STM32 – STM32CubeIDE

Petr Novák (novakpe@fel.cvut.cz) / 29.04.2021

Obsah

1 Úvod

Proč používáme ARMy od STMicroelectronics (zkráceně STM):

- SW vývojové nástroje jsou zcela zdarma a to „STM32CubeIDE“. Možnost stažení z https://www.st.com/en/development-tools/stm32cubeide.html (pro Windows / Linux / MacOS).
- Snadno dostupné a cenově velmi přijatelné HW vývojové desky STM-NUCLEO 32/64/144 nebo STM-DISCOVERY (na Aukro, Bazos, eBay, Farnell, …) a k nim i různé rozšiřující moduly.
- Všechny vývojové desky STM-NUCLEO 32/64/144 nebo STM-DISCOVERY a mnohé další obsahují přímo programátor / debug (nejsou tedy potřeba žádné další drahé HW nástroje).
- Dostatečný rozsah typů procesorů ARM Cortex M0/M3/M4/M7/… Dostupné jak v rychlé verzi, tak se sníženou spotřebou, případně jiné optimalizace / konfigurace (STM32F / STM32L / STM32G / …). Dnes i moduly s STM32 obsahující BlueTooth/WIFI.
- Snadno dostupné i samotné procesory v mnoha různých pouzdrech (8pin, 20pin, 32pin, 64pin, …), s různou velikostí FLASH/RAM a s různým počtem periferií (USB, UART, I2C, SPI, časovače, …).
- Dostupné tzv. knihovny HAL (Hardware Abstraction Layer), které poskytují velkou přenositelnost vytvořeného kódu na různé verze procesorů Cortex M0/M0/M3/M4/M7/… a tedy STM32F / STM32L / STM32G / … (samozřejmě pouze od STM).
- Vcelku velká (ne však ideální) podpora při vývoji aplikací. Generování (nejen) základního projektu pomocí GUI přímo v nástroji STM32CubeIDE.
- Vcelku velká dostupnost informací na WWW (návody / rady / diskuze / knížky).

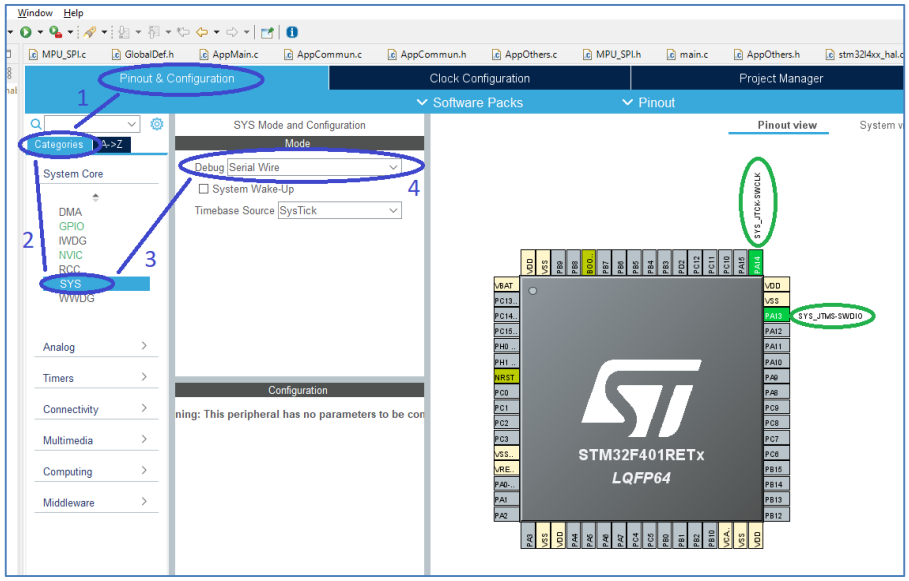
- Jedná se spíše o menší ARMy, které jsou velmi vhodné pro výukové účely, menší samostatná zařízení / projekty a zejména pro přechod z 8bit procesorů (I51/AVR/PIC/…).

2 Vytvoření nového projektu

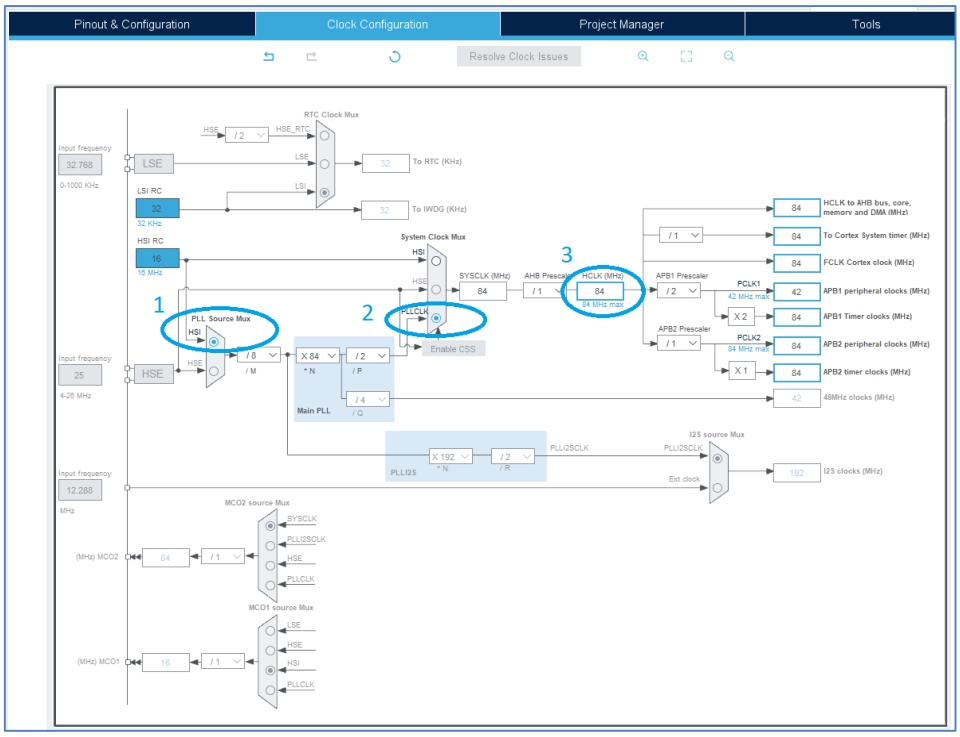
Procesory ARM jsou poněkud sofistikovanější než běžné 8bit procesory a proto i vytvoření (základu) projektu je poněkud složitější. V ARMu je potřeba nastavit nejen zdroj (interní/externí) frekvence a rychlost (pomocí PLL násobení) oscilátoru, ale často i časování pro interní sběrnice (pro různé typy interních periferií) a rovněž rychlost přístupu k FLASH/RAM (podle rychlosti jádra). Toto nastavení je vcelku složité a jakákoli chyba může způsobit zastavení procesoru (jeho interní HW ochrana). Proto je vhodné (minimálně) pro tvorbu základního / základu projektu použít nějaký GUI pomocný nástroj.
Vytvořit nový projekt lze přímo v STM32CubeIDE (V1.6.1, příklad pro NUCLEO-F401):

- **File** (menu vlevo v záhlaví aplikace)
- **New** (položka v rozbalovacím menu)
- **STM32** Project (položka v rozbalovacím pod-menu)
- Zobrazí se dialog pro výběr cílového procesoru
- V sekci **MCUs/MPUs List** (vpravo dole) vybrat požadovaný typ procesoru. Pokud se používá například deska NUCLEO je ve sloupečku **Board** uvedeno označení NUCLEO-XXXX jako pomocná informace pro výběr správného typu procesoru. Zde je vždy dobré zkontrolovat i tři písmena za číslem procesoru udávající hlavně typ pouzdra (počet pinů) a velikost paměti (FLASH). Velikost paměti RAM a FLASH lze zkontrolovat i podle dalších sloupců tabulky.
- (Pokud je potřeba použít samostatný procesor tak je potřeba kliknout obecně na řádek s procesorem. Pokud se chce použít NUCLEO, aby STM32CubeIDE generoval všechny podpůrné soubory pro NUCLEO je potřeba kliknout na odkaz NUCLEO… do sloupečku „Board“. Pokud není potřeba, tak doporučuji vybírat samotný procesor, nikoli NUCLEO.)
- **Next** (tlačítko vpravo dole)
- Zobrazí se nový dialog pro nastavení projektu
- Zadat název projektu. Doporučuji začít název projektu typem použitého procesoru, potom podtržítko a stručný název projektu (například „STM32F401_TestA“).
- Pod názvem projektu lze zadat umístění projektu. Soubory projektu se vygenerují přímo do zvoleného adresáře, proto je nutno pro projekt vytvořit nejprve adresář (například „STM32F401_TestA“) a pak jej zde vybrat aby do něho byly vloženy projektové soubory. Klidně tedy adresář a projekt nazvat stejným názvem.
- Další položky nastavit / nechat takto:
    - Targeted Language: **C** (ne C++)
    - Targeted Binary Type: **Executable** (ne Static Library)
    - Targeted Project Type: **STM32** (ne Empty)
- **Finish** (tlačítko na spodní hraně dialogu)
- Projekt se bude inicializovat. Bude se zřejmě stahovat potřebná HAL knihovna (její nejaktuálnější verze). Raději počkat až se toto vše vykoná.
- Poté bude zobrazeno pouzdro procesoru se všemi jeho vývody. Kolečkem myši lze přibližovat / vzdalovat.
- Pro činnost procesoru je potřeba vykonat základní nastavení.

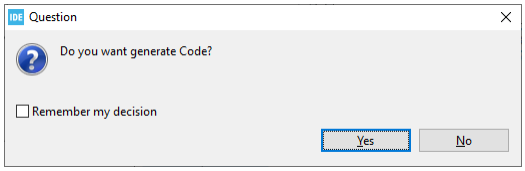
- Povolit možnost ladění / debugování. Přepnout se na záložku **Pinout & Configuration**, v **Categories** rozbalit **Systém Core** a kliknout na **SYS**. V sekci **SYS Mode and Configuration** nastavit **Debug** na **Seriál Wire** (povolení programování / ladění).



- Nastavit oscilátor a frekvenci procesoru. Přepnout se na záložku **Clock Configuration**. Zde zkontrolovat / nastavit následující:
  - (1) Zda je přepínač nastaven HSI RC (High Speed Internal RC) oscilátor.
  - (2) Zda je v přepínači použit výstup fázového závěsu PLLCLK.
  - (3) Zde lze zapsat frekvenci procesoru. Pro začátek doporučuji tu nejvyšší povolenou, tedy tu uvedenou pod tímto políčkem. Po zápisu hodnoty stisknout klávesu **Enter**. STM32CubeIDE nastaví fázoví závěs / násobičku (PLLCLK), tak aby bylo z (pevného) interního oscilátoru dosaženo požadované frekvence. Pokud se nepodaří (násobením / dělením interního oscilátoru) dosáhnout požadované frekvence, tak je hlášena chyba a je nutno zapsat jinou požadovanou / cílovou frekvenci.



- Nakonec projekt uložit klikem na ikonu diskety / disket v levém horním rohu aplikace. Bude zobrazen dialog pro potvrzení:

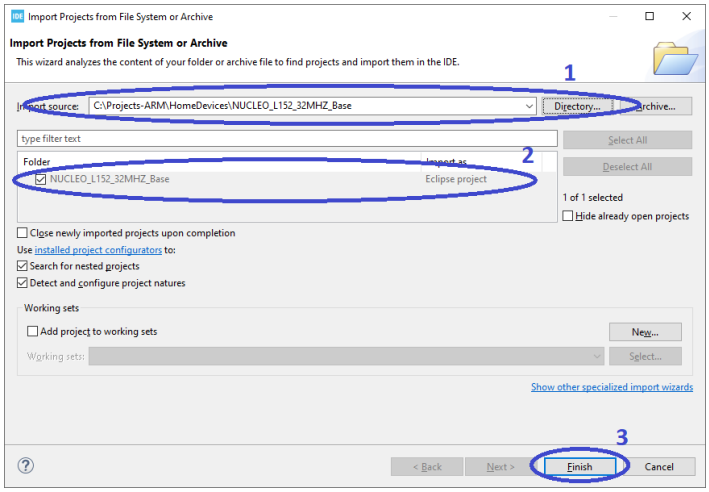


- Po potvrzení bude vygenerován projekt do zadaného umístění / adresáře.

3 Import / otevření již existujícího projektu

Pokud je potřeba přidat do STM32CudeIDE již existující projekt uložený na disku, lze postupovat následovně:

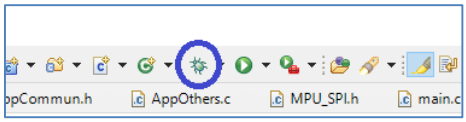
- **File** (menu vlevo v záhlaví aplikace)
- **Open projects From File Systém …** (položka v rozbalovacím menu)
- Zobrazí se dialog pro výběr umístění a projektu.



- Vybrat adresář kde jsou projektové soubory (1)
- Označit název projektu pro import / otevření (2)
- Stisknout **Finish** pro import / otevření projektu (3)
- Přidaný projekt se zobrazí v seznamu projektů (nejčastěji na levé straně).

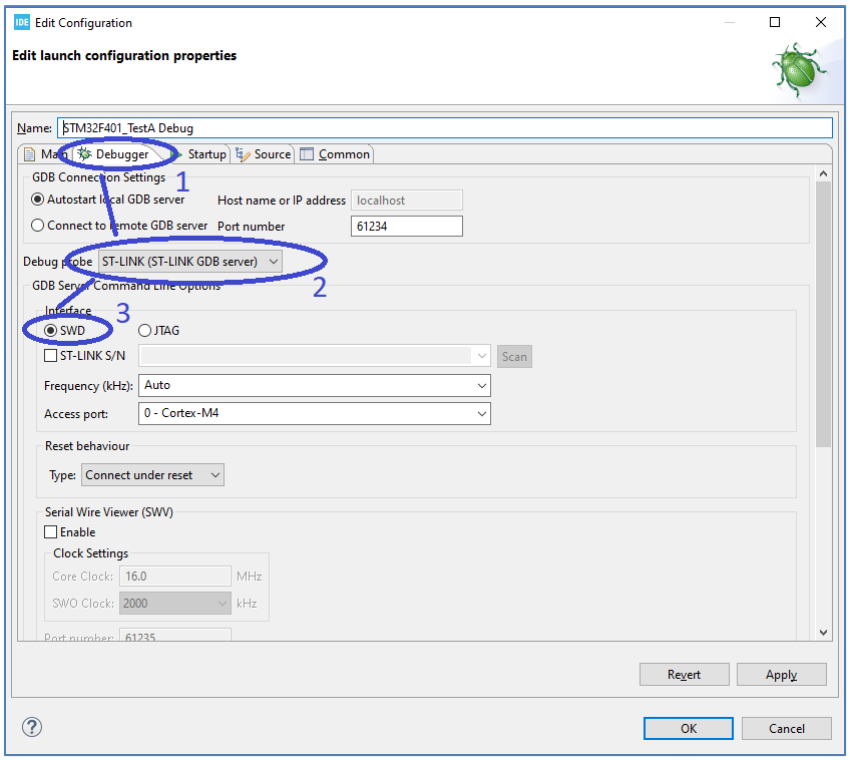
4 Nastavení programátoru a ladění

Než se začne programovat, je potřeba nastavit do projektu jaký se bude používat programátor a debagátor k ladění aplikace. Po vytvoření nového projektu je nejjednodušší stisknout tlačítko pro ladění programu, ikona zeleného brouka v horní části:



Protože není nastaven programátor, tak se zobrazí dialog pro jeho nastavení. Pokud se používá nějaká vývojová deska od STM obsahující programátor a je v ní zachovám SW z názvem ST-Link, tak v podstatě stačí tento dialog pouze potvrdit. Je však dobré zkontrolovat jeho nastavení:

- **Debugger** (záložka na dialogu)
- **Debug probe** nastavit na **ST-LINK (ST-LINK GDB server)**
- **Interface** nastavit na **SWD**



Potvrdit nastavení tlačítkem **Apply** a poté **OK**. Tento dialog lze kdykoli vyvolat pomocí:

- **Run** (menu v záhlaví aplikace)
- **Debug Configurations** … (položka v rozbaleném menu)

Zde je vše dobře nastaveno lze ověřit trasováním programu:

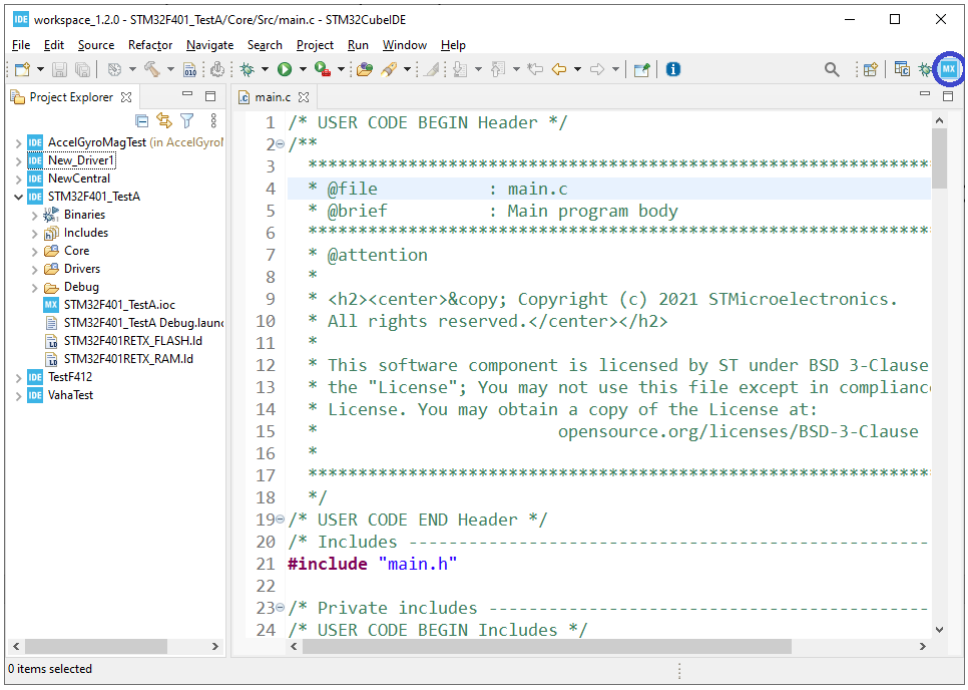
- Spustit program pomocí zeleného brouka
- Program se zastaví na řádku s „HAL_Init()“
- Pomocí žlutých šipek (v horní části aplikace) pro krok lze vykonat jeden řádek programu.

Poznámky:

- Tento postup je určen pro programátor **ST-Link** (od STM). Pokud je použít programátor **J-Link** (od SEGGER), je princip zcela stejný, instalovat SW pro **J-Link** od SEGGER a v záložce **Debugger** v položce **Debug probe** nastavit **J-Link**.

5 Přidání adresáře a souborů

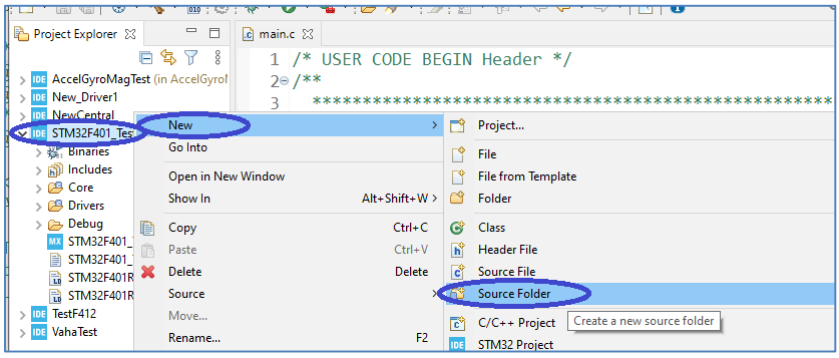
Do projektu je vždy potřeba přidat vlastní adresáře a soubory s programovým kódem. STM32CubeIDE se občas přepíná do až „podivných“ konfigurací. Pro jednoduchost je dobré mít seznam projektů a jejich adresáře a soubory na levé straně takto (pokud jsou na pravé straně tak spousta věcí nepracuje, jak je zde uvedeno):



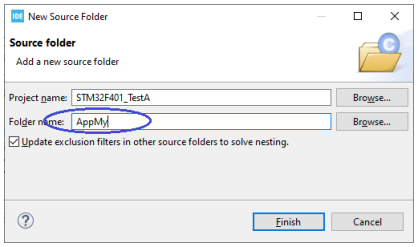
Pokud tomu takto není lze většinou tohoto zobrazení dosáhnout pomocí modrého tlačítka **MX** v levém horním rohu aplikace.

Přidání adresáře pro zdrojové soubory je následující:

- Pravé tlačítko myši na projekt v seznamu projetu. Zobrazí se lokální menu.
- **New** (položka v lokálním menu)
- **Source Folder** (položka v lokálním pod-menu) Pozor nevybírat **Folder**, ten není určen pro zdrojové soubory.



- Zobrazí se dialog pro zadání názvu adresáře.

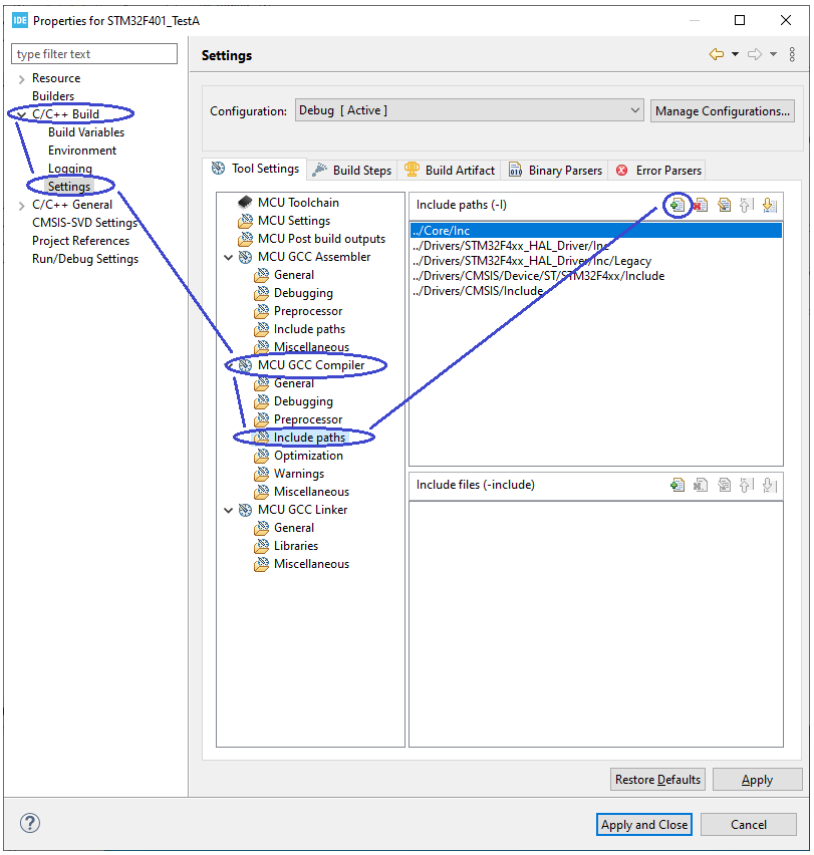


V každém projektu doporučuji vytvořit adresář **App** a zde vkládat vlastní zdrojové soubory. Toto je velmi vhodné při pozdějším přechodu na jiný typ procesoru nebo při potřebě znovu-vytvoření projektu. Stačí tedy tento adresář vzít a vložit do nového projektu. Adresáře lze samozřejmě i vnořovat.

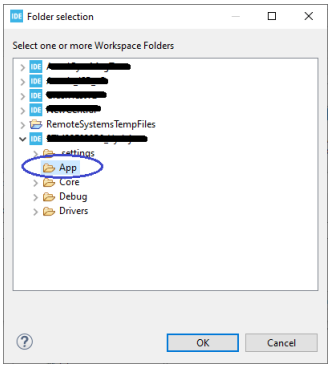
Vložení zdrojových souborů typ „.h“ a „.c“ je zcela stejné pouze je nutno ze zobrazeného lokálního pod-menu vybrat položky **Header File** nebo **Source File**. Klik pravým tlačítkem na cílový adresář pro vytvoření souborů (ne na název projektu) a názvy souborů je nutno zapsat i s příponou (tedy s „.h“ nebo „.c“).

V projektu je potřeba nastavit cesty kde jsou (zejména) hlavičkové soubory. Ty lze nastavit buď pomocí plného zadání cesty v direktivě „include“ například „#include "MyApp/MyFile.h"“ nebo nastavením cesty do adresáře v rámci projektu, což doporučuji jako pro práci jednodušší. Postup je následující:

- Pravé tlačítko na projekt v seznamu projektů
- **Properties** (položka v lokálním menu, většinou až poslední dole)
- Zobrazí se dialog konfigurace projektu.
- Rozkliknout / rozbalit skupinu **C/C++ Build** (v seznamu na levé straně)
- Kliknout na **Settings** (v rozbalené skupině)
- **Tool Settings** (záložka ve středu dialogu)
- Ve skupině **MCU GCC Compiler** kliknout na **Include paths**
- Vlevo se zobrazí seznam cest již nastavených v projektu.



- Stisknout tlačítko listu papíru se zeleným **+**. Zobrazí se dialog pro zadání nové cesty v rámci projektu.
- Stisknout **WorkSpace…** na dolním okraji dialogu. Cesta bude do našeho projektu v našem workspace. (Raději nepoužívat „File systém …“, pak dojde k závislosti na umístění projektu na disku.)
- Vybrat požadovaný adresář a stisknout **OK**.



- Poté ještě **OK** na původním dialogu. Dále **Apply** a **Apply and Close** na dialogu konfigurace projektu.
- Nově přidaná cesta se zobrazí v seznamu již vytvořených cest a je relativní v rámci WorkSpace.

Nyní lze do adresáře **App** vkládat zdrojové soubory („.c“ / „.h“) a kdekoli jinde používat pouze „include“ bez udání celé cesty. Kompilátor při hledání hlavičkových souborů bude procházet i tento adresář.

6 Možnosti tvorby projekty

Vlastní kód lze v projektu vytvářet v podstatě dvěma způsoby:

- A) Pomocí GUI nastavení v STM32CubeIDE, kdy je kód automaticky generován.
- B) Pomocí vlastního zápisu programového kódu.

První způsob se zdá na první pohled ideální, ale zdaleka není. Automaticky generovaný kód je „rozstrkán“ po mnoha souborech v projektu, což by ani tak nevadilo, ale v podstatě jej nelze přenést do jiného projektu (člověk jen tak neví, do jakého souboru bylo co vygenerováno a tedy co je potřeba kopírovat). Pokud je potřeba přejít na jiný procesor (například větší FLASH, více pinů) a jedy potřeba projekt generovat znova, tak je také potřeba pomocí GUI opět vykonat všechny nastavení pro automatické generování kódu (u většího projektu velmi náročná práce). Vyjmutí původního a přenesení do nového projektu zvládne pouze člověk dostatečně znalý struktury projektu a obsahu jeho souborů.
Avšak pro prvotní pokusy / seznámení není tento způsob programování špatný, protože nejsou potřeba velké znalosti v nastavení nějakých parametrů procesoru. Rovněž je vhodný, pokud například vlastní kód pro ovládání nějaké periferie nepracuje, tak lze automatickým generováním kódu pomocí

GUI činnost periferie ověřit a podívat se do zdrojových souborů (pokud uživatel ví, kde se generovaný kód nachází) co a jak je potřeba nastavit.
Tento přístup je celkem vhodný například pokud se programový kód nebude používat v žádném jiném projektu.

Velmi však doporučuji druhý způsob a to vlastní tvorbou programového kódu (zejména podle příkladů dostupných na WWW). Tento programový kód však ukládat (pokud to lze tak pouze) do vlastních adresářů. Při využití HAL knihoven je programový kód velmi dobře přenositelný mezi různými typy procesorů. Lze tedy vytvořit například vlastní soubory pro obsluhu různých periferii (I2C/SPI/…) obsahující vlastní funkce a ty poté snadno používat v dalších projektech (není je potřeba při novém projektu opět graficky nastavovat a generovat).

7 Doporučené uspořádání vlastních souborů

Generovaný projekt obsahuje spoustu adresářů a souborů, jejich podrobnější význam je pospán v další kapitole (kapitolách). Tato kapitola popisuje pouze souboru dodané podle aplikace a doporučuje jak je používat (adresáře i soubory).
Vytvořená aplikace (projekt) používá HAL knihovny, které jsou na HW daného procesoru velmi nezávislé, vytvořený projekt lze s různě velkým úsilím (od žádného až po …) přenést na jiný typ STM32 procesoru. Základ nového projektu (povolení SWD, nastavení hodin, …) je vždy nejlepší vytvořit pomocí grafického návrháře v prostředí STM32CubeIDE. Pak je ideální již existující zdrojové souboru původního projektu do nového pouze nakopírovat. Pro toto je vhodné dodržet nějaké ustanovení. Základní doporučení je tedy v projektu vytvořit tyto pod-adresáře (v hlavním adresáři projektu):

- **App** – Zde vkládat všechny zdrojové soubory pro vytvářenou aplikaci. Soubory určené (pouze) pro tuto aplikaci, tedy ty vytvářející činnost této aplikace.
- **Utils** – Soubory polo-univerzální sdílené pouze mezi několika projektu, například projekty stejného typu. Tedy definice pro komunikaci v projektu, vytvoření protokolu a další.
- **XUtils** – Soubory univerzální sdílené mezi aplikacemi. Tyto soubory obsahují například obsluhu obecných komunikací / protokolů, pomocné funkce a různé definice usnadňující programování. Zde je v podstatě vše nezávislé na konkrétním projektu.

V pod-adresáři **App** je vhodné vytvořit nějaké základní soubory (tyto soubory zejména zjednodušují práci na projektu):

- **Def0Global.h** (pouze hlavičkový soubor) – Soubor obsahující „include“ všech (vlastních) souborů použitých v / přidaných do projektu. Tento soubor je vhodné „includit“ do každého vlastního souboru v projektu (místo mnoha konkrétních souborů). Tento soubor je automaticky „includěn“ do všech souborů v adresáři **Utils** a **XUtils**. Soubor může rovněž obsahovat nějaké základní nastavení pro projekt.
- **Def0Project.h** (pouze hlavičkový soubor) – Nastavení pro projekt, zejména podmíněné kompilace (například verze aplikace) nebo různé parametry (například pro komunikaci).
- **Def0HW.h** (pouze hlavičkový soubor) – Definice HW pro projekt (například piny) pokud je jich málo, jinak je rozděleno do samostatných souborů podle částí projektu.

- **AppMain.h** / **AppMain.c** (hlavičkový / zdrojový soubor) – Hlavní soubory aplikace / projektu. V tomto souboru je hlavní smyčka aplikace a jsou z něho (převážně) volány všechny ostatní soubory v projektu (jejich funkce).
- **AppInit.h** / **AppInit.c** (hlavičkový / zdrojový soubor) – Pokud je složitá inicializace zařízení a nevejde se pro přehlednost do funkce **AppMainInit()** v souboru **AppMain.c** tak je ji vhodné umístit do takovéhoto samostatného souboru (soubor může samozřejmě obsahovat několik inicializačních funkcí).
- **AppOthers.h** / **AppOthers.c** (hlavičkový / zdrojový soubor) – Zejména pomocné funkce projektu.

Soubory jsou označeny **Def0…** jen proto, aby byly vhodně seřazeny v zobrazení projektu. Další vlastní soubory je vhodné nazvat **Def…** (například DefNFC.h / DefSPI.h / …).
Struktura souboru **Def0Global.h** je (zhruba) následující:

```
#ifndef DEF0GLOBAL_H_
#define DEF0GLOBAL_H_

// hlavni pro pouzity procesor (STM32F030)
// (je vhodné vždy uvést, pripadne dalsi z HAL knihoven)
#include "stm32F0xx_hal.h"

// definice / nastaveni pro projekt
#include "Def0Project.h"

// definice pro casti projektu (HW definice částí projektu)
… (#include "DefNFC.h")

// XUtils (pouzite soubory v adresari XUtils)
… (#include "XUtils00.h")

// hlavickove soubory projektu / aplikace (ty z adresare App)
#include "AppMain.h"
…

#endif /* DEF0GLOBAL_H_ */
```

V soubory **Def0Project.h** a **Def0HW.h** (pokud je použit) mohou mít zcela vlastní obsah.

Základní obsah souboru **AppMain.h** je následující (obsahuje pouze dvě funkce pro vstup do vlastní aplikace, vysvětleno dále):

```
#ifndef APPMAIN_H_
#define APPMAIN_H_

void AppMainInit();  // inicializace pred 'Loop'
```

```
void AppMainLoop();  // volano z hlavni 'Loop'

#endif /* APPMAIN */
```

Obsah souboru **AppMain.c** bude následující:

```
#include "Def0Global.h"

// prvotni inicializace pred 'Loop'
void AppMainInit()
{
…
}

// volano z hlavniho 'Loop'
void AppMainLoop()
{
…
}
```

V jednoduchém projektu je potřeba zasáhnout pouze do souboru Core/Src/main.c generovaného pomocí STM32CubeIDE. Do tohoto souboru je potřeba přidat následující řádky (červené):

```
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "Def0Global.h"
/* USER CODE END Includes */

…

/* Infinite loop */
 /* USER CODE BEGIN WHILE */
 AppMainInit();
 while (1)
 {
  AppMainLoop();
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
 }
 /* USER CODE END 3 */
```

Doporučuji více nezasahovat do souboru **main.c** a vlastní programový kód ukládat pouze do vlastních souborů v adresářích **App** / **Utils** / **XUtils**. Ze souboru **main.c** se tedy volají pouze funkce a to **AppMainInit()** a **AppMainLoop()** čímž se kód vlastní aplikace stává velmi nezávislý na generovaném

projektu. Funkce **AppMainInit()** se tedy volá pouze jednou na začátku běhu aplikace a funkce **AppMainLoop()** se volá opakovaně z hlavní smyčky umístěné v původním souboru **main.c** Funkce **AppMainLoop()** může samozřejmě obsahovat přímo nekonečnou smyčku, aby se zamezilo jejímu stále opakovanému volání z **main.c**.

Poznámky:

- Názvy souborů v adresáři **App** doporučuji začínat pomocí **App…** (například AppCommun.c/.h, AppSettings.c/.h, …). Z jejich názvu je zřejmý vztah k aplikaci / projektu, nikoli například k HAL knihovnám.
- Funkce v souborech v adresáři **App** by mohlo být vhodné začínat rovněž na **App…** (například AppCommunSend(…) nebo AppCommunReceive(…)), ale toto není to podmínkou.
- Každá část projektu umístěná v souborech **App…** (například AppCommun.c/.h, AppSettings.c/.h, atd.) by měla mít vlastní inicializační funkci (například AppCommunInit(), AppSettingsInit(), …). Tyto funkce pro inicializaci jednotlivých částí projektu by měly být vždy volány z funkce **AppMainInit()** v souboru **AppMain.c**.
- Soubory **AppInit.h** / **AppInit.c** jsou zcela na vlastním uvážení. Jsou vhodné v případě kde, je jedna velká inicializace pro celý projekt a není vhodné rozdělovat inicializaci do souborů podle částí projektu (například do AppCommun.c/.h, AppSettings.c/.h, …).

8 Struktura generovaného projektu

Lze si všimnout, že všechny soubory projetu tedy i knihovní jsou ve zdrojové formě. To je z těchto hlavních důvodů:

- Při hledání chyby je často vhodné trasovat program i do kódu knihovních souborů.
- Jsou případy kde je (skutečně) nutno do několika souborů generovaných při vytvoření projektu zasáhnout a něco v nich upravit / změnit.

Projekt obsahuje tyto výchozí adresáře:

- **Core** (adresář) – Obsahuje programové kód pro nějaké základní nastavení / spuštění procesoru a v podstatě prázdnou programovou smyčku / aplikaci.
- **Drivers** (adresář) – Obsahuje soubory / knihovny dodávané (zejména) od STM. Zde jsou i soubory pro knihovny HAL.
    - CMSIS – Základní definice a funkce pro použitý typ procesoru.
    - STM32…_HAL_Driver – Zdrojové soubory HAL pro použitý typ procesoru.

Význam některých souborů v projektu:

- **Core/Src/main.c** – Volání funkcí pro základní nastavení procesoru a prázdná programová smyčka. Obsahuje funkci pro nastavení oscilátoru a frekvence procesoru (podle nastavení v GUI při vytváření projektu).
- **Core/Src/stm32…_hal_msp.c** – Inicializace různých periferií (zejména při jejich konfiguraci přes GUI).
- **Core/Src/stm32l4xx_it.c** – Obsluhy přerušení.
- **Core/Inc/stm32l4xx_hal_conf.h** – Konfigurace jako části HAL jsou použity (podrobněji v kapitole o HAL knihovnách).

- **NazevProjektu.ioc** – Při spuštění (poklepání) tohoto souboru je spuštěn GUI návrhář.

(Popis dalších soubor bude doplněn později podle potřeby.)

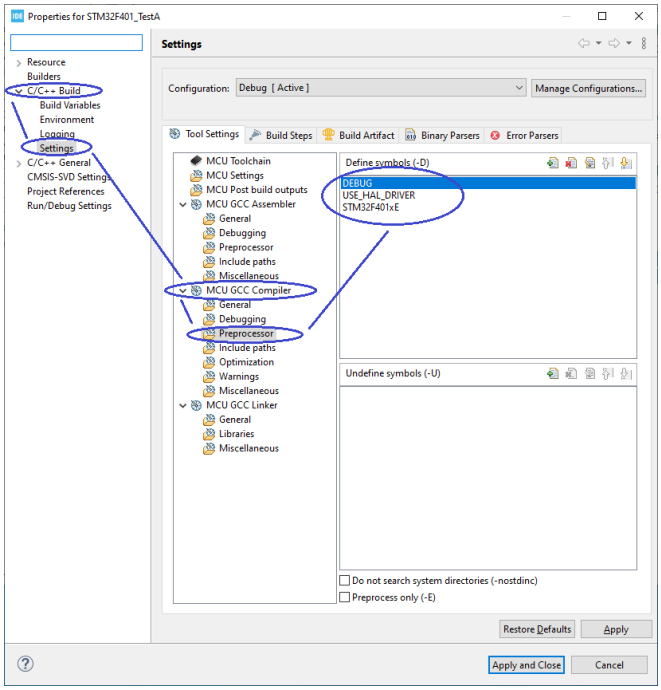
9 Knihovny HAL

Každý projekt používá nějaké knihovny, jejichž funkce jsou volány programem uživatele a tím je tedy vytvořena činnost systému. U STM se tyto knihovny jmenují HAL, což znamená „Hardware Abstraction Layer“, tedy jakási abstrakce nad HW zařízením / procesorem.

Tyto knihovny jsou různé pro různé typy procesorů, například:
- Pro STM32L4… jsou HAL knihovny označené STM32L4xx_HAL_Driver
- Pro STM32L0… jsou HAL knihovny označené STM32L0xx_HAL_Driver
- Pro STM32F4… jsou HAl knihovny označené STM32F4xx_HAL_Driver
- A tak dále.

Jejich velká výhoda však spočívá ve velmi jednotném volání funkcí, i když programový kód uvnitř funkce je pro každý typ procesoru poněkud jiný. V naprosté většině případů funkce nemají parametry (pro různé procesory by byly jiné nebo i jiný počet), ale jako parametr do funkce vstupuje struktura popisující jak se má daná funkce zachovat. Tato struktura obsahuje nějaké základní položky společné všem typům procesorů (například pořadové číslo I2C) a podle určitého typu procesoru i speciální položky (například náběžná / spádová hrana pro I2C).
Při přenosu vytvořeného programového kódu na jiný typ procesoru lze nepodporované parametry zakomentovat a potřebné přidat nebo použít podmíněný překlad pro určitý typ procesoru.

Když se vytvoří projekt tak je vždy definována konstanta zda je použita HAL knihovna a typ použitého procesoru. Definované konstanty lze zjistit klikem pravého tlačítka na projekt a vyvoláním **Properties** (položku v lokální menu, poslední). Takto definované konstanty lze samozřejmě přidat i zcela vlastní do jejich seznamu.



Lze tedy pro podmíněný překlad použít například:

```
#if defined(STM32F401xE)
…
#elsif defined(STM32F103xB)
…
#endif
```

I když je HAL knihovna určena vždy pro celou jednu řadu procesorů, například **STM32L4xx_HAL_Driver** pro **STM32F4xx**, tak je v jejich souborech spousta podmíněného překladu podle ještě konkrétnějšího typu (počet pinů, počty periferií, typy periferií, …). Konkrétní typ použitého procesoru (nebo stejná skupina) je právě definována jako konstanta v projektu a při kompilaci se překládá pouze ten kód HAL knihoven určený právě pro tento typ procesoru (rovněž proto jsou HAL knihovny ve zdrojových souborech).
Pro každou část procesoru (řízení hodin, periferie, řízení spotřeby, …) jsou určeny příslušné soubory z HAL knihovny. Pro konfiguraci HAL knihoven je důležitý soubor **Core/Inc/stm32l4xx_hal_conf.h**. V tomto souboru se povolují jaké části / soubory HAL knihovny budou do projektu zahrnuty. Pokud je tedy potřeba použít například I2C periferii / komunikaci je potřeba odkomentovat řádek:

#define HAL_I2C_MODULE_ENABLED

Tím se do kompilace projektu zahrnou i soubory z HAL knihovny určené pro obsluhu I2C. Ovšem při povolení například I2C a následné kompilaci projektu dojde k hlášení chyby, že nějaký soubor HAL knihovny není dostupný. Když se projekt vytváří tak se do projektu přidají pouze typ soubory z HAL knihovny, které jsou potřeba, ostatní nikoli (zrychluje se kompilace). Chybějící soubory je tedy nutno do projektu přidat ručně (při generování kódu pomocí GUI jsou do projektu samozřejmě přidány automaticky).

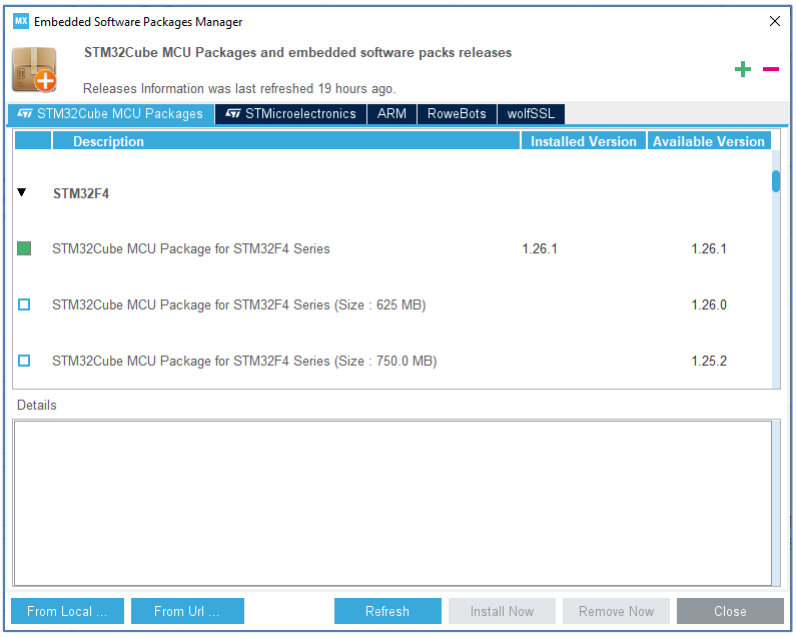
HAL knihovny se rovněž vyvíjí a jsou tedy dostupné v různých verzích. **!!! Vždy je potřeba do projektu kopírovat soubory pouze stejné verze HAL knihovny. !!!** Z tohoto důvodu je nutno vědět jaká verze HAL knihovny byla použita při generování projektu.

(jak to zjistit?)

Knihovny HAL lze stahovat od STM pomocí STM32CubeIDE. Pokud se zakládá / generuje nový projekt, tak se STM32CubeIDE vždy podívá a v případě potřeby stáhne nejnovější verzi HAL knihovny pro použitý procesor. Stažené HAL knihovny se ukládají do adresáře: „C:\Users\...\STM32Cube\Repository\“. Stažení požadované verze HAL lze vykonat i manuálně následovně:

- **Help** (menu v záhlaví aplikace)
- **Manage Embedded Software Packages** (položka v rozbalovacím menu)

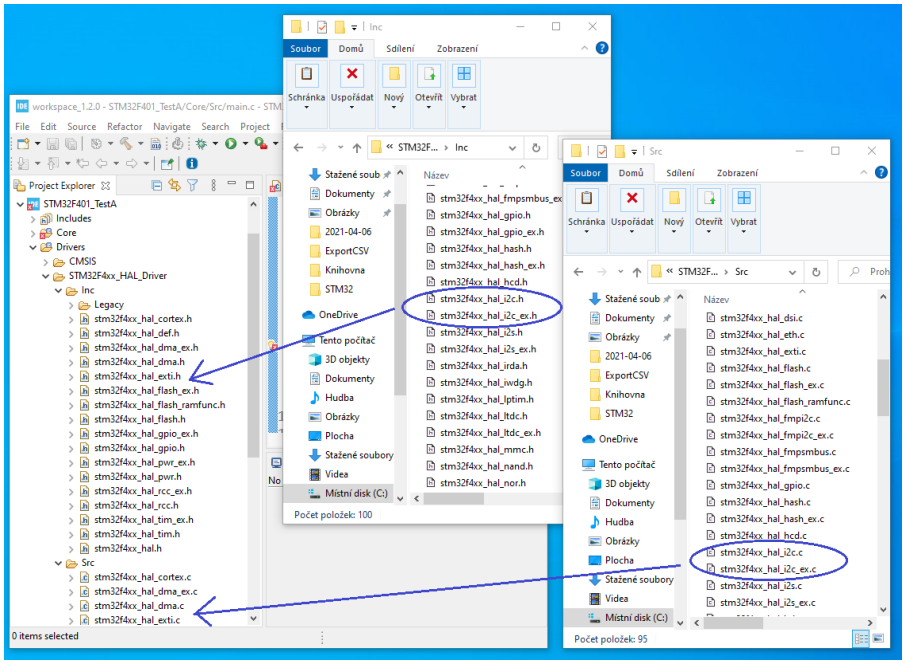
Zobrazí se dialog umožňující prohlížet jaké jsou dostupné verze HAL knihoven, pro jaké typy procesorů jsou dostupné nebo již stažené. Z obrázku je zřejmě, že HAL knihovna verze 1.26.2 pro STM32F4xx je již stažená. Existují však i starší verze této knihovny. Pokud je potřeba nějakou uvedenou verzi stáhnout, stačí ji zatrhnout a stisknout **Install Now** (tlačítko ve spodní části dialogu). Pokud je potřeba nějakou staženou verzi odinstalovat, stačí ji od-zatrhnout a stisknout **Remove Now** (tlačítko ve spodní části dialogu). Některé knihovny nejsou zrovna po rozbalení malé a na malém SSD mohou zabírat spoustu místa, pokud se již napoužívají.



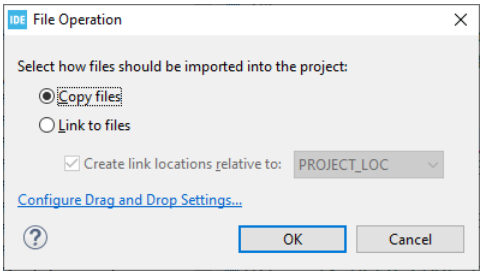
Pokud je tedy potřeba přidávat do existujícího projektu další HAL soubory (například z důvodu použití dalších periférií) je potřeba mít požadovanou HAL knihovnu (obsahující potřebné soubory) nejdříve staženou / instalovanou.

Příklad vložení dalších HAL souborů do projektu. Pokud se například povolí I2C odkomentováním řádku **#define HAL_I2C_MODULE_ENABLED** v souboru **Core/Inc/stm32l4xx_hal_conf.h** je potřeba do projektu přidat HAL soubory pro obsluhu I2C.
Řešení je následující (nevím zda existuje nějaké jednodušší):

- V cílovém projektu zobrazit / rozbalit adresáře **Drivers/STM32…_HAL_Drivers/Inc** a **Drivers/STM32…_HAL_Drivers/Inc**.
- Pomocí průzkumníku otevřít adresář umístění stažené HAL knihovny v **C:\Users\.\STM32Cube\Repository\(VerzeHAL)\Drivers\STM32F..._HAL_Driver\Inc** a **C:\Users\.\STM32Cube\Repository\(VerzeHAL)\Drivers\STM32F..._HAL_Driver\Src**.
- Najít všechny soubory obsahující název STM32…_HAL_i2c (typu „h“ i „c“).
- Přetažením zkopírovat soubory z průzkumníka do STM32CubeIDE (soubory z adresáře Inc do Inc a soubory z adresáře Src do Src).



- Při dotazu vybrat kopírovat soubory. Nedoporučuji dělat link, soubory je lepší mít přímo v projektu.



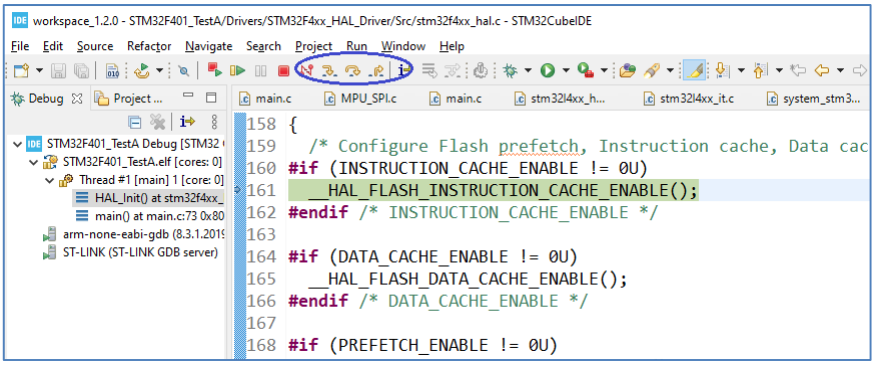
- Stejný princip platí pro všechny jiný typy částí procesoru.

10 Kompilace projektu

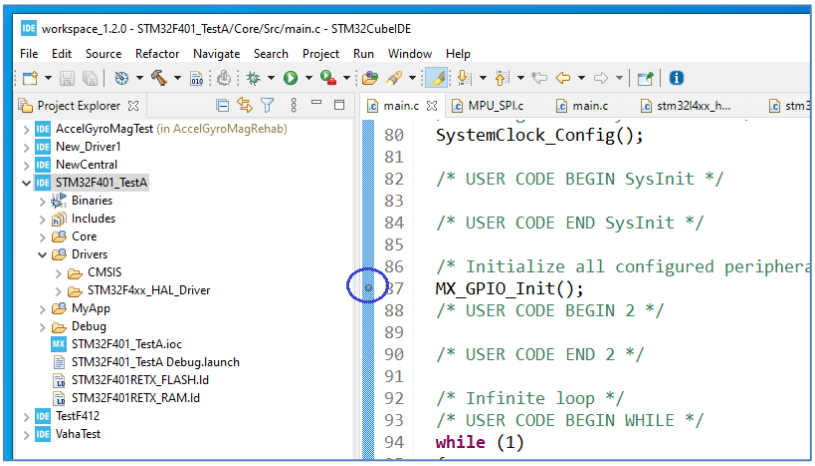
Pokud je potřeba nějaký projekt v seznamu projektů (na levé straně) zkompilovat, stačí na jeho název kliknout pravým tlačítkem a z lokálního menu vybrat **Build Project**. Stejné činnosti lze dosáhnout i stiskem ikony hnědého kladívka záhlaví aplikace.
Projekt by měl být rovněž automaticky zkompilován při jeho spuštění pomocí ikony celeného brouka v horní části aplikace.

11 Ladění programu

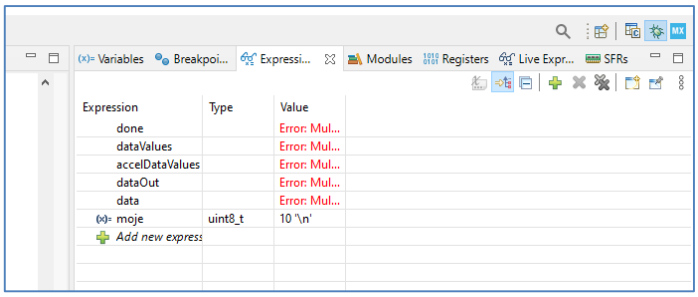
Ladící běh programu se spouští pomocí ikony zeleného brouka v horní části aplikace. Spuštěný program lze zastavit pomocí ikony červené stopky v horní části aplikace (červeně zobrazena pouze při běhu aplikace). Pouze pozastavení vykonávání programu se provede ikonou žluté stopky (žlutě zobrazena pouze při běhu aplikace). Program lze korkovat pomocí žlutých šipek (vstup do funkce, běžný krok, výstup z funkce).



Při dosažení BreakPointu lze běh programu dále spustit pomocí zelené šipky. Do programu lze v podstatě na libovolné místo vložit BreakPoint pomocí dvou-kliku na tlustější čáru vlevo od čísel řádků programového kódu. Program se při běhu na tomto řádku automaticky zastaví.



Při zastavení programu si lze zobrazit obsah všech proměnných použitých v programu pouhým najetím myší na název proměnné. Pokud je potřeba mít obsah proměnné zobrazen stále je vhodné ji zapsat do záložky **Expressions** na pravé straně (nejprve klikem na **+Add new expression** a poté zápisem jejího názvu do vytvořeného prázdného políčka).


Variables
Breakpoi...
Expressi...
Modules
Registers
Live Expr...
SFRs
Expression
Type
Value
done
Error: Mul...
dataValues
Error: Mul...
accelDataValues
Error: Mul...
dataOut
Error: Mul...
data
Error: Mul...
moje
uint8_t
10 '\n'
Add new express