ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΝΣΩΜΑΤΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ

Εργασία:

Αναφορά Πρότζεκτ Sudoku

Φοιτητές:

Βασιλείου Παναγιώτης
2017030067

Ιωάννης-Ιάσων Γεωργακάς
2017030021

December 24, 2021

Contents

1 Εισαγωγή

Σκοπός του Project είναι η επίλυση ενός παιχνιδιού Sudoku (9x9 πλέγμα) χρησιμοποιώντας έναν μικροελεγκτή AVR. Η διεπαφή του μικροελεγκτή με τον εξωτερικό κόσμο υλοποιείται με χρήση της σειριακής θύρας RS232 και με τη χρήση του τερματικού προγραμμάτος putty και του Rust προγράμματος που αναπτύχθηκε για τα πλαίσια του μαθήματος.

2 Τεχνολογία

Για την υλοποίηση του project χρησιμοποιήθηκε η πλακέτα STK500 με τον μικροελεγκτή ATmega16L με συχνότητα ρολογιού 10MHz (εξωτερικός κρύσταλλος). Τα μοντέλα του μικροελεγκτή και του κρυστάλλου τηρούν προδιαγραφές οι οποίες είναι κοινές για όλες τις ομάδες. Η συγγραφή του κώδικα πραγματοποιήθηκε στη γλώσσα C με τον avr-gcc compiler (έκδοση 5.4.0). Για το κατέβασμα του κώδικα στην πλακέτα χρησιμοποιήθηκε η πλατφόρμα Microchip Studio (έκδοση 7.0.2542) με τη χρήση του προγραμματιστή avrdude. Για την επικοινωνία της πλακέτας με τον εξωτερικό κόσμο χρησιμοποιήθηκε ένα καλώδιο USB σε Serial RS232 (βύσμα DB9).

3 Περιγραφή της υλοποίησης

Το πρότζεκτ αποτελείται απο 4 μέρη τα οποία είναι η διεπαφή της σειριακής θύρας, η επεξεργασία των εντολών, ο αλγόριθμος επίλυσης του Sudoku και η οθόνη, η οποία εμφανίζει την πρόοδο επίλυσης του Sudoku. Το ροοδιάγραμμα της main() ρουτίνας παρατίθεται στην Εικόνα 1.

3.1 Σειριακή Θύρα RS232

Η επικοινωνία του μικροελεγκτή με το χρήστη επιτυγχάνεται μέσω της σειριακής θύρας ορίζοντας τις παραμέτρους της έτσι ώστε να τηρούν τις προδιαγραφές:

- BAUD rate 9600 bps
- 8 data bits
- No parity
- 1 stop bit

Για το σετάρισμα του BAUD rate στην τιμή 9600 φορτώθηκε στον 16-bit καταχωρητή UBRR η τιμή $F\_CPU/16/USART\_BAUDRATE - 1 = 10 * 10^6/16/9600 - 1 = 64.10$, το οποίο απλοποιείται στο 64 (ο UBRR δέχεται ακέραιες τιμές). Το BAUD rate, με τιμή UBRR 64 παίρνιε την τιμή 9615, το οποίο έχει απόκλιση 0.16%.

Στη συνέχεια, ενεργοποιείται η λήψη και η μετάδοση δεδομένων μέσω των flags RXEN και TXEN του UCSRB. Το interrupt flag της λήψης, ενεργοποιείται προκειμένου ο μικροελεγκτής να λαμβάνει δεδομένα με χρήση interrupts και να μεταδίδει με χρήση polling. Το ροοδιάγραμμα της ISR USART_RXC_vect παρατίθεται στην Εικόνα 2.

Τέλος, για την ρύθμιση 8 data bits, αρχικά ενεργοποιείται ο UCSRC (ίδιος με τον UBRRH) με το URSEL bit και τίθενται τα UCSZ0 και UCSZ1 στην τιμή 1. Για τη ρύθμιση του parity και του stop bit, δεν απαιτείται κάποια ενέργεια καθώς είναι αυτόματα ρυθμισένα στην επιθυμητή τιμή (0).

Listing 1: Κώδικας αρχικοποίησης UART

```
void initUART()
{
  // Load upper 8-bits of the baud rate value into
  // the high byte of the UBRR register
  UBRRH = (BAUD_PRESCALE)>>8;
  // Load lower 8-bits of the baud rate value into
  // the low byte of the UBRR register
  UBRRL = BAUD_PRESCALE;
   // Enable reception and transmission circuitry
  UCSRB = (1 << RXEN) | (1 << TXEN);
   // Use 8-bit character sizes
  UCSRC = (1 << URSEL) | (1 << UCSZ1) | (1 << UCSZ0);
  // Enable the USART RXC and TXC interrupts
  UCSRB |= (1 << RXCIE); // | (1 << TXCIE);
}
```

3.2 Επεξεργασία των εντολών

Οι εντολές που υποστηρίζει το ενσωματώμενο σύστημα παρατίθενται στο Table 1. Οι εντολές εκτελούνται στο main loop όταν καλείται η συνάρτηση process(). Σε αυτή αρχικά διαβάζεται ο receive buffer και αποκωδικοποείται μία νέα εντολή, εφόσον υπάρχει. Στη συνέχεια εκτελείται η αντίστοιχη ενέργεια και μεταδίδεται η απάντηση από τον AVR χρησιμοποιώντας τη συνάρτηση transmit() η οποία εξετάζει εάν ο

καταχωρητής UDR είναι άδειος και επομένως έτοιμος να λάβει χαρακτήρα ώστε να τον μεταδόσει στο PC. Το ροοδιάγραμμα της ρουτίνας process() παρατίθεται στην Εικόνα 3.

Εντολή από PC ή AVR	**Ενέργεια εντολής**	**Απάντηση προς PC ή AVR**
"**AT**\r\n" (Εντολή PC)	Έναρξη επικοινωνίας με τον AVR	"**OK**\r\n" (Απάντηση AVR)
"**C**\r\n" (Εντολή PC)	Καθαρισμός πίνακα και οθόνης	"**OK**\r\n" (Απάντηση AVR)
"**N**XYVAL\r\n" (Εντολή PC)	Τιμή κελιού (VAL) sudoku[X][Υ], X,Υ ∈ [1,9]	"**OK**\r\n" (Απάντηση AVR)
"**P**\r\n" (Εντολή PC)	Ξεκινάει να παίζει-λύνει το παιχνίδι	"**OK**\r\n" (Απάντηση AVR)
"**D**\r\n" (Εντολή AVR)	Ενημερώνει το PC ότι έλυσε το Sudoku και μπορεί να ξεκινήσει τη μετάδοση των κελιών του Sudoku	"**S**\r\n" (Απάντηση/Εντολή PC)
"**S**\r\n" (Εντολή PC)	Στέλνει τα δεδομένα του 1ου κελιού του Sudoku	"**N**11VAL\r\n" (Απάντηση AVR)
"**T**\r\n" (Εντολή PC)	Στέλνει τμές κελιών στο PC, μία τιμή κάθε φορά που λαμβάνει αυτή την εντολή	"**N**XYVAL\r\n" (Απάντηση AVR)
"**B**\r\n" (Εντολή PC)	Σταμάτησε τους υπολογισμούς ή τη μετάδοση αποτελεσμάτων (warm start)	"**OK**\r\n" (Απάντηση AVR)
"**D**XY\r\n" (Εντολή PC)	Στέλνει τα περιεχόμενα του κελιού sudoku[X][Y]	"**N**XYVAL\r\n" (Απάντηση AVR)

Table 1: Σετ εντολών

3.3 Αλγόριθμός επίλυσης Sudoku

Για την επίλυση του Sudoku χρησιμοποιήθηκε ένας απλός αναδρομικός backtrack αλγόριθμος. Ο αλγόριθμος αρχικά βρίσκει ένα άδειο κελί και αποθηκεύει την 1η έγκυρη εικασία, ξεκινώντας από το 1 με μέγιστο τιμή το 9. Για να είναι έγκυρη μία εικασία,

θα πρέπει ο συγκεκιρμένος αριθμός να μην βρίσκεται ήδη στην ίδια γραμμή, στήλη και 3x3 τετράγωνο. Εφόσον το Sudoku περιέχει πλέον ένα παραπάνω συμπληρωμενο κελί, γίνεται η αναδρομική κλήση. Εάν, η συνάρτηση επίλυσης (solve) επιστρέψει 0, τότε γίνεται backtrack μηδενίζοντας το συγκεκριμένο κελί. Η solve θα επιστρέψει 0, μόνο όταν κάποιο άδειο κελί δεν μπορεί να πάρει καμία τιμή μεταξύ 1 και 9.
Λόγω του διαγωνισμού, στον οποίο θα πάρει μέρος κάθε ομάδα υλοποιήθηκε και ένας βελτιστοποιημένος αλγόριθμος επίλυσης, ωστόσο ήταν ανάγκαια η χρήση δυναμικά κατανεμημένης μνήμης και το αποτυπώμα της μνήμης ήταν μεγάλο. Τα δύο παραπάνω παρατηρήθηκαν αφού πρώτα έγινε η υλοποίηση του βελτιστοποιημένου αλγορίθμου σε απλή C και εκτελώντας τον σε επεξεργαστή γενικού σκοπού. Ως αποτέλεσμα των παραπάνω όταν εκτελέστηκε ο βελτιστοποιημένος αλγόριθμος στον ATmega16L αρκετά Sudoku δεν λύνονταν. Η ανάλυση του βελτιστοποιημένου κώδικα και η σύγκρισή του με τον απλό αλγόριθμο σε επεξεργαστή γενικού σκοπού βρίσκεται στο Παράρτημα Β. Τα αποτελέσματα της επίλυσης των Sudoku του Παραρτήματος 7 βρίσκονται στο κομμάτι "Έλεγχος σωστής λειτουργίας - Αποτελέσματα".

3.4 Οθόνη - Progress Bar

Για την απεικόνιση της προόδου επίλυσης του Sudoku χρησιμοποιήθηκαν τα LEDs κοινής ανόδου (CA) της πλακέτας. Συγκεκριμένα κατά τη διάρκεια επίλυσης του Sudoku πραγματοποιείται περιοδικά έλεγχος σχετικά με το πόσα κελιά του πίνακα είναι συμπληρωμένα, με βάση αυτόν τον αριθμό ανάβει ο ανάλογος αριθμός LEDs (Πίνακας 2). Το ροοδιάγραμμα της ISR TIMER0_COMP_vect παρατίθεται στην Εικόνα 4.
Για την υλοποίηση της περιοδικής ανανέωσης των LEDs χρησιμοποιήθηκε ο Timer 0 σε λειτουργία compare θέτοντας τον καταχωρητή OCR0 στη τιμή 125 κατά την αρχικοποίηση του Timer 0. Ο prescaler τέθηκε στην τιμή 256 θέτωντας στο καταχωρητή TCCR0 το bit CS02 ίσο με 1. Με βάση τις παραπάνω τιμές αρχικοποιήσης του Timer 0 και τον παρακάτω τύπο υπολογίστηκε ότι η οθόνη έχει ρυθμό ανανεώσης 38 Hz.
$f_{OC0} = \frac{f_{CLK}}{2*N*(1+OCR0)} = \frac{10MHz}{2*1024*126)} = 38.44Hz$

4 Χρήση της μνήμης

Οι δομές που χρησιμοποιούνται στο σύστημα είναι ένας 9x9 πίνακας στον οποίο αποθηκέυεται το sudoku, ένας buffer 256 bytes για την λήψη των χαρακτήρων από την σειριακή θύρα και ορισμένες καθολικές μεταβλητές, οι οποίες είναι volatile ώστε να μην πραγματοποιηθούν σε αυτές βελτιστοποιήσεις από τον μεταγγλωτιστή. Οι δομές που αναφέρθηκαν παραπάνω αποθηκέυονται στην SRAM, η οποία ανήκει στην data

LEDs	Clues found
LEDs off	< 10
LED00	≥ 10
LED00-01	≥ 20
LED00-02	≥ 30
LED00-03	≥ 40
LED00-04	≥ 50
LED00-05	≥ 60
LED00-06	≥ 70
LED00-07	≥ 80

Table 2: Progress Bar

memory του μικροεπξεργαστή. Λόγω της μεγάλης χωρητικότητας της flash και του σχετικά μικρού ποσοστού που καταλαμβάνει το πρόγραμμα (3124 bytes), υπάρχει η δυνατότητα να αποθηκευτούν οι παραπάνω δομές στην flash. Αυτό έχει ως πλεονέκτημα το μικρότερο access time αλλά δεν αποτελεί ορθή σχεδιαστική λύση επειδή έχει η flash αντοχή λίγο πάνω από 10,000 write/erase cycles.

5 Έλεγχος σωστής λειτουργίας - Αποτελέσματα

Για την επιβεβαίωση της σωστής λειτουργίας του συστήματος έγινε έλεγχος κάθε υποσυστήματος ξεχωριστά. Αρχικά, διαπιστώθηκε η σωστή λειτουργία της σειριακής θύρας χρησιμοποιώντας το πρόγραμμα PuTTY και εκτελώντας διάφορες εντολές του command set. Για την σωστή επικοινωνία με το PuTTY ήταν αναγκαίο να προστεθεί ένας επιπλεόν '
r' (carriage return) χαρακτήρας. Στη συνέχεια, επιβεβαιώθηκε η σωστή επίλυση των Sudoku του Παραρτήματος Α μεταξύ άλλων. Για τα παραπάνω τεστ χρησιμοποιήθηκαν οι εντολές AT, N, C, S, T, D και B.

Η επίλυση με βάση τον αλγόριθμο που αναλύθηκε στην Ενότητα 3.3 επιβεβαιώθηκε αρχικά σε ένα ξεχωριστό πρόγραμμα C και στη συνέχεια προσαρτήθηκε στο σύστημα.

Όσον αφορά το progress bar, αρχικά υλοποιήθηκε ως ξεχωριστό πρόγραμμα AVR χρησιμοποιώντας στατικά δεδομένα, επιβεβαιώθηκε ότι λειτουργεί σωστά και τέλος ενσωματώθηκε στο σύστημα.

Όταν το ολοκληρωμένο σύστημα δοκιμάστηκε με τη χρήση του PuTTY και τα boards του Παραρτήματος Α δεν διαπιστώθηκε κανένα πρόβλημα. Αντίθετα χρησιμοποιώντας το πρόγραμμα επικοινωνίας μέσω σειριακής θύρας, το οποίο αναπτύχθηκε από φοιτητή του μαθήματος, παρατηρήθηκαν τα εξής:

- Timeout errors τόσο στο κατέβασμα του sudoku με χρήση του option prog όσο και στη διαδικασία του debugging. Ο πιθανότερος λόγος των timeouts είναι η μικρή χρονική διάρκεια που έχει επιλεχθεί (50 ms κατα την συγγραφή της παρούσας αναφοράς). Το πιο σύνηθες σφάλμα επικοινωνίας με σειριακή θύρα είναι η απόκλιση μεταξύ baud rate AVR και προγράμματος PC. Αλλάζοντας τη συχνότητας στην τιμή 1.8432 MHz ("μαγική" συχνότητα) το baud rate γίνεται ακριβώς 9600 αλλά παρουσιάζονται τα ίδια σφάλματα. Σημείωση: Με τη χρήση του 10 MHz κρυστάλλου το baud rate γίνεται 9615.

- Invalid Solution το οποίο κατά πάσα πιθανότητα προκαλείται από σφάλμα στην επικοινωνία ανάμεσα στον AVR και στο πρόγραμμα που αναπτυχθήκε για τους σκοπούς του μαθήματος.

Τα παραπάνω σφάλματα δεν ήταν δυνατό να επιλυθούν λόγω αδυναμίας μεταγλώττισης του αναφερθέντος προγράμματος. Να σημειωθεί πως οι οδηγίες μεταγλώττισης του προγράμματος ακολουθήθηκαν βήμα προς βήμα αλλά δεν έγινε μεταγλώττιση.

6 Συμπεράσματα

Με την περάτωση του παρόντος πρότζεκτ, είναι πλέον εμφανές ότι η ανάπτυξη κώδικα σε έναν μικρεπεξεργαστή (είτε σε assembly, είτε σε C) είναι αρκετά πιο δύσκολη σε σχέση με την αντίστοιχη υλοποίηση σε μία high-level πλατφόρμα, η οποία τρέχει σε επεξεργαστή γενικού σκοπού. Τα προβλήματα που μπορεί να εμφανιστούν έχουν μεγαλύτερο φάσμα: από προβλήματα στον ίδιο τον MCU, το καλώδιο επικοινωνίας, τη μικρότερη υποστήριξη και την αδυναμία αποσφαλμάτωσης στο υλικό. Επίσης, η προσθήκη του πρωτοκόλλου επικοινωνίας UART προσθέτει τη δική του ποικιλία προβλημάτων, όπως ανακρίβειες στο baud rate, ελλείψη πληροφοριών και σφάλματα στα προγράμματα επικοινωνίας με τον AVR. Ωστόσο, η ανάγκη για ενσωματωμένα συστήματα είναι μεγάλη επειδή συστήματα όπως οι FPGAs, τα ASICs, οι GPUs και οι επεξεργαστες γενικού σκοπού έχουν χειρότερο συνδυασμό υψηλής κατανάλωσης ενέργειας, κόστους και πολυπλοκότητας. Συμπερασματικά, η επένδυση χρόνου στο παρόν πρότζεκτ, γνωρίζοντας κα τις τάσεις της βιομαηχανίας είναι αναμενόμενο ότι

θα αναδειχθεί πολύτιμη στην επαγγελαμτική μας καριέρα.

7 Παράρτημα Α - Τα Sudoku παζλ που χρησιμοποιήθηκαν

7.1 easy puzzle

1	7	4		9		6		
				3	8	1	5	7
5	3		7		1			4
		7	3	4	9	8		
8	4		5			3	6	
3		5			6	4	7	
2	8	6	9					1
			6	2	7		3	8
	5	3		8			9	6

7.2 easy2 puzzle

1		5	7		2	6	3	8
2					6			5
	6	3	8	4		2	1	
	5	9	2		1	3	8	
		2		5	8			9
7	1			3		5		2
		4	5	6		7	2	
5					4		6	3
3	2	6	1		7			4

7.3 inter puzzle

5	1	7	6				3	4
2	8	9			4			
3	4	6	2		5		9	
6		2					1	
	3	8			6		4	7
	9						7	8
7		3	4			5	6	

7.4 inter2 puzzle

5	1	7	6				3	4
	8	9			4			
3		6	2		5		9	
6							1	
	3				6		4	7
	9						7	8
7		3	4			5	6	

7.5 diff puzzle

		5	3					
8							2	
	7			1		5		
4					5	3		
	1			7				6
		3	2				8	
	6		5					9
		4					3	
					9	7		

7.6 hard puzzle

8	5				2	4		
7	2							9
		4						
			1		7			2
3		5				9		
	4							
				8			7	
	1	7						
				3	6		4	

8 Παράρτημα Β - Ανάλυση και απόδοση βελτιστοποιημένου αλγορίθμου

Ο βελτιστοποιημένος αλγόριθμός σε σχέση με τον απλό αλγόριθμο backtrack υπολογίζει για κάθε άδειο κελί του sudoku τα πιθάνα στοιχεία που θα μπορούσαν να βρίσοκνται εκεί. Συγκεκριμένα, ελέγχει για κάθε κελί τον υποπίνακα 3x3, τη στήλη

και τη σειρά που ανήκει και αποκλείει στοιχεία που βρίσκονται εκεί εως ότου υπάρχει ένα πιθανό στοιχείο για το εκάστοτε κελί. Η ίδια διαδικασία εκτελείται αναδρομικα για όλα τα άδεια κελιά του sudoku μέχρι να μην υπάρχει άδειο κελί στο πίνακα. Εάν, η συνάρτηση επίλυσης (solve_opt) επιστρέψει 0, τότε γίνεται backtrack μηδενίζοντας τις εικασίες. Η solve_opt θα επιστρέψει 0, μόνο όταν κάποιο άδειο κελί δεν μπορεί να πάρει καμία τιμή μεταξύ 1 και 9.

Στον Πίνακα 3 παρτίθενται τα αποτελέσματα από τη σύγκριση του απλού και του βελτιστοποιημένου αλγορίθμου επίλυσης Sudoku.

Sudoku	Επιλύθηκε	Χρόνος	Αριθμός backtracks
easy	Ναι	0.068133 ms	21
easy_opt	Ναι	0.122267 ms	0
easy2	Ναι	0.086267 ms	34
easy2_opt	Ναι	0.0958 ms	1
inter	Ναι	0.853 ms	579
inter_opt	Όχι	0.186467 ms	6
inter2	Ναι	7.866733 ms	6363
inter2_opt	Όχι	0.004379 ms	7
diff	Ναι	5.6304 ms	9949
diff_opt	Όχι	0.555 ms	9
hard	Ναι	161.5358 ms	335578
hard_opt	Όχι	0.637267 ms	11

Table 3: Μέσος χρόνος επίλυσης των παζλ του Παραρτήματος Α με χρήση του απλού και του βελτιστοποιημένου αλγορίθμου σε επεξεργαστή γενικού σκοπού.

9 Παράρτημα Γ - Flowcharts

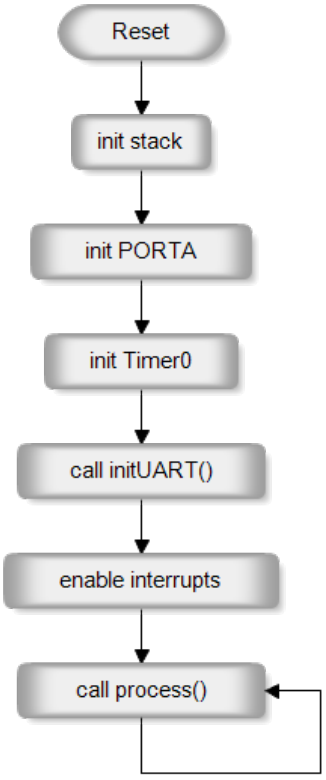

Figure 1: Ροοδιάγραμμα της main() ρουτίνας του προγράμματος.

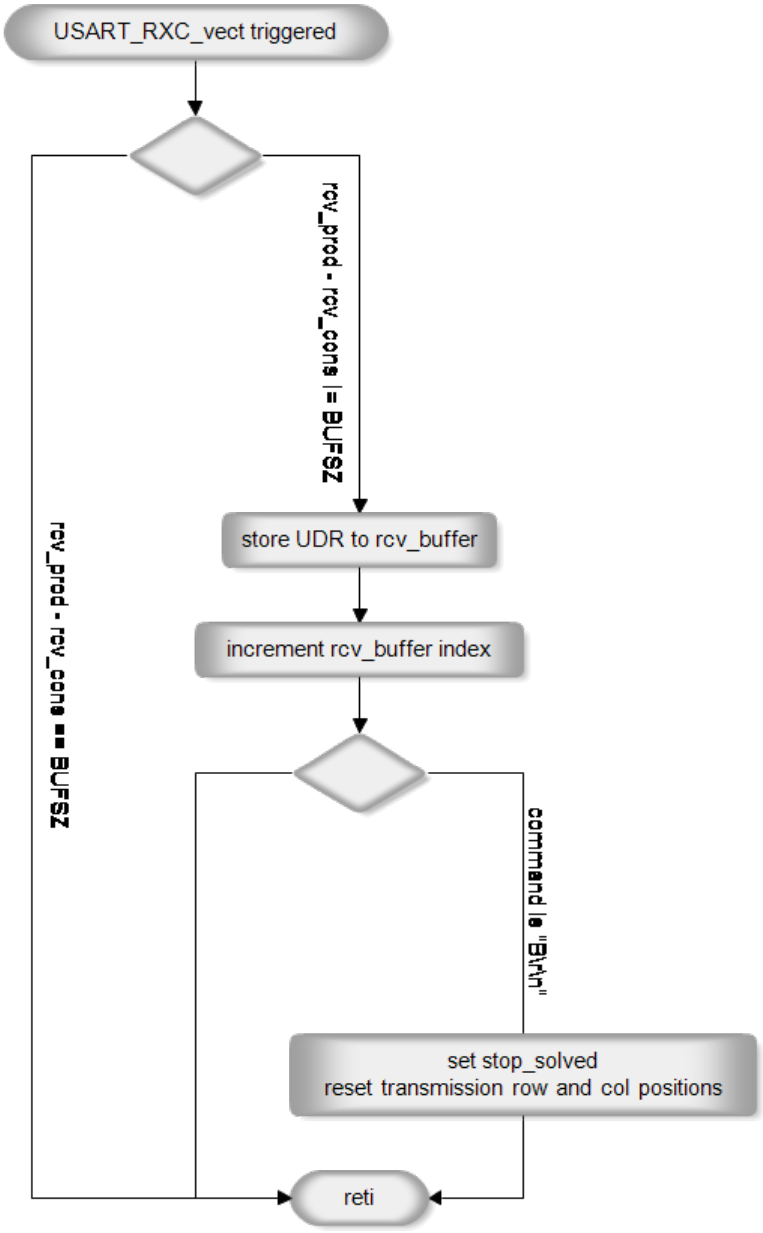

Figure 2: Ροοδιάγραμμα της ISR USART_RXC _vect.

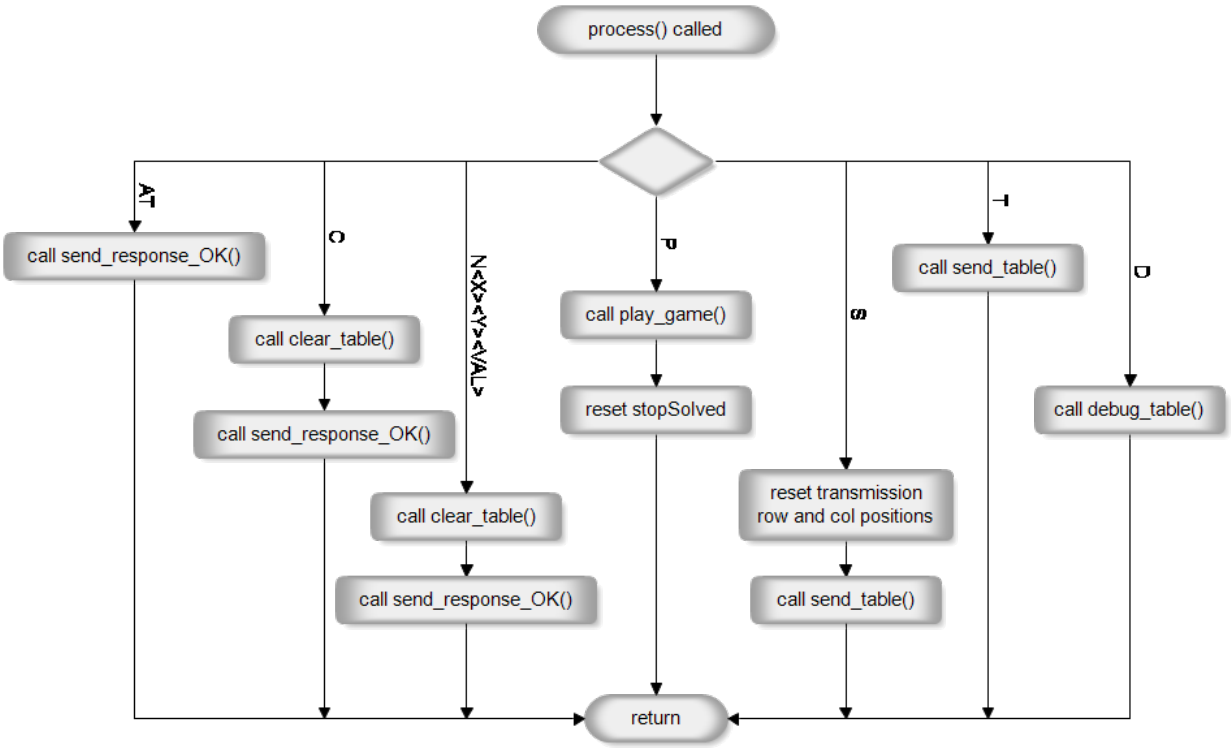

Figure 3: Ροοδιάγραμμα της process() ρουτίνας.

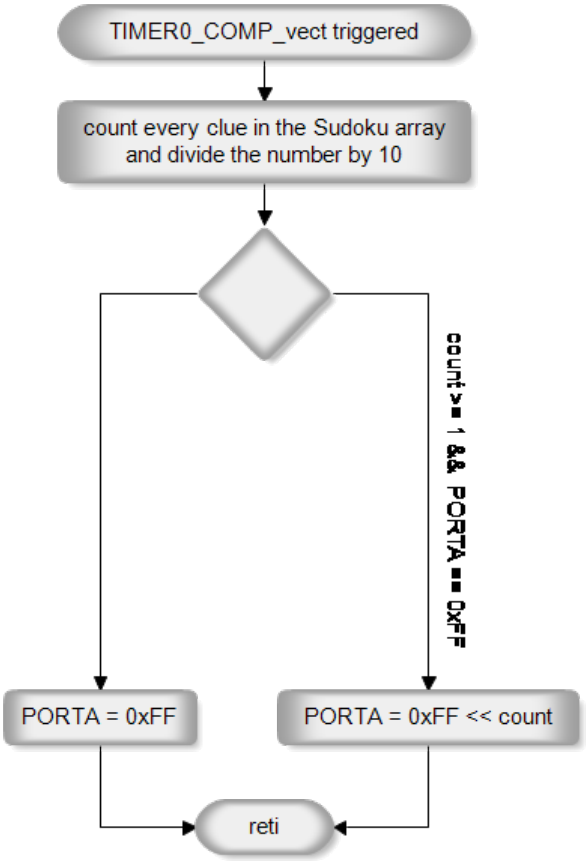

Figure 4: Ροοδιάγραμμα της ISR TIMER0_COMP _vect.