

# STM32CubeIDE, Git, základní funkce

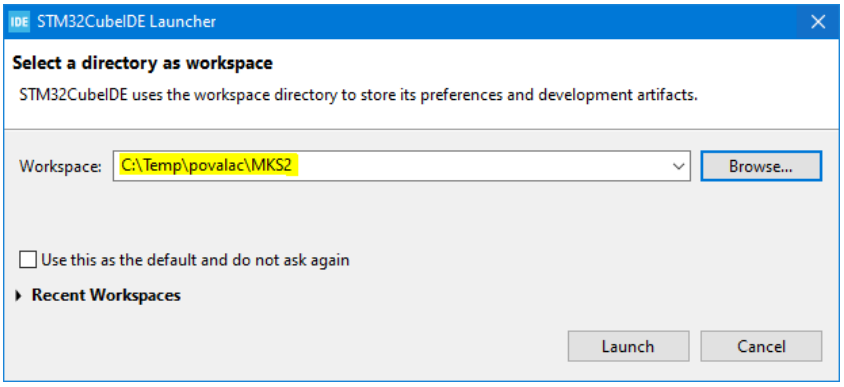
## 1 Zadání

- Seznamte se s programem TortoiseGit. Založte si repozitář na Githubu, vytvořte pracovní kopii adresáře (Git Clone).
- Seznamte se s vývojovou deskou NUCLEO-F030R8, modulem STMrel a prostředím STM32CubeIDE.
- Vytvořte jednoduchou aplikaci, která bude blikat s diodou LD2. Otestujte na vývojové desce, ověřte ladění krokováním. Uložte vytvořenou aplikaci do repozitáře (Git Commit).
- Vytvořte aplikaci, která bude realizovat blikání diodou LD2 dle předdefinovaného řetězce v morseovce. Commitujte do repozitáře. Následně kód upravte tak, aby byla sekvence morseovky zapsaná v jediné 32bitové konstantě, commitujte do repozitáře.
- Lokální repozitář uložte na server (Git Push).

## 2 Návod

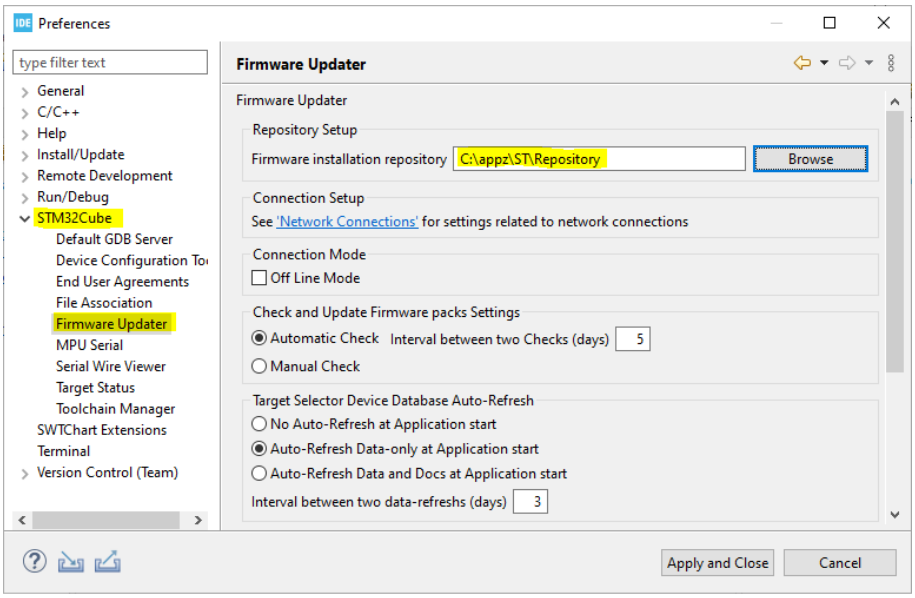
### 2.1 Základní seznámení

- Založte si účet na Githubu, vytvořte si nový repozitář nazvaný např. MKS.
- Ve složce **C:\Temp** si založte svůj pracovní adresář a v něm vytvořte pomocí operace Git Clone pracovní kopii svého osobního repozitáře. Pro přihlášení na Github využijte možnosti Sign in with your browser.
- Do této pracovní kopie přepněte workspace STM32CubeIDE (při startu nebo pomocí File / Switch Workspace).

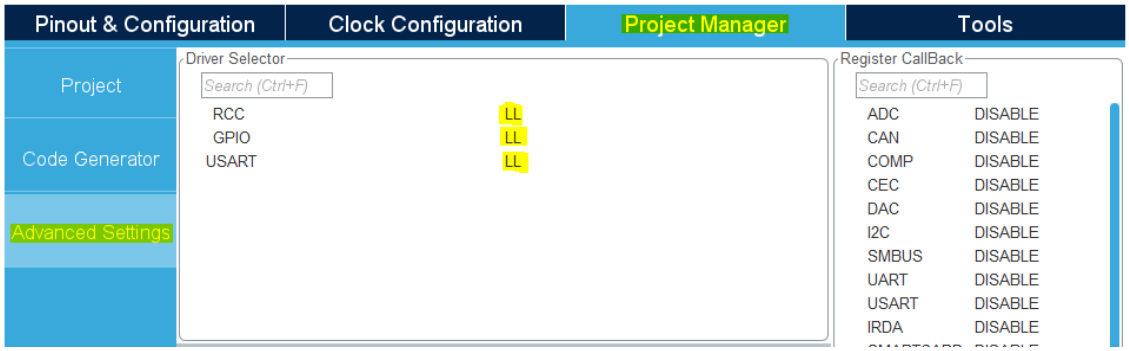



- Dále pracujte pouze na pracovní kopii.
- Potřebná schémata vývojové desky najdete v eLearningu. Modul STMrel v tomto cvičení nevyužijeme.

- Prostředí STM32CubeIDE ukládá množství dat lokálně, zejm. repozitáře s knihovnami. Aby nebylo nutné znovu vše stahovat, je vhodné nastavit cestu k centrálnímu repozitáři knihoven (platí pro učebny, ne soukromá PC) v menu Window / Preferences / STM32Cube / Firmware Updater:



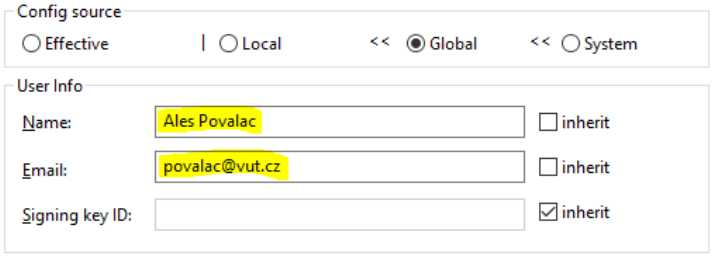

- Nový projekt s knihovnami se zakládá přes File / New / STM32 Project / **Board** Selector / NUCLEO-F030R8. Pro prvních několik cvičení budeme **využívat LL knihoven**. Potvrďte inicializaci všech periferií do výchozího nastavení. Přepnutí knihoven z komplexních HAL (které budeme využívat později) na kompaktnější LL naleznete v Project Manager / Advanced Settings / Driver Selector, pro všechny drivery vybrat LL:




- Po uložení IOC souboru dojde k automatickému vygenerování kódu, příp. lze generování spustit ručně pomocí Project / Generate Code (Alt+K).
- Překlad (Build all) lze spustit pomocí zkratky Ctrl+B, ladění (tj. spuštění, Debug) projektu pomocí F11.
- Při prvním spuštění se zobrazí nastavení debuggeru, potvrďte výchozí konfiguraci. Pokud je firmware debuggeru v připojené vývojové desce neaktuální, může být vyžádán upgrade.
- Během krokování ladění lze s výhodou používat klávesové zkratky, viz menu Run.

## 2.2 Verzování a TortoiseGit

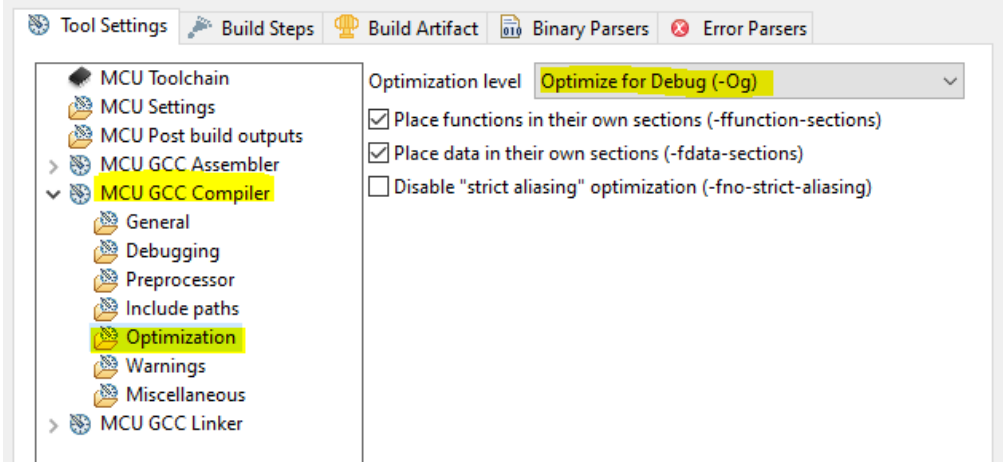
- Vždy commitujte pouze zdrojové kódy (c, h, Makefile apod.), nikoliv výsledky překladu (o, d, elf, hex, soubory workspace apod.). S výhodou lze použít ignore listu, soubor **`.gitignore`** lze vytvořit ručně nebo pomocí Git / Add to ignore list. Vhodné je ignorovat složky Debug v jednotlivých projektech (tj. **`/*/Debug`**) a složku **`.metadata`**.
- Uložení pracovní kopie do repozitáře proveďte pomocí Git Commit. Git poprvé vyzve k zadání jména a emailu.




- Uložte lokální repozitář na server (Git Push). V případě požadavku na Github přihlášení využijte možnosti Sign in with your browser. Prohlédněte si výsledek přes web Githubu.

## 2.3 Rozblikání LED

- Pokud jste potvrdili inicializaci periferií do výchozího nastavení, je LED dioda LD2 na vývojovém kitu správně nakonfigurována. Nezapomeňte přepnout knihovny z HAL na LL.
- Po uložení IOC souboru dojde k automatickému vygenerování kódu, příp. lze generování spustit ručně pomocí Project / Generate Code (Alt+K).
- Dobrým zvykem pro rozsáhlejší projekty je zapnutí optimalizace kompilátoru hned po založení projektu. Položka je poměrně hluboko v nastaveních: Project / Properties / C/C++ **Build** / Settings / MCU GCC **Compiler** / Optimization / Optimization Level, doporučené nastavení je Optimize for Debug (-Og).

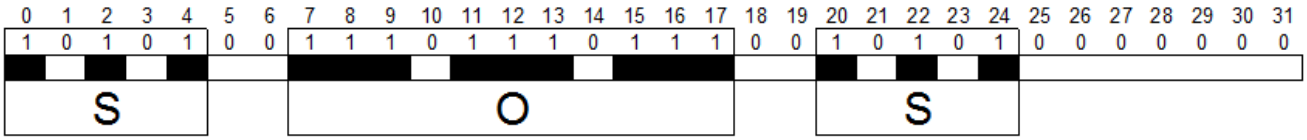



- V souboru main.c doplňujte vlastní kód vždy mezi komentáře **`USER CODE BEGIN`** a **`USER CODE END`**. Pokud toto pravidlo porušíte, bude váš kód smazán při přegenerování projektu z CubeMX.

- Inicializace periferií jsou automaticky vygenerovány, je tedy třeba doplnit kód pro samotné blikání. Ten se bude nacházet ve smyčce `while (1)`, doplňujte před komentář `/* USER CODE END WHILE */`.
- Změny výstupní úrovně pinu je možné provádět pomocí LL funkcí:
  - `LL_GPIO_SetOutputPin(LD2_GPIO_Port, LD2_Pin);`
  - `LL_GPIO_ResetOutputPin(LD2_GPIO_Port, LD2_Pin);`
- Tyto funkce využívají vygenerované definice ze souboru main.h:
  - `#define LD2_GPIO_Port GPIOA`
  - `#define LD2_Pin LL_GPIO_PIN_5`
- Podobně lze využít přímý přístup na příslušnou GPIO periferii pomocí registrů BSRR a BRR:
  - `GPIOA->BSRR = (1<<5); // set`
  - `GPIOA->BRR = (1<<5); // reset`
- Čekání lze řešit využitím `LL_mDelay(200)`, jako parametr přebírá funkce dobu čekání v milisekundách.

## 2.4 Blikání v morseovce

- Vývojová deska bude pomocí LD2 blikat v Morseově abecedě kód „SOS“.
- Sekvenci můžete uložit jako jedničky (svítí) a nuly (nesvítí) např. do pole typu `uint8_t pole[32]` a provést jeho inicializaci přímo v kódu.




- Kromě samotného pole budete potřebovat iterační proměnnou, která bude do tohoto pole ukazovat. Bude-li mít pole na indexu dle této proměnné hodnotu 0, výstup pro LD2 nastavíme na log. 0, v případě nenulové hodnoty na log. 1.
- Proveďte commit pracovní kopie do Gitu.
- Optimalizace využití paměti: Sekvence bude uložená v typu `uint32_t`, využijte binární zápis prvků (např. `0b0100`), dosáhněte stejné funkcionality. Budete potřebovat bitové operace a bitové posuny.
- Proveďte commit pracovní kopie do Gitu.
- Nezapomeňte odeslat lokální repozitář na server (Git Push).