

Programujeme STM32

bez knihoven

Ing. Vojtěch Skřivánek

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

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

```
DMA1_Channel5->CCR |= (DMA_CCR_M10 | DMA_CCR_M11 | DMA_CCR_M12 |
```

```
DMA1_Channel5->CNDTR =
```

```
DMA1_Channel5->CNDTR =
```

```
DMA1_
```

```
RCC->IOP
```

```
GPIOA->MOD
```

```
GPIOA->MOD
```

```
RCC->APB1ENR
```

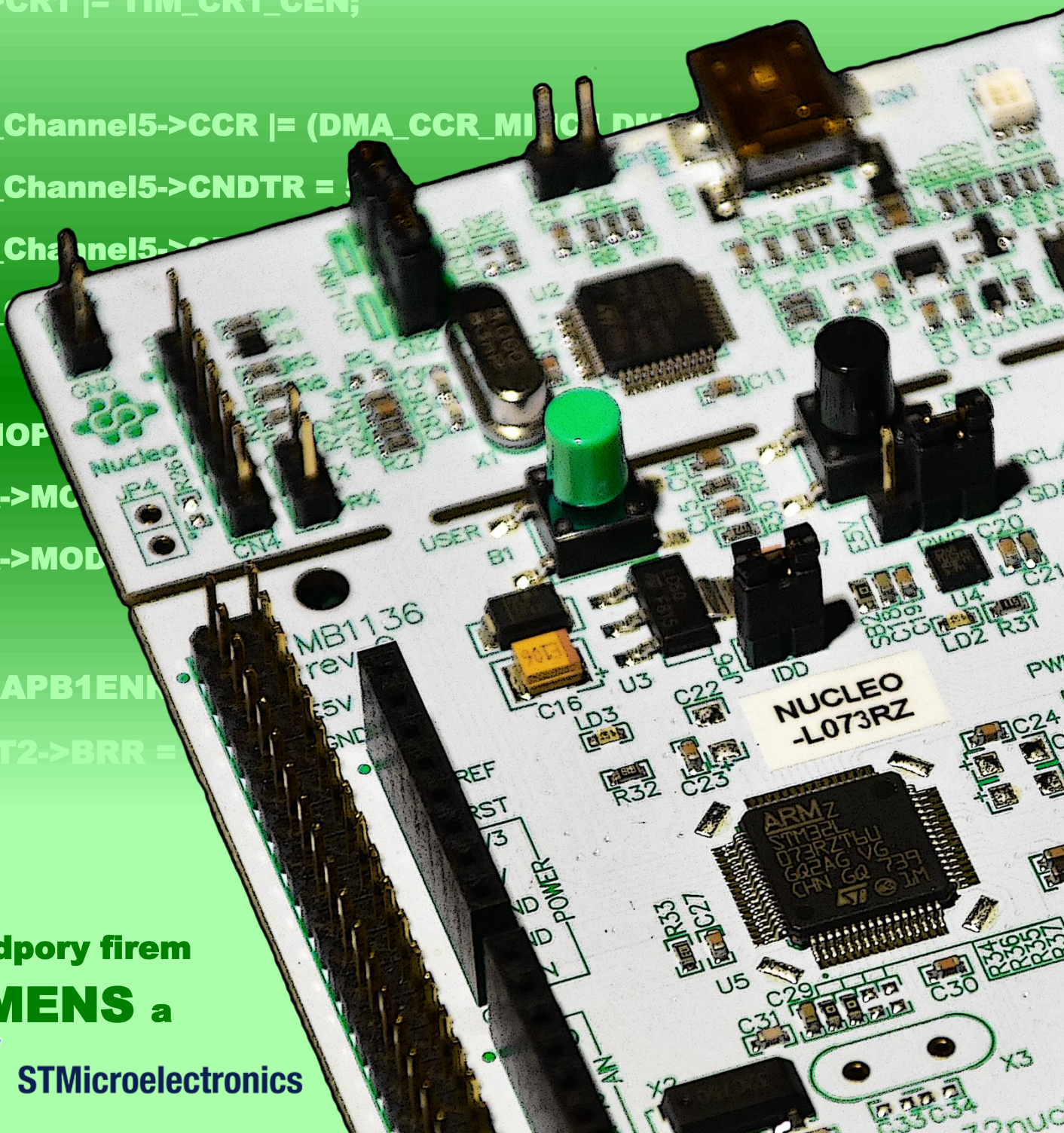
```
USART2->BRR =
```

Za podpory firem

SIEMENS a



STMicroelectronics



Programujeme STM32 bez knihoven

Ing. Vojtěch Skřivánek

Vydání první 2022

© Ing. Vojtěch Skřivánek

© Mgr. Tomáš Zahradníček - TZ-one

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 periférií	22
5.1.1.1	LED	22
5.1.1.2	EXTI	22
5.1.2	Funkce periférií	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 periférií	29
6.1.1.1	RCC - hodinový signál	29
6.1.1.2	TIM	31
6.1.2	Funkce periférií	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 periférií	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 periférií	38
6.3.1.1	TIM3	38
6.3.2	Funkce periférií	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 periférií	43
6.4.1.1	TIM	43
6.4.2	Funkce periférií	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 periférií	48
6.5.1.1	TIM2	48
6.5.1.2	TIM21	49
6.5.2	Funkce periférií	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 povel z počítače	53
7.1.1	Převodník sériové komunikace	53
7.1.2	Nastavení periférií	54
7.1.2.1	UART	54
7.1.3	Funkce periférií	55
7.1.3.1	UART - odesílání dat	55
7.1.3.2	UART - příjem dat	56
7.1.3.3	UART - callback příjmu	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 periférií	61
8.1.1.1	ADC	61
8.1.1.2	UART	63
8.1.2	Funkce periférií	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 periférií	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 periférií	75
9.1.1.1	DAC	75
9.1.2	Funkce periférií	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 periférií	79
10.1.1.1	SPI1	79
10.1.1.2	SPI2	81
10.1.2	Funkce periférií	82
10.1.2.1	SPI1 – vysílání a příjem v blokujícím režimu	82
10.1.2.2	SPI2 – vysílání 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 periférií	88
11.1.1.1	I2C1	88
11.1.1.2	I2C3	89
11.1.2	Funkce periférií	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 periférií	101
12.1.1.1	DMA	101
12.1.2	Funkce periférií	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 periférie do paměti a naopak	105
12.2.1	Nastavení periférií	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 periférií	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 periférie do periférie	111
12.3.1	Nastavení periférií	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 funkcionality, 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 periférií 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 periférií značně liší.

Nyní nechme všechny nevýhody knihoven stranou. Především s přihlédnutím k poslednímu bodu výhod knihoven může vyvstát 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 periférií. To v praxi sice znamená jejich náročnější inicializaci, ale poté jejich pohodlnější obsluhu. Navíc to, že periferie nabízí pokročilejší, více automatizované režimy (například bufferování příjmu dat digitální komunikace) neznámá, 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:

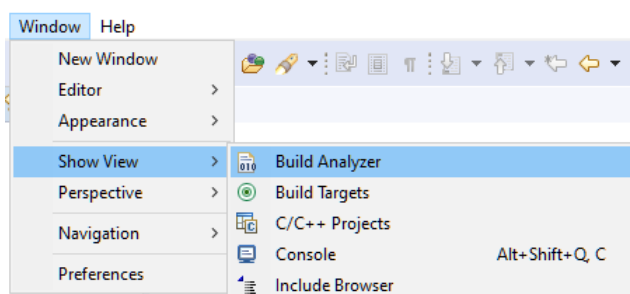
- Nastavení periferií,
- funkce periferií,
- obsluha přerušení periferií a
- tvorba programu.

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**.

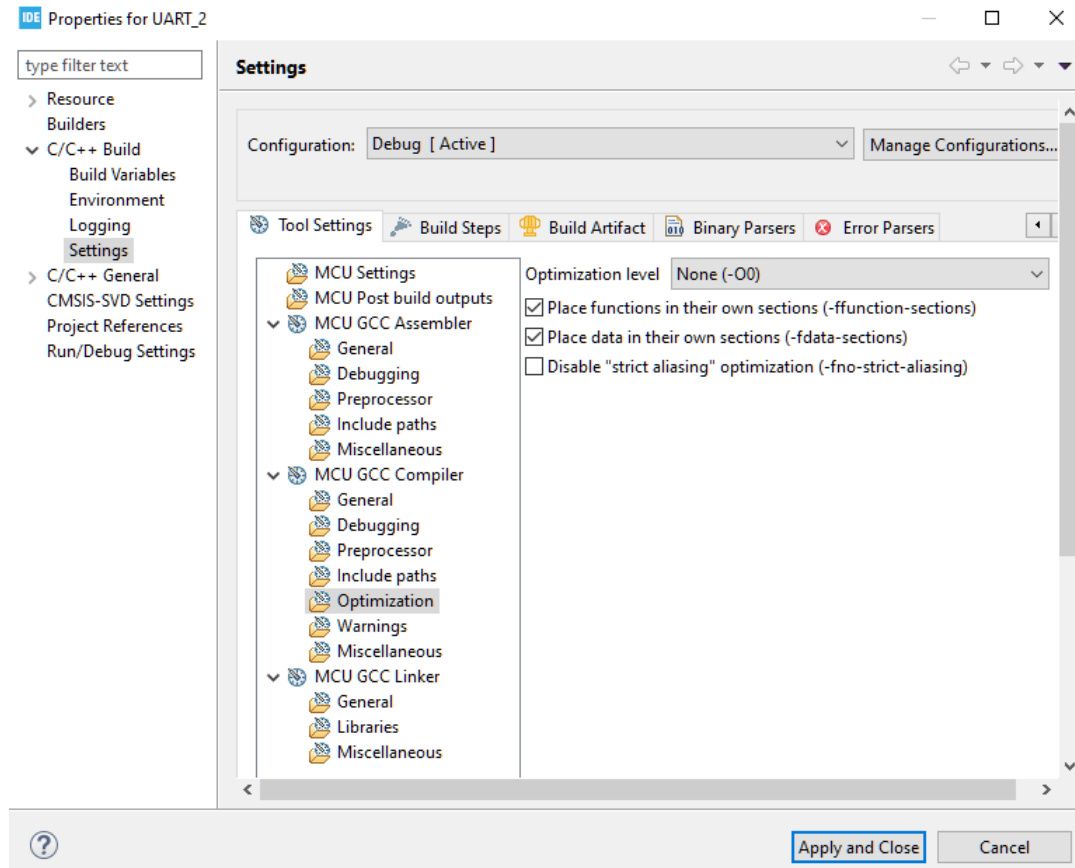


Po jeho otevření se nám zobrazí okno s informacemi o velikosti přeloženého programu.

GPIOBezKnihoven.elf - /GPIOBezKnihoven/Debug - 30.11.2020 22:20:56						
Memory Regions		Memory Details				
Region	Start address	End address	Size	Free	Used	Usage (%)
RAM	0x20000000	0x20005000	20 KB	18,47 KB	1,53 KB	7,66%
ROM	0x08000000	0x08030000	192 KB	191,43 KB	584 B	0,30%

Porovnání bude uvedeno pro tři různá nastavení optimalizace překladače 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].