

# **KAE/MINA**

## **Domácí příprava 2016/17**

Ing. Petr Weissar, Ph.D.

Ing. Petr Krist, Ph.D.

Ing. Kamil Kosturik, Ph.D.

# Program domácí přípravy

- I. Instalace Keil MDK, příp. Atollic IDE, ST-Link, analýza demo-programu, použití debuggeru**
- II. Bitové operace, maskování, GPIO
- III. Funkce z knihovny stdio – procvičení
- IV. Časování, nastavení RCC bloku, rozdělení hodinových signálů perifériím
- V. Časovač, využití přerušení
- VI. Procvičení práce s LCD, doplnění functionality
- VII. Zpracování dat ze senzorů (A/D, externí)
- VIII. DMA
- IX. Instalace a procvičení RTOS
- X. - XIII. Samostatná práce

Čísla "týdnů" odpovídají číslování "po" daném cvičení, tedy I. je plánována mezi cv. 1 a 2

# Program cvičení

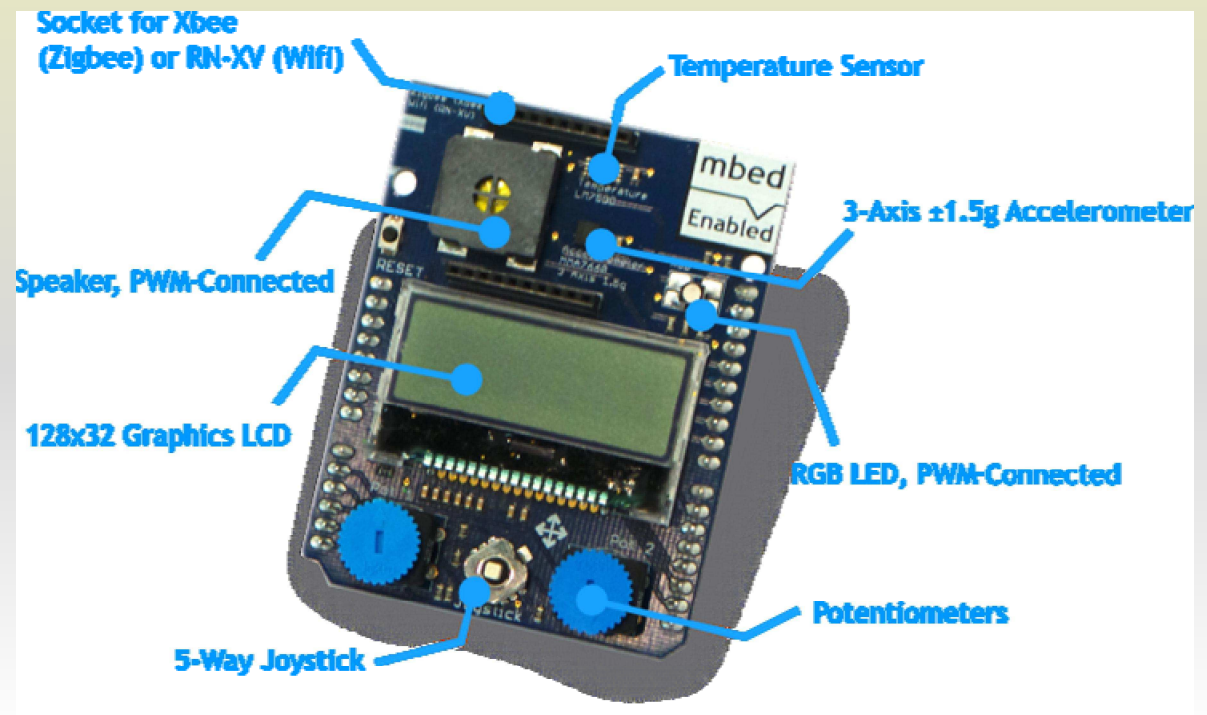
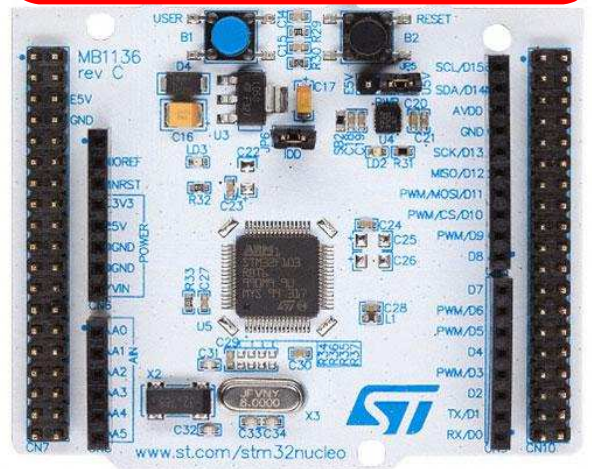
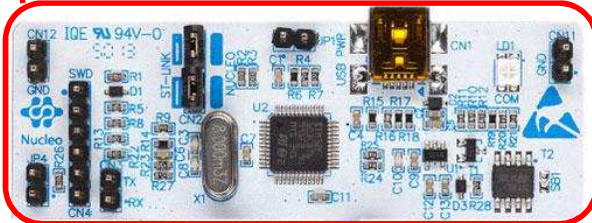
1. Úvod, rozdělení kitů, prostředí Keil, nahrání programu
2. Debug, bitové operace, GPIO
3. UART, připojení k PC, využití stdio knihovny
4. Časování, hodiny, SysTick
5. Časovač, přerušení, NVIC
6. SPI a připojení LCD
7. A/D převodník, připojení přes IIC
8. RTOS
9. Samostatná práce
10. Dtto
11. Dtto
12. Dtto
13. Dokončení a odevzdání samostatné práce

# Zdroje informací – stejné jako cvičení

- Dostupné v elektronické podobě – typicky PDF:
  - Internet – [www.st.com](http://www.st.com), [mbed.org](http://mbed.org), ...
  - Vybrané na CourseWare
- Procesor - <http://www.st.com/web/en/catalog/mmc/SC1169/SS1031/LN1565/PF164487>
  - **Reference manual** - STM32F101\_2\_3\_5\_107xx advanced ARM-based 32-bit MCUs (CD00171190)
  - **Datasheet** - STM32F103xB CD00161566.pdf
  - **Programming manual** - STM32F10x Programming manual (CD00228163)
- Nucleo kit - <http://www.st.com/web/catalog/tools/FM116/CL1620/SC959/SS1532/LN1847/PF259875>
  - <https://developer.mbed.org/platforms/ST-Nucleo-F103RB/>
  - User manual - Nucleo UserManual (DM00105823)
  - Schematic - Nucleo Schematic (MB1136)
- MBED Shield
  - <https://developer.mbed.org/components/mbed-Application-Shield/>
  - Schematic - ApplicationShield.pdf
  - Komponenty
    - Akcelerometr - MMA7660FC.pdf
    - Temperature - LM75B.pdf
    - LCD - NHD-C12832A1Z-FSW-FBW-3V3.pdf

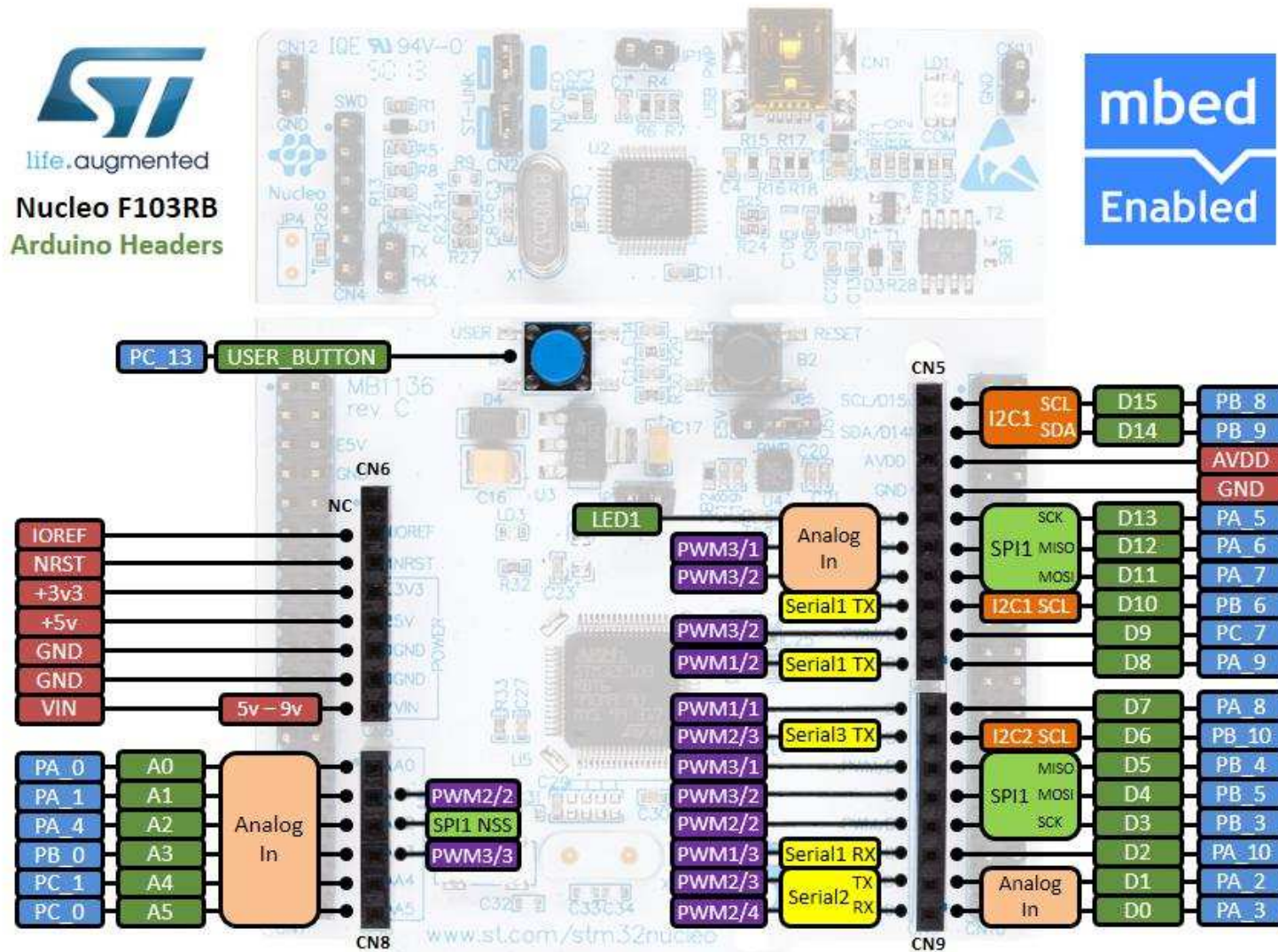
# Představení vývojového kitu

- STM32 Nucleo – osazené STM32F103RB
  - Vybrané signály na "arduino" konektorech
  - Všechny signály na "morpho" konektorech
  - Připojení USB - obsahuje ST-link
    - Implementuje 3 "device" – Debug-SWD, USB/UART, MassStorage (USB)
    - Ovladače přímo od ST
      - ST-Link, ST-Link/V2, ST-Link/V2-1 USB driver signed for XP, Windows7, Windows8
  - ke stažení např. <http://www.st.com/web/en/catalog/tools/PF260219#>
- MBED application shield – pro Arduino (3v3)

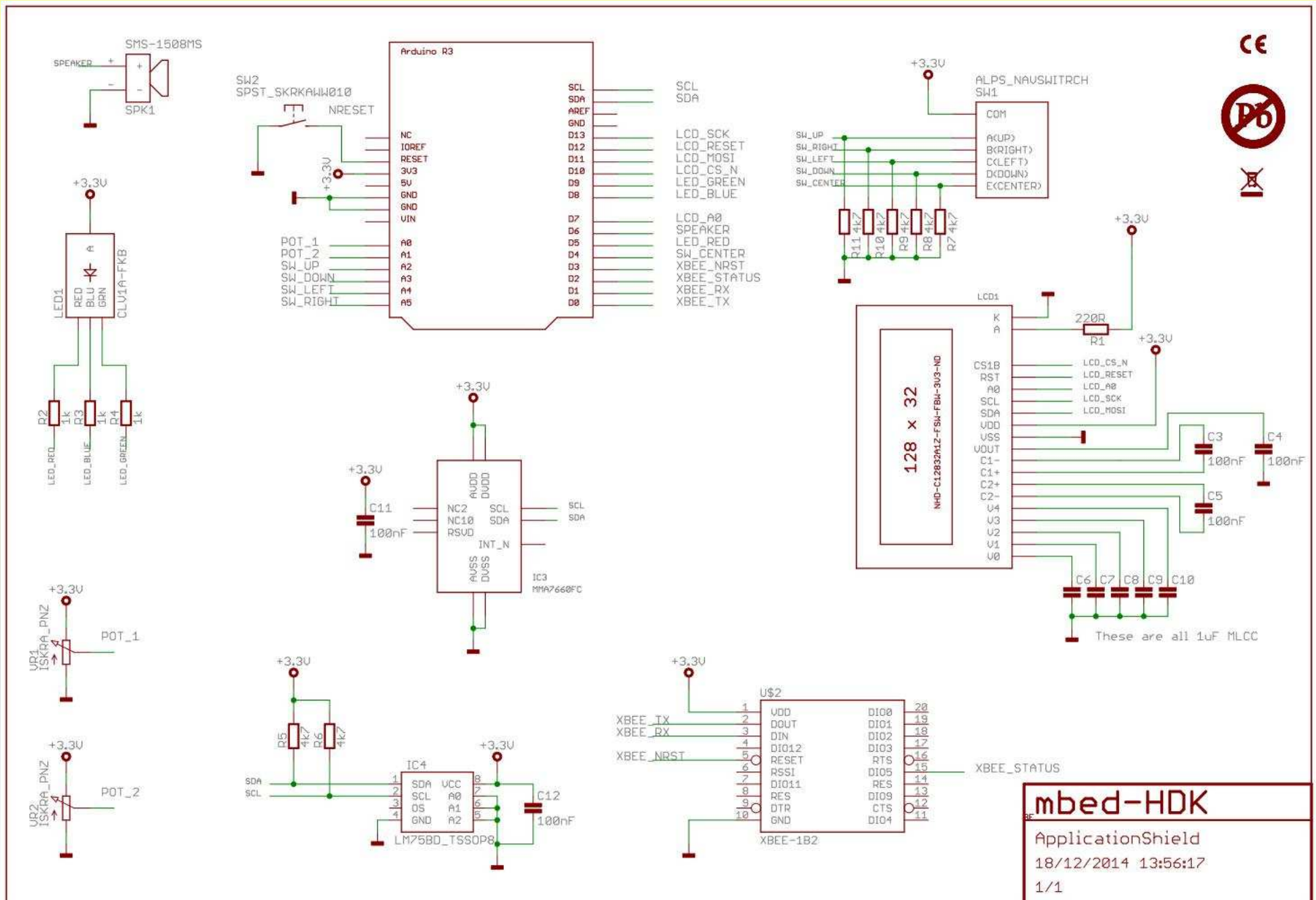




# Popis signálů Nucleo F103RB



# Schéma zapojení MBED ApplicationShield



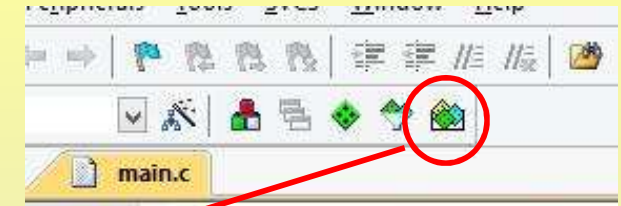
# Instalace Keil MDK 5 (verze 5.21a)

- Stáhnout instalační EXE (cca 300MB)
  - Z webu výrobce <http://www.keil.com/arm/mdk.asp>
    - Vyžaduje registraci (poměrně podrobnou)
  - DropBox - <https://dl.dropboxusercontent.com/u/22184251/MDK521a.EXE>
    - Instalace stylem Next, Next, ...
  - Při registraci vyžaduje formálně správný email
- Po prvním spuštění doporučuji nastavit prostředí do "použitelnějšího" stavu
  - Lepší font editoru pro C/C++ – např. Consolas nebo Lucida Console
  - Odsazování po 2 mezerách
  - Číslování řádků u všech souborů
  - Automatické nabízení symbolů
- Viz. video (ukázka starší verze 5.15, neliší se od používané 5.21a)
  - <https://www.youtube.com/watch?v=ncJUV81z72c> – instalace prostředí a balíčků
  - <https://www.youtube.com/watch?v=YKx17v0MKFc> – první spuštění





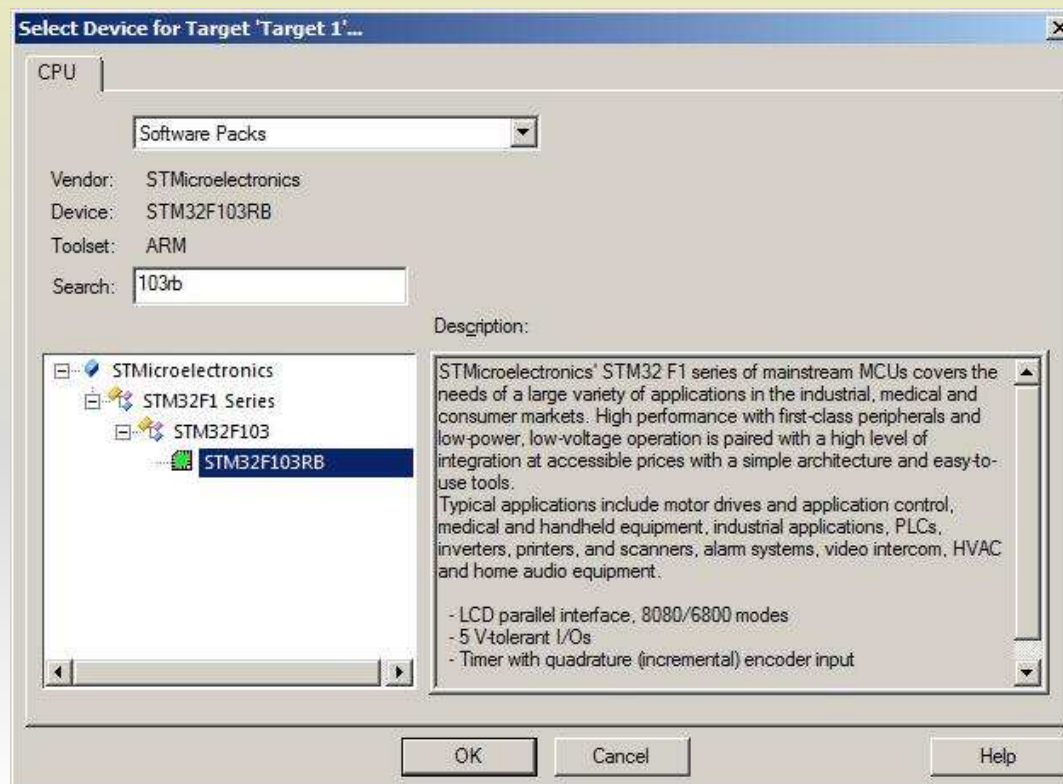
# Instalace modulů/knihoven



- Po prvním spuštění nutno nainstalovat moduly/knihovny
- Stačí vybrat "board" Nucleo 103RB a zvolit moduly:
  - **STM32F1xx\_DFP** – podpora pro řadu procesorů STM F1xx
  - **CMSIS** – knihovny pro ARM jádro
  - **ARM\_Compiler** – kompilátor+linker+utility
  - **STM32NUCLEO\_BSP** – přidává podporu pro Nucleo desky, není nezbytná
- Moduly se průběžně inovují, lze se podívat na nabídku "upgrade"
  - Projekty mají defaultně automatické využití novějších verzí
  - Příp. je možné cesty k verzi modulu nastavit ve vlastnostech projektu
- Je možné spustit více akcí, vykonávají se ve "frontě"

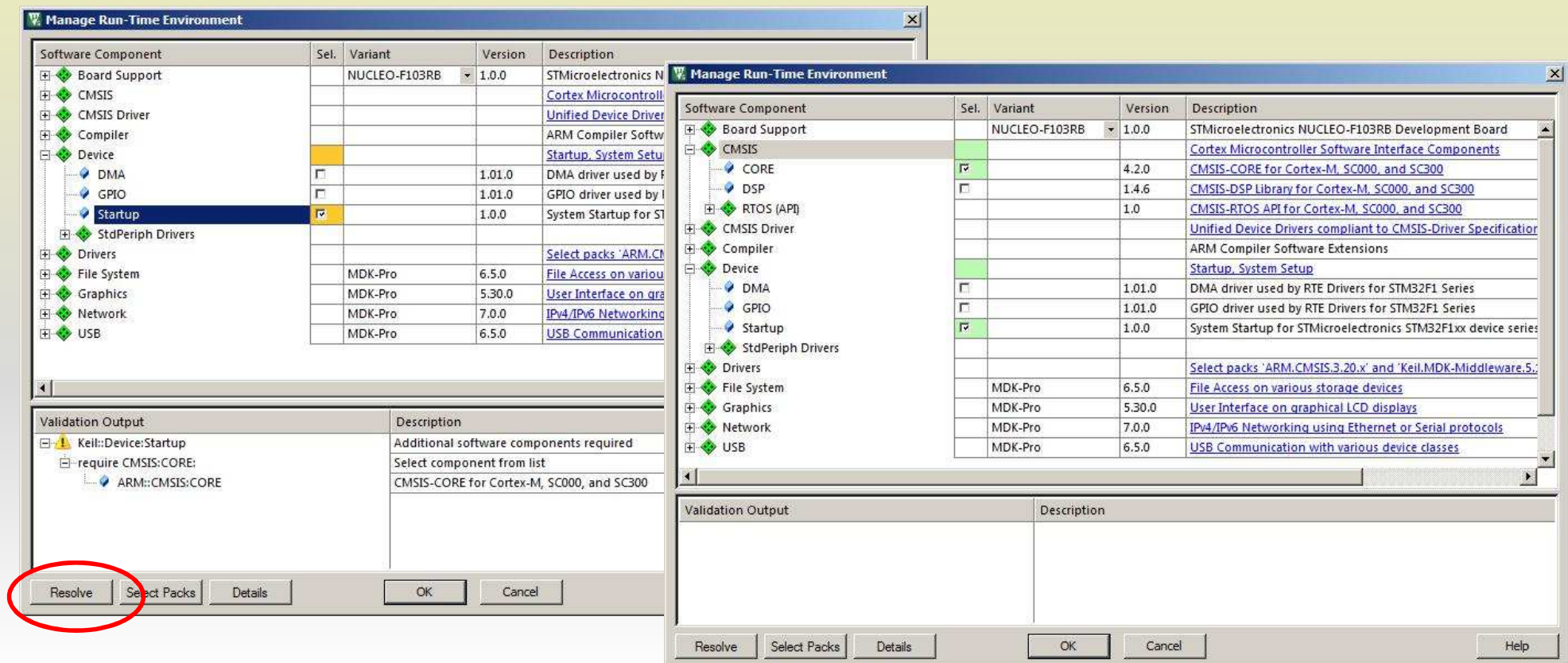
# Vytvoření nového projektu

- Každý projekt je typicky ve "svém" adresáři
- Menu Project – New uVision Project
  - Při vytváření zvolíme procesor "103rb"
    - nabízí se podle nainstalovaných balíčků

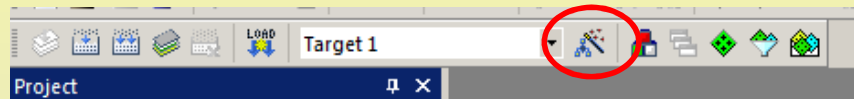


# Vytvoření nového projektu - II

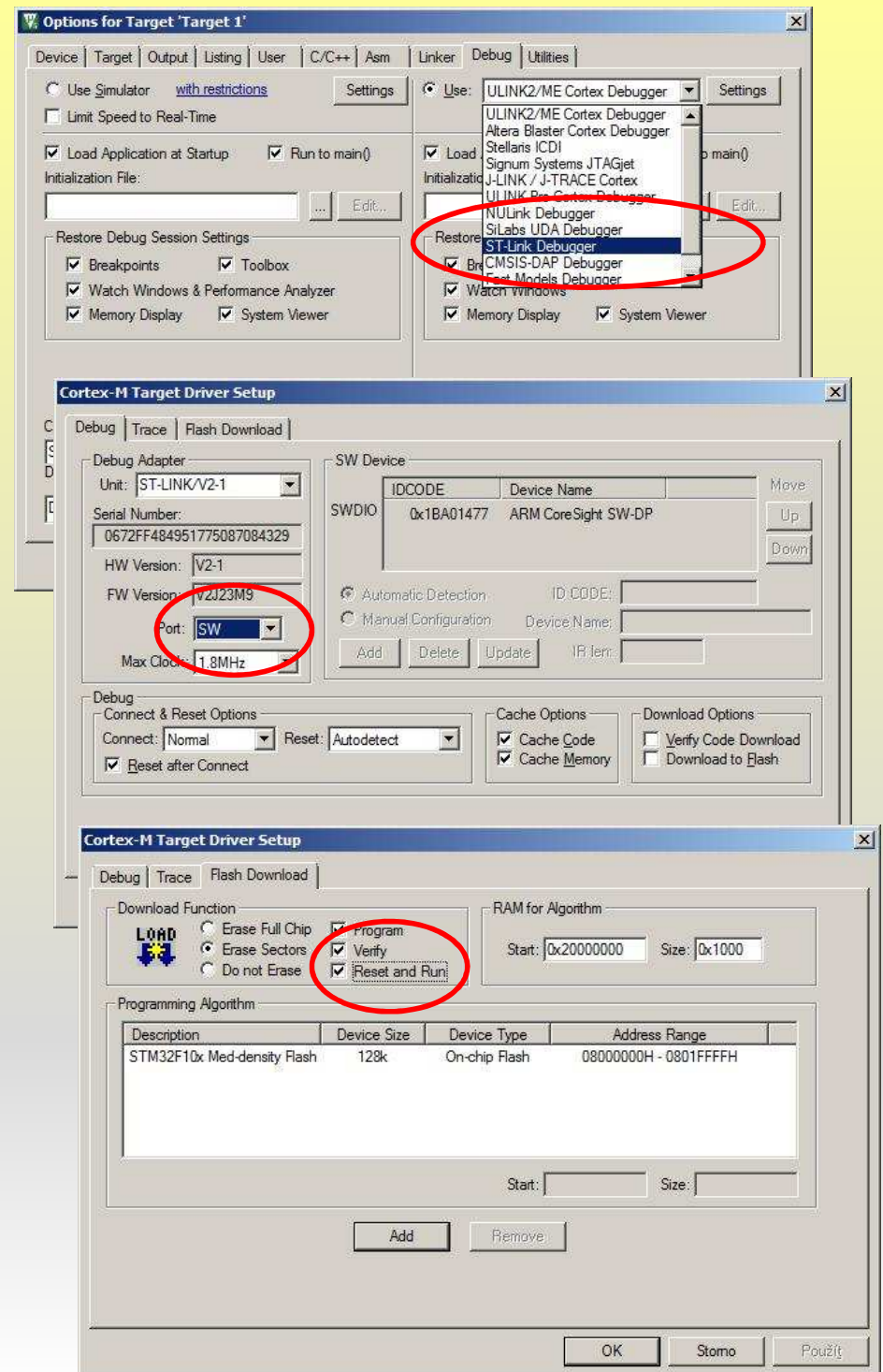
- Z nabídky modulů pro projekt stačí "Startup" v sekci Device
  - Vyžaduje doplnění CMSIS:Core – nechat vyřešit (Resolve)



# Vytvoření nového projektu - III

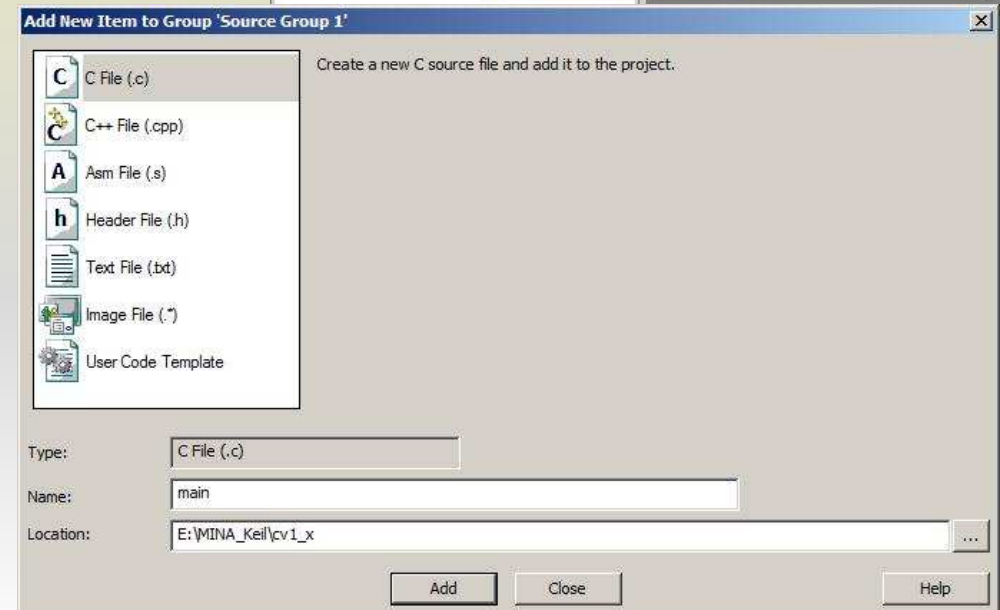
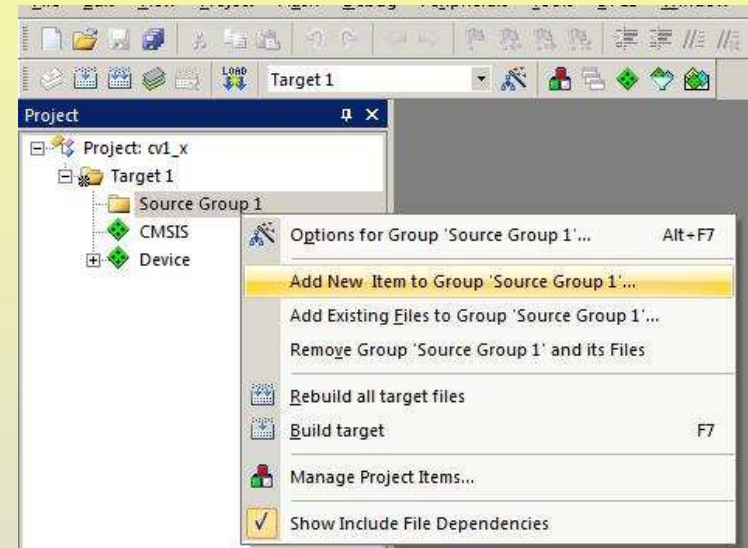


- Ve vlastnostech projektu je třeba upravit Debug
  - Vybrat ST-Link Debugger
  - Nastavit SWD režim
    - Rovnou se musí ozvat ST-Link = ukáže verzi/ID
  - Zapnout "Reset and Run" po nahrání do desky



# Vytvoření nového projektu - IV

- Přidat nový soubor s kódem
  - Do "Source Group 1"
  - Pravým myšítkem
  - Add New Item ...
- V dialogu zvolit "C File"
  - Vytvořit např. main.c





# Kód blikání LED na Nucleo desce

```
#include <stm32f10x.h> // v podstate jediny nutny include, obsahuje nazvy registru, bitu, ...

// LED na desce je pripojena na 5. bit IO brany A (= GPIOA5)

int main(void)
{
    int x;          // pomocna obecna promenna, zde pouzita jen pro dummy-pocitadlo

    if (!(RCC->APB2ENR & RCC_APB2ENR_IOPAEN)) // test zda je periferie povolena (= ma pripojene hodiny)
    {                                           // a kdyz neni, musi se povolit a vyresetovat
        RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;   // povolit hodiny pro GPIOA - je na sbernici APB2
        RCC->APB2RSTR |= RCC_APB2RSTR_IOPARST; // maskovanim nastavi RESET bit periferie do 1
        RCC->APB2RSTR &= ~RCC_APB2RSTR_IOPARST; // maskovanim nastavi RESET bit periferie do 0 = tj. RESET puls
    }

    GPIOA->CRL &= ~(0x0f << (4 * 5));          // kazdy IO bit je rizen 4-bity v registru CRL (pro nizsich 8 IO)
    // 1111 posunuta o 5 odpovida 0000 0000 1111 0000 0000 0000 0000 0000
    // bitovou negaci pak ziskame 1111 1111 0000 1111 1111 1111 1111 1111
    // a maskovanim se prislusne 4-bity vynuluji beze zmeny ostatnich
    GPIOA->CRL |= (0x03 << (4 * 5));           // hodnota 0011 pro 4 bity ridici GPIOA5
    // vystupni rezim, strmost hrany 50MHz, push-pull mod

    while(1)
    {
        GPIOA->ODR ^= (1 << 5);                // v datovem registru odpovida jednomu IO jeden bit
        // XOR-em menime 0 na 1 a naopak = zmena stavu

        for (x = 0; x < 1000000; x++)          // cekani prazdnym cyklovanim
        ;
    }
}
```

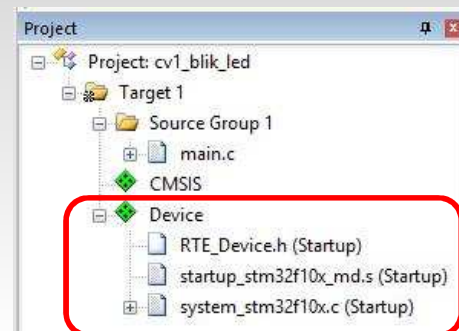
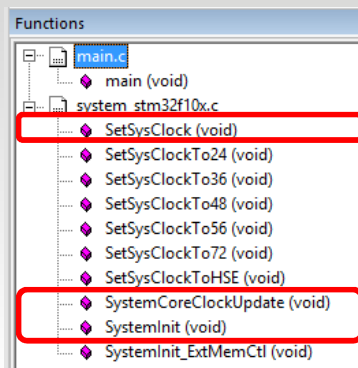
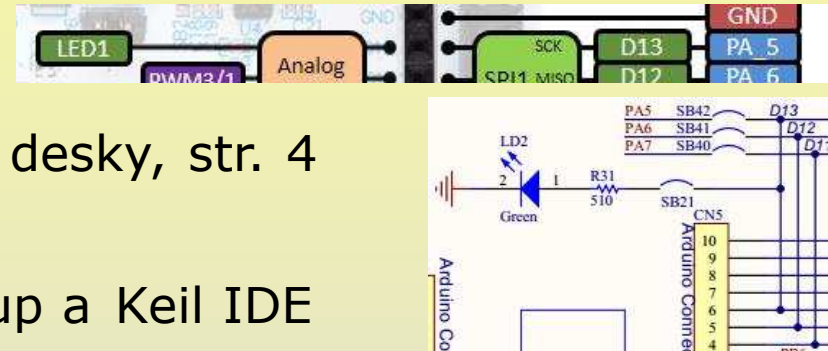
# Překlad a spuštění projektu

- Build (kompiluje změněné soubory a linkuje výsledek)
  - Ikona na liště
  - Menu: Project – Build Target
  - Klávesa F7
- Download (to HW)
  - Ikona na liště
  - Klávesa F8



# Analýza programu 1 – LED a startup

- LED zapojena z VCC
  - GPIOA 5 – viz. popis Nucleo desky
  - Svítí se log.1 – viz. schéma Nucleo desky, str. 4
- Počáteční inicializace procesoru
  - Při vytváření zvolen defaultní startup a Keil IDE přidal do projektu
    - Startup
      - ASM kód nastavující základ paměti a vektory přerušení
      - Spouští funkci SystemInit()
      - Následně "naše" hlavní main()
    - System\_stm32f10x.c
      - Základní knihovní funkce pro nastavení systému
      - Nastavuje se konstantami preprocesoru
      - Např. výběr funkce pro nastavení hodin



```
static void SetSysClock(void)
{
#ifdef SYSCLK_FREQ_HSE
    SetSysClockToHSE();
#elif defined SYSCLK_FREQ_24MHz
    SetSysClockTo24();
#elif defined SYSCLK_FREQ_36MHz
    SetSysClockTo36();
#elif defined SYSCLK_FREQ_48MHz
    SetSysClockTo48();
#elif defined SYSCLK_FREQ_56MHz
    SetSysClockTo56();
#elif defined SYSCLK_FREQ_72MHz
    SetSysClockTo72();
#endif

    /* If none of the define above is enabled,
    the HSI is used as System clock
    source (default after reset) */
}
```

# Analýza programu 1 – sběrnice, hodiny, inicializace periférie

- Jednotlivé části procesoru jsou připojeny na jednu ze sběrnic – viz. kap. 3.1 v RM
- Pro funkci periférie musí mít "povolené hodiny" z dané sběrnice
  - Řídící blok RCC, registr RCC\_APB2EN (pro sběrnici APB2)
  - Výhodně v .H souboru definováno jako "ukazatel na strukturu", která je na pevném místě paměti
    - = přístup přes "šipku"
  - Jednotlivé periférie mají odpovídající bit také definován
  - Povolení zápisem log. 1, nejlépe maskováním
- Správnou inicializaci periférie zajistí "reset-puls"
  - Opět registr v RCC, konkrétně RCC->APB2RSTR
  - Zápis log. 1 následován zápisem log. 0
    - Výhodně pomocí maskování

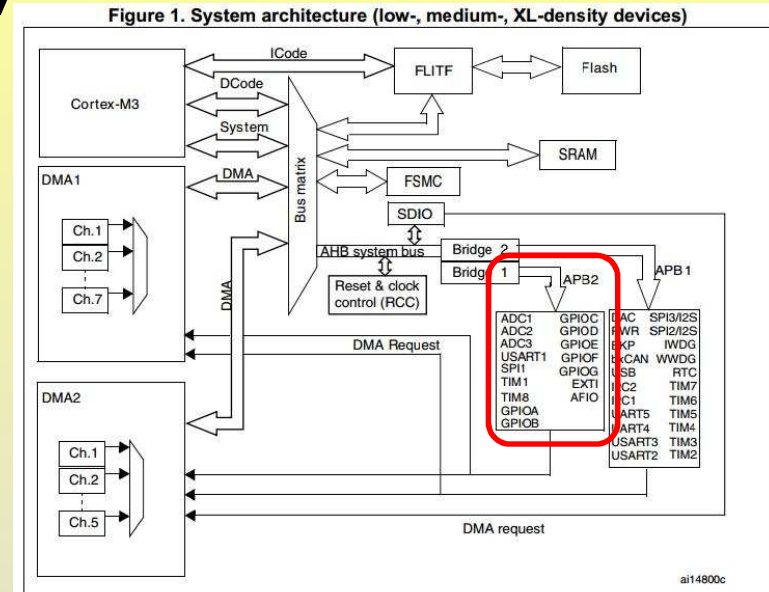
**7.3.7 APB2 peripheral clock enable register (RCC\_APB2ENR)**

Address: 0x18

Reset value: 0x0000 0000

Note: When the peripheral clock is not active, the peripheral register values may not be readable by software and the returned value is always 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										TIM11 EN	TIM10 EN	TIM9 EN	Reserved		
Reserved										rw	rw	rw	Reserved		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 EN	USART1 EN	TIM8 EN	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	IOPG EN	IOPF EN	IOPE EN	IOPD EN	IOPC EN	IOPA EN	Res.	AFIO EN	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	



# Analýza programu 1

## – GPIO režimy

### 9.2.1 Port configuration register low (GPIOx\_CRL) (x=A..G)

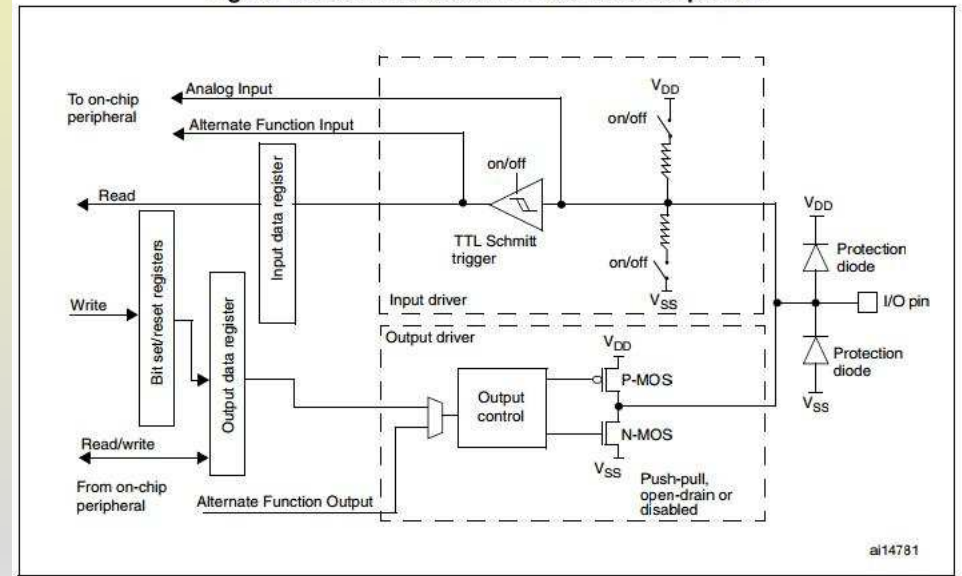
Address offset: 0x00

Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]	MODE7[1:0]	CNF6[1:0]	MODE6[1:0]	CNF5[1:0]	MODE5[1:0]	CNF4[1:0]	MODE4[1:0]	CNF3[1:0]	MODE3[1:0]	CNF2[1:0]	MODE2[1:0]	CNF1[1:0]	MODE1[1:0]	CNF0[1:0]	MODE0[1:0]
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF3[1:0]	MODE3[1:0]	CNF2[1:0]	MODE2[1:0]	CNF1[1:0]	MODE1[1:0]	CNF0[1:0]	MODE0[1:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- GPIO porty jsou 16-bitové, počet (A, B, ...) podle modelu procesoru
- Každý vývod řízen 4 bity v registru konfigurace (CRx)
  - 16x4b je 64b, proto je registr 2-dílný, CRL pro 0-7, CRH pro 8-15
  - Při nastavování vlastností GPIO vývodu je vhodné neovlivnit ostatní = nutné maskování 4-bitových "bloků"
- Základní režimy GPIO
  - Viz. také Tab. 20 a 21 v RM (kap. 9.1)
  - Bity MODEy[1:0]
    - 00: Input mode (defaultní stav po RESETu)
    - 01: Output mode, max. speed 10 MHz (strmost hrany)
    - 10: Output mode, max. speed 2 MHz
    - 11: Output mode, max. speed 50 MHz
  - Bity CNFy[1:0]
    - Input mode (MODE[1:0]=00)
      - 00: Analog mode
      - 01: Floating input (reset state)
      - 10: Input with pull-up / pull-down
      - 11: Reserved
    - Output mode (MODE[1:0] > 00)
      - 00: General purpose output push-pull
      - 01: General purpose output Open-drain
      - 10: Alternate function output Push-pull
      - 11: Alternate function output Open-drain
- Nejčastější kombinace
  - 0011 = 0x03 = Output mode 50MHz, output push-pull
  - 1000 = 0x08 = Input mode, input with pullup/down
  - 1011 = 0x0b = Output mode 50MHz, Alternate push-pull
    - Vývody jsou sdílené i s dalšími perifériemi (časovače, UART, ...) a pro využití musí mít "Alternate function"

Figure 13. Basic structure of a standard I/O port bit





# Analýza programu 1 – GPIO data

- GPIO porty mají datové registry – viz. RM 9.2.3-6
  - Input Data (GPIOx\_IDR)
    - Každý bit (0-15) odpovídá vstupní hodnotě
  - Output Data (GPIOx\_ODR)
    - Zapsaná log. hodnota se objeví na výstupu (pokud je tak nastaven CRx)
    - Téměř vždy se používá maskování, aby se neovlivnily ostatní bity
    - POZOR, operace OR, AND i XOR nejsou atomické
      - Interně proběhne sekvence Read-Modify-Write
      - Může být přerušena interruptem = problém !!
  - Bit Reset (GPIOx\_BRR) – atomická operace změny (!)
    - Vynuluje vybraný bit (resp. bity), ostatní bez změn
  - Bit Set/Reset (GPIOx\_BSRR) – atomická operace zápisu
    - Horních 16 bitů (16-31) nuluje odpovídající bit v ODR
    - Spodních 16 bitů (0-15) nastavuje výstup do log.1
  - Příklad použití BSRR

úvodní stav ODR – spodních 16b "nějaká" hodnota

ODR: 0000 0000 0000 0000 xxxx xxxx xxxx xxxx

GPIOA->BSRR = 0x0c000101; - zapsána "maskovací" hodnota

BSRR: 0000 1100 0000 0000 0000 0001 0000 0001

Výsledek v ODR – 2x vynulování, 2x "set"

ODR: 0000 0000 0000 0000 xxxx 00x1 xxxx xxx1

## 9.2.5 Port bit set/reset register (GPIOx\_BSRR) (x=A..G)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy**: Port x Reset bit y (y= 0 .. 15)

These bits are write-only and can be accessed in Word mode only.

0: No action on the corresponding ODRx bit

1: Reset the corresponding ODRx bit

*Note: If both BSx and BRx are set, BSx has priority.*

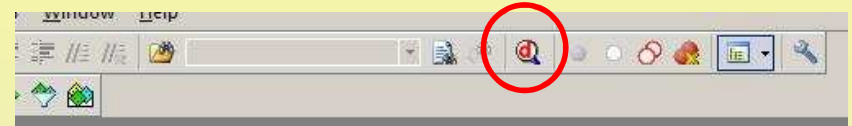
Bits 15:0 **BSy**: Port x Set bit y (y= 0 .. 15)

These bits are write-only and can be accessed in Word mode only.

0: No action on the corresponding ODRx bit

1: Set the corresponding ODRx bit

# Spuštění programu v Debug režimu



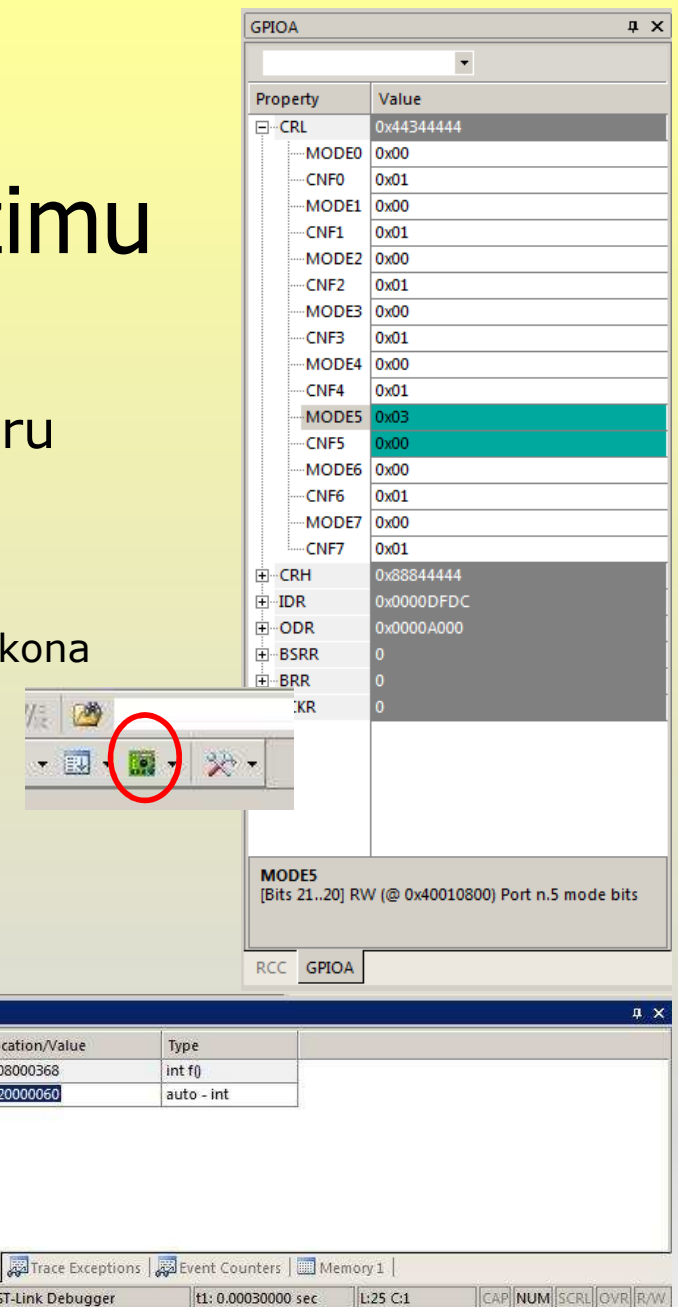
- Nezapomenout přeložit
- Před spuštěním Debug režimu se nahraje automaticky do desky



- Ovládání – ikony a klávesové zkratky
  - RST – resetuje CPU
  - Run – F5 – spustí program z aktuálního místa
  - Stop – přeruší vykonávání
  - Step – F11 – krok (příp. do funkce)
  - Step over – F10 – krok (na funkci ji vykoná jako celek)
  - Step out – Ctrl+F11 – dokončí aktuální funkci a vyskočí z ní
  - Run To Cursor – Ctrl+F10 – spustí vykonávání ke kurzoru
    - V podstatě jednorázový breakpoint
- Breakpoint – F9 – zapne/vypne, nebo kliknutím vlevo před řádkem

# Základní možnosti Debug režimu

- Lze sledovat/měnit obsah registrů procesoru
  - Registry jádra
    - Volby v menu Peripherals – Core Peripherals
  - Registry periférií
    - Volby v menu Peripherals – System Viewer nebo ikona
  - Zobrazují se bity nebo logické skupiny
- Lze sledovat/měnit hodnoty proměnných
  - Příp. View – Watch window
- Krokovat lze i v assembleru



- Viz. video [https://www.youtube.com/watch?v=\\_MFagEAPeq4](https://www.youtube.com/watch?v=_MFagEAPeq4)

# Instalace a spuštění Keil MDK 5 v Linuxu

- Ověřeno v distribuci Xubuntu 15.04
- Logicky vyžaduje pro provoz aplikaci Wine (ověřeno 1.6.2)
- Instalace IDE (5.15) bez potíží, moduly se instalují také v pořádku
- Na testovacím stroji nefunguje debugger
  - MDK má pro debug speciální DLL knihovny podle jednotlivých HW platforem, nepodařilo se mi rozběhnout
  - Program se dá editovat a přeložit
  - Výsledný AXF soubor se převede na BIN pomocí utility fromelf.exe
    - K nalezení v adresáři Keil/ARM/ARMCC/bin
    - Použitý Wine sice vypisuje chybové hlášení, ale BIN se vytvoří

```
...:~/wine/drive_c/.../Objects$ wine ~/wine/drive_c/.../fromelf.exe --bin --output=out.bin out.axf
```

- Pomocí ST-Link utilit lze nahrát BIN do desky Nucleo
  - Při instalaci nutno napřed mít v systému balík "autogen"
  - Optimálně stáhnout z github a přeložit - <https://github.com/texane/stlink>
  - Utilita st-flash a operace write od adresy 0x8000000

```
...:~/stlink.git$ sudo ./st-flash write ../wine/drive_c/.../Objects/out.bin 0x8000000
```

- Alternativně lze BIN nahrát do adresáře, který se vytvoří ve formě MassStorage

# Instalace Atollic True Studio (verze 5.3.0)

- Prostředí založené na IDE Eclipse
- Free verze také omezená 32kB výsledného kódu
- Na začátku zobrazuje "splash-screen"
- Stáhnout přímo <http://atollic.com/index.php/download/truestudio-for-arm>
  - Pozor, nutno zvolit Lite verzi
  - Instalace stylem Next, Next, ...
  - Na závěr instalace vygeneruje požadavek licenčního klíče, který je nutno odeslat na Atollic a na mail přijde příslušné "číslo", které je třeba zadat
    - Nutno tedy vyplnit funkční mail
- [https://www.dropbox.com/s/e3g988dbbzipweig/TrueSTUDIO\\_for\\_ARM\\_Lite\\_win32\\_v5.3.0\\_20150316-1058.exe?dl=0](https://www.dropbox.com/s/e3g988dbbzipweig/TrueSTUDIO_for_ARM_Lite_win32_v5.3.0_20150316-1058.exe?dl=0)
- Viz. video [https://www.youtube.com/watch?v=SpANJ\\_ALVLY](https://www.youtube.com/watch?v=SpANJ_ALVLY)

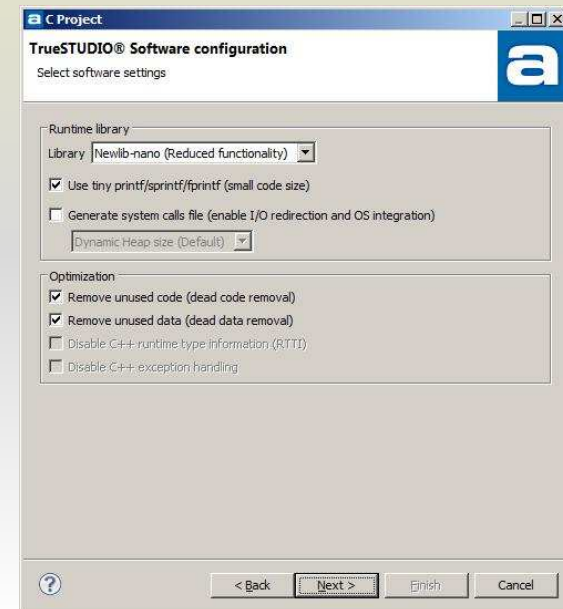
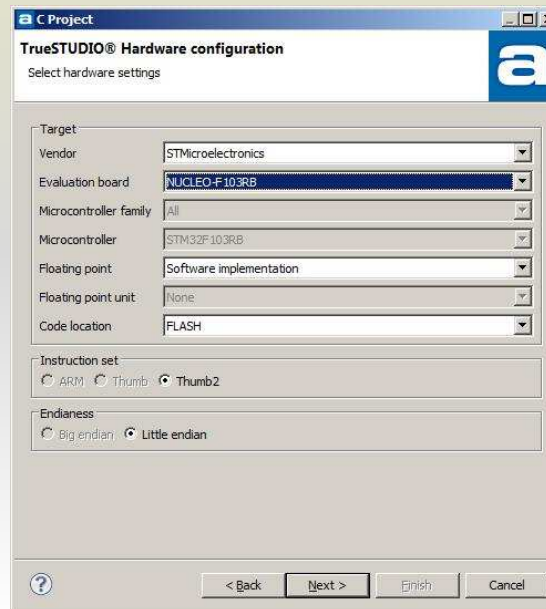
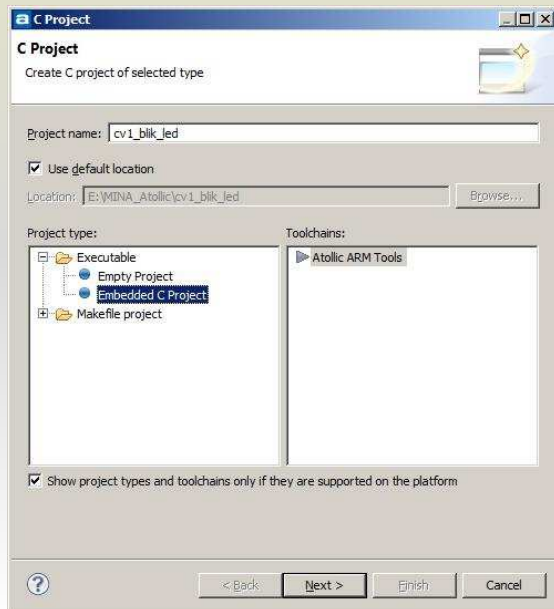


# Instalace Atollic True Studio (verze 6.0.0)

- Prostředí založené na IDE Eclipse
- Free verze také omezená 32kB výsledného kódu
- Na začátku zobrazuje "splash-screen"
- Stáhnout přímo <http://atollic.com/truestudio/>
  - Pozor, nutno zvolit Free verzi
  - Instalace stylem Next, Next, ...
- [https://dl.dropboxusercontent.com/u/22184251/TrueSTUDIO\\_for\\_ARM\\_win32\\_v6.0.0\\_20160823-1314.exe](https://dl.dropboxusercontent.com/u/22184251/TrueSTUDIO_for_ARM_win32_v6.0.0_20160823-1314.exe)
- Viz. video [https://www.youtube.com/watch?v=SpANJ\\_ALVLY](https://www.youtube.com/watch?v=SpANJ_ALVLY)
  - Sice pro starší verzi 5.3.0, v použité verzi není nutná registrace

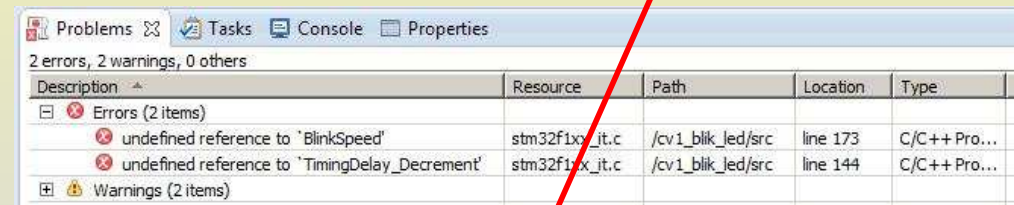
# Atollic TrueStudio – vytvoření projektu

- Otevřít nebo vytvořit Workspace
- New - C-projekt
  - Zvolit "Embedded C Project"
    - Bude vhodně nabízet odpovídající volby
  - Vybrat výrobce STMicroelectronics
  - Zvolit desku NUCLEO-F103RB (= budou se nabízet vhodné volby)
  - "Library" ponechat bez změn
    - Zatím nepotřebujeme
  - Připojení ST-Link je pro Nucleo jediné možné
  - Necháme vygenerovat Debug i Release větev



# Atollic TrueStudio – build

- Vygenerovaný projekt obsahuje sice demo "blikání", ale je třeba smazat komplet obsah souboru main.c
- Nahradit kódem z příkladu Keil – beze změn !!
- Vytvoření programu Project – Build project
  - Nebo ikona

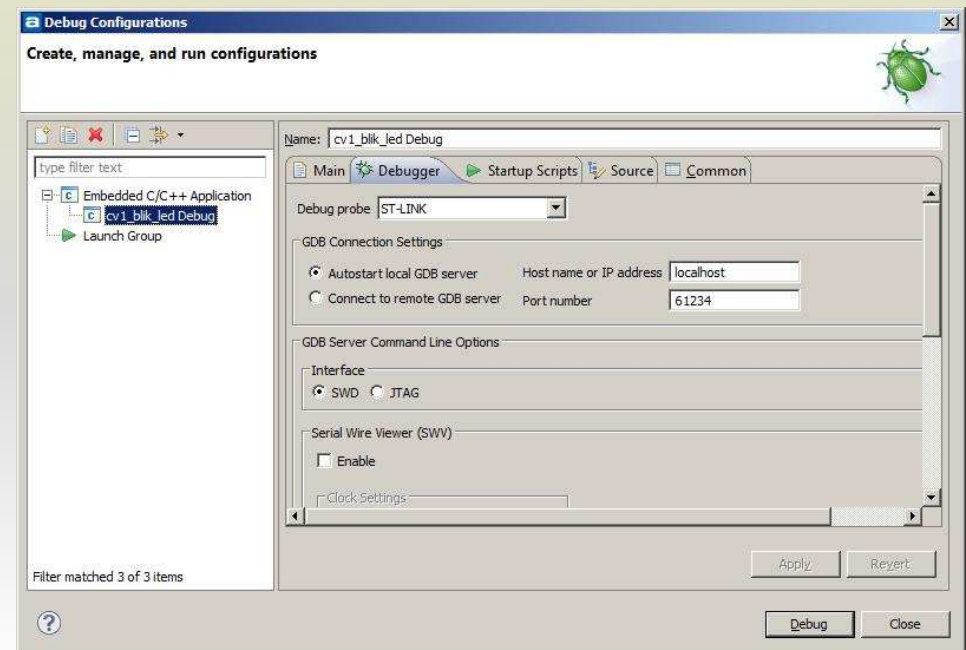


Description	Resource	Path	Location	Type
<b>Errors (2 items)</b>				
undefined reference to 'BlinkSpeed'	stm32f1xx_it.c	/cv1_blik_led/src	line 173	C/C++ Pro...
undefined reference to 'TimingDelay_Decrement'	stm32f1xx_it.c	/cv1_blik_led/src	line 144	C/C++ Pro...
<b>Warnings (2 items)</b>				

- Chyby při linkování
  - Neznámé symboly v xx\_it.c
  - Byl vygenerován pro demo, obsahuje přípravu pro obsluhu přerušení = zatím nepotřebujeme
  - Soubory stm32f1xx\_it.c a .h smazat z projektu
- Nový Build dává stejné chyby
  - Důvodem je existence minulého překladu
  - Je třeba zvolit Rebuild (= vše znova překládat)
- Zbývají 2 Warningy, které zatím ignorujeme
  - Je to daň za stejný kód jako pro Keil

# Atollic TrueStudio – spuštění (debug)

- Na počátku není konfigurován debugger
  - V menu Run – Debug Configurations ...
  - Zvolit přidání nové konfigurace (= New launch conf.)
  - V defaultním nastavením je ST-Link a SWD připojení pro Nucleo = správné nastavení
  - Ostatní parametry také vyhovují defaultně
- Zvolit Debug
  - Příště již bude tato konfigurace použita



# Atollic TrueStudio - debug

- Vytvoří se Debug "pohled"
  - Klasicky jako v každém Eclipse-based prostředí
- Krokování apod. klasicky v Eclipse
- Kromě "Registers" možno také zobrazovat "SFRs"
  - Obsah registrů periférií (= Special Function Register)
  - Pozor, při první spuštění se stává, že trvání několik vteřin, než se přehled registrů zkonfiguruje (= objeví)
  - Je možno rozkliknout až na jednotlivé bity
- Viz. video <https://www.youtube.com/watch?v=AMN1h4W87S4>





# Shrnutí získaných znalostí/dovedností – I

- ✓ Umím vytvořit nový projekt a nakonfigurovat jej
- ✓ Umím přeložit projekt
- ✓ Umím projekt nahrát do Nucleo desky
  
- ✓ Umím spustit debugger
  - Krokovat program
  - Používat breakpointy
  - Sledovat registry procesoru
  - Sledovat obsah proměnných
  
- ✓ Mám základní znalosti pro práci s GPIO
  - Periférie musí mít povolené hodiny
  - Periférie by měla projít RST cyklem po připojení hodin
  - GPIO mají různé režimy, nastavení ve 4-bitech v GPIOx->CRx
  - Hodnoty bitů možno nastavit v registru GPIOx->ODR nebo BSRR

# Program domácí přípravy

I. Instalace Keil MDK, příp. Atollic IDE, ST-Link, analýza demo-programu, použití debuggeru

## **II. Bitové operace, maskování, GPIO**

III. Funkce z knihovny stdio – procvičení

IV. Časování, nastavení RCC bloku, rozdělení hodinových signálů perifériím

V. Časovač, využití přerušení

VI. Procvičení práce s LCD, doplnění functionality

VII. Zpracování dat ze senzorů (A/D, externí)

VIII. DMA

IX. Instalace a procvičení RTOS

X. - XIII. Samostatná práce

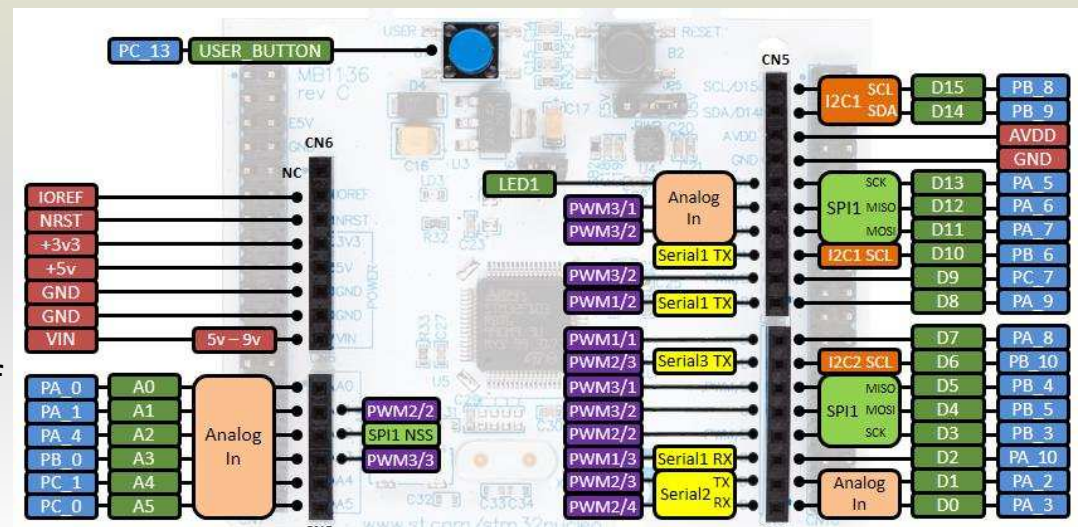
Čísla "týdnů" odpovídají číslování "po" daném cvičení, tedy I. je plánována mezi cv. 1 a 2

# Prohloubení znalostí z cvičení 2

- Nastavení IO portu jako vstupního – čtení tlačítka (PC13)
- Makro SHIFT4 pro usnadnění práce s 4-bitovou hodnotou/maskou v CRL/H konfiguračním registru
- Funkce pro nastavení libovolného GPIO signálu

```
typedef enum {portINPUT, portOUTPUT, portALTOUT, portALTOD, portALTIN, portANALOG} eIOPortType;  
int InitIOPort(GPIO_TypeDef *baseGPIO, uint32_t portNum, eIOPortType typ)  
{  
    ...  
}
```

- Připojení mbed-shield (LED svítí log. 0, tlačítko stisk' = log. 1)
  - LED R D5 = PB4
  - LED G D9 = PC7
  - **LED B D8 = PA9**
  - Down A3 = PB0
  - Left A4 = PC1
  - **Center D4 = PB5**
  - Up A2 = PA4
  - Right A5 = PC0
  - Schéma viz. ApplicationShield.pdf



# Úkol 1 – využití jiné LED

- Upravte kód tak, aby blikala zelená složka LED (PC7)
- Upravte kód pro červenou složku LED (PB4)
  - !!! NEFUNGUJE !!!
  - Vývod PB4 je sdílen signálem NJTRST (rozhraní JTAG/SWD)
    - Viz. kap 9.3.5 a Table 37 v RM
    - Nutno deaktivovat = nastavit režim 001 pro SWJ\_CFG
      - Nastavení v registru AFIO\_MAPR – viz. kap. 9.4.2
      - Nutno napřed povolit "periférii" AFIO na sběrnici APB2

```
... Na konec funkce InitIOPort ...
if ((baseGPIO == GPIOB) && (portNum == 4))
{
    // nastavuje se PB4 ?
    if (!(RCC->APB2ENR & RCC_APB2ENR_AFIOEN))
    {
        RCC->APB2ENR |= RCC_APB2ENR_AFIOEN;
        RCC->APB2RSTR |= RCC_APB2RSTR_AFIOEN;
        RCC->APB2RSTR &= ~RCC_APB2RSTR_AFIOEN;
    }

    AFIO->MAPR &= ~ AFIO_MAPR_SWJ_CFG;
    AFIO->MAPR |= AFIO_MAPR_SWJ_CFG_NOJNTRST;
}
...
```

9.4.2 AF remap and debug I/O configuration register (AFIO\_MAPR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Reserved					SWJ_CFG[2:0]			Reserved					ADC2_ETRG_REGMAP	ADC2_ETRG_REGMAP	ADC1_ETRG_REGMAP	ADC1_ETRG_REGMAP	TIM5_H4_REMAP
					W	W	W						rw	rw	rw	rw	rw
Reserved					SWJ_CFG[2:0]			Reserved					TIM5_H4_REMAP				
					W	W	W										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
PD01_REMAP	CAN_REMAP[1:0]	TIM4_REMAP	TIM3_REMAP[1:0]	TIM3_REMAP[1:0]	TIM2_REMAP[1:0]	TIM2_REMAP[1:0]	TIM1_REMAP[1:0]	USART3_REMAP[1:0]	USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_REMAP					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:27 Reserved

Bits 26:24 SWJ\_CFG[2:0]: Serial wire JTAG configuration

These bits are write-only (when read, the value is undefined). They are used to configure the SWJ and trace alternate function I/Os. The SWJ (Serial Wire JTAG) supports JTAG or SWD access to the Cortex® debug port. The default state after reset is SWJ ON without trace. This allows JTAG or SW mode to be enabled by sending a specific sequence on the JTMS / JTCK pin.

000: Full SWJ (JTAG-DP + SW-DP): Reset State

001: Full SWJ (JTAG-DP + SW-DP) but without NJTRST

010: JTAG-DP Disabled and SW-DP Enabled

100: JTAG-DP Disabled and SW-DP Disabled

Other combinations: no effect

# Úkol 2 - kombinace LED tvoří 7 barev

- "Osmá barva" je černá = nesvítí žádná LED
- Pro nastavení bitu si možno vytvořit makra

```
...  
#define SET_IO_LOW(port, num)    (port->BSRR = 1 << (num + 16))  
#define SET_IO_HIGH(port, num)   (port->BSRR = 1 << (num))  
#define TOGGLE_IO(port, num)     (port->ODR ^= (1 << num))  
...
```

- A pohodlně je používat

```
...  
    TOGGLE_IO(GPIOA, 9);  
...  
    SET_IO_HIGH(GPIOB, 4);  
...
```

# Úkol 3 – spojení joysticku a barev

- Spojte získané zkušenosti v komplexnější aplikaci
- Stiskem tlačítek se mění chování blikání LED
  - Střední = při stisknutí nebliká nic, jinak ano
  - Nahoru/dolů = mění barvu, která bliká
    - Ideálně ve zvoleném pořadí
  - Vlevo/vpravo = mění rychlost blikání
    - Ideálně z několika předem vybraných konstant



# Shrnutí získaných znalostí/dovedností – II

- ✓ Umím rozsvítit libovolnou LED na mbed-shield
- ✓ Umím zjistit stisknutí tlačítek joysticku na mbed-shield
- ✓ Umím používat společnou inicializační funkci IO portů
  - ✓ Rozšířil jsem ji pro všechny GPIOx (A-F)
  - ✓ Upravil jsem chování procesoru tak, aby se dal využívat i port PB4
- ✓ Vytvořil jsem aplikaci na "různobarevné blikání"
  - ✓ Řízeno tlačítky joysticku

# Program domácí přípravy

- I. Instalace Keil MDK, příp. Atollic IDE, ST-Link, analýza demo-programu, použití debuggeru
- II. Bitové operace, maskování, GPIO
- III. Funkce z knihovny stdio – procvičení**
- IV. Časování, nastavení RCC bloku, rozdělení hodinových signálů perifériím
- V. Časovač, využití přerušení
- VI. Procvičení práce s LCD, doplnění functionality
- VII. Zpracování dat ze senzorů (A/D, externí)
- VIII. DMA
- IX. Instalace a procvičení RTOS
- X. - XIII. Samostatná práce

Čísla "týdnů" odpovídají číslování "po" daném cvičení, tedy I. je plánována mezi cv. 1 a 2

# Funkce pro práci s UART do modulu

- Vezměte jako základ program pro komunikaci přes UART ze cvičení
  - Uvažujte variantu s MicroLIB a stdio funkcí
- Funkce fputc, fgetc a Uart2Init přesuňte do samostatného souboru
  - Bude se překládat zvlášť jako samostatný modul
  - Bude součástí naší „knihovny MINA-funkcí“
- Vytvořte příslušný hlavičkový soubor
  - Minimálně kvůli funkci Uart2Init
- Ověřte funkčnost tohoto řešení

# Domácí úkol – spojení mbed-Shield a UART komunikace

- Vytvořte projekt, kde bude možná komunikace s PC po UARTu a bude možné řídit svícení RGB LED a snímat stavy tlačítek joysticku
  - Využijte samostatného modulu s UART komunikací
  - Využijte funkcí a kódu z dřívějších projektů
- Zvolte vhodný způsob ovládání z terminálu pro
  - Nastavení svícení barvy LED
  - Nastavení blikání barvou LED
  - Zobrazení stavu tlačítek joysticku

# Shrnutí získaných znalostí/dovedností – III

- ✓ Umím nastavit sériovou komunikaci
- ✓ Umím přijímat a vysílat data pomocí stdio funkcí
  - ✓ Program testuje, zda jsou k dispozici data k příjmu
  - ✓ Funkce související s UART-komunikací jsou ve zvláštním souboru
    - ✓ Logicky včetně příslušného hlavičkového souboru
- ✓ Vytvořil jsem aplikaci pro RGB a Joystick řízenou z PC
  - ✓ Lze volit barvu svícení/blikání
  - ✓ Do PC se hlásí zvolená barva
  - ✓ Do PC se hlásí stisknuté tlačítko joysticku

# Program domácí přípravy

- I. Instalace Keil MDK, příp. Atollic IDE, ST-Link, analýza demo-programu, použití debuggeru
- II. Bitové operace, maskování, GPIO
- III. Funkce z knihovny stdio – procvičení
- IV. Časování, nastavení RCC bloku, rozdělení hodinových signálů perifériím**
- V. Časovač, využití přerušení
- VI. Procvičení práce s LCD, doplnění functionality
- VII. Zpracování dat ze senzorů (A/D, externí)
- VIII. DMA
- IX. Instalace a procvičení RTOS
- X. - XIII. Samostatná práce

Čísla "týdnů" odpovídají číslování "po" daném cvičení, tedy I. je plánována mezi cv. 1 a 2



# Aplikace/projekt využívající knihovnu

- Pro následující projekty budeme postupně budovat knihovnu (= balík funkcí)
  - Připojení do projektu pomocí přidání \*.c souborů z knihovního adresáře
    - Nejlépe do oddělené "Source group" v rámci "Target"
    - Možno přidat i více najednou pomocí "Add"
- Jednotlivé skupiny funkcí je vhodné dělit do souborů podle společné funkcionality
  - Uart
  - Mbed (zatím jen GPIO)
  - ...
- Většina "specializovaných" částí knihovny bude logicky využívat společného HW nastavení
  - Např. GPIO a piny

# Doporučené funkce pro mbed shield

- Některé mohou být nahrazeny makry
  - Pozor na závorkování
- Je vhodné využívat enum typy
  - Překladač může hlídat význam kódu (např. switch)
  - Pozor, musí být v .H souboru

```
typedef enum {portINPUT, portOUTPUT, portALTOUT, portALTIN, portANALOG} eIOPortType;

int InitIOPort(GPIO_TypeDef *baseGPIO, uint32_t portNum, eIOPortType typ)
{
    ...
}

typedef enum {colorBLACK, colorRED, colorGREEN, colorBLUE, colorYELLOW, colorCYAN, colorWHITE, colorPINK } eColors;

void SetRGBLED(eColors barva)
{
    ...
}

typedef enum {joyLEFT = 1, joyRIGHT = 2, joyUP = 4, joyDOWN = 8, joyCENTER = 16} eJoyButton;

eJoyButton GetJoyState(void) // muze byt i bitova kombinace tlacitek, zadne nestisknute = 0
{
    ...
}
```

# Shrnutí získaných znalostí/dovedností – IV

- ✓ Umím vytvořit obsluhu přerušení pro SysTick a nastavit ho
- ✓ Mám rozdělený kód mezi funkci main a knihovnu
  - ✓ V knihovně je kód pro sériovou komunikaci po UART2 pomocí funkcí z stdio
  - ✓ V knihovně jsou funkce pro využití HW mbed-shield
    - ✓ Tlačítka joysticku
    - ✓ RGB LED ve statickém režimu (ON, OFF)
- ✓ Umíme vytvořit aplikaci využívající "naši" knihovnou
  
- ✓ Upravil jsem aplikaci pro RGB a Joystick přes UART aby využívala časování pomocí SysTick

# Program domácí přípravy

- I. Instalace Keil MDK, příp. Atollic IDE, ST-Link, analýza demo-programu, použití debuggeru
- II. Bitové operace, maskování, GPIO
- III. Funkce z knihovny stdio – procvičení
- IV. Časování, nastavení RCC bloku, rozdělení hodinových signálů perifériím
- V. Časovač, využití přerušení**
- VI. Procvičení práce s LCD, doplnění functionality
- VII. Zpracování dat ze senzorů (A/D, externí)
- VIII. DMA
- IX. Instalace a procvičení RTOS
- X. - XIII. Samostatná práce

Čísla "týdnů" odpovídají číslování "po" daném cvičení, tedy I. je plánována mezi cv. 1 a 2

# Rozšíření PWM pro všechny LED v RGB

- Ze cvičení je připraven kód pro blikání LED G
- Připojení ostatních:
  - LED R - D5 = PB4 – Timer 3, Channel 1
  - LED G - D9 = PC7 – Timer 3, Channel 2
  - LED B - D8 = PA9 – Timer 1, Channel 2
- Zprovoznění LED R je jednoduchá změna kanálu
  - V TIM3->CCMR1 jsou to bity OC1M
  - V TIM3->CCER nutno povolit CC1E
  - A střída se nastavuje v CCR1
  - POZOR - TIM3\_REMAP v AFIO musí být nastaven na 10 = "partial"
    - Viz. RM 9.4.2
    - Nelze tedy využívat PWM pro R i pro G najednou !!!
- Zprovoznění LED B = Timer1
  - Nutno povolit – POZOR – běží z APB2
    - Zohlednit při výpočtu rychlosti
  - Pracuje se s TIM1->... Registry
  - TIM1\_REMAP v AFIO zřejmě zůstane na 00 ("noremap"), aby PA9 byl CH2

# Převodní tabulka PWM = svit LED

- Závislost jasu LED (a světelných zdrojů obecně) na PWM není lineární
  - Cca od 50% do 100% PWM svítí skoro stejně
  - Závislost je možné linearizovat pomocí S-křivky
    - <http://electronics.stackexchange.com/questions/1983/correcting-for-non-linear-brightness-in-leds-when-using-pwm>
    - [http://www.pyroelectro.com/tutorials/fading\\_led\\_pwm/theory2.html](http://www.pyroelectro.com/tutorials/fading_led_pwm/theory2.html)



# Shrnutí získaných znalostí/dovedností – V

- ✓ Umím vytvořit obsluhu přerušení od časovače
- ✓ Využívám funkce z knihovny
  - ✓ V knihovně je kód pro sériovou komunikaci po UART2 pomocí funkcí z stdio
  - ✓ V knihovně jsou funkce pro využití HW mbed-shield
    - ✓ Tlačítka joysticku
    - ✓ RGB LED ve statickém režimu (ON, OFF)
- ✓ Pomocí PWM umím generovat různý jas na složkách RGB LED
  - ✓ Jedním z demo režimů je "dýchání"

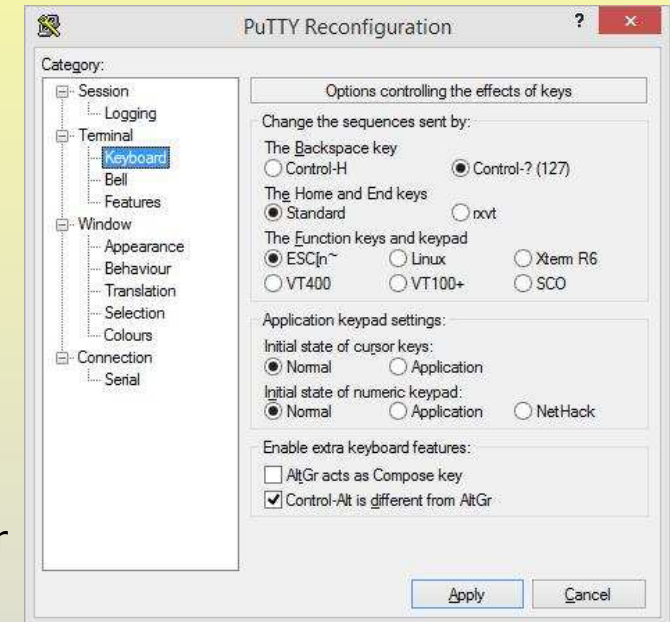
# Program domácí přípravy

- I. Instalace Keil MDK, příp. Atollic IDE, ST-Link, analýza demo-programu, použití debuggeru
- II. Bitové operace, maskování, GPIO
- III. Funkce z knihovny stdio – procvičení
- IV. Časování, nastavení RCC bloku, rozdělení hodinových signálů perifériím
- V. Časovač, využití přerušení
- VI. Procvičení práce s LCD, doplnění functionality**
- VII. Zpracování dat ze senzorů (A/D, externí)
- VIII. DMA
- IX. Instalace a procvičení RTOS
- X. - XIII. Samostatná práce

Čísla "týdnů" odpovídají číslování "po" daném cvičení, tedy I. je plánována mezi cv. 1 a 2

# Spojení modulů pro práci s UART a LCD

- Vytvořte aplikaci, která
  - Umí přijímat data z sériového portu (UART)
  - Zobrazovat na LCD
  - Využijte funkce z knihovny stdio
  - Vhodně reagujte na speciální znaky
    - CR (carriage return)
    - LF (line feed)
      - Pozor na nastavení terminálu, co posílá po stisku Enter
      - Zjistit např. pomocí debuggeru
    - ... Další ?
  - Pomocí vybraného znaku/kódu smažte displej
    - Začne se potom zobrazovat od pozice 0,0
    - V LCD knihovně lze využít funkci MBED\_LCD\_FillDisp
- Do knihovny přidejte funkci pro inverzní výpis textu
  - Např. MBED\_LCD\_WriteCharXY\_negative
  - Využijte tuto funkci při zobrazování v aplikaci
    - Pro přepínání pozitiv/negativ zvolte vhodný znak/kód



# Shrnutí získaných znalostí/dovedností – VI

- ✓ Umím přidat do projektu "knihovnu" pro LCD displej
- ✓ Umím vypsát na LCD text
  - ✓V knihovně je k dispozici funkce MBED\_LCD\_WriteCharXY
  - ✓Využívá se základní font 8x8
- ✓ Umím přijímat data (text) z UARTu (terminál v PC) a zobrazovat je na LCD
  - ✓Speciální znakem možno "smazat displej" a psát o pozice 0,0
  - ✓Zobrazování respektuje znaky CR a LF
  - ✓Je možné přepnout na "negativní" zobrazení a zpět

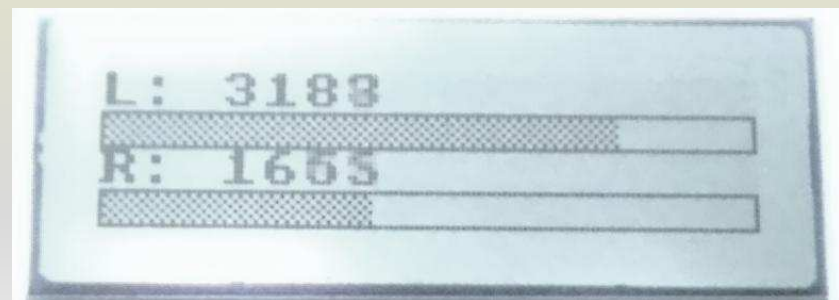
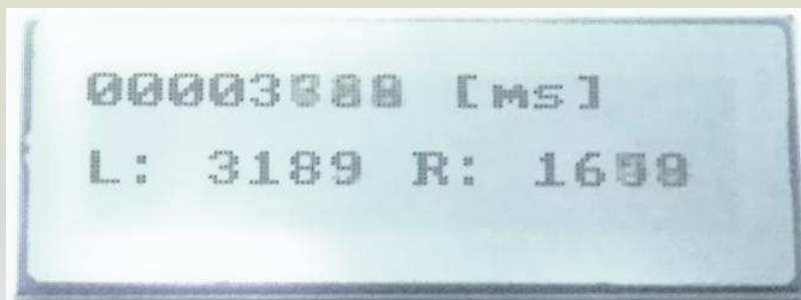
# Program domácí přípravy

- I. Instalace Keil MDK, příp. Atollic IDE, ST-Link, analýza demo-programu, použití debuggeru
- II. Bitové operace, maskování, GPIO
- III. Funkce z knihovny stdio – procvičení
- IV. Časování, nastavení RCC bloku, rozdělení hodinových signálů perifériím
- V. Časovač, využití přerušení
- VI. Procvičení práce s LCD, doplnění functionality
- VII. Zpracování dat ze senzorů (A/D, externí)**
- VIII. DMA
- IX. Instalace a procvičení RTOS
- X. - XIII. Samostatná práce

Čísla "týdnů" odpovídají číslování "po" daném cvičení, tedy I. je plánována mezi cv. 1 a 2

# Spojení A/D převodníku a LCD

- Vytvořte aplikaci, která bude měřit pomocí A/D převodníku
  - Vstupy jsou potenciometry na mbed-kitu
  - Vypisujte hodnoty:
    - Číselně změřená data (0 - 4095)
    - Číselně napětí (0 - 3,3V)
    - Ve formě "bar-grafu"



- Pokud to bude možné, vyzkoušejte přerušení od dokončení převodu A/D převodníku



# Shrnutí získaných znalostí/dovedností – VII

- ✓ Umím používat A/D převodník
- ✓ Umím spojit data z A/D převodníku s LCD
  - ✓ Zobrazují se hodnoty z A/D jako číslo
  - ✓ Zobrazují se hodnoty z A/D ve formě bargrafu

# Program domácí přípravy

- I. Instalace Keil MDK, příp. Atollic IDE, ST-Link, analýza demo-programu, použití debuggeru
- II. Bitové operace, maskování, GPIO
- III. Funkce z knihovny stdio – procvičení
- IV. Časování, nastavení RCC bloku, rozdělení hodinových signálů perifériím
- V. Časovač, využití přerušení
- VI. Procvičení práce s LCD, doplnění functionality
- VII. Zpracování dat ze senzorů (A/D, externí)

## **VIII.DMA**

- IX. Instalace a procvičení RTOS
- X. - XIII. Samostatná práce

Čísla "týdnů" odpovídají číslování "po" daném cvičení, tedy I. je plánována mezi cv. 1 a 2

# Ověření efektů DMA přenosů MEM2MEM

- Navažte na příklady předvedené na cv.

```
CoreClock: 72000000  
FOR Array copy: 713 ms  
FOR PTR copy: 485 ms  
DMA copy: 172 ms
```

- Během každého průchodu cyklem inkrementujte proměnnou

- Simuluje to vykonávání dalšího kódu
- V kopírovacím cyklem při každém kroku cyklu
- Při DMA při každém průchodu cyklem

```
while(!(DMA1->ISR & DMA_ISR_TCIF7))
```

- Doporučuji využít typ uint64\_t

- V printf existuje makro PRIu64 umožňující zpracování 64b hodnot – viz.:

```
printf("DMA copy: %d ms, %" PRIu64 "\r\n", tickEnd - tickStart, cntMax);
```

- Očekávané výsledky:

```
CoreClock: 72000000  
FOR Array copy: 827 ms, 2048000  
FOR PTR copy: 513 ms, 2048000  
DMA copy: 172 ms, 646677
```

# Shrnutí získaných znalostí/dovedností – VIII

- ✓ Umím používat DMA přenosy
- ✓ Ověřil jsem efekty DMA přesnosů