Sve zasluge idu kolegama: Krunoslav Šokić, Karolina, Jablan i Limun2904
(http://www.fer2.net/showpost.php?p=1978953&postcount=8)
(http://www.fer2.net/showthread.php?t=64932)
(http://www.fer2.net/showpost.php?p=2396745&postcount=21)

Zadatak 1:

```
//  Potrebno  je  napisati  C  program koji neprekidno cita stanje prikljucka 28 na sklopu
//PIOB. Kad je na tom prikljucku logicka jedinica, potrebno je postaviti vrijednost 0xAA na
//prikljucke od 0 do 7 na sklopu PIOB. Kad je na tom prikljucku logicka nula, potrebno je
//spomenute prikljucke skupa PIOB držati u stanju visoke impedancije. Preostalim prikljuccima
//sklopa PIOB nije dopušteno pristupati.

#include <AT91SAM7X512.h>
#include <stdio.h>

int main (void) {
    *AT91C_PMC_PCER = (1<<AT91C_ID_PIOB); //takt za PIOB
    *AT91C_PIOB_PER = 1<<28; // Paljenje adekvatnih prikljucaka
    *AT91C_PIOB_ODR = 1<<28;  // zelimo onemoguciti izlazno pojacalo za pin PIOB 28 zato se
upisuje 1 u ODR registar
    // zelimo koristiti pinove PIOB 0-7, zato moramo pojacalo za te pinove omoguciti
          *AT91C_PIOB_PER = 0xFF<<0;
          *AT91C_PIOB_OER = 0xFF<<0;

    while (1) {
        if (*AT91C_PIOB_PDSR & (1<<28)){
                *AT91C_PIOB_OER  = 0xFF;  //ne znamo kakvo je prije bilo stanje, ako je bilo u
stanju visoke impedancije, moramo upaliti izlazno pojacalo
                *AT91C_PIOB_CODR = 0xFF; // zelimo iskljuciti prvo te prikljucke, da bi bili
sigurni da ce stanje u iducem koraku biti 0xAA, mislim da u ovom zadatku cak to i nije tako
bitno....ali u zivotu ce biti bitno :)
                *AT91C_PIOB_SODR = 0xAA; // postavljamo stanje 0xAA na pinove 0-7
        }
        else
            *AT91C_PIOB_ODR=0xFF; // te prikljucke stavljamo u stanje visoke impedancije 0-7
    }
}
```

Zadatak 2:

```
//  Potrebno  je  generirati  digitalni  pilasti  signal  koji  poprima  vrijednosti  od  0x0  do
//0xFFFF. Uzorke signala treba generirati u prekidnoj funkciji koja poslužuje prekid PIT
//sklopa. U svakom pozivu funkcija treba povecati vrijednost signala za 1. U istoj funkciji
//potrebno  je izvršiti i upisivanje signala u izlazno sklopovlje. Kao izlazne prikljucke koristiti
//prikljucke 0 do 15 sklopa PIOB. Period uzorkovanja pilastog signala iznosi 2ms. Pretpostaviti
//da procesor radi na taktu frekvencije 32MHz.


#include <at91sam7x128.h>
#include <stdio.h>

void pit_handler (void) __irq;
void spurious_handler (void) __irq;

unsigned int brojac=0;

int main (void){
    *(AT91C_AIC_SMR+AT91C_ID_SYS)=(0<<0)|(0<<5); //ID za SYS prekid, bit 0 za prioritet, 0 za rastuci
brid;
    *(AT91C_AIC_SVR+AT91C_ID_SYS)= (unsigned int) pit_handler; //prekidni vektor
    *AT91C_AIC_SPU = (unsigned int) spurious_handler; //prekidni vektor za neregularni prekid
    *AT91C_AIC_IECR=1<<AT91C_ID_SYS;

    *AT91C_PITC_PIMR=(3<<24)|(3999<<0); //ukljucit PITIEN i PITN, postaviti PIV na 3999->brojanje do
2ms

    *AT91C_PMC_PCER=(1<<AT91C_ID_PIOB); //takt na PIOB;
    *AT91C_PIOB_PER=0xFFFF; //izlazi 0 do 15 ukljucivanje takta;
    *AT91C_PIOB_OER=0xFFFF; //izlazno pojacalo;
```

```
    *AT91C_PIOB_SODR=(0xFFFF & brojac);


    while(1);
    }

void pit_handler (void) __irq {


    //*AT91C_PIOB_SODR=(0xFFFF & brojac); //postavljanje izlaza;

    brojac++;
    if(brojac==(0xFFFF+1))    brojac=0;
    *AT91C_PIOB_CODR=0xFFFF;
    *AT91C_PIOB_SODR=(0xFFFF & brojac); //kada pila predje max vrijednost brise se izlazni registar i
upisuje se na njega 0;


    *AT91C_AIC_EOICR=*AT91C_PITC_PIVR; //citanje i pisanje u registre gdje to treba;
}



void spurious_handler (void) __irq {
        *AT91C_AIC_EOICR=*AT91C_PITC_PIVR; //citanje i pisanje u registre gdje to treba;
}
```

Zadatak 3:

```
//  Napisati  funkciju  za  konfiguriranje  PLL  sklopa  tako  da  on  na  svom  izlazu  daje
//frekvenciju koja je 10 puta veca od ulazne frekvencije. U funkciji je potrebno cekati dok
//završi prijelazna pojava. Odabrati maksimalno moguce vrijeme utitravanja PLL sklopa.
//Pretpostaviti da PLL radi u frekvencijskom podrucju od 80 do 160MHz tj. vrijednost polja
//OUT iznosi 00b.

#include <AT91SAM7X128.h>
#include <stdio.h>

void PLLinit (void) {
    *AT91C_PMC_PLLR=(9<<16)|(0b0<<14)|(0b0<<15)|(0x3F<<8)|(1<<0); //MUL=9 DIV=1 PLLCOUNT=63 (0x3F)
OUT=00b
// ispravak kolege Limun2904: moze i 0b00<<14 ali onda nam ne treba 0b0<<15 ista je stvar

    while ((*AT91C_PMC_SR)&4); //cekanje dok se LOCK bit ne makne;

}

int main (void){
    PLLinit();
}
```

Zadatak 4:

main.c:

```
//  Napisati program koji u petlji cita znakove sa serijskog sucelja. Kad se pojavi znak 'd',
//na serijsko sucelje potrebno je ispisati poruku "Doslo je slovo!". Program se mora izvršavati
//na  sklopovlju  mikrokontrolera  AT91SAM7X.  Kao  izlaznu  jedinicu  koristiti  serijsko sucelje
//USART0. Za citanje i ispis potrebno je koristiti funkcije scanf() i printf(). Potrebno je
//koristiti  redefiniranje  ulazno-izlaznih  funkcija.  Pretpostaviti  da  je  funkcija  za
//inicijalizaciju zadana i zove se init USART0().

#include <stdio.h>
#include <at91sam7x128.h>

extern void init_USART0 (void);
int main (void) {

    char slovo;
    init_USART0();


    //sendchar_USART0('d');
```

```
    while (slovo !='d'){
        scanf("%c", &slovo);

        if (slovo=='d') printf ("Doslo je slovo!");
        else printf ("Nije doslo slovo");
    }

    while(1);

}

char receivechar_USART0 (void){
    while (!(*AT91C_US0_CSR & AT91C_US_RXRDY)); //cekanje dok sklop ne bude spreman za primanje ??

    return *AT91C_US0_RHR;
}


void sendchar_USART0 (ch) {
    while (!(*AT91C_US0_CSR & AT91C_US_TXRDY)); //cekanje spremnost za slanje;

    *AT91C_US0_THR=ch;
}

// Za slucaj da treba napisat i init_USART0
//
// To ide u poseban fajl: init_USART0.c
//
// #include <at91sam7x128.h>
// #define MCK 48000000   // fMCK=48MHz
// #define BR 9600        // Zeljena brzina prijenosa
// #define CD (MCK/16/BR) // konst. za generator takta za prijenos
// void init_USART0 (void) {
//    *AT91C_PMC_PCER=(1<<AT91C_ID_US0) | // Takt za USART0
//                    (1<<AT91C_ID_PIOA); // Takt za PIOA
//    *AT91C_PIOA_PDR=AT91C_PA1_TXD0; // Dodjela prikljucaka
//
//    *AT91C_US0_CR=AT91C_US_TXDIS | // Onemoguci odasiljac
//                  AT91C_US_RSTTX;  // Resetiraj odasiljac
//    *AT91C_US0_MR=AT91C_US_USMODE_NORMAL | // normalan nacin rada
//                  AT91C_US_CLKS_CLOCK |   // takt za USART je MCK
//                  AT91C_US_CHRL_8_BITS |  // znak ima 8 bitova
//                  AT91C_US_NBSTOP_1_BIT | // 1 stop bit
//                  AT91C_US_PAR_NONE;      // nema pariteta
//                                        // MSBF=0 => prvo ide LSB
//    *AT91C_US0_BRGR=CD; // inic. generatora takta za prijenos
//    *AT91C_US0_CR=AT91C_US_TXEN;          // enable odasiljac
// }
```

retarget.c:

```
#include <stdio.h>
#include <at91sam7x128.h>

#pragma import (__use_no_semihosting_swi)

extern void sendchar_USART0 (int ch);
extern char receivechar_USART0 (void);
extern void init_USART0 (void);
int zastavica;
int slovo;
struct __FILE {
    int handle;
};

FILE __stdout;
FILE __stdin;

int fputc (int ch, FILE *f){
    sendchar_USART0 (ch);
    return (ch);
}

int ferror (FILE *f) {
    return 0;
}
```

```
int fgetc(FILE *f){
    if(zastavica==1) {zastavica=0; return slovo;} //ovo je nesto cudno;
    return (receivechar_USART0());
}

int _backspace(){
    zastavica=1;
    return 0;
}

void _sys_exit (int return_code){
    while(1);
}
```

Zadatak 5:

```
//Potrebno je napisati program za konfiguriranje sklopovlja PIOB. Program mora ocitati stanje
prikljucka 18 sklopa PIOB. Ako je na tom prikljucku jedinica ,
//potrebno je konfigurirati prikljucke 19-22 sklopa PIOB kao izlazne prikljucke periferije A. U
slucaju da je na tom prikljucku nula,
//prikljucke 19-21 sklopa PIOB konfigurirati kao izlazne sa spojenim priteznim otpornikom, a
prikljucak 22 sklopa PIOB konfigurirati
//kao ulazni sa ukljucenim „glitch“ filtrom
//te omoguciti dodavanje zahtjeva za prekid toga prikljucka na razini PIO sklopa.



/* Algoritam kak tocno rjesiti ovaj zadatak....
*  1. Inicijalizirati PIOB 18 kao ulazni.
*  2. Provjeriti stanje na PIOB 18.
*  3. Ako je PIOB 18 jednak 1 onda na prikljucke PIOB 19-22  preusmjeriti na PIOA 19-22.
*     (nije greska ovo s A i B; pogledaj salabahter, tamo su 2 linije 'izlaz periferije A' i isto za
B). i ispisati 0xAA.
*  4. Ako je PIOB 18 jednak 0 onda na prikljucke (valjda 19-22?!) od PIOB konfigurirati kao izlazne sa
spojenim pull up otpornikom i PIOB 22 kao ulazni s ukljucenim 'glitch' filtrom i omoguciti zahtjev za
prekid (samo zahtjeva PIO omogucavanje, ne AIC omogucavanje prekida).
*/

#include <AT91SAM7X512.h>
#include <stdio.h>

int main (void) {
         // 1. Inicijalizirati PIOB 28 kao ulazni
    *AT91C_PMC_PCER = (1<<AT91C_ID_PIOB); //takt za PIOB
           *AT91C_PIOB_ODR = 1 << 18; // zelimo onemoguciti izlazno pojacalo PIOB 28, no za to jos
moramo i
           *AT91C_PIOB_PER = 1 << 18; // omoguciti onaj gornji multiplekser da selektira prvi ulaz
pomocu PER registra
           *AT91C_PIOB_IFER = 0;    // donji multiplekser samo propusta stanje s pina u mikrokontroler

         // U zadatku pise da se smao jednom provjeri stanje PIOB 28, zato nema tu nikakvog while-a...
         // 2. Provjeriti stanje na PIOB 28.
         if(*AT91C_PIOB_PDSR & 1<<18){    // provjerava se stanje pina PIOB 28, pogledaj salabahter,
vidis da u tom registru je zapisano stanje
                  // 3. dio algoritma
                  *AT91C_PIOB_PDR = 0x0F << 19;  // multiplekser selektira ulaz 0, da smo htjeli
selektirati ulaz 1, upisivali bi tu konstantu u PER.
                  *AT91C_PIOB_ASR = 0x0F << 19;  // multiplekser mora odabrati 0ti ulaz (jer je to
perif. A) a to se radi s registrom ASR tako da upisemo jedinice na mjesta koje pinove preusmjeravamo
         }
         else{
                  // 4. dio algoritma
                  *AT91C_PIOB_OER = 0x7 << 19;
                  *AT91C_PIOB_PER = 0x7 << 19; //ovo je klasika kad se neki prikljucci definiraju kao
izlazni definiranje PIOB 22 kao ulaznog s ukljucenim glitch filtrom
                *AT91C_PIOB_PPUER=(0x7<<19); //Pritezni otpornik
                  *AT91C_PIOB_ODR = 1 << 22;   // izlazno pojacalo cemo onemoguciti, obzirom da ovaj
pin ne koristimo kao izlazni, ne treba nam to pojacalo
                  *AT91C_PIOB_PER = 1 << 22;   // s prethodnom linijom stanje za iskljucivanje
pojacala smo doveli na ulaz multipleksera,
                                               //s ovom linijom se omogucava prolaz prvog kanala
multipleksera
                  //glitch i omogucavanje zahtjeva za prekid
                  *AT91C_PIOB_IFER = 1 << 22;  // glitch, selektira se prvi kanal od multipleksera
                  *AT91C_PIOB_IER  = 1 << 22;  // omogucavanje interrupta s tog pina. Da je u zadatku
bilo receno da zaista omogucimo prekid onda bi morali pisat sve ono s AIC periferijom
}
```

```
        while(1);
        }
```

Zadatak 6:

```
//Potrebno je napisati funkciju koja mijenja izvor i frekvenciju takta mikrokontrolera.
//Ulazni parametar funkcije je ASCII znak. Ako je ulazni parametar jednak „p"
//sklopovlje je potrebno konfigurirati da izvor takta bude PLL sklop, a frekvencija na kojoj radi uC
iznosi 30 MHz.
//Pretpostaviti da kristalni oscilator radi na freq. 20 MHz te da on vec radi u stacionarnom stanju.
Odabrati max.
//vrijeme utitravanja PLL sklopa. Osigurati cekanja tijekom prijelaznih pojava. Ako ulazni parametar
nije jednak „p" , zadržati postojece stanje izvora takta.

#include <stdio.h>
#include <AT91SAM7X128.h>

int main (void){

}

void clk_promjena (char slovo){
    if (slovo == 'p') {
    *AT91C_CKGR_PLLR = (2<<16)|(0b00<<14)|(63<<8)|(1<<0); //MUL=2 DOV=1 PLLCOUNT= 0x3F OUT=0b00;
    while((*AT91C_PMC_SR & (1<<2)) == 0); //cekanje do se ne makne LOCK, ->utitravanje PLL-a, kad je
jednak 1 znaci da je jos uvijek 'zakljucan' tj. nije spreman

    *AT91C_PMC_MCKR=(1<<2); //prescaler = 1
    while ((*AT91C_PMC_SR & (1<<3)) == 0); // cekanje da se utitra MCK, provjerava se zastavica
MCKRDY, kad je MCKRDY=1  spremno je

    *AT91C_PMC_MCKR=(3<<0); //PLL = 3, MainClock=1, SlowClock=0
    while ((*AT91C_PMC_SR & (1<<3)) == 0); // utitravanje MCK-a
    }
}
```

Zadatak 7:

```
// Zadatak 1. (5 bodova)
// Na skup prikljucaka PIOA spojeno je vanjsko sklopovlje. Na prikljucke 0-7 dovodi se
// podatak koji je zapisan u dvojnom komplementu.
//Na prikljucak 28 spojena je tipka prema masi. Potrebno je napisati program koji neprekidno cita
stanje tipke. Dok je tipka pritisnuta, potrebno je na prikljucke od 16 do 23 neprekidno upisivati
apsolutnu vrijednost podatka koji je procitan s prikljucaka 0-7.  kad je tipka spojena prema masi
znaci da kad je spojena citamo 0, a kad je odspojena cita se 1.

#include <stdio.h>
#include <at91sam7x128.h>

int main (void){
  *AT91C_PMC_PCER=(1<<AT91C_ID_PIOA);
  *AT91C_PIOA_PER=(1<<28)|(0xFF<<16)|(0xFF<<0); //Upravljanje prikljuccima 16 do 23, 0 do 7 i tipkom;
  *AT91C_PIOA_ODR=(0xFF)|(AT91C_PIO_PA28); //0-7 su ulazni prikljucci; 28 je tipka prema masi;
  *AT91C PIOA PPUDR=AT91C PIO PA28; //pull-down tipka
  *AT91C_PIOA_OER=(0xFF<<16); // prikljucci 16 do 23 su izlazi;

         while (1){
           int tmp = (*AT91C_PIOA_PDSR & (1<<28)) >> 28; // izolira se 28.bit i onda se pomice na
nulto mjesto
                  tmp = ~tmp; // ako je tmp 1 znaci da je stisnuta, ako je tmp 0 onda nije stisnuta
                  if(tmp == 1){
                           int value = (*AT91C_PIOA_PDSR & 0xFF); // value je procitana vrijednost s
prikljucaka
                           if((value & 0x80) != 0){ // ako je MSB bit procitane vrijednosti jednak 1
onda se mora invertirati broj
                                    *AT91C_PIOA_CODR = 0xFF;
                                    *AT91C_PIOA_SODR = 0-value; //ispis  pozitivne vrijednosti
                           }
                           else{
                                    *AT91C_PIOA_CODR = 0xFF;
                                    *AT91C_PIOA_SODR = value;
                           }
                  }
```

```
  }
}
```

Zadatak 8:

```
// Zadatak 2. (5 bodova)
// Potrebno je inicijalizirati sklop PIOB  tako da pojava aktivnog signala na prikljuccima 3, 7 i
// 11 uzrokuje zahtjev za prekid. Potrebno je omoguciti prekid i dodijeliti mu najvišu razinu
// prioriteta. Napisati potrebne prekidne funkcije. Prekidna funkcija koja poslužuje zahtjev za
// prekid PIOB mora brojati prekide koji su došli sa svakog pojedinog prikljucka.

#include <at91sam7x128.h>

unsigned int brojac3=0;
unsigned int brojac7=0;
unsigned int brojac11=0;

void piob_handler (void) __irq;
void spu_handler (void) __irq;

int main (void) {

    *AT91C_PMC_PCER=(1<<AT91C_ID_PIOB); //takt na PIOB;
    *AT91C_PIOB_PER=(AT91C_PIO_PB3)|(AT91C_PIO_PB7)|(AT91C_PIO_PB11);
    *AT91C_PIOB_ODR=(AT91C_PIO_PB3)|(AT91C_PIO_PB7)|(AT91C_PIO_PB11); // Ulazni prikljucci
    *AT91C_PIOB_IER=(AT91C_PIO_PB3)|(AT91C_PIO_PB7)|(AT91C_PIO_PB11); //Omogucavanje prekida na
prikljuccima 3, 7 i 11;
    *(AT91C_AIC_SMR+AT91C_ID_PIOB)=(0<<5)|(7<<0); //Dodjeljivanje prioriteta i okidanja na rastuci
brid;
    *(AT91C_AIC_SVR+AT91C_ID_PIOB)=(unsigned int) piob_handler; //prekidni vektori;
    *AT91C_AIC_SPU= (unsigned int) spu_handler; //prekidni vektori;

    while(1);
}

/* Ovo je bilo u originalu
void piob_handler (void) __irq {
        if (*AT91C_PIOB_ISR & AT91C_PIO_PB3) brojac3++; //brojanje prekida sa pojedinog prikljucka
        if (*AT91C_PIOB_ISR & AT91C_PIO_PB7) brojac7++;
        if (*AT91C_PIOB_ISR & AT91C_PIO_PB11) brojac11++;
*/
// no, kolega Jablan kaže: zadatak 8, pripazite na čitanje izvora prekida. Registar
*AT91C_PIOB_ISR sadrži prekidne zastavice koje su se dogodile od trenutka zadnjeg čitanja. Svaki
put kad čitate taj registar on se resetira. U gornjem rješenju prvo će pročitati dal ima prekida
za B3 i resetirat se pa ne radi kako treba. Trebalo bi prvo spremit citavi registar u varijablu
pa preko varijable provjeravat bitove.
void piob_handler (void) __irq {
         maska = *AT91C_PIOB_ISR;
         if (maska & 1<<3) br3++;
         if (maska & 1<<7) br7++;
         if (maska & 1<<11) br11++;

         // *AT91C AIC EOICR=*AT91C PITC PIVR; // ovo je stajalo u originalu
         // ispravak kolege Limun2904
         *AT91C_AIC_EOICR=0; // ono s PIVR nema smisla zato sto uopce ne koristimo taj sklop, a
upisom bilo cega u ovaj registar brisemo zastavicu

}

void spu_handler (void) __irq {
        *AT91C_AIC_EOICR=0;
}
```

Zadatak 9:

```
// Zadatak 3. (4 bodova)
// Potrebno je napisati funkciju koja mijenja nacin rada sklopovlja za reset. Ulazni parametar
// funkcije je ASCII znak. Ako je ulazni parametar jednak 'R', sklopovlje je potrebno
// konfigurirati tako pritisak na tipku za reset uzrokuje stvarni reset mikrokontrolera. Ako je
// ulazni parametar jednak 'I', sklopovlje treba konfigurirati tako da pritisak na tipku generira
// zahtjev za prekid. Prekid treba omoguciti iskljucivo na razini reset kontrolera.
//

#include <at91sam7x128.h>
void funkcija (char ulaz){
    if (ulaz =='R'){
```

```
        *AT91C_RSTC_RMR=(1<<0); //omogucavanje reseta pritiskom na tipku
    }
    if (ulaz=='I'){
        *AT91C_RSTC_RMR = (0XA5 << 24) | (1 << 4) | (0 << 0); //omogucavanje reseta i prekida
         // ispravak kolege Jablan - u originalu bilo: *AT91C_RSTC_RMR=(1<<0)|(1<<4);

    }

}
```

Zadatak 10:

```
// Zadatak 2 (9 bodova)
// Potrebno je napisati programsku podršku koja nakon svakih 10ms procita podatak s prikljucka
// 7 sklopa PIOB, invertira  ga i postavi na prikljucak 3 sklopa PIOB. Ova operacija mora biti
// upravljana iz prekidne funkcije  koja poslužuje prekid sklopa  Periodic Interval Timer.
// Pretpostaviti frekvenciju takta od 16MHz.
//

#include <at91sam7x128.h>
#include <stdio.h>


void PIT_handler(void) __irq;
void spurious_handler (void) __irq;


int main(void){

    *(AT91C_AIC_SMR+AT91C_ID_SYS)=(0<<0)|(0<<5);
    *(AT91C_AIC_SVR+AT91C_ID_SYS)=(unsigned int)PIT_handler; //Prekidni vektori;
    *AT91C_AIC_SPU = (unsigned int) spurious_handler; //Prekidni vektori;
    *AT91C_AIC_IECR=AT91C_ID_SYS;

    *AT91C_PMC_PCER=(1<<3); // Takt na PIOB;
    *AT91C_PIOB_PER=AT91C_PIO_PB3 | AT91C_PIO_PB7; //Upravljanje prikljuckom 3 i 7;
    *AT91C_PIOB_ODR=AT91C_PIO_PB7; //PB7 je ulazni prikljucak
    *AT91C_PIOB_OER=AT91C_PIO_PB3; //PB3 je izlazni prikljucak;


    *AT91C_PITC_PIMR=(3<<24)|(9999<<0); //Podesavanje PIT sklopovlja;



    while(1);

}

void PIT_handler (void) __irq{
         int tmp = *AT91C_PIOB_PDSR >> 7; //tu pomicem 7. bit na 0.mjesto
         tmp = tmp & 0x01; // izoliram samo taj bit, znaci vrijednost tmp moze biti 0 ili 1
         tmp = ~tmp; //invertiram, tako da znam trebam li ispisati 0 ili 1 na PIOB3
         if(tmp == 1){
                  *AT91C_PIOB_SODR= 1<<3; // kada se upisuje 1 onda s ekoristi registar SODR
         }
         else{
                  *AT91C_PIOB_CODR= 1<<3; // kada se upisuje 0 na pin koristi se CODR
         }

         *AT91C_AIC_EOICR=*AT91C_PITC_PIVR;
}

void spurious_handler(void) __irq {
    *AT91C AIC EOICR=*AT91C PITC PIVR;
    }
```

MI 13/14 by kolegica Karolina
(http://www.fer2.net/showpost.php?p=2290259&postcount=2)

29. 4. 2014.

Projektiranje ugradbenih računalnih sustava
Međuispit

Zadatak 1. (6 bodova)

Potrebno je napisati program koji neprekidno čita stanje priključka 18 na sklopu PIOB. Kad je na tom priključku logička jedinica, potrebno je invertirati stanje priključaka od 4 do 11 na sklopu PIOB. Kad je na tom priključku logička nula, potrebno je spomenute priključke skupa PIOB dodijeliti periferiji A ili B. Ako je stanje priključka 1 sklopa PIOB logička jedinica, priključcima pristupa periferija A, a ako je logička nula onda pristupa periferija B. Preostalim priključcima sklopa PIOB nije dopušteno pristupati.

Zadatak 2. (7 bodova)

Potrebno je generirati sinusni signal čija je širina riječi 16 bitova, a vrijednosti amplituda leže u intervalu [-1,1). Pretpostaviti da su uzorci koji odgovaraju prvoj četvrtini perioda **sinusa** pohranjeni u podatkovnoj memoriji počevši od lokacije 0x00202000, te da ih ima ukupno 1000. Uzorke sinusnog signala potrebno je ispisivati na priključke 0 do 15 sklopa PIOB. Ispis treba izvesti u funkciji koja poslužuje prekid PIT sklopa. Zadana je frekvencija uzorkovanja od 1 kHz. Pretpostaviti da procesor radi na taktu frekvencije 16MHz.

Zadatak 3. (5 bodova)

Potrebno je napisati funkciju koja mijenja način rada sklopovlja za reset. Ulazni parametar funkcije je ASCII znak. Ako je ulazni parametar jednak 'R', sklopovlje je potrebno konfigurirati tako pritisak na tipku za reset uzrokuje stvarni reset mikrokontrolera. Ako je ulazni parametar jednak 'I', sklopovlje treba konfigurirati tako da pritisak na tipku generira zahtjev za prekid. Prekid treba omogućiti isključivo na razini reset kontrolera.

Zadatak 4. (8 bodova)

Napisati program koji u petlji čita 8-bitne vrijednosti s priključaka PA0-PA7. Kad se pojavi vrijednost na priključcima, potrebno ju je ispisati na serijsko sučelje. Program se mora izvršavati na sklopovlju mikrokontrolera AT91SAM7X. Za čitanje i ispis potrebno je koristiti funkcije `scanf()` i `printf()`. Potrebno je koristiti redefiniranje ulazno-izlaznih funkcija uz pretpostavku da su poznate sljedeće definicije:

```
typedef struct __FILE FILE;
extern FILE __stdin, __stdout, __stderr;
int fputc (int c, FILE * f );
int ferror (FILE * f);
int fgetc (FILE * f );
int __backspace(FILE * f);
```

Pretpostaviti da funkcija `__backpace()` uvijek vraća vrijednost 0.

Funkcije za inicijalizaciju priključaka i serijskog sučelja zadane su i zovu se `init_PIOA()` i `init_USART0()`.

Zadatak 5. (3 bodova)

Opisati osnovnu inicijalizaciju memorijskog kontrolera. Kad je potrebno podesiti broj stanja čekanja a kad to nije potrebno? Opisati inicijalizaciju PLL sklopovlja.

Zadatak 6. (3 bodova)

Opisati kako je izvedeno napajanje mikrokotrolera familije AT91SAM7X. Nacrtati izvedbu serijskog regulatora koji daje glavno napajanje. Opisati priključak za brisanje memorije i nacrtati spoj vanjskih komponenata koje je potrebno ugraditi na taj priključak.

Zadatak 7. (3 bodova)

Skicirati pojednostavljenu blokovsku shemu sklopa Real-time Timer i opisati način njegovog rada.

Zadatak 1:

```
#include <stdio.h>
#include <at91sam7x128.h>

int main (void){
  *AT91C_PMC_PCER = 1 << AT91C_ID_PIOB;

         // PIOB 1 i 18 su ulazni
         *AT91C_PIOB_ODR = (1<<18) | (1<<1);
         *AT91C_PIOB_PER = (1<<18) | (1<<1);
         *AT91C_PIOB_IFDR = (1<<18) | (1<<1); // kolega Limun2904 kaže ovo ne treba

         // PIOB 4-11 su izlazni
         *AT91C_PIOB_OER = 0XFF<<4;
         *AT91C_PIOB_PER = 0XFF<<4;
         //ne pise da moramo neko inicijalno stanje postaviti...

         while(1){
                  if(*AT91C_PIOB_PDSR & (1<<18)){
                           int tmp = *AT91C_PIOB_PDSR;
                           tmp = ~tmp; // invertiranje bitova
                           *AT91C_PIOB_CODR = 0XFF << 4; // zelimo trenutno stanje na tom pinu
pobrisati tako da mozemo napraviti toggle
                           *AT91C_PIOB_SODR = tmp & (0xFF<<4);
                  }
                 else{
                           if(*AT91C_PIOB_PDSR & 1<<1){
                                    *AT91C_PIOB_ASR = 0XFF << 4; // multipleksor bira A izlaz
                                    *AT91C_PIOB_PDR = 0XFF << 4; // multipleksor bira nulti ulaz za
izlaz, tako da proslijedi izlaz iz perif. A
                           }
                           else{
                                    *AT91C_PIOB_BSR = 0XFF << 4; // multipleksor bira A izlaz
                                    *AT91C_PIOB_PDR = 0XFF << 4; // multipleksor bira nulti ulaz za
izlaz, tako da proslijedi izlaz iz perif. A
                           }
                  }
         }

}
```

Zadatak 2:

```
#include <stdio.h>
#include <at91sam7x128.h>

#define START 0X202000
unsigned int volatile * const adresa = (unsigned int *) START;

void sys_handler(void) __irq;
void spurious_handler(void) __irq;


short offset=0; // short mora biti jer inkrementiramo za 2 bajta, int ne moze biti, jer su njegove
adrese poravnate na 4bajta...
int data=0;
int period=0;

// 0 - prva cetvrtina, 1 - druga cetvrtina, 2 - treca cetvrtina, 3 - cetvrta cetvrtina
// Prva cetvrtina      - upis po rastucim adresama
// Druga cevrtina      - upis po padajucim adresama
// Treca cetvrtina      - inverzni upis po rastucim adresama
// Cetvrta cetvrtina - inverzni upis po padajucim adresama


int main (void){
           *AT91C_PMC_PCER = 1 << AT91C_ID_PIOB;

    *(AT91C_AIC_SMR + AT91C_ID_SYS) = (0<<5) | (0<<0);
    *(AT91C_AIC_SVR + AT91C_ID_SYS) = (unsigned int) sys_handler;
    *(AT91C_AIC_SPU) = (unsigned int) spurious_handler;
    *AT91C_AIC_IECR  = 1<<AT91C_ID_SYS;

    *AT91C_PITC_PIMR = (1<<25) | (1<<24) | (999);   // parametri PIT interrupta

           // postaviti 0-15 pinove u 1
```

```
    *AT91C_PIOB_PER = (0xFFFF);
    *AT91C_PIOB_OER = (0xFFFF);
    *AT91C_PIOB_OWER = (0xFFFF);

    while(1);

}

void sys handler(void)   irq {
  data = *(adresa+ 2*offset);  // citanje podatka sa bazne adrese + pomak - crveno je ispravak kolege Jablan koji kaže: u 2. zadatku mi se cini da bi trebalo offset povecavati za 2 te da nije toliko bitno da varijabla offset bude tipa shot. Tj. ako već čitamo poluriječ iz memorije, onda bi varijabla data trebala biti short. A podatak čitamo sa adresa poravnatih na 2, tj. 0x202000, 0x202002, 0x202004...
         if (period==2 || period==3) { // ako se radi o negativnoj poluperiodi
    data = 0 - data; // negativna vrijednost
  }

  *AT91C_PIOB_ODSR = data & 0xFF;

         switch(period){
                  case 0: if(offset < 1000) ++offset;
                           else {--offset; ++period;}
                           break;
                  case 1: if(offset > 0) --offset;
                           else {offset = 1; ++period;}
                           break;
                  case 2: if(offset < 1000) ++offset;
                           else {--offset; ++period;}
                           break;
                  case 3: if(offset > 0) --offset;
                           else {offset = 1; period = 0;}
                           break;
         }

  *AT91C_AIC_EOICR = *AT91C_PITC_PIVR; // PITS reset
}

void spurious_handler(void) __irq {

    *AT91C_AIC_EOICR = 666; // hehehehehe

}
```

Zadatak 3:

```
void reset_init(char x){
         if(x == 'R'){
                  // pogeldajte salic str. 3
                  // ako je R moramo napraviti pravi reset mikrokontrolera
                  // to se omogucuje tako da se upise ona lozinka='KEY' koja je 0xA5 - tako je definirano od ATMEL-a
                  // i jos moramo upisati 1 na bit za enable
                  *AT91C_RSTC_RMR = (0XA5 << 24) | (0 << 0);
         }
         else if( x == 'I'){
                  // AIC je ona periferija za interrupt
                  // da bi to omogucili vidite da postoji jedna zastavica URSTIEN koja mora zatvorit sklopku da bi interrupt otisao do AIC
                  // takoder u tom registru morate upisat onaj KEY i ofc jos jednu sklopkicu zatvorit-URSTEN
                  // mislim da ne treba nista raditi po registrima RSTC_SR
                  *AT91C_RSTC_RMR = (0XA5 << 24) | (1 << 4) | (1 << 0); // RMR je taj registar RSTC_MR, u salicu je napisano kao RMR, ATMEL <3

                  //obzirom da pise da interrupte napravimo samo na razini reset periferije, ne treba nista s onim AIC raditi :)
         }
}
```

Zadatak 4:

```
#include <stdio.h>
#include <at91sam7x128.h>

extern void init USART0(void);
extern void init_PIOA(void);

int main (void){
    char ch;
           init_PIOA();
    init_USART0();

    while (1){
        scanf("%c", &ch);
        printf ("%c", ch);
          }
}
```

retarget.c

```
#include <stdio.h>
#include <at91sam7x128.h>

#pragma import (__use_no_semihosting_swi)

extern void USART0_send(int ch); // obzirom da USART nismo ucili, mislim da ne trebamo znati kako se
USART-om prenose i primaju znakovi
extern int USART0 receiv(void);     // tako da sam ostavile ove funkcije kao da su definirane negdje
drugdje :)

struct __FILE {
    int handle;
};

FILE   stdout,   stdin;

int fputc (int ch, FILE *f){
    USART0_send(ch);
    return ch;
}

int ferror (FILE *f) {
    return 0;
}

int fgetc(FILE *f){
    return USART0_receiv();
}

int __backspace(){
    return 0;
}

void _sys_exit (int return_code){
    while(1);
}
```