ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ & ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

Εργαστήριο
Μικροελεγκτές - Ενσωματωμένα Συστήματα

1ος ΚΥΚΛΟΣ

Μικροελεγκτές Atmel – AVR

ΑΣΚΗΣΗ 1η

Εξοικείωση με το λογισμικό AVR Studio
Χρονισμός – Λειτουργίες I/O

Γ. Καλτσάς
Καθηγητής

Αθήνα 2019

Εισαγωγή

Η πρώτη ενότητα του εργαστηρίου «Μικροελεγκτές – Ενσωματωμένα Συστήματα» έχει αντικείμενο την μελέτη των μικροελεγκτών της οικογένειας AVR της Atmel. Πρόκειται για 8-bit RISC μικροελεγκτές οι οποίοι βρίσκουν ευρύτατες εφαρμογές στον αυτόματο έλεγχο, στη βιομηχανία, στα ενσωματωμένα συστήματα (embedded systems), στην διαμεταγωγή δεδομένων κτλ.

Η μελέτη των συγκεκριμένων μικροελεγκτών θα εστιαστεί στο επίπεδο ανάπτυξης εφαρμογών με χρήση συμβολικής γλώσσας (assembly language). Για το λόγο αυτό θα χρησιμοποιηθεί ένα ειδικό λογισμικό που παρέχεται από την ATMEL-Microchip (https://www.microchip.com/mplab/avr-support/atmel-studio-7) και ονομάζεται AVR Studio. Το συγκεκριμένο πρόγραμμα είναι ένα ολοκληρωμένο παραθυρικό περιβάλλον ανάπτυξης (IDE: Integrated Development Environment) και διόρθωσης (debugging) εφαρμογών βασισμένων σε μικροελεγκτές AVR. Το λογισμικό AVR Studio παρέχει εργαλείο διαχείρισης έργων (project manager tool), συντάκτη αρχείων πηγής (source file editor), προσομοιωτή O.K. (chip simulator) και περιβάλλον εξομοίωσης ενσωματωμένου κυκλώματος (In-circuit emulator interface) για μικροελεγκτές της οικογένειας AVR. Επιπλέον, το συγκεκριμένο λογισμικό υποστηρίζει το αναπτυξιακό STK600, το οποίο επιτρέπει τον προγραμματισμό όλων των διατάξεων AVR, και τον εξομοιωτή JTAG (on-chip emulator). Το AVR Studio έχει απόλυτα δομημένη αρχιτεκτονική (modular architecture) που επιτρέπει ακόμη περισσότερη αλληλεπίδραση με τους προμηθευτές συμπληρωματικού λογισμικού (3rd party software vendors).

Γενικά Χαρακτηριστικά Μικροελεγκτών Της Οικογένειας AVR

Οι μικροελεγκτές της οικογένειας AVR περιλαμβάνουν έναν επεξεργαστή RISC ο οποίος έχει σχεδιαστεί σύμφωνα με την αρχιτεκτονική Harvard. Σύμφωνα με την συγκεκριμένη αρχιτεκτονική, χρησιμοποιούνται διαφορετικές μονάδες μνήμης για την αποθήκευση του προγράμματος και των δεδομένων. Κατά συνέπεια η CPU συνεργάζεται ταυτόχρονα με μια μνήμη προγράμματος (τύπου Flash) και με δύο μονάδες μνήμης, όπου αποθηκεύονται τα δεδομένα – Data memories (τύπων SRAM και EEPROM). Για τους τρόπους προσπέλασης μνήμης θα αναφερθούμε εκτενέστερα σε επόμενη άσκηση του συγκεκριμένου κύκλου.

Η οικογένεια των μικροελεγκτών AVR παρουσιάζει κάποια γενικά χαρακτηριστικά:

- ✓ Ενσωματωμένη (on-chip) μνήμη ταχείας αποθήκευσης (τύπου Flash) με δυνατότητα προγραμματισμού εντός του συστήματος (ISP – In System Programmable).
- ✓ Ενσωματωμένη μνήμη δεδομένων τύπου SRAM και EEPROM.
- ✓ Παρέχονται 32 καταχωρητές εργασίας των 8-bit. Ο μεγάλος σχετικά αυτός αριθμός καταχωρητών δίνει την δυνατότητα σε διάφορες μεταβλητές να αποθηκεύονται εντός της CPU και όχι σε μνήμη, καθώς η διαδικασία προσπέλασης μνήμης θεωρείται χρονοβόρα.
- ✓ Λειτουργία σε συχνότητες χρονισμού έως 32 MHz. Οι περισσότερες εντολές συμβολικής γλώσσας ολοκληρώνονται σε μια περίοδο του κεντρικού σήματος

χρονισμού. Οι προσπελάσεις στη μνήμη και οι διακλαδώσεις (όταν πραγματοποιούνται) απαιτούν δύο κύκλους ρολογιού.

- ✓ Διατίθεται εσωτερικό κύκλωμα επανατοποθέτησης κατά την εφαρμογή τάσης τροφοδοσίας (POR – Power On Reset).
- ✓ Διατίθεται ενσωματωμένος προγραμματιζόμενος χρονιστής με ιδιαίτερη μονάδα διαίρεσης συχνότητας (prescaler).
- ✓ Διατίθενται πηγές εσωτερικών ή εξωτερικών διακοπών.
- ✓ Παρέχεται προγραμματιζόμενος χρονιστής επιτήρησης (WDT) οδηγούμενος από ιδιαίτερο ταλαντωτή.
- ✓ Παρέχονται λειτουργίες ηρεμίας (Sleep) και αποκοπής (Power Down).
- ✓ Σε πολλούς από τους μικροελεγκτές της σειράς διατίθεται ενσωματωμένος ταλαντωτής τύπου RC.

Στην παρούσα άσκηση θα αναλυθούν τρεις βασικές λειτουργίες με τη βοήθεια του λογισμικού AVR Studio:

Α) Η εισαγωγή δεδομένων σε καταχωρητές εργασίας
Β) Η διαδικασία εισόδου-εξόδου δεδομένων, που πραγματοποιείται μέσω των παράλληλων θυρών.
Γ) Ο χρονισμός, ο οποίος είναι πρωταρχικής σημασίας στην εκτίμηση του χρόνου εκτέλεσης του αναπτυσσόμενου κώδικα.

Εισαγωγή Δεδομένων σε Καταχωρητή Εργασίας

Κατά την συγγραφή προγραμμάτων συμβολικής γλώσσας είναι απαραίτητο να εισάγονται δεδομένα σε καταχωρητές εργασίας. Η βασική εντολή, η οποία «φορτώνει» σε έναν καταχωρητή μια συγκεκριμένη τιμή 8-bit είναι η LDI (Load Immediate). Η σύνταξή της ακολουθεί τον γενικό τύπο:

LDI Rd, K ***16 ≤ d ≤ 31, 0 ≤ K ≤ 255***

Κατά συνέπεια μέσω της εντολής LDI μπορεί να φορτωθεί άμεσα ένας καταχωρητής της δεύτερης 16αδας (R16 → R31) με μια τιμή 8-bit. Η συγκεκριμένη τιμή μπορεί γενικά να αναφερθεί σε τρία συστήματα αρίθμησης:

- ➢ Στο δεκαδικό όταν δεν χρησιμοποιείται κανένα πρόθεμα.
- ➢ Στο δεκαεξαδικό όταν χρησιμοποιείται το πρόθεμα “$” ή “0x”.
- ➢ Στο δυαδικό όταν χρησιμοποιείται το πρόθεμα “0b”.

Κατά συνέπεια τα παρακάτω παραδείγματα είναι ισοδύναμα και φορτώνουν στον καταχωρητή R21 την τιμή 255:

```
LDI R20, 255
LDI R20, $FF
LDI R20, 0xFF
LDI R20, 0b11111111
```

Προφανώς επειδή δεν υπάρχει η δυνατότητα να αποδοθεί άμεσα τιμή σε καταχωρητή της πρώτης 16αδας (R0 → R15) χρησιμοποιείται έμμεση απόδοση μέσω της εντολής ***MOV***:

Παράδειγμα:

```
LDI R17, 0x0F      ; Φόρτωση της τιμής 0x0F (16) στον καταχωρητή R17
MOV R5, R17        ; Αντιγραφή της συγκεκριμένης τιμής στον καταχωρητή R5
```

Λόγου του ότι στις περισσότερες των περιπτώσεων χρησιμοποιείται η φόρτωση καταχωρητών με τις τιμές 0 και 255 παρέχονται δύο ακόμη εντολές για ευκολία:

- Η εντολή ***SER***, η οποία συντάσσεται με έναν καταχωρητή και του αποδίδει την τιμή 255
- Η εντολή ***CLR*** , η οποία συντάσσεται με έναν καταχωρητή και μηδενίζει το περιεχόμενό του.

Παραδείγματα:

```
SER R23      ; Είναι ισοδύναμη με την εντολή LDI R23, 255
CLR R30      ; Είναι ισοδύναμη με την εντολή LDI R30, 0
```

Παρατήρηση:
Για την εισαγωγή σχολίων σε προγράμματα συμβολικής γλώσσας του AVR χρησιμοποιείται το σύμβολο του Ελληνικού ερωτηματικού «;». Οτιδήποτε ακολουθεί το ερωτηματικό θεωρείται σχόλιο και δεν λαμβάνεται υπ' όψη από τον συμβολομεταφραστή.

Οι Θύρες Παράλληλης Εισόδου – Εξόδου Της Οικογένειας AVR

Όλοι οι μικροελεγκτές της οικογένειας AVR διαθέτουν μεταξύ άλλων και κάποιο αριθμό ακροδεκτών εισόδου – εξόδου, ξεκινώντας από 3 που συναντούμε στον μικροελεγκτή AT90S2323, μέχρι και 68, που συναντούμε στον μικροελεγκτή Mega6490. Όλοι οι ακροδέκτες εισόδου – εξόδου των μικροελεγκτών AVR έχουν ικανότητα ρεύματος που ανέρχεται στη τιμή των 20 mA, ποσότητας ικανής για απευθείας οδήγηση διόδων LED χωρίς την ανάγκη χρήσης κατάλληλης εξωτερικής μονάδας απομόνωσης (buffer).

Για όλες τις θύρες εισόδου – εξόδου του συστήματος υπάρχουν συνολικά τρεις διευθύνσεις (PIN, PORT, DDR) στη μνήμη εισόδου – εξόδου, οι οποίες σχετίζονται με κάθε μία από αυτές. Η μία από τις διευθύνσεις αυτές (DDR) απαιτείται για τον καθορισμό της κατεύθυνσης των ακροδεκτών, δηλαδή καθορίζει ποιοι από τους ακροδέκτες θα λειτουργούν ως είσοδοι και ποιοι ως έξοδοι. Η δεύτερη διεύθυνση (PORT) αφορά στα δεδομένα που πρόκειται να εγγραφούν σε εκείνους (ή και όλους) τους ακροδέκτες που έχουν προγραμματιστεί ως έξοδοι, ενώ τέλος, η τρίτη διεύθυνση (PIN) αφορά στα δεδομένα που διαβάζονται από εκείνους (ή και όλους) τους ακροδέκτες που έχουν προγραμματιστεί ως είσοδοι.

Οι παραπάνω διευθύνσεις συμβολίζονται ως DDRx, PORTx και PINx, για τη συγκεκριμένη φυσική θύρα x. Ο καταχωρητής DDRx είναι ο λεγόμενος καταχωρητής κατεύθυνσης. Τοποθετώντας ένα bit του καταχωρητή αυτού σε λογικό 1, ο

ακροδέκτης της θύρας x στον οποίο αντιστοιχεί, θα λειτουργεί ως έξοδος. Μετά από αυτή την ενέργεια μπορούμε να οδηγούμε κατά βούληση την λογική στάθμη του αντίστοιχου ακροδέκτη εξόδου χρησιμοποιώντας κατάλληλη εντολή (OUT).

Παρόμοια, για να διαβάσουμε τη τιμή του λογικού επιπέδου ενός ακροδέκτη εισόδου, χρησιμοποιούμε τον καταχωρητή PINx. Η μονάδα στην οποία αντιστοιχεί ο καταχωρητής PINx, συνδέεται απευθείας στον ακροδέκτη της θύρας. Τοποθετώντας το bit του καταχωρητή PORTx σε λογικό 1 ενεργοποιείται μια εσωτερική αντίσταση πρόσδεσης η οποία συνδέεται στον αντίστοιχο ακροδέκτη. Η τιμή της εσωτερικής αυτής αντίστασης κυμαίνεται από 30 ως 150 Kohm. Η αντίστοιχη τιμή του ρεύματος που διαρρέει την παραπάνω αντίσταση κυμαίνεται μεταξύ 33 και 160μA.
Αντίθετα, τοποθετώντας σε λογικό 0 το αντίστοιχο bit στην διεύθυνση PORTx, η εσωτερική αντίσταση αποσυνδέεται και ο ακροδέκτης εισόδου έρχεται σε κατάσταση υψηλής αντίστασης.

Παράλληλη Είσοδος – Έξοχος Δεδομένων σε Μικροελεγκτή AVR ATmega2560

Για τις ανάγκες του συγκεκριμένου εργαστηρίου θα χρησιμοποιηθεί ο μικροελεγκτής AVR ATmega2560. Ο μικροελεγκτής ATmega2560 διαθέτει 86 γραμμές I/O σε έντεκα (11) 8-bits θύρες παράλληλης εισόδου-εξόδου, οι οποίες αναφέρονται ως A, B, C, … , K, L. Κάθε θύρα χρησιμοποιεί τρεις διευθύνσεις μνήμης (όπως έχει αναφερθεί και στην προηγούμενη παράγραφο), μία για τον καταχωρητή δεδομένων (Data Register) την PORTX όπου X= A, B, C, … , L, μία για τον καταχωρητή κατεύθυνσης δεδομένων (Data Direction Register) την DDRX και μία για την είσοδο της θύρας (Input Pins Address) την PINX. Η διεύθυνση εισόδου της θύρας είναι μόνο ανάγνωσης, ενώ του καταχωρητή δεδομένων και του καταχωρητή κατεύθυνσης δεδομένων είναι ανάγνωσης - γραφής.

Όταν θέλουμε να χρησιμοποιήσουμε κάποια pins μιας θύρας (ή και όλα) για έξοδο δεδομένων θα πρέπει πρώτα να θέσουμε τα αντίστοιχα bits του καταχωρητή DDRX σε κατάσταση 1. Αν θέλουμε να τα χρησιμοποιήσουμε για είσοδο, θα πρέπει να θέσουμε 0 στα αντίστοιχα bits του καταχωρητή DDRX. Όταν χρησιμοποιούμε την θύρα για είσοδο αναφερόμαστε στη διεύθυνση PINX (διαβάζουμε τα δεδομένα εισόδου) ενώ όταν την χρησιμοποιούμε για έξοδο αναφερόμαστε στη διεύθυνση PORTX (γράφουμε τα δεδομένα εξόδου).

Στον παρακάτω πίνακα αναγράφονται οι διευθύνσεις των τριών καταχωρητών για κάθε μία από τις θύρες A, B, C και D. Οι συγκεκριμένες διευθύνσεις βρίσκονται στο χώρο I/O της μνήμης SRAM.

	PIN	**DDR**	**PORT**
Θύρα A	0x20	0x21	0x22
Θύρα B	0x23	0x24	0x25
Θύρα C	0x26	0x27	0x28
Θύρα D	0x29	0x2A	0x2B
…	…	…	…
Θύρα L	0x109	0x10A	0x10B

Στη γενική περίπτωση που χρειάζεται μια θύρα να λειτουργήσει σαν έξοδος, απαιτούνται οι εξής λειτουργίες:

- Το περιεχόμενου κάποιου καταχωρητή τίθεται σε 0xFF, μέσω της εντολής ***LDI***.
- Το περιεχόμενο του καταχωρητή αντιγράφεται στον καταχωρητή κατεύθυνσης (DDR) της συγκεκριμένης θύρας, μέσω της εντολής ***OUT***.
- Στη συνέχεια η θύρα είναι έτοιμη να λειτουργήσει ως έξοδος, οπότε μπορούμε να τοποθετήσουμε οποιαδήποτε τιμή στον αντίστοιχο καταχωρητή δεδομένων (PORT), μέσω της εντολής ***OUT***.

Παράδειγμα:

Στο επόμενο παράδειγμα η θύρα B τίθεται ως έξοδος και στη συνέχεια αντιγράφεται σε αυτή η τιμή 0x23.

```
LDI R20, 0xFF        ; Το περιεχόμενο του καταχωρητή R20 γίνεται 0xFF
OUT DDRB, R20        ; Όλα τα bit του καταχωρητή DDRB τίθενται σε λογικό 1.

LDI R20, 0x23        ; Το περιεχόμενο του καταχωρητή R20 γίνεται 0x23
OUT PORTB, R20       ; Το περιεχόμενο του καταχωρητή PORTB και κατά
                       συνέπεια η έξοδος της θύρας B, γίνεται 0x23.
```

Στη γενική περίπτωση που χρειάζεται μια θύρα να λειτουργήσει σαν είσοδος, απαιτούνται οι εξής λειτουργίες:

- Το περιεχόμενου κάποιου καταχωρητή τίθεται σε 0x00, μέσω της εντολής ***LDI***.
- Το περιεχόμενο του καταχωρητή αντιγράφεται στον καταχωρητή κατεύθυνσης (DDR) της συγκεκριμένης θύρας, μέσω της εντολής ***OUT***.
- Στη συνέχεια η θύρα είναι έτοιμη να λειτουργήσει ως είσοδος, οπότε μπορούμε να διαβάσουμε οποιαδήποτε τιμή από την είσοδο της θύρας (PIN), μέσω της εντολής ***IN***.

Παράδειγμα:

Στο επόμενο παράδειγμα η θύρα D τίθεται ως είσοδος και στη συνέχεια αντιγράφεται η τιμή της στον καταχωρητή R21.

```
LDI R20, 0x00        ; Το περιεχόμενο του καταχωρητή R20 γίνεται 0x00
OUT DDRD, R20        ; Όλα τα bit του καταχωρητή DDRD τίθενται σε λογικό 0.

IN R21, PIND         ; Στον καταχωρητή R21 αποθηκεύεται η τιμή της εισόδου
                       της θύρας D (PIND).
```

Ανάπτυξη Προγράμματος Συμβολικής Γλώσσας Για Είσοδο – Έξοδο Δεδομένων Με το Λογισμικό AVR Studio

Στη συνέχεια θα αναπτυχθεί πρόγραμμα συμβολικής γλώσσας του AVR μέσω του AVR Studio το οποίο θα επιτελεί την εξής λειτουργία: θα λαμβάνει ένα byte από την θύρα Α και θα το αντιγράφει στην θύρα Β.

Αρχικά, σύμφωνα με τις συνθήκες του προβλήματος, θα πρέπει να τεθεί η θύρα Α ως είσοδος (αφού θα διαβάζονται δεδομένα από αυτή) και η θύρα Β ως έξοδος (αφού θα γράφονται δεδομένα σε αυτή). Στη συνέχεια θα πρέπει να δημιουργηθεί ένας συνεχώς επαναλαμβανόμενος βρόχος (loop) που θα εκτελεί την διαδικασία αντιγραφής δεδομένων. Για την δημιουργία του συγκεκριμένου βρόχου θα χρησιμοποιηθεί η εντολή ***rjmp***, η οποία συνάσσεται με μια ετικέτα (label) και στέλνει τον έλεγχο του προγράμματος στο σημείο που σηματοδοτεί η ετικέτα (unconditional jump). Σύμφωνα με τα παραπάνω, ο κώδικας του προγράμματος θα είναι ο εξής:

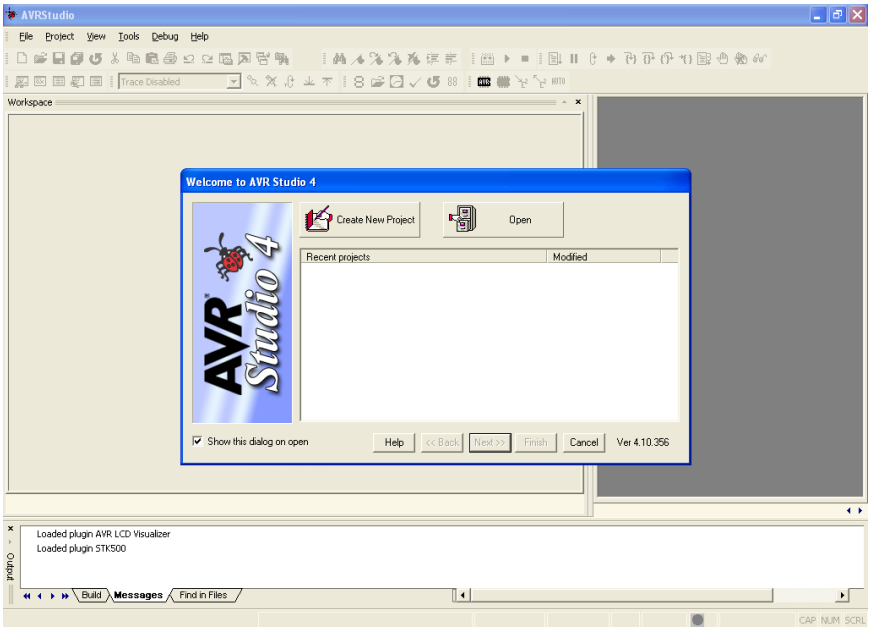
```
        ldi r25,0           ; Το περιεχόμενο του καταχωρητή R25 γίνεται 0x00
        out DDRA,r25        ; Η πόρτα Α τίθεται σε λειτουργία εισόδου

        ldi r25,255         ; Το περιεχόμενο του καταχωρητή R25 γίνεται 0xFF
        out DDRB,r25        ; Η πόρτα Β τίθεται σε λειτουργία εξόδου

start:
        in r23,PINA         ; Διαβάζεται η είσοδος της πόρτας Α (PINA) και το
                            ; περιεχόμενό της αποθηκεύεται στον καταχωρητή R23.
        out PORTB,r23       ; Το περιεχόμενο του καταχωρητή R23 αντιγράφεται στην
                            ; έξοδο της πόρτας Β (PORTB).
        rjmp start          ; Ο έλεγχος μεταφέρεται στο σημείο start για να ξαναρχίσει
                            ; η όλη διαδικασία.
```

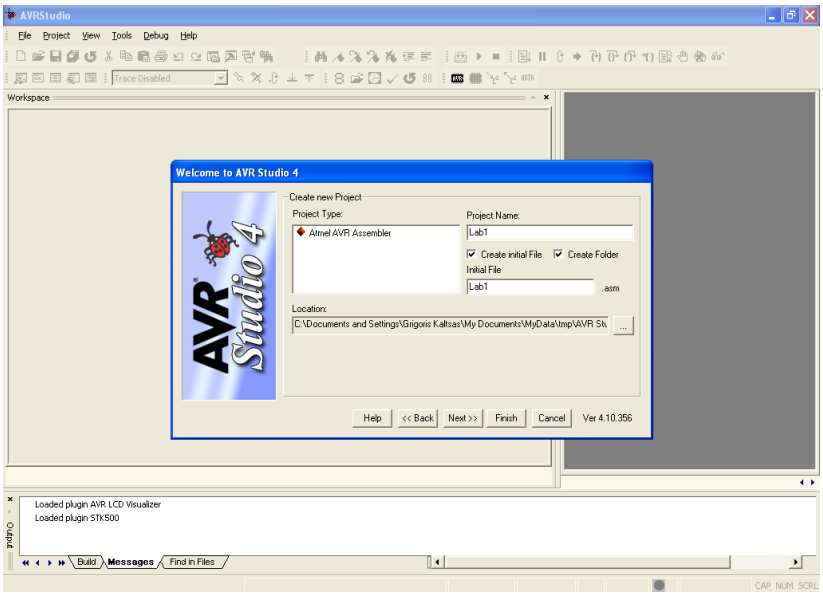
Η προσομοίωση της λειτουργίας του παραπάνω προγράμματος θα γίνει μέσω του λογισμικού AVR Studio. Στο συγκεκριμένο λογισμικό χρησιμοποιείται η έννοια του έργου (project). Αμέσως μετά την έναρξη του προγράμματος εμφανίζεται μια οθόνη που προτρέπει το χρήστη να δημιουργήσει ένα νέο project (ή να ανοίξει κάποιο ήδη υπάρχον), όπως φαίνεται στο σχήμα 1.

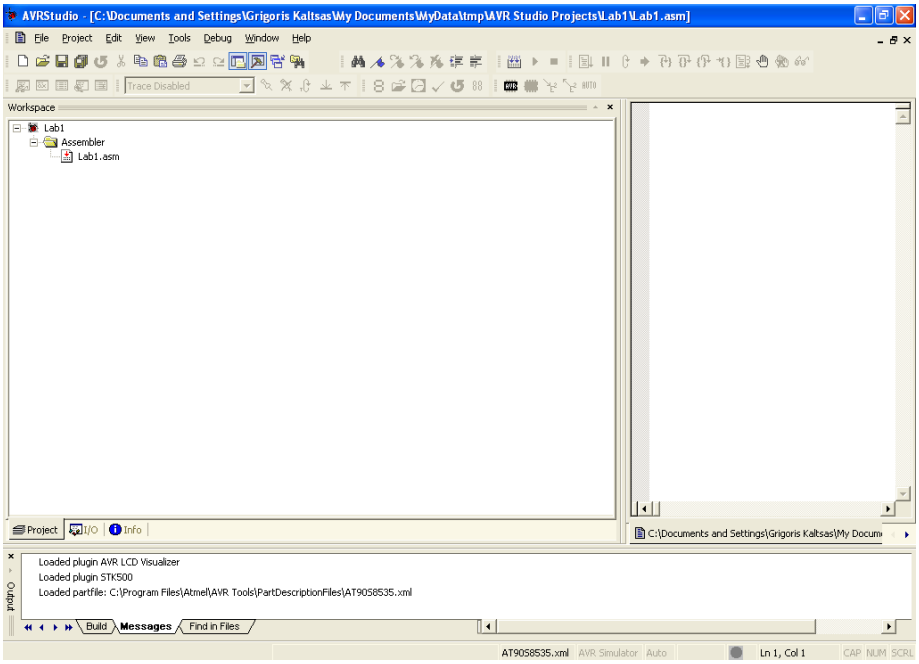
Ενεργοποιώντας την επιλογή “Create New Project” εμφανίζεται μια νέα οθόνη στην οποία προτρέπεται ο χρήστης να εισάγει το όνομα του νέου project που θέλει να δημιουργήσει, όπως φαίνεται στο σχήμα 2. Οι δύο επιλογές “Create Initial File”και “Create Folder” καλό είναι να είναι ενεργοποιημένες, έτσι ώστε όλα τα αρχεία που θα δημιουργηθούν στο project να βρίσκονται συγκεντρωμένα σε ένα κατάλογο με το ίδιο όνομα. Με την εισαγωγή του ονόματος στο πεδίο “Project Name”, ταυτόχρονα εμφανίζεται το ίδιο όνομα στο πεδίο “Initial File”. Στο συγκεκριμένο παράδειγμα έχει εισαχθεί το όνομα “Lab1”.

Μετά το πάτημα του πλήκτρου “Next” ο χρήστης προτρέπεται για να εισάγει την πλατφόρμα προσομοίωσης και τον τύπο του μικροελεγκτή. Οι επιλογές στα δύο πεδία “Debug Platform” και “Device” πρέπει να είναι “AVR Simulator” και “mega2560” αντίστοιχα.

Τελικά αφού πατηθεί το πλήκτρο “Finish” το project δημιουργείται και εμφανίζονται στην οθόνη τρία πεδία:


AVRStudio
File Project View Tools Debug Help
Trace Disabled
Workspace
Welcome to AVR Studio 4
Create New Project
Open
Recent projects
Modified
AVR Studio 4
Show this dialog on open
Help
<< Back
Next >>
Finish
Cancel
Ver 4.10.356
Loaded plugin AVR LCD Visualizer
Loaded plugin STK500
Output
Build
Messages
Find in Files
CAP NUM SCRL


AVRStudio
File Project View Tools Debug Help
Trace Disabled
Workspace
Welcome to AVR Studio 4
Create new Project
Project Type:
Atmel AVR Assembler
Project Name:
Lab1
Create initial File
Create Folder
Initial File
Lab1
.asm
Location:
C:\Documents and Settings\Grigoris Kaltsas\My Documents\MyData\tmp\AVR St
Help
<< Back
Next >>
Finish
Cancel
Ver 4.10.356
AVR Studio 4
Loaded plugin AVR LCD Visualizer
Loaded plugin STK500
Output
Build
Messages
Find in Files
CAP NUM SCRL


AVRStudio - [C:\Documents and Settings\Grigoris Kaltsas\My Documents\MyData\tmp\AVR Studio Projects\Lab1\Lab1.asm]
File Project Edit View Tools Debug Window Help
Trace Disabled
Workspace
Lab1
Assembler
Lab1.asm
Project I/O Info
C:\Documents and Settings\Grigoris Kaltsas\My Docum
Loaded plugin AVR LCD Visualizer
Loaded plugin STK500
Loaded partfile: C:\Program Files\Atmel\AVR Tools\PartDescriptionFiles\AT90S8535.xml
Output
Build Messages Find in Files
AT90S8535.xml AVR Simulator Auto
Ln 1, Col 1
CAP NUM SCRL

Ανάλυση Εργαστηριακής Άσκησης

1) Το πρόγραμμα που ακολουθεί θέτει περιοδικά 0 και 1 το πρώτο bit της πόρτας Β.
Α) Πληκτρολογήστε το συγκεκριμένο πρόγραμμα και παρατηρήστε τη λειτουργία του μέσω του προσομοιωτή AVRStudio.
Β) Υπολογίστε τον αριθμό των περιόδων του ρολογιού (clock cycles) που απαιτούνται για να αλλάξει το bit PORTB.0 από την κατάσταση 1 σε 0 και από 0 σε ένα. Τι παρατηρείτε; (Η εντολή nop απλά καθυστερεί την εκτέλεση του προγράμματος μια περίοδο ρολογιού).
Γ) Τροποποιείστε το πρόγραμμα ώστε και οι δύο μεταπτώσεις (από 1 σε 0 και από 0 σε 1) να πραγματοποιούνται σε 7 περιόδους ρολογιού.
Δ) Τροποποιείστε το πρόγραμμα ώστε και οι δύο μεταπτώσεις (από 1 σε 0 και από 0 σε 1) να πραγματοποιούνται σε 803 περιόδους ρολογιού. Υπολογίστε το πραγματικό χρονικό διάστημα μεταξύ των δύο μεταπτώσεων.

```
LDI R24,255
OUT DDRB,R24

LDI R24,1
LDI R25,0

START:

OUT PORTB,R24

LDI R16,1
LOOP1:
NOP
DEC R16
BRNE LOOP1

OUT PORTB,R25

LDI R16,1
LOOP2:
NOP
DEC R16
BRNE LOOP2

RJMP START
```

2) Αναπτύξτε πρόγραμμα, το οποίο να λαμβάνει ένα byte από την πόρτα D και να το αντιγράφει στην πόρτα Β. Παρατηρήστε τη λειτουργία του μέσω του προσομοιωτή AVRStudio. Σχολιάστε την συμπεριφορά της εισόδου της θύρας PINB.

3) Το πρόγραμμα το οποίο ακολουθεί, θέτει την θύρα Β σαν έξοδο και δημιουργεί σε αυτή συνεχώς έναν απαριθμητή modulo 256, με αυξανόμενη μέτρηση.

```
LDI R20,255
OUT DDRB,R20
LDI R20,0

START:
OUT PORTB,R20
INC R20
RJMP START
```

Πληκτρολογήστε το ανωτέρω πρόγραμμα και παρατηρήστε τη λειτουργία του μέσω του προσομοιωτή AVRStudio. Στη συνέχεια τροποποιείστε το ώστε ο απαριθμητής να είναι modulo 16 με ελαττούμενη μέτρηση.

4) Αναπτύξτε πρόγραμμα, το οποίο να αναβοσβήνει διαδοχικά το high nibble και το low nibble της θύρας B με τυχαία συχνότητα. Τροποποιήσετε το πρόγραμμα αυτό, ώστε να αναβοσβήνουν τα περιττά (PB.1, PB.3, PB.5, PB.7) και τα άρτια (PB.0, PB.2, PB.4, PB.6) leds με τυχαία συχνότητα.

5) Αναπτύξτε πρόγραμμα το οποίο να θέτει την θύρα B σαν έξοδο και να δημιουργεί σε αυτή έναν μετρητή Johnson (χρησιμοποιείστε τις εντολής ROL ή ROR). Στη συνέχεια τροποποιείστε το συγκεκριμένο πρόγραμμα ώστε το ανακυκλούμενο 1 να μην διέρχεται από το C flag (μετρητής με διαδοχικές καταστάσεις 1, 2, 4, 8, 16, 32, 64, 128).

6) Αναπτύξτε πρόγραμμα, το οποίο να ανάβει τα leds της θύρας B με την ακόλουθη σειρά α) PB.0, β) PB.0 , PB.1, γ) PB.0 , PB.1, PB.2 δ) PB.0 , PB.1, PB.2, PB.3 κλπ. Όταν ανάψουν όλα τα leds να μηδενίζονται και να ξαναρχίζει η διαδικασία.

7) Το πρόγραμμα το οποίο ακολουθεί θέτει την θύρα D σαν είσοδο και την θύρα B σαν έξοδο.
A) Αναλύστε την λειτουργία του και εκτελέστε το μέσω του προσομοιωτή AVRStudio.
B) Τροποποιείστε το πρόγραμμα έτσι ώστε οι λειτουργίες INCREMENT, DECREMENT και ROTATE να μην διακόπτονται όταν αλλάζει η τιμή της εισόδου PIND.

```
; Setup
ldi r20, 0xFF
out DDRB, r20
ldi r20, 0x00
out DDRD, r20

ldi r21, 1
ldi r22, 1

START:
        ; Read portD
        in r20, PIND

        cpi r20, 0x01
        breq INCREMENT

        cpi r20, 0x02
        breq DECREMENT

        cpi r20, 0x04
        breq ROTATE

rjmp START

INCREMENT:
        out PORTB, r21
        inc r21
        rjmp START

DECREMENT:
        out PORTB, r21
        dec r21
        rjmp START

ROTATE:
        out PORTB, r22
        rol r22
        tst r22
        brne CONTINUE
        clc
        ldi r22, 1
CONTINUE:
        rjmp START
```

ΠΑΡΑΡΤΗΜΑ:

ΑΝΑΛΥΣΗ ΧΡΗΣΙΜΟΠΟΙΟΥΜΕΝΩΝ ΕΝΤΟΛΩΝ

IN - Load an I/O Port to Register

Περιγραφή:
Φορτώνει δεδομένα από έναν I/O χώρο (Πόρτες, Μετρητές κλπ.) στον καταχωρητή Rd.

Λειτουργία:
(i) Rd ≤ P

Σύνταξη:	**Τελεστές:**	**Μετρητής προγράμματος:**
(i) IN Rd,P	0 ≤ d ≤ 31, 0 ≤ P ≤ 63	PC ← PC + 1

Παράδειγμα:
```
in r25,$16 ; Read Port B
cpi r25,4 ; Compare read value to constant
breq exit ; Branch if r25=4
...
exit: nop ; Branch destination (do nothing)
```

Words: 1 (2 bytes)
Cycles: 1

OUT - Store Register to I/O port

Περιγραφή:
Αποθηκεύει δεδομένα από τον καταχωρητή Rr σε ένα χώρο I/O (Πόρτες, Μετρητές κτλ).

Λειτουργία:
(i) P ≤ Rr

Σύνταξη:	**Τελεστές:**	**Μετρητής προγράμματος:**
(i) OUT P, Rr	0 ≤ r ≤ 31, 0 ≤ P ≤ 63	PC ← PC + 1

Παράδειγμα:
```
clr r16 ; Clear r16
ser r17 ; Set r17
out $18,r16 ; Write zeros to Port B
nop ; Wait (do nothing)
out $18,r17 ; Write ones to Port B
```

Words: 1 (2 bytes)

Cycles: 1

CBI - Clear Bit in I/O Register

Περιγραφή:
Μηδενίζει ένα καθορισμένο bit σε ένα I/O καταχωρητή. Αυτή η εντολή λειτουργεί με τους μικρότερους 32 I/O καταχωρητές – διευθύνσεις 0-31.

Λειτουργία:
(i) I/O(P,b) ← 0

Σύνταξη:	**Τελεστές:**	**Μετρητής προγράμματος:**
(i) CBI P,b	0 ≤ P ≤ 31, 0 ≤ b ≤ 7	PC ← PC + 1

Παράδειγμα:
```
cbi $12,7 ; Clear bit 7 in Port D
```

Words: 1 (2 bytes)
Cycles: 2

SBI - Set Bit in I/O Register

Περιγραφή:
Τοποθετεί μία μονάδα σε ένα συγκεκριμένο bit σε έναν I/O καταχωρητή. Αυτή η εντολή λειτουργεί με τους κατώτερους 32 I/O καταχωρητές (διευθύνσεις 0-31).

Λειτουργία:
(i) I/O(P,b) ← 1

Σύνταξη:	**Τελεστές:**	**Μετρητής προγράμματος:**
(i) SBI P,b	0 ≤ P ≤ 31, 0 ≤ b ≤ 7	PC ← PC + 1

Παράδειγμα:
```
out $1E,r0 ; Write EEPROM address
sbi $1C,0 ; Set read bit in EECR
in r1,$1D ; Read EEPROM data
```

Words: 1 (2 bytes)
Cycles: 2

LDI - Load Immediate

Περιγραφή:
Φορτώνει μία σταθερά των 8 bit απευθείας στους καταχωρητές 16 έως 31.

Λειτουργία:
(i) Rd ← K

Σύνταξη:	**Τελεστές:**	**Μετρητής προγράμματος:**
(i) LDI Rd,K	16 ≤ d ≤ 31, 0 ≤ K ≤ 255	PC ← PC + 1

Παράδειγμα:
```
clr r31 ; Clear Z high byte
ldi r30,$F0 ; Set Z low byte to $F0
lpm  ; Load constant from program
     ; memory pointed to by Z
```

Words: 1 (2 bytes)
Cycles: 1

ROL - Rotate Left trough Carry

Περιγραφή:
Μεταφέρει όλα τα bits στον Rd μία θέση αριστερά. Το C flag μεταφέρεται στο bit 0 του Rd. Το bit 7 μεταφέρεται στο C flag.

Λειτουργία:
C ←b7 - - - - - - - - - - - - - - - - - - b0 ←C

Σύνταξη:	**Τελεστές:**	**Μετρητής προγράμματος:**
(i) ROL Rd	0 ≤ d ≤ 31	PC ← PC + 1

Παράδειγμα:
```
rol r15 ; Rotate left
brcs oneenc ; Branch if carry set
...
oneenc: nop ; Branch destination (do nothing)
```

Words: 1 (2 bytes)
Cycles: 1

ROR - Rotate Right trough Carry

Περιγραφή:
Μεταφέρει όλα τα bits στον Rd μία θέση δεξιά. Το C flag μεταφέρεται στο bit 7 του Rd. Το bit 0 μεταφέρεται στο C flag.

Λειτουργία:
C →b7 - - - - - - - - - - - - - - - - - - b0 →C

Σύνταξη:	**Τελεστές:**	**Μετρητής προγράμματος:**
(i)ROR Rd	0 ≤ d ≤ 31	PC ← PC + 1

Παράδειγμα:

```
ror r15 ; Rotate right
brcc zeroenc ; Branch if carry cleared
...
zeroenc: nop ; Branch destination (do nothing)
```

Words: 1 (2 bytes)
Cycles: 1