Programujeme STM32

bez knihoven

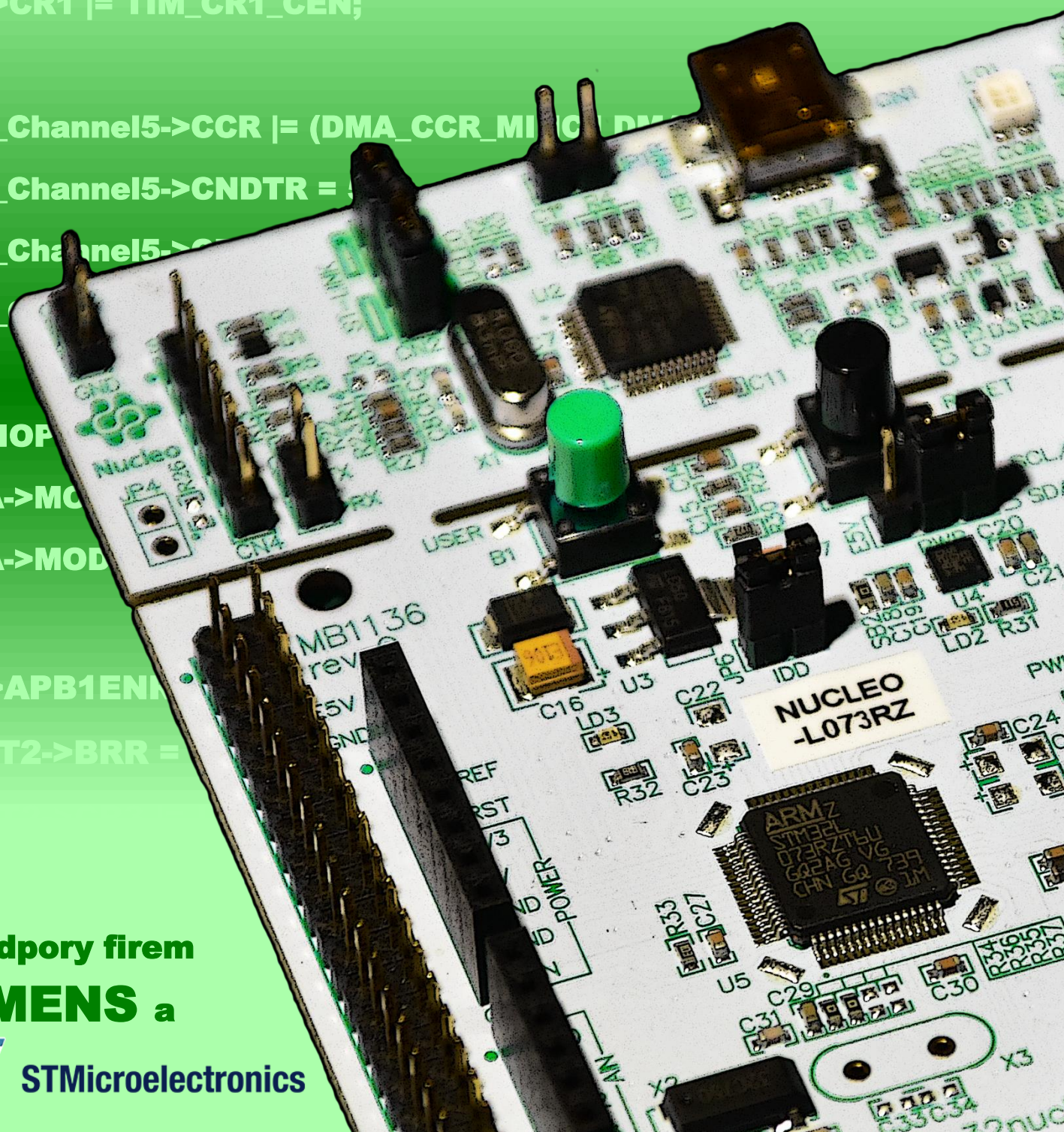
Ing. Vojtěch Skřivánek

```
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

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:

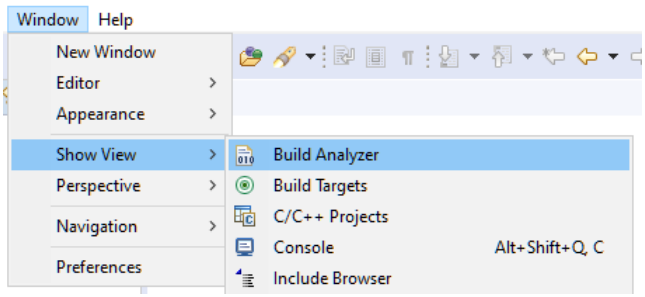
- Nastavení periferií,
- funkce periferií,
- obsluha přerušení periferií a
- tvorba programu.

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

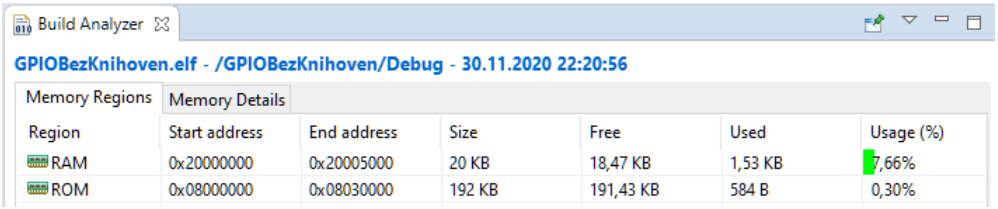
Důležité je upozornit, že kód některých funkcionalit neukazuje, jak danou problematiku řešit nejlépe, ale co možná nejjednodušeji. Účelem je snadné pochopení funkce periferií a programu. Stejně tak jsou všechny funkce v programu určené pouze pro jeden účel a jednu konkrétní periferii.

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***.

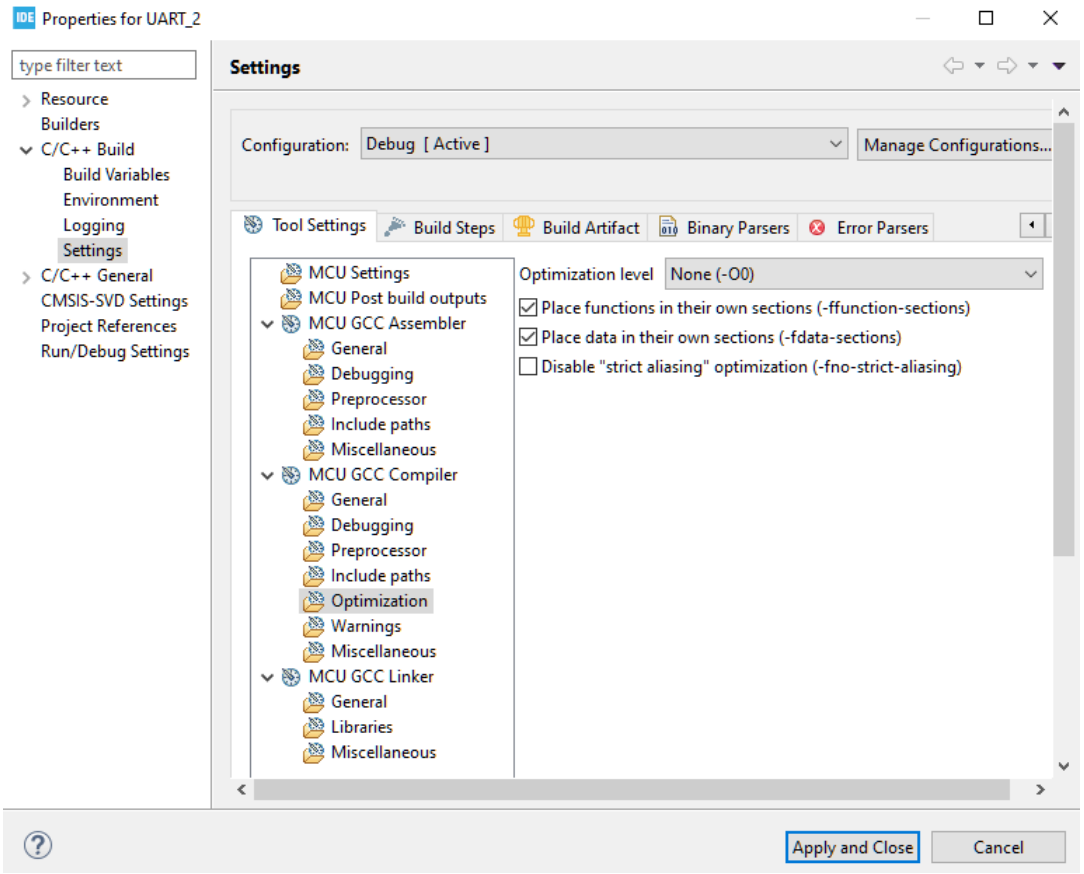


Po jeho otevření se nám zobrazí okno s informacemi o velikosti přeloženého programu.



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].