Obsluha platformy FITkit3 (a.k.a. Minerva) se zaměřením na MCU ARM Kinetis-K60 a vývojové prostředí Kinetis Design Studio

Michal Bidlo

Obsah

- Tento materiál obsahuje step-by-step návod k základnímu používání FITkitu3 z pohledu programování mikrokontroléru Kinetis-K60.
- Součástí je též demonstrační příklad obsluhy tlačítek a soustavy LED prostřednictvím univerzálních I/O portů s použitím časovače a obsluhou přerušení.
- Další část zahrnuje (formou demo ukázek):
  - Informace k možným způsobům přístupu k registrům jednotlivých modulů MCU,
  - Konstanty, makra a další podpůrné funkce,
  - Praktické aspekty programování obsluhy přerušení

Instalace IDE Kinetis Design Studio (KDS)

- KDS je ke stažení ze stránek společnosti Freescale:
  http://www.freescale.com/tools/software-and-tools/run-time-software/kinetis-software-and-tools/ides-for-kinetis-mcus/kinetis-design-studio-integrated-development-environment-ide:KDS_IDE?code=KDS_IDE&nodeId=0152109D3F1E8C1EB4&fpsp=1&tab=Design_Tools_Tab

- Před stažením je nutno se zaregistrovat, jako adresu můžete uvést sídlo fakulty (Božetěchova 2, 61266 Brno)
- KDS je k dispozici pro různé platformy (včetně Linux-based v podobě předpřipravených balíčků)
- Stáhněte si potřebnou verzi dle preferovaného OS a nainstalujte obvyklým způsobem. Licence netřeba.

  Doporučujeme stáhnout verzi KDS 3.0.0.

- Dokumentace k použitému MCU je dostupná zde:
  http://cache.freescale.com/files/32bit/doc/ref_manual/K60P144M100SF2V2RM.pdf

Konfigurace KDS v OS Linux

- KDS 3.0 je 32-bitová aplikace, pokud má být provozována na 64-bit systému, je nutno doinstalovat následující balíčky **ve verzi pro architekturu i386:**
  - libc6
  - libstdc++
  - libncurses5
  - Libusb
- V Debian-like OS typicky přidáním architektury i386 zadáním:

```
sudo dpkg  --add-architecture i386
sudo apt-get update
```

- Následně dohledejte a nainstalujte příslušné balíčky pomocí nástroje aptitude, v názvu musí mít suffix „:i386“.

Konfigurace KDS v OS Linux

- Zprovoznění komunikace s připojeným kitem vyžaduje instalaci příslušných ovladačů, v našem případě PEMicro USB Drivers, přidáním příslušných pravidel pro udev, což lze provést následujícím způsobem:

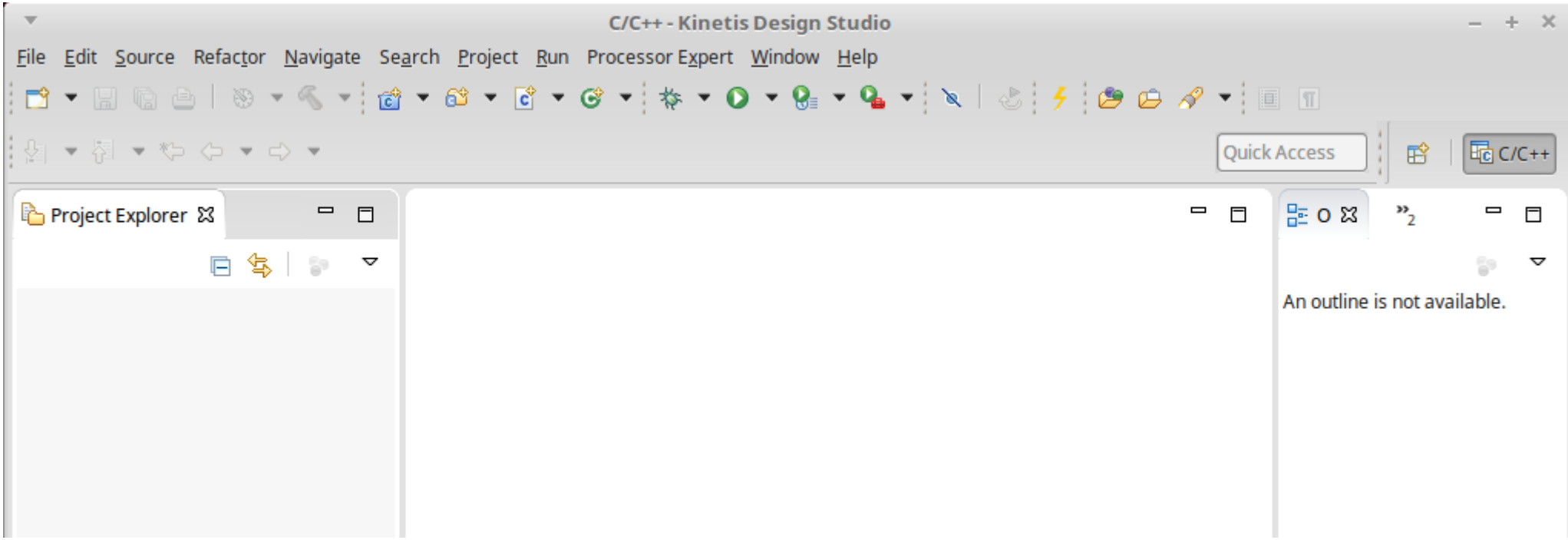
  S oprávněním roota spusťe skript setup.sh v adresáři /opt/Freescale/KDS_3.0.0/pemicro/drivers/libusb_64_32.

  Dojde k nakopírování souboru s pravidly 28-pemicro.rules do /etc/udev/rules.d, případnou chybu s udevcontrol lze ignorovat, podstatné je, aby došlo ke zkopírování tohoto souboru a spuštění /sbin/udevadm control --reload-rules, což lze provést i ručně (případně restartovat počítač).

- Tím je vše připraveno k použití.

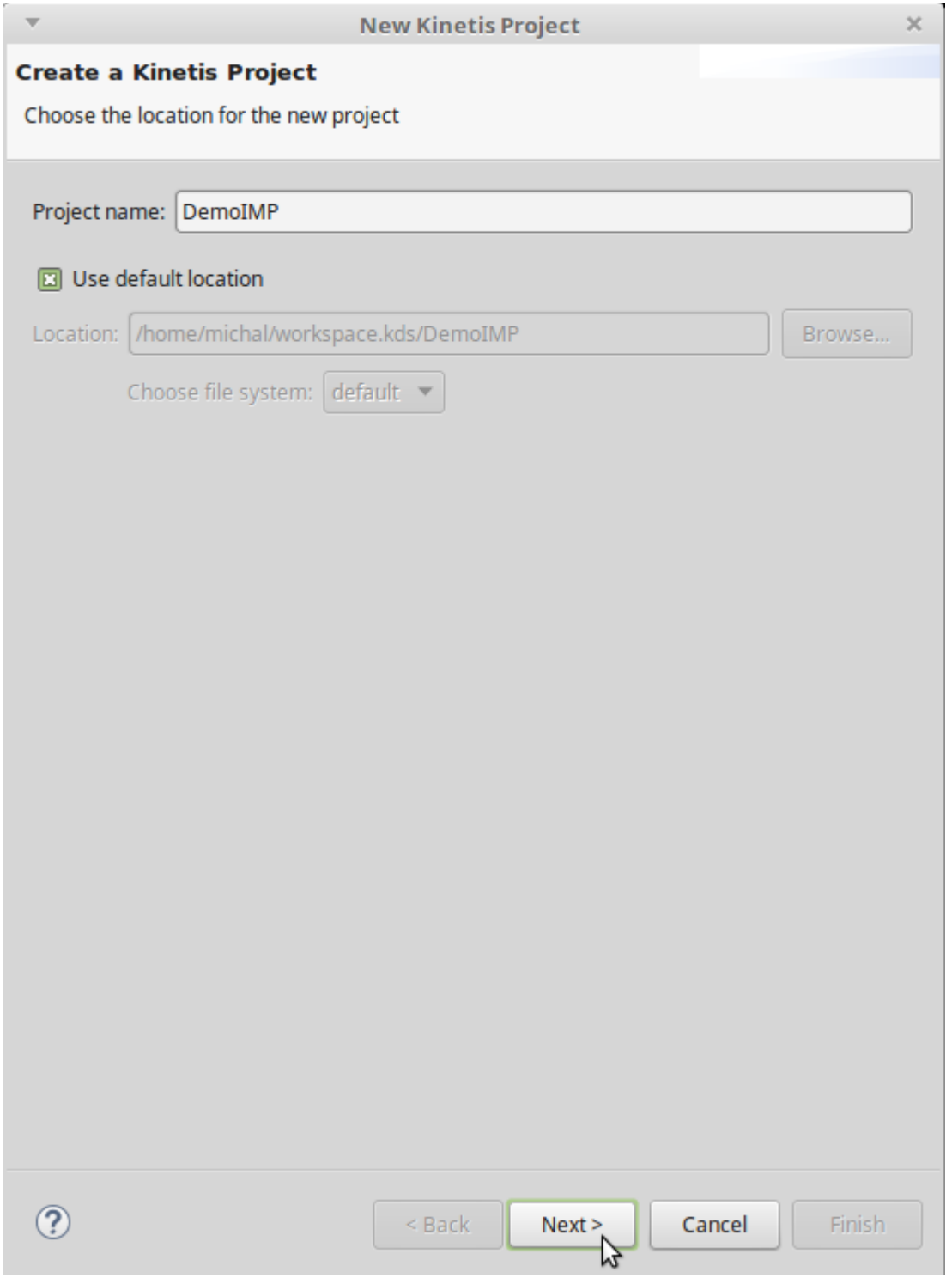
První spuštění KDS, základní nastavení

- Po spuštění KDS obvykle vyzve k volbě adresáře pro tzv. workspace. Jedná se o adresář, kam budou ukládány jednotlivé projekty a nastavení. Obvykle je potřeba toto provést pouze jednou.
- Chcete-li později umístění pro workspace změnit, je možné toto provést menu File → Switch Workspace. Po nastavení workspace se KDS obvykle automaticky restartuje.
- Zavřete kartu uvítacího okna, dostanete se do výchozího IDE KDS:



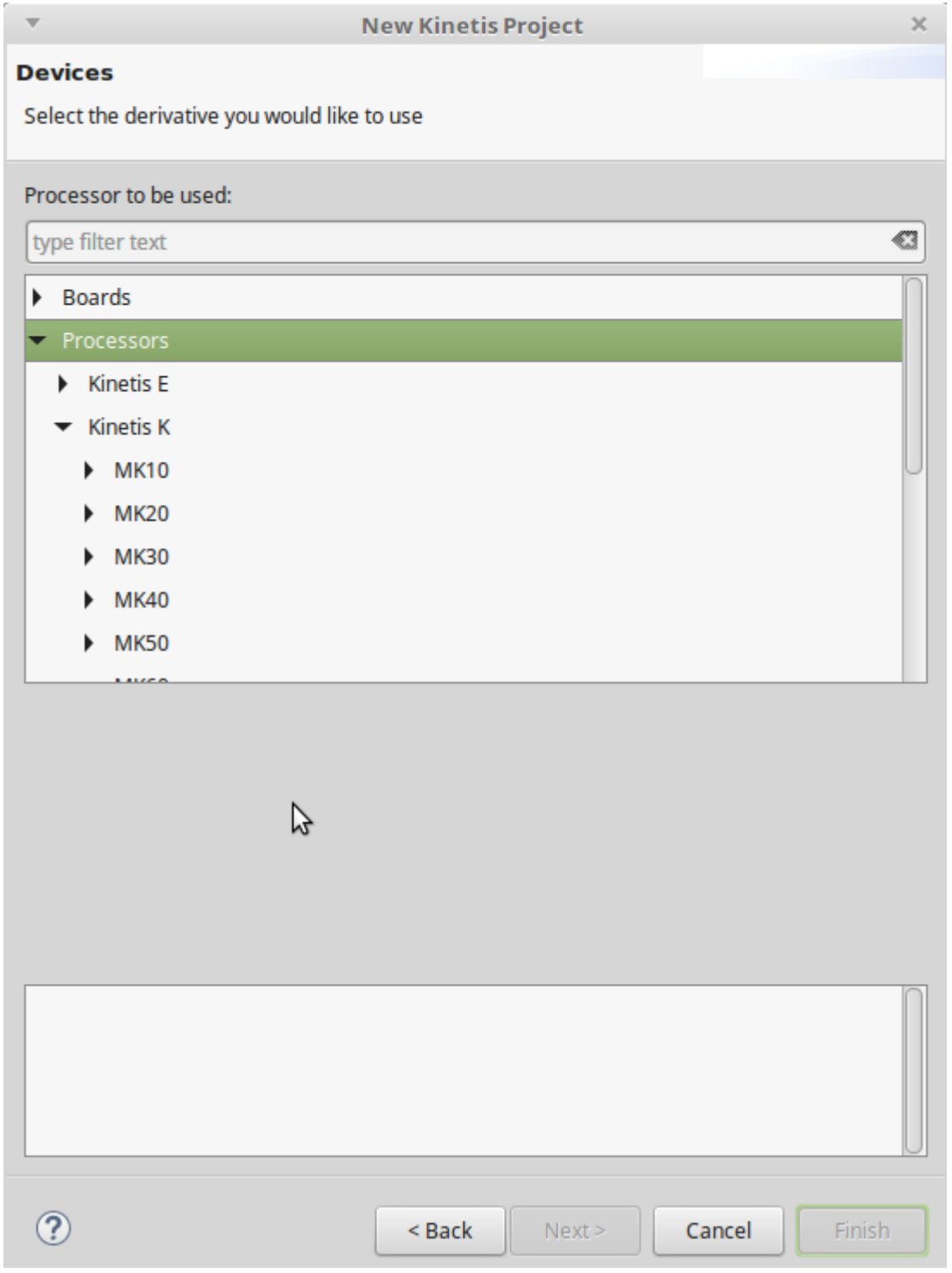
Vytvoření projektu

- Pro potřeby další činnosti (programování, ladění) je nutné založit nový projekt. Toto provedete v menu **File → new → Kinetis Project**
- Zadejte název projektu a jeho umístění (doporučuji zvolit „use default location“, umístí se do adresáře nastaveného pro workspace. Stiskněte tlačítko **Next >**



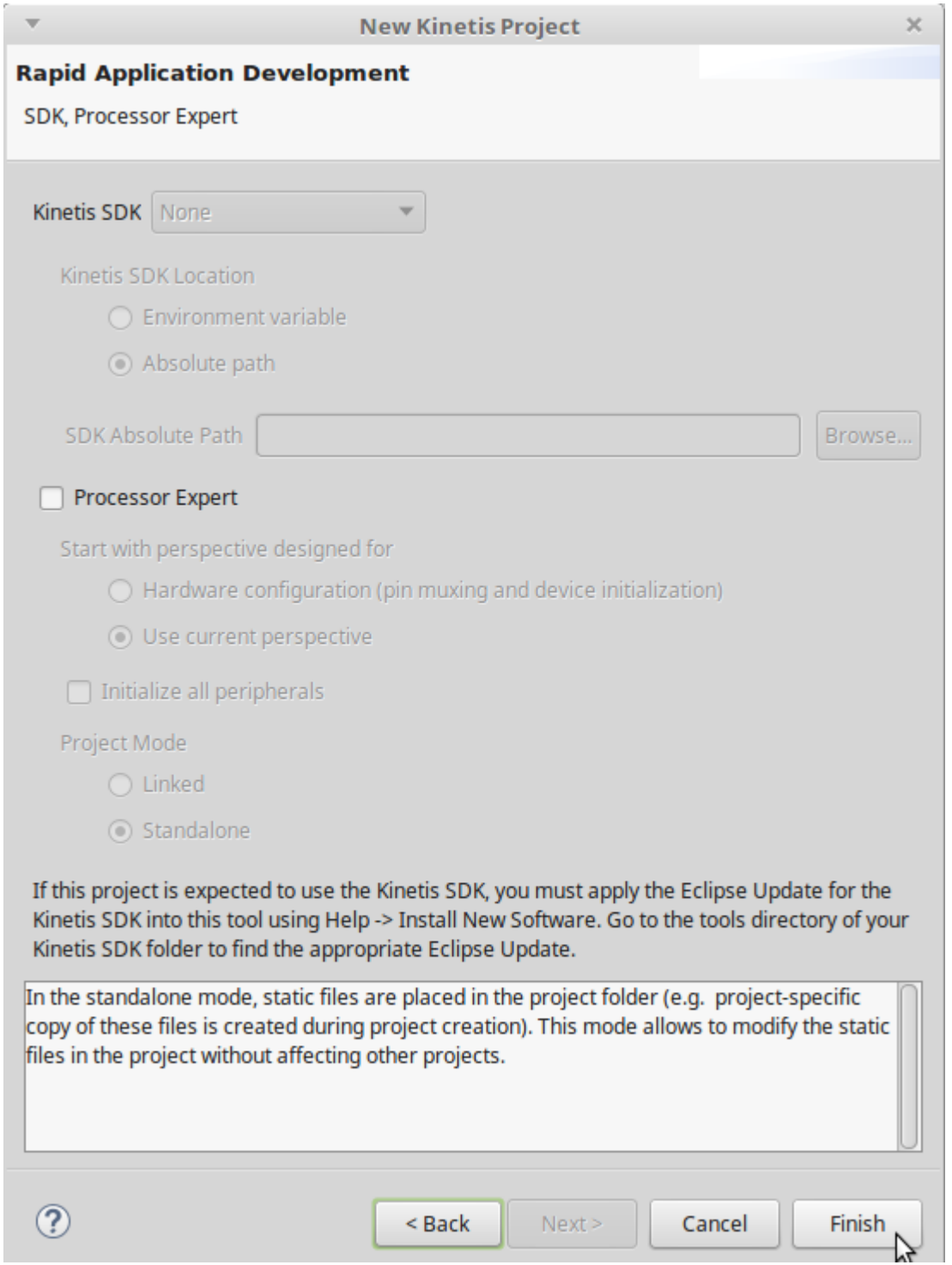
Vytvoření projektu

- Dále je potřeba vybrat MCU. V zobrazeném okně zvolte **Processors → Kinetis K → MK60 → MK60D (100MHz) → MK60DN512xxx10**
- Označte uvedený model a stiskněte tlačítko **Next >**



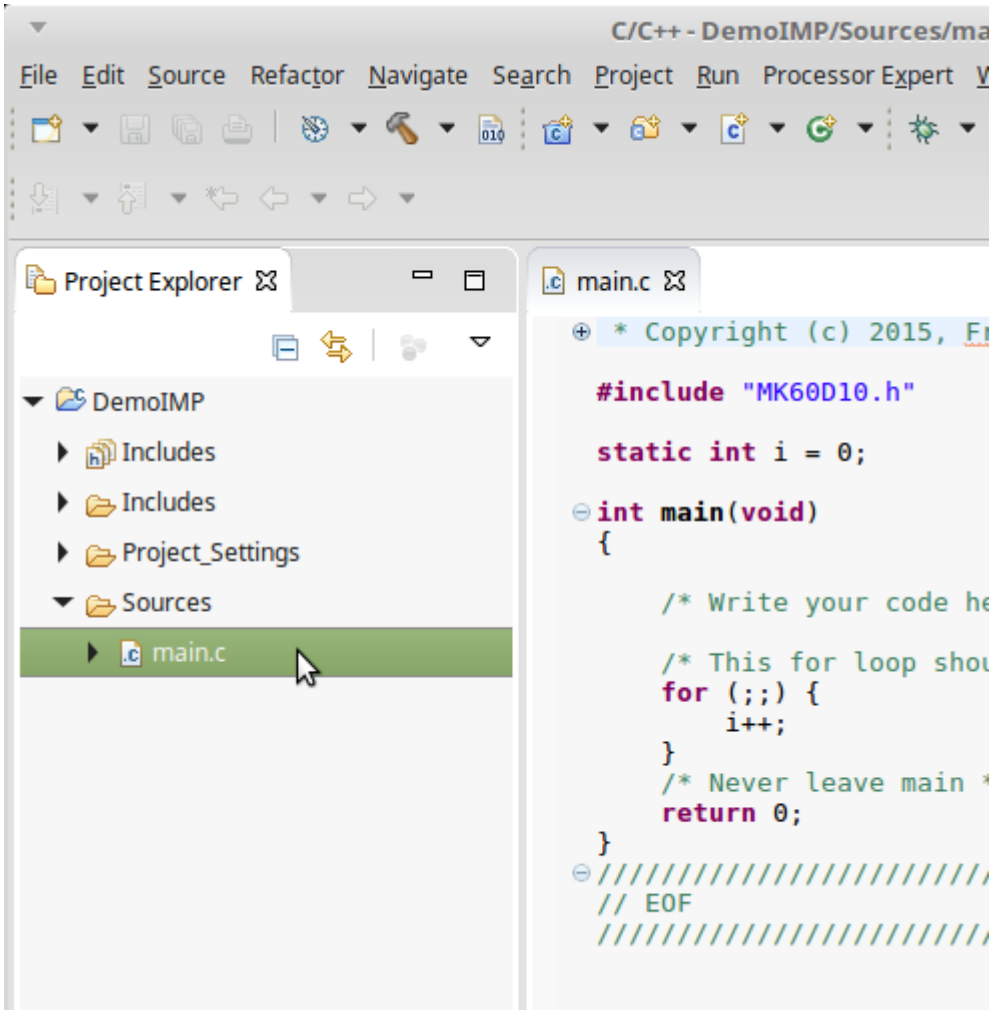
Vytvoření projektu

- Tím je nastavení projektu dokončeno, v dalším okně ponechte implicitní volby (viz okno vpravo) a stiskněte tlačítko **Finish.**



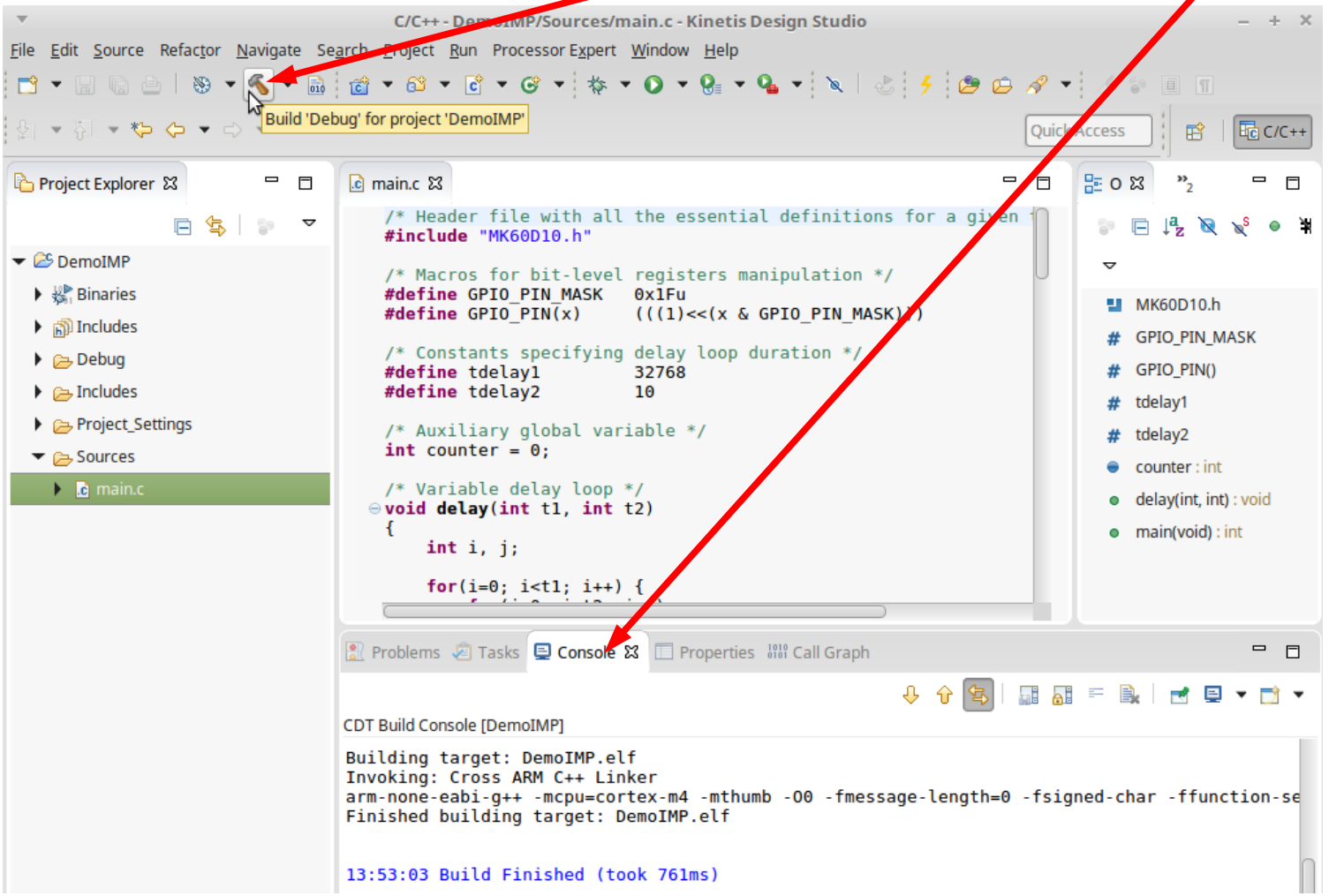
Otevření src

- Po návratu do hlavního IDE byste měli vlevo na kartě Project Explorer vidět položku s názvem vašeho projektu. Rozbalením stromu projektu poklepejte v sekci **Sources** na main.c, otevře se zdrojový kód projektu.
- Tento soubor kompletně nahraďte vzorovým programem (ctrl-c, ctrl-v) přiloženým k této prezentaci.



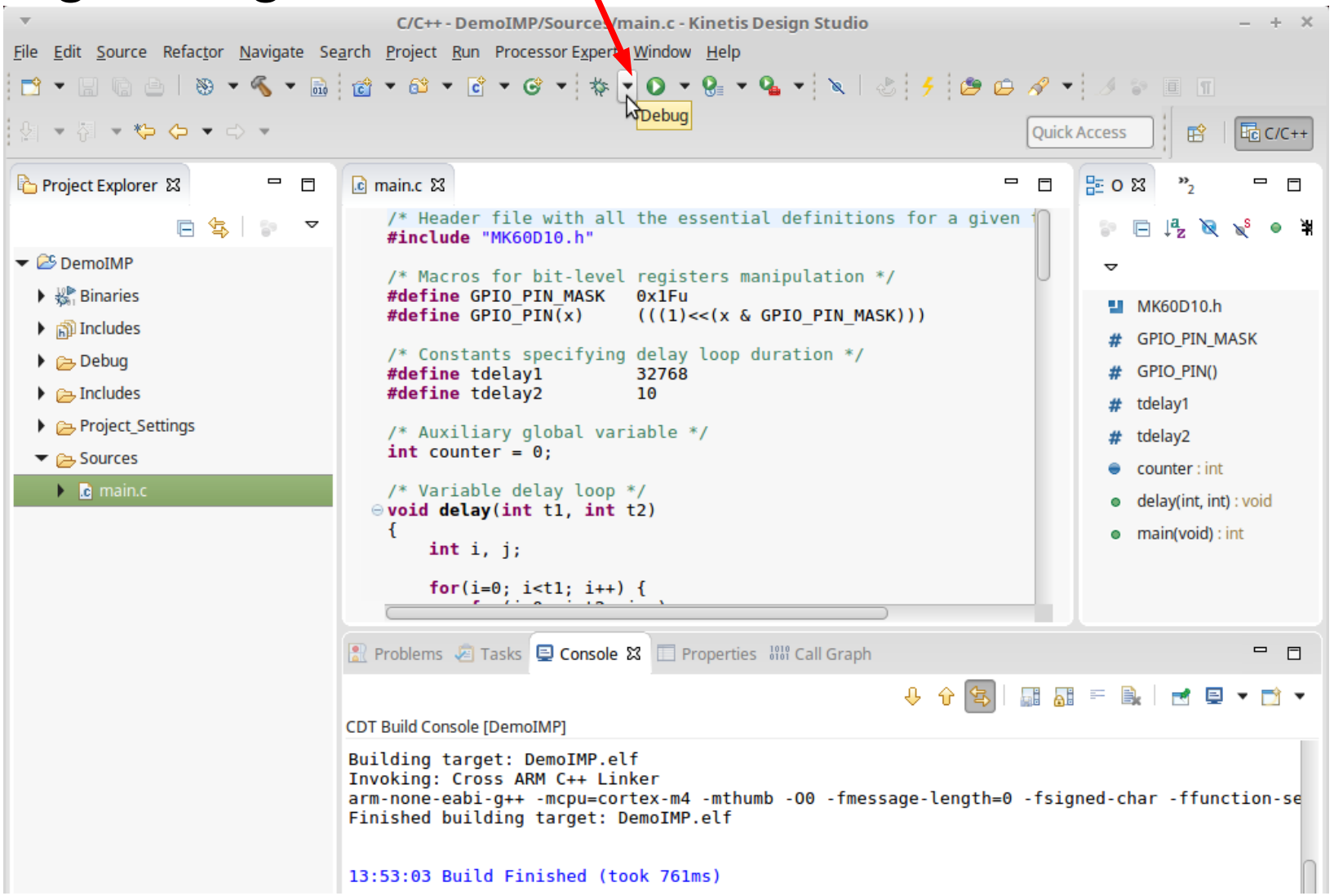
Sestavení projektu ze zdrojového kódu

- Kompilace a sestavení je možná buď z menu **Project → Build Project** nebo pohoblněji z nástrojové lišty (ctrl-B).
- Po překladu zkontrolujte, zda vše proběhlo v pořádku.



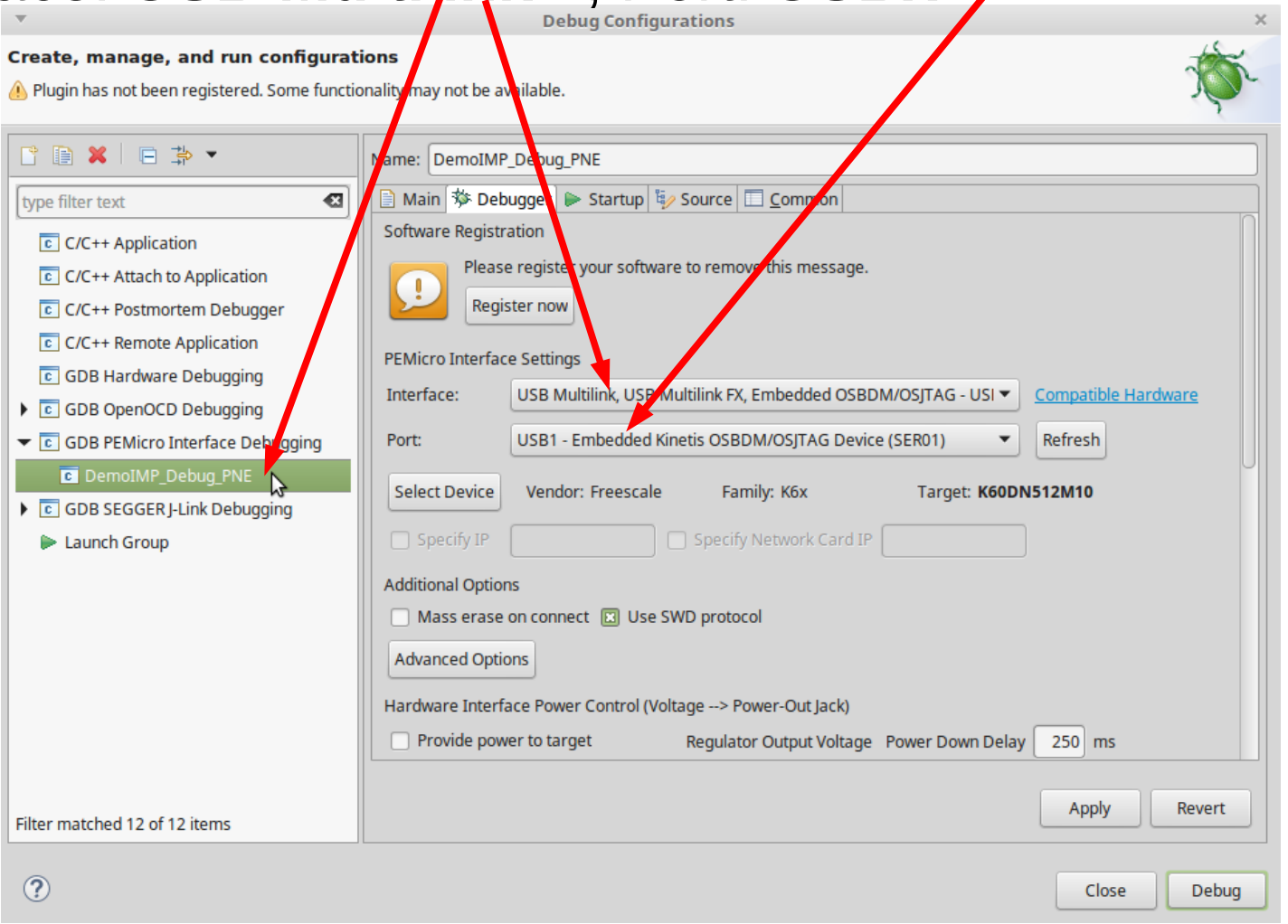
Nastavení konfigurace pro ladění projektu

- Připojte kit k PC pomocí USB kabelu typu A-B.
- Následně je nutné u ikony „brouka“ Debug z podmenu zvolit **Debug Configurations...**



Nastavení konfigurace pro ladění projektu

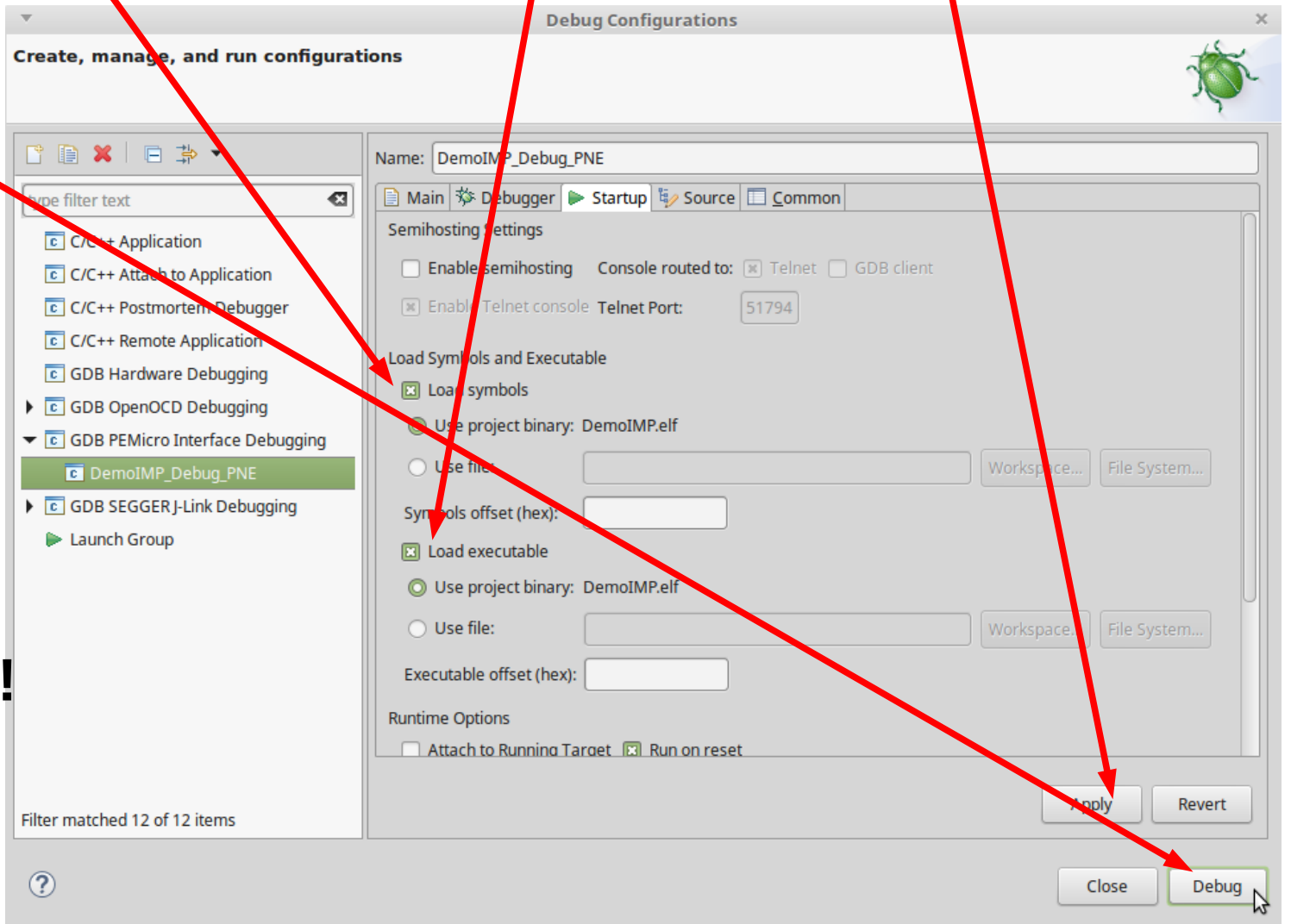
- Ze zobrazeného okna vlevo rozbalte položku **GDB PEMicro Interface Debugging** a zvolte **DemoIMP_Debug_PNE.**
- Na kartě Debugger zkontrolujte, zda systém připojený kit rozpoznal. Interface: **USB Multilink...**, Port: **USBx – Embedded...**
- Ostatní volby ponechte na implicitním nastavení.
- Po tomto nastavení je další volání debuggeru možné přes F11 a volbou **_Debug_PNE.**



Nastavení konfigurace pro ladění projektu

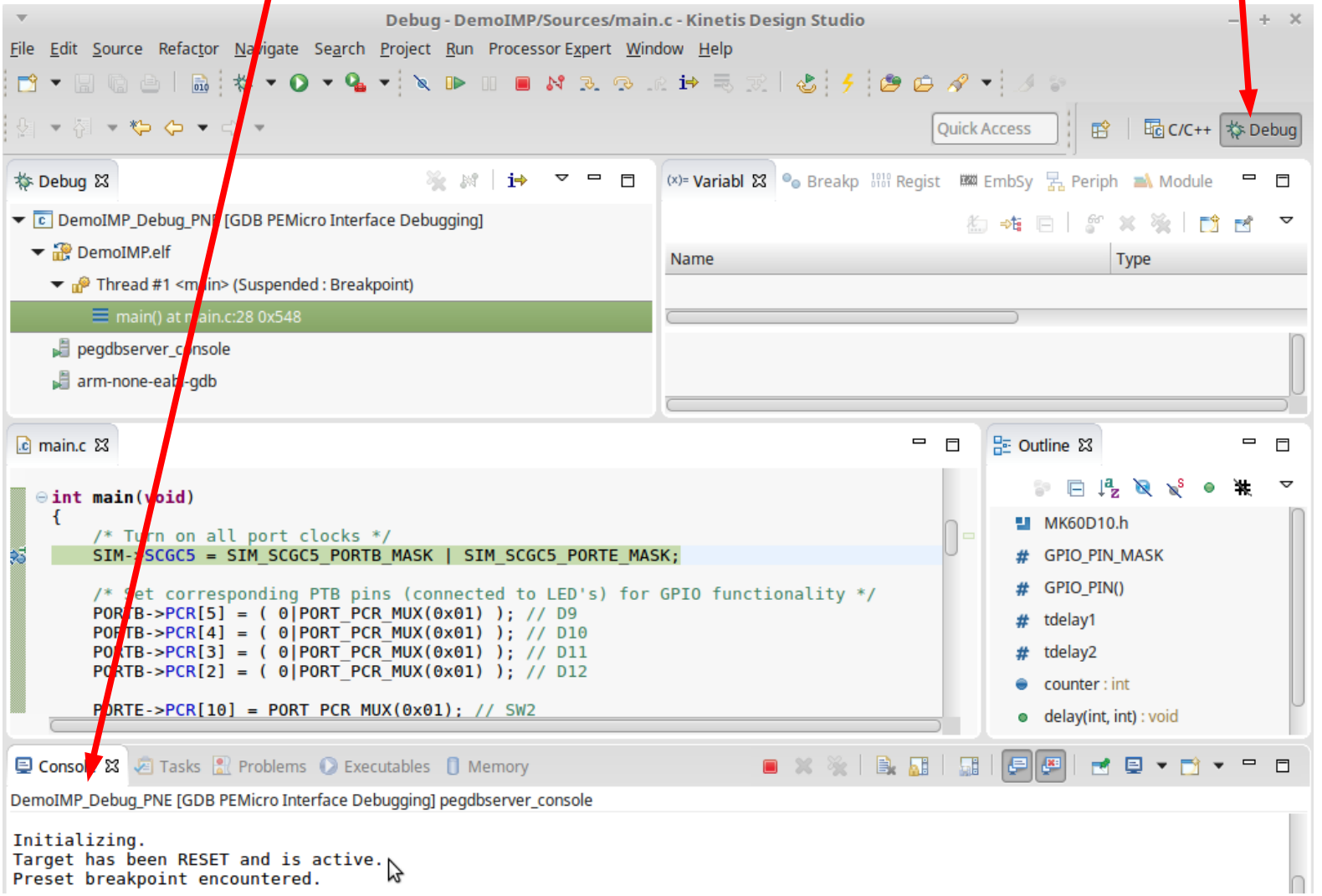
- Na kartě Startup zkontrolujte, zda jsou zaškrtnuta nastavení **Load symbols** a **Load executable**, stiskněte Apply**.**
- Stiskněte Debug, na případný dotaz ohledně debug perspective zvolte Yes.

  Reakce bývá delší, **proto nezmatkujte!**



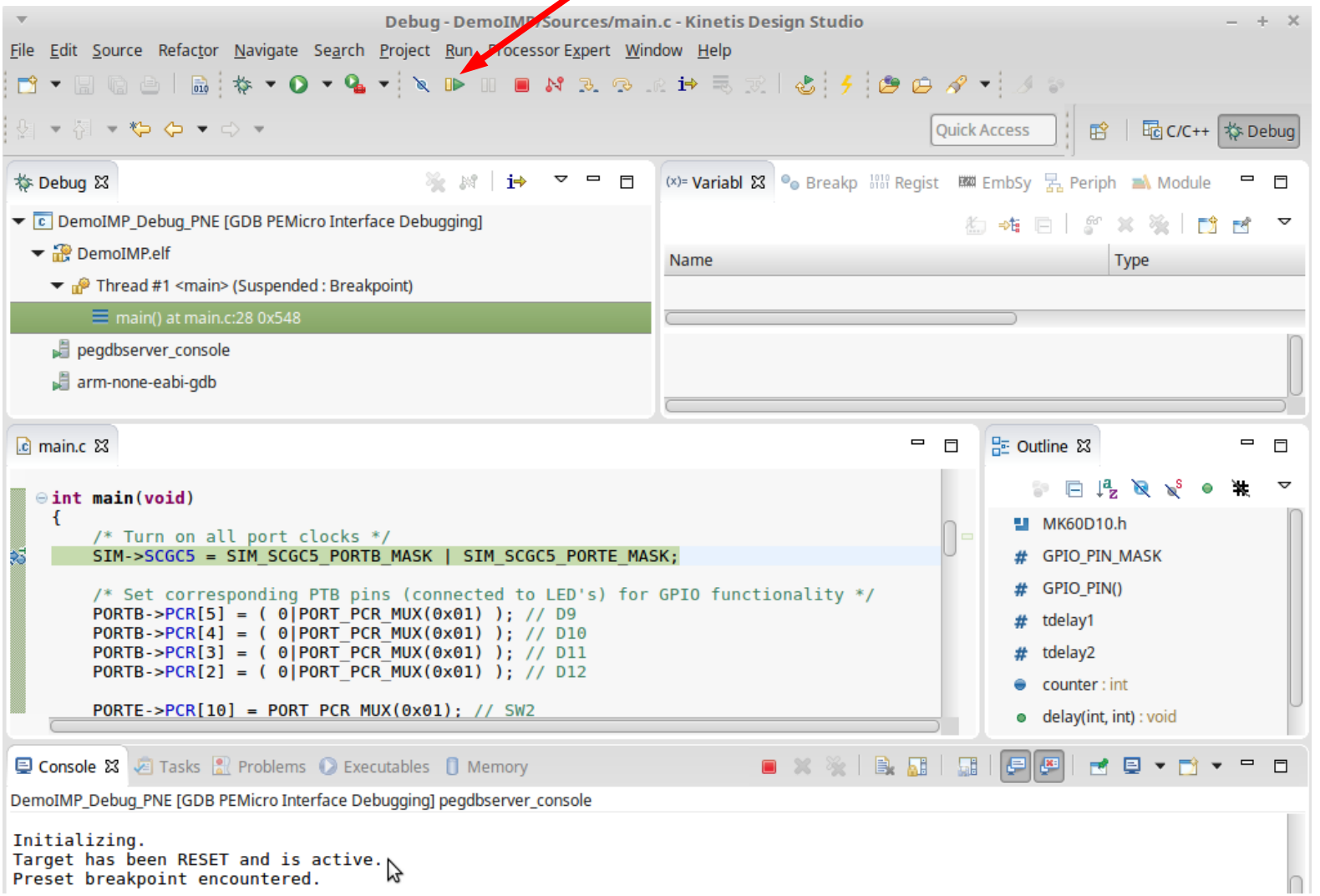
Naprogramování MCU

- Po proběhnutí flashování do MCU byste se měli dostat do režimu Debug. Zde zkontrolujte, zda naprogramování proběhlo úspěšně (výpis na konzoli).



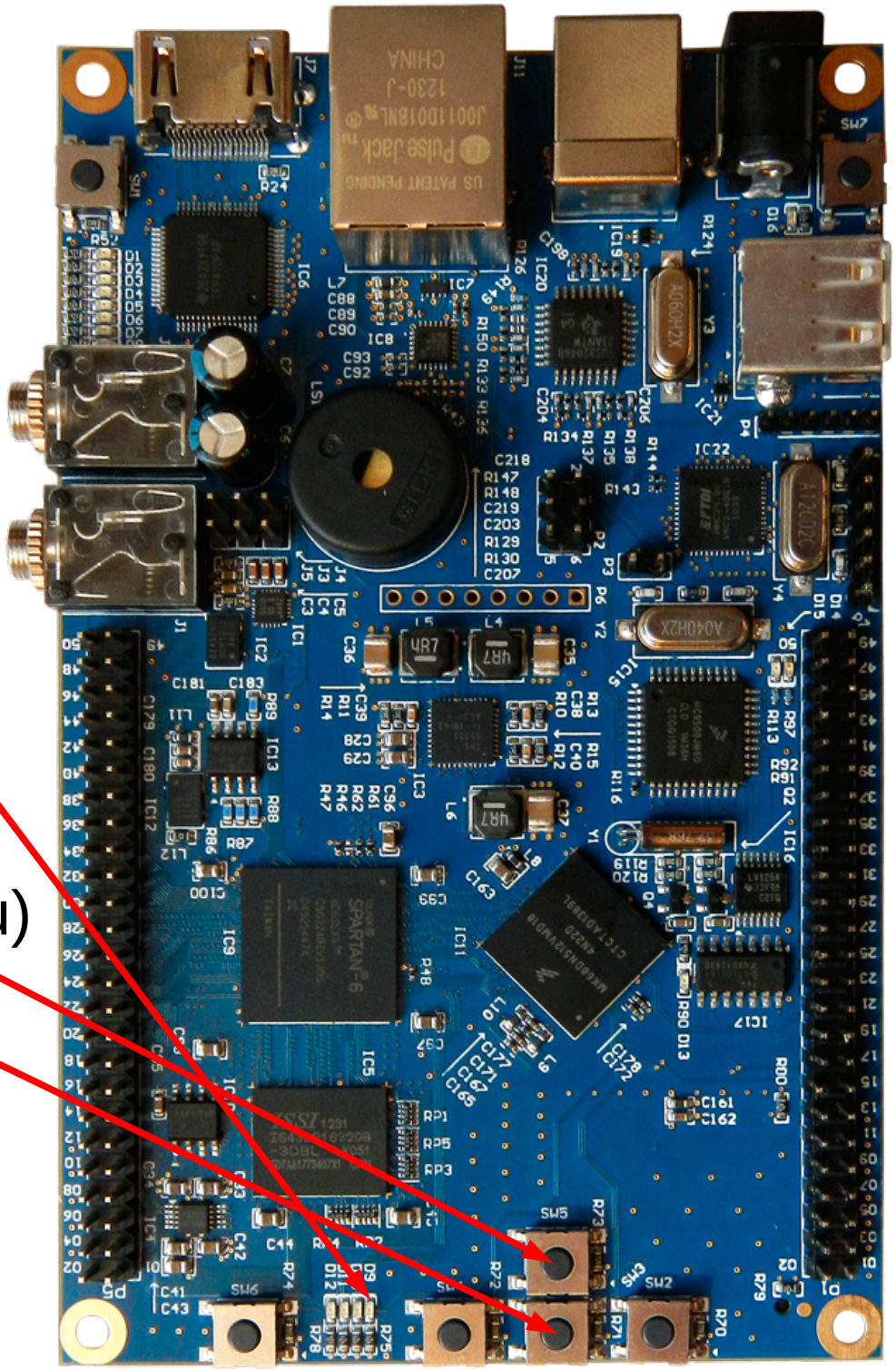
Spuštění aplikace na MCU

- Oznámení na konzoli „Target has been RESET and is active.“ značí úspěšné spojení debuggeru s kitem, nyní je možné aplikaci spustit stiskem Resume nebo klávesou F8
- Od této chvíle app již běží v MCU.



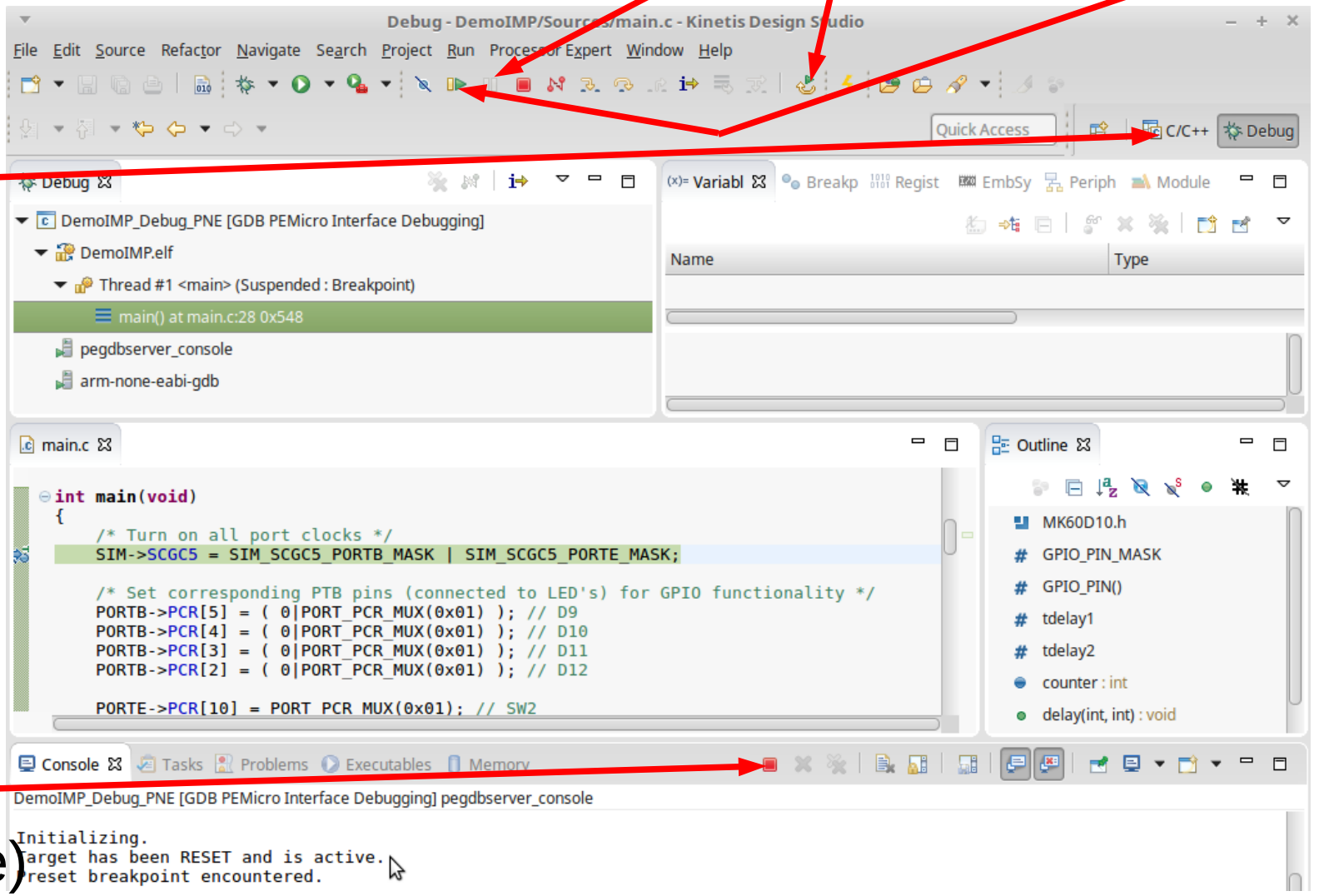
Kontrola úspěšného běhu aplikace

- Vzorová demo aplikace využívá časovače LPTMR0 (Low-Power Timer) generujícího přerušení pro blikání LEDkou D9, přičemž frekvenci blikání lze měnit tlačítky SW5 (nahoru) a SW3 (dolů).



Ukončení a znovuspuštění aplikace v MCU

- Pozastavení běhu aplikace provedete stiskem Suspend.
- Znovuspuštění naprogramovaného MCU **bez nutnosti opětovného flashování** je možné přes **Restart...** a poté opět F8.
- Návrat do zdrojáku stiskem
- Před opětovným flashováním nutno původní debug-sekvenci zastavit (Terminate)



Poznámky k obsluze aplikace v MCU

- Po správném naprogramování MCU je kit provozuschopný i bez opětovného flashování, stačí připojit napájení či přes USB k počítači a aplikace v kitu se ihned aktivuje
- Občas se může stát (jako u každého složitého SW), že se KDS začně chovat podivně, obvykle pomůže uložení rozpracované činnosti, restart KDS, příp. přepojení kitu.
- Ačkoliv FITkit3 obsahuje též FPGA (Xilinx Spartan-6), v rozsahu programování ARM MCU je plně obsluhovatelný pomocí KDS bez nutnosti instalace SW Xilinx pro práci s VHDL / FPGA.
- Práce s FPGA je zatím nad rámec možností IMP.

Struktury modulů MCU, registry, makra, příznakové masky

Vše je definováno v hlavičkových souborech v sekci Include hlavního adresáře projektu (zejména soubor MK60D10.h).

1. varianta: odkaz na registr přes strukturu pomocí ukazatele, např:

```
/* Turn on all port clocks */
SIM->SCGC5 = SIM_SCGC5_PORTB_MASK | SIM_SCGC5_PORTE_MASK;
```

2. varianta: přímá specifikace názvu registru (symbolické makro):

```
SIM_SCGC5 |= SIM_SCGC5_LPTIMER_MASK;

LPTMR0_CSR &= ~LPTMR_CSR_TEN_MASK;
```

Příklad vícebitové masky - ekviv. zápis:

```
PORTB->PCR[5] = PORT_PCR_MUX(0x01); // D9
GPIOB_PCR5 = PORT_PCR_MUX(0x01);
```

```
/** SIM - Register Layout Typedef */
typedef struct {
  __IO uint32_t SOPT1;
  __IO uint32_t SOPT1CFG;
       uint8_t RESERVED_0[4092];
  __IO uint32_t SOPT2;
       uint8_t RESERVED_1[4];
  __IO uint32_t SOPT4;
  __IO uint32_t SOPT5;
       uint8_t RESERVED_2[4];
  __IO uint32_t SOPT7;
       uint8_t RESERVED_3[8];
  __I  uint32_t SDID;
  __IO uint32_t SCGC1;
  __IO uint32_t SCGC2;
  __IO uint32_t SCGC3;
  __IO uint32_t SCGC4;
  __IO uint32_t SCGC5;
  __IO uint32_t SCGC6;
  __IO uint32_t SCGC7;
  __IO uint32_t CLKDIV1;
  __IO uint32_t CLKDIV2;
  __IO uint32_t FCFG1;
  __I  uint32_t FCFG2;
  __I  uint32_t UIDH;
  __I  uint32_t UIDMH;
  __I  uint32_t UIDML;
  __I  uint32_t UIDL;
} SIM_Type, *SIM_MemMapPtr;
```

Praktické aspekty programování obsluhy přerušení (příklad s časovačem)

- Pro správnou obsluhu pčerušení je nutno:

1. Povolit přerušení na úrovni MCU aktivací příslušného čísla žádosti – IRQ (specifické pro každou periferii). z MK60D10.h

```
NVIC_EnableIRQ(LPTMR0_IRQn);
```

2. Aktivovat přerušení v řídicím registru dané periferie – událostí vyvolávajících přerušení od jedné periferie může být i více.

Timer Compare Flag

Low Power Timer Control Status Register (LPTMRx_CSR)

7	6	5	4	3	2	1	0
TCF	TIE	TPS		TPP	TFC	TMS	TEN
w1c							
0	0	0	0	0	0	0	0

3. Definovat podprogram obsluhy přerušení. Název podprogramu viz **Project_Settings/Startup_Code/startup_MK60D10.S.** **(Tím je definován příslušný vektor přerušení!!!!)**

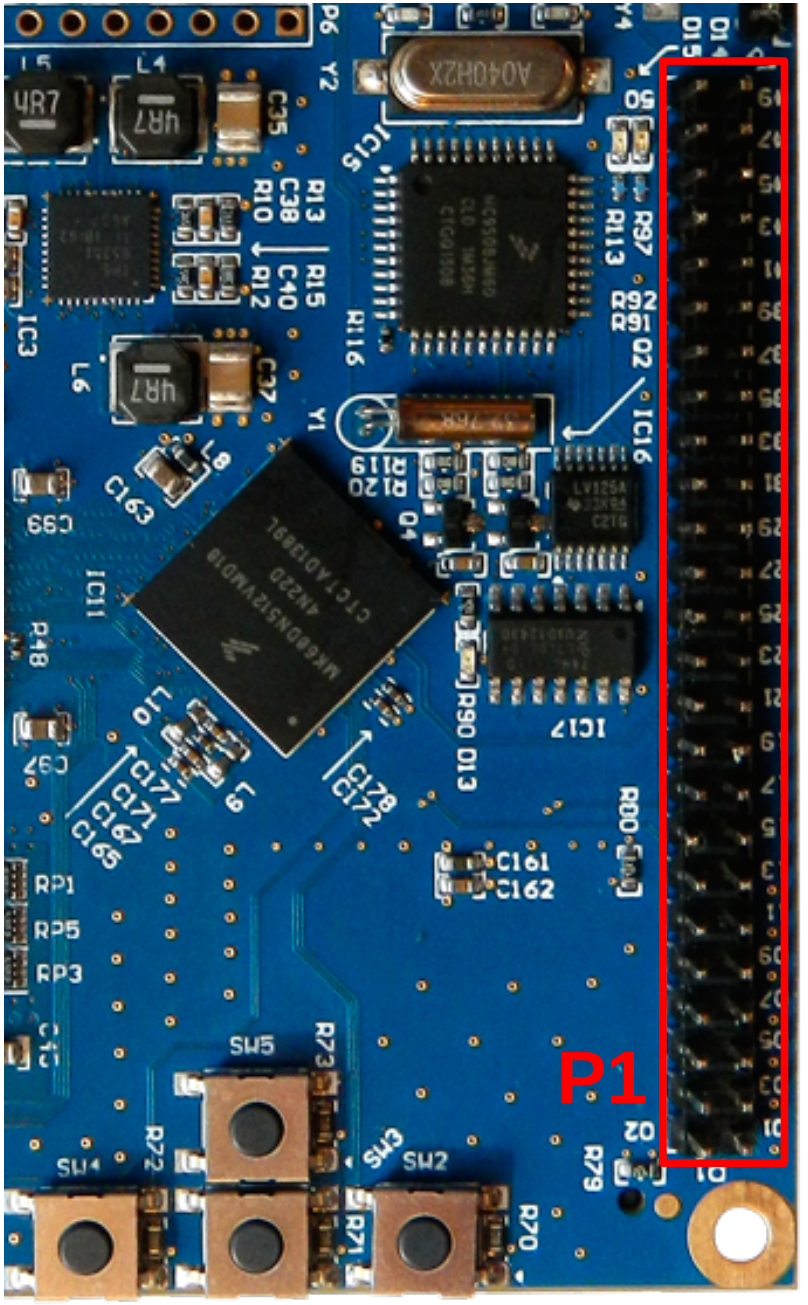
```
void LPTMR0_IRQHandler(void)
{
    LPTMR0_CMR = compare;
    LPTMR0_CSR |=  LPTMR_CSR_TCF_MASK;
    GPIOB_PDOR ^= LED_D9;
}
```

Použití externích portů MCU na FITkitu3

- Ačkoliv samotný MCU nabízí několik desítek GPIO vývodů (pinové pole P1), **univerzální použití umožňuje na FITkitu3 pouze 19 vývodů –** vizte schéma kitu, str. 6 a 8: vodiče popsané PTAx, PTEx (celkem 13), dále pak 6 vývodů:

  MCU_I2C0_SCL,
  MCU_I2C0_SDA,
  MCU_SPI2_CLK,
  MCU_SPI2_CS,
  MCU_SPI2_MOSI,
  MCU_SPI2_MISO.

- Na desce jsou to piny v poli P1 č. 17-28 a 34-40.



Usnadnění práce s KDS - přehled

- KDS využívá pro překlad a ladění nástroje GNU (gcc/g++, debugger gdb atd.), které je možno používat i mimo IDE KDS.
- Využijeme připravené skripty, které jsou k dispozici u vzorové aplikace:
  - *x00-make.sh*
  - *x01-flash.sh*
  - *x02-run.sh*
- Jejich přizpůsobení je možné pro libovolný OS podporovaný ze strany KDS. Zde využijeme např. OS Linux.
- Nástroje používané ve skriptech (vše je součástí KDS):
  - make
  - gdb server (pegdbserver_console)
  - debugger gdb (arm-none-eabi-gdb)

- **Dále ukážeme způsob, jak zefektivnit činnnosti při programování MCU, abychom nemuseli pokaždé spouštět poměrně náročné IDE.**

Usnadnění práce s KDS - konfigurace

- **Nejprve je třeba nastavit cestu k výše uvedeným nástrojům KDS v systémové proměnné PATH.**
  Např. v Linuxu s KDS 3.0 jsou to cesty:

  /opt/Freescale/KDS_3.0.0/toolchain/bin (pro nástroje v make a gdb)

  /opt/Freescale/KDS_3.0.0/eclipse/plugins/com.pemicro.debug.gdbjtag.pne_2.0.8.201504092111/lin
  (pro pegdbserver_console)

- Postupem uvedeným dříve vytvořte projekt v IDE KDS (pojmenujte jej např. FITkit3-demo).
  - Adresář s projektem si můžete umístit dle potřeby,
  - **název adresáře s projektem však musí být shodný s názvem projektu zadaným v KDS**!!
    (Implicitně to tak je, takže název neměňte.)
- Skripty (*x00-make.sh, x01-flash.sh, x02-run.sh*) zkopírujte do podadresáře Sources v projektu.

Usnadnění práce s KDS – konfigurace

- Ve skriptu *x01-flash.sh* nastavte název elf souboru (**stejný jako název projektu**) u příkazu:

  -ex 'load ../Debug/FITkit3-demo.elf'

- Na prvním řádku skriptů *x01-flash.sh* a *x02-run.sh* nastavte model MCU (pro FITkit3: Freescale_K6x_K60DN512M10):

  pegdbserver_console -startserver -device=Freescale_K6x_K60DN512M10 &

  Seznam podporovaných MCU lze získat spuštěním:

  pegdbserver_console -devicelist

Princip skriptů *flash* a *run*: nejprve se spustí debug-server na pozadí, dalším příkazem bude spuštěn arm-debugger, který se přes debug-server připojí k MCU a podle zadaných příkazů debuggeru (-ex) provede naprogramování MCU binárkou .elf, případně aktivaci aplikace v MCU (podobně jako stisk F8/Resume v KDS).

Usnadnění práce s KDS – použití skriptů

- Po provedení konfigurace (jednorázově pro daný projekt) je možné provádět překlad a ladění aplikace v MCU pohodlně z příkazového řádku nez nutnosti spouštět IDE KDS:
  - vstupte do podadresáře Sources v projektu,
  - proveďte překlad spuštěním skriptu *x00-make.sh*,
  - naprogramujte MCU spuštěním skriptu *x01-flash.sh*,
  - spusťte skript *x02-run.sh*, který otevře konzoli gdb a aktivuje aplikaci v MCU,
  - CTRL-C ukončí běh aplikace (_reset) i běh gdb (quit).
- Pozn: konzole gdb je plnohodnotné ladicí prostředí, které podporuje množství dalších příkazů (vizte dokumentaci).