## Άσκηση 6 Γ. Ασκήσεις πάνω στους χρονιστές(Timers)

**Να υπολογισθεί η περίοδος του κύκλου μηχανής στο παρακάτω διάγραμμα και η αρχική τιμή που πρέπει να δίνεται στον Timer0 έτσι ώστε να εκτελούνται διακοπές κάθε 50 ms. Η συχνότητα στην έξοδο του κρυσταλλικού ταλαντωτή είναι 48 MHz. Ο προγραμματιζόμενος διαιρέτης να τεθεί στην τιμή .**

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

**Στη συνέχεια να γραφεί πρόγραμμα με ρουτίνα διακοπών από τον Timer0 με το οποίο αναβοσβήνουν τα LED τα οποία συνδέονται στην πόρτα Β ως εξής:**

**To LED0 κάθε 100 ms To LED1 κάθε 150 ms To LED2 κάθε 200 ms Το LED3 κάθε 300 ms**

*Υπόδειξη: Θα χρησιμοποιήσετε 8 μετρητές διακοπών, counter0, counter1, counter2, counter3, counter4, counter5, counter6, counter7 που θα παίρνουν αρχικές τιμές 2, 3, 4, 6.*

*Για παράδειγμα το LED4 θα αλλάζει κατάσταση μετά από 6 διακοπές ( 6 Χ 50ms = 300 ms ).*

*Οι μετρητές διακοπών σε κάθε διακοπή, δηλαδή κάθε 50 ms, θα ελαττώνονται κατά 1. Όταν φθάνουν στην τιμή 0 το αντίστοιχο LED θα αλλάζει κατάσταση και οι μετρητές διακοπών θα παίρνουν και πάλι την αρχική τους τιμή.*



Κύκλωμα άσκησης 6B (Τα LED αναβοσβήνουν, το καθένα σε διαφορετικό χρόνο. Οι χρόνοι που αναβοσβήνουν είναι πολλαπλάσια του χρόνου διακοπής από τον timer0, ο οποίος είναι 50 ms)

Α. Υπολογισμός της αρχικής τιμής του Timer0 ώστε να συμβαίνουν διακοπές κάθε 50 ms

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

Η συχνότητα στην είσοδο του χρονιστή 0(Timer0), δηλαδή στην έξοδο του προγραμματιζόμενου διαιρέτη θα είναι:

fέξοδος προγραμματιζόμενου διαιρέτη=

*στην είσοδο του προγραμματιζόμενου διαιρέτη(prescaler).*

Η περίοδος στην είσοδο του χρονιστή 0(Timer0), δηλαδή στην έξοδο του προγραμματιζόμενου διαιρέτη θα είναι:

T έξοδος προγραμματιζόμενου διαιρέτη=83,33 ns X 32 =2666ns

Μπορεί να υπολογισθεί και όπως παρακάτω

Τέξοδος προγραμματιζόμενου διαιρέτη

Θα πρέπει να υπολογισθεί η αρχική τιμή στον timer0 έτσι ώστε να συμβαίνει υπερχείλιση του timer0 κάθε 50 ms.

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:

50ms=50 000 μs=50 000 000 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)]x2666 ns= 50 000 000 ns.

Υπολογίζουμε την αρχική τιμή του timer0.

65536-(Αρχική τιμή του Timer0)=

**(Αρχική τιμή του Timer0)=65536-18755 =46781**

**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

int counter0=2; //Δήλωση μεταβλητών για μέτρηση των διακοπών

int counter1=3; //Καθορίζονται οι αρχικές τους τιμές

int counter2=4;

int counter3=6;

// 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

// 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(46781); //αρχική τιμή του timer0 ώστε η επόμενη διακοπή να συμβεί σε

// χρόνο ίσο με 50 ms

counter0--; counter1--;counter2--;counter3--;

if (counter0==0){

counter0=2;

PORTB=PORTB^0b00000001; // με την λογική πράξη του αποκλειστικού

//ή (Exclusive OR) ανάμεσα στην PORTB και

//την τιμή 00000001 αλλάζουμε την κατάσταση

//του bit RB0 της PORTB

// Κάθε 2 διακοπές αλλάζει όλα τα bit RB0 της πόρτας Β

}

if (counter1==0){

counter1=3;

PORTB=PORTB^0b00000010; // με την λογική πράξη του αποκλειστικού

//ή (Exclusive OR) ανάμεσα στην PORTB και

//την τιμή 00000010 αλλάζουμε την κατάσταση

//του bit RB1 της PORTB

// Κάθε 3 διακοπές αλλάζει όλα τα bit RB1 της πόρτας Β

}

if (counter2==0){

counter2=4;

PORTB=PORTB^0b00000100; // με την λογική πράξη του αποκλειστικού

//ή (Exclusive OR) ανάμεσα στην PORTB και

//την τιμή 000000100 αλλάζουμε την κατάσταση

//του bit RB2 της PORTB

// Κάθε 4 διακοπές αλλάζει όλα τα bit RB2 της πόρτας Β

}

if (counter3==0){

counter3=6;

PORTB=PORTB^0b00001000; // με την λογική πράξη του αποκλειστικού

//ή (Exclusive OR) ανάμεσα στην PORTB και

//την τιμή 00001000 αλλάζουμε την κατάσταση

//του bit RB3 της PORTB

// Κάθε 6 διακοπές αλλάζει όλα τα bit RB3 της πόρτας Β

}

} // κλείνει η αγκύλη της ρουτίνας εξυπηρέτησης της

// διακοπής

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

// Αρχή ρουτίνας αρχικών ρυθμίσεων…………………………………………………

void init (void) {

SETUP\_TIMER\_0(T0\_INTERNAL | T0\_DIV\_32 );// Ρύθμιση του προγραμματιζόμενου

// διαιρέτη στην τιμή

set\_timer0(46781); // Αρχική τιμή του timer0

// ώστε να συμβαίνουν διακοπές

// κάθε 50 ms

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………………………………………………….