Εργαστήριο 2 - Ομάδα 3

Γαλανομάτης Σωκράτης - Τσακιρίδου Δήμητρα Μαρία

- Υλοποίηση κώδικα:

Αρχικά στον κώδικα, πέρα από τις βασικές βιβλιοθήκες (stdio.h, stdint.h) περιλαμβάνονται αρχεία .h βιβλιοθηκών από τις οποίες έχουμε αξιοποιήσει συναρτήσεις:
platform.h -> γίνονται ορισμοί που αφορούν την πλατφόρμα (platform) όπου τρέχει το πρόγραμμα. Πιο συγκεκριμένα, αξιοποιούμε τους ακροδέκτες (pins PA_5, PC_13) της πλακέτας για το LED και το κουμπί. Επίσης, ορίζονται οι εντολές LED_ON και LED_OFF που καθορίζουν αν τα LEDs είναι ενεργά όταν η έξοδος είναι υψηλή (high) ή χαμηλή (low) και αντιστοιχίζονται σε τιμές 1 και 0.
uart.h -> uart_init(uint32_t baud), uart_enable(void), uart_tx(uint8_t c), uart_print(char *str), uart_set_rx_callback(void (*callback)(uint8_t c))
queue.h -> queue_init(Queue *queue, uint32_t size), queue_enqueue(Queue *queue, uint8_t item), queue_dequeue(Queue *queue, uint8_t *item)
leds.h -> leds_init(void), leds_set(int red_on, int green_on, int blue_on)
gpio.h -> gpio_set(Pin pin, int value), gpio_get(Pin pin), gpio_set_mode(Pin pin, PinMode mode), gpio_set_trigger(Pin pin, TriggerMode trig), gpio_set_callback(Pin pin, void (*callback)(int status))
timer.h -> timer_init(uint32_t timestamp), timer_set_callback(void (*callback)(void)), timer_enable(void), timer_disable(void)

Οι global μεταβλητές switch_press_count και led_state είναι δηλωμένες με τη λέξη-κλειδί volatile, η οποία υποδεικνύει ότι μπορεί να αλλάξουν απρόσμενα λόγω εξωτερικών παραγόντων (όπως interrupts).

Στη συνέχεια ορίζουμε κάποιες συναρτήσεις πριν από την main():
void uart_rx_isr(uint8_t rx): Αυτή η συνάρτηση καλείται όταν λαμβάνεται ένας χαρακτήρας από την UART. Ελέγχει αν ο χαρακτήρας που λήφθηκε είναι ένας από τους χαρακτήρες που επιτρέπονται (αριθμοί 0-9, backspace ή enter) και, αν ναι, τον αποθηκεύει στην ουρά rx_queue.
void switch_init(void): Αυτή η συνάρτηση ρυθμίζει τον διακόπτη, καθώς ορίζει το pin PC_13 ως είσοδο και στην gpio_set_trigger χρησιμοποιείται η παράμετρος Rising που σημαίνει ότι η συνάρτηση button_press_isr θα καλείται όταν πατηθεί το κουμπί και το σήμα αλλάξει από χαμηλό σε υψηλό.
int switch_get(Pin pin): Αυτή η συνάρτηση επιστρέφει την κατάσταση του διακόπτη που ορίζεται από τον παράμετρο pin. Εάν ο διακόπτης είναι ενεργοποιημένος, επιστρέφεται 0, αλλιώς επιστρέφεται 1.
void button_press_isr(int sources): Αυτή η συνάρτηση είναι μια ISR που καλείται όταν πατηθεί ο διακόπτης. Αυξάνει τον μετρητή switch_press_count κατά ένα, τυπώνει την κατάσταση του LED, εναλλάσσει την κατάσταση του, τυπώνει τον αριθμό των πατημάτων του διακόπτη (switch_press_count) και απενεργοποιεί τον χρονοδιακόπτη για να κρατήσει την τελική κατάσταση.
void odd_digit_isr(): Μια ISR συνάρτηση που καλείται όταν ληφθεί ένας περιττός αριθμός από την UART. Ενεργοποιεί τον timer και εναλλάσσει την κατάσταση του LED κάθε 0,5 δευτερόλεπτα.

void even_digit_isr(): Επίσης, μία ISR συνάρτηση που καλείται όταν ληφθεί ένας άρτιος αριθμός από την UART και απενεργοποιεί τον χρονοδιακόπτη, άρα κρατάει σταθερή κατάσταση.
void timer_callback(): Κάθε φορά που τελειώνει ο χρόνος του timer καλείται η ISR συνάρτηση odd_digit_isr().

Έπειτα σχετικά με τον κώδικα μέσα στην main() συνάρτηση:

Αρχικοποιούμε τις μεταβλητές που θα χρησιμοποιήσουμε. Πιο συγκεκριμένα ο rx_char για την αποθήκευση των χαρακτήρων που λαμβάνονται από την UART, ο πίνακας buff για την αποθήκευση των εισόδων του χρήστη, μια μεταβλητή buff_index για την παρακολούθηση του μήκους του πίνακα buff, μια μεταβλητή number για την αποθήκευση των αριθμητικών εισόδων του χρήστη, καθώς και έναν πίνακα led για πιθανές εξόδους LED.

Μετά αρχικοποιούμε την ουρά rx_queue, τη UART, το LED, τον διακόπτη και τον timer:
queue_init(&rx_queue, 128): αρχικοποιεί την ουρά rx_queue με μέγεθος 128, για την αποθήκευση των χαρακτήρων που λαμβάνονται από την UART.
uart_init(115200): αρχικοποιεί την UART για επικοινωνία με ρυθμό μετάδοσης 115200 bps.
uart_set_rx_callback(uart_rx_isr): ορίζει τη συνάρτηση uart_rx_isr ως callback που καλείται κάθε φορά που λαμβάνεται ένας χαρακτήρας από την UART και αυτή η συνάρτηση προσθέτει τους λαμβανόμενους χαρακτήρες στην ουρά rx_queue.
uart_enable(): ενεργοποιεί την λειτουργία της UART
leds_init(): αρχικοποιεί τα LEDs.
switch_init(): αρχικοποιεί τον διακόπτη με βάση τη συνάρτηση που έχουμε φτιάξει πιο πάνω.
gpio_set_callback(PC_13, button_press_isr): ορίζει τη συνάρτηση button_press_isr ως callback που καλείται όταν γίνει πατημένος ο διακόπτης.
timer_init(500000): αρχικοποιεί τον χρονοδιακόπτη με περίοδο 500ms.
timer_set_callback(timer_callback): ορίζει τη συνάρτηση timer_callback ως callback που καλείται όταν παρέλθει ο χρόνος του χρονοδιακόπτη.
timer_disable(): Αυτή η εντολή απενεργοποιεί τον χρονοδιακόπτη, που πιθανόν ήταν αναμμένος σε προηγούμενη εκτέλεση του προγράμματος.
__enable_irq(): Αυτή η εντολή ενεργοποιεί τις διακοπές, επιτρέποντας την εκτέλεση των ISR (Interrupt Service Routines).

Ακολουθεί ο κύριος βρόχος επανάληψης που τρέχει απεριόριστα. Σε αυτόν τον βρόχο γίνεται έλεγχος για την προτεραιότητα της εισόδου του διακόπτη, αφού σε περίπτωση που πατηθεί το κουμπί ενώ ο χρήστης καταχωρεί το input του, θα πρέπει να έχει προτεραιότητα η είσοδος από το διακόπτη. Αν ο διακόπτης πατηθεί, λοιπόν, καλείται η συνάρτηση button_press_isr για να εκτελέσει τις κατάλληλες ενέργειες ανάλογα με την κατάσταση του LED και να εκτυπώσει πόσες φορές έχει πατηθεί ο διακόπτης. . Διαφορετικά, ζητείται από το χρήστη να εισάγει έναν κωδικό ΑΕΜ.

Μέχρι να ληφθεί ο χαρακτήρας '\r' (Enter) ή μέχρι ο δείκτης buff_index να φτάσει το μέγιστο μέγεθος του buffer BUFF_SIZE, ώστε να μην υπερχειλιστεί, ακολουθείται η εξής διαδικασία:

while (!queue_dequeue(&rx_queue, &rx_char)) __WFI(): Αυτή η εντολή περιμένει να υπάρξει κάποιος χαρακτήρας στην ουρά rx_queue και στη συνέχεια τον αφαιρεί. Η συνάρτηση queue_dequeue που καλείται ελέγχει αν υπάρχει χαρακτήρας στην ουρά και επιστρέφει true αν ναι ή false αν όχι. Το __WFI() είναι ένας μηχανισμός που βάζει τον μικροελεγκτή σε κατάσταση χαμηλής κατανάλωσης ενέργειας (Wait For Interrupt) μέχρι να λάβει ένας νέος χαρακτήρας από τη σειριακή θύρα.
if (rx_char == 0x7F) { ... } else { ... }: Αν ο χαρακτήρας είναι το backspace (0x7F), τότε αυτό το τμήμα κώδικα μειώνει τον δείκτη buff_index κατά ένα και στέλνει τον χαρακτήρα backspace πίσω στη σειριακή θύρα για να διαγράψει τον προηγούμενο χαρακτήρα που είχε εκτυπωθεί στην οθόνη.
else { ... }: Αν ο ληφθείς χαρακτήρας δεν είναι backspace, τότε αποθηκεύεται στον buffer buff στη θέση buff_index και αυξάνεται ο δείκτης buff_index. Έπειτα, ο χαρακτήρας αποστέλλεται πίσω στη σειριακή θύρα για να εμφανιστεί στην οθόνη. Αντικατάσταση του τελευταίου χαρακτήρα με '\0', ώστε να δηλωθεί το τέλος του string.

Στη συνέχεια, πρώτα ελέγχεται αν το μήκος του buffer (buff_index) είναι μικρότερο από το μέγεθος του buffer και αν είναι, σημαίνει ότι δόθηκε μια έγκυρη είσοδος. Έπειτα, γίνεται μετατροπή του περιεχομένου του buffer σε ακέραιο αριθμό με τη χρήση της συνάρτησης sscanf και αν επιστρέφει 1, τότε ελέγχεται αν το τελευταίο ψηφίο του ακεραίου είναι άρτιος ή περιττός. Αν είναι άρτιος, καλείται η συνάρτηση even_digit_isr() για να αντιμετωπιστεί η άρτια περίπτωση, ενώ αν είναι περιττός καλείται η συνάρτηση odd_digit_isr() για την περίττη περίπτωση. Σε κάθε περίπτωση, εκτυπώνεται η κατάσταση του LED με τη χρήση της συνάρτησης uart_print.

- Προβλήματα που αντιμετωπίσαμε:

Στο συγκεκριμένο πρότζεκτ κυρίως μας απασχόλησε περισσότερο το να κατανοήσουμε τους drivers που μας δίνονται και να επιλέξουμε ποιες συναρτήσεις θα αξιοποιήσουμε και από ποιους. Πρακτικά προβλήματα που αντιμετωπίσαμε είναι ότι καμιά φορά μετά το debug, ο κώδικας κολλούσε (βέβαια αυτό καμιά φορά οφειλόταν σε κάποιο infinite loop που είχαμε δημιουργήσει καταλάθος). Ένα ακόμη πρόβλημα το οποίο λύθηκε όμως πολύ γρήγορα, αφορά την εμφάνιση της κατάστασης του LED μέσω UART, καθώς στην αρχή είχαμε ορίσει μέσα στην odd_digit_isr() να κάνει την εκτύπωση κάθε αλλαγής κατάστασης, με αποτέλεσμα να δυσκολεύει η εκτύπωση του AEM σε μία μόνο γραμμή. Αυτό το λύσαμε βάζοντας απλά “The LED is blinking” μέσα στην main() κάθε φορά που καλείται η συνάρτηση. Τέλος, αρκετό χρόνο μας πήρε να κάνουμε λειτουργικό τον κώδικα για τον διακόπτη, καθώς στην αρχή είχαμε βάλει έναν if έλεγχο, ο οποίος δεν ικανοποιούνταν και έτσι δεν καλούταν το κομμάτι του κώδικα στην button_press_isr().

- Testing:

Το testing, δεν μπορούσε να γίνει με την χρήση του simulator, παρά μόνο όταν ήμασταν στο εργαστήριο έχοντας την πλακέτα. Έτσι, αφού κάναμε build τον κώδικα, ξεκινούσαμε το debug session όπου για να ελέγξουμε αν η υλοποίηση μας είναι σωστή, βλέπαμε πως μεταβάλλονταν οι τιμές των registers στον πίνακα αριστερά, ποιες τιμές αποθηκεύονταν στις μεταβλητές μας και πότε εκτυπώνονται, στην αρχή με printf και έπειτα με UART. Για να ελέγξουμε όσες περιπτώσεις μπορούμε, στο debugging τρέχαμε τον κώδικα με διάφορα breakpoints, ώστε να δούμε σε ποια σημεία κολλάει και σε ποια δεν μπαίνει καθόλου.