Κεφάλαιο 5ο

Αρχιτεκτονική και προγραμματισμός του μικροελεγκτή PIC

Περιεχόμενα

Στόχοι του κεφαλαίου:

Όταν ολοκληρώσεις το κεφάλαιο αυτό θα μπορείς να...

- περιγράφεις την δομή και λειτουργία του μικροελεγκτή PIC.
- κατανοείς τι είναι μία εντολή Assembly και πώς, εκτελείται από το σύστημα.
- περιγράφεις πώς γίνεται η αποθήκευση και ανάκληση δεδομένων.
- διαβάζεις κώδικα γλώσσας Assembly.
- γράφεις στοιχειώδη κώδικα σε γλώσσα Assembly.

Εισαγωγή

Στα προηγούμενα κεφάλαια είδαμε τις βασικές αρχές δομής και λειτουργίας ενός μικροϋπολογιστικού συστήματος. Μάθαμε, ότι όλα αυτά αποτελούν εφαρμογές των ψηφιακών ηλεκτρονικών. Ακόμη, είδαμε το δυαδικό και το δεκαεξαδικό σύστημα και πώς μπορούμε να κάνουμε τις διάφορες αριθμητικές πράξεις με αυτά. Επιπλέον, μιλήσαμε, διεξοδικά, για την αρχιτεκτονική των μικροϋπολογιστικών συστημάτων, τους μικροεπεξεργαστές, τους μικροελεγκτές, καθώς και τις ομοιότητες και τις διαφορές τους. Τέλος, είδαμε πώς ένας μικροεπεξεργαστής ή μικροελεγκτής μπορεί να συνδεθεί με τις κατάλληλες περιφερειακές μονάδες, σε μία πρακτική εφαρμογή.

Στο κεφάλαιο αυτό θα μελετήσουμε έναν πραγματικό μικροελεγκτή, που παρά το μικρό του κόστος, μπορεί να χρησιμοποιηθεί σε πάρα πολλές, μικρές και μεγάλες, εφαρμογές. Πρόκειται για τον μικροελεγκτή PIC, της εταιρείας Microchip. Ο συγκεκριμένος μικροελεγκτής χρησιμοποιείται και στις εργαστηριακές ασκήσεις του μαθήματός μας.

Αφού δούμε την αρχιτεκτονική του και τα διάφορα επιμέρους τμήματά του, θα μάθουμε πώς μπορούμε να τον προγραμματίσουμε, γράφοντας ένα πρόγραμμα σε γλώσσα Assembly. Επίσης, θα αναφερθούμε στις δυνατότητες που αυτή η γλώσσα μας προσφέρει. Άλλωστε, οι δυνατότητες αυτές είναι που την κάνουν να διαφοροποιείται από τις υπόλοιπες γλώσσες προγραμματισμού.

5.1 Δομή του μικροελεγκτή PIC

5.1.1 Γενικά στοιχεία

Στο τρίτο κεφάλαιο του βιβλίου, είδαμε ότι υπάρχουν αρκετές οικογένειες μικροελεγκτών. Επίσης, είπαμε ότι οι μικροεπεξεργαστές και μικροελεγκτές χωρίζονται σε δύο μεγάλες κατηγορίες, αυτών με αρχιτεκτονική CISC και αυτών με RISC. Ο μικροελεγκτής που θα μελετήσουμε είναι ο PIC, της εταιρείας Microchip και είναι αρχιτεκτονικής RISC.

Η δομή του μικροελεγκτή PIC μπορεί να χωριστεί σε δύο μέρη, τον πυρήνα (core) και τις περιφερειακές μονάδες του (peripheral units). Ο πυρήνας του μικροελεγκτή αποτελείται από όλα εκείνα τα στοιχεία, τα οποία είναι απολύτως απαραίτητα για την λειτουργία του. Οι περιφερειακές μονάδες βρίσκονται ενσωματωμένες στον μικροελεγκτή και είναι αυτές που τον κάνουν να διαφέρει από έναν μικροεπεξεργαστή.

Στον πυρήνα του PIC ανήκουν οι:

- Κεντρική μονάδα επεξεργασίας
- Μνήμη
- Εντολές
- Λειτουργίες διακοπών

Εδώ, πρέπει να σημειώσουμε ότι, λόγω της σημαντικότητάς τους, έχουμε συμπεριλάβει και τις εντολές στον πυρήνα του PIC, παρόλο που πρόκειται μάλλον για κάποιο λογικό παρά υλικό στοιχείο του μικροελεγκτή.

Στις περιφερειακές μονάδες ανήκουν:

- Οι θύρες εισόδου / εξόδου γενικής χρήσης
- Οι μετρητές χρόνου (τρεις μονάδες)
- Η μονάδα διαμόρφωσης πλάτους
- Οι θύρες σειριακής επικοινωνίας (τρεις θύρες)
- Η θύρα παράλληλης επικοινωνίας
- Η μονάδα παραγωγής τάσης αναφοράς
- Οι συγκριτές
- Ο μετατροπέας αναλογικού σήματος σε ψηφιακό

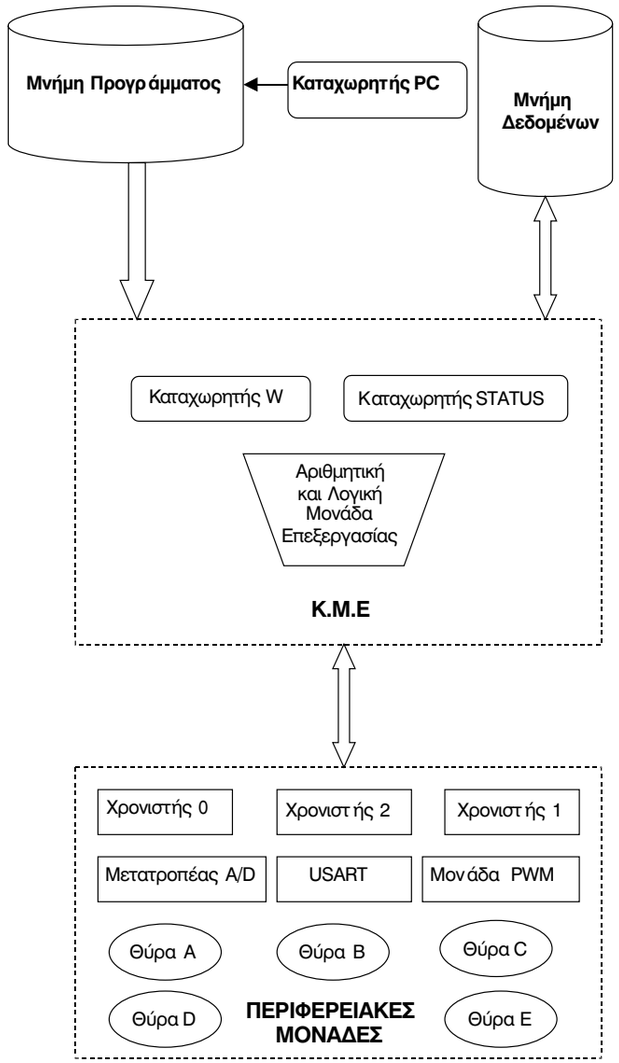
Στη συνέχεια, θα αναλύσουμε τα παραπάνω στοιχεία και θα δούμε πώς αυτά λειτουργούν και συνεργάζονται στις διάφορες εφαρμογές του PIC.

5.1.2 Κεντρική Μονάδα Επεξεργασίας

Η κεντρική μονάδα επεξεργασίας (Κ.Μ.Ε), γνωστή και ως CPU, εκτελεί τις εντολές του προγράμματος που έχουμε αποθηκεύσει σε μία μνήμη η οποία ονομάζεται μνήμη προγράμματος. Από τη μνήμη αυτή, η κεντρική μονάδα επεξεργασίας φέρνει, με τη

σειρά, τις εντολές του προγράμματος, τις αποκωδικοποιεί και τις εκτελεί. Εδώ, πρέπει να σημειώσουμε ότι ο PIC αναγνωρίζει τριανταπέντε (35) εντολές προγραμματισμού. Για τις εντολές αυτές θα μιλήσουμε αναλυτικά στις ενότητες που ακολουθούν.

Μέσα στην κεντρική μονάδα επεξεργασίας, βρίσκεται και η αριθμητική και λογική μονάδα (Α.Λ.Μ.). Το σχήμα 5.1 παρουσιάζει την Κ.Μ.Ε. μαζί με τα άμεσα, με αυτήν, συνδεόμενα στοιχεία του PIC. Οι αριθμητικές πράξεις που μπορεί να εκτελεί είναι η πρόσθεση και η αφαίρεση. Επίσης, έχει τη δυνατότητα να εκτελεί λογικές πράξεις (AND, OR, XOR, κτλ). Η μονάδα επεξεργάζεται δεδομένα μήκους οκτώ δυαδικών ψηφίων (8-bit).

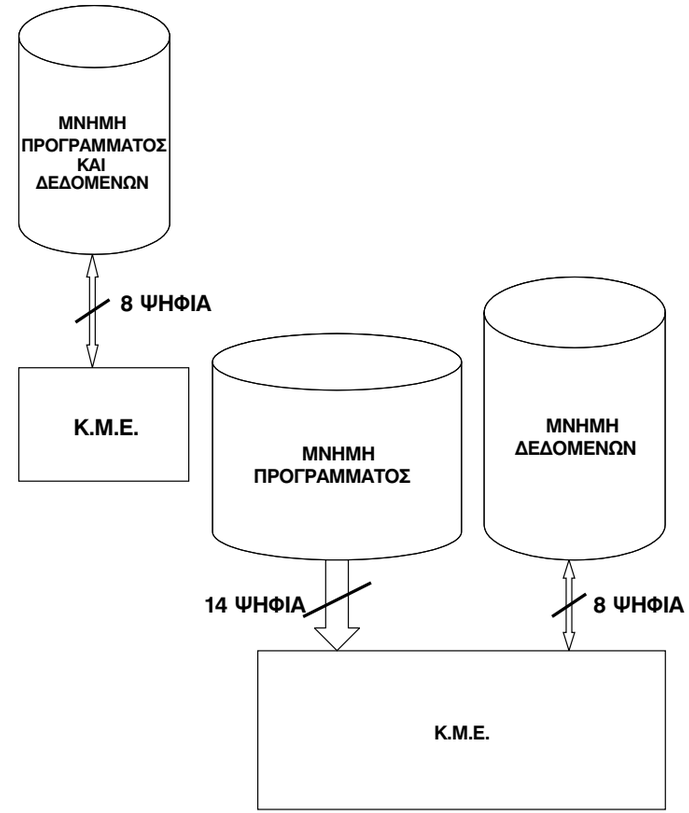


Σχήμα 5.1 *Η Δομή μικροελεγκτή PIC.*

Για τη σωστή ανεύρεση των εντολών, που πρέπει να εκτελεσθούν, η κεντρική μονάδα επεξεργασίας χρησιμοποιεί ένα μετρητή προγράμματος. Στην πραγματικότητα, πρόκειται για έναν καταχωρητή, ο οποίος περιέχει τη διεύθυνση της μνήμης στην οποία βρίσκεται αποθηκευμένη η εντολή που πρέπει να εκτελεσθεί. Ο καταχωρητής αυτός ονομάζεται Program Counter (PC).

5.1.3 Μνήμη

Στη σχεδίαση μικροεπεξεργαστών και μικροελεγκτών ακολουθούνται δύο αρχιτεκτονικές. Στην πρώτη χρησιμοποιείται μία μνήμη τόσο για την αποθήκευση του προγράμματος όσο και για την αποθήκευση των δεδομένων. Στη δεύτερη χρησιμοποιούνται δύο ξεχωριστές μνήμες. Η μία χρησιμοποιείται για την αποθήκευση του προγράμματος και λέγεται μνήμη προγράμματος ενώ η άλλη για την αποθήκευση των δεδομένων και λέγεται μνήμη δεδομένων. Οι δύο αρχιτεκτονικές παρουσιάζονται στο σχήμα 5.2.



Σχήμα 5.2 *Αρχιτεκτονικές μνήμης μικροελεγκτών. (α) Ενιαία μνήμη προγράμματος και δεδομένων. (β) Ξεχωριστή μνήμη προγράμματος και δεδομένων (μικροελεγκτής PIC).*

Ο μικροελεγκτής PIC ακολουθεί την δεύτερη αρχιτεκτονική. Στην δεύτερη αρχιτεκτονική, εντολές και δεδομένα κινούνται σε ξεχωριστούς διαδρόμους (διαύλους) με αποτέλεσμα αυτό να μπορεί να γίνει όχι μόνον με πολύ μεγαλύτερη ταχύτητα αλλά ακόμη και την ίδια χρονική στιγμή. Αντίθετα, στην πρώτη αρχιτεκτονική, εντολές και δεδομένα μοιράζονται τον ίδιο διάδρομο με αποτέλεσμα να ελαττώνεται η ταχύτητα μεταφοράς τους. Επιπλέον, το πλεονέκτημα της δεύτερης αρχιτεκτονικής, να χρησιμοποιεί ξεχωριστούς χώρους μνήμης για την αποθήκευση των δεδομένων και του προγράμματος, δίνει τη δυνατότητα χρησιμοποίησης μνήμης με διαφορετικό μήκος λέξης. Έτσι, στην περίπτωση του PIC, η μνήμη προγράμματος έχει μήκος λέξης δεκατεσσάρων (14) δυαδικών ψηφίων (bits) αντί των οκτώ (8), της μνήμης των δεδομένων, με σκοπό όλες οι εντολές να κωδικοποιούνται σε μία λέξη. Θυμίζουμε ότι, γενικά, οι εντολές των μικροεπεξεργαστών και μικροελεγκτών μπορεί να έχουν μήκος μίας, δύο ή, ακόμη, και περισσοτέρων λέξεων, με αντίστοιχη, βέβαια, αύξηση του χρόνου εκτέλεσής τους.

Το μέγεθος της μνήμης προγράμματος κυμαίνεται από 2 ως 8KBytes και, συνήθως, είναι τύπου Flash. Η συγκεκριμένη τεχνολογία επιτρέπει όχι μόνον την εγγραφή αλλά και το σβήσιμο της μνήμης να γίνεται με ηλεκτρικό τρόπο. Αυτό σημαίνει ότι ο προγραμματισμός του μικροελεγκτή γίνεται εύκολα ενώ αυτός βρίσκεται συνδεδεμένος στο κύκλωμα της εκάστοτε εφαρμογής.

Το μέγεθος της μνήμης δεδομένων αποτελείται από τρία τμήματα, με μέγεθος 128Bytes το κάθε ένα, δηλαδή 384Bytes συνολικά. Το κάθε τμήμα αποτελείται τόσο από καταχωρητές γενικού όσο και ειδικού σκοπού. Μερικοί από τους καταχωρητές ειδικού σκοπού χρησιμοποιούνται για τον έλεγχο του πυρήνα του PIC ενώ άλλοι για τον έλεγχο των περιφερειακών του.

Στις επόμενες ενότητες του κεφαλαίου αυτού, θα αναφερθούμε αναλυτικά στον τρόπο με τον οποίο ο μικροελεγκτής μας χρησιμοποιεί τις μνήμες που προαναφέραμε.

5.1.4 Εντολές

Όσον αφορά την σχεδίαση εντολών οι μικροεπεξεργαστές και μικροελεγκτές ακολουθούν είτε την αρχιτεκτονική RISC είτε την αρχιτεκτονική CISC. Ενδεικτικά αναφέρουμε ότι μεγάλοι κεντρικοί ηλεκτρονικοί υπολογιστές βασίζονται σε μικροεπεξεργαστές αρχιτεκτονικής RISC, ενώ ο προσωπικός μας ηλεκτρονικός υπολογιστής βασίζεται σε μικροεπεξεργαστή αρχιτεκτονικής CISC. Οι μικροεπεξεργαστές και μικροελεγκτές αρχιτεκτονικής RISC χαρακτηρίζονται από το γεγονός ότι διαθέτουν μικρό αριθμό εντολών, μπορούν όμως να ικανοποιήσουν όλες τις πιθανές απαιτήσεις προγραμματισμού.

Ο μικροελεγκτής PIC, ακολουθεί την αρχιτεκτονική RISC και έχει συνολικά τριανταπέντε (35) εντολές, μήκους μίας λέξης (14 bit). Έτσι, σε αντίθεση με τους μικροελεγκτές αρχιτεκτονικής CISC, ο PIC εκτελεί την κάθε εντολή σε έναν κύκλο

μηχανής, με αποτέλεσμα τη σημαντική βελτίωση της ταχύτητας επεξεργασίας. Εδώ, ας τονίσουμε ότι μοναδική εξαίρεση αποτελούν οι εντολές διακλάδωσης, οι οποίες εκτελούνται σε δύο κύκλους μηχανής. Συγκριτικά αναφέρουμε ότι ο γνωστός μικροεπεξεργαστής Z80, ο οποίος είναι αρχιτεκτονικής CISC, έχει εντολές που εκτελούνται σε δέκα (10) ή και περισσότερους κύκλους μηχανής.

Οι εντολές του PIC και ο προγραμματισμός του, θα μας απασχολήσουν εκτενώς στη συνέχεια του μαθήματός μας.

5.1.5 Λειτουργίες Διακοπών

Την ώρα που ο PIC εκτελεί ένα πρόγραμμα που του έχουμε δώσει, μπορεί να συμβούν διάφορα γεγονότα, στο περιβάλλον που αυτός ελέγχει. Ανάλογα με τη σημασία του κάθε γεγονότος, ενδεχομένως, να χρειαστεί να διακόψει την εκτέλεση του κυρίως προγράμματος για να ασχοληθεί με το γεγονός αυτό. Ο PIC για να αντιληφθεί την ύπαρξη των γεγονότων αυτών έχει αναθέσει σε κάποιες από τις περιφερειακές του μονάδες να διενεργούν διάφορους ελέγχους. Αυτό μπορεί να γίνεται ανεξάρτητα από τις λοιπές λειτουργίες του μικροελεγκτή. Όταν μία από τις μονάδες εντοπίσει το γεγονός, τότε, στέλνει ένα σήμα στην K.M.E. του μικροελεγκτή να διακόψει την εκτέλεση του προγράμματος που εκτελεί. Το σήμα λέγεται διακοπή και προκαλεί την άμεση εκτέλεση ενός τμήματος κώδικα, το οποίο λέγεται ρουτίνα εξυπηρέτησης της διακοπής.

Ως παράδειγμα μπορούμε να αναφέρουμε τη χρήση ενός μικροϋπολογιστικού συστήματος βασισμένου στον PIC, που χρησιμοποιείται για τον έλεγχο μίας κατεργασίας μετάλλου σε ένα βιομηχανικό περιβάλλον. Εκεί, ανάμεσα στις άλλες ασχολίες του, ο PIC, πρέπει και να ρυθμίζει τη θερμοκρασία σε κάποιον κλίβανο. Επίσης, για λόγους ασφαλείας, μετρά τη θερμοκρασία σε διάφορα σημεία της κατεργασίας. Την ώρα, λοιπόν, που εκτελεί μία συνηθισμένη διαδικασία, π.χ. την αποστολή κάποιων δεδομένων από τη μνήμη του σε έναν κεντρικό Η/Υ, διαπιστώνεται επικίνδυνη αύξηση της θερμοκρασίας σε κάποιο σημείο της κατεργασίας. Τότε, το περιφερειακό που την εντόπισε, προκαλεί μια διακοπή στην K.M.E. Η αποστολή σταματά και η K.M.E. καλείται να εκτελέσει την ρουτίνα που συνοδεύει τη διακοπή. Η ρουτίνα περιέχει εντολές που σβήνουν τον κλίβανο, ανοίγουν κάποιους ανεμιστήρες ψύξης και ενεργοποιούν τη σειρήνα κινδύνου.

Το παράδειγμα, που περιγράψαμε, δείχνει χαρακτηριστικά τη λειτουργία των διακοπών. Ωστόσο, το γεγονός ότι αναφέρεται σε μία κατάσταση κινδύνου δε σημαίνει ότι μόνον τότε μπορεί να έχουμε μία διακοπή. Διακοπές έχουμε και σε διάφορες άλλες απλές περιπτώσεις.

Για παράδειγμα, σε μία άλλη περίπτωση, η μονάδα που προκάλεσε τη διακοπή, μπορεί να είναι ένας μετατροπέας αναλογικού σήματος σε ψηφιακό. Τότε, με το σήμα που στέλνει στην K.M.E., ο μετατροπέας δηλώνει ότι έχει τελειώσει τη μετατροπή και

το αποτέλεσμα είναι έτοιμο. Στην περίπτωση αυτή, η Κ.Μ.Ε. διακόπτει την εκτέλεση των εργασιών που έκανε, μέχρι εκείνη την στιγμή, για να παραλάβει το αποτέλεσμα και, πιθανόν, να το επεξεργαστεί και να το αποθηκεύσει. Κάτι παρόμοιο συμβαίνει και με τη θύρα σειριακής επικοινωνίας, Όταν ολοκληρωθεί η λήψη ή η αποστολή κάποιου δεδομένου, το περιφερειακό, ειδοποιεί, με μία διακοπή, την Κ.Μ.Ε. για να το παραλάβει ή για να στείλει το επόμενο, αντίστοιχα.

Ο PIC δέχεται ένα πλήθος διακοπών, οι οποίες, κατά κύριο λόγο, προέρχονται από τις διάφορες περιφερειακές του μονάδες. Συνήθως, μία περιφερειακή μονάδα μπορεί να δώσει ένα σήμα διακοπής. Ωστόσο, υπάρχουν μονάδες που δίνουν περισσότερες διακοπές, όπως, για παράδειγμα, το περιφερειακό σειριακής επικοινωνίας. Τονίζεται ότι, έχουμε τη δυνατότητα να απενεργοποιούμε κάποιες από τις διακοπές, ή και όλες μαζί, όταν η εκτέλεση κάποιας εργασίας δεν πρέπει να διακοπεί.

Το θέμα είναι αρκετά μεγάλο και, για το λόγο αυτό, θα το εξετάσουμε, διεξοδικά, στο τέλος του κεφαλαίου, καθώς και στο επόμενο κεφάλαιο, όπου θα δούμε πώς χρησιμοποιούνται και προγραμματίζονται οι περιφερειακές μονάδες του PIC.

5.1.6 Περιφερειακές μονάδες

Όπως είπαμε και σε προηγούμενα κεφάλαια, η κύρια διαφορά των μικροελεγκτών από τους μικροεπεξεργαστές είναι ότι, οι πρώτοι, έχουν ενσωματωμένα διάφορα περιφερειακά. Αυτό τους κάνει κατάλληλους για πολλές εφαρμογές όπου το άμεσο ζητούμενο είναι η χρήση περιφερειακών μονάδων, όπως ο A/D μετατροπέας, η θύρα σειριακής επικοινωνίας, κτλ. Μία εφαρμογή μπορεί να είναι ο έλεγχος θερμοκρασίας ενός κλιβάνου, όπως στο παράδειγμα που είδαμε παραπάνω. Πολλές φορές, ένας μικροελεγκτής επιλέγεται για μία εφαρμογή με γνώμονα το είδος και τις δυνατότητες των περιφερειακών που διαθέτει. Ο μικροελεγκτής PIC έχει ενσωματωμένα αρκετά περιφερειακά, που του δίνουν τη δυνατότητα να χρησιμοποιηθεί για ένα πλήθος εφαρμογών.

Ο μικροελεγκτής μας έχει πέντε θύρες εισόδου / εξόδου, των οκτώ (8) δυαδικών ψηφίων (bits). Αυτές μπορούν να χρησιμοποιηθούν είτε σαν απλές θύρες είτε σαν θύρες των υπολοίπων περιφερειακών που διαθέτει. Η τέταρτη και πέμπτη θύρα μπορούν να χρησιμοποιηθούν και για παράλληλη επικοινωνία.

Ακόμη, διαθέτει τρεις μετρητές χρόνου, που του δίνουν μεγάλες δυνατότητες σε εφαρμογές όπου οι πολλαπλές μετρήσεις χρόνου είναι απαραίτητες. Παράλληλα, είναι εφικτή η παραγωγή παλμών ελεγχόμενης διάρκειας, κάτι που είναι πολύ χρήσιμο για την παραγωγή ρυθμιζόμενης συνεχούς τάσης.

Επιπροσθέτως, ο μικροελεγκτής παρέχει έξοδο παλμοσειράς με ρυθμιζόμενο εύρος (PWM). Αυτή μπορεί να χρησιμοποιηθεί για την ρύθμιση διαφόρων βιομηχανικών συσκευών (π.χ. βαλβίδες).

Στον μικροελεγκτή μας δεν λείπει και η δυνατότητα σειριακής επικοινωνίας. Μάλιστα, διαθέτει δύο περιφερειακά: Ένα για σύγχρονη ή ασύγχρονη επικοινωνία, του γνωστού

μας τύπου USART και ένα για σύγχρονη, μόνον, επικοινωνία, το οποίο ονομάζεται SSP (Synchronous Serial Port).

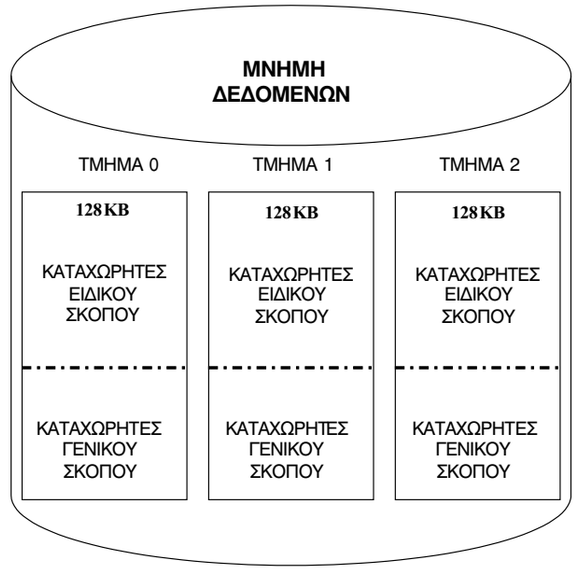
Επιπλέον, διαθέτει ένα μετατροπέα αναλογικού σήματος σε ψηφιακό. Ο μετατροπέας αυτός είναι οκτώ καναλιών, που σημαίνει ότι έχει τη δυνατότητα να λαμβάνει και μετατρέπει οκτώ διαφορετικά αναλογικά σήματα. Το αποτέλεσμα της μετατροπής, για κάθε κανάλι, είναι ένας αριθμός 8 bit.

5.2 Καταχωρητές

Έχουμε δει σε προηγούμενα μαθήματα, τι είναι ένας καταχωρητής και πώς αυτός υλοποιείται με λογικές πύλες. Επίσης, έχουμε επανειλημμένα τονίσει ότι ένας καταχωρητής δεν είναι τίποτε περισσότερο από μία στοιχειώδη μνήμη. Ακόμη, κάναμε μία σύντομη αναφορά στο θέμα αυτό και στην αρχή του μαθήματός μας. Ήρθε, λοιπόν, η στιγμή να δούμε την εφαρμογή όλων αυτών σε έναν πραγματικό μικροελεγκτή.

Οι καταχωρητές είναι ένα από τα βασικότερα στοιχεία της αρχιτεκτονικής ενός *μικροελεγκτή*. Η ευκολία και οι δυνατότητες προγραμματισμού του μικροελεγκτή έχουν άμεση σχέση με το πλήθος, το είδος και τις δυνατότητές των καταχωρητών του. Κάθε εντολή ενός προγράμματος χρησιμοποιεί έναν τουλάχιστον καταχωρητή.

Ο μικροελεγκτής PIC χρησιμοποιεί δύο ομάδες καταχωρητών, τις οποίες μπορούμε να αναζητήσουμε στη μνήμη δεδομένων του PIC, όπως φαίνεται στο σχήμα 5.3.



Σχήμα 5.3 *Μνήμη δεδομένων, τριών τμημάτων, μικροεπεξεργαστή PIC.*

Η πρώτη ομάδα, που βρίσκεται στις χαμηλότερες διευθύνσεις, περιέχει καταχωρητές ειδικών λειτουργιών (special function registers), όπως αυτών του ελέγχου των περιφερειακών που βρίσκονται ενσωματωμένα στον μικροελεγκτή. Η δεύτερη ομάδα περιέχει καταχωρητές γενικής χρήσης και αναφέρεται ως αρχείο καταχωρητών γενικού σκοπού (general purpose register file). Πέραν αυτών, ο μικροελεγκτής διαθέτει τους καταχωρητές W και PC, στον οποίο αναφερθήκαμε στην αρχή του κεφαλαίου.

Ο καταχωρητής W λέγεται και καταχωρητής εργασίας. Είναι ανεξάρτητος από τους υπόλοιπους και βρίσκεται άμεσα συνδεδεμένος με την αριθμητική και λογική μονάδα του PIC. Αυτό του δίνει κάποια μοναδικά πλεονεκτήματα, με αποτέλεσμα να είναι απαραίτητος για την εκτέλεση κάποιων εντολών. Για παράδειγμα, μπορούμε να δώσουμε εντολή πρόσθεσης δύο καταχωρητών, μόνον αν ο ένας από αυτούς είναι ο W.

Ο μετρητής προγράμματος, PC, είναι κοινό στοιχείο της αρχιτεκτονικής όλων των μικροεπεξεργαστών και μικροελεγκτών. Στην ουσία, είναι ο μόνος τρόπος με τον οποίο η Κ.Μ.Ε. μπορεί να βρει στην μνήμη την επόμενη εντολή που πρέπει να εκτελέσει. Οι εντολές ενός προγράμματος, συνήθως, εκτελούνται με την σειρά που βρίσκονται αποθηκευμένες στην μνήμη. Ο PC, σε κάθε εντολή που πρόκειται να εκτελεσθεί, αυξάνεται κατά 1, έτσι ώστε να έχει τη διεύθυνση της επόμενης εντολής αυτής. Όπως θα δούμε και στη συνέχεια, αυτό δεν συμβαίνει πάντα.

Πολλές φορές είναι αναγκαίο να εκτελεσθεί μία εντολή που βρίσκεται αποθηκευμένη αρκετές θέσεις μνήμης μακριά από την τελευταία εντολή που εκτελέσθηκε. Αυτό επιτυγχάνεται με την εκτέλεση μίας εντολής άλματος, η οποία αλλάζει τη ροή εκτέλεσης του προγράμματος. Τότε, στον PC καταχωρείται η διεύθυνση της εντολής που πρέπει να εκτελεσθεί.

Ο μετρητής προγράμματος του PIC έχει μήκος 13 bit. Άρα, μπορούν να παρασταθούν 213 αριθμοί, δηλαδή από 0 έως 8191. Αυτοί οι αριθμοί αντιπροσωπεύουν τις αντίστοιχες διευθύνσεις στην μνήμη προγράμματος. Συνεπώς, ο PIC μπορεί έχει μέχρι 8KB μνήμης προγράμματος, όπως άλλωστε είπαμε και στην ενότητα 5.1.3.

Πολύ σημαντικός, για τα προγράμματά μας, είναι και ο καταχωρητής STATUS, ο οποίος ανήκει στην κατηγορία των καταχωρητών ειδικών λειτουργιών. Αυτόν τον καταχωρητή μπορούμε να τον δούμε στο σχήμα 5.4.

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

Σχήμα 5.4 *Ο καταχωρητής STATUS.*

Τα bits του καταχωρητή που θα μας απασχολήσουν στο μάθημά μας είναι τα 6, 5, 2, 1 και 0, ενώ δε θα ασχοληθούμε με τα υπόλοιπα. Ας δούμε, λοιπόν, με την σειρά τα πέντε αυτά bits:

- **RP1 - RP0**: Bits επιλογής τμήματος (128 Bytes) μνήμης δεδομένων.
  00 = Τμήμα 0 (0 - 7F H),
  01 = Τμήμα 1 (80 H - FF H)
  10 = Τμήμα 2 (100 H - 17F H)
  11 = Δε χρησιμοποιείται
- **Z**: Bit ένδειξης μηδενισμού αποτελέσματος.
  1 = Το αποτέλεσμα μίας αριθμητικής ή λογικής πράξης είναι 0. 0 = Το αποτέλεσμα μίας αριθμητικής ή λογικής πράξης δεν
  είναι 0.
- **DC**: Bit ένδειξης ενδιάμεσου κρατούμενου για τις αριθμητικές πράξεις πρόσθεσης και αφαίρεσης.
  1 = ΄Υπαρξη κρατουμένου, που παράγεται από το 4ο χαμηλότερο bit του αποτελέσματος. 0 = Δεν υπάρχει κρατούμενο, που να παράγεται από το 4ο χαμηλότερο bit του αποτελέσματος.
- **C**: Bit ένδειξης κρατούμενου για τις αριθμητικές πράξεις πρόσθεσης και αφαίρεσης.
  1 = ΄Υπαρξη κρατουμένου, που παράγεται από το περισσότερο
  σημαντικό bit του αποτελέσματος. 0 = Δεν υπάρχει κρατούμενο, που να παράγεται από το περισσότερο σημαντικό bit του αποτελέσματος.
  Σημείωση: Το bit αυτό χρησιμοποιείται και στις εντολές περιστροφής καταχωρητή, όπως θα δούμε στη συνέχεια.

Τα τρία τελευταία bit μας δηλώνουν μία κατάσταση και, για το λόγο αυτό, τα ονομάζουμε και σημαίες. Για παράδειγμα, το bit C λέγεται και σημαία κρατούμενου. Πολλές εντολές, όταν εκτελεσθούν, μαζί με τη λειτουργία που εκτελούν, επηρεάζουν και τις σημαίες αυτές. Περισσότερα, για τις σημαίες, θα δούμε στις επόμενες ενότητες, όταν θα εξετάσουμε τις εντολές του PIC.

5.3 Τύποι εντολών του PIC

Ο μικροελεγκτής PIC έχει εντολές μήκους μίας λέξης. Μία λέξη, στην προκειμένη περίπτωση, αποτελείται από δεκατέσσερα (14) δυαδικά ψηφία. Η δομή της λέξης διαφέρει από εντολή σε εντολή. Σε όλες τις λέξεις, το πρώτο τμήμα περιέχει τον κωδικό της εντολής (OPCODE), ενώ το υπόλοιπο περιέχει πληροφορίες για την εκτέλεση της εντολής. Αυτός κωδικός είναι καθορισμένος από την αρχιτεκτονική του συστήματος, είναι μοναδικός για κάθε εντολή και δεν μπορεί να αλλαχθεί. Η Κ.Μ.Ε., διαβάζοντάς τον κωδικό μίας εντολής, γνωρίζει, επακριβώς, τις εργασίες που πρέπει να εκτελέσει, ενώ οι απαιτούμενες, για την εκτέλεσή τους, πληροφορίες, βρίσκονται στο υπόλοιπο τμήμα της λέξης. Τους κωδικούς των εντολών ενός μικροεπεξεργαστή ή μικροελεγκτή, μπορούμε να τους αναζητήσουμε στο εγχειρίδιό του.

Οι εντολές του PIC χωρίζονται σε τέσσερις κατηγορίες, ως εξής:

- Εντολές επεξεργασίας byte (byte - oriented)
- Εντολές επεξεργασίας bit (bit - oriented)
- Εντολές άλματος (αλλαγής ροής προγράμματος)
- Λοιπές εντολές

Στην κάθε κατηγορία αντιστοιχεί μία συγκεκριμένη δομή. Στις δύο πρώτες, η εντολή χωρίζεται σε τρία τμήματα ενώ, στις άλλες δύο, σε δύο.

Γενικά, ο κωδικός των εντολών έχει μήκος έξι (6) δυαδικά ψηφία (bits). Εξαίρεση αποτελεί ο κωδικός των εντολών της κατηγορίας επεξεργασίας bit που έχει μήκος τέσσερα (4), καθώς επίσης και των εντολών αλλαγής ροής προγράμματος, που έχει τρία (3). Το σχήμα 5.5 παρουσιάζει τον τρόπο με τον οποίο χωρίζεται, κατά περίπτωση, μία λέξη εντολής.

13 12 11 10 9 8	7	4 6 5 3 2 1 0
Κωδικός εντολής	d	Διεύθυνση καταχωρητή στην μνήμη δεδομένων

(α) Εντολές επεξεργασίας Byte (Το d είναι ψηφίο επιλογής καταχωρητή για την αποθήκευση του αποτελέσματος)

13 12 11 10	9 8 7	4 6 5 3 2 1 0
Κωδικός εντολής	Αριθμός Ψηφίου	Διεύθυνση καταχωρητή στην μνήμη δεδομένων

(β) Εντολές επεξεργασίας bit

13 12 11	10 9 8 7	4 6 5 3 2 1 0
Κωδικός εντολής		Διεύθυνση

(γ) Εντολές Άλματος

13 12 11	10 9 8 7	4 6 5 3 2 1 0
Κωδικός εντολής		Δεδομένο

(δ) Λοιπές εντολές

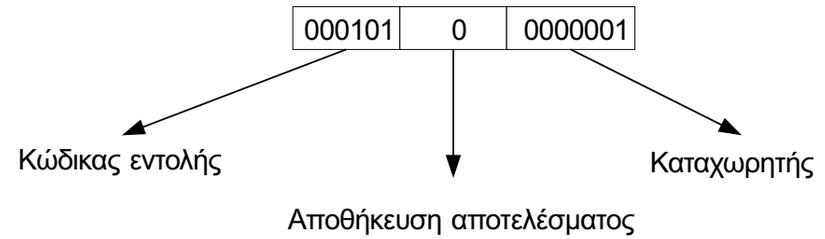
Σχήμα 5.5 *Δομή εντολών μικροεπεξεργαστή PIC*

Στη συνέχεια θα δούμε ένα παράδειγμα εντολής, το οποίο θα μας βοηθήσει να καταλάβουμε, καλύτερα, πώς πραγματικά δομείται μία εντολή.

Έστω, ότι θέλουμε να εκτελέσουμε τη λογική πράξη ΚΑΙ (AND) μεταξύ δύο καταχωρητών. Ο πρώτος είναι ο καταχωρητής W και ο δεύτερος ο καταχωρητής που βρίσκεται στην διεύθυνση 1 της μνήμης δεδομένων. Ο δεύτερος ονομάζεται TMR0 και θα μιλήσουμε για αυτόν στο επόμενο κεφάλαιο, όταν θα δούμε τις περιφερειακές μονάδες του PIC. Το αποτέλεσμα, της λογικής πράξης, θέλουμε να το αποθηκεύσουμε

σε έναν από τους δύο καταχωρητές, π.χ. στον W.

Η κατάλληλη εντολή ανήκει στην κατηγορία εντολών επεξεργασίας Byte. Όπως φαίνεται στο σχήμα 5.5(α), οι εντολές αυτού του τύπου χωρίζονται σε τρία τμήματα. Άρα, η λέξη της εντολής που θέλουμε, έχει τον κωδικό και δύο ορίσματα. Ο κωδικός της έχει μήκος 6 bit και είναι ο 000101. Το πρώτο όρισμα έχει μήκος 1 bit. Με αυτό μπορούμε να επιλέξουμε σε ποιόν από τους δύο καταχωρητές θα αποθηκευτεί το αποτέλεσμα. Με 0 το αποτέλεσμα αποθηκεύεται στο καταχωρητή W, ενώ με 1 στον άλλον. Το δεύτερο όρισμα έχει μήκος 7 bit και προσδιορίζει τον καταχωρητή TMR0, χρησιμοποιώντας τη διεύθυνσή του στη μνήμη δεδομένων. Εδώ, ας θυμηθούμε ότι ο PIC διαθέτει 128 καταχωρητές άρα, το μήκος των 7 bit επαρκεί για να αποθηκεύσει την διεύθυνση των καταχωρητών. Συνεπώς, η λέξη της εντολής είναι η:



Επειδή είναι πολύ δύσκολο να γράψουμε ή να διαβάσουμε ένα πρόγραμμα με εντολές στην παραπάνω μορφή, χρησιμοποιούμε μία συμβολική γραφή. Για παράδειγμα, η παραπάνω εντολή συμβολίζεται ως εξής:

ANDWF TMR0, 0

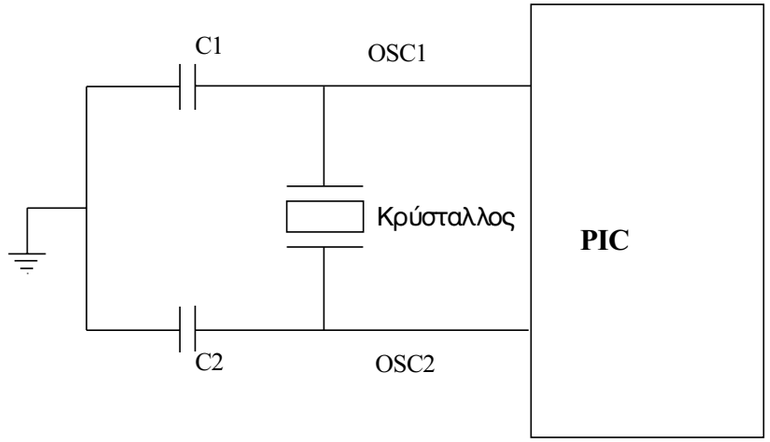
Ο συμβολισμός ANDWF αντιστοιχεί στον κωδικό 000101, ενώ ο καταχωρητής TMR0 βρίσκεται στη διεύθυνση 0000001 της μνήμης δεδομένων και το μηδέν δηλώνει που θα αποθηκευτεί το αποτέλεσμα.

Με παρόμοιο τρόπο σχηματίζονται και οι άλλες εντολές του PIC. Για παράδειγμα η εντολή 01010100000001 θέτει το τρίτο σημαντικό bit του καταχωρητή TMR0, στο 1. Η εντολή αυτή συμβολίζεται ως BSF TMR0,2. Επίσης, η εντολή 101 10111110011 αλλάζει τη ροή εκτέλεσης του προγράμματος εκτελώντας άλμα στη θέση 5F3 H της μνήμης προγράμματος. Αυτή η εντολή συμβολίζεται ως GOTO 05F3 H.

Είδαμε, λοιπόν, τη μορφή που σχηματίζονται οι εντολές του PIC και πώς ο μικροελεγκτής αναλύει και χρησιμοποιεί την πληροφορία που αυτές περιέχουν. Στη συνέχεια θα μιλήσουμε για το χρόνο που κάνει ο PIC για να εκτελέσει τις εντολές αυτές.

5.4 Κύκλος εκτέλεσης εντολής

Ο μικροελεγκτής διαθέτει εσωτερική μονάδα χρονισμού και παραγωγής παλμών. Είναι αρκετή η σύνδεση ενός εξωτερικού κρυστάλλου και δύο πυκνωτών, για την παραγωγή παλμών ρολογιού. Η συνδεσμολογία παρουσιάζεται στο σχήμα 5.6.



Σχήμα 5.6 *Το κύκλωμα παραγωγής παλμών του μικροελεγκτή PIC.*

Το εξωτερικό κύκλωμα μπορεί να παράγει σήμα ρολογιού από 455kHz ως 20MHz, ανάλογα με τον κρύσταλλο που θα χρησιμοποιηθεί. Ο PIC λαμβάνει τους παλμούς αυτούς στον ακροδέκτη OSC1.

Όπως ξέρουμε, ο μικροελεγκτής εκτελεί μία-μία τις εντολές ενός προγράμματος που υπάρχει στην μνήμη του. Η διαδικασία εκτέλεσης μία εντολής είναι να έρθει η εντολή αυτή από την μνήμη στην Κ.Μ.Ε. και μετά να αποκωδικοποιηθεί και να εκτελεσθεί. Ο μικροελεγκτής αρχίζει τη διαδικασία αυξάνοντας τον καταχωρητή PC κατά 1 και, αμέσως μετά, ανακαλεί την εντολή από τη θέση της μνήμης προγράμματος που αυτός υποδεικνύει. Για όλες τις εντολές του PIC, αυτή η διαδικασία διαρκεί έναν κύκλο εντολής.

Μόνη εξαίρεση αποτελούν οι εντολές άλματος, που αλλάζουν τη ροή του προγράμματος, δηλαδή, τον καταχωρητή PC. Στην περίπτωση αυτή, η επόμενη εντολή που πρέπει να ανακληθεί δεν είναι αυτή που περιμένει στη σειρά. Για να πάρει ο PC την σωστή τιμή, ώστε να υποδείξει την επόμενη εντολή που πρέπει να εκτελεσθεί, χρειάζεται και έναν ακόμη κύκλο εντολής. Έτσι οι εντολές άλματος χρειάζονται δύο κύκλους εντολής για να εκτελεσθούν.

Για να καταλάβουμε καλύτερα την όλη διαδικασία, ας δούμε σε πόσο χρόνο εκτελούνται οι εντολές του επομένου προγράμματος:

05F0 H: ADDWF TMR0, 0
05F1 H: GOTO 05F3 H
05F2 H: BSF TMR0, 1
05F3 H: BSF TMR0, 2
05F4 H: ……..
05F5 H: ……..
05F5 H: ……..

Όλες οι παραπάνω εντολές εκτελούνται σε έναν κύκλο εντολής, εκτός από την εντολή άλματος, GOTO 05F3 H, που εκτελείται σε δύο κύκλους. Άρα, η εκτέλεση του όλου προγράμματος διαρκεί 4 κύκλους. Ας προσέξουμε ότι οι εντολές που εκτελέσθηκαν ήταν 3, αφού η εντολή BSF TMR0, 1 δεν εκτελείται, δεδομένου ότι μετά την εντολή άλματος εκτελείται η εντολή BSF TMR0, 2.

Γνωρίζοντας πόσους κύκλους εντολής χρειάζεται η κάθε εντολή για να ανακληθεί από την μνήμη και να εκτελεσθεί, μπορούμε να υπολογίσουμε τους συνολικούς κύκλους που χρειάζεται έναν πρόγραμμα ή μία ρουτίνα για να εκτελεσθεί. Την χρονική διάρκεια ενός κύκλου εντολής μπορούμε να την υπολογίσουμε αν γνωρίζουμε την συχνότητα λειτουργίας του PIC, σύμφωνα με την επόμενη σχέση:

Διάρκεια κύκλου εντολής = 4 / (Συχν. Λειτουργίας PIC)

Συνεπώς, αν υποθέσουμε ότι ο PIC, στο παραπάνω παράδειγμα, έχει συχνότητα λειτουργίας 20MHz τότε η διάρκεια του κάθε κύκλου εντολής είναι 0,2μsec. Άρα, η εκτέλεση του όλου προγράμματος θα διαρκέσει 0,8μsec.

5.5 Η γλώσσα Assembly του PIC

Στην ενότητα 5.4 είδαμε ότι ο PIC, όπως και κάθε άλλος μικροελεγκτής και μικροεπεξεργαστής, ανακαλεί από τη μνήμη μία σειρά λέξεων που εμπεριέχουν κωδικούς εντολών και πληροφορίες σε δυαδική μορφή. Αυτή η σειρά λέξεων είναι, στην πραγματικότητα, το πρόγραμμα που εκτελεί Έτσι, το πρόγραμμα, που εξετάσαμε στην προηγούμενη ενότητα, είναι γραμμένο στη μνήμη προγράμματος, αρχίζοντας από τη θέση 05F0 H, ως εξής

05F0 H: 00010100000001
05F1 H: 10110111110011
05F2 H: 01010010000001
05F3 H: 01010110000001
05F4 H: ……..
05F5 H: ……..
05F5 H: ……..

Είναι δυνατό να γράψουμε απευθείας ένα πρόγραμμα στη μνήμη προγράμματος στην παραπάνω μορφή. Άλλωστε, αυτός είναι ο τρόπος που γράφονταν τα προγράμματα όταν η εξέλιξη των ηλεκτρονικών υπολογιστών ήταν ακόμη σε αρχικό στάδιο. Ένα πρόγραμμα στην παραπάνω μορφή λέμε ότι είναι γραμμένο σε γλώσσα μηχανής. Βέβαια, γίνεται αμέσως αντιληπτό πόσο δύσκολο είναι να γράψουμε ένα πρόγραμμα, εκατό και πλέον εντολών, στη μορφή αυτή. Μάλιστα, ακόμη δυσκολότερο είναι να το διορθώσουμε στην περίπτωση που εντοπίσουμε κάποιο σφάλμα, κατά τη διάρκεια της εκτέλεσής του.

Για το λόγο αυτό, σχεδιάστηκε μία γλώσσα που οι εντολές της να αντιστοιχούν άμεσα σε αυτές του μικροελεγκτή. Στην πραγματικότητα πρόκειται για μία απεικόνιση των εντολών του μικροελεγκτή σε μία εύληπτη μορφή. Η γλώσσα αυτή ονομάστηκε Assembly. Μία πρώτη γνωριμία με την συγκεκριμένη γλώσσα κάναμε στα παραδείγματα των δύο προηγουμένων ενοτήτων. Εκεί, είδαμε πώς συμβολίζονται οι εντολές που χρησιμοποιήσαμε στη γλώσσα αυτή.

Όμως, πριν ξεκινήσουμε να βλέπουμε τις εντολές της γλώσσας Assembly του μικροελεγκτή PIC, ας μιλήσουμε, λίγο, για τα πλεονεκτήματα και τα μειονεκτήματά της σε σχέση με τις άλλες γλώσσες προγραμματισμού. Η Assembly, επειδή είναι άμεση συνέχεια της γλώσσας μηχανής μας δίνει τη δυνατότητα να επέμβουμε απευθείας στον PIC και να τον ελέγξουμε πλήρως και, μάλιστα, σε επίπεδο μηχανής. Επίσης, όπως είδαμε και στην προηγούμενη ενότητα, μπορούμε να υπολογίσουμε με ακρίβεια τον χρόνο εκτέλεσης ενός προγράμματος ή ενός μέρος αυτού. Αυτά είναι τα χαρακτηριστικά που κάνουν τη γλώσσα αναντικατάστατη στο χώρο της ανάπτυξης μικροϋπολογιστικών συστημάτων. Ωστόσο, η γλώσσα Assembly μειονεκτεί σε θέματα ευκολίας προγραμματισμού. Για παράδειγμα, είναι αρκετά δύσκολο να χειριστούμε αρχεία και, ακόμη δυσκολότερο, να φτιάξουμε μία βάση δεδομένων. Επιπρόσθετα, η γλώσσα Assembly, είναι διαφορετική για κάθε μικροελεγκτή ή μικροεπεξεργαστή, αφού έχει άμεση σχέση με τη γλώσσα μηχανής του.

Σε αντίθεση, οι γλώσσες υψηλότερου επιπέδου μας δίνουν τη δυνατότητα να γράψουμε σύνθετα προγράμματα με μεγαλύτερη ευκολία. Για παράδειγμα, μία εντολή της γλώσσας Basic μπορεί να αντιστοιχεί σε πολλές εντολές της Assembly. Επίσης, ο κώδικας που θα γραφεί σε μία τέτοια γλώσσα, μπορεί να λειτουργήσει σε μικροϋπολογιστικά συστήματα βασισμένα σε διάφορους μικροεπεξεργαστές ή μικροελεγκτές. Φυσικά, είναι αδύνατο να μετρήσουμε, με ακρίβεια, το χρόνο εκτέλεσης ενός προγράμματος γραμμένου σε γλώσσα υψηλού επιπέδου. Επίσης, είναι πολύ δύσκολο να επέμβουμε άμεσα στα διάφορα περιφερειακά του μικροελεγκτή.

Σε μικροϋπολογιστικά συστήματα που χρησιμοποιούνται σε μεγάλες εφαρμογές ακολουθείται μία μέση οδός. Το μεγαλύτερο μέρος της εφαρμογής γράφεται σε γλώσσα C, ενώ μέρος της, όπου αυτό είναι απαραίτητο, γράφεται σε γλώσσα Assembly. Αυτό γίνεται γιατί η γλώσσα C, όπως και άλλες, δέχεται εμφύτευση κώδικα Assembly.

Αφού είπαμε μερικά γενικά στοιχεία για τη γλώσσα Assembly, είναι καιρός, πλέον, να προχωρήσουμε και στις εντολές της. Πρέπει να σημειώσουμε ότι δε θα ακολουθήσουμε την κατηγοριοποίηση των εντολών που κάναμε όταν μιλούσαμε για τη μορφή τους. Στην παρούσα ενότητα μας ενδιαφέρει περισσότερο η πρακτική χρήση των εντολών στον προγραμματισμό του PIC. Έτσι, λοιπόν, θα τροποποιήσουμε λίγο τις κατηγορίες αυτές και θα παρουσιάσουμε τις εντολές ως ακολούθως:

- Εντολές αλλαγής περιεχομένου καταχωρητή

- Εντολές αύξησης / μείωσης τιμής καταχωρητή
- Εντολές αριθμητικών πράξεων
- Εντολές λογικών πράξεων
- Εντολές επεξεργασίας bit
- Εντολές άλματος (αλλαγής ροής προγράμματος)

Στις εντολές που θα παρουσιάσουμε θα αναφέρουμε διαδοχικά τη σύνταξή τους, πώς ορίζονται τα ορίσματά τους, τη λειτουργία που εκτελούν σε μορφή συνάρτησης και μία σύντομη περιγραφή τους. Επίσης, θα χρειαστούμε τον πίνακα 5.1 που δείχνει τα ονόματα των καταχωρητών και την διεύθυνσή τους στη μνήμη δεδομένων. Ας δούμε, τώρα, με την σειρά τις εντολές αυτές.

5.5.1 Εντολές αλλαγής περιεχομένου καταχωρητή

Όλες οι εντολές αναφέρονται, άμεσα ή έμμεσα, σε κάποιον καταχωρητή. Προφανώς, είναι σημαντικό, πριν μιλήσουμε για οποιαδήποτε άλλη εντολή, να ξέρουμε πώς μπορούμε να φορτώσουμε μία συγκεκριμένη τιμή σε έναν καταχωρητή, καθώς, επίσης, και πώς να τον καθαρίσουμε. Οι πρώτες, λοιπόν, εντολές που θα δούμε, στον PIC, είναι αυτές της φόρτωσης ενός καταχωρητή με μία τιμή που θέλουμε καθώς και του μηδενισμού του.

• MOVLW Δ

Η πρώτη εντολή που θα δούμε είναι αυτή της απευθείας φόρτωσης του καταχωρητή W με μία τιμή. Ο καταχωρητής W χρησιμοποιείται πολύ στις υπόλοιπες εντολές και είναι, βασικό, να μπορούμε να ορίσουμε την τιμή του εύκολα. Ακολούθως, παραθέτουμε έναν πίνακα με τα συνοπτικά στοιχεία της εντολής.

ΣΥΝΤΑΞΗ	MOVLW Δ
ΟΡΙΣΜΑ	Το Δ παίρνει τιμή από 0 έως 255.
ΣΗΜΑΙΕΣ	ΚΑΜΙΑ

Περιγραφή: Το δεδομένο Δ φορτώνεται στον καταχωρητή W. (σχήμα 5.7)

Σχήμα 5.7 *Φόρτωμα του καταχωρητή W με ένα δεδομένο Δ με την εντολή MOVLW Δ.*

Όπως βλέπουμε το δεδομένο παίρνει τιμές από το 0 ως το 255, που σημαίνει ότι έχει μήκος ένα byte. Αυτό είναι πολύ λογικό αφού φορτώνεται στον καταχωρητή W που έχει και αυτός το ίδιο μήκος. Καμία από τις σημαίες δεν ενημερώνεται.
Ας δούμε και ένα παράδειγμα με την εντολή αυτή.

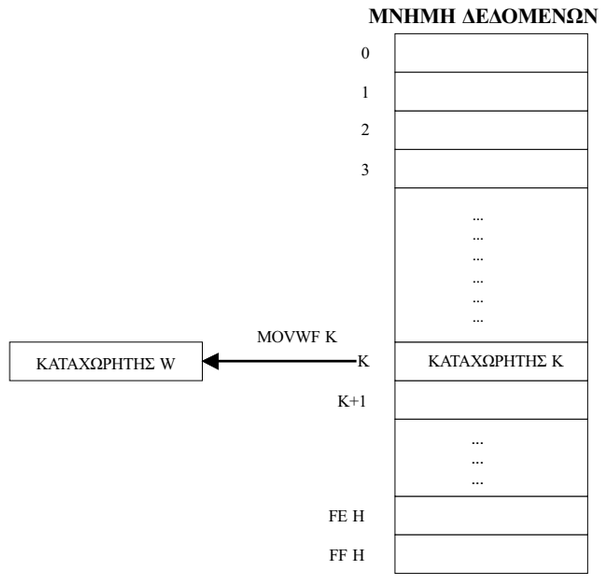
Παράδειγμα
Έστω ότι, αρχικά, ο καταχωρητής W = 7A H και ότι δίνουμε την εντολή MOVLW 35 H. Μετά την εκτέλεσή της ο καταχωρητής θα είναι W = 35 H.

• MOVWF K

Με την προηγούμενη εντολή, μπορούμε να φορτώσουμε μία τιμή μόνο στον καταχωρητή W. Είναι φανερό ότι, συχνά χρειαζόμαστε να δώσουμε κάποια τιμή και σε άλλους καταχωρητές. Ο PIC διαθέτει μία εντολή που φορτώνει σε έναν καταχωρητή το περιεχόμενο του W.

ΣΥΝΤΑΞΗ	MOVWF K Το K παίρνει τιμές από 0
ΟΡΙΣΜΑ	έως 127 ή είναι το συμβολικό όνομα ενός καταχωρητή.
ΣΗΜΑΙΕΣ	ΚΑΜΙΑ

Περιγραφή: Το περιεχόμενο του καταχωρητή W, μεταφέρεται στον καταχωρητή K (σχήμα 5.8).



Σχήμα 5.8 *Φόρτωση των περιεχομένων του καταχωρητή K στον καταχωρητή W με την εντολή MOVWF K.*

Ο καταχωρητής K ορίζεται από τον αριθμό του, ο οποίος παίρνει τιμές από το 0 ως το 127 και αντιστοιχεί στην διεύθυνσή του στην περιοχή της μνήμης δεδομένων. Επειδή, οι καταχωρητές ειδικού σκοπού έχουν και κάποιο συμβολικό όνομα, π.χ. TMR0, αντί της διεύθυνσης 1, μπορούμε να χρησιμοποιήσουμε το όνομα αυτό. Όπως και στην προηγούμενη εντολή μεταφοράς, δεν επηρεάζονται οι σημαίες. Ας δούμε και εδώ ένα παράδειγμα.

Παράδειγμα
Θεωρούμε ότι, αρχικά, ο καταχωρητής W = 16 H και ότι ο καταχωρητής TMR0 = 6A

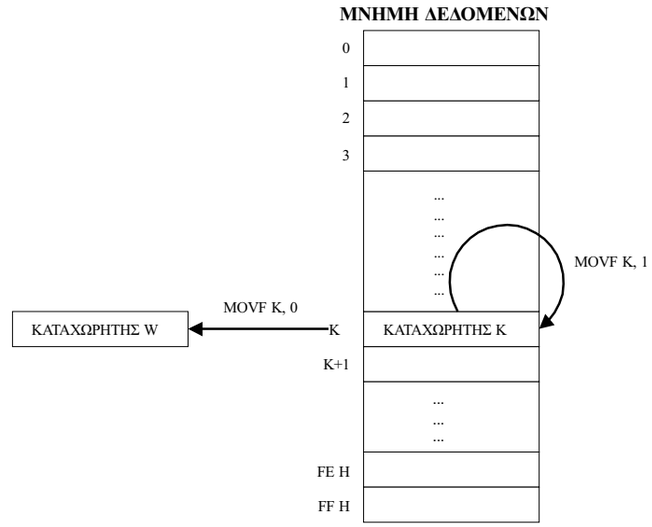
Έστω ότι δίνουμε την εντολή MOVWF TMR0. Μετά την εκτέλεσή της ο καταχωρητής TMR0 = 16 H. Με τη συγκεκριμένη εντολή δεν μπορούμε να φορτώσουμε απευθείας σε έναν καταχωρητή μία συγκεκριμένη τιμή, παρά μόνον του καταχωρητή W. Ωστόσο, αυτό μπορεί να γίνει έμμεσα. Ας υποθέσουμε ότι θέλουμε να φορτώσουμε στον TMR0 την τιμή 23 H. Τότε, μπορούμε, πρώτα, να δώσουμε στον καταχωρητή W την τιμή αυτή και, μετά, να φορτώσουμε στον TMR0 την τιμή του W. Συνεπώς, οι εντολές που πρέπει δώσουμε είναι:
MOVLW 23 H
MOVWF TMR0

• MOVF K, d

Μία ακόμη εντολή, της κατηγορίας αυτής, είναι η φόρτωση του καταχωρητή W με τα περιεχόμενα του καταχωρητή K.

ΣΥΝΤΑΞΗ	MOVF K, d
ΟΡΙΣΜΑΤΑ	• Το K παίρνει τιμές από 0 έως 127 ή είναι το συμβολικό όνομα ενός καταχωρητή. • Το d παίρνει τιμές από 0 έως 127.
ΣΗΜΑΙΑ	Z

Περιγραφή: Το περιεχόμενο του καταχωρητή K, μεταφέρεται στον καταχωρητή W, αν το d = 0, ή στον εαυτό του, αν το d = 1. (σχήμα 5.9)



Σχήμα 5.9 *Φόρτωση των περιεχομένων του καταχωρητή K στον καταχωρητή W με την εντολή MOVF K, 0 ή στον εαυτό του με την εντολή MOVF K, 1.*

Όπως και στην προηγούμενη εντολή, που παρουσιάσαμε, ο καταχωρητής K ορίζεται από τον αριθμό του, ο οποίος παίρνει τιμές από το 0 ως το 127 και αντιστοιχεί στην διεύθυνσή του στην περιοχή της μνήμης δεδομένων ή το συμβολικό του όνομα, αν έχει. Σε αντίθεση με τις άλλες δύο εντολές φόρτωσης, η εντολή αυτή ενημερώνει τη σημαία του μηδενισμού (Z). Σε περίπτωση που το d = 1, ο καταχωρητής φορτώνει το περιεχόμενό του στον εαυτό του. Αυτό, σε συνδυασμό με την σημαία Z, μπορεί να χρησιμεύσει για τον έλεγχο του αν το περιεχόμενο του καταχωρητή είναι 0.

Ας δούμε και δύο παραδείγματα με την εντολή αυτή. Θα χρησιμοποιήσουμε, πάλι, τον καταχωρητή TMR0, που χρησιμοποιήσαμε και προηγουμένως.

1ο Παράδειγμα

Θεωρούμε ότι, αρχικά, ο καταχωρητής W = 26 H και ότι ο καταχωρητής TMR0 = 1A H. Έστω ότι δίνουμε την εντολή MOVF TMR0, 0. Μετά την εκτέλεσή της, ο καταχωρητής W = 1A H, ενώ η σημαία Z = 0, ανεξάρτητα με το τι ήταν προηγουμένως.

2ο Παράδειγμα

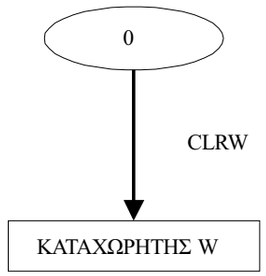
Ας θεωρήσουμε, τώρα, ότι, αρχικά, ο καταχωρητής W = 26 H και ότι ο καταχωρητής TMR0 = 0. Έστω ότι δίνουμε την εντολή MOVF TMR0, 1. Επειδή το d = 1, ο καταχωρητής W δεν αλλάζει αφού ο TMR0 φορτώνεται στον εαυτό του. Επειδή το περιεχόμενό του είναι 0, η σημαία Z γίνεται 1, ανεξάρτητα με το τι ήταν προηγουμένως.

• CLRW

Συχνά, κατά τη διάρκεια ενός προγράμματος χρειάζεται να μηδενίσουμε τον καταχωρητή W. Για την διαδικασία αυτή έχει προβλεφθεί η εντολή αυτή:

ΣΥΝΤΑΞΗ	CLRW
ΟΡΙΣΜΑ	KANENA
ΣΗΜΑΙΑ	Z

Περιγραφή: Ο καταχωρητής W μηδενίζεται και η σημαία Z γίνεται 1 (σχήμα 5.10).



Σχήμα 5.10 *Μηδενισμός καταχωρητή W με την εντολή CLRW.*

Η εντολή δε δέχεται κανένα όρισμα. Αυτό που την διαφοροποιεί από την εντολή "MOVLW 0" είναι ότι με την εκτέλεσή της, η σημαία Z γίνεται 1.
Ακολουθεί ένα παράδειγμα με την εντολή αυτή.

Παράδειγμα

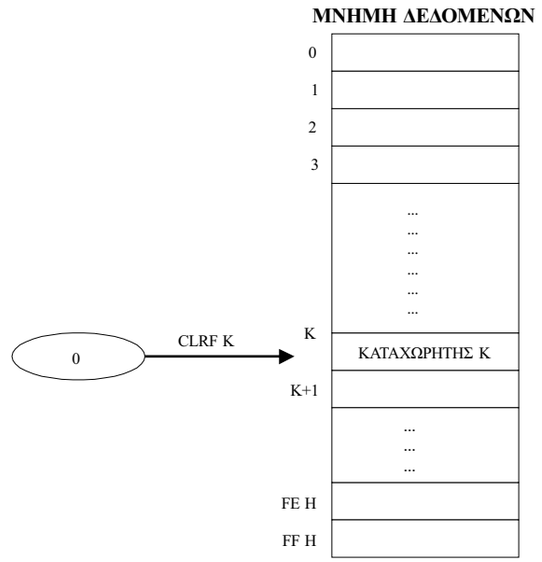
Έστω ότι, αρχικά, ο καταχωρητής W = 71 H και ότι δίνουμε την εντολή CLRW. Μετά την εκτέλεσή της ο καταχωρητής θα είναι W = 0, ενώ το Z = 1, ανεξάρτητα από την τιμή που είχε πριν.

• CLRF K

Η τελευταία εντολή, της κατηγορίας αυτής, είναι του μηδενισμού του περιεχομένου ενός καταχωρητή.

ΣΥΝΤΑΞΗ	CLRF K
ΟΡΙΣΜΑ	Το K παίρνει τιμές από 0 έως 127 ή είναι το συμβολικό όνομα ενός καταχωρητή.
ΣΗΜΑΙΑ	Z

Περιγραφή: Το περιεχόμενο του καταχωρητή K μηδενίζεται και η σημαία Z γίνεται 1 (σχήμα 5.11).



Σχήμα 5.11 *Μηδενισμός καταχωρητή K με την εντολή CLRF K.*

Όπως και με την εντολή μηδενισμού του καταχωρητή W, έτσι και εδώ, ο μηδενισμός του καταχωρητή κάνει την σημαία Z ίση με 1. Ας δούμε, και εδώ, ένα παράδειγμα.

Παράδειγμα

Θεωρούμε ότι ο καταχωρητής TMR0 = C3 H. Έστω ότι δίνουμε την εντολή CLRF TMR0. Μετά την εκτέλεσή της, ο καταχωρητής TMR0 = 0 και η σημαία Z = 1, ανεξάρτητα με το τι ήταν προηγουμένως.

Ερωτήσεις

1. Με ποια εντολή μπορούμε να δώσουμε στον καταχωρητή W την τιμή B9 H;
2. Με ποιες εντολές μπορούμε να δώσουμε στον καταχωρητή TMR0 την τιμή 5;
3. Ποιες από τις παρακάτω τιμές μπορούμε να δώσουμε στον καταχωρητή W και γιατί:

   23 H, 0A H, 5, 50F H, 0FF H, 111 H, 035 H
4. Πώς μπορούμε να φορτώσουμε στον καταχωρητή TMR0 το περιεχόμενο του καταχωρητή W.

5.5.2 Εντολές αύξησης / μείωσης τιμής καταχωρητή

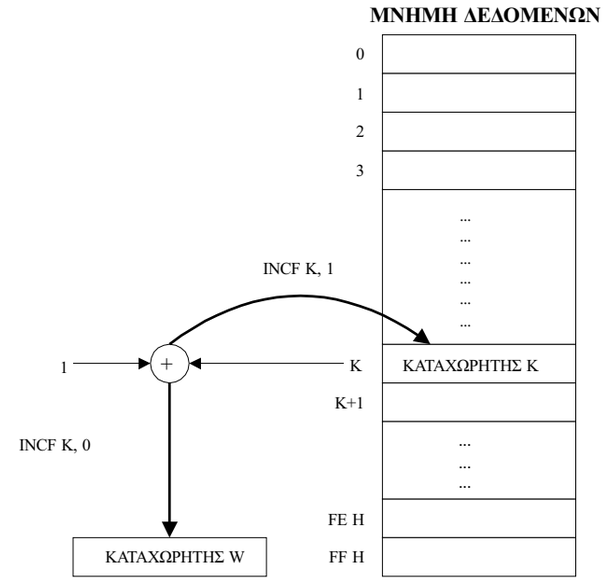
Σε πολλές περιπτώσεις χρειαζόμαστε να αυξήσουμε ή να μειώσουμε την τιμή ενός καταχωρητή κατά 1. Τέτοιες λειτουργίες είναι απαραίτητες σε διαδικασίες μέτρησης επανάληψης βρόχων (loops) μέσα σε ένα πρόγραμμα. Ο PIC δέχεται δύο εντολές αύξησης καταχωρητή και δύο μείωσης.

• INCF K, d

Η πρώτη εντολή, της κατηγορίας αυτής, αυξάνει το περιεχόμενο ενός καταχωρητή K κατά ένα.

ΣΥΝΤΑΞΗ	INCF K, d
ΟΡΙΣΜΑΤΑ	• Το K παίρνει τιμές από 0 έως 127 ή είναι το συμβολικό όνομα ενός καταχωρητή. • Το d παίρνει τιμές από 0 έως 127.
ΣΗΜΑΙΑ	Z

Περιγραφή: Το περιεχόμενο του καταχωρητή K, αυξάνεται κατά 1 και το αποτέλεσμα αποθηκεύεται στον καταχωρητή W, αν το d = 0, ή στον καταχωρητή K, αν το d = 1 (σχήμα 5.12).



Σχήμα 5.12 *Αύξηση του περιεχομένου του καταχωρητή K κατά 1 και αποθήκευση του αποτελέσματος στον καταχωρητή W με την εντολή INCF K, 0 ή στον καταχωρητή K με την εντολή INCF K, 1.*

Και στην περίπτωση αυτή, ο καταχωρητής K ορίζεται από τον αριθμό του, ο οποίος παίρνει τιμές από το 0 ως το 127 και αντιστοιχεί στην διεύθυνσή του στην περιοχή της μνήμης δεδομένων ή το συμβολικό του όνομα, αν έχει. Η παρούσα εντολή επηρεάζει μόνον την σημαία του μηδενισμού (Z). Ας δούμε τώρα και δύο παραδείγματα με την εντολή αυτή.

1ο Παράδειγμα

Θεωρούμε ότι, αρχικά, ο καταχωρητής TMR0 = 16 H και ο W = 8A H.
Αν δώσουμε την εντολή:
INCF TMR0, 1
Τότε οι καταχωρητές,
TMR0 = 17 H (ο καταχωρητής επηρεάζεται, αφού d = 1) και
W = 8A H (ο καταχωρητής δεν επηρεάζεται, αφού d = 1)
Ενώ η σημαία,
Z = 0 (αφού το αποτέλεσμα δεν είναι μηδέν), ανεξάρτητα με το τι ήταν στην αρχή.

2ο Παράδειγμα

Θεωρούμε ότι, αρχικά, ο καταχωρητής TMR0 = FF H και ο W = 8A H.
Αν δώσουμε την εντολή:
INCF TMR0, 0

Τότε οι καταχωρητές,
TMR0 = FF H (ο καταχωρητής δεν επηρεάζεται, αφού d = 0) και
W = 0 (ο καταχωρητής επηρεάζεται, αφού d = 0)
Ενώ η σημαία,
Z = 1 (αφού το αποτέλεσμα είναι μηδέν), ανεξάρτητα με το τι ήταν στην αρχή.

• INCFSZ K, d

Αυτή η εντολή είναι μία παραλλαγή της προηγούμενης. Εδώ, η εντολή εκτελείται όπως ακριβώς η προηγούμενη. Η διαφορά έγκειται στην επόμενη εντολή που είναι να εκτελεστεί. Αν το αποτέλεσμα είναι μηδενικό, τότε δεν εκτελείται η πρώτη σε σειρά εντολή αλλά η αμέσως επόμενη από αυτήν.

ΣΥΝΤΑΞΗ	INCFSZ K, d
ΟΡΙΣΜΑΤΑ	• Το K παίρνει τιμές από 0 έως 127 ή είναι το συμβολικό όνομα ενός καταχωρητή. • Το d παίρνει τιμές από 0 έως 127.
ΣΗΜΑΙΑ	ΚΑΜΙΑ

Περιγραφή: Ο καταχωρητής K, αυξάνεται κατά 1 και το αποτέλεσμα αποθηκεύεται στον W, αν το d = 0, ή στον καταχωρητή K, αν το d = 1. Αν το αποτέλεσμα της αύξησης είναι μηδενικό, η αμέσως επόμενη εντολή παρακάμπτεται.

Αν προσέξουμε λίγο περισσότερο τη διαδικασία θα παρατηρήσουμε ότι η εντολή όταν το αποτέλεσμα της αύξησης είναι μηδενικό επεμβαίνει στην τιμή του PC. Τότε, σύμφωνα με τα όσα είπαμε στην ενότητα 5.5 για τον χρόνο εκτέλεσης μίας εντολής, η εντολή χρειάζεται 2 κύκλους. Ας δούμε τώρα και ένα μικρό πρόγραμμα ως παράδειγμα για την περίπτωση αυτή.

Παράδειγμα
΄Εστω ότι έχουμε το ακόλουθο πρόγραμμα:

```
INCFSZ TMR0, 1
MOVLW 23 H
INCF TMR0, 1
```

Ας θεωρήσουμε ότι, αρχικά, ο καταχωρητής TMR0 = FF H και ο W = 12 H. Τότε, μετά την εκτέλεση της κάθε εντολής , οι καταχωρητές θα έχουν τις τιμές:

Εντολή: INCFSZ TMR0, 1
W = 12 H,

TMR0 = 0
Κύκλοι εκτέλεσης εντολής: 2

Εντολή: MOVLW 23 H
Δεν εκτελείται.

Εντολή: INCF TMR0, 1
W = 12 H,
TMR0 = 1
Κύκλοι εκτέλεσης εντολής: 1

Επειδή, το αποτέλεσμα της πρώτης εντολής είναι 0, αμέσως μετά εκτελείται η τρίτη εντολή ενώ η δεύτερη παρακάμπτεται.
Τώρα, ας θεωρήσουμε ότι, αρχικά, ο καταχωρητής TMR0 = A5 H και ο W = 12 H.
Τότε, μετά την εκτέλεσή του, οι καταχωρητές θα έχουν τις τιμές:

Εντολή: INCFSZ TMR0, 1
W = 12 H,
TMR0 = 1
Κύκλοι εκτέλεσης εντολής: 1

Εντολή: MOVLW 23 H
W = 23 H,
TMR0 = 1
Κύκλοι εκτέλεσης εντολής: 1

Εντολή: INCF TMR0, 1
W = 23 H,
TMR0 = 2
Κύκλοι εκτέλεσης εντολής: 1

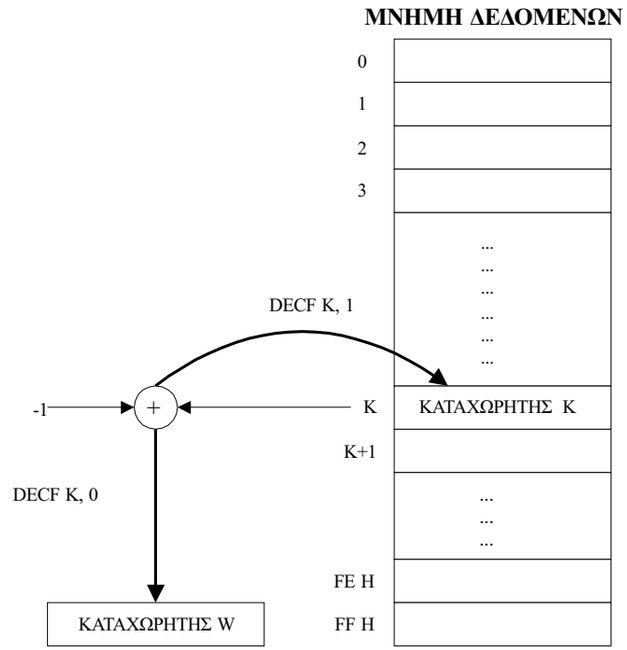
Ας προσέξουμε ότι, και στις δύο περιπτώσεις, το πρόγραμμα εκτελείται σε 3 κύκλους εντολής.

• DECF K, d

Πρόκειται για εντολή μείωσης περιεχομένου καταχωρητή. Είναι η αντίστοιχη της εντολής INCF, την οποία αναλύσαμε, λεπτομερώς, προηγουμένως.

ΣΥΝΤΑΞΗ	DECF K, d
ΟΡΙΣΜΑΤΑ	• Το K παίρνει τιμές από 0 έως 127 ή είναι το συμβολικό όνομα ενός καταχωρητή. • Το d παίρνει τιμές από 0 έως 127.
ΣΗΜΑΙΑ	Z

Περιγραφή: Ο καταχωρητής K, μειώνεται κατά 1 και το αποτέλεσμα αποθηκεύεται στον W, αν το d = 0, ή στον καταχωρητή K, αν το d = 1 (σχήμα 5.13).



Σχήμα 5.13 *Μείωση του περιεχομένου του καταχωρητή K κατά 1 και αποθήκευση του αποτελέσματος στον καταχωρητή W με την εντολή DECF K, 0 ή στον καταχωρητή K με την εντολή DECF K, 1.*

Δε θα επεκταθούμε περισσότερο, αφού η μόνη της διαφορά από την αντίστοιχή της είναι ότι αντί να αυξάνει, μειώνει κατά 1.

• DECFSZ K, d

Παρόμοια με την πρώτη και η δεύτερη εντολή μείωσης περιεχομένου καταχωρητή είναι η αντίστοιχη της εντολής INCFSZ, για την οποία μιλήσαμε προηγουμένως.

ΣΥΝΤΑΞΗ	DECFSZ K, d
ΟΡΙΣΜΑΤΑ	• Το K παίρνει τιμές από 0 έως 127 ή είναι το συμβολικό όνομα ενός καταχωρητή. • Το d παίρνει τιμές από 0 έως 127.
ΣΗΜΑΙΑ	ΚΑΜΙΑ

Περιγραφή: Το περιεχόμενο του καταχωρητή K, μειώνεται κατά 1 και το αποτέλεσμα αποθηκεύεται στον W, αν το d = 0, ή στον καταχωρητή K, αν το d = 1. Αν το αποτέλεσμα της μείωσης είναι μηδέν η αμέσως επόμενη εντολή παρακάμπτεται.

Ούτε εδώ θα επεκταθούμε περισσότερο. Όπως και στην προηγούμενη εντολή, η μόνη της διαφορά, από την αντίστοιχη εντολή αύξησης, είναι ότι αντί να αυξάνει, μειώνει κατά 1.

Ερωτήσεις

1. Πώς μπορούμε να αυξήσουμε το περιεχόμενο του καταχωρητή W κατά 1;
2. Τι κάνει η εντολή DECF TMR0, 1;
3. Τι τιμή παίρνει η σημαία Z και οι καταχωρητές W και TMR0, μετά την εκτέλεση της εντολής 'DECF TMR0, 0', με TMR0=5;

5.5.3 Εντολές αριθμητικών πράξεων

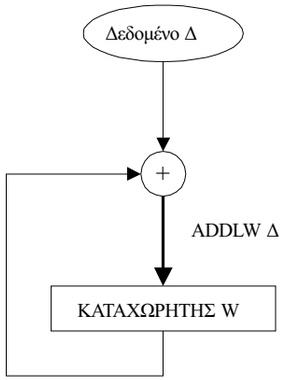
Ένας μικροελεγκτής ή μικροεπεξεργαστής θα ήταν άχρηστος αν δεν μπορούσε να κάνει αριθμητικές πράξεις. Ο PIC διαθέτει δύο εντολές για πρόσθεση και δύο για αφαίρεση. Με την βοήθεια αυτών, μπορεί να πραγματοποιήσει και τις πράξεις του πολλαπλασιασμού και διαίρεσης.

• ADDLW Δ

Η πρώτη εντολή πρόσθεσης, που θα μάθουμε, αφορά την απλή διαδικασία της πρόσθεσης δύο αριθμών. Την εντολή την έχουμε δει και σαν παράδειγμα στην ενότητα 5.4.

ΣΥΝΤΑΞΗ	ADDLW Δ
ΟΡΙΣΜΑΤΑ	Το Δ παίρνει τιμή από 0 έως 255.
ΣΗΜΑΙΕΣ	C, DC, Z

Περιγραφή: Το δεδομένο Δ προστίθεται στον καταχωρητή W και το αποτέλεσμα αποθηκεύεται στον καταχωρητή W (σχήμα 5.14).



Σχήμα 5.14 *Πρόσθεση του δεδομένου Δ στα περιεχόμενα του καταχωρητή W και αποθήκευση του αποτελέσματος στον W με την εντολή ADDLW Δ.*

Το δεδομένο Δ παίρνει τιμές από το 0 ως το 255, που σημαίνει ότι έχει μήκος ένα byte. Αυτό είναι πολύ λογικό αφού προστίθεται στον καταχωρητή W που έχει και αυτός το ίδιο μήκος. Επίσης, όπως άλλωστε αναμέναμε, η συγκεκριμένη πράξη επηρεάζει τις σημαίες του κρατούμενου (C), του ενδιάμεσου κρατούμενου (DC), δηλαδή κρατούμενου μεταξύ του 3ου και του 4ου bit, και του μηδενισμού (Z). Τέλος, η εντολή χρησιμοποιεί άμεση διευθυνσιοδότηση.

Θα παραθέσουμε, τώρα, δύο παραδείγματα με τα οποία θα κατανοήσουμε καλύτερα τη λειτουργία της εντολής.

1ο Παράδειγμα

Έστω ότι W = 6D H και ότι δίνουμε την εντολή ADDLW D1 H τότε:

$$\begin{array}{rrcl} & 0110\ 1101 & = & 6D\ H \\ + & 1101\ 0001 & = & D1\ H \\ \hline & 0011\ 1110 & = & 3E\ H \end{array}$$

Σύμφωνα με την εντολή, το αποτέλεσμα θα αποθηκευτεί στον W και άρα θα έχουμε W = 3E H, ενώ οι σημαίες θα γίνουν:

C = 1 (η πράξη δίνει κρατούμενο),

DC = 0 (η πράξη δεν δίνει ενδιάμεσο κρατούμενο) και

Z = 0 (το αποτέλεσμα δεν είναι 0),

ανεξάρτητα με τις τιμές που είχαν πριν την εκτέλεση της εντολής.

2ο Παράδειγμα

Έστω ότι W = 95 H και ότι δίνουμε την εντολή ADDLW 2C H τότε:

$$\begin{array}{rrcl} & 0010\ 1100 & = & 2C\ H \\ + & 1001\ 0101 & = & 95\ H \\ \hline & 1100\ 0001 & = & C1\ H \end{array}$$

Σύμφωνα με την εντολή, το αποτέλεσμα θα αποθηκευτεί στον W και άρα θα έχουμε W = C1 H, ενώ οι σημαίες θα γίνουν:

C = 0 (η πράξη δε δίνει κρατούμενο),

DC = 1 (η πράξη δίνει ενδιάμεσο κρατούμενο) και

Z = 0 (το αποτέλεσμα δεν είναι 0),

ανεξάρτητα με τις τιμές που είχαν πριν την εκτέλεση της εντολής.

• ADDWF K, d

Συνεχίζοντας θα περιγράψουμε μία δεύτερη εντολή πρόσθεσης, λίγο πιο πολύπλοκη από την πρώτη. Εδώ, πάλι, χρησιμοποιούμε τον καταχωρητή W. Όμως, αυτή την φορά, η πρόσθεση γίνεται με έναν αριθμό που βρίσκεται στον καταχωρητή, η διεύθυνση του οποίου υποδεικνύεται από το όρισμα K.

ΣΥΝΤΑΞΗ	ADDWF K, d
ΟΡΙΣΜΑΤΑ	• Το K παίρνει τιμές από 0 έως 127 ή είναι το συμβολικό όνομα ενός καταχωρητή. • Το d παίρνει τιμές από 0 έως 127.
ΣΗΜΑΙΑ	Z, C, DC

Περιγραφή: Το περιεχόμενο του W, προστίθεται στο περιεχόμενο του καταχωρητή K και το αποτέλεσμα αποθηκεύεται στον W, αν το d = 0, ή στον καταχωρητή K, αν το d = 1 (σχήμα 5.15).

Σχήμα 5.15 *Πρόσθεση του περιεχομένου του καταχωρητή W στο περιεχόμενο του καταχωρητή K και αποθήκευση του αποτελέσματος στον W με την εντολή ADDWF K, 0 ή στον K με την εντολή ADDWF K, 1.*

Ο καταχωρητής Κ ορίζεται όπως και στις άλλες εντολές που είδαμε έως τώρα. Επίσης, όπως άλλωστε και στην προηγούμενη εντολή πρόσθεσης, επηρεάζονται οι σημαίες του κρατούμενου (C), του ενδιάμεσου κρατούμενου (DC) και του μηδενισμού (Z).

Θα παραθέσουμε, πάλι, δύο παραδείγματα με τα οποία θα κατανοήσουμε την λειτουργία καθώς και την διαφορά της από την προηγούμενη.

1ο Παράδειγμα

Έστω ότι W = A3 H, TMR0 = 15 H και ότι δίνουμε την εντολή ADDWF TMR0, 0, τότε:

$$\begin{array}{rrcr} & 1010\ 0011 & = & \text{A3 H} \\ + & \underline{0001\ 0101} & = & \underline{\text{15 H}} \\ & 1011\ 1000 & = & \text{B8 H} \end{array}$$

Σύμφωνα με την εντολή, το αποτέλεσμα θα αποθηκευτεί στον W και άρα θα έχουμε W = B8 H, ενώ οι σημαίες θα γίνουν:

C = 0 (η πράξη δε δίνει κρατούμενο),
DC = 0 (η πράξη δε δίνει ενδιάμεσο κρατούμενο) και
Z = 0 (το αποτέλεσμα δεν είναι 0),

ανεξάρτητα με τις τιμές που είχαν πριν την εκτέλεση της εντολής.

2ο Παράδειγμα

Έστω ότι W = DB H, TMR0 = 85 H και ότι δίνουμε την εντολή ADDWF TMR0, 1 τότε:

$$\begin{array}{rrcr} & 1101\ 1011 & = & \text{DB H} \\ + & \underline{1000\ 0101} & = & \underline{\text{85 H}} \\ & 0110\ 0000 & = & \text{60 H} \end{array}$$

Σύμφωνα με την εντολή, το αποτέλεσμα (60 H) θα αποθηκευτεί στον TMR0, ενώ ο W θα κρατήσει την αρχική του τιμή (DB H). Οι σημαίες θα γίνουν:

C = 1 (η πράξη δίνει κρατούμενο),
DC = 1 (η πράξη δίνει ενδιάμεσο κρατούμενο) και
Z = 0 (το αποτέλεσμα δεν είναι 0),

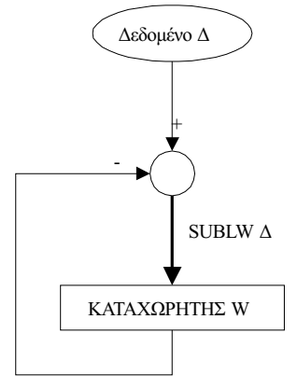
ανεξάρτητα με τις τιμές που είχαν πριν την εκτέλεση της εντολής.

• SUBLW Δ

Οι αριθμητικές εντολές περιλαμβάνουν και την εντολή της αφαίρεσης. Η εντολή, στη δομή και λειτουργία της, μοιάζει με αυτή της πρόσθεσης ενός αριθμού με τον καταχωρητή W. Ας δούμε και εδώ την εντολή σε έναν συνοπτικό πίνακα.

ΣΥΝΤΑΞΗ	SUBLW Δ
ΟΡΙΣΜΑΤΑ	Το Δ παίρνει τιμή από 0 έως 255.
ΣΗΜΑΙΕΣ	C, DC, Z

Περιγραφή: Το περιεχόμενο του καταχωρητή W, αφαιρείται από το δεδομένο και το αποτέλεσμα αποθηκεύεται στον καταχωρητή W (σχήμα 5.16).



Σχήμα 5.16 *Αφαίρεση του περιεχομένου του καταχωρητή W από το δεδομένου Δ και αποθήκευση του αποτελέσματος στον W με την εντολή SUBLW Δ.*

Όπως και στην αντίστοιχη εντολή πρόσθεσης, το δεδομένο Δ παίρνει τιμές από το 0 ως το 255. Επίσης, και εδώ, η συγκεκριμένη πράξη επηρεάζει τις σημαίες του κρατούμενου (C), του ενδιάμεσου κρατούμενου (DC) και του μηδενισμού (Z). Η εντολή χρησιμοποιεί άμεση διευθυνσιοδότηση.

Στην συνέχεια, δίνουμε, τρία παραδείγματα τα οποία θα μας βοηθήσουν να κατανοήσουμε καλύτερα τη λειτουργία της εντολής.

1ο Παράδειγμα

Έστω ότι W = 6D H και ότι δίνουμε την εντολή SUBLW D1 H τότε:

$$\begin{array}{rrcr} & 1101\ 0001 & = & \text{D1 H} \\ - & \underline{0110\ 1101} & = & \underline{\text{6D H}} \\ & 0110\ 0100 & = & \text{64 H} \end{array}$$

Σύμφωνα με την εντολή, το αποτέλεσμα θα αποθηκευτεί στον W και άρα θα

έχουμε W = 64 H, ενώ οι σημαίες θα γίνουν:

C = 0 (η πράξη δε δίνει κρατούμενο),
DC = 1 (η πράξη δίνει ενδιάμεσο κρατούμενο) και
Z = 0 (το αποτέλεσμα δεν είναι 0),

ανεξάρτητα από τις τιμές που είχαν πριν την εκτέλεση της εντολής.

2ο Παράδειγμα

Έστω ότι W = A4 H και ότι δίνουμε την εντολή SUBLW 21 H τότε:

	0010 0001	=	21 H
-	1010 0100	=	A4 H
	0111 1101	=	7D H

Σύμφωνα με την εντολή, το αποτέλεσμα θα αποθηκευτεί στον W και άρα θα έχουμε W = 7D H, ενώ οι σημαίες θα γίνουν:

C = 1 (η πράξη δίνει κρατούμενο),
DC = 1 (η πράξη δίνει ενδιάμεσο κρατούμενο) και
Z = 0 (το αποτέλεσμα δεν είναι 0),

ανεξάρτητα με τις τιμές που είχαν πριν την εκτέλεση της εντολής.

3ο Παράδειγμα

Έστω ότι W = 21 H και ότι δίνουμε την εντολή ADDLW 21 H, τότε:

	0010 0001	=	21 H
-	0010 0001	=	21 H
	0000 0000	=	0

Σύμφωνα με την εντολή, το αποτέλεσμα θα αποθηκευτεί στον W και άρα θα έχουμε W = 0, ενώ οι σημαίες θα γίνουν:

C = 0 (η πράξη δε δίνει κρατούμενο),
DC = 0 (η πράξη δε δίνει ενδιάμεσο κρατούμενο) και
Z = 1 (το αποτέλεσμα είναι 0),

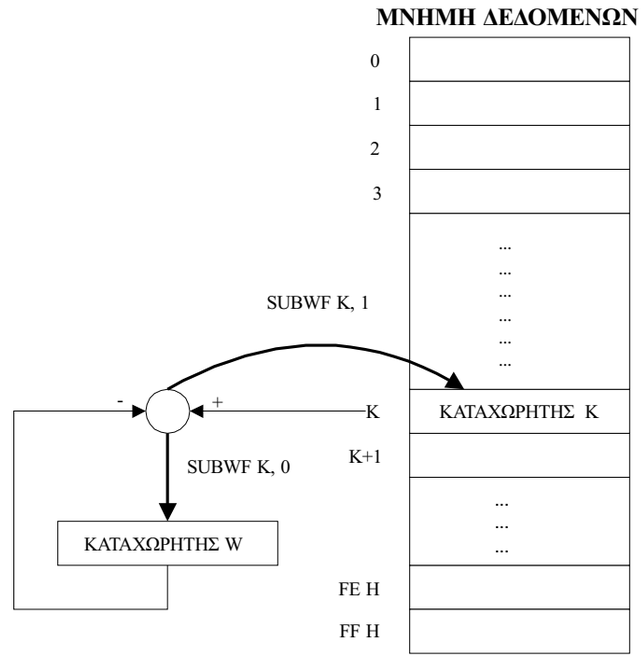
ανεξάρτητα με τις τιμές που είχαν πριν την εκτέλεση της εντολής.

• SUBWF K, d

Οι αριθμητικές εντολές περιλαμβάνουν και την εντολή της αφαίρεσης. Η εντολή, στη δομή και λειτουργία της, μοιάζει με αυτή της πρόσθεσης ενός αριθμού με τον καταχωρητή W.

ΣΥΝΤΑΞΗ	SUBWF K, d
ΟΡΙΣΜΑΤΑ	• Το K παίρνει τιμές από 0 έως 127 ή είναι το συμβολικό όνομα ενός καταχωρητή. • Το d παίρνει τιμές από 0 έως 127.
ΣΗΜΑΙΕΣ	Z, C, DC

Περιγραφή: Το περιεχόμενο του W, αφαιρείται από το περιεχόμενο του καταχωρητή K και το αποτέλεσμα αποθηκεύεται στον W, αν το d = 0, ή στον καταχωρητή K, αν το d = 1 (σχήμα 5.17).



Σχήμα 5.17 *Αφαίρεση του περιεχομένου του καταχωρητή W από το περιεχόμενο του καταχωρητή K και αποθήκευση του αποτελέσματος στον W με την εντολή SUBWF K, 0 ή στον K με την εντολή SUBWF K, 1.*

Όπως στην αντίστοιχη εντολή πρόσθεσης, ο καταχωρητής παίρνει τιμές από το 0 ως το 127 ή το συμβολικό του όνομα, αν έχει. Παρομοίως με την προηγούμενη εντολή αφαίρεσης οι σημαίες που επηρεάζονται είναι του κρατούμενου (C), του ενδιάμεσου

κρατούμενου (DC) και του μηδενισμού (Z). Στην συνέχεια, δίνουμε, τρία παραδείγματα τα οποία θα μας βοηθήσουν να κατανοήσουμε καλύτερα τη λειτουργία της εντολής.

1ο Παράδειγμα
Έστω ότι W = 6D H, TMR0 = D1 H και ότι δίνουμε την εντολή SUBWF TMR0, 0, τότε:

	1101 0001	=	D1 H
-	0110 1101	=	6D H
	0110 0100	=	64 H

Σύμφωνα με την εντολή, το αποτέλεσμα θα αποθηκευτεί στον W και, άρα, θα έχουμε W = 64 H, ενώ οι σημαίες θα γίνουν:

C = 0 (η πράξη δε δίνει κρατούμενο),
DC = 1 (η πράξη δίνει ενδιάμεσο κρατούμενο) και
Z = 0 (το αποτέλεσμα δεν είναι 0),

ανεξάρτητα από τις τιμές που είχαν πριν την εκτέλεση της εντολής.

2ο Παράδειγμα
Έστω ότι W = 67 H, TMR0 = A4 H και ότι δίνουμε την εντολή SUBWF TMR0, 1, τότε:

	0110 0111	=	67 H
-	1010 0100	=	A4 H
	1100 0011	=	C3 H

Σύμφωνα με την εντολή, το αποτέλεσμα θα αποθηκευτεί στον TMR0, αφού το d = 1, και, άρα, θα έχουμε TMR0 = C3 H, ενώ οι σημαίες θα γίνουν:

C = 1 (η πράξη δίνει κρατούμενο),
DC = 0 (η πράξη δε δίνει ενδιάμεσο κρατούμενο) και
Z = 0 (το αποτέλεσμα δεν είναι 0),

ανεξάρτητα με τις τιμές που είχαν πριν την εκτέλεση της εντολής.

3ο Παράδειγμα
Έστω ότι W = 21 H, TMR0 = 21 H και ότι δίνουμε την εντολή SUBWF TMR0, 0, τότε:

	0010 0001	=	21 H
-	0010 0001	=	21 H
	0000 0000,	=	0

Σύμφωνα με την εντολή, το αποτέλεσμα θα αποθηκευτεί στον W και άρα θα έχουμε W = 0, ενώ οι σημαίες θα γίνουν:

C = 0 (η πράξη δε δίνει κρατούμενο),
DC = 0 (η πράξη δε δίνει ενδιάμεσο κρατούμενο) και
Z = 1 (το αποτέλεσμα είναι 0),

ανεξάρτητα με τις τιμές που είχαν πριν την εκτέλεση της εντολής.

Ερωτήσεις

1. Ποια τιμή παίρνουν οι σημαίες C, DC και Z, μετά την εκτέλεση της εντολής ADDWF
2. TMR0, 1, με TMR0=0D H και W=13 H;
3. Που αποθηκεύεται το αποτέλεσμα της προηγουμένης ερώτησης;

5.5.4 Εντολές λογικών πράξεων και διαδικασιών

Πέρα από τις εντολές αριθμητικών πράξεων, ο PIC, διαθέτει και μία ομάδα εντολών λογικών πράξεων. Οι εντολές αυτές είναι πολύ χρήσιμες για τον έλεγχο των ενσωματωμένων περιφερειακών μονάδων. Παραδείγματα θα δούμε σε εφαρμογές που θα παρουσιαστούν σε επόμενες ενότητες.

Σε αυτήν την ενότητα θα συμπεριλάβουμε και τρεις εντολές που, μπορούμε να πούμε ότι εκτελούν λογικές διαδικασίες στους καταχωρητές. Οι δύο από αυτές αφορούν ενέργειες περιστροφής καταχωρητών. Η τρίτη αφορά την ανταλλαγή των τεσσάρων σημαντικότερων bits με τα τέσσερα λιγότερο σημαντικά.

Ο PIC μπορεί να εκτελέσει τις λογικές πράξεις:

- ΚΑΙ (AND)
- Η (OR)
- Αποκλειστικό Η (XOR)
- Συμπλήρωμα ως προς 1
- Περιστροφή προς τα δεξιά
- Περιστροφή προς τα αριστερά
- Ανταλλαγή bits

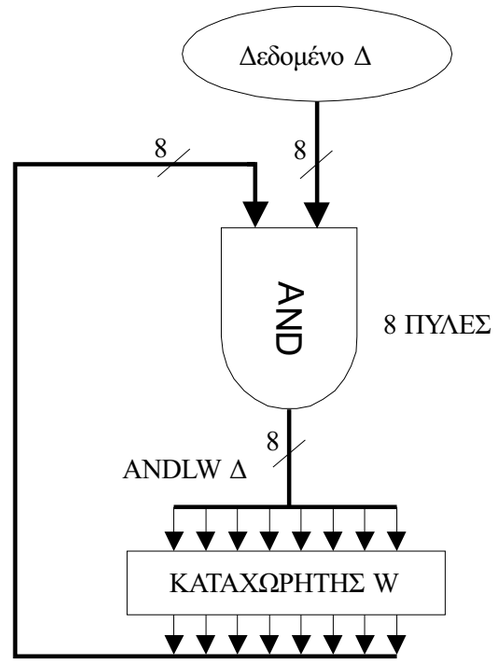
Για τις λογικές πράξεις του AND και του OR, ο PIC διαθέτει, για την κάθε μία, από δύο εντολές. Στην συνέχεια, θα παρουσιάσουμε με τη σειρά τις εντολές αυτές.

• ANDLW Δ

Η πρώτη λογική εντολή, που θα αναλύσουμε, αφορά την απλή διαδικασία της λογικής πράξης ΚΑΙ δύο αριθμών. Παραθέτουμε, λοιπόν, τον πίνακα με τα βασικότερα στοιχεία της εντολής.

ΣΥΝΤΑΞΗ	ANDLW Δ
ΟΡΙΣΜΑΤΑ	Το Δ παίρνει τιμή από 0 έως 255.
ΣΗΜΑΙΑ	Z

Περιγραφή: Εκτέλεση της πράξης του λογικού ΚΑΙ μεταξύ του περιεχομένου του καταχωρητή W και του δεδομένου Δ. Το αποτέλεσμα αποθηκεύεται στον καταχωρητή W (σχήμα 5.18).



Σχήμα 5.18 *Εκτέλεση της λογικής πράξης ΚΑΙ μεταξύ του περιεχομένου του καταχωρητή W και του δεδομένου Δ και αποθήκευση του αποτελέσματος στον W με την εντολή ANDLW Δ.*

Και στην κατηγορία των λογικών εντολών, το δεδομένο Δ παίρνει τιμές από το 0 ως το 255. Άλλωστε, το δεδομένο πρέπει να έχει το ίδιο μήκος με τον καταχωρητή W, αφού εκτελείται η λογική πράξη ΚΑΙ, bit προς bit, μεταξύ τους. Η συγκεκριμένη πράξη επηρεάζει τη σημαία του μηδενισμού (Z).

Στη συνέχεια, θα δούμε δύο παραδείγματα με αυτήν την εντολή.

1ο Παράδειγμα

Έστω ότι W = 35 H και ότι δίνουμε την εντολή ANDLW A4 H τότε:

```
    0011 0101   (35 H)
AND 1010 0100   (A4 H)
    0010 0100   (24 H)
```

Όπως είπαμε, το αποτέλεσμα αποθηκεύεται στον W άρα, θα έχουμε W = 24 H, ενώ η σημαία Z θα γίνει 0.

2ο Παράδειγμα

Έστω ότι W = AB H και ότι δίνουμε την εντολή ANDLW 14 H τότε:

```
    1010 1011   (AB H)
AND 0001 0100   (14 H)
    0000 0000   (0)
```

Σύμφωνα με την εντολή, το αποτέλεσμα θα αποθηκευτεί στον W και άρα θα έχουμε W = 0, ενώ η σημαία Z θα γίνει ίση με 1, ανεξάρτητα με την τιμή που είχε πριν την εκτέλεση της εντολής.

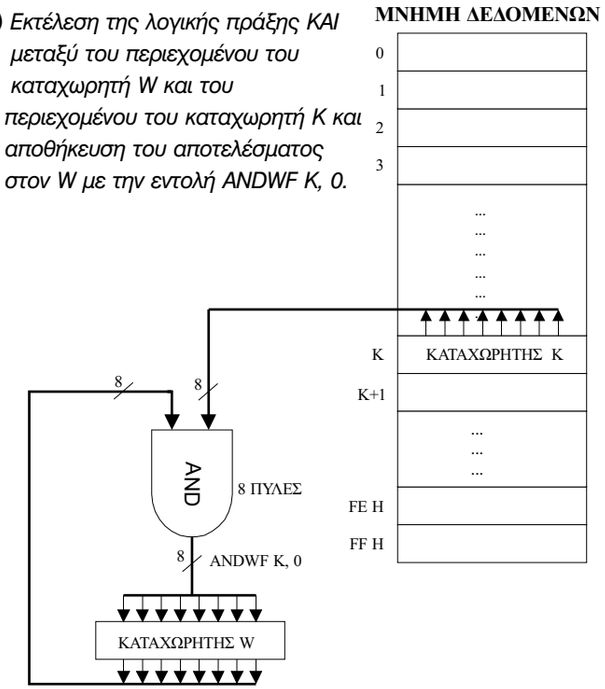
• ANDWF K, d

Η δεύτερη εντολή με την οποία μπορούμε να εκτελέσουμε την λογική πράξη ΚΑΙ, εκτελείται μεταξύ του καταχωρητή W και ενός άλλου καταχωρητή. Όπως και στις αντίστοιχες εντολές αριθμητικών πράξεων, χρησιμοποιούμε ως όρισμα, τη διεύθυνση στη μνήμη δεδομένων, του καταχωρητή αυτού.

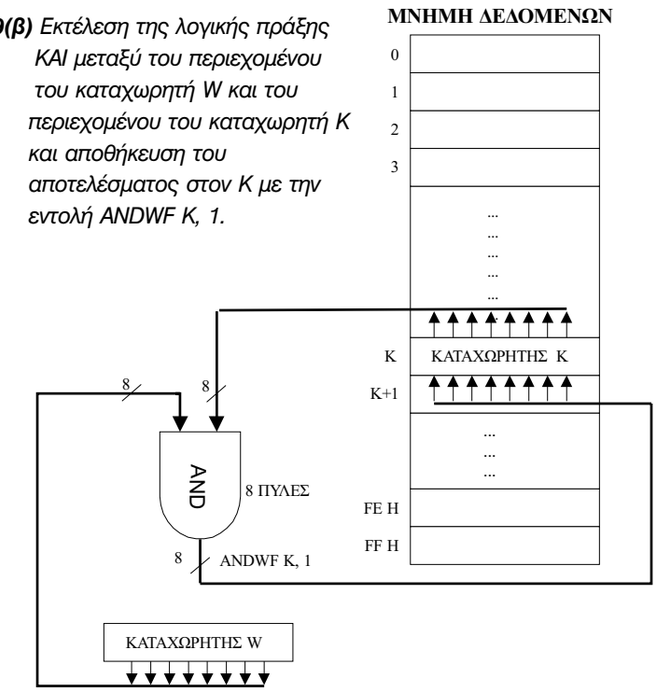
ΣΥΝΤΑΞΗ	ANDWF K, d
ΟΡΙΣΜΑΤΑ	• Το K παίρνει τιμές από 0 έως 127 ή είναι το συμβολικό όνομα ενός καταχωρητή. • Το d παίρνει τιμές από 0 έως 127.
ΣΗΜΑΙΑ	Z

Περιγραφή: Εκτέλεση της πράξης του λογικού ΚΑΙ μεταξύ του περιεχομένου του W και του περιεχομένου του καταχωρητή K. Το αποτέλεσμα αποθηκεύεται στον W, αν το d = 0 (σχήμα 5.19α), ή στον καταχωρητή K, αν το d = 1 (σχήμα 5.19β).

Σχήμα 5.19(α) *Εκτέλεση της λογικής πράξης ΚΑΙ μεταξύ του περιεχομένου του καταχωρητή W και του περιεχομένου του καταχωρητή Κ και αποθήκευση του αποτελέσματος στον W με την εντολή ANDWF K, 0.*



Σχήμα 5.19(β) *Εκτέλεση της λογικής πράξης ΚΑΙ μεταξύ του περιεχομένου του καταχωρητή W και του περιεχομένου του καταχωρητή Κ και αποθήκευση του αποτελέσματος στον Κ με την εντολή ANDWF K, 1.*



Ο καταχωρητής Κ ορίζεται όπως συνήθως. Η εντολή ενημερώνει την σημαία του μηδενισμού (Z).
Θα παραθέσουμε δύο παραδείγματα, που είναι ενδεικτικά της λειτουργίας της εντολής και της διαφοράς της από την προηγούμενη.

1ο Παράδειγμα
Θεωρούμε ότι ο καταχωρητής W = A3 H και ο TMR0 = 15 H. Έστω ότι δίνουμε την εντολή ANDWF TMR0, 0, τότε:

	1010 0011	(A3 H)
AND	0001 0101	(15 H)
	0000 0001	(1)

Σύμφωνα με την εντολή, το αποτέλεσμα θα αποθηκευτεί στον W, δηλαδή W = 1, ενώ η σημαία Z θα γίνει 0, αφού το αποτέλεσμα δεν είναι μηδενικό.

2ο Παράδειγμα
Θεωρούμε ότι ο καταχωρητής W = 5A H και ο TMR0 = 85 H. Έστω ότι δίνουμε την εντολή ANDWF TMR0, 1 τότε:

	0101 1010	(5A H)
AND	1000 0101	(85 H)
	0000 0000	(0)

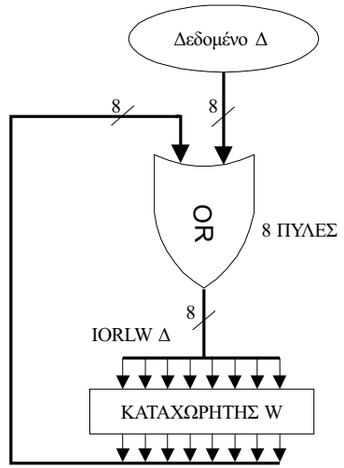
Σύμφωνα με την εντολή, το αποτέλεσμα θα αποθηκευτεί στον TMR0 και άρα θα έχουμε TMR0 = 0, ενώ η σημαία Z θα γίνει 1, αφού το αποτέλεσμα είναι 0. Ο καταχωρητής W διατηρεί την τιμή του.

• IORLW Δ

Η δεύτερη λογική πράξη, που μπορεί να εκτελέσει ο PIC, είναι το λογικό Η (OR). Η εντολή, που θα αναλύσουμε, εδώ, είναι η αντίστοιχη της ANDLW για το λογικό Η.

ΣΥΝΤΑΞΗ	IORLW Δ
ΟΡΙΣΜΑΤΑ	Το Δ παίρνει τιμή από 0 έως 255.
ΣΗΜΑΙΑ	Z

Περιγραφή: Εκτέλεση της πράξης του λογικού Η μεταξύ του περιεχομένου του καταχωρητή W και του δεδομένου Δ. Το αποτέλεσμα αποθηκεύεται στον καταχωρητή W (σχήμα 5.20).



Σχήμα 5.20 *Εκτέλεση της λογικής πράξης Η μεταξύ του περιεχομένου του καταχωρητή W και του δεδομένου Δ και αποθήκευση του αποτελέσματος στον W με την εντολή IORLW Δ.*

Όπως πάντα, το δεδομένο παίρνει τιμές από το 0 ως το 255. Η συγκεκριμένη πράξη επηρεάζει την σημαία του μηδενισμού (Z). Ακολούθως, παραθέτουμε δύο παραδείγματα αυτής της εντολής.

1ο Παράδειγμα

Έστω ότι W = 11 Η και ότι δίνουμε την εντολή IORLW 96 Η τότε:

```
      0001 0001     (11 H)
OR    1001 0110     (96 H)
      1001 0111     (97 H)
```

Όπως είπαμε, το αποτέλεσμα αποθηκεύεται στον W και, άρα, θα έχουμε W = 97 Η, ενώ η σημαία Z = 0, ανεξάρτητα με την τιμή που είχε πριν την εκτέλεση της εντολής.

2ο Παράδειγμα

Έστω ότι W = 0 και ότι δίνουμε την εντολή IORLW 0 τότε:

```
      0000 0000     (0)
OR    0000 0000     (0)
      0000 0000,    (0)
```

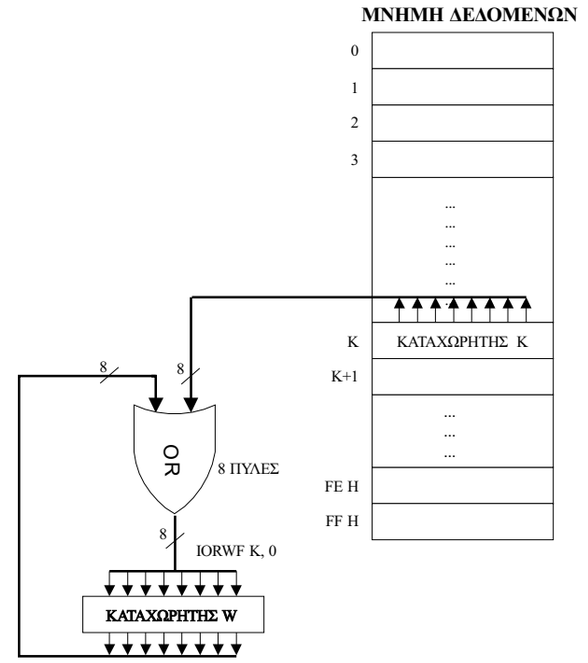

Σύμφωνα με την εντολή, το αποτέλεσμα θα αποθηκευτεί στον W και άρα θα έχουμε W = 0, ενώ η σημαία Z θα γίνει ίση με 1, ανεξάρτητα με την τιμή είχε πριν την εκτέλεση της εντολής.

• IORWF K, d

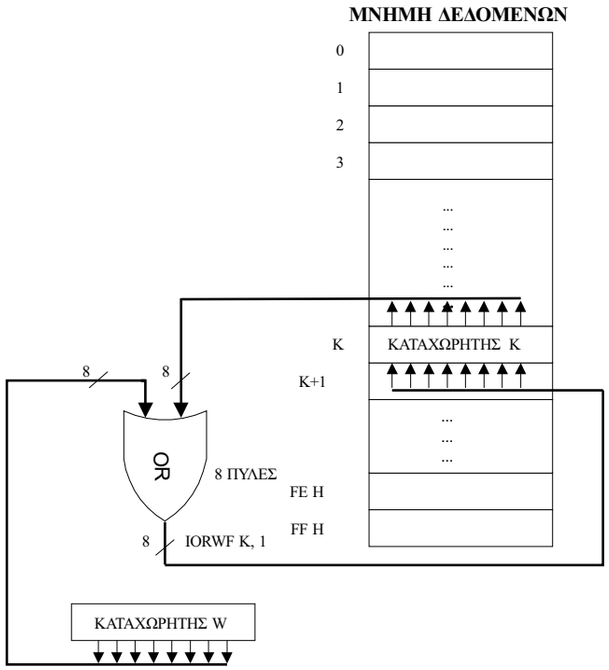
Η δεύτερη εντολή με την οποία μπορούμε να εκτελέσουμε την πράξη του λογικού Η, εκτελείται μεταξύ του καταχωρητή W και ενός άλλου καταχωρητή. Όπως και στην αντίστοιχη εντολή λογικού ΚΑΙ, χρησιμοποιούμε ως όρισμα, τη διεύθυνση στη μνήμη δεδομένων, του καταχωρητή αυτού.

ΣΥΝΤΑΞΗ	ANDWF K, d
ΟΡΙΣΜΑΤΑ	• Το K παίρνει τιμές από 0 έως 127 ή είναι το συμβολικό όνομα ενός καταχωρητή. • Το d παίρνει τιμές από 0 έως 127.
ΣΗΜΑΙΑ	Z

Περιγραφή: Εκτέλεση της πράξης του λογικού Η μεταξύ του περιεχομένου του W και του περιεχομένου του καταχωρητή K. Το αποτέλεσμα αποθηκεύεται στον W, αν το d = 0 (σχήμα 5.21α,) ή στον καταχωρητή, αν το d = 1 (σχήμα 5.21β).



Σχήμα 5.21α *Εκτέλεση της λογικής πράξης Η μεταξύ του περιεχομένου του καταχωρητή W και του περιεχομένου του καταχωρητή K και αποθήκευση του αποτελέσματος στον W με την εντολή IORWF K, 0.*



Σχήμα 5.21β *Εκτέλεση της λογικής πράξης Η μεταξύ του περιεχομένου του καταχωρητή W και του περιεχομένου του καταχωρητή K και αποθήκευση του αποτελέσματος στον K με την εντολή IORWF K, 1.*

Και εδώ, η διεύθυνση του καταχωρητή K παίρνει τιμές από το 0 έως το 127 ή το συμβολικό του όνομα, αν έχει. Επίσης, όπως άλλωστε και στην προηγούμενη εντολή λογικού Η, ενημερώνεται η σημαία του μηδενισμού (Z).

1ο Παράδειγμα

Θεωρούμε ότι ο καταχωρητής W = E4 H και ο TMR0 = 5. Έστω ότι δίνουμε την εντολή IORWF TMR0, 0, τότε:

```
      1110 0100      (E4 H)
OR    0000 0101      (5)
      1110 0101      (E5 H)
```

Σύμφωνα με την εντολή, το αποτέλεσμα θα αποθηκευτεί στον W, δηλαδή W = E5 H, ενώ η σημαία Z θα γίνει 0.

2ο Παράδειγμα

Θεωρούμε ότι ο καταχωρητής W = 0 και ο TMR0 = 0. Έστω ότι δίνουμε την εντολή IORWF TMR0, 1 τότε:


```
      0000 0000      (0)
OR    0000 0000      (0)
      0000 0000      (0)
```

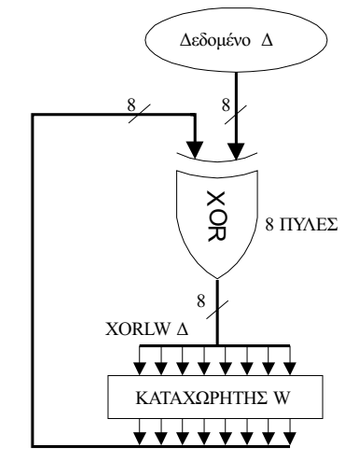
Σύμφωνα με την εντολή, το αποτέλεσμα θα αποθηκευτεί στον TMR0 και, συνεπώς, θα έχουμε TMR0 = 0, ενώ η σημαία Z θα γίνει 1, αφού το αποτέλεσμα είναι 0. Ο καταχωρητής W διατηρεί την τιμή του.

• XORLW Δ

Η τρίτη λογική πράξη, που μπορεί να εκτελέσει ο PIC, είναι του λογικού αποκλειστικού Η (XOR). Η εντολή, που θα αναλύσουμε, είναι η αντίστοιχη των ANDLW και IORLW για το αποκλειστικό Η.

ΣΥΝΤΑΞΗ	XORLW Δ
ΟΡΙΣΜΑΤΑ	Το Δ παίρνει τιμές από 0 έως 255.
ΣΗΜΑΙΑ	Z

Περιγραφή: Εκτέλεση της πράξης του αποκλειστικού Η (XOR) μεταξύ του περιεχομένου του καταχωρητή W και του δεδομένου Δ. Το αποτέλεσμα αποθηκεύεται στον καταχωρητή W (σχήμα 5.22).



Σχήμα 5.22 *Εκτέλεση της λογικής πράξης αποκλειστικού Η μεταξύ του περιεχομένου του καταχωρητή W και του δεδομένου Δ και αποθήκευση του αποτελέσματος στον W με την εντολή XORLW Δ.*

Το δεδομένο Δ παίρνει τιμές από το 0 ως το 255. Παρομοίως, με τις προηγούμενες λογικές εντολές, η παρούσα επηρεάζει την σημαία του μηδενισμού (Z). Συνεχίζουμε με δύο παραδείγματα της εντολής αυτής.

1ο Παράδειγμα

Έστω ότι W = 19 H και ότι δίνουμε την εντολή XORLW 97 H τότε:

```
      0001 1001    (1A H)
XOR   1001 0111    (97 H)
      1000 1110    (8D H)
```

Ως γνωστόν, το αποτέλεσμα αποθηκεύεται στον W και, άρα, έχουμε W = 8D H, ενώ η σημαία Z γίνεται 0, αφού το αποτέλεσμα είναι διάφορο του μηδενός.

2ο Παράδειγμα

Έστω ότι W = CB H και ότι δίνουμε την εντολή XORLW CB H τότε:

```
      1100 1011    (CB H)
XOR   1100 1011    (CB H)
      0000 0000    (0)
```

Σύμφωνα με την εντολή, το αποτέλεσμα θα αποθηκευτεί στον W και, άρα, θα έχουμε W = 0, ενώ η σημαία Z θα γίνει 1, αφού έχουμε μηδενικό αποτέλεσμα, ανεξάρτητα με την τιμή που είχε πριν την εκτέλεση της εντολής.

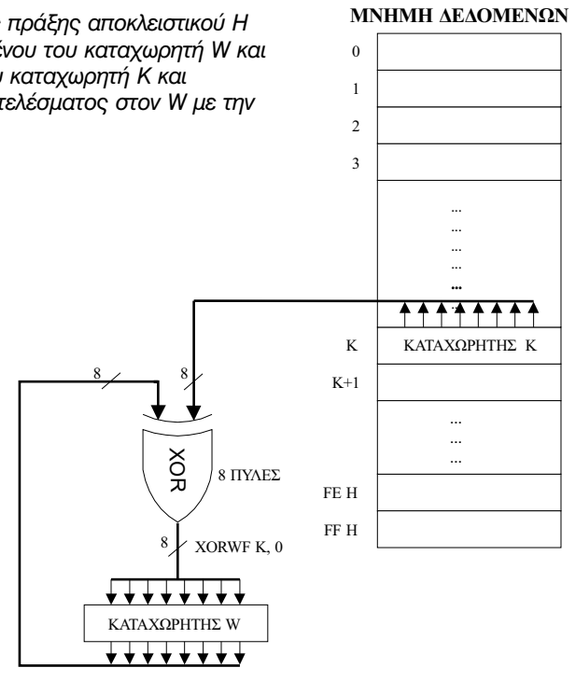
- **XORWF K, d**

Η δεύτερη εντολή με την οποία μπορούμε να εκτελέσουμε την πράξη του λογικού αποκλειστικού Η, εκτελείται μεταξύ του καταχωρητή W και ενός άλλου καταχωρητή. Όπως και στην αντίστοιχη εντολή λογικού Η, χρησιμοποιούμε ως όρισμα, την διεύθυνση στην μνήμη δεδομένων, του καταχωρητή αυτού.

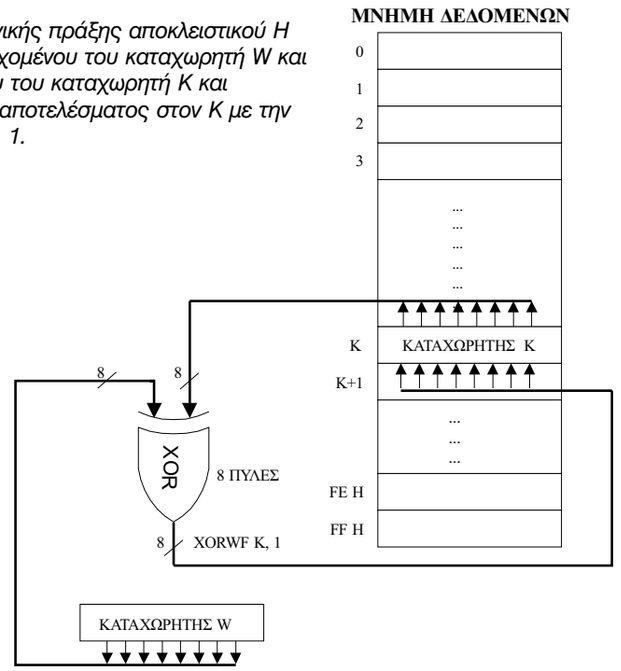
ΣΥΝΤΑΞΗ	XORWF K, d
ΟΡΙΣΜΑΤΑ	• Το K παίρνει τιμές από 0 έως 127 ή είναι το συμβολικό όνομα ενός καταχωρητή. • Το d παίρνει τιμές από 0 έως 127.
ΣΗΜΑΙΑ	Z

Περιγραφή: Εκτέλεση της πράξης του αποκλειστικού Η μεταξύ του περιεχομένου του W και του περιεχομένου του καταχωρητή K. Το αποτέλεσμα αποθηκεύεται στον W, αν το d = 0 (σχήμα 5.23α), ή στον καταχωρητή K, αν το d = 1 (σχήμα 5.23β).

Σχήμα 5.23α *Εκτέλεση της λογικής πράξης αποκλειστικού Η μεταξύ του περιεχομένου του καταχωρητή W και του περιεχομένου του καταχωρητή K και αποθήκευση του αποτελέσματος στον W με την εντολή XORWF K, 0.*



Σχήμα 5.23β *Εκτέλεση της λογικής πράξης αποκλειστικού Η μεταξύ του περιεχομένου του καταχωρητή W και του περιεχομένου του καταχωρητή K και αποθήκευση του αποτελέσματος στον K με την εντολή XORWF K, 1.*



Το όρισμα Κ αντιστοιχεί στη διεύθυνση του καταχωρητή που θέλουμε. Επίσης, όπως άλλωστε και στην προηγούμενη εντολή αποκλειστικού Η, ενημερώνεται η σημαία του μηδενισμού (Z).

Ακολουθούν, δύο παραδείγματα, με την εντολή αυτή.

1ο Παράδειγμα

Θεωρούμε ότι ο καταχωρητής W = ED H και ο TMR0 = A5 H. Έστω ότι δίνουμε την εντολή XORWF TMR0, 0, τότε:

	1110 1101	(ED H)
XOR	1010 0101	(A5 H)
	0100 1000	(48 H)

Σύμφωνα με την εντολή, το αποτέλεσμα θα αποθηκευτεί στον W, δηλαδή W = 48 H, ενώ η σημαία Z θα γίνει 0, αφού το αποτέλεσμα δεν είναι 0. Ο TMR0 θα παραμείνει ως έχει.

2ο Παράδειγμα

Θεωρούμε ότι ο καταχωρητής W = 62 H και ο TMR0 = 62 H. Έστω ότι δίνουμε την εντολή XORWF TMR0, 1 τότε:

	0110 0010	(62 H)
XOR	0110 0010	(62 H)
	0000 0000	(0)

Σύμφωνα με την εντολή, το αποτέλεσμα θα αποθηκευτεί στον TMR0 και, άρα, θα έχουμε TMR0 = 0, ενώ η σημαία Z θα γίνει 1, αφού το αποτέλεσμα είναι 0. Ο καταχωρητής W διατηρεί την τιμή του.

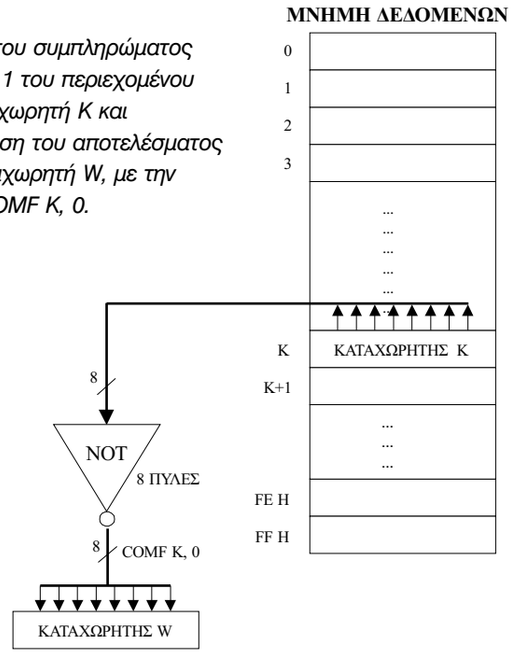
• COMF K, d

Εκτός των γνωστών λογικών πράξεων, του ΚΑΙ και του Η, οι λογικές εντολές περιλαμβάνουν και την εντολή του συμπληρώματος.

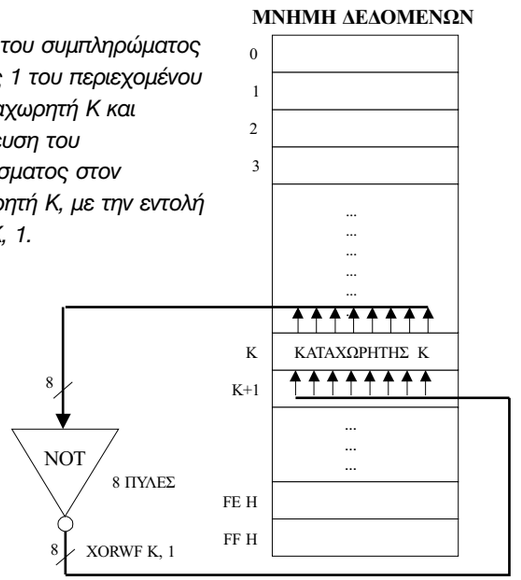
ΣΥΝΤΑΞΗ	COMF K, d
ΟΡΙΣΜΑΤΑ	• Το K παίρνει τιμές από 0 έως 127 ή είναι το συμβολικό όνομα ενός καταχωρητή. • Το d παίρνει τιμές από 0 έως 127.
ΣΗΜΑΙΑ	Z

Περιγραφή: Το συμπλήρωμα ως προς 1 του καταχωρητή Κ, αποθηκεύεται στον W, αν το d = 0 (σχήμα 5.24α), ή στον καταχωρητή Κ, αν το d = 1(σχήμα 5.24β).

Σχήμα 5.24α *Εύρεση του συμπληρώματος ως προς 1 του περιεχομένου του καταχωρητή Κ και αποθήκευση του αποτελέσματος στον καταχωρητή W, με την εντολή COMF K, 0.*



Σχήμα 5.24β *Εύρεση του συμπληρώματος ως προς 1 του περιεχομένου του καταχωρητή Κ και αποθήκευση του αποτελέσματος στον καταχωρητή Κ, με την εντολή COMF K, 1.*



Στη συνέχεια, δίνουμε, δύο παραδείγματα, για να δούμε καλύτερα πώς λειτουργεί η εντολή αυτή.

1ο Παράδειγμα
Έστω ότι W = 6D H και TMR0 = 71 H.
Δίνουμε την εντολή COMF TMR0, 0.
Τότε, το αποτέλεσμα θα είναι 8E H και θα αποθηκευτεί στον W, δηλαδή W = 8E H, αφού το όρισμα d = 0.
Η σημαία Z = 0, αφού το αποτέλεσμα δεν είναι 0, ανεξάρτητα με το τι ήταν πριν.

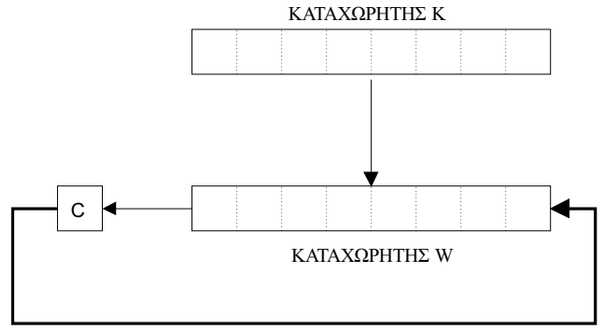
2ο Παράδειγμα
Έστω ότι W = 5A H και TMR0 = FF H.
Δίνουμε την εντολή COMF TMR0, 1.
Τότε, το αποτέλεσμα θα είναι 0 και θα αποθηκευτεί στον TMR0, δηλαδή TMR0 = 0, αφού το όρισμα d = 1.
Η σημαία Z = 1, αφού το αποτέλεσμα είναι 0, ανεξάρτητα με το τι ήταν πριν.

• RLF K, d

Όπως είπαμε, ο PIC διαθέτει δύο εντολές που του δίνουν τη δυνατότητα να εκτελεί περιστροφή στο περιεχόμενο ενός καταχωρητή. Ας δούμε, λοιπόν, το συνοπτικό πίνακα της πρώτης εντολής, που περιστρέφει το περιεχόμενο του καταχωρητή προς τα αριστερά, και, στη συνέχεια, ένα επεξηγηματικό σχήμα.

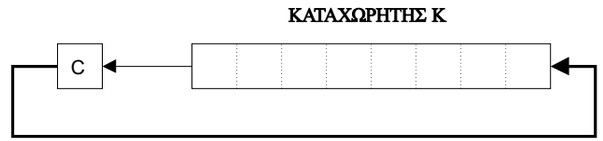
ΣΥΝΤΑΞΗ	RLF K, d
ΟΡΙΣΜΑΤΑ	• Το K παίρνει τιμές από 0 έως 127 ή είναι το συμβολικό όνομα ενός καταχωρητή. • Το d παίρνει τιμές από 0 έως 127.
ΣΗΜΑΙΕΣ	C

Περιγραφή: Το περιεχόμενο του καταχωρητή K και η σημαία C περιστρέφονται προς τα αριστερά κατά ένα bit. Το αποτέλεσμα αποθηκεύεται στον W, αν το d = 0 (σχήμα 5.25α), ή στον καταχωρητή K, αν το d = 1 (σχήμα 5.25β)



RLF K, 0

Σχήμα 5.25α *Αριστερή περιστροφή του περιεχομένου του καταχωρητή K, μέσω σημαίας C, και αποθήκευση του αποτελέσματος στον καταχωρητή W, με την εντολή RLF K, 0.*



RLF K, 1

Σχήμα 5.25β *Αριστερή περιστροφή του περιεχομένου του καταχωρητή K, μέσω σημαίας C, και αποθήκευση του αποτελέσματος στον καταχωρητή K, με την εντολή RLF K, 1.*

Η περιστροφή γίνεται όπως φαίνεται στα σχήματα 5.25α και 5.25β.

Τα ορίσματα της εντολής δηλώνουν ό,τι και στις υπόλοιπες εντολές. Η εντολή επηρεάζει τη σημαία του κρατούμενου, C, αφού αυτή παίρνει μέρος στην περιστροφή. Ακολουθεί ένα παράδειγμα που βοηθά στη κατανόηση της λειτουργίας της εντολής αυτής.

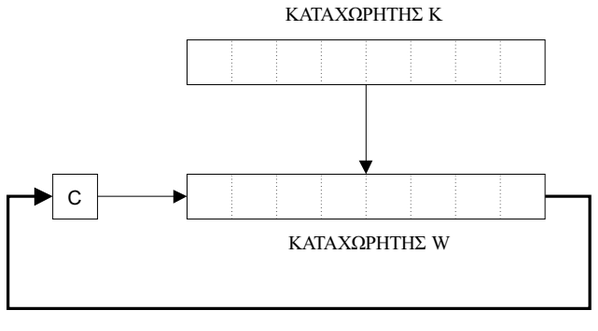
Παράδειγμα
Έστω ότι ο καταχωρητής W = 54 H, ο TMR0 = AB H και η σημαία C = 0.
Δίνουμε την εντολή RLF TMR0, 0. Το αποτέλεσμα θα αποθηκευτεί στον W, αφού το όρισμα d = 0, και θα είναι W = 56 H, ενώ το C = 1. Ο καταχωρητής TMR0 διατηρεί την αρχική τιμή του.

• RRF K, d

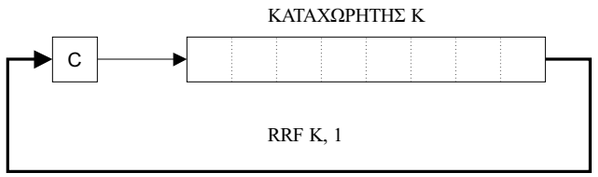
Η δεύτερη εντολή περιστροφής που διαθέτει ο PIC, περιστρέφει το περιεχόμενο ενός καταχωρητή προς τα δεξιά. Ας δούμε, λοιπόν, το συνοπτικό πίνακα της εντολής και ένα επεξηγηματικό σχήμα.

ΣΥΝΤΑΞΗ	RRF K, d
ΟΡΙΣΜΑΤΑ	• Το K παίρνει τιμές από 0 έως 127 ή είναι το συμβολικό όνομα ενός καταχωρητή. • Το d παίρνει τιμές από 0 έως 127.
ΣΗΜΑΙΑ	C

Περιγραφή: Το περιεχόμενο του καταχωρητή K και η σημαία C περιστρέφονται προς τα δεξιά κατά ένα bit. Το αποτέλεσμα αποθηκεύεται στον W, αν το d = 0 (σχήμα 5.26α), ή στον καταχωρητή K, αν το d = 1 (σχήμα 5.26β).



Σχήμα 5.26α *Δεξιά περιστροφή του περιεχομένου του καταχωρητή K, μέσω σημαίας C, και αποθήκευση του αποτελέσματος στον καταχωρητή W, με την εντολή RRF K, 0.*



Σχήμα 5.26β *Δεξιά περιστροφή του περιεχομένου του καταχωρητή K, μέσω σημαίας C, και αποθήκευση του αποτελέσματος στον καταχωρητή K, με την εντολή RRF K, 1.*

Η εντολή είναι παρόμοια με την προηγούμενη. Η διαφορά είναι ότι η περιστροφή γίνεται προς τα δεξιά, όπως φαίνεται στα σχήματα 5.26α και 5.26β.

Επίσης, και εδώ, τα ορίσματα της εντολής δηλώνουν ό,τι και στις υπόλοιπες εντολές. Η εντολή επηρεάζει τη σημαία του κρατούμενου, C, αφού και αυτή παίρνει μέρος στην περιστροφή.

Στην συνέχεια, δίνουμε, ένα παράδειγμα με την εντολή για την πλήρη κατανόηση της λειτουργίας της.

Παράδειγμα

Έστω ότι ο καταχωρητής W = 54 H, ο TMR0 = AB H και η σημαία C = 0, όπως και στο

παράδειγμα της προηγούμενης εντολής.

Δίνουμε την εντολή RRF TMR0, 1.

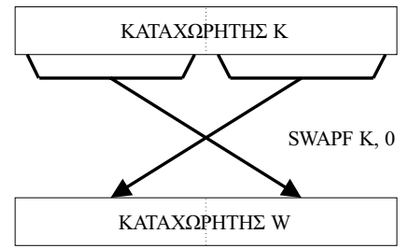
Το αποτέλεσμα θα αποθηκευτεί στον καταχωρητή TMR0, αφού το όρισμα d = 1, και θα είναι TMR0 = 55 H, ενώ το C = 1. Ο καταχωρητής W διατηρεί την αρχική τιμή του, αφού, στην πραγματικότητα δε χρησιμοποιείται.

• SWAPF K, d

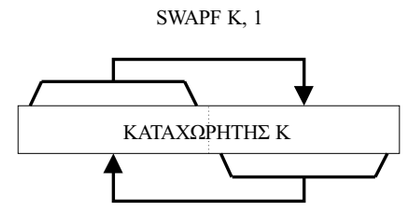
Πέραν των εντολών περιστροφής, ο PIC έχει τη δυνατότητα με μία μόνον εντολή να αντιμεταθέσει τα 4 περισσότερο σημαντικά bits, ενός καταχωρητή, με τα 4 λιγότερο σημαντικά. Ο συνοπτικός πίνακας της εντολής έχει ως εξής:

ΣΥΝΤΑΞΗ	SWAPF K, d
ΟΡΙΣΜΑΤΑ	• Το K παίρνει τιμές από 0 έως 127 ή είναι το συμβολικό όνομα ενός καταχωρητή. • Το d παίρνει τιμές από 0 έως 127.
ΣΗΜΑΙΑ	ΚΑΜΙΑ

Περιγραφή: Τα bits 0 έως 3, του καταχωρητή K, αντιμετατίθενται με τα bits 4 ως 7, του ίδιου καταχωρητή. Το αποτέλεσμα αποθηκεύεται στον W, αν το d = 0 (σχήμα 5.27α), ή στον καταχωρητή K, αν το d = 1 (σχήμα 5.27β).



Σχήμα 5.27α *Αντιμετάθεση των δύο μισών του καταχωρητή K και αποθήκευση του αποτελέσματος στον καταχωρητή W, με την εντολή SWAPF K, 0.*



Σχήμα 5.27β *Αντιμετάθεση των δύο μισών του καταχωρητή K και αποθήκευση του αποτελέσματος στον καταχωρητή K, με την εντολή SWAPF K, 1.*

Τα ορίσματα της εντολής έχουν την ίδια σημασία με αυτά στις υπόλοιπες εντολές. Η εντολή δεν επηρεάζει καμία σημαία. Ακολουθεί ένα παράδειγμα με την εντολή.

Παράδειγμα
Έστω ότι ο καταχωρητής W = 43 H και ο καταχωρητής TMR0 = 61 H.
Δίνουμε την εντολή SWAP TMR0, 1.
Το αποτέλεσμα θα αποθηκευτεί στον καταχωρητή TMR0, αφού το όρισμα d = 1, και θα είναι TMR0 = 16 H, ενώ οι σημαίες δεν επηρεάζονται. Ο καταχωρητής W διατηρεί την αρχική τιμή του, αφού, στην πραγματικότητα δε χρησιμοποιείται.

- **NOP**

Η εντολή αυτή δεν κάνει απολύτως τίποτε παρά να καταλαμβάνει μία θέση στην μνήμη προγράμματος και να εισάγει μία καθυστέρηση ενός κύκλου εντολής.

ΣΥΝΤΑΞΗ	NOP
ΟΡΙΣΜΑΤΑ	ΚΑΝΕΝΑ
ΣΗΜΑΙΑ	ΚΑΜΙΑ

Περιγραφή: Δεν εκτελείται καμία λειτουργία.

Δεν υπάρχει λόγος να παραθέσουμε κάποιο παράδειγμα με την εντολή, αφού είναι πολύ απλή.

Ερωτήσεις

1. Ποιο το αποτέλεσμα της εντολής 'ADDWF TMR0, 1', όπου TMR0=77 H και W=77 H;
2. Σε ποιόν καταχωρητή θα αποθηκευτεί το αποτέλεσμα της προηγούμενης πράξης και ποια θα είναι η τιμή της σημαίας Z;
3. Τι τιμή θα πάρει η σημαία Z αν πριν από την εντολή της πρώτης ερώτησης εκτελέσουμε την εντολή 'COMF TMR0, 0';
4. Ποια η τιμή της σημαίας C μετά την εκτέλεση της εντολής 'RLF TMR0, 0', όπου TMR0=8A H. Σε ποιόν καταχωρητή αποθηκεύεται το αποτέλεσμα της πράξης;

5.5.5 Εντολές επεξεργασίας bit

Πέραν, των εντολών ενός μικροεπεξεργαστή ή μικροελεγκτή που διαχειρίζονται byte, υφίσταται συχνά η ανάγκη να επέμβουμε μόνο σε ένα bit, χωρίς να επηρεάζουμε τα υπόλοιπα bits του καταχωρητή. Ο PIC διαθέτει τέσσερις (4) εντολές που εκτελούν λειτουργίες επεξεργασίας bit σε έναν καταχωρητή, τις οποίες θα δούμε ακολούθως.

- **BSF K, b**

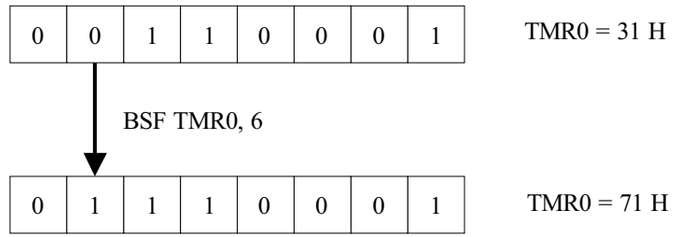
Η πρώτη εντολή χρησιμοποιείται για να θέσει στο 1 ένα bit ενός καταχωρητή. Στη συνέχεια δίνουμε τον πίνακα με τα κυριότερα στοιχεία της εντολής.

ΣΥΝΤΑΞΗ	BSF K, b
ΟΡΙΣΜΑΤΑ	• Το K παίρνει τιμές από 0 έως 127 ή είναι το συμβολικό όνομα ενός καταχωρητή. • Το b παίρνει τιμές από 0 έως 7.
ΣΗΜΑΙΑ	ΚΑΜΙΑ

Περιγραφή: Το bit του καταχωρητή K, που η θέση του δηλώνεται από τον αριθμό b, γίνεται 1.

Ο καταχωρητής επιλέγεται όπως και στις προηγούμενες περιπτώσεις. Το b δηλώνει την θέση του bit, που θα γίνει 1, στον καταχωρητή. Παρακάτω, βλέπουμε και ένα παράδειγμα.

Παράδειγμα
Έστω ότι TMR0 = 31 H (0011 0001B).
Δίνουμε την εντολή BSF TMR0, 6.
Τότε, το αποτέλεσμα γίνεται:
TMR0 = 71 H (0111 0001B)
Οι σημαίες δεν επηρεάζονται, ενώ το bit 6 γίνεται 1 ανεξάρτητα από την τιμή που είχε προηγουμένως (σχήμα 5.28).



Σχήμα 5.28 *Τοποθέτηση του 1 στο 6ο ψηφίο του καταχωρητή TMR0, με την εντολή BSF TMR0, 6.*

- **BCF K, b**

Η δεύτερη εντολή χρησιμοποιείται για να μηδενίσει ένα bit ενός καταχωρητή.

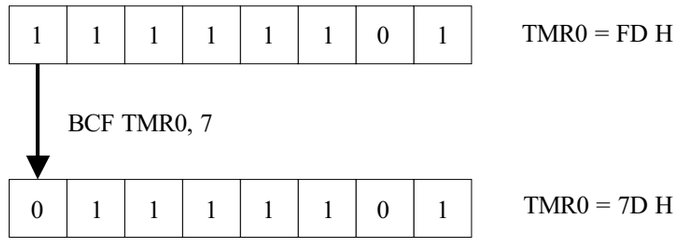
ΣΥΝΤΑΞΗ	BCF K, b
ΟΡΙΣΜΑΤΑ	• Το K παίρνει τιμές από 0 έως 127 ή είναι το συμβολικό όνομα ενός καταχωρητή. • Το b παίρνει τιμές από 0 έως 7.

ΣΗΜΑΙΕΣ	ΚΑΜΙΑ

Περιγραφή: Το bit b του καταχωρητή γίνεται 0.

Για τα ορίσματα ισχύει ό,τι και στην προηγούμενη εντολή, μόνο που στην περίπτωση αυτή το b δείχνει το bit που θα γίνει 0. Για να κατανοήσουμε καλύτερα την εντολή, παραθέτουμε ένα παράδειγμά της.

Παράδειγμα
Έστω ότι TMR0 = FD H (1111 1101B).
Δίνουμε την εντολή BSF TMR0, 7.
Τότε, το αποτέλεσμα γίνεται:
TMR0 = 7D H (0111 1101B)
Οι σημαίες δεν επηρεάζονται, ενώ το bit 7 γίνεται 0 ανεξάρτητα από την προηγούμενη τιμή του (σχήμα 5.29).



Σχήμα 5.29 *Τοποθέτηση του 0 στο 7ο ψηφίο του καταχωρητή TMR0, με την εντολή BCF TMR0, 7.*

• BTFSS K, b

Η παρούσα εντολή ελέγχει την τιμή του bit, που η θέση στον καταχωρητή K του δηλώνεται με τον αριθμό b, και αν το bit αυτό είναι 1 δεν εκτελείται η αμέσως επόμενη εντολή.

ΣΥΝΤΑΞΗ	BTFSS K, b
ΟΡΙΣΜΑΤΑ	• Το K παίρνει τιμές από 0 έως 127 ή είναι το συμβολικό όνομα ενός καταχωρητή. • Το b παίρνει τιμές από 0 έως7.
ΣΗΜΑΙΕΣ	ΚΑΜΙΑ

Περιγραφή: Ελέγχεται το bit, που η θέση του στον καταχωρητή K δηλώνεται με τον αριθμό b και αν είναι 1, τότε εκτελείται η μεθεπόμενη εντολή ενώ αν είναι 0, εκτελείται η επόμενη.

Το b δηλώνει την θέση του bit στον καταχωρητή K. Σε περίπτωση που διαπιστωθεί ότι το συγκεκριμένο bit είναι 1, τότε, εκτελείται η μεθεπόμενη εντολή. Η διαδικασία διαρκεί 2 κύκλους μηχανής, όπως έχουμε εξηγήσει στην ενότητα 5.4. Αν ο έλεγχος διαπιστώσει ότι το bit είναι 0, τότε, η εκτέλεση του προγράμματος γίνεται κανονικά. Το bit του καταχωρητή δεν αλλάζει ούτε οι σημαίες επηρεάζονται.

Παράδειγμα
Έστω το ακόλουθο πρόγραμμα:

1. MOVLW 84 H
2. MOVWF TMR0, 1
3. BTFSS TMR0, 2
4. BSF TMR0, 1
5. BSF TMR0, 3

Το πρόγραμμα, με τις δύο πρώτες εντολές, φορτώνει στον καταχωρητή TMR0 την τιμή 84 H (1000 0100 B). Η εντολή BTFSS TMR0, 2, ελέγχει τον καταχωρητή και βλέπει ότι το bit 2 (αρίθμηση από 0 ως 7) είναι 1. Τότε, εκτελεί την πέμπτη εντολή και παρακάμπτει την τέταρτη. Μετά την εκτέλεση του προγράμματος ο καταχωρητής TMR0 = 8C H.

Αν, αρχικά, ο καταχωρητής TMR0 λάμβανε την τιμή 80 H, τότε το πρόγραμμα θα εκτελούσε και την τέταρτη εντολή. Έτσι, μετά την εκτέλεσή του, ο καταχωρητής TMR0 = 8A H.

Ας προσέξουμε ότι, και στις δύο περιπτώσεις, το πρόγραμμα θέλει 5 κύκλους μηχανής για να εκτελεσθεί.

• BTFSC K, b

Πρόκειται για εντολή παρόμοια με την προηγούμενη. Η διαδικασία είναι ακριβώς η ίδια αλλά εκτελείται όταν το συγκεκριμένο bit είναι 0 αντί 1.

ΣΥΝΤΑΞΗ	BTFSC K, b
ΟΡΙΣΜΑΤΑ	• Το K παίρνει τιμές από 0 έως 127 ή είναι το συμβολικό όνομα ενός καταχωρητή. • Το b παίρνει τιμές από 0 έως 7.
ΣΗΜΑΙΕΣ	ΚΑΜΙΑ

Περιγραφή: Ελέγχεται το bit, που η θέση του στον καταχωρητή K δηλώνεται με τον αριθμό b και αν είναι 0, τότε εκτελείται η μεθεπόμενη εντολή ενώ αν είναι 1, εκτελείται η επόμενη.

Δεν θα επεκταθούμε περισσότερο, αφού, όπως είπαμε, είναι παρόμοια με την προηγούμενη.

Ερωτήσεις

1. Έστω ότι TMR0=35 H. Ποια η τιμή του καταχωρητή αυτού μετά την εκτέλεση της εντολής 'BSF TMR0, 1';
2. Ποια η τιμή του καταχωρητή της προηγουμένης ερώτησης μετά την εκτέλεση της εντολής 'BCF TMR0, 4';
3. Αν αρχικά ο καταχωρητής TMR0=35 H, ποια τιμή θα έχει μετά την εκτέλεση του παρακάτω προγράμματος:

```
BTFSC TMR0, 3
BSF TMR0, 1
BSF TMR0, 7
```

5.5.6 Εντολές άλματος (αλλαγής ροής προγράμματος)

Τελευταία, αφήσαμε την κατηγορία εντολών άλματος, οι οποίες προκαλούν αλλαγή στην ροή του προγράμματος. Σε ένα πρόγραμμα οι εντολές εκτελούνται με την σειρά με την οποία είναι γραμμένες. Όμως, συχνά, είναι απαραίτητο να αλλάξουμε την ροή αυτή. Ο PIC υποστηρίζει την διαδικασία αυτή μέσω κάποιων εντολών που διαθέτει. Ας δούμε, τώρα, την πρώτη από τις εντολές αυτές, ενώ άλλες τρεις θα περιγράψουμε αφού μιλήσουμε για τις ρουτίνες.

• GOTO Διεύθυνση

Μετά την εκτέλεση της εντολής GOTO *Διεύθυνση* η εκτέλεση του προγράμματος συνεχίζει με την εντολή στη θέση *Διεύθυνση*.

ΣΥΝΤΑΞΗ	GOTO Διεύθυνση
ΟΡΙΣΜΑΤΑ	Η Διεύθυνση παίρνει τιμές από 0 έως 2047.
ΣΗΜΑΙΕΣ	ΚΑΜΙΑ

Περιγραφή: Ο καταχωρητής PC φορτώνει τη Διεύθυνση που δίνουμε.

Επίσης, πρέπει να τονίσουμε ότι η εντολή εκτελείται σε 2 κύκλους εντολής, όπως είπαμε και στην ενότητα 5.5.

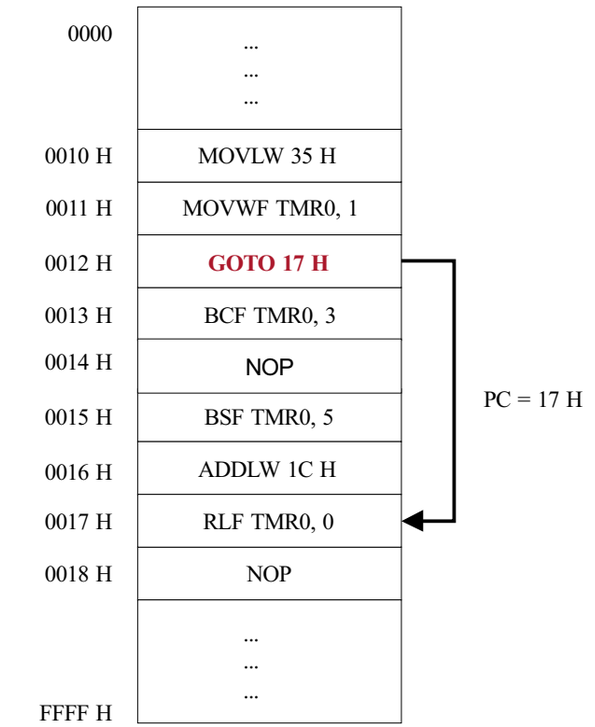
Ακολουθεί ένα παράδειγμα με την εντολή αυτή.

Παράδειγμα

Έστω το ακόλουθο πρόγραμμα, το οποίο αρχίζει στην διεύθυνση 10 H: 010 H: MOVLW 35 H
011 H: MOVWF TMR0, 1
012 H: GOTO 17 H
013 H: BCF TMR0, 3

014 H: NOP
015 H: BSF TMR0, 5
016 H: ADDLW 1C H
017 H: RLF TMR0, 0
018 H: NOP

Η εντολή GOTO φορτώνει τον καταχωρητή PC με την τιμή 17 H. Έτσι, μετά την εντολή αυτή, ο PIC θα εκτελέσει την εντολή που βρίσκεται στη διεύθυνση 17 H, δηλαδή την RLF TMR0, 0 (σχήμα 5.30).



Σχήμα 5.30 *Η εκτέλεση του προγράμματος από τη θέση 12 H της μνήμης προγράμματος, συνεχίζεται στη θέση 17 H, με την εντολή GOTO 17 H, η οποία φορτώνει στον καταχωρητή PC την τιμή 17 H.*

Το πρόγραμμα θα εκτελεστεί σε 6 κύκλους εντολής.

5.6 Μεθοδολογία προγραμματισμού σε γλώσσα Assembly

Στην προηγούμενη ενότητα είδαμε αναλυτικά τις εντολές του PIC. Εύκολα, μπορούμε να παρατηρήσουμε ότι δεν υπάρχουν εντολές αντίστοιχες με αυτές των γλωσσών

υψηλού επιπέδου, όπως αυτών της γλώσσας C ή BASIC. Στην πραγματικότητα, κάθε εντολή των γλωσσών υψηλού επιπέδου μεταφράζεται σε μία σειρά εντολών Assembly από το σύστημα.

Ας μην ξεχνάμε τη φύση της γλώσσας Assembly. Στις άλλες γλώσσες προγραμματίζουμε γράφοντας κώδικα που μοιάζει με τον τρόπο που σκεφτόμαστε. Η Assembly βρίσκεται πολύ κοντά στην μηχανή. Έτσι, όταν γράφουμε κώδικα, πρέπει να έχουμε στο μυαλό μας τον τρόπο με τον οποίο λειτουργεί ο μικροελεγκτής. Αυτό σημαίνει ότι πρέπει να αναλύουμε το εκάστοτε πρόβλημα στις στοιχειώδεις συνιστώσες του. Να εξετάζουμε κάθε μία ξεχωριστά και, κατόπιν, να βλέπουμε πώς μπορούμε να τις συνθέσουμε για να το λύσουμε.

Στην παρούσα ενότητα θα δούμε πώς μπορούμε να υλοποιήσουμε διάφορες λογικές, οι οποίες είναι απαραίτητες στον προγραμματισμό. Συγκεκριμένα, θα εξετάσουμε πώς μπορούμε να γράψουμε σε γλώσσα Assembly τις ακόλουθες διαδικασίες:

- Ανάγνωση - Εγγραφή μνήμης
- Ρουτίνες
- Αλλαγή ροής προγράμματος υπό συνθήκη
- Χρονική καθυστέρηση
- Βρόχος WHILE - DO

Οι διαδικασίες που θα παρουσιάσουμε είναι υποδειγματικές του τρόπου που λειτουργεί ο PIC. Τονίζουμε ότι δεν είναι οι μόνες που μπορεί να υλοποιήσει ο μικροελεγκτής μας. Κάθε ζητούμενη διαδικασία είναι δυνατόν, με τον κατάλληλο προγραμματισμό, να υλοποιηθεί. Ωστόσο, η ανάλυση των παραπάνω διαδικασιών και η προσεκτική εξέταση του τρόπου λειτουργίας τους, θα μας δώσει τη δυνατότητα να γράψουμε κώδικα για παρόμοιες περιπτώσεις που είναι πιθανό να συναντήσουμε.

Πριν, όμως, προχωρήσουμε στην επεξήγηση όλων αυτών, ας δούμε τι μπορεί να περιέχει μία γραμμή κώδικα Assembly. Πέραν της εντολής και των ορισμάτων της, η γραμμή κώδικα μπορεί να περιέχει μία ετικέτα και σχόλια. Η διάταξη είναι η εξής:

	Ετικέτα	**Εντολή**	**Ορίσματα**	**;Σχόλια**
Π.χ.	ROUT1	MOVLW	45 H	;W = 45 H

Η ετικέτα είναι ένα συμβολικό όνομα της διεύθυνσης του προγράμματος, ενώ τα σχόλια χωρίζονται από την υπόλοιπη γραμμή με το σύμβολο ';'.

5.6.1 Ανάγνωση - Εγγραφή μνήμης

Είναι αδύνατο να υπάρξει πρόγραμμα που να μη γράφει στη μνήμη δεδομένα ή, έστω, να διαβάζει από αυτήν. Στις γλώσσες υψηλού επιπέδου, μπορούμε να εκτελέσουμε αυτές τις διαδικασίες με απλές εντολές, όπως READ (δεδομένο) ή WRITE (δεδομένο), χωρίς να ενδιαφερόμαστε για τη φυσική θέση που αυτό αποθηκεύεται στην μνήμη δεδομένων.

Όταν γράφουμε ένα πρόγραμμα σε γλώσσα Assembly, τότε, είμαστε υποχρεωμένοι να επιλέξουμε εμείς πού ακριβώς στην μνήμη θα γραφεί το συγκεκριμένο δεδομένο, δηλαδή σε ποια διεύθυνσή της. Αυτός είναι ένας λόγος για τον οποίο τα προγράμματα, που είναι γραμμένα στην γλώσσα αυτή, είναι αρκετά πολύπλοκα. Όμως, για τον ίδιο λόγο, με την γλώσσα αυτή, μπορούμε να ελέγξουμε πλήρως τον μικροελεγκτή μας.

Όπως είδαμε και στην αρχή του κεφαλαίου μας, η μνήμη του PIC αναφέρεται ως αρχείο καταχωρητών. Στο μικροελεγκτή μας, λοιπόν, οι θέσεις μνήμης αναφέρονται και σαν καταχωρητές. Για παράδειγμα, η θέση μνήμης με διεύθυνση 30 H, λέγεται και καταχωρητής 30 H. Στην συνέχεια, θα εξετάσουμε δύο αντιπροσωπευτικές περιπτώσεις ανάγνωσης και εγγραφής στους καταχωρητές αυτούς:

- **Εγγραφή δεδομένου σε καταχωρητή**

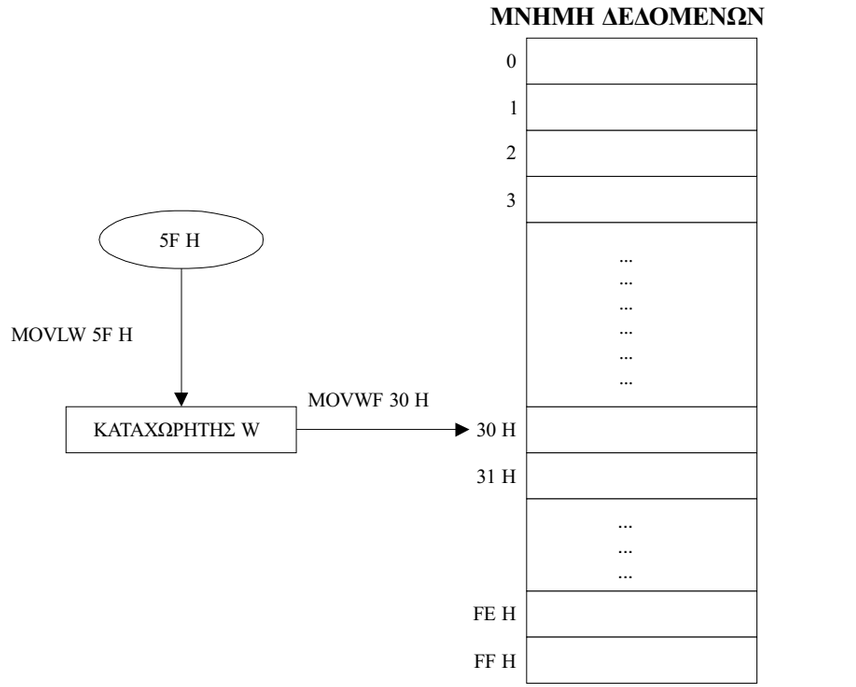
Η διαδικασία της εγγραφής δεδομένου σε έναν καταχωρητή αποτελεί βασικό στοιχείο κάθε προγράμματος. Μέσα σε κάθε πρόγραμμα που φτιάχνουμε, θα χρειαστεί πολλές φορές να γράψουμε ένα δεδομένο σε έναν καταχωρητή, δηλαδή σε μία θέση μνήμης. Ο PIC μας δίνει τη δυνατότητα να κάνουμε την εγγραφή αυτή χρησιμοποιώντας μία εντολή μόνον στον καταχωρητή W. Για όλους τους άλλους καταχωρητές πρέπει να χρησιμοποιήσουμε τον καταχωρητή W σαν ενδιάμεσο βήμα.

Για παράδειγμα, έστω ότι θέλουμε να γράψουμε την τιμή 5F H στον καταχωρητή (θέση μνήμης) 30 H, τότε θα χρειαστούμε τις ακόλουθες εντολές:

Εντολές:

```
MOVLW 5F H   ; Φόρτωσε στον W το 5F H
MOVWF 30 H   ; Φόρτωσε στον 30 H την τιμή του W
```

Μετά την εκτέλεση των εντολών ο καταχωρητής 30 H παίρνει την τιμή 5F H (σχήμα 5.31).



Σχήμα 5.31 *Εγγραφή του δεδομένου 5F H στον καταχωρητή 30 H, με την διαδοχική εκτέλεση των εντολών MOVLW 5F H και MOVWF 30 H.*

• Εγγραφή περιεχομένου καταχωρητή σε καταχωρητή

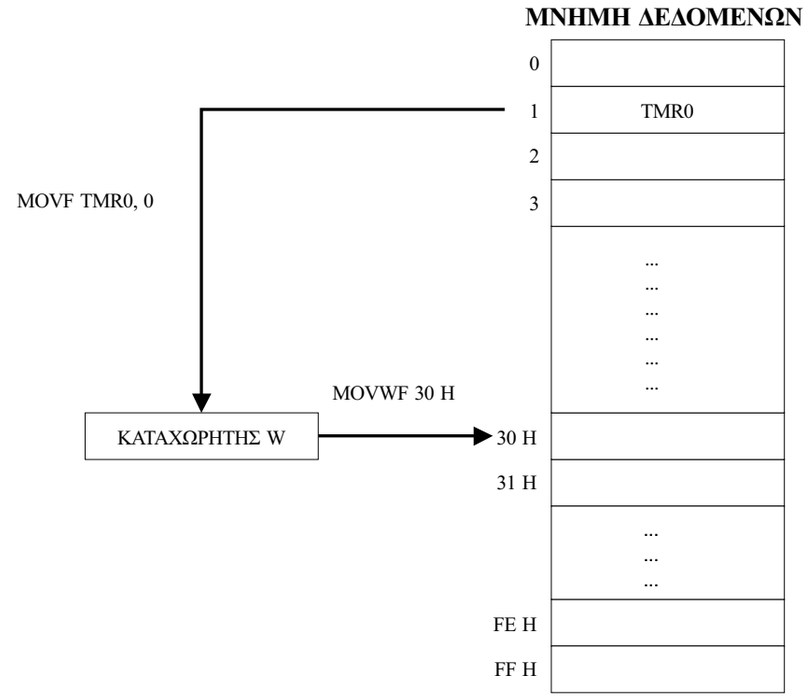
Η δεύτερη διαδικασία, που θα δούμε, αφορά την εγγραφή του περιεχομένου ενός καταχωρητή σε έναν άλλο. Είναι, και αυτή, μία διαδικασία που θα χρειαστεί να την επαναλάβουμε πολλές φορές μέσα σε κάθε πρόγραμμα που γράφουμε. Όπως και στην προηγούμενη διαδικασία, που είδαμε, έτσι και εδώ, ο PIC μας παρέχει τη δυνατότητα να κάνουμε την εγγραφή αυτή χρησιμοποιώντας μία εντολή μόνον όταν ο ένας από τους δύο καταχωρητές είναι ο W. Σε κάθε άλλη περίπτωση, εγγραφής του περιεχομένου ενός καταχωρητή σε έναν άλλο, πρέπει να χρησιμοποιήσουμε τον καταχωρητή W σαν ενδιάμεσο βήμα.

Για παράδειγμα, έστω ότι θέλουμε να γράψουμε το περιεχόμενο του καταχωρητή TMR0 στον καταχωρητή (θέση μνήμης δεδομένων) 30 H, τότε θα χρειαστούμε τις ακόλουθες εντολές:

• Εντολές:

```
MOVF TMR0, 0   ; Φόρτωσε στον W την τιμή του TMR0
MOVWF 30 H     ; Φόρτωσε στον 30 H την τιμή του W
```


Μετά την εκτέλεση των εντολών ο καταχωρητής 30 H παίρνει την τιμή του καταχωρητή TMR0 (σχήμα 5.32).



Σχήμα 5.32 *Εγγραφή του περιεχομένου του καταχωρητή TMR0 στον καταχωρητή 30 H, με την διαδοχική εκτέλεση των εντολών MOVF TMR0, 0 και MOVWF 30 H.*

Σημείωση: Τα παραπάνω είναι μόνον παραδείγματα. Οι διαδικασίες αυτές ισχύουν για οποιονδήποτε καταχωρητή της μνήμης δεδομένων και για οποιαδήποτε τιμή δεδομένου.

5.6.2 Ρουτίνες

Η έννοια της ρουτίνας, στη γλώσσα Assembly, είναι ίδια με αυτή των γλωσσών υψηλού επιπέδου. Ένα καλά δομημένο πρόγραμμα χρησιμοποιεί ρουτίνες όταν υπάρχει σειρά εντολών που επαναλαμβάνεται συχνά.

Η πρώτη γραμμή της ρουτίνας δηλώνεται με μία ετικέτα, η οποία αποτελεί και το όνομα της ρουτίνας. Η ρουτίνα τελειώνει με την εντολή RETURN, η οποία επιστρέφει τον έλεγχο στο κυρίως πρόγραμμα.

Για παράδειγμα, αν θέλουμε, μέσα σε ένα πρόγραμμα, να διαβάζουμε τακτικά τα περιεχόμενα του καταχωρητή PORTA, με διεύθυνση 5, και να τα αποθηκεύουμε στον καταχωρητή TMR0, τότε μπορούμε να γράψουμε μία ρουτίνα με ένα όνομα, π.χ. READ_PORTA, ως εξής:

```
READ_PORTA   MOVF 5, 0        ; Φόρτωσε τα περιεχόμενα του καταχωρητή
                              ; PORTA στον καταχωρητή W.
             MOVWF TMR0       ; Φόρτωσε τα περιεχόμενα του καταχωρητή W
                              ; στον καταχωρητή TMR0.
             RETURN           ; Επέστρεψε στο κυρίως πρόγραμμα.
```

Η ρουτίνα καλείται μέσα σε ένα πρόγραμμα με την εντολή CALL, ως εξής:

```
Εντολή 1
Εντολή 2
...
...
...
Εντολή k
CALL READ_PORTA
Εντολή k+2
...
...
...
CALL READ_PORTA
...
...
...
```

Στην συνέχεια θα περιγράψουμε την εντολή CALL και δύο ακόμη εντολές του μικροελεγκτή μας, οι οποίες χρησιμοποιούνται στις ρουτίνες:

• CALL Όνομα ή Διεύθυνση

Με την εντολή αυτή ο PIC εκτελεί ως επόμενη εντολή την πρώτη εντολή της ρουτίνας που βρίσκεται στην θέση που δείχνουν το Όνομα ή η Διεύθυνση.

ΣΥΝΤΑΞΗ	CALL Όνομα ή Διεύθυνση
ΟΡΙΣΜΑΤΑ	Όνομα ή διεύθυνση ρουτίνας
ΣΗΜΑΙΕΣ	ΚΑΜΙΑ

Περιγραφή: Η εντολή καλεί τη ρουτίνα με το όνομά της ή με την διεύθυνση στην οποία αρχίζει. Πιο συγκεκριμένα ο PC φορτώνεται με την διεύθυνση της ρουτίνας.

Σε σχέση με την εντολή GOTO, η παρούσα εντολή κάνει και μία επιπλέον ενέργεια. Αποθηκεύει τη διεύθυνση της αμέσως επόμενης εντολής, δηλαδή την PC + 1, σε μία ειδική μνήμη που λέγεται σωρός. Έτσι, όταν ο PIC τελειώσει την εκτέλεση της

ρουτίνας, θα μπορέσει να επιστρέψει, μετά την CALL, στην εντολή. Στην επόμενη εντολή, που θα περιγράψουμε, θα μιλήσουμε περισσότερο για τη σημασία αυτής της ενέργειας.

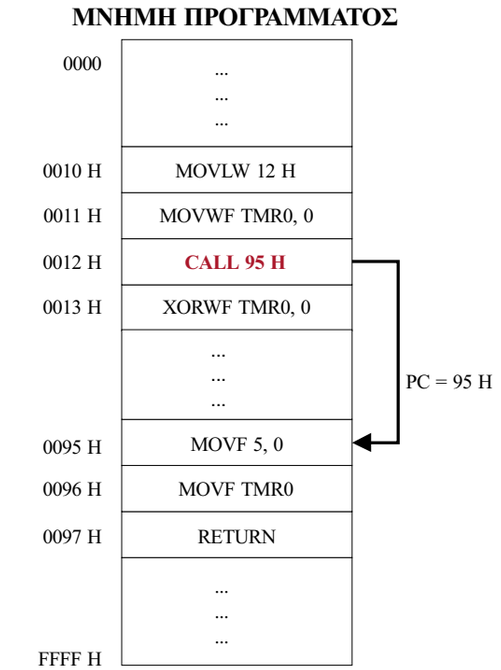
Και αυτή η εντολή εκτελείται σε 2 κύκλους εντολής. Ακολουθεί ένα παράδειγμα, το οποίο υποδεικνύει τη χρήση της εντολής:

Παράδειγμα

Έστω το ακόλουθο πρόγραμμα, το οποίο αρχίζει στη διεύθυνση 10 H και ότι η ρουτίνα που θα καλέσει είναι η READ_PORTA, την οποία είδαμε πριν λίγο, και αρχίζει στην διεύθυνση 95 H:

```
010 H: MOVLW 12 H       ;Φορτώνει στον W την τιμή 12 H.
011 H: MOVWF TMR0, 0    ;Φορτώνει στον TMR0 την τιμή  0.
012 H: CALL 95 H        ;Καλεί την READ_PORTA (ή CALL READ_PORTA).
013 H: XORWF TMR0, 0    ;Εκτελεί την λογική πράξη XOR μεταξύ των W και TMR0.
.
.
.
```

εντολή αυτή, ο PIC θα εκτελέσει τη ρουτίνα, της οποίας η πρώτη εντολή αρχίζει στη διεύθυνση 95 H (σχήμα 5.33).



Σχήμα 5.33 *Κλήση της ρουτίνας, που ξεκινά από την διεύθυνση 95 H, με την εντολή CALL 95 H, η οποία φορτώνει στον καταχωρητή PC την τιμή 95 H.*

Ας θυμηθούμε, εδώ, ότι αντί του 95 H θα μπορούσαμε να χρησιμοποιήσουμε και το όνομα της ρουτίνας, δηλαδή το READ_PORTA.

Όταν, η ρουτίνα τελειώσει, ο PIC θα εκτελέσει την αμέσως επόμενη της CALL, εντολή, δηλαδή την XORWF TMR0, 0. Η διεύθυνσή της φορτώνεται στον καταχωρητή PC στο τέλος της ρουτίνας. Στο παράδειγμά μας, η διεύθυνση αυτή είναι 13 H.

• RETURN

Μετά την εκτέλεσή της, μία ρουτίνα πρέπει να επιστρέψει στην επόμενη εντολή της CALL και να συνεχίσει ομαλά την εκτέλεση του κυρίως προγράμματος. Στην προηγούμενη εντολή είδαμε ότι, για να γίνει αυτό, χρειάζεται η διεύθυνση αυτή να έχει αποθηκευτεί στο σωρό, έτσι ώστε να είναι δυνατή η ανάκλησή της, με την κατάλληλη εντολή. Ο πίνακας της εντολής αυτής, παρατίθεται ακολούθως.

ΣΥΝΤΑΞΗ	RETURN
ΟΡΙΣΜΑΤΑ	KANENA
ΣΗΜΑΙΕΣ	KAMIA

Περιγραφή: Μετά την εκτέλεση μίας ρουτίνας, επιστρέφει τον έλεγχο από την ρουτίνα στο κυρίως πρόγραμμα, στην επόμενη εντολή της CALL από την οποία κλήθηκε η ρουτίνα αυτή.

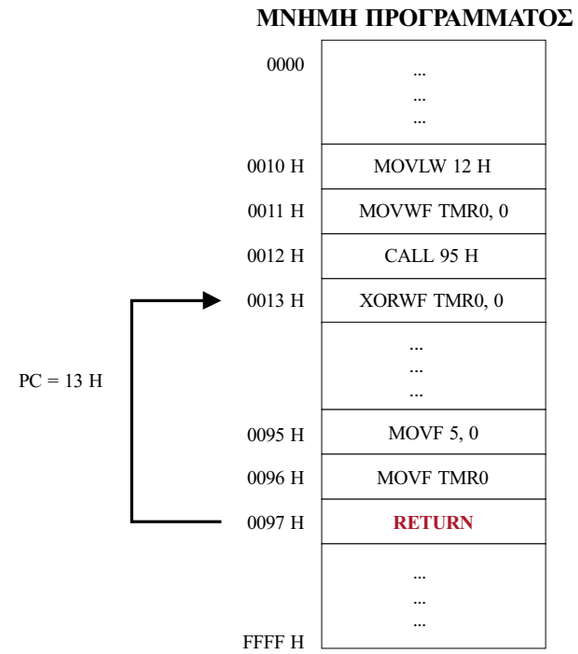
Με αυτήν την εντολή πρέπει να τελειώνει κάθε ρουτίνα που γράφουμε. Και αυτή η εντολή εκτελείται σε 2 κύκλους εντολής. Ακολουθεί ένα παράδειγμα, το οποίο υποδεικνύει τη χρήση της εντολής.

Παράδειγμα

Έστω το πρόγραμμα που είδαμε στο παράδειγμα της εντολής CALL. Το κυρίως πρόγραμμα αρχίζει στη διεύθυνση 10 H, ενώ η ρουτίνα READ_PORTA αρχίζει στη διεύθυνση 95 H.

Σύμφωνα με τα όσα είπαμε για την εντολή CALL, όταν αυτή εκτελεσθεί, ο έλεγχος μεταφέρεται στη διεύθυνση 95 H, δηλαδή ο PC από 12H γίνεται 95 H, αντί 13 H, που είναι η διεύθυνση της επόμενης, κατά σειρά, εντολής. Πριν όμως φορτωθεί στον PC η τιμή 95 H, η τιμή PC + 1, δηλαδή η 13 H, αποθηκεύεται στο σωρό.

Οι εντολές της ρουτίνας θα εκτελεσθούν κανονικά, μία-μία. Στο τέλος, η εντολή RETURN θα ανακαλέσει την αποθηκευμένη τιμή PC + 1, δηλαδή την 13 H, και θα την τοποθετήσει στον PC (σχήμα 5.34). Έτσι, ο έλεγχος θα επιστρέψει στο κυρίως πρόγραμμα και ο PIC θα εκτελέσει την αμέσως επόμενη της CALL, εντολή, δηλαδή την XORWF TMR0, 0.



Σχήμα 5.34 *Επιστροφή, από την ρουτίνα στο κυρίως πρόγραμμα, με την εντολή RETURN, η οποία φορτώνει στον καταχωρητή PC την τιμή 13 H.*

• RETFIE

Ας παρουσιάσουμε, τώρα, μία παραλλαγή της προηγούμενης εντολής, Η εντολή χρησιμοποιείται στην περίπτωση των διακοπών, για τις οποίες θα μιλήσουμε σε επόμενη ενότητα.

ΣΥΝΤΑΞΗ	RETFIE
ΟΡΙΣΜΑΤΑ	KANENA
ΣΗΜΑΙΕΣ	KAMIA

Περιγραφή: Επιστρέφει τον έλεγχο στο κυρίως πρόγραμμα, μετά από την εκτέλεση μίας ρουτίνας διακοπής και ενεργοποιεί τις διακοπές του συστήματος.

Με αυτήν την εντολή πρέπει να τελειώνει κάθε ρουτίνα που εξυπηρετεί διακοπές. Η μόνη της διαφορά από την προηγούμενη είναι ότι ενεργοποιεί τις διακοπές του συστήματος.

5.6.3 Αλλαγή ροής προγράμματος υπό συνθήκη

Υπάρχει το ενδεχόμενο να θέλουμε να εκτελέσουμε ένα τμήμα κώδικα, για

παράδειγμα μία ρουτίνα, μόνον κάτω από ορισμένες συνθήκες.

Στη γλώσσα Assembly ο έλεγχος της ροής του προγράμματος ασκείται μέσω των σημαιών. Πολλές εντολές επηρεάζουν τις σημαίες. Για παράδειγμα, μερικές από τις εντολές, όταν μηδενίζουν το περιεχόμενο ενός καταχωρητή, θέτουν τη σημαία του μηδενισμού (Z), στο 1. Είναι δυνατό, βλέποντας την κατάσταση των σημαιών, να συμπεράνουμε έμμεσα την κατάσταση ενός καταχωρητή.

Έστω, λοιπόν, ότι θέλουμε να εκτελέσουμε την ρουτίνα, του παραδείγματος της προηγούμενης ενότητας, μόνον αν ο καταχωρητής W είναι 0. Τότε μπορούμε να συμπληρώσουμε το κυρίως πρόγραμμα ως εξής:

```
...
IORLW 0              ; W = W OR 0
BTFSS STATUS, 2      ; Αν Z = 1 παράκαμψε την επομένη εντολή.
CALL READ_PORTA      ; Κάλεσε την ρουτίνα READ_PORTA. ...
...
...
IORLW 0              ; W = W OR 0
BTFSS STATUS, 2      ; Αν Z = 1 παράκαμψε την επομένη εντολή.
CALL READ_PORTA      ; Κάλεσε την ρουτίνα READ_PORTA.
...
...
...
...
...
```

Όπως βλέπουμε, χρησιμοποιούμε ένα τέχνασμα για να διαπιστώσουμε αν ο καταχωρητής W είναι 0. Εκτελούμε τη λογική πράξη Η (OR) μεταξύ του περιεχομένου του καταχωρητή W και του αριθμού 0. Στην ουσία η πράξη αφήνει το περιεχόμενο του καταχωρητή W το ίδιο. Άρα, η μόνη περίπτωση να έχουμε μηδενικό αποτέλεσμα είναι όταν η αρχική τιμή του W είναι 0. Σε περίπτωση που το αποτέλεσμα είναι 0, η εντολή, που χρησιμοποιούμε, θέτει την σημαία Z στο 1. Αντίθετα, αν το αποτέλεσμα είναι διάφορο του μηδενός το Z γίνεται 0. Συνεπώς, με μία εντολή και χωρίς στην ουσία να αλλάξουμε το περιεχόμενο του W, μπορούμε να πάρουμε στη σημαία Z την ένδειξη του κατά πόσο ο καταχωρητής W είναι 0. Παρόμοια τεχνάσματα χρησιμοποιούμε συχνά στον προγραμματισμό σε γλώσσα Assembly.

Αμέσως μετά, ακολουθεί η εντολή BTFSS. Η εντολή αυτή ελέγχει την κατάσταση της σημαίας Z, που είναι το bit 2 του καταχωρητή STATUS, και αν Z = 1 η αμέσως επόμενη εντολή, δηλαδή η CALL, δεν εκτελείται.

Η λύση που ακολουθούμε στο παράδειγμα δεν είναι μοναδική. Σε μία παρόμοια περίπτωση, θα μπορούσαμε να έχουμε την εντολή GOTO αντί της CALL. Επίσης, ας θυμηθούμε ότι υπάρχουν και άλλες εντολές πλην της BTFSS, μπορούν να αλλάξουν τη ροή του προγράμματος ανάλογα με την τιμή της σημαίας μηδενισμού.

5.6.4 Χρονική καθυστέρηση

Υπάρχουν πολλές εφαρμογές μικροϋπολογιστικών συστημάτων, όπου ο υπολογισμός του χρόνου εκτέλεσης ενός ορισμένου τμήματος κώδικα είναι απολύτως απαραίτητος. Παράδειγμα αποτελεί η δημιουργία ενός βρόχου συγκεκριμένης χρονικής καθυστέρησης.

Ας υποθέσουμε, λοιπόν, ότι θέλουμε να φτιάξουμε μία ρουτίνα που κάθε φορά που θακαλείται θα εισάγει μία καθυστέρηση 200μsec στην εκτέλεση του προγράμματος. Το πρώτο πράγμα που πρέπει να ξέρουμε είναι η συχνότητα του ρολογιού του PIC. Αν στην περίπτωσή μας είναι 4MHz, τότε, σύμφωνα με τα όσα είπαμε στην ενότητα 5.4, ο χρόνος εκτέλεσης 1 κύκλου εντολής είναι 4/4MHz, δηλαδή 1μsec. Άρα, τα 200μsec αντιστοιχούν σε 200 κύκλους εντολής.

Κατασκευάζουμε, λοιπόν, μία ρουτίνα, με το όνομα DELAY. Η ρουτίνα χρησιμοποιεί τον καταχωρητή TMR0 σαν μετρητή επαναλήψεων ενός βρόχου. Ο αριθμός των επαναλήψεων που θα κάνει ο βρόχος περνούν στην ρουτίνα από τον καταχωρητή W. Η ρουτίνα έχει ως εξής:

```
DELAY  MOVF TMR0,W     ; TMR0 = W (1 κύκλος)
LOOP1  NOP             ; (1 κύκλος)
       NOP             ; (1 κύκλος)
       DECFSZ TMR0, 1  ; Μείωσε τον TMR0 κατά 1 και αν
                       ; γίνει 0 πήγαινε στο RETURN (1 ή 2 κύκλοι)
       GOTO LOOP1      ; Επανάλαβε τον βρόχο (2 κύκλος)
       NOP             ; (1 κύκλος)
RETURN                 ; Επιστροφή στο κυρίως πρόγραμμα (2 κύκλοι)
```

Το κυρίως πρόγραμμα έχει ως εξής:

```
       ...
       ...
       ...
       MOVLW N         ; όπου N ο αριθμός των επαναλήψεων
CALL DELAY             ; (2 κύκλοι)
       ...
       ...
       ...
```

Στα σχόλια βάζουμε σε παρένθεση τους κύκλους εντολής που θέλει η κάθε μία εντολή για να εκτελεσθεί. Μπορούμε να υπολογίσουμε τον αριθμό τον επαναλήψεων ως εξής:

Ο βρόχος θέλει τους εξής κύκλους εντολής για να εκτελεσθεί:

ΚΥΚΛΟΙ:	(N	- 1) x	(1	+	1	+	1	+	2)	+	(1	+	1	+	2)
ΕΝΤΟΛΕΣ:			NOP		NOP		DECFSZ		GOTO		NOP		NOP		DECFSZ
			Όλοι οι βρόχοι πλην του τελευταίου								Τελευταίος βρόχος				

Επίσης, έχουμε άλλους 4 κύκλους εντολής 1 από την πρώτη εντολή της ρουτίνας, δηλαδή την MOVFW 1, 1 από την προτελευταία εντολή, δηλαδή την NOP, και 2 από την τελευταία, δηλαδή την RETURN. Ακόμη, άλλοι 2 κύκλους θέλει η ίδια η εντολή κλήσης της ρουτίνας, δηλαδή η CALL. Σύνολο, λοιπόν έχουμε 6 επιπλέον κύκλους εντολής.

Αν τα συγκεντρώσουμε όλα αυτά μπορούμε να γράψουμε ότι $5 \times (N - 1) + 10 = 200$, το οποίο μας δίνει ότι ο αριθμός των επαναλήψεων πρέπει να είναι 39.

5.6.5 Βρόχος WHILE - DO

Ας δούμε, τώρα, μία γενικότερη περίπτωση χρήσης βρόχου, δηλαδή την επανάληψη εκτέλεσης ενός συνόλου εντολών. Σε γλώσσες υψηλού επιπέδου ο βρόχος υλοποιείται με εντολές της μορφής WHILE - DO, REPEAT - UNTIL, FOR - DO, κτλ. Για παράδειγμα:

```
Y = 0;
WHILE (συνθήκη, π.χ. Y ? 10)
Εντολή1;
Εντολή2;
...
Αύξησε Y κατά 1;
DO      ; Επαναλαμβάνονται οι εντολές όσο Y ? 10. Αν Y>10 προχωράμε στην
        ; επόμενη, της DO, εντολή.
Ή
Y = 35;
REPEAT
Εντολή1;
Εντολή2;
...
Μείωσε Y κατά 1;
UNTIL (συνθήκη, π.χ. Y = 10)    ; Ο βρόχος αυτός θα εκτελείται μέχρι το Y=10.
Τότε,                           ; προχωράμε επόμενη, της UNTIL, εντολή.
```

Όπου το Y είναι μία μεταβλητή.

Στη λογική αυτή γίνεται χρήση λειτουργιών όπως της αυξομείωσης μετρητή, της αλλαγής ροής προγράμματος υπό συνθήκη, κτλ. Με τη γλώσσα Assembly η

υλοποίηση της λογικής αυτής είναι σαφώς πολυπλοκότερη. Εδώ, πρέπει να ορισθεί ένας καταχωρητής ως μετρητής των επαναλήψεων που θα γίνουν. Επίσης, με κάποιον τρόπο πρέπει να διαπιστωθεί ότι ο μετρητής έφτασε στην τιμή που θέλουμε.

Θα παρακολουθήσουμε, τώρα, τη λύση ενός συνηθισμένου προβλήματος στους μικροελεγκτές.

Πρόβλημα

Έστω, λοιπόν, ότι θέλουμε να φτιάξουμε ένα πρόγραμμα που να προσθέτει 7 φορές τον αριθμό 5 και το αποτέλεσμα να αποθηκεύεται στον καταχωρητή 55 H.

Ανάλυση

Το πρόβλημα αναλύεται σε δύο μέρη:

1. Εύρεση αποτελέσματος.
2. Αποθήκευσή του στον καταχωρητή 55 H.

Επειδή έχουμε επαναλαμβανόμενες εντολές (7 προσθέσεις), μπορούμε να χρησιμοποιήσουμε κάποιο βρόχο.

Λύση

Μέχρι τώρα έχουμε δει πώς μπορούμε να γράψουμε κάτι στη μνήμη και πώς να δημιουργήσουμε βρόχους. Άρα, αν συνδυάσουμε τις γνώσεις που αποκτήσαμε ως τώρα, μπορούμε να γράψουμε τον ακόλουθο κώδικα:

```
        ; Ορίζουμε τον καταχωρητή 21 H, σαν μετρητή των 7 προσθέσεων.
        MOVLW 7         ;W = 7
        MOVWF 21 H      ;Περιεχόμενο καταχωρητή 21 H = 7.

        ; Δίνουμε το 0 σαν αρχική τιμή στον καταχωρητή W.
        MOVLW 0

        ; Αρχή βρόχου
START   ADDLW 5H        ;Προσθέτουμε στον W το 5.
        DECFSZ 21 H, 1  ;Μειώνουμε το περιεχόμενο της διεύθυνσης 21 H κατά 1 και
                        ;αν το αποτέλεσμα είναι 0 πηγαίνουμε στο τέλος, αλλιώς
        GOTO START      ;επαναλαμβάνουμε το βρόχο
        END             ;Τέλος
```

Ερώτηση: Ποια αριθμητική πράξη υλοποιεί, στην πραγματικότητα, ο βρόχος;

5.7 Οι διακοπές του PIC

Όπως είπαμε και στην εισαγωγή του κεφαλαίου, ο PIC δέχεται διακοπές από πολλές διαφορετικές πηγές. Κάθε περιφερειακή μονάδα μπορεί να δώσει μία διακοπή. Μάλιστα, μερικές από αυτές μπορούν να δώσουν περισσότερες από μία, όπως για παράδειγμα η περιφερειακή μονάδα σύγχρονης και ασύγχρονης σειριακής επικοινωνίας (USART).

Θα εξετάσουμε τις βασικές αρχές της λειτουργίας των διακοπών του PIC. Στο επόμενο κεφάλαιο θα εξετάσουμε μερικές από τις περιφερειακές μονάδες του PIC και θα καταλάβουμε, μέσα από παραδείγματα, τη χρησιμότητα των διακοπών για κάθε μία από αυτές.

Σε κάθε διακοπή αντιστοιχούν δύο δυαδικά ψηφία (bits). Το πρώτο χρησιμοποιείται για την ενεργοποίηση ή απενεργοποίησή της, ενώ το δεύτερο, που καλείται σημαία (flag), γίνεται 1 όταν προκαλείται μια διακοπή. Τα bits συμβολίζονται με το όνομα του περιφερειακού, που προκάλεσε την διακοπή, ακολουθούμενο από τα γράμματα IE (interrupt enable), για το πρώτο, και IF (interrupt flag), για το δεύτερο. Για παράδειγμα, για το περιφερειακό TIMER0, το οποίο θα μελετήσουμε στο επόμενο κεφάλαιο, τα αντίστοιχα bits είναι T0IE και T0IF.

Τα bits, που προαναφέραμε, περιέχονται σε καταχωρητές που χρησιμοποιούνται αποκλειστικά για τις διακοπές. Οι καταχωρητές είναι οι:

- **INTCON**
- **PIE1**
- **PIR1**
- **PIE2**
- **PIR2**

Στην συνέχεια θα εξετάσουμε τα bits του καταχωρητή INTCON και θα δούμε συνοπτικά τους υπόλοιπους καταχωρητές, αφού όσα από τα bits τους μας ενδιαφέρουν για το μάθημά μας, θα τα δούμε, αναλυτικά, στο επόμενο κεφάλαιο.

5.7.1 Καταχωρητής INTCON

Ο καταχωρητής αυτός φαίνεται στο σχήμα 5.35. Περιέχει το bit ενεργοποίησης / απενεργοποίησης διακοπής καθώς και το bit της σημαίας της, για τις περιφερειακές μονάδες TIMER0 και θύρα B. Στην δεύτερη η διακοπή προκαλείται αν γίνει αλλαγή σήματος σε ένα από τους ακροδέκτες 4 ως 7. Επίσης, ο καταχωρητής περιέχει τα αντίστοιχα bits για μία εξωτερική διακοπή, που μπορεί να δεχθεί στον ακροδέκτη 0 της θύρας B. Τέλος, τα άλλα δύο bits του καταχωρητή χρησιμοποιούνται για τον έλεγχο της γενικής ενεργοποίησης ή μη όλων των διακοπών, το ένα, και των διακοπών μόνον των περιφερειακών μονάδων, το άλλο.

Τον καταχωρητή μπορούμε όχι μόνον να τον διαβάσουμε αλλά και να τον γράψουμε.

7	6	5	4	3	2	1	0
GIE	**PEIE**	**T0IE**	**INTE**	**RBIE**	**T0IF**	**INTF**	**RBIF**

Σχήμα 5.35: *Ο καταχωρητής INTCON*

Στη συνέχεια περιγράφουμε τη σημασία των bits του:

- **GIE:** Bit ενεργοποίησης όλων των διακοπών.
  1 = Ενεργοποιεί όλες τις μη απενεργοποιημένες διακοπές.
  0 = Απενεργοποιεί όλες τις διακοπές.
- **PEIE:** Bit ενεργοποίησης όλων των διακοπών των περιφερειακών.
  1 = Ενεργοποιεί όλες τις μη απενεργοποιημένες διακοπές των περιφερειακών.
  0 = Απενεργοποιεί όλες τις διακοπές των περιφερειακών.
- **T0IE:** Bit ενεργοποίησης διακοπής υπερχείλισης TIMER0. 1 = Ενεργοποίηση διακοπής υπερχείλισης TIMER0.
  0 = Απενεργοποίηση διακοπής υπερχείλισης TIMER0.
- **INTE:** Bit ενεργοποίησης εξωτερικής διακοπής RB0/INT.
  1 = Ενεργοποίηση εξωτερικής διακοπής RB0/INT.
  0 = Απενεργοποίηση εξωτερικής διακοπής RB0/INT.
- **RBIE:** Bit ενεργοποίησης διακοπής αλλαγής σήματος θύρας B.
  1 = Ενεργοποίηση διακοπής αλλαγής σήματος θύρας B.
  0 = Απενεργοποίηση διακοπής αλλαγής σήματος θύρας B.
- **T0IF:** Σημαία διακοπής υπερχείλισης TIMER0. 1 = Ο καταχωρητής TMR0 έχει υπερχειλίσει.
  0 = Ο καταχωρητής TMR0 δεν έχει υπερχειλίσει.
- **INTF:** Σημαία εξωτερικής διακοπής RB0/INT.
  1 = Υπάρχει εξωτερική διακοπή RB0/INT.
  0 = Δεν υπάρχει εξωτερική διακοπή RB0/INT.
- **RBIF:** Σημαία διακοπής αλλαγής σήματος θύρας B.
  1 = Τουλάχιστον ένας από τους ακροδέκτες 4 ως 7 της θύρας B άλλαξε κατάσταση.
  0 = Κανένας από τους ακροδέκτες 4 ως 7 της θύρας B δεν άλλαξε κατάσταση.

5.7.2 Καταχωρητές PIE1 και PIR1

Ο καταχωρητής PIE1 περιέχει τα bits ενεργοποίησης / απενεργοποίησης και ο καταχωρητής PIR1 τα bits των σημαιών των διακοπών που προκαλούνται από τις ακόλουθες περιφερειακές μονάδες:

1. Θύρα παράλληλης επικοινωνίας (PSP).
2. A/D μετατροπέας.
3. Θύρα σύγχρονης / ασύγχρονης επικοινωνίας (USART).

4. Θύρα σύγχρονης επικοινωνίας (SSP).
5. Μονάδα CCP1
6. TIMER2

5.7.3 Καταχωρητές PIE2 και PIF2

Είναι καταχωρητές αντίστοιχοι με τους PIE1 και PIF1. Δεν θα ασχοληθούμε με αυτούς διότι αναφέρονται σε λειτουργίες του PIC, τις οποίες δεν θα εξετάσουμε.

Ας μιλήσουμε, όμως, λίγο περισσότερο για το τι, στην ουσία, σημαίνει η εμφάνιση μίας διακοπής και πώς καλείται η ρουτίνα εξυπηρέτησής της, για να εκτελεσθεί.
Η εμφάνιση μίας διακοπής σημαίνει δύο πράγματα:

1. Στον καταχωρητή PC φορτώνεται η διεύθυνση 4, της μνήμης προγράμματος
2. Η σημαία του περιφερειακού που προκάλεσε τη διακοπή, εμφανίζεται στον κατάλληλο καταχωρητή.

Συνεπώς, η επόμενη εντολή που θα εκτελεσθεί είναι αυτή που βρίσκεται αποθηκευμένη στην διεύθυνση 4. Όμως, εμείς, με την εμφάνιση της διακοπής, θέλουμε να εκτελείται μία ρουτίνα, η οποία, ασφαλώς, καταλαμβάνει πολύ περισσότερο χώρο από μία θέση μνήμης. Η τεχνική που χρησιμοποιούμε, είναι να γράψουμε τη ρουτίνα σε κάποιες θέσεις μνήμης. Έπειτα, μπορούμε να τοποθετήσουμε στη θέση 4 την εντολή GOTO με όρισμα τη θέση μνήμης από την οποία αρχίζει η ρουτίνα. Για παράδειγμα αν γράψουμε τη ρουτίνα, εξυπηρέτησης της διακοπής, στις θέσεις 83 H ως 9A H και της δώσουμε το όνομα (ετικέτα) ROYTINA1, τότε, στη θέση 4 θα γράψουμε την εντολή GOTO 83 H (ή GOTO ROUTINA1). Αυτός ο τρόπος χρησιμοποιεί τη δυνατότητα της μηχανής (Hardware), με την εμφάνιση μίας διακοπής, να μεταβιβάζει την εκτέλεση του προγράμματος σε άλλη διεύθυνση. Αυτό μπορεί να γραφεί ως εξής:

```
0004 H: GOTO 83 H      Εντολή που εκτελείται με την
                       εμφάνιση της διακοπής
0005 H: ...
...
...
...
001A H: ...
001B H: ...            Κυρίως πρόγραμμα
...
...
...
...
```



```
0083 H: ...
...                    Ρουτίνα εξυπηρέτησης διακοπής
...
009A H: RETFIE
...
```

Ένας δεύτερος τρόπος με τον οποίο χειριζόμαστε τις διακοπές είναι η μέθοδος της δειγματοληψίας (polling). Σε αυτήν την περίπτωση, απενεργοποιούμε τη διακοπή. Στη συνέχεια, ανά τακτά χρονικά διαστήματα, ελέγχουμε μέσα από το πρόγραμμά μας την εμφάνιση ή μη της αντίστοιχης σημαίας. Όταν εμφανιστεί η σημαία καλούμε τη ρουτίνα εξυπηρέτησης της διακοπής και την εκτελούμε.

Για παράδειγμα, η σειριακή θύρα επικοινωνίας USART, όταν μεταδίδει μία σειρά δεδομένων, δηλώνει με τη διακοπή ότι έχει στείλει ένα δεδομένο και είναι έτοιμη να της στείλουμε το επόμενο. Αν θέλουμε να μην διακόπτεται η εκτέλεση του προγράμματός μας, κάθε φορά που συμβαίνει αυτό, και να ελέγχουμε εμείς το πότε θα της στείλουμε το επόμενο δεδομένο για να το μεταδώσει, τότε, ενεργούμε ως ακολούθως:

1. Απενεργοποιούμε τη διακοπή θέτοντας το bit TXIE του καταχωρητή PIE1 στο 0.

2. Γράφουμε μία ρουτίνα εξυπηρέτησης όπου στέλνει το επόμενο δεδομένο στο περιφερειακό.

3. Έπειτα, ελέγχουμε ανά τακτά χρονικά διαστήματα, με τη χρήση ενός βρόχου, αν το bit TXIF του καταχωρητή PIR1 είναι 1. Όταν συμβεί αυτό χρησιμοποιούμε την τεχνική της αλλαγής προγράμματος υπό συνθήκη, που περιγράψαμε στην προηγούμενη ενότητα, για να εκτελέσουμε τη ρουτίνα εξυπηρέτησης του περιφερειακού.

Το πρόγραμμά μας πρέπει να ακολουθεί την ακόλουθη λογική:

```
        BCF PIR1, TXIF    ;Απενεργοποίηση της διακοπής του USART (μετάδοση)

START   ...               ;Έναρξη του κυρίου προγράμματος
        ...
        ...
        ...
        ...       ...
        BTFSC PIR1, TXIF      ;Αν TXIF = 0, παράκαμψη της επομένης εντολής
        CALL ROUT_USART       ;Κλήση της ρουτίνας εξυπηρέτησης του
                              περιφερειακού
        GOTO START            ;Επανάληψη του βρόχου
        ...
        ...
```

```
            ...
            ...
Ρουτίνα εξυπηρέτησης του USART, σε λειτουργία μετάδοσης
ROUT_USART  ...
            ...
            ...
            ...
            RETIF  Επιστροφή από εκεί που σταμάτησε (Δηλαδή,
                   η επόμενη εντολή είναι η GOTO START)
```


1. ***Douglas Hall***, "Micropocessors and Interfacing - Programming and Hardware", εκδόσεις Mc Graw Hill
2. ***Microchip Technology Inc.***, "PICmicro Mid - Range MCU Family", Reference Manual, 1997
3. ***Microchip Technology Inc.***, "PIC 16F87X 28/40-pin 8-Bit CMOS FLASH Microcontrollers", Reference Manual, 1999

Κεφάλαιο 6ο

Χρήση του μικροελεγκτή σε εφαρμογές

Περιεχόμενα

Στόχοι του κεφαλαίου:

Όταν ολοκληρώσεις το κεφάλαιο αυτό θα μπορείς να...

αναφέρεις τις βασικές περιφερειακές συσκευές του μικροελεγκτή PIC.
περιγράφεις πώς γίνεται ο έλεγχος μίας περιφερειακής συσκευής μέσω της γλώσσας Assembly.
διαβάζεις και να εξηγείς κώδικα προγραμματισμού των περιφερειακών συσκευών.
γράφεις στοιχειώδη προγράμματα λειτουργίας των συσκευών αυτών.

Είπαμε και στο προηγούμενο κεφάλαιο ότι η διαφορά ενός μικροελεγκτή από ένα μικροεπεξεργαστή είναι ότι πρώτος διαθέτει ενσωματωμένες περιφερειακές συσκευές. Η καταλληλότητα ή μη ενός μικροελεγκτή για μία συγκεκριμένη εφαρμογή κρίνεται από τον αριθμό και το είδος των περιφερειακών που διαθέτει.

Στο προηγούμενο κεφάλαιο μιλήσαμε αναλυτικά για τις εντολές του PIC και την γλώσσα Assembly. Στο κεφάλαιο αυτό, θα μιλήσουμε αναλυτικότερα για τις περιφερειακές συσκευές του PIC και πώς μπορούμε να τις προγραμματίσουμε.

Το πλήθος των ενσωματωμένων περιφερειακών που διαθέτει, κάνουν τον μικροελεγκτή PIC κατάλληλο για ένα ευρύ φάσμα εφαρμογών. Θα εστιάσουμε την προσοχή μας στις ακόλουθες μονάδες του:

Θύρες εισόδων / εξόδων
Χρονιστές
Διαμορφωτής εύρους παλμού (PWM)
Θύρα σειριακής επικοινωνίας
Μετατροπέας αναλογικού σήματος σε ψηφιακό

Τα περιφερειακά του PIC, του δίνουν μεγάλη ευελιξία. Για παράδειγμα, για το περιφερειακό της σειριακής επικοινωνίας, μπορούμε να ορίσουμε την συχνότητα μετάδοσης, το αν η επικοινωνία θα είναι σύγχρονη ή ασύγχρονη, καθώς και πολλά άλλα.

Οι ορισμοί αυτοί, στην ουσία είναι παράμετροι που ρυθμίζουν την λειτουργία του περιφερειακού. Σε κάθε λειτουργία αντιστοιχεί και μία παράμετρος, η οποία είναι ένας δυαδικός αριθμός 8 bits. Συνήθως, ένα περιφερειακό χρειάζεται για την ρύθμιση του, μία ή δύο παραμέτρους, τις οποίες λαμβάνει μέσω κάποιων ειδικών καταχωρητών. Αυτοί οι καταχωρητές λέγονται καταχωρητές ελέγχου και είναι συγκεκριμένοι για κάθε περιφερειακή συσκευή. Εμείς, λοιπόν, αφού αποφασίσουμε το πώς πρέπει να δουλέψει το περιφερειακό, πρέπει να βρούμε τις τιμές αυτών των παραμέτρων και να τις φορτώσουμε στους καταχωρητές ελέγχου του. Αυτό ονομάζεται αρχικοποίηση της περιφερειακής μονάδας.

Πέρα των καταχωρητών ελέγχου του, ένα περιφερειακό διαθέτει επιπλέον καταχωρητές. Αυτούς τους χρησιμοποιεί για να αποθηκεύει τα δεδομένα που προκύπτουν από την λειτουργία του.

Στην συνέχεια, θα εξετάσουμε με την σειρά τις μονάδες αυτές ώστε να κατανοήσουμε τον τρόπο λειτουργίας τους και την χρησιμότητά τους. Παράλληλα, θα βλέπουμε και, ενδεικτικά, παραδείγματα αρχικοποίησής τους.

6.1 Θύρες εισόδων / εξόδων

Μπορούμε να πούμε ότι οι θύρες εισόδου / εξόδου, γενικού σκοπού, του PIC, είναι οι απλούστερες από τις περιφερειακές συσκευές του. Επιτρέπουν στον μικροελεγκτή μας να συνδέεται με άλλες συσκευές και να τις ελέγχει.

Ωστόσο, μερικοί από τους ακροδέκτες (pin) των θυρών αυτών έχουν διπλή λειτουργία. Συγκεκριμένα, υπάρχουν ακροδέκτες που χρησιμοποιούνται και από άλλες περιφερειακές μονάδες του μικροελεγκτή. Όπως γίνεται αντιληπτό, σε αυτήν την περίπτωση, δεν μπορούμε να τις χρησιμοποιήσουμε και σαν γενικής χρήσης εισόδους / εξόδους.

Ο PIC διαθέτει 5 θύρες εισόδου / εξόδου, που ονομάζονται θύρες A, B, C, D και E. Για την λειτουργία της κάθε θύρας απαιτείται ο αντίστοιχος καταχωρητής TRIS και ο αντίστοιχος καταχωρητής PORT. Υπάρχουν 5 καταχωρητές TRIS, οι TRISA, TRISB, TRISC, TRISD, και TRISE, που αντιστοιχούν σε κάθε μία από τις θύρες. Παρομοίως, υπάρχουν και 5 καταχωρητές PORT, οι PORTA, PORTB, PORTC, PORTD και PORTE. Οι καταχωρητές TRIS βρίσκονται στο τμήμα 1 της μνήμης δεδομένων, ενώ οι καταχωρητές PORT, βρίσκονται στο τμήμα 0. Όλες οι θύρες έχουν 8 ακροδέκτες εισόδου / εξόδου. Εξαίρεση αποτελούν η θύρα A που έχει μόνον 6 (RA0 - RA5) και η θύρα E που έχει μόνον 3 (RE0 - RE2).

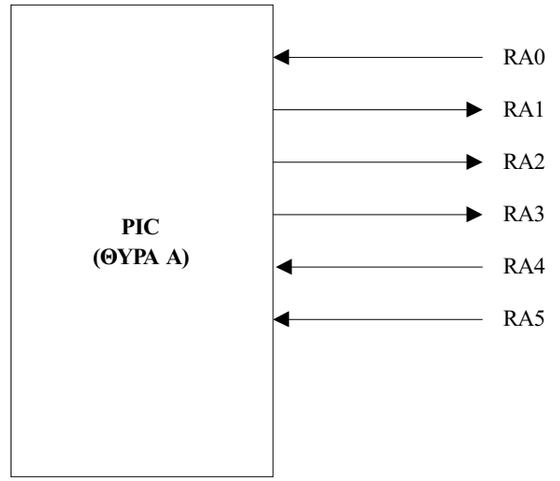
Το αν θα χρησιμοποιηθεί σαν είσοδος ή έξοδος ένας ακροδέκτης μίας θύρας, ελέγχεται από τον αντίστοιχο καταχωρητή TRIS, που βρίσκεται στο τμήμα 1 της μνήμης δεδομένων. Κάθε bit του καταχωρητή αντιστοιχεί σε έναν ακροδέκτη της θύρας. Αν το bit έχει τιμή "1", τότε, ο αντίστοιχος ακροδέκτης χρησιμοποιείται σαν είσοδος, ενώ αν έχει τιμή "0", τότε, χρησιμοποιείται σαν έξοδος. Για παράδειγμα, αν το ψηφίο 0 του καταχωρητή TRISA είναι 1, τότε ο ακροδέκτης RA0 της Θύρας A χρησιμοποιείται σαν είσοδος. Ο κάθε ακροδέκτης μίας θύρας μπορεί να ορισθεί ανεξάρτητα από τους άλλους. Έτσι, μπορούμε να έχουμε μία θύρα που να έχει ορισμένους κάποιους από τους ακροδέκτες της ως εισόδους και τους υπόλοιπους ως εξόδους.

Τα δεδομένα που θέλουμε να στείλουμε στην έξοδο μίας θύρας τα γράφουμε στον αντίστοιχο καταχωρητή PORT, π.χ. για την Θύρα C στον καταχωρητή PORTC Επίσης, διαβάζουμε δεδομένα εισόδου από τον ίδιο καταχωρητή. Στην συνέχεια, θα μελετήσουμε τις θύρες και τις βασικότερες λειτουργίες τους.

6.1.1 Θύρα A

Οι έξι ακροδέκτες, RA0 - RA5, της θύρας A μπορούν να χρησιμοποιηθούν σαν είσοδοι ή σαν έξοδοι. Ας δούμε, λοιπόν, ένα παράδειγμα προγραμματισμού της. Έστω, ότι

θέλουμε να χρησιμοποιήσουμε τους ακροδέκτες RA0, RA4 και RA5 σαν εξόδους, ενώ τους υπόλοιπους ακροδέκτες, RA1, RA2 και RA3, σαν εισόδους, όπως στο σχήμα 6.1.



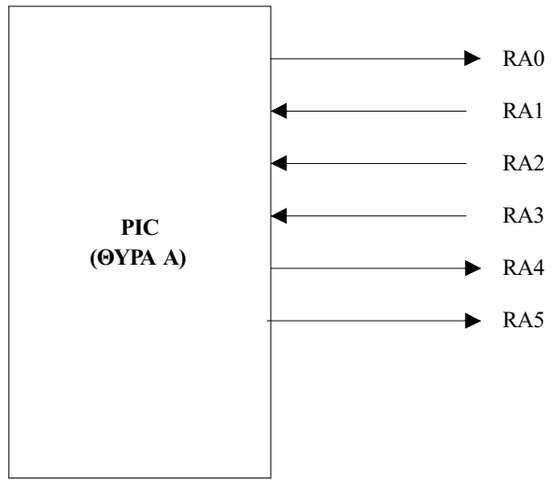
Σχήμα 6.1 *Η θύρα A με ορισμό ακροδεκτών RA0, RA4 και RA5 σαν εξόδους και RA1, RA2, RA3 σαν εισόδους.*

Τότε, όπως είπαμε προηγουμένως, για να χρησιμοποιήσουμε τους ακροδέκτες με τον τρόπο αυτό, πρέπει να δώσουμε την τιμή "1" στα bits του καταχωρητή TRISA που, αντιστοιχούν στους ακροδέκτες εισόδου και την "0" σε αυτούς που αντιστοιχούν στους ακροδέκτες εξόδου. Άρα, πρέπει να φορτώσουμε στον καταχωρητή TRISA την τιμή 11001110 (CE H). Ας τονίσουμε ότι, τα bits 6 και 7, που τους έχουμε δώσει τιμή "1", δεν έχουν σημασία, αφού η θύρα A δεν έχει 6ο και 7ο ακροδέκτη.

Εδώ, πρέπει να σημειώσουμε ότι, οι ακροδέκτες, της θύρας A, μπορούν να χρησιμοποιηθούν και σαν αναλογικές είσοδοι για άλλα περιφερειακά. Έτσι, πριν ορίσουμε το αν θα είναι είσοδοι ή έξοδοι, πρέπει να ορίσουμε ότι θα τους χρησιμοποιήσουμε για ψηφιακή λειτουργία. Αυτό γίνεται δίνοντας την τιμή 6 στον καταχωρητή ADCON1, για τον οποίο θα μιλήσουμε στην ενότητα 6.5, στο τέλος του κεφαλαίου.

7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	1	1	**0**

Σχήμα 6.2: *Ο καταχωρητής ADCON1 φορτωμένος με την τιμή 6, που ορίζει τους ακροδέκτες της θύρας A να έχουν ψηφιακή λειτουργία.*



Σχήμα 6.2 *Η θύρα Α με ορισμό ακροδεκτών RA0, RA4 και RA5 σαν εξόδους και RA1, RA2, RA3 σαν εισόδους.*

Ακολουθεί ο κώδικας αρχικοποίησης της θύρας, τον οποίο και εξηγούμε αμέσως μετά.

```
BCF STATUS, RP0   ; Επιλογή τμήματος 0 της μνήμης
BCF STATUS, RP1   ;δεδομένων (RP0 = 0 και RP1 = 0).

 CLRF PORTA       ;Μηδενισμός της εξόδου της θύρας

BSF STATUS, RP0   ;Επιλογή τμήματος 1 της μνήμης
                  ;δεδομένων (RP0 = 1).

MOVLW 6           ;Ορισμός των ακροδεκτών για ψηφιακή
MOVWF ADCON1      ;λειτουργία

MOVLW CE H        ; Φόρτωση τιμής 11001110 (CE H) στον TRISA
MOVWF TRISA       ;΄Ετσι επιλέγονται τα RA0, RA4 και RA5 σαν
                  ;έξοδοι και RA1 - RA3 σαν είσοδοι
```

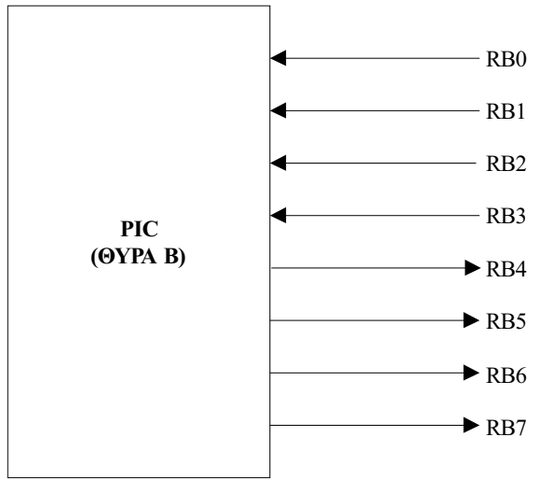
Ας δούμε, τώρα, αναλυτικά γιατί επιλέξαμε τις εντολές αυτές. Οι δύο πρώτες εντολές χρησιμοποιούνται για να μηδενίσουμε, αρχικά, την έξοδο της θύρας. Αυτό δεν είναι απόλυτα απαραίτητο να γίνει αλλά είναι καλή τεχνική προγραμματισμού να ξεκινήσουμε την έξοδο της θύρας από μία συγκεκριμένη τιμή. Ας προσέξουμε ότι ο καταχωρητής PORTA βρίσκεται στο τμήμα 0 της μνήμης δεδομένων. Συνεπώς, θα πρέπει, πρώτα, να επιλέξουμε αυτό το τμήμα της μνήμης, πριν προσπαθήσουμε να δώσουμε κάποια εντολή για τον καταχωρητή. Όπως είδαμε στο προηγούμενο κεφάλαιο του βιβλίου μας, η επιλογή τμήματος μνήμης δεδομένων, γίνεται με τον

μηδενισμό των bits RP0 και RP1 του καταχωρητή STATUS, που σημαίνει επιλογή του τμήματος 0 της μνήμης δεδομένων. Ακολούθως, θέτοντας το bit RP0 του καταχωρητή STATUS στο 1, επιλέγουμε το τμήμα 1 της μνήμης δεδομένων, αφού εκεί βρίσκεται τόσο ο καταχωρητής ADCON1 όσο και ο καταχωρητής TRISA. Στην συνέχεια, ορίζουμε ότι οι ακροδέκτες θα χρησιμοποιηθούν για ψηφιακά σήματα, φορτώνοντας στον καταχωρητή ADCON1 την τιμή 6 και, αμέσως μετά, ορίζουμε τους ακροδέκτες ως εισόδους ή εξόδους, φορτώνοντας στον καταχωρητή TRISA την τιμή CE H.

6.1.2 Θύρα Β

Η θύρα Β έχει 8 ακροδέκτες, που ο καθένας τους μπορεί να γίνει είσοδος ή έξοδος ανάλογα με το αν το αντίστοιχο bit του καταχωρητή TRISB είναι 1 ή 0. Για παράδειγμα, ο ορισμός των ακροδεκτών RB0 - RB3 ως είσοδοι και των υπολοίπων ως έξοδοι, όπως στο σχήμα 6.3, γίνεται με την φόρτωση της τιμής 00001111 (0F H) στον καταχωρητή TRISB. Ο κώδικας αρχικοποίησης έχει ως εξής:

```
BCF STATUS, RP0     ;Επιλογή τμήματος 0 της μνήμης
BCF STATUS, RP1     ;δεδομένων (RP0 = 0 και RP1 = 0).

 CLRF POTRB         ;Μηδενισμός της εξόδου της θύρας

BSF STATUS, RP0     ;Επιλογή τμήματος 1 της μνήμης
                    ;δεδομένων (RP0 = 1).
MOVLW 0F H          ;Φόρτωση τιμής 00001111 (0F H)  στον TRISB
MOVWF TRISB         ;΄Ετσι επιλέγονται τα RB0 - RB3 σαν είσοδοι
```



Σχήμα 6.3 *Η θύρα Β με ορισμό ακροδεκτών RB0 - RB3 ως είσοδοι και RB4 - RB7 ως έξοδοι.*

Επιπρόσθετα, οι ακροδέκτες RB4 - RB7 παρέχουν μία ακόμη λειτουργία. Δίνουν σήμα διακοπής μόλις η είσοδος κάποιας από αυτές αλλάξει τιμή. Αυτό, βέβαια, γίνεται με την προϋπόθεση ότι οι ακροδέκτες αυτοί λειτουργούν ως είσοδοι. Αν κάποιος ακροδέκτης έχει ορισθεί ως έξοδος, τότε, αυτός δεν μπορεί να προκαλέσει διακοπή. Για να μπορέσει μία αλλαγή στην είσοδο να γίνει αντιληπτή και να προκληθεί διακοπή, οι τιμές της συγκρίνονται με τις τελευταίες τιμές που διαβάστηκαν από τον καταχωρητή PORTB. Η διακοπή, που προκαλείται με τον τρόπο αυτό, ενεργοποιείται ή απενεργοποιείται με το bit RBIE ενώ η σημαία της είναι το bit RBIF. Αυτά τα δύο bits βρίσκονται στον καταχωρητή INTCON, όπως φαίνεται στο σχήμα 6.4.

7	6	5	4	3	2	1	0
			RBIE				**RBIF**

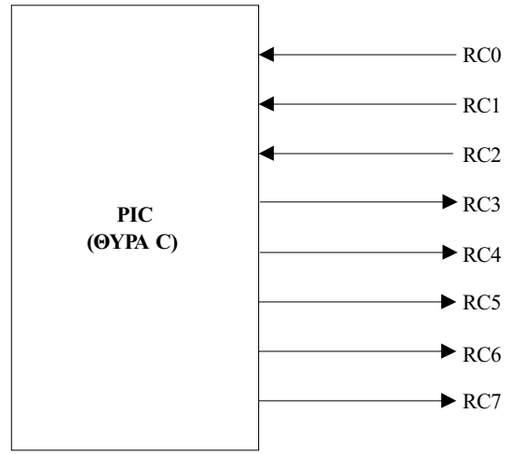
Σχήμα 6.4 *Η σημαία RBIF και το ψηφίο RBIE στον καταχωρητή INTCON.*

Κάθε φορά που δεχόμαστε μια διακοπή και εκτελούμε την ρουτίνα εξυπηρέτησής της, πρέπει στη συνέχεια να την απενεργοποιούμε ώστε η λειτουργία του προγράμματός μας να συνεχιστεί κανονικά. Για να μπορέσουμε να το κάνουμε αυτό, πρέπει να εκτελέσουμε μία εντολή που να διαβάζει ή να γράφει στον καταχωρητή PORTB. Αυτό θα προκαλέσει την ενημέρωσή του και έτσι θα σταματήσει να παρουσιάζεται η διαφορά κατά την διάρκεια της σύγκρισης, που αναφέραμε στην προηγούμενη παράγραφο. Ακολούθως, πρέπει να θέσουμε την σημαία RBIF στο 0. Αν δεν ενημερώσουμε τον καταχωρητή PORTB, η διαφορά θα εξακολουθεί να υπάρχει και η σημαία θα γίνεται συνεχώς 1, έστω και αν προσπαθούμε να την κάνουμε 0.

Τον ακροδέκτη RB0, της θύρας B μπορούμε να τον χρησιμοποιήσουμε για την λήψη μίας ακόμη διακοπής από τις εξωτερικές συσκευές με τις οποίες επικοινωνεί ο PIC. Εδώ, η διακοπή δεν προκαλείται από την αλλαγή της κατάστασης της εισόδου στον ακροδέκτη αυτόν αλλά κατά την διάρκεια του ανερχόμενου ή κατερχόμενου μετώπου. Συγκεκριμένα, όταν το bit INTEDG του καταχωρητή OPTION_REG είναι 1 τότε η διακοπή προκαλείται σε ανερχόμενο μέτωπο του ακροδέκτη RB0, ενώ όταν είναι 0 η διακοπή προκαλείται σε κατερχόμενο. Επίσης, την θύρα B αφορά και το bit 7 του καταχωρητή OPTION_REG, το οποίο και θα θέτουμε πάντα στο 1. Τα bits INTEDG και 7, του καταχωρητή OPTION_REG, φαίνονται στο σχήμα 6.5.

7	6	5	4	3	2	1	0
1	**INTEDG**						

Σχήμα 6.5 *Το ψηφίο INTEDG στον καταχωρητή OPTION_REG.*



Σχήμα 6.5 *Η θύρα C με τους ακροδέκτες RC0 - RC2 ορισμένους ως εισόδους και τους Rc3 - RC7 ως εξόδους.*

Ένα παράδειγμα εφαρμογής αυτής της λειτουργίας διακοπής, του ακροδέκτη RB0 της θύρας B, είναι όταν συνδέουμε ένα πληκτρολόγιο στον ακροδέκτη αυτό. Η εφαρμογή αυτή συνιστάται όταν θέλουμε να επαναφέρουμε τον PIC από κατάσταση SLEEP, με το πάτημα ενός πλήκτρου. Η περίπτωση είναι ανάλογη με ότι συμβαίνει στους προσωπικούς Η/Υ μας. Εκεί, μετά από ένα διάστημα που ο Η/Υ μένει ανενεργός, σβήνει η οθόνη. Η λειτουργία της επανέρχεται με το πάτημα ενός πλήκτρου ή την κίνηση του ποντικιού.

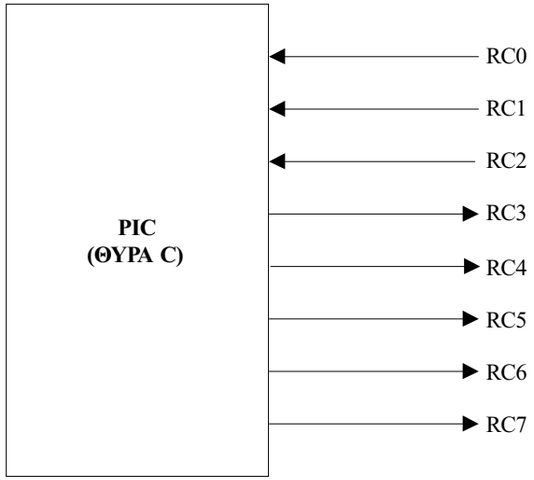
6.1.3 Θύρα C

Η θύρα C έχει 8 ακροδέκτες εισόδου - εξόδου. Όπως και στις προηγούμενες θύρες, έτσι και εδώ, ο κάθε ακροδέκτης μπορεί να ορισθεί ανεξάρτητα από τους υπολοίπους. Για παράδειγμα, ο ορισμός των ακροδεκτών RC0 - RC2 ως είσοδοι και των υπολοίπων ως έξοδοι, όπως στο σχήμα 6.6, γίνεται με την φόρτωση της τιμής 00000111 (03 H) στον καταχωρητή TRISC.

Ο κώδικας αρχικοποίησης είναι:

```
BCF STATUS, RP0      ; Επιλογή τμήματος 0 της μνήμης
BCF STATUS, RP1      ;δεδομένων (RP0 = 0 και RP1 = 0).

CLRF POTRC           ;Μηδενισμός της εξόδου της θύρας

BSF STATUS, RP0      ;Επιλογή τμήματος 1 της μνήμης
                     ;δεδομένων.
MOVLW 03 H           ;Φόρτωση τιμής 00000111 (03 H) στον TRISC
```

```
MOVWF TRISC          ;Έτσι επιλέγονται τα RC0 - RC2 σαν
                     ;είσοδοι και τα RC3 - RC7 σαν έξοδοι
```



Σχήμα 6.6 *Η θύρα C με τους ακροδέκτες RC0 - RC2 ορισμένους ως εισόδους και τους RC3 - RC7 ως εξόδους.*

Όταν προγραμματίζουμε την θύρα για μία συγκεκριμένη λειτουργία, όπως στο παραπάνω παράδειγμα, πρέπει να προσέχουμε και την λειτουργία των άλλων περιφερειακών του PIC. Κάποια από αυτά μπορεί να χρησιμοποιούν τους ακροδέκτες της θύρας. Στην περίπτωση αυτή, είναι δυνατόν, να αλλάζουν τα bits του καταχωρητή TRISC και να μετατρέπουν κάποιους ακροδέκτες σε εισόδους ή εξόδους ανάλογα με την λειτουργία τους.

6.1.4 Θύρες D και E

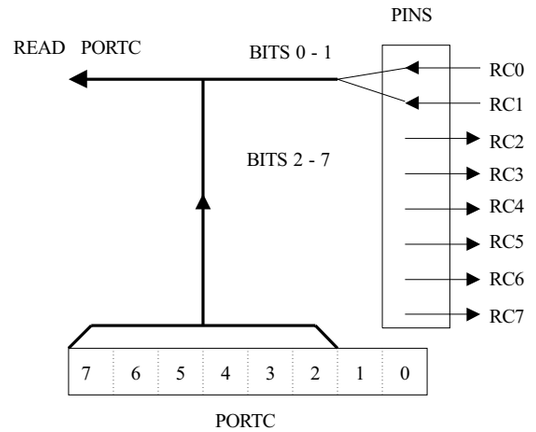
Η θύρα D έχει 8 ακροδέκτες εισόδου / εξόδου ενώ η θύρα E έχει μόνον 3. Η λειτουργία τους είναι πανομοιότυπη με την γενική λειτουργία των θυρών A, B και C και, για τον λόγο αυτό, δεν θα την αναλύσουμε.

Τέλος, αφού μιλήσαμε για όλες τις θύρες του PIC, θα πρέπει να τονίσουμε ένα χαρακτηριστικό της λειτουργίας των καταχωρητών PORT. Ενώ η εγγραφή ενός δεδομένου σε έναν από τους καταχωρητές PORT σημαίνει φόρτωμα του δεδομένου στον καταχωρητή αυτό, δηλαδή, ό,τι έχουμε μάθει έως τώρα, η ανάγνωση ενός καταχωρητή PORT σημαίνει ανάγνωση της κατάστασης των ακροδεκτών της αντίστοιχης θύρας.

Πρέπει να ξέρουμε ότι οι εντολές εγγραφής, όπως είναι οι εντολές επεξεργασίας bit, πρώτα διαβάζουν τα δεδομένα, μετά τα τροποποιούν και, τέλος, τα αποθηκεύουν ξανά στο ίδιο μέρος. Για παράδειγμα η εντολή "BCF 30 H, 3", πρώτα διαβάζει το περιεχόμενο του καταχωρητή 30 H, μετά μηδενίζει το bit 3 της τιμής που διάβασε και,

τέλος, γράφει το αποτέλεσμα, πίσω, στον καταχωρητή 30 H. Ωστόσο, αν εκτελέσουμε μία εντολή εγγραφής στον καταχωρητή PORT μίας θύρας, που έχει κάποιους από τους ακροδέκτες της ορισμένους ως εισόδους, τότε, τα πράγματα είναι διαφορετικά. Η εντολή εγγραφής θα διαβάσει την κατάσταση των ακροδεκτών, αντί του καταχωρητή PORT και, στη συνέχεια, αφού τις τροποποιήσει θα τις γράψει στον καταχωρητή PORT. Μόνον οι ακροδέκτες που είναι έξοδοι έχουν τις τιμές των αντιστοίχων bits του καταχωρητή PORT, ενώ όσοι είναι είσοδοι παίρνουν την τιμή τους από το σύστημα που τους οδηγεί.

Ας δούμε, λοιπόν, ένα παράδειγμα. Έστω ότι η θύρα C έχει τους ακροδέκτες RC0 και RC1 ορισμένους ως εισόδους και τους RC2 - RC5 ορισμένους ως εξόδους. Επίσης, ας υποθέσουμε ότι ο καταχωρητής PORTC έχει το περιεχόμενο 00101110 (2E H) ενώ η κατάσταση των ακροδεκτών εισόδου είναι RC0 =1 και RC1 = 0. Τότε, η εντολή "BCF PORTC, 3", διαβάζει την τιμή 00101101 (2D H) και, ακολούθως, μηδενίζει το bit 3. Το αποτέλεσμα, 00100101 (25 H), γράφει στον καταχωρητή PORTC. Αυτό, βεβαίως, είναι διαφορετικό από το αποτέλεσμα 00100110 (26 H), που θα παίρναμε, με την ίδια εντολή, αν αντί του PORTC είχαμε, για παράδειγμα, τον καταχωρητή 30 H. Το σχήμα 6.7 δείχνει παραστατικά το παράδειγμα αυτό.



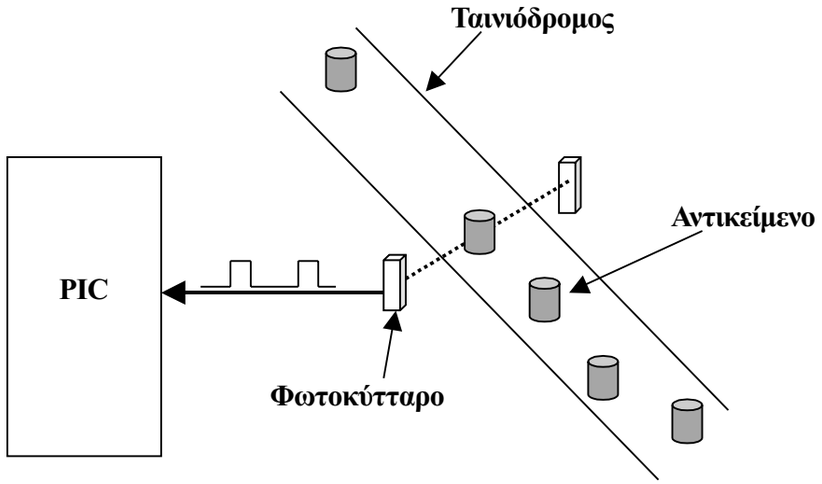
Σχήμα 6.7 *Εκτέλεση εντολής ανάγνωσης στον καταχωρητή PORTC της θύρας C, με τους ακροδέκτες RC0 και RC1 ορισμένους ως εισόδους, ενώ τους RC2 έως RC7, ως εξόδους.*

Ερωτήσεις

1. Με ποιόν καταχωρητή μπορούμε να ορίσουμε σαν είσοδο ή έξοδο τον ακροδέκτη μίας θύρας;
2. Πώς πρέπει να τροποποιήσουμε το πρόγραμμα αρχικοποίησης της θύρας C, έτσι ώστε να ορίσουμε τους ακροδέκτες RC3 και RC6 ως εξόδους και τους υπόλοιπους ως εισόδους;

6.2 Χρονιστές

Η δεύτερη κατηγορία περιφερειακών συσκευών, που θα μελετήσουμε, είναι οι χρονιστές. Πρόκειται για συσκευές που αυξάνουν ή μειώνουν την τιμή μίας παραμέτρου κατά μία μονάδα. Οι χρονιστές έχουν δύο τρόπους λειτουργίας. Στον πρώτο, η μεταβολή της τιμής αυτής γίνεται περιοδικά, με συχνότητα που καθορίζεται από ένα ρολόι. Στο δεύτερο τρόπο, λειτουργούν ως μετρητές κάποιων παλμών που δέχονται από το εξωτερικό περιβάλλον. Ένα παράδειγμα είναι η σύνδεση ενός κυκλώματος με φωτοκύτταρο στον κατάλληλο ακροδέκτη του PIC. Το φωτοκύτταρο χρησιμοποιείται για την μέτρηση αντικειμένων που περνούν με έναν ταινιόδρομο. Κάθε φορά που ένα αντικείμενο περνά, το φωτοκύτταρο στέλνει σήμα στο χρονιστή του PIC, ο οποίος και καταγράφει την μέτρηση. Το παράδειγμα παρουσιάζεται στο σχήμα 6.8.



Σχήμα 6.8 *Ο μικροελεγκτής PIC σε λειτουργία μέτρησης αντικειμένων.*

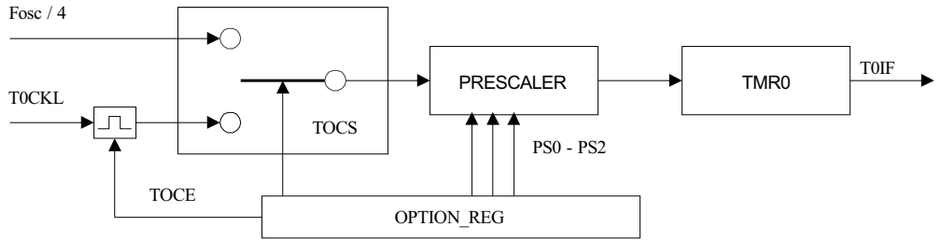
Ο PIC διαθέτει 3 διαφορετικούς τύπους χρονιστών, τους TIMER0, TIMER1 και TIMER2. Εμείς, θα δούμε, αναλυτικά, τη λειτουργία των TIMER0 και TIMER2. Η λειτουργία του TIMER1 είναι παρόμοια.

6.2.1 Ο Χρονιστής TIMER0

Η συγκεκριμένη συσκευή διαθέτει ένα διαιρέτη συχνότητας 8 bits και έναν μετρητή 8 bits. Η τιμή των μετρήσεων του TIMER0 αποθηκεύεται στον καταχωρητή TMR0, που βρίσκεται στο τμήμα 2 της μνήμης δεδομένων. Ο έλεγχος του χρονιστή TIMER0 γίνεται αποκλειστικά από τα bits 0 - 5 του καταχωρητή OPTION_REG, που βρίσκεται στο τμήμα 1 της μνήμης δεδομένων. Ο καταχωρητής OPTION_REG φαίνεται στο σχήμα 6.9, ενώ το διάγραμμα του χρονιστή παρουσιάζεται στο σχήμα 6.10.

7	6	5	4	3	2	1	0
		TOCS	**TOCE**	**0**	**PS2**	**PS1**	**PS0**

Σχήμα 6.9 *Ο καταχωρητής OPTION_REG*



Σχήμα 6.10 *Ο χρονιστής TIMER0.*

Τον καταχωρητή αυτό μπορούμε όχι μόνο να τον διαβάσουμε αλλά και να τον γράψουμε. Ο καταχωρητής, μεταξύ άλλων, περιέχει και bits για τον έλεγχο του TMR0 διαιρέτη. Ας δούμε, τώρα, την σημασία του κάθε bit του καταχωρητή ελέγχου OPTION_REG για την λειτουργία του χρονιστή.

- ***Bits 7-6:*** Δεν έχουν σημασία για την λειτουργία του χρονιστή.

- ***TOCS:*** Επιλογή πηγής ρολογιού
  1 = Πηγή συνδεδεμένη στον ακροδέκτη TOCLK (Λειτουργία μετρητή παλμών)
  0 = Εσωτερική πηγή ρολογιού (Λειτουργία χρονιστή).

- ***TOSE:*** Επιλογή μετώπου πηγής
  1 = Αύξηση με κατερχόμενο μέτωπο του παλμού στον ακροδέκτη TOCLK
  0 = Αύξηση με ανερχόμενο μέτωπο του παλμού στον ακροδέκτη TOCLK

- ***Bit 3:*** Θα το θεωρούμε πάντα "0".

- ***PS2-PS0:*** Επιλογή λόγου διαίρεσης

Τιμή Bits	Ρυθμός TMR0
000	1:2
001	1:4
010	1:8
011	1:16
100	1:32
101	1:64
110	1:128
111	1:256

Για να επιλέξουμε τη λειτουργία του περιφερειακού ως χρονιστή θέτουμε το bit TOCS = 0. Τότε ο χρονιστής θα αρχίσει να αυξάνεται. Μπορούμε, επίσης, να του ορίσουμε την τιμή, από την οποία θέλουμε να ξεκινήσει να αυξάνει, στον καταχωρητή TMR0. Ωστόσο, πρέπει να προσέξουμε διότι οι δύο πρώτοι κύκλοι δεν καταγράφονται. Για να υπερκεράσουμε αυτό το πρόβλημα, πρέπει να βάλουμε, στον καταχωρητή TMR0, τιμή κατά δύο μονάδες μεγαλύτερη από αυτή που θέλουμε να ξεκινά.

Για να λειτουργήσουμε τη συσκευή ως μετρητή, πρέπει να θέσουμε το bit TOCS = 1. Τότε, η συσκευή μετρά τους παλμούς που δέχεται στον ακροδέκτη T0CLK. Η αύξηση γίνεται όταν ο παλμός βρίσκεται είτε στο ανερχόμενο είτε στο κατερχόμενο μέτωπό του. Αυτό το ορίζουμε από το bit TOSE, όπως είδαμε παραπάνω.

Ο λόγος διαίρεσης ορίζεται από τα bits 0 ως 2. Όταν ο διαιρέτης χρησιμοποιείται από τον χρονιστή TIMER0, όλες οι εντολές που γράφουν στον καταχωρητή TMR0 μηδενίζουν τον διαιρέτη. Επίσης, ο καταχωρητής αυξάνεται ανά τόσους παλμούς, όσους έχουμε ορίσει στον διαιρέτη. Για παράδειγμα, αν βάλουμε στα bits PS2 - PS0 την τιμή 000 τότε ο διαιρέτης έχει την τιμή 2 και η αύξηση θα γίνεται κάθε 2 κύκλους εντολής. Παράλληλα, όμως, αυξάνεται ανάλογα και ο χρόνος που καθυστερεί μέχρι να αρχίσει να μετρά. Έτσι, στο παράδειγμά μας, αντί των δύο κύκλων εντολών καθυστέρηση που θα είχαμε, τώρα, θα έχουμε την διπλή, δηλαδή 4.

Στην περίπτωση που ο καταχωρητής TMR0 υπερχειλίσει, δηλαδή μετά από την τιμή FF H φορτώσει την τιμή 0, τότε προκαλεί διακοπή. Η σημαία που υποδηλώνει την διακοπή είναι το bit T0IF του καταχωρητή INTCON. Η σημαία τίθεται πάλι στο 0 από την ρουτίνα εξυπηρέτησης της διακοπής, ώστε η διακοπή να μπορεί να ενεργοποιηθεί ξανά. Η διακοπή μπορεί να απενεργοποιηθεί, μόνιμα, αν θέσουμε το bit T0IE, του καταχωρητή INTCON, στο 0. Οι δύο σημαίες, που αναφέραμε, φαίνονται στο σχήμα 6.11.

7	6	5	4	3	2	1	0
		TOIE			**TOIF**		

Σχήμα 6.11 *Οι σημαίες T0IE και T0IF του καταχωρητή INTCON.*

Στην συνέχεια θα δούμε πώς μπορούμε να αρχικοποιήσουμε το περιφερειακό αυτό, ως χρονιστή και ως μετρητή.

- **Αρχικοποίηση ως χρονιστή (εσωτερική πηγή παλμών).**

Έστω ότι θέλουμε να ορίζουμε τον TIMER0 να αυξάνεται χρησιμοποιώντας το εσωτερικό ρολόι (TOCS = 0), στο ανερχόμενο μέτωπό του (TOSE = 0), με διαιρέτη 1:16 (PS2 - PS0 = 011). Τότε, πρέπει να φορτώσουμε στον καταχωρητή OPTION_REG (σχήμα 6.9) την τιμή 11000011 (C3 H). Ας σημειωθεί ότι τα bits 6 και 7 είναι αδιάφορα για την λειτουργία αυτή και τα βάζουμε στο "1". Ο κώδικας αρχικοποίησης, έχει ως ακολούθως:


```
CLRF TMR0          ;Μηδένισε TMR0
CLRF INTCON        :Απενεργοποίησε τις διακοπές και μηδένισε το TOIF

BCF STATUS, RP0    ;Επιλογή του τμήματος 1 της μνήμης δεδομένων
BSF STATUS, RP1

MOVLW 83 H         ;Με βάση τη τιμή 83 H ορίζουμε λειτουργία χρονιστή
MOVWF OPTION_REG   ;με το εσωτερικό ρολόι, στο ανερχόμενο μέτωπο,
                   ;με διαιρέτη 1:16

BCF STATUS, RP0    ;Επιλογή του τμήματος 0 της μνήμης δεδομένων

BSF INTCON, TOIE   ;Ενεργοποίηση της διακοπής TMR0
BSF INTCON, GIE    ;Ενεργοποίηση όλων των διακοπών
```

- **Αρχικοποίηση ως μετρητή (εξωτερική πηγή παλμών).**

Έστω ότι θέλουμε να ορίζουμε τον TIMER0 να μετρά τους εξωτερικούς παλμούς που δέχεται στο T0CLK (TOCS = 1), στο κατερχόμενο μέτωπό τους (TOSE = 1), με διαιρέτη 1:256 (PS2 - PS0 = 111). Τότε, πρέπει να φορτώσουμε στον καταχωρητή OPTION_REG την τιμή 00110111 (37 H). Ας σημειωθεί ότι τα bits 6 και 7 είναι αδιάφορα για την λειτουργία αυτή και τα βάζουμε στο "0". Ο πλήρης κώδικας αρχικοποίησης, έχει ως ακολούθως:

```
CLRF TMR0          ;Μηδένισε TMR0
CLRF INTCON        :Απενεργοποίησε τις διακοπές και μηδένισε το TOIF

BCF STATUS, RP0    ;Επιλογή του τμήματος 1 της μνήμης δεδομένων
BSF STATUS, RP1

MOVLW 37 H         ;Αύξηση του χρονιστή από το εξωτερικό παλμό που
MOVWF OPTION_REG   ;δέχεται στο T0CLK, με διαιρέτη 1:256

BCF STATUS, RP0    ;Επιλογή του τμήματος 0 της μνήμης δεδομένων

BSF INTCON, TOIE   ;Ενεργοποίηση της διακοπής TMR0
BSF INTCON, GIE    ;Ενεργοποίηση όλων των διακοπών
```

Και στις δύο περιπτώσεις, όπως μπορούμε να παρατηρήσουμε, ο κώδικας αρχικοποίησης ξεκινά με την απενεργοποίηση των διακοπών. Ακολούθως, προχωρά στην τοποθέτηση της κατάλληλης, για την κάθε περίπτωση, τιμής στον καταχωρητή OPTION_REG. Τέλος, οι διακοπές ενεργοποιούνται ξανά.

6.2.2 Ο Χρονιστής TIMER 2

Όπως και ο TIMER0 έτσι και αυτή η περιφερειακή μονάδα χρονισμού / μέτρησης, ο TIMER2, είναι 8 bits. Ο TIMER2, όμως, διαθέτει δύο διαιρέτες, αντί του ενός που έχει ο TIMER0. Ο ένας διαιρέτης, που λέγεται prescaler, διαιρεί τους παλμούς εισόδου πριν από τον μετρητή, όπως και στον TIMER0. Ο άλλος διαιρέτης, που λέγεται postscaler, διαιρεί το αποτέλεσμα της μέτρησης. Ο χρονιστής μπορεί να χρησιμοποιηθεί σαν βάση χρόνου για την περιφερειακή συσκευή διαμόρφωσης πλάτους παλμού (PWM).

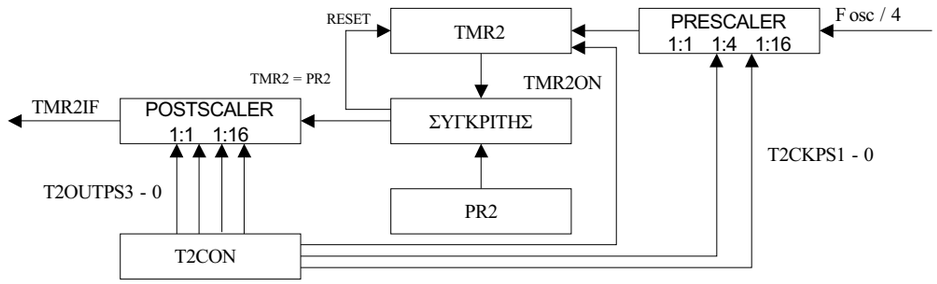
Οι μετρήσεις καταγράφονται στον καταχωρητή TMR2, ενώ ο καταχωρητής ελέγχου του είναι ο T2CON, ο οποίος φαίνεται στο σχήμα 6.12. Και οι δύο καταχωρητές βρίσκονται στο τμήμα 0 της μνήμης δεδομένων.

7	6	5	4	3	2	1	0
-	**TOUTPS3**	**TOUTPS2**	**TOUTPS1**	**TOUTPS0**	**TMR2ON**	**T2CKPS1**	**T2CKPS0**

Σχήμα 6.12 *Ο καταχωρητής T2CON*

Τον καταχωρητή T2CON μπορούμε όχι μόνον να τον διαβάσουμε αλλά και να τον γράψουμε. Ο καταχωρητής αυτός περιέχει bits για τον καθορισμό του τρόπου λειτουργίας του TIMER2 και του διαιρέτη του. Το διάγραμμα του TIMER2, με τους δύο διαιρέτες και τους καταχωρητές που χρησιμοποιεί, φαίνεται στο σχήμα 6.13.

Στην συνέχεια παρουσιάζουμε την σημασία του κάθε bit του καταχωρητή ελέγχου για την λειτουργία του TIMER2.



Σχήμα 6.13 *Ο χρονιστής TIMER2.*

- ***Bit7:*** Δεν έχει σημασία. Διαβάζεται σαν "0".

- ***TOUTPS3 - TOUTPS0:*** Επιλογή του λόγου διαίρεσης του αποτελέσματος του μετρητή (postscale).
  0000 = 1:1
  0001 = 1:2
  0010 = 1:3
  ...
  ...

  ...
  1111 = 1:16

- ***TMR2ON:*** Bit ενεργοποίησης TIMER2.
  1 = TIMER2 ενεργοποίηση
  0 = TIMER2 απενεργοποίηση

- ***T2CKPS1 - T2CKPS0:*** Επιλογή του λόγου διαίρεσης πριν από τον μετρητή (prescale). 00 = Διαίρεση με 1
  01 = Διαίρεση με 4
  10 = Διαίρεση με 16
  11 = Διαίρεση με 16

Ο χρονιστής δέχεται παλμούς ρολογιού μόνον από το ρολόι του PIC, με συχνότητα Fosc/4. Μπορούμε να διαιρέσουμε την συχνότητα αυτή, κατά 1, 4 και 16, χρησιμοποιώντας τον διαιρέτη πριν από τον μετρητή. Όπως είδαμε και προηγουμένως, η επιλογή γίνεται από τα bits T2CKPS1 - T2CKPS0 του καταχωρητή T2CON.

Πέραν των καταχωρητών που προαναφέραμε, η συσκευή αυτή διαθέτει και έναν άλλο καταχωρητή, τον PR2, που βρίσκεται στο τμήμα 1 της μνήμης δεδομένων. Και αυτός ο καταχωρητής μπορεί να γραφεί και να διαβαστεί. Ο καταχωρητής TMR2 συγκρίνεται με τον PR2 και αν είναι ίσοι, αρχίζει να μετρά, πάλι, από την αρχή, δηλαδή από το 0. Έτσι, βάζοντας μία τιμή στον PR2, μπορούμε να ρυθμίσουμε την συχνότητα με την οποία ο TMR2 μηδενίζεται.

Όπως και στις άλλες δύο περιπτώσεις χρονιστών, όταν υπερχειλίσει ο καταχωρητής που καταγράφει την μέτρηση, προκαλείται διακοπή. Στον TIMER2 η διακοπή υποδεικνύεται από την σημαία TMR2IF του καταχωρητή PIR1 (σχήμα 6.14). Τη διακοπή μπορούμε να απενεργοποιήσουμε από το bit TMR2IE του καταχωρητή PIE1 (σχήμα 6.15). Πρέπει, και εδώ, να προσέξουμε η ρουτίνα εξυπηρέτησης, που θα φτιάξουμε, να καθαρίζει την σημαία της διακοπής, έτσι ώστε να μπορεί αυτή να χρησιμοποιηθεί ξανά.

7	6	5	4	3	2	1	0
						TMR2IF	

Σχήμα 6.14 *Η σημαία TMR2IF στον καταχωρητή PIR1.*

7	6	5	4	3	2	1	0
						TMR2IE	

Σχήμα 6.15 *Το bit TMR2IE στον καταχωρητή PIE1.*

Στην συνέχεια θα δούμε ένα παράδειγμα αρχικοποίησης του TIMER2 και ορισμού των παραμέτρων του:

Παράδειγμα αρχικοποίησης TIMER2

Έστω ότι θέλουμε να χρησιμοποιήσουμε τον TIMER2 με λόγο διαίρεσης 1:16 των παλμών εισόδου του μετρητή (T2CKPS1 - T2CKPS0 =11) και λόγο διαίρεσης 1:15 του αποτελέσματος του μετρητή (TOUTPS3 - TOUTPS0 = 1110). Τις τιμές αυτές θα τις δώσουμε με τον TIMER2 σταματημένο (TMR2ON = 0). Έτσι με βάση τα προηγούμενα, πρέπει να φορτώσουμε τον καταχωρητή T2CON (σχήμα 6.12) με την τιμή 01110011 (72 H). Αμέσως μετά θα εκκινήσουμε τον μετρητή θέτοντας το bit TMR2ON στο "1".

Η διακοπή, που υποδηλώνεται με την σημαία TMR2IF, προκαλείται όταν η τιμή στον TMR2 γίνει ίση με την τιμή στον PR2. Η σημαία τίθεται πάλι στο 0 από την ρουτίνα εξυπηρέτησης της διακοπής, ώστε η διακοπή να μπορεί να ενεργοποιηθεί ξανά.

Στο παράδειγμά μας, δεν διακόπτεται η εκτέλεση του προγράμματος, για να εκτελεστεί η ρουτίνα εξυπηρέτησης του TIMER2, κάθε φορά που η τιμή του TMR2 γίνει ίση με αυτή του PR2 (διακοπή). Για την διαχείριση της διακοπής χρησιμοποιούμε την μέθοδο δειγματοληψίας, όπως την είδαμε στην ενότητα 5.7, του κεφαλαίου 5. Έτσι, αφού απενεργοποιήσουμε όλες τις διακοπές, ελέγχουμε ανά τακτά χρονικά διαστήματα, με τη χρήση ενός βρόχου, αν το bit TMR2IF του καταχωρητή PIR1 είναι 1. Επίσης, χρησιμοποιούμε την τεχνική της αλλαγής της ροής προγράμματος υπό συνθήκη, που περιγράψαμε στην ενότητα 5.6, του κεφαλαίου 5, για να εκτελέσουμε τη ρουτίνα εξυπηρέτησης του περιφερειακού, όταν συμβεί η διακοπή. Ο κώδικας είναι ο ακόλουθος:

```
;Παύση TIMER2, Λόγος διαίρεσης 1:1 και στους δύο διαιρέτες
        CLRF T2CON
;
;Μηδενισμός καταχωρητή TMR2
        CLRF TMR2
;
;Απενεργοποίηση διακοπών
        CLRF INTCON
;
;Επιλογή τμήματος 1 της μνήμης δεδομένων
        BSF STATUS, RP0
;
;Απενεργοποίηση των διακοπών των περιφερειακών
        CLRF PIE1                    ;
;Επιλογή τμήματος 0 της μνήμης δεδομένων
        BCF STATUS, RP0
;
; Μηδενισμός σημαιών περιφερειακών διακοπών
        CLRF PIR1
```



```
;
;Διαιρέτης Prescaler = 1:16, T2CKPS1 - T2CKPS0 =11

;Διαιρέτης Postscaler = 1:15, TOUTPS3 - TOUTPS0 = 1110
;Παύση TIMER2, TMR2ON = 0
        MOVLW 72 H
        MOVWF T2CON
;
;Εκκίνηση TIMER2
        BSF T2CON, TMR2ON
;
;
;Η διακοπή του Timer2 είναι απενεργοποιημένη. Ο έλεγχος γίνεται
;με δειγματοληψία της σημαίας υπερχείλισης TMR2IF και την χρησιμοποίηση της
;τεχνικής της αλλαγής της ροής προγράμματος υπό συνθήκη.
;
T2_OVFL_WAIT

;΄Ελεγχος διακοπής (bit TMR2IF).
;Αν δεν υπάρχει διακοπή (TMR2IF = 0) εκτέλεσε την επομένη εντολή.
;Αν υπάρχει διακοπή (TMR2IF = 1) εκτέλεσε την μεθεπομένη εντολή.
        BTFSS PIR1, TMR2IF
GOTO T2_OVFL_WAIT    ; Επανέλαβε το βρόχο.
;
;Ο χρονιστής έχει υπερχειλίσει. Μηδένισε τη σημαία TMR2IF. BCF PIR1, TMR2IF
```

Ερωτήσεις

1. Ποιοι είναι οι δύο τρόποι λειτουργίας ενός χρονιστή;
2. Σε ποιόν καταχωρητή μπορούμε να βρούμε την τιμή της μέτρησης του TIMER0;
3. Τι προκαλεί την μεταβολή στην τιμή της μέτρησης;
4. Ποια η λειτουργία του διαιρέτη;

6.3 Μονάδα Σύλληψης / Σύγκρισης / Διαμόρφωσης εύρους παλμού (PWM)

Στην παρούσα ενότητα θα εξετάσουμε μία περιφερειακή συσκευή του PIC, η οποία μπορεί να λειτουργήσει με τρεις διαφορετικούς τρόπους, ως ακολούθως:

- Μονάδα Σύλληψης
- Μονάδα Σύγκρισης
- Μονάδα Διαμόρφωσης εύρους παλμού (PWM)

Αυτή η περιφερειακή συσκευή λέγεται CCP. Ο μικροελεγκτής PIC διαθέτει δύο τέτοιες περιφερειακές συσκευές, οι οποίες λειτουργούν πανομοιότυπα, τις CCP1 και CCP2. Εμείς θα εξετάσουμε την πρώτη.

Η μονάδα CCP1 περιέχει δύο καταχωρητές, οι οποίοι βρίσκονται στο τμήμα 0 της μνήμης δεδομένων. Ο ένας καταχωρητής, ο οποίος ονομάζεται CCP1CON, είναι 8 bits, και χρησιμοποιείται για τον έλεγχο της συσκευής. Ο άλλος, ο CCPR1, είναι 16 bits και τον χρησιμοποιεί κατά την λειτουργία της. Ο καταχωρητής αυτός μπορεί να λειτουργήσει σαν καταχωρητής σύλληψης 16 bits, σύγκρισης 16 bits και PWM 10 bits, ανάλογα με τον τρόπο λειτουργίας της. Επειδή είναι 16 bits, αναφερόμαστε σε αυτόν σαν να ήταν δύο καταχωρητές, οι CCPR1H και CCPR1L, που είναι το περισσότερο και λιγότερο σημαντικό byte του, αντίστοιχα.

Σε κάθε λειτουργία της, η μονάδα CCP1, χρησιμοποιεί και έναν από τους χρονιστές του PIC, που είδαμε στην προηγούμενη ενότητα. Στις δύο πρώτες λειτουργίες χρησιμοποιεί τον TIMER1, ενώ στην τελευταία, την PWM, τον TIMER2. Επίσης, χρησιμοποιεί τον ακροδέκτη CCP2 σαν είσοδο ή έξοδο, ανάλογα με την λειτουργία της.

Θα εξετάσουμε, τώρα, τον καταχωρητή ελέγχου και θα δούμε την σημασία των bits του. Ο καταχωρητής φαίνεται στο σχήμα 6.16. Τον καταχωρητή μπορούμε όχι μόνον να τον διαβάσουμε αλλά και να τον γράψουμε. Ο καταχωρητής περιέχει bits για την επιλογή του τρόπου λειτουργίας της συσκευής.

7	6	5	4	3	2	1	0
-	-	**DCIB1**	**DCIB0**	**CCPIM3**	**CCPIM2**	**CCPIM1**	**CCPIM0**

Σχήμα 6.16 *Ο καταχωρητής CCP1CON*

Η σημασία των bits του έχει ως εξής:

- ***Bits 7 και 6:*** Δεν έχουν σημασία και διαβάζονται ως 0.

- ***DC1B1- DC1B0:*** Bits PWM λειτουργίας
  Είναι το δύο λιγότερο σημαντικά bits, δηλαδή το 1 και το 0, από τα 10 bits της λειτουργίας PWM. Τα υπόλοιπα 8 βρίσκονται στον καταχωρητή CCPR1L. Για τις άλλες δύο λειτουργίες της συσκευής δεν έχουν σημασία.

- ***CCP1M3 - CCP1M0:*** Bits επιλογής λειτουργίας
  0000 = Απενεργοποίηση όλων των λειτουργιών. Επανεκκίνηση (reset) συσκευής. 0100 - 0111 = Λειτουργίες σύλληψης.
  1000 - 1011 = Λειτουργίες σύγκρισης.

11xx = PWM λειτουργία. Τα bits CCP1M0 και CCP1M1 δεν έχουν σημασία και μπορεί να είναι οτιδήποτε.

Στην συνέχεια θα περιγράψουμε τις τρεις λειτουργίες αυτής της περιφερειακής συσκευής.

6.3.1 Λειτουργία Σύλληψης

Στην λειτουργία αυτή, όταν η μονάδα CCP δεχθεί σήμα στον ακροδέκτη CCP1, διαβάζει την τιμή του καταχωρητή TMR1, που, όπως είπαμε, ανήκει στον TIMER1, και την αποθηκεύει στον καταχωρητή CCPR1. Ως σήμα λογίζεται ένα από τα ακόλουθα:

- Μία κατερχόμενο μέτωπο παλμού.
- Μία ανερχόμενο μέτωπο παλμού.
- Μία ανερχόμενο μέτωπο κάθε 4ου παλμού.
- Μία ανερχόμενο μέτωπο κάθε 16ου παλμού.

Η λειτουργία ονομάζεται σύλληψη.

6.3.2 Μονάδα Σύγκρισης

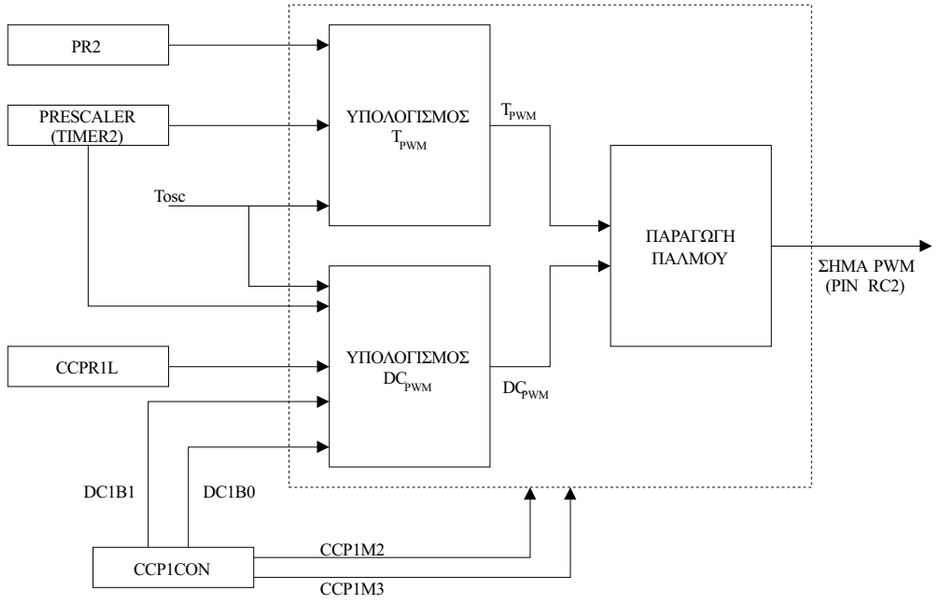
Στην λειτουργία αυτή, η μονάδα CCP1 συγκρίνει τα περιεχόμενα του καταχωρητή CCPR1 με τα περιεχόμενα του καταχωρητή TMR1. Όταν διαπιστώσει ισότητα θέτει τον ακροδέκτη CCP1 από το 0 στο 1 ή από το 1 στο 0, ανάλογα με τον ορισμό που της δώσαμε στα bits CCP1M3 - CCP1M0 του καταχωρητή CCP1CON.

6.3.3 Μονάδα Διαμόρφωσης εύρους παλμού (PWM)

Αυτή η λειτουργία έχει πολύ μεγάλη σημασία σε περιπτώσεις όπου ο PIC χρησιμοποιείται για την ανάπτυξη μικροϋπολογιστικών συστημάτων για βιομηχανικό περιβάλλον. Η λειτουργία πολλών συσκευών ελέγχεται από την στάθμη της ενεργού τάσης του σήματος εισόδου. Για παράδειγμα υπάρχουν ηλεκτρικές βαλβίδες με μεταβαλλόμενο άνοιγμα ή κλείσιμο που είναι ελεγχόμενες από την ενεργό τάση εισόδου. Η στάθμη της ενεργού τάσης μίας παλμοσειράς εξαρτάται από το εύρος των παλμών σε σχέση με την περίοδο, δηλαδή το duty cycle. Ως duty cycle μίας παλμοσειράς ορίζουμε τον λόγο της χρονικής διάρκειας του παλμού της προς την περίοδο της. Συχνά, το duty cycle μετράται %. Για παράδειγμα duty cycle 25%, το οποίο σημαίνει ότι η διάρκεια του παλμού μίας παλμοσειράς είναι το 25% όλης της περιόδου της. Ο παλμός δίνεται στον ακροδέκτη RC2, της θύρας C.

Η λειτουργία επιλέγεται από τα bits CCP1M3 - CCP1M0 του καταχωρητή CCP1CON,

όπως άλλωστε και οι άλλες δύο. Ο παλμός παράγεται στον ακροδέκτη CCP1, τον οποίο πρέπει να ορίσουμε ως έξοδο στον καταχωρητή TRIS. Επίσης, ο χρονιστής που χρησιμοποιείται είναι ο TIMER2, αντί του TIMER1 που χρησιμοποιούν οι άλλες λειτουργίες. Το σχήμα 6.17 παρουσιάζει την μονάδα PWM και τους καταχωρητές που χρησιμοποιεί.



Σχήμα 6.17 *Η μονάδα PWM και οι καταχωρητές που χρησιμοποιεί.*

Μπορούμε να ορίσουμε την περίοδο της PWM, την οποία συμβολίζουμε με TPWM, δίνοντας μία τιμή στον καταχωρητή PR2. Ας θυμηθούμε ότι, ο PR2 είναι ο καταχωρητής που καθορίζει την τιμή όπου θα μηδενίζεται ο καταχωρητής TMR2 του TIMER2. Τότε, την περίοδο την υπολογίζουμε, σε μονάδες χρόνου, ως εξής:

$$T_{PWM} = [(PR2) + 1] \cdot 4 \cdot T_{OSC} \cdot (\text{prescale διαιρέτης TIMER2}) \qquad (6.1)$$

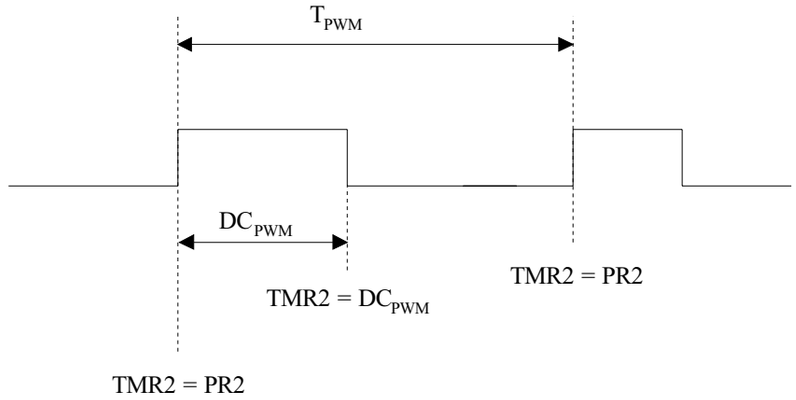
Όπως είναι αναμενόμενο, η συχνότητα PWM είναι το αντίστροφο της περιόδου, δηλαδή:

$$F_{PWM} = 1 / T_{PWM} \qquad (6.2)$$

Το PWM duty cycle, το οποίο συμβολίζουμε με DC_{PWM}, μπορούμε να το ορίσουμε από τα 2 bits, DC1B1- DC1B0, του καταχωρητή ελέγχου CCP1CON, που είναι και τα λιγότερο σημαντικά, και τα 8 bits του καταχωρητή CCPR1L. Όλα μαζί σχηματίζουν μία λέξη των 10 bits, που θα την ονομάσουμε DC1B. Το duty cycle υπολογίζεται, ως εξής:

$$DC_{PWM} = DC1B \cdot T_{OSC} \cdot (\text{prescale διαιρέτης TIMER2}) \qquad (6.3)$$

Το σχήμα 6.18 δείχνει παραστατικά την έξοδο PWM, στον ακροδέκτη RC2, και τη σημασία της περιόδου T_{PWM} και των τιμών DC_{PWM}, TMR2 και PR2. Στην συνέχεια θα δούμε μερικά παραδείγματα, τα οποία θα μας βοηθήσουν να καταλάβουμε πώς μπορούμε να υπολογίσουμε τις διάφορες παραμέτρους της λειτουργίας PWM.



Σχήμα 6.18 *Η έξοδος PWM και η σημασία της περιόδου T_{PWM} και των τιμών DC_{PWM}, TMR2 και PR2.*

Παράδειγμα 1 - Εύρεση τιμής PR2

Έστω ότι η ζητούμενη συχνότητα PWM, η F_{PWM}, είναι η 78,125 kHz και ότι ο PIC λειτουργεί με F_{OSC} = 20 MHz.

Αν ορίσουμε τον prescale διαιρέτη του TIMER2 στο 1, θέτοντας στον καταχωρητή T2CON τα ψηφία T2CKPS1 - T2CKPS0 = 00, τότε, από τον τύπο 6.1, μπορούμε να υπολογίσουμε την τιμή που πρέπει να δώσουμε στον καταχωρητή PR2, ως εξής:

T_{PWM} = 1/78,125 kHz = [(PR2) +1] · 4 · (1 / 20 MHz) · 1
12,8 μs = [(PR2) +1] · 4 · 0,05 μs · 1
PR2 = 63

Πριν δούμε ένα δεύτερο παράδειγμα, θα ανακεφαλαιώσουμε τα όσα είπαμε για την PWM λειτουργία της περιφερειακής συσκευής CCP1 του μικροελεγκτή μας. Ας δούμε, λοιπόν, επιγραμματικά τα βήματα που πρέπει να ακολουθήσουμε για να θέσουμε σε λειτουργία το περιφερειακό αυτό:

1. Καθορισμός της περιόδου της PWM από τον καταχωρητή PR2.
2. Καθορισμός του duty cycle της PWM από την λέξη DC1B (bits DC1B9 - DC1B0).
3. Ορισμός του ακροδέκτη CCP1 με μηδενισμό του bit PWM1 του καταχωρητή TRISC.

4. Καθορισμός του λόγου της prescale διαίρεσης από τον καταχωρητή T2CON του TIMER2.
5. Ορισμός της λειτουργίας PWM της συσκευής CCP1 από τον καταχωρητή CCP1CON.

΄Εχοντας υπόψη όλα αυτά, μπορούμε να δούμε το ακόλουθο παράδειγμα:

Παράδειγμα 2 - Αρχικοποίηση μονάδας PWM για 20% duty cycle

΄Εστω ότι θέλουμε να χρησιμοποιήσουμε το μικροελεγκτή PIC, με ρολόι 20 MHz, για να παράγουμε μία παλμοσειρά με duty cycle 20% και συχνότητα F_{PWM} = 156,3 kHz, δηλαδή T_{PWM} = 6,4μs.

Άρα, η διάρκεια του παλμού πρέπει να είναι 1,28 μs. Θα χρησιμοποιήσουμε τον χρονιστή TIMER2, με τον prescale διαιρέτη του ορισμένο στο 1, δηλαδή τα bits T2CKPS1 και T2CKPS2 του καταχωρητή ελέγχου, T2CON, του χρονιστή πρέπει να είναι στο 0.

Με αυτά τα δεδομένα, από τον τύπο 6.1, βρίσκουμε ότι πρέπει να δώσουμε στον καταχωρητή PR2 τιμή 31, δηλαδή 1F H. Στην συνέχεια, από τον τύπο 6.3, βρίσκουμε ότι, κατά προσέγγιση, η τιμή της λέξης DC1B πρέπει να είναι 26, δηλαδή 1A H. Όπως είπαμε η λέξη DC1B έχει μήκος 10 bits και, συνεπώς, είναι η:

CCPR1L								CCP1CON	
7	6	5	4	3	2	1	0	DC1B1	DC1B0
0	0	0	0	0	1	1	0	1	0

Πέρα από τα bits DC1B1 και DC1B0, που είδαμε παραπάνω, στον καταχωρητή ελέγχου CCP1CON, πρέπει να ορίσουμε και την λειτουργία PWM με τα bits CCP1M3 - 0 = 1100. Άρα, πρέπει να του φορτώσουμε την τιμή 00101100 (2C H) Ο κώδικας αρχικοποίησης της περιφερειακής συσκευής CCP1, για την παραπάνω λειτουργία PWM, έχει ως ακολούθως:

```
    CLRF CCP1CON        ; Απενεργοποίηση CCP
    CLRF TMR2           ; Μηδενισμός TIMER2

    MOVLW 1F H          ;
    MOVWF PR2           ; PR2 = 31 (1F H)

                        ;Duty Cycle το 20% της TPWM
    MOVLW 6
    MOVWF CCPR1L
;Απενεργοποίηση διακοπών - Μηδενισμός T0IF
    CLRF INTCON
;Επιλογή τμήματος 1 της μνήμης δεδομένων
    BSF STATUS, RP0
```



```
;Ορισμός ακροδέκτη PWM1 (RC2 της θύρας C) ως έξοδο
    BCF TRISC, PWM1
;Απενεργοποίηση περιφερειακών διακοπών.
    CLRF PIE1
;Επιλογή τμήματος 0 της μνήμης δεδομένων
    BCF STATUS, RP0

;Μηδενισμός των σημαιών των περιφερειακών διακοπών
   CLRF PIR1

;Επιλογή της λειτουργίας PWM, DC1B1 = 1 και DC1B0 = 0.
    MOVLW 2C H
    MOVWF CCP1CON

    BSF T2CON, TMR2ON          ;Εκκίνηση TIMER2

;Απενεργοποιημένη διακοπή CCP1.
;΄Ελεγχος διακοπής (bit TMR2IF).
;Αν δεν υπάρχει διακοπή (TMR2IF = 0) εκτέλεσε την επομένη εντολή.
;Αν υπάρχει διακοπή (TMR2IF = 1) εκτέλεσε την μεθεπομένη εντολή.

PWM_Period_Match
        BTFSS PIR1, TMR2IF
GOTO PWM_Period_Match  ;Επανέλαβε το βρόχο.

;Μηδενισμός σημαίας διακοπής TMR2IF
   BCF PIR1, TMR2IF
```

Ερωτήσεις

1. Τι πετυχαίνουμε με τη χρήση της μονάδας PWM; Που μπορεί να χρησιμοποιηθεί;
2. Ποια από τις περιφερειακές συσκευές, που είδαμε στις προηγούμενες ενότητες, χρησιμοποιεί η παρούσα συσκευή για την λειτουργία της;
3. Ποιος είναι ο καταχωρητής ελέγχου της παρούσας συσκευής και από ποια bits του μπορούμε να ορίσουμε τη συσκευή να λειτουργεί ως μονάδα PWM.

6.4 Θύρα σειριακής επικοινωνίας

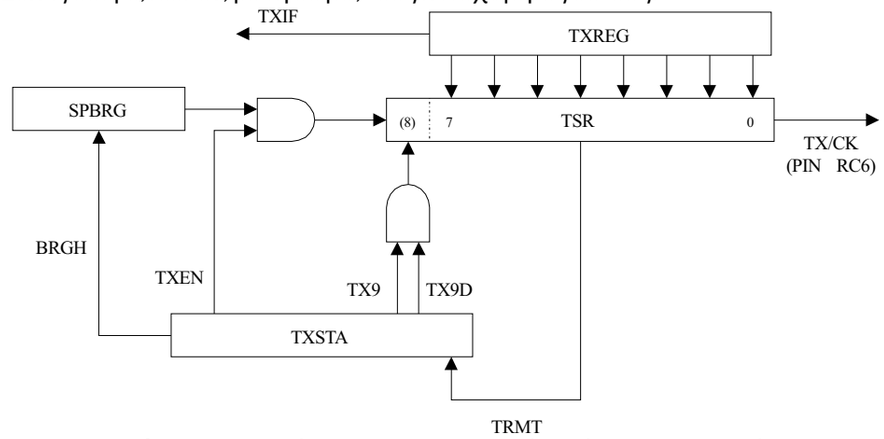
Στην παρούσα ενότητα θα μελετήσουμε την περιφερειακή συσκευή σειριακής επικοινωνίας του μικροελεγκτή μας. Αυτή η συσκευή ονομάζεται USART και

υποστηρίζει τόσο σύγχρονη όσο και ασύγχρονη σειριακή επικοινωνία. Συγκεκριμένα, το περιφερειακό αυτό μπορεί να χρησιμοποιηθεί για ταυτόχρονα αμφίδρομη ασύγχρονη επικοινωνία (full duplex). Αυτό σημαίνει ότι τα δεδομένα μπορούν να μεταδίδονται την ίδια χρονική στιγμή και προς τις δύο κατευθύνσεις. Με τον τρόπο αυτό, μπορεί να συνδεθεί με τερματικά και προσωπικούς Η/Υ. Επίσης, μπορεί να λειτουργήσει και με διαδοχικά αμφίδρομη σύγχρονη επικοινωνία (half duplex). Στην περίπτωση αυτή τα δεδομένα μπορούν να μεταδίδονται και προς τις δύο κατευθύνσεις αλλά όχι ταυτόχρονα. Δηλαδή, πρώτα γίνεται η μετάδοση προς τη μία κατεύθυνση, και όταν τελειώσει αυτή, μπορεί να ξεκινήσει προς την άλλη. Με αυτόν τον τρόπο, μπορεί να επικοινωνήσει με άλλες εξωτερικές περιφερειακές συσκευές, όπως μετατροπείς αναλογικού σήματος σε ψηφιακό ή ψηφιακού σε αναλογικό, σειριακές μνήμες, κτλ.
Επιγραμματικά, λοιπόν, η συσκευή USART, μπορεί να λειτουργήσει με 3 τρόπους:

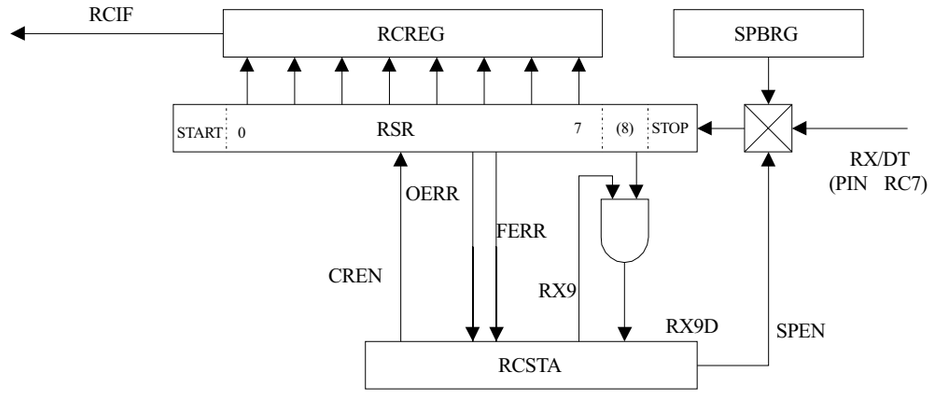
- Ασύγχρονο (full duplex)
- Σύγχρονο - Master (half duplex)
- Σύγχρονο - Slave (half duplex)

Εμείς θα εξετάσουμε διεξοδικά μόνον τον ασύγχρονο τρόπο λειτουργίας, ενώ θα αναφερθούμε επιγραμματικά στο σύγχρονο. Οι ακροδέκτες RC6 και RC7 της θύρας C του PIC, χρησιμοποιούνται ως οι ακροδέκτες TX/CK και RX/DT της σειριακής θύρας. Αυτοί ορίζονται από το bit SPEN και τα bits 6 και 7 του καταχωρητή TRISC, τα οποία πρέπει να τεθούν στο 1. Έτσι, οι απαιτούμενοι, για τη λειτουργία του περιφερειακού, ακροδέκτες TX/CK και RX/DT μπορούν να λειτουργήσουν σωστά.

Σε αντίθεση με τις υπόλοιπες περιφερειακές συσκευές, που εξετάσαμε, η παρούσα διαθέτει δύο καταχωρητές ελέγχου, τους TXSTA και RCSTA. Ο πρώτος χρησιμοποιείται για τη λειτουργία του USART ως πομπού και ο δεύτερος για την λειτουργία του ως δέκτη. Η μονάδα USART, σε λειτουργία ασύγχρονης εκπομπής, παρουσιάζεται στο σχήμα 6.19, ενώ σε λειτουργία ασύγχρονης λήψης, στο σχήμα 6.20. Ας δούμε, λοιπόν, με τη σειρά, τους καταχωρητές αυτούς.



Σχήμα 6.19 *Η μονάδα USART σε λειτουργία ασύγχρονης εκπομπής.*



Σχήμα 6.20 *Η μονάδα USART σε λειτουργία ασύγχρονης λήψης.*

Ο καταχωρητής TXSTA φαίνεται στο σχήμα 6.21 και χρησιμοποιείται, εκτός από τον ορισμό της λειτουργίας εκπομπής, και για την ένδειξη της κατάστασης του καταχωρητή εκπομπής. Εδώ, θα αναφερθούμε μόνον στις επιλογές που αφορούν την ασύγχρονη επικοινωνία.

7	6	5	4	3	2	1	0
0	**TX9**	**TXEN**	**0**	**0**	**BRGH**	**TRMT**	**TX9D**

Σχήμα 6.21 *Ο καταχωρητής TXSTA στην ασύγχρονη επικοινωνία.*

Η σημασία των bits είναι η ακόλουθη:

- ***Bit 7:*** Δεν έχει σημασία στην ασύγχρονη επικοινωνία. Το θέτουμε στο "0".

- ***TX9:*** Επιλογή bits επικοινωνίας
  1 = Επιλογή επικοινωνίας 9 bits
  0 = Επιλογή επικοινωνίας 8 bits

- ***TXEN:*** Bit ενεργοποίησης λειτουργίας εκπομπής
  1 = Ενεργοποίηση λειτουργίας εκπομπής
  0 = Απενεργοποίηση λειτουργίας εκπομπής

- ***Bit 4:*** Στην την ασύγχρονη επικοινωνία, το θέτουμε στο "0".

- ***Bit 3:*** Δεν χρησιμοποιείται. Διαβάζεται σαν '0'.

- ***BRGH:*** Bit επιλογής υψηλού Baud Rate
  Ασύγχρονη λειτουργία
  1 = Υψηλή ταχύτητα
  0 = Χαμηλή ταχύτητα

- ***TRMT:*** Bit ένδειξης κατάστασης καταχωρητή TSR
  1 = TSR άδειος
  0 = TSR γεμάτος

- ***TX9D:*** 9ο Εκπεμπόμενο bit. Μπορεί να είναι το bit ισοτιμίας.

Μπορούμε να δώσουμε τιμές σε όλα τα bits του καταχωρητή, εκτός, φυσικά, του 3ου, που δεν χρησιμοποιείται, και του 1ου, που είναι μόνον αναγνώσιμο. Όμως, ας τονίσουμε ότι, για να έχουμε ασύγχρονη επικοινωνία πρέπει να θέτουμε στο "0" τα bits 4 και 7, όπως είπαμε προηγουμένως.

Ο δεύτερος καταχωρητής ελέγχου, ο RCSTA φαίνεται στο σχήμα 6.22 και χρησιμοποιείται, εκτός από τον ορισμό της λειτουργίας λήψης, και για την ένδειξη της κατάστασης του καταχωρητή λήψης. Και σε αυτόν τον καταχωρητή, θα αναφερθούμε μόνον στις επιλογές που αφορούν την ασύγχρονη επικοινωνία.

7	6	5	4	3	2	1	0
SPEN	**RX9**	**0**	**CREN**	**0**	**FERR**	**OERR**	**RX9D**

Σχήμα 6.22 *Ο καταχωρητής RCSTA στην ασύγχρονη επικοινωνία.*

Ας δούμε, και εδώ, πώς μπορούν να ορισθούν τα bits του καταχωρητή:

- ***SPEN:*** Bit ενεργοποίησης σειριακής θύρας.
  1 = Ενεργοποίηση σειριακής θύρας (Ορίζει τα RX/DT και TX/CK σαν ακροδέκτες της σειριακής θύρας)
  0 = Απενεργοποίηση σειριακής θύρας

- ***RX9:*** Επιλογή bits επικοινωνίας
  1 = Επιλογή επικοινωνίας 9 bits
  0 = Επιλογή επικοινωνίας 8 bits

- ***Bit 5:*** Δεν χρησιμοποιείται στην ασύγχρονη επικοινωνία. Το θέτουμε στο "0".

- ***CREN:*** Bit ενεργοποίησης συνεχούς λήψης
  Ασύγχρονη λειτουργία
  1 = Ενεργοποίηση συνεχούς λήψης
  0 = Απενεργοποίηση συνεχούς λήψης

- ***Bit 3:*** Δεν χρησιμοποιείται. Διαβάζεται σαν '0'.

  FERR: Bit ένδειξης σφάλματος πλαισίου
  1 = Σφάλματος πλαισίου (Μπορεί να ενημερωθεί με ανάγνωση του καταχωρητή RCREG και να λάβει το επόμενο σωστό byte)
  0 = Κανένα σφάλμα πλαισίου

- ***OERR:*** Bit ένδειξης σφάλματος επικάλυψης
  1 = Σφάλμα επικάλυψης (Μπορεί να μηδενιστεί, μηδενίζοντας το bit CREN.)
  0 = Κανένα σφάλμα επικάλυψης

- ***RX9D:*** 9ο λαμβανόμενο bit. Μπορεί να είναι το bit ισοτιμίας.

Μπορούμε να δώσουμε τιμές στα τέσσερα περισσότερο σημαντικά bits του καταχωρητή, δηλαδή τα bits 4 ως 7. Τα bits 0 ως 2 είναι μόνο αναγνώσιμα, ενώ το 3ο bit δεν χρησιμοποιείται.

Όπως εύκολα καταλαβαίνουμε, από την πολυπλοκότητα των καταχωρητών ελέγχου, το περιφερειακό μπορεί να λειτουργήσει με πολλούς τρόπους. Γενικά, όμως, όπως είπαμε και στην αρχή, υπάρχουν δύο τρόποι επικοινωνίας, ο σύγχρονος και ο ασύγχρονος. Στο σύγχρονο, τα δεδομένα στέλνονται με σταθερό ρυθμό, σύμφωνο με αυτόν του ρολογιού της σειριακής επικοινωνίας. Μαζί με τα δεδομένα στέλνεται και το ρολόι. Έτσι, η αρχή και το τέλος τους, είναι γνωστά στον δέκτη, αφού αυτός συγχρονίζεται με τον πομπό, με τη βοήθεια του ρολογιού αυτού. Άρα, μπορούν να σταλούν πολλά δεδομένα μαζί, χωρίς να υπάρχει η ανάγκη, το κάθε ένα από αυτά, να γνωστοποιεί στον δέκτη, την αρχή και το τέλος του. Σε αντίθεση, στην ασύγχρονη επικοινωνία ο δέκτης δεν συγχρονίζεται με τον πομπό και, συνεπώς, δεν γνωρίζει τον ακριβή χρόνο έλευσης ενός δεδομένου. Εμείς, εδώ, θα περιοριστούμε στην μελέτη του ασύγχρονου τρόπου επικοινωνίας.

Για να επιλυθεί το προαναφερόμενο πρόβλημα της ασύγχρονης επικοινωνίας του περιφερειακού, η κάθε λέξη που στέλνεται περικλείεται από 2 ή 3 επιπλέον bits. Το πρώτο καθορίζει την αρχή της λέξης που αποστέλλεται. Ακολουθούν τα bits της λέξης και η αποστολή τελειώνει με τα υπόλοιπα 1 ή 2 που σηματοδοτούν το τέλος του. Το σχήμα 6.23 δείχνει παραστατικά τα παραπάνω.

Λογικό 1

Αρχή	0	1	2	3	4	5	6	7	Τέλος

Λογικό 0

Σχήμα 6.23 *Τα bits που στέλνονται στην ασύγχρονη επικοινωνία.*

Το πλεονέκτημα της ασύγχρονης επικοινωνίας είναι ότι δεν απαιτείται να στέλνονται συνεχώς δεδομένα. Ένα παράδειγμα αυτής της επικοινωνίας είναι η σύνδεση του πληκτρολογίου με τον υπολογιστή, όπου ο ρυθμός πληκτρολόγησης δεν είναι σταθερός.

Ο ρυθμός μετάδοσης λέγεται Baud rate και ταυτίζεται, στην περίπτωσή μας, με τον ρυθμό μετάδοσης των bits. Μπορούμε να υπολογίσουμε τον ρυθμό αυτόν ως εξής:

Για χαμηλή ταχύτητα (BRGH = 0):

$$\text{Baud Rate} = F_{osc} / (64\,(X+1)) \qquad (6.4)$$

Για υψηλή ταχύτητα (BRGH = 1):

$$\text{Baud Rate} = F_{osc} / (16\,(X+1)) \qquad (6.5)$$

Την τιμή X, που θα υπολογίσουμε, για το Baud Rate που επιθυμούμε, θα την τοποθετήσουμε στον καταχωρητή SPBRG.

Η επιλογή της ασύγχρονης λειτουργίας γίνεται από το bit SYNC του καταχωρητή ελέγχου TXSTA, όταν αυτό τεθεί στο 0. Η μονάδα ασύγχρονης επικοινωνίας δομείται από 4 βασικά στοιχεία:

- Γεννήτρια Baud Rate
- Κύκλωμα Δειγματοληψίας.
- Ασύγχρονος εκπομπός
- Ασύγχρονος δέκτης

Λειτουργία ασύγχρονης εκπομπής

Ας δούμε, τώρα, τον τρόπο της λειτουργίας εκπομπής. Χρησιμοποιούνται δυο καταχωρητές οι TSR και TXREG. Ο βασικός καταχωρητής εκπομπής είναι ο TSR, ο οποίος λαμβάνει τα δεδομένα του από τον καταχωρητή TXREG. Στον τελευταίο γράφουμε το byte που θέλουμε να αποστείλουμε, το οποίο μεταφέρεται στον TSR και από εκεί εκπέμπεται από τον ακροδέκτη TX/CK. Εδώ, ας σημειώσουμε ότι ο καταχωρητής TSR δε βρίσκεται στην μνήμη δεδομένων και δεν μπορούμε να τον χρησιμοποιήσουμε στα προγράμματά μας. Η μεταφορά του byte, από τον έναν καταχωρητή στον άλλο, γίνεται όταν ο TSR είναι άδειος, δηλαδή όταν έχει στείλει και το bit τέλους εκπομπής. Όταν η μεταφορά του byte, από τον TXREG στον TSR, ολοκληρωθεί, ο TXREG είναι άδειος. Αυτό προκαλεί την ενεργοποίηση μίας διακοπής, που υποδηλώνεται με την σημαία TXIF, του καταχωρητή PIR1 (σχήμα 6.24). Η διακοπή αυτή μπορεί να ενεργοποιείται ή απενεργοποιείται από το bit TXIE, του καταχωρητή PIE1 (σχήμα 6.25). Ωστόσο, η σημαία TXIF μπορεί να γίνει 1 ανεξάρτητα από το αν η διακοπή είναι ενεργοποιημένη ή μη, ενώ δεν υπάρχει τρόπος να την μηδενίσουμε με κάποια εντολή. Η σημαία μηδενίζεται μόνον όταν φορτωθούν νέα δεδομένα στον καταχωρητή TXREG.

7	6	5	4	3	2	1	0
			TXIF				

Σχήμα 6.24 *Η σημαία TXIF στον καταχωρητή PIR1.*

7	6	5	4	3	2	1	0
			TXIE				

Σχήμα 6.25 *Η σημαία TXIE στον καταχωρητή PIE1.*

Αντίστοιχο με το bit TXIF, που δείχνει την κατάσταση του καταχωρητή TXREG, είναι το bit TRMT, του καταχωρητή TXSTA, που δείχνει την κατάσταση του καταχωρητή TSR. Το bit τίθεται στο 1, όταν ο καταχωρητής αυτός είναι άδειος. Αντίθετα, όμως, με τον καταχωρητή TXREG, εδώ, καμία διακοπή δεν προκαλείται. Για να καταλάβουμε αν ο TSR είναι άδειος, πρέπει να ελέγχουμε το bit αυτό.

Για να ξεκινήσει η μετάδοση πρέπει να θέσουμε το bit TXEN, του καταχωρητή ελέγχου TXSTA, στο 1 και να τοποθετήσουμε στον TXREG αυτό που θέλουμε να στείλουμε.

Ας δούμε, τώρα, επιγραμματικά τα βήματα που πρέπει να ακολουθήσουμε για να ορίσουμε μία ασύγχρονη σειριακή εκπομπή:

1. Αρχικοποίηση του καταχωρητή SPBRG για το κατάλληλο baud rate. Αν είναι επιθυμητή υψηλή ταχύτητα τότε το bit BRGH = 1.
2. Ενεργοποίηση ασύγχρονης σειριακής θύρας με τα bits SYNC = 0 και SPEN = 1.
3. Αν οι διακοπές είναι επιθυμητές, τότε, τα bits TXIE = 1, GIE = 1 και PEIE = 1.
4. Αν η μετάδοση 9 bits είναι επιθυμητή, τότε, το bit TX9 = 1.
5. Εκκίνηση εκπομπής με το bit TXEN = 1, το οποίο θα θέσει το bit TXIF στο 1.
6. Αν επιλεγεί μετάδοση 9 bits το 9° bit πρέπει να φορτωθεί στο bit TX9D.
7. Το φόρτωμα των δεδομένων στον καταχωρητή TXREG σηματοδοτεί την έναρξη της μετάδοσης.

Λειτουργία ασύγχρονης λήψης

Η λειτουργία λήψης ακολουθεί παρόμοια λογική με αυτή της λειτουργίας εκπομπής, μόνον που εδώ η διαδικασία γίνεται αντίστροφα.

Τα δεδομένα λαμβάνονται στον ακροδέκτη RX/DT. Η λειτουργία λήψης ορίζεται από το bit CREN, του καταχωρητή ελέγχου. Όπως και στη περίπτωση της εκπομπής, έτσι και εδώ, χρησιμοποιούνται δύο καταχωρητές, οι RSR και RCREG. Κατά τη διαδικασία λήψης ο RSR λαμβάνει τα bits, το έναν μετά το άλλο, από τον ακροδέκτη RX/DT. Μόλις λάβει και το τελευταίο bit, δηλαδή το bit 'τέλος', μεταφέρει το δεδομένο στον καταχωρητή RCREG. Μόλις γίνει η μεταφορά προκαλείται διακοπή και η σημαία RCIF τίθεται στο 1, του καταχωρητή PIR1 (σχήμα 6.26). Η διακοπή μπορεί να ενεργοποιηθεί ή απενεργοποιηθεί από το bit RCIE, του καταχωρητή PIE1 (σχήμα 6.27). Όταν διαβάσουμε τον καταχωρητή RCREG, αυτός αδειάζει και η σημαία μηδενίζεται.

7	6	5	4	3	2	1	0
		RCIF					

Σχήμα 6.26 *Η σημαία RCIF στον καταχωρητή PIR1*

7	6	5	4	3	2	1	0
		RCIE					

Σχήμα 6.27 *Η σημαία RCIE στον καταχωρητή PIE1*

Κατά τη διαδικασία λήψης, δύο σφάλματα μπορούν να συμβούν. Το πρώτο είναι το σφάλμα επικάλυψης, που υποδηλώνεται από το bit OERR, του καταχωρητή ελέγχου RCSTA. Το σφάλμα επικάλυψης συμβαίνει όταν, κατά τη διάρκεια της λήψης, ο καταχωρητής ανιχνεύσει το bit τέλους. Τότε, θα επιχειρήσει να μεταφέρει το δεδομένο στον RCREG. Αν ο καταχωρητής δεν έχει ακόμη διαβασθεί και, συνεπώς, δεν είναι άδειος, θα προκληθεί σφάλμα επικάλυψης. Ο λόγος είναι ότι ο RCREG θα φορτώσει το νέο δεδομένο, με αποτέλεσμα το προηγούμενο να χαθεί. Αν συμβεί αυτό, η μεταφορά των δεδομένων από τον καταχωρητή RSR στον καταχωρητή RCREG

σταματά και πρέπει να μηδενίσουμε το bit OERR, για να συνεχιστεί η διαδικασία λήψης. Αυτό μπορούμε να το κάνουμε αν μηδενίσουμε το bit CREN και αμέσως μετά το θέσουμε στο 1 (reset).

Το δεύτερο σφάλμα είναι το σφάλμα πλαισίου, που υποδηλώνεται από το bit FERR, του καταχωρητή ελέγχου RCSTA. Αν προσέξουμε το σχήμα 6.23, θα δούμε ότι το bit τέλους έχει λογικό 1. Το λάθος πλαισίου προέρχεται από τη λήψη του bit αυτού με λογικό 0, κάτι που δεν είναι αναμενόμενο, και σημαίνει ότι η όλη μετάδοση είχε κάποιο πρόβλημα. Τότε, το bit FERR γίνεται 1. Για να πάρουμε τη σωστή τιμή του bit αυτού, πρέπει να διαβάσουμε τον καταχωρητή ελέγχου RCSTA, πριν διαβάσουμε τον καταχωρητή RCREG, αφού το τελευταίο θα φορτώσει στα bits FERR και RX9D, νέες τιμές.

Ας δούμε, τώρα, επιγραμματικά τα βήματα που πρέπει να ακολουθήσουμε για να ορίσουμε μία ασύγχρονη σειριακή λήψη:

1. Αρχικοποίηση του καταχωρητή SPBRG για το κατάλληλο baud rate. Αν είναι επιθυμητή υψηλή ταχύτητα τότε το bit BRGH = 1.
2. Ενεργοποίηση ασύγχρονης σειριακής θύρας με τα bits SYNC = 0 και SPEN = 1.
3. Αν οι διακοπές είναι επιθυμητές, τότε, τα bits RCIE = 1, GIE = 1 και PEIE = 1.
4. Αν η μετάδοση 9 ψηφίων είναι επιθυμητή, τότε, το bit RX9 = 1.
5. Εκκίνηση λήψης με το bit CREN = 1.
6. Η σημαία RCIF γίνεται 1, όταν η μετάδοση έχει ολοκληρωθεί, ενώ προκαλείται διακοπή αν το bit RCIE = 1.
7. Ανάγνωση του καταχωρητή RCSTA για την λήψη του 9ου bit, αν αυτό έχει ορισθεί, και έλεγχος για πιθανά σφάλματα που μπορεί να προκύψουν κατά την λήψη.
8. Ανάγνωση του δεδομένου από τον καταχωρητή RCREG.
9. Αν προκύψουν σφάλματα η λήψη σταματά. Η επανεκκίνησή της γίνεται με reset του bit CREN.

Ας δούμε τώρα ένα παράδειγμα ασύγχρονης λειτουργίας αυτής της περιφερειακής συσκευής. Έστω ότι θέλουμε η συσκευή να λειτουργεί με Baud rate 9600, με επικοινωνία 8 bits και ότι ο PIC λειτουργεί στα 20MHz.

Για χαμηλή ταχύτητα (BRGH = 0), από τον τύπο 6.4, πρέπει να δώσουμε στον καταχωρητή SPBRG την τιμή 31 (1F H). Ακόμη, για τη λειτουργία εκπομπής πρέπει να ορίσουμε επικοινωνία 8 bits (TX9 = 0) και να την ενεργοποιήσουμε (TXEN = 1). Άρα, πρέπει να δώσουμε για την εκπομπή την τιμή 0100000 (40 H) στον καταχωρητή TXSTA. Ακόμη, για τη λειτουργία λήψης πρέπει να ορίσουμε επικοινωνία 8 bits (RX 9 = 0) και να την ενεργοποιήσουμε (RCEN = 1). Επίσης, πρέπει να ενεργοποιήσουμε τους ακροδέκτες TX/CK και RX/DT (SPEN = 1). Άρα, πρέπει να δώσουμε για την λήψη την τιμή 10010000 (90 H) στον καταχωρητή RCSTA. Ας σημειώσουμε ότι θέτουμε στο "0" όσα από τα bits των δύο καταχωρητών δεν χρησιμοποιούνται. Στην συνέχεια, παρουσιάζουμε τον κώδικα αρχικοποίησης της περιφερειακής συσκευής για τη λειτουργία της ασύγχρονης μετάδοσης / λήψης:

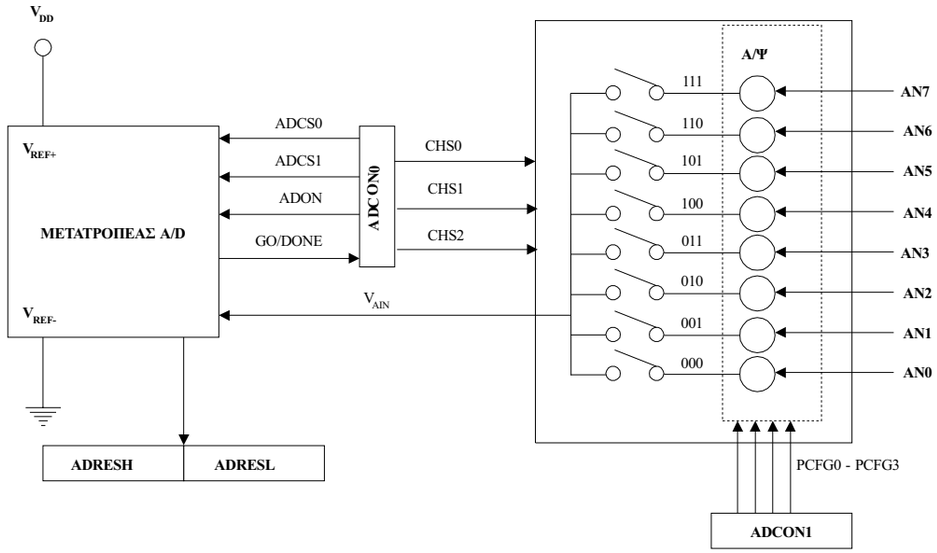

```
BSF STATUS,RP0 ; Επιλογή τμήματος 1 μνήμης.

; Ορισμός Baud rate στα 9600 (τύπος 6.4)
MOVLW 1F H
MOVWF SPBRG

; ΑΡΧΙΚΟΠΟΙΗΣΗ ΠΟΜΠΟΥ.
; Επιλογή και ενεργοποίηση λειτουργίας ασύγχρονης
; μετάδοσης 8 ψηφίων χαμηλής ταχύτητας.
MOVLW 40 H
MOVWF TXSTA

BSF PIE1,TXIE ; Ενεργοποίηση διακοπής μετάδοσης
BSF PIE1,RCIE ; Ενεργοποίηση διακοπής λήψης

; ΑΡΧΙΚΟΠΟΙΗΣΗ ΔΕΚΤΗ
BCF STATUS,RP0 ; Επιλογή τμήματος 0 μνήμης.
; Επιλογή και ενεργοποίηση λειτουργίας λήψης 8 ψηφίων.
MOVLW 90 H
MOVWF RCSTA
```

Ερωτήσεις

1. Με ποιους τρόπους μπορεί να λειτουργήσει η συσκευή σειριακής επικοινωνίας;
2. Πόσους και ποιους καταχωρητές ελέγχου έχει η παρούσα περιφερειακή συσκευή;
3. Ποιες είναι οι σημαίες διακοπής στην ασύγχρονη λειτουργία και σε ποιόν καταχωρητή μπορούμε να τις βρούμε;

6.5 Μετατροπέας αναλογικού σήματος σε ψηφιακό

Η περιφερειακή συσκευή μετατροπής αναλογικού σήματος σε ψηφιακό (A/D) διαθέτει 8 αναλογικές εισόδους (AN0 - AN7), που βρίσκονται στις θύρες A (RA0/AN0, RA1/AN1, RA2/AN2, RA3/AN3 και RA5/AN4) και E (RE0/AN5, RE1/AN6 και RE2/AN7). Το αναλογικό σήμα μετατρέπεται σε ένα δυαδικό αριθμό 10 bits. Το περιφερειακό χρησιμοποιεί την τάση λειτουργίας του PIC, V_{DD}, ως τάση αναφοράς, την οποία και συνδέουμε στο V_{REF+}, ενώ συνδέουμε το V_{REF-} στη γείωση. Το διάγραμμα λειτουργίας του A/D μετατροπέα φαίνεται στο σχήμα 6.27.



Σχήμα 6.27 *Η μονάδα μετατροπής A/D.*

Η συσκευή χρησιμοποιεί 4 καταχωρητές για να λειτουργήσει. Αυτοί είναι οι:

- A/D Καταχωρητής ελέγχου 0 (ADCON0)
- A/D Καταχωρητής ελέγχου 1 (ADCON1)
- A/D Καταχωρητής των 8 πιο σημαντικών bits του αποτελέσματος (ADRESH)
- A/D Καταχωρητής των 2 λιγότερο σημαντικών bits του αποτελέσματος (ADRESL)

Με τον πρώτο καταχωρητή ελέγχουμε τη λειτουργία της περιφερειακής συσκευής. Τον δεύτερο τον χρησιμοποιούμε για να ορίσουμε τη λειτουργία των ακροδεκτών που θα χρησιμοποιηθούν. Τους ακροδέκτες μπορούμε να τους ορίσουμε σαν αναλογικές εισόδους ή σαν ψηφιακές εισόδους / εξόδους. Ο τρίτος και ο τέταρτος καταχωρητής (ADRESH και ADRESL) αποθηκεύουν το αποτέλεσμα της A/D μετατροπής, όπως φαίνεται στο σχήμα 6.28.

ΑΠΟΤΕΛΕΣΜΑ A/D ΜΕΤΑΤΡΟΠΗΣ									
BITS ADRESH								BITS ADRESL	
7	6	5	4	3	2	1	0	7	6

Σχήμα 6.28 *Ο τρόπος αποθήκευσης του αποτελέσματος των 10 bits, της A/D μετατροπής, στους καταχωρητές ADRESH και ADRESL.*

Θα περιγράψουμε, τώρα, τους δύο καταχωρητές ελέγχου και, στη συνέχεια, θα μελετήσουμε τη λειτουργία του περιφερειακού. Ξεκινάμε, λοιπόν, με τον καταχωρητή ADCON0, που βρίσκεται στο τμήμα 0 της μνήμης δεδομένων, ο οποίος φαίνεται στο σχήμα 6.29.

7	6	5	4	3	2	1	0
ADCS1	ADCS0	CHS2	CHS1	SHS0	GO/DONE	-	ADON

Σχήμα 6.29 *Ο καταχωρητής ADCON0*

- ***ADCS1 - ADCS0:*** Bit επιλογής ρολογιού μετατροπής A/D.
  00 = FOSC/2
  01 = FOSC/8
  10 = FOSC/32
  11 = FRC (πηγή ρολογιού ο εσωτερικός A/D RC ταλαντωτής)

- ***CHS2 - CHS0:*** Bits επιλογής αναλογικού καναλιού.
  000 = κανάλι 0, (AN0)
  001 = κανάλι 1, (AN1)
  010 = κανάλι 2, (AN2)
  011 = κανάλι 3, (AN3)
  100 = κανάλι 4, (AN4)
  101 = κανάλι 5, (AN5)
  110 = κανάλι 6, (AN6)
  111 = κανάλι 7, (AN7)

- ***GO/DONE:*** Bit ένδειξης κατάστασης A/D μετατροπής, όταν το bit ADON είναι 1.
  Αν GO/DONE = 1, η A/D μετατροπή είναι σε εξέλιξη.
  Αν GO/DONE = 0, η A/D μετατροπή ολοκληρώθηκε και το αποτέλεσμα είναι έτοιμο.

- ***Bit 1:*** Δε χρησιμοποιείται. Διαβάζεται ως "0".

- ***ADON:*** Bit εκκίνησης A/D μετατροπέα.
  1 = A/D Εκκίνηση μετατροπέα.
  0 = A/D Παύση μετατροπέα.

Μπορούμε να γράψουμε και να διαβάσουμε όλα τα bits του καταχωρητή αυτού. Ο επόμενος καταχωρητής, που θα εξετάσουμε, είναι ο ADCON1, που βρίσκεται στο τμήμα 1 της μνήμης δεδομένων, τον οποίον και παρουσιάζουμε στο σχήμα 6.30.

7	6	5	4	3	2	1	0
ADCS1	ADCS0	CHS2	CHS1	SHS0	GO/DONE	-	ADON

Σχήμα 6.30 *Ο καταχωρητής ADCON1*

- ***Bit 7:*** Τίθεται στο "0".

- ***Bits 6 έως 4:*** Δεν χρησιμοποιούνται. Διαβάζονται σαν λογικό 0.

- ***PCFG3 - PCFG0:*** Bits ορισμού ακροδεκτών A/D μετατροπέα.
  A = Αναλογική είσοδος
  Ψ = Ψηφιακή είσοδος / έξοδος

Στις περισσότερες εφαρμογές χρησιμοποιούνται οι παρακάτω καταστάσεις:

PCFG3 - 0	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
0000	A	A	A	A	A	A	A	A
0010	Ψ	Ψ	Ψ	A	A	A	A	A
0100	Ψ	Ψ	Ψ	Ψ	A	Ψ	A	A
0110	Ψ	Ψ	Ψ	Ψ	Ψ	Ψ	Ψ	Ψ
1001	Ψ	Ψ	A	A	A	A	A	A
1110	Ψ	Ψ	Ψ	Ψ	Ψ	Ψ	Ψ	A

Η λειτουργία του A/D μετατροπέα είναι σχετικά απλή. Μόλις ολοκληρωθεί η μετατροπή, το αποτέλεσμα φορτώνεται στους καταχωρητές ADRESH και ADRESL, που βρίσκονται, αντιστοίχως, στα τμήματα 0 και 1 της μνήμης δεδομένων. Παράλληλα μηδενίζεται το bit GO/DONE του καταχωρητή ADCON0 και η σημαία A/D διακοπής, η ADIF του καταχωρητή PIR1 (σχήμα 6.31), γίνεται 1.

7	6	5	4	3	2	1	0
	ADIF						

Σχήμα 6.31 *Η σημαία ADIF στον καταχωρητή PIR1.*

7	6	5	4	3	2	1	0
	ADIE						

Σχήμα 6.32 *Η σημαία ADIE στον καταχωρητή PIE1.*

Αυτό που μας ενδιαφέρει περισσότερο είναι ο χρόνος που χρειάζεται για να ολοκληρωθεί η διαδικασία μετατροπής. Από αυτόν μπορούμε να υπολογίσουμε τη μέγιστη συχνότητα των σημάτων, $F_{Σήματος}$, που μπορούμε να μετατρέπουμε σε ψηφιακά, χωρίς να υπεισέρχεται σημαντικό σφάλμα, ως εξής:

$$F_{Μετατροπής} = 2 \cdot F_{Σήματος} \qquad (6.6)$$

Με τον τρόπο αυτό δεν χάνουμε πληροφορία από το μετρούμενο σήμα. Ο χρόνος μετατροπής εξαρτάται από την συχνότητα λειτουργίας του PIC και από την τιμή που θα δώσουμε στα bits ADCS1 και ADCS0. Την επιλογή μας μπορούμε αν την κάνουμε με την βοήθεια του ακόλουθου πίνακα:

ΠΙΝΑΚΑΣ ΥΠΟΛΟΓΙΣΜΟΥ ΧΡΟΝΟΥ A/D ΜΕΤΑΤΡΟΠΗΣ ΓΙΑ ΤΟ ΚΑΘΕ BIT ΤΟΥ ΑΠΟΤΕΛΕΣΜΑΤΟΣ

Πηγή ρολογιού A/D		Συχνότητα Λειτουργίας PIC			
Περίοδος	**ADCS1 - 0**	**20 MHz**	**5 MHz**	**1.25 MHz**	**333.33 kHz**
2Tosc	**00**	-	-	1,6μs	6μs
8Tosc	**01**	-	1,6μs	6,4μs	24μs
32Tosc	**10**	1,6μs	6,4μs	25,6μs	96μs
RC	**11**	2-6μs	2-6μs	2-6μs	2-6μs

Ο χρόνος που φαίνεται στον πίνακα είναι ίσος με την περίοδο του ρολογιού του κυκλώματος μετατροπής. Δεδομένου ότι ο μετατροπέας χρειάζεται περίπου 10 κύκλους για να ολοκληρώσει μια μετατροπή στην περίπτωση που για παράδειγμα η συχνότητα λειτουργίας είναι 20MHz η ελάχιστη περίοδος του ρολογιού είναι 1,6μs. Συνολικά χρειάζεται ο μετατροπέας χρόνο ίσο με 16 μs για να ολοκληρώσει μια μετατροπή δηλαδή μπορεί, να δέχεται δείγματα με συχνότητα 62,5 KHz.

Όταν θέλουμε να χρησιμοποιήσουμε το περιφερειακό αυτό, πρέπει να προσέξουμε να ορίσουμε τους ακροδέκτες, που χρησιμοποιεί, σαν εισόδους, στον καταχωρητή TRIS. Αν, κατά λάθος, ορίσουμε κάποιον ακροδέκτη σαν έξοδο, τότε, ο A/D μετατροπέας θα δεχθεί σαν είσοδο την τάση εξόδου του ακροδέκτη. Αυτό συμβαίνει γιατί το περιφερειακό λειτουργεί ανεξάρτητα του ορισμού των ακροδεκτών. Στην προκειμένη περίπτωση, δεν θα έχουμε καμία ένδειξη ότι λάβαμε λάθος μέτρηση.

Στην συνέχεια παρουσιάζουμε ένα παράδειγμα αρχικοποίησης του περιφερειακού:

Παράδειγμα Αρχικοποίησης A/D περιφερειακού

΄Εστω ότι θέλουμε να ορίσουμε τους 8 ακροδέκτες ως αναλογικές εισόδους (PCFG3 - PCFG0 = 0000). Τότε, θα πρέπει να δώσουμε στον καταχωρητή ADCON1 την τιμή 0. Επίσης, έστω ότι θέλουμε να ενεργοποιήσουμε τον A/D μετατροπέα (ADON = 1), ώστε να χρησιμοποιεί το RC ρολόι (ADCS1 - ADCS0 = 11), και να δέχεται το δείγμα από το κανάλι 0 (CHS2 -CHS0 =000). Επίσης, πρέπει να προσέξουμε η εκκίνηση της A/D μετατροπής (GO/DONE = 1) να γίνεται με χωριστή εντολή από την ενεργοποίηση της συσκευής (ADON = 1), ώστε να αφήσουμε λίγο χρόνο να σταθεροποιηθεί το σήμα στην είσοδο. Συνεπώς, πρέπει να δώσουμε στον καταχωρητή ADCON0 την τιμή 11000001 (C1 H). Ο κώδικας αρχικοποίησης έχει ως ακολούθως:

```
; Και οι 8 ακροδέκτες ορίζονται ως αναλογικές είσοδοι
BSF STATUS, RP0 ; Επιλογή τμήματος 1 μνήμης.
MOVLW 0
MOVWF ADCON1
```

```
Επιλογή RC Ρολογιού - Ενεργοποίηση A/D - Επιλογή καναλιού 0
BCF STATUS, RP0 ; Επιλογή τμήματος 0 μνήμης.
MOVLW C1 H
MOVWF ADCON0

;Μηδενισμός σημαίας - Ενεργοποίηση διακοπών
BCF PIR1, ADIF ;Μηδενισμός bit σημαίας A/D διακοπής
BSF PIE1, ADIE ; Ενεργοποίηση διακοπής A/D
BSF INTCON, PEIE ;Ενεργοποίηση διακοπής περιφερειακών
BSF INTCON, GIE ;Ενεργοποίηση όλων των διακοπών

; Εκκίνηση A/D μετατροπής.
BSF ADCON0, GO
```

Όταν λάβουμε την διακοπή (ADIF=1), τότε διαβάζουμε το περιεχόμενο των καταχωρητών ADRESH και ADRESL, το οποίο, όπως είπαμε, είναι το αποτέλεσμα της μετατροπής.

Ερωτήσεις

1. Πόσες εισόδους διαθέτει η περιφερειακή συσκευή A/D μετατροπής;\
2. Πως μπορούμε να επιλέξουμε ποια είσοδο (κανάλι) θα μετατρέψει η συσκευή;
3. Πόσα δυαδικά bits είναι μία μετατροπή;
4. Από πού μπορούμε να διαβάσουμε το αποτέλεσμα της μετατροπής;

6.6 Επίλογος

Στο βιβλίο ασχοληθήκαμε με τη χρήση των μικροελεγκτών σε μικροϋπολογιστικές εφαρμογές. Μελετήσαμε το μικροελεγκτή PIC της εταιρείας Microchip, ο οποίος ανήκει στην οικογένεια των μικροελεγκτών και μικροεπεξεργαστών με αρχιτεκτονική RISC. Εδώ, πρέπει να τονίσουμε ότι εξετάσαμε μόνο λίγες από τις πολλές περιφερειακές συσκευές που διαθέτει.

Στο εμπόριο, ο PIC, διατίθεται σε πολλές εκδόσεις, με λιγότερα ή περισσότερα περιφερειακά, με μεγαλύτερες ή μικρότερες μνήμες διαφορετικών τεχνολογιών, κτλ.

Ακόμη, διαφέρει σε σχήμα, αριθμό ακροδεκτών, καθώς και κόστος, ανάλογα με τις δυνατότητες που προσφέρει. Έτσι, ο μικροελεγκτής μας μπορεί να χρησιμοποιηθεί σε πολλές εφαρμογές, από απλές μέχρι ιδιαίτερα πολυσύνθετες.

Ολοκληρώνοντας, πρέπει να προσθέσουμε ότι στην αγορά υπάρχουν και πολλοί άλλοι αξιόλογοι μικροελεγκτές, όπως των εταιρειών Intel, Motorola, κ.α. Ο καλύτερος μικροελεγκτής είναι αυτός που είναι ο περισσότερο κατάλληλος, τόσο σε δυνατότητες όσο και σε κόστος, για την καθεμία εφαρμογή ξεχωριστά.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. ***Microchip Technology Inc.***, "PICmicro Mid - Range MCU Family" Reference Manual, 1997
2. ***Microchip Technology Inc.***, "PIC 16F87X 28/40-pin 8-Bit CMOS FLASH Microcontrollers", Reference Manual, 1999