Μικροεπεξεργαστές και Περιφερειακά

Αναφορά – Εργασία 2

- Όνομα: Στασινός Αλκιβιάδης AEM: 9214

Περιγραφή λειτουργίας κώδικα

Στην εκφώνηση της εργασίας ζητείται η υλοποίηση δύο διαφορετικών εκδοχών του προγράμματος. Μία κατά την οποία το πρόγραμμα θα ανάβει το LED και θα μετράται ο χρόνος που χρειάστηκε ο χρήστης να αντιδράσει και να πατήσει το κουμπί ώστε να σβήσει το LED, και άλλη μία στην οποία θα συμβαίνει το ανίστροφο. Δηλαδή το LED θα σβήνει και θα μετράται ο χρόνος αντίδρασης του χρήστη να το ανάψει. Ο τρόπος που υλοποιείται αυτή η λειτουργικότητα είναι μέσω ενός **#define** στην αρχή του source, το οποίο ορίζει το **workingMode** σε "**IturnThemOff".** "Όταν υπάρχει αυτό το define τότε το πρόγραμμα θα λειτουργεί με βάση την πρώτη εκδοχή, ειδάλλως θα δουλεύει σύμφωνα με τη δεύτερη.

Απαραίτητο **initialization** στη main.

Αρχικά γίνεται το initialization του LED με τη ***leds_init()***, ενώ αμέσως μετά πραγματοποιείται το configuration του switch (δηλαδή του button μας) σε **PullUp mode** με **Rising trigger** και **callback** τη συνάρτηση **button_press_isr.** Συνοπτικά, τα παραπάνω σημαίνουν ότι ενεργοποιείται η σύνδεση με την εσωτερική **PullUp resistor** και έτσι στη default κατάσταση το button μας (**PC_13**) είναι συνδεδεμένο με το λογικό **1.** Όταν ο χρήστης το πατήσει θα πάει στο λογικό **0** και μόλις το ελευθερώσει, δηλαδή κατά τη **Rising** ακμή του σήματος, τότε θα καλεστεί ο **Interrupt Handler** με όνομα **button_press_isr.**

Επιπλέον τα Pins **P_DBG_ISR** και **P_DBG_MAIN** δηλώνονται ως pin εξόδου (**Output mode).** Ψάχνοντας στο αρχείο **platform.h** μπορούμε να δούμε ότι τα pins αυτά αντιστοιχούν στη πραγματικότητα στα **GPIO PC_8** και **PC_6** αντίστοιχα. Τέλος πριν το κύριο κομμάτι ενεργοποιούμε τα interrupts με ***__enable_irq()*** .

Ειδοποίηση του χρήστη πως οι μετρήσεις επρόκειτο να ξεκινήσουν.

Μετά το initialization , με τη χρήση της **delay_ms()** το LED θα παραμείνει σβηστό για 2 δευτερόλεπτα και ύστερα θα ξεκινήσει κανονικά το πρόγραμμα. Έτσι οι μετρήσεις δε θα ξεκινήσουν αμέσως μόλις ανοίξει το σύστημα και ο χρήστης θα μπορεί να ετοιμαστεί για τις επικείμενες δοκιμές.

Αρχικοποίηση των μεταβλητών πριν το βρόχο **while(1)**.

Πριν τον βασικό βρόχο **while(1)** της main αρχικοποιούνται οι απαραίτητες για τις μετρήσεις μεταβλητές. Πρώτα η **sample_count** που θα έχει κάθε φορά την τιμή των συλλεγμένων δειγμάτων με μέγιστη τιμή το **5** όπως ζητείται. Όταν συλλεχθεί το $5^{ο}$ δείγμα και καταγραφούν τα αποτελέσματα τότε επαναφέρεται στο **0**. Μετά δηλώνεται ένας πίνακας **5** θέσεων τύπου **unsigned int** με όνομα **reactions** στον οποίο θα αποθηκεύουμε το αποτέλεσμα κάθε χρόνου αντίδρασης (σε **ms**) για πέντε αντιδράσεις και αρχικοποιούμαι όλα τα στοιχεία του στο **0**. Ύστερα η **unsigned int** μεταβλητή **counter** η οποία όπως θα δούμε χρησιμεύει στον υπολογισμό του χρόνου αντίδρασης και η μεταβλητή **mean** όπου θα αποθηκευτεί η μέση τιμή των αντιδράσεων κάθε γύρου (5 δείγματα) . Επιπλέον ορίζεται η μεταβλητή **delaytime** η οποία θα περιέχει τον αριθμό των **ms** κατά τα οποία, μετά την καταγραφή των αποτελεσμάτων κάθε μέτρησης, θα καθυστερήσει το επόμενο άναμμα/σβήσιμο του LED. Η μεταβλητή αυτή παίρνει κάθε φορά τυχαία τιμή μεταξύ **1000** και **10000 ms** ώστε να προσομοιώσουμε ένα περιβάλλον όπου ο χρήστης δε μπορεί να προσαρμόσει τις αντιδράσεις του και να εκτιμά εύκολα το χρόνο για την επανενεργοποίηση του LED.

While (1) Βρόχος - Κύριο κομμάτι της main.

Βρισκόμενοι πλέον στο εσωτερικό του βρόχου **while(1)** αρχικά εκτελείται η εντολή ***gpio_toggle(P_DBG_MAIN)*** η οποία εναλλάσσει το σήμα **P_DBG_MAIN** σε κάθε επανάληψη. Ύστερα ανάλογα την εκδοχή του προγράμματος που ορίζεται στο

αρχικό **#define** θα εκτελεστεί είτε η ***leds_set(1,1,1)*** είτε η ***leds_set(0,0,0)*** . Το ποια εντολή εκτελείται ελέγχεται μέσω των preprocessor directives **#ifdef mode** και **#ifndef mode**. Έστω ότι λειτουργούμε βάσει της πρώτης εκδοχής. Από τη στιγμή που ανάβει το LED και όσο το κουμπί δεν έχει πατηθεί, τότε στο εσωτερικό του loop **while(!button_pressed)** αυξάνουμε τη **unsigned**

int μεταβλητή με όνομα **counter** κατά ένα κάθε **1000** κύκλους ρολογιού. Μόλις πατηθεί το κουμπί θα εκτελεστεί ο **interrupt handler** και θα θέσει τη μεταβλητή **button_pressed** σε **1**. Έτσι η συνθήκη του loop θα είναι **false** και η ροή του προγράμματος θα συνεχίσει εκτός του **while(!button_pressed)**. Αν συμβεί αυτό, αμέσως μετά το LED θα σβήσει, θα αποθηκεύσουμε στη κατάλληλη θέση (ανάλογα το sample_count) του πίνακα **reactions** την τιμή του **counter** διαιρεμένη με **16** και το **sample_count** θα αυξηθεί κατά ένα. Αποθηκεύοντας το **counter / 16** έχουμε τα milliseconds του χρόνου αντίδρασης. Αυτό συμβαίνει επειδή το default ρολόι του μικροεπεξεργαστή είναι **16 MHz** , κάτι που μπορούμε να δούμε και από την μεταβλητή **SystemCoreClock**, άρα ένας κύκλος ρολογιού διαρκεί 1 / (16 * 10^6) seconds ή **1 / (16 * 10^3) ms**. Έτσι, αφού ο **counter** αυξάνει κάθε **1000** κύκλους, **αρκεί να διαιρέσουμε την τιμή του με 16 ώστε να έχουμε αποτέλεσμα σε ms**. Ύστερα γίνεται έλεγχος για το αν συλλέχθηκαν 5 δείγματα και αν αυτή η συνθήκη είναι **true** τότε η τιμή του **sample_count** γίνεται reset σε **0** και με ένα **for** loop 5 επαναλήψεων αθροίζουμε τα στοιχεία του πίνακα **reactions** διαιρεμένα με **5** ώστε το τελικό άθροισμα που αποθηκεύεται στη μεταβλητή **mean** να είναι ο **μέσος όρος** των αποτελεσμάτων (ms). Η τελική τιμή της **mean** αποθηκεύεται στη θέση μνήμης **0x20000562** μέσω της συνάρτησης ***storeresults(unsigned int)***. Η συνάρτηση αυτή γράφτηκε σε **assembly** και θα αποθηκεύσει το argument της στη μνήμη 0x20000562. Τέλος τα στοιχεία του πίνακα **reactions** επαναφέρονται στο **0** , και μέσω της built-in συνάρτησης του Keil **rand()** επιλέγεται ένας τυχαίος αριθμός από **1000** έως **10000 ms** για την καθυστέρηση του επόμενου ανάμματος.

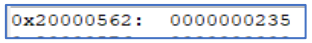
Interrupt Handler - button_press_isr

Ο **interrupt handler** που εκτελείται μόλις πατηθεί το κουμπί περιλαμβάνει όσο το δυνατόν λιγότερες εντολές ώστε η εκτέλεση του να είναι στιγμιαία και να μην παρακωλύει τη ροή του προγράμματος ή να δημιουργηθεί συμφόρηση από pending interrupts. Αρχικά τίθεται το debug σήμα του handler σε **1** με ***gpio_set(P_DBG_ISR, 1)*** και ύστερα ο handler αλλάζει την τιμή της μεταβλητής **button_pressed** σε 1 ώστε να τερματίσει το while loop που βρίσκεται εσωτερικά στη main. Η μεταβλητή **button_pressed** είναι δηλωμένη globally ως **volatile** για να αποφευχθεί τυχόν optimization από τον compiler. Τέλος επαναφέρεται το debug σήμα σε **0** με ***gpio_set(P_DBG_ISR, 0)*** .

Testing

Για τη διασφάλιση της καλής λειτουργίας του κώδικα διενεργήθηκε μια σειρά από δοκιμές. Μέσω του Keil ενεργοποιώντας τη λειτουργία **debug** μπορούμε να προσθέσουμε στο **watch window** τα στοιχεία του πίνακα **reactions** μέσω των οποίων θα βλέπουμε τα **milliseconds** που χρειάστηκε ο χρήστης να αντιδράσει για κάθε δείγμα. Επίσης προσθέτουμε τη μεταβλητή **counter** η οποία θα μας δείχνει τους κύκλους ρολογιού που πέρασαν σε κάθε πάτημα διά **1000**, τη **sample_count** ώστε να βλέπουμε πόσα δείγματα έχουν συλλεχθεί και τέλος το μέσο όρο **mean**. Παράλληλα κάθε 5 δείγματα ελέγχουμε στο **memory window** τη τιμή της διεύθυνσης **0x20000562** όπου αποθηκεύονται τα αποτελέσματα μας σε **ms**. Έτσι διαπιστώνουμε πως οι τιμές γράφονται επιτυχώς. Επιπλέον , λόγω του ότι ο **counter** αυξάνεται κάθε **1000** κύκλους, στη χειρότερη περίπτωση το interrupt θα έρθει ακριβώς πριν την έναρξη του delay των 1000 κύκλων. Έτσι ο counter θα αυξηθεί λανθασμένα μία φορά. Ωστόσο αυτό το worst-case scenario αντιστοιχεί σε χρονική απόκλιση 1 / 16 = 62.5 **us** , δηλαδή πολύ μικρή. Παρατίθεται και ένα στιγμιότυπο μετά το 5ο πάτημα του κουμπιού. Σε αυτό βλέπουμε πως οι τιμές των στοιχείων του πίνακα **reactions** περιέχουν όπως ήταν αναμενόμενο τους χρόνους αντίδρασης για κάθε πάτημα και μάλιστα οι τιμές είναι λογικές, καθώς βρίσκονται εντός του προβλεπόμενου χρόνου αντίδρασης ενός ανθρώπου **200-500 ms** . Επίσης από την τιμή του **counter** βλέπουμε πως για το **5ο δείγμα** μέχρι να πατήσει ο χρήστης του κουμπί πέρασαν 3600 * 10^3 κύκλοι. Εύκολα διαπιστώνεται η εγκυρότητα της τιμής αφού 3600 / 16 = 225 = **reactions[4] .** Τελευταία βλέπουμε **τη μέση τιμή** των αποτελεσμάτων στα **235** και ελέγχουμε από το **memory window** πως αυτή έχει εγγραφεί επιτυχώς στη διεύθυνση μνήμης **0x20000562.**

Name	Value
sample_count	5
counter	3600
reactions[0]	193
reactions[1]	243
reactions[2]	220
reactions[3]	301
reactions[4]	225
mean	235



Αποτελέσματα Μετρήσεων

Στον πίνακα δεξιά παρατίθενται οι μετρήσεις 5 δοκιμών από 5 διαφορετικούς χρήστες.

	1η Δοκιμή	2η Δοκιμή	3η Δοκιμή	4η Δοκιμή	5η Δοκιμή
1ο Δείγμα	276	246	319	263	227
2ο Δείγμα	354	493	202	229	238
3ο Δείγμα	294	322	263	218	234
4ο Δείγμα	495	225	304	348	204
5ο Δείγμα	374	265	295	230	203
Μέση τιμή (ms)	358.6	310.2	276.6	257.6	221.2