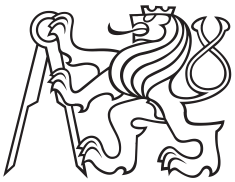
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra měření

Firmware pro měřicí přístroj s mikrořadičem STM32G431

Bc. Petr David

Vedoucí: doc. Ing. Jan Fischer, CSc.
Květen 2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **David** Jméno: **Petr** Osobní číslo: **420110**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Kybernetika a robotika**

Studijní obor: **Kybernetika a robotika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Firmware pro měřicí přístroj s mikrořadičem STM32G431

Název diplomové práce anglicky:

Firmware for measuring instrument based on microcontroller STM32G431

Pokyny pro vypracování:

V návaznosti na přístroj vyvinutý v rámci DP [1] vytvořte firmware pro mikrořadič STM32G431 tak, aby jej ve spolupráci s PC aplikací Zero eLab Viewer bylo možno využít jako jednoduchý, avšak komplexní měřicí přístroj pro výukové účely. V případě potřeby proveďte nutné úpravy PC aplikace. Přístroj bude zahrnovat funkce osciloskopu i se zobrazením průběhů logických kanálů, dále funkce impulsního a signálového generátoru, čítače a voltmetru se záznamem. Při návrhu firmware můžete též využít vhodné bloky vytvořené v rámci prací [2] a [3]. Výsledný přístroj otestujte a ověřte jeho parametry.

Seznam doporučené literatury:

[1] Berlinger, A.: „Implementace přístrojových funkcí mikrořadiči STM32“, diplomová práce ČVUT – FEL, 2016
[2] Cejp M.: „Virtuální přístroj s mikrořadičem pro analýzu signálu v modulační doméně“, bakalářská práce, ČVUT – FEL, 2017
[3] Dujava J., „Softwarově definované osciloskopy s terminálovým rozhraním“, diplomová práce ČVUT – FEL, 2022
[4] Cejp M.: „Virtuální přístroj s mikrořadičem pro analýzu signálu v modulační doméně“, bakalářská práce, ČVUT – FEL, 2017

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Jan Fischer, CSc. katedra měření FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **06.09.2022** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce:
do konce letního semestru 2023/2024

doc. Ing. Jan Fischer, CSc. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____ Datum převzetí zadání

_____ Podpis studenta

Poděkování

Děkuji ČVUT, že mi je tak dobrou *alma mater*.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 10. května 2023

Abstrakt

Abstrakt v češtině

Klíčová slova: slovo, klíč

Vedoucí: doc. Ing. Jan Fischer, CSc.

Abstract

Abstract in English

Keywords: word, key

Title translation: Firmware for measuring instrument based on microcontroller STM32G431

Obsah

Obrázky

Tabulky

ctuthesis t1606152353

Kapitola 1

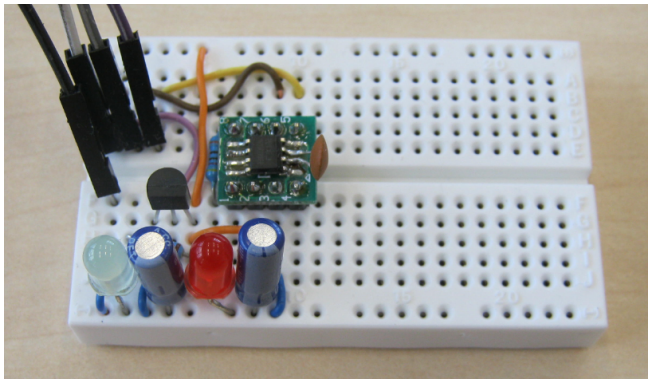
Úvod

Kapitola 2

Rozbor

2.1 PC aplikace Zero Elab Viewer

V roce 2016 v rámci své diplomové práce pan Ing.Adam Berlinger vytvořil aplikaci Zero Elab Viewer fungující jako grafické rozhraní pro komunikaci se softwarově definovanými nástrojem(SDI) založeném na STM32F042. Aplikace společně s FW umožňuje používání MCU jako dostupnou náhradu za laboratorní přístroje jako jsou voltmetr, osciloskop, impulsní generátor, signálový generátor nebo logický analyzátor. Aplikace se od té doby stále používá v hodinách praktické elektroniky a od té doby vznikla ještě řada implementací FW pro jiné MCU. Nejnověji například verze pro mikrokontrolér ATMega 328 pro lidi se zkušenostmi s ARDUINO nebo zatím nejmenší(a nejlevnější) varianta SDI přístoje vzniklá na katedře měření využívající 8-pinovou variantu mikrokontroleru STM32G030.



Obr. 2.1: STM32G031 zapojen na nepájivém poli převzato z [3]

2.1.1 Existující FW

Zatím poslední zveřejněný FW pro již zmíněný mikrokontrolér STM32G030 implementuje přístroje s následujícími parametry [3]:

- Osciloskop- záznam až 2048 vzorků, rozlišení 12 bitů,
- Rychlost záznamu až 1x 2 MS/s (nebo 2x 1 MS/s, 3 x 666 kS/s).
- Ve stroboskopickém módu - až 64 MS/s - rozlišení s intervalem 15,6 ns.
- Impulsní generátor PWM , nastavení střídy 0 až 100 %; nastavení frekvence 1 Hz až 32 MHz
- Voltmetr tři kanály, 0 až + 3,3 V, 100 odměrů/s, průměrování z 1 až 256 odměrů, možnost funkce záznamu

2.2 Použitý MCU STM32G431

Navržený mikrokontrolér s jádrem Arm® 32-bit Cortex®-M4 může běžet až s maximální frekvencí jádra 170MHz a tak ve srovnání s předchozími implementacemi na STM32F042(48MHz) a STM32G030(64MHz) nabízí výrazně větší výpočetní výkon. Navíc obsahuje více dostupných periferií vhodným pro implementaci SDI jako jsou 2 ADC převodníky či větší počet periferií čítačů(14 vs 8 u G030).

Kapitola 3

Realizace FW

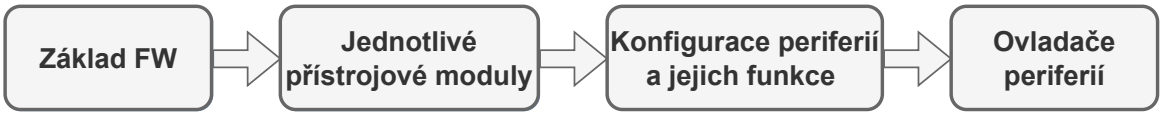
3.1 Struktura FW

Jak vyplývá ze zadání, firmware vznikal v návaznosti na existující přístroj využívající stávající PC aplikací Zero eLab Viewer a jíž touto skutečností byl návrh firmware mikrokontroléru částečně vymezen. V zájmu zpětné kompaktibility by totiž úpravy aplikace neměly ohrozit fungování předchozích implementací na jiných mikrokontrolérech, kterých je již celá řada. Především tedy komunikační protokol, který i když je v některých případech limitující, nemohl být upraven. Dále pak aplikace určuje, jaké softwarově definované přístoje lze implementovat a jaké budou mít možnosti ovládání či nastavení. Dále jsem se při návrhu zaměřil na tyto body

- **Samostatná využitelnost jednotlivých přístrojových bloků**
  Rozdělit implementaci jednotlivých přístrojů do samostatně fungujících bloků je výhodné z 2 důvodů. Zaprvé lze tak jednodušeji výsledný FW přizpůsobit pro různé mikrokontroléry v závislosti na dostupných periferií a velikosti FLASH paměti.Druhým důvodem je pak možnost použití kódu i v jiných aplikacích.

- **Jednoduchá záměna HW prostředků**
  Bylo žádoucí aby vznikající FW byl flexibilní co se týče použitých HW prostředků a tedy bylo jednoduché upravovat například použité piny, DMA kanály či čítače. Toto dále zjednodušuje další adaptaci firmware pro jiné MCU.

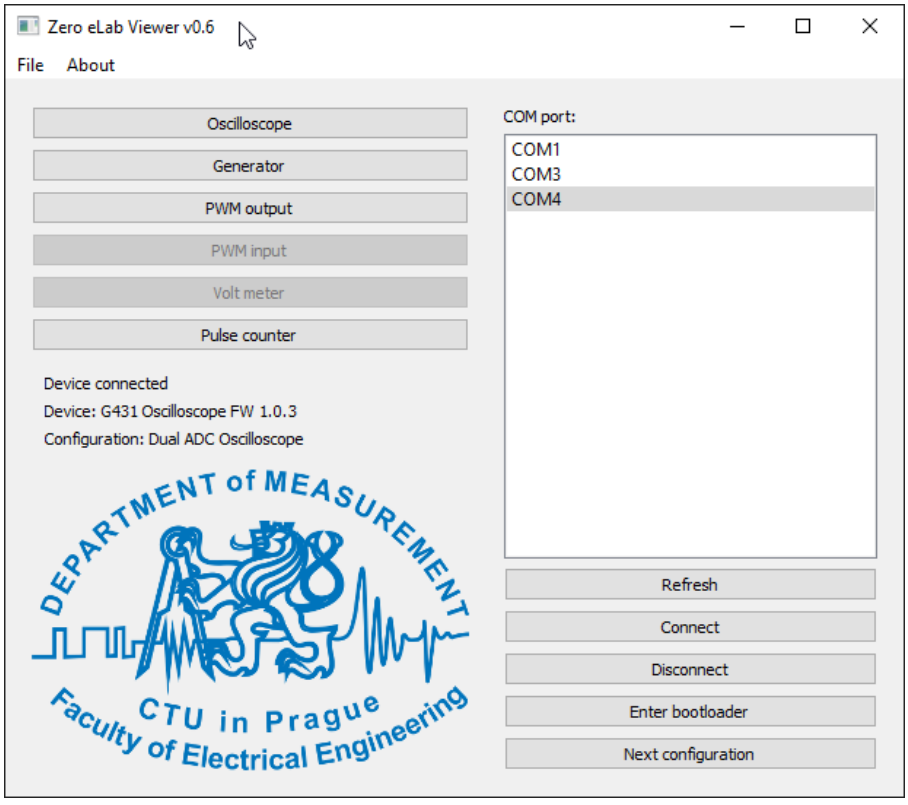
- **Kompaktibilita s generátorem inicializačního kódu**
  Nástroj STM32CubeMX velmi usnadňuje začátek vytváření FW díky využití grafického rozhraní pro definování počáteční konfigurace MCU. Pokud se zachová struktura generovaného kódu, lze pak nástroj opětovně využít v případě, že chceme jednoduše upravit HW konfiguraci.



Obr. 3.1: Kategorie zdrojového kodu a jejich posloupnost závislosti

3.1.1 Využití možnosti přepínání mezi konfiguracemi

Především z důvodů omezených hardwarových prostředků na různých mikrokontrolérech je využito možnosti přepínání různých přístrojových konfigurací. Konfigurací v tomto případě myslíme set nástrojů, které mohou být použity současně. Přístrojové moduly pak mohou sdílet periferie za předpokladu, že v dané konfiguraci je aktivní pouze jeden z těch modulů, který má přístup k dané periferii. Příkladem může být sdílení ADC převodníku modulem Voltmetru a modulem Osciloskopu, které pak tedy nemohou být aktivní zároveň. Jako je vidět na obrázku 3.2 hlavní okna PC aplikace.



Obr. 3.2: Hlavní okno aplikace Zero eLab Viewer zobrazující aktivní moduly dané konfigurace

3.1.2 Využití LL ovladačů

Jedná se o hardwarově orientovanou knihovnu dodávanou pro STM32 mikrokontrolery. Její využití je velmi podobné využití CMSIS ovladačů s tím rozdílem, že nabízí určité rozšířené možnosti por-

tovatelnosti mezi jednotlivými rodinami mikrokontrolerů a sadu rozšiřujících API pro zjednodušení implementace některých úkonů jako je například inicializace periferií. K použití teto knihovny je oproti pravděpodobně známější knihovně HAL zapotřebí znalost jednotlivých periferií, jelikož velká část definovaných funkcí je pouze jednořádková modifikace registrů bez kontrol vstupních paramentrů a uživatel tedy musí být více obeznámen s tím co dělá. Výhodou oproti HAL je výrazně menší paměťová náročnost.

Pro vytvoření představy o rozdílnosti paměťové náročnosti jednotlivých knihoven jsem vytvořil pomocí STM32CubeMX dva minimální projekty, jejichž úkolem bylo nastavení systémových hodin blikání jednou LED. V prvním případě byly periferie RCC(Reset and clock control) a GPIO inicializovány prostřednictvím HAL knihovny a v druhém případě pomocí LL ovladačů a byly použity odpovídají zpožďovací funkce společně s přepínáním výstupu pinu. Výsledkem bylo, že po zkompilování verze s HAL knihovnou využívala asi o 86 procent více FLASH.

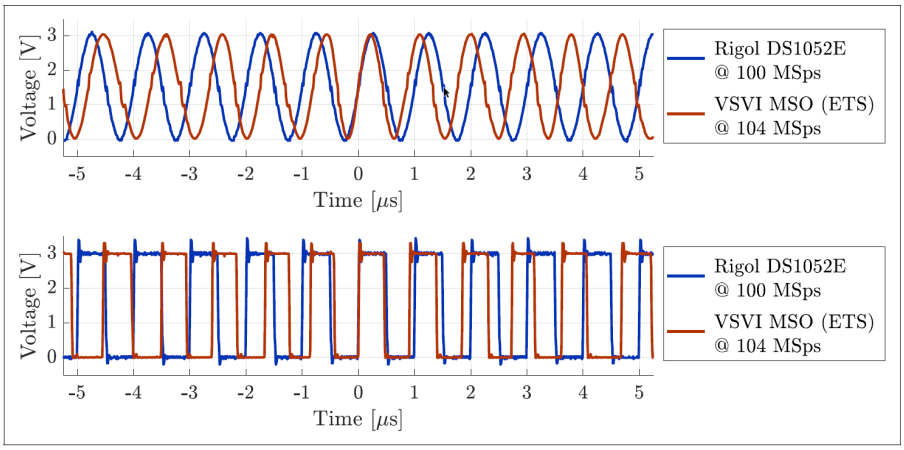
	FLASH	RAM
LL ovlačače	3.04 kB	1,53 kB
HAL ovladače	5,66 kB	1,55 kB

Tab. 3.1: Srovnání paměťové náročnosti jednoduchého programu s využitím různých knihoven

3.1.3 Core coupled memory CCM SRAM

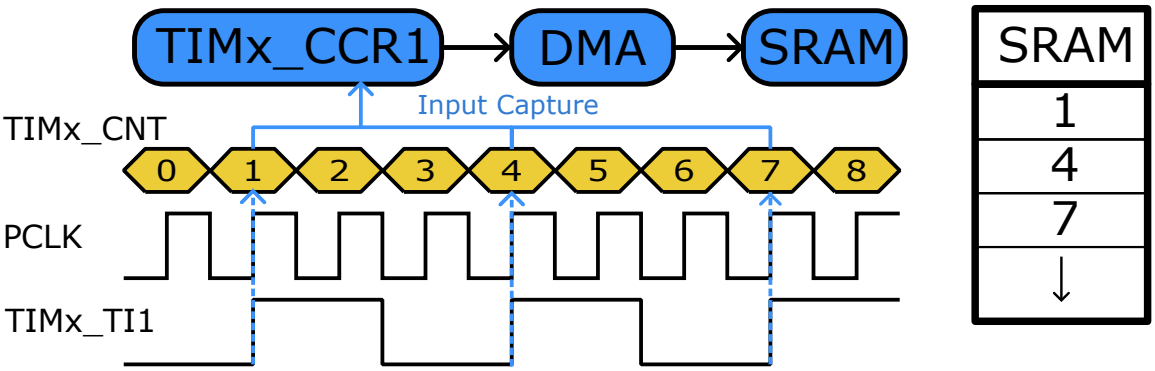
3.2 Rozpoznání frekvence externího krystalu HSE

Pro chod mikrokontroléru je zapotřebí zdroj hodinového signálu pro generování systémových hodin(System Core clock) dále jen SYSCLK. Jako základní varianta zdroje hodinového signálu se používá interní vysoko-rychlostní oscilátor(HSI), jehož výstupní frekvence ve srovnání s externími zdroji hodinového signálu vykazuje řádově vyšší nepřesnost a vyšší závislost na změnách teplot viz srovnávací tabulka 3.2.Tento rozdíl je pak obzvlášť podstatný při realizaci funkce osciloskopu v režimu vzorkování v ekvivalentním čase(ETS), kde dochází k výraznému zkreslení měřeného signálu viz obrázek 3.4. Na tomto obrázku je zobrazen zkreslený záznam signálu s G431 využívajícím HSI jako zdroj hodinového signálu a druhá stopa je měřena osciloskopem Rigol DS1052E jehož přesnost vzorkovací frekvence je $\pm 0.005\%$ [5]. Z obrázku je zřejmá vhodnost použití zdroje hodinového signálu s vyšší přesností než vykazuje HSI.



Obr. 3.3: Zkreslení průběhu měřeného signálu v důsledku nestability HSI převzato z [2]

Jako zdroj vysokorychlostního externě získaného hodinové signálu(HSE) lze buď využít krystalu buzeného pomocí MCU(HSE crystal) nebo jiného externího zdroje signálu(HSE bypass). Případný externí signál musí pak splňovat nějaké podmínky a to například: pro STM32G431 musí být v rozsahu 4-48MHz a mít střídu 40-60%. Ve výuce laboratorních měření na katedře měření jsou k dispozici krystaly různých výstupních frekvencí převážně pak 8 MHz, 12MHz a 16MHz.Pro účely co nejflexibilnějšího laboratorního přístroje se zdálo účelné naprogramovat firmware pro použití s různými oscilátory tohoto typu.Tedy aby funkce FW nebyla závislá na přítomnosti krystalu ani jeho výstupní frekvenci. Toho bylo docíleno změřením výstupní frekvence oscilátoru a nastavení výsledné frekvence systémových hodin pomocí interního obvodu fázového závěsu(PLL), tak aby výsledná frekvence SYSCLK nebyla na použitém krystalu závislá.



Obr. 3.4: Měření periody vstupního signálu pomocí "Input-capture"a DMA

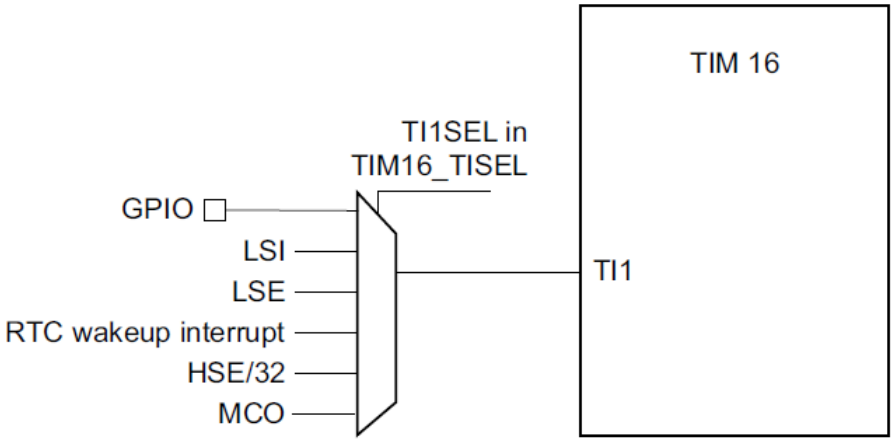
Krátkodobá nestabilita frekvence (jitter)

	Δf f	Δ
STM32G431 HSI 16 MHz	± 1%	±1%
Adafruit krystal 16 MHz	±0.003%	±0.005%

Tab. 3.2:

3.2.1 Měření frekvence HSE

Existují různé způsoby měření frekvence externího hodinového signálu, ale jako nejúčelnější se v tomto případě zdálo použití čítače v režimu "Input capture"(IC) a měření délky periody externího signálu. Na rodině mikrokontrolérů STM32G4 mají čítače TIM16 a TIM17 možnost interně přivést HSE již se sníženou frekvencí. Frekvence HSE je totiž ještě před přivedením na vstup čítače zpracovaná obvodem, který frekvenci 32krát sníží. Pro tento vstup čítače se sníženou frekvencí se pak používá označení HSE32 jako je vidět na obrázku 3.5 z dokumentace.



Obr. 3.5: Vstupy dostupné na kanálu číslo 1 čítače TIM16. Převzato z[6]

Měření periody signálu HSE32 probíhá potom tak, že měříme počet cyklů čítače mezi jednotlivými náběžnými hranami nebo sestupnými hranami. Tento počet cyklů nám pak určuje poměr mezi frekvencí externího hodinového signálu na vstupu f_{IN} a hodinového signálu, který pro svůj chod využívá periferie čítače f_{TIM}. Pro správné měření je tedy podstatné, aby interní hodinový signál čítače byl výrazně vyšší než frekvence na měřeném vstupu. Pro získání přesnějšího odhadu vstupní frekvence můžeme zaznamenat více hodnot hodnot po sobě a ty zprůměrovat. Frekvence vstupu je poté rovna:

$$f_{\mathrm{IN}} = \frac{f_{\mathrm{TIM}}}{N_{\mathrm{p}}} \tag{3.1}$$

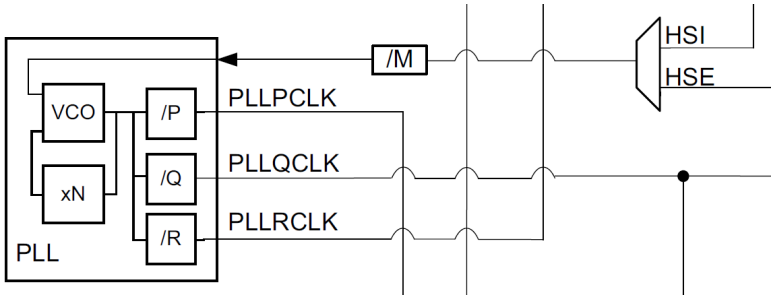
Pro určení frekvence HSE pak ještě musíme získanou hodnotu vynásobit 32:

$$f_{\mathrm{HSE}} = 32 \cdot f_{\mathrm{IN}} \tag{3.2}$$

3.2.2 Využití PLL

Obvod PLL je další z možných zdrojů hodinového signálů systémových hodin, který má navíc programovatelné dělení a násobení vstupní frekvence. Jako vstup pak lze použít HSI nebo HSE ve stanovené rozsahu. Například 2.66-16MHz pro stm32G431[8]. Dle obrázku 3.6 lze vidět, jak vstupní hodinový signál vstupujícího do PLL bloku nejdříve prochází přes děličku signálu M, dále se hodinový signál

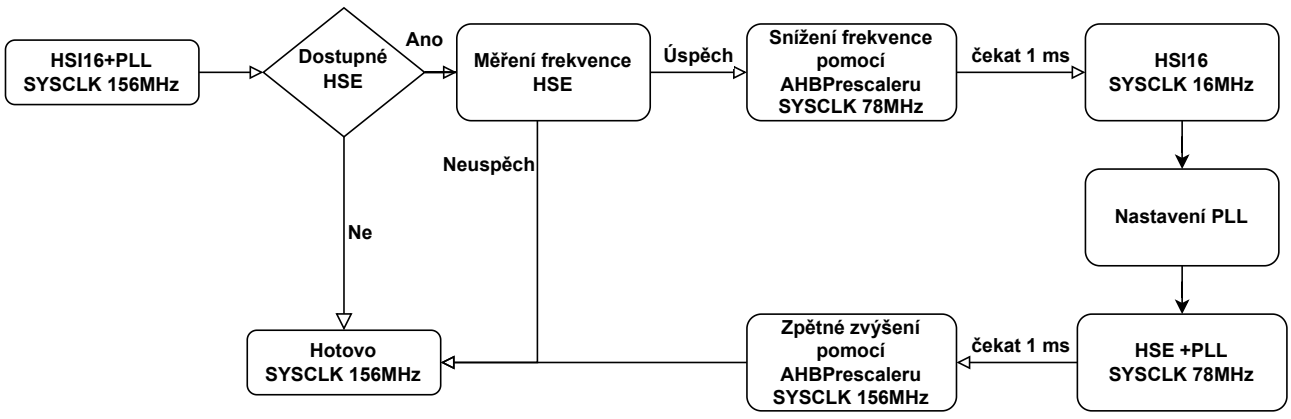
násobí N a tento signál jde pak na 3 různé výstupy s vlastními děličkami. Dále v kapitole o ADC ukazuji výhodu této možnosti více výstupů z obvodu PLL, díky které vstupní hodiny ADC nemusí být závislé na frekvenci systémových hodin.



Obr. 3.6: Vyobrazení interního PLL bloku. Převzato z

3.2.3 Změna zdroje hodinového signálu systémových hodin

Na obrázku 3.7 je popsaný postup změny zdroje SYSCLK. Při změně vstupního signálu PLL nelze PLL používat, tedy je nejdříve zapotřebí přepnout systémové hodiny na interní oscilátor 16MHz. Dle doporučení v [6] je při velkých rozdílech frekvencí mezi výstupem PLL(SYSCLK>80MHz) zapotřebí přidat mezikrok s využitím AHB předděličky hodinového signálu systémových hodin. hodnota předděličky se nastavuje RCC_CFGR registru. Díky tomu například ve svém programu zmenším frekvenci systémových na polovinu tedy 78 MHz a pak až nastavuji jako zdroj HSI146. Doporučená doba setrvání v tomto mezikroku je alespoň 1μs. V mém řešení program čeká 1ms s využitím připravených funkcí obsahujícím čekání v jednotkách ms.



Obr. 3.7: Postup změny zdroje hodinového signálu

3.3 Realizace funkce Voltmetru

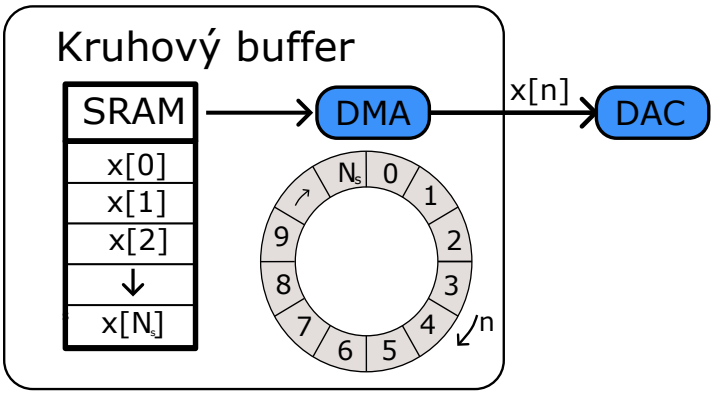
3.3.1 Nejistota měření

3.4 Realizace měření frekvence a střídy

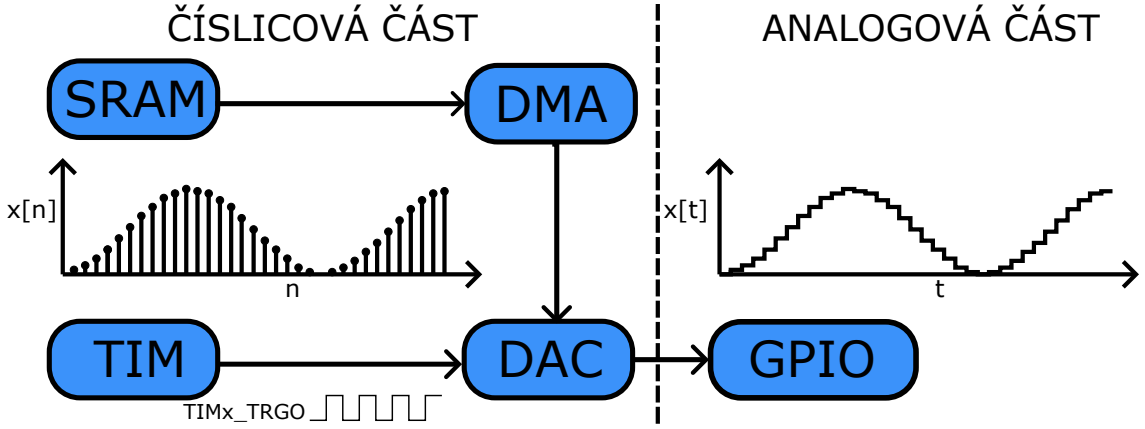
3.5 Realizace generátoru funkcí

3.5.1 Generování signálu

Jedním ze způsobů jak generovat průběhy různých signálů je přichystáním jednotlivých vzorků do paměti SRAM a s využitím DMA data použít jako kruhového bufferu (Obr 3.8). DMA pak může cyklicky posílat data do DAC převodníku bez vlivu na výpočetní výkon mikrokontroléru.



Obr. 3.8: Popis funkce kruhového bufferu jako zdroj datových vzorků pro DAC převodník



Obr. 3.9: Diagram využití DAC převodníku a dalších periferií pro generování analogového signálu

Optimalizace délky bufferu

Obvyklý postup je si stanovit počet vzorků signálu N_s na jednu periodu generovaného signálu v závislosti na velikosti dostupné paměti a určit frekvenci f_{out} s jakou se generovaný signál má opakovat. Poté se obvykle stanovuje dělící poměr DIV pro čítač časové základny vzorkování v závislosti na frekvence vstupních hodin toho čítače f_{PCLK}.

$$\mathrm{DIV} = \frac{f_{\mathrm{PCLK}}}{f_{\mathrm{out}} \cdot N_{\mathrm{s}}} \tag{3.3}$$

Nicméně zde narážíme na limit přesnosti nastavení výstupní frekvence signálu v podobě nastavení dělícího poměru, který může nabývat pouze celočíselných hodnot. Pro zlepšení rozmezí nastavitelných frekvencí výstupního signálu jsem se inspiroval článkem [4] a do svého řešení adaptoval optimalizaci délky bufferu, která umožňuje iterativním zkracováním délky bufferu minimalizovat chybu výstupní frekvence.

V rovnicích 3.4 uvádím vztahy pro výpočet reálné výstupní frekvence f_{out}, která je tedy závislá na počtu vzorků v kruhovém bufferu (N_s) a vzorkovací frekvenci (f_s). Vzorkovací frekvence je pak zase podílem frekvence vstupních hodin čítače f_{PCLK} a dělícího poměru DIV.

$$f_{\mathrm{out}} = \frac{f_{\mathrm{s}}}{N_{\mathrm{s}}}, \quad f_{\mathrm{s}} = \frac{f_{\mathrm{PCLK}}}{\mathrm{DIV}} \quad \Longrightarrow \quad f_{\mathrm{out}} = \frac{f_{\mathrm{PCLK}}}{\mathrm{DIV} \cdot N_{\mathrm{s}}}, \quad f_{\mathrm{s}} = \frac{f_{\mathrm{out}}}{\mathrm{DIV} \cdot N_{\mathrm{s}}} \tag{3.4}$$

Dělící poměr DIV je závislý na nastavení ARR(auto-reload register) a PSC(Prescaler Register) registrech čítače. Zde je podstatné uvést, že ke skutečným hodnotám registrů ARR a PSC se musí přičíst 1 abychom získali správnou hodnotu jake ve vidět ve vzorci 3.5 Při optimalizaci délky bufferu tedy v prvním kroku nehledáme jen dělící poměr DIV, ale faktorizaci tohoto dělícího čísla, která nám minimalizuje rozdíl mezi cílenou a vzorkovací frekvencí f_s a reálnou vzorkovací frekvencí dosažitelnou

při daných vstupních hodinách čítače. Pro nalezení optimálních hodnot ARR a PSC jsem využil algoritmu popsaného v práci [2].

$$\mathrm{DIV} = (\mathrm{ARR} + 1) \cdot (\mathrm{PSC} + 1) \tag{3.5}$$

Algoritmus spočívá v iterativním zvyšování hodnoty PSC a dopočtu vhodných adeptů na ARR registr. Jeho výsledkem jsou pak 2 existující faktorizace takové, že použitím jedné faktorizace získáme frekvenci menší nebo rovnu požadované a druhou faktorizací frekvenci vyšší nebo rovnu. Porovnáním dosažitelných frekvencí pak můžeme zvolit faktorizaci, která nám zajistí menší odchylku od požadované frekvence. Kompletní popis algoritmu je dostupný v citované práci.

Dále jsem při optimalizaci délky bufferu postupoval podle pseudo-kódu uvedeného v [4]. Nejdříve jsem určil maximální délku bufferu v závislosti na maximální vzorkovací frekvenci DAC převodníku a požadované frekvenci signálu. Následovně jsem cyklicky zmenšoval délku bufferu. Pro každou délku bufferu jsem pomocí výše již zmíněného algoritmu získal DIV a vypočítal odchylku od požadované frekvence signálu. Pokud byla odchylka menší než nejmenší zatím dosažena uloží se DIV a algoritmus pokračuje dále dokud nenastane jedna z ukončujících podmínek. Těmi je buď dosažení odchylky menší než 0,1Hz nebo zkrácení výsledné délky bufferu na 1/4 původní délky.

Při testování funkčnosti optimalizace se potvrdili pozorování uvedené [4] a to, že touto metodou lze získat poměrně velké přesnosti při frekvencích do 500 Hz. S tím jak dosáhneme frekvence 1000 Hz, tak se začne maximální velikost bufferu snižovat a zřejmě s tím se začne snižovat možnost optimalizace frekvence signálu. Algoritmus jsem testoval za následujících podmínek:

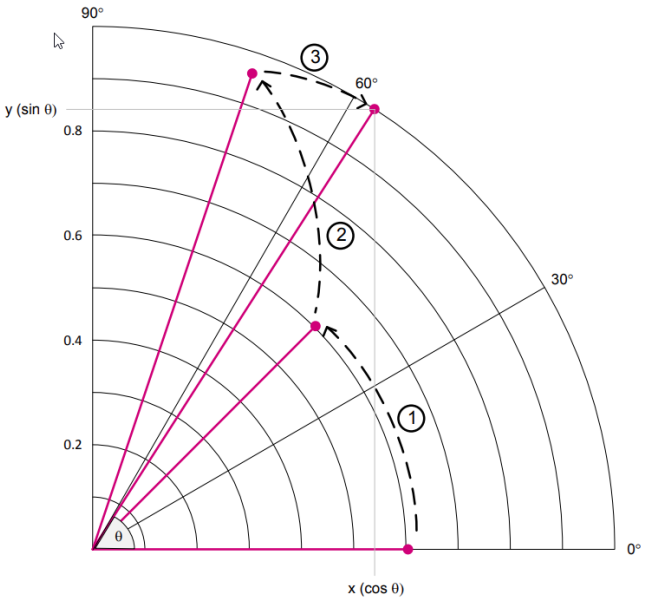
- Frekvence vstupního signálu po krocích 1 Hz
- Maximální vzorkovací frekvence DAC převodníku 1 Msps
- Maximální délka bufferu - 1000
- Frekvence vstupních hodin periferie čítače- PCLK 156 MHz
- Změřené odchylky jsou pouze teoretické odchylky neuvažující nepřesnost frekvence krystalu.

	Maximální absolutní odchylka		Maximální relativní odchylka	
Rozsah frekvencí	Odchylka [Hz]	Frekvence[Hz]	Odchylka [%]	Frekvence[Hz]
1-500 Hz	0,0058	477	0,0015	314
500-1000 Hz	0,0212	977	0,0022	977
1000 - 2500 Hz	0,1894	2361	0,0087	1951
2500 - 5000 Hz	0,8407	4683	0,0179	4683
5000 - 1000 Hz	31,694	9968	0,3179	9968

Tab. 3.3: Maximálních dosažené odchylky frekvencí výstupních signálů pro různá rozmezí frekvencí

Využití CORDIC

CORDIC(COordinate ROtation DIgital Computer) je HW akcelerátor designovaný pro urychlení kalkulací vybraných matematický funkcí jako jsou trigonometrické a hyperbolické funkce.[10]. Pro získání výsledku používá metody postupné aproximace a případě trigonometrických funkcí algoritmus konverguje pro hodnoty $-\pi$ až π radiánů. Algoritmus využívá číselné reprezentace s pevnou řádovou čárkou. Výpočet funkce sinu daného úhlu pak probíhá postupnými rotacemi počátečního vektoru o postupně se zmenšující úhly.S tím, že HW implementace používá pro výpočet pouze operace sčítání, posuvu a pevného scaling faktoru. Výsledek je pak připravený ve fixním počtu kroků v závislosti na požadované přesnosti výsledku. Průběh kalkulace je pak vidět na obrázku.



Obr. 3.10: Výpočet sinus a cosinu uhlu postupnou aproximací použitou v CORDIC převzato z [7]

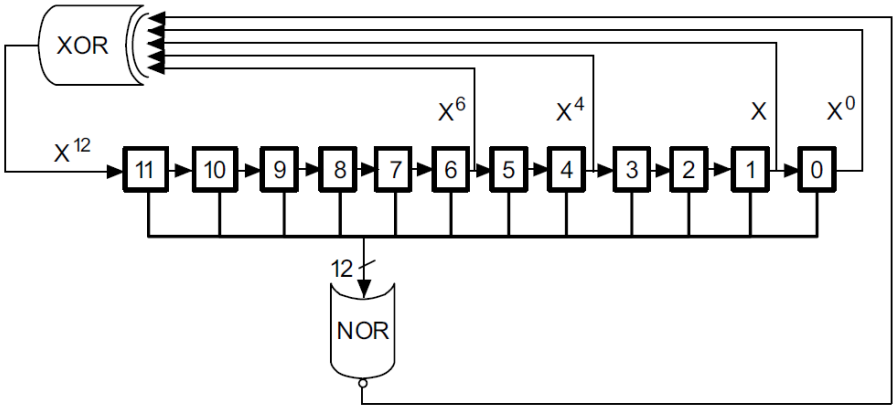
Ve své práci jsem využil přítomnosti této periferie pro urychlení výpočtu hodnot sinus při přípravě bufferu pro generování sinusového signálu. Pro porovnání časové náročnosti výpočtu hodnot bufferu jsem buffer o 1000 vzorcích nejdříve vypočítal pomocí CORDIC a pro porovnání jsem pak využil výpočtu pomocí běžné funkce `sinf(x)` z knihovny `<math.h>`. Výsledná doba měření pak byla za celý výpočet bufferu tedy kromě samotného výpočtu výpočty sinu postupně se zvyšující fáze, tak zároveň vynásobením spočítané hodnoty podle nastaveného rozkmitu a posun dle nastaveného offsetu. Navíc v případě použití CORDIC je nejdříve zapotřebí hodnoty převést do formátu s pevnou desetinou čárkou q1.31 a následně zpět, protože periferie s jiným formátem dat neumí pracovat. Uběhlý čas jsem pak porovnával při nastavené optimalizaci kodu se zaměřením na rychlost (-OSpeed). Výsledkem bylo že výpočet pomocí CORDIC byl v průměru asi o 120% rychlejší.

Použitá metoda	Doba plnění bufferu
CORDIC	37541 cyklů
sinf(x) z math.h	85573 cyklů

Tab. 3.4: Doba trvání přípravy bufferu o délce 1000 vzorků v závislosti na použité metodě

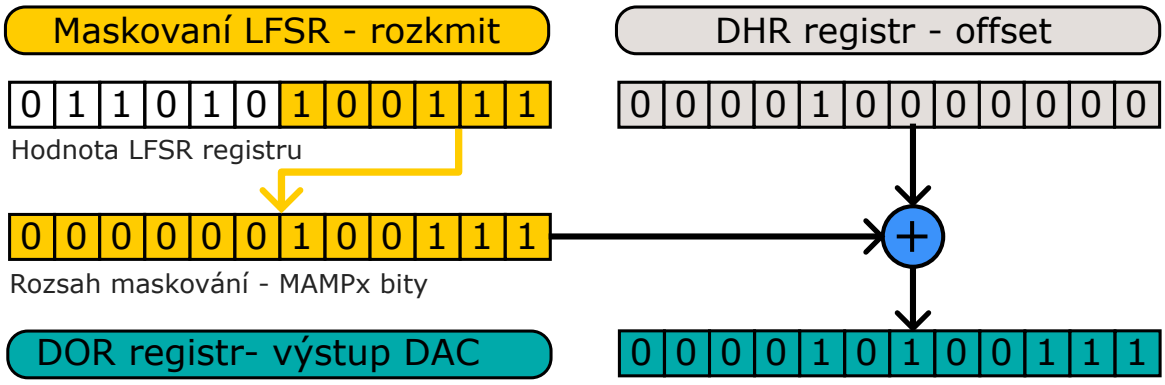
3.5.2 Generování šumu

DAC převodník přítomný u MCU rodiny STM32G4 má implementován LFSR(linear feedback shift register) umožňující pomocí převodníku generování pseudo-šumu o nastavitelném offsetu a rozkmitu. Hodnota registru je vždy přepočítána po triggerování DAC převodníku podle specifického algoritmu popsaného na obrázku 3.11. Generování šumu ADC převodníkem může být využito pro různé demonstrace



Obr. 3.11:

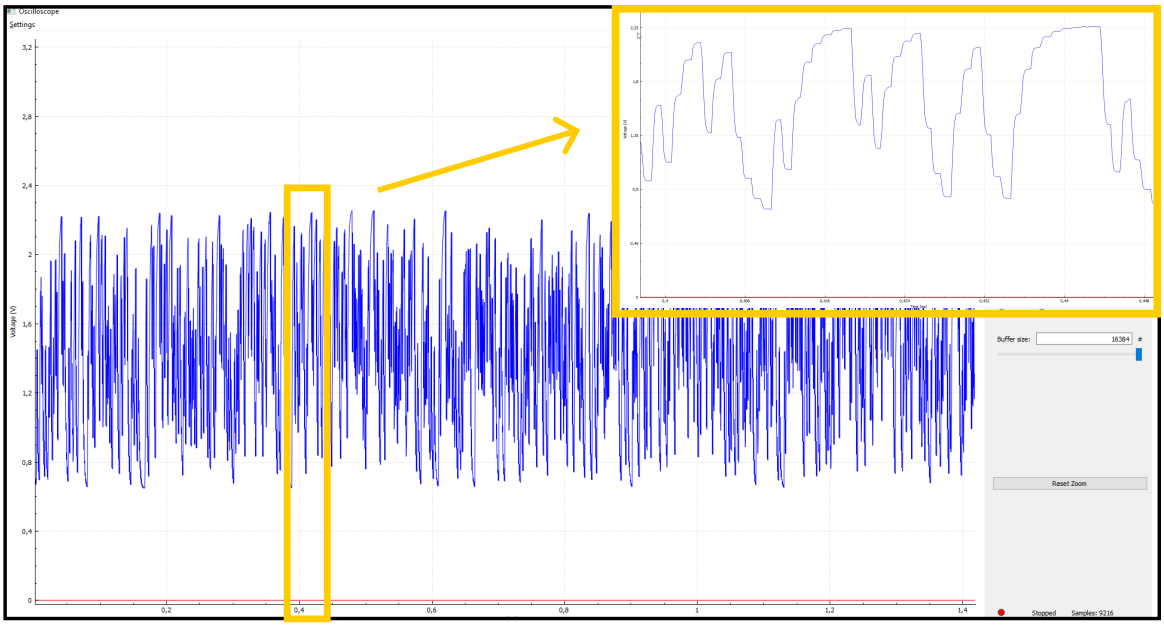
Samotné využití LFSR v DAC převodníku spočívá maskováním bitů tohoto registru pro získání určitého rozmezí(rozkmitu) výstupních hodnot a následným přičtením této hodnoty získané maskováním do DHR(Data-Hold) registru. Hodnota registru DHR určuje nejmenší hodnotu posílanou na výstupu DAC převodníku - tedy offset výsledného signálu. Součet maskované hodnoty LFSR a DHR registrů se pak každý posílá na výstup DAC převodníku (DOR registr). Celý postup je zobrazený na obrázku 3.12 Takto produkovaný šum má plochou spektrální distribuci a může být s určitou rezervou považován za bílý šum, s tím rozdílem, že narozdíl od pravého bílého šumu nemá Gaussovské(normální) rozdělení, ale rovnoměrné.



Obr. 3.12: Využití LFSR registru pro generování šumu

Výsledný generovaný šum je limitován maximální vzorkovací frekvencí DAC převodníku. V případě STM32G431 je to 1 Msps. Na obrázku 3.13. Co se týče nastavení parametrů generovaného signálu tak

offset se dá nastavit jako jakákoliv hodnota v celém rozsahu DAC převodníku, menší rozsah hodnot jde ale reálně nastavit pro rozkmit generovaného signálu. Ze strany počítačové aplikace se sice zdá, že ho lze nastavovat v desetinnách procenta nicméně tím že jediný možný způsob přepočtu LFSR registru je maskováním, tak lze velikost rozkmitu nastavit pouze ve 12 krocích.



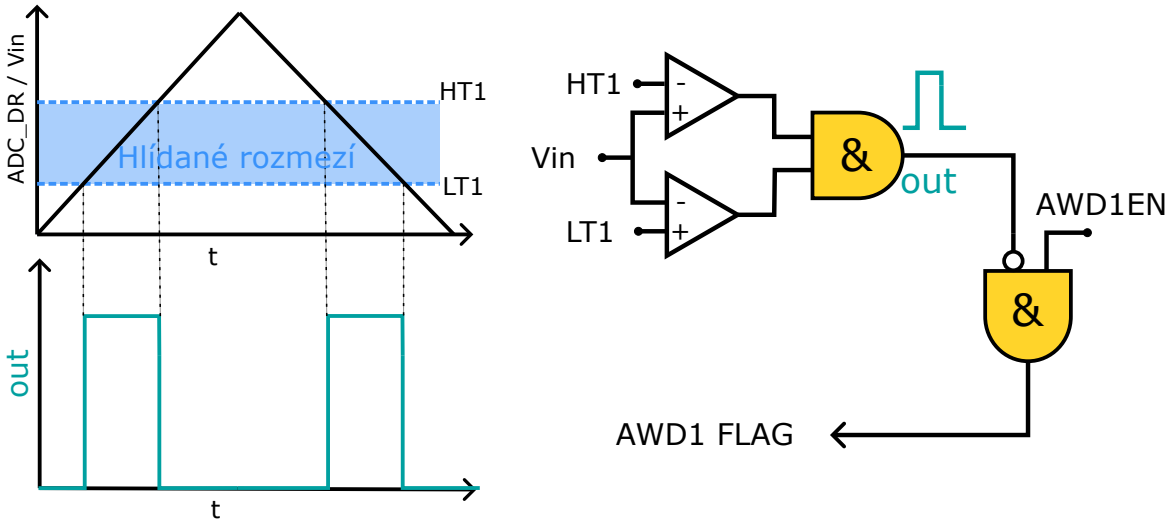
Obr. 3.13: Záznam generovaného signálu šumu- offset 20% a rozkmit 50%

3.6 Realizace osciloskopu

Modul osciloskopu je nejkomplexnější z modulů implementovaných v této práci a též využívající nejvíce HW prostředků. K implementaci na STM32G431 jsem využil přítomnosti 2 ADC převodníku, triggerování pomocí analog watchdog (AWDG) funkce přítomných ADC převodníků, DMA řadiče a 2 vzájemně propojených zřetězených čítačů.

3.6.1 Řešení triggerování osciloskopu

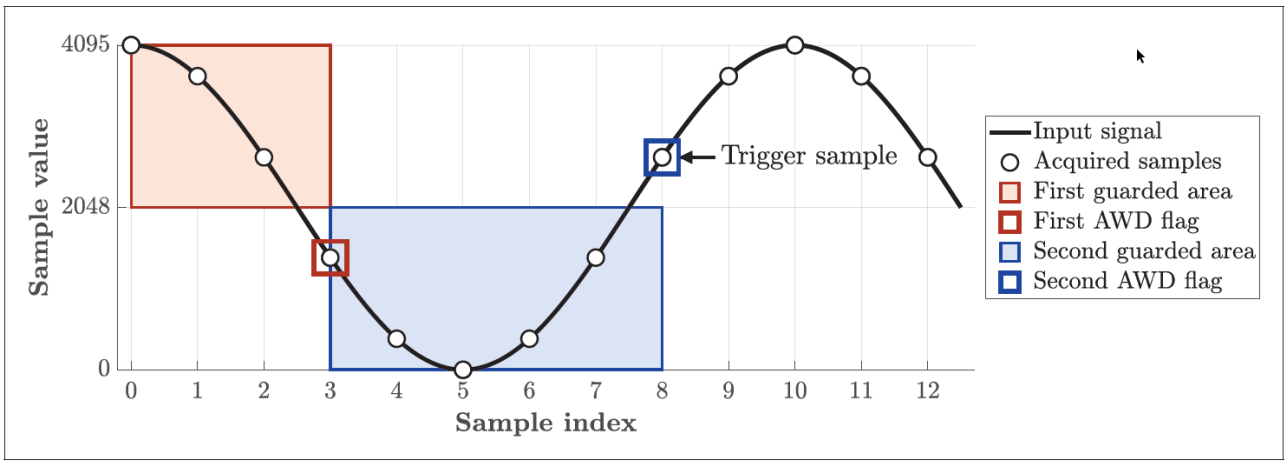
Určení okamžiku náběžné nebo sestupné hrany vstupního signálu je podstatnou funkcí osciloskopu pro zobrazování jak přechodných jevů tak periodických průběhů signálu. Existuje více přístupů, jak takovou funkci implementovat. Jednou z nejjednoduších variant je SW orientované řešení, kdy jsou naměřená data vždy cyklicky kontrolována, zda došlo k překročení zvolené napěťové úrovně a popřípadě zastavit další sběr dat. Takové řešení bylo například použito v původní variantě FW pro STM32F042 v práci [1]. Nevýhodou tohoto řešení je vyšší výpočetní náročnost ve srovnání s více HW zaměřenými řešeními, jaká jsou například využití komparátorů nebo funkce AWDG, kterou disponují ADC převodníky na mikrokontrolérech STM32G4 a kterou jsem se rozhodl uplatnit v této práci já.



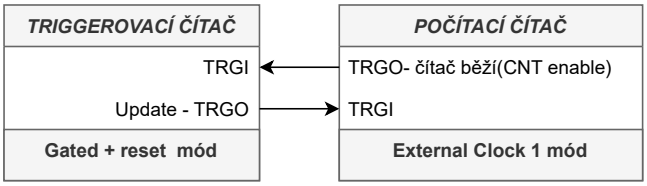
Obr. 3.14: Pre

AWDG umožňuje vyvolat přerušení v momentě, kdy se daném kanálu objeví napětí mimo nastavené rozmezí. Abychom mohli určit okamžik, kdy došlo k poklesu pod určitou úroveň(sestupná hrana) nebo naopak k překročení napěťové urovně(náběžná hrana) je potřeba využití AWDG ve dvou fázích znázorněných na obrázku 3.15.

- **Fáze 1**
  Při hledání okamžiku nástupní hrany v první fázi nejdříve nastavíme rozmezí nad střeženou napěťovou úrovní. Jakmile se napětí klesne mimo rozmezí dojde k přerušení a přesuneme se do fáze 2
- **Fáze 2**
  V této fázi víme že signál je pod nastavenou napěťovou úrovní. Tedy nastavíme nové střežené rozmezí napětí a víme, že jakmile dojde k přerušení, že nastal okamžik, který chceme označit jako moment nástupní hrany a uložit si číslo vzorku, kdy k tomuto došlo.



Obr. 3.15: 2-fázové triggerování na nástupnou hranu signálu s využitím AWDG- převzato z [2]



Obr. 3.16: Vzájemné propojení dvou čítačů

Po vyhodnocení, že došlo k události triggeru, potřebujeme zjistit číslo vzorku, kdy moment nastal a nastavit odměření zbývajících vzorků signálů, tak aby nové vzorky zapisované do kruhového bufferu nepřepsaly data vstupního signálu před touto událostí. Zde jsem využil dvou vzájemně propojených čítačů. Funkce prvního čítače je triggerování ADC převodníku a funkce druhého čítače je počítání odměřených vzorků a poté po triggeru zastavení triggerování po daném počtu vzorků.

- **Vzorkovací čítač**
  Vzorkovací čítač spouští vzorkování ADC převodníku. Ve spojitosti s osciloskopy se také používá termín generátor časové základny. V závislosti na propojení interních signálů daného mikrokontroléru můžeme pro spouštění ADC buď použít nastavitelný TRGO výstup čítače nebo jeden z jeho "Capture-Compare"kanalů. V případě využití TRGO výstupu se zvolí slouží UPDATE událost a kdy CNT registr dosáhne hodnoty ARR registru a čítač začíná počítat znovu od začátku. Hodnota prescaler a ARR registru tak určuje vzorkovací frekvenci. Vstupním hodinový signálem jsou interní hodiny o frekvenci systémových hodin. Navíc tento čítač operuje v slave modu "combine gated + reset", kdy tento čítač běží jen pokud na trigger vstupu(TRGI) je nastavena logická úroveň '1'. Jakmile dojde k poklesu na logickou úroveň(1) čítač se zastaví a zároveň se vyresetuje- tedy hodnota čítače CNT se vynuluje.
- **Čítač vzorků**
  Jak již název napovídá funkce tohoto čítače je určení pozice odebíraného vzorku v bufferu. Aktuální hodnota čítače odpovídá pozici v kruhovém bufferu následujícího snímaného vzorku. Tento čítač funguje ve funkci external clock 1 kdy čítač počítá náběžné hrany na vstupu TRGI. Na výstupu TRGO je potom stav čítače odpovídající enable bitu.

Vzájemné fungování by se pak dalo popsat následovně: pokud je zapnutý čítač vzorků, tak běží zároveň triggerovací čítač. Čítač vzorků čítá počet TRGO pulzů triggerovacího čítače a pokud čítač vzorků zastavíme, tak se zastaví triggerovací čítač a tím vzorkování ADC převodníkem. Pokud pak počítací čítač běží tzv 'one pulse' modu tak se oba čítače zastaví po odběru stanoveného počtu vzorků.

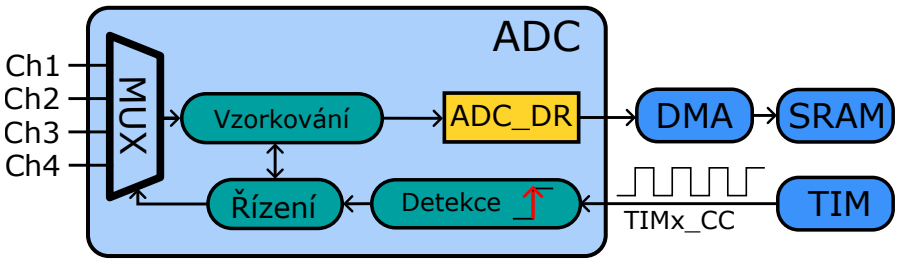
3.6.2 Módy vzorkování

Pokud máme k dispozici pouze jeden ADC převodník, musíme v případě měření signálu na více kanálech vstup ADC převodníku přepínat mezi jednotlivými kanály. V případě využití dvou a více ADC převodníků máme daleko větší variablitu jak vzorkovat daný set analogových kanálů. Podrobněji se jednotlivým módům věnuje práce [2]. STM32G431 má k dispozci 2 ADC, které lze využít pro zvýšení

Analogové kanály	CH1	CH2	CH3	CH4	Vnitřní reference
Přítomnost na Pinu	PA0	PA1	PA2	PA3	-
Dostupné ADC	ADC 1, ADC2	ADC1, ADC2	ADC1	ADC1	ADC1
Vnitřní kanál ADC	Kanál 1	Kanál 2	Kanál 3	Kanál 4	Kanál 18

Tab. 3.5: Přehled jednotlivých analogových kanálů a jejich na dostupnosti na ADC převodnících

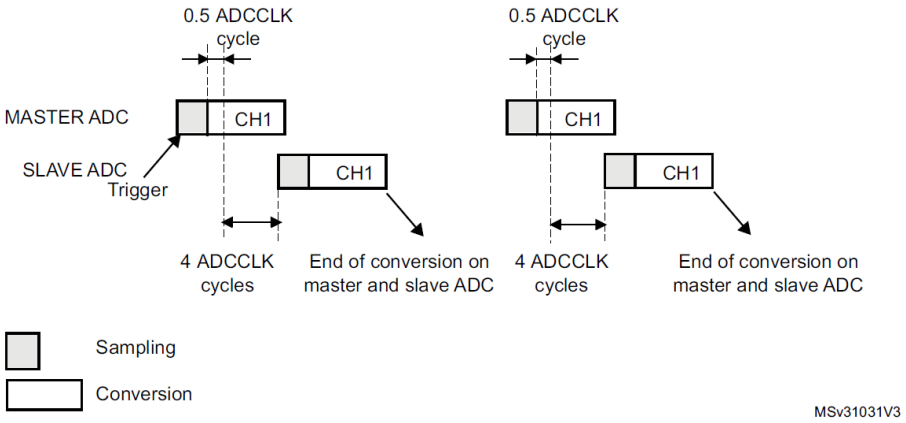
vzorkovací frekvence, respektive zvýšení doby vzorkování pro stejnou vzorkovací frekvenci oproti použití 1 ADC. Výběr použitého módu vzorkování pak záleží na zvolených kanálech, respektive na jejich počtu a na jejich přítomnosti na jednotlivých ADC převodnících. Například kanál osciloskopu číslo 1 je na pinu PA0 a na tomto pinu také může být připojen na vstup převodníků ADC 1(kanál 1) a ADC 2(kanál1), ale například kanál osciloskopu číslo 3 je na pinu PA2, který se sice dá připojit na vstup převodníku ADC1, ale už se nedá připojit na vstup převodníku ADC2



Obr. 3.17: Převzato z [6]

Vzorkování 1 kanálu

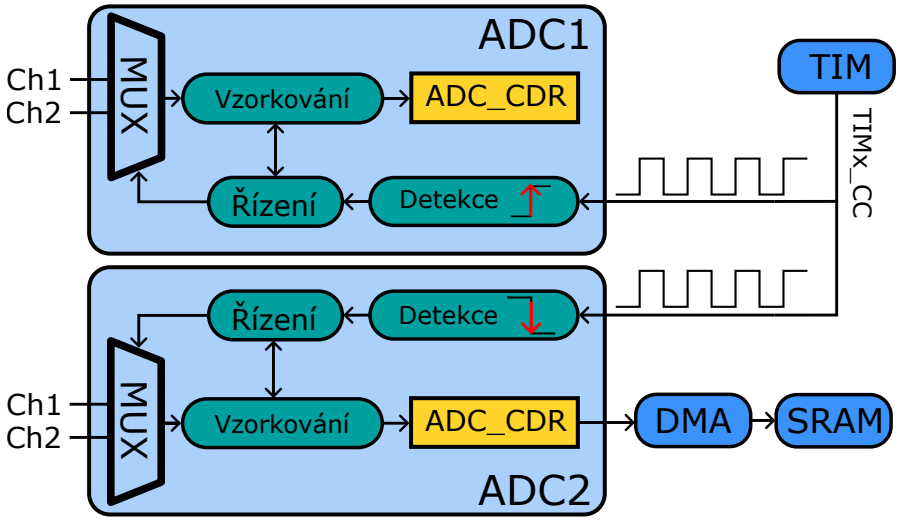
Pokud je zvolený kanál osciloskopu přiveditelný na vstupy obou ADC převodníků, můžeme využít tzv. "Dual -interleaved modu". Tento režim spočívá ve střídavém vzorkování jedním převodníkem a v průběhu konverze odebraného vzorku vzorkováním převodníkem druhým. U mikrokontroléru rodiny STM32G4 minimální doba odběru jednoho vzorku(vzorkování + konverze) trvá 15 period hodinového signálu ADC převodníku. V případě dual - interleaved modu lze pak vzorkovat zvolený kanál druhým ADC v momentě kdy na prvním ADC převodníku ještě probíhá konverze viz obrázek 3.18. Výsledně lze pak odebrat vzorek až každých 8 cyklů. Což dovoluje téměř zdvojnásobit maximální vzorkovací frekvenci. Druhou výhodou pak je možnost zvolení delší doby vzorkování při měření s nižší vzorkovací frekvencí více v kapitole 3.6.3.



Obr. 3.18: Převzato z [6]

Problémem běžně popisovaného dual interleaved režimu je že vzorkování podřazeného ADC2 je spuštěno pevně stanový počet cyklů po vzorkování ADC1, což může zkreslit průběh měřeného signálu, pokud časový interval mezi odběry vzorků není stejná. V mé implementaci tedy používám něco jako Independent interleaved mod. V tomto režimu využívám output compare kanálu čítače, abych spouštěl ADC převodníky střídavě tak, aby mezi po sobě jdoucími vzorky byla pokud možno stejná vzdálenost. Na výstupu čítače tedy je nastaven obdelníkový signál se střídou 50% , ADC1 začíná vzorkovat na hranu náběžnou a ADC2 na hranu sestupnou. Každý cyklus čítače tedy znamená odebrání 2 vzorků.

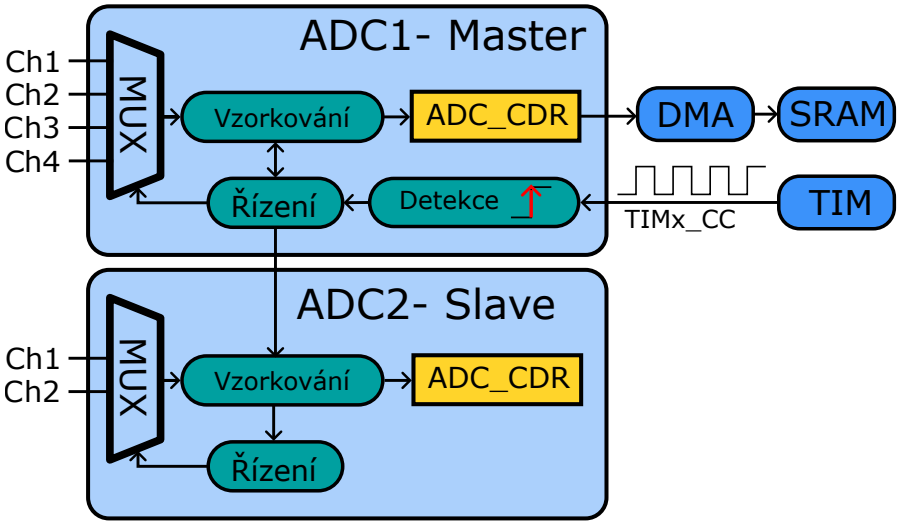
Oproti klasickému dual interleaved modu tento "independent interleaved"mód využívá nezávislého triggerování obou ADC převodníků. Nicméně stále je využito možnosti vyčítání naměřených dat ze sdílených data registů jako v případě obvyklého 'dual-interleaved' modu. Díky využití sdílených data registrů lze snížit vytížení DMA a AHB sběrnice a vyčítat data z obou ADC převodníků najednou a to s použitím 1 DMA kanálu přenášející 32 bitů . Přenos je pak spouštěn událostí dokončení konverze na ADC2.



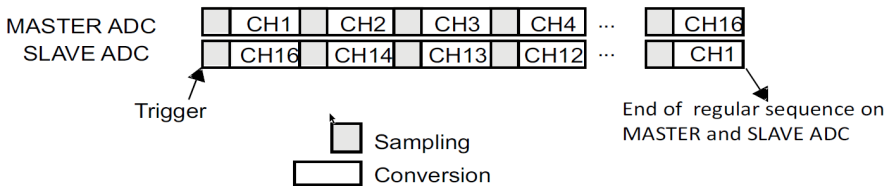
Obr. 3.19: Převzato z [6]

Vzorkování 2 a více kanálů

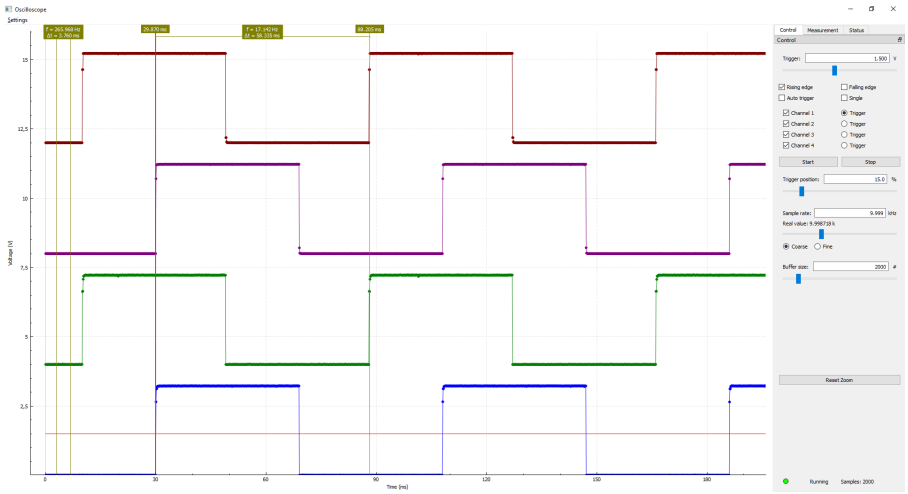
Pokud je to možné v případě zvolení 2 a více kanálů, je výhodné použít Dual Simultaneus režim, který nám umožní vzorkování více kanálů najednou. V mém FW jsem implementoval automatické přiřazování kanálů jednotlivým převodníků, podle jejich dostupnosti. Přiřazování probíhá tak že nejdříve se přiřadí kanály, které jsou dostupné pouze na jednom z převodníků a pak se posloupnosti kanálů doplní kanály dostupnými na obou převodnících tak, aby pokud možno oba ADC převodníky snímali stejný počet kanálů. Vznikají tak různé konfigurace na základě zvolených kanálů. Při analogových kanálech, tak jak jsou popsány v tabulce 3.5 tak je při zvolení jakýkoliv 2 kanálů CH1-CH4 zajištěno snímání ve stejný čas, kromě kombinace CH3+Ch4, které jsou oba dostupné pouze na ADC1 a tedy musí být vzorkovány multiplexováním vstupu ADC převodníku. To má za následek v případě této kombinace sníženou maximální vzorkovací frekvenci v porovnání s jinými 2 kanálovými konfiguracemi. V případě zvolení 3 nebo 4 kanálů se pak ukázalo jako nejvhodnější řešení pevné nastavení skenování všech 4 kanálů, kdy jsou najednou snímany kanály CH1+CH3 a CH2+CH4. Po navzorkování bufferu je pak případně nezvolených kanál vyfiltrován z bufferu před posláním naměřených dat do počítače.



Obr. 3.20: Převzato z [6]



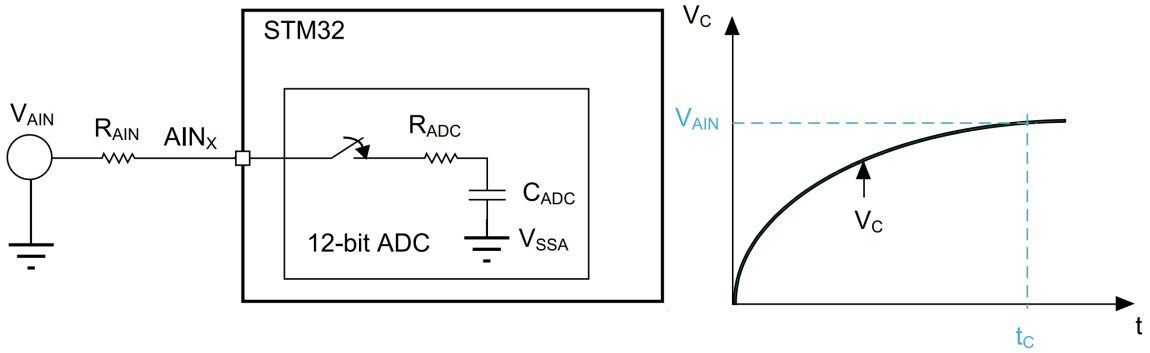
Obr. 3.21: Dual Regular simultaneus mod převzato z [6]



Obr. 3.22: Zpoždění kanálu 2 a 4 za kanálem 1 a 3 při použití Dual simultaneus modu a stroboskopickém vzorkování s ekvivalentní vzokovací frekvencí 78MHz

3.6.3 Nastavení délky vzorkování a vliv na maximální vstupní odpor

Impedance analogového zdroje signálu, respektive sériový odpor(R_{AIN}) mezi zdrojem a pinem ovlivňují proud nabíjející vzorkovací kondenzátor. Časová konstanta nabíjení t_c potom určuje minimální dobu vzorkování, tak aby se vzorkovací kondenzátor měl čas nabít nabít na vstupní napětí(s tolerancí 1/2 LSB) V_{AIN} jako je znázorněno na obrázku 3.23. Čas vzorkování je limitován vzorkovací frekvencí a tedy pro různé vzorkovací frekvence existuje nějaký maximální vstupní odpor (R_{AIN}) takový, aby se měření neodchylovalo o více než ±0.5LSB.

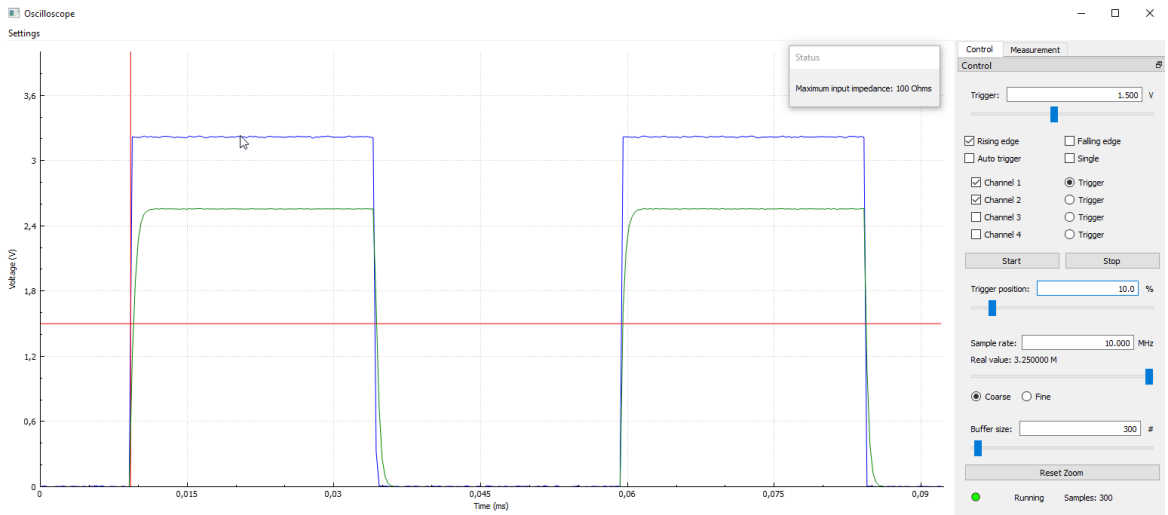


Obr. 3.23: převzato z [9]

Doba vzorkování se nastavuje v počtu period hodinového signálu ADC (ADC_CLK) převodníku s tím že na výběr je typicky pro daný ADC převodník sada dostupných hodnot s tím, že dále lze pak ovlivnit dobu vzorkování změnou frekvence již zmíněného ADC_CLK. Jako nejlepší řešení se jevilo zvolit pro danou vzorkovací frekvenci a zvolený počet kanálů maximální možnou dobu vzorkování takovou, aby se stihly všechny zrovna zvolené kanály navzorkovat a zkonvertovat před tím, než přijde z

triggerovacího čítače pokyn k dalšímu odběru vzorků. Mikrokontroler poté informuje uživatel skrz PC aplikaci o maximální impedanci zdroje signálu, aby měření nebylo zkresleno.

$$t_c = (R_{\mathrm{ADC}} + R_{\mathrm{AIN}}) \times C_{\mathrm{ADC}} \tag{3.6}$$



Obr. 3.24: Efekt nedodržení maximálního vstupního odporu

3.7 Logický analyzátor

Kapitola 4

Ověření funkčnosti

Kapitola 5

Zhodnocení

Kapitola 6

Závěr

Příloha A

Literatura

[1] Berlinger, A. Implementace přístrojových funkcí s využitím mikrořadičů stm32, 2016. [cit. 2023-05-23].

[2] Dujava, J. Softvérovo definované osciloskopy s terminálovým rozhraním, 2023. [cit. 2023-03-15].

[3] Fischer, J. G0-lab s stm32g031. [online], 7 2017. [cit. 2020-05-09].

[4] Hladik, J. Single chip software defined instrumentation for educational purposes. [online], 05 2017. [cit. 2023-05-08].

[5] RIGOL TECHNOLOGIES, I. Datasheetds1000e, ds1000d series digital oscilloscopes. [online], 12 2015. [cit. 2020-04-13].

[6] STMicroelectronics. Reference manual RM0440 stm32g4 series advanced arm®-based 32-bit mcus. [online], 1 2017. [cit. 2023-04-30].

[7] STMicroelectronics. An5325 how to use the cordic to perform mathematical functions on stm32 mcus. [online], 02 2021. [cit. 2023-04-29].

[8] STMicroelectronics. *Datasheet DS12589 STM32G431x6 STM32G431x8 STM32G431xB*. Rev 6, 10 2021. [cit. 2023-05-06].

[9] STMicroelectronics. Application note AN2834 how to get the best adc accuracy in stm32 microcontrollers. [online], 01 2022. [cit. 2023-05-02].

[10] STMicroelectronics. How to use the cordic to perform mathematical functions on stm32 mcus. [online], 02 2023. [cit. 2023-05-05].

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **David** Jméno: **Petr** Osobní číslo: **420110**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Kybernetika a robotika**

Studijní obor: **Kybernetika a robotika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Firmware pro měřicí přístroj s mikrořadičem STM32G431

Název diplomové práce anglicky:

Firmware for measuring instrument based on microcontroller STM32G431

Pokyny pro vypracování:

V návaznosti na přístroj vyvinutý v rámci DP [1] vytvořte firmware pro mikrořadič STM32G431 tak, aby jej ve spolupráci s PC aplikací Zero eLab Viewer bylo možno využít jako jednoduchý, avšak komplexní měřicí přístroj pro výukové účely. V případě potřeby proveďte nutné úpravy PC aplikace. Přístroj bude zahrnovat funkce osciloskopu i se zobrazením průběhů logických kanálů, dále funkce impulsního a signálového generátoru, čítače a voltmetru se záznamem. Při návrhu firmware můžete též využít vhodné bloky vytvořené v rámci prací [2] a [3]. Výsledný přístroj otestujte a ověřte jeho parametry.

Seznam doporučené literatury:

[1] Berlinger, A.: „Implementace přístrojových funkcí mikrořadiči STM32“, diplomová práce ČVUT – FEL, 2016
[2] Cejp M.: „Virtuální přístroj s mikrořadičem pro analýzu signálu v modulační doméně“, bakalářská práce, ČVUT – FEL, 2017
[3] Dujava J., „Softwarově definované osciloskopy s terminálovým rozhraním“, diplomová práce ČVUT – FEL, 2022
[4] Cejp M.: „Virtuální přístroj s mikrořadičem pro analýzu signálu v modulační doméně“, bakalářská práce, ČVUT – FEL, 2017

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Jan Fischer, CSc. katedra měření FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **06.09.2022** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce:
do konce letního semestru 2023/2024

doc. Ing. Jan Fischer, CSc. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____ Datum převzetí zadání

_____ Podpis studenta