησης ή μείωσης των τιμών τους και πράξεις σε επίπεδο bit.

Στην επόμενη κατηγορία εντολών, ανήκουν οι εντολές «ελέγχου εκτέλεσης». Αυτές, τις αποτελούν οι εντολές άλματος (goto), κλήσης υπορουτίνας (call) και οι εντολές επιστροφής (return) από κάποια υπορουτίνα, καθώς επίσης και οι εντολές διακλάδωσης υπό συνθήκη (btfss, btfsc, decfsz κλπ).

Στην επόμενη κατηγορία ανήκουν οι εντολές «ελέγχου του μικροεπεξεργαστή». Οι εντολές αυτές επηρεάζουν βασικά τη λειτουργία του επεξεργαστή και τα κυκλώματα που συσχετίζονται με αυτόν (π.χ. clrwdt, sleep, clrf).

Η τελευταία κατηγορία αποτελείται από τις εντολές «χειρισμού των bit των καταχωρητών» (τοποθέτηση ή μηδενισμός bit). Με τις εντολές αυτές ελέγχουμε άμεσα, κάθε ένα ξεχωριστό bit των καταχωρητών. Η πιο προφανής χρήση των εντολών αυτών είναι ο άμεσος έλεγχος επιμέρους κυκλωμάτων και ακροδεκτών του μικροελεγκτή.

2.7 Τρόποι προσπέλασης της μνήμης

Στο σημείο αυτό αναφέρουμε ότι τις θέσεις της μνήμης RAM μπορούμε να τις προσπελάσουμε με *απευθείας* (*direct*) και *έμμεσο* (*indirect*) τρόπο (ή αλλιώς *διευθυνσιοδότηση*), όπως και στους συνηθισμένους μικροεπεξεργαστές. Στην απευθείας διευθυνσιοδότηση, η διεύθυνση της μνήμης του καταχωρητή που θέλουμε να προσπελάσουμε αναφέρεται με την αριθμητική ή τη συμβολική της ονομασία στη

Πίνακας 2.1: Οι εντολές του μικροελεγκτή PIC

Μνημονικό / Τελεστής		Λειτουργία	Σημαία – Flag που επηρεάζεται
Εντολές χειρισμού ψηφιολέξεων - Byte oriented file register operations			
ADDWF	f, d	Πρόσθεσε το W και το f	C, DC, Z
ANDWF	f, d	Κάνε την λογική πράξη AND ανάμεσα στο W και το f	Z
CLRF	f	Μηδένισε το f	Z
CLRW	-	Μηδένισε το W	Z
COMF	f, d	Φτιάξε το συμπλήρωμα του f και αποθήκευσέ το στο d	Z
DECF	f, d	Μείωσε την τιμή του f	Z
DECFSZ	f, d	Μείωσε την τιμή του f, παρέκαμψε την επόμενη εντολή αν ο f γίνει 0	
INCF	f, d	Αύξησε την τιμή του f	Z
INCFSZ	f, d	Αύξησε την τιμή του f, παρέκαμψε την επόμενη εντολή αν ο f γίνει 0	
IORWF	f, d	Κάνε την λογική πράξη IOR ανάμεσα στο W και το f	Z
MOVF	f, d	Μετέφερε το περιεχόμενο του f	Z
MOVWF	f	Μετέφερε το περιεχόμενο του W στο f	
NOP	-	Εντολή δίχως λειτουργία (απλή χρονική καθυστέρηση ενός κύκλου μηχανής)	
RLF	f, d	Μετέφερε προς τα αριστερά το περιεχόμενο του f μέσο του ψηφίου Carry	C
RRF	f, d	Μετέφερε προς τα δεξιά το περιεχόμενο του f μέσο του ψηφίου Carry	C
SUBWF	f, d	Αφαίρεσε το περιεχόμενο του W από τον καταχωρητή f	C, DC, Z

SWAPF	f, d	Αντιμετάθεσε τα δύο μισά της ψηφιολέξης (Byte) στο f	
Εντολές χειρισμού ψηφίων - Bit oriented file register operations			
BCF	f, b	Μηδένισε το ψηφίο b του καταχωρητή f	
BSF	f, b	Κάνε λογικό 1 το ψηφίο b του καταχωρητή f	
BTFSC	f, b	Εξέτασε το ψηφίο b του καταχωρητή f, παρέκαμψε την επόμενη εντολή αν είναι 0	
BTFSS	f, b	Εξέτασε το ψηφίο b του καταχωρητή f, παρέκαμψε την επόμενη εντολή αν είναι 1	
Εντολές πράξεων με σταθερούς αριθμούς. Εντολές ελέγχου προγράμματος			
ADDLW	k	Πρόσθεσε τον σταθερό αριθμό k με το W	C, DC, Z
ANDLW	k	Κάνε την λογική πράξη AND ανάμεσα στο k και το W	Z
CALL	k	Κάλεσε την υπορουτίνα k	
CLRWDT	-	Μηδένισε τον επιτηρητή Watchdog Timer	$\overline{TO}$, $\overline{PD}$
GOTO	k	Πήγαινε και εκτέλεσε την εντολή που υπάρχει στην διεύθυνση k	
IORLW	k	Κάνε την λογική πράξη IOR ανάμεσα στο k και το W	Z
MOVLW	k	Μετέφερε το περιεχόμενο του k στο W	
RETFIE	-	Επέστρεψε στην διεύθυνση που ήσουν πριν συμβεί η διακοπή (interrupt)	
RETLW	k	Επέστρεψε από υπορουτίνα και φόρτωσε τον σταθερό αριθμό k στο W	
RETURN	-	Επέστρεψε από υπορουτίνα	
SLEEP	-	Ενεργοποίησε την λειτουργία χαμηλής κατανάλωσης (Sleep - κατανάλωση 2μA)	$\overline{TO}$, $\overline{PD}$
SUBLW	k	Αφαίρεσε το περιεχόμενο του W από το σταθερό αριθμό k	C, DC, Z
XORLW	k	Κάνε την λογική πράξη XOR ανάμεσα στο k και το W	Z

θέση του *τελεστέου* (*operand*). Η σύνταξη της εντολής είναι **movwf *F***, όπου *F* η θέση μνήμης όπου θα καταχωρηθεί το περιεχόμενο του *w*. Στην έμμεση διευθυνσιοδότηση, ο τελεστέος δίνει μια διεύθυνση μνήμης, στην οποία ευρίσκεται αποθηκευμένη η πραγματική διεύθυνση που θέλουμε να προσπελάσουμε.

2.8 Καταχωρητές Ειδικού Σκοπού

Δεν είναι δυνατό να επεκταθούμε σε λεπτομερή περιγραφή όλων των καταχωρητών ειδικού σκοπού (SFR) στο πλαίσιο αυτής της περιληπτικής αναφοράς. Εντελώς εισαγωγικά θα κάνουμε μια νύξη για το νόημα και τη χρήση των πιο απαραίτητων. Αναγκαστικά, ο χρήστης θα πρέπει να ανατρέξει στα εξειδικευμένα εγχειρίδια της Microchip για επιπλέον λεπτομέρειες. Τα 14 bits των εντολών του 16F877, αν και επιτρέπουν την εγγραφή κάθε εντολής σε μια μοναδική *λέξη*, που εκτελείται συνήθως σε έναν κύκλο, οδηγούν σε κάποιους περιορισμούς. Έτσι, δεν μπορούμε να ορίσουμε παραπάνω από έναν καταχωρητή ως τελεστέο σε κάθε εντολή. Παρακάτω θα γνωρίσουμε δύο βασικούς καταχωρητές, που παίζουν καίριο ρόλο σε κάθε εφαρμογή.

2.8.1 Ο καταχωρητής w

Ο καταχωρητής μέσω του οποίου πραγματοποιούνται οι περισσότερες πράξεις και λειτουργίες είναι ο λεγόμενος *καταχωρητής εργασίας* (*working register*), που αναφέρεται ως *καταχωρητής **w***. Αυτός είναι το αντίστοιχο του *συσσωρευτή* (*accumulator*), που χρησιμοποιείται σε δημοφιλείς μικροεπεξεργαστές, όπως τους 6502, 8085 και 8051.

Ο καταχωρητής **w** έχει εύρος 8 bits και δεν διευθυνσιοδοτείται. Μπορούμε να φανταζόμαστε ότι ενσωματώνεται στην αριθμητική μονάδα. Περιλαμβάνεται στις περισσότερες εντολές των μικροελεγκτών PIC, διότι μέσω αυτού γίνονται οι μετακινήσεις δεδομένων και οι πράξεις ανάμεσα στους καταχωρητές. Στον καταχωρητή **w** μπορεί να γίνει προσπέλαση άμεσα και να φορτωθεί κάποιο αριθμητικό δεδομένο, με την εντολή **movlw k**, όπου k ο αριθμός (literal ή κυριολέκτημα). Κάθε αριθμητική πράξη που επιτελείται στον PIC, χρησιμοποιεί τον καταχωρητή **w**. Παραδείγματος χάριν, αν θέλουμε να προσθέσουμε τα περιεχόμενα δύο καταχωρητών, πρέπει να μεταφέρουμε το περιεχόμενο του πρώτου καταχωρητή στον **w** και στη συνέχεια να το προσθέσουμε με το περιεχόμενο του δεύτερου καταχωρητή.

Οι ελεγκτές PIC διαθέτουν αρκετά ισχυρή αρχιτεκτονική από την άποψη ότι το αποτέλεσμα μιας αριθμητικής πράξης μπορεί να αποθηκευτεί ή στον καταχωρητή "**w**", ή στον καταχωρητή προέλευσης των δεδομένων. Αποθηκεύοντας το αποτέλεσμα στον καταχωρητή προέλευσης εξαλείφεται ουσιαστικά η ανάγκη χρήσης πρόσθετων εντολών για την αποθήκευση αυτή. Με το τρόπο αυτό, η μεταφορά των αποτελεσμάτων απλουστεύεται και γίνεται αποδοτικότερη.

2.8.2 Ο καταχωρητής STATUS

Ο καταχωρητής **STATUS** (Καταχωρητής Κατάστασης) αποτελεί το βασικό καταχωρητή που χρησιμοποιείται για τον έλεγχο της εκτέλεσης του προγράμματος. Ο καταχωρητής αυτός χωρίζεται σε τρία τμήματα. Το πρώτο τμήμα περιέχει τις σημαίες (Flags) ή bits κατάστασης της εκτέλεσης (τις "**Z**", "**dc**" και "**C**"). Τα τρία αυτά bits απεικονίζουν τη κατάσταση της εκτέλεσης του προγράμματος. Το bit "**Z**", ή η σημαία του μηδενός (Zero Flag), τίθεται σε λογικό "1" όταν το αποτέλεσμα κάποιας πράξης γίνει ίσο με το μηδέν (add, sub, clear, πράξεις λογικής επεξεργασίας). Η σημαία κρατουμένου (Carry Flag) "**C**" τίθεται σε λογικό "1" όταν το αποτέλεσμα κάποιας πράξης γίνει μεγαλύτερο από 255 (0x0FF), και χρησιμοποιείται για να δηλώσει ότι πρέπει να ενημερωθούν και τα υψηλότερης τάξης bytes που είναι σχετικά με το αποτέλεσμα.

Η σημαία δεκαδικού κρατουμένου (Decimal Carry Flag) "**dc**", τίθεται σε λογικό "1" όταν τα τέσσερα λιγότερο σημαντικά bits (nibble) του αποτελέσματος μιας αριθμητικής πράξης, δώσουν αριθμό μεγαλύτερο από 15.

Αυτά τα bits, που αντιπροσωπεύουν οι σημαίες κατάστασης, μπορούν να διαβαστούν και να εγγραφούν από το χρήστη, ενώ επίσης ενημερώνεται η κατάστασή τους, κάθε φορά που εκτελείται μια εντολή.

Τα επόμενα δύο bits του καταχωρητή **STATUS** δείχνουν το τρόπο με τον οποίο ο επεξεργαστής ανταποκρίθηκε κατά την εκτέλεση της διαδικασίας έναρξης του προγράμματος ή κατά την επιστροφή του από έναν ανενεργό κύκλο (sleep). Ο σκοπός για τον οποίο χρησιμοποιούνται τα bits αυτά είναι να μπορεί το πρόγραμμα της εφαρμογής να αντιλαμβάνεται την αιτία για την οποία ο έλεγχος του επεξεργαστή βρέθηκε στην αρχική θέση εκτέλεσης του προγράμματος.

Τα άλλα δύο bits, «*RP0»* και «*RP1»* χρησιμοποιούνται αποκλειστικά για τη προσπέλαση των δύο υψηλότερων σελίδων της μνήμης. Είναι δυνατή τόσο η ανάγνωση όσο και η εγγραφή σε αυτά. Το bit *"irp",* χρησιμοποιείται για την επιλογή έμμεσης διευθυνσιοδότησης.

Πίνακας 2.2: Ο καταχωρητής STATUS και τα bits που τον αποτελούν

7	6	5	4	3	2	1	0
IRP	RP1	RP0	-	-	Z	DC	C

3. Γενικά θέματα προγραμματισμού των μικροελεγκτών PIC16F

3.1 Βασικές εντολές αντιγραφής και μεταφοράς δεδομένων

Η εντολή **MOVWF** Reg

αντιγράφει το περιεχόμενο του καταχωρητή εργασίας **w** στον καταχωρητή **Reg**.

Η εντολή **MOVF** Reg, w

αντιγράφει το περιεχόμενο του καταχωρητή **Reg** στον καταχωρητή εργασίας **w**.

Η εντολή **MOVLW** k

μεταφέρει την ποσότητα **k** στον καταχωρητή εργασίας **w**, με τη διαφορά, ότι εδώ το **k** είναι ο συγκεκριμένος αριθμός και δεν αντιπροσωπεύει μία θέση μνήμης, όπως προηγουμένως ο **Reg**. Ο αριθμός k αναφέρεται ως literal ή κυριολέκτημα.

Με βάση τα παραπάνω, η εντολή

MOVLW 0C

μεταφέρει τον δεκαεξαδικό αριθμό 0C στον καταχωρητή εργασίας **w**, ενώ η εντολή

MOVLW Reg

θα μεταφέρει στον **w** όχι το περιεχόμενο της διεύθυνσης **Reg**, αλλά την ίδια τη διεύθυνση **Reg**. Ας ονομάσουμε, λοιπόν, τον καταχωρητή στη διεύθυνση hex0F με το ψευδώνυμο **START**:

START **equ** 0F

Τότε η εντολή

MOVLW START

έχει σαν αποτέλεσμα να μεταφερθεί στον καταχωρητή εργασίας **w** ο αριθμός 0F, ενώ η εντολή

MOVF START, w

Αντιγράφει στον **w** το περιεχόμενο της διεύθυνσης 0F.

3.2 Έλεγχος της ροής του προγράμματος

Η ροή του προγράμματος στους μικροελεγκτές PIC ελέγχεται μέσω εντολών διακλάδωσης χωρίς συνθήκη και μέσω εντολών διακλάδωσης υπό συνθήκη. Επίσης, ελέγχεται με την κλήση υπορουτινών. Τις περιπτώσεις αυτές εξετάζουμε στις επόμενες παραγράφους.

3.2.1 Εντολή goto

Η εντολή goto διακόπτει την ομαλή ροή του προγράμματος και φορτώνει μια νέα τιμή στον απαριθμητή προγράμματος (Program counter ή PC). Η σύνταξη της εντολής είναι:

goto *label* ή **goto** *address*

παράδειγμα:

goto *main* ; πήγαινε στην εντολή με ετικέτα *main*

ή

goto 08 ; πήγαινε στη διεύθυνση προγράμματος 08

3.2.2 Εντολές διακλάδωσης υπό συνθήκη

Οι εντολές αυτές είναι εντολές ελέγχου bit. Δηλαδή, ελέγχουν τη λογική κατάσταση ενός bit κάποιου καταχωρητή και διακλαδίζουν τη ροή του προγράμματος, ανάλογα με το αποτέλεσμα του ελέγχου. Αν η συνθήκη που ελέγχουμε συντρέχει, τότε δεν εκτελείται η αμέσως επόμενη εντολή, αλλά η μεθεπόμενη (“skip”). Αν η συνθήκη δεν συντρέχει, τότε εκτελείται κανονικά η επόμενη εντολή. Τέτοιες εντολές είναι οι εξής:

btfss Reg, b ;κάνε έλεγχο του bit b του καταχωρητή Reg και να παρακάμψεις την

;επόμενη εντολή εάν το bit είναι λογικό 1.

Η εντολή διαβάζεται ως εξής: Bit Test f Skip on Set

Παράδειγμα:

btfss STATUS, Z

Ο έλεγχος του προγράμματος θα παρακάμψει την επόμενη εντολή, εάν η σημαία του μηδενός (Z) στον καταχωρητή **STATUS** είναι 1.

Αντίστοιχα, η εντολή **btfsc** μεταφέρει τον έλεγχο στην μεθεπόμενη εντολή, αν το bit b του καταχωρητή Reg είναι λογικό 0:

btfsc Reg, b ;κάνε έλεγχο στο bit b του καταχωρητή Reg και να παρακάμψεις

;την αμέσως επόμενη εντολή, εάν το bit b είναι λογικό 0.

Η εντολή διαβάζεται ως εξής: Bit Test f Skip on Clear

Παράδειγμα:

btfsc PORTD, 1

Ο έλεγχος του προγράμματος θα παρακάμψει την επόμενη εντολή εάν ο ακροδέκτης RD1 της PORTD είναι 0.
Ο συνδυασμός των παραπάνω εντολών με μια εντολή **goto $-1** (πήγαινε στην αμέσως προηγούμενη εντολή), δημιουργεί ένα βρόχο ελέγχου στον οποίο παγιδεύεται η εκτέλεση του προγράμματος έως ότου εκπληρωθεί η συνθήκη:

btfsc PORTD, 1

goto $-1

επόμενη εντολή...

Στον παραπάνω βρόχο θα γίνεται διαρκώς ο έλεγος της λογικής κατάστασης του ακροδέκτη RD1, μέχρι να βρεθεί ότι ο ακροδέκτης έλαβε λογικό 0. Τότε, η εντολή **goto $-1** θα παρακαμφθεί και η εκτέλεση του προγράμματος θα μεταβεί στην *επόμενη εντολή* και θα συνεχίσει εκτελώντας με τη σειρά όσες έπονται.

Δύο ακόμη εντολές διακλάδωσης υπό συνθήκη, αυξάνουν και μειώνουν, αντίστοιχα, το περιεχόμενο ενός καταχωρητή *f* και παρακάπτουν την επόμενη αντολή αν το αποτέλεσμα της εντολής είναι μηδέν (δηλαδή η σημαία του μηδενός Z γίνει 1):

incfsz Reg ;αύξησε το περιεχόμενο του καταχωρητή Reg και παρέκαμψε
;την επόμενη εντολή αν το αποτέλεσμα είναι μηδέν

decfsz Reg ;μείωσε το περιεχόμενο του καταχωρητή Reg και παρέκαμψε
;την επόμενη εντολή αν το αποτέλεσμα είναι μηδέν

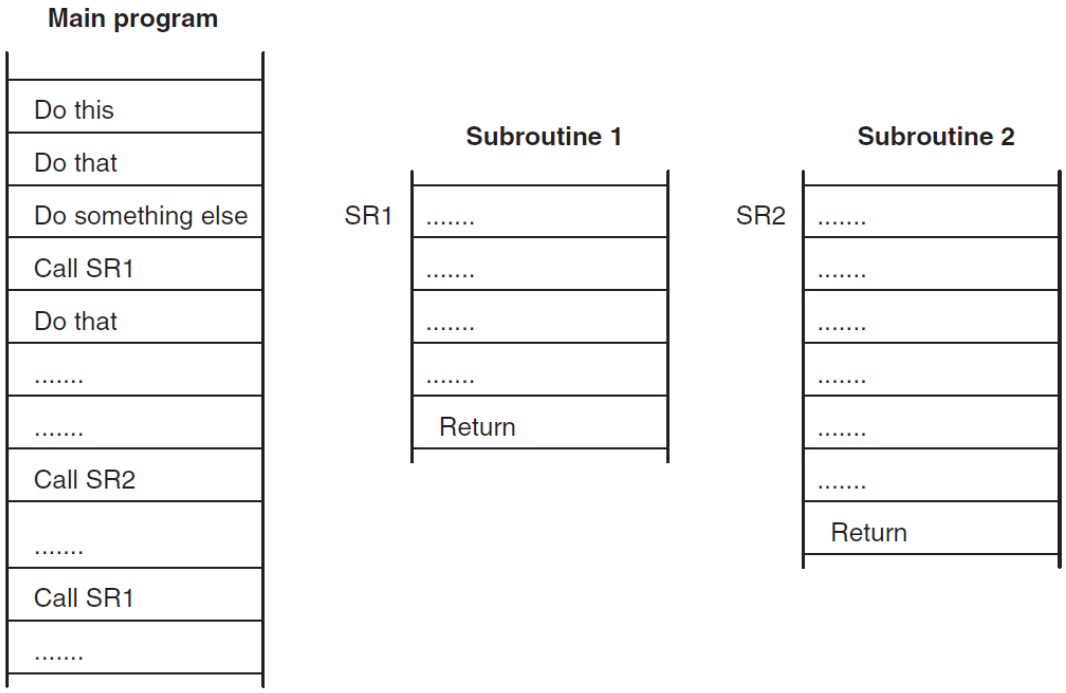
π.χ.

incfsz 20h

3.3 Υπορουτίνες

Οι υπορουτίνες επιτρέπουν να γράφουμε δομημένα προγράμματα και να επαναχρησιμοποιούμε χρήσιμο κώδικα. Μια υπορουτίνα ονομάζεται με μια ετικέτα που προηγείται της πρώτης εντολής της υπορουτίνας και τελειώναι με μια εντολή **return**. Η κλήση της γίνεται με μια εντολή **call**. Όταν ολοκληρωθεί η εκτέλεση της υπορουτίνας η εκτέλεση του προγράμματος επιστρέφει στην εντολή που ακολουθεί την **call**. Σε ένα πρόγραμμα μπορούμε να έχουμε πολλές υπορουτίνες, όπως φαίνεται στο παρακάτω σχήμα 3.1. Επίσης, είναι δυνατό να καλούμε μια υπορουτίνα μέσα από μια άλλη υπορουτίνα.

Κάθε φορά που καλείται μια υπορουτίνα, το περιεχόμενο του απαριθμητή εντολών (Program Counter) αποθηκεύεται στη στοίβα (stack). Όταν εκτελείται η εντολή **return**, τότε ανακτάται η τελευταία διεύθυνση που αποθηκεύτηκε στο σωρό και η εκτέλεση του προγράμματος μεταβαίνει στο σημείο αυτό.



Σχήμα 3.1 Οργάνωση προγράμματος με υπορουτίνες

3.4 Δομή ενσωματωμένων εφαρμογών

Ο κώδικας ενός μικροελεγκτή περιλαμβάνει τις απαραίτητες ενέργειες για την εκτέλεση του βρόχου ελέγχου που γνωρίσαμε σε προηγούμενες παραγράφους (παράγραφοι 1.2.1 και 2.1). Ο βρόχος ελέγχου είναι μια περιοδική διεργασία, και πρέπει να επαναλαμβάνεται κυκλικά. Άρα, το κυρίως πρόγραμμα (main) αντιπροσωπεύει έναν ατέρμονα βρόχο της μορφής

```
        initialize peripherals
main    do this
        do that
        read current sensor value
        compare current sensor value with reference value
        if current value < reference value
                do this
        else
                do something else
        goto main
```

Ο παραπάνω κώδικας είναι ψευδοκώδικας. Γενικά, ένας απλός κώδικας σε assembly έχει την εξής γενική δομή:

- Σύντομη τεκμηρίωση του κώδικα με τη μορφή σχολίων
- Συμπερίληψη (include) αρχείων δηλώσεων ανάλογα με τον μικροελεγκτή
- Οδηγία (macro-directive) ορισμού των ψηφίων διαμόρφωσης (configuration bits)
- Δηλώσεις ψευδωνύμων θέσεων μνήμης για ευκολία στη χρήση μεταβλητών
- Δηλώσεις μετάβασης στην αρχική θέση μνήμης προγράμματος (reset vector)
- Αρχικοποίηση περιφερειακών, όπως θύρες, χρονιστές κλπ.
- Κυρίως πρόγραμμα, όπου επιτελείται ο κυρίως βρόχος του προγράμματος

Ας σημειωθεί ότι τα bits διαμόρφωσης μπορούν να οριστούν και εκτός του κώδικα της εφαρμογής, μέσω του μενού εντολών του περιβάλλοντος ανάπτυξης. Τα bits αυτά διαμορφώνουν συγκεκριμένες γενικές ρυθμίσεις για τον τρόπο λειτουργίας της συσκευής και δεν μπορούν να αλλάξουν μέσω εντολών του προγράμματος.
Παρακάτω δίνεται ένα απλός κώδικας, που περιλαμβάνει τα παραπάνω βασικά στοιχεία για μια εφαρμογή, που απλά εισάγει τα δεδομένα που υπάρχουν στους ακροδέκτες της θύρας D και τα εξάγει αυτούσια στους ακροδέκτες της θύρας B. Επίσης, αποθηκεύει τα δεδομένα σε μια θέση μνήμης (20h), για ενδεχόμενη περαιτέρω χρήση. Στη θέση μνήμης έχουμε δώσει το μνημονικό ψευδώνυμο TEMP. Ότι ακολουθεί το ελληνικό ερωτηματικό σε μια γραμμή είναι σχόλιο και δεν εκτελείται.

```
;-------------------------------------------------------------------------------
; project_name: DIR \ test1
;TITLE: this program reads PORTD and transfers data to PORTB
;6.10.2012 by John

#include "p16f877.inc"

; Here we optionally set Configuration Word bits:
__CONFIG _CP_OFF & _WDT_OFF & _XT_OSC & _PWRTE_OFF & _CPD_OFF &
_WRT_ENABLE_ON & _BODEN_ON & _LVP_OFF
;WDT OFF, Power-up timer ON, code protect OFF, XT oscillator,
;Low-Voltage Programming OFF
TEMP equ 20h                    ;declare an alias for memory position 20h
        Org 00                  ;define the Reset vector
;Initialization of peripherals
start   bsf STATUS, RP0         ;select memory bank 1
        movlw b'00000111'
        movwf TRISD             ;the three first bits of portA are input
        movlw 00
        movwf TRISB             ;all pins of portB are output
        bcf STATUS, RP0         ;return to bank0
        clrf PORTB
```

```
;main code starts here
main        clrf PORTB
            movf PORTD, w
            movwf TEMP
            movwf PORTB
            goto main
end
```

Κώδικας 3.1 Δομή ενσωματωμένης εφαρμογής

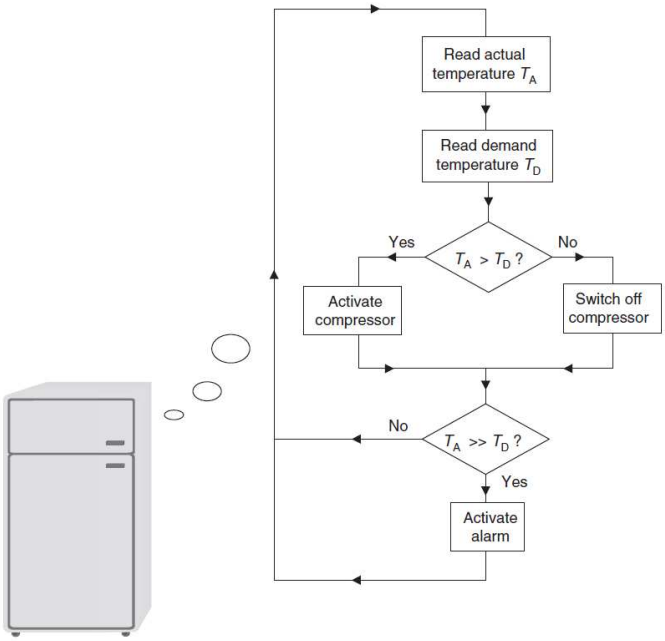
Ένας αποτελεσματικός τρόπος οργάνωσης μιας ενσωματωμένης εφαρμογής σε γλώσσα assembly είναι να οργανώνουμε τον κώδικα με βάση υπορουτίνες. Έτσι, το παραπάνω πρόγραμμα θα μπορούσε να γραφεί ως εξής

```
; project_name: DIR \ test1
;TITLE: this program reads PORTD and transfers data to PORTB
;6.10.2012 by John

#include "p16f877.inc"

__CONFIG _CP_OFF & _WDT_OFF & _XT_OSC & _PWRTE_OFF & _CPD_OFF &
_WRT_ENABLE_ON & _BODEN_ON & _LVP_OFF
TEMP equ 20h                    ;declare an alias for memory position 20h
            Org 00              ;define the Reset vector
;main code starts here
main        call init
loop        call read_display
            goto loop
;subroutines
init        bsf STATUS, RP0     ;select memory bank 1
            movlw b'00000111'
            movwf TRISD         ;the three first bits of portA are input
            movlw 00
            movwf TRISB         ;all pins of portB are output
            bcf STATUS, RP0     ;return to bank0
            clrf PORTB
            return
read_display movf PORTD, w
            movwf TEMP
            movwf PORTB
            return
end
```

Κώδικας 3.2 Δόμηση της εφαρμογής με χρήση υπορουτινών

3.5 Υλοποίηση διακλάδωσης υπό συνθήκη

Στο παρακάτω σχήμα 3.2 φαίνεται ένα τυπικό σύστημα που ενσωματώνει ένα μικροελεγκτή προκειμένου να υλοποιήσει το βρόχο ελέγχου. Το ψυγείο ενεργοποιεί το συμπιεστή, που η λειτουργία του στηρίζεται σε έναν ηλεκτρικό κινητήρα, προκειμένου να μειώσει τη θερμοκρασία του χώρου της ψύξης. Αντίστοιχα, σταματά τη λειτουργία του συμπιεστή, όταν η θερμοκρασία μειωθεί αρκετά. Αν, πάλι, η πόρτα του ψυγείου μείνει ανοιχτή για αρκετή ώρα, ώστε η θερμοκρασία ανεβεί πολύ, τότε το σύστημα παράγει μια ηχητική ειδοποίηση (alarm). Η θερμοκρασία κάθε στιγμή συγκρίνεται με τη θερμοκρασία αναφοράς, προκειμένου να ληφθούν οι αποφάσεις. Το διάγραμμα ροής της εφαρμογής αποτυπώνει τις παραπάνω ανάγκες επεξεργασίας.



Σχήμα 3.2 Διάγραμμα ροής του ελεγκτή του ψυγείου

Ένα κρίσιμο ζητούμενο είναι η υλοποίηση της διακλάδωσης υπό συνθήκη, που απορρέει από τη σύγκριση της τρέχουσας θερμοκρασίας με τη θερμοκρασία αναφοράς. Παρακάτω παρουσιάζεται ένα τμήμα κώδικα που υλοποιεί τη δομή IF…THEN…ELSE σε γλώσσα assembly.

```
movf  TA, w          ;transfer TA into w
subwf TD, w          ;compare TD with TA (TD+(-TA))
btfss STATUS,C       ;if Carry=0 then result is negative (TD<TA)
do this              ;in which case do this
btfsc STATUS,C       ;if Carry=1 then the of subwf is positive (TD>TA)
do that          ;in which case do something else
```

Κώδικας 3.3 Υλοποίηση διακλάδωσης IF…THEN…ELSE

Τα βασικά χαρακτηριστικά είναι τα εξής: α. Η σύγκριση των δύο θερμοκρασιών γίνεται με αφαίρεση. β. Αν το αποτέλεσμα της αφαίρεσης είναι θετικό, τότε το bit **C** του καταχωρητή **STATUS** είναι μονάδα. Σε αντίθετη περίπτωση, δηλαδή όταν η αφαίρεση παράγει αρνητικό αποτέλεσμα, το bit C είναι μηδέν. Σημειώνεται ότι η αφαίρεση εκτελείται με πρόσθεση του συμπληρώματος ως προς 2 του περιεχομένου του W. γ. Η διπλή χρήση της εντολής διακλάδωσης υπό συνθήκη (btfsc και btfss) εξασφαλίζει ότι θα εκτελεστεί η εντολή *do this* ή η εντολή *do that* και ποτέ και οι δύο. Φυσικά, οι *do this* και *do that* αντιστοιχούν σε ψευδοκώδικα και παριστάνουν ό,τι πρέπει να εκτελεστεί στη μία ή στην άλλη περίπτωση.
Αν αυτό που εκτελείται με τις ψευδοεντολές *do this* και *do that* περιλαμβάνει παραπάνω από μία εντολές, τότε μπορεί να γραφεί με τη μορφή υπορουτίνας και να κληθεί με εντολή call:

call *do_this*

3.5 Έμμεση προσπέλαση με τους καταχωρητές FSR και INDF

Ο ρόλος της έμμεσης διευθυνσιοδότησης είναι πολύ σημαντικός στην προσπέλαση πινάκων, δηλαδή συνόλου δεδομένων που έχουν διαταχθεί στη σειρά μέσα στη μνήμη, καταλαμβάνοντας πολλές θέσεις. Για το σκοπό αυτό οι μικροελεγκτές PIC διαθέτουν έναν καταχωρητή «δείκτη», τον **FSR**. Οι τιμές που καταχωρούνται στον **FSR** είναι θέσεις μνήμης δεδομένων. Έτσι, οι εντολές

movlw 20h

movwf FSR

καταχωρούν στον **FSR** τη διεύθυνση 20h. Αν στη συνέχεια, γράψουμε την εντολή

movf INDF, w

τότε προσπελάζεται η διεύθυνση που έχει καταχωρηθεί στον **FSR** και το περιεχόμενό της εγγράφεται στον καταχωρητή εργασίας **w**. Ο INDF στην παραπάνω εντολή δεν είναι πραγματικός καταχωρητής, αλλά η χρήση του όπως παραπάνω είναι μια συμβολική έκφραση, που όταν εκτελείται προσπελάζει τη διεύθυνση που περιέχεται στον **FSR** και μεταφέρει το περιεχόμενό της στον **w**.

Παρακάτω θα δώσουμε ένα παράδειγμα προσπέλασης πίνακα, με τη βοήθεια του καταχωρητή «δείκτη» **FSR**.

Έστω ο παρακάτω πίνακας τιμών στη μνήμη RAM

Mnemonic name	Address in RAM	Example value
startf	0x20	0x09
	0x21	0x07
	0x22	0x05
	0x23	0x1A
endf	0x24	0x2B

Τότε οι παρακάτω εντολές προσπελάζουν τον πίνακα από την αρχική διεύθυνση μέχρι την τελική και προβάλλουν το περιεχόμενο στη θύρα B:

```
        movlw startf
        movwf FSR          ;load start memory location in FSR
loop2   movf INDF,w        ;access memory location in FSR
        movwf PORTB        ;display memory content on PORTB
        movf FSR,w
        sublw endf         ;check if current memory is last
        btfsc STATUS,Z
        goto label
        incf FSR,1         ;access next memory location
        goto loop2
label   goto label         ;endless loop
```

Κώδικας 3.4 Προσπέλαση θέσεων μνήμης με τον καταχωρητή **FSR**.

Προφανώς, ο παραπάνω κώδικας δεν είναι ολοκληρωμένος, καθώς δεν περιέχει παρά μόνον το τμήμα του κυρίως προγράμματος που προσπελάζει τις θέσεις μνήμης. Η αφαίρεση που επιτελείται ανάμεσα στην τρέχουσα θέση μνήμης και στην τελευταία (endf) είναι ένας μηχανισμός με τον οποίο ελέγχεται αν ολοκληρώθηκε η προσπέλαση του πίνακα ή πρέπει να επαναληφθεί ο βρόχος loop2.

3.4 Εργαλεία Λογισμικού για τον Προγραμματισμό του Μικροελεγκτή PIC16F877

Έχοντας αποκτήσει κανείς μια εικόνα για την αρχιτεκτονική του μικροελεγκτή PIC16F84 και για τις βασικές εντολές της μνημονικής γλώσσας (*assembly*), επόμενο είναι να αναρωτιέται με ποια εργαλεία θα μπορέσει να αναπτύξει πρακτικές εφαρμογές, προγραμματίζοντας κατάλληλα τον PIC. Η παράγραφος αυτή περιέχει ορισμένες απαντήσεις, που θα καθοδηγήσουν τον αρχάριο χρήστη στην αντιμετώπιση πρακτικών προβλημάτων.

- **Ο κειμενογράφος (text editor):** Με τον κειμενογράφο συντάσσουμε το πηγαίο αρχείο σε μνημονική γλώσσα, το οποίο κατόπιν θα μεταφράσουμε και θα αποθηκεύσουμε στη μνήμη ROM του μικροελεγκτή. Τέτοιο πρόγραμμα μπορεί να είναι ένας οποιοσδήποτε κειμενογράφος χαρακτήρων ASCII, για παράδειγμα ο

κειμενογράφος EDIT του DOS ή το Notepad των Windows. Το τελικό αρχείο κειμένου πρέπει να έχει επέκταση .ASM γι' αυτό πρέπει να αποθηκεύσει κανείς το αρχείο που πληκτρολόγησε δίνοντας ως επέκταση τα αρχικά ".ASM".

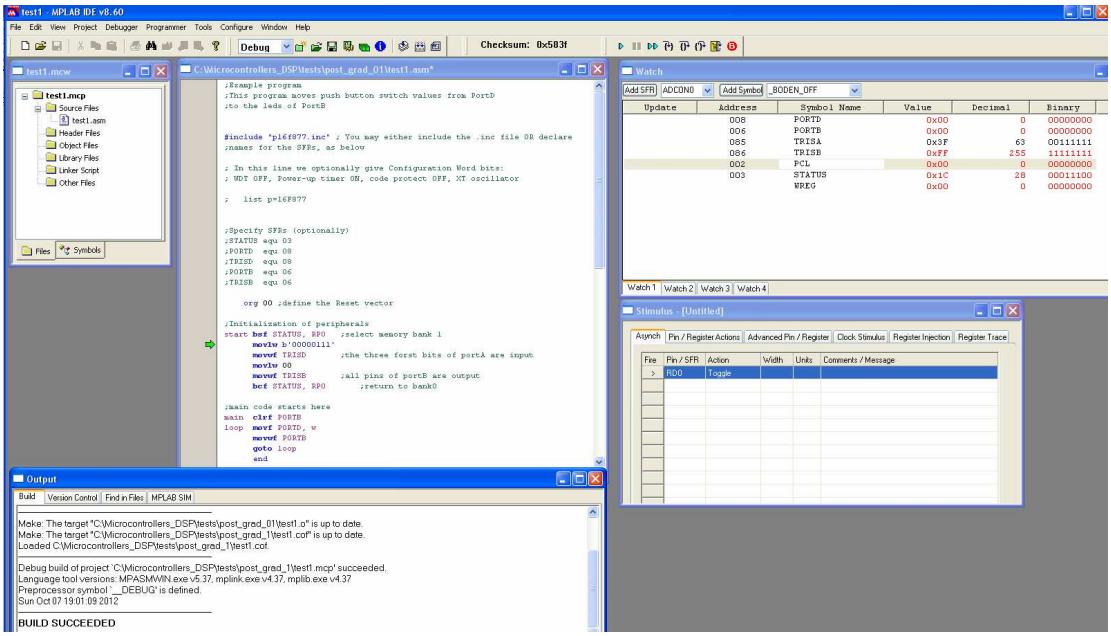
- **Ο συμβολομεταφραστής (assembler):** Πρόκειται για το πρόγραμμα που μεταφράζει το πηγαίο αρχείο .ASM σε δεκαεξαδική μορφή, κατάλληλη να τη διαχειριστεί ο προγραμματιστής της μνήμης EEPROM του μικροελεγκτή. Αν και κυκλοφορούν πολλοί συμβολομεταφραστές, μία εύκολη και καλή λύση είναι να χρησιμοποιήσει κανείς το πρόγραμμα MPASM της εταιρείας Microchip, το οποίο αποτελεί μέρος του ολοκληρωμένου περιβάλλοντος MPLAB και διατίθεται χωρίς χρέωση στην ιστοσελίδα της εταιρείας. Κατά τη μετάφραση παράγονται τα μηνύματα σφαλμάτων, μετά τη διόρθωση των οποίων ο συμβολομεταφραστής παράγει ένα αρχείο με επέκταση .HEX, το οποίο μπορεί να χρησιμοποιηθεί για τα επόμενα βήματα. Επίσης, παράγονται αρχεία που περιέχουν αναφορές σφαλμάτων καθώς και αρχεία εισόδου για τον προσομοιωτή (simulator).

- **Ο προσομοιωτής (simulator):** Αυτός είναι ένα βοηθητικό και όχι υποχρεωτικό πρόγραμμα. Τέτοιο πρόγραμμα είναι το MSIM της εταιρείας Microchip. Το πρόγραμμα αυτό χρησιμοποιεί σαν αρχείο εισόδου ένα αρχείο με επέκταση .INI που παράγεται από το πρόγραμμα MPASM κατά τη διαδικασία της συμβολομετάφρασης. Το πρόγραμμα MSIM παράγει μία βήμα προς βήμα προσομοίωση της εκτέλεσης του προγράμματος και επιδεικνύει τις τιμές που λαμβάνουν οι διάφοροι καταχωρητές ειδικού σκοπού και οι θέσεις της μνήμης RAM. Με τον τρόπο αυτόν ο χρήστης μπορεί να βεβαιωθεί ότι το πρόγραμμα επιτελεί ακριβώς αυτό για το οποίο προορίζεται και να ανιχνεύσει τα τυχόν σφάλματα που υπάρχουν.

Τα παραπάνω εργαλεία μπορεί να τα βρει κανείς ενσωματωμένα σε ολοκληρωμένα αναπτυξιακά εργαλεία λογισμικού, όπως είναι το πρόγραμμα MPLAB της εταιρείας Microchip. Το πρόγραμμα αυτό αναφέρεται ως *Ολοκληρωμένο Περιβάλλον Ανάπτυξης Εφαρμογών* (*IDE – Integrated Development Environment*). Είναι ένα σπονδυλωτό παραθυρικό πρόγραμμα που περιέχει σε ενιαίο γραφικό περιβάλλον τον κειμενογράφο, τον συμβολομεταφραστή MPASM και τον προσομοιωτή MSIM της εταιρείας Microchip. Το πρόγραμμα MPLAB δημιουργεί *σχέδια εργασίας* (*projects*) που περιέχουν όλα τα απαραίτητα αρχεία κάθε εφαρμογής. Για τον προγραμματισμό της συσκευής μπορεί να συνδεθεί με κυκλώματα προγραμματισμού, όπως ο PICSTART Plus ή ο PICkit3. Το πρόγραμμα διατίθεται δωρεάν από την εταιρεία Microchip, στην ιστοσελίδα www.microchip.com (βλέπε και Παράρτημα Α).

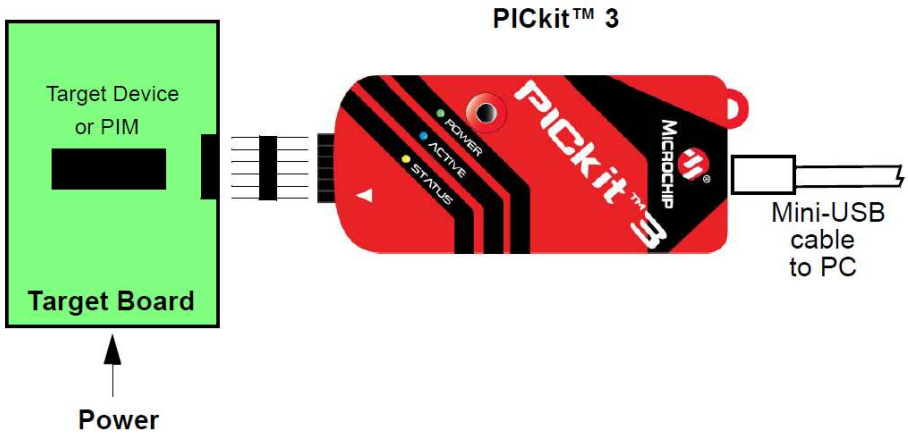
Τα παραπάνω λογισμικό μπορεί να χρησιμοποιηθεί για την ανάπτυξη εφαρμογών με κάθε μικροελεγκτή της εταιρείας Microchip και όχι μόνον με τον PIC16F877.

Στην παρακάτω εικόνα φαίνεται το γραφικό περιβάλλον του προγράμματος MPLAB, όπου τα διάφορα παράθυρα είναι ο κειμενογράφος, η κονσόλα όπου παράγονται τα μηνύματα εξόδου (Output) και παράθυρα προσομοίωσης του κώδικα.



Εικόνα 3.1: Το περιβάλλον MPLAB

Ο προγραμματισμός του μικροελεγκτή γίνεται με ειδικά κυκλώματα, που συνδέονται στον υπολογιστή. Τα κυκλώματα αυτά καταλήγουν σε ακροδέκτες του μικροελεγκτή, που διασυνδέονται με τη μνήμη προγράμματος. Η συσκευή της εικόνας 3.2 είναι ο προγραμματιστής PICkit3, που προγραμματίζει τον μικροελεγκτή πάνω στο κύκλωμα (in-circuit programmer)



Εικόνα 3.2: Προγραμματιστής (In-circuit programmer-debugger) PICkit3

4. Είσοδος/έξοδος και χρονισμός του μικροελεγκτή PIC

4.1 Είσοδος/έξοδος και καταχωρητές θυρών

Οι καταχωρητές **PORTB** και **TRISB**, **PORTD** και **TRISD**, καθώς και οι αντίστοιχοι καταχωρητές των υπόλοιπων θυρών, χρησιμεύουν στη λειτουργία εισόδου/εξόδου, δηλαδή στην ανάγνωση και εγγραφή δεδομένων από και προς τις θύρες. Η εγγραφή σε έναν καταχωρητή **TRIS** ορίζει ποιοι ακροδέκτες της αντίστοιχης θύρας λειτουργούν σαν είσοδοι και ποιοι σαν έξοδοι.

Εγγράφοντας στον καταχωρητή **TRISB** τον δυαδικό αριθμό 11110000, ορίζουμε τα τέσσερα λιγότερο σημαντικά bits της θύρας B ως εξόδους και τα τέσσερα περισσότερο σημαντικά bits ως εισόδους (0 = έξοδος, 1 = είσοδος):

movlw b'11110000'
movwf TRISB

Ας θυμηθούμε ότι η θύρες **B, C, D** έχουν οκτώ ακροδέκτες, η θύρα **A** έχει έξι και η θύρα **E** έχει τρεις (στον μικροελεγκτή PIC16F877). Όλοι αυτοί μπορούν να οριστούν σε κάθε στιγμή ως είσοδοι ή έξοδοι. Επίσης, ας θυμηθούμε ότι η πρόσβαση στον καταχωρητή **TRISB** για τον ορισμό της διεύθυνσης λειτουργίας της θύρας **B**, απαιτεί μετάβαση στη σελίδα 1 της μνήμης (bit RP0 του καταχωρητή **STATUS** σε λογικό ένα). Το ίδιο, βέβαια, συμβαίνει και με τους υπόλοιπους καταχωρητές **TRIS**. Όταν τελειώσει η διαδικασία αρχικοποίησης των θυρών, πρέπει να επιστρέψουμε στη βασική σελίδα μνήμης. Ο ολοκληρωμένος κώδικας για την αρχικοποίηση της θύρας B έχει ως εξής:

bsf STATUS, RP0
movlw b'11110000'
movwf TRISB
bcf STATUS, RP0

Η εντολή ανάγνωσης ενός καταχωρητή θύρας, π.χ. του **PORTB**, διαβάζει την κατάσταση των ακροδεκτών της θύρας και τη μεταφέρει στον καταχωρητή εργασίας **w**:

movf PORTB, w

Η εντολή εγγραφής του **w** στον καταχωρητή **PORTB** εμφανίζει το περιεχόμενο του **w** στους αντίστοιχους ακροδέκτες:

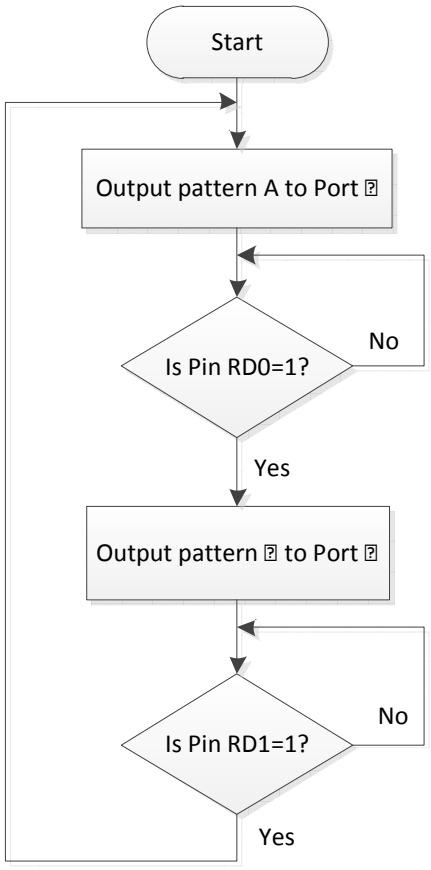
movwf PORTB

Οι μικροελεγκτές PIC έχουν τη δυνατότητα μέσω ειδικών εντολών να εγγράφουν ένα μόνο bit κάποιου καταχωρητή, αγνοώντας το υπόλοιπο μέρος του συγκεκριμένου byte:

bsf PORTB, 2 ;θέσε σε λογικό 1 το bit RB2

4.1.2 Προγραμματισμός των θυρών PIO του PIC16F877.

Το παρακάτω πρόγραμμα επιδεικνύει τον τρόπο προγραμματισμού των θυρών Parallel I/O (PIO) του PIC για είσοδο και για έξοδο. Δύο διακόπτες πιέσεως (push buttons) συνδέονται με τους ακροδέκτες RD0 και RD1 και όταν πιέζονται μεταφέρουν στους ακροδέκτες τάση 5V, που αντιστοιχεί σε λογικό 1. Στην αρχή, ο κώδικας εξάγει ένα φωτεινό μοτίβο στη θύρα B και αναμένει την πίεση του πρώτου διακόπτη. Στη συνέχεια, μόλις ο διακόπτης πιεστεί, αλλάζει το φωτεινό μοτίβο και αναμένει την πίεση του δεύτερου διακόπτη. Όταν πιεστεί κι αυτός, η διαδικασία επαναλαμβάνεται, με την εμφάνιση του πρώτου μοτίβου. Το διάγραμμα ροής του κώδικα παρουσιάζεται στο σχήμα 4.1.



Σχήμα 4.1 Το διάγραμμα ροής της εφαρμογής εισόδου/εξόδου

Ο κώδικας που υλοποιεί τις παραπάνω λειτουργίες παρατίθεται παρακάτω:

```
#include "p16F877.inc"
        ;Initialization
        Org 0                   ;Start from program memory address 0
        bsf STATUS, RP0         ;Change to bank1
```

```
        movlw b'00000000'    ;Make all pins of portB Outputs
        movwf TRISB
        movlw b'00011111'    ;Make 5 pins of portD inputs
        movwf TRISD
        bcf STATUS, RP0      ;Return to bank0

main    movlw b'01010101'    ;Output to portB
        movwf PORTB
        btfss PORTD, 0       ;Wait until pin0 of portD receives 1
        goto $-1
        movlw b'10101010'    ;If pin0 of portD is zero, change output combination to
                             ;portB
        movwf PORTB
        btfss PORTD, 1       ;Wait until pin1 of portD receives 1
        goto $-1
        goto main
        end
```

Κώδικας 4.1 Παράδειγμα προγραμματισμού θυρών εισόδου/εξόδου

4.2 Χρονισμός

4.2.1 Κύκλος εκτέλεσης εντολής

Οι περισσότερες εντολές των μικροελεγκτών PIC εκτελούνται σε έναν κύκλο εκτέλεσης (instruction cycle). Αυτό είναι αποτέλεσμα της αρχιτεκτονικής RISC που υποστηρίζουν αυτές οι μονάδες μικροελεγκτών. Κάθε κύκλος εκτέλεσης περιλαμβάνει τα βήματα fetch, decode, execute και store, οπότε κάθε εντολή ολοκληρώνεται σε τέσσερις περιόδους του εξωτερικού ρολογιού. Γενικά αναφερόμαστε στη συχνότητα του εξωτερικού ρολογιού ως f_{osc} και στην περίοδο ως T_{osc}. Άρα

fetch, decode, execute and store = 1 instruction cycle

Διάρκεια ενός κύκλου εντολής = $4*T_{osc}$ (4.1)

Ο κύκλος εκτέλεσης εντολής ορίζει την περίοδο μιας εσωτερικής γεννήτριας παλμών, που ονομάζεται ρολόϊ εντολών του PIC.

Γνωρίζοντας πόσους εξωτερικούς παλμούς χρειάζεται κάθε εντολή για να ανακτηθεί από την μνήμη και να εκτελεστεί μπορούμε να υπολογίσουμε το συνολικό χρόνο που χρειάζεται ένα πρόγραμμα ή μια ρουτίνα για να εκτελεστεί.

Άρα, τη χρονική διάρκεια κάθε εντολής μπορούμε να την υπολογίσουμε αν γνωρίζουμε την συχνότητα του εξωτερικού ταλαντωτή, σύμφωνα με τη σχέση:

Διάρκεια κύκλου εντολής = 4 / (Συχνότητα εξωτερικού κρυστάλλου) (4.2)

Τυπική συχνότητα λειτουργίας των μικροελεγκτών PIC είναι 4MHz, οπότε η διάρκεια κάθε κύκλου εντολής είναι **1μsec.** Για κρύσταλλο 8MHz η διάρκεια του κύκλου εντολής είναι 0,5 μsec.

Όπως είναι εύκολα κατανοητό, η διάρκεια εκτέλεσης ενός προγράμματος, ισούται με τον αριθμό τον εντολών που το αποτελούν επί τη διάρκεια του κύκλου εκτέλεσης της εντολής. Κάποιες εντολές χρειάζονται παραπάνω από έναν κύκλο εντολής για να εκτελεστούν. Τέτοιες είναι οι εντολές αλλαγής της ροής του προγράμματος (Π.χ. goto, call) που χρειάζονται δύο κύκλους εντολής για να εκτελεστούν.
Με βάση τα παραπάνω μπορούν να δημιουργηθούν κατάλληλες υπορουτίνες καθυστέρησης.

4.2.2 Θέματα χρονισμού και καθυστέρησης της εκτέλεσης (delay)

Όπως αναφέρθηκε στην προηγούμενη παράγραφο, αν η συχνότητα του εξωτερικού κρυστάλλου είναι f_{osc} = 4MHz, τότε η περίοδος είναι

$T_{osc} = 1/f_{osc} = 0{,}25$ μs.

Άρα, ένας πλήρης κύκλος εκτέλεσης εντολής διαρκεί 1μs.

Συνεπώς, σε ένα τέτοιο σύστημα η εκτέλεση των περισσότερων εντολών διαρκεί 1μs.

Εξαίρεση αποτελούν οι εντολές διακλάδωσης, που αναφέρθηκαν στις προηγούμενες παραγράφους. Έτσι, η εντολή goto, η εντολή call και η εντολή return διαρκούν δύο κύκλους εκτέλεσης εντολής, δηλαδή για εξωτερικό κρύσταλλο με συχνότητα 4MHz αυτές οι εντολές διαρκούν 2μs.

Οι εντολές ελέγχου bit διαρκούν 1 κύκλο εντολής αν δεν εκτελούν παράκαμψη της επόμενης εντολής (δηλαδή δεν συντρέχει η συνθήκη τους), αλλά διαρκούν 2 κύκλους, αν εκτελούν την παράκαμψη.

Έτσι, η παρακάτω υπορουτίνα delay παράγει μια συνολική καθυστέρηση 1ms.

```
;With crystal XT and fosc=4MHz, instruction cycle is 1 μs
;branch instructions (call, goto) take 2μs

delay   movlw 0F9       ;load decimal 249
        nop             ;first two instructions take 2μs      } 4μs
micro4  addlw 0FF       ;add -1 in 2's complement. 1μs        }
```

```
        btfss STATUS, Z     ;no skip: 1μs. With skip: 2μs
        goto micro4         ;2μs
        return              ;2μs
```

Κώδικας 4.2: Υπορουτίνα καθυστέρησης, για συνολική καθυστέρηση 1ms

Αρχικά φορτώνεται ο W με τον δεκαδικό 249 (διάρκεια 1μs) και ακολουθεί μια εντολή nop που απλώς διαρκεί 1μs χωρίς να επιτελεί τίποτε (no operation). Ο βρόχος micro4 - goto micro4 διαρκεί ακριβώς 4μs, διότι η πρώτη εντολή (addlw) διαρκεί 1μs, η εντολή btfss διαρκεί 1μs όταν δεν συντρέχει η συνθήκη και η goto διαρκεί 2 μs. Όταν συντρέχει η συνθήκη Z=1, τότε η btfss διαρκεί 2 μs. Άρα σε κάθε περίπτωση ο βρόχος διαρκεί 4 μs. Σε κάθε εκτέλεση του βρόχου, η εντολή addlw προσθέτει στο 249 το -1 (FF ως προσημασμένος, με το συμπλήρωμα ως προς 2), οπότε το αποτέλεσμα της άθροισης θα γίνει μηδέν σε 249 κύκλους εκτέλεσης του βρόχου, δηλαδή σε συνολική διάρκεια:

249 x 4 μs = 996 μs

Στη διάρκεια αυτή πρέπει να προστεθούν και τα 2 μs των δύο πρώτων εντολών, καθώς και τα 2μs της εντολής call delay. Σύνολο, 1000 μs, δηλαδή 1ms.

Η παρακάτω παραλλαγή είναι μια υπορουτίνα που ονομάζεται delay_ms και δημιουργεί καθυστέρηση όσα ms έχουν εγγραφεί στον καταχωρητή msec (από 1 έως 255).

```
delay_ms    movwf msec
loop        movlw 0F8       ;decimal 248
            call micro4     ;248x4+2=994μs
            nop
            nop
            decfsz msec, f
            goto loop
            return
```

Κώδικας 4.3 Υπορουτίνα καθυστέρησης έως 255 ms

Η υπορουτίνα delay_ms καλεί την υπορουτίνα micro4 που περιγράψαμε παραπάνω 248 φορές (συνολική διάρκεια 994 μs) και επαναλαμβάνει την κλήση αυτή μέσα σε ένα βρόχο που εκτελείται msec φορές. Ο βρόχος προσθέτει κάθε φορά τα μs που υπολείπονται, μέχρι τα 1000μs. Έτσι, ο κώδικας

```
        msec equ 30h
        movlw d'35'
        call delay_ms
```

παράγει εξαιτίας της delay_ms μια καθυστέρηση 35 ms.

4.2.3 Χρονιστές (timers) στους PIC

Ο PIC διαθέτει τρεις μετρητές: δύο οκτάμπιτους (8bits) χρονιστές/μετρητές (timer/counter), τους Timer0,Timer2 και έναν δεκαξάμπιτο (16bits), τον Timer1.

Ο Timer0, είναι ένας πλήρως προγραμματιζόμενος απαριθμητής, του οποίου το περιεχόμενο προσπελάζεται μέσω του ειδικού καταχωρητή TMR0. Σε κάθε παλμό ρολογιού, το περιεχόμενο του TMR0 αυξάνει κατά ένα. Το κύκλωμα μπορεί να λειτουργήσει με δύο τρόπους.

α) *Ως χρονιστής (timer)*: Στην περίπτωση αυτή η πηγή χρονισμού είναι ο εσωτερικός κύκλος των εντολών. Το περιεχόμενο του TMR0 σε λειτουργία χρονιστή αυξάνει κατά ένα σε κάθε κύκλο εντολής. Ο κύκλος της εντολής διαρκεί όσο η περίοδος του εξωτερικού κρυστάλλου επί τέσσερα ($4*T_{osc}$). Η αντίστοιχη συχνότητα είναι ίση με την συχνότητα f_{osc} του κρυστάλλου διά 4. Χρησιμοποιώντας τον Timer0 σε λειτουργία εσωτερικού χρονισμού, έχουμε στη διάθεσή μας ένα αρκετά αξιόπιστο σήμα χρονισμού που μπορεί να χρησιμοποιηθεί για τον ακριβή προσδιορισμό χρονικών διαστημάτων.

β) *Ως απαριθμητής εξωτερικών συμβάντων (counter)*: Στην περίπτωση αυτή χρησιμοποιείται ένας εξωτερικός παλμός χρονισμού. Το εξωτερικό σήμα χρονισμού συνδέεται με τον ακροδέκτη TOCKI (RA4). Η χρήση εξωτερικής πηγής χρονισμού επιτρέπει την απαρίθμηση εξωτερικών συμβάντων, με τη μορφή παλμών. Το περιεχόμενο του Timer0 αυξάνει κατά ένα σε κάθε παλμό της εξωτερικής πηγής χρονισμού.

Και στις δύο περιπτώσεις, ο **TMR0** μετρά από την αρχική τιμή που καταχωρήσαμε έως 0xFF (στο δεκαεξαδικό σύστημα).

Όπως θα δούμε, η επιλογή του είδους της πηγής χρονισμού του TMR0, επιτυγχάνεται με τη χρήση του bit T0CS του ειδικού καταχωρητή OPTION_REG.

Τα bits του καταχωρητή **OPTION_REG** καθορίζουν εάν θα μπει σε λειτουργία ο απαριθμητής και με ποιον από όλους τους δυνατούς τρόπους θα εργαστεί. Ο πίνακας 4-1 παρουσιάζει τα bits του καταχωρητή OPTION_REG και τη σημασία τους.

Πίνακας 4-1
Τα bits του καταχωρητή OPTION_REG, ο οποίος ρυθμίζει τη λειτουργία του χρονιστή TMR0

b7	b6	b5	b4	b3	b2	b1	b0
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

- **Bits 7-6**: Δεν έχουν σημασία για την λειτουργία του χρονιστή (τα θέτουμε 11).
- **T0CS**: Επιλογή πηγής ρολογιού
  1 = Πηγή συνδεδεμένη στον ακροδέκτη T0CKI (Λειτουργία μετρητή παλμών)

  0 = Εσωτερική πηγή ρολογιού (Λειτουργία χρονιστή).
- **T0CE:** Επιλογή θετικού ή αρνητικού μετώπου παλμού (αναφέρεται σε εξωτερική πηγή χρονισμού)
  1 = Αύξηση με κατερχόμενο μέτωπο του παλμού στον ακροδέκτη T0CLK

  0 = Αύξηση με ανερχόμενο μέτωπο του παλμού στον ακροδέκτη T0CLK
- **PSA**: "0", όταν θέλουμε να συνδέσουμε τον διαιρέτη συχνότητας.
  "1", όταν θέλουμε να αποσυνδέσουμε τον διαιρέτη συχνότητας.
- **PS2-PS0***:* Επιλογή λόγου διαίρεσης

Εάν επιθυμούμε να αυξηθεί ο χρόνος τον οποίο μετρά ο απαριθμητής **TMR0**, μπορούμε να χρησιμοποιήσουμε έναν διαιρέτη συχνότητας (*prescaler*). Αυτός μπορεί να ρυθμιστεί κατάλληλα, ώστε να διαιρεθεί η συχνότητα αύξησης του μετρητή κατά μία επιθυμητή τιμή. (1:2, 1:4, 1:8, ... 1:256).

Πίνακας 4-2

Διαίρεση της συχνότητας με την οποία αυξάνεται ο TMR0

Τιμή προμετρητή			Ρυθμός TMR0
PS2	PS1	PS0	
0	0	0	1:2
0	0	1	1:4
0	1	0	1:8
0	1	1	1:16
1	0	0	1:32
1	0	1	1:64
1	1	0	1:128
1	1	1	1:256

Η ρύθμιση του διαιρέτη συχνότητας γίνεται ορίζοντας κάποια bits του καταχωρητή **OPTION_REG** (Πίνακας 4-1). Συγκεκριμένα, ορίζουμε τα τρία πρώτα bits (PS0, PS1, και PS2) του καταχωρητή **OPTION_REG** από p=000 έως p=111 (δηλαδή στο δεκαδικό σύστημα από 0 έως 7), και έτσι μεταβάλλουμε τον ρυθμό του απαριθμητή **TMR0** κατά τον παράγοντα $1:2^{p+1}$

Εάν το bit PSA του καταχωρητή **OPTION_REG** τεθεί σε λογικό ένα, τότε ο διαιρέτης αποσυνδέεται από τον απαριθμητή **TMR0** και συνδέεται με τον χρονιστή επιτήρησης (**WDT**, βλέπε παρακάτω). Στην περίπτωση αυτή δεν συμβαίνει καμία καθυστέρηση στο χρόνο αύξησης του περιεχομένου του απαριθμητή **TMR0** και αυτός αυξάνει κατά ένα σε κάθε κύκλο εντολής.

Εάν το bit PSA του καταχωρητή **OPTION_REG** τεθεί σε λογικό μηδέν, τότε ο διαιρέτης συχνότητας συνδέεται με τον απαριθμητή **TMR0** και διαιρεί τη συχνότητα αύξησης κατ' ελάχιστο διά δύο.

Όταν ο Timer0 λειτουργεί ως χρονιστής, τότε ο χρόνος που χρειάζεται για να υπερχειλίσει ο καταχωρητής TMR0 δίνεται από τη σχέση:

$$\text{Delay} = \frac{(256 - \text{αρχικό περιεχόμενο του TMR0}) * \text{λόγος διαίρεσης συχνότητας} * 4}{\text{Συχνότητα κρυστάλλου}} \quad (4.3)$$

Όταν ο απαριθμητής **Timer0** υπερχειλίσει, δηλαδή όταν ο καταχωρητής του **TMR0** μεταβεί από την ανώτερη τιμή 0xFF στην αρχική τιμή 0x00, τότε μια σημαία σηματοδοτεί την υπερχείλιση μεταβαίνοντας σε λογικό 1. Η σημαία αυτή ονομάζεται T0IF και βρίσκεται στον καταχωρητή INTCON (b2). Στο σημείο αυτό, και εφόσον έχει ενεργοποιηθεί η αντίστοιχη διακοπή (*interrupt*), δημιουργείται ένα *σήμα διακοπής*. Η εκτέλεση του προγράμματος διακόπτεται και καλείται η ρουτίνα εξυπηρέτησης της διακοπής. Πιο λεπτομερής αναφορά στα σήματα διακοπών στους PIC γίνεται σε επόμενη παράγραφο. Ας σημειωθεί ότι η συγκεκριμένη χρήση του απαριθμητή/χρονιστή **Timer0** για τη δημιουργία διακοπών χρησιμοποιείται σε εφαρμογές στις οποίες η καταμέτρηση του χρόνου έχει κρίσιμη σημασία. Παράδειγμα τέτοιας εφαρμογής είναι η χρήση του μικροελεγκτή για λήψη και εκπομπή σημάτων, με τη μέθοδο της ασύγχρονης σειριακής επικοινωνίας.

Στο σημείο αυτό θα γίνει μια σύντομη αναφορά σε έναν ακόμη χρονιστή που διαθέτει ο PIC και ονομάζεται χρονιστής επιτήρησης (Watch-Dog Timer ή WDT). Ο χρονιστής επιτήρησης, όταν ενεργοποιείται, επανεκκινεί (reset) τη CPU σε τακτά χρονικά διαστήματα, ώστε να αποκλείεται η παγίδευση της εκτέλεσης του κώδικα σε κάποιο αδιέξοδο. Η ενεργοποίηση και η απενεργοποίηση του χρονιστή επιτήρησης γίνεται με τη βοήθεια των bits διαμόρφωσης, που εγγράφονται στη λέξη διαμόρφωσης όταν η συσκευή προγραμματίζεται. Ανάμεσα στα bits διαμόρφωσης (configuration bits) υπάρχει ένα που ενεργοποιεί ή απενεργοποιεί τον WDT (WDT_ON ή WDT_OFF). Περισσότερα για τα bits διαμόρφωσης αναφέρονται στην παράγραφο 3.4 και στα εργαστηριακά φύλλα έργου.

Όταν ο διαιρέτης συχνότητας αποσυνδέεται από το κύκλωμα του χρονιστή, τότε συνδέεται με τον WDT (PSA=1). Στην περίπτωση αυτή, εφόσον έχει ενεργοποιηθεί ο

WDT η συχνότητα επανεκκίνησης υποδιαιρείται με τον τρόπο που εμφανίζεται στον πίνακα 4-2.

4.3 Προσομοίωση της λειτουργίας του χρονιστή

Ο παρακάτω κώδικας προγραμματίζει τον χρονιστή Timer0 και επιτρέπει να παρατηρήσουμε την υπερχείλιση του καταχωρητή **TMR0** σε περιβάλλον προσομοίωσης (βλέπε και εργαστηριακά φύλλα έργου).

```
#include "P16F877.inc"
        Org 0                   ;Το πρόγραμμα ξεκινά από τη θέση μνήμης προγρ. 0
        bsf STATUS, RP0         ;Μεταβαίνουμε στην σελίδα 1 της μνήμης
        movlw b'11010001'       ;Διαίρεση συχνότητας διά 4
        movwf OPTION_REG        ;Εγγράφουμε τον OPTION REG
        bcf STATUS, RP0         ;Μεταβαίνουμε στην σελίδα 0 της μνήμης
        movlw 0F0h              ;Θέτουμε στον TMR0 αρχική τιμή 240
        movwf TMR0
        bcf INTCON, T0IF        ;Μηδενίζουμε την σημαία T0IF
loop    goto loop               ;Βρόχος αναμονής
        END
```

Κώδικας 4.4 Ρύθμιση και φόρτωση του χρονιστή, με σκοπό την προσομοίωση της διαδικασίας

Στον παραπάνω κώδικα, θέτοντας το PSA bit του OPTION_REG μηδέν, συνδέουμε τον χρονιστή TIMER0 με τον διαιρέτη συχνότητας. Θέτοντας τα bits PS2:PS0=001 ορίζουμε τη συχνότητα αύξησης του TIMER0 ίση με τη συχνότητα ρολογιού εντολών διά 4. Όταν η εκτέλεση του προγράμματος φτάσει στον βρόχο αναμονής «loop GOTO loop», τότε ο TIMER0 αυξάνεται κατά 1 για κάθε δύο εκτελέσεις της εντολής βρόχου. Ο λόγος είναι ότι η εντολή GOTO διαρκεί δύο κύκλους εντολής (2x2=4 ο λόγος διαίρεσης).

4.4 Ρολόϊ πραγματικού χρόνου

Το παρακάτω πρόγραμμα μετρά με ακρίβεια το χρονικό διάστημα που ορίζουμε στον καταχωρητή SEC. Ο εξωτερικός ταλαντωτής έχει συχνότητα 8MHz. Ο προγραμματισμός του χρονιστή Timer0 γίνεται με βάση τη σχέση (4.3).

```
#include "P16F877.INC"
        Org 0
CENT equ 20h                ; Δίνουμε στη θέση μνήμης 20h το όνομα CENT
SEC equ 21h                 ; Δίνουμε στη θέση μνήμης 21h το όνομα SEC
```

```
        movlw d'5'
        movwf SEC              ;Ορίζουμε χρόνο 5 sec και το αποθηκεύουμε στην θέση SEC
        clrf CENT              ;Μηδενίζουμε την θέση CENT (εκατοστά του δευτερολέπτου)
        bsf STATUS, RP0        ;Μεταβαίνουμε στην σελίδα μνήμης 1
        movlw b'11111100'
        movwf TRISB            ; Κάνουμε τα δύο χαμηλότερα bits της θύρας B έξοδο
        movlw b'11010111'
        movwf OPTION_REG       ;Ορίζουμε διαίρεση συχνότητας 1/256 PS2:PS0=111
        bcf STATUS, RP0        ; Επιστρέφουμε στην σελίδα μνήμης 0
        movlw b'00000010'      ; Ανάβουμε το δεύτερο LED της PORTB
        movwf PORTB
loop1   movlw d'178'
        movwf TMR0             ; Χρόνος=(256-178)*256*4/8 σε μsec (=0,01 sec). Σχέση (4.3).
        bcf INTCON, T0IF       ; Μηδενίζουμε τη σημαία T0IF
loop2   btfss INTCON, T0IF     ; Αναμονή 0,01 sec
        goto loop2
        incf CENT,1
        movlw d'100'
        subwf CENT,w
        btfss STATUS,Z
        goto loop1             ; Επαναλαμβάνει την αναμονή 100 φορές
        clrf CENT
        decfsz SEC,f
        goto loop1             ; Επαναλαμβάνει τη συνολική αναμονή SEC φορές
        movlw b'00000001'
        movwf PORTB            ; Ανάβει το πρώτο LED
loop3   goto loop3
        END
```

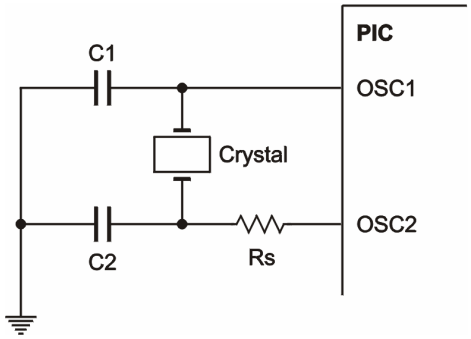
Κώδικας 4.5 Μέτρηση δευτερολέπτων με χρήση του Timer0

4.4 Εξωτερικός ταλαντωτής

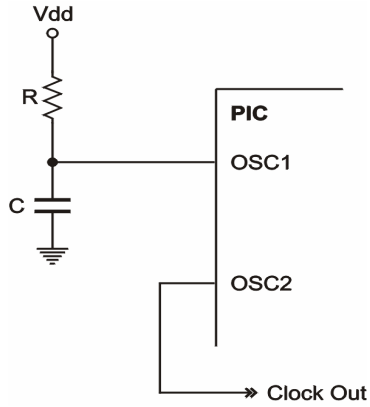
Θα κλείσουμε τη σύντομη αυτή παρουσίαση των κυκλωμάτων χρονισμού με μία αναφορά στα εξωτερικά κυκλώματα που μπορούμε να χρησιμοποιήσουμε, προκειμένου να εξασφαλίσουμε τους κύκλους του ωρολογιακού σήματος για τα κυκλώματα του μικροελεγκτή. Το Σχήμα 4.2 παρουσιάζει πώς συνδέουμε έναν κρυσταλλικό ταλαντωτή με τη βοήθεια δύο πυκνωτών. Ας σημειωθεί ότι μία τιμή 22-33pF για τους πυκνωτές C1 και C2 είναι κατάλληλη στις περισσότερες περιπτώσεις.

Ένας ταλαντωτής με ιδιοσυχνότητα 4MHz είναι μια συνηθισμένη και φθηνή λύση, η οποία μάλιστα εξασφαλίζει ικανοποιητικές ταχύτητες για τις περισσότερες εφαρμογές. Επιπλέον έχει το πλεονέκτημα ότι μας βοηθά να προγραμματίζουμε εφαρμογές στις

οποίες ο χρονισμός αποτελεί κρίσιμο παράγοντα. Ο λόγος είναι ότι με αυτήν την εξωτερική πηγή χρονισμού, η εκτέλεση κάθε εντολής διαρκεί ακριβώς 1μs, αφού όπως προαναφέραμε ο κύκλος εκτέλεσης κάθε εντολής διαρκεί τέσσερις συνολικά κύκλους του εξωτερικού ρολογιού. Έτσι διευκολύνεται σημαντικά η καταμέτρηση του χρόνου κατά την εκτέλεση του προγράμματος.



Σχήμα 4.2 Σύνδεση κρυσταλλικού ταλαντωτή



Σχήμα 4.3 Σύνδεση ταλαντωτή RC

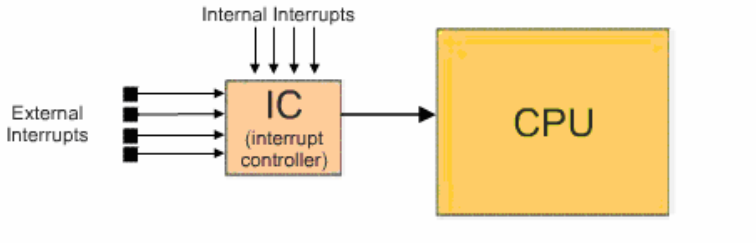
Σε απλούστερες εφαρμογές, όπου η ταχύτητα και η ακρίβεια της συχνότητας του ωρολογιακού σήματος δεν παίζουν ιδιαίτερο ρόλο, είναι δυνατό να χρησιμοποιούμε για τον χρονισμό απλούς ταλαντωτές RC. Μία τέτοια τυπική συνδεσμολογία παρουσιάζεται στο Σχήμα 4.3.

Ο χρονισμός των διαφόρων εφαρμογών απαιτεί συχνά την αναμονή σε κάποιο βρόγχο καθυστέρησης. Η χρονική ακρίβεια με την οποία εκτελείται μια υπορουτίνα καθυστέρησης επιτρέπει την υλοποίηση εφαρμογών με ιδιαίτερες απαιτήσεις στην καταμέτρηση χρονικών διαστημάτων.

5. Σήματα διακοπής στους μικροελεγκτές PIC16F

5.1 Σήματα Διακοπής (Interrupts)

Τα σήματα διακοπής αποτελούν γενικά τον πιο αποδοτικό τρόπο χρήσης της κεντρικής μονάδας επεξεργασίας (CPU), ειδικά όταν αυτή πρόκειται να εξυπηρετήσει περισσότερες από μία ανάγκες. Ας υποθέσουμε ότι ο μικροελεγκτής πρέπει να εκτελέσει μία συγκεκριμένη ακολουθία εντολών, όταν κάποιος ακροδέκτης εισόδου δεχτεί λογικό 1. Ένας τρόπος είναι η CPU να ελέγχει διαρκώς τον συγκεκριμένο ακροδέκτη, κάτι που την υποχρεώνει να αφιερώνει χρήσιμο χρόνο σε μια ελάχιστα παραγωγική εργασία. Ο άλλος τρόπος είναι το λογικό 1 να απευθύνεται σε κάποιον από τους ακροδέκτες που ο μικροελεγκτής χρησιμοποιεί για να δέχεται σήματα διακοπών. Μέχρι να δεχτεί κάποια διακοπή, η CPU μπορεί να εκτελεί τη βασική αλληλουχία εντολών του κυρίως προγράμματος, για παράδειγμα να παίρνει μετρήσεις κάποιου φυσικού μεγέθους. Όταν δεχτεί ένα σήμα διακοπής, η CPU σταματά την εκτέλεση του προγράμματος, αποθηκεύει τη διεύθυνση της επόμενης εντολής στη *στοίβα*, πηγαίνει στην εντολή 4 του κυρίως προγράμματος και συνεχίζει την εκτέλεση από εκεί. Συνήθως, στη θέση 4 βρίσκεται μια εντολή GOTO που στέλνει το πρόγραμμα στην υπορουτίνα που εξυπηρετεί τις ανάγκες της διακοπής. Για παράδειγμα, η υπορουτίνα αυτή μπορεί να θέσει για λίγο σε λειτουργία κάποιον βηματικό κινητήρα, πριν επιστρέψει ο έλεγχος στο κυρίως πρόγραμμα και το σύστημα συνεχίσει να εκτελεί μετρήσεις.



Σχ. 5.1 Ο ελεγκτής διακοπών ενός μικροελεγκτή

Στο σχ. 5.1 φαίνεται η σχέση του ελεγκτή διακοπών ενός μικροελεγκτή με την CPU και του εξωτερικούς ακροδέκτες του κυκλώματος.

Ο μικροελεγκτής PIC16F877 μπορεί να δεχτεί σήματα διακοπών κυρίως από τις εξής πηγές.

- Εξωτερική διακοπή από τον ακροδέκτη RB0/INT.
- Υπερχείλιση του απαριθμητή-χρονιστή **Timer0**.

- Κάποια αλλαγή της κατάστασης των ακροδεκτών RB7-RB4 της θύρας **B**.

5.2 Ο καταχωρητής ελέγχου διακοπών (INTCON)

Ο καταχωρητής που ρυθμίζει την ενεργοποίηση των διακοπών και καταγράφει ποιες διακοπές σημειώθηκαν είναι ο καταχωρητής **INTCON** (*Interrupt Control*). Η ονομασία και η σειρά των bits αυτού του καταχωρητή παρουσιάζεται στον Πίνακα 4-1.

Πίνακας 5-1
Τα bits του καταχωρητή ελέγχου διακοπών INTCON

b7	b6	b5	b4	b3	b2	b1	b0
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF

Το σημαντικότερο bit του καταχωρητή **INTCON**, το bit b7, είναι το bit της γενικής ενεργοποίησης των διακοπών (*General Interrupt Enable*, *GIE*) και όταν είναι μηδέν δεν επιτρέπει να συμβεί καμία διακοπή. Τα υπόλοιπα bits του καταχωρητή διακρίνονται σε δύο κατηγορίες, αυτά που ενεργοποιούν τις επιμέρους διακοπές (IE) και αυτά που δηλώνουν ότι σημειώθηκε κάποια διακοπή (*σημαίες διακοπών*, IF). Καθένας από τους βασικούς τύπους διακοπών, που αναφέραμε προηγουμένως έχει το δικό του ζευγάρι IE και IF. Έτσι, για να ενεργοποιηθεί η δυνατότητα να εμφανιστεί διακοπή εξαιτίας της υπερχείλισης του καταχωρητή **TMR0**, του χρονιστή **Timer0,** πρέπει πρώτα να τεθεί σε λογικό 1 το bit GIE και κατόπι να τεθεί σε λογικό 1 το bit 5 (T0IE). Μόλις εμφανιστεί η υπερχείλιση, η αντίστοιχη σημαία διακοπής T0IF τίθεται σε λογικό 1 και ακολουθεί η διαδικασία που περιγράψαμε: η τιμή του απαριθμητή προγράμματος, που περιέχει τη διεύθυνση της επόμενης εντολής αποθηκεύεται στη *στοίβα* (*stack*) και το πρόγραμμα μεταβαίνει στη διεύθυνση 0x004 της μνήμης του προγράμματος, προκειμένου να εξυπηρετηθεί η διακοπή.

Το bit που ενεργοποιεί τις διακοπές που προέρχονται από τα τέσσερα ανώτερα bits της θύρας **B** είναι το bit RBIE και η αντίστοιχη σημαία είναι το bit RBIF. Παρομοίως, οι διακοπές που προέρχονται από σήματα στον ακροδέκτη RB0/INT ενεργοποιούνται και σημειώνονται από το bit 4 (INTE) και το bit 1 (INTF) αντίστοιχα, του καταχωρητή **INTCON**. Το είδος του μετώπου (θετικό ή αρνητικό) που προκαλεί τη διακοπή στον ακροδέκτη RB0/INT ορίζεται από το bit INTEDG του καταχωρητή **OPTION_REG**.

Όταν καλείται η υπορουτίνα εξυπηρέτησης της διακοπής, απενεργοποιούνται οι επιπλέον διακοπές και έτσι το σύστημα δεν μπορεί να δεχτεί άλλη διακοπή μέχρι να επιστρέψει από αυτήν που ήδη άρχισε να εξυπηρετεί. Στο τέλος της υπορουτίνας της διακοπής ο χρήστης οφείλει να μηδενίσει εκ νέου τη σημαία της διακοπής και να ενεργοποιήσει ξανά τις διακοπές, τοποθετώντας σε λογικό 1 το bit GIE. Αυτό επιτυγχάνεται με χρήση της εντολής **RETFIE** στο τέλος της υπορουτίνας της διακοπής. Η εντολή αυτή είναι μια παραλλαγή της εντολής **RETURN**, και με αυτήν αφενός επιτυγχάνεται η επιστροφή στο κυρίως πρόγραμμα, στη διεύθυνση της εντολής που

αποθηκεύτηκε στη στοίβα, και αφετέρου ενεργοποιούνται ξανά οι διακοπές, με τοποθέτηση του GIE σε λογικό 1.

5.3 Κώδικας επίδειξης σημάτων διακοπής

Το επόμενο πρόγραμμα επιδεικνύει τη λειτουργία των σημάτων διακοπής στους μικροελεγκτές PIC. Το κυρίως πρόγραμμα βρίσκεται μετά την ετικέτα Start. Αυτό, κάνει απλά τις κατάλληλες αρχικοποιήσεις στις τιμές των καταχωρητών και στη συνέχεια μπαίνει σε έναν βρόγχο αναμονής (loop goto loop).

Η υπορουτίνα διακοπής ακολουθεί αμέσως μετά την δήλωση Org 4. Τοποθετείται στη μνήμη προγράμματος ξεκινώντας από την θέση μνήμης 04. Η λειτουργία της είναι να αυξάνει έναν μετρητή κατά ένα, κάθε φορά που συμβαίνει εξωτερική διακοπή. Το αποτέλεσμα της απαρίθμησης των διακοπών εμφανίζεται στη θύρα C.

Η πηγή των σημάτων διακοπής ορίζεται να είναι ο ακροδέκτης RB0 της θύρας B του μικροελεγκτή. Η διακοπή αυτή ενεργοποιείται με τη βοήθεια του bit INTE του καταχωρητή INTCON.

```
#include "P16F877.inc"

TEMP equ 20h
        Org 0
        goto start              ;Πήγαινε στο κυρίως πρόγραμμα
        Org 4                   ;Τοποθέτησε την επόμενη εντολή στη θέση 4 του προγράμματος

        incf TEMP, F            ;Αύξησε τον καταχωρητή TEMP κατά 1
        movf TEMP,W
        movwf PORTC             ;Μετέφερε τον TEMP στην θύρα C
        bcf INTCON, INTF        ;Μηδένισε την σημαία της διακοπής
        retfie                  ;Επέστρεψε από την διακοπή
start
        clrf TEMP               ;Μηδένισε τον TEMP
        bsf STATUS, RP0         ;Πήγαινε στην σελίδα μνήμης 1
        movlw b'00000000'
        movwf TRISC             ;Κάνε την θύρα C έξοδο
        movlw b'00000001'
        movwf TRISB             ;Κάνε τον ακροδέκτη RB0 (INT) είσοδο
        bcf STATUS, RP0         ;Επέστρεψε στη σελίδα μνήμης 0
        bsf INTCON, INTE        ;Ενεργοποίησε την διακοπή RB0
        bcf INTCON, INTF        ;Μηδένισε την σημαία της διακοπής RB0
        bsf INTCON, GIE         ;Ενεργοποίησε τις διακοπές
loop    goto loop               ;Περίμενε για διακοπή
        END
```

Κώδικας 5.1 Κώδικας που κάνει χρήση σήματος διακοπής από τον ακροδέκτη INT

5.4 Μετάβαση σε υπορουτίνα διακοπής – Context saving

Ένα τελευταίο σημείο προσοχής, που πρέπει να λαμβάνεται υπόψη όταν συμβαίνει μία διακοπή είναι ότι κατά τη διάρκεια της διακοπής πιθανότατα θα μεταβληθούν οι τιμές σημαντικών καταχωρητών, γεγονός που μπορεί να δημιουργήσει προβλήματα μετά τη επιστροφή στο κυρίως πρόγραμμα. Για το λόγο αυτό είναι σκόπιμο οι τιμές των σημαντικών καταχωρητών να αποθηκεύονται κατά την εκκίνηση της υπορουτίνας της διακοπής, ώστε να ανακτώνται μετά την εκτέλεση της υπορουτίνας. Οι σημαντικότεροι καταχωρητές που είναι σκόπιμο να αποθηκεύονται και να ανακτώνται είναι ο καταχωρητής εργασίας **W** και ο καταχωρητής κατάστασης **STATUS**.
Παρακάτω, παρουσιάζεται ο κώδικας για την αποθήκευση των βασικών καταχωρητών, καθώς και ο κώδικας για την ανάκτησή τους, κατά το πέρας της υπορουτίνας διακοπής.

```
;Στο κυρίως πρόγραμμα:
SaveWReg equ 0C      ;Ορίζουμε έναν καταχωρητή με όνομα SaveWReg
SaveStatus equ 0D    ;Ορίζουμε έναν καταχωρητή με όνομα SaveStatus

;Στην αρχή της υπορουτίνας διακοπής γράφουμε τις παρακάτω εντολές αποθήκευσης
movwf SaveWReg
swapf STATUS, w      ;Ανταλλαγή των τετράδων του καταχωρητή STATUS
movwf SaveStatus
```

Κώδικας 5.2 Αποθήκευση του περιεχομένου των βασικών καταχωρητών κατά τη μετάβαση σε υπορουτίνα εξυπηρέτησης διακοπής

Η εντολή **SWAPF**, που χρησιμοποιείται στο σημείο αυτό του προγράμματος, ανταλλάσσει τα τέσσερα κατώτερα bits (b0, b1, b2 και b3) με τα τέσσερα ανώτερα bits (b4, b5, b6 και b7). Ο λόγος που χρησιμοποιούμε αυτήν την εντολή για τη μεταφορά δεδομένων είναι ότι ειδικά αυτή η εντολή αφήνει αμετάβλητο τον καταχωρητή κατάστασης. Έτσι είμαστε βέβαιοι ότι δεν θα μεταβληθεί το περιεχόμενο του καταχωρητή **STATUS** κατά τη διαδικασία αποθήκευσης και ανάκτησης των σημαντικών καταχωρητών.

```
swapf SaveStatus, W  ;Ανταλλαγή τετράδων του καταχωρητή SaveStatus
                     ;και μεταφορά στον W
movwf STATUS         ;Αντιγραφή των περιεχομένων του καταχωρητή W στον
                     ;καταχωρητή STATUS
swapf SaveWReg       ;Ανταλλαγή των τετράδων του καταχωρητή SaveWReg
swapf SaveWReg, w    ;Επαναφορά στον W της αρχικής του τιμής
```

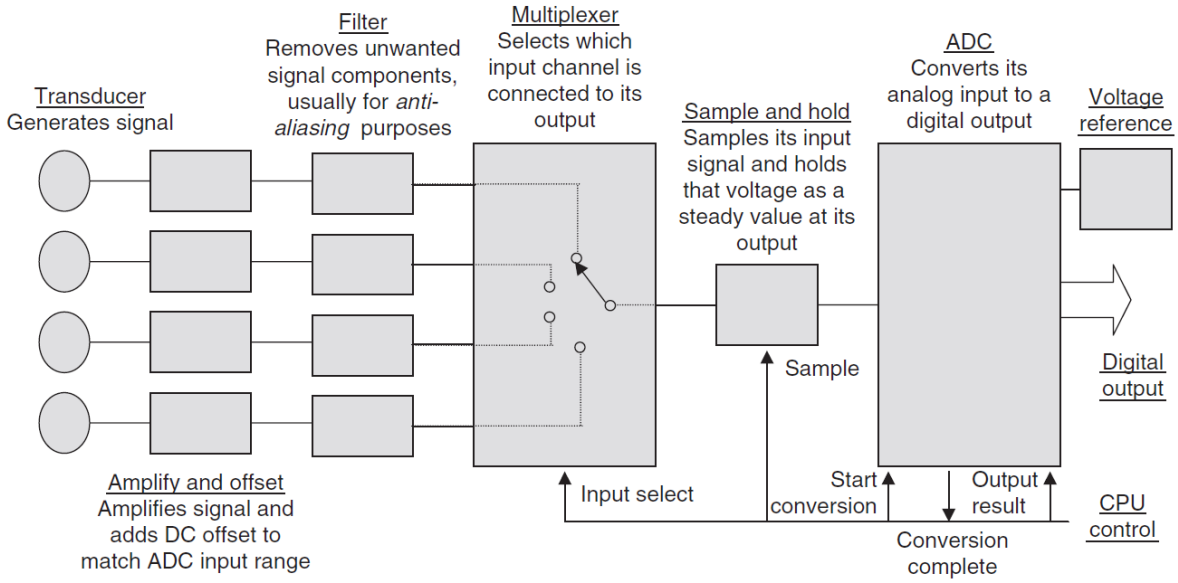
Κώδικας 5.3 Ανάκτηση του περιεχομένου των βασικών καταχωρητών κατά την επιστροφή από υπορουτίνα εξυπηρέτησης διακοπής

6. Μετατροπή αναλογικού σήματος σε ψηφιακό στους μικροελεγκτές PIC16F

6.1 Εισαγωγή

Στο σχήμα 6.1 φαίνεται ένα τυπικό σύστημα συλλογής δεδομένων (data acquisition). Αποτελείται από τους αισθητήρες, που μετατρέπουν τη φυσική ποσότητα που μετράται σε αναλογικό ηλεκτρικό σήμα, τα κυκλώματα ρύθμισης (signal conditioning), που ενισχύουν και φιλτράρουν το σήμα, και τέλος το σύστημα μετατροπής του αναλογικού σήματος σε ψηφιακό (ADC – analog to digital converter).

Το τελευταίο αυτό μέρος περιλαμβάνει συνήθως 1. έναν πολυπλέκτη, που επιλέγει το σήμα που θα υποστεί τη μετατροπή, 2. ένα κύκλωμα δειγματοληψίας και συγκράτησης (sample and hold) το οποίο λαμβάνει ένα στιγμιαίο δείγμα της αναλογικής τάσης και με τη βοήθεια ενός πυκνωτή συγκρατεί την τιμή του όσο χρειάζεται προκειμένου να γίνει η μετατροπή του δείγματος σε δυαδικό κώδικα και τέλος 3. ένα κύκλωμα που παράγει ένα δυαδικό κώδικα με *n* bits, που η δεκαδική στάθμη του είναι ανάλογη με την αρχική αναλογική τιμή. Δηλαδή, όσο μεγαλύτερη η αναλογική τάση του σήματος εισόδου, τόσο υψηλότερη είναι η κβαντική στάθμη του δυαδικού κώδικα που παράγεται στην έξοδο. Προφανώς, ένας ADC με *n* bits υποστηρίζει στάθμες από 0 μέχρι 2^n-1.



Σχ. 6.1 Ένα τυπικό σύστημα συλλογής δεδομένων

Ένας ADC λαμβάνει μια τάση αναφοράς (Voltage reference - V_{ref}) η οποία ρυθμίζει ποια αναλογική τάση εισόδου θα αντιστοιχίζεται στην πρώτη, τη δεύτερη, μέχρι και τη μέγιστη ψηφιακή κβαντική στάθμη. Για παράδειγμα, ένας μετατροπέας 8-bit, θα υποστηρίζει συνολικά 256 κβαντικές στάθμες. Όταν η στάθμη αναφοράς V_{ref} είναι 5V, τότε η τάση αυτή μοιράζεται σε όλες τις διαθέσιμες κβαντικές στάθμες, οπότε από τη μια στάθμη στην επόμενη αντιστοιχεί ένα μικρό βήμα αναλογικής τάσης V_{step}, ως εξής:

$$V_{step} = \frac{V_{ref}}{256} = \frac{5V}{256} = 0{,}0195V \qquad (6.1)$$

Αν η αναλογική τάση της εισόδου είναι V_{an}, τότε η κβαντική στάθμη που αντιστοιχεί είναι

$$\kappa\beta\alpha\nu\tau\iota\kappa\acute{\eta}\ \sigma\tau\acute{\alpha}\theta\mu\eta = \frac{V_{an}}{V_{step}} \qquad (6.2)$$

Έτσι, ο μετατροπέας αντιστοιχίζει κάθε αναλογική τάση της εισόδου του σε κβαντικές στάθμες, σύμφωνα με τον Πίνακα 6-1.

Πίνακας 6-1

Αντιστοιχία αναλογικής τάσης και δυαδικής τιμής σε ADC 8-bit

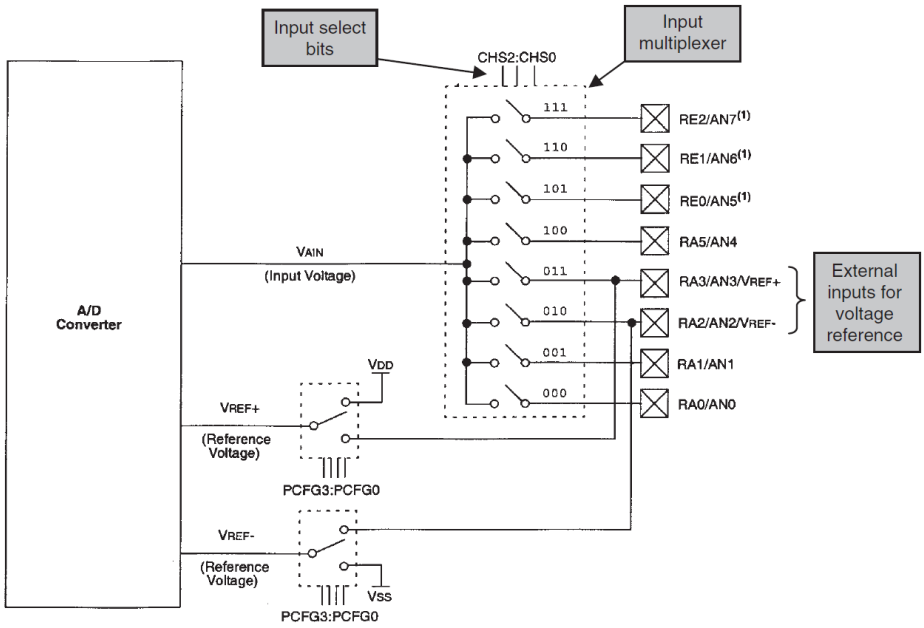
Αναλογική είσοδος (V_{an})	**Κβαντική στάθμη εξόδου**	**Δυαδική τιμή εξόδου**
0 V	0	00000000
0,9945 V	51	00110011
2,496 V	128	10000000
3,49 V	179	10110011
4,98 V	255	11111111

6.2 Κύκλωμα ADC στον μικροελεγκτή PIC16F877

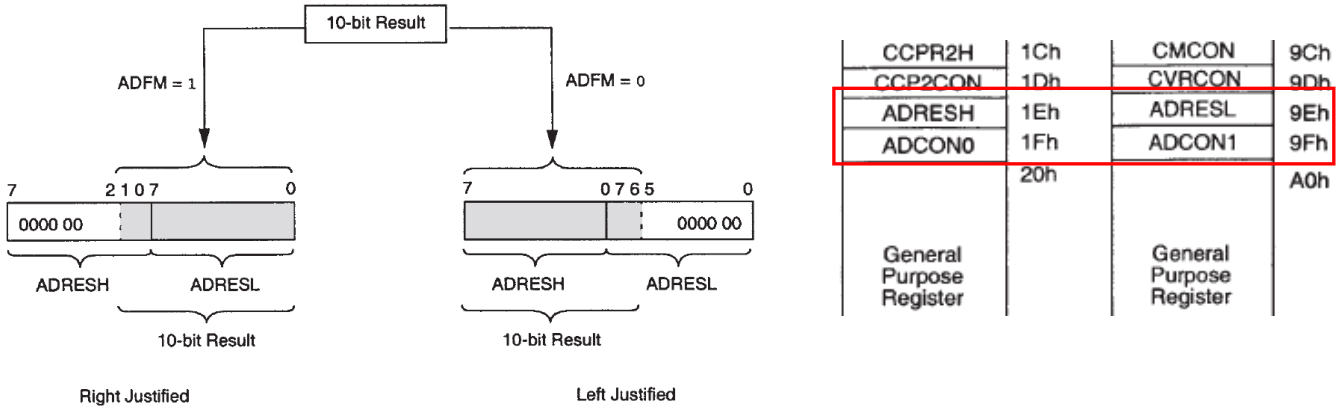
Το σχήμα 6.2 παρουσιάζει το κύκλωμα μετατροπής αναλογικού σήματος σε ψηφιακό στους μικροελεγκτές της Microchip. Ένας αριθμός ακροδεκτών εισόδου προορίζεται να λάβει είσοδο αναλογικού σήματος. Για να λειτουργήσουν αυτοί οι ακροδέκτες ως αναλογικές είσοδοι θα πρέπει να γίνει η κατάλληλη επιλογή, μέσω ενός καταχωρητή που ρυθμίζει τη λειτουργία του ADC. Στην είσοδο υπάρχει το κύκλωμα πολυπλεξίας που επιλέγει ένα αναλογικό σήμα εισόδου. Η επιλογή του καναλιού εισόδου που μετατρέπεται κάθε φορά, γίνεται πάλι με τη βοήθεια ειδικού καταχωρητή (input select

bits). Η τάση αναφοράς τίθεται αυτόματα στα 5V που είναι η τάση τροφοδοσίας, αλλά μπορεί και να ρυθμιστεί.

Ο μετατροπέας του ADC του PIC16F877 έχει ανάλυση 10 bit. Αυτό σημαίνει ότι υποστηρίζει 1024 κβαντικές στάθμες. Όταν ολοκληρώνεται μια μετατροπή, το αποτέλεσμα τοποθετείται σε δύο καταχωρητές, τους ADRESH και ADRESL. Το αποτέλεσμα μπορεί να τοποθετηθεί με δεξιά ή αριστερή στοίχιση, όπως φαίνεται στο σχήμα 6.3, ανάλογα με τις ρυθμίσεις που θα γίνουν στον καταχωρητή ADCON1.



Σχ. 6.2 Κύκλωμα ADC στους μικροελεγκτές PIC.



Σχ. 6.3 Καταχώρηση αποτελέσματος μετατροπής στους καταχωρητές ADRESH και ADRESL, με δεξιά και αριστερή στοίχιση. Δεξιά στο σχήμα, φαίνονται οι βασικοί καταχωρητές που σχετίζονται με τον ADC.

6.3 Οι βασικοί καταχωρητές του μετατροπέα ADC

Όπως όλα τα περιφερειακά του PIC, έτσι και ο ADC ρυθμίζεται ώστε να επιτελεί τις επιθυμητές λειτουργίες, με τη βοήθεια ορισμένων καταχωρητών ειδικού σκοπού. Οι βασικοί καταχωρητές είναι ο ADCON0 και ADCON1. Όπως αναφέρθηκε, οι καταχωρητές ADRESL και ADRESH αποθηκεύουν το αποτέλεσμα της μετατροπής. Με αριστερή στοίχιση, ο ADRESH αποθηκεύει τα 8 πιο σημαντικά bits της μετατροπής, άρα μπορεί να χρησιμοποιηθεί αν επιθυμούμε να απλουστεύσουμε τον μετατροπέα και να τον χρησιμοποιήσουμε σαν να είχε ανάλυση 8-bits.
Όλες τις λεπτομέρειες για τις λειτουργίες που ρυθμίζουν τα διάφορα bits των καταχωρητών, ο ενδιαφερόμενος χρήστης μπορεί να τις βρει στα εγχειρίδια χρήσης του μικροελεγκτή που δημοσιεύονται διαδικτυακά από την εταιρία Microchip. Οι απαραίτητες ρυθμίσεις περιγράφονται συνοπτικά παρακάτω.

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

bit 7-6 **ADCS1:ADCS0:** A/D Conversion Clock Select bits (ADCON0 bits in **bold**)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	Fosc/2
0	01	Fosc/8
0	10	Fosc/32
0	11	FRC (clock derived from the internal A/D RC oscillator)
1	00	Fosc/4
1	01	Fosc/16
1	10	Fosc/64
1	11	FRC (clock derived from the internal A/D RC oscillator)

bit 5-3 **CHS2:CHS0:** Analog Channel Select bits
000 = Channel 0 (AN0)
001 = Channel 1 (AN1)
010 = Channel 2 (AN2)
011 = Channel 3 (AN3)
100 = Channel 4 (AN4)
101 = Channel 5 (AN5)
110 = Channel 6 (AN6)
111 = Channel 7 (AN7)

bit 2 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit
When ADON = 1:
1 = A/D conversion in progress (setting this bit starts the A/D conversion which is automatically cleared by hardware when the A/D conversion is complete)
0 = A/D conversion not in progress

bit 1 **Unimplemented:** Read as '0'

bit 0 **ADON:** A/D On bit
1 = A/D converter module is powered up
0 = A/D converter module is shut-off and consumes no operating current

Σχ. 6.4 Ο καταχωρητής ADCON0 και οι λειτουργίες που επιτελούν τα bits που αυτός διαθέτει.

1. Επιλογή συχνότητας ρολογιού ADC: Ο μετατροπέας χρονίζεται με τη βοήθεια γεννήτριας παλμών, που λαμβάνεται με υποδιαίρεση της συχνότητας του εξωτερικού κρυστάλλου. Η υποδιαίρεση ρυθμίζεται με τα bits ADCS2, ADCS1, ADCS0, από τα οποία τα δύο λιγότερα σημαντικά ανήκουν στον καταχωρητή ADCON0, ενώ το πιο σημαντικό (ADCS2) ανήκει στον ADCON1 (βλέπε σχήματα 6.4 και 6.5). Μια κατάλληλη υποδιαίρεση για το ρολόϊ του μετατροπέα είναι Fosc/16, που αντιστοιχεί σε bits επιλογής 101.
2. Τα bits CHS2:CHS0 του ADCON0 επιλέγουν κανάλι εισόδου, μέσω του πολυπλέκτη εισόδου. Έτσι, για να γίνει η μετατροπή του καναλιού Ch0 (RA0) αυτά πρέπει να λάβουν την τιμή 000.
3. Ο καταχωρητής ADCON1 διαθέτει τέσσερα bits PCFG3:PCFG0 που ρυθμίζουν τη διαμόρφωση των εισόδων AN0 έως AN7, δηλαδή ποιες θα λειτουργούν ως αναλογικές και ποιες ως ψηφιακές. Επίσης, κάποιες μπορούν να διαμορφωθούν ώστε να δεχτούν είσοδο αναφοράς.
4. Το bit ADFM του καταχωρητή ADCON1 ρυθμίζει τη στοίχιση της λέξης 10-bit που παράγει ο ADC. Συνιστάται αριστερή στοίχιση (ADFM=0).

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7 **ADFM:** A/D Result Format Select bit
1 = Right justified. Six (6) Most Significant bits of ADRESH are read as '0'.
0 = Left justified. Six (6) Least Significant bits of ADRESL are read as '0'.

bit 6 **ADCS2:** A/D Conversion Clock Select bit (ADCON1 bits in shaded area and in **bold**)

bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C/R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2/1
011x	D	D	D	D	D	D	D	D	—	—	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1/2

A = Analog input D = Digital I/O

Σχ. 6.5 Ο καταχωρητής ADCON1 και οι λειτουργίες που επιτελούν τα bits που αυτός διαθέτει.

Τέλος, για να λειτουργήσει ο ADC πρέπει να θέσουμε το bit ADON του ADCON0 σε λογικό 1 (power up). Για να γίνει η δειγματοληψία μιας νέας αναλογικής τάσης, η σημαία ADIF πρέπει να μηδενιστεί. Είναι απαραίτητο να προβλέψουμε μια μικρή περίοδο αδράνειας, ώστε να σταθεροποιηθεί η τάση στην είσοδο του κυκλώματος δειγματοληψίας και συγκράτησης. Αυτό επιτυγχάνεται με δύο ή τρεις εντολές nop.
Για να ξεκινήσει μια νέα μετατροπή θέτουμε σε λογικό 1 το bit GO του ADCON0.
Όταν ολοκληρωθεί η μετατροπή η σημαία ADIF γίνεται 1. Η σημαία αυτή βρίσκεται στον καταχωρητή PIR1, που ρυθμίζει τα σήματα διακοπής περιφερειακών συσκευών. Όταν η σημαία ADIF γίνει 1, το αποτέλεσμα βρίσκεται στους καταχωρητές ADRESH, ADRESL, και μπορεί να διαβαστεί. Παρακάτω δίνεται ένα πλήρες παράδειγμα κώδικα για τη μετατροπή ADC.

```
        #include “p16f877.inc”
        Org 00

        ;initialize ADC
        bsf STATUS, RP0
        movlw b'00011111'    ;5 first bits of PORTA inputs
        movwf TRISA
        movlw b'00000000'
        movwf TRISB
        movlw b'01000010'    ;left justified, ADCS2=1, 3 Dig 5 Analog ch
        movwf ADCON1
        bcf STATUS, RP0
        movlw b'01000001'    ;101 Fosc/16, ch0, ADON
        movwf ADCON0
        clrf PORTB

        ;conversion
loop1   bcf PIR1, ADIF
        nop                  ;wait for the output of S&H circuit to stabilize
        nop
        nop
        bsf ADCON0, GO
wait    btfss PIR1, ADIF
        goto wait

        ;read ADC
        movf ADRESH, w
        movwf PORTB
        goto loop1
        end
```

Κώδικας 6.1 Μετατροπή αναλογικού σήματος σε ψηφιακό (χρήση μόνον 8-bit)

7. Ασύγχρονη σειριακή μετάδοση –το κύκλωμα UART

7.1 Σύγχρονη και ασύγχρονη σειριακή μετάδοση

Συχνά μια εφαρμογή πραγματικού χρόνου χρειάζεται να επικοινωνήσει με άλλα μικροϋπολογιστικά συστήματα ή συσκευές, ειδικά σε κατανεμημένα συστήματα, όπου η πληροφορία παράγεται τοπικά, μεταδίδεται μέσω καναλιών και συλλέγεται ή υφίσταται επεξεργασία κεντρικά. Η μετάδοση της πληροφορίας μπορεί να γίνει παράλληλα, με ταυτόχρονη μετάδοση όλων των bits κάθε ψηφιολέξης ή σειριακά. Στη σειριακή μετάδοση η πληροφορία μεταδίδεται ως μια ακολουθία από bits από τον πομπό προς το δέκτη. Γενικά διακρίνουμε τη σύγχρονη και την ασύγχρονη σειριακή μετάδοση.

Στη *σύγχρονη* σειριακή επικοινωνία πομπός και δέκτης πρέπει να είναι συγχρονισμένοι για την εκπομπή και λήψη των δεδομένων. Ταυτόχρονα με την εκπομπή των δεδομένων εκπέμπεται και το clock, για τον συγχρονισμό της επικοινωνίας.

Στην *ασύγχρονη* μετάδοση, τα δεδομένα μπορεί να εμφανιστούν στη γραμμή οποιαδήποτε χρονική στιγμή, χωρίς κανένα συγχρονισμό με ωρολογιακά σήματα. Ο μόνος απαραίτητος χρονισμός υλοποιείται με το start bit, ενώ ο συγχρονισμός πομπού και δέκτη επιτυγχάνεται διατηρώντας σταθερό το ρυθμό εκπομπής και λήψης (baud rate).

Στο κεφάλαιο αυτό θα μελετήσουμε τη χρήση της σειριακής θύρας ως μονάδας ασύγχρονης σειριακής επικοινωνίας στον μικροελεγκτή PIC16F877.

7.2 Ασύγχρονη Σειριακή Επικοινωνία

Η ασύγχρονη σειριακή επικοινωνία εξυπηρετεί τη μετάδοση χαρακτήρων που εκπέμπονται από κάποιο πομπό χωρίς κανένα συγχρονισμό, όπως συμβαίνει με τους αλφαριθμητικούς χαρακτήρες που δημιουργούνται όταν πιέζουμε τα πλήκτρα ενός πληκτρολογίου.

Κάθε τέτοιος χαρακτήρας μετατρέπεται σε ψηφιολέξη μέσω του κώδικα ASCII και η ακολουθία bits που αντιστοιχεί σε αυτόν εμφανίζεται στη γραμμή μετάδοσης. Ο δέκτης πρέπει να είναι σε θέση να αναγνωρίζει ότι έφτασε ένας χαρακτήρας και να δέχεται τα bits του χαρακτήρα με τη σωστή σειρά και χωρίς απώλειες.

Για τον παραπάνω σκοπό, τα bits της ασύγχρονης σειριακής μετάδοσης οργανώνονται σε ομάδες των εννέα έως δώδεκα bits συνολικά, οι οποίες περιέχουν κάποιους χαρακτήρες έναρξης και λήξης (Σχήμα 7.1). Το πρώτο bit κάθε πλαισίου είναι το λεγόμενο START BIT, το οποίο αντιστοιχεί σε λογικό μηδέν. Ακολουθεί η σειρά των ψηφίων του χαρακτήρα που αποστέλλεται. Για παράδειγμα, εάν αποστέλλεται το κεφαλαίο γράμμα Α, τότε η ακολουθία των ψηφίων θα είναι 01001011. Μετά από τα bits του χαρακτήρα ακολουθεί ένα bit άρτιας ή περιττής *ισοτιμίας* (*parity*), το οποίο ενεργοποιεί μία διαδικασία ελέγχου σφαλμάτων, για να ανιχνεύσει τυχόν λάθη που

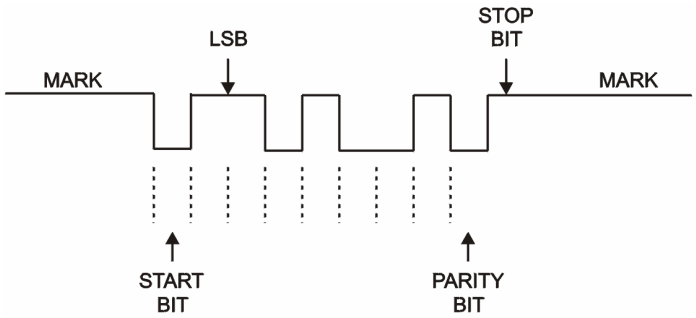
συνέβησαν κατά τη μετάδοση. Το πλαίσιο κλείνει με ένα ή δύο STOP BITS, που υποδηλώνουν το τέλος του χαρακτήρα και την κατάσταση αναμονής για τον επόμενο. Η λογική κατάσταση των ψηφίων λήξης (STOP BITS) είναι το λογικό ένα.

Όταν δεν μεταδίδεται κάποιος χαρακτήρας, τότε λέμε ότι η σύνδεση είναι *ανενεργή*, οπότε η γραμμή ευρίσκεται σε λογικό ένα. Η κατάσταση αυτή ονομάζεται «*συνθήκη MARK*». Όταν μεταδοθεί το bit έναρξης (START BIT) και φθάσει στο δέκτη, ο δέκτης καταλαβαίνει ότι ακολουθούν τα bits του χαρακτήρα που αποστέλλεται, οπότε ενεργοποιεί το σύστημα χρονισμού του και διαβάζει με τη σειρά τα επόμενα bits μέχρι τα bits λήξης (STOP BITS). Στη συνέχεια τίθεται και πάλι στην κατάσταση αναμονής.

Είναι φανερό ότι η διάρκεια του κάθε εκπεμπόμενου bit στην ασύγχρονη σειριακή μετάδοση πρέπει να είναι αυστηρά η ίδια, ώστε να μπορεί ο δέκτης, με βάση κάποιο σύστημα χρονισμού, να διακρίνει τα bits μεταξύ τους. Συνεπώς ο πομπός και ο δέκτης πρέπει να συμφωνούν ως προς την ταχύτητα της σειριακής μετάδοσης των bits. Η ταχύτητα αυτή ορίζει τον λεγόμενο «*ρυθμό μετάδοσης*» (*baud rate*), που μετριέται σε *bits ανά δευτερόλεπτο* (*bits/sec* ή *bps*). Συνήθεις ρυθμοί στις ασύγχρονες σειριακές επικοινωνίες είναι 2400, 4800, 9600, 14400, 19200, 28800 και 33600 bits/sec. Η μέγιστη ταχύτητα που υποστηρίζει μια θύρα UART κατά την αποστολή χαρακτήρων από έναν υπολογιστή προς μία συσκευή επικοινωνίας είναι 115.2 kbps. Τυπική ασύγχρονη σειριακή συσκευή είναι το modem, που διασυνδέει τον υπολογιστή ή κάποιο τερματικό με την τηλεφωνική γραμμή.

Σαν παράδειγμα αναφέρουμε ότι, για να έχουμε ρυθμό μετάδοσης 9600 bps, η διάρκεια του κάθε bit πρέπει να είναι 104 μs. Κάθε χαρακτήρας θα διαρκεί στη γραμμή 1.14 ms. *Ας σημειωθεί ότι η ακρίβεια στη διάρκεια του κάθε bit είναι σημαντική, ώστε να υπάρχει συγχρονισμός πομπού και δέκτη*.

Το βασικό μειονέκτημα της ασύγχρονης σειριακής μετάδοσης είναι η ανάγκη που προκύπτει για START και STOP bits στην αρχή και στο τέλος κάθε χαρακτήρα. Με τον τρόπο αυτό επιβαρύνεται σημαντικά η διαδικασία της μετάδοσης με επιπλέον bits που δεν αντιπροσωπεύουν χρήσιμη πληροφορία.



Σχήμα 7.1 Μορφή του σήματος στην ασύγχρονη σειριακή μετάδοση χαρακτήρα

7.3 Το Πρωτόκολλο RS-232C

Το πρωτόκολλο RS-232C επιτρέπει την ασύγχρονη σειριακή επικοινωνία ανάμεσα σε δύο συσκευές. Εάν επιθυμούμε να συνδέσουμε περισσότερες από δύο συσκευές σε έναν υπολογιστή, χρειαζόμαστε περισσότερες από μία σειριακές θύρες. Υπάρχουν, βέβαια, και άλλα σειριακά πρωτόκολλα, όπως το πρωτόκολλο σύγχρονης επικοινωνίας I^2C, που επιτρέπουν τη διασύνδεση πολλών συσκευών σε ένα σειριακό κύκλωμα.

Το πρωτόκολλο RS-232C χρησιμοποιεί αρνητική ψηφιακή λογική και μεγάλες στάθμες, ώστε να επιτρέπει τη διάδοση του σήματος σε μεγάλες αποστάσεις χωρίς απώλειες. Αυτά έχουν σαν αποτέλεσμα οι τάσεις του πρωτοκόλλου RS-232C να μην είναι συμβατές με τις στάθμες TTL. Τα επίπεδα τάσεων του πρωτοκόλλου RS-232C, σύμφωνα με τις προδιαγραφές που θέσπισε η Ένωση EIA, φαίνονται στον πίνακα 7.1.

Αν και σύμφωνα με το πρωτόκολλο ο μέγιστος ρυθμός μετάδοσης (baud rate) δεν ξεπερνά τα 19.2 kbps, οι σημερινές ταχύτητες μπορεί να είναι πολύ μεγαλύτερες.

Με χρήση του πρωτοκόλλου RS-232C, ένα τερματικό χαρακτήρων (ASCII terminal) μπορεί να αποστείλει μέσω μιας γραμμής επικοινωνίας δεδομένα, σύμφωνα με τους κανόνες της ασύγχρονης σειριακής μετάδοσης που περιγράψαμε στην παραπάνω παράγραφο.

Όταν η γραμμή είναι ανενεργή, ευρίσκεται σε συνθήκη MARK, δηλαδή −12 V περίπου, που αντιστοιχούν σε λογικό ένα. Η γραμμή ενεργοποιείται με τη συνθήκη SPACE (λογικό μηδέν ή +12V). Ακολουθεί η μετάδοση επτά ή οκτώ bits για τον αποστελλόμενο χαρακτήρα, ένα προαιρετικό bit άρτιας ή περιττής ισοτιμίας (parity) και ένα ή δύο STOP bits (συνθήκη MARK), που σηματοδοτούν το τέλος του χαρακτήρα (βλέπε Σχήμα 7.1).

Πίνακας 7.1 Προδιαγραφές τάσεων και ρευμάτων πρωτοκόλλου RS232

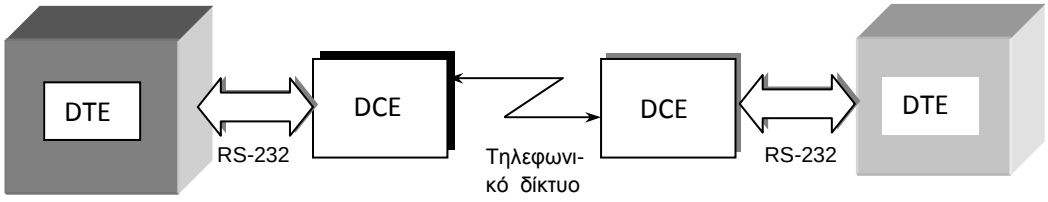
1. Το λογικό 0, που λέγεται και **SPACE**, βρίσκεται μεταξύ +3 και +25 V (στην πράξη λαμβάνονται και εκπέμπονται από +5 έως +15 V).
2. Το λογικό 1, που λέγεται και **MARK**, βρίσκεται μεταξύ −3 και −25V (πρακτικά από −5 έως −15V).
3. Η περιοχή από −3V έως +3V δεν αντιπροσωπεύει καθορισμένη λογική στάθμη.
4. Κανένας από τους ακροδέκτες της σειριακής θύρας δεν μπορεί να δεχτεί δυναμικό μεγαλύτερο από 25V σε σχέση με τη γη.
5. Το μέγιστο ρεύμα δεν μπορεί να ξεπερνά τα 500mA.

7.4 Τύποι Ασύγχρονων Σειριακών Συσκευών (DTE/DCE)

Το πρωτόκολλο RS-232 καθορίζει δύο τύπους συσκευών. Ο πρώτος τύπος συσκευής ονομάζεται *Τερματική Συσκευή Δεδομένων* (*Data Terminal Equipment* ή *DTE*). Ο δεύτερος τύπος συσκευής ονομάζεται *Συσκευή Επικοινωνίας Δεδομένων* (*Data Communications Equipment* ή *DCE*).

Κάθε προσωπικός υπολογιστής η μικροϋπολογιστής σε τσιπ που διαθέτει θύρα RS-232 είναι συσκευή DTE.

Για τη διασύνδεση δύο τερματικών σταθμών σε μεγάλες αποστάσεις μέσω σειριακής θύρας παρεμβάλλονται συσκευές επικοινωνίας, όπως εικονίζεται στο Σχήμα 7.2. Τυπικές τέτοιες συσκευές είναι τα modem, που αναφέρονται ως *συσκευές DCE*.

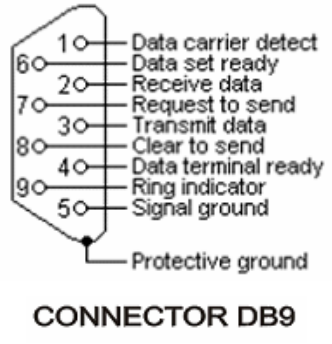


Σχήμα 7.2 Τυπικές διασυνδέσεις DTE/DCE και DCE/DCE

Εκτός από τα modem, στη σειριακή θύρα ενός υπολογιστή μπορεί να συνδεθούν κι άλλες συσκευές που δεν προορίζονται για επικοινωνία, αλλά για μετρήσεις, έλεγχο κυκλωμάτων κ.λπ. Πολλά πολύμετρα, πηγές, ελεγκτές (controllers), αισθητήρες, ακόμη και οικιακές ηλεκτρονικές συσκευές όπως video, δορυφορικοί δέκτες κλπ. έρχονται εξοπλισμένα με σειριακή θύρα RS-232. Οι συσκευές αυτές συνήθως είναι διαμορφωμένες σαν DCE και το καλώδιο που τις συνδέει με τον υπολογιστή καθώς και τα σήματα ελέγχου που ανταλλάσσουν μαζί του είναι ίδια με αυτά ενός modem.

7.5 Ακροδέκτες της σειριακής θύρας

Ανάμεσα στους ακροδέκτες της σειριακής θύρας διακρίνουμε τους *ακροδέκτες* ή *γραμμές δεδομένων* (*data lines*) και τους *ακροδέκτες* ή *γραμμές ελέγχου* (*control lines*). Οι σπουδαιότεροι ακροδέκτες είναι αυτοί που μεταφέρουν τα δεδομένα εκπομπής και λήψης, δηλαδή την πληροφορία προς την μία ή την άλλη κατεύθυνση. Όλοι οι υπόλοιποι ακροδέκτες είναι βοηθητικοί αλλά απαραίτητοι για την επικοινωνία ενός DTE με ένα DCE. Ο σύνδεσμος της σειριακής θύρας είναι εννέα ακροδεκτών, τύπου DB-9.



Σχ. 7.3 Σύνδεσμος DB-9 για τη σύνδεση της σειριακής θύρας

Οι γραμμές δεδομένων ονομάζονται TXD, RXD και SGND, και αντιστοιχούν στους ακροδέκτες 2 και 3, (Πίνακας 7-2). Η γραμμή TXD προορίζεται για την εκπομπή των σειριακών δεδομένων, η γραμμή RXD για τη λήψη των δεδομένων, ενώ η γραμμή SGND είναι ο αγωγός αναφοράς των τάσεων που διαδίδονται στις πρώτες δύο γραμμές. Το σχήμα 7.3 παρουσιάζει τη μορφή και τους ακροδέκτες του τυπικού συνδέσμου.

Οι σπουδαιότερες από τις γραμμές ελέγχου είναι οι RTS, CTS, DSR και DTR. Όταν ο υπολογιστής (DTE) θέλει να στείλει δεδομένα σε μια συσκευή DCE, ενεργοποιεί τη γραμμή RTS (*Request To Send*) θέτοντας στη γραμμή τη συνθήκη SPACE. Εάν η συσκευή DCE έχει χώρο για να δεχθεί δεδομένα, τότε απαντά ενεργοποιώντας τη γραμμή CTS (*Clear To Send*).

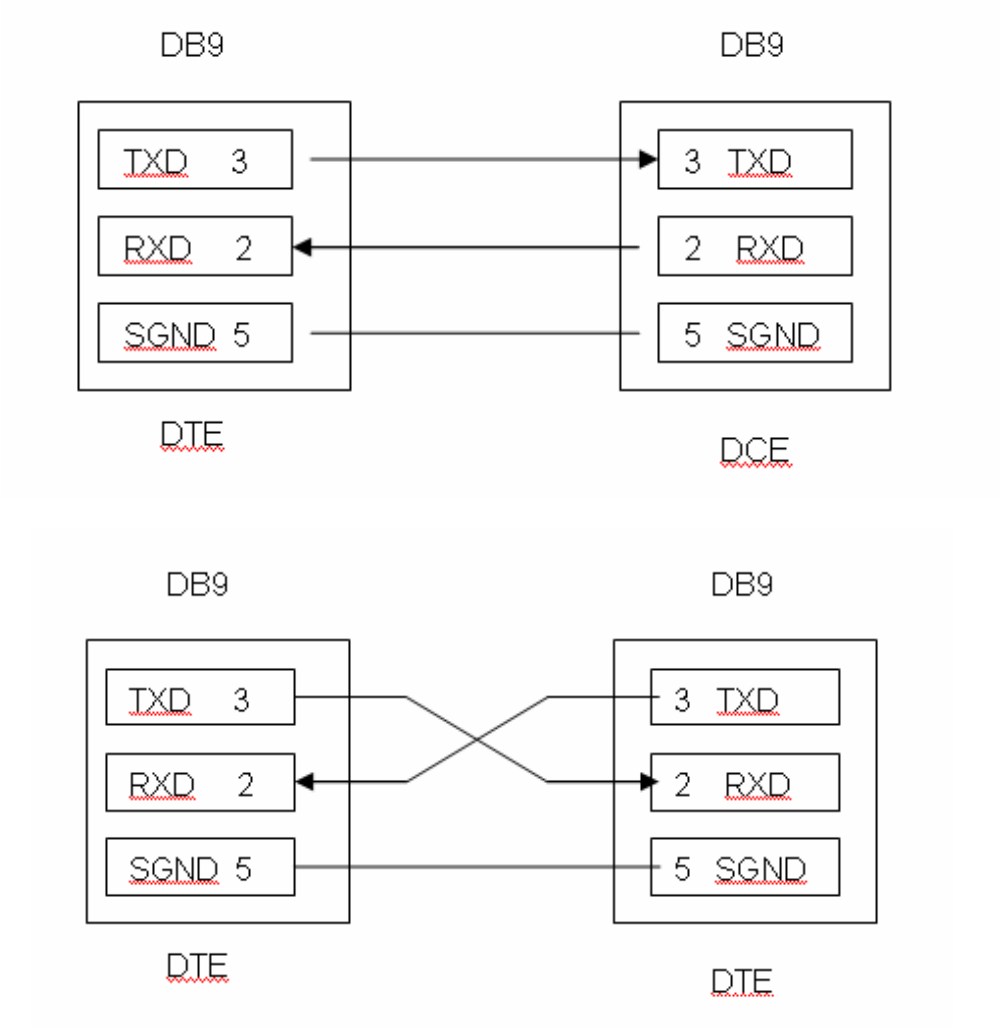
Πίνακας 7-2
Ονομασία και αντιστοιχία των ακροδεκτών της σειριακής θύρας

Ακροδέκτες σε σύνδεσμο D-9	Κωδική ονομασία	Όνομα
3	TXD	Transmit Data
2	RXD	Receive Data
5	SGND	Signal Ground
7	RTS	Request To Send
8	CTS	Clear To Send
6	DSR	Data Set Ready
4	DTR	Data Terminal Ready
1	CD	Carrier Detect
9	RI	Ring Indicator

Αντίστοιχα, όταν το modem θέλει να στείλει δεδομένα, πληροφορεί τη θύρα UART ενεργοποιώντας τη γραμμή DSR (*Data Set Ready*). Εάν ο υπολογιστής από την πλευρά του είναι έτοιμος να λάβει δεδομένα, απαντά θέτοντας την κατάσταση SPACE στον ακροδέκτη DTR (*Data Terminal Ready*). Εάν, πάλι, δεν είναι έτοιμος να λάβει δεδομένα, τότε θέτει συνθήκη MARK στη γραμμή DTR.

Από τους παραπάνω ακροδέκτες, οι απαραίτητοι είναι αυτοί που αντιστοιχούν σε γραμμές επικοινωνίας (2,3,5). Οι γραμμές ελέγχου χρησιμοποιούνται προαιρετικά, μόνον αν επιθυμούμε να υλοποιήσουμε το αντίστοιχο πρωτόκολλο.

Τυπική σύνδεση ανάμεσα σε τερματική συσκευή DTE και σε συσκευή επικοινωνίας DCE φαίνεται στο σχήμα 7.4. Στο ίδιο σχήμα φαίνεται και η σύνδεση ανάμεσα σε τερματικές συσκευές (DTE/DTE), για παράδειγμα ανάμεσα σε δύο προσωπικούς υπολογιστές ή ανάμεσα σε δύο μικροελεγκτές, που επικοινωνούν μέσω σειριακής θύρας.



Σχ. 7.4 Τυπικές συνδέσεις DTE/DCE και DTE/DTE

7.4 Μονάδα ασύγχρονης επικοινωνίας στον PIC16F877

Η Σύγχρονη-Ασύγχρονη μετάδοση (USART) είναι μία από τις δύο σειριακές μονάδες εισόδου – εξόδου που διαθέτει ο μικροελεγκτής PIC16F877. Αυτή μπορεί να διαμορφωθεί ως ένα ασύγχρονο σύστημα το οποίο μπορεί να επικοινωνεί με περιφερειακές συσκευές, όπως LCD displays και προσωπικούς υπολογιστές ή μπορεί να διαμορφωθεί ως ένα σύγχρονο σύστημα το οποίο μπορεί να επικοινωνεί με κυκλώματα όπως A/D ή D/A, μνήμες EEPROM και άλλα. Για τη σειριακή επικοινωνία ο PIC16F877 χρησιμοποιεί τις γραμμές TXD και RXD και SGND του πίνακα 7.2. Οι ακροδέκτες αυτοί είναι οι 25, 26 και 31 αντίστοιχα (βλέπε σχ. 2.5, κεφ. 2). Οι δύο πρώτοι αντιστοιχούν στους ακροδέκτες RC6 και RC7 της θύρας C.

Όλοι οι ακροδέκτες του κυκλώματος UART είναι συμβατοί με τα επίπεδα TTL. Άρα, ανάμεσα στο UART και τον D-τύπου συνδετήρα της σειριακής θύρας παρεμβάλλονται μετατροπείς στάθμης, ώστε τα σήματα του UART να γίνουν συμβατά με τα επίπεδα της λογικής του πρωτοκόλλου RS-232. Οι περισσότεροι Η/Υ χρησιμοποιούν για το σκοπό αυτό τα ολοκληρωμένα κυκλώματα DS1489 για τη λήψη και DS1488 για την εκπομπή σημάτων. Στην παράγραφο 7.9 θα γίνει αναφορά στον μετατροπέα στάθμης MAX232.

Κάθε κύκλωμα UART περιέχει ένα κύκλωμα χρονισμού, που ονομάζεται *γεννήτρια ρυθμού* (*baud rate generator*). Για τη λειτουργία του κυκλώματος χρονισμού απαιτείται ένας εξωτερικός κρύσταλλος, που στην περίπτωση του PIC είναι ο εξωτερικός κρύσταλλος χρονισμού. Η συχνότητα του κρυστάλλου μετατρέπεται εσωτερικά σε ρυθμό εκπομπής και λήψης (baud rate), με τη βοήθεια ενός προγραμματιζόμενου διαιρέτη συχνότητας (baud rate generator).

Ο τρόπος λειτουργίας του κυκλώματος UART προγραμματίζεται με τη βοήθεια ορισμένων καταχωρητών, τους οποίους ο υπολογιστής βλέπει σαν θέσεις μνήμης. Με τη βοήθεια των καταχωρητών αυτών μπορεί ο χρήστης να έχει πλήρη έλεγχο της διαδικασίας εισόδου/εξόδου μέσω της σειριακής θύρας.

Η Ασύγχρονη σειριακή θύρα αποτελείται από τα ακόλουθα σημαντικά κυκλώματα:

- Γεννήτορας ρυθμού μετάδοσης.
- Το κύκλωμα.
- Ασύγχρονος μεταδότης.
- Ασύγχρονος λήπτης.

7.5 Καταχωρητές της ασύγχρονης σειριακής θύρας

Για να επικοινωνήσει ο PIC μέσω της σειριακής θύρας, με περιφερειακές συσκευές ή υπολογιστές χρησιμοποιεί τους εξής καταχωρητές ειδικού σκοπού:

- **TXSTA**: Transmit Status and Control Register-Καταχωρητής κατάστασης και ελέγχου της αποστολής των δεδομένων.
- **RCSTA**: Receive Status and Control Register-Καταχωρητής κατάστασης και ελέγχου της λήψης των δεδομένων.

- **SPBRG**: Baud Rate Generation Register – Καταχωρητής παραγωγής του ρυθμού μετάδοσης baud rate.
- **TXREG**: Καταχωρητής αποστολής δεδομένων.
- **RCREG**: Καταχωρητής λήψης δεδομένων.

Οι πρώτοι τρεις καταχωρητές ρυθμίζουν την λειτουργία της μονάδας USART, ενώ τους άλλους δύο τους χρησιμοποιούμε ως καταχωρητές αποθήκευσης, για να μεταφέρουμε δεδομένα από και προς την USART.

7.6 Αρχικοποίηση των καταχωρητών της σειριακής θύρας

Για την ενεργοποίηση της σειριακής θύρας, το bit SPEN (RCSTA<7>) πρέπει να τεθεί σε λογικό '1'.
Για την ενεργοποίηση της ασύγχρονης επικοινωνίας το bit SYNC του καταχωρητή TXSTA (TXSTA<4>) τίθεται σε λογικό 1.
Τα bits TRISC < 7:6 > πρέπει να τεθούν σε κατάσταση "10", για να διαμορφωθούν τα pins RC6/TX/CX και RC7/RX/DT της USART ως σειριακή έξοδος και είσοδος αντίστοιχα. Το πρώτο (RC6) εκπέμπει δεδομένα, ενώ το δεύτερο (RC7) λαμβάνει δεδομένα.
Για τον καθορισμό του baud rate εγγράφουμε τον καταχωρητή SPBRG με μια τιμή που προκύπτει από τις παρακάτω σχέσεις. Σημειώνεται ότι σε περίπτωση που επιθυμούμε χαμηλούς ρυθμούς μετάδοσης, το bit BRGH (TXSTA<2>) είναι μηδέν, οπότε χρησιμοποιούμε την πρώτη σχέση, ενώ αν επιθυμούμε υψηλούς ρυθμούς μετάδοσης το bit αυτό τίθεται σε λογικό 1 και η τιμή SPBRG προκύπτει από τη δεύτερη σχέση:

$$\text{For } \mathbf{BRGH} = 0 \qquad \text{Baud rate} = \frac{f_{\text{osc}}}{64([\mathbf{SPBRG}] + 1)}$$

$$\text{For } \mathbf{BRGH} = 1 \qquad \text{Baud rate} = \frac{f_{\text{osc}}}{16([\mathbf{SPBRG}] + 1)}$$

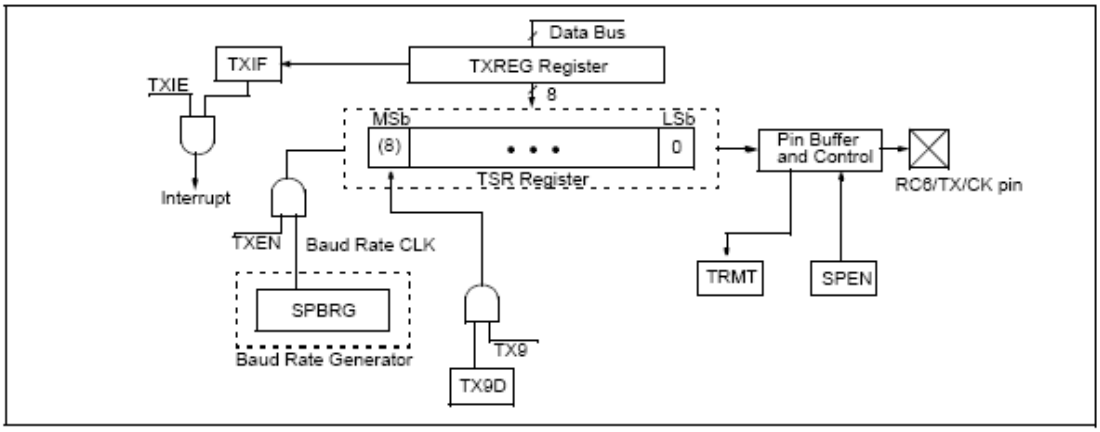
Η Ασύγχρονη επικοινωνία χρησιμοποιεί τη μορφή μετάδοσης που περιγράψαμε στην παράγραφο 7.1: ένα bit START, οχτώ ή εννιά bit δεδομένων και ένα STOP bit. Η USART στέλνει και λαμβάνει το LSB (Least Significant Bit) πρώτα. Η αποστολή και η λήψη είναι λειτουργικά ανεξάρτητες, όμως χρησιμοποιούν την ίδια μορφή δεδομένων και τον ίδιο ρυθμό μετάδοσης.

7.7 Ασύγχρονη Αποστολή

Ο πυρήνας της μετάδοσης είναι ένας εσωτερικός καταχωρητής, ο καταχωρητής TSR. Ο TSR καταχωρητής περιέχει τα δεδομένα του buffer μετάδοσης TXREG (όπου ο TXREG περιέχει τα δεδομένα αποστολής που φορτώνουμε εμείς από το Software). Ο TSR δεν φορτώνεται με δεδομένα έως ότου έρθει το STOP bit από την προηγούμενη φόρτωση. Μόλις έρθει το STOP bit τα δεδομένα (εάν υπάρχουν) μεταφέρονται από τον TXREG

στον TSR. Μόλις ο TXREG στείλει τα δεδομένα στον TSR, υψώνεται σημαία ότι ο TXREG είναι άδειος. Αυτό δηλώνεται από το bit TXIF (PIR1<4>) με την ένδειξη ‘1’. Αυτή η διακοπή μπορεί να ενεργοποιηθεί θέτοντας ‘1’ το bit TXIE (PIE1<4>). Αν το TXIE είναι ‘0’ το TXIF θα γίνει ‘1’,αλλά δεν θα παραχθεί σήμα διακοπής. Ο TXIF γίνεται ‘0’ μόλις φορτωθούν καινούργια δεδομένα στον TXREG.

Η μετάδοση μπορεί να ενεργοποιηθεί θέτοντας ‘1’ το bit TXEN (TXSTA<5>). Η μετάδοση στην πραγματικότητα δεν θα πραγματοποιηθεί μέχρι ο TXREG να φορτωθεί με δεδομένα και ο Γεννήτορας Ρυθμού Μετάδοσης να παράγει τον χρονισμό.



Σχήμα 7.5: Διάγραμμα Ασύγχρονης Αποστολής Δεδομένων

Πίνακας 7.3: Καταχωρητές που σχετίζονται με την Ασύγχρονη Αποστολή

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	R0IF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
19h	TXREG	USART Transmit Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Για να καθορίσουμε μια Ασύγχρονη Αποστολή ακολουθούμε τα εξής βήματα:

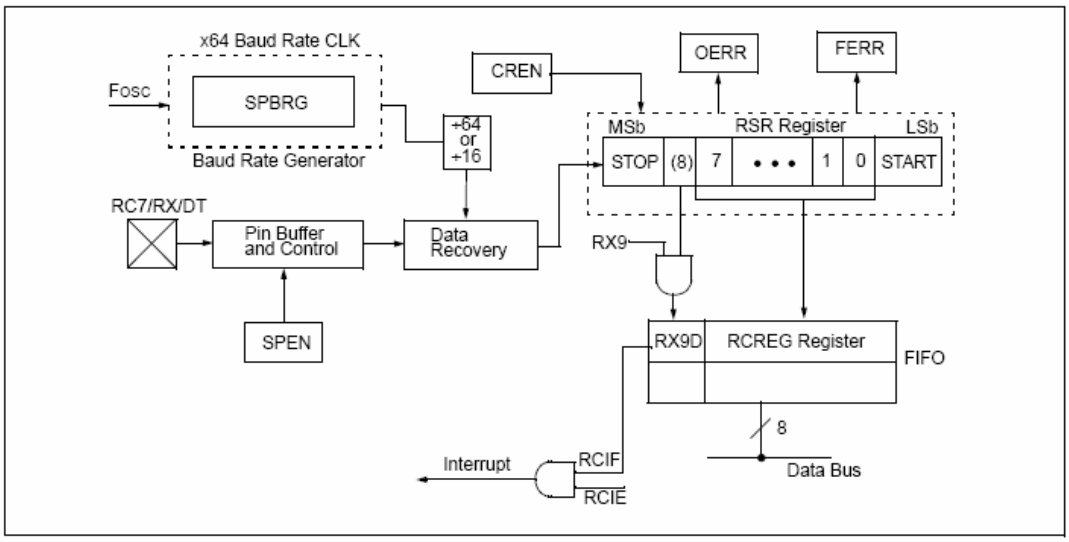
1. Αρχικοποιούμε τον καταχωρητή SPBRG με το κατάλληλο ρυθμό μετάδοσης. Εάν επιθυμούμε υψηλούς ρυθμούς μετάδοσης, θέτουμε ‘1’ το bit BRGH
2. Ενεργοποιούμε την Ασύγχρονη Σειριακή πόρτα θέτοντας ‘0’ το bit SYNC (TXSTA<4>) και ‘1’ το bit SPEN (RCSTA<7>)

3. Εάν επιθυμούμε διακοπές, τότε ενεργοποιούμε το bit TXIE (PIE1<4>)
4. Ενεργοποιούμε την μετάδοση θέτοντας ‘1’ το bit TXEN (TXSTA<5>), το οποίο θα θέσει με την σειρά του ‘1’ το bit TXIF (PIR1<4>)
5. Φορτώνουμε στον TXREG τα δεδομένα (ξεκινάει η μετάδοση)
6. Εάν χρησιμοποιούμε διακοπές, σιγουρευόμαστε ότι τα bits (6 και 7) του καταχωρητή INTCON είναι ‘1’

Στο Σχήμα 7.5 παρουσιάζεται το διάγραμμα της ασύγχρονης αποστολής των δεδομένων της σειριακής επικοινωνίας.

7.8 Ασύγχρονη Λήψη

Αντίστοιχα με όσα συμβαίνουν στην ασύγχρονη σειριακή εκπομπή συμβαίνουν και κατά την σειριακή λήψη. Τα δεδομένα λαμβάνονται από το pin RC7/RX/DT και μεταφέρονται στον Data Recovery. Ο πυρήνας της λήψης είναι ο εσωτερικός καταχωρητής RSR. Μετά τη λήψη του STOP bit τα δεδομένα λήψης τα οποία βρίσκονται στον RSR, μεταφέρονται στον καταχωρητή RCREG (αν αυτός είναι άδειος). Αν η μεταφορά ολοκληρωθεί η σημαία του καταχωρητή RCIF (PIR1<5>), γίνεται ‘1’. Η πραγματική διακοπή μπορεί να ενεργοποιηθεί θέτοντας ‘1’ το bit RCIE, του καταχωρητή PIE<5>.
Η διαδικασία της ασύγχρονης σειριακής λήψης των δεδομένων παρουσιάζεται στο Σχήμα 7.6.



Σχήμα 7.6: Διάγραμμα Ασύγχρονης Λήψης Δεδομένων

Για να καθορίσουμε μια Ασύγχρονη Λήψη ακολουθούμε τα εξής βήματα:

1.Φόρτωση του SPBRG για το κατάλληλο ρυθμό μετάδοσης. Αν θέλουμε υψηλό ρυθμό μετάδοσης τότε το bit BRGH πρέπει να τεθεί ‘1’

2. Ενεργοποιούμε την ασύγχρονη σειριακή πόρτα με το μηδενισμό του SYNC bit και θέτοντας ‘1’ το bit SPEN

3. Αν η διακοπή είναι επιθυμητή θέτουμε ‘1’ το bit RCIE

4.Ενεργοποιούμε την λήψη θέτοντας ‘1’ το bit CREN

5. Η σημαία του bit RCIF θα είναι ‘1’ όταν η λήψη είναι ολοκληρωμένη και μια διακοπή θα λάβει χώρα εφόσον το RCIE bit θα είναι ‘1’

6. Διαβάζουμε τα 8-bit δεδομένων που λήφθηκαν διαβάζοντας τον καταχωρητή RCREG

7. Εάν οποιοδήποτε σφάλμα λάβει χώρα, μηδενίζουμε το σφάλμα, μηδενίζοντας το bit CREN

8. Αν χρησιμοποιούμε διακοπή σιγουρευόμαστε ότι το GIE και το PEIE (7^{o} και 6^{o} bit αντίστοιχα) του καταχωρητή INTCON είναι ‘1’

Πίνακας 7.4: Καταχωρητές που σχετίζονται με την Ασύγχρονη Λήψη

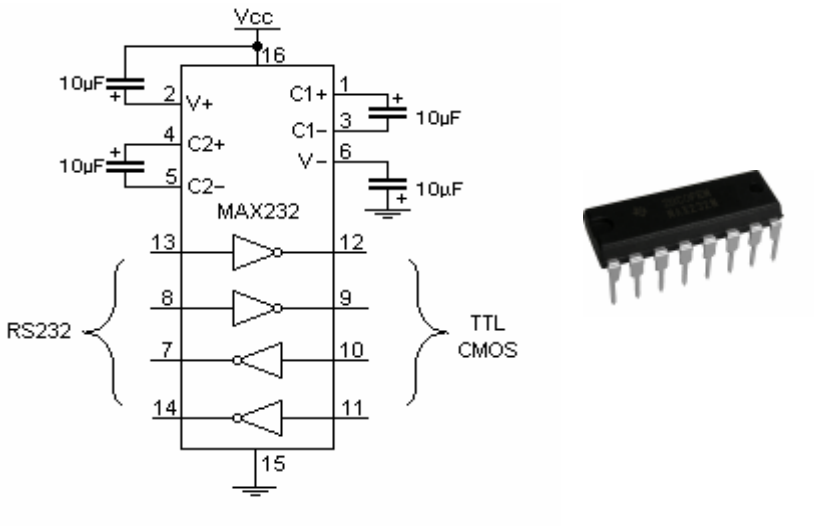
Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	R0IF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
1Ah	RCREG	USART Receive Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

7.9 Το κύκλωμα MAX232

Τα διάφορα ψηφιακά ολοκληρωμένα κυκλώματα που χρησιμοποιούμε στις κατασκευές μας δέχονται στάθμες TTL ή CMOS, στις οποίες συνήθως το λογικό ‘0’ αντιστοιχεί σε 0 Volts και το λογικό ΄1΄σε 5 Volts. Για να διασυνδέσουμε επομένως κάποια εφαρμογή με τη σειριακή θύρα, θα πρέπει να παρεμβάλουμε κάποια κυκλώματα που θα μετατρέπουν τις στάθμες του πρωτοκόλλου RS-232 (Πίνακας 7.1) στις παραπάνω απλές στάθμες TTL.

Ένα κύκλωμα που χρησιμοποιείται ευρύτατα για τέτοιο σκοπό είναι το ολοκληρωμένο κύκλωμα MAX232 και τα συμβατά με αυτό. Το κύκλωμα αυτό περιέχει δύο γραμμές εκπομπής και δύο γραμμές λήψης. Λειτουργεί με μια απλή τροφοδοσία +5V και παράγει τα -10 και +10 Volts που απαιτούνται για τις στάθμες της θύρας RS-232 με τη βοήθεια μιας αντλίας φορτίσεως.

Άλλοι τυπικοί μετατροπείς στάθμης για το πρωτόκολλο RS-232 είναι τα ολοκληρωμένα κυκλώματα DS1488 και DS1489. Το πρώτο εκτελεί εκπομπή δεδομένων προς τη θύρα RS-232 και το δεύτερο εκτελεί λήψη δεδομένων από την θύρα RS-232. Κάθε κύκλωμα περιέχει τέσσερις αντιστροφείς ενός μόνο τύπου, δηλαδή εκπομπούς ή

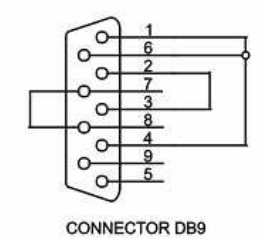


Σχήμα 7.7: Τυπική εφαρμογή του ολοκληρωμένου κυκλώματος MAX232

δέκτες. Στο Σχήμα 7.7 φαίνεται η τυπική εφαρμογή του ολοκληρωμένου κυκλώματος MAX232.

7.10 Παράδειγμα κώδικα για την ασύγχρονη σειριακή επικοινωνία

Παρακάτω παρατίθεται παράδειγμα κώδικα που στέλνει έναν χαρακτήρα και στη συνέχεια λαμβάνει έναν χαρακτήρα. Ο πιο πρόσφορος τρόπος για να χρησιμοποιηθεί ο κώδικας αυτούσιος είναι μια εφαρμογή “echo”, όπου η σειριακή θύρα στέλνει χαρακτήρα και αμέσως λαμβάνει πίσω τον χαρακτήρα που μόλις έστειλε. Αυτό μπορεί να γίνει με κατάλληλη επιστροφή του σήματος TX στον ακροδέκτη RX, δηλαδή βραχυκυκλώνοντας του ακροδέκτες 2 και 3 του σειριακού συνδέσμου (σχήμα 7.8).
Ο παρακάτω κώδικας είναι δομημένος με υπορουτίνες. Η υπορουτίνα initialize επιτελεί την αρχικοποίηση των καταχωρητών της σειριακής θύρας. Η υπορουτίνα send στέλνει έναν χαρακτήρα και η υπορουτίνα receive λαμβάνει. Ανάμεσα σε διαδοχικές αποστολές-λήψεις το σύστημα αναμένει την ενεργοποίηση ενός διακόπτη πιέσεως (push-button). Η υπορουτίνα καθυστέρησης delay_ms χρησιμοποιείται για τον αποκλυδωνισμό (debouncing) του διακόπτη.



Σχήμα 7.8 Σύνδεση βρόχου αυτοακρόασης (τάπα) για λειτουργία echo

```
#include "p16f877.inc"
__CONFIG _CP_OFF & _WDT_OFF & _XT_OSC & _PWRTE_OFF & _CPD_OFF &
_WRT_ENABLE_ON & _BODEN_ON & _LVP_OFF

msec equ 20h
                Org 0
                call initialize
                movlw 0xAA              ;test pattern
                movwf PORTB
main            btfss PORTC, 0          ;is button pressed?
                goto $-1
                movlw d'35'             ;wait 25ms to debounce
                call delay_ms
                btfsc PORTC, 0          ;is button released?
                goto $-1
                movlw d'35'             ;wait 25ms to debounce
                call delay_ms
                movf PORTD,w
                call send
                call receive
                movwf PORTB
                goto main

;initialize serial port
initialize      bsf STATUS,RP0
                 movlw 0x00
                movwf TRISB             ;PORTB output
                movlw b'10000001'       ;RC7 (RX) input, RC0 input
                 movwf TRISC
                movlw b'00100100'        ;Transmit enable, high speed baud rate
                movwf TXSTA
                movlw d'103'            ;2404 bps
                movwf SPBRG
                bcf STATUS,RP0
                movlw b'10010000' ;port is on, 8-bit transfer, no address detect
                movwf RCSTA
                return

;subroutine for transmit
send            btfss PIR1,TXIF         ;Edw perimenei mexri to TXIF na ginei '1', pou
                                        ;shmainei adeios TXREG
```

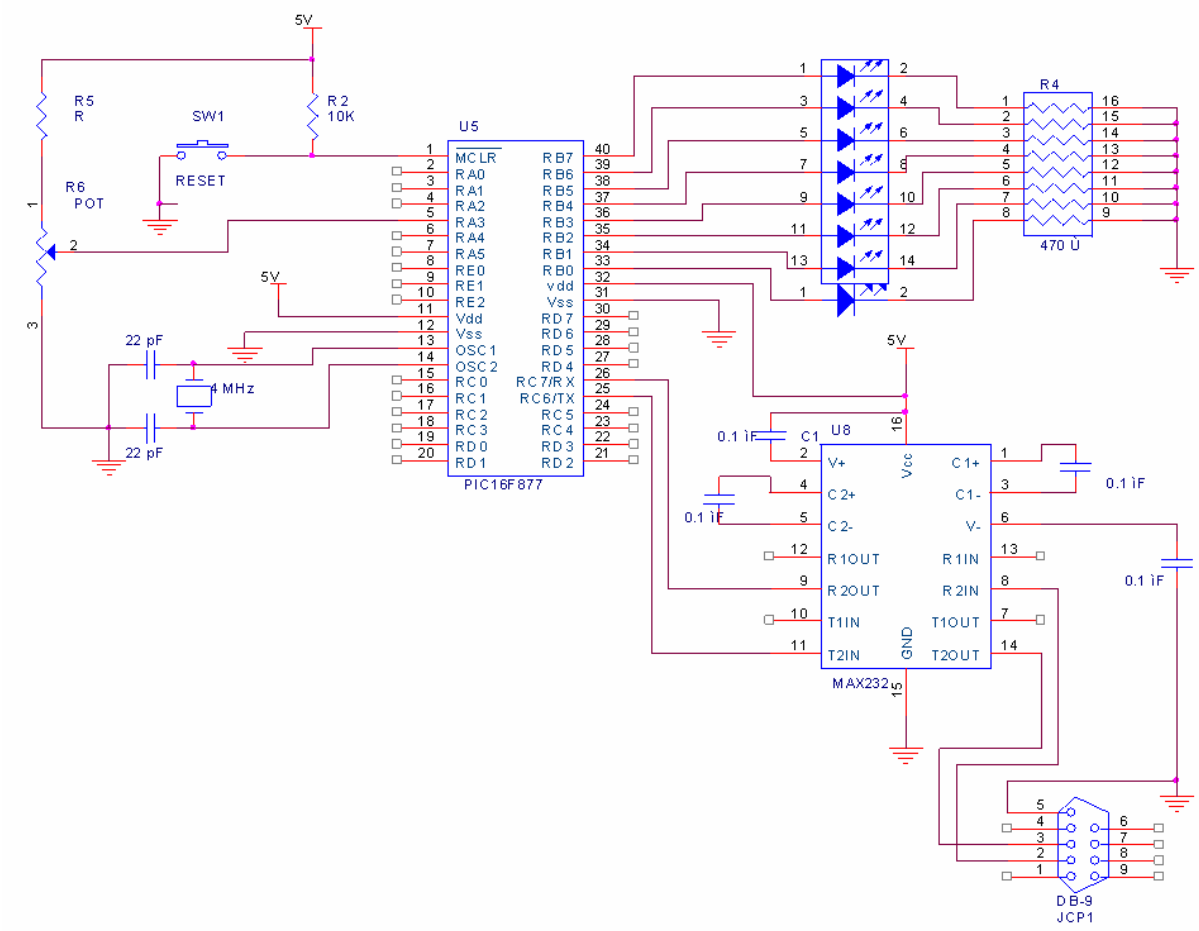
```
                goto $-1                ;wste na fortwsw ta dedomena ston TXREG.
                movwf TXREG
                return

;subroutine for receive
receive         btfss PIR1,RCIF         ;otan ginei 1 exei ftasei xarakthras
                goto receive             ;wait
                movf RCREG,w
                return

;delay subroutine
delay_ms         movwf msec
loop            movlw 0xF8
                call micro4
                nop
                nop
                decfsz msec, f
                goto loop
                return
micro4          addlw 0FF               ;add -1 in 2's complement
                btfss STATUS,Z          ;no skip 1μs. With skip 2μ
                goto micro4
                return

end
```

7.11 Ολοκληρωμένη εφαρμογή συλλογής και αποστολής δεδομένων

Στο σχήμα 7.9 παρουσιάζεται μια ολοκληρωμένη εφαρμογή με βάση τον μικροελεγκτή PIC16F877. Ο μικροελεγκτής συλλέγει τιμές μέσω του μετατροπέα αναλογικού σήματος σε ψηφιακό, διαβάζοντας τις τάσεις στην μεσαία λήψη ενός ποτενσιομέτρου. Στη θέση του ποτενσιομέτρου μπορεί να βρίσκεται ένα αισθητήρας που μετρά κάποιο φυσικό μέγεθος (π.χ. θερμοκρασία, πίεση, φωτεινή στάθμη κλπ.). Στο κύκλωμα του σχήματος 7.9 χρησιμοποιούμε το κανάλι 3 (RA3) του μετατροπέα ADC. Στο ίδιο κύκλωμα φαίνονται και οι υπόλοιπες απαραίτητες τυπικές συνδέσεις, όπως του κρυσταλλικού ταλαντωτή και της τροφοδοσίας. Η θύρα B χρησιμοποιείται ως έξοδος και στους ακροδέκτες της συνδέονται LEDs για την απεικόνιση των μετρήσεων. Έτσι, κάθε τιμή που συλλέγει ο ADC μπορεί να απεικονίζεται στη θύρα B. Οι ακροδέκτες 25 και 26 χρησιμοποιούνται ως TX και RX της σειριακής θύρας και συνδέονται με το MAX232 για τη μετατροπή της στάθμης στα επίπεδα που προβλέπει το πρωτόκολλο RS232C. Για να ολοκληρωθεί η εφαρμογή χρειάζεται να συνδεθεί ο συνδετήρας DB-9 με τη σειριακή θύρα ενός προσωπικού υπολογιστή ή με έναν μετατροπέα USB-to-serial. Από τη μεριά

του ξενιστή (host) υπολογιστή πρέπει να εκτελείται μια εφαρμογή επικοινωνίας με τη σειριακή θύρα. Για το σκοπό αυτό ο χρήστης μπορεί να χρησιμοποιήσει το περιβάλλον προγραμματισμού Matlab ή το περιβάλλον συλλογής και επεξεργασίας δεδομένων LabVIEW.

Ο ενδιαφερόμενος χρήστης μπορεί να δημιουργήσει τις εφαρμογές του μικροελεγκτή και του ξενιστή υπό μορφή άσκησης, λαμβάνοντας υπόψη όσα αναπτύχθηκαν στο παρόν και στο προηγούμενο κεφάλαιο. Για την ανάπτυξη εφαρμογών σειριακής επικοινωνίας στον ξενιστή μπορεί να ανατρέξει και στο λήμμα 8 της βιβλιογραφίας.



Σχήμα 7.9 Ολοκληρωμένη εφαρμογή για τη λήψη δεδομένων μέσω του μετατροπέα ADC και αποστολή των δεδομένων σε υπολογιστή μέσω της σειριακής θύρας.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Πεκμεστζή Κιαμάλ, “Συστήματα μικροϋπολογιστών” (τόμος 2). Μικροελεγκτές AVR και PIC, Εκδόσεις Συμμετρία, 2009.

2. Σ. Αλατσαθιανός, Μικροελεγκτές PIC, Εκδόσεις Γκιούρδα, 2008.

3. Tim Wilmhurst, “An Introduction to the design of small-scale embedded systems”, Palgrave, 2001.

4. Tim Wilmhurst, “Designing embedded systems with PIC microcontrollers”, Newness.

5. Ιωάννη Καλόμοιρου, “Έλεγχος κυκλωμάτων και μετρήσεων με Η/Υ”, Εκδόσεις Τζιόλα, 2001.