MIKROPROCESORSKI SOFTVER VEŽBA 1

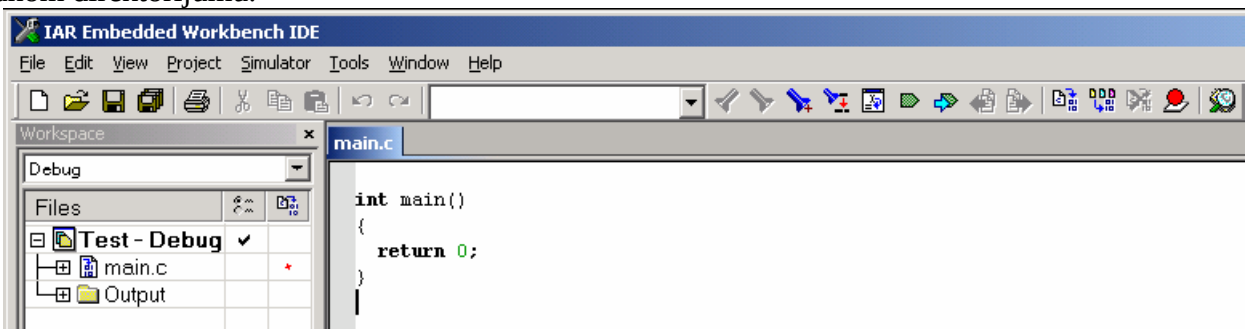
CILJ VEŽBE

- Upoznavanje sa razvojnim okruženjem za mikrokontroler ARM_LPC2148
- Upoznavanje sa aplikacijom ***IAR Embedded Workbench 4.0 Kickstart*** za programiranje mikrokontrolera. Kreiranje novog projekta.
- Upoznavanje sa okruženjem **ISIS *Proteus Design Suite*** za simulaciju rada mikrokontrolera i njegovih periferija.
- Rešavanje praktičnih problema korišćenjem ulazno-izlaznog porta opšte namene (***General Purpose Input Output*** - GPIO).
- Samostalna izrada jednostavnih programa od ideje do testiranja.

ZADATAK 1: Izraditi program za sabiranje dva broja i rezultate pratiti u Debuger-u.

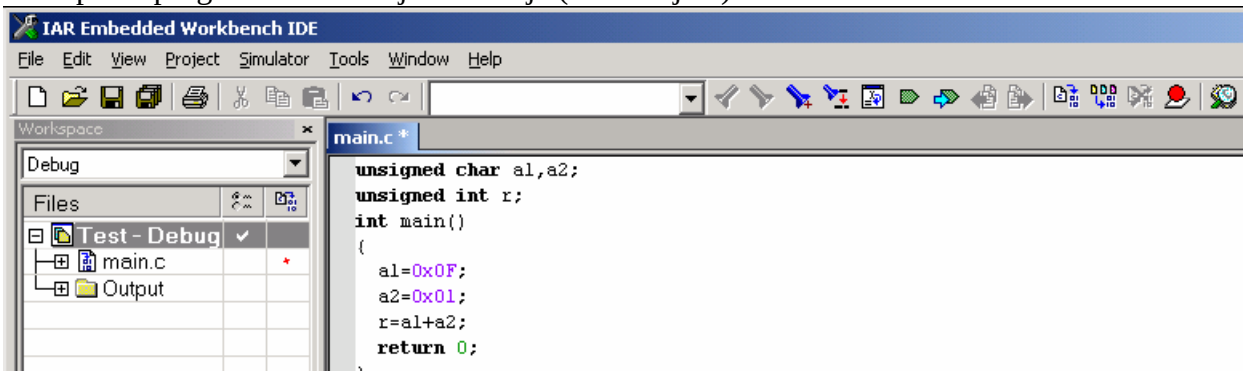
PROCEDURA:

1. Iz **Start** menija pokrenuti aplikaciju ***IAR Embedded Workbench*.** Otvoriti dokument ***IAR Embedded Workbench.pdf*** koji se nalazi u direktorijumu **VEZBE_ LPC2148**. Prateći korake navedene u dokumentu kreirati novi C projekat sa `main()` funkcijom, pod nazivom **Test**. Razvojno okruženje bi trebalo da izgleda kao Ilustracija 1. Sačuvati radni prostor (File -> Save Workspace) pod imenom „Test“ u radnom direktorijumu.



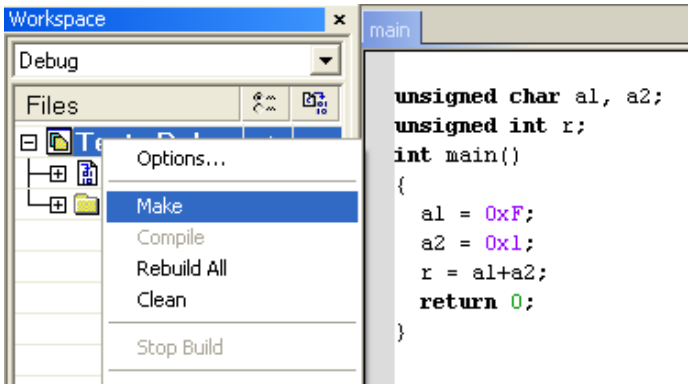
Ilustracija 1 Razvojno okruženje Test projekta sa main funkcijom(u Debug režimu)

2. Napisati program za sabiranje dva broja (Ilustracija 2)

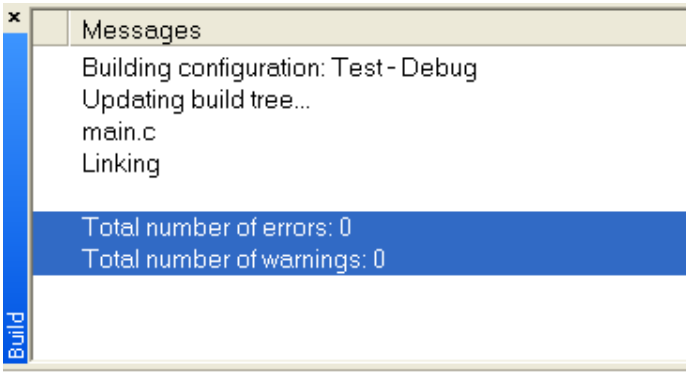


Ilustracija 2 Program za sabiranje dva broja

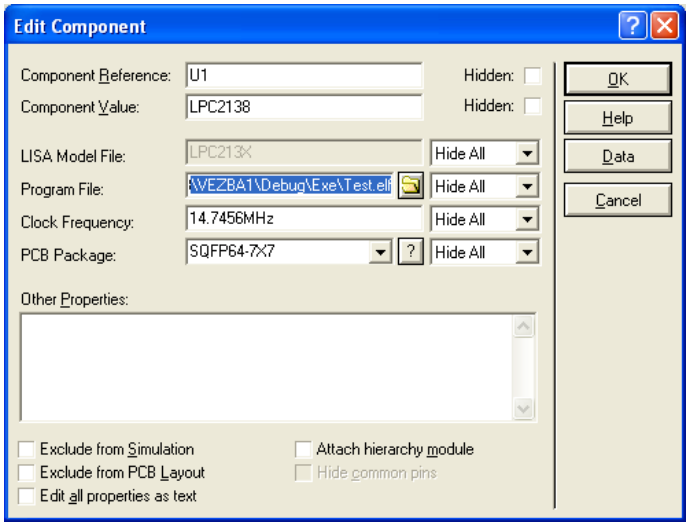
3. Kreirati izlazni fajl odabirom opcije **Make**, kao na slici.



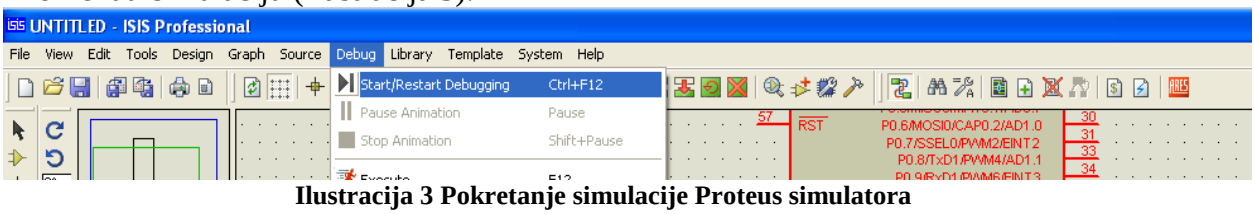
4. Uveriti se da je kreiranje izlazne datoteke prošlo bez greške, kao na slici. Ukoliko nije bilo grešaka, naša izlazna datoteka se nalazi u radnom direktorijumu na putanji: **VEZBE_LPC2148\VEZBA1\Debug\Exe\ Test.elf**. Nju je sada potrebno učitati u simulator.



5. U **Proteus** simulatoru podesiti komponentu mikrokontrolera **U1** (**desni klik→Edit Properties**) tako da putanja do izvršne datoteke bude do datoteke **Test.elf**, kao slici:

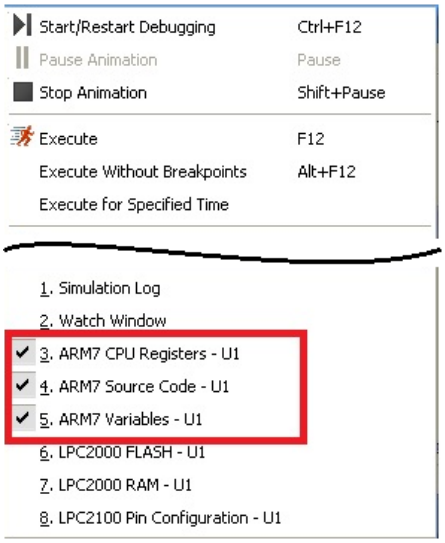


6. Pokrenuti simulaciju (Ilustracija 3).

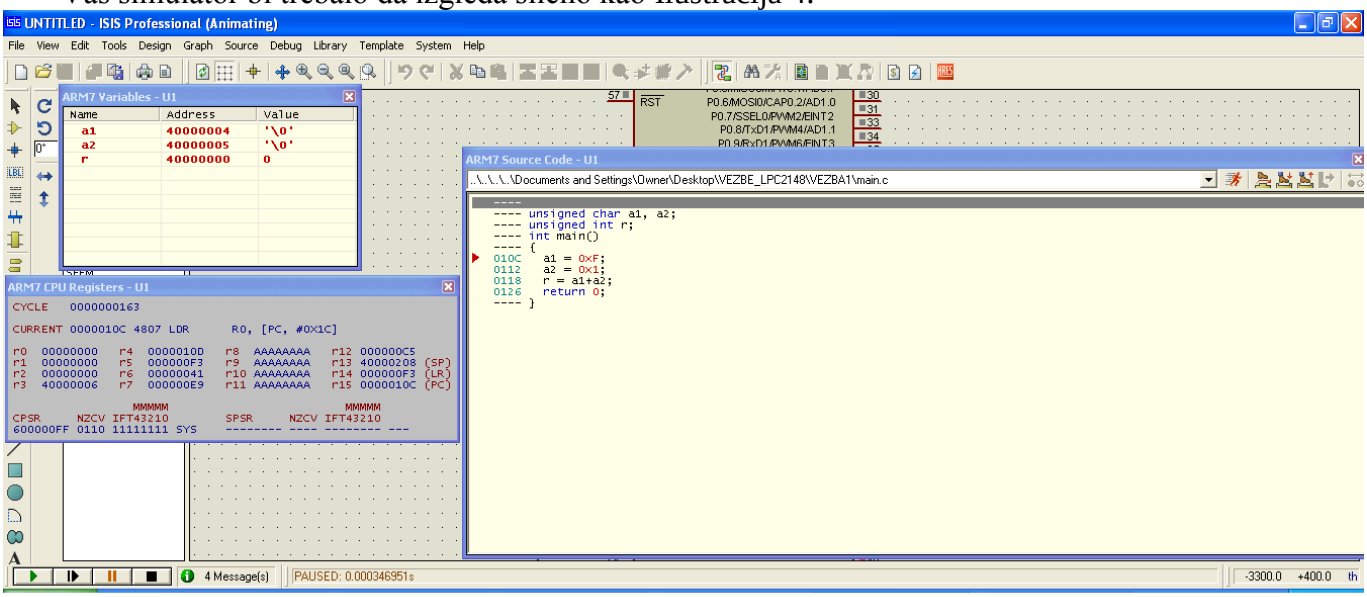


Ilustracija 3 Pokretanje simulacije Proteus simulatora

Ukoliko se nije automatski otvorio, otvorite prozor za posmatranje izlaznog koda, registara procesora i promenljivih, odabirom opcija kao na slici:



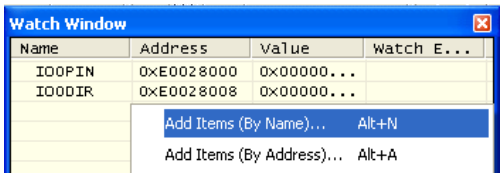
Vaš simulator bi trebalo da izgleda slično kao Ilustracija 4:



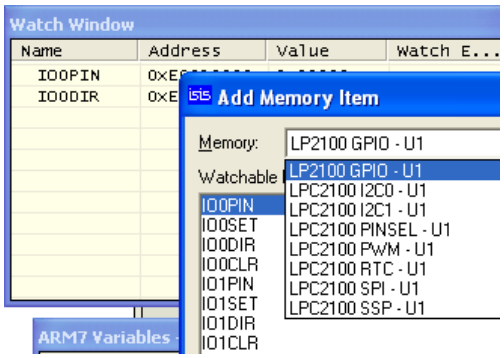
Ilustracija 4 Izgled simulatora prilikom simulacije

Program je zaustavljen na prvoj naredbi u main programu (Ilustracija 4). U desnom prozoru je prikazan izvorni C kod programa. U gornjem levom prozoru vidimo fizičke adrese i vrednosti promenljivih koje su do ovog trenutka deklarisane. U donjem levom uglu vidimo stanja registara procesora.

Ukoliko želimo da posmatramo registre periferija mikrokontrolera, potrebno je uključiti i **Watch Window**. Registre koje želimo da posmatramo dodajemo odabirom opcije **desni klik→Add Items (By Name)**.



Iz padajućeg menija odabiramo kojoj periferiji pripada registar, a zatim iz liste registara odabiramo onaj čije vrednosti želimo da posmatramo.



7. Izvršavati naredbe korak po korak pomoću **Debug** komandi u gornjem desnom uglu Source Code prozora i posmatrati promene vrednosti promenljivih.

ZADATAK 2: Izraditi program koji bi po startovanju naizmenično palio i gasio neku od svetlećih dioda, tako da dioda svetli približno pola sekunde i isto toliko je ugašena.

Uvod:

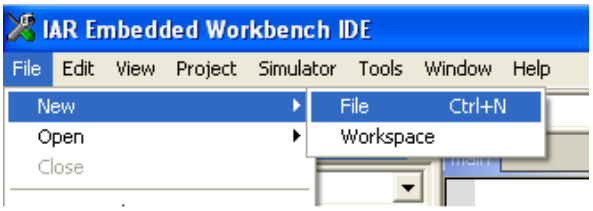
GPIO periferija se sastoji od dva porta (PORT0 i PORT1) pri čemu svaki od njih sadrži do 32 ulazno-izlazne linije. Svakoj ulazno-izlaznoj liniji je pridružena po jedna nožica mikrokontrolera i po jedan bit u odgovarajućem registru podataka (IO0PIN i IO1PIN). Stanje svakog bita u IOPIN registrima je povezano sa logičkim stanjem odgovarajuće nožice mikrokontrolera. Na primer, logičko stanje bita 15 porta 0 (P0.15) odgovara logičkom stanju nožice 45 na mikrokontroleru.

REŠENJE:

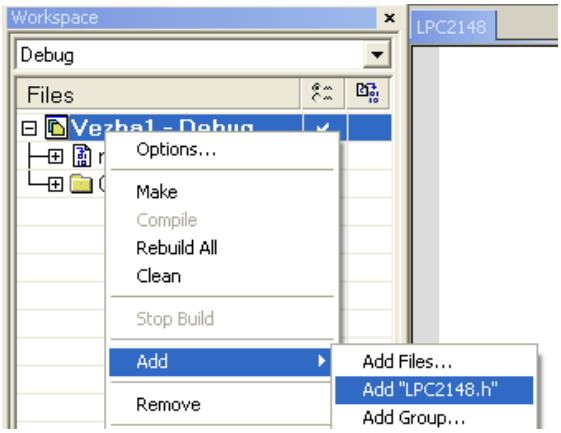
Korišćenjem uputstva sa standardnom uvodnom procedurom za rešavanje zadataka, sa početka praktikuma, pokrenuti i podesiti razvojno okruženje, a zatim uraditi specifična podešavanja za ovaj zadatak koja slede u nastavku.

1. **Kreiranje datoteke zaglavlja (*header file*)**
   Obzirom da je C programski jezik višeg nivoa, u naredbama nije podržano direktno korišćenje adresa memorijskih lokacija, registara i periferija, već je potrebno za takve svrhe koristiti pokazivače. Kako bismo olakšali korišćenje registara periferija i učinili kôd preglednijim, pre početka razvoja kôda glavnog programa deklarisaćemo simbolička imena za periferije koje ćemo koristiti u vežbi. To ćemo uraditi u posebnoj datoteci zaglavlja (***header***) koju ćemo nazvati **LPC2148.h** i uključiti je (pomoću direktive #***include***) u glavni program.

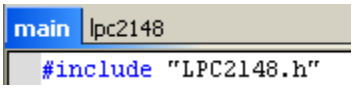
   Kreirati novu datoteku **LPC2148.h** i snimiti je. Na slici je prikazano kako se projektu dodaju nove datoteke. Nakon kreiranja, datoteku treba snimiti odabirom **File→Save** pod imenom „**LPC2148.h**“ u radni direktorijum vežbe.



   Nakon snimanja, datoteku treba dodati u projekat odabirom opcije nad projektom: **Desni klik→Add→Add „LPC2148.h“**.



   U **main** programu dodati na početku direktivu:



2. **Deklarisanje simboličkih imena periferija**
   Za potrebe prvog zadatka od periferija je potrebno da koristimo samo jednu liniju jednog GPIO porta. Stoga ćemo deklarisati samo simbolička imena za pokazivače na registre posredstvom kojih se upravlja portom IOPORT0, čije adrese treba da pronađemo u user manualu za procesor LPC2148:

```
IOPIN0        0xE0028000
IOSET0        0xE0028004
IODIR0        0xE0028008
IOCLR0        0xE002800C
```

   Deklarisanje simboličkog imena u C programskom jeziku se radi pomoću direktive ***#define***, a tip pokazivača na memorijski lokaciju se, u zavisnosti od veličine podatka kojem se pristupa, definiše kao:
   `(*((volatile unsigned char *)` – pokazivač na memorijsku lokaciju jednobajtnog podatka;
   `(*((volatile unsigned short *)` – pokazivač na memorijsku lokaciju dvobajtnog podatka;
   `(*((volatile unsigned long *)` – pokazivač na memorijsku lokaciju četvorobajtnog podatka.
   Obzirom da su svi GPIO registri 32-bitni, koristićemo četvorobajtni tip pokazivača. Prema tome, deklaracija registara IOPORT0 izgleda kao Ilustracija 5:

main | lpc2148

```
#define IOPIN0        (*((volatile unsigned long *) 0xE0028000))
#define IOSET0        (*((volatile unsigned long *) 0xE0028004))
#define IODIR0        (*((volatile unsigned long *) 0xE0028008))
#define IOCLR0        (*((volatile unsigned long *) 0xE002800C))
```

Ilustracija 5 Deklarisanje IOPORT0 registara

3. **Deklaracija tipova**
   Na sličan način (kao za tipove pokazivača na memorijske lokacije u prethodnom odeljku) deklarisaćemo u datoteci i takva skraćena simbolička imena za osnovne C tipove, koja će u nazivu govoriti koliko bita nosi podatak tog tipa, kao i da li predstavlja označeni ili neoznačeni broj:

```
/* Skracena notacija za tipove podataka */
#define C8      char
#define U8      unsigned char
#define U16     unsigned short
#define U32     unsigned long
#define S8      signed char
#define S16     signed short
#define S32     signed long
```

4. **Deklaracija pinova**
   Deklarisaćemo simboličko ime za broj koji ima jedinicu na mestu bita koji želimo da pogodimo na IOPORT0:

```
#define LED07 1<<7
```

5. **Inicijalizacija GPIO periferije**
   Na početku glavnog programa deklarisati funkciju

```
void Init_GPIO(void);
```

   Na kraju glavnog programa definisati funkciju Init_GPIO, u kojoj jedino što treba inicijalizovati je da jedna linija porta bude izlazna, na primer P0.7, setovanjem odgovarajućeg pina u IODIR0 registru (*default* smer za sve linije GPIO je ulazni):

```
void Init_GPIO(void)
{
  IODIR0 |=1<<7;
}
```

6. Na početku glavnog programa deklarisati funkciju

```
void Kasni(U32 vreme);
```

kojoj se kao parameter prosleđuje 32-bitni broj, a služi za vremensko kašnjnje u programu. Funkcija treba da umanjuje ulazni parametar za jedan dok on ne postane 0. Ovakvo umanjivanje broja 360000 traje približno jednu sekundu.
Na kraju glavnog programa definisati funkciju Kasni:

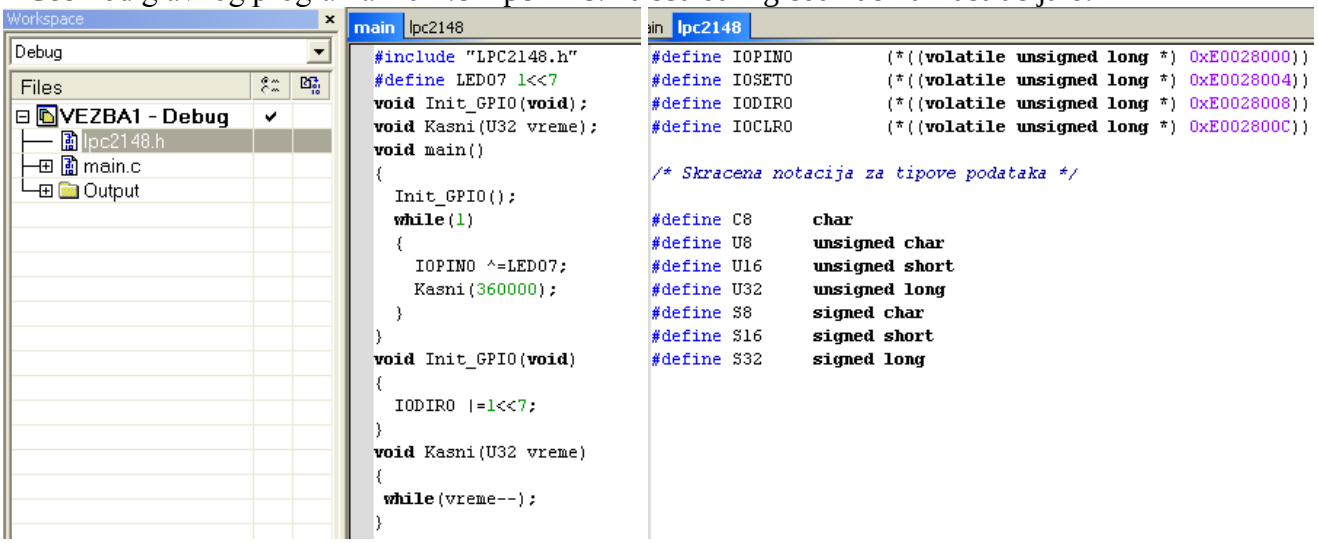
```
void Kasni(U32 vreme)
{
 while(vreme--);
}
```

7. **Glavna programska funkcija**

Na početku glavne programske funkcije najpre ćemo inicijalizovati GPIO periferiju pozivom funkcije `Init_GPIO()`:

```
void main( void )
{
    Init_GPIO();
    while( 1 )
    {
            IOPIN0 ^=LED07;
            Kasni(360000);
    }
}
```

Nakon inicijalizacije izvršava se beskonačna petlja u kojoj se naizmenično upisuje logička 1 i logička 0 na mesto bita 7 registra IO0PIN (IO0PIN.7), što uzrokuje odgovarajuće promene naponskih nivoa nožice pridružene ovom bitu GPIO porta. Između ovih promena stanja pravi se pauza od približno 500 ms pozivom funkcije Kasni().
Ceo kôd glavnog programa main.c i lpc2148.h treba da izgleda kao na Ilustracija 6:



Ilustracija 6 Ceo program vežba 1 - zadatak 1.

Na simulatoru povezati nožicu odabrane izlazne linije GPIO-a mikrokontorlera sa nekom od svetlećih dioda. Podesiti u simlatoru putanju do kreirane .elf datoteke i pokrenuti njegovo izvršavanje.

ZADATAK 3: Izraditi program koji bi na pritisak tastera pokretao treperenje neke od svetljećih dioda.

PROCEDURA:

- Slično kao u drugom zadatku, podesiti registre mikrokontrolera tako da se jedna od linija ulazno-izlaznog porta GPIO (npr. P0.6) koristi kao izlazna, a neka druga kao ulazna (npr P0.8).
- Projekat kreiran u okruženju ***IAR Embedded Workbench*** za potrebe prvog zadatka, izmeniti tako da se paljenje i gašenje svetlećih dioda pokrene tek nakon što se sa odabrane ulazne linije ulazno-izlaznog porta (GPIO) pročita stanje pritisnutog tastera. Prevesti program.
- Nožicu odabrane ulazne linije GPIO-a mikrokontrolera u simulatoru povezati sa nekim od tastera.
- Nožicu odabrane izlazne linije GPIO-a mikrokontrolera u simulatoru povezati sa nekom od svetlećih dioda.
- Podesiti u simlatoru putanju do kreirane .elf datoteke i pokrenuti njegovo izvršavanje.