ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΜΜΥ

ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ & ΥΛΙΚΟΥ

ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ ΓΙΑ ΤΟ ΜΑΘΗΜΑ:

ΗΡΥ 411 – ΕΝΣΩΜΑΤΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ

ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ 2020

Καθ. Α. Δόλλας

Εργαστήριο 2

ΕΞΟΙΚΕΙΩΣΗ ΜΕ ΤΗΝ ΟΙΚΟΓΕΝΕΙΑ ΜΙΚΡΟΕΛΕΓΚΤΩΝ ATMEL AVR – Μία Απλή Οθόνη 7-Segment LED

ΕΚΔΟΣΗ : 1.0

Προθεσμία: Παρασκευή 16 Οκτωβρίου 2020, έως τα μεσάνυχτα
Ηλεκτρονική υποβολή στο Webcourses

Όλα τα Εργαστήρια είναι ΑΤΟΜΙΚΑ και όχι κατά ομάδες

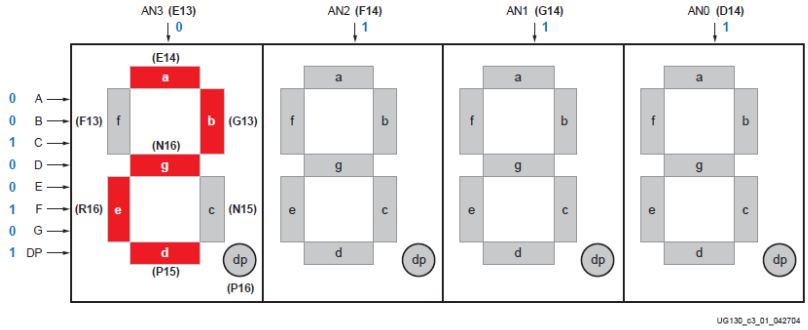
Σκοπός – Βήματα

Σκοπός του εργαστηρίου είναι η περαιτέρω εξοικείωση με το περιβάλλον ανάπτυξης για μικροελεγκτή AVR, με την δημιουργία απλού προγράμματος για οδήγηση με πολυπλεξία στον χρόνο μίας οθόνης 7-segment LED για (έως) οκτώ ψηφία. **Για τώρα, παραμένουμε στην γλώσσα Assembly.** Το εργαστήριο αυτό από πλευράς μαθησιακής στοχεύει σε μία περισσότερο ρεαλιστική χρήση του TIMER με ένα παράδειγμα πολύ κοντά στην πραγματικότητα, και έχει τρία μέρη: τον ορισμό των φυσικών διεπαφών για οδήγηση της οθόνης, τον ορισμό των λογικών διεπαφών για το περιεχόμενο που θέλουμε να παρουσιάσουμε στην οθόνη, και, τον κώδικα που (χρησιμοποιώντας παραλλαγή του κώδικα του πρώτου εργαστηρίου) δημιουργεί την οθόνη με πολυπλεξία στον χρόνο. Ο νέος κώδικας, όπως και στο πρώτο εργαστήριο, είναι λίγες γραμμές (ο καινούργιος κώδικας είναι λίγες δεκάδες γραμμές Assembly, ανάλογα πως τον υλοποιεί κανείς). Για το εργαστήριο αυτό **δεν** θα μας απασχολήσει ότι από κυκλωματικής άποψης οι έξοδοι που οδηγούν ολόκληρο το κάθε 7-segment LED μπορεί να χρειαστεί να οδηγήσουν κάποιο

τρανζίστορ για να μπορεί να δώσει αρκετό ρεύμα - εφόσον οι τιμές στις αντίστοιχες θύρες είναι οι σωστές το εργαστήριο θεωρείται πλήρες,και άρα ούτε μελετούμε ούτε σχεδιάζουμε κυκλωματικά την φυσική διεπαφή. Όπως πάντα, είναι σημαντικό να γραφτεί μία κατά το δυνατόν σύντομη αλλά περιεκτική αναφορά με τον τρόπο που λύθηκαν τα προβλήματα, ορίστηκαν οι διεπαφές (μαζί με το πως χρησιμοποιείται η μνήμη), πως έγινε η βασική δομή του κώδικα,κλπ. Κατά πάγια πρακτική, το κυρίως πρόγραμμα πρέπει να έχει αρχικοποιήσεις και να καταλήγει σε ένα ατέρμονα βρόχο.

Περιγραφή του Εργαστηρίου

Η λειτουργία των 7-segment LED display μας είναι γνωστή από την Προχωρημένη Λογική Σχεδίαση, μάλιστα κάποιοι/κάποιες από εσάς έχετε υλοποιήσει σε εργαστήρια τέτοια κυκλώματα σε VHDL. Στο παρόν εργαστήριο θα δούμε το ίδιο θέμα από την οπτική της υλοποίησης σε γλώσσα Assembly του AVR. Υπάρχουν πολλά παραδείγματα τέτοιων κωδίκων στο Internet αλλά στο εργαστήριο αυτό ζητάμε να γράψετε τον δικό σας κώδικα ώστε να μάθετε από την διαδικασία, καθότι στο μέλλον θα πρέπει να τον αλλάξετε/προσαρμόσετε σε ένα μεγαλύτερο κομμάτι κώδικα. Ξαναθυμίζουμε το κύκλωμα των 7-Segment LED, με ένα διάγραμμα από την ύλη της Προχωρημένης Λογικής Σχεδίασης:



Το παραπάνω διάγραμμα είναι για τέσσερα ψηφία, κοινής ανόδου (AN3..AN0), με την τυποποιημένη ονομασία των ακροδεκτών A-G, DP. Για το εργαστήριο αυτό μην σας απασχολεί η υποδιαστολή DP και μην σας απασχολεί οποιαδήποτε πλήρης αλφανουμερική ή δεκαεξαδική αναπαράσταση, τα δεδομένα ανά 7-segment LED θα είναι ένας αριθμός BCD, δηλαδή στην δεκαδική περιοχή 0-9. Οι οθόνες θα είναι κοινής ανόδου, όπως φαίνεται και στο διάγραμμα. Τα δικά μας ψηφία όμως θα είναι οκτώ και όχι τέσσερα.

Εκτέλεση του Εργαστηρίου

Παρότι σε μελλοντικά εργαστήρια μπορεί να αλλάξουμε τους ακροδέκτες και τις θύρες που χρησιμοποιούμε, για το εργαστήριο αυτό θα χρησιμοποιήσουμε την θύρα A του AVR (PA0

– PA7) για τα 7 τμήματα κάθε αριθμού, (PA0 = A, PA1 = B, … PA6 = G, PA7 = DP), και την θύρα C για την ενεργοποίηση του ψηφίου που βγαίνει στην οθόνη (PC0 = Least Significant Digit, …, PC7 = Most Significant Digit). Για πρακτικούς λόγους θα κρατήσουμε την εξής σύμβαση, με την κατανόηση ότι αν είχαμε μία πραγματική εφαρμογή κάποιες επιλογές μπορεί να ήταν λίγο διαφορετικές: επειδή τα 7-segment LED είναι CA, αναμμένο LED σημαίνει 0 στην αντίστοιχη έξοδο, επομένως, για παράδειγμα, για να βγάλουμε στην έξοδο τον αριθμό 1_{10} (εκφρασμένο σαν BCD, δηλαδή με περιεχόμενα μνήμης 0x01), τα περιεχόμενα της θύρας A θα είναι 0xF9 (μηδενικά στο PA1, PA2 και άσσοι τα υπόλοιπα). Η θύρα C ουσιαστικά βγάζει στην έξοδο ένα μετρητή δακτυλίου, με άσσο στο ψηφίο που θέλουμε να δείξουμε, αν π.χ. στο παραπάνω παράδειγμα θέλουμε να δείξουμε το δεκαδικό 1 στην έκτη θέση από τις οκτώ, τότε στην θύρα C έχουμε 0x20 ώστε μόνο το PC5 να είναι άσσος). Πολλοί/πολλές από εσάς μπορεί να παρατηρήσετε ότι αν χρειαζόμασταν εξωτερικά τρανζίστορ για να δίνουμε ρεύμα σε ολόκληρο το 7-segment LED θα χρησιμοποιούσαμε τρανζίστορ PNP και σε αυτή την περίπτωση θα αντιστοιχούσε σε 0 έξοδο η οθόνη που θέλουμε να ανάψουμε, αλλά αυτό είναι μία λεπτομέρεια που αλλάζει πάρα πολύ εύκολα, και δεν αλλάζει καθόλου την πολυπλοκότητα ή το μέγεθος του κώδικά μας.

Μέχρι τώρα έχουμε δει ότι υλοποιούμε για την πολυπλεξία στον χρόνο ένα μετρητή δακτυλίου στην θύρα C και έναν αποκωδικοποιητή (BCD-to-7-Segment) στην θύρα A. Μας λείπουν μόνο οι πληροφορίες για το που βρίσκονται τα δεδομένα μας, καθώς και πόσο συχνά χρειάζεται να κάνουμε την πολυπλεξία στον χρόνο,και πως.

Τα δεδομένα προς εμφάνιση αλλά και τα δεδομένα του ποια LED ανάβουμε θα βρίσκονται στην κυρίως μνήμη (RAM). Διαλέξετε και στις δύο περιπτώσεις μία βολική αρχική θέση (π.χ. με τα τέσσερα LSB να είναι 0) και αρχικοποιήσετε την. Παρότι για το εργαστήριο αυτό δεν βαθμολογείστε στο να μπορείτε να αλλάζετε δεδομένα μνήμης, κλπ. (επομένως αν σας βολεύει η flash ή η EPROM για το εργαστήριο αυτό είναι OK), πολύ σύντομα θα μας χρειαστεί η RAM, οπότε flash για το ποιο segment ανάβει ανά αριθμό και κανονική RAM για τα περιεχόμενα είναι μία καλή ιδέα.

Η ελάχιστη συχνότητα με την οποία αλλάζετε την οθόνη είναι 30 φορές το δευτερόλεπτο, και επειδή δεν κάνετε πολυπλεξία σε επίπεδο segment αλλά ολόκληρου 7-segment LED, έχετε 8 ψηφία επί 30 ανανεώσεις ανά δευτερόλεπτο = 240Hz. Πρακτικά, το 1msec από το προηγούμενο εργαστήριο μπορεί να χρησιμοποιηθεί αυτούσιο, ή να αλλάξετε λίγο την περίοδο του TIMER, ότι πάντως και να κάνετε πρέπει να το περιγράψετε

στην αναφορά.

Παρουσίαση /Εξέταση

Επειδή το μάθημα γίνεται χωρίς βοηθούς, ισχύουν για την εξέταση όσα ίσχυαν και στο Εργαστήριο 1, δηλαδή ανεβάζετε στο Webcourses ολόκληρο το Atmel Studio Project με όλους τους κώδικες πηγής και την αναφορά και ότι άλλο είναι χρήσιμο (π.χ. αρχεία εισόδου) σε ένα zip/gzip. Επί αυτού θα εξεταστείτε κάποια στιγμή.

ΣΗΜΑΝΤΙΚΗ ΠΑΡΑΤΗΡΗΣΗ: Ο τρόπος που δουλεύουν τέτοιες οθόνες είναι όσον αφορά την δομή ακριβώς αυτός που κάνουμε, δηλαδή έχουμε (σε Assembly) κάποιο κώδικα (στην δική μας περίπτωση έναν ατέρμονα βρόχο) για κυρίως πρόγραμμα, κάποια ορισμένη διεπαφή (π.χ. στην μνήμη) για το που αποθηκεύουμε τους αριθμούς που θέλουμε να δείξουμε, και το ισοδύναμο του display driver είναι κ κώδικας του εργαστηρίου. Έτσι, αν κάποιος θέλει να αλλάξει κάποιο ψηφίο, απλά γράφει την νέα τιμή στην μνήμη και η οθόνη αυτόματα δείχνει την νέα τιμή. Για τους προφανείς λόγους η διεπαφή έχει νόημα να είναι σε κωδικοποίηση BCD ώστε να μην χρειάζεται μετατροπή πολλές φορές στον τρόπο που θα απεικονίσουμε τελικά.

ΒΟΗΘΗΜΑ 1: Για το εργαστήριο αυτό δεν ανησυχούμε για τυχόν ψηφία που κανονικά δεν θα δείχναμε, επομένως πρέπει και τα οκτώ ψηφία να έχουν κάποια τιμή 0_{10}-9_{10} (χωρίς λάθη), και δείχνουμε αυτούσια την τιμή αυτή και στα οκτώ ψηφία. Επίσης δεν ανησυχούμε για το ότι σε ένα Byte αποθήκευσης θα μπορούσαμε να έχουμε δύο ψηφία BCD – ένα Byte ανά δεκαδικό ψηφίο είναι αποδεκτό.

ΒΟΗΘΗΜΑ 2: Ρώτησαν πολλοί/πολλές στο μάθημα πως μπορούμε να αποφύγουμε το να έχουμε μεγάλα δέντρα με διακλαδώσεις. Η απάντηση είναι ότι εφόσον μπορούμε να εγγυηθούμε πως οι αριθμοί που απεικονίζουμε είναι σωστοί, και διαλέξουμε προσεκτικά το να αποθηκεύσουμε τα περιεχόμενα της επιθυμητής εξόδου για το 0 σε διεύθυνση που καταλήγει (τέσσερα LSB) σε 0x0, τότε η θέση μνήμης με τα περιεχόμενα του 7-segment LED για κάθε ψηφίο είναι η αρχική, συν τον αριθμό που θέλουμε να αναπαραστήσουμε.

ΒΟΗΘΗΜΑ 3: Επίτηδες δεν έχουμε περιγράψει το πως ξέρουμε π.χ. ότι δείξαμε το 2° ψηφίο ώστε όταν ξυπνήσει ο TIMER να δείξουμε το τρίτο. Πρακτικά αυτό σημαίνει ότι μπορούν να δεσμευθούν κάποιοι καταχωρητές για τώρα, αργότερα θα έχουμε μία λύση περισσότερο ρεαλιστική, όπως π.χ. να υπάρχει συγκεκριμένη θέση στην μνήμη που κρατάει αυτή την πληροφορία. Δείτε επίσης τι πληροφορία μπορείτε να πάρετε από τον ίδιο τον PORT C DATA REGISTER – μας δίνει μία καλή

λύση;

ΒΟΗΘΗΜΑ 4: Είναι σημαντικό να έχουμε καλές πρακτικές στους κώδικές μας. Αυτό σημαίνει ότι όταν γράψετε κώδικα και είναι να αλλάξετε ψηφίο, δεν μπορείτε ούτε να αλλάξετε τα περιεχόμενα του Port C και κατόπιν τα περιεχόμενα του Port A στην νέα τιμή (γιατί θα δείξουμε λίγο από τα παλιά περιεχόμενα του Port A πριν ανανεωθεί η οθόνη), ούτε να αλλάξετε τα δεδομένα του Port A στα σωστά δεδομένα πρώτα και μετά αυτά του Port C γιατί θα δείξετε για λίγο τον νέο αριθμό στο Port C. Η «σωστή» προσέγγιση είναι να γράψετε 0xFF στον Port A ώστε να μην δείχνει τίποτα, να αλλάξετε τον Port C, και κατόπιν να γράψετε στον Port A την νέα (σωστή) τιμή.

ΠΡΟΣΟΧΗ (τα ξέρετε, αλλά τα ξαναθυμίζουμε)!

1) Η προεργασία να είναι σε ηλεκτρονική μορφή και μαζί με αρχεία με κώδικες που να μπορούμε να εκτελέσουμε. Το αρχείο πρέπει να το υποβάλλετε στο Webcourses.

2) Η έλλειψη προετοιμασίας ή επαρκούς τεκμηρίωσης οδηγεί σε απόρριψη.

3)Η διαπίστωση αντιγραφής σε οποιοδήποτε σκέλος της άσκησης οδηγεί στην απόρριψη όλων των εμπλεκομένων από το σύνολο των εργαστηριακών ασκήσεων, άρα και του μαθήματος. Αυτό γίνεται οποιαδήποτε στιγμή στη διάρκεια του εξαμήνου. Ως αντιγραφή νοείται και μέρος της αναφοράς, π.χ. σχήματα.

ΚΑΛΗ ΕΠΙΤΥΧΙΑ! ☺