ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΤΕΧΝΟΛΟΓΙΑΣ & ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ

Ver. 2, Rev. 1

ΕΓΧΕΙΡΙΔΙΟ ΑΣΚΗΣΕΩΝ ΕΡΓΑΣΤΗΡΙΟΥ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ

ΧΑΡΙΔΗΜΟΣ ΒΕΡΓΟΣ
ΝΙΚΟΛΑΟΣ ΚΩΣΤΑΡΑΣ

ΠΑΤΡΑ 2007

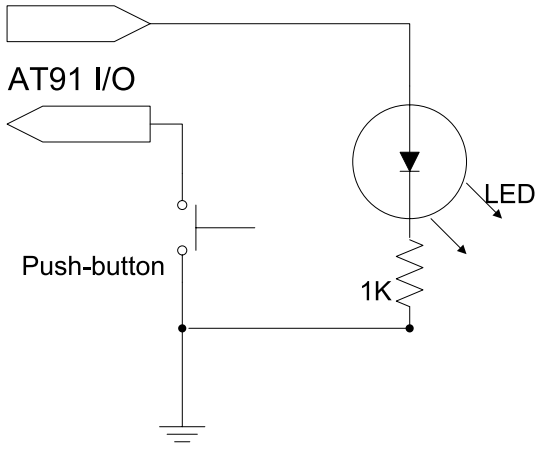
Περιεχόμενα

Εισαγωγική Άσκηση

Εργαστηριακή Άσκηση 1

i. Σκοπός

Σκοπός αυτής της εργαστηριακής άσκησης είναι η εξοικείωσή σας με τη μονάδα εισόδου / εξόδου του AT91, τη μονάδα διαχείρισης διακοπών (interrupts) και τη χρήση του μετρητή του συστήματος (Timer / Counter). Στο τέλος αυτής της άσκησης θα είστε σε θέση να προγραμματίζετε το AT91 ώστε να μπορεί να δεχθεί είσοδο από κάποια μονάδα και να διαβιβάσει δεδομένα στην ίδια ή κάποια άλλη, να διαχειρίζεστε σήματα διακοπών και να χρησιμοποιείτε το μετρητή του συστήματος ώστε να προγραμματίσετε χρονικές ακολουθίες.

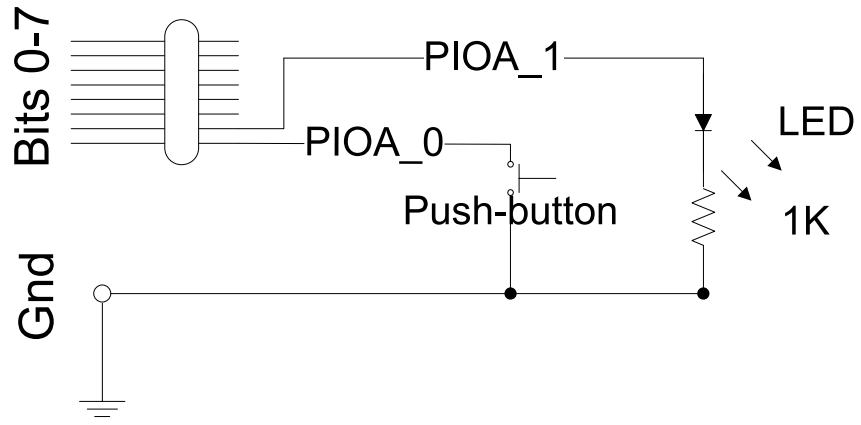


Το κύκλωμα το οποίο πρέπει να αναπτύξετε στη πλακέτα επέκτασης (breadboard) φαίνεται στην παραπάνω εικόνα, και αποτελείται από ένα διακόπτη push-button, ένα LED και μία αντίσταση 1K. Στην άσκηση καλείστε να προγραμματίσετε το AT91 έτσι ώστε να ελέγχει (μέσω της υπομονάδας εξόδου) το εικονιζόμενο LED. Το LED θα πρέπει είτε να αναβοσβήνει με συχνότητα 1 δευτερολέπτου είτε να παραμένει στην ίδια κατάσταση (σβηστό ή αναμμένο). Κάθε διαδοχικό πάτημα του διακόπτη σηματοδοτεί την εναλλαγή μεταξύ αυτών των δύο καταστάσεων.

Από το σχηματικό της προηγούμενης εικόνας προκύπτει πως κάθε πάτημα του διακόπτη προκαλεί ένα χαμηλό δυναμικό στη γραμμή εισόδου προς το AT91. Θα προγραμματίσουμε συνεπώς έτσι το AT91 ώστε χαμηλό δυναμικό σε αυτή τη γραμμή να προκαλεί μια διακοπή. Η ρουτίνα εξυπηρέτησης διακοπών θα περιέχει τον απαραίτητο κώδικα ώστε να ελέγχει ποια περιφερειακή συσκευή προκάλεσε τη διακοπή. Αν η διακοπή έχει προκληθεί από το πάτημα του διακόπτη και το LED δεν είναι στην κατάσταση **¨ΑΝΑΒΟΣΒΗΝΕΙ¨**, τότε θα σηματοδοτηθεί η έναρξη λειτουργίας της μονάδας μέτρησης (Timer) για τη μέτρηση χρονικού διαστήματος ίσου με 1 δευτερόλεπτο, διαφορετικά αν το LED είναι στην κατάσταση **¨ΑΝΑΒΟΣΒΗΝΕΙ¨** θα σηματοδοτηθεί η λήξη λειτουργίας της μονάδας μέτρησης. Πέρα από το πάτημα του διακόπτη, θα προγραμματίσουμε το AT91 ώστε διακοπή να μπορεί να προκληθεί και από την εξάντληση της επιθυμητής μέτρησης από το μετρητή του συστήματος. Αν προκληθεί συνεπώς διακοπή από την ολοκλήρωση της μέτρησης του ενός δευτερολέπτου, η υπομονάδα εξόδου που ελέγχει τον ακροδέκτη στον οποίο είναι συνδεδεμένο το LED θα προγραμματιστεί έτσι ώστε να αντιστραφεί η κατάσταση του LED (από ενεργό να γίνει ανενεργό ή το αντίστροφο). Επιπλέον, θα προγραμματιστεί η μονάδα μέτρησης, ώστε να μετρήσει εκ νέου χρονικό διάστημα ίσο με 1 δευτερόλεπτο.

ii. Κύκλωμα

Η πλήρης συνδεσμολογία που θα πρέπει να αναπτύξετε παρουσιάζεται στο επόμενο διάγραμμα. Οι επιβλέποντες του εργαστηρίου θα σας εξηγήσουν σε ποια σημεία της πλακέτας επέκτασης θα βρείτε τα σήματα τροφοδοσίας και γής καθώς και τους ακροδέκτες 0 (PIOA_0) και 1 (PIOA_1) της μονάδας εισόδου / εξόδου.



Όπως παρατηρείτε, για τη διασύνδεση του κυκλώματος που αναπτύξατε με το AT91 έχουν χρησιμοποιηθεί μόνο οι ακροδέκτες 0 και 1 της μονάδας εισόδου / εξόδου. Οι παράμετροι λειτουργίας των ακροδεκτών της μονάδας εισόδου / εξόδου καθορίζονται μέσω

του προγραμματισμού των καταχωρητών της μονάδας αυτής. *Αναλυτικός οδηγός για τον προγραμματισμό των καταχωρητών αυτής της μονάδας παρατίθεται στο Παράστημα Β.*

Ο ακροδέκτης 0, που συνδέεται με τον διακόπτη, πρέπει να ρυθμιστεί σε λειτουργία εισόδου. Όσο ο διακόπτης δεν είναι πατημένος, η γραμμή δεν οδηγείται από εξωτερική πηγή. Για να μην μένει σε απροσδιόριστη κατάσταση, είναι απαραίτητο να ενεργοποιηθεί η εσωτερική pull-up αντίσταση, ώστε το δυναμικό εισόδου της γραμμής να διατηρείται υψηλό. Όταν πατηθεί ο διακόπτης, η γραμμή 0 θα οδηγηθεί με χαμηλό δυναμικό (η εσωτερική αντίσταση που είναι της τάξης των 100K, θα αρχίσει να διαρέεται από ρεύμα πολύ χαμηλής έντασης). Η αλλαγή της κατάστασης της γραμμής 0 θα ενεργοποιήσει μια διακοπή. *Αναλυτικός οδηγός για τον προγραμματισμό της μονάδας διαχείρησης διακοπών παρατίθεται στο Παράρτημα Γ.* Τέλος, θα πρέπει να λάβετε υπ' όψιν σας ότι ο ελεγκτής εισόδου / εξόδου θα αντιληφθεί τόσο το πάτημα, όσο και την απελευθέρωση του διακόπτη (και στις δυο περιπτώσεις υπάρχει αλλαγή κατάστασης δυναμικού εισόδου), οπότε πρέπει να προβλεφθεί το γεγονός πως θα δημιουργηθούν 2 σήματα διακοπών.

Ο ακροδέκτης 1 πρέπει να ρυθμιστεί σε λειτουργία εξόδου. Χαμηλό δυναμικό σε αυτόν τον ακροδέκτη συνεπάγεται σβήσιμο του LED. Υψηλό δυναμικό αντίθετα συνεπάγεται διαφορά τάσης στα άκρα της φωτοδιόδου μεγαλύτερη των 0.7V και συνεπώς ενεργοποίηση της πηγής φωτός.

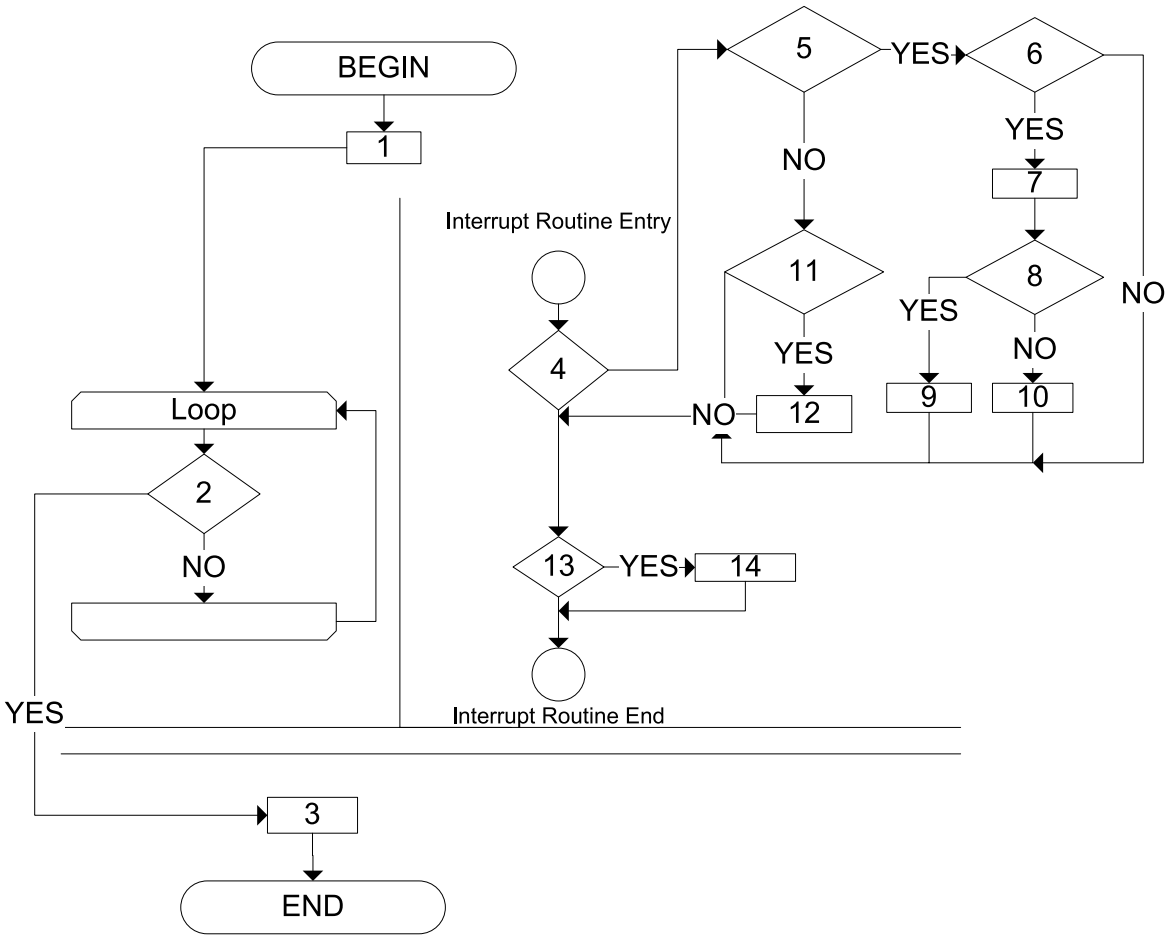
Για τη μέτρηση του διαστήματος μεταξύ των 2 καταστάσεων του LED στην κατάσταση ¨**ΑΝΑΒΟΣΒΗΝΕΙ**¨, θα χρησιμοποιήσουμε τη μονάδα μέτρησης του AT91. *Αναλυτικός οδηγός για τον προγραμματισμό αυτής της μονάδας παρατίθεται στο Παράστημα Δ.*

Χρειάζεται πολύ προσοχή στη συνδεσμολογία του κυκλώματος, ώστε να μη δημιουργηθεί βραχυκύκλωμα ανάμεσα σε γραμμή τροφοδοσίας (Vcc ή γραμμή εξόδου με υψηλό δυναμικό) και γείωση!!!

iii. Υλοποίηση

Ο ζητούμενος προγραμματισμός του AT91 για την επίλυση της άσκησης ακολουθεί το παρακάτω flowchart. Στο flowchart έχουμε αριθμήσει με $1, 2, \ldots, 14$ τα διαφορετικά τμήματα του κώδικα που παρατίθενται με την ίδια αρίθμηση παρακάτω. Για παράδειγμα, στο σημείο 2 γίνεται ένας ατέρμονος βρόχος μέχρι να πατηθεί το πλήκτρο e, οπότε και τερματίζει το πρόγραμμα. Ο κώδικας που αντιστοιχεί στο 2 του flowchart είναι αυτός που παρατίθεται στο υποκεφάλαιο «2. Κεντρικός βρόχος».

Το αρχείο "header.h" το οποίο παρατίθεται στο παράρτημα Α, περιέχει τις δηλώσεις των δομών που χρησιμοποιούνται στον υπόλοιπο κώδικα (συμβατές με τη σημειολογία των παραρτημάτων Β, Γ και Δ) και κάποιες εντολές αρχικοποίησης. Θα πρέπει να περιλαμβάνετε το αρχείο "header.h" στον κώδικά σας για τις επόμενες ασκήσεις, χωρίς να το μεταβάλλετε.



Διάγραμμα ροής προγράμματος

1. Αρχικοποίηση συστήματος

Αρχικοποιεί το πρόγραμμα, τις μεταβλητές του συστήματος και επιτρέπει την πρόσβαση στα περιφερειακά.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <unistd.h>
#include <sys/mman.h>
#include <stdio.h>
#include <stdlib.h>

#include <header.h>

#define PIOA_ID         2
#define TC0_ID          17

#define BUT_IDLE        0
```

```
#define BUT_PRESSED     1
#define BUT_RELEASED    2

#define LED_IDLE        0
#define LED_FLASHING    1

void FIQ_handler(void);

PIO* pioa = NULL;
AIC* aic = NULL;
TC*  tc = NULL;

unsigned int button_state = BUT_IDLE;
unsigned int led_state = LED_IDLE;

int main( int argc, const char* argv[] ){
unsigned int gen;

STARTUP;                        //ΑΡΧΙΚΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΟΣ
tc->Channel_0.RC = 8192;        //ΠΕΡΙΟΔΟΣ 1 ΔΕΥΤΕΡΟΛΕΠΤΟ
tc->Channel_0.CMR = 2084;       // SLOW CLOCK, WAVEFORM, DISABLE CLK ON RC COMPARE
tc->Channel_0.IDR = 0xFF;       //ΑΠΕΝΕΡΓΟΠΟΙΗΣΗ ΟΛΩΝ ΤΩΝ ΔΙΑΚΟΠΩΝ
tc->Channel_0.IER = 0x10;       //ΕΝΕΡΓΟΠΟΙΗΣΗ ΜΟΝΟ ΤΟΥ RC COMPARE

aic->FFER = (1<<PIOA_ID) | (1<<TC0_ID); //ΟΙ ΔΙΑΚΟΠΕΣ 2,17 ΕΙΝΑΙ ΤΥΠΟΥ FIQ
aic->IECR = (1<<PIOA_ID) | (1<<TC0_ID); //ΕΝΕΡΓΟΠΟΙΗΣΗ ΔΙΑΚΟΠΩΝ:PIOA & TC0
pioa->PUER  = 0x01;                     //ΕΝΕΡΓΟΠΟΙΗΣΗ ΣΤΗ ΓΡΑΜΜΗ 0:PULL-UP
pioa->ODR   = 0x01;                     //ΓΡΑΜΜΗ 0: ΛΕΙΤΟΥΡΓΙΑ ΕΙΣΟΔΟΥ
pioa->CODR  = 0x02;                     //ΓΡΑΜΜΗ 1: ΔΥΝΑΜΙΚΟ ΕΞΟΔΟΥ LOW
pioa->OER   = 0x02;                     //ΓΡΑΜΜΗ 1: ΛΕΙΤΟΥΡΓΙΑ ΕΞΟΔΟΥ

gen         = pioa->ISR;                //PIOA: ΕΚΚΑΘΑΡΙΣΗ ΑΠΟ ΤΥΧΟΝ ΔΙΑΚΟΠΕΣ
pioa->PER   = 0x03;                     //ΓΡΑΜΜΕΣ 0,1:ΓΕΝΙΚΟΥ ΣΚΟΠΟΥ
gen         = tc->Channel_0.SR;         //TC0: ΕΚΚΑΘΑΡΙΣΗ ΑΠΟ ΤΥΧΟΝ ΔΙΑΚΟΠΕΣ
aic->ICCR   = (1<<PIOA_ID)|(1<<TC0_ID); //AIC: ΕΚΚΑΘΑΡΙΣΗ ΑΠΟ ΤΥΧΟΝ ΔΙΑΚΟΠΕΣ
pioa->IER   = 0x01;                     //ΕΝΕΡΓΟΠΟΙΗΣΗ ΔΙΑΚΟΠΩΝ ΣΤΗ ΓΡΑΜΜΗ 0
```

2. Κεντρικός βρόχος

Εκτελείται μέχρι να πατηθεί το πλήκτρο 'e' + enter. Εδώ μπορεί να προστεθεί ο κώδικας επεξεργασίας των δεδομένων. Θεωρείται ως νήμα χαμηλής προτεραιότητας, διότι μπορεί να διακοπεί από τη ρουτίνα εξυπηρέτησης διακοπών.

```
while( (tmp = getchar()) != 'e')
{

}
```

3. Τερματισμός συστήματος

Επιστρέφει τις δεσμευμένες περιοχές στο σύστημα και απενεργοποιεί τις διακοπές.

```
aic->IDCR = (1<<PIOA_ID) | (1<<TC0_ID); // ΔΙΑΚΟΠΗ ΤΩΝ AIC interrupts
tc->Channel_0.CCR = 0x02;               // ΑΠΕΝΕΡΓΟΠΟΙΗΣΗ ΤΟΥ Timer
CLEANUP;
return 0;
}
```

Έλεγχος διακόπτη

4. Έλεγχος μονάδας εισόδου/εξόδου. Εξετάζει αν η μονάδα εισόδου/εξόδου προκάλεσε τη διακοπή, ή περιμένει να εξυπηρετηθεί.

5. Έλεγχος κατάστασης διακόπτη. Εξετάζει αν ο διακόπτης πατήθηκε ή αφέθηκε.

6. Έλεγχος σημειωμένης κατάστασης διακόπτη. Ελέγχει αν η κατάσταση στην οποία είχε σημειωθεί πως βρισκόταν ο διακόπτης, πριν αρχίσει να εξυπηρετείται η διακοπή, είναι η "ΜΗ ΠΑΤΗΜΕΝΟΣ".

7. Αλλαγή τρέχουσας κατάστασης. Σημειώνει πως η κατάσταση του διακόπτη είναι "ΠΑΤΗΜΕΝΟΣ".

8. Έλεγχος κατάστασης LED Ελέγχει αν το LED έχει σημειωθεί πως είναι σε κατάσταση "ΔΕΝ ΑΝΑΒΟΣΒΗΝΕΙ".

9. Ενεργοποίηση Μετρητή Ξεκινά τη μέτρηση του 1 δευτερολέπτου και σημειώνει πως η κατάσταση του LED είναι "ΑΝΑΒΟΣΒΗΝΕΙ".

```
void FIQ_handler(void)
{
unsigned int data_in = 0;
unsigned int fiq = 0;
unsigned int data_out;

fiq = aic->IPR; //ΕΝΤΟΠΙΣΜΟΣ ΠΕΡΙΦΕΡΕΙΑΚΟΥ ΠΟΥ ΠΡΟΚΑΛΕΣΕ ΤΗ ΔΙΑΚΟΠΗ

if( fiq & (1<<PIOA_ID) ){   //ΕΛΕΓΧΟΣ ΓΙΑ PIOA
    data_in = pioa->ISR;        //ΕΚΚΑΘΑΡΙΣΗ ΤΗΣ ΠΗΓΗΣ ΤΗΣ ΔΙΑΚΟΠΗΣ
    aic->ICCR = (1<<PIOA_ID);   //ΕΚΚΑΘΑΡΙΣΗ ΤΗΣ ΔΙΑΚΟΠΗΣ ΑΠΟ AIC
    data_in = pioa->PDSR;       //ΑΝΑΓΝΩΣΗ ΤΙΜΩΝ ΕΙΣΟΔΟΥ
    if(  data_in & 0x01 ){      //ΔΙΑΚΟΠΤΗΣ ΠΑΤΗΜΕΝΟΣ;
        if(button_state == BUT_IDLE){
            button_state = BUT_PRESSED;
            if( led_state == LED_IDLE ){    //ΑΝ ΔΕΝ ΑΝΑΒΟΣΒΗΝΕΙ
                tc->Channel_0.CCR = 0x05;   //ΕΝΑΡΞΗ ΜΕΤΡΗΤΗ
                led_state = LED_FLASHING;
            }
```

Έλεγχος διακόπτη

10. Απενεργοποίηση Μετρητή. Απενεργοποιεί τον μετρητή και σημειώνει πως το LED είναι σε κατάσταση "ΔΕΝ ΑΝΑΒΟΣΒΗΝΕΙ".

11. Έλεγχος σημειωμένης κατάστασης διακόπτη. Ελέγχει αν η κατάσταση στην οποία είχε σημειωθεί πως βρισκόταν ο διακόπτης, πριν αρχίσει να εξυπηρετείται η διακοπή, είναι η "ΠΑΤΗΜΕΝΟΣ".

12. Αλλαγή τρέχουσας κατάστασης. Σημειώνει πως η κατάσταση του διακόπτη είναι "ΜΗ ΠΑΤΗΜΕΝΟΣ".

```
            else{
                tc->Channel_0.CCR = 0x02;   //ΔΙΑΚΟΠΗ ΜΕΤΡΗΤΗ
                led_state = LED_IDLE;
            }
        }
    }else{
        if(button_state == BUT_PRESSED)
            button_state = BUT_IDLE;
    }
}
```

Έλεγχος Μετρητή

13. Έλεγχος Μετρητή. Εξετάζει αν η Μονάδα Μετρητή προκάλεσε τη διακοπή, ή περιμένει να εξυπηρετηθεί.

14. Επανέναρξη Μέτρησης. Επαναπρογραμματίζει τη Μονάδα Μετρητή για να ξεκινήσει πάλι η μέτρηση του 1 δευτερολέπτου.

```
    if( fiq & (1<<TC0_ID) ){
        data_out = tc->Channel_0.SR;//ΕΚΚΑΘΑΡΙΣΗ ΤΗΣ ΠΗΓΗΣ ΤΗΣ ΔΙΑΚΟΠΗΣ
        aic->ICCR = (1<<TC0_ID);    //ΕΚΚΑΘΑΡΙΣΗ ΔΙΑΚΟΠΗΣ ΚΑΙ ΑΠΟ ΑΙC
        data_out = pioa->ODSR;      //ΑΝΑΓΝΩΣΗ ΤΙΜΩΝ ΕΞΟΔΟΥ
        pioa->SODR =  data_out & 0x02;
        pioa->CODR = data_out & 0x02;
        tc->Channel_0.CCR = 0x05;
    }
}
```

Παρατηρήστε πως στην αρχή του προγράμματος εκτελείται κλήση της μακροεντολής STARTUP. Η μακροεντολή STARTUP περιέχει τον κώδικα που είναι απαραίτητος για την αρχικοποίηση του συστήματος. Μια από τις διαδικασίες που εκτελούνται είναι και η ενημέρωση του λειτουργικού συστήματος για τη θέση μνήμης όπου έχει τοποθετηθεί η ρουτίνα εξυπηρέτησης διακοπών. Η ρουτίνα εξυπηρέτησης διακοπών πρέπει να έχει πάντα το όνομα FIQ_handler, διότι η μακροεντολή STARTUP κάνει χρήση του συγκεκριμένου ονόματος για να την δηλώσει στο λειτουργικό σύστημα. Μόλις ενεργοποιηθεί κάποια από τις διακοπές που έχουν οριστεί ως ενεργές κατά την αρχικοποίηση των περιφερειακών συσκευών, ο επεξεργαστής θα διακόψει τη ροή εκτέλεσης του προγράμματος και θα μεταβεί στην ρουτίνα εξυπηρέτησης διακοπών. Στη ρουτίνα εξυπηρέτησης πρέπει να γίνει έλεγχος για να εντοπιστεί το περιφερειακό που προκάλεσε τη διακοπή. Αν έχει προκληθεί από τη μονάδα εισόδου/εξόδου, θα γίνει έλεγχος της κατάστασης του διακόπτη και του LED, ώστε να υπολογιστεί η νέα κατάσταση στην οποία θα εισέλθει το LED. Αν έχει προκληθεί από τη μονάδα μετρητή, θα γίνει έλεγχος της τρέχουσας κατάστασης του LED (αναμμένο/σβηστό), θα γίνει αντιστροφή της κατάστασης του LED και θα επαναπρογραμματιστεί η μονάδα μετρητή.

Τέλος, παρατηρείστε πως το πρόγραμμα ολοκληρώνεται με την κλήση της μακροεντολής CLEANUP. Η μακροεντολή CLEANUP περιέχει τον απαραίτητο κώδικα για την αποδέσμευση των πηγών του συστήματος που είχαν δεσμευτεί με την κλήση της μακροεντολής STARTUP.

iv. Αποσφαλμάτωση

Η αποσφαλμάτωση του κυκλώματος και ο έλεγχος ορθής λειτουργίας, θα σας βοηθήσει στο να εντοπίσετε σημεία του κυκλώματός σας που δεν εμφανίζουν την αναμενόμενη

συμπεριφορά (π.χ. βραχυκυκλωμένα, ανοιχτά κυκλώματα λόγω κακής επαφής καλωδίων κλπ). Η μεθοδολογία που μπορείτε να ακολουθήσετε είναι η ρύθμιση των γραμμών εισόδου / εξόδου με τη βοήθεια εργαλείων της κονσόλας του λειτουργικού συστήματος, η εγγραφή τιμών στους καταχωρητές που ελέγχουν την έξοδο ώστε να διαπιστωθεί αν το LED αναβοσβήνει και η ανάγνωση τιμών από τους καταχωρητές εισόδου, ώστε να διαπιστωθεί η αλλαγή του δυναμικού εισόδου κατά το πάτημα του διακόπτη.

Τα εργαλεία που θα χρησιμοποιήσετε είναι τα **mw** για την εγγραφή τιμών στις επιθυμητές θέσεις μνήμης και **md** για την ανάγνωση τιμών από τις επιθυμητές θέσεις μνήμης.

Υπενθύμιση

Η εντολή md 0 <phy_addr> <word_count> χρησιμοποιείται για να εμφανίσει τα περιεχόμενα της μνήμης. Ξεκινά από τη διεύθυνση phy_addr και τυπώνει τις πρώτες <word_count> διαδοχικές λέξεις. Για παράδειγμα, η εντολή

md 0 0xFFFFF400 4

θα τυπώσει τις τιμές των 4 πρώτων καταχωρητών της μονάδας εισόδου / εξόδου. Η εντολή mw 0 <phy_addr> <value> χρησιμοποιείται για να αποθηκεύσει την τιμή <value> στη θέση μνήμης <phy_addr>, εκτελώντας προσπέλαση λέξης (4 bytes). Για παράδειγμα, η εντολή

mw 0 0xFFFFF400 0x10

εκτελεί την εγγραφή της τιμής 16 στον καταχωρητή PIO_PER.
Επειδή η ανάγνωση και η εγγραφή γίνονται με προσπέλαση λέξης, οι διευθύνσεις μνήμης που εισάγονται ως παράμετροι στις 2 αυτές εντολές πρέπει να είναι ακέραια πολλαπλάσια του 4.

Για παράδειγμα, για να ρυθμίσετε την γραμμή 1 σε λειτουργία εξόδου, εκτελέστε τις επόμενες εντολές:

Εντολή	**Λειτουργία Γραμμής**	**Καταχωρητής**
mw 0 0xFFFFF400 0x02	Γενικού Σκοπού	PIO_PER
mw 0 0xFFFFF410 0x02	Έξοδος	PIO_OER
mw 0 0xFFFFF430 0x02	Υψηλό δυναμικό	PIO_SODR
mw 0 0xFFFFF434 0x02	Χαμηλό δυναμικό	PIO_CODR

Η εγγραφή στον καταχωρητή PIO_SODR θα ενεργοποιήσει το LED, ενώ η εγγραφή στον PIO_CODR θα απενεργοποιήσει το LED. Με τις επόμενες εντολές γίνεται ο έλεγχος της γραμμής 0, στην οποίο είναι συνδεδεμένος ο διακόπτης:

Εντολή	**Λειτουργία Γραμμής**	**Καταχωρητής**
mw 0 0xFFFFF400 0x01	Γενικού Σκοπού	PIO_PER
mw 0 0xFFFFF414 0x01	Είσοδος	PIO_ODR
mw 0 0xFFFFF464 0x01	Ενεργοποίηση Pull-up	PIO_PUER
md 0 0xFFFFF43C 1	Ανάγνωση δυναμικού εισόδου	PIO_PDSR

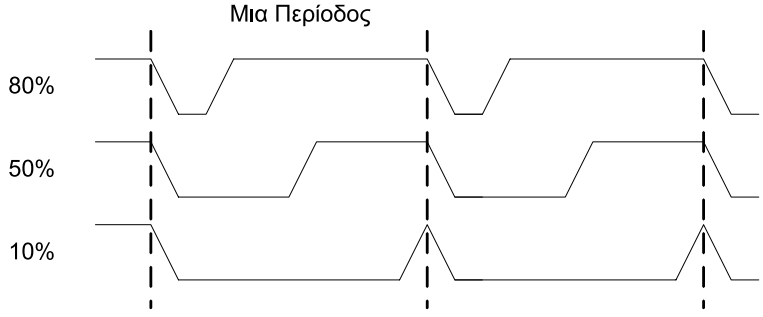
Αν ο διακόπτης είναι πατημένος, η τιμή του bit 0 της λέξης που θα εμφανίσει η εντολή md θα είναι 0 (χαμηλό δυναμικό), ενώ αν δεν είναι πατημένος, η τιμή του bit 0 της λέξης που θα εμφανιστεί θα είναι 1 (λόγω της pull-up).

Διαμόρφωση Εύρους Παλμού

Εργαστηριακή Άσκηση 2

i. Σκοπός

Σκοπός αυτής της εργαστηριακής άσκησης είναι η χρήση της μονάδας εισόδου / εξόδου για εφαρμογή της τεχνικής διαμόρφωσης εύρους παλμού (Pulse Width Modulation PWM) στην οδήγηση LED bar. Η τεχνική της διαμόρφωσης εύρους παλμού σχετίζεται με το χρονικό διάστημα κατά το οποίο μια γραμμή εξόδου διατηρεί το δυναμικό της υψηλό (High). Σε μια χρονική περίοδο, ο λόγος του χρονικού διαστήματος κατά το οποίο το σήμα εξόδου είναι High προς το συνολικό διάστημα της περιόδου ονομάζεται *λόγος κύκλου (duty cycle)*. Για παράδειγμα, στα παρακάτω διαγράμματα παρουσιάζονται κυματομορφές εξόδου με duty cycle ίσο με 80%, 50% και 10% αντίστοιχα:



ii. Υποερώτημα 1. Εφαρμογή τεχνικής PWM σε οδήγηση LED bar

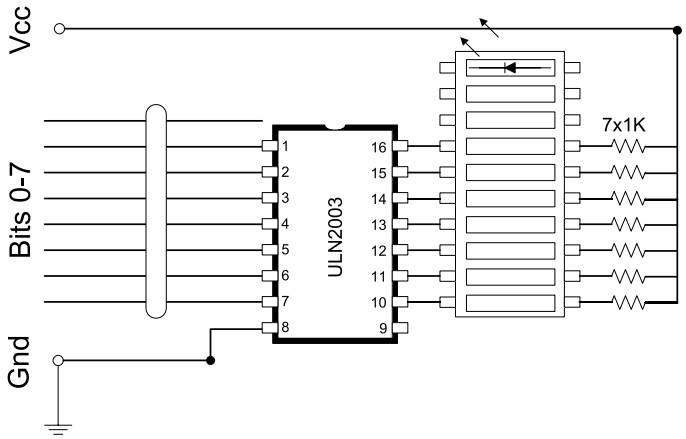
Στο πρώτο μέρος της εργαστηριακής άσκησης καλείστε να συνδέσετε το σύστημα AT91 με ένα LED bar, μέσω της μονάδας εισόδου/εξόδου. Για καλύτερη οδήγηση του εξωτερικού κυκλώματος, θα χρησιμοποιηθούν εξωτερικοί buffers.

iii. Κύκλωμα

Υλικά

Για την ανάπτυξη του 1ου κυκλώματος θα χρειαστείτε 1 πολλαπλή αντίσταση 1K, 1 σειρά LED (LED bar) και 1 κύκλωμα οδήγησης (ULN2003). Για το δεύτερο κύκλωμα θα χρειαστείτε επιπλέον ένα διακόπτη (push-button).

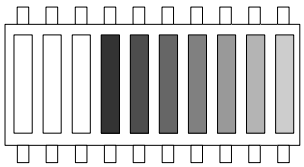
Η συνδεσμολογία του κυκλώματος είναι η εξής:



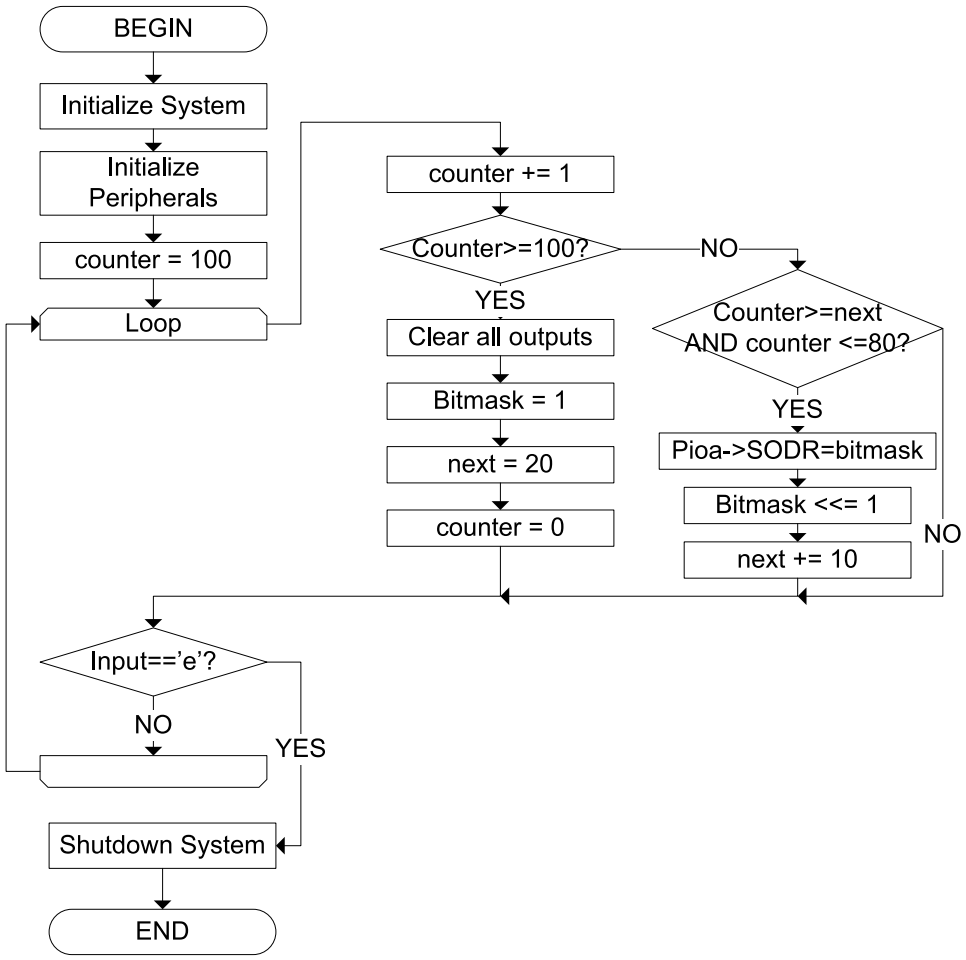
Η οδήγηση των LEDs γίνεται με την τεχνική PWM και σε καθένα από αυτά αντιστοιχίζεται διαφορετικό duty cycle. Ο πίνακας αντιστοίχισης είναι ο ακόλουθος:

Line	Duty Cycle
PIOA_0	80%
PIOA_1	70%
PIOA_2	60%
PIOA_3	50%
PIOA_4	40%
PIOA_5	30%
PIOA_6	20%

Το αποτέλεσμα της εκτέλεσης πρέπει να έχει την ακόλουθη μορφή:



Για να δημιουργήσετε την επιθυμητή διαμόρφωση των εξόδων, μπορείτε να ακολουθήσετε το ακόλουθο ενδεικτικό διάγραμμα ροής:



Αφού γίνει η αρχικοποίηση του συστήματος, η ροή εκτέλεσης θα εισέλθει στον κεντρικό βρόχο. Η αρχική τιμή του counter είναι 100, έτσι η συνθήκη ελέγχου counter>100 θα είναι αληθής κατά την πρώτη εκτέλεση του βρόχου. Η ροή εκτέλεσης θα περάσει αρχικά από τις εντολές αρχικοποίησης των μεταβλητών bitmask, next & counter, ενώ θα απενεργοποιηθούν και όλες οι έξοδοι (θα σβήσουν όλα τα LEDs). Στη συνέχεια, η μεταβλητή counter θα αυξάνεται σε κάθε βρόχο και μόλις φτάσει στην τιμή next (η οποία αρχικά είναι 20), θα ενεργοποιηθεί η γραμμή που αντιστοιχεί στην τιμή της μεταβλητής bitmask (επειδή αρχικά η bitmask είναι 1, γράφοντας την τιμή της στον καταχωρητή PIO_SODR θα ενεργοποιηθεί η γραμμή 0). Η next θα γίνει 30 και η bitmask θα γίνει 2. Μόλις η συνθήκη counter>=next γίνει πάλι αληθής, η εγγραφή της τιμής της bitmask στον PIO_SODR θα ενεργοποιήσει τη γραμμή 1. Σημειώστε ότι η γραμμή 0 είναι ήδη ενεργοποιημένη για 10 επαναλήψεις. Η διαδικασία αυτή θα επαναληφθεί μέχρι ο counter να φτάσει την τιμή

100, όπου θα επαναρχικοποιηθούν οι τιμές των μεταβλητών bitmask, next & counter και θα απενεργοποιηθούν όλες οι έξοδοι. Παρατηρείστε πως η γραμμή 0 είναι ενεργοποιημένη από τον 20ο μέχρι τον 100ο κύκλο επανάληψης (80%), η γραμμή 1 είναι ενεργοποιημένη από τον 30ο μέχρι τον 100ο κύκλο επανάληψης (70%) κλπ.

iv. Οδηγίες αποσφαλμάτωσης υποερωτήματος 1

Ξεκινήστε με την υλοποίηση του κυκλώματος. Βεβαιωθείτε ότι οι συνδεσμολογίες είναι σωστές και ότι δεν υπάρχουν βραχυκυκλώματα ανάμεσα σε γραμμές τροφοδοσίας και γείωσης. Εξακριβώστε την ορθότητα του κυκλώματος με τη βοήθεια της εντολής mw του Linux. Συγκεκριμένα, θέστε τις γραμμές PIOA_0 - PIOA_6 σε κατάσταση λειτουργίας γενικού σκοπού (εγγραφή στον καταχωρητή PIO_PER, διεύθυνση 0xFFFFF400) και σε κατάσταση εξόδου (εγγραφή στον καταχωρητή PIO_OER, διεύθυνση 0xFFFFF410). Οδηγήστε όλες τις εξόδους σε χαμηλό δυναμικό (εγγραφή στον καταχωρητή PIO_CODR, διεύθυνση 0xFFFFF434) και αρχίστε να ενεργοποιείτε τις εξόδους μια-προς-μία. Δηλαδή, αρχικά οδηγήστε την PIOA_0 σε υψηλό δυναμικό, επιβεβαιώστε ότι ανάβει στο αντίστοιχο LED και ξαναοδηγήστε την σε χαμηλό δυναμικό. Επαναλάβετε τα ίδια βήματα και για τις υπόλοιπες 6 γραμμές. Αν υπάρχει κάποιο βραχυκύκλωμα ή πρόβλημα στο κύκλωμα θα φανεί επειδή δεν θα ανάβει το αντίστοιχο LED ή θα ανάψουν περισσότερα από ένα.

v. Υποερώτημα 2. Ολίσθηση φωτεινότητας

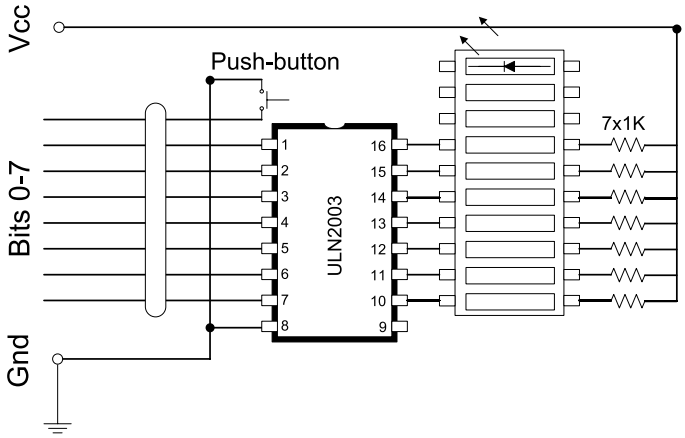
Στο δεύτερο μέρος της εργαστηριακής άσκησης καλείστε να επαυξήσετε τη προηγούμενη λύση σας, έτσι ώστε να εκτελείται δεξιά ή αριστερή κυκλική ολίσθηση με συχνότητα 5Hz, μετά από πάτημα εξωτερικού διακόπτη. Συγκεκριμένα, τα LEDs θα ξεκινούν, με duty cycle ίδιο με αυτό του πρώτου μέρους της άσκησης. Όταν το σύστημα βρίσκεται σε αυτή την κατάσταση (κατάσταση IDLE), το πάτημα του διακόπτη θα ενεργοποιεί την κυκλική ολίσθηση προς τα δεξιά (το σύστημα θα μεταβαίνει στην κατάσταση ROTATING_RIGHT). Όταν το σύστημα βρίσκεται στην κατάσταση ROTATING_RIGHT, το πάτημα του διακόπτη θα ενεργοποιεί την κυκλική ολίσθηση προς τα αριστερά (το σύστημα θα μεταβαίνει στην κατάσταση ROTATING_LEFT). Όταν το σύστημα βρίσκεται στην κατάσταση ROTATING_LEFT, το πάτημα του διακόπτη θα ενεργοποιεί την κατάσταση IDLE και θα σταματά η ολίσθηση.

- Κυκλική ολίσθηση δεξιά: Το duty cycle της γραμμής **n** θα μεταφέρεται στο duty cycle της γραμμής **n-1**, για $n = 6...1$. Το duty cycle της γραμμής 0 θα μεταφέρεται στην 6η γραμμή.

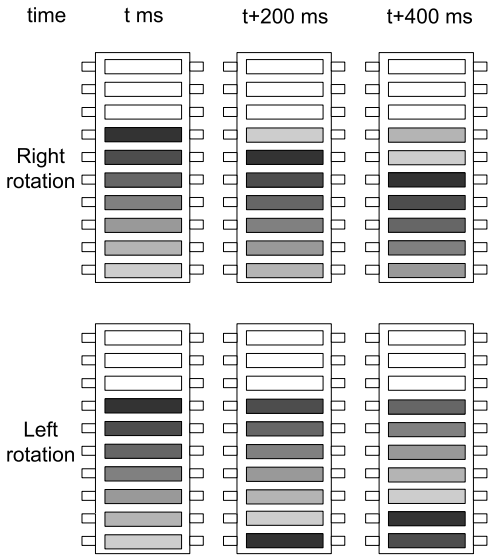
- Κυκλική ολίσθηση αριστερά: Το duty cycle της γραμμής ν θα μεταφέρεται στο duty cycle της γραμμής **n+1**, για $n = 5...0$. Το duty cycle της γραμμής 6 θα μεταφέρεται στην γραμμή 0.

vi. Οδηγίες υλοποίησης υποερωτήματος 2

Τροποποιήστε την συνδεσμολογία του κυκλώματος, έτσι ώστε να συμπεριλάβετε και τον διακόπτη, όπως στο σχήμα.

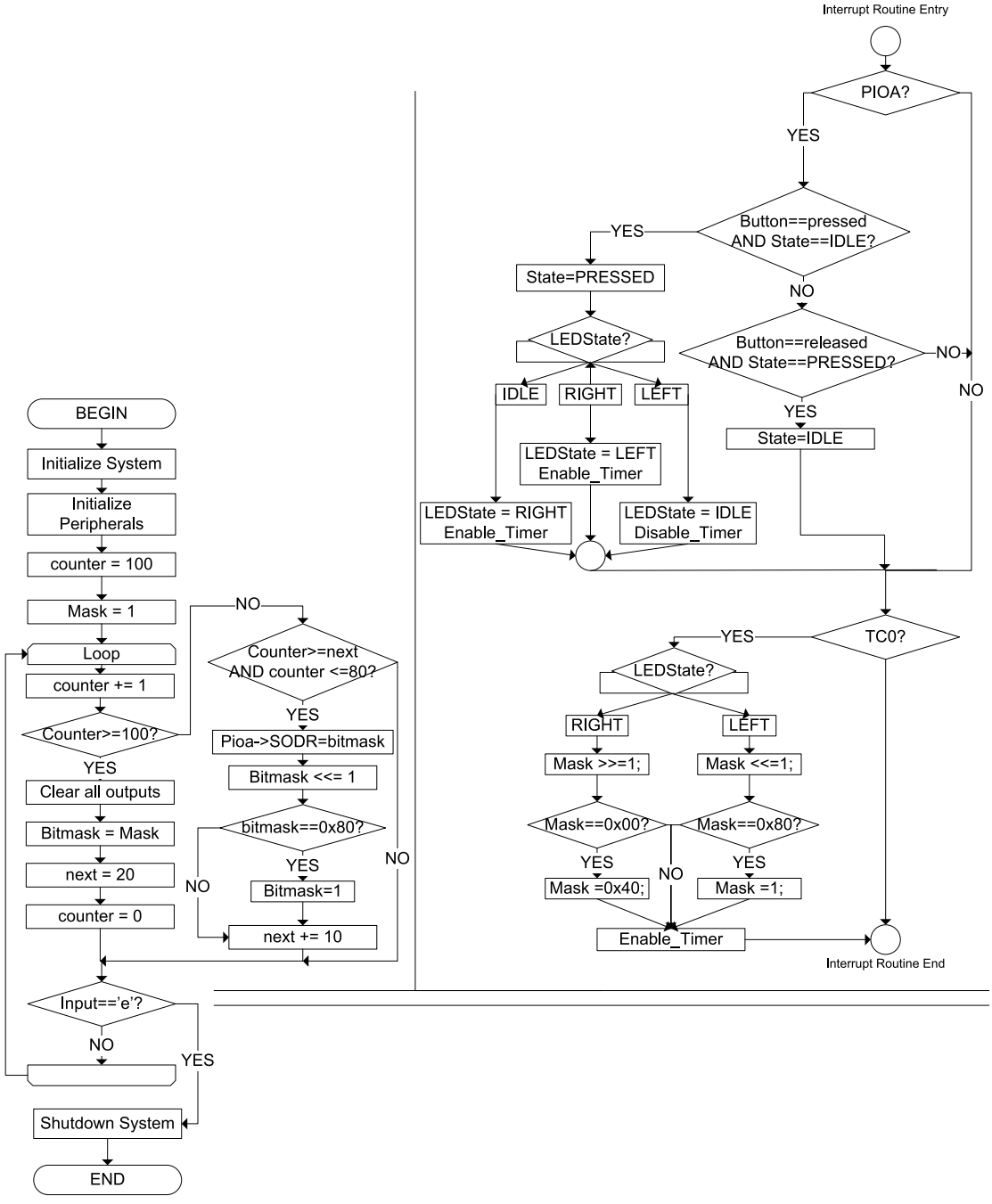


Το αποτέλεσμα που πρέπει να εμφανιστεί απεικονίζεται στο επόμενο σχήμα:



Παρατηρείστε πως η ολίσθηση μπορεί να υλοποιηθεί με τη χρήση μιας μεταβλητής μάσκας (mask) η οποία θα ολισθαίνει δεξιά ή αριστερά και θα αποτελεί την τιμή αρχικοποίησης της μεταβλητής bitmask που χρησιμοποιήσαμε στο προηγούμενο ενδεικ-

τικό διάγραμμα ροής. Ένα ενδεικτικό διάγραμμα ροής για το τρέχον υποερώτημα είναι συνεπώς το ακόλουθο :



Ρολόι

Εργαστηριακή Άσκηση 3

i. Σκοπός

Στην άσκηση αυτή καλείστε να φτιάξετε ένα ρολόι δευτερολέπτων, 2 ψηφίων, με λειτουργία "stopwatch". Το ρολόι σας θα ξεκινά από την τιμή 0 και θα αυξάνεται ανά 1 μέχρι την τιμή 59 (με συχνότητα 1 Hz), απ'οπου θα επιστρέφει πάλι στο 0.

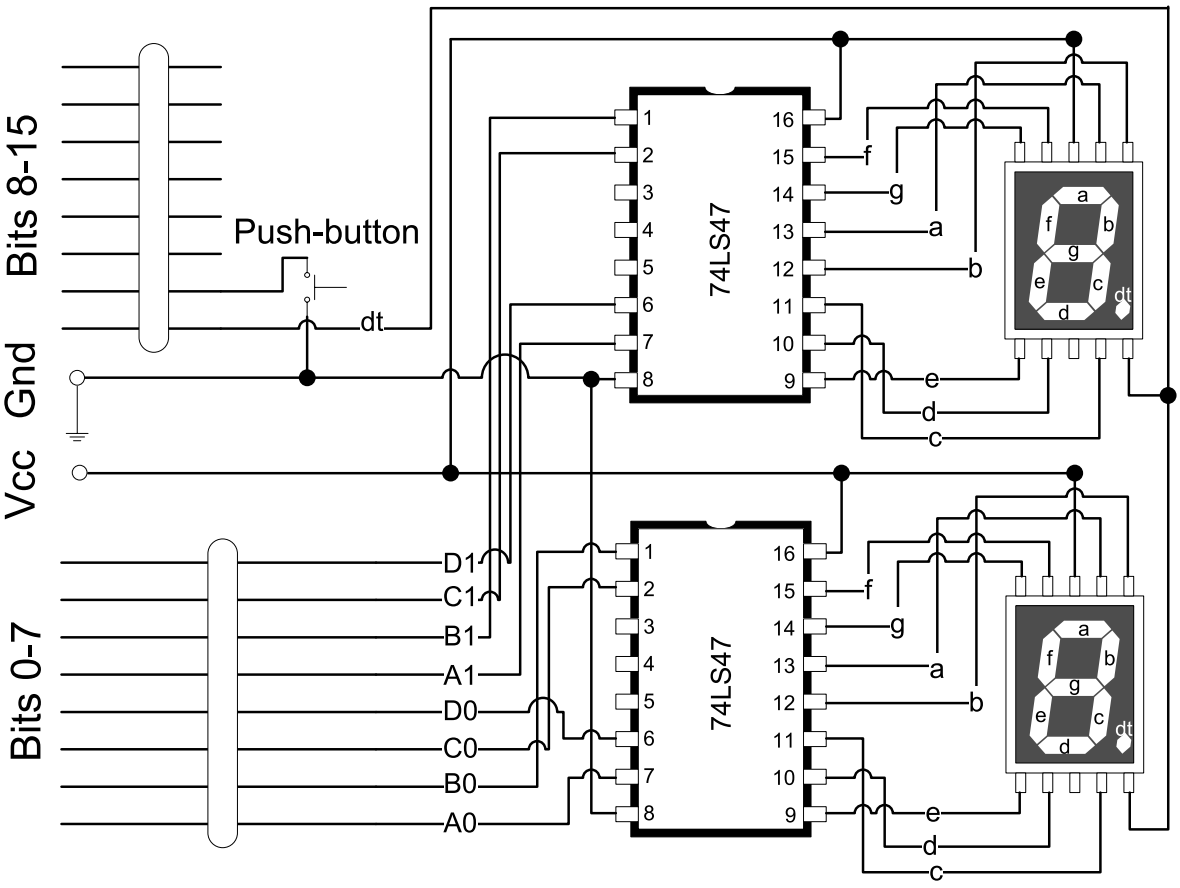
Το σύστημα ξεκινά από την κατάσταση IDLE, στην οποία το ρολόι ξεκινά να μετρά και τα 7-segment displays απεικονίζουν την τιμή του. Το πάτημα του εξωτερικού διακόπτη θα σταματά την ανανέωση των 7-segment displays (δηλαδή τα 7-segment displays θα διατηρούν την τελευταία τιμή του ρολογιού πριν το πάτημα του διακόπτη), ενώ το ρολόι εσωτερικά θα συνεχίζει να μετρά. Αυτή τη νέα κατάσταση θα την αποκαλούμε κατάσταση HOLD. Η κατάσταση αυτή υποδεικνύεται από το αναβόσβημα των τελειών των 7-segment displays με συχνότητα 2 Hz). Πάτημα του διακόπτη ενώ το σύστημα βρίσκεται σε κατάσταση HOLD, προκαλεί μετάβαση και πάλι στην κατάσταση IDLE, με διαρκή και πάλι απεικόνιση της τιμής του ρολογιού και απενεργοποίηση των τελειών. Τέλος, ανεξάρτητα από τη κατάσταση στην οποία βρίσκεται το σύστημα, πάτημα του διακόπτη για χρονικό διάστημα μεγαλύτερο του 1 δευτερολέπτου θα πρέπει να επαναφέρει το ρολόι στην τιμή 0.

ii. Κύκλωμα

Υλικά

Τα υλικά που θα χρειαστείτε για να υλοποιήσετε το κύκλωμα είναι 2 7-segment displays και 2 BCD to 7-segment display drivers.

Το λογικό διάγραμμα του κυκλώματος που θα πρέπει να αναπτύξετε στην πλακέτα επέκτασης είναι το ακόλουθο :



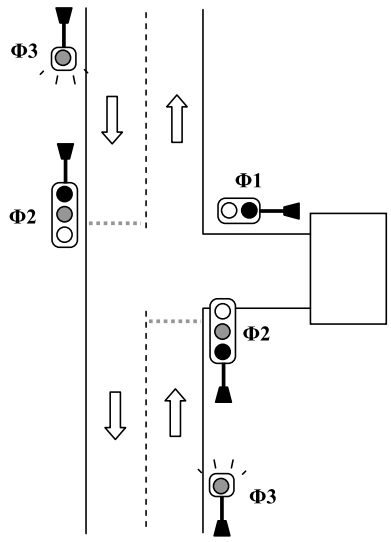
Οι ακροδέκτες 0 έως 8 της μονάδας εισόδου / εξόδου θα πρέπει να χρησιμοποιηθούν ως έξοδοι, ενώ ο ακροδέκτης 9 ως είσοδος με pull-up αντίσταση.

Φωτεινοί Σηματοδότες

Εργαστηριακή Άσκηση 4

i. Σκοπός

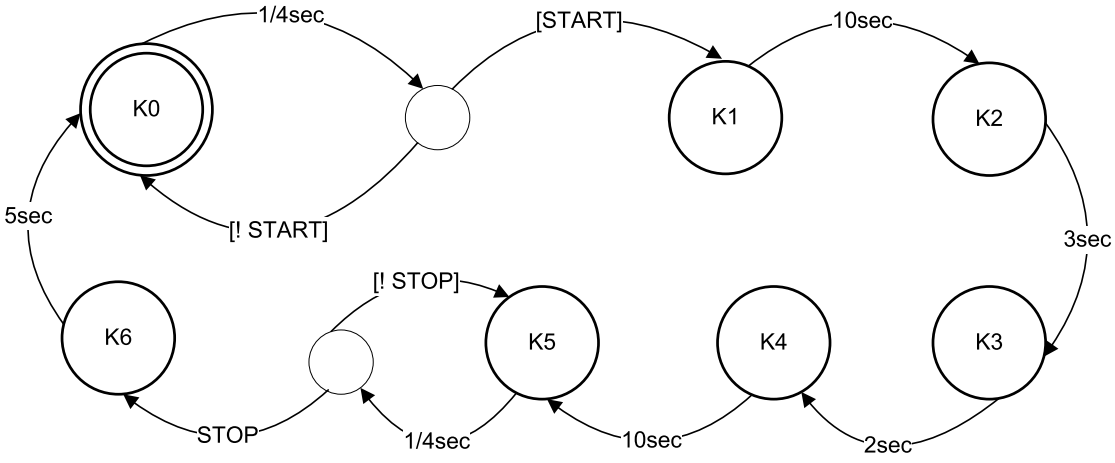
Σκοπός αυτής της εργαστηριακής άσκησης είναι η υλοποίηση ενός συστήματος ελέγχου φωτεινών σηματοδοτών, για ένα μικρό οδικό δίκτυο, με ειδικούς περιορισμούς (δες παρακάτω σχήμα). Ενας πυροσβεστικός σταθμός έχει έξοδο σε οδό διπλής κυκλοφορίας. Η κυκλοφορία στην οδό ρυθμίζεται από τους φωτεινούς σηματοδότες Φ2 (ΚΟ-ΚΙ-ΠΡ) και Φ3 (ΚΙ, αναβοσβήνει περιοδικά ανά 0,5 σες). Η έξοδος των πυροσβεστικών οχημάτων ρυθμίζεται από τον σηματοδότη Φ1 (ΚΟ-ΠΡ). Σε κανονικές συνθήκες ο Φ1 είναι κόκκινος, ο Φ2 πράσινος και ο Φ3 σβηστός. Η κυκλοφορία στην οδό σταματά με την πίεση ενός διακόπτη START, επιτρέποντας την έξοδο των πυροσβεστικών οχημάτων, και συνεχίζεται με την πίεση διακόπτη STOP. Για την αποφυγή σφαλμάτων χειρισμού ο Φ1 μένει πράσινος τουλάχιστον για ένα ορισμένο χρονικό διάστημα (βλ. πίνακα στην επόμενη σελίδα), ακόμα κι αν ο διακόπτης STOP πιεστεί νωρίτερα.



(ΚΟ=κόκκινος, ΚΙ=κίτρινος, ΠΡ=πράσινος, OFF=σβηστός, BLN=αναβοσβήνει περιοδικά ανά 0,5 sec)

κατάσταση	**Φ1**	**Φ2**	**Φ3**	**διάρκεια (sec)**	**επόμενη κατάσταση**	**παρατηρήσεις**
Κ0 (αρχική)	ΚΟ	ΠΡ	OFF	1/4	εάν πιέστηκε το START: Κ1 αλλιώς: Κ0	περιοδική ανίχνευση για πίεση START, ο διακόπτης αυτός θέτει ένα flag "request_service"
Κ1	ΚΟ	ΠΡ	BLN	10	Κ2	
Κ2	ΚΟ	ΚΙ	BLN	3	Κ3	
Κ3	ΚΟ	ΚΟ	BLN	2	Κ4	
Κ4	ΠΡ	ΚΟ	BLN	10	Κ5	τουλάχιστον 10 sec
Κ5	ΠΡ	ΚΟ	BLN	1/4	εάν πιέστηκε το STOP: Κ6 αλλιώς: Κ5	το STOP απενεργοποιεί το flag "request_service"
Κ6	ΚΟ	ΚΟ	OFF	5	Κ0	επιστροφή στην αρχική κατάσταση

Η ακολουθία ενδείξεων των σηματοδοτών ορίζεται από την ακόλουθη μηχανή πεπερασμένων καταστάσεων (Finite State Machine FSM).

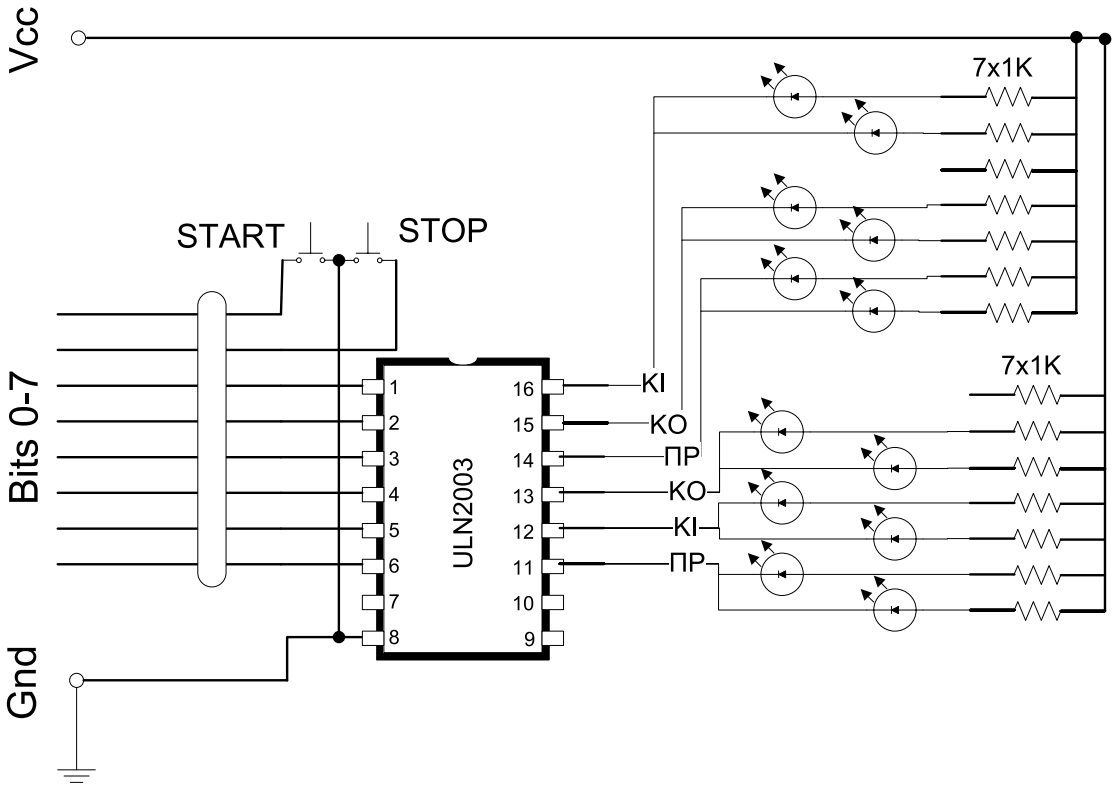


ii. Κύκλωμα

Υλικά

Τα υλικά που θα χρειαστείτε για την υλοποίηση του κυκλώματος είναι 2 πολλαπλές αντιστάσεις 1K, 2 διακόπτες τύπου push-button, 10 LEDs (3 ΚΟ, 3 ΠΡ, 4 ΚΙ) και 1 ULN2003 κύκλωμα οδήγησης.

Το λογικό διάγραμμα του κυκλώματος που θα πρέπει να αναπτύξετε στην πλακέτα επέκτασης είναι το ακόλουθο :



Οι ακροδέκτες 0 έως 6 της μονάδας εισόδου / εξόδου θα πρέπει να χρησιμοποιοηθούν ως έξοδοι, ενώ οι ακροδέκτες 7 και 8 ως είσοδοι με pull-up αντίσταση.

Παιχνίδι Pong

Εργαστηριακή Άσκηση 5

i. Σκοπός

Ο σκοπός αυτής της άσκησης είναι η σχεδίαση και ανάπτυξη του παιχνιδιού pong. Το παιχνίδι αυτό προσομοιώνει ένα παιχνίδι ping-pong, όπου ο κάθε παίκτης προσπαθεί να προλάβει το μπαλάκι που του πετά ο αντίπαλος, χτυπώντας το με μια ρακέτα. Στην ηλεκτρονική του έκδοση, το μπαλάκι αντικαθίσταται με ένα LED και η ρακέτα με ένα διακόπτη τύπου push-button. Οι κανόνες του παιχνιδιού είναι:

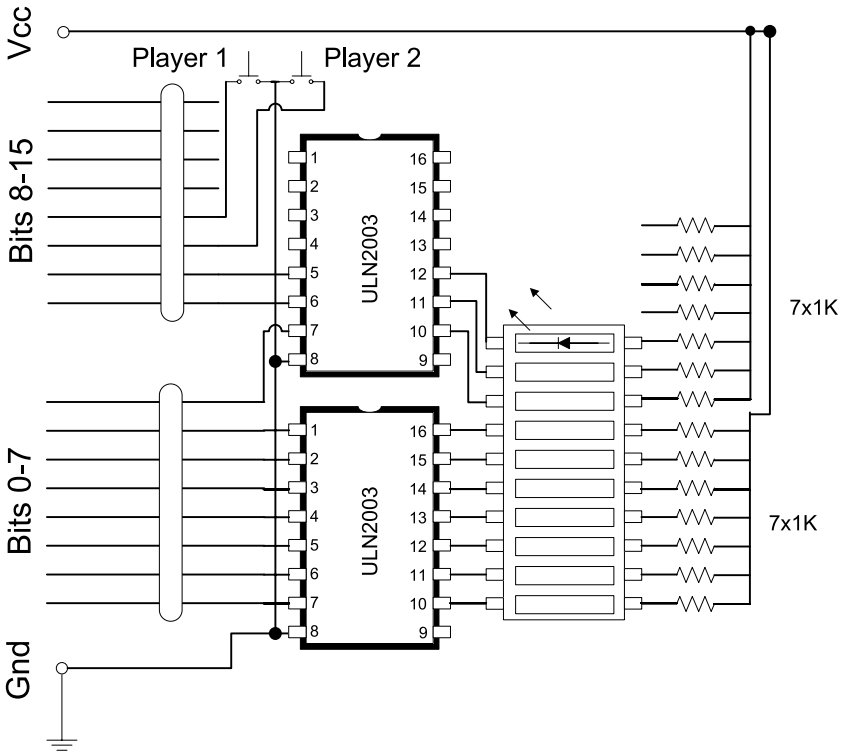
Το μπαλάκι ξεκινά από έναν παίκτη με συγκεκριμένη/σταθερή ταχύτητα και κινείται προς τον αντίπαλό του. Μόλις φτάσει στο τελευταίο σημείο της τροχιάς του, ο αντίπαλος καλείται να πιέσει τον διακόπτη του, ώστε να αναγκάσει το μπαλάκι να αλλάξει κατεύθυνση. Αυτά τα βήματα επαναλαμβάνονται μέχρι να χάσει ένας παίκτης το μπαλάκι (να μην προλάβει να πατήσει το διακόπτη του έγκαιρα). Ο αντίπαλος του παίκτη που δεν πίεσε το διακόπτη έγκαιρα, ανταμοίβεται με έναν πόντο. Μόλις κάποιος παίκτης συγκεντρώσει 3 πόντους θεωρείται νικητής και το παιχνίδι σταματά.

Το μπαλάκι θα προσομοιωθεί με τη χρήση του LED bar. Κάθε στιγμή θα είναι ενεργοποιημένο μόνο ένα από τα LEDs, το οποίο θα συμβολίζει τη θέση της μπάλας. Ανάλογα με την κατεύθυνση της μπάλας, θα σβήνει το τρέχον και θα ανάβει το επόμενο LED. Όταν το μπαλάκι φτάσει στα όρια και δεν έχει πατηθεί ο αντίστοιχος διακόπτης, θα απενεργοποιούνται όλα τα LEDs (κανένα από τα LEDs δεν θα παραμένει αναμένο). Η ανίχνευση της κατάστασης των διακοπτών (πατημένοι ή ελεύθεροι) θα γίνεται με polling και όχι με interrupts. Όταν ολοκληρώνεται το παιχνίδι, πρέπει να ενεργοποιούνται όλα τα LEDs του LED bar και να παραμένουν σε αυτή την κατάσταση μέχρι να ξεκινήσει νέος γύρος. Για να ξεκινήσει νέος γύρος παιχνιδιού, πρέπει να πατηθεί κάποιος διακόπτης. Ο παίκτης που θα πατήσει πρώτος το διακόπτη του για να ξεκινήσει το παιχνίδι, θα έχει αρχικά την μπάλα στην κατοχή του (η μπάλα θα ξεκινά από την πλευρά του προς τον αντίπαλο). Σχεδιάστε το διάγραμμα μετάβασης καταστάσεων και στη συνέχεια υλοποιήστε το στο σύστημα AT91.

ii. Κύκλωμα

Τα υλικά που θα χρειαστείτε για την υλοποίηση του κυκλώματος είναι 2 πολλαπλές αντιστάσεις 1K, 2 διακόπτες τύπου push-button, 1 LED bar και 2 ULN2003 κυκλώματα οδήγησης.

Το λογικό διάγραμμα του κυκλώματος που θα πρέπει να αναπτύξετε στην πλακέτα επέκτασης είναι το ακόλουθο :



Οι ακροδέκτες 0 έως 9 της μονάδας εισόδου / εξόδου θα πρέπει να χρησιμοποιηθούν ως έξοδοι, ενώ οι ακροδέκτες 10 και 11 ως είσοδοι με pull-up αντίσταση.

iii. Επιπλέον χαρακτηριστικά (Προαιρετικό)

Για να βελτιωθεί η ποιότητα του παιχνιδιού, είναι επιθυμητό να προστεθούν τα εξής χαρακτηριστικά :

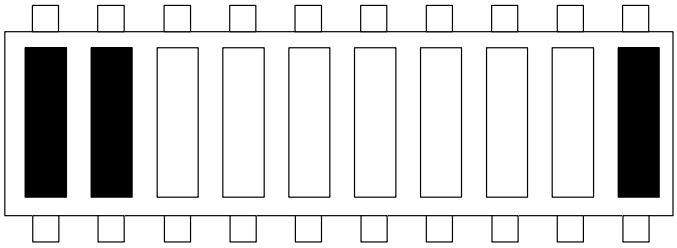
1. *Κάθε φορά που ένας παίκτης χάνει την μπάλα, θα εμφανίζεται το τρέχον σκορ για 1 δευτερόλεπτο και μετά θα συνεχίζεται το παιχνίδι.*

2. *Η ταχύτητα της μπάλας θα αυξάνεται μετά από 10 συνεχόμενες αλλαγές κατεύθυνσης. Το βήμα αύξησης της ταχύτητας μπορείτε να το επιλέξετε εσείς.*

3. *Για να αποφευχθούν περιπτώσεις συνεχούς πατήματος του διακόπτη, θα πρέπει να τεθούν όρια. Σε κάθε πάτημα, η διάρκεια του πατήματος δεν θα ξεπερνά το χρόνο που χρειάζεται η μπάλα να διανύσει 2 σημεία (δηλαδή να εμφανιστεί 2 LEDs παραπέρα) και δεν θα λαμβάνεται υπόψη οποιαδήποτε αλλαγή της κατάστασης του διακόπτη για χρόνο ίσο με το χρόνο που απαιτεί η μπάλα να διανύσει 4 σημεία.*

iv. Οδηγίες υλοποίησης

Για κάθε παίκτη, η απεικόνιση του σκορ θα γίνεται με την ενεργοποίηση LEDs του LED bar. Ξεκινώντας από το LED που αντιστοιχεί στην πλησιέστερη θέση της μπάλας προς τον κάθε παίκτη και συνεχίζοντας προς το κέντρο του LED bar, θα πρέπει να ενεργοποιούνται τόσα LEDs όσος είναι και ο αριθμός πόντων του κάθε παίκτη. Ένα παράδειγμα παρουσιάζεται στην επόμενη εικόνα, όπου ο παίκτης στα δεξιά έχει 1 πόντο και ο παίκτης στα αριστερά έχει 2 πόντους:

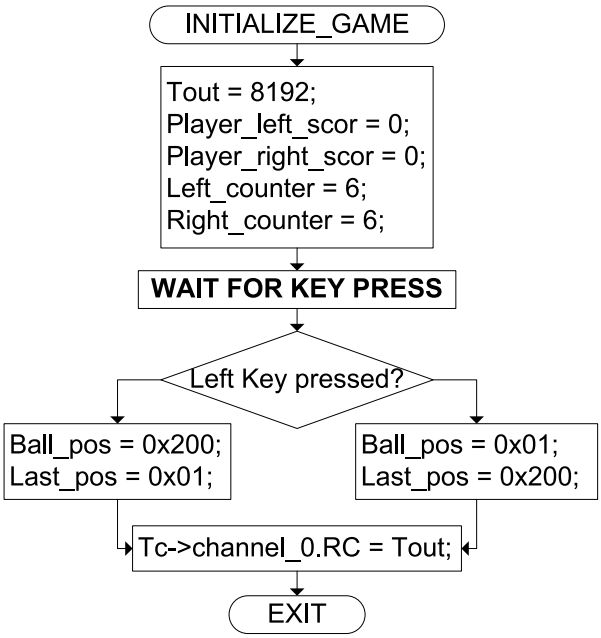


Η διάρκεια πατήματος μπορεί να υλοποιηθεί με τον εξής τρόπο:

Ορίστε μια μεταβλητή But_activeQ για κάθε διακόπτη, η οποία θα ενεργοποιείται όποτε ανιχνεύεται πάτημα του διακόπτη και θα απενεργοποιείται μετά από 2 χρονικές περιόδους της μπάλας, ανεξάρτητα από την κατάσταση του διακόπτη. Η απενεργοποίηση μπορεί να γίνει χρησιμοποιώντας μια μεταβλητή But_DownQ ως μετρητή, η οποία θα αρχικοποιείται στην τιμή 6 όταν ανιχνεύεται ένα πάτημα του διακόπτη και η But_DownQ έχει την τιμή 0. Η τιμή της μεταβλητής, όσο είναι μεγαλύτερη από το 0, θα μειώνεται κατά 1 μονάδα κάθε φορά που ολοκληρώνει τη μέτρησή του ο Timer (δηλαδή μέσα στη ρουτίνα εξυπηρέτησης). Μόλις η τιμή της But_DownQ γίνει μικρότερη του 4, η But_activeQ πρέπει να απενεργοποιηθεί. Στο πρόγραμμα, στα σημεία όπου ελέγχεται η τρέχουσα κατάσταση του διακόπτη, χρησιμοποιείστε την μεταβλητή But_activeQ αντί για την τιμή που λαμβάνετε από την γραμμή της μονάδας εισόδου/εξόδου που είναι συνδεδεμένη με τον διακόπτη.

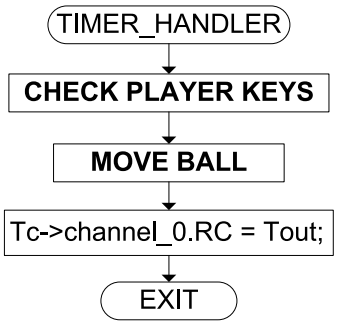
Από τη στιγμή που ανιχνεύεται το πάτημα του διακόπτη, η δυνατότητα επόμενης ανίχνευσης πρέπει να αναστέλεται για 6 χρονικές περιόδους της μπάλας. Αυτό μπορεί

να υλοποιηθεί πάλι με τη χρήση της μεταβλητής But_downQ. Μόλις ανιχνευτεί το πάτημα του διακόπτη, πρέπει να γίνει έλεγχος της τιμής της But_downQ. Αν είναι μεγαλύτερη του 0 δεν θα γίνει καμία ανανέωση της μεταβλητής But_activeQ. Αν έχει την τιμή 0, τότε και μόνο τότε θα πρέπει να τεθεί η But_activeQ στην τιμή 1.



Σχήμα 5.1: Αρχικοποίηση των μεταβλητών κατά την έναρξη του κάθε παιχνιδιού

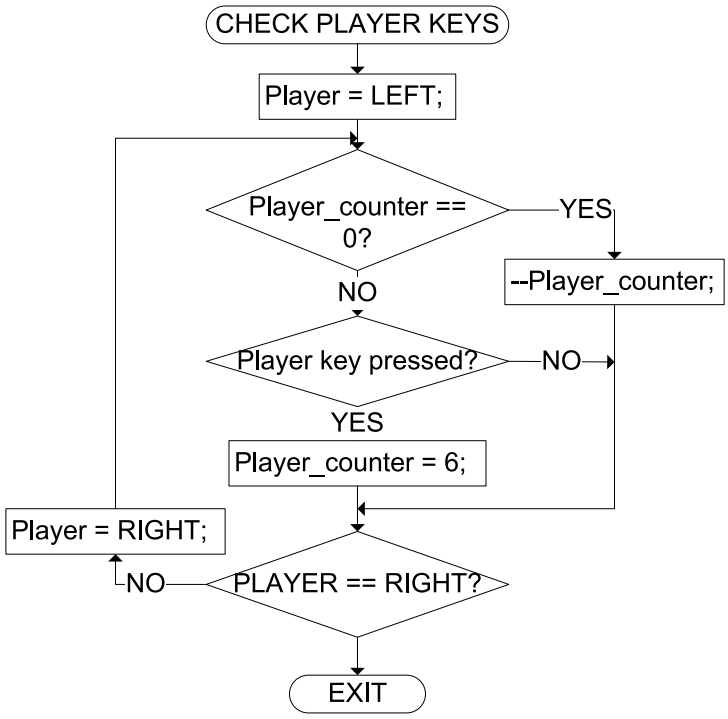
Τα ενδεικτικά βήματα που περιγράφει το διάγραμμα ροής 5.1 πρέπει να εκτελούνται κατά την αρχικοποίηση των μεταβλητών της μηχανής πεπερασμένων καταστάσεων, όταν δηλαδή ξεκινά ένας γύρος παιχνιδιού.



Σχήμα 5.2: Ρουτίνα εξυπηρέτησης διακοπών του μετρητή συστήματος

Το διάγραμμα 5.2 περιγράφει τα βήματα μιας ενδεικτικής ρουτίνας εξυπηρέτησης διακοπών για τον μετρητή του συστήματος. Οι καταστάσεις Check_player_keys και Move_ball

αναλύονται με περισσότερες λεπτομέρειες στα διαγράμματα 5.3 και 5.4 αντίστοιχα.

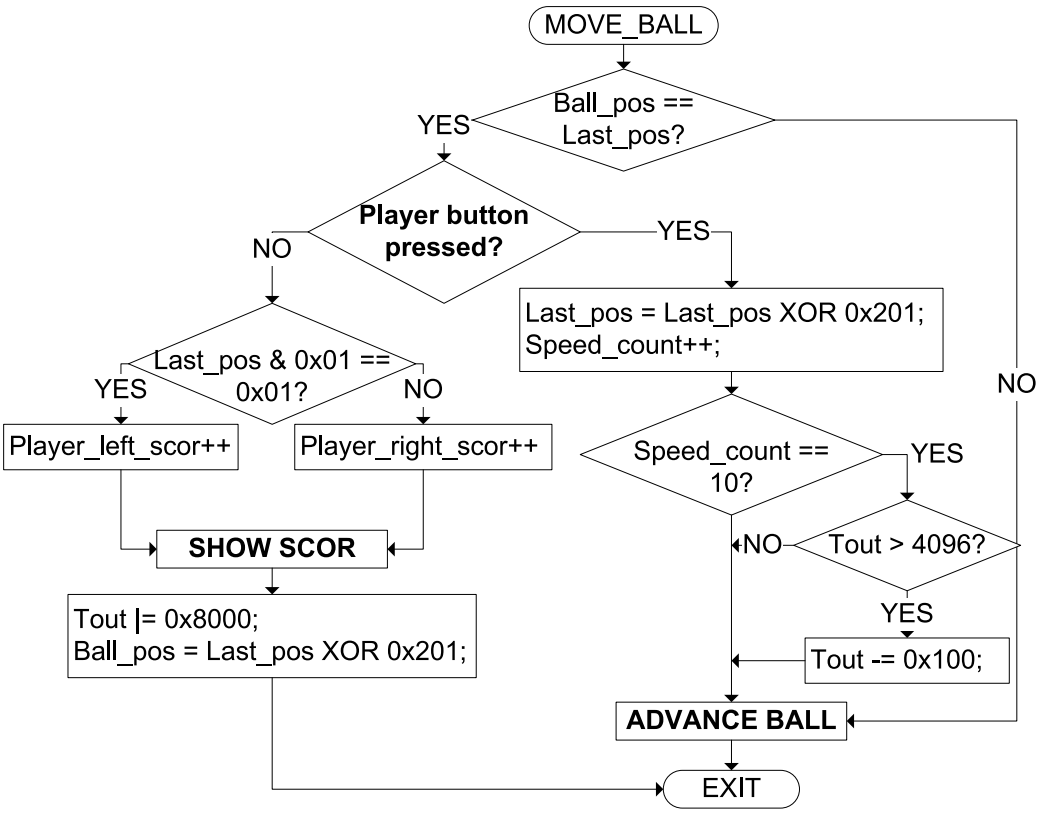


Σχήμα 5.3: Έλεγχος χρονικών περιορισμών σχετικά με το πάτημα των διακοπτών

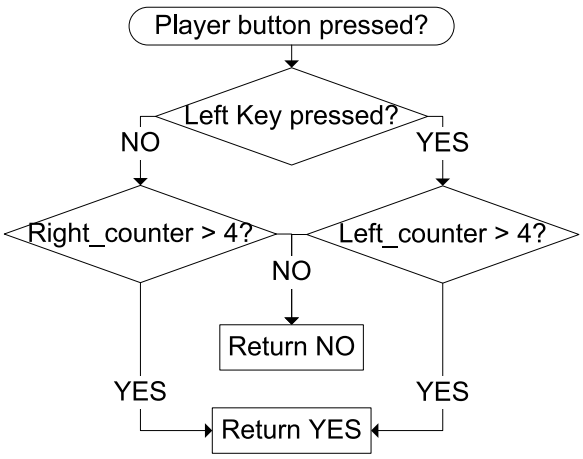
Το διάγραμμα 5.3 παραθέτει τα ενδεικτικά βήματα που χρειάζεται να εκτελεστούν, ώστε να γίνει ο έλεγχος για τους χρονικούς περιορισμούς που τίθενται, όσον αφορά τα χρονικά διαστήματα κατά τα οποία οι διακόπτες θεωρούνται πατημένοι ή όχι.

Το διάγραμμα 5.4 παραθέτει τα ενδεικτικά βήματα που χρειάζεται να εκτελεστούν, ώστε να γίνει ο έλεγχος για τους χρονικούς περιορισμούς που τίθενται, όσον αφορά τα χρονικά διαστήματα κατά τα οποία οι διακόπτες θεωρούνται πατημένοι ή όχι.

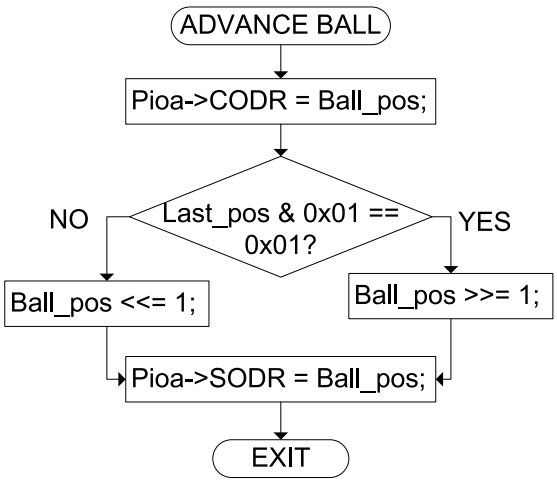
Το διάγραμμα 5.5 παραθέτει τα ενδεικτικά βήματα που χρειάζεται να εκτελεστούν, για ανιχνευθεί το αν έχει πατηθεί κάποιος από τους διακόπτες.



Σχήμα 5.4: Υπολογισμός της νέας θέσης της μπάλας



Σχήμα 5.5: Έλεγχος για πατημένους διακόπτες



Σχήμα 5.6: Απεικόνιση της μπάλας στο LED bar

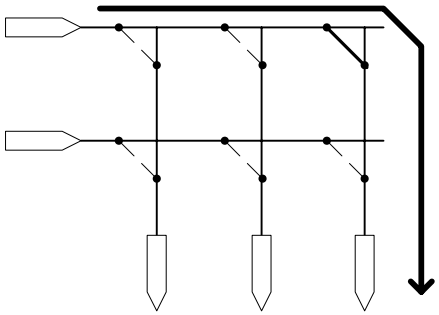
Πληκτρολόγιο

Εργαστηριακή Άσκηση 6

i. Σκοπός

Ο σκοπός αυτής της εργαστηριακής άσκησης είναι η υλοποίηση ενός mini πληκτρολογίου (keypad). Η τεχνική που θα χρησιμοποιηθεί είναι η σάρωση (scan-code). Παρότι το mini πληκτρολόγιό μας αποτελείται από ένα μικρό σύνολο πλήκτρων, ο αριθμός τους είναι σχετικά μεγάλος σε σχέση με τις γραμμές εισόδου/εξόδου του συστήματος. Για το λόγο αυτό θα χρησιμοποιήσουμε τεχνικές πολύπλεξης των γραμμών εισόδου/εξόδου. Το πλεονέκτημα της πολύπλεξης είναι ο περιορισμός του αριθμού των απαιτούμενων γραμμών εισόδου/εξόδου για τη διασύνδεση του πληκτρολογίου. Το μειονέκτημα είναι η κατάργηση της δυνατότητας ταυτόχρονης πίεσης δύο ή περισσότερων πλήκτρων του.

Η πολύπλεξη ορίζει πως τα πλήκτρα ομαδοποιούνται σε γραμμές και στήλες, σχηματίζοντας ένα ορθογώνιο πλέγμα (δες την εικόνα που ακολουθεί), όπου κάθε γραμμή και κάθε στήλη συνδέεται με γραμμές εισόδου/εξόδου. Τα πλήκτρα της κάθε γραμμής έχουν τον πρώτο ακροδέκτη τους συνδεδεμένο στην ίδια γραμμή εισόδου/εξόδου, ενώ τα πλήκτρα της κάθε στήλης έχουν τον δεύτερο ακροδέκτη τους συνδεδεμένο στην ίδια γραμμή εισόδου/εξόδου. Όπως μπορείτε να διαπιστώσετε, ο μέγιστος αριθμός πλήκτρων σε ένα τέτοιο πλέγμα είναι ίσος με το γινόμενο γραμμών και στηλών. Οι γραμμές εισόδου/εξόδου που συνδέονται με τις γραμμές του πλέγματος ρυθμίζονται σε λειτουργία εξόδου, ενώ οι γραμμές εισόδου/εξόδου που συνδέονται με τις στήλες του πλέγματος ρυθμίζονται σε λειτουργία εισόδου.

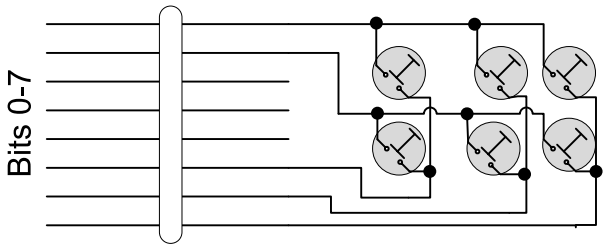


Όπως παρατηρείτε στο διάγραμμα, υπάρχουν 2 γραμμές και 3 στήλες, οι οποίες σχηματίζουν ένα πλέγμα. Οι γραμμές και οι στήλες δεν συνδέονται απευθείας, αλλά μέσω των διακοπτών που τοποθετούνται στα σημεία διασταύρωσης των γραμμών και συμβολίζονται με την πλάγια γραμμή. Ο διακόπτης της πρώτης γραμμής και τρίτης στήλης έχει πατηθεί και έχει κλείσει το κύκλωμα που σχηματίζεται ανάμεσα στον ακροδέκτη που οδηγεί την πρώτη γραμμή και τον ακροδέκτη που οδηγείται από την τρίτη στήλη. Οι ακροδέκτες που είναι συνδεδεμένοι με τις στήλες 1 και 2 δεν οδηγούνται, διότι δεν έχει πατηθεί κάποιος διακόπτης που να κλείνει το κύκλωμα ανάμεσα σε αυτούς και τους ακροδέκτες που οδηγούν τις γραμμές.

ii. Κύκλωμα

Υλικά

Τα υλικά που θα χρειαστείτε για την υλοποίηση του κυκλώματος είναι 6 διακόπτες τύπου push-button.



iii. Οδηγίες υλοποίησης

Επειδή οι γραμμές εισόδου δεν οδηγούνται όταν οι διακόπτες δεν είναι πατημένοι, είναι απαραίτητο να ενεργοποιηθούν οι εσωτερικές pull-up αντιστάσεις. Κάθε φορά που είναι επιθυμητή η ανάγνωση της κατάστασης του keypad, εκτελούνται τα παρακάτω βήματα, τόσες φορές όσες και ο αριθμός των γραμμών του πλέγματος:

Σε κάθε επανάληψη, μια από τις γραμμές εξόδου (που συνδέεται σε μια γραμμή του πλέγματος) οδηγείται με χαμηλό δυναμικό και οι υπόλοιπες γραμμές εξόδου οδηγούνται με υψηλό δυναμικό.
Εκτελείται ανάγνωση της κατάστασης των γραμμών εισόδου και εντοπίζεται αν υπάρχει κάποια γραμμή με χαμηλό δυναμικό. Αν εντοπιστεί τέτοια γραμμή, σημαίνει πως είναι πατημένο το πλήκτρο στην γραμμή που οδηγήθηκε και στην στήλη όπου εμφανίζεται χαμηλό δυναμικό.

Ο αριθμός της γραμμής του πατημένου πλήκτρου και ο αριθμός της στήλης του, συνδιάζονται μεταξύ τους για να παράγουν ένα διακριτό/μοναδικό αριθμό ο οποίος αντιστοιχεί στο πλήκτρο που πατήθηκε (ονομαζεται scan-code). Ένας εύκολος τρόπος παραγωγής του διακριτού κωδικού είναι ο πολλαπλασιασμός του αριθμού της γραμμής του πλέγματος του πατημένου πλήκτρου με τον αριθμό των πλήκτρων που υπάρχουν σε κάθε γραμμή και η πρόσθεση του αριθμού της στήλης του πατημένου πλήκτρου. Για παράδειγμα, αν σε ένα πλέγμα 4x4 πατηθεί το πλήκτρο (2,1), ο διακριτός κωδικός του πλήκτρου είναι 2*4 + 1 = 9 (η αρίθμηση γραμμών και στηλών γίνεται ξεκινώντας από το 0).

Η ανάγνωση της κατάστασης του keypad είναι επιθυμητό να γίνεται ανά τακτά χρονικά διαστήματα, κάτι που μπορεί να υλοποιηθεί με τη χρήση του Timer και με συχνότητα 5 Hz. Η επαναληπτική διαδικασία ανίχνευσης θα γίνει μέσα στην ρουτίνα εξυπηρέτησης διακοπών. Το scan-code που υπολογίζεται μπορεί να καταχωρηθεί σε μια καθολική μεταβλητή[1] και να γίνεται συνεχής έλεγχος στον κεντρικό βρόγχο για το πότε η μεταβλητή έχει τιμή διαφορετική από 255. Όταν ανιχνευτεί ότι η τιμή της μεταβλητής είναι διαφορετική του 255, πρέπει η τιμή της να προβάλεται στο lcd (με τη χρήση της συνάρτησης printf) και στη συνέχεια η μεταβλητή να τίθεται πάλι στην τιμή 255.

[1] Μεταβλητή δηλωμένη εκτός συναρτήσεων και προσπελάσιμη από όλες τις συναρτήσεις.

Παράρτημα Α΄

HEADER.H

```
typedef volatile struct _PIO
{
    unsigned int    PER;         /** PIO Enable register      (Write-only)*/
    unsigned int    PDR;         /** PIO Disable register     (Write-only)*/
    unsigned int    PSR;         /** PIO Status register      (Read-only)*/
    unsigned int    Reserved0;   /** Unused register */

    unsigned int    OER;         /** Output Enable register       (Write-only)*/
    unsigned int    ODR;         /** Output Disable register      (Write-only)*/
    unsigned int    OSR;         /** Output Status register       (Read-only) */
    unsigned int    Reserved1;   /** Unused register */

    unsigned int    IFER;        /** Glitch Input filter Enable (Write-only)*/
    unsigned int    IFDR;        /** Glitch Input filter Disable(Write-only)*/
    unsigned int    IFSR;        /** Glitch Input filter Status (Read-only)*/
    unsigned int    Reserved2;   /** Unused register */

    unsigned int    SODR;        /** Set Output Data register     (Write-only)*/
    unsigned int    CODR;        /** Clear Output Data register   (Write-only)*/
    unsigned int    ODSR;        /** Output Data Status register (Read-only)*/
    unsigned int    PDSR;        /** Pin Data Status register     (Read-only)*/

    unsigned int    IER;         /** Interrupt Enable register    (Write-only)*/
    unsigned int    IDR;         /** Interrupt Disable register   (Write-only)*/
    unsigned int    IMR;         /** Interrupt Mask register      (Write-only)*/
    unsigned int    ISR;         /** Interrupt Status register    (Read-only)*/

    unsigned int    MDER;        /** Multi-driver Enable (Write-only)*/
    unsigned int    MDDR;        /** Multi-driver Disable(Write-only)*/
    unsigned int    MDSR;        /** Multi-driver Status (Read-only)*/
    unsigned int    Reserved3;   /** Unused register*/

    unsigned int    PUDR;        /** Pull-up Disable register(Write-only)*/
    unsigned int    PUER;        /** Pull-up Enable register (Write-only)*/
    unsigned int    PUSR;        /** Pull-up Status register (Read-only)*/
    unsigned int    Reserved4;   /** Unused register*/

    unsigned int    ASR;             /** Peripheral A select (Write-only)*/
    unsigned int    BSR;             /** Peripheral B select (Write-only)*/
    unsigned int    ABSR;            /** Peripheral AB Status(Read-only)*/
    unsigned int    Reserved5[9];    /** Unused register*/

    unsigned int    OWER;            /** Output write enable (Write-only)*/
```

```
    unsigned int    OWDR;           /** Output write disable(Write-only)*/
    unsigned int    OWSR;           /** Output write Status (Read-only)*/
}PIO;

typedef volatile struct _AIC
{
    unsigned int    SMR[32];        /** Source mode register    (Read-Write)*/
    unsigned int    SVR[32];        /** Source vector register  (Read-Write)*/

    unsigned int    IVR;            /** Interrupt vector register       (Read-only)*/
    unsigned int    FVR;            /** Fast Interrupt vector register  (Read-only)*/
    unsigned int    ISR;            /** Interrupt status register       (Read-only)*/
    unsigned int    IPR;            /** Interrupt pending register      (Read-only)*/
    unsigned int    IMR;            /** Interrupt mask register         (Read-only)*/
    unsigned int    CISR;           /** Core Interrupt status register  (Read-only)*/
    unsigned int    Reserved1[2];   /** Unused register */

    unsigned int    IECR;           /** Interrupt enable command register   (Write-only)*/
    unsigned int    IDCR;           /** Interrupt disable command register  (Write-only)*/
    unsigned int    ICCR;           /** Interrupt clear command register    (Write-only)*/
    unsigned int    ISCR;           /** Interrupt set command register      (Write-only)*/
    unsigned int    EICR;           /** End of Interrupt command register   (Write-only)*/

    unsigned int    SPUR;           /** Spurious Interrupt vector register  (Read-Write)*/
    unsigned int    DCR;            /** Debug control register              (Read-Write)*/
    unsigned int    Reserved2;      /** Unused register*/

    unsigned int    FFER;           /** Fast Forcing enable register    (Write-only)*/
    unsigned int    FFDR;           /** Fast Forcing disable register   (Write-only)*/
    unsigned int    FFSR;           /** Fast Forcing status register    (Write-only)*/
}AIC;

typedef volatile struct _TCCHAN
{
    unsigned int    CCR;            /** Channel Control Register    (Write-only)*/
    unsigned int    CMR;            /** Channel Mode Register       (Read-Write)*/
    unsigned int    Reserved1[2];   /** Unused register*/
    unsigned int    CV;             /** Counter Value               (Read-only)*/
    unsigned int    RA;             /** Register A                  (Read-Write)*/
    unsigned int    RB;             /** Register B                  (Read-Write)*/
    unsigned int    RC;             /** Register C                  (Read-Write)*/
    unsigned int    SR;             /** Status Register             (Read-only)*/
    unsigned int    IER;            /** Interrupt Enable Register   (Write-only)*/
    unsigned int    IDR;            /** Interrupt Disable Register  (Write-only)*/
    unsigned int    IMR;            /** Interrupt Mask Register     (Read-only)*/
    unsigned int    Reserved2[4];   /** Unused register*/
}TCCHAN;

typedef volatile struct _TC
{
    TCCHAN  Channel_0;
    TCCHAN  Channel_1;
    TCCHAN  Channel_2;

    unsigned int    BCR;            /** Block Control Register  (Write-only)*/
    unsigned int    BMR;            /** Block Mode Register     (Read-Write)*/
}TC;

/// ΕΝΑΡΞΗ ΑΡΧΙΚΟΠΟΙΗΣΗΣ ΣΥΣΤΗΜΑΤΟΣ
/// ΠΡΕΠΕΙ ΝΑ ΓΙΝΕΤΑΙ ΠΑΝΤΑ ΣΤΗΝ ΑΡΧΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ
#define STARTUP                                                 \
    unsigned int _FIQtmp;                                       \
    /* ΑΠΑΡΑΙΤΗΤΟ ΓΙΑ ΠΡΟΣΠΕΛΑΣΗ ΜΝΗΜΗΣ ΣΥΣΤΗΜΑΤΟΣ  */          \
```

```
    int fd = open("/dev/mem", O_RDWR | O_SYNC);                          \
    /* ΑΠΑΡΑΙΤΗΤΟ ΓΙΑ ΕΓΓΡΑΦΗ ΡΟΥΤΙΝΑΣ ΕΞΥΠΗΡΕΣΗΣ ΔΙΑΚΟΠΩΝ */          \
    int fd2 = open("/proc/FIQ", O_RDWR | O_SYNC);                        \
    /* ΕΝΕΡΓΟΠΟΙΗΣΗ ΔΙΚΑΙΩΜΑΤΩΝ ΠΡΟΣΠΕΛΑΣΗΣ ΣΤΗΝ*/ /**PIOA & AIC*/     \
    char* mptr = mmap(0, 0x1000, PROT_READ | PROT_WRITE,                \
    MAP_SHARED, fd, 0xFFFFF000);                                         \
    /* ΑΠΑΡΑΙΤΗΤΟ ΓΙΑ ΠΡΟΣΠΕΛΑΣΗ ΤΩΝ*/ /**TIMERS*/                      \
    char* pptr = mmap(0, 0x1000, PROT_READ | PROT_WRITE,                 \
    MAP_SHARED, fd, 0xFFFA0000);                                         \
    /* ΕΛΕΓΧΟΣ ΑΠΟΚΤΗΣΗΣ ΔΙΚΑΙΩΜΑΤΩΝ */                                  \
    if(mptr == MAP_FAILED || pptr == MAP_FAILED){                        \
        close(fd);                                                       \
        close(fd2);                                                      \
        exit(1);                                                         \
    }                                                                    \
    /* ΟΡΙΖΟΥΜΕ ΔΙΕΥΘΥΝΣΗ ΤΗΣ*/ /**PIOA*/                                 \
    pioa = (PIO*)(mptr + 0x400);                                         \
    /* ΟΡΙΖΟΥΜΕ ΔΙΕΥΘΥΝΣΗ ΤΗΣ*/ /**AIC*/                                  \
    aic = (AIC*)(mptr);                                                  \
    /* ΟΡΙΖΟΥΜΕ ΔΙΕΥΘΥΝΣΗ ΤΟΥ*/ /**TC*/                                  \
    tc = (TC*)(pptr);                                                    \
    _FIQtmp = (unsigned int)FIQ_handler;                                 \
    write(fd2, &_FIQtmp, sizeof(unsigned int));                          \
    _FIQtmp = fcntl(STDIN_FILENO, F_GETFL, 0);                           \
    _FIQtmp |= O_NONBLOCK;                                               \
    fcntl(STDIN_FILENO, F_SETFL, _FIQtmp)
/// ΛΗΞΗ ΑΡΧΙΚΟΠΟΙΗΣΗΣ

/// ΕΝΑΡΞΗ ΕΠΑΝΑΦΟΡΑΣ ΣΥΣΤΗΜΑΤΟΣ
/// ΠΡΕΠΕΙ ΝΑ ΓΙΝΕΤΑΙ ΠΑΝΤΑ ΣΤΟ ΤΕΛΟΣ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ
#define CLEANUP                                                          \
    _FIQtmp = fcntl(STDIN_FILENO, F_GETFL, 0);                           \
    _FIQtmp &=  O_NONBLOCK;                                              \
    fcntl(STDIN_FILENO, F_SETFL, _FIQtmp);                               \
    /* ΕΠΙΣΤΡΟΦΗ ΔΕΣΜΕΥΜΕΝΗΣ ΜΝΗΜΗΣ */                                   \
    munmap(mptr, 0x1000);                                                \
    munmap(pptr, 0x1000);                                                \
    close(fd);                                                           \
    close(fd2)
/// ΛΗΞΗ ΕΠΑΝΑΦΟΡΑΣ ΣΥΣΤΗΜΑΤΟΣ

#define DISABLE_FIQ                 \
    {                               \
        unsigned int __Reg_save;    \
        asm("mrs␣%0,␣cpsr;"         \
            "orr␣%0,␣%0,␣#0x40;"    \
            "msr␣cpsr_c,␣%0;"       \
            :"=r" (__Reg_save)      \
            :"r"(__Reg_save)        \
            );}

#define ENABLE_FIQ                  \
    {                               \
        unsigned int __Reg_save;    \
        asm("mrs␣%0,␣cpsr;"         \
            "bic␣%0,␣%0,␣#0x40;"    \
            "msr␣cpsr_c,␣%0;"       \
            :"=r" (__Reg_save)      \
            :"r"(__Reg_save)        \
            );}
```

Παράρτημα Β΄

Μονάδα εισόδου/εξόδου

iv. Γενικά

Το σύστημα AT91 παρέχει τη δυνατότητα διασύνδεσης με εξωτερικές περιφερειακές συσκευές, μέσω 96 προγραμματιζόμενων γραμμών επικοινωνίας. Για τη διαχείριση της λειτουργίας των γραμμών, το σύστημα διαθέτει τρεις εξειδικευμένες περιφερειακές μονάδες (Parallel Input / Output - PIO), τις μονάδες ελέγχου εισόδου / εξόδου A, B και C. Η κάθε μια από αυτές τις τρεις μονάδες ελέγχει 32 γραμμές επικοινωνίας. Για τις ανάγκες του εργαστηρίου, χρησιμοποιείται μόνο η μονάδα ελέγχου εισόδου / εξόδου A (PIOA).

Κάθε γραμμή επικοινωνίας μπορεί να προγραμματιστεί ανεξάρτητα από κάθε άλλη ως προς τη λειτουργία της. Ο καθορισμός των παραμέτρων λειτουργίας κάθε γραμμής επικοινωνίας (π.χ. λειτουργία γενικού σκοπού, λειτουργία εισόδου ή εξόδου κλπ.) γίνεται με την εγγραφή των κατάλληλων τιμών στους καταχωρητές της κάθε μονάδας ελέγχου εισόδου / εξόδου. Η κάθε γραμμή μπορεί να χρησιμοποιηθεί είτε σαν είσοδος / έξοδος γενικού σκοπού, είτε σαν είσοδος / έξοδος ελεγχόμενη από κάποιο ενσωματωμένο περιφεριακό. Σε όλες τις εργαστηριακές ασκήσεις, οι γραμμές εισόδου / εξόδου θα τίθενται σε λειτουργία γενικού σκοπού. Η λειτουργία κάθε γραμμής επικοινωνίας καθορίζεται από ένα bit στους καταχωρητές καθορισμού λειτουργίας της κάθε μονάδας. Αυτό σημαίνει πως σε κάθε καταχωρητή, το bit i καθορίζει τη συμπεριφορά της γραμμής i.

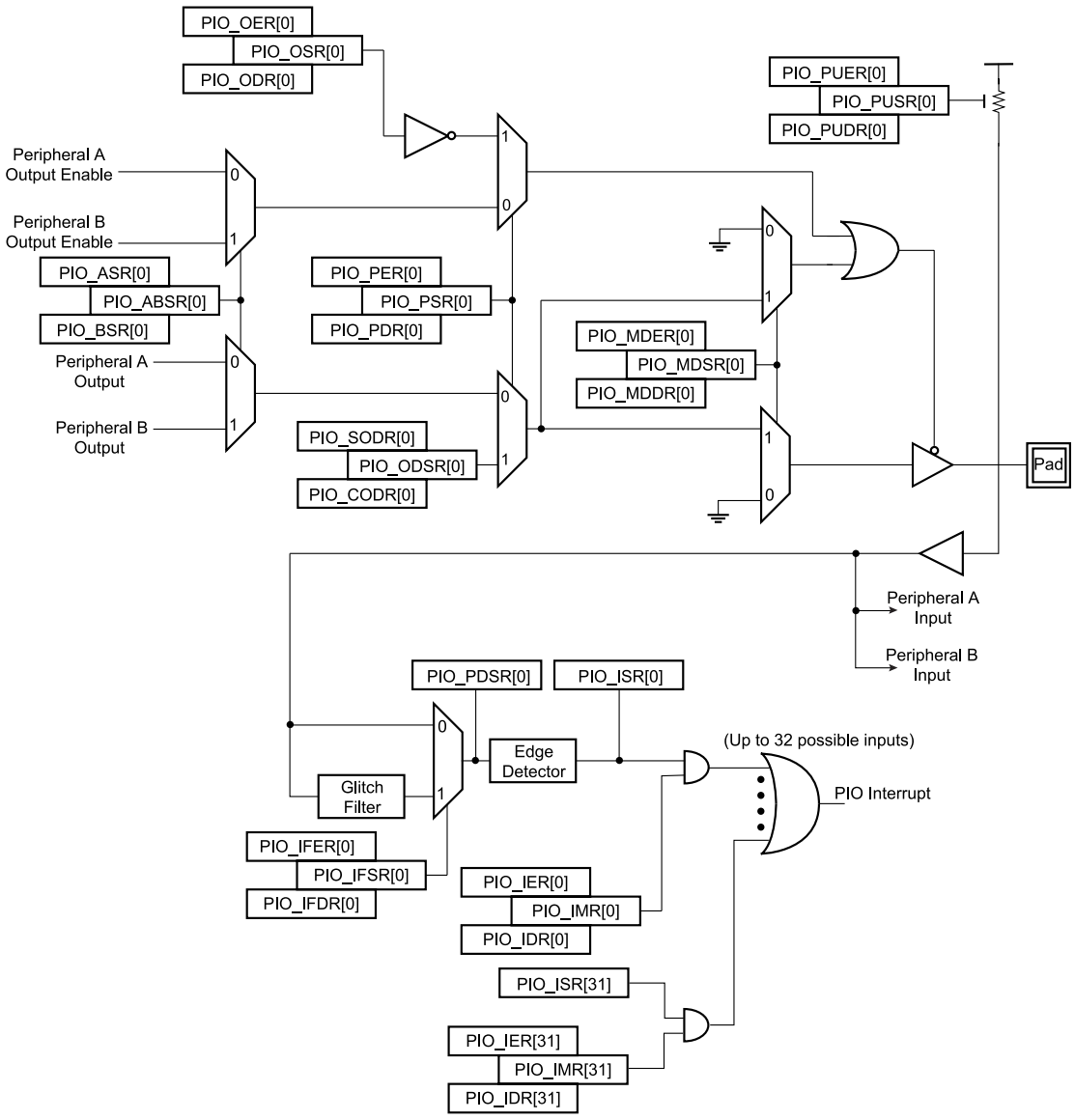
v. Καταχωρητές ελέγχου

Οι καταχωρητές καθορισμού λειτουργίας αναγράφονται στον ακόλουθο πίνακα. Η στήλη Access καθορίζει τους δυνατούς τρόπους προσπέλασης κάθε καταχωρητή. Η φυσική υλοποίηση του κυκλώματος που ελέγχει τη λειτουργία της γραμμής επικοινωνίας 0 φαίνεται στην εικόνα Β΄.1 και αποτελείται κατά μεγάλο μέρος από πολυπλέκτες και τα δυαδικά

ψηφία των καταχωρητών ελέγχου στη θέση 0. Αντίστοιχα κυκλώματα ελεγχόμενα από τα κατάλληλα δυαδικά ψηφία των καταχωρητών ελέγχου υπάρχουν για κάθε γραμμή επικοινωνίας (υπάρχουν δηλαδή 96 αντίτυπα του κυκλώματος της εικόνας Β'.1.

Φυσική διεύθυνση: 0xFFFFF400

Offset	Register Name	Access
0x0000	PIO Enable Register PIO_PER	Write-only
0x0004	PIO Disable Register PIO_PDR	Write-only
0x0008	PIO Status Register PIO_PSR	Read-only
0x000C	Reserved	
0x0010	Output Enable Register PIO_OER	Write-only
0x0014	Output Disable Register PIO_ODR	Write-only
0x0018	Output Status Register PIO_OSR	Read-only
0x001C	Reserved	
0x0020	Glitch Input Filter Enable Register PIO_IFER	Write-only
0x0024	Glitch Input Filter Disable Register PIO_IFDR	Write-only
0x0028	Glitch Input Filter Status Register PIO_IFSR	Read-only
0x002C	Reserved	
0x0030	Set Output Data Register PIO_SODR	Write-only
0x0034	Clear Output Data Register PIO_CODR	Write-only
0x0038	Output Data Status Register PIO_ODSR	Read-only
0x003C	Pin Data Status Register PIO_PDSR	Read-only
0x0040	Interrupt Enable Register PIO_IER	Write-only
0x0044	Interrupt Disable Register PIO_IDR	Write-only
0x0048	Interrupt Mask Register PIO_IMR	Read-only
0x004C	Interrupt Status Register PIO_ISR	Read-only
0x0050	Multi-driver Enable Register PIO_MDER	Write-only
0x0054	Multi-driver Disable Register PIO_MDDR	Write-only
0x0058	Multi-driver Status Register PIO_MDSR	Read-only
0x005C	Reserved	
0x0060	Pull-up Disable Register PIO_PUDR	Write-only
0x0064	Pull-up Enable Register PIO_PUER	Write-only
0x0068	Pad Pull-up Status Register PIO_PUSR	Read-only
0x006C	Reserved	
0x0070	Peripheral A Select Register PIO_ASR	Write-only
0x0074	Peripheral B Select Register PIO_BSR	Write-only
0x0078	AB Status Register PIO_ABSR	Read-only
0x007C - 0x009C	Reserved	
0x00A0	Output Write Enable PIO_OWER	Write-only
0x00A4	Output Write Disable PIO_OWDR	Write-only
0x00A8	Output Write Status Register PIO_OWSR	Read-only



Σχήμα Β'.1: Διάγραμμα ελέγχου γραμμής

Σημείωση

Για να υπολογιστεί η φυσική διεύθυνση κάθε καταχωρητή, πρέπει να προστεθεί η μετατόπιση που αντιστοιχεί στον καταχωρητή στη φυσική διεύθυνση της μονάδας. Για παράδειγμα, η διεύθυνση του καταχωρητή PIO_OER είναι:

0x10 + 0xFFFFF400 = 0xFFFFF410

Υπενθυμίζεται ότι κάθε bit σε κάθένα από τους παραπάνω καταχωρητές αντιστοιχεί σε μια γραμμή επικοινωνίας. Το λιγότερο σημαντικό bit αντιστοιχεί στη γραμμή PIO_0 της κάθε μονάδας, ενώ το περισσότερο σημαντικό bit στη γραμμή PIO_31.

Κάθε γραμμή επικοινωνίας pin μπορεί να προγραμματιστεί σαν γραμμή γενικού σκοπού (1 στο αντίστοιχο bit του PIO_PER) ή σαν γραμμή ελεγχόμενη από κάποιο ενσωματωμένο περιφεριακό (1 στο αντίστοιχο bit του PIO_PDR). Η τρέχουσα κατάσταση εμφανίζεται στον PIO_PSR, όπου η τιμή 1 σημαίνει ότι η γραμμή είναι γενικού σκοπού και η τιμή 0 σημαίνει ότι η γραμμή ελέγχεται από περιφερειακό. Για παράδειγμα, με την εντολή

pioa->PIO_PER = 0x802;

θέτουμε τις γραμμές 1 και 11 σε κατάσταση γενικού σκοπού. Με την εντολή

status = pioa->PIO_PSR;

αποθηκεύουμε στη μεταβλητή status την τρέχουσα κατάσταση λειτουργίας.

Μια γραμμή γενικού σκοπού μπορεί να λειτουργήσει είτε σαν είσοδος (γράφοντας 1 στο αντίστοιχο bit του PIO_ODR), είτε σαν έξοδος (γράφοντας 1 στο αντίστοιχο bit του PIO_OER). Στον PIO_OSR αναφέρεται η τρέχουσα λειτουργία της κάθε γραμμής, όπου η τιμή 1 σε κάποιο bit σημαίνει ότι η αντίστοιχη γραμμή είναι έξοδος, ενώ η τιμή 0 σε κάποιο bit σημαίνει ότι η αντίστοιχη γραμμή είναι είσοδος.

Αν μια γραμμή γενικού σκοπού λειτουργεί σαν έξοδος, τα δεδομένα εξόδου της προέρχονται από τον καταχωρητή PIO_ODSR. Τα bits του καταχωρητή ενεργοποιούνται με την εγγραφή της τιμής 1 στην αντίστοιχη θέση του PIO_SODR και απενεργοποιούνται με την εγγραφή της τιμής 1 στην αντίστοιχη θέση του PIO_CODR. Για παράδειγμα, οι εντολές

pioa->PIO_SODR = 0x100;
pioa->PIO_CODR = 0x1;

θα θέσουν την έξοδο της 8ης γραμμής σε υψηλό δυναμικό και την έξοδο της γραμμής 0 σε χαμηλό δυναμικό. Μπορείτε να ανατρέχετε στην εικόνα Β'.1, για να βλέπετε την αντιστοιχία μεταξύ του προγραμματιστικού μοντέλου και της φυσικής υλοποίησης.

Υπάρχει και η δυνατότητα απευθείας εγγραφής στον PIO_ODSR, ώστε η ενεργοποίηση και απενεργοποίηση των εξόδων να γίνει με μια εντολή. Τα bits που επηρεάζονται από την απευθείας εγγραφή στον PIO_ODSR είναι όσα έχουν την τιμή 1 στον καταχωρητή PIO_OWSR. Τα bits του PIO_OWSR ενεργοποιούνται με την εγγραφή της τιμής 1 στα αντίστοιχα bits του PIO_OWER και απενεργοποιούνται με την εγγραφή της τιμής 1 στα αντίστοιχα bits του PIO_OWDR. Για παράδειγμα, αν ο PIO_OWSR έχει την τιμή 0xF1, η εγγραφή της τιμής 0xCF στον PIO_ODSR θα ενεργοποιήσει τις γραμμές 0,6 και 7 και θα απενεργοποιήσει τις γραμμές 4 και 5.

Αν μια γραμμή γενικού σκοπού λειτουργεί σαν είσοδος, η στάθμη του τρέχοντος δυναμικού εισόδου της μπορεί να αναγνωστεί από τον καταχωρητή PIO_PDSR, όπου η τιμή 1 σε κάποιο bit σημαίνει πως η αντίστοιχη γραμμή δέχεται σαν είσοδο υψηλό δυναμικό, ενώ η τιμή 0 σημαίνει πως η αντίστοιχη γραμμή δέχεται σαν είσοδο χαμηλό δυναμικό.

Σε πολλές εφαρμογές, η οδήγηση μιας γραμμής εισόδου δεν είναι συνεχής και υπάρχουν διαστήματα όπου η γραμμή δεν οδηγείται από κάποιο εξωτερικό περιφερειακό (η στάθμη του δυναμικού εισόδου της γραμμής είναι απροσδιόριστη). Ένα παράδειγμα είναι η διασύνδεση μιας γραμμής με ένα εξωτερικό διακόπτη, ο οποίος έχει το ένα άκρο του συνδεδεμένο στη γραμμή και το άλλο στη γείωση. Όταν δεν είναι πατημένος, η γραμμή δεν είναι συνδεδεμένη στο χαμηλό δυναμικό (γείωση). Αυτή η κατάσταση λειτουργίας πρέπει να αποφεύγεται, διότι μπορεί να προκαλέσει απροσδιόριστη συμπεριφορά στο σύστημα (π.χ. μπορεί να ανιχνευθεί χαμηλό δυναμικό χωρίς να έχει πατηθεί ο διακόπτης). Μια λύση σε αυτό το πρόβλημα είναι η ενεργοποίηση των εσωτερικών pull-up αντιστάσεων που συνδέουν την γραμμή με την τροφοδοσία, μέσω μιας αντίστασης 100K. Το δυναμικό εισόδου της γραμμής, όταν αυτή δεν οδηγείται, είναι υψηλό λόγω της διασύνδεσης της γραμμής με την τροφοδοσία. Όταν ο διακόπτης πατηθεί, το δυναμικό εισόδου θα γίνει χαμηλό (εφόσον δεν παρεμβάλεται μεγάλη αντίσταση ανάμεσα στη γραμμή και την γείωση).

Όπως καταλαβαίνετε, το μειονέκτημα αυτής της λύσης είναι η μεγάλη κατανάλωση που εμφανίζεται στο μονοπάτι ανάμεσα στην τροφοδοσία, την pull-up αντίσταση και τη γείωση. Γιάυτό και η pull-up αντίσταση πρέπει να ενεργοποιείται μόνο όταν δεν υπάρχει συνεχής οδήγηση της γραμμής. Ο καταχωρητής που ενεργοποιεί την pull-up αντίσταση είναι ο PIO_PUER (η εγγραφή της τιμής 1 σε bits αυτού του καταχωρητή θέτει τα αντίστοιχα bits του PIO_PUSR στην τιμή 0). Ο καταχωρητής που απενεργοποιεί την pull-up

αντίσταση είναι ο PIO_PUDR (η εγγραφή της τιμής 1 σε bits αυτού του καταχωρητή θέτει τα αντίστοιχα bits του PIO_PUSR στην τιμή 1). Στον καταχωρητή PIO_PUSR υπάρχει η τρέχουσα κατάσταση λειτουργίας των pull-up αντιστάσεων.

Μια γραμμή όταν προγραμματιστεί σαν είσοδος έχει τη δυνατότητα να παράγει interrupts αν ανιχνευθεί αλλαγή της κατάστασης του σήματος εισόδου (από 0 σε 1 και το αντίθετο). Οι γραμμές που μπορούν να παράγουν interrupts είναι αυτές για τις οποίες ο PIO_IMR έχει στα αντίστοιχα bits την τιμή 1. Τα bits αυτού του καταχωρητή ενεργοποιούνται με την εγγραφή της τιμής 1 στο αντίστοιχο bit του PIO_IER και απενεργοποιούνται με την εγγραφή της τιμής 1 στο αντίστοιχο bit του PIO_IDR.

Μόλις ανιχνευθεί η αλλαγή κατάστασης εισόδου, ενεργοποιείται το bit του PIO_ISR που αντιστοιχεί στη γραμμή που συνέβη η αλλαγή κατάστασης. Όσο είναι ενεργοποιημένο κάποιο bit του PIO_ISR, η μονάδα ελέγχου interrupts δέχεται αιτήσεις διακοπής. Συνεπώς, μόλις η ροή του προγράμματος μεταβεί στη ρουτίνα εξυπηρέτησης του interrupt, πρέπει να απενεργοποιηθούν τα ενεργοποιημένα bits του PIO_ISR. Ο καθαρισμός των bits αυτού του καταχωρητή γίνεται με την ανάγνωσή του (δηλαδή αν προσπελαστεί για να αναγνωστεί το περιεχόμενό του, τα bits του θα απενεργοποιηθούν αυτόματα και οποιαδήποτε επόμενη ανάγνωσή του, πριν δημιουργηθεί επόμενο interrupt, θα έχει σαν αποτέλεσμα το 0).

vi. Παράδειγμα

Υποθέστε πως η εφαρμογή που καλείστε να υλοποιήσετε απαιτεί τα εξής :

1. Οι ακροδέκτες 0, 4 και 12 να ρυθμιστούν σε λειτουργία εξόδου.
2. Οι ακροδέκτες 1, 2, 3, 5 και 8 να ρυθμιστούν σε λειτουργία εισόδου.
3. Οι γραμμές εισόδου 1 και 2 να παράγουν διακοπή σε κάθε αλλαγή της κατάστασης του δυναμικού εισόδου τους.
4. Η γραμμή εισόδου 1 είναι συνδεδεμένη σε διακόπτη, ενώ η γραμμή 2 είναι συνδεδεμένη σε εξωτερικό περιφερειακό (η γραμμή οδηγείται συνεχώς).
5. Οι γραμμές 12, 4 και 0 πρέπει να οδηγήσουν εξωτερικό κύκλωμα με τις τιμές: {010}, {110}, {000}, {101}.

Ρύθμιση λειτουργίας γραμμών επικοινωνίας.

pioa->PIO_PER = 0x113F; //Ρύθμιση των γραμμών σε λειτουργία γενικού σκοπού.

pioa->PIO_POER = 0x1011; //Ρύθμιση λειτουργίας εξόδου.
pioa->PIO_PODR = 0x12E; //Ρύθμιση λειτουργίας εισόδου.
pioa->PIO_PUER = 0x2; //Ενεργοποίηση της pull-up αντίστασης.

Οδήγηση εξωτερικού κυκλώματος.

1ος τρόπος	**2ος τρόπος**	**I/O 12**	**I/O 4**	**I/O 0**
pioa->PIO_CODR = 0x1001; **pioa->PIO_SODR = 0x10;**	**pioa->PIO_OWDR = 0xFFFFFFFF;** **pioa->PIO_OWER = 0x1011;** **pioa->PIO_ODSR = 0x10;**	0	1	0
pioa->PIO_CODR = 0x1; **pioa->PIO_SODR = 0x1010;**	**pioa->PIO_ODSR = 0x1010;**	1	1	0
pioa->PIO_CODR = 0x1011;	**pioa->PIO_ODSR = 0x00;**	0	0	0
pioa->PIO_CODR = 0x10; **pioa->PIO_SODR = 0x1001;**	**pioa->PIO_ODSR = 0x1001;**	1	0	1

Παράρτημα Γ΄

Μονάδα ελέγχου διακοπών

i. Γενικά

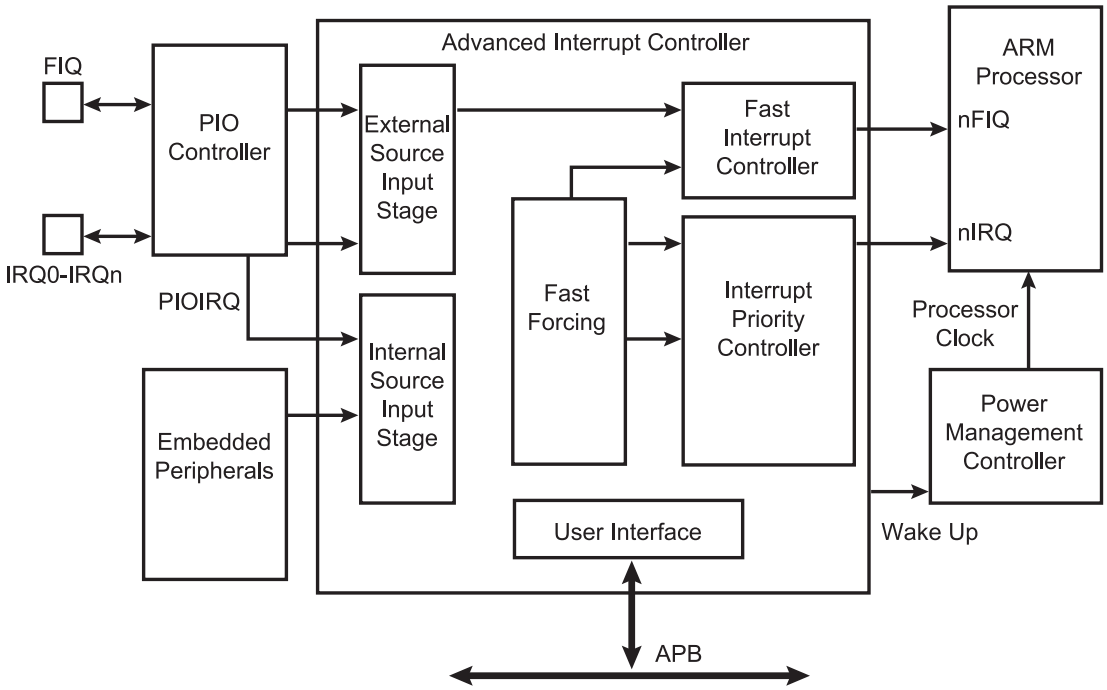
Οι διακοπές (interrupts) χρησιμοποιούνται ως σήματα εισόδου στον επεξεργαστή του συστήματος, για να του ανακοινώσουν πως κάποιο από τα περιφερειακά απαιτεί εξυπηρέτηση. Ο επεξεργαστής με τη σειρά του, διακόπτει τη ροή εκτέλεσης του προγράμματος και μεταβαίνει σε προσυμφωνημένη θέση μνήμης, ανάλογα με τον τύπο της διακοπής. Στο σύστημα AT91 ο επεξεργαστής μεταβαίνει στις εξής θέσεις μνήμης:

Όνομα διακοπής	**Διεύθυνση**	**Κατάσταση**	**Αιτία**
Reset	0xFFFF0000	Supervisor	Εκκίνηση του συστήματος
Undefined instruction	0xFFFF0004	Undefined	Εκτέλεση εντολής που δεν ανήκει στο σύνολο εντολών.
Software Interrupt	0xFFFF0008	Supervisor	Εκτέλεση εντολής SWI
Abort prefetch	0xFFFF000C	Abort	Λήψη εντολής από περιοχή μνήμης που δεν υπάρχουν δικαιώματα πρόσβασης
Abort data	0xFFFF0010	Abort	Λήψη δεδομένων από περιοχή μνήμης που δεν υπάρχουν δικαιώματα πρόσβασης
Reserved	0xFFFF0014	Reserved	
IRQ	0xFFFF0018	IRQ	Ενεργοποίηση γραμμής IRQ
FIQ	0xFFFF001C	FIQ	Ενεργοποίηση γραμμής FIQ

Πίνακας Γ.2: Διάνυσμα διακοπών

Οι πέντε πρώτες διακοπές σχετίζονται με τη συμπεριφορά του επεξεργαστή απέναντι σε βασικά σφάλματα εκτέλεσης. Οι δυο τελευταίες διακοπές σχετίζονται με σήματα διακοπής που στέλνει η μονάδα ελέγχου διακοπών στον επεξεργαστή. Ο επεξεργαστής διαθέτει μόνο 2 γραμμές εξωτερικών διακοπών, την IRQ που αποτελεί τη γραμμή διακοπών κανονικής προτεραιότητας και την την FIQ που αποτελεί τη γραμμή διακοπών υψηλής προτεραιότη-

τας. Αν δηλαδή ενεργοποιηθεί η γραμμή FIQ κατά της διάρκεια της εκτέλεσης της ρουτίνας εξυπηρέτησης ενός IRQ, η ροή του προγράμματος θα μεταφερθεί στην ρουτίνα εκτέλεσης των FIQ.



Σχήμα Γ.1: Διάγραμμα μονάδας ελέγχου διακοπών

Η μονάδα ελέγχου διακοπών είναι υπεύθυνη για τη διαχείριση των διακοπών (interrupts) που αποστέλλονται από τα περιφερειακά προς τον επεξεργαστή. Η λειτουργία της είναι η πολύπλεξη των γραμμών διακοπών. Μέσω αυτής της μονάδας γίνεται ο καθορισμός της προσβασιμότητας των interrupts (ενεργοποίηση/απενεργοποίηση), όπως και η ανάγνωση της τρέχουσας κατάστασής τους (ενεργά, ανενεργά ή προς εξυπηρέτηση). Όπως παρατηρείτε στο διάγραμμα Γ.1, οι πηγές διακοπών προς τη μονάδα ελέγχου διακοπών είναι οι γραμμές επικοινωνίας εισόδου / εξόδου και τα εσωτερικά περιφερειακά. Η μονάδα ελέγχου διακοπών εκτελεί απλούς ελέγχους για να καθορίσει το είδος της τρέχουσας διακοπής (FIQ ή IRQ), όπως και το αν θα προωθηθεί στον επεξεργαστή. Ο επεξεργαστής ειδοποιείται για την ύπαρξη της διακοπής μέσω των γραμμών FIQ και IRQ και επικοινωνεί με τη μονάδα ελέγχου διακοπών μέσω του διαύλου APB, ώστε να αναγνώσει τις πληροφορίες σχετικά με την τρέχουσα διακοπή για την οποία ειδοποιήθηκε.

Οι 32 πηγές διακοπών του συστήματος είναι:

Πίνακας Κωδικών Περιφερειακών

ID	Κωδικός	Περιφερειακό	ID	Κωδικός	Περιφερειακό
0	AIC	Advanced Interrupt Controller	**13**	SPI1	Serial Peripheral Interface 1
1	SYSIRQ	System Interrupt	**14**	SSC0	Synchronous Serial Controller
2	PIOA	Parallel I/O Controller A	**15**	SSC1	Synchronous Serial Controller
3	PIOB	Parallel I/O Controller B	**16**	SSC2	Synchronous Serial Controller
4	PIOC	Parallel I/O Controller C	**17**	TC0	Timer/Counter 0
5	Reserved		**18**	TC1	Timer/Counter 1
6	US0	USART 0	**19**	TC2	Timer/Counter 2
7	US1	USART 1	**20**	UHP	USB Host Port
8	US2	USART 2	**21**	LCDC	LCD Controller
9	MCI	Multimedia Card Interface	**22-28**	Reserved	
10	UDP	USB Device Port	**29**	AIC	Advanced Interrupt Controller
11	TWI	Two-Wire Interface	**30**	AIC	Advanced Interrupt Controller
12	SPI0	Serial Peripheral Interface 0	**31**	AIC	Advanced Interrupt Controller

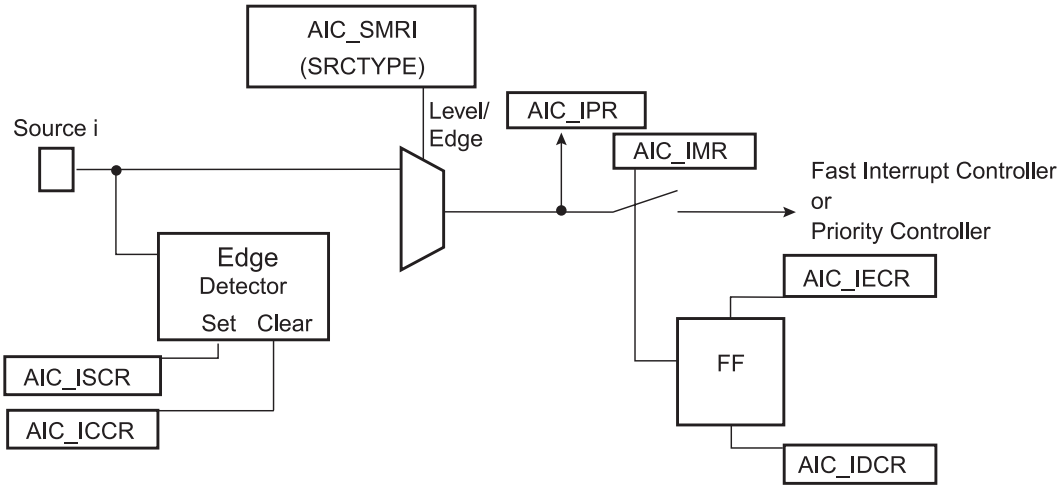
ii. Καταχωρητές ελέγχου

Οι καταχωρητές καθορισμού λειτουργίας αναγράφονται στον ακόλουθο πίνακα. Η στήλη Access καθορίζει τους δυνατούς τρόπους προσπέλασης κάθε καταχωρητή. Η φυσική υλοποίηση του κυκλώματος που ελέγχει τη λειτουργία της γραμμής επικοινωνίας 0 φαίνεται στην εικόνα Γ.2.

Φυσική διεύθυνση: 0xFFFFF000

Μετατόπιση	Όνομα Καταχωρητή	Access
0x00-0x7C	Source Mode Register 0 AIC_SMR0	Read/Write
0x80-0xFC	Source Vector Register 0 AIC_SVR0	Read/Write
0x100	Interrupt Vector Register AIC_IVR	Read-only
0x104	Fast Interrupt Vector Register AIC_FVR	Read-only
0x108	Interrupt Status Register AIC_ISR	Read-only
0x10C	Interrupt Pending Register AIC_IPR	Read-only
0x110	Interrupt Mask Register AIC_IMR	Read-only
0x114	Core Interrupt Status Register AIC_CISR	Read-only
0x118	Reserved	
0x11C	Reserved	
0x120	Interrupt Enable Command Register AIC_IECR	Write-only
0x124	Interrupt Disable Command Register AIC_IDCR	Write-only
0x128	Interrupt Clear Command Register AIC_ICCR	Write-only
0x12C	Interrupt Set Command Register AIC_ISCR	Write-only
0x130	End of Interrupt Command Register AIC_EOICR	Write-only
0x134	Spurious Interrupt Vector Register AIC_SPU	Read/Write

0x138	Debug Control Register AIC_DCR	Read/Write
0x13C	Reserved	
0x140	Fast Forcing Enable Register AIC_FFER	Write-only
0x144	Fast Forcing Disable Register AIC_FFDR	Write-only
0x148	Fast Forcing Status Register AIC_FFSR	Read-only



Σχήμα Γ.2: Διάγραμμα ελέγχου διακοπής

Ο καταχωρητής AIC_ISR περιέχει στα 5 λιγότερο σημαντικά bits του τον κωδικό του περιφερειακού που προκάλεσε το τρέχον interrupt (*Πίνακας Κωδικών Περιφερειακών*). Ο καταχωρητής AIC_IPR περιέχει τα interrupts που δεν έχουν εξυπηρετηθεί ακόμα (το λιγότερο σημαντικό bit αντιστοιχεί σε εξωτερική διακοπή, το bit 1 σε διακοπή συστήματος και τα υπόλοιπα αντιστοιχούν στα περιφερειακά).

Οι διακοπές που προωθούνται στον επεξεργαστή καθορίζονται από τον καταχωρητή AIC_IMR, όπου η τιμή 1 σε κάποιο bit σημαίνει πως οι διακοπές από το αντίστοιχο περιφερειακό θα προωθηθούν, ενώ η τιμή 0 σημαίνει πως οι διακοπές δεν θα προωθηθούν. Η ενεργοποίηση ενός bit του AIC_IMR γίνεται με την εγγραφή της τιμής 1 στο αντίστοιχο bit του AIC_IECR, ενώ η απενεργοποίηση ενός bit του AIC_IMR γίνεται με την εγγραφή της τιμής 1 στο αντίστοιχο bit του AIC_IDCR. Για παράδειγμα, η εντολή

```
aic->IECR = (1<<2);
```

ενεργοποιεί την προώθηση των διακοπών από τη μονάδα PIOA.

Μια διακοπή μπορεί να χαρακτηριστεί είτε ως IRQ, είτε ως FIQ. Οι διακοπές που θα χρησιμοποιηθούν για την υλοποίηση των εργαστηριακών ασκήσεων πρέπει να είναι αποκλειστικά τύπου FIQ. Ο χαρακτηρισμός γίνεται μέσω του καταχωρητή AIC_FFSR, όπου η τιμή 1 σε κάποιο bit σημαίνει πως το αντίστοιχο περιφερειακό θα δημιουργεί διακοπές τύπου FIQ, ενώ η τιμή 0 σημαίνει πως το αντίστοιχο περιφερειακό θα δημιουργεί διακοπές τύπου IRQ. Η ενεργοποίηση ενός bit του AIC_FFSR γίνεται με εγγραφή της τιμής 1 στο αντίστοιχο bit του AIC_FFER, ενώ η απενεργοποίηση ενός bit του AIC_FFSR γίνεται με εγγραφή της τιμής 1 στο αντίστοιχο bit του AIC_FFDR.

Όταν ενεργοποιηθεί μια διακοπή και εκτελεστεί η ρουτίνα εξυπηρέτησης της, **πρέπει πρώτα** να καθαριστεί η ένδειξη από την πηγή της (δηλαδή από τον υπεύθυνο καταχωρητή του περιφερειακού που δημιούργησε τη διακοπή) και στη συνέχεια από την μονάδα ελέγχου διακοπών. Ο καθαρισμός στην μονάδα ελέγχου διακοπών γίνεται με την εγγραφή της τιμής 1 στο bit του AIC_ICCR που αντιστοιχεί στο εκάστοτε περιφερειακό.

iii. Παράδειγμα

Η δυνατότητα χρήσης της ρουτίνας εξυπηρέτησης διακοπών ενεργοποιείται με την εγγραφή της διεύθυνσης της ρουτίνας στην κατάλληλη θέση του διανύσματος διακοπών που περιγράφηκε στον πίνακα Γ'.2. Το είδος των διακοπών που θα χρησιμοποιηθεί για τις ασκήσεις που θα υλοποιηθούν στο σύστημα AT91 είναι οι FIQ διακοπές. Το λειτουργικό σύστημα Linux έχει ήδη τοποθετήσει στη θέση μνήμης 0xFFFF001C τη δική του ρουτίνα εξυπηρέτησης διακοπών, η οποία εκτελεί τις ρουτίνες που έχουν εισαχθεί στο σύστημα με την εγγραφή στο αρχείο συστήματος /proc/FIQ. Αυτό σημαίνει πως για να εισάγετε τη δική σας ρουτίνα εξυπηρέτησης στο σύστημα, αρκεί η εγγραφή της διεύθυνσής της στο αρχείο /proc/FIQ. Η διαδικασία αυτή έχει υλοποιηθεί στο αρχείο HEADER.H και απαιτεί το όνομα της ρουτίνας εξυπηρέτησης διακοπών να είναι FIQ_handler. Στο ακόλουθο πρόγραμμα, θα γίνει η ρύθμιση της γραμμής εισόδου PIOA_9 σε λειτουργία εισόδου και θα ενεργοποιηθεί η δυνατότητα παραγωγής διακοπών με την αλλαγή της στάθμης του δυναμικού εισόδου της. Η ρουτίνα εξυπηρέτησης διακοπών θα αυξάνει ένα μετρητή.

Ρύθμιση λειτουργίας γραμμών επικοινωνίας.

```
unsigned int tmp;                  //Μεταβλητή γενικού σκοπού.
pioa->PIO_PER = 0x200;             //Ρύθμιση της γραμμής 9 σε λειτουργία γενικού σκοπού.
pioa->PIO_PODR = 0x200;            //Ρύθμιση της γραμμής 9 σε λειτουργία εισόδου.
pioa->PIO_PUER = 0x200;            //Ενεργοποίηση της pull-up αντίστασης.
tmp = pioa->PIO_ISR;               //Μηδενισμός καταχωρητή διακοπών με την ανάγνωση.
pioa->PIO_IER = 0x200;             //Ενεργοποίηση παραγωγής διακοπών.
```

```
aic->ICCR = (1<<2);          //Μηδενισμός υπαρχόντων διακοπών από PIOA.
aic->FFER = (1<<2);          //Οι διακοπές από την PIOA είναι FIQ.
aic->IECR = (1<<2);          //Ενεργοποίηση προώθησης διακοπών από PIOA.

volatile unsigned int count = 0;
void FIQ_handler(void){
unsigned int fiq = 0;        //Μεταβλητή που θα περιέχει την πηγή της διακοπής.
fiq = aic->IPR;              //Εντοπισμός περιφερειακού που προκάλεσε τη διακοπή.

if( fiq & 1<<2 ){            //Αν το προκάλεσε η PIOA.
fiq = pioa->ISR;             //Μηδενισμός καταχωρητή διακοπών με την ανάγνωση.
aic->ICCR = (1<<2);          //Μηδενισμός διακοπής από την AIC.
count++;}}                   //Αύξηση μετρητή.
```

Παράρτημα Δ΄

Μονάδα Μετρητή

iv. Γενικά

Η μονάδα μετρητή περιλαμβάνει 3 ανεξάρτητους μετρητές καθένας εύρους 16 bit. Κάθε μετρητής ξεκινά από το 0 και μετρά μέχρι το 0xFFFF. Όταν φτάσει σε αυτή την τιμή δημιουργεί διακοπή λόγω υπερχείλισης και ξεκινά πάλι την μέτρηση από το 0. Κάθε μετρητής μπορεί να δεχτεί σήμα ρολογιού από 8 διαφορετικές πηγές, εκ των οποίων οι πρώτες 5 αφορούν εσωτερικά ρολόγια, ενώ οι υπόλοιπες 3 αφορούν εξωτερικά σήματα. Στις εργαστηριακές ασκήσεις θα χρησιμοποιήσετε μόνο τον μετρητή 0.

v. Καταχωρητές ελέγχου

Η εικόνα Δ΄.1 παρουσιάζει τη φυσική υλοποίηση κάθε μετρητή. Η επιθυμητή λειτουργία κάθε μετρητή καθορίζεται από τις τιμές διάφορων καταχωρητών ελέγχου που αναφέρονται στον ακόλουθο πίνακα :

Φυσική διεύθυνση: 0xFFFA0000

Μετατόπιση	**Όνομα Καταχωρητή**	**Access**
0x00	Channel Control Register TC_CCR	Write-only
0x04	Channel Mode Register TC_CMR	Read-Write
0x08	Reserved	
0x0C	Reserved	
0x10	Counter Value TC_CV	Read-only
0x14	Register A TC_RA	Read-Write
0x18	Register B TC_RB	Read-Write
0x1C	Register C TC_RC	Read-Write
0x20	Status Register TC_SR	Read-only
0x24	Interrupt Enable Register TC_IER	Write-only
0x28	Interrupt Disable Register TC_IDR	Write-only

0x2C	Interrupt Mask Register TC_IMR	Read-only

Ο μετρητής ενεργοποιείται με εγγραφή της τιμής 1 στο λιγότερο σημαντικό bit του TC_CCR (bit 0) και απενεργοποιείται με την εγγραφή της τιμής 1 στο 1ο bit του καταχωρητή TC_CCR. Ο μετρητής επανατίθεται στην τιμή 0 με την εγγραφή της τιμής 1 στο 2ο bit του καταχωρητή TC_CCR. Για παράδειγμα, η εντολή:

tc->Channel_0.CCR = 0x5;

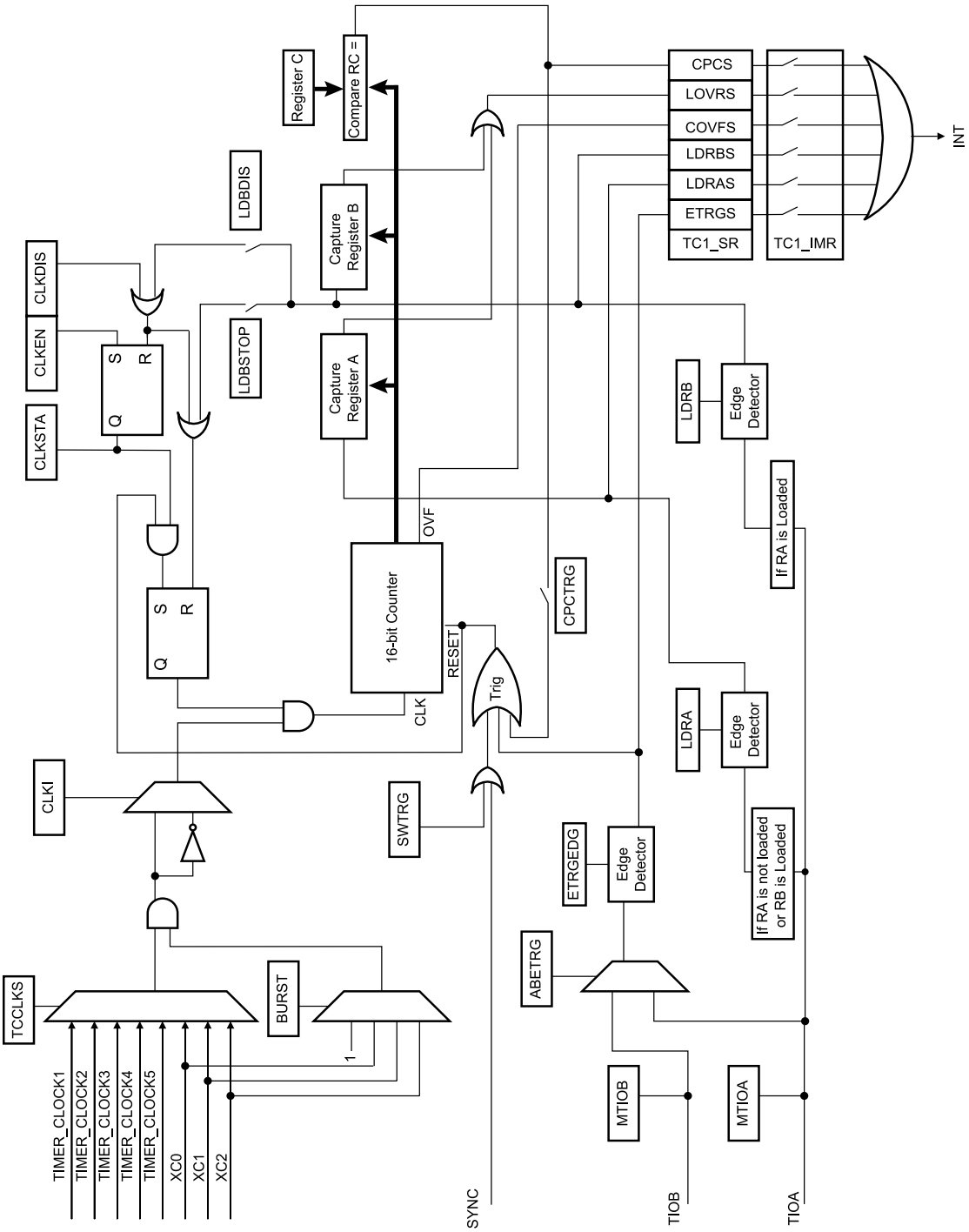
θα ενεργοποιήσει τον μετρητή και θα ξεκινήσει τη μέτρηση από την τιμή 0.

Οι παράμετροι λειτουργίας του μετρητή καθορίζονται από τον TC_CMR, όπου τα 3 λιγότερο σημαντικά bits καθορίζουν το ρολόι που θα χρησιμοποιηθεί, το 7ο bit ορίζει το αν ο μετρητής θα σταματήσει τη μέτρηση μόλις η τιμή του γίνει ίση με τον RC (αυτό συμβαίνει όταν το bit 7 έχει την τιμή 1) και το bit 15 που ορίζει ότι η λειτουργία είναι τύπου μέτρησης και πρέπει πάντα να έχει την τιμή 1. Τα υπόλοιπα bits πρέπει να διατηρούνται στην τιμή 0.

Bit 2	Bit 1	Bit 0	Clock	Description
0	0	0	TIMER_CLOCK1	48 MHz
0	0	1	TIMER_CLOCK2	12 MHz
0	1	0	TIMER_CLOCK3	3 MHz
0	1	1	TIMER_CLOCK4	750 KHz
1	0	0	TIMER_CLOCK5	8,192 KHz
1	0	1	XC0	External
1	1	0	XC1	External
1	1	1	XC2	External

Στον καταχωρητή TC_CV υπάρχει η τρέχουσα τιμή του μετρητή. Στους καταχωρητές TC_RA, TC_RB και TC_RC μπορεί να γίνει εγγραφή τιμών, οι οποίες συγκρίνονται με την τρέχουσα τιμή του μετρητή. Όταν η τιμή του μετρητή ξεπεράσει την τιμή κάποιου από τους 3 καταχωρητές, ενεργοποιούνται γραμμές εξόδου ή δημιουργούνται διακοπές, ανάλογα με τις παραμέτρους που έχουν εγγραφεί στον TC_CMR.

Ο καταχωρητής TC_IMR καθορίζει το ποιες διακοπές θα προωθηθούν στην μονάδα ελέγχου. Αν η τιμή ενός bit του TC_IMR είναι ίση με το 1, τότε η αντίστοιχη διακοπή θα προωθηθεί, ενώ αν είναι 0 δεν θα προωθηθεί. Τα bits του TC_IMR ενεργοποιούνται με την εγγραφή της τιμής 1 στα αντίστοιχα bits του TC_IER και απενεργοποιούνται με



Σχήμα Δ'.1: Διάγραμμα μονάδας μετρητή

την εγγραφή της τιμής 1 στα αντίστοιχα bits του TC_IDR. Ο καταχωρητής TC_SR διατηρεί τα γεγονότα που έχουν συμβεί μετά από την τελευταία ανάγνωσή του και μηδενίζεται με προσπέλαση για ανάγνωση. Αν είναι ενεργοποιημένο στον καταχωρητή TC_IMR το αντίστοιχο bit κάποιου γεγονότος, η διακοπή που θα δημιουργηθεί θα προωθείται στη μονάδα ελέγχου διακοπών μέχρι να γίνει ανάγνωση του TC_SR. Τα γεγονότα που υποστηρίζονται είναι:

Bit	**Event**	**Bit**	**Event**
0	COVFS Counter Overflow	4	CPCS RC Compare
1	LOVRS Load Overrun	5	LDRAS RA Loading
2	CPAS RA Compare	6	LDRBS RB Loading
3	CPBS RB Compare	7	ETRGS External Trigger

vi. Παράδειγμα

Υποθέστε πως η εφαρμογή που καλείστε να υλοποιήσετε απαιτεί την ενεργοποίηση διακοπής με συχνότητα 5 Hz. Αυτό μπορεί να επιτευχθεί με την χρήση της μονάδας μετρητή και την ρύθμιση μέτρησης χρονικού διαστήματος ίσου με 200ms. Χρησιμοποιούμε μόνο το μετρητή 0, οι καταχωρητές του οποίου μπορούν να προσπελαστούν μέσω της δομής tc->Channel_0.

Ρύθμιση λειτουργίας μετρητή.

```
tc->Channel_0.RC = 1638;        //Περίπου 200 ms
tc->Channel_0.CMR = 0x2084;     //Slow clock, count, stop on RC compare
tc->Channel_0.IDR = 0xFF;       //Απενεργοποίηση όλων των διακοπών
tc->Channel_0.IER = 0x10;       //Ενεργοποίηση διακοπής μόλις ο RC γίνει ίσος με τον μετρητή
tc->Channel_0.CCR = 5;          //Έναρξη μέτρησης
```

Μέσα στη ρουτίνα εξυπηρέτησης διακοπών, όταν ανιχνευτεί διακοπή από το μετρητή, πρέπει να εκτελεστούν οι εξής εντολές:

Ρουτίνα εξυπηρέτησης διακοπής μετρητή.

```
dummy = tc->Channel_0.SR;   //Μηδενισμός της πηγής διακοπής
tc->Channel_0.CCR = 5;      //Έναρξη μέτρησης
```