Μικροεπεξεργαστές και Περιφερειακά

Αναφορά Εργαστηρίου 2

Ονοματεπώνυμο	ΑΕΜ	E-mail
Κωνσταντίνος Πράττης	10056	kprattis@ece.auth.gr
Βασιλική Ρίζου	10021	vasilikirn@ece.auth.gr

Εισαγωγή

Η παρούσα αναφορά αποσκοπεί στην επεξήγηση της εργασίας που ανατέθηκε στα πλαίσια του 2ου εργαστηρίου, του μαθήματος Μικροεπεξεργαστές και Περιφερειακά. Η παρούσα εργασία καλύπτει τον προγραμματισμό σε C ενός μικροελεγκτή ARM. Στόχος είναι, μέσω σειριακής διεπαφής με την UART να ζητείται από το χρήστη η εισαγωγή του ΑΕΜ του. Ανάλογα το τελευταίο ψηφίο (περιττό ή άρτιο) που αυτός έχει εισάγει, θα καλείται μια ρουτίνα ISR, προς εξυπηρέτηση του interrupt, η οποία κατά περίπτωση θα ενεργοποιεί ή θα απενεργοποιεί ένα LED. Σε περίπτωση που πατηθεί ο διακόπτης, θα πρέπει μέσω κλήσης κατάλληλης ISR ρουτίνας, να αλλάζει το state του LED. Επιπλέον, θα πρέπει η προτεραιότητα εξυπηρέτησης του interrupt που προέρχεται από το πάτημα του διακόπτη να είναι μεγαλύτερη από αυτή της εισαγωγής των ψηφίων του ΑΕΜ από τον χρήστη. Να σημειωθεί ότι το LED που χρησιμοποιήθηκε είναι στο pin *P_LED_R* όπως ορίζεται στην *gpio.h*.

Interrupt Handler για το Button – *switch_ISR*

Η συνάρτηση που καλείται κάθε φορά που λαμβάνεται ένα interrupt από το κουμπί είναι η *switch_ISR*. Σύμφωνα με τις οδηγίες της εκφώνησης, η συνάρτηση αυτή αλλάζει την κατάσταση του LED, δηλαδή αν το LED είναι “On” το πηγαίνει στη θέση “Off” και αντίστροφα, ενώ ταυτόχρονα αυξάνει και έναν *counter* ο οποίος μετράει πόσες φορές συνολικά από την αρχή του προγράμματος πατήθηκε το κουμπί.

Για την υλοποίηση των παραπάνω, έγινε χρήση της ρουτίνας ***gpio_toggle*** της βιβλιοθήκης *gpio.h*, η οποία δέχεται ως όρισμα ένα pin και αντιστρέφει λογικά την τιμή της εξόδου του.

Ακόμα, ο *counter* πρέπει να είναι static μεταβλητή ώστε αφενός να μην την πειράξει ο *compiler*, κάτι που είναι πιθανό αφού δεν χρησιμοποιείται μέσα στην main αλλά μόνο στους ISR, ενώ παράλληλα για να διατηρεί και την τιμή του ακόμα και μετά το πέρας της εκτέλεσης της συνάρτησης.

Τέλος, η συνάρτηση αυτή τυπώνει μέσω *UART* ανάλογα με την κατάσταση του LED (που βρίσκεται με gpio_get) και με τη βοήθεια της *sprintf* και της *uart_print*, την τιμή του *counter* και την κατάσταση του LED.

Interrupt Handler για την uart – *getchar_ISR*

Η εν λόγω ρουτίνα, καλείται κάθε φορά που ο χρήστης εισάγει έναν χαρακτήρα μέσω σειριακής διεπαφής με την UART. Σύμφωνα με τις οδηγίες, διακρίνονται 2 περιπτώσεις:

- *Αν το τελευταίο ψηφίο του ΑΕΜ του χρήστη είναι περιττός αριθμός, θα πρέπει να ανάβει το LED.*

- *Αν το τελευταίο ψηφίο του ΑΕΜ του χρήστη είναι άρτιος αριθμός, θα πρέπει να απενεργοποιείται το LED.*

Επιπλέον, θα πρέπει κάθε αλλαγή της κατάστασης του LED, να εκτυπώνεται μέσω της UART. Έτσι, η λογική που ακολουθήθηκε είναι η εξής. Κάθε φορά που καλείται η getchar_ISR, ελέγχεται αρχικά αν ο χαρακτήρας που εισήγαγε ο χρήστης είναι ο Carriage return ('\r'). Εάν κάτι τέτοιο δεν ισχύει, τότε αποθηκεύεται ο συγκεκριμένος χαρακτήρας, που είναι ο τελευταίο που εισήγαγε ο χρήστης, στην μεταβλητής ***static uint8_t last_digit_AEM.*** Διαφορετικά, ο τελευταίος χαρακτήρας είναι τιμή της ***last_digit_AEM***. Σε αυτή την περίπτωση καλείται η ***gpio_set(P_LED, last_digit_AEM % 2)***, της βιβλιοθήκης gpio.h, η οποία θέτει την τιμή του συγκεκριμένου pin (***P_LED***) στην τιμή ***last_digit_AEM % 2***. Άρα, εάν το τελευταίο ψηφίο του ΑΕΜ είναι περιττό, τότε ***last_digit_AEM % 2 = 1*** και άρα το LED ανάβει (αν προηγουμένως ήταν απενεργοποιημένο), διαφορετικά ***last_digit_AEM % 2 = 0*** και άρα το LED απενεργοποιείται (αν προηγουμένως ήταν αναμμένο). Στην πρώτη περίπτωση, εμφανίζεται επιπλέον και το μήνυμα "*Odd AEM, Led is on*" , ενώ στην δεύτερη "*Even AEM, Led is off*". Τελικά, ξαναεμφανίζεται σχετικό μήνυμα στον χρήστη, για την εισαγωγή του ΑΕΜ του.

Συνάρτηση main

Εφόσον υλοποιήθηκαν οι interrupt handlers, η βασική δουλειά της main είναι να αρχικοποιήσει το σύστημα και μετά να μπει σε μία άδεια while(1) λούπα, όπου θα περιμένει να έρθουν τα interrupts από τα περιφερειακά.

Για την αρχικοποίηση:

- Ενεργοποιούμε τα interrupts με ***__enable_irq()***
- Αρχικοποιούμε τη UART με ***uart_init(115200)*** και ***uart_enable()***, ενώ ορίζουμε και τον interrupt handler του receive (rx) να είναι ο getchar_ISR με την ***uart_set_rx_callback***.
- Καλούμε την ***leds_init*** του leds.h ώστε να αρχικοποιηθούν τα leds.
- Για το κουμπί, αρχικά θέτουμε για το αντίστοιχο pin (***P_SW***) το mode σε ***PullDown*** και το trigger σε ***Rising*** και στη συνέχεια ορίζουμε τον interrupt handler του pin να είναι ο switch_ISR, όλα με τις κατάλληλες συναρτήσεις από το gpio.h. Ο λόγος που θέλουμε το mode να είναι σε PullDown, είναι έτσι ώστε όταν το κουμπί δεν πατιέται, αντί να αφήνεται να αιωρείται, να λαμβάνει συγκεκριμένη τιμή. Και μάλιστα, θα θέλαμε όταν το switch δεν είναι πατημένο, η τιμή του pin να είναι LOW, ενώ αντίστοιχα όταν πατιέται η τιμή του να είναι HIGH. Έτσι, το default state του P_SW είναι 0, και μόνο όταν αυτό πατιέται είναι 1. Όσον αφορά το Rising mode στο trigger, αυτό επιλέγεται, καθότι θα θέλαμε το συγκεκριμένο pin να γίνεται triggered, κατά την μετάβαση από το λογικό 0 στο λογικό 1.
- Τέλος, για να είναι το κουμπί σε μεγαλύτερη προτεραιότητα από την uart, χρησιμοποιείται η εντολή ***NVIC_SetPriority(EXTI15_10_IRQn, 1)***, ώστε να θέσουμε το priority στην έξοδο του κουμπιού, την οποία αναζητήσαμε στα αρχεία .h ίσο με 1. Το κουμπί αντιστοιχεί στην πόρτα C, στο pin 13. Η συνάρτηση, gpio_set_callback, θέτει την προτεραιότητα όλων των pin από 10- 15 ίση με 3.