ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

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

Practice leaflet: Module_2-6 Timer

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

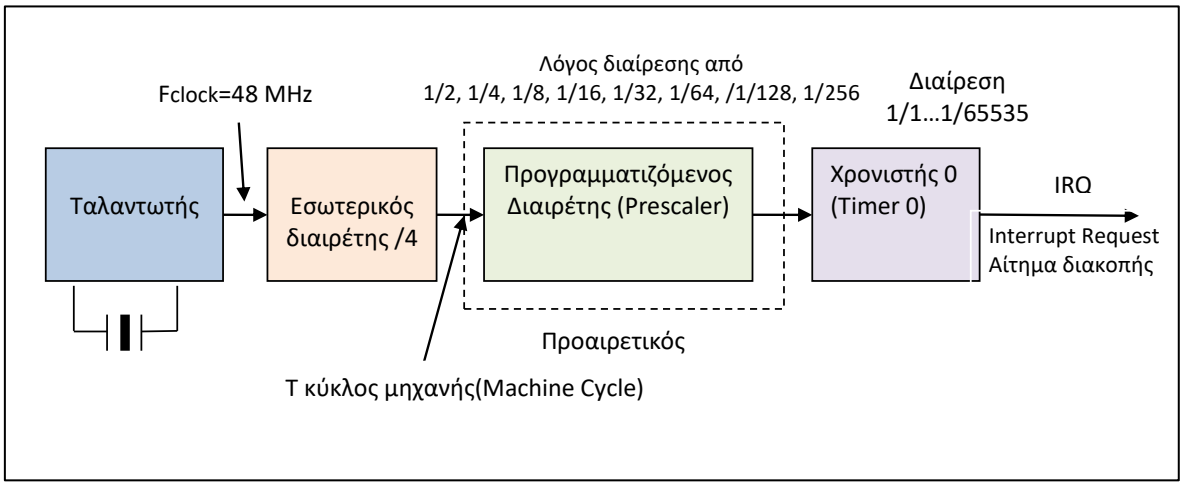
Αυτό το έργο έχει χρηματοδοτηθεί με την υποστήριξη της Ευρωπαϊκής Επιτροπής. Αυτή η έκθεση αντικατοπτρίζει μόνο τις απόψεις του συγγραφέα και η Επιτροπή δεν μπορεί να θεωρηθεί υπεύθυνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν.

Πίνακας Περιεχομένων

Δραστηριότητες

1. Διακοπή από τον Timer0 κάθε 10ms

Να γραφεί πρόγραμμα με ρουτίνα διακοπών από τον Timer0 με το οποίο αναβοσβήνουν τα LED τα οποία συνδέονται στην πόρτα Β κάθε 200 ms. Προσοχή! Στο πρόγραμμα δεν θα χρησιμοποιηθούν καθόλου συναρτήσεις χρονοκαθυστέρησης. Ως βάση χρόνου, θα ρυθμιστεί ο Timer0 να προκαλεί μία διακοπή κάθε 10ms.



⇨ Υπολογισμός της αρχικής τιμής του Timer0 ώστε να συμβαίνουν διακοπές κάθε 10 ms

Η συχνότητα στην είσοδο του προγραμματιζόμενου διαιρέτη (prescaler) θα είναι: 48/4 MHz = 12 MHz

Η περίοδος αυτή ονομάζεται κύκλος μηχανής (MC Machine Cycle) και θα είναι: T =1/12MHz=0,08333μs=83,33 ns. **Δηλαδή ένας κύκλος μηχανής είναι 83,33 ns**

Η συχνότητα στην είσοδο του χρονιστή 0 (Timer0), δηλαδή στην έξοδο του prescaler θα είναι: 12*1/64 MHz=0,1875MHz

Αντίστοιχα, η περίοδος στην είσοδο του χρονιστή 0 (Timer0), δηλαδή στην έξοδο του prescaler θα είναι: 83,33 ns X 64 =5333ns

⇨ Θα πρέπει να υπολογισθεί η αρχική τιμή στον timer0 έτσι ώστε **να συμβαίνει υπερχείλιση του timer0 κάθε 10 ms.**

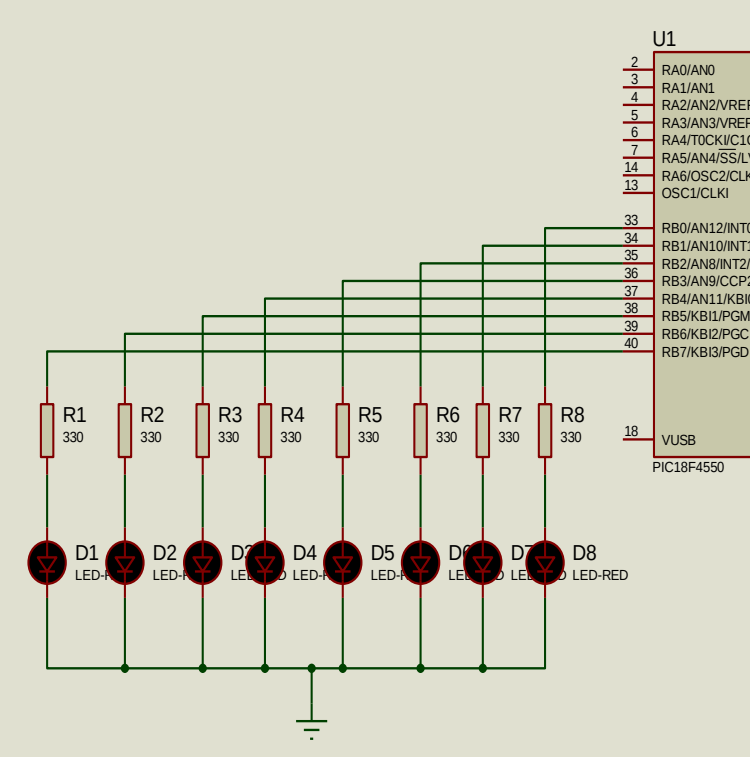
Υπερχείλιση του timer0 σημαίνει μετάβαση από την τιμή FFFF στην τιμή 0000. Ο χρόνος που χρειάζεται ο timer0 για να μεταβεί από την αρχική τιμή που θα του δοθεί μέχρι να γίνει υπερχείλιση (και επομένως διακοπή) θα πρέπει να είναι:

10ms=10 000 μs=10 000 000 ns Υπενθυμίζεται ότι (FFFF)h=(65535)d

Το πλήθος των βημάτων από την αρχική τιμή του Timer0 έως ότου γίνει υπερχείλιση θα είναι: 65536-(Αρχική τιμή του Timer0). Δηλαδή θα πρέπει:
[65536-(Αρχική τιμή του Timer0)]x5333 **ns**= 10 000 000 **ns**.

65536-(Αρχική τιμή του Timer0)=(10 000 000)/5333⇔65536-(Αρχική τιμή του Timer0)=1875⇔

(Αρχική τιμή του Timer0)=65536-1875 =**63661**

(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 PORTB με 8 LEDs  *Figure 1. Διασύνδεση*

Βήμα 2
(5 λεπτά)
	Μελετήστε τον παρακάτω κώδικα
	
	`#include <main.h>`
	`#byte PORTB=0xF81`
	
	`void init (void);`
	`//Δήλωση της ρουτίνας αρχικοποίησης`
	`void timer0_int(void);`
	`//Δήλωση της ρουτίνας διακοπών από τον timer0`
	`int counter1=20;`
	`//Δήλωση μεταβλητής για μέτρηση των διακοπών.`
	`//Στη μεταβλητή αυτή δίνεται η αρχική τιμή 20.`
	
	`void main(){`
	`  init();  // Κλήση της ρουτίνας των αρχικών ρυθμίσεων`
	`  while (TRUE){`
	`  // το κύριο πρόγραμμα δεν κάνει τίποτα. Εκτελεί έναν ατέρμονα βρόχο`
	`  }`
	`}`
	
	`// Αρχή ρουτίνας εξυπηρέτησης της διακοπής`
	`// Οδηγία ότι η επόμενη ρουτίνα είναι η ρουτίνα εξυπηρέτησης της από τον Timer0`
	`#INT_TIMER0`
	`void timer0_int(void){`
	`  //αρχική τιμή του timer0 ώστε η επόμενη διακοπή να συμβεί σε χρόνο ίσο με 10 ms`
	`  set_timer0(63661);`
	`  counter1--;`
	`  if (counter1==0){`
	`  counter1=20;`
	`  PORTB=PORTB^0b11111111;`
	`  // με την λογική πράξη του αποκλειστικού`
	`  //ή (Exclusive OR) ανάμεσα στην PORTB και`
	`  //την τιμή 11111111 αλλάζουμε την κατάσταση`
	`  //όλων των bit της PORTB`
	`  // Κάθε 20 διακοπές αλλάζουν όλα τα bit της πόρτας B`
	`  }`
	`}`
	
	`// Αρχή ρουτίνας αρχικών ρυθμίσεων`
	`void init (void) {`
	`  // Ρύθμιση του προγραμματιζόμενου διαιρέτη στην τιμή 1/64`
	`  SETUP_TIMER_0(T0_INTERNAL | T0_DIV_64 );`
	`  //  Αρχική τιμή του  timer0 ώστε να συμβαίνουν διακοπές κάθε 10 ms`
	`  set_timer0(63661);`
	`  // Ενεργοποίηση της διακοπής από τον timer0`
	`  enable_interrupts(INT_TIMER0);`
	`  // Ενεργοποίηση του γενικού διακόπτη των διακοπών`
	`  enable_interrupts(GLOBAL);`
	
	`  set_tris_b(0x00);`
	`  PORTB=0x00;`
	`}`
	
	`//  Interrupt  κάθε  (65536-63661)*[1/(Fclock/4)]*Prescaler  = 9.994ms`

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

2. Συνάρτηση καθυστέρησης με χρήση του timer0

Σε αυτήν την δραστηριότητα θέλουμε με την χρήση του Timer0 να δημιουργήσουμε μία δική μας συνάρτηση "delay" που να διαρκεί 100μs.

- ⇨ Υπολογισμός της αρχικής τιμής του Timer0 ώστε να συμβαίνουν διακοπές κάθε 100 μs

Η συχνότητα στην είσοδο του προγραμματιζόμενου διαιρέτη (prescaler) θα είναι: 48/4 MHz = 12 MHz

Η περίοδος αυτή ονομάζεται κύκλος μηχανής (MC Machine Cycle) και θα είναι: T =1/12MHz=0,08333μs=83,33 ns. **Δηλαδή ένας κύκλος μηχανής είναι 83,33 ns**

Έστω ότι **δεν** χρησιμοποιούμε προγραμματιζόμενο διαιρέτη! Η περίοδος στην είσοδο του χρονιστή 0 (Timer0) θα είναι: 83,33 ns X 1 =83,33ns

- ⇨ Θα πρέπει να υπολογισθεί η αρχική τιμή στον timer0 έτσι ώστε **να συμβαίνει υπερχείλιση του timer0 κάθε 100 μs.**

Υπερχείλιση του timer0 σημαίνει μετάβαση από την τιμή FFFF στην τιμή 0000. Ο χρόνος που χρειάζεται ο timer0 για να μεταβεί από την αρχική τιμή που θα του δοθεί μέχρι να γίνει υπερχείλιση (και επομένως διακοπή) θα πρέπει να είναι:

100 μs=100 000 ns Υπενθυμίζεται ότι (FFFF)h=(65535)d

Το πλήθος των βημάτων από την αρχική τιμή του Timer0 έως ότου γίνει υπερχείλιση θα είναι: 65536-(Αρχική τιμή του Timer0). Δηλαδή θα πρέπει:
[65536-(Αρχική τιμή του Timer0)]x83,33 **ns**= 100 000 **ns**.

65536-(Αρχική τιμή του Timer0)=(100 000)/83,33⇔

⇔ (Αρχική τιμή του Timer0) = 65536-1200 =**64336**

(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`
	
	`//δηλώσεις συναρτήσεων`
	`void init (void);`
	`void timer0_int(void);`
	`//συνάρτηση που με χρήση Timer0 θα μετράει 100μs`

```
void mydelay_100us(int);

int32 counter_time=0;
//Δήλωση μεταβλητής για μέτρηση των διακοπών.
//Στη μεταβλητή αυτή δίνεται η αρχική τιμή 0.
//Θα αυξάνεται κατά 1 κάθε 100 μs
int32 counter_time_old=0;
//Μεταβλητή που χρησιμοποιείται στην αυτοσχέδια ρουτίνα
καθυστέρησης.
int32 aaa=1;
// ακέραιη μεταβλητή που χρησιμοποιείται σαν όρισμα στην
//αυτοσχέδια ρουτίνα καθυστέρησης

// Κύριο πρόγραμμα
void main(){
   init();     // Κλήση της ρουτίνας των αρχικών ρυθμίσεων
   while (TRUE){
      PORTB=PORTB^0b11111111;
      mydelay_100us(500);
      //καθυστέρηση 500X100μs=50 000 μs= 50 ms
   }
}


// Αρχή ρουτίνας εξυπηρέτησης της διακοπής
#INT_TIMER0
void timer0_int(void) {
   set_timer0(64336);
   //αρχική τιμή του timer0 ώστε η επόμενη διακοπή να συμβεί
σε
   // χρόνο ίσο με 100 μs
   counter_time++;
   // Ο μετρητής διακοπών αυξάνεται κατά 1 κάθε 100 μs
   //  Χρησιμοποιείται για να μετράμε καθυστέρηση σε
   // πολλαπλάσια των 100 μs
}


// Αρχή ρουτίνας αρχικών ρυθμίσεων
void init (void) {
   SETUP_TIMER_0(T0_INTERNAL | T0_DIV_1 );
   // Ρύθμιση του προγραμματιζόμενου διαιρέτη(prescaler) στην
τιμή 1
   set_timer0(64336);
   //  Αρχική τιμή του timer0 ώστε να συμβαίνουν διακοπές κάθε
100 μs
   enable_interrupts(INT_TIMER0);
   // Ενεργοποίηση της διακοπής από τον timer0
   enable_interrupts(GLOBAL);
   // Ενεργοποίηση του γενικού διακόπτη των διακοπών

   set_tris_b(0x00);
   PORTB=0x00;
}

//Κώδικας  ρουτίνας καθυστέρησης aaax100μs ----------------------
void mydelay_100us(aaa){
   counter_time_old=counter_time;
   // Ο counter_time_old παίρνει την τιμή του χρόνου
   // την στιγμή που μπαίνουμε στην ρουτίνα
   while(counter_time < counter_time_old+aaa) {  }
   //Αναμονή έως ότου ο μετρητής χρόνου φτάσει
   // την τιμή του χρόνου τη στιγμή που μπαίνουμε στην ρουτίνα
+ aaa
}
```

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