## Άσκηση 6 Δ. Συνάρτηση καθυστέρησης με χρήση του timer0

1. **Να ρυθμιστεί η τιμή του Prescaler και του timer0 έτσι ώστε να συμβαίνουν διακοπές κάθε 100 μs.**
2. **Να κατασκευαστεί συνάρτηση** **mydelay\_100us(a) η οποία να προκαλεί καθυστέρηση ax100μs.**
3. **Να γραφεί πρόγραμμα που να χρησιμοποιεί την παραπάνω συνάρτηση mydelay\_100us(a) και να αναβοσβήνει τα LED που είναι συνδεδεμένα στην πόρτα B κάθε 100 ms (T/2= 50 ms).**

Ταλαντωτής

Εσωτερικός διαιρέτης /4

Προγραμματιζόμενος Διαιρέτης (Prescaler)

Χρονιστής 0

(Timer 0)

Λόγος διαίρεσης από

1/2, 1/4, 1/8, 1/16, 1/32, 1/64, /1/128, 1/256

Διαίρεση 1/1…1/65535

Προαιρετικός

IRQ

Fclock=48 MHz

Τ κύκλος μηχανής(Machine Cycle)

Interrupt Request

Αίτημα διακοπής



Κύκλωμα άσκησης 6Δ (Αναβόσβημα των LED κάθε 50 ms με την **αυτοσχέδια συνάρτηση** καθυστέρησης mydelay\_ms(a)

**Λύση:**

Α. Υπολογισμός της αρχικής τιμής του Timer0 ώστε να συμβαίνουν διακοπές κάθε 100 μs

Η συχνότητα στην είσοδο του προγραμματιζόμενου διαιρέτη, δηλαδή στην έξοδο του εσωτερικού διαιρέτη θα είναι:

Η συχνότητα στην έξοδο του εσωτερικού διαιρέτη θα είναι:

fέξοδος εσωτερικού διαιρέτη=

Η περίοδος στην έξοδο του εσωτερικού διαιρέτη ονομάζεται κύκλος μηχανής(MC Machine Cycle) και θα είναι:

Tέξοδος εσωτερικού διαιρέτη =0,08333μs=83,33 ns

Δηλαδή ένας κύκλος μηχανής είναι 83,33 ns

Έστω ότι δεν χρησιμοποιούμε προγραμματιζόμενο διαιρέτη.

Η περίοδος στην είσοδο του timer θα είναι 83,33 ns

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

Υπερχείλιση του timer0 σημαίνει μετάβαση από την τιμή FFFF στην τιμή 0000.

O 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 000ns

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.

Υπολογίζουμε την αρχική τιμή του timer0.

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

**Πρόγραμμα.**

#include <main.h>

#byte PORTB =0xF81 // καθορισμός του καταχωρητή δεδομένων της

//πόρτας Β

void init (void); // Δήλωση της ρουτίνας αρχικοποίησης.

void timer0\_int(void); //Δήλωση της ρουτίνας διακοπών από τον timer0

void mydelay\_100us(int); // Δήλωση της αυτοσχέδιας ρουτίνας καθυστέρησης ax100 μs

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() { // Ανοίγει η αγκύλη της main

init(); // Κλήση της ρουτίνας των αρχικών ρυθμίσεων

while (TRUE){

} // το κύριο πρόγραμμα δεν κάνει τίποτα. Εκτελεί έναν ατέρμονα βρόχο

} // κλείνει η αγκύλη του main

// Τέλος κύριου προγράμματος

// Αρχή ρουτίνας εξυπηρέτησης της διακοπής…………………………………………..

#INT\_TIMER0 // Οδηγία ότι η επόμενη ρουτίνα είναι η ρουτίνα εξυπηρέτησης της

// από τον 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++ // O μετρητής διακοπών αυξάνεται κατά 1 κάθε 100 μs

// Χρησιμοποιείται για να μετράμε καθυστέρηση σε

// πολλαπλάσια των 100 μs

}

// Τέλος ρουτίνας εξυπηρέτησης της διακοπής………………………………………

// 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); //Η πόρτα Β γίνεται έξοδος

PORTB=0x00; //Αρχική τιμή 0 στην πόρτα Β

}

// Τέλος ρουτίνας αρχικών ρυθμίσεων………………………………………………….

// Ορισμός ρουτίνας καθυστέρησης aaax100μs

void mydelay\_100us(aaa) {

counter\_time\_old=counter\_time; // O 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

}

//τέλος της ρουτίνας καθυστέρησης aaa x 100 μs