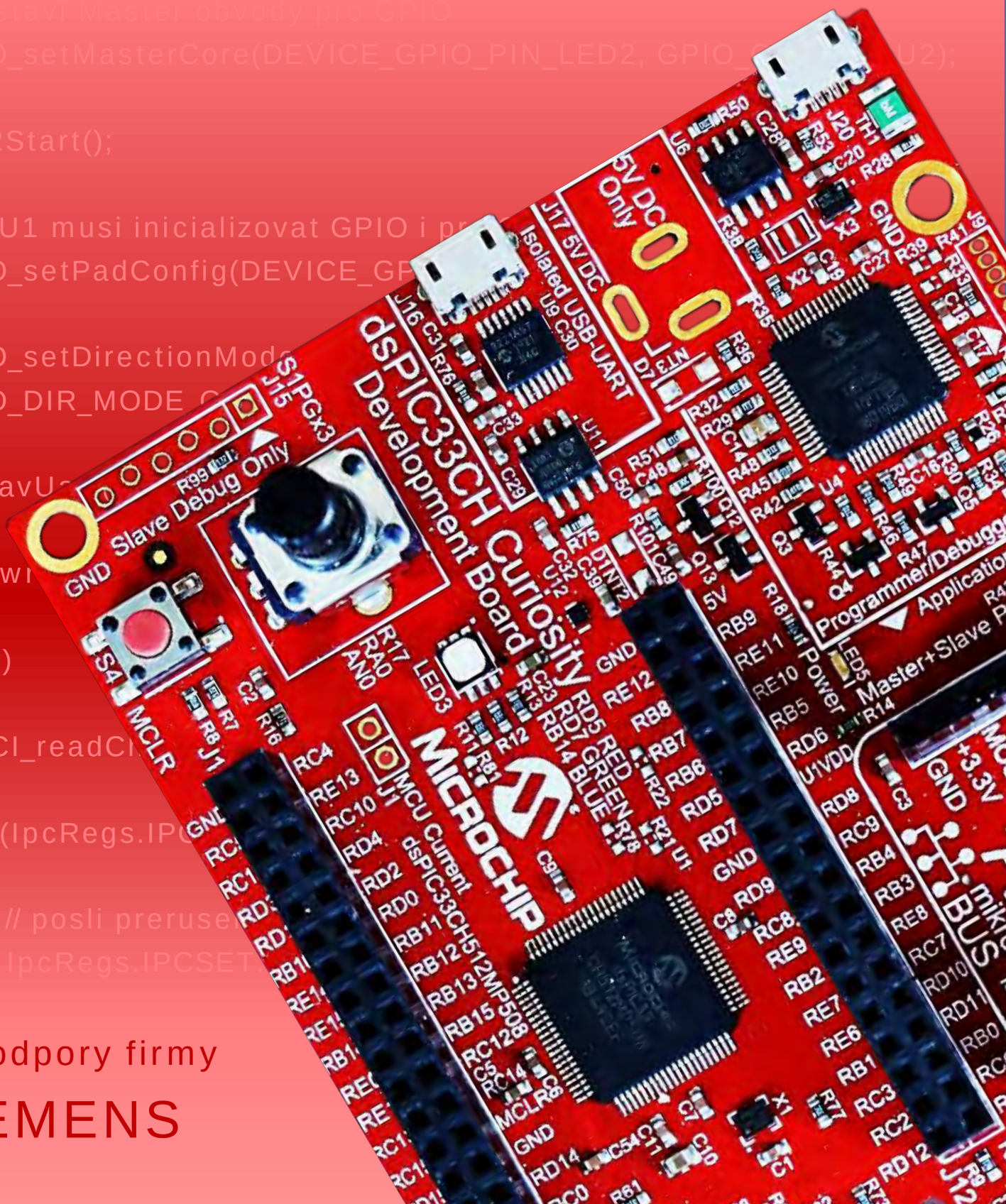
Programujeme d**voujádrové** kontrolery

Ing. Vojtěch Skřivánek

```
// nastavi Master obvody pro GPIO
GPIO_setMasterCore(DEVICE_GPIO_PIN_LED2, GPIO_CORE_CPU2);

cpu2Start();

// CPU1 musi inicializovat GPIO i pro
GPIO_setPadConfig(DEVICE_GP

GPIO_setDirectionMode
GPIO_DIR_MODE_

nastavU

SCI_wr

for(;;)
{
   SCI_readCh

   if((IpcRegs.IPC
   {
      // posli preruse
      IpcRegs.IPCSET
```



za podpory firmy

SIEMENS

Poděkování

Děkuji firmě

SIEMENS

za podporu při psaní této knihy.

Vydání první 2025

ISBN: 978-80-7539-213-8 (PDF verze)

Obsah

Kapitola 1

Úvod

V dnešním světě technického rozmachu jsou požadavky na výkon a energetickou šetrnost elektronických zařízení stále vyšší. Důvody jsou například neustále se zvyšující standardy zákazníků, či integrace umělé inteligence tam, kde ještě donedávna nebyla. A tak se v praxi stává, že jednojádrové mikrokontrolery již nejsou dostatečně výkonné. Z tohoto důvodu se na trhu stále častěji objevují kontrolery, které obsahují více než jednu procesorovou jednotku.

Výhody vícejádrových kontrolerů jsou zřejmé.

- První výhodou je poměr ceny a výkonu. Jeden vícejádrový kontroler je levnější varianta než více jednojádrových. Důvodem je, že jsou obě jádra na jednom čipu, nezabírají tedy tolik místa na plošném spoji, který tedy může být menší. Komunikace mezi procesory je rychlejší. Jádra sdílejí paměti a periferie, kterých je zpravidla dost na to, aby pokryly naše požadavky, takže neplýtváme zdroji jako v případě více plně vybavených jednojádrových kontrolerů.
- Druhou výhodou je poměr výkonu a spotřeby energie. Efektivita kontroleru se snižuje, čím více se jeho taktovací kmitočet blíží jeho maximální hranici. Řešením je tedy vícejádrový kontroler, jehož procesorové jednotky pracují na nižších kmitočtech, kde je jejich efektivita vysoká. Toto řešení je energeticky výhodnější než náhrada kontrolerem s vyšším maximálním kmitočtem, nebo více separovanými kontrolery.
- Další výhodou je lepší separace programu. Více jader poskytuje možnost rozdělit program například na časově kritickou část, vyžadující vyšší výkon, a udržovací "pomalý"program. Také je možné použít kombinaci více operačních systémů. Program je možné rozdělit na části, které musí či nemusí splňovat požadavky na bezpečnost. Více jader umožňuje rychleji zpracovávat větší množství přerušení. Více separovaných programů je snazší ladit. Všechny tyto výhody mají i oddělené mikrokontrolery, avšak, jak již bylo řečeno, vícejádrový kontroler toto dokáže efektivněji.

Častým využitím druhého jádra kontroleru je jeho použití k implementaci komunikačních protokolů, jako jsou například Bluetooth, Wifi nebo Modbus komunikace.

Abychom nebyli jednostranní, práce s vícejádrovými kontrolery má i své neduhy.

- Pokud nejsou jádra dobře spravována, může být jich energetická spotřeba vyšší.
- Pokud není potenciál jader naplno využit, jde o dražší řešení, než je použití výkonného jednojádrového kontroleru.
- Ladění programu může být v případě meziprocesorové komunikace náročnější.
- Správa paměti a návrh programu může být komplikovanější.

Jak je vidět, vícejádrové kontrolery nabízí výhody i nevýhody. Velmi záleží na jejich použití pro daný účel. Jedno je ale jisté - v dnešním světě kladoucím stále větší důraz na výkon, budou tyto kontrolery stále častěji používány ve vestavěných systémech.

1.1 Motivace knihy

Tato kniha se zabývá dvoujádrovými kontrolory, a to především proto, že se ve vestavěných systémech, hned po jednojádrových, vyskytují nejčastěji.

Kniha si klade za cíl čtenáře seznámit s běžnými dvoujádrovými mikrokontrolery, které jsou dostupné na trhu. Kniha zdaleka nepokrývá všechny varianty kontrolerů, ale zaměřuje se na cenově dostupné kontrolery k běžnému využití dodávané největšími výrobci kontrolerů.

Čtenář se o každém v této knize uvedeném kontroleru dozví, jakou má architekturu, jaké možnosti meziprocesorové komunikace nabízí, jak je využívat a jaké zdroje mu mohou pomoci při tvorbě vlastního programu.

Kniha se zabývá především specifickými nástroji meziprocesorové komunikace každého kontroleru. Obecné rady a postupy programování aplikací pro vícejádrové kontrolery nejsou náplní této knihy.

1.2 Struktura knihy

Kniha obsahuje celkem pět hlavních kapitol.

Každá z nich se zabývá jedním dvoujádrovým kontrolerem. Každý kontroler je vyráběn jinou firmou, aby byly vidět rozdíly v architektuře kontrolerů, vývojových prostředích, založení projektu, psaní programu a hlavně možnostech meziprocesorové komunikace.

V každé z této kapitol je nejprve popsán kontroler a vývojová deska, se kterou se pracuje v příkladech. Všechny vývojové desky je možné běžně zakoupit na internetových obchodech prodávající elektrotechnické součástky. Byly vybírány tak, aby byly cenově dostupné. Kontrolery byly voleny na základě dostupnosti a výbavy meziprocesorové komunikace.

Dále je popsána instalace a použití vývojového prostředí, které je využito pro tvorbu příkladů.

Samostatná podkapitola je věnována založení projektu pro dvoujádrový kontroler v daném vývojovém prostředí. Jak zjistíme, ne vždy je taková věc jednoduchá.

Poslední nejobsáhlejší podkapitola se již zabývá nástroji meziprocesorové komunikace daného kontroleru. Po výčtu všech nástrojů jsou tyto nástroje blíže popsány a práce s nimi je následně ukázána na příkladech.

Všechny příklady a errata je možné stáhnout na webové stránce www.ProgramujemeKontrolery.cz.

1.3 Fonty textu

Anglické zkratky, názvy nabídek a políček vývojového prostředí a názvy registrů jsou vždy napsány tučnou kurzívou - např. ***AutoReload*** registr. Ač by bylo konzistentnější používat v knize buď pouze české, nebo pouze anglické názvosloví, existuje-li český ekvivalent (např. názvu registru), je použit, jelikož lépe zapadne do věty, která je pak srozumitelnější.

Odkazy na použité zdroje jsou uvedeny číslem v hranatých závorkách – např. [1].

Odkazy na body v obrázcích a grafech budou popsány tučným písmem v závorce - např. **(3)**.

Kapitola 2

STM32WL55

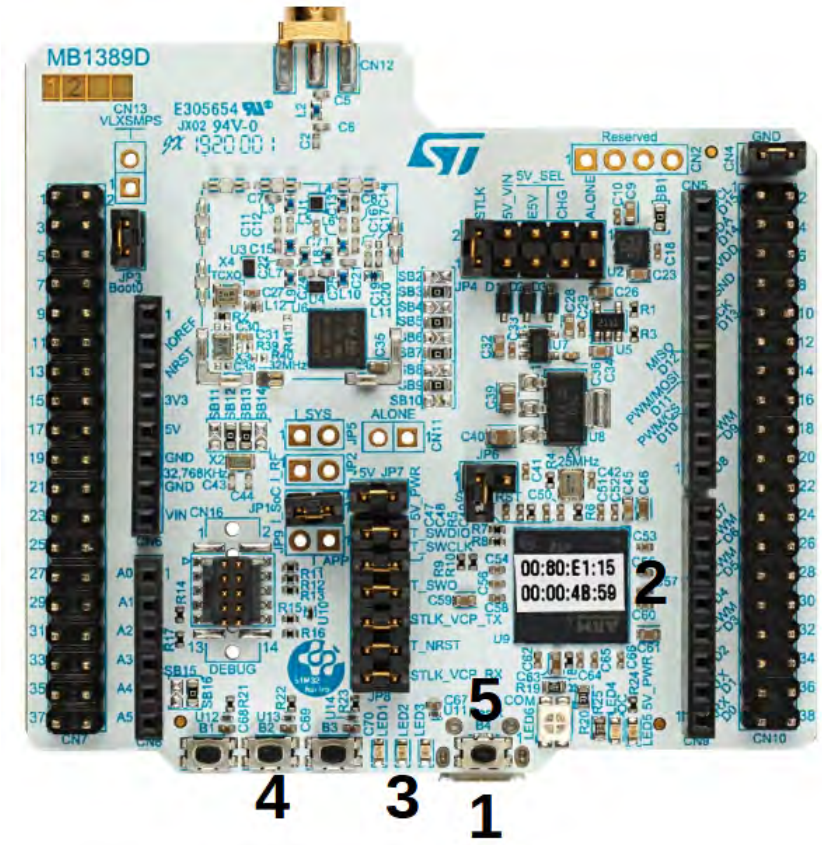
32bitový mikrokontroler ***STM32WL55*** od firmy ***STMicroelectronics*** je kontroler se dvěma jádry ***ARM***. Jádra ***Cortex-M4*** (označováno jako ***CPU1*** nebo ***M4***) a ***Cortex-M0+*** (označováno jako ***CPU2*** nebo ***M0P***) pracují na maximální frekvenci 48 MHz.

Obě jádra sdílejí veškerou programovou a datovou paměť. Sdílejí se tedy i všechny speciální funkční registry, z čehož vyplývá sdílené užívání periferií. Obě jádra mohou bez jakéhokoliv přiřazení využívat jakoukoliv periferii. To může být v některých případech velice výhodné, jindy naopak.

Periferie tohoto kontroleru nijak nevybočují ze standardní výbavy kontrolerů určených pro běžné použití. Jeho specialitou je zabudovaný modul pro bezdrátovou komunikaci ***LoRa***. Tento modul umožňuje snadný a energetický úsporný bezdrátový přenos dat. Díky této kombinaci periferií je kontroler vhodný pro použití v oblasti ***IoT***.

2.1 Vývojová deska

V této kapitole budeme používat vývojovou desku ***NUCLEO-WL55JC1***



Deska obsahuje programátor a ***debugger*** s integrovaným převodníkem sériové komunikace ***UART*** na ***USB*** **(1)** (spodní strana vývojové desky).

Mimo kontroleru **(2)** a velkého množství konektorů (včetně možnosti připojení antény pro Bluetooth komunikaci) se na desce nachází tři uživatelské ***LED*** **(3)**, tři uživatelská **(4)** a jedno resetovací tlačítko **(5)**.

2.2 Vývojové prostředí

Výrobce čipu poskytuje zdarma vývojové prostředí ***STM32CubeIDE***, ve kterém jsou vytvořeny všechny příklady této části.

Toto prostředí založené na prostředí ***Eclipse*** v sobě obsahuje konfigurátor periferií čipu, programování a správu projektu, nahrání programu do mikrokontroleru a možnost jeho ladění.

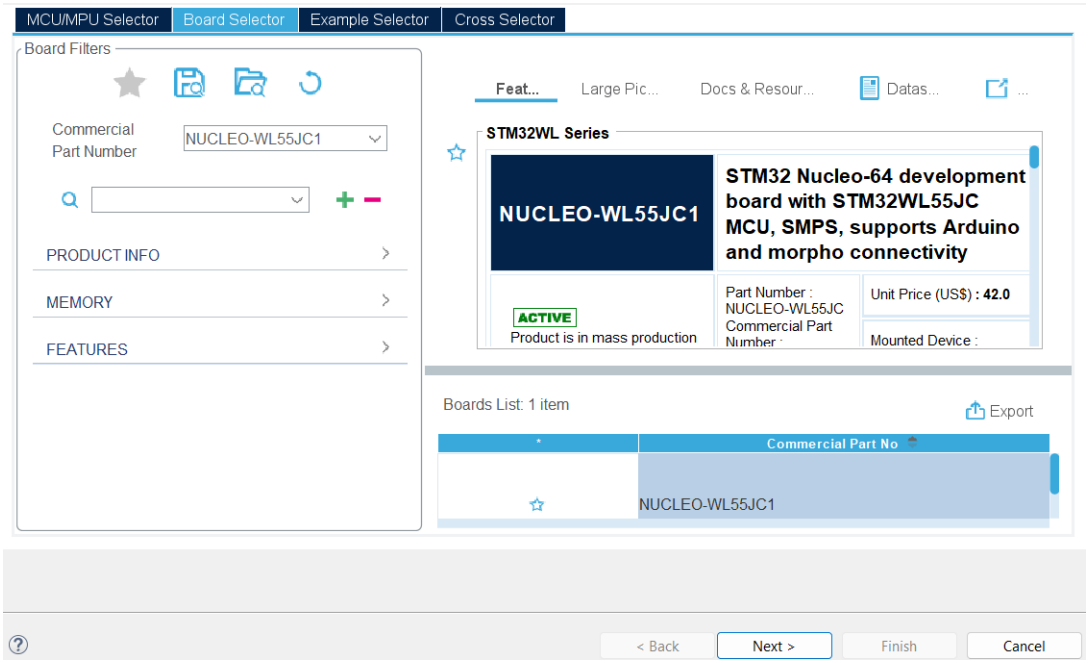
S instalací vývojového prostřední se automaticky nainstaluje i ovladač programátoru, který je umístěný na vývojové desce desce, a překladač ze zdrojového kódu na strojový.

STM32CubeIDE je možné po registraci na webových stránkách firmy STMicroelectronics www.st.com stáhnout zcela zdarma. Nejsnazším způsobem nalezení odkazu ke stažení je zadat do internetového vyhledávače heslo „STM32CubeIDE“ a pravděpodobně hned první odkaz bude mířit na správnou stránku, kde najdete také návod na instalaci a manuál k použití.

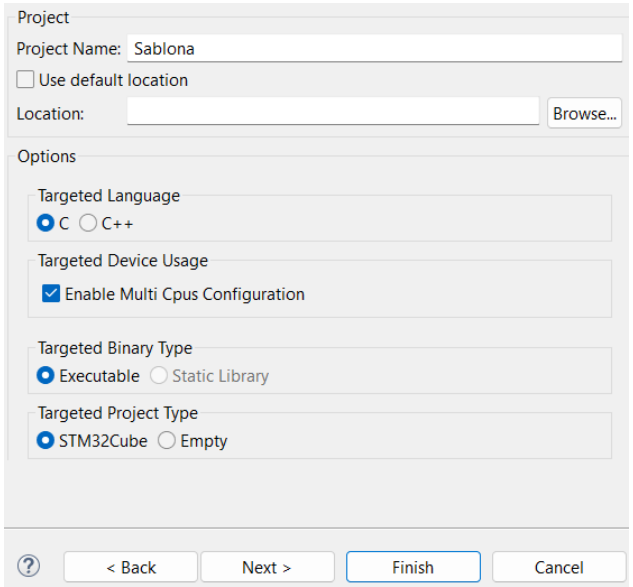
Věřím, že není nutné popisovat instalaci vývojového prostředí, která je plně automatická a intuitivní a nainstaluje i všechny nutné ovladače.

2.3 Založení projektu

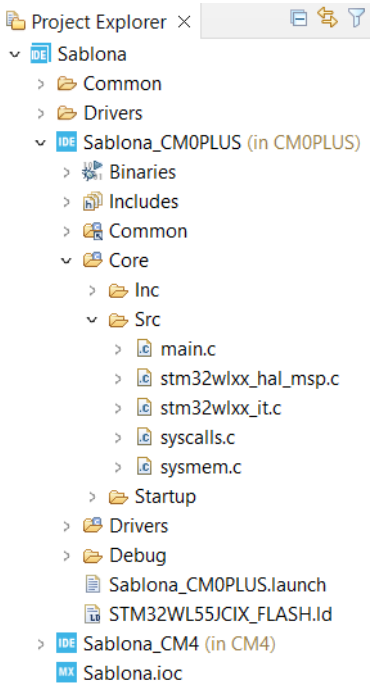
Po spuštění vývojového prostředí vybereme z horní nabídky možnost ***File->New->STM32 project***. Po chvíli se otevře okno s nabídkou všech kontrolerů, které firma ***STM*** vyrábí. My se však přepneme do karty s výčtem všech vyráběných vývojových desek. V této nabídce do vyhledávacího filtru napíšeme název námi používané vývojové desky a zvolíme ji ve výčtu desek.



Po výběru desky uvidíme nové dialogové okno, v němž určíme název a umístění projektu. Nastavení necháme ve výchozím stavu a stiskneme tlačítko ***Finish***.



Po potvrzení nastavení periferií do jejich výchozí režimu se vytvoří projekt, který nalezneme v levé části prostředí. Hlavní projekt obsahuje konfigurační soubor (s příponou ***.ioc***) a dva vnořené projekty s totožnou strukturou. Každý z vnořených projektů patří jednomu jádru. Ve vnořených projektech nalezneme ovladače, knihovny, ale především soubor s uživatelským programem ***main.c***. Také zde nalezneme spojovací programy těchto projektů (s příponou ***.ld***).



Když do spojovacích programů nahlédneme, uvidíme, že každé jádro využívá jinou část programové a datové paměti. Díky tomu se v pamětích programy ani data jednotlivých jader nepřekrývají.

STM32WL55JCIX_FLASH.ld

```
43
44 /* Memories definition */
45 MEMORY
46 {
47   RAM    (xrw)    : ORIGIN = 0x20000000, LENGTH = 32K
48   FLASH    (rx)    : ORIGIN = 0x08000000, LENGTH = 128K
49 }
```

Spojovací program jádra CPU1

STM32WL55JCIX_FLASH.ld

```
43
44 /* Memories definition */
45 MEMORY
46 {
47   RAM    (xrw)    : ORIGIN = 0x20008000, LENGTH = 32K
48   FLASH    (rx)    : ORIGIN = 0x08020000, LENGTH = 128K
49 }
```

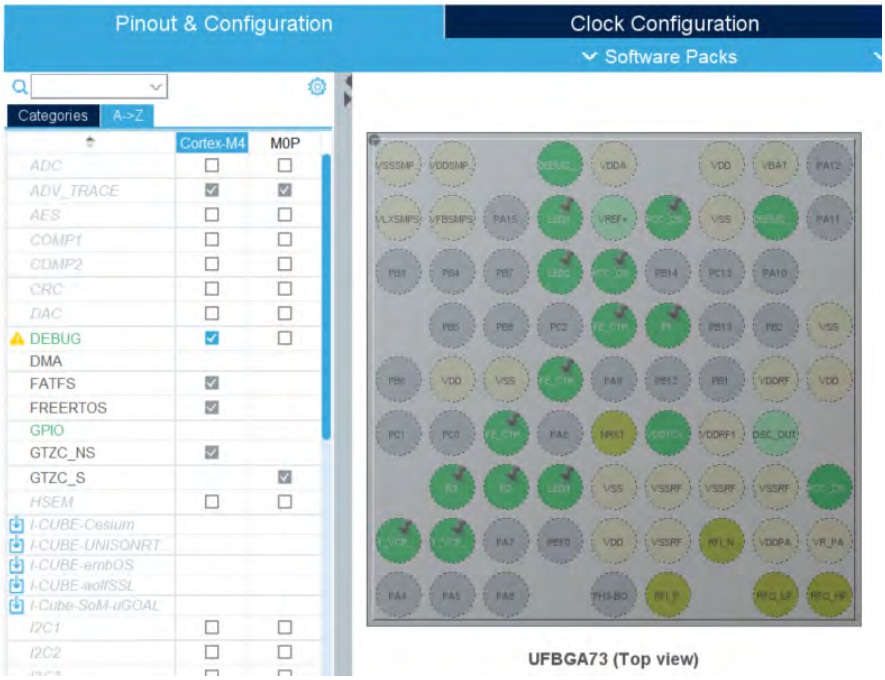
Spojovací program jádra CPU2

Adresa programu, kam jádro po resetu skočí, je předem dána hodnotou registru ***FLASH_SRRVR*** (***FLASH Secure SRAM start address and CPU2 Reset Vector Register***). Nižších 16 bitů tohoto registru nazvaných ***SBRV*** (***CPU2 Boot Reset Vector***) obsahuje adresový posun programové paměti. Hodnota je dána v počtu slov (4 bajtů). Výchozí hodnota adresového posunu po resetu kontroleru je *0x8000*. Vynásobeno 4 bajty jde o posun *0x20000*, což odpovídá hodnotě ve spojovacím programu jádra ***CPU2*** - *0x08020000*. Na této adrese se nachází funkce, která po provedení nezbytných inicializačních operací skočí do hlavního programu. Tato funkce se nachází v souboru (s příponou ***.s***) ve složce ***Startup***. Program je psán v jazyce ***assembler***.

startup_stm32wl55jcix.s

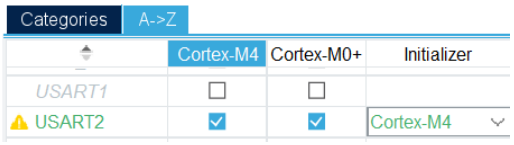
```
97 /* Call static constructors */
98   bl __libc_init_array
99 /* Call the application's entry point.*/
100  bl main
```

Pokud otevřeme soubor s nastavením kontroleru (s příponou ***.ioc***), spustí se grafický konfigurátor. V něm nalezneme mimo rozložení pinů kontroleru také seznam dostupných periferií.



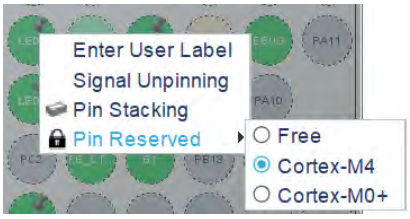
Ve výchozím nastavení je většina pinů připravena pro nastavení, u kterého výrobce předpokládá, že jej budeme chtít. To se týká například i pinů, na které jsou připojené uživatelské ***LED*** a sériová komunikace.

Abychom mohli sériovou komunikaci používat oběma jádry, je třeba v seznamu periferií aktivovat ***USART2*** pro obě jádra. Inicializaci přenecháme jádru ***CPU1*** - tedy ***CM4***. To znamená, že ve vygenerovaném programu jádra ***CPU1*** bude volána inicializační funkce této periferie.

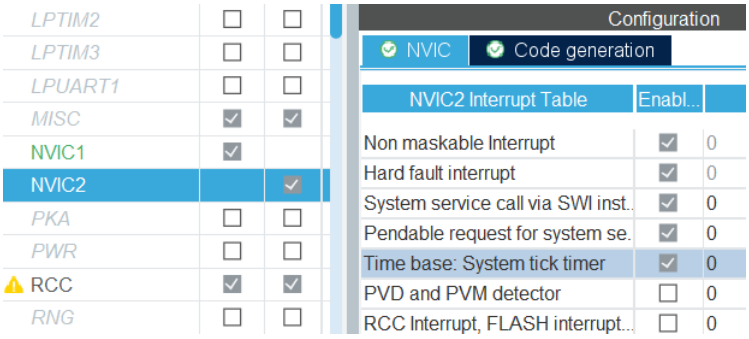


Piny (***PA2*** a ***PA3***) a parametry sériové komunikace už jsou nastaveny, a nemusíme tedy nic měnit. Rychlost přenosu je nastavena na 115200 bitů za sekundu.

Piny pro uživatelské ***LED*** (***PB9***, ***PB11*** a ***PB15***) jsou již také připraveny. V našich příkladech budeme využívat ale využívat pouze ***LED1*** připojenou na pin ***PB15***. Využívat ji bude jádro ***CPU1***. Aby byl pin ve vygenerovaném programu jádra ***CPU1*** inicializován knihovní funkcí, je třeba tento pin danému jádru přiřadit. (Pin může být používán oběma jádry, ale inicializace proběhne pouze v jednom z nich.) To provedeme kliknutím pravého tlačítka myši na daný pin a výběrem jádra, které si pin rezervuje.



Abychom mohli v přerušení používat knihovní funkci pro zpoždění ***HAL_Delay()***, musíme nastavit vysokou prioritu přerušení systémového časovače (***systick***). To uděláme v nabídkách ***NVIC1*** a ***NVIC2*** tak, že prioritu tohoto přerušení nastavíme na nulu.



Když nyní konfiguraci uložíme, dialogové okno nám nabídne vygenerování kódu, které přijmeme. Soubory ***main.c*** v obou projektech nyní obsahují inicializační funkce, které odpovídají nastavení v grafickém konfigurátoru.

Pokud se podíváme do hlavního programu jádra ***CPU1***, uvidíme v něm knihovní funkce pro nastavení hodinového signálu a periferií. Dále v ní vidíme funkci na spuštění druhého jádra. Tu v našem ukázkovém projektu, který budeme používat jako šablonu, zakomentujeme a nahradíme vlastním příkazem se stejnou funkcionalitou.

```
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Boot CPU2 */
  //HAL_PWREx_ReleaseCore(PWR_CORE_CPU2);

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */

  // zapne jadro CPU2
  PWR->CR4 |= PWR_CR4_C2BOOT;

  while (1)
  {
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

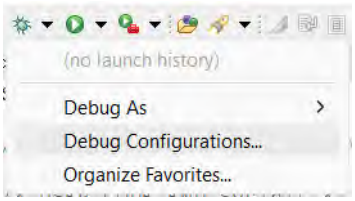
Je důležité, abychom uživatelský kód vždy umisťovali mezi komentáře ***/* USER CODE BEGIN */*** a ***/* USER CODE END */***. Pokud tak neuděláme, změníme nastavení v grafickém konfigurátoru a vygenerujeme nový kód, o náš uživatelský kód přijdeme.

Nyní se podívejme do hlavního programu jádra ***CPU2*** (***CM0***). V něm chybí volání vygenerované funkce pro inicializaci sériové komunikace. Inicializace periferie je provedena v jádru ***CPU1***, a není tu proto nutná. Periferii můžeme bez problémů používat pomocí zápisů a čtení registrů. Pokud ale chceme používat knihovní funkce na její ovládání, musíme i zde manuálně přidat její volání. K inicializaci periferie tedy dojde dvakrát, ale jelikož jde o stejné nastavení, nevadí to.

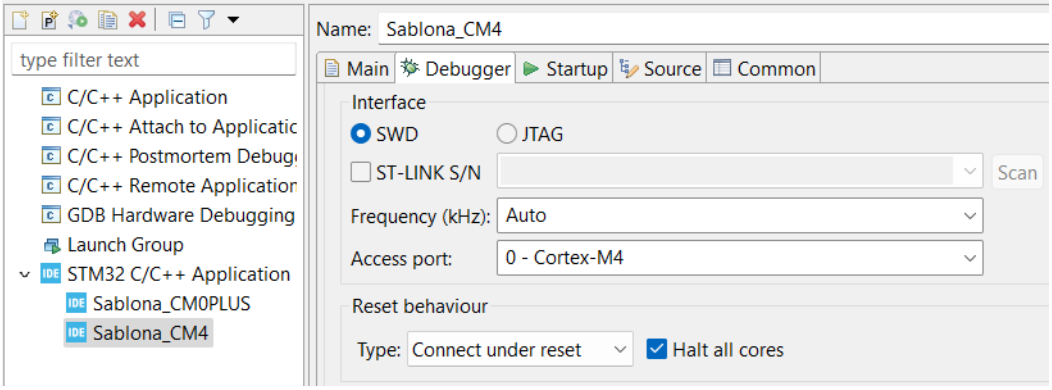
```
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  /* USER CODE BEGIN 2 */
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  MX_USART2_UART_Init();

  while (1)
  {
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

Šablony obou projektů jsou hotové. Nyní nastavíme projekt tak, aby přeložil a nahrál oba programy zároveň.

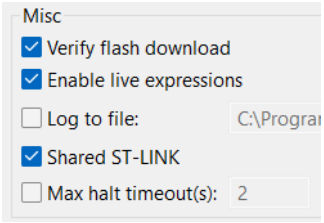
V nabídce projektů v levé části obrazovky vybereme projekt, který má příponu ***CM4***. To je projekt hlavního jádra kontroleru (***CPU1***). Na horní listě klikneme na tlačítko ladění, které má obrázek zeleného brouka. Tím se otevře nabídka vytvoření nového profilu ladění programu. (Pokud již profil existuje, je možné jej upravit kliknutím na dolu směřující šipku vedle tlačítka ladění a otevřením nastavení profilu.)



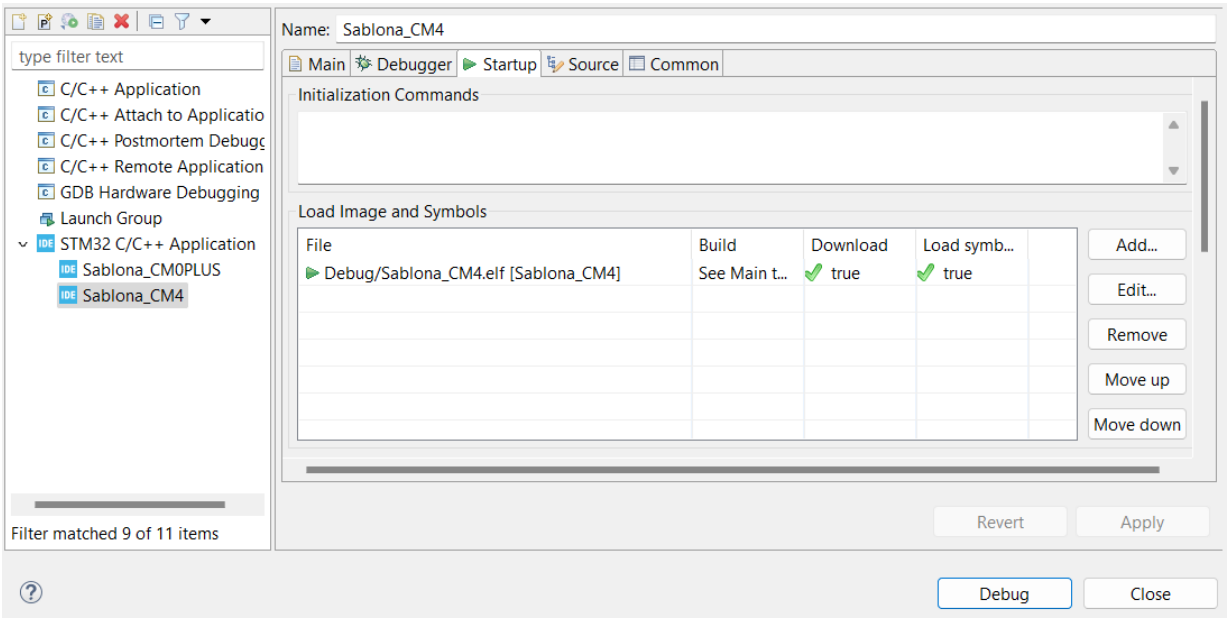
V této nabídce otevřeme kartu ***Debugger***, ve které nastavíme chování po resetu. Zvolíme ***Connect under reset*** a zaškrtneme možnost ***Halt all cores***.



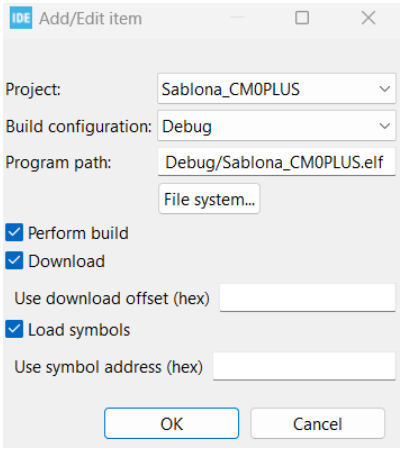
Ve spodní části této karty ověříme, že je zaškrtnuta možnost ***Shared ST-LINK***.



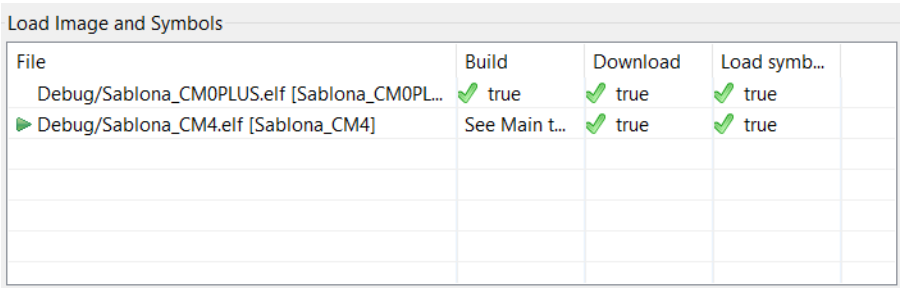
Nyní se přepneme do karty ***Startup***. V ní v okně ***Load Images and Symbols*** vidíme binární soubor tohoto projektu (s příponou ***.elf***).



Musíme sem přidat binární soubor projektu s příponou ***CM0PLUS***. To uděláme tak, že klikneme na tlačítko ***Add***. Otevře se nám nabídka, v níž vybereme binární soubor projektu s příponou ***CM0PLUS*** a zaškrtneme všechny možnosti ve spodní části nabídky. (Pokud v nabídce binární soubor nenaleznete, je nutné projekt s příponou ***CM0PLUS*** nejprve přeložit.)

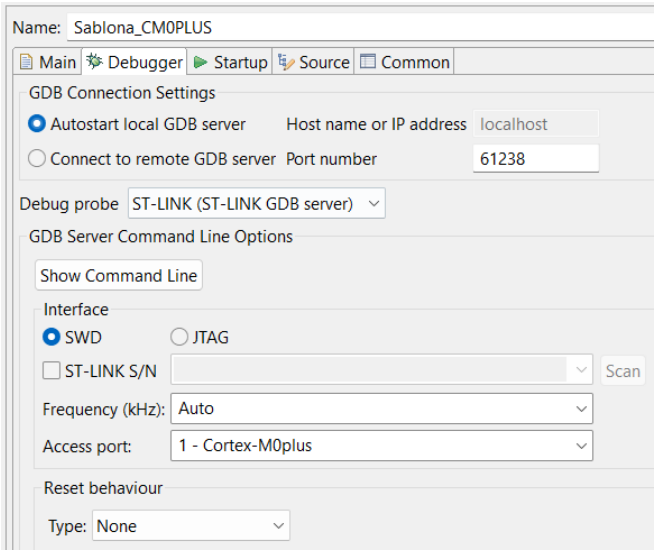


Po potvrzení by mělo výsledné nastavení vypadat následovně:

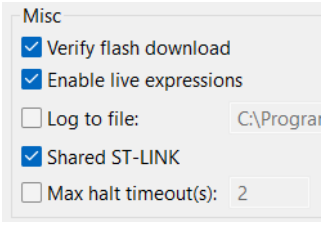


Tím je nastavení ladění jádra ***CPU1*** hotové. Uložíme jej stiskem tlačítka ***Apply***.

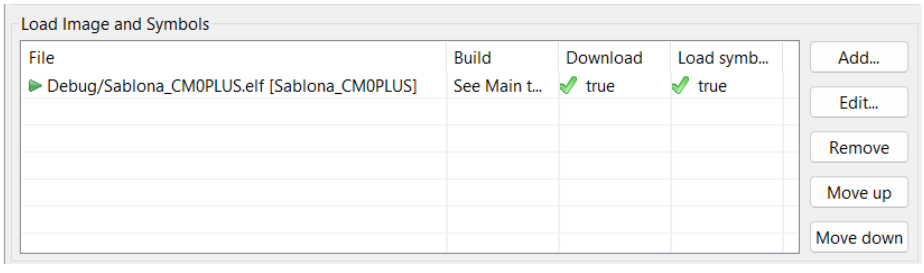
Stejným způsobem otevřeme nabídku nastavení ladění jádra ***CPU2*** (***CM0***) - klikneme tedy na projekt s příponou ***CM0PLUS***, a poté na tlačítko s ikonou zeleného brouka. Otevře se nabídka vytvoření nového profilu ladění programu, v níž zobrazíme kartu ***Debugger***. V té změníme číslo portu, které nesmí být stejné jako v nastavení pro jádro ***CPU1***. Dle doporučení v dokumentaci [4] jej nastavíme na hodnotu *61238*. Dále změníme nastavení ***Reset behaviour*** na hodnotu ***None***.



Ve spodní části této karty opět ověříme, že je zaškrtnuta možnost ***Shared ST-LINK***.



Opět se přepneme do karty ***Startup***. V ní v okně ***Load Images and Symbols*** vidíme binární soubor tohoto projektu.



Vážení čtenáři, právě jste dočetli ukázku z knihy ***Programujeme dvojjádrové kontrolery*.**
Pokud se Vám ukázka líbila, na našem webu si můžete zakoupit celou knihu.