INTERNATIONAL HELLENIC UNIVERSITY

Διεθνές Πανεπιστήμιο της Ελλάδος

Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων

Ενσωματωμένα Συστήματα

(6ο εξάμηνο)

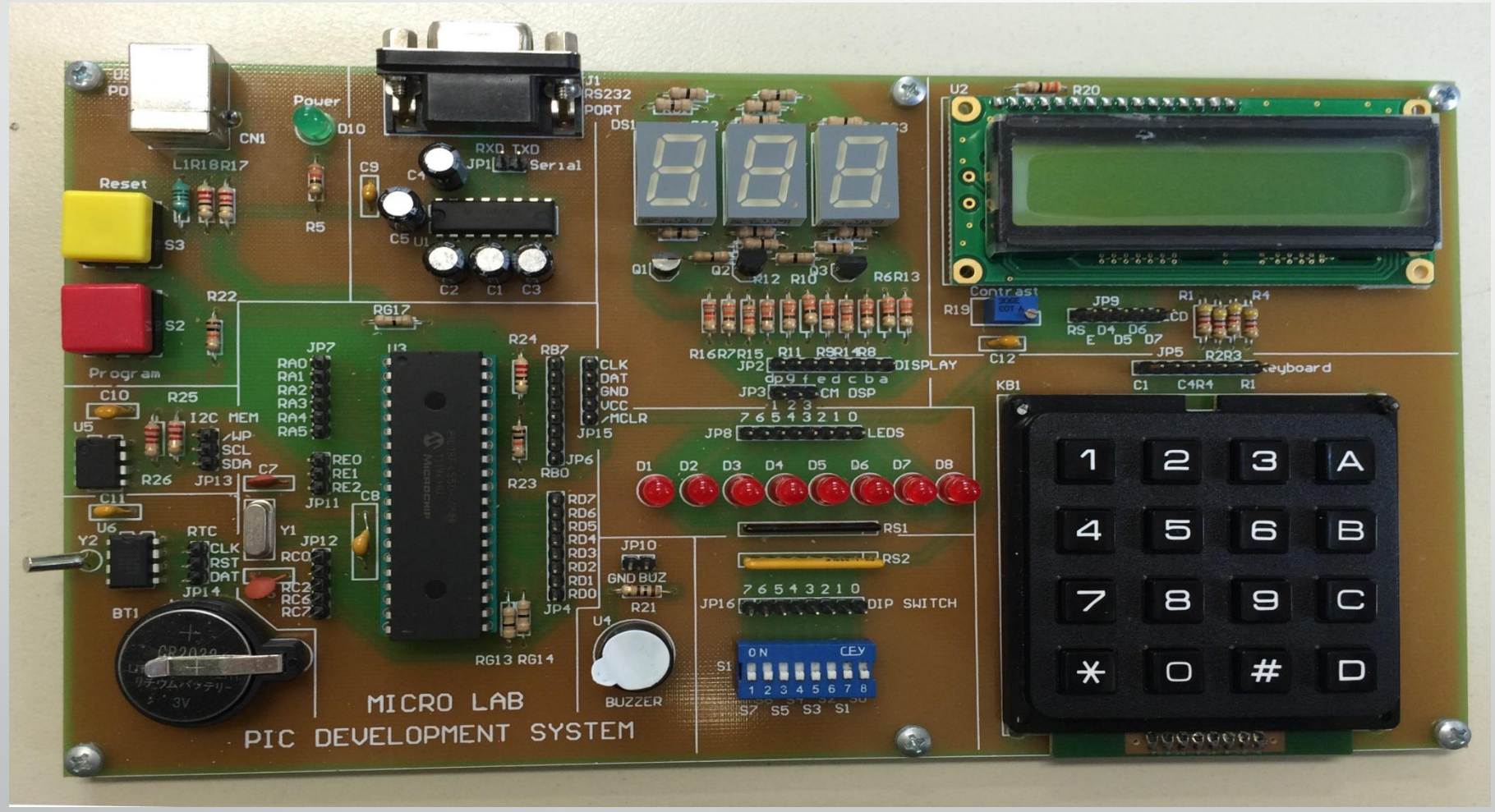
07-Χρονιστές - Timers

Διδάσκουσα: Παπαδοπούλου Μαρία

Επίκουρη Καθηγήτρια

Θεσσαλονίκη 2025

Πλακέτα ανάπτυξης εφαρμογών



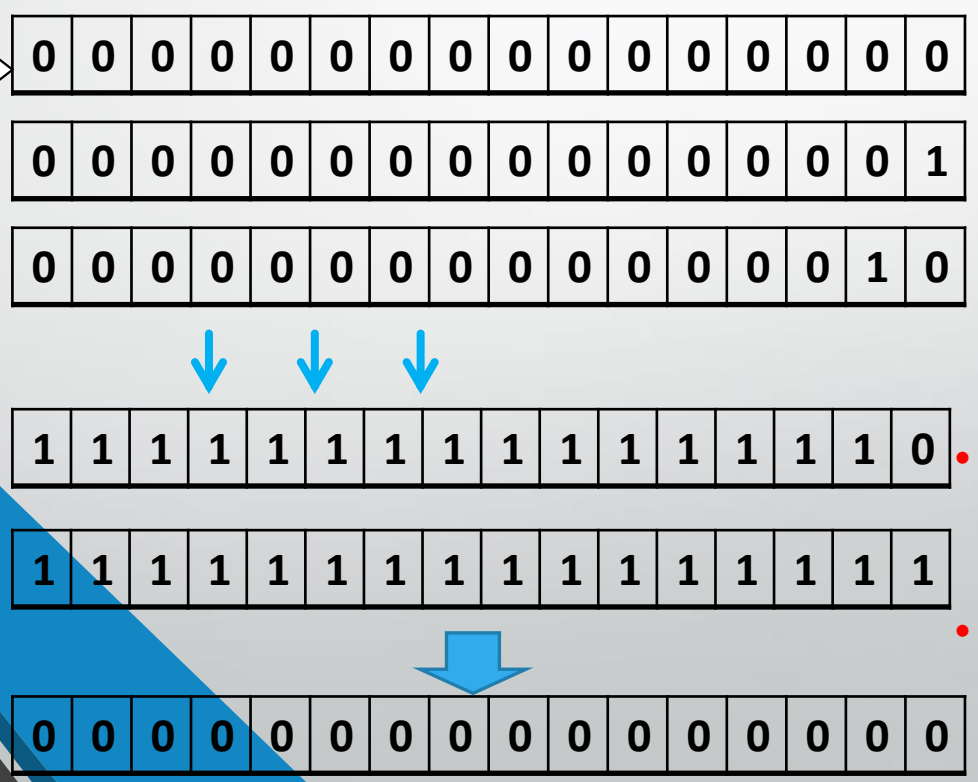
Χρονιστής - Timer

- Τι είναι οι Timers (Χρονιστές) του μικροελεγκτή;

  Είναι καταχωρητές των 8 bit ή των 16 bit των οποίων το περιεχόμενο αυξάνει ανά συγκεκριμένο χρονικό διάστημα.

Κάθε παλμός αυξάνει το περιεχόμενο του timer κατά 1

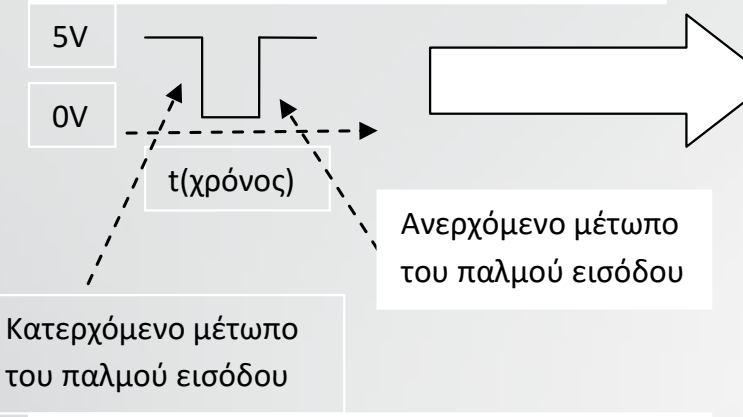
Στο πρόγραμμα μας δηλώνουμε αν θέλουμε το περιεχόμενο του timer να αυξάνεται κατά το ανερχόμενο ή το κατερχόμενο μέτωπο του παλμού εισόδου.



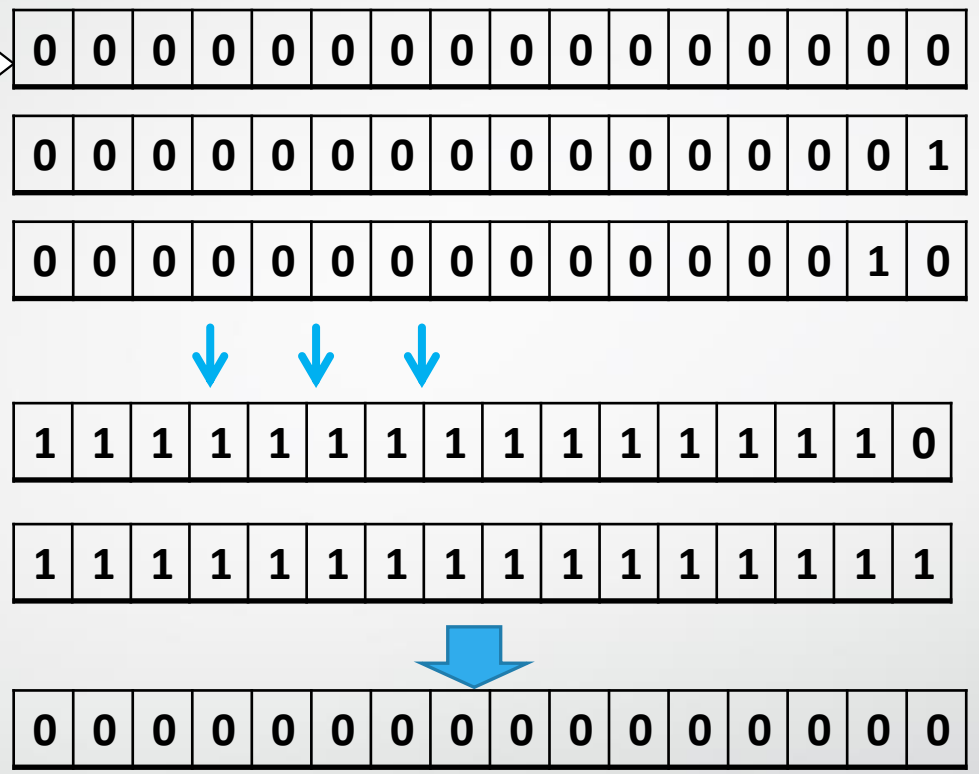
- **Προσοχή, μετά την τιμή 111 … 111 (= 65535) ο timer παίρνει την τιμή 000 … 000**
- **Σε αυτήν τη μετάβαση μπορούμε να προγραμματίσουμε να συμβαίνει μια διακοπή (interrupt)**

Χρονιστής - Timer

Κάθε παλμός αυξάνει το περιεχόμενο του timer κατά 1



Στο πρόγραμμα μας δηλώνουμε αν θέλουμε το περιεχόμενο του timer να αυξάνεται κατά το ανερχόμενο ή το κατερχόμενο μέτωπο του παλμού εισόδου.



Ερώτηση:

Από πού προέρχονται οι παλμοί που αυξάνουν την τιμή του Timer;

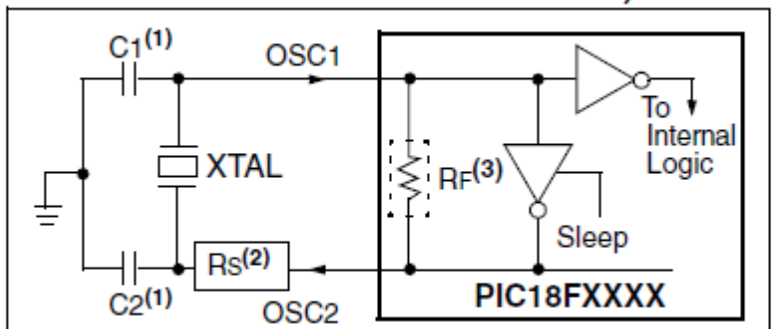
Απάντηση:

- Μπορεί να προέρχονται από μια εξωτερική πηγή και να εφαρμόζονται σε έναν συγκεκριμένο ακροδέκτη του μικροελεγκτή
- Μπορεί να προέρχονται από ένα κύκλωμα στο εσωτερικό του μικροελεγκτή, το οποίο ονομάζεται εσωτερικός ταλαντωτής (Oscillator)
- Στο πρόγραμμα, με μια εντολή, δηλώνουμε την πηγή των παλμών που θέλουμε να χρησιμοποιηθεί

Τι είναι ο εσωτερικός ταλαντωτής (internal oscillator) του μΕ;

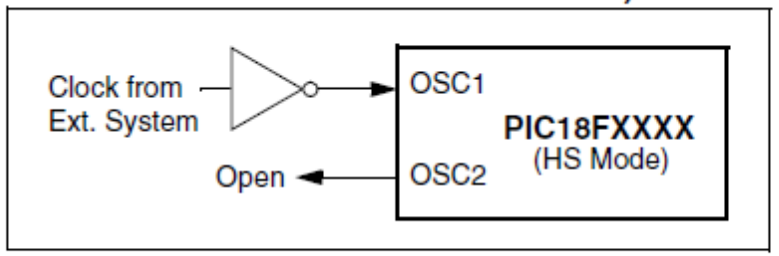
Manual PIC18F4550, σελίδα 25

FIGURE 2-2: CRYSTAL/CERAMIC RESONATOR OPERATION (XT, HS OR HSPLL CONFIGURATION)



Note 1: See Table 2-1 and Table 2-2 for initial values of C1 and C2.
2: A series resistor (RS) may be required for AT strip cut crystals.
3: RF varies with the oscillator mode chosen.

FIGURE 2-3: EXTERNAL CLOCK INPUT OPERATION (HS OSC CONFIGURATION)

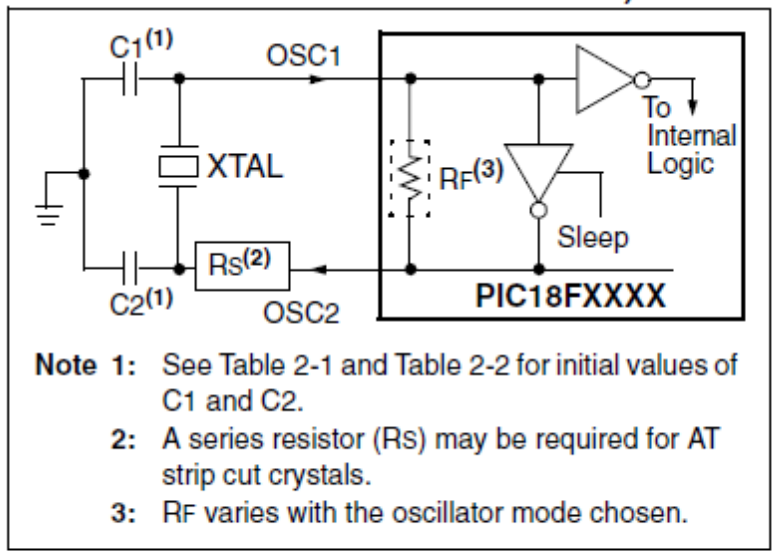


- Μπορεί να χρησιμοποιηθεί και εξωτερικός ταλαντωτής για την παραγωγή παλμών (Clock from External System)

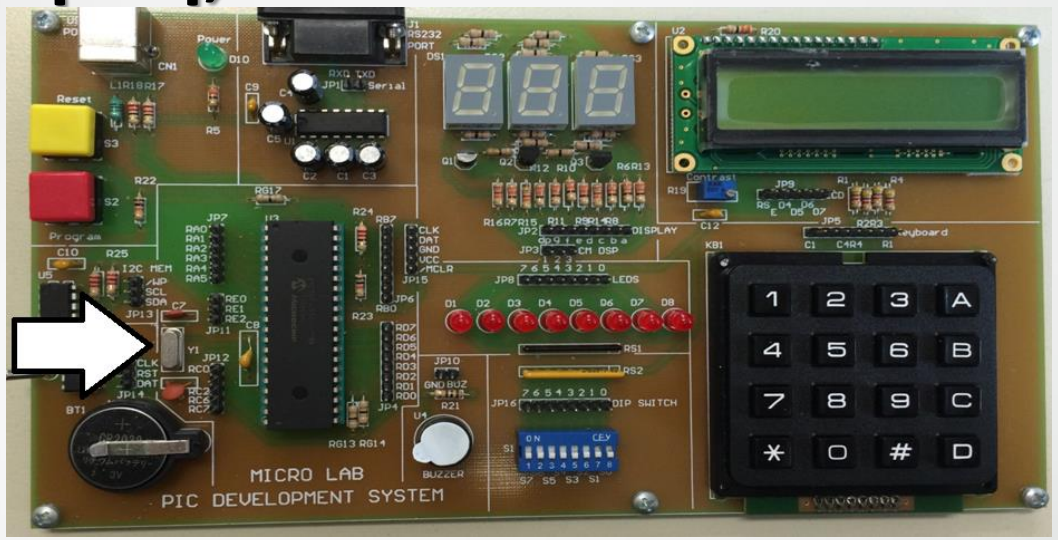
- Ο ταλαντωτής είναι κύκλωμα στο εσωτερικό του μΕ, το οποίο παράγει τετραγωνικούς παλμούς. Αυτοί οι παλμοί χρησιμοποιούνται για τη λειτουργία των λογικών κυκλωμάτων του μΕ
- Το κύκλωμα αυτό χρησιμοποιεί ένα εξάρτημα έξω από το μΕ, το οποίο λέγεται κρύσταλλος (XTAL)

Τι είναι ο κρύσταλλος (quartz) που χρησιμοποιείται στον ταλαντωτή του μικροελεγκτή;

FIGURE 2-2: CRYSTAL/CERAMIC RESONATOR OPERATION (XT, HS OR HSPLL CONFIGURATION)



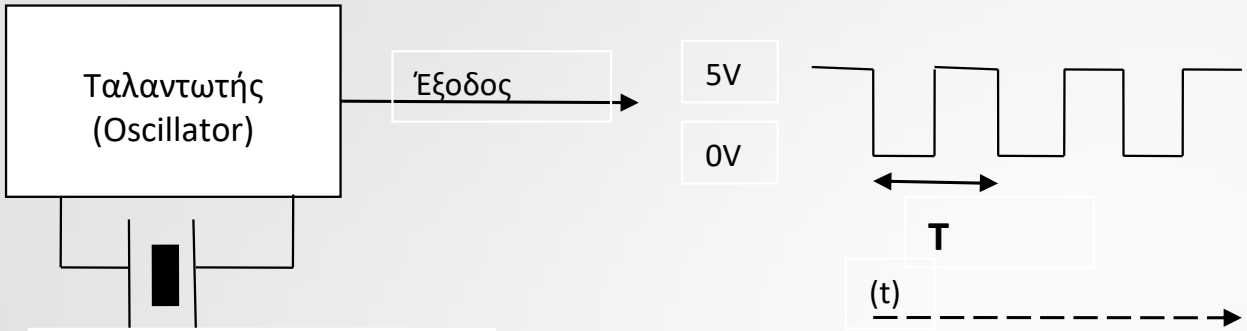
Note 1: See Table 2-1 and Table 2-2 for initial values of C1 and C2.
2: A series resistor (RS) may be required for AT strip cut crystals.
3: RF varies with the oscillator mode chosen.



- Το βέλος δείχνει τον εξωτερικό κρύσταλλο (quartz)
- Δίπλα στον κρύσταλλο φαίνονται, σαν φακές, οι πυκνωτές C1 και C2

- Ο κρύσταλλος (quartz) είναι ηλεκτρονικό εξάρτημα το οποίο χρησιμοποιείται σε κάποιο κύκλωμα το οποίο λέγεται **ταλαντωτής κρυστάλλου**
- **Ο ταλαντωτής κρυστάλλου παράγει παλμούς εξαιρετικά σταθερής συχνότητας** π.χ. συχνότητας F=20,237123 MHz
- **Χωρίς κρύσταλλο (quartz) δεν μπορεί να παραχθεί σταθεροποιημένη συχνότητα**
- Οι ταλαντωτές κρυστάλλου χρησιμοποιούνται και σε όλα τα ηλεκτρονικά ρολόγια. Αποτελούν τη λεγόμενη πηγή χρόνου

Τι είναι ο ταλαντωτής κρυστάλλου

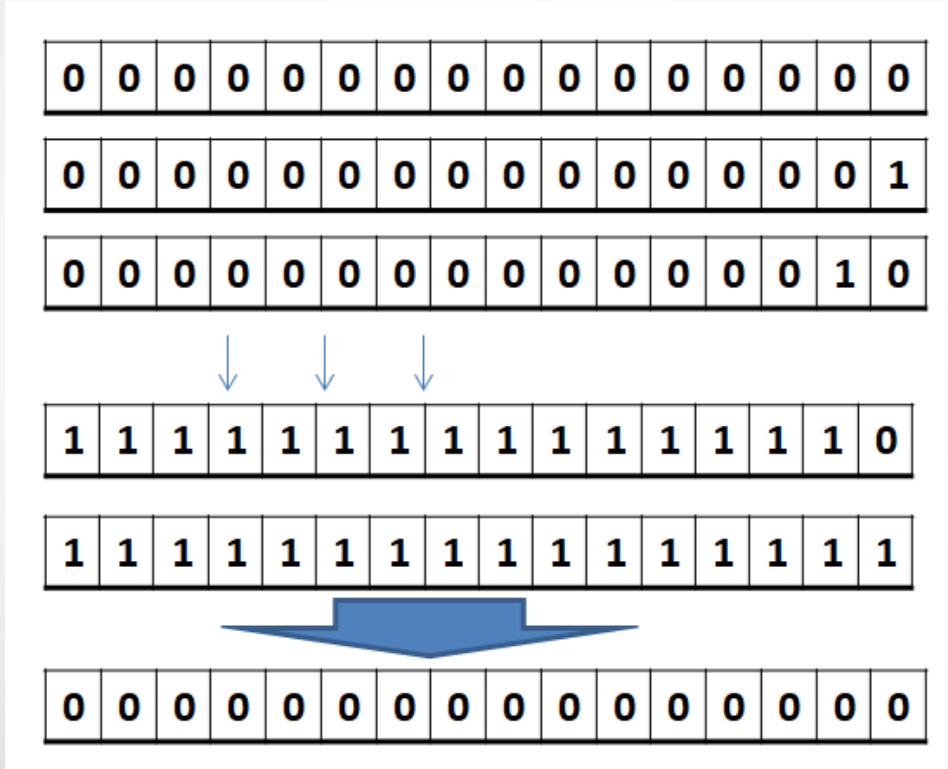


$$f = \frac{1}{T}$$

- Ο ταλαντωτής είναι ένα κύκλωμα το οποίο παράγει μια περιοδική κυματομορφή, όπως για παράδειγμα τους τετραγωνικούς παλμούς τάσης του σχήματος
- Η τετραγωνική αυτή περιοδική κυματομορφή χρησιμοποιείται στους μικροεπεξεργαστές για τον συγχρονισμό της λειτουργίας των λογικών κυκλωμάτων τους
- Η έννοια του ταλαντωτή είναι γενική και περιλαμβάνει και τα κυκλώματα τα οποία παράγουν ημιτονοειδείς κυματομορφές τάσης, όπως αυτές που χρησιμοποιούνται στις τηλεπικοινωνίες

Πού χρησιμοποιούνται οι timers (ρολόγια) του μικροελεγκτή;

1. Για να μετράμε χρόνο
2. Για να προγραμματίσουμε τον μικρολεγκτή να εκτελέσει μια συγκεκριμένη λειτουργία σε μια χρονική στιγμή
3. Για να προγραμματίσουμε, ανά ορισμένα χρονικά διαστήματα, το μικροελεγκτή να εκτελεί μια συγκεκριμένη λειτουργία, π.χ. να μετράει και να καταγράφει τη θερμοκρασία κάθε 10 sec



- Μπορούμε να προγραμματίσουμε ώστε σε κάθε μετάβαση από 111…111 σε 000…000 να εκτελείται μια ρουτίνα διακοπής από τον Timer (Timer Interrupt)
- Αυτή η διακοπή είναι μια **εσωτερική διακοπή** (Internal Interrupt) γιατί προκαλείται από το εσωτερικό του μικροελεγκτή και όχι από ένα σήμα που εφαρμόζεται σε έναν εξωτερικό ακροδέκτη του μικροελεγκτή

Πόσα ρολόγια (timers) έχει ο Μικρολεγκτής PIC 18F4550;

Timer0 module
Timer1 module
Timer2 module
Timer3 module

- Ο καθένας από αυτούς του τέσσερις Timers έχει διαφορετικά υποσυστήματα για ειδικές εφαρμογές.

Timer0 module

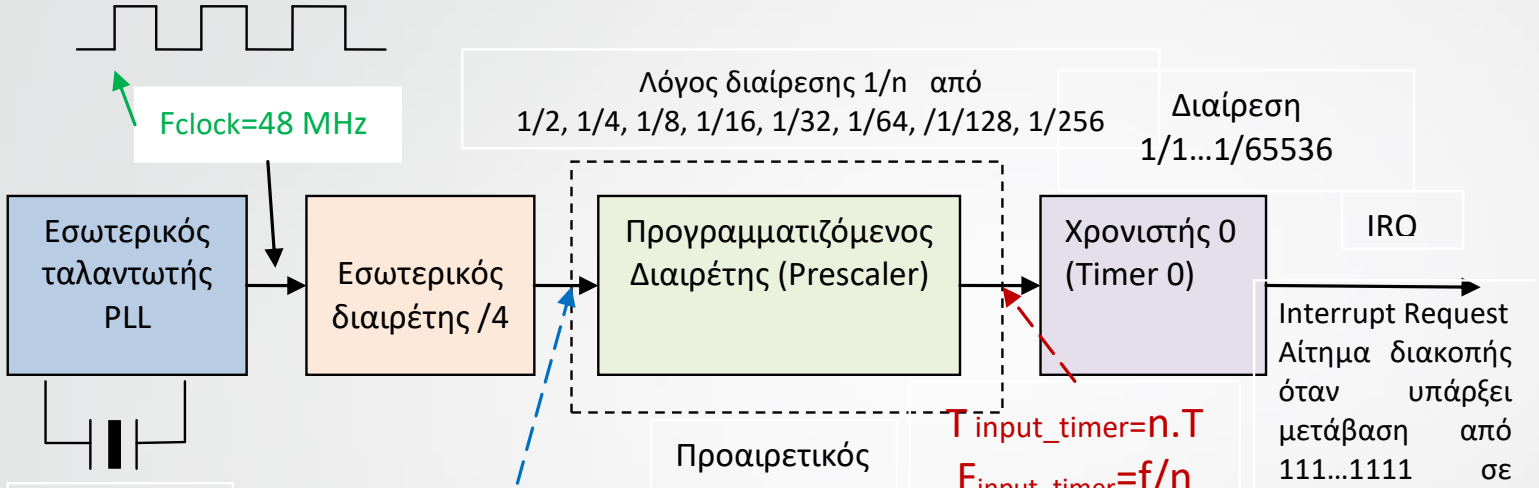
module → υποσύστημα, ενότητα

- Ο Timer0 είναι ένας timer των 8 bit ή των 16 bit
- Με μια εντολή ρυθμίζουμε πώς θα χρησιμοποιηθεί ο timer0
- Εάν θέλουμε να χρησιμοποιηθεί σαν timer των 8 bit θα πρέπει να συμπληρώσουμε τη σχετική παράμετρο στην εντολή ρύθμισης του timer0

setup_timer_0(T0_INTERNAL|T0_DIV_256|**T0_8_BIT**);

Αναζήτηση στο manual του CCS C Compiler

Με τι ρυθμό αυξάνεται το περιεχόμενο του timer0;



T κύκλος μηχανής(Machine Cycle)

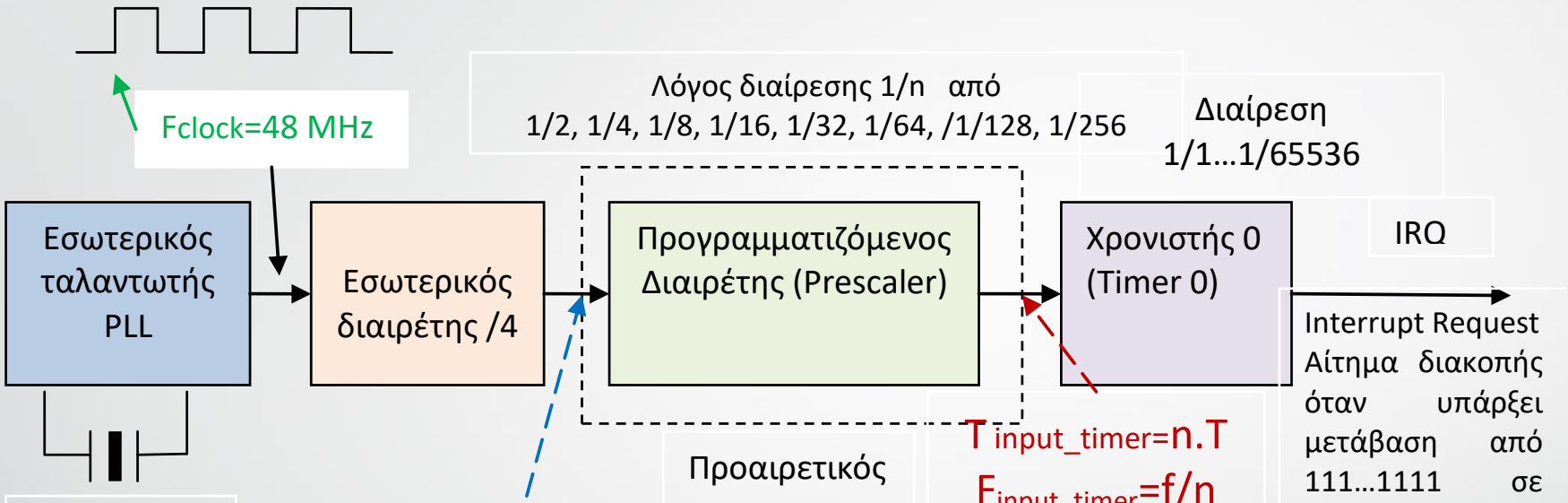
$$T = \frac{1}{12MHz} = \frac{1}{12x10^6 Hz} = 0{,}08333x10^{-6} = 83{,}33x10^{-9} = 83{,}33\ ns$$

Ο κύκλος μηχανής MC είναι 83,33 ns MC → Machine Cycle

$f = \frac{1}{T}$ =12 MHz συχνότητα στην είσοδο του προγραμματιζόμενου διαιρέτη (prescaler)

- Όταν δεν χρησιμοποιείται ο προγραμματιζόμενος διαιρέτης, ο timer0 αυξάνει κατά 1 κάθε 83,33 ns
- Όταν ο προγραμματιζόμενος διαιρέτης τεθεί στην τιμή ½, ο timer0 αυξάνει κατά 1 κάθε 2 x 83,33 ns =166,66 ns
- Όταν ο προγραμματιζόμενος διαιρέτης τεθεί στην τιμή 1/16, ο timer0 αυξάνει κατά 1 κάθε 16 x 83,33 ns =1333,28 ns = 1,33328 μs
- Κ.Ο.Κ.

Πώς ρυθμίζεται η τιμή του προγραμματιζόμενου διαιρέτη συχνότητας;



T κύκλος μηχανής(Machine Cycle)

$$T = \frac{1}{12MHz} = \frac{1}{12x10^6 Hz} = 0{,}08333x10^{-6} = 83{,}33x10^{-9} = 83{,}33\ ns$$

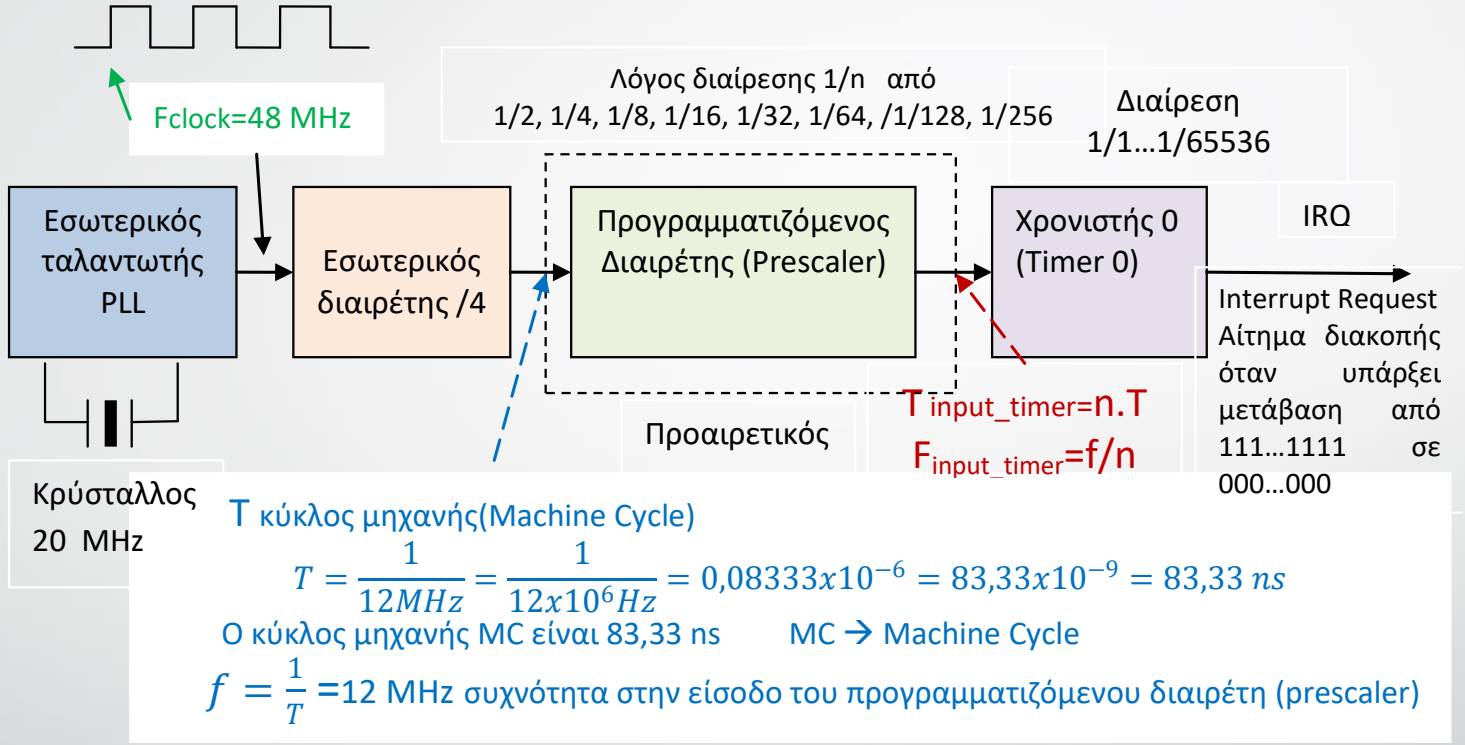
Ο κύκλος μηχανής MC είναι 83,33 ns MC → Machine Cycle

$f = \frac{1}{T}$ =12 MHz συχνότητα στην είσοδο του προγραμματιζόμενου διαιρέτη (prescaler)

setup_timer_0(T0_INTERNAL|**T0_DIV_256**);

Κάθε πότε θα συμβαίνουν διακοπές, αν θέσουμε τον προγραμματιζόμενο διαιρέτη συχνότητας στην τιμή 1/256;

setup_timer_0(T0_INTERNAL|**T0_DIV_256**);

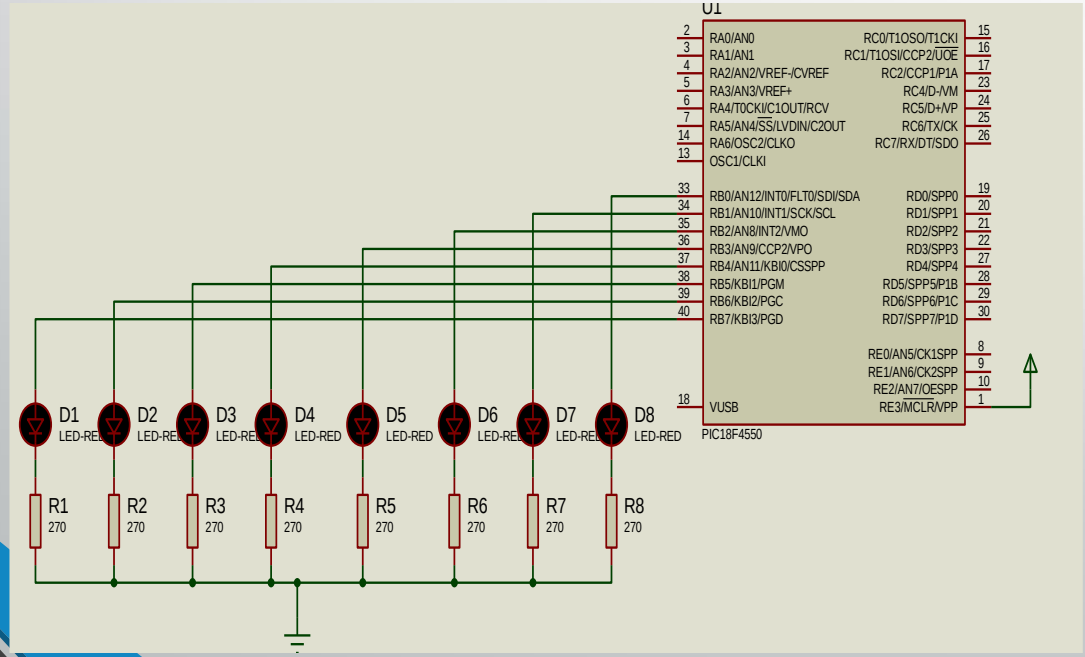


- Θα συμβαίνουν διακοπές κάθε: 83,33ns x 256 x 65536 = 1398045409 ns = 1,398 s
- **Πώς ενεργοποιούμε τις διακοπές από τον Timer0;** ***Απάντηση στο manual του CCS***

```
enable_interrupts(GLOBAL);          // Ενεργοποίηση του γενικού διακόπτη των διακοπών
enable_interrupts(INT_TIMER0);      // Ενεργοποίηση της διακοπής από τον timer0
```

Παράδειγμα άσκησης με διακοπές από τον timer0

- Να γραφεί πρόγραμμα στο οποίο είναι ενεργοποιημένες οι διακοπές από τον timer0 σε λειτουργία 16 bit.
- Ο προγραμματισμένος διαιρέτης συχνότητας να τοποθετηθεί στην τιμή 1/16.
- Η πόρτα Β να χρησιμοποιείται σαν έξοδος, και κάθε φορά που συμβαίνει μια διακοπή το περιεχόμενο της να αυξάνει κατά 1. Η αρχική τιμή της πόρτας Β να είναι 0.
- Να υπολογισθεί ανά πόσο χρονικό διάστημα θα αυξάνεται το περιεχόμενο της πόρτας Β κατά 1.

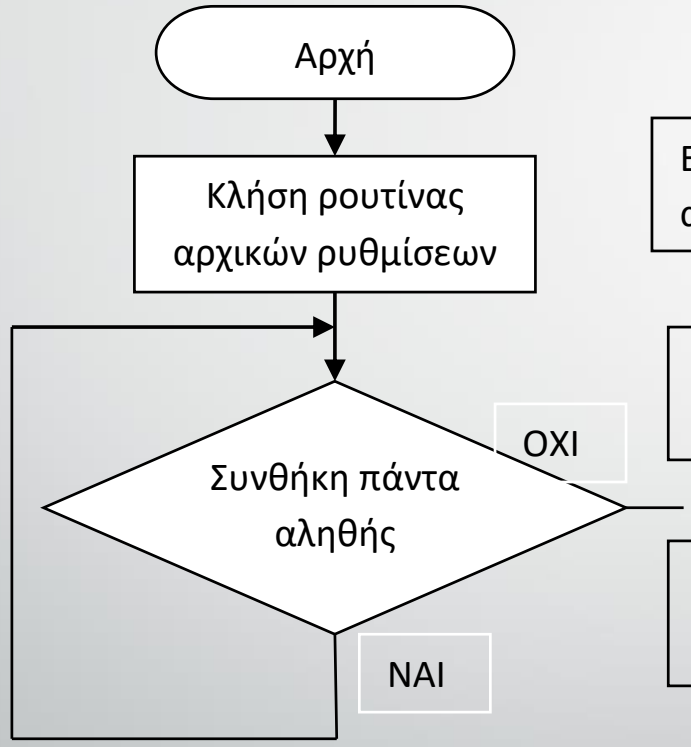


Δομή του προγράμματος

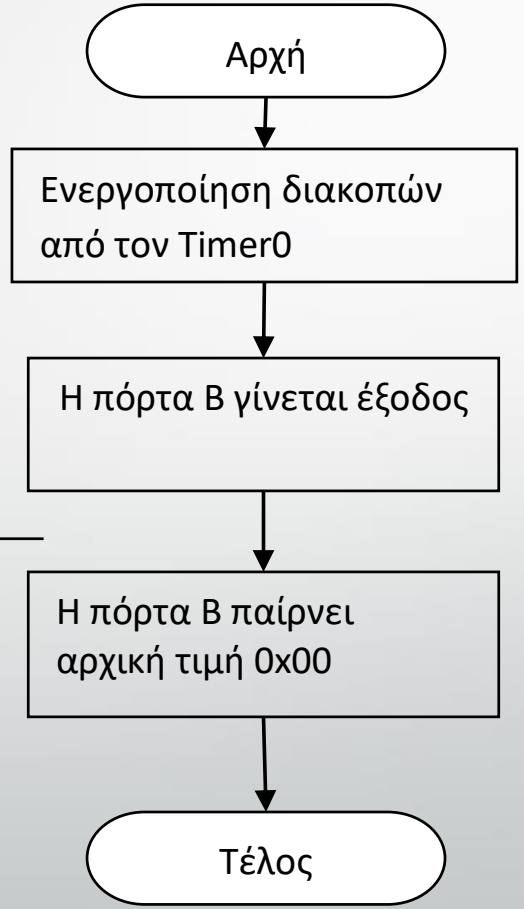
1. Κύριο πρόγραμμα **main(){ }**
   Δεν κάνει τίποτα
2. Πρόγραμμα αρχικών ρυθμίσεων **init(){ }**
   Πόρτες εισόδου εξόδου, ρυθμίσεις διακοπών από τον timer0
3. Ρουτίνα διακοπών από **timer0_int(void){ }**
   Αυξάνει το περιεχόμενο της πόρτας Β κατά 1

Διάγραμμα ροής

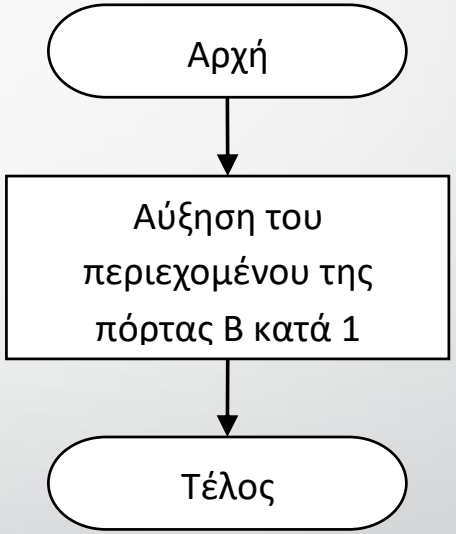
Κύριο πρόγραμμα



Ρουτίνα αρχικών ρυθμίσεων



Ρουτίνα διακοπών από τον timer0



Πρόγραμμα (κύριο πρόγραμμα)

```
#include <main.h>
#byte PORTB=0xF81 // Καθορισμός του καταχωρητή δεδομένων της πόρτας B

void init (void);           //Δήλωση της ρουτίνας αρχικοποίησης
void timer0_int(void);      //Δήλωση της ρουτίνας διακοπών από τον timer0

    // *** Κύριο πρόγραμμα ***
void main() {       // Ανοίγει η αγκύλη της main
    init();         // Κλήση της ρουτίνας των αρχικών ρυθμίσεων
    while (TRUE){   // Το κύριο πρόγραμμα δεν κάνει κάτι. Εκτελεί έναν ατέρμονα βρόχο
    }
}   // *** Κλείνει η αγκύλη main ***
```

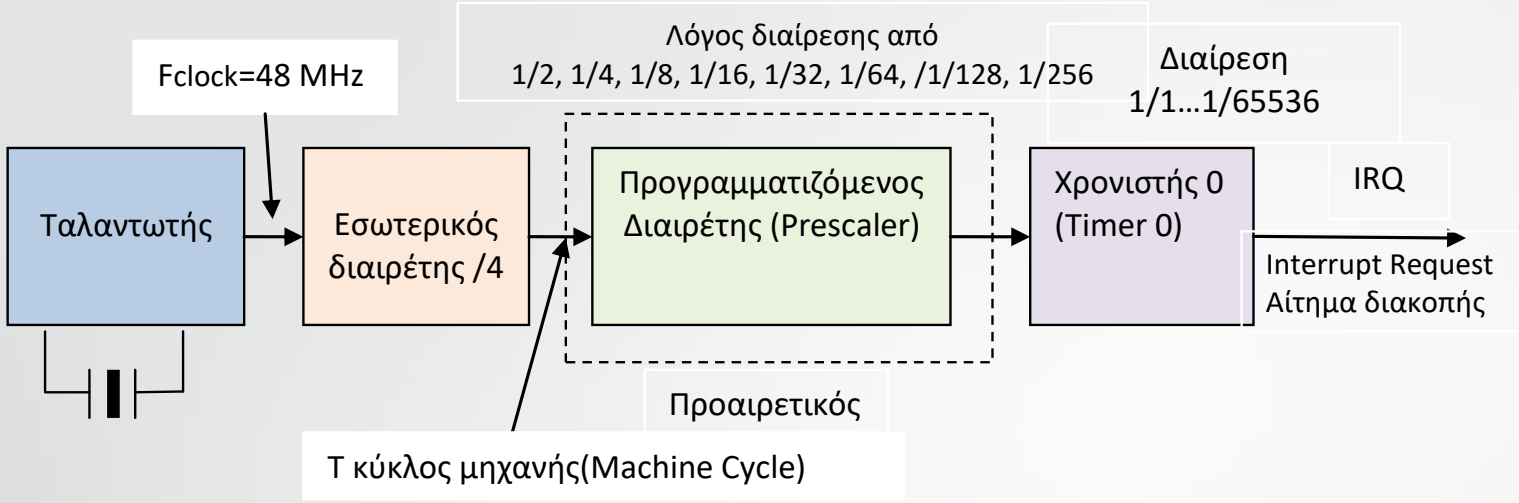
Ρουτίνα εξυπηρέτησης διακοπής από τον timer0

```
// *** Αρχή ρουτίνας εξυπηρέτησης της διακοπής ***
#INT_TIMER0     // Οδηγία ότι η επόμενη ρουτίνα είναι η ρουτίνα εξυπηρέτησης της
                // από τον Timer0

void timer0_int(void){
    PORTB = PORTB + 1;      // Αύξηση του περιεχομένου της πόρτας B κατά 1
}   // Κλείνει η αγκύλη της ρουτίνας εξυπηρέτησης της διακοπής
// *** Τέλος ρουτίνας εξυπηρέτησης της διακοπής ***
```

Ρουτίνα αρχικών ρυθμίσεων

```
// *** Αρχή ρουτίνας αρχικών ρυθμίσεων ***
void init (void) {
    SETUP_TIMER_0(T0_INTERNAL | T0_DIV_16 ); // Παλμοί από τον εσωτερικό ταλαντωτή
                                             // (internal oscillator)
                                             // Ρύθμιση του προγραμματιζόμενου
                                          // διαιρέτη στην τιμή 1/16
    enable_interrupts(GLOBAL);          // Ενεργοποίηση του γενικού διακόπτη των διακοπών
    enable_interrupts(INT_TIMER0);      // Ενεργοποίηση της διακοπής από τον timer0

set_tris_b(0x00);  //Η πόρτα Β γίνεται έξοδος

    PORTB=0x00;          //Αρχική τιμή 0 στην πόρτα Β
} // Τέλος ρουτίνας αρχικών ρυθμίσεων******
```

Υπολογισμός του χρόνου μεταξύ δύο διαδοχικών διακοπών (1/2)



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

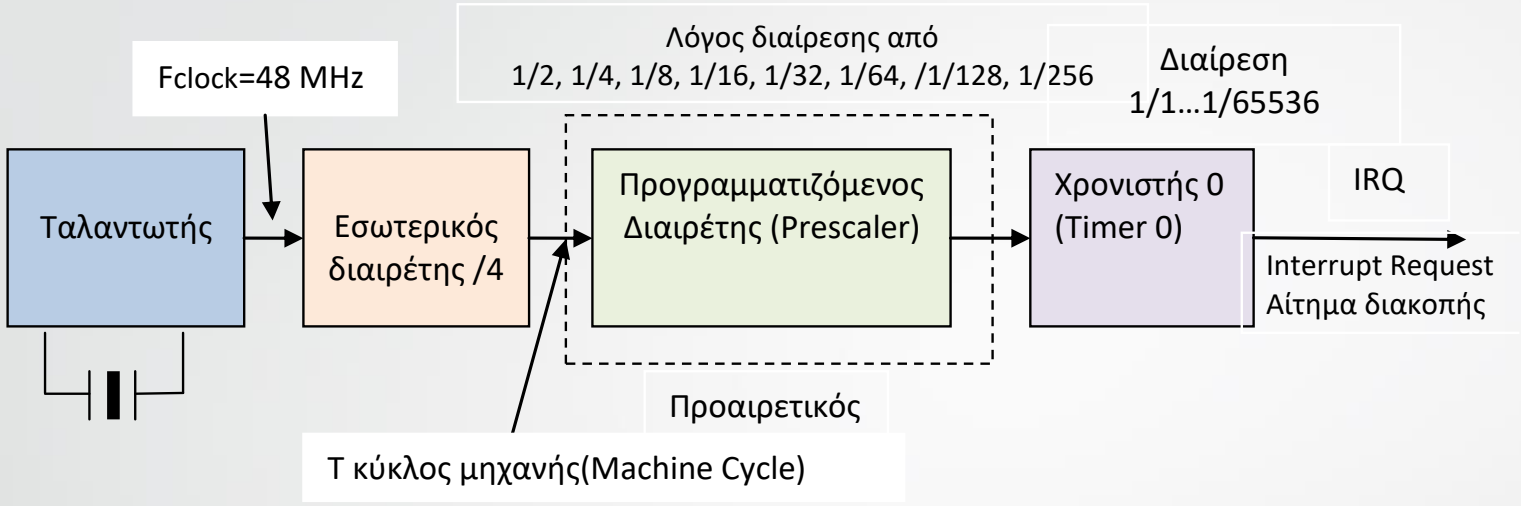
$f_{\text{έξοδος εσωτερικού διαιρέτη}} = 48 * \frac{1}{4}\ MHz = 12\ MHz$

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

$T_{\text{έξοδος εσωτερικού διαιρέτη}} = \frac{1}{12MHz}$=0,08333μs=83,33 ns

- **ΑΡΑ, ένας κύκλος μηχανής είναι ίσος με 83,33 ns**

Υπολογισμός του χρόνου μεταξύ δύο διαδοχικών διακοπών (2/2)



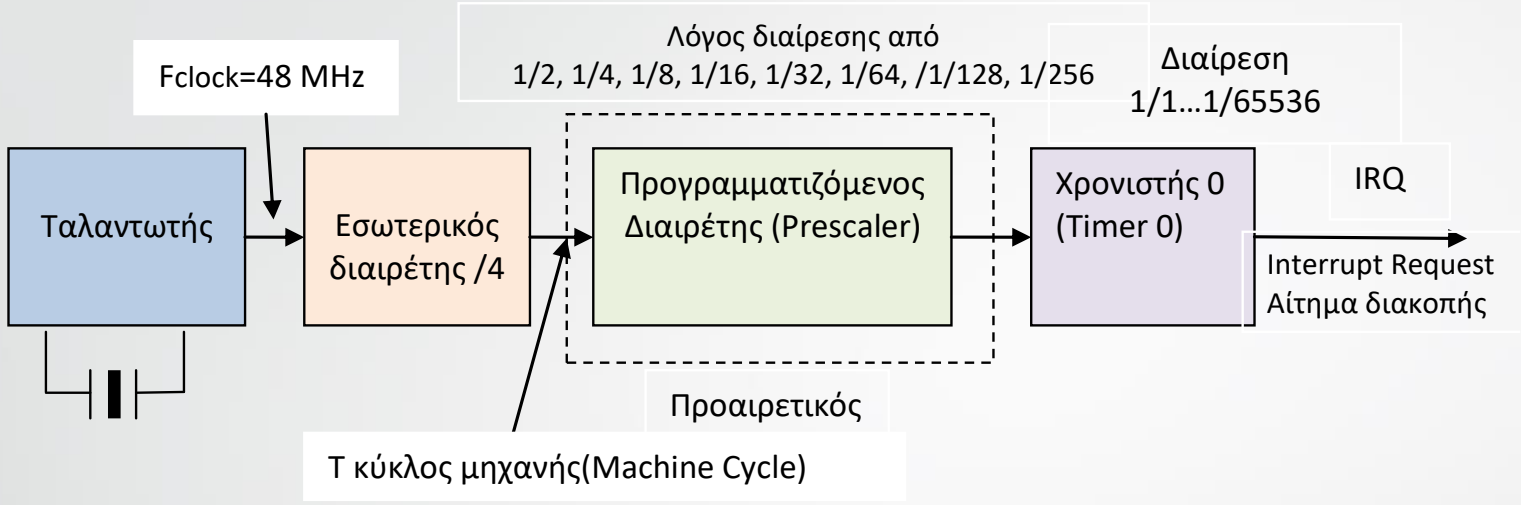
- Ο κύκλος μηχανής Τ υπολογίσθηκε σε 83,33 ns
- Η περίοδος στην έξοδο του προγραμματιζόμενου διαιρέτη (είσοδος timer) θα είναι:

  16 x 83,33 ns = 1333,28 ns =1,333 μs

  δεδομένου ότι επιλέχθηκε λόγος διαίρεσης 1/16
- Η υπερχείλιση του timer0 (δηλαδή η μετάβαση από 111…111 σε 000…000) θα συμβαίνει κάθε:

  1,333 μs x 65536 = 87359,488 μs =87,359 ms

Υπολογισμός ώστε να συμβαίνουν διακοπές ανά συγκεκριμένο χρονικό διάστημα, για παράδειγμα κάθε 50 ms (1/2)

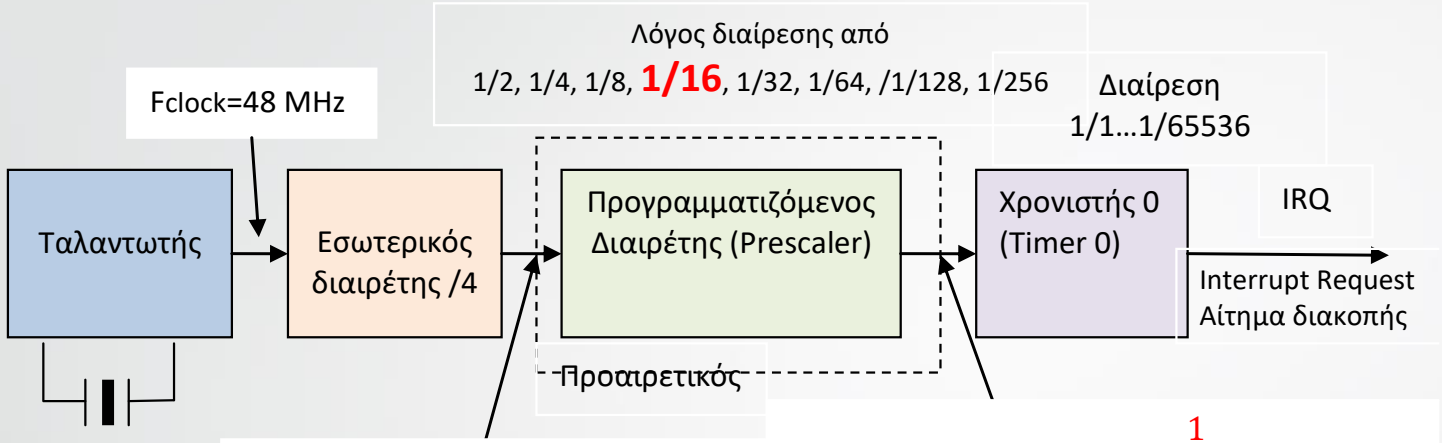


1. Στο παραπάνω διάγραμμα υπολογίσθηκε ο χρόνος του κύκλου μηχανής ίσος με 83,33 ns
2. Εάν επιλεγεί λόγος διαίρεσης 1/16, ο χρόνος μεταξύ δύο διαδοχικών διακοπών, δηλαδή μεταξύ δύο διαδοχικών μεταβάσεων από 111…111 σε 000…000, είναι 87,359 ms
3. **Πώς μπορούμε να ρυθμίσουμε τον timer ώστε οι διακοπές να συμβαίνουν κάθε 50 ms;**

Απάντηση: Τοποθετώντας στον timer μετά τη μετάβαση από 111…111 σε 000….000 (η οποία προκαλεί τη διακοπή) μια αρχική τιμή π.χ. 20000. Σε αυτήν την περίπτωση, για τη νέα υπερχείλιση του timer δεν θα χρειάζονται 65536 παλμοί αλλά μόνο (65536-20000) παλμοί.

Την αρχική τιμή που θα πρέπει να βάλουμε στον timer, ώστε να έχουμε διακοπές κάθε 50 ms, θα πρέπει να την υπολογίσουμε.

Υπολογισμός ώστε να συμβαίνουν διακοπές ανά συγκεκριμένο χρονικό διάστημα για παράδειγμα κάθε 50 ms (2/2)



Tinput-timer=16 x 83,33 ns=1333,28 ns=1,33328μs

Για να έχουμε διακοπές κάθε 50 ms θα πρέπει η αρχική τιμή **y** που τίθεται στον **timer** μετά από **κάθε διακοπή** να πληροί τη σχέση:

(65536 – y) x 1,33328 μs = 50000 μs =>
65563 – y = 50000/1,33328 = 37501 (γίνεται στρογγυλοποίηση στην ακέραια τιμή) =>
y = 65536 – 37501 = 28035

Αυτή είναι η αρχική τιμή που πρέπει να τεθεί στον Timer ώστε να έχουμε διακοπές κάθε 50 ms

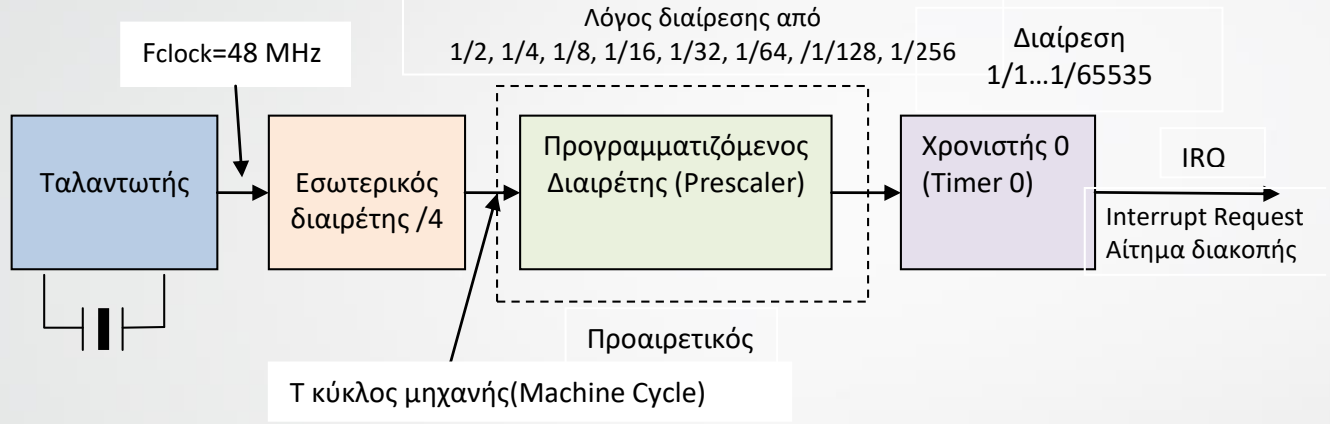
Πώς θα ρυθμίσουμε το σύστημα ώστε μετά από κάθε διακοπή ο timer να μην παίρνει την αρχική τιμή 0 αλλά 28035;

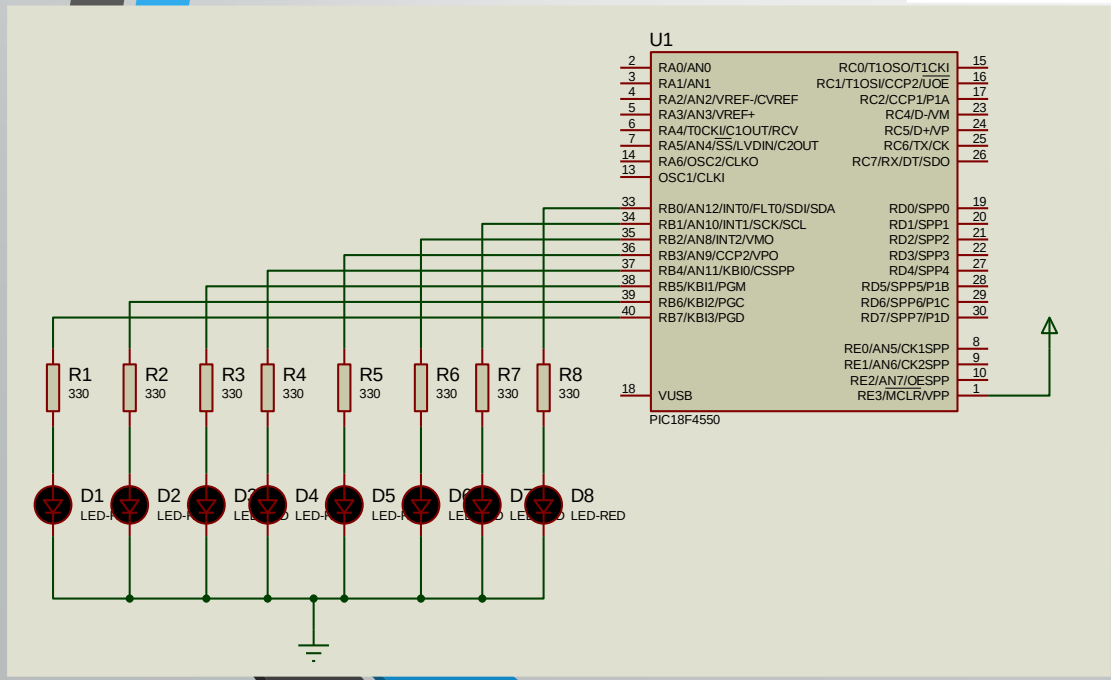
- Μέσα στη ρουτίνα διακοπών θα γράψουμε την κατάλληλη εντολή:

```
#INT_TIMER0              // Οδηγία ότι η επόμενη ρουτίνα είναι η ρουτίνα εξυπηρέτησης της
                         // από τον Timer0
void timer0_int(void){

        set_timer0(28035);      // Αποδίδεται αρχική τιμή στον timer0 ώστε η επόμενη
                                // διακοπή να συμβεί σε χρόνο ίσο με 50 ms
                                // Η επόμενη διακοπή θα γίνει μετά από 65536-28035 παλμούς
                                // στην είσοδο του timer.

        // .........
        // Άλλες εντολές της ρουτίνας διακοπών από τον timer0
        //.........

}   // Αγκύλη κλεισίματος της ρουτίνας διακοπών
```

Άσκηση 6α. 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 10 ms. Η συχνότητα στην έξοδο του κρυσταλλικού ταλαντωτή είναι 48 MHz. Ο προγραμματιζόμενος διαιρέτης να τεθεί στην τιμή 1/64.

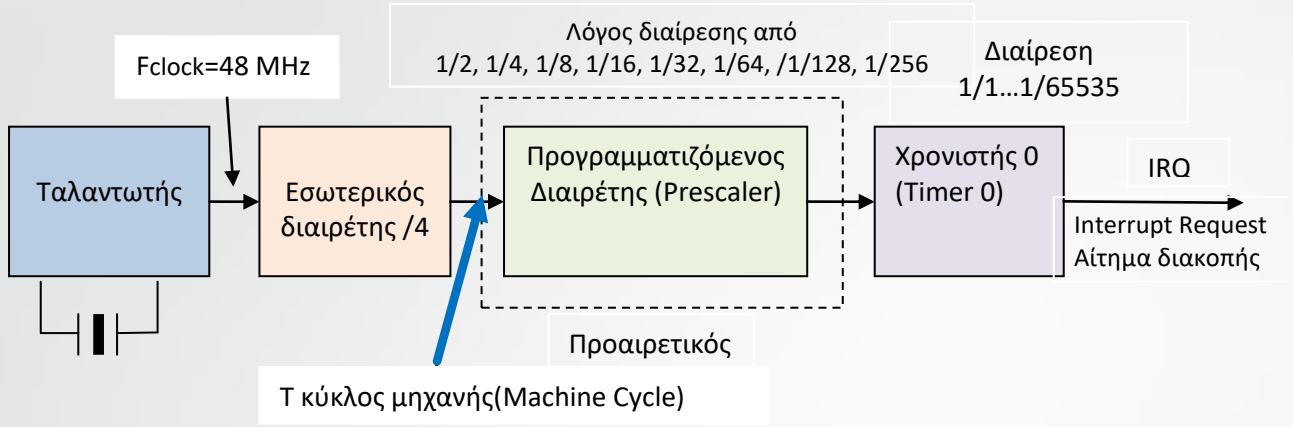




Στη συνέχεια να γραφεί πρόγραμμα με ρουτίνα διακοπών από τον timer0 με το οποίο αναβοσβήνουν τα LED, τα οποία συνδέονται στην πόρτα B, κάθε 200 ms.

Προσοχή! Στο πρόγραμμα ΔΕΝ θα χρησιμοποιηθούν συναρτήσεις χρονοκαθυστέρησης.

Άσκηση 6α. Ασκήσεις πάνω στους χρονιστές (Timers)



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

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

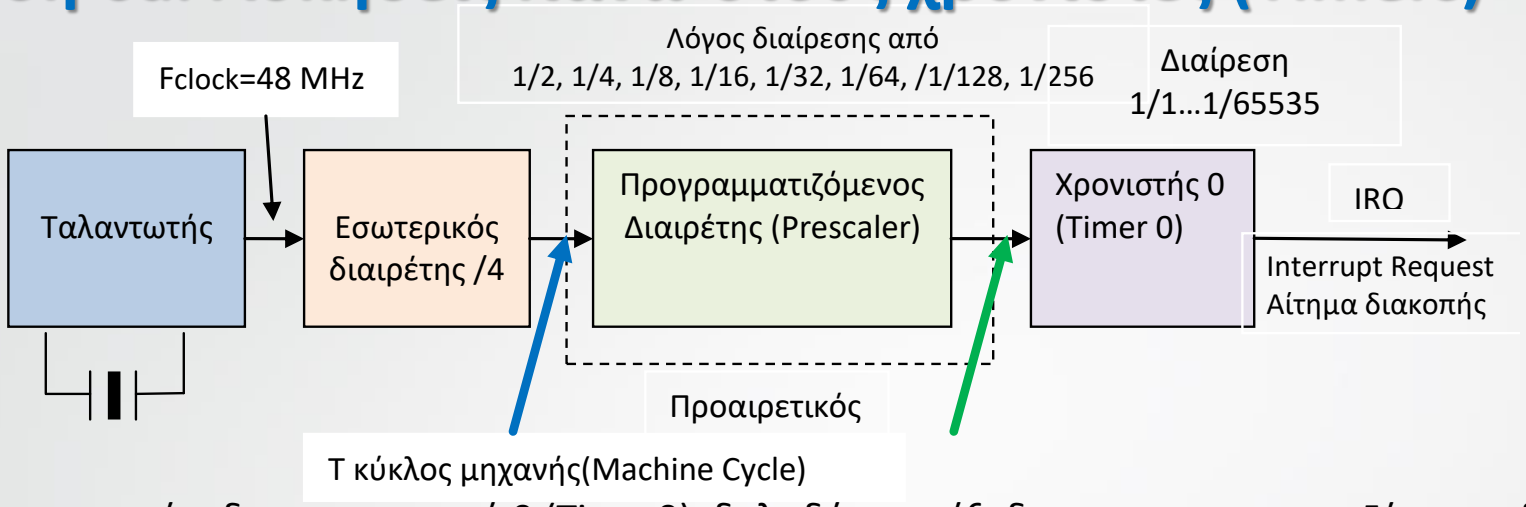
$f_{\text{έξοδος εσωτερικού διαιρέτη}} = 48 * \frac{1}{4}\ MHz = 12\ MHz$

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

$T_{\text{έξοδος εσωτερικού διαιρέτη}} = \frac{1}{12MHz}$=0,08333μs=83,33 ns

ΑΡΑ, ένας κύκλος μηχανής είναι ίσος με 83,33 ns

Άσκηση 6α. Ασκήσεις πάνω στους χρονιστές (Timers)



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

$f_{\text{έξοδος προγραμματιζόμενου διαιρέτη}} = 12 * \frac{1}{64} MHz = 0{,}1875\ MHz$

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

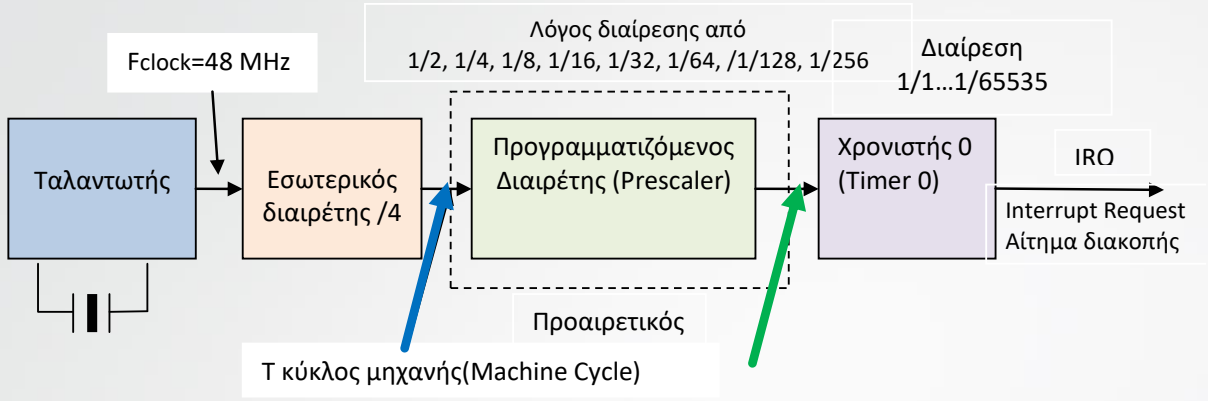
$T_{\text{έξοδος προγραμματιζόμενου διαιρέτη}}$ = 83,33 ns x 64 = 5333 ns

Μπορεί να υπολογισθεί και ως εξής:

$T_{\text{έξοδος προγραμματιζόμενου διαιρέτη}} = \frac{1}{0{,}1875 MHz} = 5{,}333 \mu s = 5333\ ns$

Με βάση την παραπάνω πληροφορία, θα πρέπει να υπολογισθεί η αρχική τιμή στον timer0 έτσι ώστε να συμβαίνει υπερχείλιση του κάθε **10 ms**.

Άσκηση 6α. Ασκήσεις πάνω στους χρονιστές (Timers)



- 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:

  10 ms = 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)] x 5333 ns = 10 000 000 ns

- Υπολογίζουμε την αρχική τιμή του timer0:

  65536 - (Αρχική τιμή Timer0) = $\frac{10\ 000\ 000}{5333} \Leftrightarrow 65536 - (Αρχική\ τιμή\ Timer0) = 1875 \Leftrightarrow$

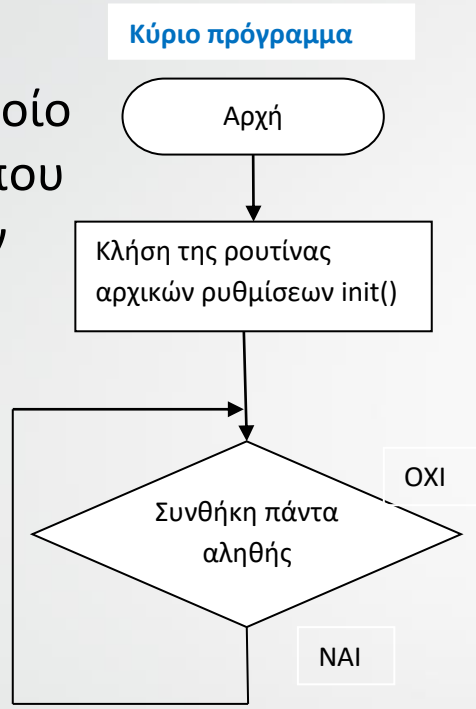
  Αρχική τιμή Timer0 = 65536 - 1875 = 63661

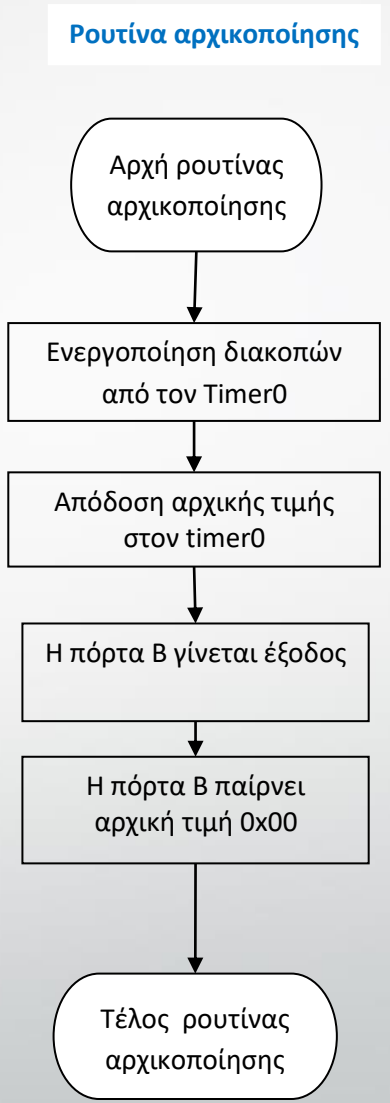
Άσκηση 6α. Ασκήσεις πάνω στους χρονιστές (Timers)

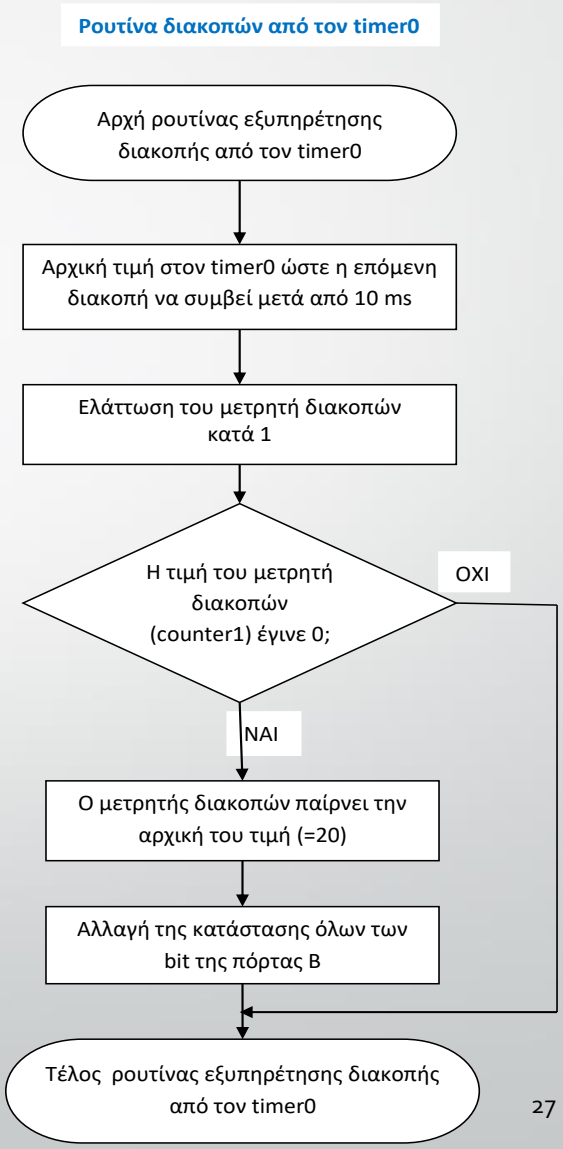
Διάγραμμα ροής του προγράμματος με το οποίο αναβοσβήνουν τα LED που είναι συνδεδεμένα στην πόρτα Β κάθε 200 ms.

Προσοχή!

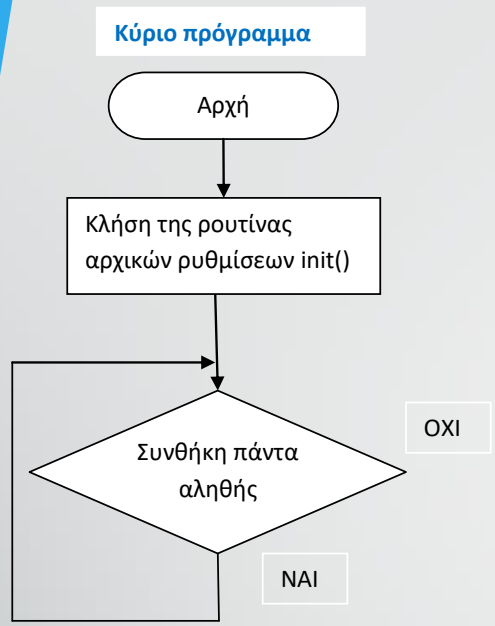
200 ms = 20 x 10 ms



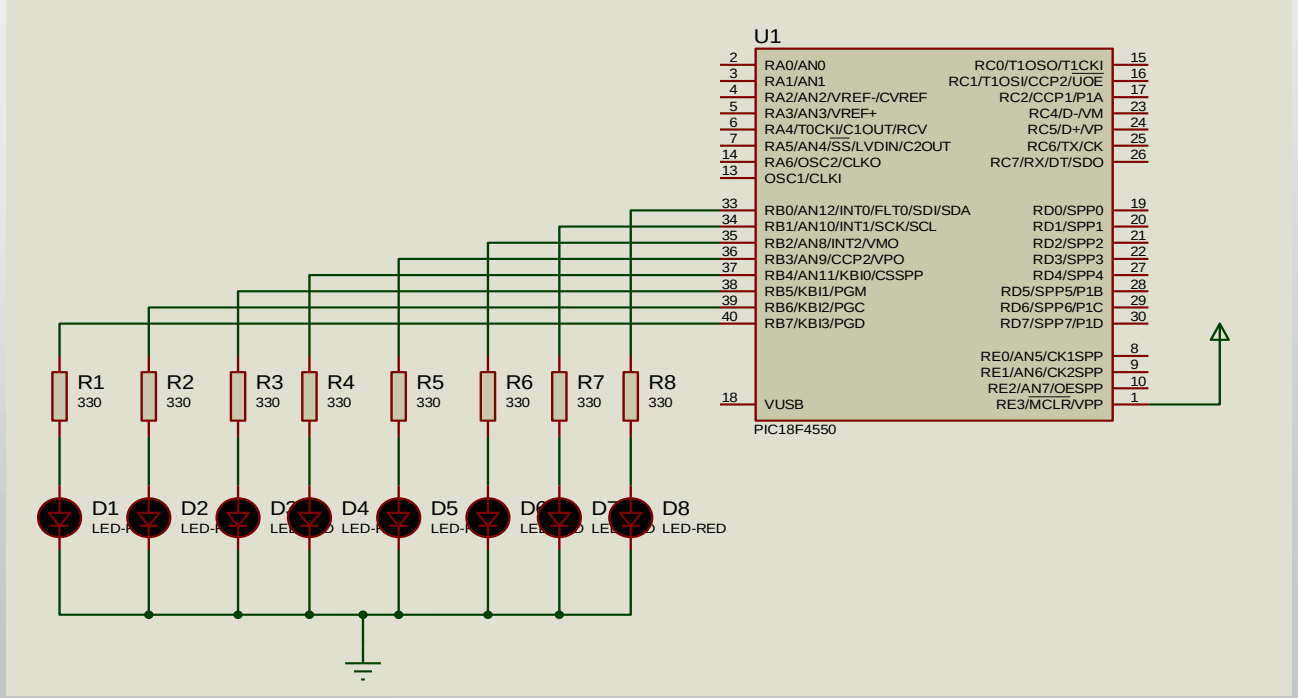




Κύριο πρόγραμμα

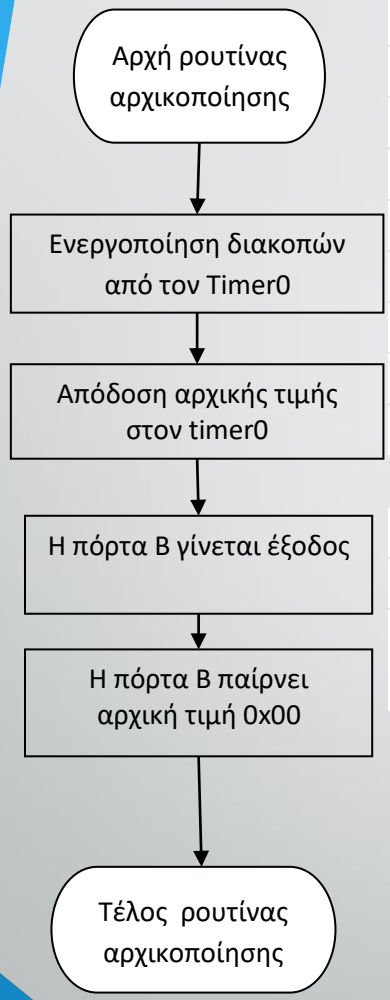



```c
#include <main.h>
#byte PORTB=0xF81 // Καθορισμός του καταχωρητή δεδομένων της πόρτας B

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.

// *** Κύριο πρόγραμμα ***
void main() { // Ανοίγει η αγκύλη της main
     init(); // Κλήση της ρουτίνας των αρχικών ρυθμίσεων
     while (TRUE){ // Το κύριο πρόγραμμα δεν κάνει κάτι. Εκτελεί έναν ατέρμονα βρόχο
     }
} // Κλείνει η αγκύλη main
```



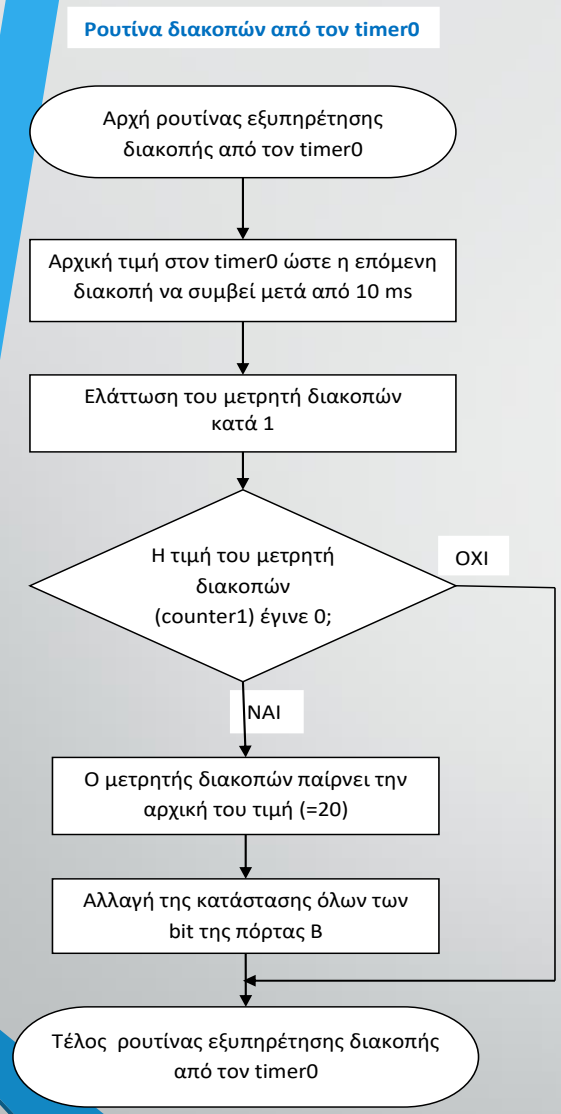
Ρουτίνα αρχικών ρυθμίσεων

Ρουτίνα αρχικοποίησης




```
// Αρχή ρουτίνας αρχικών ρυθμίσεων******
void init (void) {
    SETUP_TIMER_0(T0_INTERNAL | T0_DIV_64 ); // Ρύθμιση του
                                             // προγραμματιζόμενου
                                             // διαιρέτη στην τιμή 1/64
    set_timer0(63661); // Αρχική τιμή του timer0 ώστε να συμβαίνουν
                       // διακοπές κάθε 10 ms
    enable_interrupts(GLOBAL);// Ενεργοποίηση του γενικού διακόπτη
                              // των διακοπών
    enable_interrupts(INT_TIMER0); // Ενεργοποίηση της διακοπής από τον
                                   // timer0

    set_tris_b(0x00); //Η πόρτα Β γίνεται έξοδος
    PORTB = 0x00; //Αρχική τιμή 0 στην πόρτα Β
}
// Τέλος ρουτίνας αρχικών ρυθμίσεων******
```

Ρουτίνα εξυπηρέτησης διακοπών από τον timer0




```
// Αρχή ρουτίνας εξυπηρέτησης της διακοπής*****
#INT_TIMER0     // Οδηγία ότι η επόμενη ρουτίνα είναι η ρουτίνα εξυπηρέτησης της
                // από τον Timer0
void timer0_int(void){// Ανοίγει η αγκύλη της ρουτίνας εξυπηρέτησης της διακοπής
     set_timer0(63661);  // Αρχική τιμή του timer0 ώστε η επόμενη διακοπή να συμβεί σε
                         // χρόνο ίσο με 10 ms
     counter1--;
     if (counter1==0){    // Ανοίγει η αγκύλη του if
          counter1=20;
          PORTB=PORTB^0b11111111; // Με τη 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
     }                    // Κλείνει η αγκύλη του if
}                       // Κλείνει η αγκύλη της ρουτίνας εξυπηρέτησης της διακοπής
// Τέλος ρουτίνας εξυπηρέτησης της διακοπής************
```

Προσοχή!

- Μετράμε 20 διακοπές. Μετά από 20 διακοπές αλλάζουμε την κατάσταση της πόρτας Β.
- Τη μεταβλητή counter1 τη χρησιμοποιούμε για να μετράμε τις διακοπές
- Αφού μετρήσουμε 20 διακοπές, στη συνέχεια δίνουμε στη μεταβλητή counter1 ξανά την τιμή 20 ώστε να μπορούμε να μετρήσουμε ξανά άλλες 20 διακοπές κ.ο.κ.

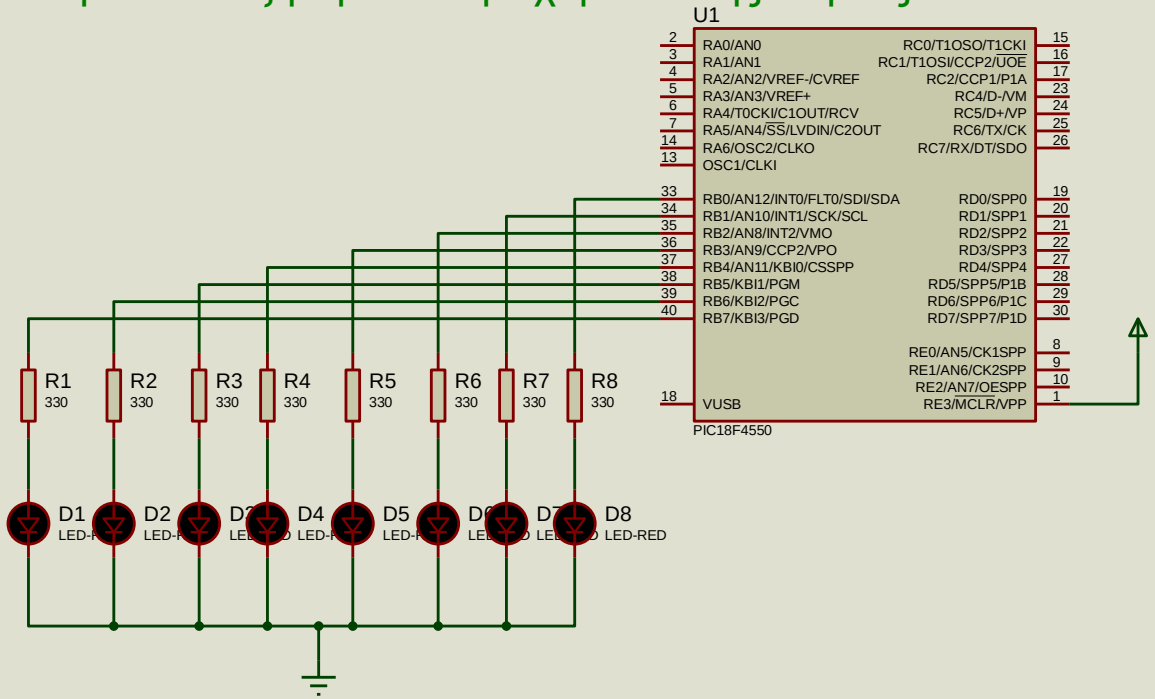
Άσκηση 6b. Αύξηση του περιεχομένου της πόρτας B με χρήση μετρητή διακοπών

Να γραφεί πρόγραμμα με το οποίο να αυξάνεται το περιεχόμενο της πόρτας B κάθε 200 ms. Να γίνουν οι κατάλληλες ρυθμίσεις ώστε οι διακοπές από τον Timer0 να συμβαίνουν κάθε 10 ms.

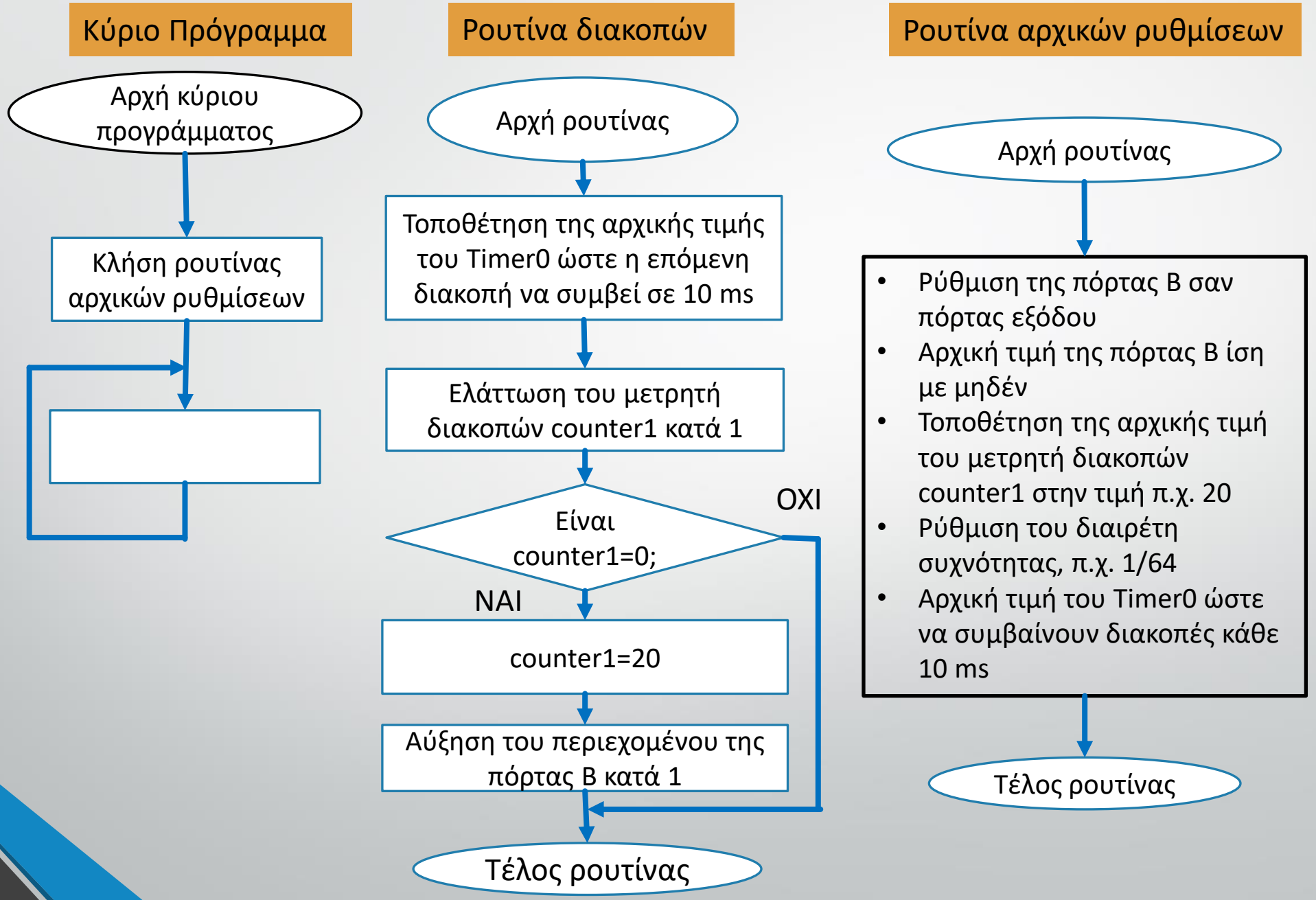
Έχει γίνει υπολογισμός και τοποθετήθηκαν ρυθμίσεις ώστε οι διακοπές να συμβαίνουν κάθε 10 ms

Χρησιμοποιείται μια μεταβλητή, counter1, ώστε να μετράμε τις διακοπές
Όταν μετρηθούν 20 διακοπές αυξάνει το περιεχόμενο της πόρτας B κατά 1

Έτσι γίνεται αύξηση του περιεχομένου της πόρτας B κάθε 20 X10 ms = 200 ms



Άσκηση 6b. Διάγραμμα ροής του προγράμματος



Άσκηση 6b. Το κύριο πρόγραμμα

```
#include <main.h>
#byte PORTB=0xF81    // Καθορισμός της θέσης του καταχωρητή δεδομένων της
                     //πόρτας B στη μνήμη δεδομένων του μικροελεγκτή
void init (void); //Δήλωση της ρουτίνας αρχικοποίησης
void timer0_int(void); //Δήλωση της ρουτίνας διακοπών από τον timer0
int16 counter1=20;   //Δήλωση μεταβλητής για μέτρηση των διακοπών.
                     //Στη μεταβλητή αυτή δίνεται η αρχική τιμή 20.
// Κύριο πρόγραμμα***
void main(){ // Ανοίγει η αγκύλη της main
    init(); // Κλήση της ρουτίνας των αρχικών ρυθμίσεων
    while (TRUE){    // Το κύριο πρόγραμμα δεν κάνει κάτι
    }                // Εκτελεί έναν ατέρμονα βρόχο
} // Κλείνει η αγκύλη main
```

Άσκηση 6b. Η ρουτίνα διακοπών από τον timer0

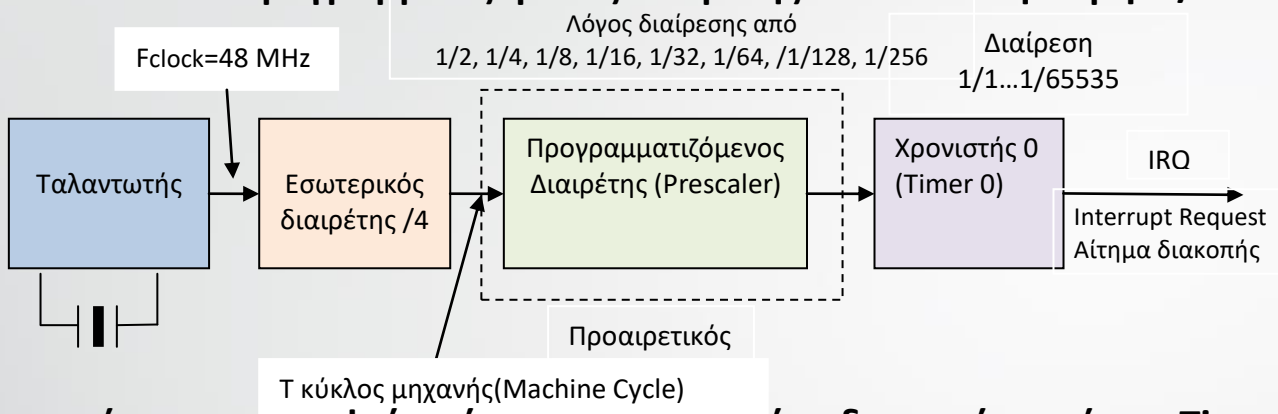
```
// Αρχή ρουτίνας εξυπηρέτησης της διακοπής***
#INT_TIMER0 // Οδηγία ότι η επόμενη ρουτίνα είναι η ρουτίνα εξυπηρέτησης
            // από τον Timer0
void timer0_int(void){
    set_timer0(63661); // Αρχική τιμή του timer0 ώστε η επόμενη διακοπή να συμβεί σε
                       // χρόνο ίσο με 10 ms
    counter1--;
    if (counter1==0){
        counter1=20;
        PORTB=PORTB+1; // Κάθε 20 διακοπές ο μετρητής διακοπών counter1
                       // γίνεται 0. Τότε εκτελείται αυτό που είναι μέσα
                       // στις αγκύλες του if. Ο μετρητής διακοπών counter1
                       // ξαναπαίρνει την τιμή 20 και εκτελείται η εντολή
                       // PORTB=PORTB+1
    } // Κλείνει η αγκύλη του if
} // Κλείνει η αγκύλη της ρουτίνας εξυπηρέτησης της διακοπής
// Τέλος ρουτίνας εξυπηρέτησης της διακοπής*****
```

Άσκηση 6b. Ρουτίνα αρχικών ρυθμίσεων

```
// Αρχή ρουτίνας αρχικών ρυθμίσεων***
void init (void) {
    SETUP_TIMER_0(T0_INTERNAL | T0_DIV_64 ); // Ρύθμιση του προγραμματιζόμενου
                                             // διαιρέτη στην τιμή 1/64
    set_timer0(63661); // Αρχική τιμή timer0 ώστε να συμβαίνουν διακοπές κάθε 10 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*****
```

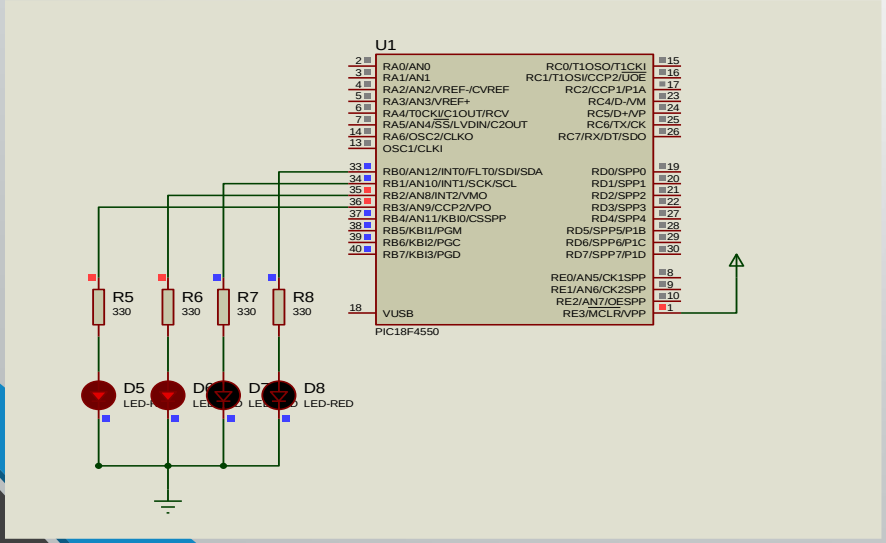
Άσκηση 6c. Ασκήσεις πάνω στους χρονιστές (Timers)

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 1/32



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:

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



Ρυθμίζουμε ώστε να πραγματοποιούνται διακοπές κάθε 50 ms

Το LED0 θα αλλάζει κατάσταση κάθε 2 διακοπές

Το LED1 θα αλλάζει κατάσταση κάθε 3 διακοπές

Το LED2 θα αλλάζει κατάσταση κάθε 4 διακοπές

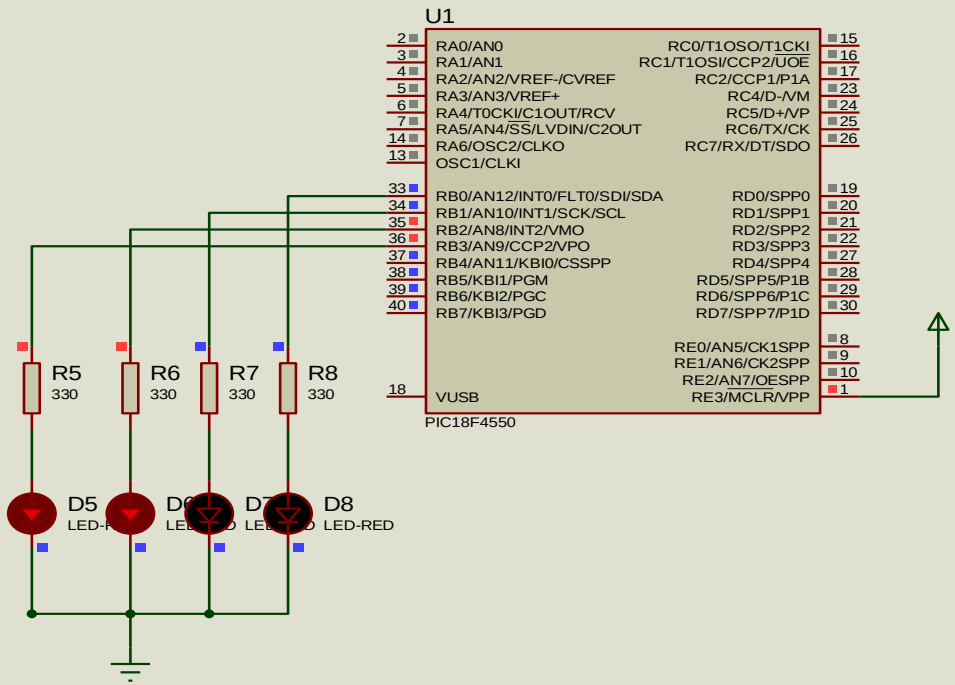
Το LED3 θα αλλάζει κατάσταση κάθε 6 διακοπές

Το αναβόσβημα των LED θα γίνεται μέσα στη ρουτίνα εξυπηρέτησης διακοπών από τον timer0

Τα LED θα αναβοσβήνουν ως εξής:

LED0 κάθε 100 ms LED1 κάθε 150 ms LED2 κάθε 200 ms LED3 κάθε 300 ms

Οι διακοπές θα συμβαίνουν κάθε 50 ms



100 ms = 2 x 50 ms

150 ms = 3 x 50 ms

200 ms = 4 x 50 ms

300 ms = 6 x 50 ms

Θα χρησιμοποιηθούν 4 μετρητές διακοπών:

counter0, counter1, counter2 και counter3

Άσκηση 6c. Δομή του προγράμματος

Κύριο πρόγραμμα: **main(){ }** // Δεν θα κάνει κάτι

Ρουτίνα αρχικών ρυθμίσεων: **init() { }** // Αρχικές ρυθμίσεις

Ρουτίνα διακοπών: **#INT_TIMER0** // Οδηγία για ρουτίνα διακοπών

void timer0_int(void){

- Η ρουτίνα διακοπών θα εκτελείται κάθε 50 ms
- Θα υπάρχουν 4 μετρητές διακοπών που θα παίρνουν τις αρχικές τιμές: counter0=2, counter1=3, counter2 4, counter3=6
- Σε κάθε διακοπή, δηλαδή κάθε 50 ms, θα ελαττώνεται το περιεχόμενο του κάθε μετρητή κατά 1
- Όταν ένας μετρητής γίνει 0 θα αλλάζει την κατάσταση του αντίστοιχου LED και θα ξαναπαίρνει την αρχική του τιμή.

}

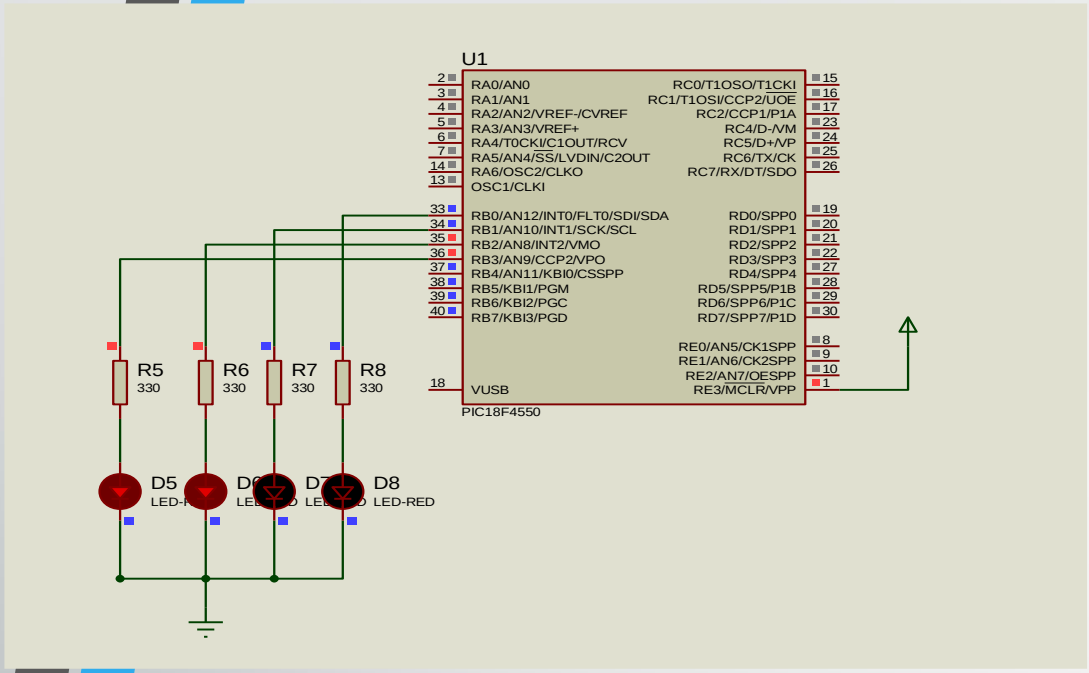
Άσκηση 6c. Κύριο πρόγραμμα

```
#include <main.h>
#byte PORTB=0xF81          // Καθορισμός του καταχωρητή δεδομένων της
                           // πόρτας B
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 τιμές των μεταβλητών
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 κάτι. Εκτελεί έναν ατέρμονα βρόχο
}  // Κλείνει η αγκύλη main
```

Άσκηση 6c. Ρουτίνα αρχικών ρυθμίσεων

```
// Αρχή ρουτίνας αρχικών ρυθμίσεων……
void init (void) {

        SETUP_TIMER_0(T0_INTERNAL | T0_DIV_32 );    // Ρύθμιση του προγραμματιζόμενου
                                                    // διαιρέτη στην τιμή 1/32

        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……
```

Άσκηση 6c. Ρουτίνα διακοπών από τον timer0



Το LED0 θα αλλάζει κατάσταση κάθε 2 διακοπές

Το LED1 θα αλλάζει κατάσταση κάθε 3 διακοπές

Το LED2 θα αλλάζει κατάσταση κάθε 4 διακοπές

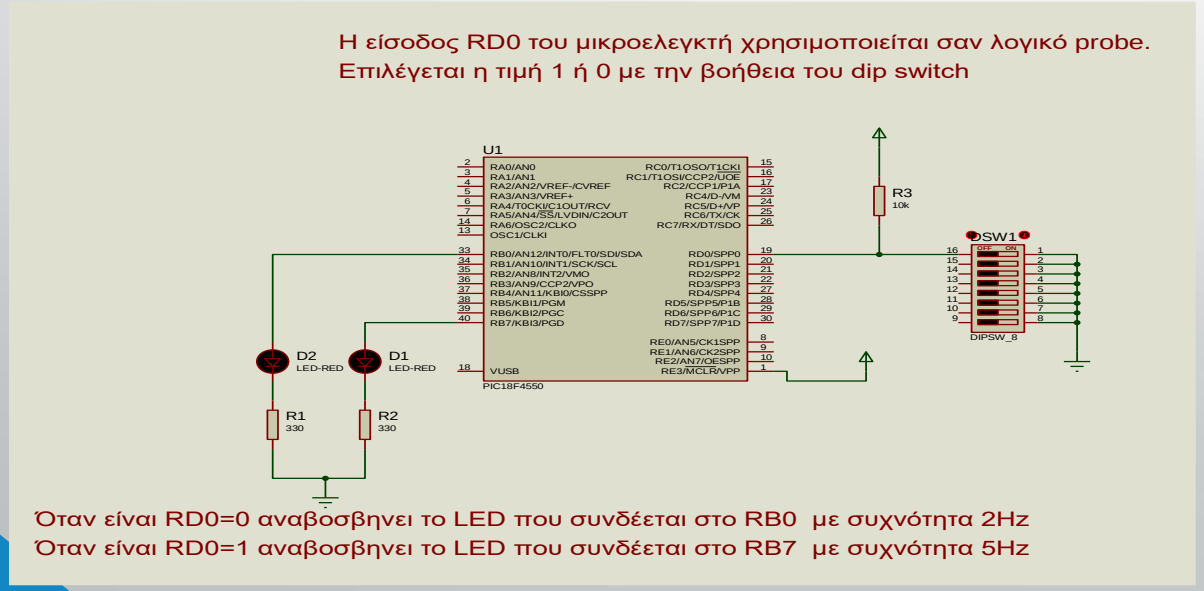
Το LED3 θα αλλάζει κατάσταση κάθε 6 διακοπές

```
// 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 τη 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 B
        }
        if (counter1==0){
                counter1=3;
                PORTB=PORTB^0b00000010;        // Με τη 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 B
        }
        if (counter2==0){
                counter2=4;
                PORTB=PORTB^0b00000100;        // Με τη 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 B
        }

        if (counter3==0){
                counter3=6;
                PORTB=PORTB^0b00001000;        // Με τη 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 B
        }
} // κλείνει η αγκύλη της ρουτίνας εξυπηρέτησης της διακοπής
// Τέλος ρουτίνας εξυπηρέτησης της διακοπής……
```

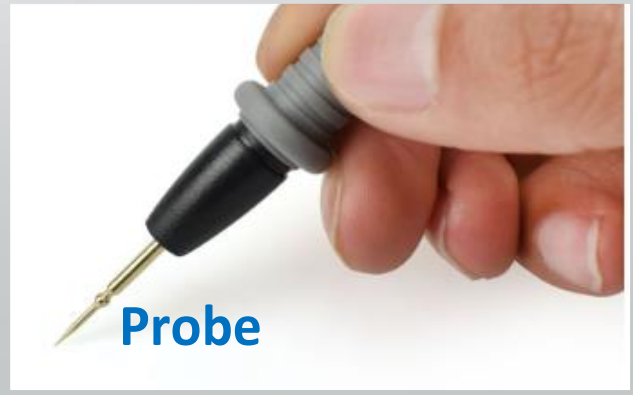
Άσκηση 6d. Ασκήσεις πάνω στους χρονιστές (Timers)

Εφαρμογή για τη δημιουργία λογικού probe με χρήση διακοπών από τον Timer0 (1)

- **Να γραφεί πρόγραμμα σε γλώσσα προγραμματισμού C για το μικροελεγκτή PIC 18F4550 με το οποίο ο μικροελεγκτής θα λειτουργεί σαν λογικό probe (ακροδέκτης με τον οποίο ανιχνεύουμε σε ένα σημείο του κυκλώματος την ύπαρξη του λογικού 0 ή του λογικού 1).**
- **Η είσοδος του λογικού probe θα είναι ο ακροδέκτης RD0.**
- **Αν ο ακροδέκτης RD0 είναι σε λογικό 0 θα πρέπει να αναβοσβήνει το LED0, το οποίο είναι συνδεδεμένο στον ακροδέκτη RB0, με συχνότητα 2 Hz.**
- **Αν ο ακροδέκτης RD0 είναι σε λογικό 1 θα πρέπει να αναβοσβήνει το LED7, το οποίο είναι συνδεδεμένο στον ακροδέκτη RB7, με συχνότητα 5 Hz.**

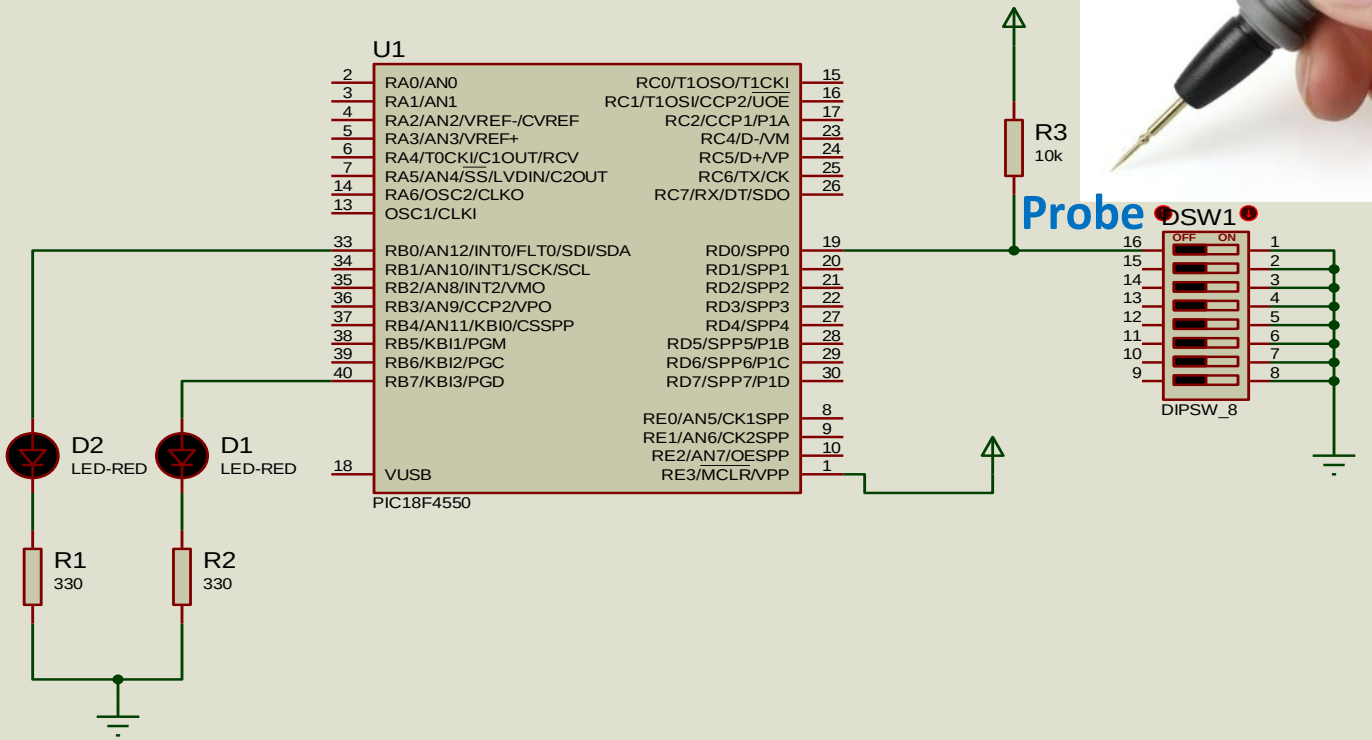


- **Να χρησιμοποιηθεί η μέθοδος της διακοπής από υπερχείλιση του timer0.**



Κύκλωμα άσκησης 6d

Η είσοδος RD0 του μικροελεγκτή χρησιμοποιείται σαν λογικό probe.
Επιλέγεται η τιμή 1 ή 0 με την βοήθεια του dip switch



Όταν είναι RD0=0 αναβοσβηνει το LED που συνδέεται στο RB0 με συχνότητα 2Hz
Όταν είναι RD0=1 αναβοσβηνει το LED που συνδέεται στο RB7 με συχνότητα 5Hz

Α) Έστω ότι έχει ρυθμιστεί ώστε οι διακοπές (interrupts) από τον timer0 να συμβαίνουν κάθε 50 ms.

Αν RD0=0 θα πρέπει το LED0 που συνδέεται στο RB0 να αναβοσβήνει με συχνότητα 2 Hz.

Η περίοδος θα είναι: $T = \frac{1}{f} = \frac{1}{2Hz} = 0{,}5\ \text{sec} = 500\text{ms} = 10\ \text{x}\ 50\ \text{ms}.$

Το LED0 θα πρέπει να αλλάζει κατάσταση ανά χρονικό διάστημα: $\frac{T}{2} = 250\ ms = 5\ x\ 50\ ms$

Δηλαδή αν RD0=0, τότε το LED0 θα πρέπει να αλλάζει κατάσταση κάθε 5 διακοπές

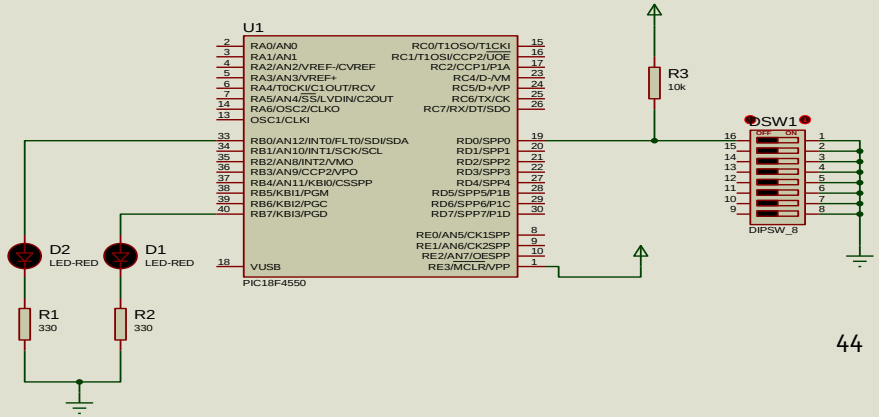
Αν RD0=1 θα πρέπει το LED7 που συνδέεται στο RB7 να αναβοσβήνει με συχνότητα 5 Hz.

Η περίοδος θα είναι: $T' = \frac{1}{f'} = \frac{1}{5Hz} = 0{,}2\ \text{sec} = 200\text{ms} = 4\ \text{x}\ 50\ \text{ms}.$

Το LED7 θα πρέπει να αλλάζει κατάσταση ανά χρονικό διάστημα: $\frac{T'}{2} = 100\ ms = 2\ x\ 50\ ms$

Δηλαδή αν RD0=1, τότε το LED7 θα πρέπει να αλλάζει κατάσταση κάθε 2 διακοπές.

Η είσοδος RD0 του μικροελεγκτή χρησιμοποιείται σαν λογικό probe.
Επιλέγεται η τιμή 1 ή 0 με την βοήθεια του dip switch

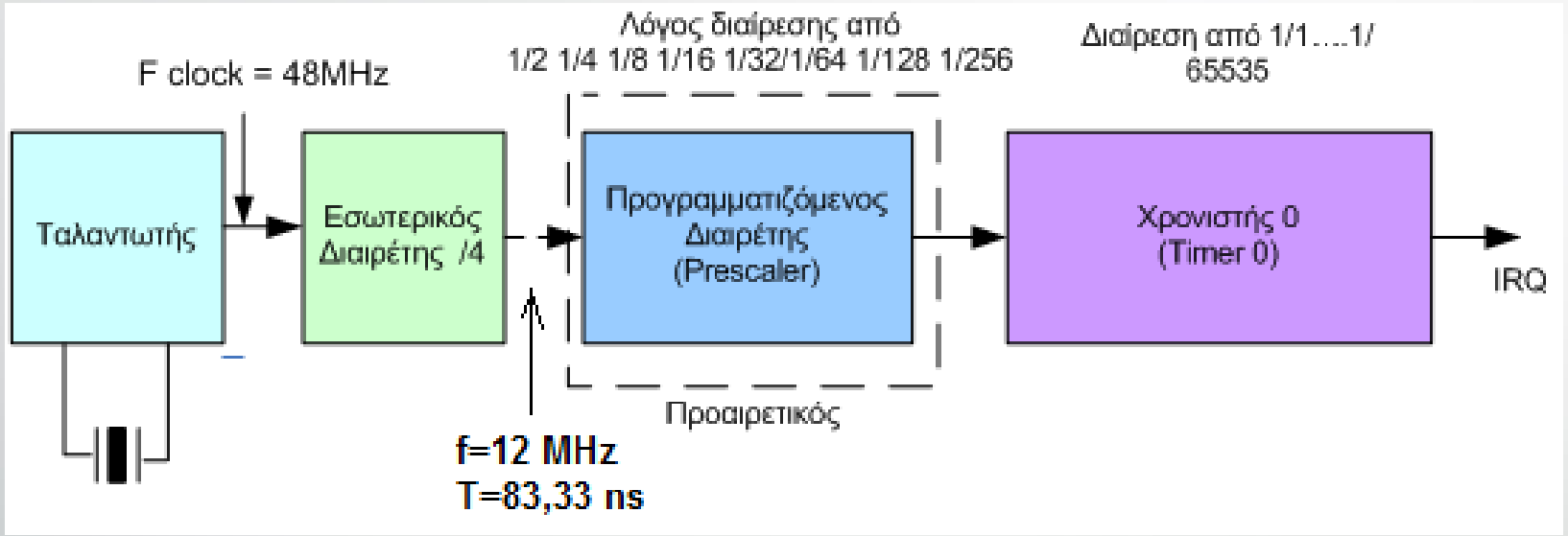


Όταν είναι RD0=0 αναβοσβηνει το LED που συνδέεται στο RB0 με συχνότητα 2Hz
Όταν είναι RD0=1 αναβοσβηνει το LED που συνδέεται στο RB7 με συχνότητα 5Hz

Β) <u>Υπολογισμός της αρχικής τιμής του timer0 ώστε να συμβαίνουν διακοπές κάθε 50 ms.</u>

Επιλέγεται τιμή του προγραμματιζόμενου διαιρέτη ίση με $\frac{1}{64}$.

Υπολογισμός της αρχικής τιμής του timer0 για να έχουμε διακοπές κάθε 50 ms:

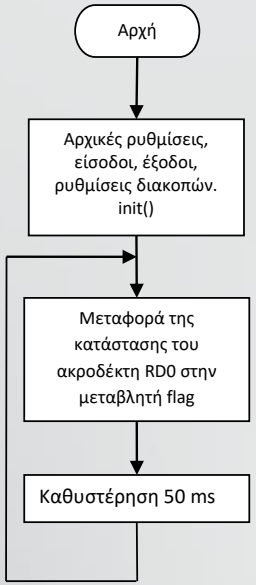


50.000.000 nsec = 83,333 nsec * 64 * (65536 – αρχική τιμή του μετρητή)⇔

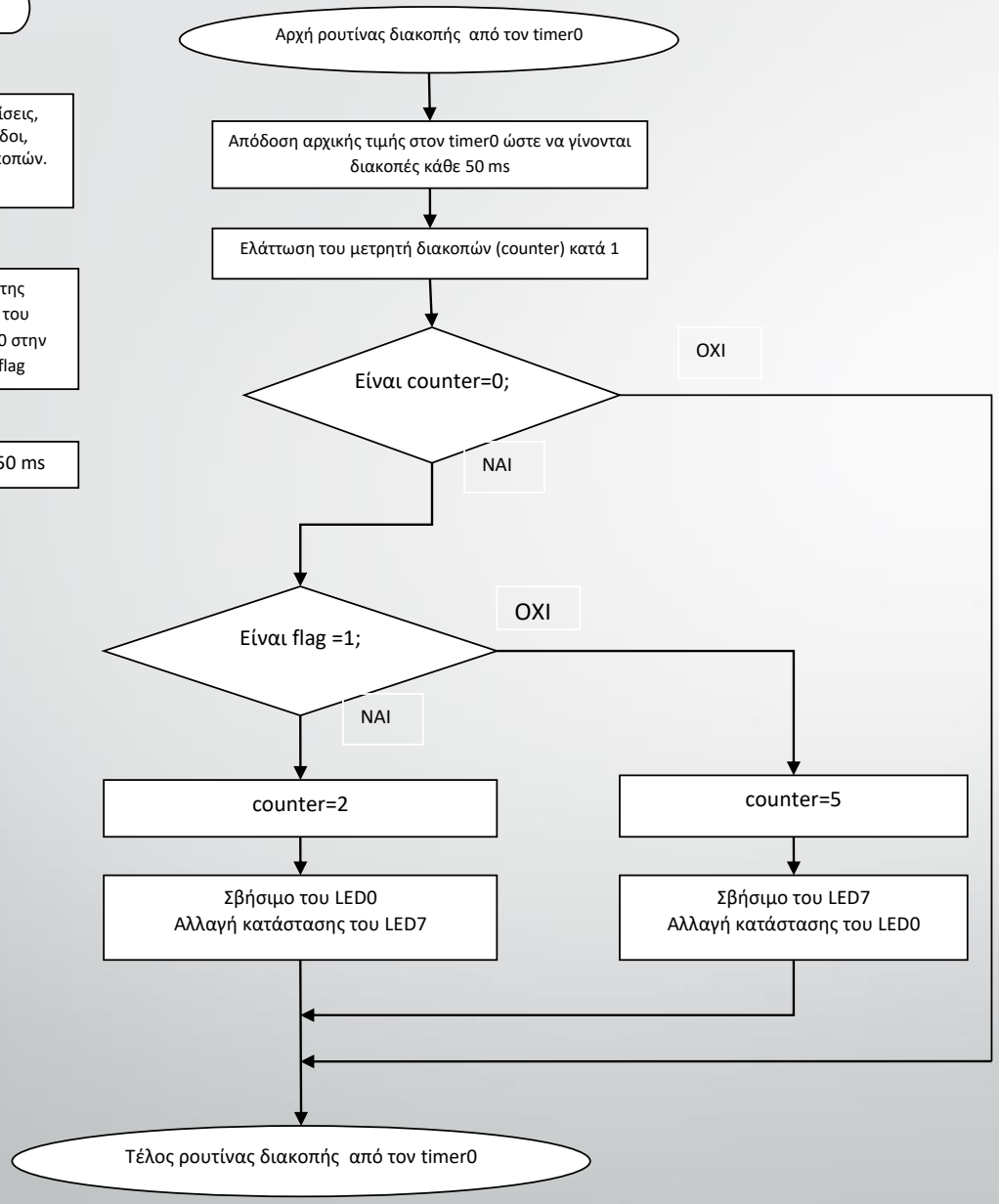
Αρχική τιμή μετρητή = $\frac{83{,}333X64X65536-50.000.000}{83{,}333X64}$ = 56161

Με αρχική τιμή στον timer0=56161 θα έχουμε διακοπές κάθε 50 ms

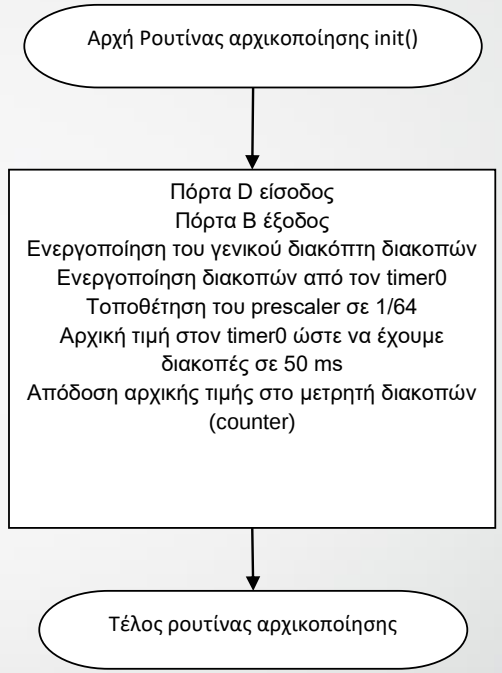
Κύριο πρόγραμμα



Ρουτίνα διακοπών



Ρουτίνα αρχικοποίησης

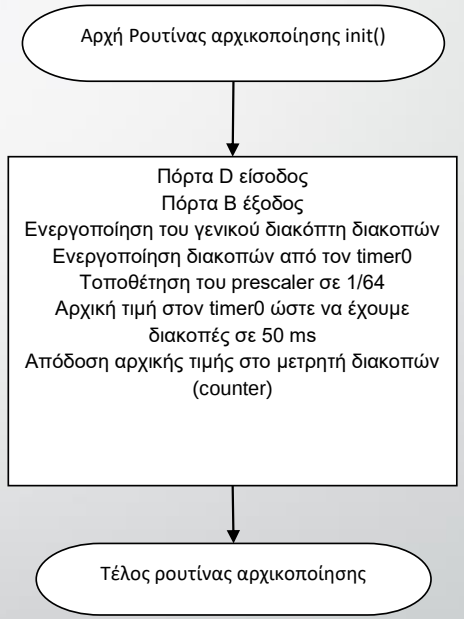


Άσκηση 6d. Δηλώσεις πάνω από την main(){ }

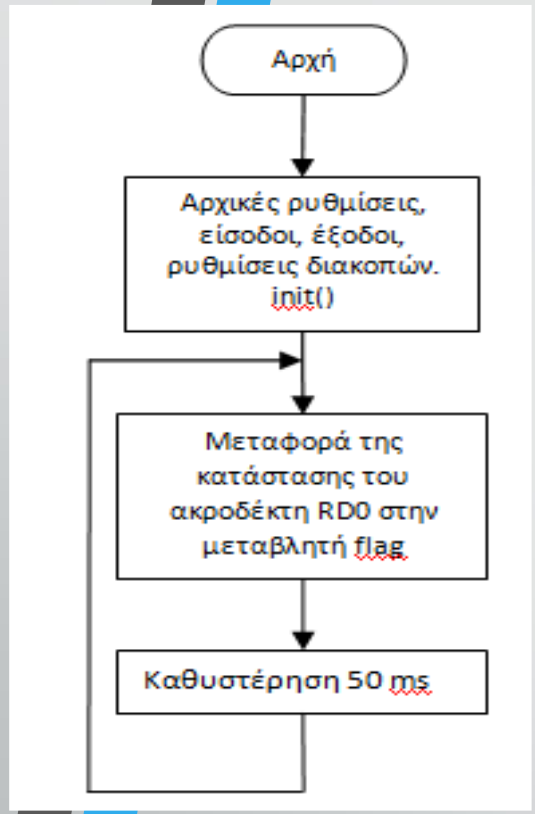

```
#include <main.h>
#byte PORTB       =0xF81  // Ορισμός των θυρών με τη θέση τους στην μνήμη
#byte PORTD       =0xF83


// Προσοχή στους παρακάτω ορισμούς πράξεων Toggle_Led0  και Toggle_Led7
#define Toggle_Led0 PORTB=PORTB^0x01;  // Αλλαγή κατάστασης του bit RB0.  ^ σημαίνει exclusive OR.
#define Toggle_Led7 PORTB=PORTB^0x80;  // Αλλαγή κατάστασης του bit RB7.  ^ σημαίνει exclusive OR.

// Με το Toggle_Led0 αλλάζουμε την κατάσταση του bit 0 της πόρτας B
// Με το Toggle_Led7 αλλάζουμε την κατάσταση του bit 7 της πόρτας B

// Προσοχή στους παρακάτω ορισμούς πράξεων clear_Led0 και clear_Led7
#define clear_Led0  PORTB=PORTB&0b11111110; // Μηδενισμός του bit RB0
#define clear_Led7  PORTB=PORTB&0b01111111; // Μηδενισμός του bit RB7

// Με το clear_Led0 μηδενίζουμε το bit 0 της πόρτας B και αφήνουμε τα άλλα bit αμετάβλητα
//Με το clear_Led7 μηδενίζουμε το bit 7 της πόρτας B και αφήνουμε τα άλλα bit αμετάβλητα

int8  counter;    // Μεταβλητή για ρύθμιση συχνότητας  αναβοσβησίματος του led. Μετρητής
                  // διακοπών. Είναι global μεταβλητή, τοποθετείται πάνω από τη main()
int1 flag=0;      // Τιμή της μεταβλητής flag=1 όταν RD0=1 και flag=0 όταν RD0=0
                  // Είναι global μεταβλητή, τοποθετείται πάνω από τη main()

// Δήλωση συναρτήσεων
void timer0_int(void);  //Οι συναρτήσεις που θα χρησιμοποιηθούν δηλώνονται πάνω από τη main()
void init (void);
```

Άσκηση 6d. Συνάρτηση αρχικών ρυθμίσεων


```
// Ρουτίνα αρχικοποίησης
void init (void){
        set_tris_b(0x00);       // Καθορισμός της πόρτας B ως εξόδου
        set_tris_d(0xff);       // Καθορισμός της πόρτας D ως εισόδου
        PORTB = 0;
        counter=5;              // Αρχική τιμή του counter που αντιστοιχεί σε RD0=0
        SETUP_TIMER_0(T0_INTERNAL | T0_DIV_64 );      //Prescaler=64
        set_timer0(56161);      //  Αρχική τιμή του  μετρητή timer0 για διακοπές
                                //κάθε 50 ms
        enable_interrupts(INT_TIMER0);      // Ενεργοποίηση της
                                            //διακοπής του timer0
        enable_interrupts(GLOBAL);          // Ενεργοποίηση του γενικού
                                            // διακόπτη των διακοπών
}
```



Άσκηση 6d. Η συνάρτηση main() { }




```
// Κύρια συνάρτηση
// Ελέγχει την είσοδο του probe(RD0) και
// αλλάζει την κατάσταση του flag σε 1 ή σε 0
void main() {
        init();
        while (TRUE){
                flag=input(PIN_D0);     // Μεταβλητή στην οποία αποδίδεται η κατάσταση
                                        // του ακροδέκτη RD0 στη μεταβλητή flag
                delay_ms(50);           // Καθυστέρηση για αποφυγή φαινομένου
                                        // αναπηδήσεων κατά την αλλαγή
                                        // κατάστασης της εισόδου από το RD0
        }
} // Κλείνει η main
```

Άσκηση 6d. Ρουτίνα διακοπών timer0_int(){ }


```
// Ρουτίνα διακοπής από τον timer0. Αλλάζει την κατάσταση του led0 ή του led7.
#INT_TIMER0    // Directive. Οδηγία προς τον Compiler. Δηλώνει ότι η επόμενη ρουτίνα είναι ρουτίνα
               // διακοπών από τον timer0
void timer0_int(void){
                    set_timer0(56161);  // Αρχική τιμή του μετρητή
                                        // για να συμβεί η επόμενη
                                        // διακοπή σε 50ms
                    counter--;          // Ελαττώνεται ο μετρητής διακοπών

// Τι συμβαίνει όταν ο counter γίνει 0 και το probe είναι σε λογικό 0.

                    if (counter==0 && flag==0){ // Τι συμβαίνει όταν το probe(RD0) είναι 0
                        counter=5;
                        Toggle_Led0;    // Αλλαγή κατάστασης του led0
                                        // της πόρτας B
                        clear_Led7;     // Σβήνει το led7
                    }
// Τι συμβαίνει όταν το probe(RD0) γίνει ο και το probe είναι σε λογικό 1.

                    if (counter==0 && flag==1){ // Τι συμβαίνει όταν το probe(RD0) είνα
                        counter=2;
                        Toggle_Led7;    // Αλλαγή κατάστασης του led7
                                        // της πόρτας B
                        clear_Led0;     // Σβήνει το led0
                    }
}
```