ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

Output 2: Online Course for Microcontrollers: syllabus, open educational resources

Practice leaflet: Module_2-5 Interrupts

Lead Partner: International Hellenic University (IHU)

Erasmus+

GRANT NUMBER: 2020-1-PL01-KA226-HE-095653

Δήλωση

Αυτό το αρχείο συντάχθηκε στο πλαίσιο του έργου ENGINE. Όπου έχουν χρησιμοποιηθεί άλλα δημοσιευμένα και αδημοσίευτα υλικά, αυτά έχουν αναγνωριστεί.

Πνευματική ιδιοκτησία

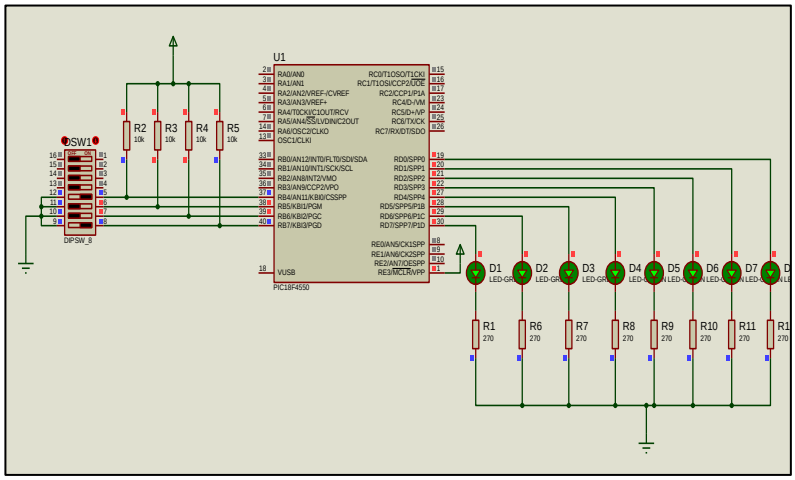
Αυτό το έργο έχει χρηματοδοτηθεί με την υποστήριξη της Ευρωπαϊκής Επιτροπής. Αυτή η έκθεση αντικατοπτρίζει μόνο τις απόψεις του συγγραφέα και η Επιτροπή δεν μπορεί να θεωρηθεί υπεύθυνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν.

Πίνακας Περιεχομένων

Δραστηριότητες

1. Εναλλαγή καταστάσεων στην PORTD

Να γραφεί πρόγραμμα με το οποίο προκαλείται μια διακοπή από αλλαγή κατάστασης στους ακροδέκτες RB4, RB5, RB6, RB7. Κάθε φορά που αλλάζει η κατάσταση σε έναν από τους διακόπτες που είναι συνδεδεμένοι στου ακροδέκτες RB4, RB5, RB6, RB7 να αλλάζει η κατάσταση όλων των bit της πόρτας D. Η αρχική τιμή της πόρτας D να είναι FF.

(20 λεπτά)	**Βήμα 1. Υλοποίηση του κυκλώματος** **Βήμα 2. Ολοκλήρωση κώδικα** **Βήμα 3. Μεταφορά κώδικα στον μικροελεγκτή** **Βήμα 4. Έλεγχος λειτουργίας**
Βήμα 1 (10 λεπτά)	Υλοποίηση του κυκλώματος. Διασύνδεση PORTD με 8 LEDs, και διακόπτες με την PORTB.  *Figure 1. Διασύνδεση*

Βήμα 2
(5 λεπτά)
	Μελετήστε και **συμπληρώστε** τον παρακάτω κώδικα
	
	`#include <main.h>`
	`#byte PORTD=0xF83`
	`#byte PORTB=0xF81`
	
	`//δήλωση συναρτήσεων`
	`void rb(void);`
	`void init (void);`
	
	`void main(){`
	`  ---------- //καλώ την συνάρτηση που κάνει αρχικοποιήσεις`
	`  while(TRUE){;}  //για πάντα`
	`  }`
	
	`//Ρουτίνα εξυπηρέτησης διακοπής από τα RB4~RB7`
	`void init (){`
	`  ---------- //ορισμός PORTB ως είσοδος`
	`  set_tris_d(0x00); //ορισμός PORTD ως έξοδος`
	`  enable_interrupts(INT_RB);`
	`  ----------  //ενεργοποίηση "γενικού διακόπτη"`
	`interrupt`
	`  PORTD=0xFF;  //αρχική κατάσταση στην PORTD`
	`}`
	
	`#INT_RB`
	`void rb(){  //ρουτίνα εξυπηρέτησης διακοπής`
	`  int8 a;`
	`  PORTD=PORTD^0xFF;`
	`  a=PORTB;`
	`}`

Βήμα 3 (3 λεπτά)	Δημιουργήστε το hex file και φορτώστε το στον μικροελεγκτή
Βήμα 4 (2 λεπτά)	Ελέγξτε ότι το κύκλωμα λειτουργεί σωστά

2. Εύρεση bit που προκάλεσε το interrupt

Σε αυτήν την δραστηριότητα θέλουμε κάθε φορά που ένα pin από τα RR4~RB7 προκαλεί διακοπή, να εμφανίζεται στο αντίστοιχο pin της PORTD.

(10 λεπτά)	***** Το κύκλωμα είναι ίδιο με την προηγούμενη δραστηριότητα ***** **Βήμα 1. Μελέτη του κώδικα** **Βήμα 2. Μεταφορά κώδικα στον μικροελεγκτή** **Βήμα 3. Έλεγχος λειτουργίας**

Βήμα 1
(4 λεπτά)
	Μελετήστε τον παρακάτω κώδικα
	
	`#include <main.h>`
	`#byte PORTB=0xF81`
	`#byte PORTD=0xF83`
	
	`// Δήλωση της ρουτίνας διακοπής από τους ακροδέκτες RB4, RB5, RB6, RB7`
	`void rb(void) ;`
	`void init(void);`
	
	`//Global μεταβλητή για να κρατήσουμε την τελευταία τιμή της πόρτας B`
	`int8 lastPORTB;`
	
	`void main(){`
	`  // ρουτίνα αρχικοποίησης, ρυθμίσεις εισόδων/εξόδων, διακοπών κλπ.`
	`  init();`
	`  // το κύριο πρόγραμμα δεν κάνει τίποτα`
	`  while(TRUE){;}`
	`}`
	
	`void init (void){`
	`  set_tris_d(0x00); //  Καθορισμός της πόρτας D ως εξόδου`
	`  enable_interrupts(GLOBAL);  //Ενεργοποίηση του γενικού διακόπτη διακοπών`
	`  enable_interrupts(INT_RB);  //Ενεργοποίηση διακοπής από αλλαγή`
	`  //κατάστασης στους ακροδέκτες`
	`  // RB4, RB5, RB6, RB7`
	`  PORTD=0x00;`
	`  lastPORTB=PORTB;`
	`}`
	
	`#INT_RB  // Διακοπή που προκαλείται από αλλαγή`
	`  // κατάστασης στους ακροδέκτες RB4, RB5, RB6, RB7`
	`void rb (void){ // Αρχή της ρουτίνας εξυπηρέτησης της διακοπής rb`
	`  int8 changes;  // Ορισμός μεταβλητής των 8 bit.`
	`  changes = lastPORTB ^ PORTB;  // Γίνεται 1 το bit που άλλαξε`
	`  //Εμφανίζεται σαν 1 στη μεταβλητή changes το bit που άλλαξε.`
	`  // Μεταφέρεται η νέα τιμή της Πόρτας B`
	`  //Στην τιμή της μεταβλητής lastPORTB`
	`  lastPORTB=PORTB;`
	`  PORTD=changes;`
	`  //καθυστέρηση για αποφυγή αναπηδήσεων`
	`  delay_ms (100);`
	`}`

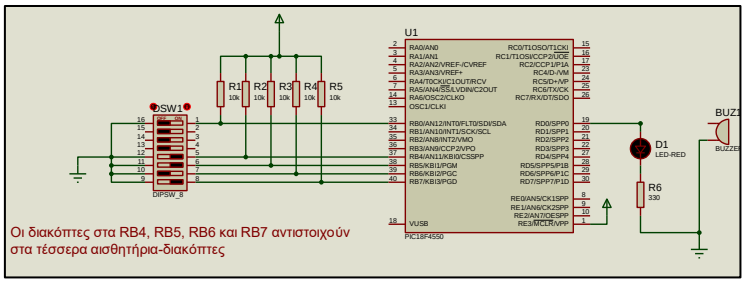
Βήμα 2 (4 λεπτά)	Δημιουργήστε το hex file και φορτώστε το στον μικροελεγκτή
Βήμα 3 (2 λεπτά)	Ελέγξτε ότι το κύκλωμα λειτουργεί σωστά

3. Σύστημα συναγερμού αυτοκινήτου

Έστω ότι οι τέσσερεις ακροδέκτες RB4, RB5, RB6, RB7 είναι συνδεδεμένοι σε 4 υδραργυρικούς διακόπτες οι οποίοι είναι τοποθετημένοι στα άκρα ενός σταυρού και κλείνουν όταν το αυτοκίνητο μετακινηθεί προς μια από τις 4 κατευθύνσεις.

Αν το σύστημα είναι ενεργοποιημένο και αλλάξει η κατάσταση τουλάχιστον σε έναν από τους ακροδέκτες RB4…RB7 και το σύστημα συνεχίζει να είναι ενεργοποιημένο και για τα επόμενα 3 sec, τότε χτυπάει μια σειρήνα για 3 sec και στην συνέχεια το σύστημα περιμένει για νέα αλλαγή κατάστασης στους ακροδέκτες RB4 έως RB7.

Ο οπλισμός του συστήματος, δηλαδή η ενεργοποίηση του, γίνεται από τον ακροδέκτη RB0. Όταν RB0=1 το σύστημα είναι οπλισμένο (ενεργοποιημένο). Όταν RB0=0 το σύστημα είναι απενεργοποιημένο.

(20 λεπτά)	**Βήμα 1. Υλοποίηση του κυκλώματος** **Βήμα 2. Μεταφορά κώδικα στον μικροελεγκτή** **Βήμα 3. Έλεγχος λειτουργίας**
Βήμα 1 (3 λεπτά)	Υλοποίηση του κυκλώματος. Διασύνδεση RD0 με LED, και διακόπτες με την PORTB.  ***Figure 2.*** *Διασύνδεση*

Βήμα 2
(14 λεπτά)
	Μελετήστε τον παρακάτω κώδικα
	`#include <main.h>`
	`#byte PORTB=0xF81 //ορισμός των θυρών με την θέση τους στην μνήμη`
	`#byte PORTD=0xF83`
	`// Δήλωση συναρτήσεων, global μεταβλητών`
	`void init (void);`
	`void rb (void);`
	`void main(){`
	`  init();  //κλήση της ρουτίνας αρχικοποίησης`
	`  while (TRUE) {;}`
	`}`
	`//ρουτίνα αρχικοποίησης`
	`void init (void){`
	`  set_tris_b(0xff);  // Καθορισμός της πόρτας B ως εισόδου`
	`  set_tris_d(0x00);  //  Καθορισμός της πόρτας D ως εξόδου`
	`  enable_interrupts(GLOBAL);  //Ενεργοποίηση του γενικού διακόπτη διακοπών`

	enable_interrupts(INT_RB); //Ενεργοποίηση διακοπής από αλλαγή
	//κατάστασης στους ακροδέκτες RB4, RB5, RB6, RB7
	PORTD=0x00; //αρχική τιμή 0 στην θύρα D
	}
	
	//ρουτίνα διακοπής
	#INT_RB
	void rb (void){
	if(input(PIN_B0)==1){
	delay_ms(3000); //Αναμονή για 3 δευτερόλεπτα
	if(input(PIN_B0)==1){ //Αν συνεχίσει να είναι ενεργοποιημένος
	//ο συναγερμός ενεργοποιείται η σειρήνα
	output_high(PIN_D0);//Ενεργοποίηση σειρήνας
	delay_ms(3000); //αναμονή για 3 sec
	output_low(PIN_D0);//Απενεργοποίηση σειρήνας
	}
	}
	}

Βήμα 3 (3 λεπτά)	Δημιουργήστε το hex file και φορτώστε το στον μικροελεγκτή. Ελέγξτε ότι το κύκλωμα λειτουργεί σωστά