Εργαστήριο 3

Ονομα: Μαυρογιώργης Δημήτρης
ΑΜ: 2016030016
ΗΡΥ 411 - Ενσωματωμένα Συστήματα Μικροεπεξεργατών
Πολυτεχνειο Κρητης

April 19, 2021

Σκοπός

Σκοπός του τρίτου εργαστηρίου είναι η εξοικείωσή μας με τη σειριακή θύρα RS-232. Στο συγκεκριμένο εργαστήριο έχουμε δύο ανεξάρτητες διεργασίες που θα χρησιμοποιούν διαφορετικά Interrupts η κάθε μία. Η πρώτη διεργασία είναι αυτή που θα κάνει την ανανέωση των δεδομένων που εμφανίζουμε στα 7-segment LED, ενώ η δεύτερη θα επεξεργάζεται τα δεδομένα που λαμβάνονται και αποστέλονται από και προς τη σειριακή θύρα RS-232.

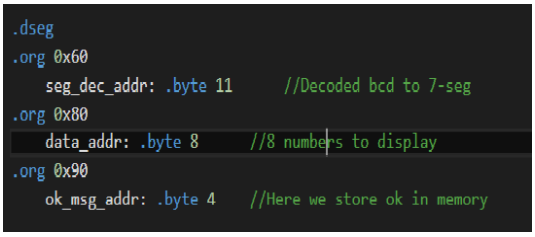
Περιγραφή της υλοποίησης

Στο συγκεκριμενο εργαστήριο εκτος απο τις θύρες A και C του προηγουμενου εργαστηριου χρησιμοποιήθηκαν και οι RXD, TXD για receive και transmit μέσω της σειριακής θύρα αντιστοιχα. Αρχικά, όπως λέει στο εγχειρίδιο του ATmega16, θα πρέπει να διασφαλίσουμε ότι τα Interrupts είναι απενεργοποιημένα και μόλις τελειώσει η αρχικοποίηση να ενεργοποιηθούν. Επιπλέον, για τη λειτουργία της σειρακής θύρας έπρεπε να ενεργοποιήσουμε το receive και transmit κάνοντας "1" τα bit RXEN και TXEN του καταχωρητή UCSRB. Για την ενεργοποίηση του rceive interrupt αρχικοποιήθηκε σε "1" και το bit RXCIE.

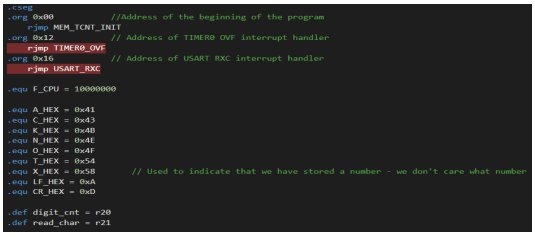
Κατόπιν αρχικοποιήθηκε ο καταχωρητής UBRRL με την τιμή "0x41" (65_{10}) που είναι η τιμή BAUD για baud rate 9600 και συχνότητα 10 MHz. Τέλος, αφού έγινε "1" το bit URSEL του καταχωρητή UCSRC, για να μπορέσουμε να αρχικοποιήσουμε το συγκεκριμένο καταχωρητή και όχι το UBRR, τέθηκαν σε "1" τα bit UCSZ0 και UCSZ1 για να δηλώσουμε ότι έχουμε 8-bit κατά τη μετάδοση και λήψη. Τα bit UPM0 και UPM1 έμειναν "0", καθώς δεν έχουμε parity mode, ενώ το bit USBS έμεινε "0", επειδή έχουμε 1 stop bit.

Επεξήγηση Κώδικα
Αρχικά, στο data segment δεσμευουμε χωρο 4 byte για το OK<CR><LF>. Έπειτα δηλώθηκαν καποιες σταθερές που είναι οι HEX τιμές των γραμμάτων εντολών A, T, C, N, O, K και ο χαρακτήρας X που χρησιμοποιήθηκε για να ξέρουμε κάθε φορά ότι ήρθε κάποιος αριθμός απο τη θύρα. Παράλληλα, δημιουργήθηκε μια ρουτίνα για clear της μνήμης, με την οποία αποθηκεύουμε την τιμή "0x0A" που σημαίνει ότι τα LED είναι σβηστά. Επιπλέον, επειδή κατα την ανανέωση των 7-segment διβάζουμε από τη διεύθυνση "0x80" προς την "0x87" και η ανανέωση γίνεται από το LS Digit προς το MS Digit, χρειάζεται μια ρουτίνα για να κάνει shift τα δεδομένα, ώστε να εμφανίζονται σωστά.

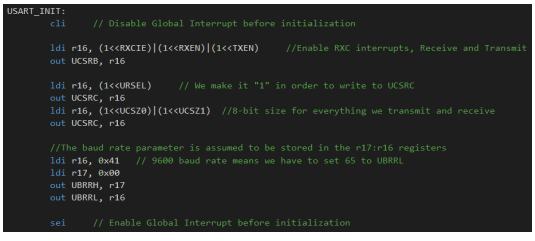
Όσον αφορά το handler για τη σειριακή θύρα και το receive των δεδομένων, η διεύθυνση μνήμης που βρίσκεται είναι η "0x16". Η υλοποίησή του είναι πολύ απλή, καθώς το μόνο που χρειάζεται να γίνει είναι να διαβάσουμε την τιμή του UDR σε ένα καταχωρητή και να ελέγχουμε με τον καταχωρητή r21, που έγινε define ως read_char, ότι διαβάζονται σωστά οι εντολές και ανάλογα με την εντολή να κάνουμε την κατάλληλη ενέργεια και να στείλουμε OK<CR><LF>, μόλις λάβουμε και το <LF> της εντολής που εκτελέστηκε.



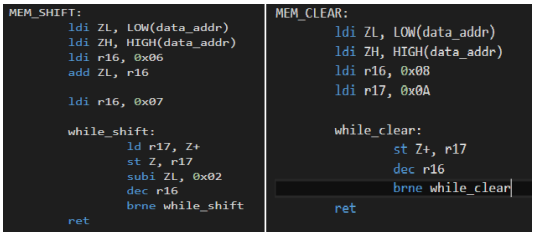
(a) Data segment



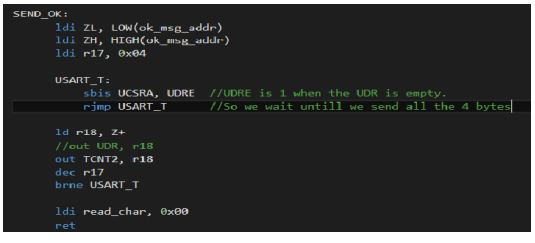
(b) Code segment - Definitions



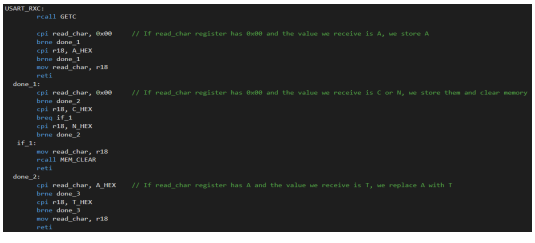
(c) Usart Initialization



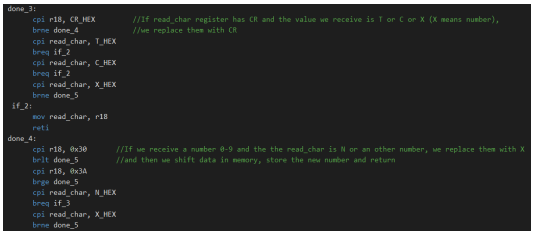
(d) Memory clear and data shift fucntion



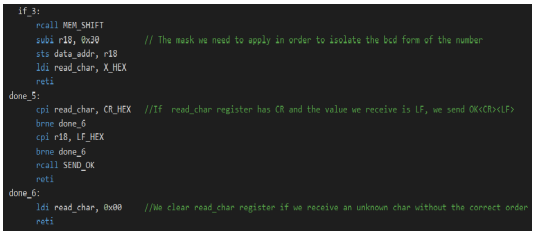
(e) Function for sending OK



(f) Usart Interrupt Handler - A



(g) Usart Interrupt Handler - B

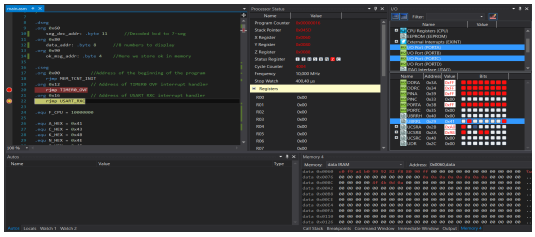


(h) Usart Interrupt Handler - C

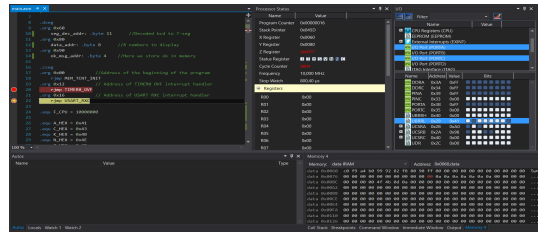
Προσομοίωση Αποτελεσμάτων

Όπως φαίνεται και στην εικόνα (a) όλες οι αρχικοποιήσεις που περιγράφτηκαν παραπάνω γίνονται σωστά και για το USART και για τον TIMER/COUNTER0. Επιπλέον, Βλέπουμε ότι οι αποκωδικοποιήσεις των BCD, οι αριθμοί προς εμφάνιση και το OK<CR><LF> αποθηκεύονται στην SRAM. Στην επόμενη εικόνα παρατηρούμε ότι εκτελείται ο USART_RXC handler και αποθηκεύεται ο αριθμός 0 που λαμβάνουμε μέσω του USART. Στην εικόνα (c) βλέπουμε ότι κάποια στιγμή εκτελείται ο TIMER0_OVF handler και εμφανίζεται η τιμή 2 στο πρώτο digit. Έπειτα μετά από αρκετές εκτελέσεις του USART_RXC handler, βλέπουμε ότι και οι οχτώ τιμές που λήφθηκαν μέσω του USART αποθηκεύονται σωστά στη μνήμη (εικόνα (e)).

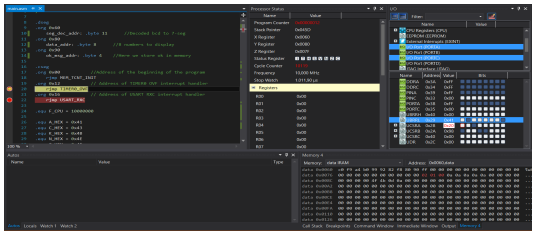
Στην ίδια εικόνα βλέπουμε ότι έρχεται για δεύτερη φορά ο TIMER0_OVF handler και στην επόμενη εικόνα βλέπουμε ότι εμφανίζεται το 6 που αποθηκεύσαμε στη δεύτερη θέση μνήμης στο digit 2. Μόλις τελειώσει η λήψη και του <LF>, εκτελείται μόνο ο TIMER0_OVF και γίνεται σωστά η ανανέωση των ψηφίων, όπως φαίνεται και στις επόμενες εικόνες. Στις εικόνες (g) και (h) φαίνεται ότι μετα την εμφάνιση του 6, εμφανίζεται το 5 στο ψηφίο 3 και στην επόμενη εκτέλεση εμφανίζεται το 4 στο digit 4 που είναι αποθηκευμένα στην τρίτη και τέταρτη θέση μνήμης αντίστοιχα. Συνεπώς, βλέπουμε ότι η λήψη και εμφάνιση των αριθμών γίνεται σωστά. Τέλος, όσον αφορά το OK<CR><LF> που στέλνουμε μετά από κάθε εντολή, φαίνεται ότι αποστέλονται σωστά τα byte OK<CR><LF> στο αρχείο lab3.log, αφού γίνεται η εγγραφή τους σε HEX μορφή σε αυτό το αρχείο.



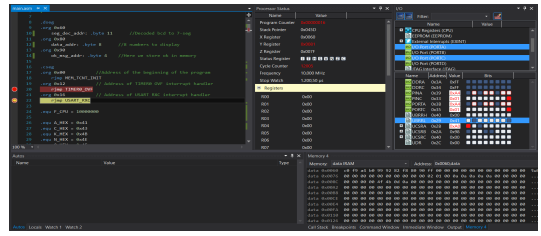
(a) Results from Atmel Studio 7 - Initialization of registers and memory



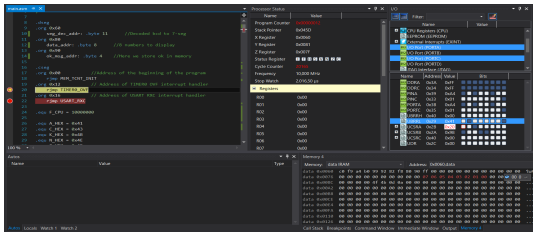
(b) Results from Atmel Studio 7 - Store of the first number



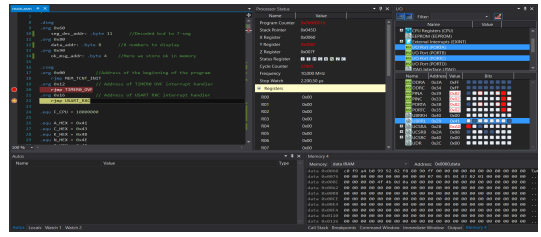
(c) Results from Atmel Studio 7 - Three number written and excecution of TIMER0 OVF handler



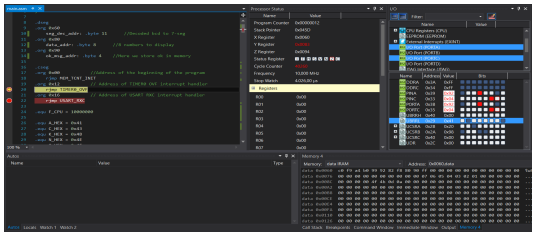
(d) Results from Atmel Studio 7 - After the execution of TIMER0 OVF handler



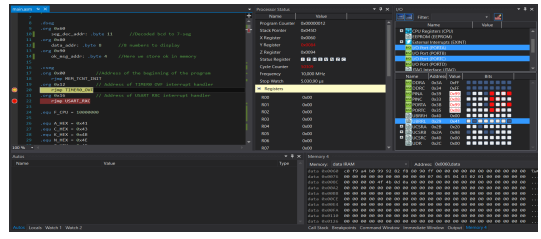
(e) Results from Atmel Studio 7 - All numbers stored and TIMER0 OVF handler is executed once again



(f) Results from Atmel Studio 7 - After the second execution of TIMER0 OVF handler



(g) Results from Atmel Studio 7 - After the third execution of TIMER0 OVF handler



(h) Results from Atmel Studio 7 - After the forth execution of TIMER0 OVF handler