Představení laboratorního kitu

IMP – demo cvičení

Ing. Václav ŠIMEK

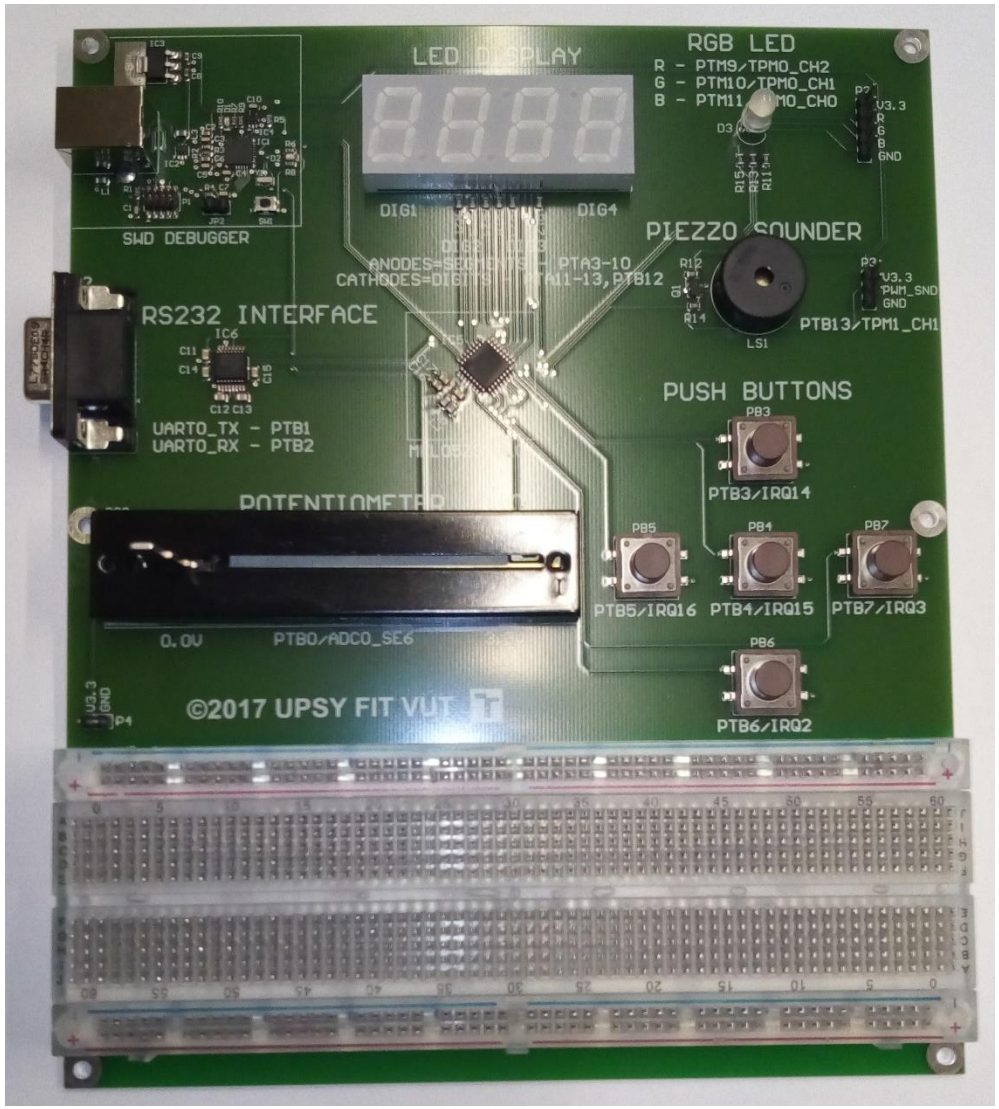
Vysoké učení technické v Brně, Fakulta informačních technologií
Božetěchova 1/2, 612 66 Brno, Česká republika
simekv@fit.vutbr.cz

30.09.2022

Obsah

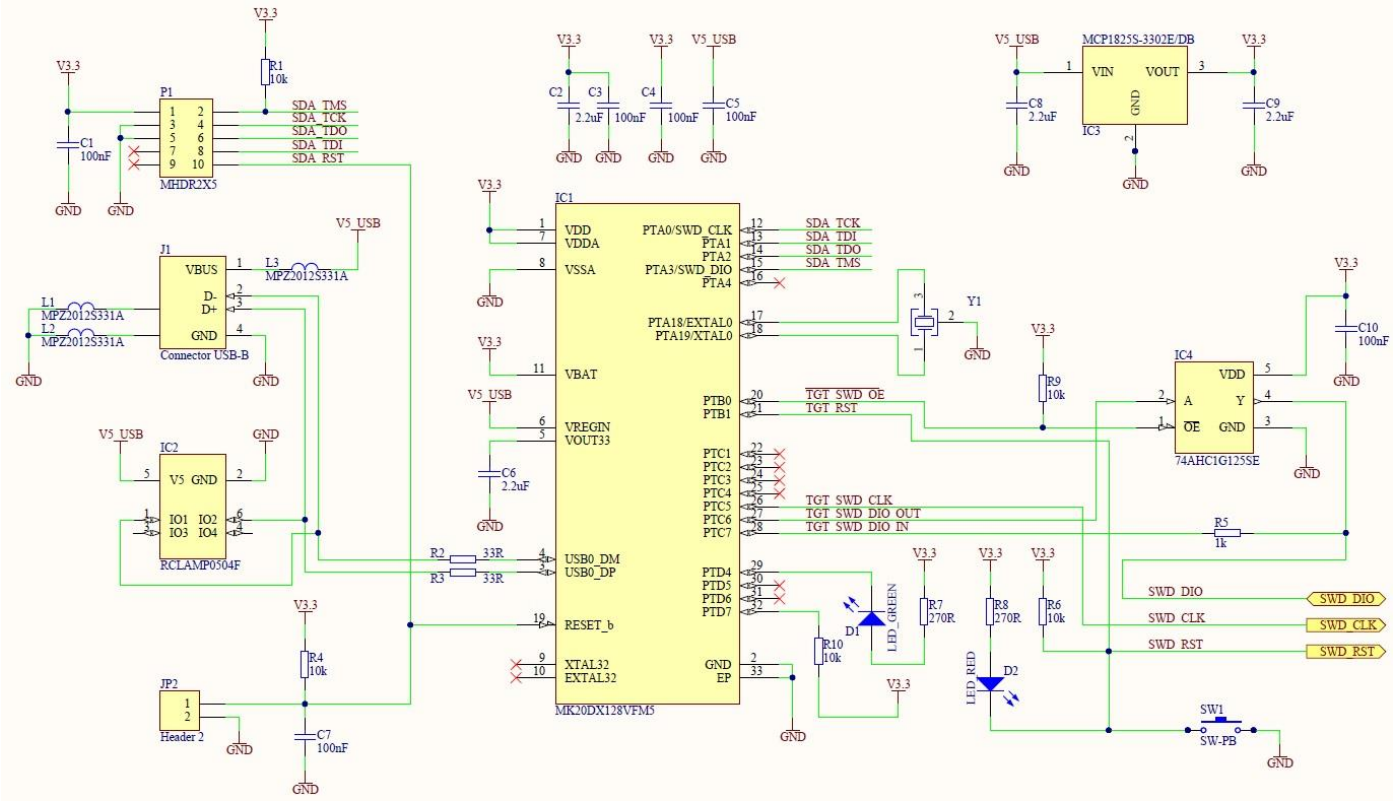
- Výukový kit s 32-bit ARM MCU
- Schéma zapojení a popis periferií
- Vývojové prostředí KDS
- Praktická ukázka kódu

Výukový kit s 32-bit ARM MCU – finální verze



V této podobě je kit nainstalován v laboratoři L306

Schéma zapojení – ladicí rozhraní OpenSDA



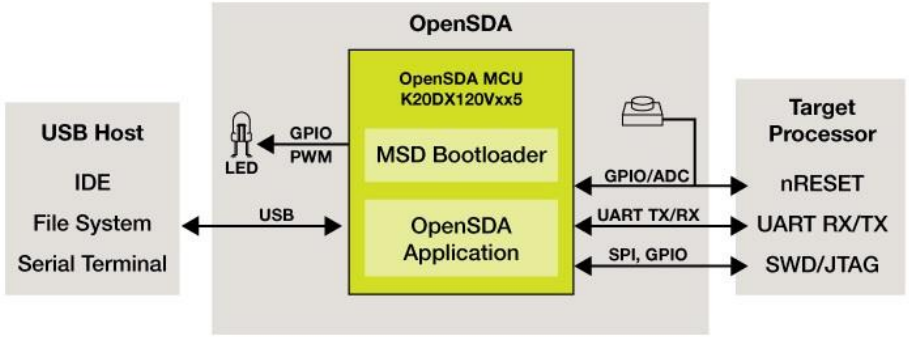


Schéma zapojení – mikrokontrolér MKL05Z32

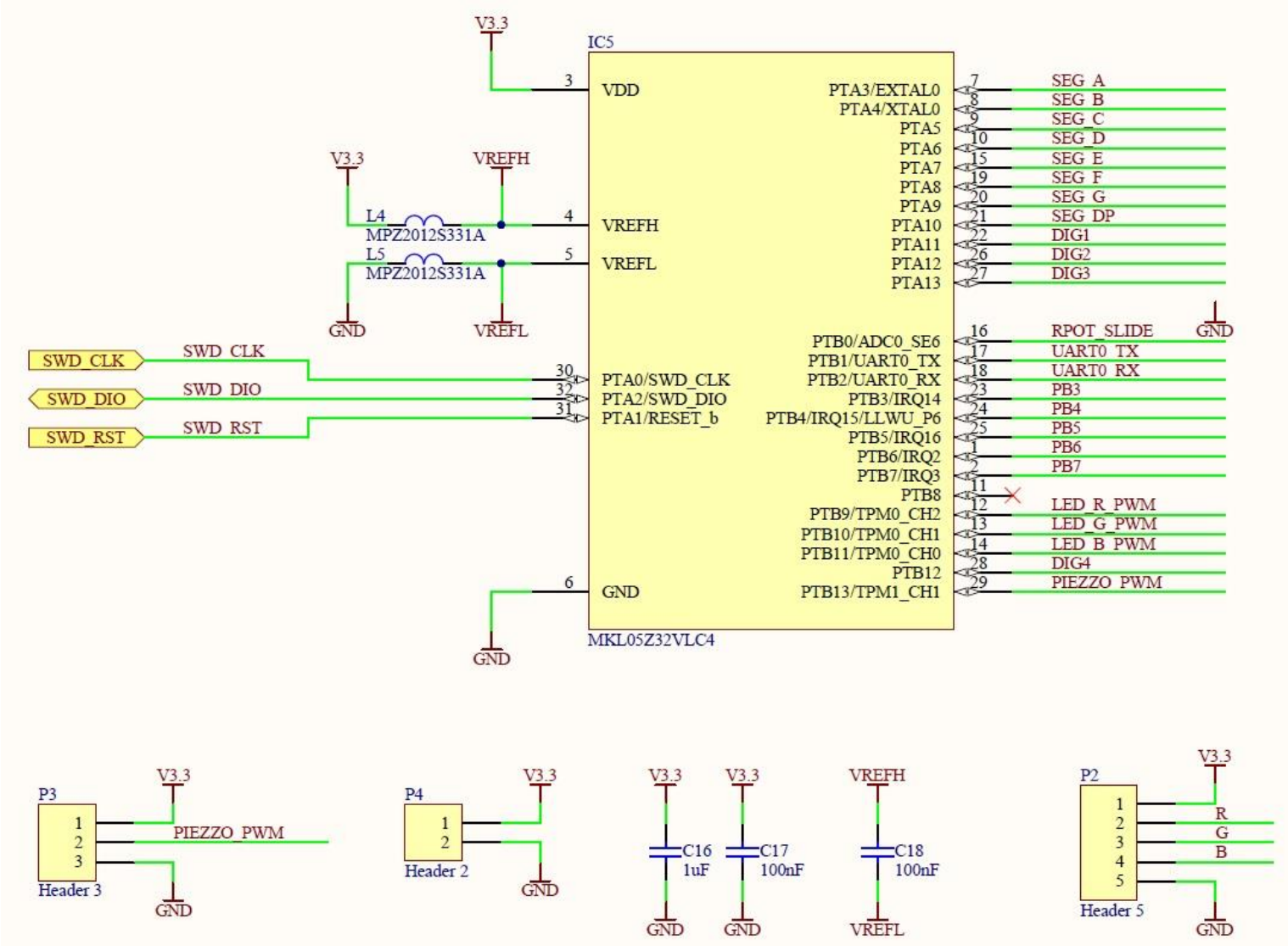


Schéma zapojení – sériová linka RS232

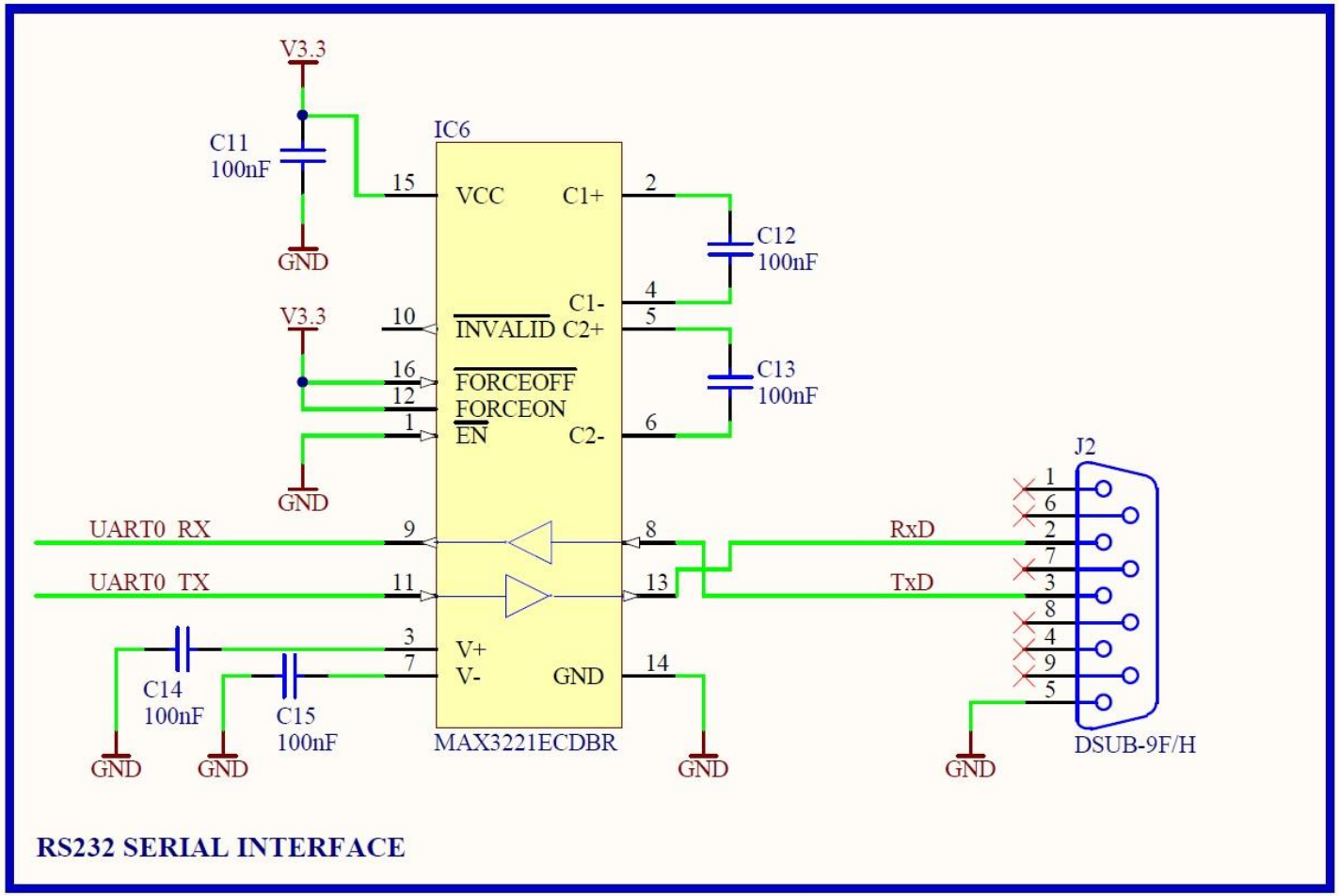


Schéma zapojení – mechanická tlačítka

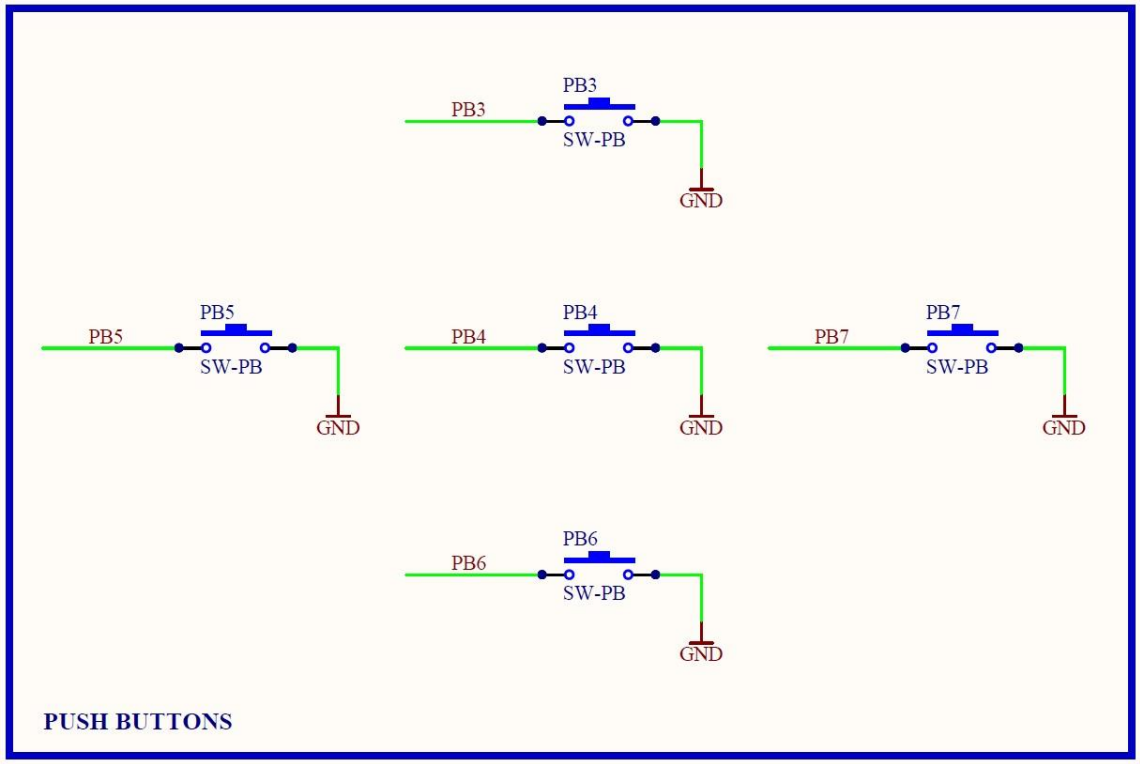
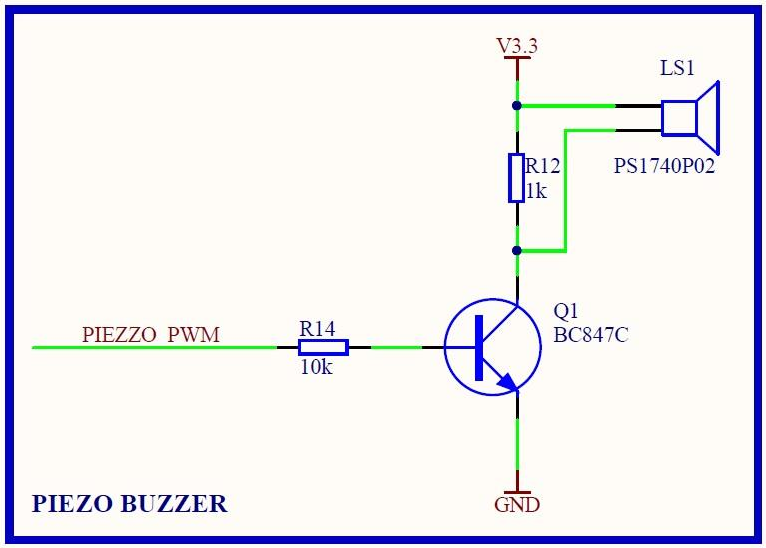


Schéma zapojení – piezzo bzučák a RGB led



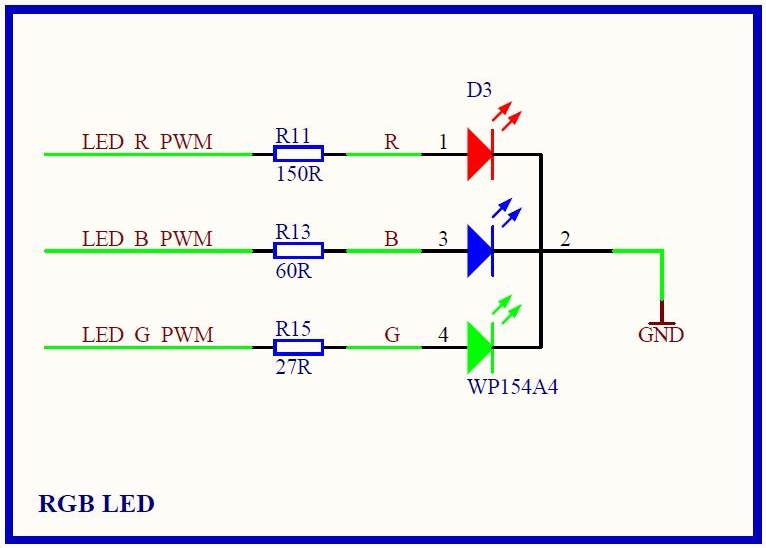
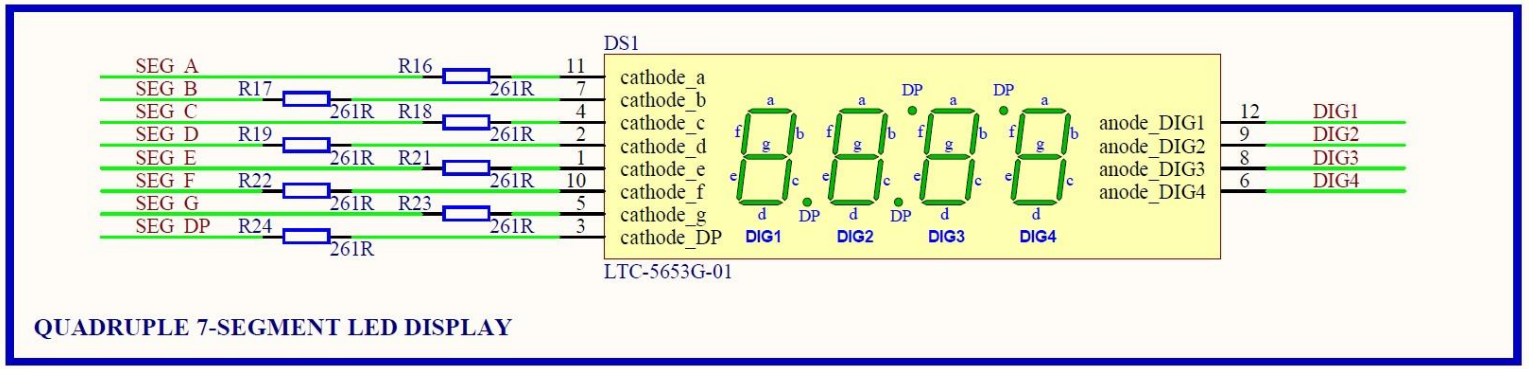
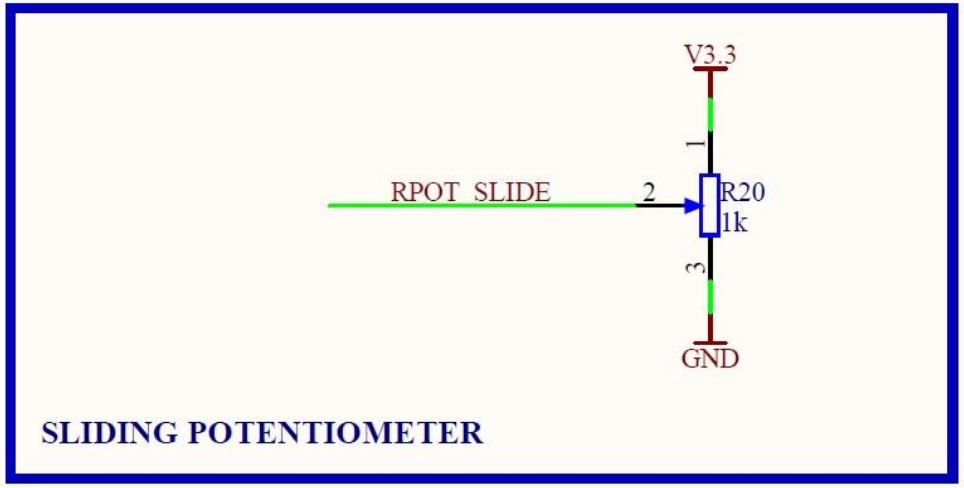
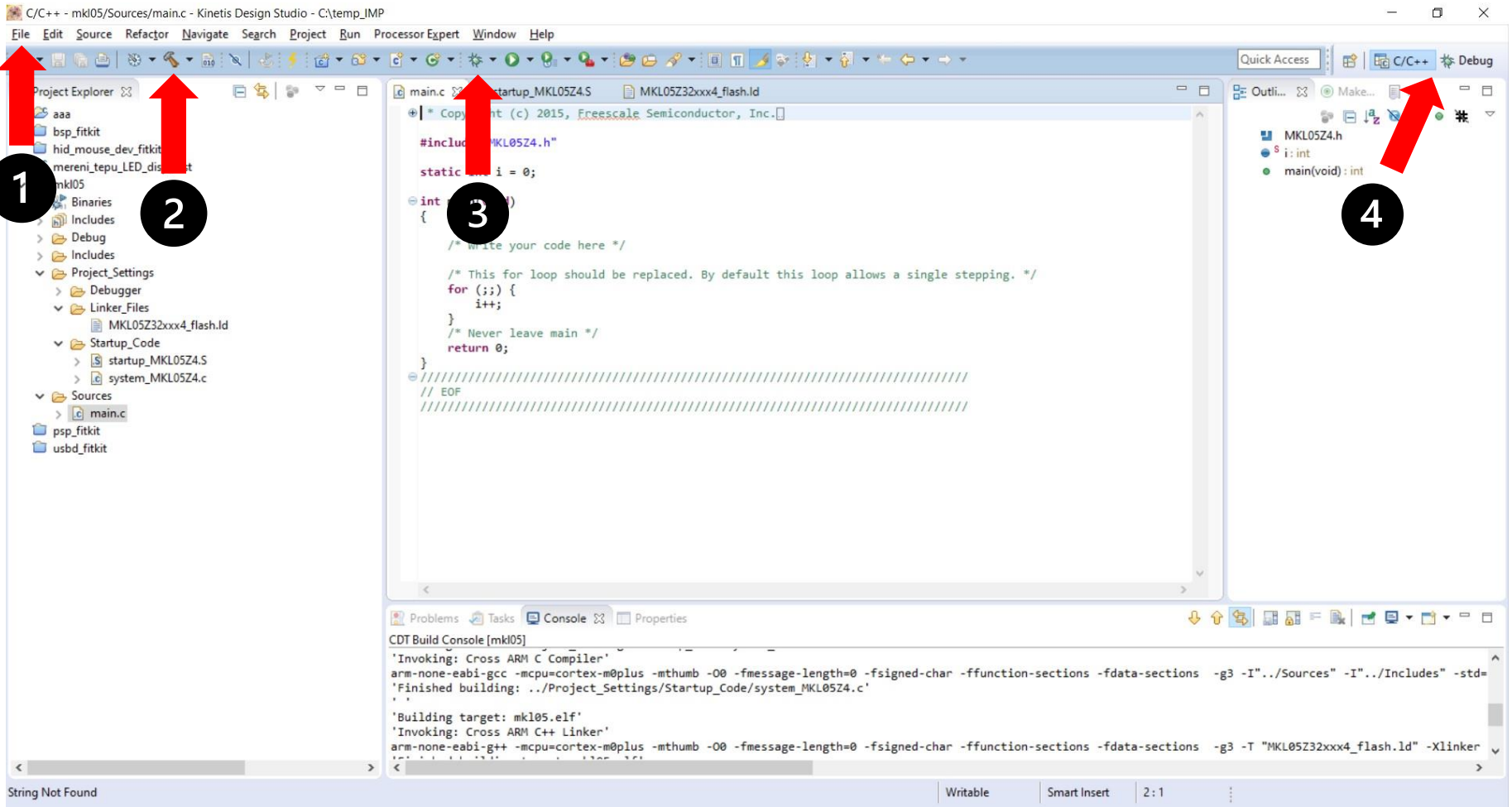


Schéma zapojení – 4-místný LED displej a potenciometr





Vývojové prostředí KDS – základní ovládací prvky



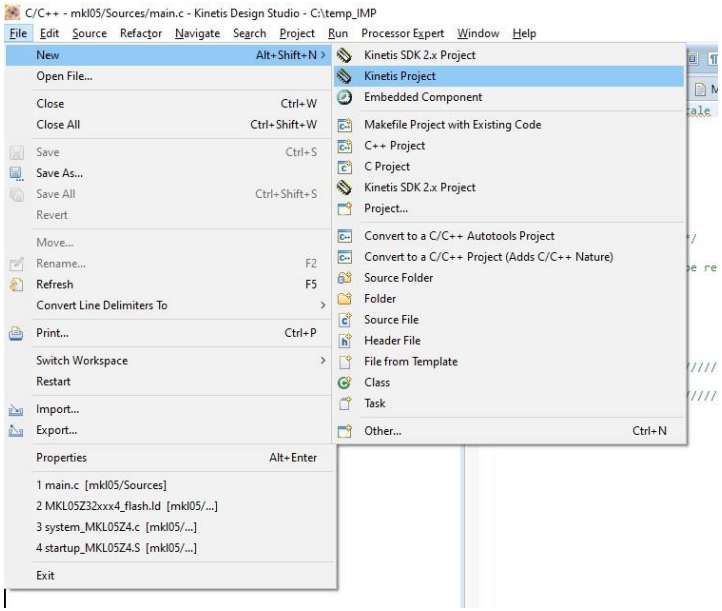
1) funkce pro založení projektu a jeho základní konfiguraci

2) překlad projektu

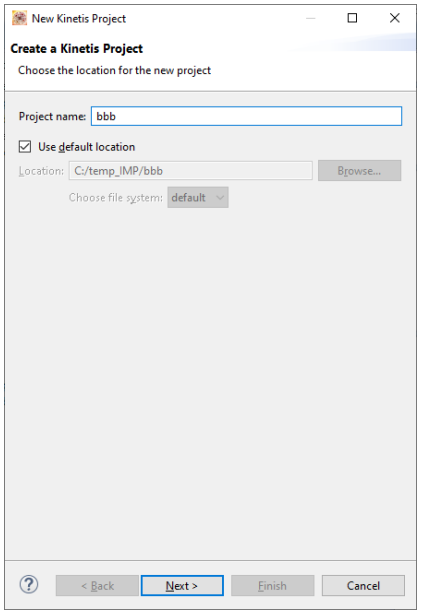
3) nastavení ladicího rozhraní a nahrání výsledné binárky do MCU

4) přepínání mezi editací kódu a laděním (***C/C++ perspective***, ***Degug perspective***)

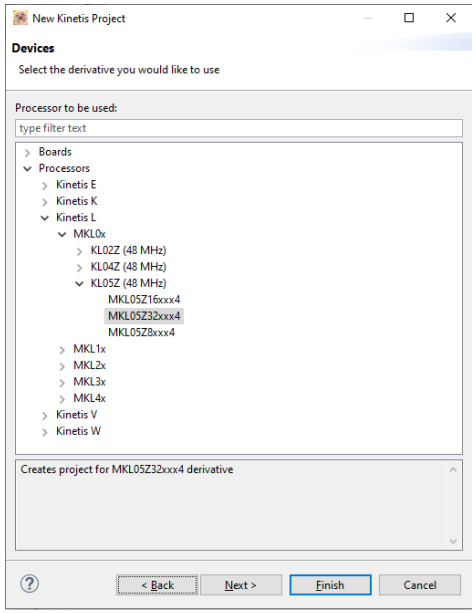
Vývojové prostředí KDS – postup při založení projektu



1



2



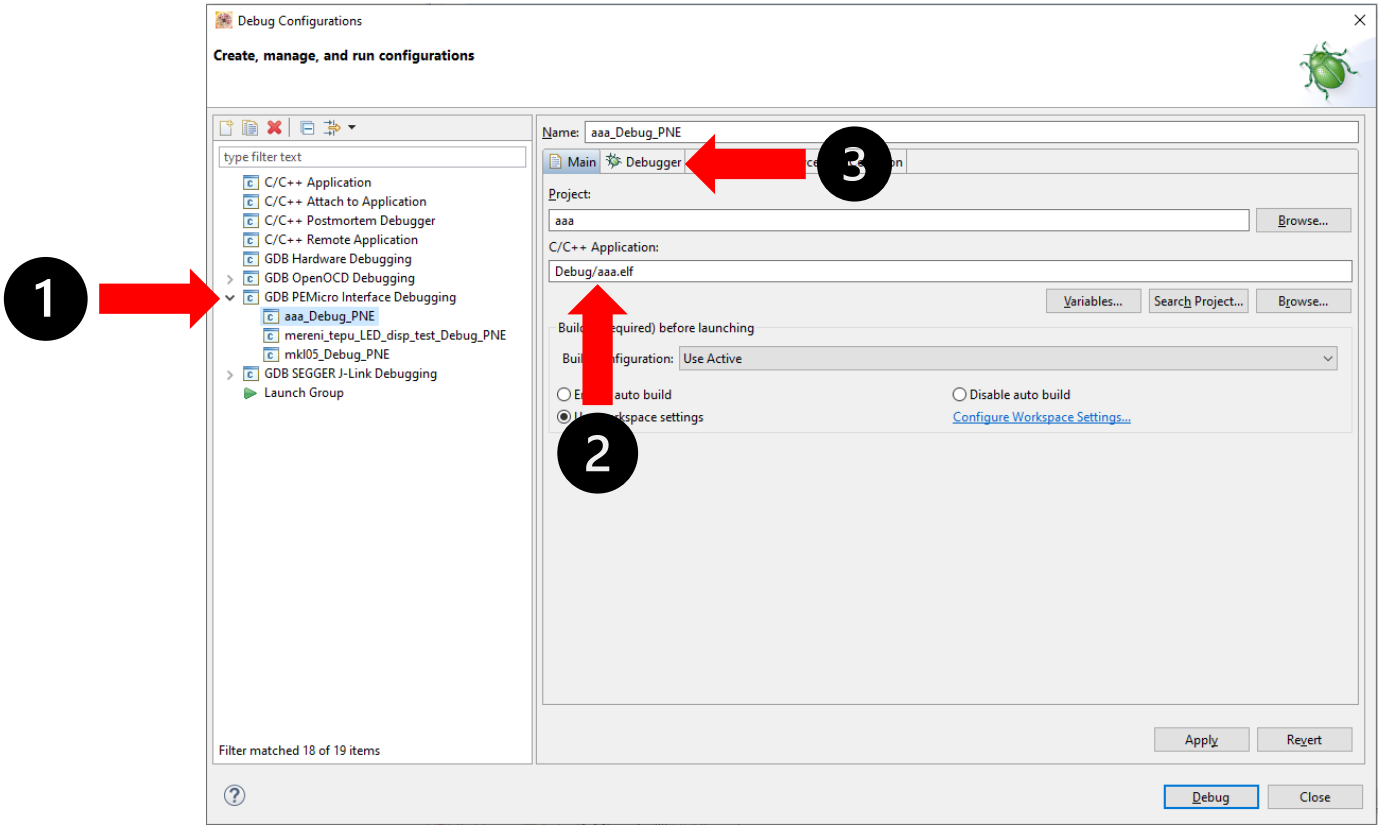
3

1) v menu „*File*“ zvolíme položku **NEW** a dále **Kinetis Project**

2) do kolonky „*Project name*“ zadáme název projektu

3) v menu „*Devices*“ vybereme požadovaný typ mikrokontroleru, tj. MKL05Z32

Vývojové prostředí KDS – konfigurace ladicího rozhraní

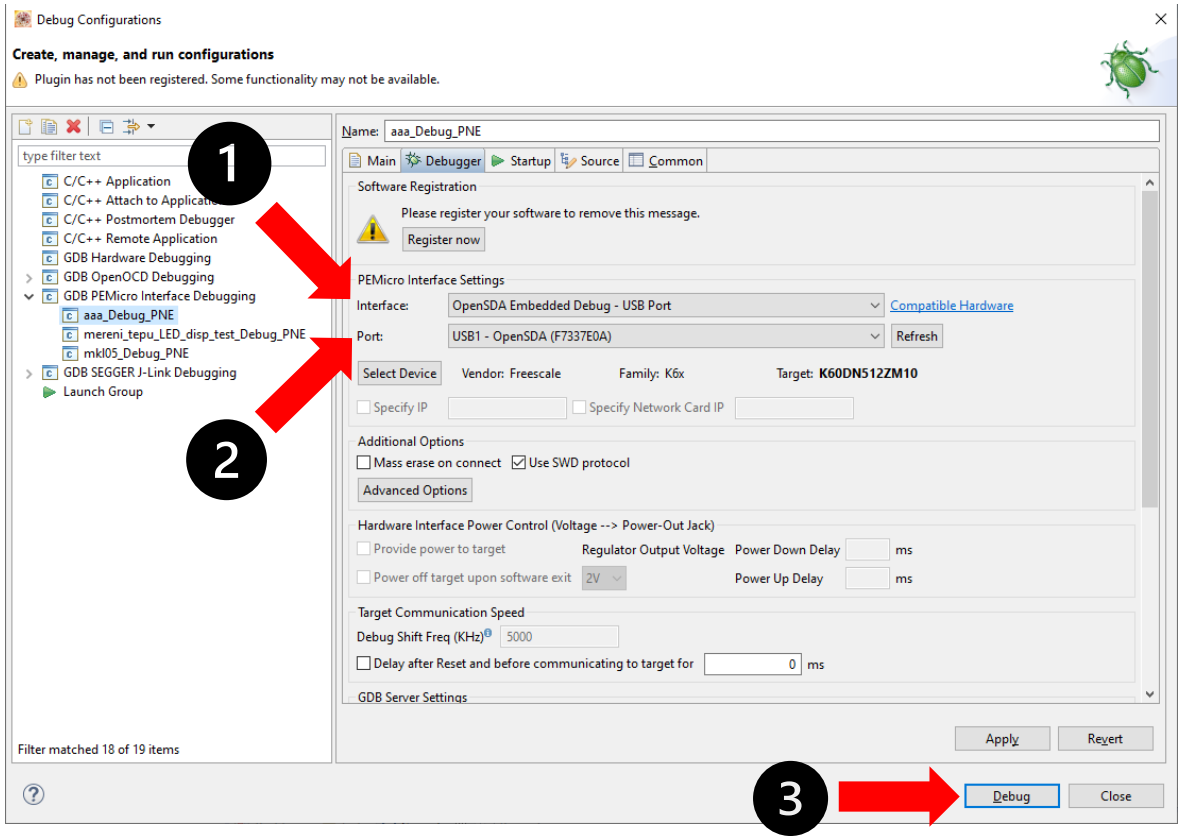


1) ve sloupci vlevo najdeme rozbalovací menu „*GDB PEMicro Interface Debugging*“, v němž klikneme na položku s názvem našeho projektu

2) zkontrolujeme, zda řádek „*Project*“ obsahuje název našeho projektu a zda je na řádku „C/C++ Application“ uvedena cesta k výsledné binárce

3) přejdeme na záložku „Debugger“ a dokončíme nastavení ladicího rozhraní pro náš projekt

Vývojové prostředí KDS – konfigurace ladicího rozhraní

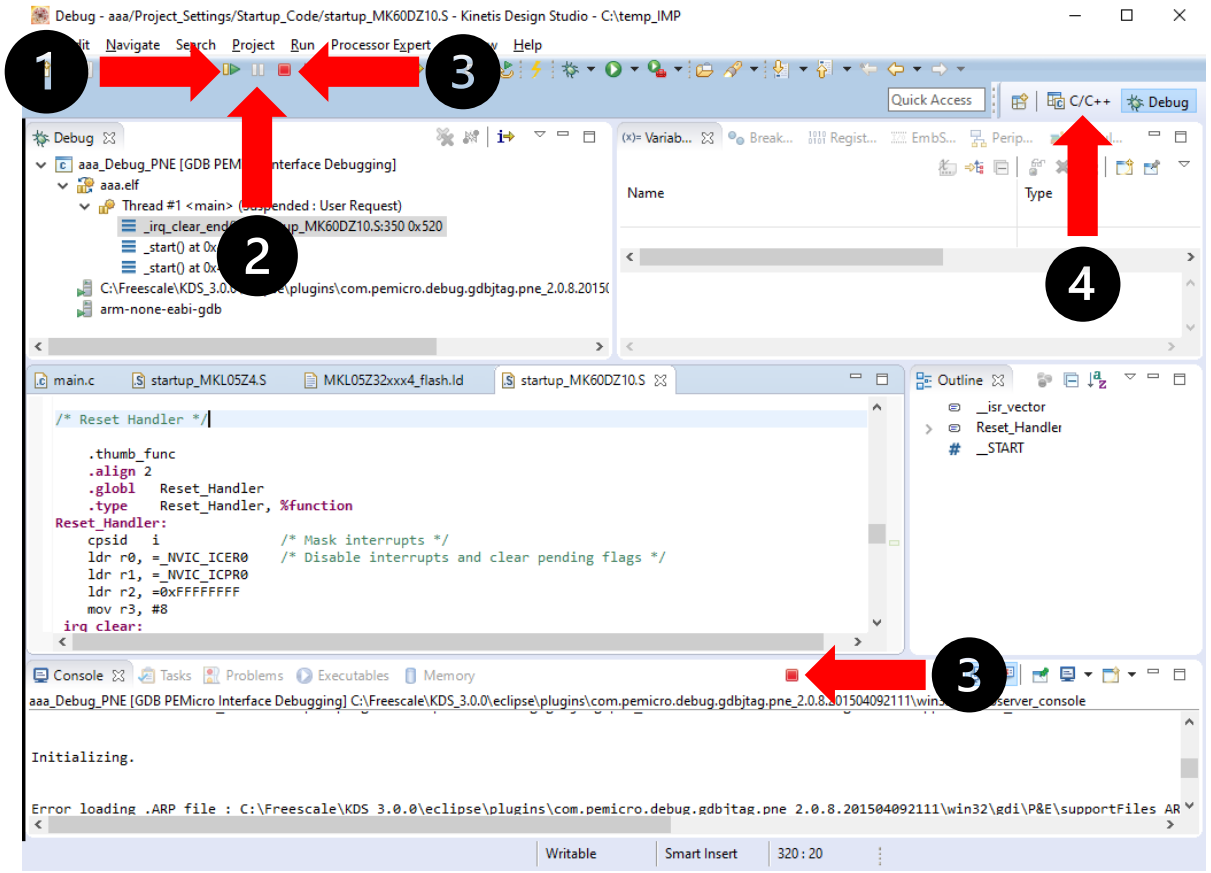


1) zkontrolujeme, že řádek „*Interface*“ obsahuje správný typ debuggeru pro náš projekt, tj. ***OpenSDA Embedded Debug – USB Port***

2) dále by měla být na řádku „Port“ vidět konkrétní instance připojeného debuggeru, který se nachází na laboratorním kitu

3) pokud jsou všechna nastavení v pořádku, můžeme proces nahrávání kódu do MCU a jeho ladění zahájit kliknutím na ikonku „*Debug*“

Vývojové prostředí KDS – spuštění nahraného kódu



1) zelená šipka spustí výsledný kód našeho projektu, který byl nahrán do MCU

2) tlačítko „pauza“ pozastaví provádění kódu, pro pokračování provádění kódu naší aplikace klikneme na zelenou šipku

3) pokud již chceme ukončit ladění aplikace, je třeba kliknout na červený čtvereček a ověřit, že se jeho barva změnila na „šedivou“, jinak nepůjde ladění opět spustit

4) po ukončení ladění klikneme na pohled „C/C++“ a přepneme se zpět do editačního režimu

Ukázka kódu – ovládání piezzo bzučáku

```
#include "MKL05Z4.h"                          // Just an ordinary delay loop

void delay(long long bound) {
     long long i;
     for(i=0;i<bound;i++);
}

int main(void){
     SIM->COPC = SIM_COPC_COPT(0x00);          // Just disable the usage of WatchDog feature
     SIM->SCGC5 = SIM_SCGC5_PORTB_MASK;        // Turn on clocks for PORTA and PORTB

     PORTB->PCR[13] = ( 0|PORT_PCR_MUX(0x01) ); // Set corresponding PORTB port pins as outputs
     PTB->PDDR = GPIO_PDDR_PDD( 0x2000 );      // "1" configures given pin as an output

     for (;;) {
          PTB->PDOR = GPIO_PDOR_PDO(0x2000);
          delay(500);
          PTB->PDOR = GPIO_PDOR_PDO(0x0000);
          delay(500);
     }
     return 0;
}
```

Vývojové prostředí KDS – názvy přerušovacích rutin

```
/* External Interrupts*/
    .long   DMA0_IRQHandler                 /* DMA channel 0 transfer complete*/
    .long   DMA1_IRQHandler                 /* DMA channel 1 transfer complete*/
    .long   DMA2_IRQHandler                 /* DMA channel 2 transfer complete*/
    .long   DMA3_IRQHandler                 /* DMA channel 3 transfer complete*/
    .long   Reserved20_IRQHandler           /* Reserved interrupt*/
    .long   FTFA_IRQHandler                 /* Command complete and read collision*/
    .long   LVD_LVW_IRQHandler              /* Low-voltage detect, low-voltage warning*/
    .long   LLWU_IRQHandler                 /* Low leakage wakeup Unit*/
    .long   I2C0_IRQHandler                 /* I2C0 interrupt*/
    .long   Reserved25_IRQHandler           /* Reserved interrupt*/
    .long   SPI0_IRQHandler                 /* SPI0 single interrupt vector for all sources*/
    .long   Reserved27_IRQHandler           /* Reserved interrupt*/
    .long   UART0_IRQHandler                /* UART0 status and error*/
    .long   Reserved29_IRQHandler           /* Reserved interrupt*/
    .long   Reserved30_IRQHandler           /* Reserved interrupt*/
    .long   ADC0_IRQHandler                 /* ADC0 interrupt*/
    .long   CMP0_IRQHandler                 /* CMP0 interrupt*/
    .long   TPM0_IRQHandler                 /* TPM0 single interrupt vector for all sources*/
    .long   TPM1_IRQHandler                 /* TPM1 single interrupt vector for all sources*/
    .long   Reserved35_IRQHandler           /* Reserved interrupt*/
    .long   RTC_IRQHandler                  /* RTC alarm*/
    .long   RTC_Seconds_IRQHandler          /* RTC seconds*/
    .long   PIT_IRQHandler                  /* PIT interrupt*/
    .long   Reserved39_IRQHandler           /* Reserved interrupt*/
    .long   Reserved40_IRQHandler           /* Reserved interrupt*/
    .long   DAC0_IRQHandler                 /* DAC0 interrupt*/
    .long   TSI0_IRQHandler                 /* TSI0 interrupt*/
    .long   MCG_IRQHandler                  /* MCG interrupt*/
    .long   LPTMR0_IRQHandler               /* LPTMR0 interrupt*/
    .long   Reserved45_IRQHandler           /* Reserved interrupt*/
    .long   PORTA_IRQHandler                /* PORTA Pin detect*/
    .long   PORTB_IRQHandler                /* PORTB Pin detect*/
```

Děkuji za pozornost…