Ενσωματωμένα Συστήματα Μικροεπεξεργαστών – ΗΡΥ411

Αναφορά Εργαστηρίου 8 – Υπορουτίνες, και ένας πολύ απλός δρομολογητής

Ομάδα: LAB41145877
Κιούλος Ευάγγελος 2016030056

Εισαγωγή:

Σκοπός της όγδοης εργαστηριακής άσκησης ήταν η δημιουργία τριών απλών υπορουτινών και ενός απλού δρομολογητή. Το time slice για την εναλλαγή διεργασιών είναι 100msec. Για το εργαστήριο χρησιμοποιήθηκε ο κώδικας του εργαστηρίου 5 χωρίς τον οδηγό για το seven segment display.

Υλοποίηση Εργαστηρίου:

Αρχικά, υλοποιήθηκαν 3 απλές υπορουτίνες. Η υπορουτίνα “void process_1(void)” υλοποιεί έναν απλό μετρητή BCD ο οποίος μετράει από την τιμή “0000” μέχρι την τιμή “1111”. Μόλις φτάσει στην τιμή “1111”, πηγαίνει στην τιμή “0000” και ξεκινά να μετράει από την αρχή. Η τιμή της εξόδου της ρουτίνας μετά από κάθε κλήση αποθηκεύεται στην διεύθυνση “0x0080” της μνήμης RAM. Η υπορουτίνα “void process_2(void)” εναλλάσσει την τιμή “10101010” με “01010101”. Η τιμή της εξόδου της ρουτίνας μετά από κάθε κλήση αποθηκεύεται στην διεύθυνση “0x0081” της μνήμης RAM. Η υπορουτίνα “void process_3(void)” υλοποιεί έναν ring counter ο οποίος ξεκινά από την τιμή “0x01”. Η τιμή της εξόδου της ρουτίνας μετά από κάθε κλήση αποθηκεύεται στην διεύθυνση “0x0082” της μνήμης RAM. Και οι τρείς ρουτίνες βγάζουν την έξοδο τους στο PORTA. Κάθε ρουτίνα κάνει μόνο μία αλλαγή στην έξοδο κάθε φορά που καλείται.

Στην συνέχεια, χρησιμοποιώντας την λειτουργία της σειριακής θύρας από το εργαστήριο 5, προστέθηκε η λειτουργία για εκτέλεση των εντολών “Sx<CR><LF>” για ενεργοποίηση μίας διεργασίας και “Qx<CR><LF>” για απενεργοποίηση μίας διεργασίας, με το “x” να είναι ‘1’, ‘2’ ή ‘3’. Για την ενεργοποίηση μίας διεργασίας εισάγουμε την τιμή “0x01” σε μία διεύθυνση μνήμης(ξεχωριστή διεύθυνση για κάθε διεργασία) και αντίστοιχα για την απενεργοποίηση της εισάγουμε την τιμή “0x00”. Για την υπορουτίνα 1 χρησιμοποιούμε την διεύθυνση “0x006B”, για την υπορουτίνα 2 την διεύθυνση “0x006C” και για την υπορουτίνα 3 την διεύθυνση “0x006D”. Επιπλέων, χρησιμοποιείται και η διεύθυνση “0x006E” η τιμή της οποίας δείχνει ποια διεργασία εκτελείται κάθε στιγμή.

ΣΗΜΕΙΩΣΗ: Η παραπάνω λύση, παρόλο που είναι απλή από άποψη κώδικα είναι αρκετά σπάταλη καθώς χρησιμοποιούμε 4 bytes της μνήμης. Αντ’ αυτού, θα μπορούσαμε για παράδειγμα να χρησιμοποιήσουμε μόνο 1 byte, στο οποίο τα τέσσερα πιο σημαντικά bit θα μας έδειχναν ποια διεργασία εκτελείται και τα τέσσερα λιγότερο σημαντικά bit ποιες διεργασίες είναι ενεργοποιημένες.

Για την υλοποίηση του time slice, χρησιμοποιήθηκε ο 16bit timer/counter1 σε CTC mode, με prescaler $f_{clk}/64$ και με τον καταχωρητή OCR1A να έχει την τιμή “15625”. Η τιμή του OCR1A υπολογίστηκε μέσο του τύπου $f_{OCnA} = \frac{f_{clk}}{2N(1+OCRnA)}$(σελίδα 102 ATmega16 manual). Επιπλέων, ενεργοποιήθηκε το Compare Match A interrupt του timer/counter1. Στον interrupt handler του timer1, διαβάζουμε την τιμή της διεύθυνσης “0x006E” και με την χρήση ενός

switch statement, ελέγχουμε ποια διεργασία εκτελείται σε αυτό το time slice και μεταβαίνουμε στην επόμενη ενεργοποιημένη διεργασία, ελέγχοντας ποιες διεργασίες είναι ενεργές, διαβάζοντας τις τιμές των διευθύνσεων “0x006B”,“0x006C” και “0x006D”. Για την μετάβαση στην επόμενη διεργασία, αλλάζουμε την τιμή της διεύθυνσης “0x006E”. Για παράδειγμα, αν οι διεργασίες ‘1’ και ‘3’ είναι ενεργοποιημένες και η διεργασία που εκτελείται σε αυτό το time slice είναι η ‘1’, τότε η διεύθυνση “0x006E” θα πάρει την τιμή “0x03” και θα επιστρέψουμε από το interrupt. Οι διεργασίες αλλάζουν κυκλικά. Αν καμία διεργασία δεν είναι ενεργοποιημένη, τότε η διεύθυνση “0x006E” παίρνει την τιμή “0x00”.

Τέλος, στην main συνάρτηση του προγράμματος, χρησιμοποιούμε ένα ατέρμονο while loop, μέσα στο οποίο διαβάζουμε την τιμή της διεύθυνσης “0x006E” και με την χρήση ενός switch statement, εκτελούμε την αντίστοιχη διεργασία. Αν η τιμή της διεύθυνσης “0x006E” είναι ‘0’ τότε δεν εκτελούμε τίποτα και το πρόγραμμα τρέχει έναν ατέρμονο βρόγχο για τα επόμενα time slices μέχρι να ενεργοποιηθεί κάποια διεργασία. Οι υπορουτίνες, εκτελούνται συνεχόμενα μέχρι να τελειώσει το time slice τους.

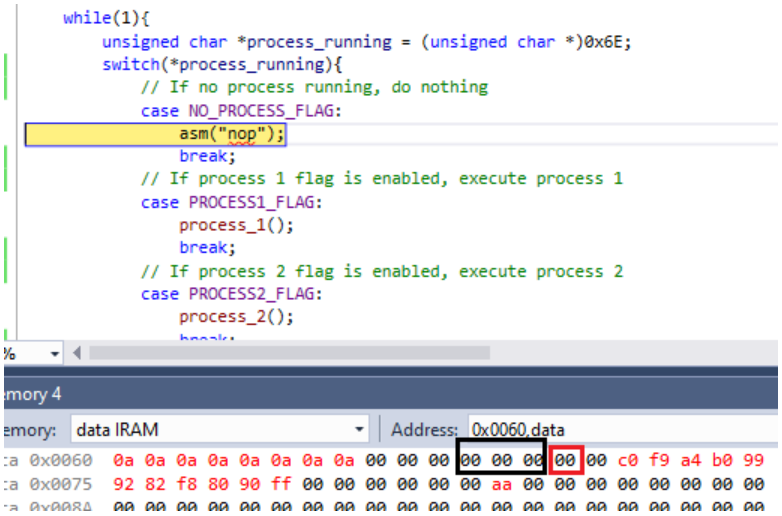
Αποτελέσματα Προσομοίωσης:

Για την προσομοίωση θα εκτελέσουμε το σενάριο:

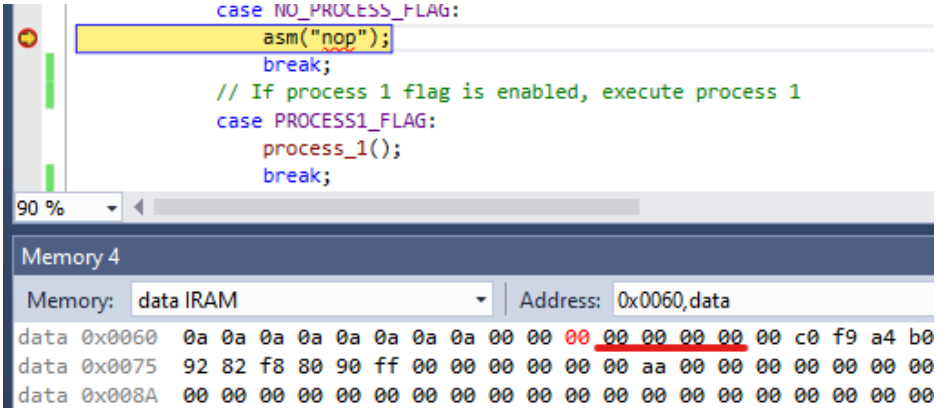
- Καμία ενεργή διεργασία αρχικά.
- Απενεργοποίηση της διεργασίας ‘1’.
- Ενεργοποίηση της διεργασίας ‘1’.
- Εκτέλεση της διεργασίας ‘1’.
- Ενεργοποίηση της διεργασίας ‘3’.
- Εκτέλεση των διεργασιών ‘1’ και ‘3’.
- Ενεργοποίηση της διεργασίας ‘2’.
- Εκτέλεση και των τριών διεργασιών.
- Απενεργοποίηση και των τριών διεργασιών.

Για την ενεργοποίηση και την απενεργοποίηση των διεργασιών χρησιμοποιήθηκαν τα αρχεία “.stim” που υπάρχουν στον παραδοτέο φάκελο.

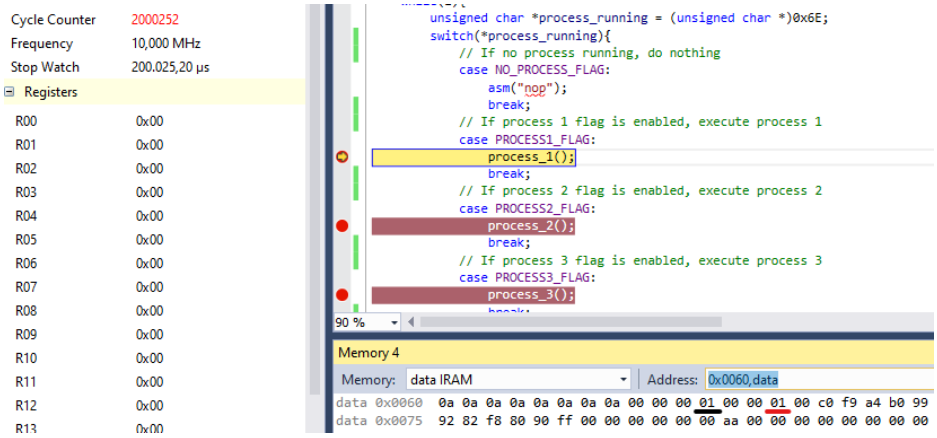
Αρχικά, δεν έχουμε καμία ενεργοποιημένη διεργασία όπως φαίνεται στο μαύρο πλαίσιο της παρακάτω εικόνας που δείχνει τις τιμές των διευθύνσεων “0x06C-0x06D” και στο κόκκινο πλαίσιο βλέπουμε ότι η τιμή της διεργασίας που εκτελείται τώρα είναι ‘0’(διεύθυνση “0x006E”), επομένως, το πρόγραμμα τρέχει έναν ατέρμονο βρόχο.



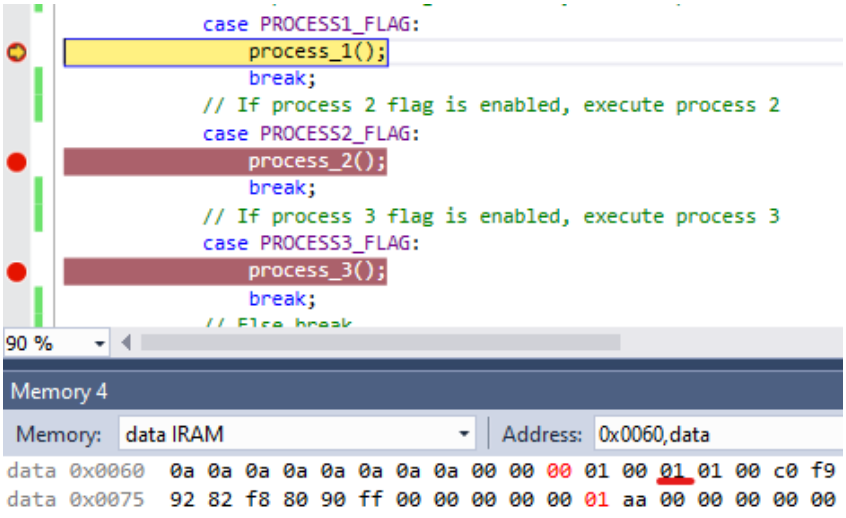
Στην συνέχεια τρέχουμε το αρχείο “Q_1_CR_LF.stim” και παρατηρούμε ότι απενεργοποιώντας μια διεργασία που δεν είναι ενεργοποιημένη, δεν θα αλλάξει κάτι στην μνήμη μας(οι τιμές με την κόκκινη υπογράμμιση παραμένουν ίδιες).



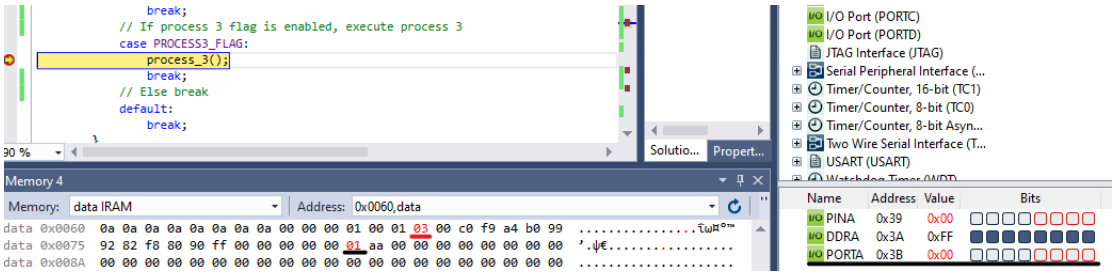
Στο επόμενο time slice(δηλαδή μετά από 100msec) ενεργοποιούμε τη διεργασία ‘1’ εκτελώντας το αρχείο “S_1_CR_LF.stim”. Παρατηρούμε ότι στην μνήμη η διεύθυνση “0x006E” (με την μαύρη υπογράμμιση στην παρακάτω εικόνα) έχει την τιμή ‘1’ που δείχνει ότι η διεργασία 1 ενεργοποιήθηκε και μετά από 100msec(δηλαδή μετά από ένα time slice) η τιμή της διεύθυνσης “0x006E”(κόκκινη υπογράμμιση) έχει την τιμή ‘1’.



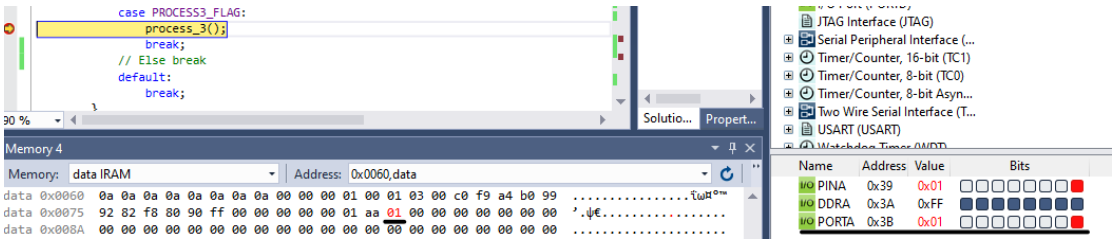
Στην συνέχεια ενεργοποιούμε τη διεργασία ‘3’ εκτελώντας το αρχείο “S_3_CR_LF.stim”. Όπως βλέπουμε παρακάτω, το flag ενεργοποίησης της διεργασίας ‘3’ έχει την τιμή ‘1’.



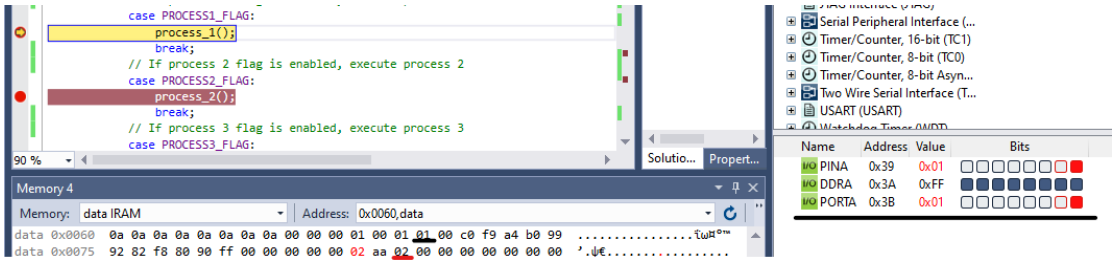
Συνεχίζοντας την εκτέλεση του προγράμματος μέχρι να τελειώσει το time slice. Παρατηρούμε ότι μετά το interrupt, όταν επιστρέψουμε στην main, το πρόγραμμα αρχίζει την εκτέλεση της διεργασίας 3. Η τιμή του flag για την διεργασία που εκτελείται τώρα έχει την τιμή ‘3’ όπως φαίνεται και παρακάτω(κόκκινη υπογράμμιση).



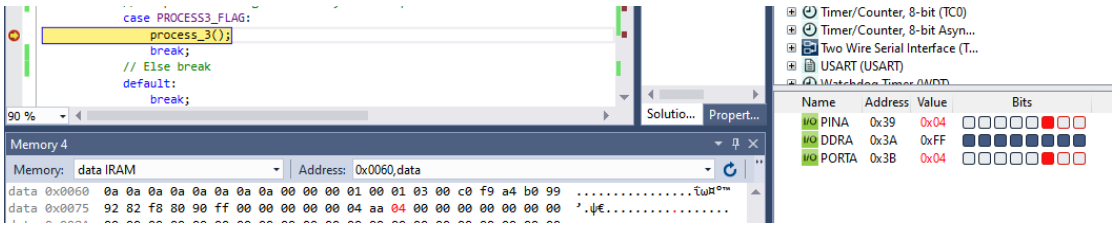
Στην παραπάνω εικόνα, δεν έχουμε εκτελέσει ακόμα την διεργασία 3 και παρατηρούμε ότι στο port A υπάρχει η τελευταία έξοδος της υπορουτίνας ‘1’ που είναι το “0x00” και στην μνήμη υπάρχει η επόμενη τιμή της δηλαδή η τιμή “0x01”(μαύρη υπογράμμιση). Εκτελώντας την υπορουτίνα ‘3’ παρατηρούμε ότι στο port A υπάρχει η έξοδος της υπορουτίνας ‘3’και η αντίστοιχη τιμή έχει αποθηκευτεί στην μνήμη(μαύρη υπογράμμιση).



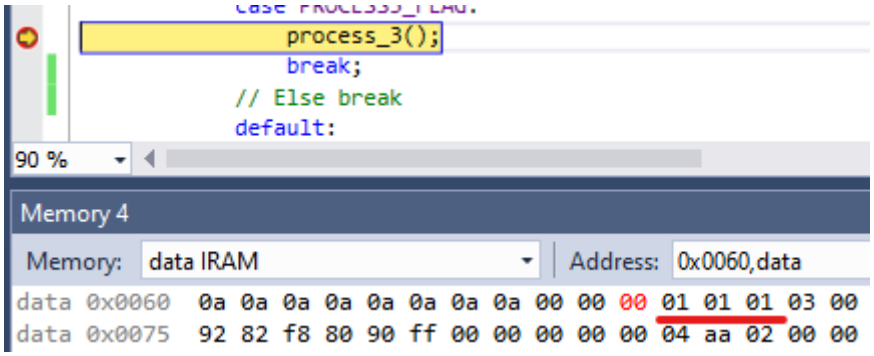
Συνεχίζοντας την εκτέλεση του προγράμματος μέχρι το τέλος του time slice παρατηρούμε ότι το πρόγραμμα μεταβαίνει στην διεργασία ‘1’ και στο port A υπάρχει η τελευταία τιμή που υπήρχε στην μνήμη, δηλαδή η τιμή 1.



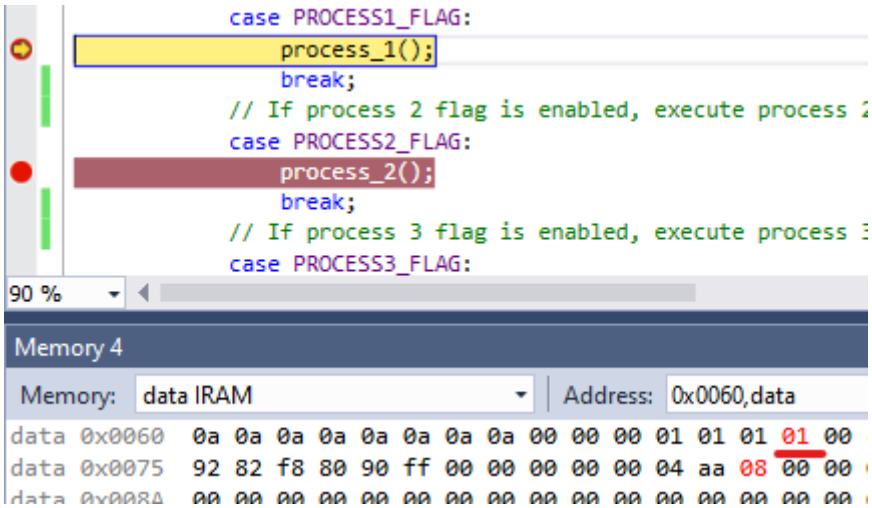
Στην παραπάνω εικόνα η κόκκινη υπογράμμιση δείχνει την τελευταία τιμή της υπορουτίνας ‘3’ η οποία είναι η τιμή “0x02”. Να υπενθυμίσουμε ότι υπορουτίνα ‘3’ υλοποιεί έναν ring counter, επομένως, όταν έρθει η υπορουτίνα ‘3’ ξανά, η τιμή στο port A θα πρέπει να είναι η τιμή “0x04”. Συνεχίζοντας την εκτέλεση του προγράμματος, μεταβαίνουμε στην διεργασία ‘3’ μετά το τέλος του time slice, όπως φαίνεται και στην παρακάτω εικόνα.



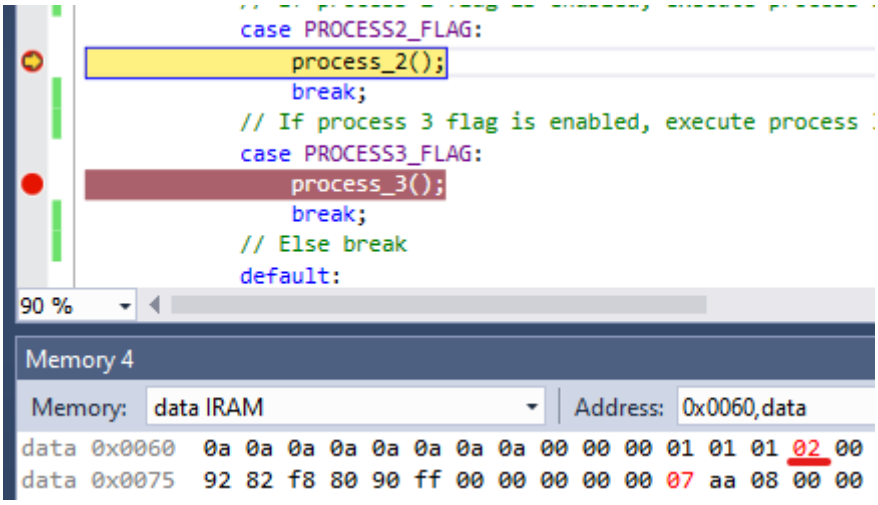
Παρατηρούμε ότι, την πρώτη φορά που θα τρέξουμε την υπορουτίνα ‘3’ παρατηρούμε ότι η έξοδος της στο port A θα είναι πράγματι η τιμή “0x04” όπως προβλέψαμε παραπάνω. Στην συνέχεια, ενώ εκτελούμε την υπορουτίνα ‘3’, ενεργοποιούμε τη διεργασία ‘2’ εκτελώντας το αρχείο “S_2_CR_LF.stim”. Μετά την εκτέλεση του αρχείου “.stim” παρατηρούμε ότι το flag ενεργοποίησης της διεργασίας ‘2’ έχει την τιμή ‘1’. Όπως βλέπουμε και παρακάτω, πλέων και οι τρείς διεργασίες είναι ενεργοποιημένες.



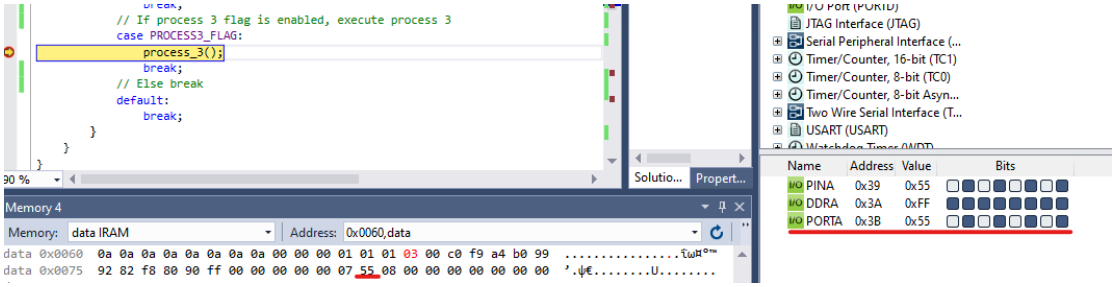
Μόλις τελειώσει το time slice, μεταβαίνουμε στην υπορουτίνα ‘1’ όπως φαίνεται και παρακάτω.



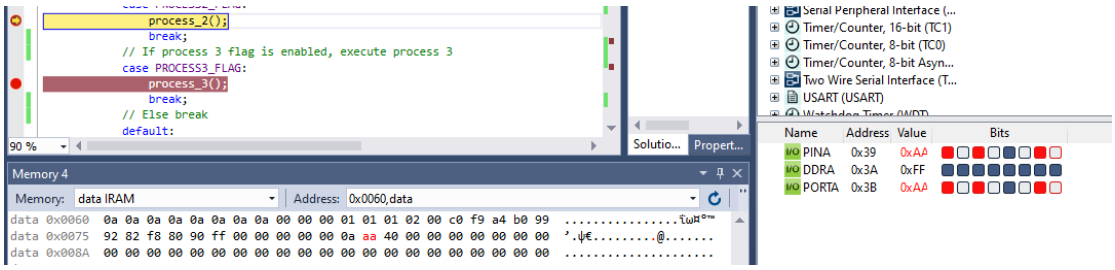
Στο επόμενο time slice, όπως βλέπουμε και παρακάτω, το πρόγραμμα θα μεταβεί στην υπορουτίνα ‘2’.



Στην παραπάνω εικόνα, βλέπουμε ότι το flag που δείχνει την υπορουτίνα του τωρινού time slice έχει την τιμή ‘2’(κόκκινη υπογράμμιση). Συνεχίζοντας την εκτέλεση του προγράμματος, στο επόμενο time slice, το πρόγραμμα μεταβαίνει στην διεργασία ‘3’.



Παρατηρούμε ότι, η τελευταία τιμή της εξόδου της υπορουτίνας ‘2’ είναι η τιμή “01010101” (δηλ. “0x55”). Η υπορουτίνα ‘2’ κάνει την εναλλαγή “01010101” (δηλ. “0x55”) σε “10101010” (δηλ. “0xAA”) και το αντίστροφο. Επομένως, την επόμενη φορά που θα εκτελεστεί η υπορουτίνα ‘2’, η τιμή στη έξοδο της θα είναι η τιμή “10101010”(δηλ. “0xAA”). Πράγματι, όπως βλέπουμε στη παρακάτω εικόνα, όταν έρθει ξανά η υπορουτίνα ‘2’, παρατηρούμε ότι η τιμή του port A μετά την πρώτη εκτέλεση της θα είναι “10101010”(δηλ. “0xAA”).



Στην συνέχεια, απενεργοποιούμε και τις τρείς υπορουτίνες εκτελώντας τα αρχεία “Q_1_CR_LF.stim”, “Q_2_CR_LF.stim” και “Q_3_CR_LF.stim”. Μετά την εκτέλεση των αρχείων, παρατηρούμε ότι τα flags που δείχνουν την κατάσταση μιας διεργασίας(ενεργοποιημένη ή απενεργοποιημένη) έχουν την τιμή ‘0’(κόκκινη υπογράμμιση) και το flag για την διεργασία του τωρινού time slice(μαύρη υπογράμμιση) έχει την τιμή ‘0’, που αντιστοιχεί σε έναν ατέρμονο βρόγχο όπου δεν έχουμε καμία διεργασία. Επιπλέον, στην συγκεκριμένη υλοποίηση, οι τελευταίες τιμές των διεργασιών που έχουν απενεργοποιηθεί, έχουν παραμείνει στην μνήμη(πράσινη υλοποίηση). Τα παραπάνω αποτελέσματα φαίνονται στην παρακάτω εικόνα.