Programujeme STM32

bez knihoven

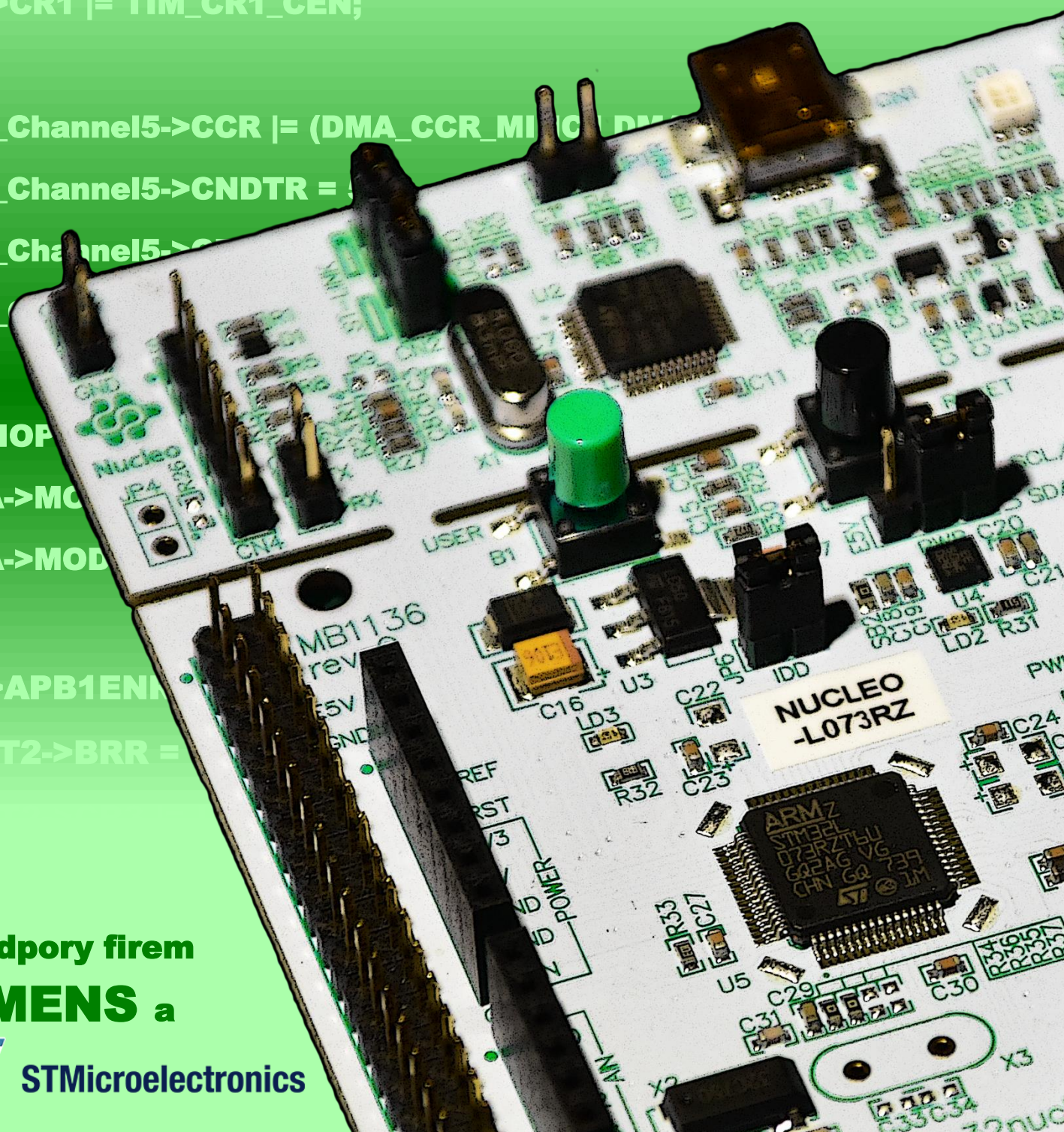
Ing. Vojtěch Skřivánek

```
TIM2->CR2 |= TIM_CR2_MMS_1;
TIM2->CR1 |= TIM_CR1_CEN;

DMA1_Channel5->CCR |= (DMA_CCR_MI
DMA1_Channel5->CNDTR =
DMA1_Cha nel5->
DMA1_

RCC->IOP
GPIOA->MO
GPIOA->MOD

RCC->APB1ENR
USART2->BRR =
```



Za podpory firem

SIEMENS a

STMicroelectronics

Programujeme STM32
bez knihoven

Ing. Vojtěch Skřivánek

Vydání první 2022

ISBN: 978-80-7539-134-6 (PDF verze)
ISBN: 978-80-7539-135-3 (ePub verze)
ISBN: 978-80-7539-136-0 (mobi verze)

Poděkování

Děkuji firmám

SIEMENS

a

za podporu při psaní této knihy.

Obsah

1 Úvod **1**
- 1.1 Motivace knihy . . . 1
- 1.2 Struktura knihy . . . 3
- 1.3 Fonty textu . . . 4

2 Vývojové nástroje **5**
- 2.1 Vývojová deska . . . 6
- 2.2 Vývojové prostředí . . . 7
- 2.3 Dokumentace . . . 7
- 2.4 Shrnutí . . . 8

3 Založení projektu **9**

4 Vstupně/výstupní piny **15**
- 4.1 Tlačítkem rozsvícená LED . . . 15
  - 4.1.1 Práce s registry při tvorbě programu . . . 16
    - 4.1.1.1 Nalezení symbolického názvu registru . . . 16
    - 4.1.1.2 Nalezení symbolického názvu bitu . . . 16
    - 4.1.1.3 Zápis do registru . . . 17
  - 4.1.2 Tvorba programu . . . 17
  - 4.1.3 Porovnání . . . 18

5 Přerušení **21**
- 5.1 Tlačítkem rozsvícená LED pomocí přerušení . . . 22
  - 5.1.1 Nastavení periferií . . . 22
    - 5.1.1.1 LED . . . 22
    - 5.1.1.2 EXTI . . . 22
  - 5.1.2 Funkce periferií . . . 24
    - 5.1.2.1 LED - přepínání . . . 24
  - 5.1.3 Obsluha přerušení . . . 24
  - 5.1.4 Tvorba hlavního programu . . . 26
  - 5.1.5 Zapojení . . . 26
  - 5.1.6 Porovnání . . . 27

6 Časovač **29**
- 6.1 Blikání pomocí časovače s přerušením . . . 29
  - 6.1.1 Nastavení periferií . . . 29
    - 6.1.1.1 RCC - hodinový signál . . . 29
    - 6.1.1.2 TIM . . . 31
  - 6.1.2 Funkce periferií . . . 32
    - 6.1.2.1 TIM - start . . . 32

6.1.3 Obsluha přerušení . . . 32
6.1.4 Tvorba programu . . . 33
6.1.5 Porovnání . . . 34
6.2 Blikání pomocí OC režimu . . . 35
6.2.1 Nastavení periferií . . . 35
6.2.1.1 TIM . . . 35
6.2.2 Tvorba programu . . . 37
6.2.3 Provnání . . . 37
6.3 Blikání s nastavením délky pomocí IC režimu . . . 38
6.3.1 Nastavení periferií . . . 38
6.3.1.1 TIM3 . . . 38
6.3.2 Funkce periferií . . . 39
6.3.2.1 TIM3 start . . . 39
6.3.3 Obsluha přerušení . . . 40
6.3.4 Tvorba programu . . . 41
6.3.5 Zapojení . . . 41
6.3.6 Rozbor programu . . . 42
6.3.7 Porovnání . . . 42
6.4 Nastavení jasu LED pomocí PWM režimu . . . 43
6.4.1 Nastavení periferií . . . 43
6.4.1.1 TIM . . . 43
6.4.2 Funkce periferií . . . 44
6.4.2.1 TIM2 - změna střídy . . . 44
6.4.3 Tvorba programu . . . 45
6.4.4 Rozbor programu . . . 45
6.4.5 Porovnání . . . 46
6.5 Časovač spuštěný čítačem vnějších událostí . . . 47
6.5.1 Nastavení periferií . . . 48
6.5.1.1 TIM2 . . . 48
6.5.1.2 TIM21 . . . 49
6.5.2 Funkce periferií . . . 50
6.5.2.1 TIM21 - start . . . 50
6.5.3 Tvorba programu . . . 50
6.5.4 Zapojení . . . 51
6.5.5 Porovnání . . . 51

7 UART 53
7.1 LED rozsvícená povelem z počítače . . . 53
7.1.1 Převodník sériové komunikace . . . 53
7.1.2 Nastavení periferií . . . 54
7.1.2.1 UART . . . 54
7.1.3 Funkce periferií . . . 55
7.1.3.1 UART - odesílání dat . . . 55
7.1.3.2 UART - příjem dat . . . 56
7.1.3.3 UART - callback přijmu . . . 58
7.1.4 Obsluha přerušení . . . 59
7.1.5 Tvorba programu . . . 60
7.1.6 Porovnání . . . 60

8 ADC **61**
8.1 Přerušované měření dvou kanálů . . . 61
8.1.1 Nastavení periferií . . . 61
8.1.1.1 ADC . . . 61
8.1.1.2 UART . . . 63
8.1.2 Funkce periferií . . . 63
8.1.2.1 ADC - start . . . 63
8.1.2.2 UART – odešli data v blokujícím režimu . . . 64
8.1.3 Obsluha přerušení . . . 64
8.1.4 Tvorba programu . . . 65
8.1.5 Zapojení . . . 66
8.1.6 Porovnání . . . 67
8.2 Jednotlivé měření dvou kanálů s externím spouštěním . . . 68
8.2.1 Nastavení periferií . . . 68
8.2.1.1 ADC . . . 68
8.2.1.2 EXTI . . . 70
8.2.2 Obsluha přerušení . . . 71
8.2.3 Tvorba programu . . . 72
8.2.4 Zapojení . . . 72
8.2.5 Porovnání . . . 73

9 DAC **75**
9.1 9.1 Nastavení jasu LED pomocí DAC . . . 75
9.1.1 Nastavení periferií . . . 75
9.1.1.1 DAC . . . 75
9.1.2 Funkce periferií . . . 76
9.1.2.1 DAC - nastavení výstupní hodnoty . . . 76
9.1.3 Tvorba programu . . . 76
9.1.4 Porovnání . . . 77

10 SPI **79**
10.1 Obousměrná SPI komunikace Master<->Slave . . . 79
10.1.1 Nastavení periferií . . . 79
10.1.1.1 SPI1 . . . 79
10.1.1.2 SPI2 . . . 81
10.1.2 Funkce periferií . . . 82
10.1.2.1 SPI1 – vysílaní a příjem v blokujícím režimu . . . 82
10.1.2.2 SPI2 – vysílaní a příjem v neblokujícím režimu . . . 82
10.1.3 Obsluha přerušení . . . 84
10.1.4 Tvorba programu . . . 84
10.1.5 Zapojení . . . 85
10.1.6 Rozbor programu . . . 85
10.1.7 Porovnání . . . 86

11 I2C **87**
11.1 Jednosměrná komunikace Master->Slave a Master<-Slave . . . 87
11.1.1 Nastavení periferií . . . 88
11.1.1.1 I2C1 . . . 88
11.1.1.2 I2C3 . . . 89
11.1.2 Funkce periferií . . . 90
11.1.2.1 I2C1 - vysílání . . . 90

11.1.2.2 I2C1 - příjem . . . 92
11.1.2.3 I2C3 - vysílání . . . 93
11.1.2.4 I2C3 - příjem . . . 94
11.1.3 Obsluha přerušení . . . 95
11.1.3.1 I2C1 . . . 95
11.1.3.2 I2C3 . . . 96
11.1.4 Tvorba programu . . . 97
11.1.5 Zapojení . . . 98
11.1.6 Rozbor programu . . . 98
11.1.7 Porovnání . . . 100

12 DMA **101**
12.1 DMA přenos z paměti do paměti . . . 101
12.1.1 Nastavení periferií . . . 101
12.1.1.1 DMA . . . 101
12.1.2 Funkce periferií . . . 102
12.1.2.1 DMA - start . . . 102
12.1.3 Obsluha přerušení . . . 103
12.1.4 Tvorba programu . . . 103
12.1.5 Rozbor programu . . . 104
12.1.6 Porovnání . . . 104
12.2 DMA přenos z periferie do paměti a naopak . . . 105
12.2.1 Nastavení periferií . . . 105
12.2.1.1 UART . . . 105
12.2.1.2 DMA5 . . . 106
12.2.1.3 DAC . . . 107
12.2.1.4 DMA4 . . . 108
12.2.1.5 TIM . . . 108
12.2.2 Funkce periferií . . . 109
12.2.2.1 DMA5 - start . . . 109
12.2.2.2 DMA4 - start . . . 109
12.2.2.3 TIM - start . . . 109
12.2.3 Tvorba programu . . . 110
12.2.4 Porovnání . . . 110
12.3 DMA přenos z periferie do periferie . . . 111
12.3.1 Nastavení periferií . . . 111
12.3.1.1 ADC . . . 111
12.3.1.2 DMA1 . . . 113
12.3.2 Tvorba programu . . . 114
12.3.3 Zapojení . . . 115
12.3.4 Porovnání . . . 116

13 Závěr **117**

Kapitola 1

Úvod

Tato kniha navazuje na knihu „Programujeme STM32 – zdolejte jednočipy profesionálů“. Proto, pokud ještě nemáte zkušenosti s kontrolery STM32, doporučuji nejprve přečíst tu.

Stejně jako předchozí kniha se i tato zabývá devíti nejrozšířenějšími periferiemi mikrokontrolerů STM32. Rozdíl je však v přístupu k těmto periferiím.

Kniha se již nevěnuje tomu, jak periferie fungují, ani jak používat standardní knihovny. Je spíše sbírkou příkladů, které ukazují, jak periferie využívat bez použití knihoven. Prací přímo s registry jednotlivých periferií poskytne čtenáři povědomí o tom, jaké registry a jaká nastavení jsou pro danou periferii kontroleru STM32 typickými. Všechny kontrolery STM32 mají totiž jednotné názvosloví i účel registrů (kontrolní, konfigurační, stavový, datový ...). Z toho důvodu bude snadné zorientovat se i při práci s jiným kontrolerem.

Použití knihoven často skryje, co všechno je nutné v periferii nastavit, aby plnila svůj účel. Dvojnásobně to platí u spolupráce dvou periferií. Tím, že příklady v této knize nepoužívají knihovny, pronikneme hlouběji do funkce mikrokontroleru. Uvědomíme si i věci, které se na první pohled nemusí zdát zcela zřejmé.

Předpokládá se, že čtenář umí zacházet s vývojovým prostředím STM32CubeIDE a ví, jak nahrát a ladit program kontroleru. Dalším nutným předpokladem je obecná teoretická znalost funkce všech periferií a komunikačních protokolů, se kterými tato kniha pracuje.

1.1 Motivace knihy

Jaký je vlastně důvod nepoužívat oficiální knihovny k ovládání periferií mikrokontroleru? Jaký má smysl psát si vlastní knihovní funkce? Není to zbytečná práce skýtající riziko vytvoření chyby?

Hlavními důvody, proč nevyužít knihovny a napsat si vlastní kód, jsou:

- Velikost programu:

  Knihovní funkce jsou navrženy tak, aby byly univerzální. Ať už chceme nastavit nebo použít periferii jakýmkoliv způsobem, knihovní funkce si s tím poradí. Tato univerzálnost je bohužel vykoupena velkým množstvím kódu, který se v praxi vůbec nevyužije, jelikož vždy chceme danou periferii využít pouze v jednom konkrétním režimu.

- Rychlost programu:

  Tento bod souvisí s předchozím. Jelikož jsou knihovní funkce univerzální, obsahují velké množství podmínek a větvení. Pokud přesně víme, jak periferii použít, je možné vytvořit funkce s minimem podmínek, které program zpomalují.

- Testovatelnost programu:

  Pokud pracujeme na programu produktu, který musí fungovat bezpečně, měla by být každá funkcionalita, každá větev, ideálně každý řádek kódu otestován. Otestovat knihovní funkci je velmi pracné a především zbytečné, pokud využíváme pouze její část. Navíc by program, který má získat bezpečnostní certifikaci, neměl obsahovat nevyužitý (mrtvý) kód. Knihovní funkce jsou takového kódu plné.

Dalším argumentem může například to, že ne vždy knihovní funkce nabízí všechny možnosti využití periferie.

Na druhou stranu by měly být zmíněny i výhody použití knihoven:

- Knihovní funkce neobsahují chyby (většinou):

  Oficiální knihovní funkce jsou otestované. Neopomíjejí žádné nastavení periferie, které může programátor při psaní vlastních knihoven vynechat, a program se může v neočekávaných podmínkách chovat nestandardně. POZOR! U nových rodin čipů je možné, že i oficiální knihovny obsahují doposud neodhalenou chybu (především u funkcí málo používaných periferií nebo jejich režimů)

- Knihovní funkce jsou přenositelné:

  Tato výhoda neplatí vždy, ale skutečně je možné, že velkou část svého programu můžete použít i v případě, že se rozhodnete jej spustit na jiném, podobném kontroleru. Některé knihovní funkce mají velmi často stejný tvar i použití u více rodin kontrolerů.

- Knihovní funkce nevyžadují detailní znalost kontroleru:

  Zřejmě největší výhodou použití knihovních funkcí je to, že není potřeba znát, jak kontroler funguje. Jaké registry je potřeba nastavit, aby periferie dělala to či ono, není naše starost. Pokud chceme například přijmout data pomocí digitální komunikace, použijeme knihovní funkci a ta to provede, aniž bychom museli vědět, jaké operace je pro to nutné udělat. A je pravdou, že některé kontrolery se v použití některých periferií značně liší.

Nyní nechme všechny nevýhody knihoven stranou. Především s přihlédnutím k poslednímu bodu výhod knihoven může vyvstat otázka: Proč se učit do detailu, jak nastavit a využívat periferie tohoto kontroleru, když u jiného mohou fungovat odlišně?

Ano, je pravda že jiný kontroler STM32 nebude fungovat stejně. Kontroler STM32L073RZ je ale jedním z nejjednodušších kontrolerů. Můžeme tedy očekávat, že to, co je nutné nastavit u něj, bude potřeba udělat i u komplexnějších rodin, jelikož základní nastavení bývá shodné.

U kontrolerů z vyšších řad jsou větší možnosti nastavení a sofistikovanější chování periferií. To v praxi sice znamená jejich náročnější inicializaci, ale poté jejich pohodlnější obsluhu. Navíc to, že periferie nabízí pokročilejší, více automatizované režimy (například bufferování přijmu dat digitální komunikace) neznamená, že je musíme používat. Velmi často je možné i pokročilejší periferie komplexnějších kontrolerů používat v základním režimu.

1.2 Struktura knihy

Každá z devíti kapitol se zabývá jednou periferií. Obsahuje jeden nebo více příkladů kódu jejího využití. Příklady v této knize věrně kopírují ukázky z předchozí „Programujeme STM32 – zdolejte jednočipy profesionálů“. Na konci každého příkladu tyto kódy porovnáme z různých hledisek.

Oproti předchozí knize není tato kniha učebnicí, takže obsahuje o poznání méně textu. Avšak o to více je v ní kódu, který je textem okomentovaný.

Na začátku každého příkladu je jeho zadání. Po něm následuje tvorba a popis kódu, který zadání splní. Kód se nejčastěji skládá z následujících částí:

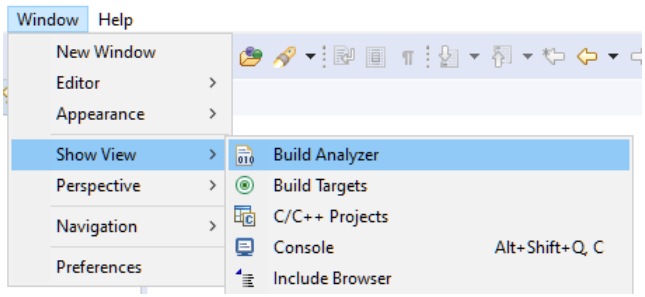
- Nastavení periferií,
- funkce periferií,
- obsluha přerušení periferií a
- tvorba programu.

V některých příkladech jsou jisté části vynechány, v jiných jsou části navíc.

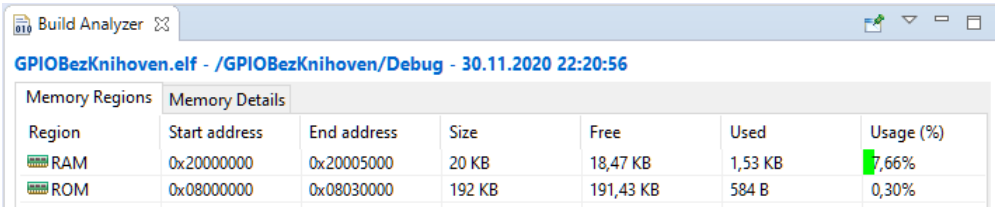
Důležité je upozornit, že kód některých funkcionalit neukazuje, jak danou problematiku řešit nejlépe, ale co možná nejjednodušeji. Účelem je snadné pochopení funkce periferií a programu. Stejně tak jsou všechny funkce v programu určené pouze pro jeden účel a jednu konkrétní periferii.

Errata a projekty příkladů z této knihy je možné najít na www.programujemekontrolery.cz.

V závěru většiny příkladů je porovnání kódu za použití knihoven a bez nich. Nejčastěji se jedná o porovnání velikosti a rychlosti kódu. Rychlost je porovnána měřením délky pulzu výstupního pinu, který ohraničuje měřenou část kódu. Při porovnávání rychlosti je samozřejmě nastavená stejná frekvence systémového hodinového signálu. Velikost kódu je porovnána na základě výstupních informací překladače. Můžeme si ji zobrazit v okně ***Build Analyzer***.

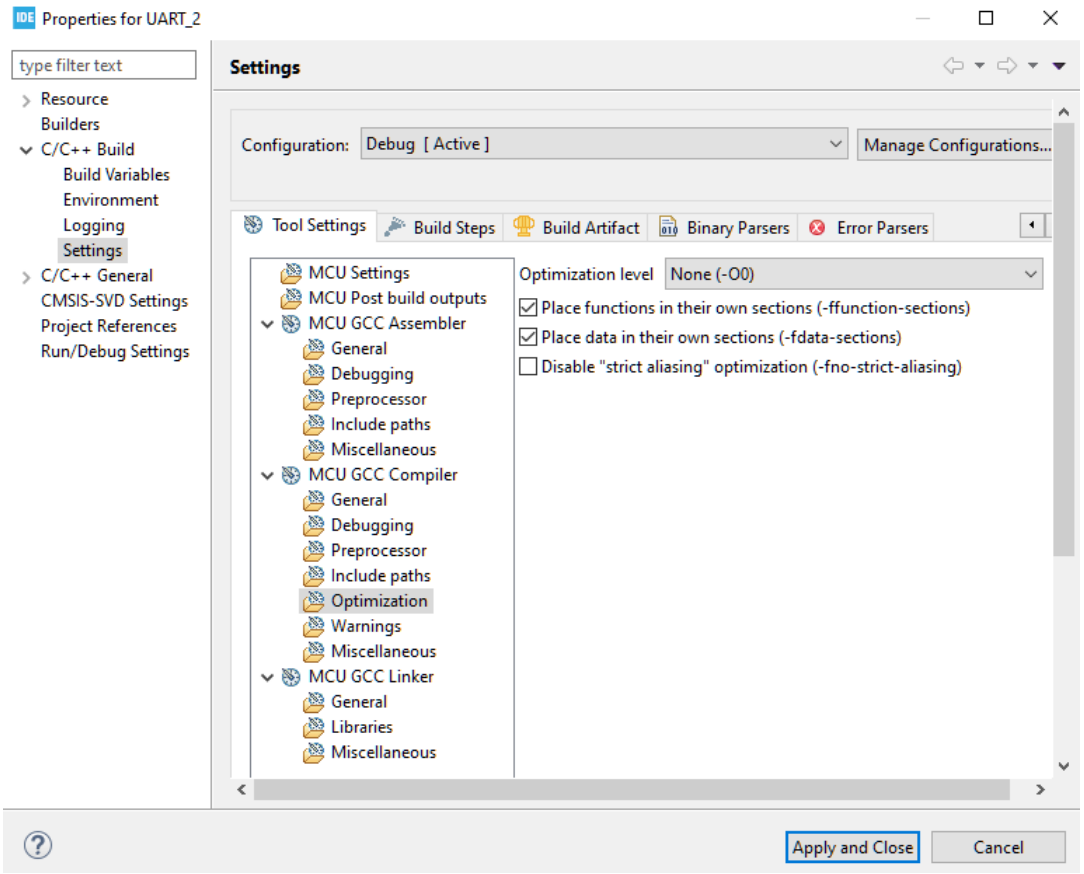


Po jeho otevření se nám zobrazí okno s informacemi o velikosti přeloženého programu.



Porovnání bude uvedeno pro tři různá nastavení optimalizace překladu kódu. První porovnání bude uvedeno pro překlad bez optimalizace (***-O0***), druhé pro optimalizaci s ohledem na nejmenší velikost kódu (***-Os***) a třetí při optimalizaci se zaměřením na nejrychlejší program (***-Ofast***).

Způsob optimalizace lze změnit v nastavení překladače projektu.



Za poznámku stojí, že při tvorbě programu pro bezpečné zařízení nebývá povoleno využívat jakékoliv optimalizace.

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 ten, jelikož lépe zapadne do věty, která je pak srozumitelnější. Odkazy na použité zdroje jsou uvedeny číslem v hranatých závorkách – např. [1].

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

Kapitola 2

Vývojové nástroje

K programování mikrokontrolerů jsou zapotřebí tři základní nástroje.

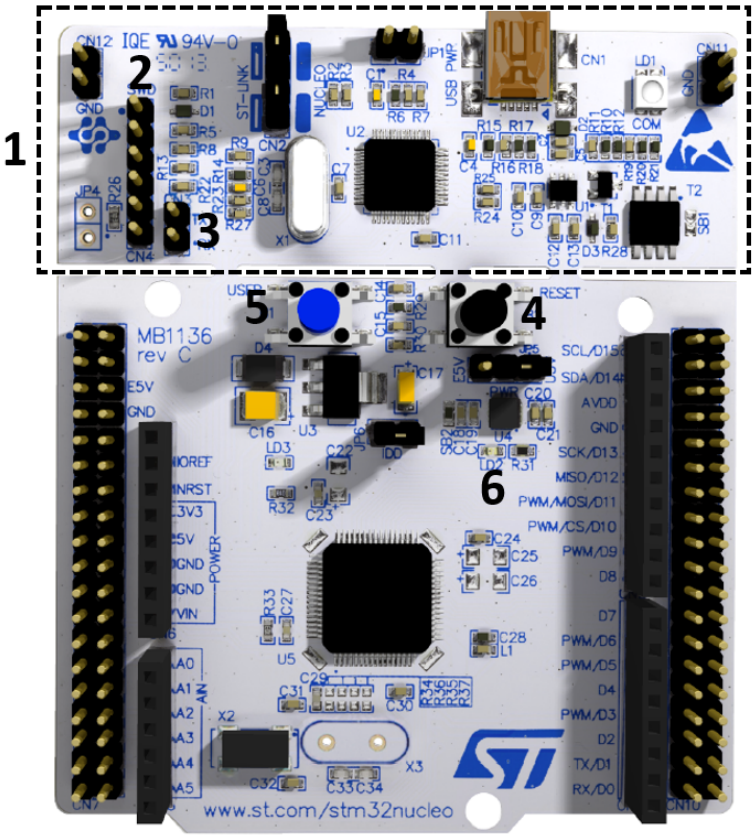
Prvním z nich je samozřejmě samotný mikrokontroler. Nejsnazším způsobem, jak mít k dispozici funkční mikrokontroler připravený k programování a používání jeho periferií, je opatřit si vývojovou desku. Firma STMicroelectronics nabízí širokou škálu cenově příznivých vývojových desek. Mezi nejznámější patří řady desek ***Nucleo*** a ***Discovery***. Ty se od sebe liší tím, že zatímco ***Nucleo*** obsahuje převážně pouze mikrokontroler a konektory, ***Discovery*** dost často skýtá také rozličné senzory, tlačítka, LED, displeje a jiné prvky, připojené přímo ke kontroleru a připravené k okamžitému použití.

Druhým potřebným nástrojem je programátor mikrokontroleru, který dokáže nahrát binární kód z PC do paměti programu čipu. Tento programátor je možné koupit samostatně, ale výhodou všech vývojových desek firmy STMicroelectronics je, že každá z nich má v sobě programátor a debugger (používaný k ladění programu) již integrovaný.

Zbývá poslední nástroj, ačkoliv jde vlastně o nástroje dva, které je však možno opět získat v jednom balíčku. Prvním je vývojové prostředí, ve němž je možné snadno psát kód a spravovat projekt programu. Druhým je překladač zdrojového kódu, psaného v jazyce C, do strojového binárního kódu, kterému rozumí daný kontroler. Firma STMicroelectronics nabízí zdarma své vývojové prostředí ***STM32CubeIDE***, ve kterém je možné napsat zdrojový kód, přeložit jej do strojového kódu a pomocí programátoru jej nahrát do programové paměti mikrokontroleru.

2.1 Vývojová deska

Vývojová deska, která je použita na všechny příklady v této knize, je z řady desek ***Nucleo*** s názvem ***NUCLEO-L073RZ***.



Vývojová deska NUCLEO-L073RZ [2]

Deska obsahuje, mimo mikrokontroleru ***STM32L073RZ***, také již zmíněný programátor a debugger sloužící k ladění programu **(1)**. Programátor se nachází v horní části desky, kterou je možné od spodní odlomit. K programování kontrolerů mimo vývojovou desku slouží konektor ***CN4*** v levé části **(2)**. Horní část také obsahuje převodník z USB komunikace na ***UART***, jehož vývody CN3 jsou vpravo od programovacího konektoru **(3)**. Ani jeden z těchto konektorů není třeba používat, jelikož programátor i převodník jsou s kontrolerem spojeny můstky mezi horní a dolní částí desky.

Na dolní části desky se nachází dvě tlačítka, jedno z nich - ***B2*** - slouží k resetování programu mikrokontroleru *(4)*, druhé - ***B1*** - je uživatelské **(5)**, které je připojené na pin mikrokontroleru a jež bude často využívané v našich příkladech.

Kromě LED signalizující funkci programátoru a správné napájení je na desce umístěna také jedna uživatelská LED ***LD2*** **(6)**, kterou lze pinem mikrokontroleru ovládat.

Na krajích desky pak nelze přehlédnout dva typy konektorů. Vnitřní tvořené dutinkami jsou kompatibilní se všemi rozšiřujícími deskami pro platformu ***Arduino***. Vnější hřebínkové jsou přivedeny ke všem pinům kontroleru.

POZOR!! Ne všechny konektory jsou skutečně připojeny k pinům kontroleru. Někdy je nutné na příslušné místo na desce připájet nulový rezistor, aby došlo k propojení. Důvodem je, že daný pin je již použit například programátorem. Detaily je možné najít v dokumentaci ***Nucleo*** desky [4].

Kromě výše zmíněných funkcí deska ještě nabízí možnost připájení vlastního přesného oscilátoru, místo na měření proudové spotřeby a přepnutí na externí napájení. Nic z toho ale v této knize není využito.

2.2 Vývojové prostředí

Jak již bylo zmíněno, výrobce čipu poskytuje zdarma vývojové prostředí ***STM32CubeIDE***, ve kterém jsou vytvořeny všechny příklady této knihy.

Toto prostředí v sobě obsahuje konfigurátor periferií čipu, programování a správu projektu, nahrání programu do mikrokontroleru a možnost jeho ladění (debuggování).

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

STM32CubeIDE je možné po registraci na webových stránkách firmy STMicroelectronics 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 Dokumentace

K práci s mikrokontrolerem a vývojovou deskou přijdou vhod tři dokumenty.

Prvním a nejdůležitějším je referenční manuál mikrokontroleru [1]. Tento dokument obsahuje kompletní popis kontroleru, jeho funkcí a periferií. Po jeho otevření je nutné nezaleknout se počtu stránek, jelikož ne všechny jsou stejně důležité. Všechny kapitoly o periferiích, které kontroler má, mají stejnou strukturu. Po obecném úvodu a výpisu funkcí periferie následuje blokový diagram periferie. Kapitola pokračuje popisem jednotlivých funkcí a režimů periferie. Poslední částí kapitoly v referenčním manuálu jsou popisy registrů spjatých s danou periferií. S nimi budeme pracovat v našich příkladech, a proto této části dokumentace budeme věnovat největší pozornost.

Je vhodně při práci na příkladech této knihy nahlížet do referenčního manuálu a význam a možnosti registrů v něm paralelně sledovat.

Druhým dokumentem je uživatelský manuál vývojové desky ***Nucelo*** [2]. V něm je mimo jiné možné najít, jaké piny kontroleru jsou připojeny k dutinkovým a hřebínkovým konektorům desky, či na kterých pinech je připojeno uživatelské tlačítko a uživatelská LED. Také se v něm nachází informace, které piny nejsou s konektory spojeny vůbec, a zda je možné to dodatečně udělat připájením nulového rezistoru na konkrétní pájicí plošky na desce.

Poslední, spíše doplňkový dokument, je katalogový list (***datasheet***) [4] rodiny mikrokontroleru. V dokumentu jsou uvedena obecná data o mikrokontroleru, jako druhy, počet a stručný popis všech periferií, velikosti pamětí, dostupná pouzdra, statické a dynamické charakteristiky (napájení a spotřeba čipu, teplotní charakteristiky, chyby převodníků…), názvy pinů pouzder, informace o napájení čipu, informace pro návrh plošného spoje a pro osazení. Tento dokument je zde uveden především kvůli důležité tabulce popisující jednotlivé piny kontroleru, která je zmíněna v kapitole o vstupně/výstupních pinech.

Všechny tyto dokumenty přijdou vhod v momentě, kdy se chystáme pracovat s jeho periferií nebo je potřeba k mikrokontroleru připojit externí periferii.

2.4 Shrnutí

Po přečtení této kapitoly by měl mít čtenář k úspěšnému naprogramování příkladů uvedených v této knize připraveny následující nejnutnější věci:

- vývojovou desku **NUCLEO-L073RZ**,
- **USB** kabel k připojení desky k počítači,
- počítač s nainstalovaným vývojovým prostředím **STM32CubeIDE**,
- a **několik vodičů** k vzájemnému propojení vnějších konektorů vývojové desky.

S touto nezbytnou výbavou, k níž je doporučeno mít k dispozici zmíněnou dokumentaci, je možné se směle pustit do čtení následujících kapitol, které odhalují programování mikrokontrolerů STM32.

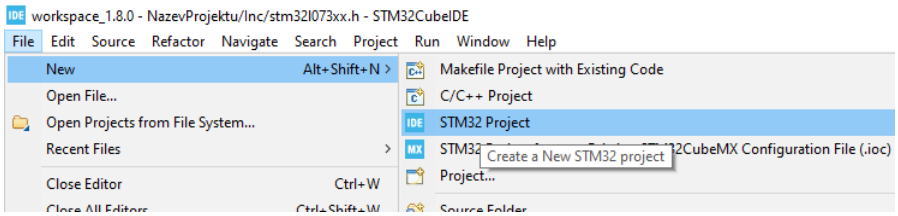
Kapitola 3

Založení projektu

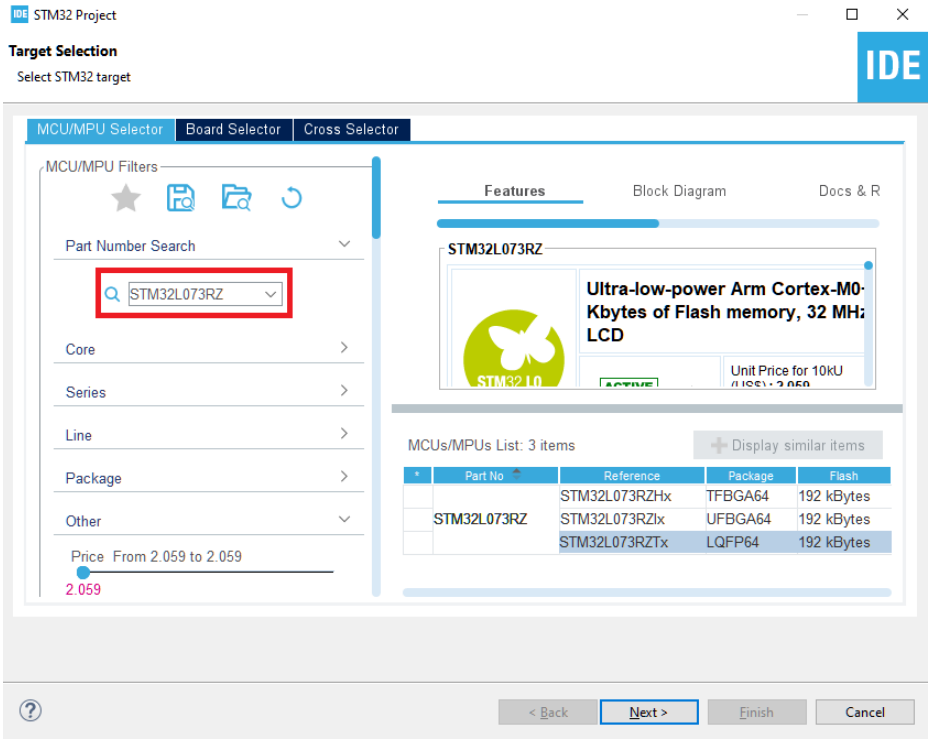
Před každým příkladem v této knize je potřeba vytvořit projekt bez přidaných knihoven. Tato kapitola ukazuje, jak toho dosáhnout. Postup je platný pro ***STM32CubeIDE*** verze 1.8.0.

V případě, že uvedený postup nefunguje (patrně z důvodu nové verze vývojového prostředí), vyhledejte aktualizovaný postup na webové stránce www.programujemekontrolery.cz. Pokud se ani na ní nebude nalézat funkční postup, kontaktujte autora knihy na emailové adrese *VojtechSkrivanek@seznam.cz*.

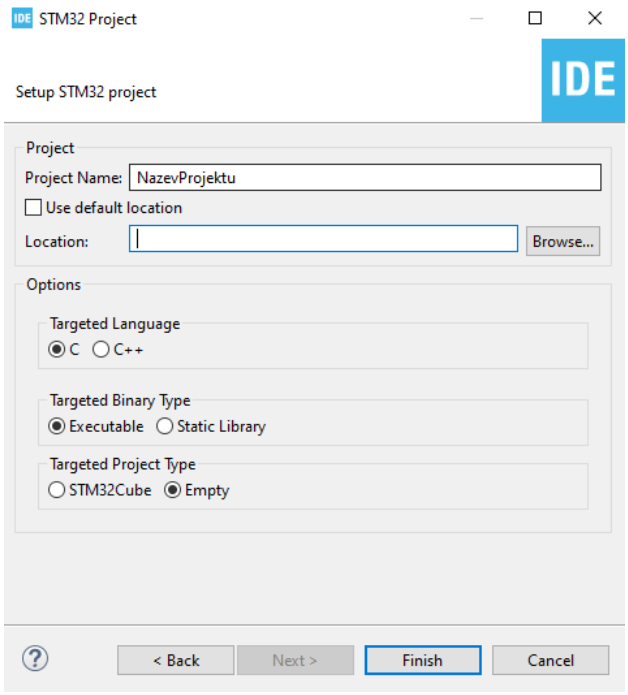
Po spuštění ***STM32CubeIDE*** otevřeme nabídku ***File->New->STM32 Project***.



Zobrazí se nabídka dostupných kontrolerů, v níž pomocí vyhledávače nalezneme kontroler ***STM32L073RZTx***. Stiskneme tlačítko ***Next***.

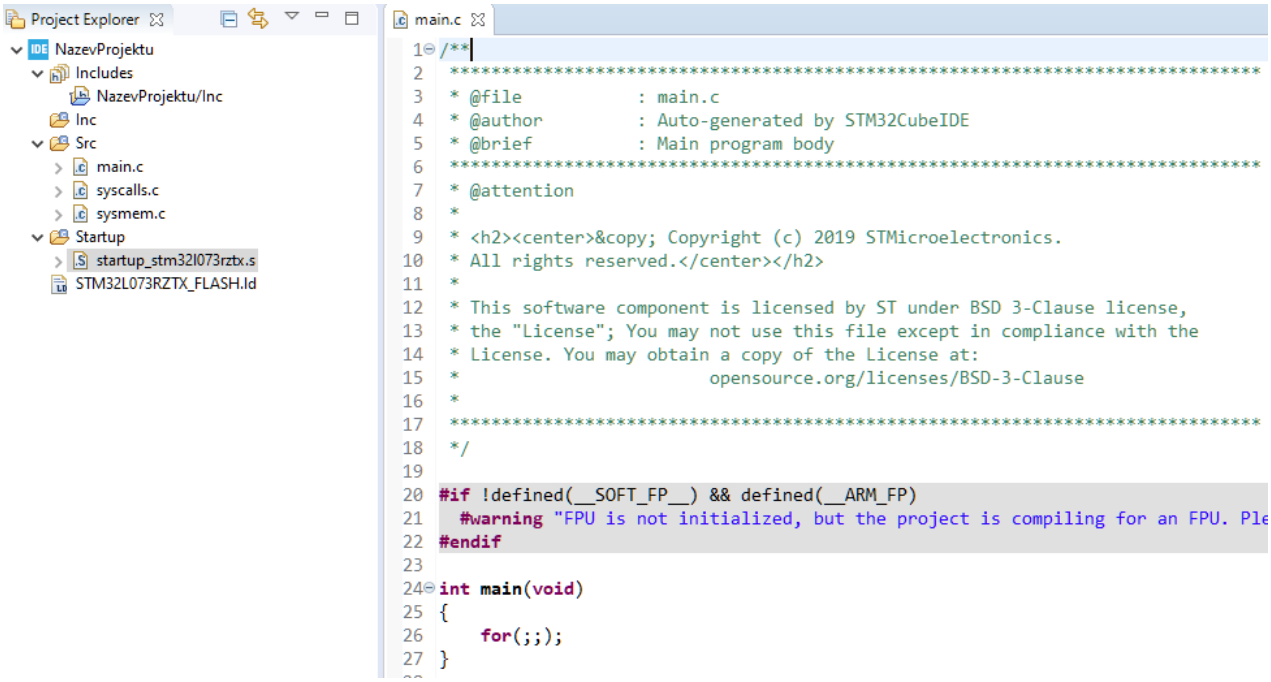


Tím se otevře poslední nabídka, v níž uděláme jednu změnu. V nejnižší nabídce ***Targeted Project Type*** místo možnosti ***STM32Cube*** zvolíme ***Empty***. Tím nastavíme, že nechceme v projektu používat grafický konfigurátor periferií ani oficiální knihovny.



Po zadání názvu a umístění projektu (POZOR! Cesta k projektu nesmí obsahovat českou diakritiku!) můžeme stisknout tlačítko ***Finish***.

Výsledkem je vytvoření zcela prázdného projektu, který obsahuje pouze základní soubory, jako je ***main.c***, bez jakékoliv funkcionality, a ***startup*** psaný v assembleru. Ten kontroler uvede do provozu.



Ve složce ***Inc***, v níž se vždy nacházely knihovny, je nyní prázdná. To jsme očekávali, ale přeci jen v ní něco potřebovat budeme. Jsou to definované názvy registrů kontroleru a jejich jednotlivých bitů. Dále budeme

potřebovat názvy registrů jádra ***ARM***. Tyto názvy nalezneme ve dvou souborech. Ty nejsnáze získáme tím, že založíme ještě jeden projekt. Budeme postupovat tak, jako bychom chtěli používat grafický konfigurátor a knihovny. To znamená, že v závěrečném okně předchozího postupu necháme nejnižší nabídce ***Targeted Project Type*** volbu ***STM32Cube***.

Vytvoří se nám druhý projekt, který již obsahuje knihovny. V něm najdeme ony dva soubory, jež potřebujeme.

Prvním je soubor ***stm32l073xx.h***. Ten obsahuje definice spojení s daným kontrolerem. Najdeme jej ve složce projektu ***Drivers->CMSIS->Device->ST->STM32L0xx->Inlcude***. Pokud jej za držení klávesy ***Ctrl*** přetáhneme do složky ***Inc*** našeho projektu bez knihoven, zobrazí se nám okno, kde potvrdíme, že chceme soubor zkopírovat.