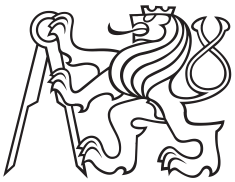
Diplomová práce



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

© ČVUT v Praze, Design: ČVUT v Praze, VIC

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

V rámci výuky na katedře měření v předmětech zaměřující se na elektroniku se již nějaký čas používají softwarově definované přístroje(SDI), které slouží jako dostupná náhrada za laboratorní vybavení jako jsou osciloskopy nebo signálové generátory. Ač tyto přístroje nedosahují parametrů profesionálních přístrojů, tak existuje mnoho aplikací, kde jejich výkon ve smyslu maximální vzorkovací frekvence či přesnosti určení napětí je dostatečné a přístroje tak mohou v těchto případech profesinální přístroje nahradit.

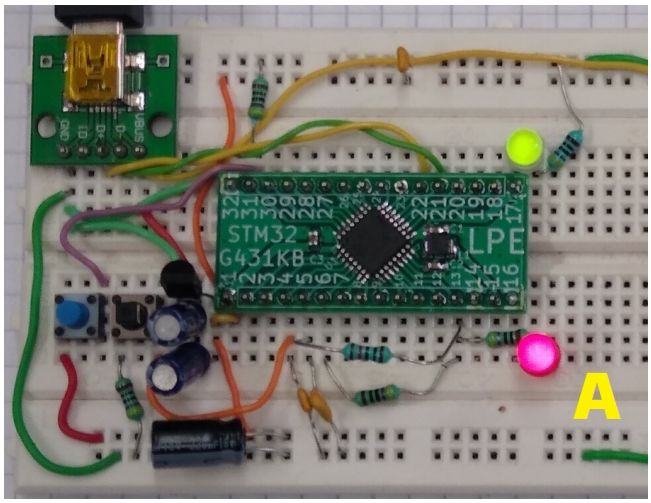
Konkrétně v předmětu laboratoře průmyslové elektroniky(LPE) se právě používá SDI běžící na MCU STM32F042 v kombinaci s aplikací Zero Elab Viewer pro různé experimenty jako je měření časové konstanty RC článku, zapojování různých obvodů s operačními zesilovači a mnoho dalších. SDI se v rámci výuky také používá pro diagnostiku správnosti funkce programů, které studenti sami programují na výukových modulech s nepájivými poli.

Nově se studenti učí programovat MCU řady STM32G431, které oproti původně používanému MCU STM32F042 nabízí vyšší výpočetní výkon a také nové periferie jako je například DAC převodník. Vhodnost využití tohoto MCU jakožto SDI již ukázala dříve vzniklá implementace SDI na tomto MCU spolupracující s aplikací VSVI, která také vznikla zde na fakultě. Ovládání této aplikace se ale prokázalo méně intuitivní než je tomu v případě Zero Elab Viewer a zároveň studenti stále v těchto využívají hotový modul s STM32F042 a tak vznikla myšlenka využít STM32G431 k implementace SDI spolupracují Zero Elab Viewer a přidat rozšířit funkcionalitu pro studenty a vyučující známé aplikaci.

Kapitola 2

Rozbor

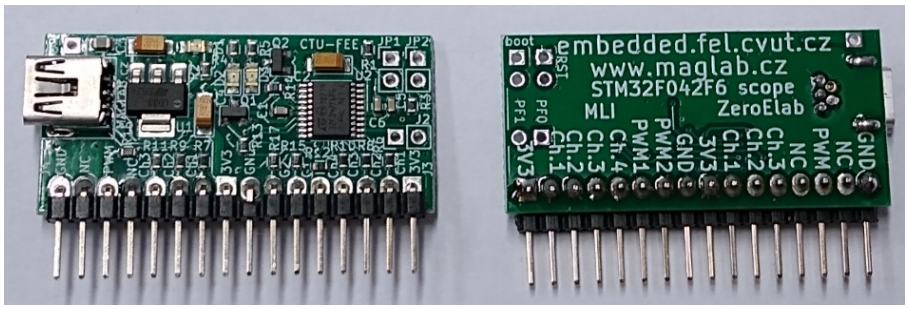
Jak již bylo výše zmíněno v laboratoři se dlouhodobě používá SDI vzniklý okolo STM32F042 společně s aplikací Zero Elab Viewer. V rámci výuky programování mikrokontrolérů se přešlo na nový modul(obr. 2.1) s výkonnějším MCU STM32G431, který obsahuje jádro ARM Cortex-M4 a vyšší počet periferií než doposud používaná STM32F042. Otevřely se tím možnosti pro implementace SDI s lepšími parametry, jako je například modul vzniklý v rámci práce [4] spolupracující s aplikací VSVI. Ten oproti dosud používanému řešení nabízel nově funkční generátor, komplexní pulzní generátor a také výrazně vyšší vzorkovací frekvence osciloskopu dosahující až 6.5 MSps.



Obr. 2.1: Modul s STM32G431 na nepájivém poli používaný v rámci výuky LPE, převzato z [1]

Nicméně jiné ovládání (TUI) oproti klasickému grafickému rozhraní a také vyšší komplexnost tohoto řešení způsobili horší uživatelskou zkušenost a vznikla poptávka po vytvoření FW pro STM32G431, který by spolupracoval s původní aplikací Zero Elab Viewer a zároveň přinášel zlepšení parametrů jako maximální vzorkovací frekvence a rozšíření funkcionalit. Zároveň studenti mají k dispozici modul (obr. 2.2) pro méně náročná měření, a tedy v případě přecházení od jedné implementace SDI k druhé by pak

odpadala nutnost se orientovat v novém prostředí.

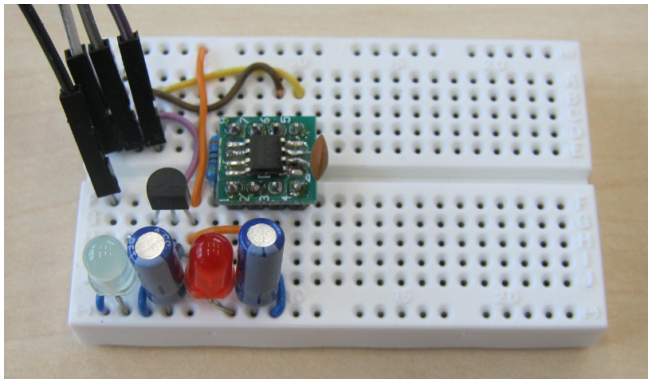


Obr. 2.2: Modul SDI s STM32F042 s předchystaným FW pro Zero Elab Viewer

Úkolem této práce tedy je vytvořit 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. Nejdříve bylo nutno tedy analyzovat protokol existující aplikace a vytvořit firmware tak, aby využil všech funkcí, které aplikace již nabízí. Dále by mělo dojít ke zlepšení parametrů oproti původní verzi SDI pro STM32F042 a to především dosáhnout vyšších vzorkovacích frekvencí osciloskopu a logického analyzátoru, využít větších paměťových kapacit, které STM32G431 nabízí pro delší záznamovou paměť a v neposlední řadě také implementovat nové funkce jako je generátor funkcí. Přístroj by měl dále zahrnovat funkce impulsního generátoru, čítače a voltmetru se záznamem.

2.1 PC aplikace Zero Elab Viewer

V roce 2016 v rámci své diplomové práce [2] pan Ing.Adam Berlinger vytvořil aplikaci Zero Elab Viewer fungující jako grafické rozhraní pro komunikaci se softwarově definovaným 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 ve výuce praktické elektroniky a od té doby vznikla ještě řada implementací FW pro jiné MCU. Nejnověji například verze pro mikrokontrolér ATMega 328 pro uživatele se zkušenostmi s ARDUINO nebo zatím nejmenší (a nejlevnější) varianta SDI přístoje vzniklá využívající 8-pinovou variantu mikrokontroleru STM32G030.



Obr. 2.3: STM32G031 zapojen na nepájivém poli převzato z [5]

2.1.1 Existující SDI FW pro spolupráci s aplikací Zero Elab Viewer

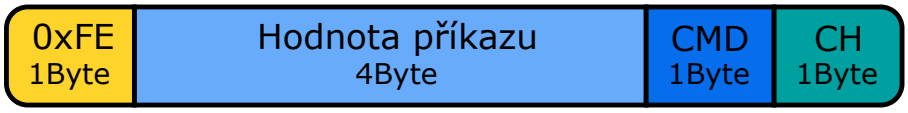
- Firmware pro SDI s STM32F042
  - Osciloskop- záznam až 1152 vzorků, rozlišení 12 bitů,
  - Rychlost záznamu až 1x 600 kS/s
  - Ve stroboskopickém módu - až 48 MS/s - rozlišení s intervalem 15,6 ns.
  - Impulsní generátor PWM , nastavení střídy 0 až 100 %; nastavení frekvence 1 Hz až 24 MHz
  - Voltmetr tři kanály, 0 až + 3,3 V, 100 odměrů/s, průměrování z 1 až 256 odměrů, možnost funkce záznamu
- Firmware pro SDI STM32G030
  - 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
- Firmware pro SDI s ATMega 328
  - Dvoukanálový Osciloskop- záznam až 768 vzorků, rozlišení 8 bitů,
  - Rychlost záznamu až 1x 80 kS/s (nebo 2x 50 kS/s, 3 x 666 kS/s).
  - Ve stroboskopickém módu - až 4 MS/s - rozlišení s intervalem 250 ns.
  - Impulsní generátor PWM , nastavení střídy 0 až 100 %; nastavení frekvence 0.5 Hz až 4 MHz
  - Voltmetr tři kanály, 0 až +5 V, rozlišení 10 bitů, 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 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).

2.3 Komunikační protokol aplikace ZeroElab Viewer

Komunikační protokol aplikace Zero Elab Viewer je postaven kolem 2 druhů zpráv. Druhy zpráv jsou od sebe navzájem odlišeny prvním bytem zprávy. Zaprvé jsou to zprávy typu příkazů, (CMD), které slouží k oboustrannému předávání pokynů. Jedná se o jednoduché zprávy, ve kterých pole CH určuje, kterému modulu je příkaz určen, tedy jestli se jedná o příkaz voltmetru, pulzního generátoru nebo například obecný příkaz měnící konfiguraci přístroje. Dále pole CMD rozlišuje, o jaký druh příkazu se jedná. Typicky příkaz společný pro jednotlivé moduly je zapnutí/vypnutí běhu daného modulu a hodnota příkazu pak rozšiřuje možnost předaných parametrů spojených s daným příkazem. Druhým typem zpráv jsou je data, která míří z MCU do aplikace v PC specifikovaná číslem kanálu(modulu), z kterého data pocházejí, délkou data po které pak už následují samotná data. Oba formáty zpráv jsou naznačeny na obrázcích 2.4 a 2.5.



Obr. 2.4: Rozložení bytů ve zprávě typu příkaz(CMD)

Obr. 2.5: Rozložení bytů ve zprávě typu data

Kapitola 3

Realizace FW

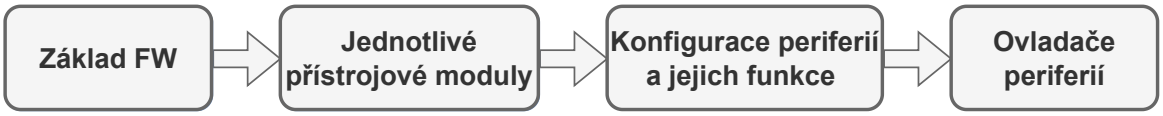
3.1 Struktura FW

Jak vyplývá ze zadání, firmware vznikal v návaznosti na existující přístroj využívající stávající PC aplikací Zero eLab Viewer a již touto skutečností byl návrh firmware mikrokontroléru částečně vymezen. Přestože určité změny PC aplikace by byli přípustné, tak bylo podstatné aby přes jakékoliv provedené změny aplikace zůstala použitelná i pro všechny dříve vzniklé implementace SDI pro jiné mikrokontroléry. Především pak komunikační protokol, který se v některých případech prokázal limitující nemohl být upraven. Dále není jednoduché přidávat funkcionality se kterými původně aplikace nepočítala a tedy aplikace rovnou určuje jaké softwarově definované přístoje lze implementovat a jaké budou mít možnosti ovládání či nastavení. Při návrhu Fw jsem se pak 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 možnost použití částí kódu jako příklady pro návrh nově vznikajících SDI přístrojů .

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

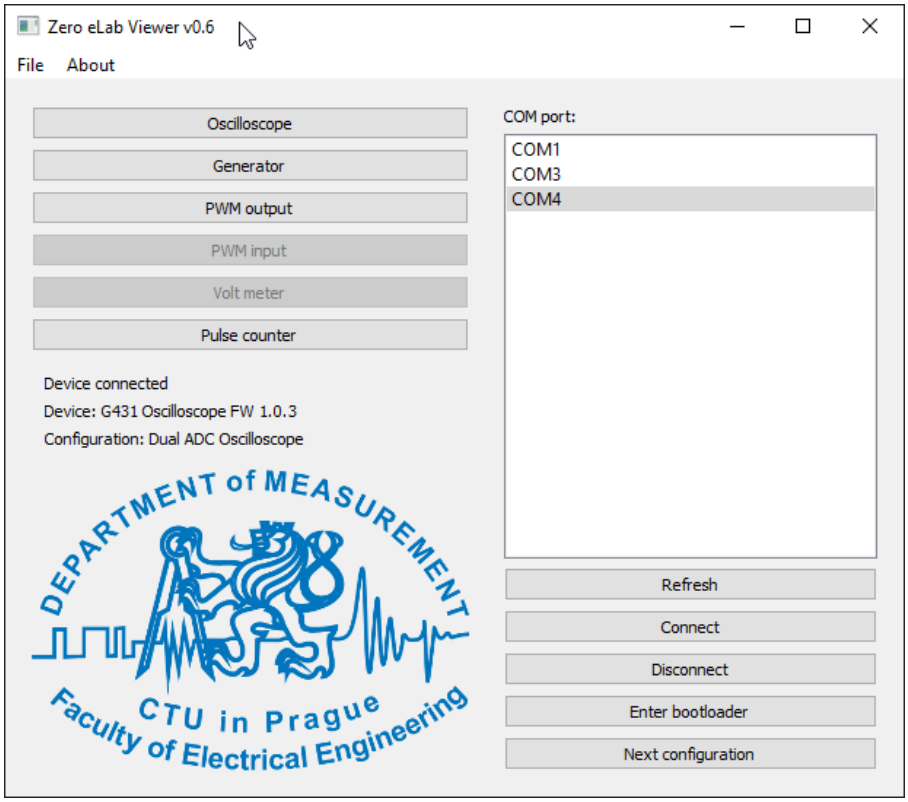
- **Kompaktibilita s generátorem inicializačního kódu**
  Nástroj STM32CubeMX velmi usnadňuje začátek vytváření FW díky využití grafického rozhraní pro definování počáteční konfigurace MCU. Pokud se zachová struktura generovaného kódu, lze pak nástroj opětovně využít v případě, že chceme jednoduše upravit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



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

3.1.2 Využití LL ovladačů

Firma STMicroelectronics pro podporu svého portfólia mikrokontrolérů nabízí tzv. LL ovladače usnadňující psaní firmware. Jedná se o hardwarově orientovanou knihovnu jejíž využití je velmi podobné

využití CMSIS ovladačů s tím rozdílem, že nabízí určité rozšířené možnosti portovatelnosti mezi jednotlivými rodinami mikrokontrolérů a sadu rozšiřujících API pro zjednodušení implementace některých úkonů, jako je například inicializace periferií. K použití této knihovny je oproti pravděpodobně známější knihovně HAL zapotřebí znalost jednotlivých periferií, jelikož velká část definovaných funkcí je pouze jednořádková modifikace registrů bez kontrol vstupních parametrů a uživatel tedy musí být více obeznámen s tím co dělá. Výhodou LL oproti HAL je výrazně menší délka generovaného kódu a tedy nižší paměťová náročnost.

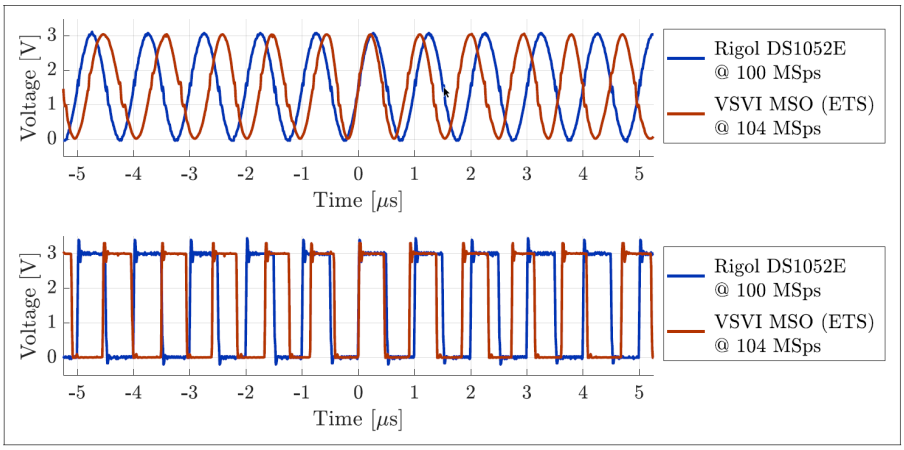
Pro vytvoření představy o rozdílnosti paměťové náročnosti jednotlivých knihoven jsem vytvořil pomocí STM32CubeMX dva minimální projekty, jejichž úkolem bylo pouze inicializace systémových hodin a dále blikání jednou LED. V prvním případě byly periferie RCC(Reset and clock control) a GPIO inicializovány prostřednictvím HAL knihovny a v druhém případě pomocí LL ovladačů. Dále byly použity odpovídají zpožďovací funkce společně s přepínáním výstupu pinu. Výsledkem bylo, že po zkompilování verze s HAL knihovnou využívala asi o 46 procent více FLASH paměti.

	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

3.2 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 3.3. 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 $\pm 0.005\%$ [7]. 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 [4]

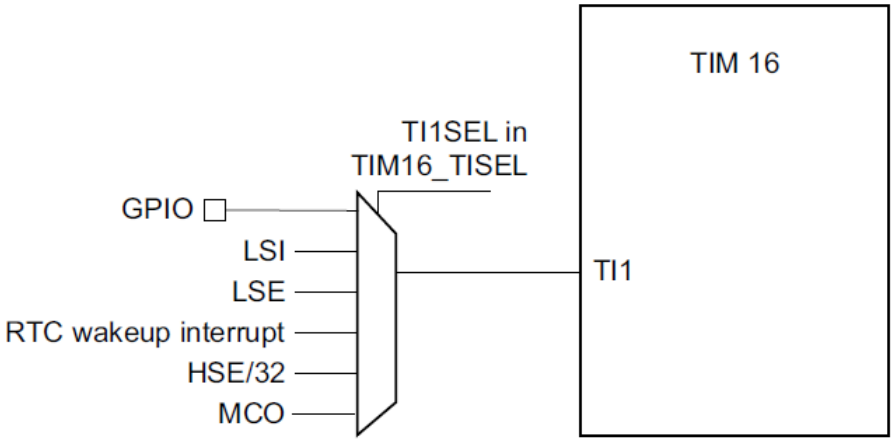
Jako zdroj vysokorychlostního externě získaného hodinové signálu (HSE) lze buď využít krystalu buzeného pomocí MCU (HSE crystal) nebo jiného externího zdroje signálu (HSE bypass) splňující podmínky na maximální frekvenci a střídu hodinového signálu pro dané MCU. Například pro STM32G431 musí být externí signál 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.

	Absolutní odchylka frekvence	Odchylka frekvence způsobená změnou teploty
STM32G431 HSI 16 MHz	± 1%	±1% (0 až +85°C)
Adafruit krystal 16 MHz	±0.003%	±0.005% (-20 až +70°C)

Tab. 3.2:

3.2.1 Měření frekvence HSE

Abychom mohli správně nastavit parametry PLL podle použitého krystalu, potřebujeme znát jeho frekvenci. Variantou by mohlo být ponechat na uživateli, aby zadal použitou frekvenci krystalu například nějakou zprávou přijatou z aplikace v počítači. Nicméně pro to by se musela upravit aplikace. Jako optimálnější řešení se zdálo frekvenci dostupného krystalu změřit. Na rodině mikrokontrolérů STM32G4 mají čítače TIM16 a TIM17 možnost interně přivést HSE 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 HSE/32 jako je vidět na obrázku 3.4 z dokumentace.



Obr. 3.4: Vstupy dostupné na kanálu číslo 1 čítače TIM16. Převzato z[9]

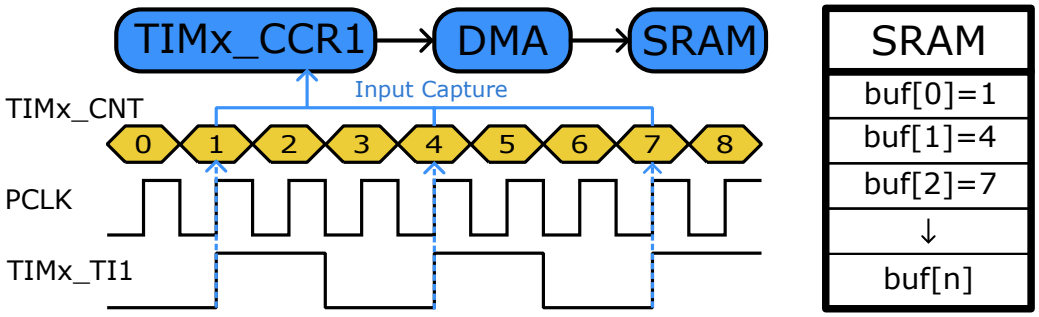
Standardní způsob měření periody vstupního signálu probíhá potom tak, že měříme počet uběhlých cyklů čítače mezi jednotlivými náběžnými hranami nebo sestupnými hranami jako je vyobrazeno na obrázku 3.5. Tento počet cyklů nám pak určuje poměr mezi frekvencí externího hodinového signálu na vstupu f_{IN} a hodinového signálu, který pro svůj chod využívá periferie čítače f_{TIM}. Druhou zmíněnou frekvenci(f_{TIM}) typicky známe. Pro správné měření je také podstatné, aby frekvence interního hodinového signálu čítače byla výrazně vyšší než frekvence na měřeném vstupu. Pro získání přesnějšího odhadu vstupní frekvence můžeme zaznamenat více period po sobě a ty zprůměrovat. Na obrázku 3.5 je popsán případ, kdy ze záznamu paměti je vidět, že jednotlivé odebrané vzorky jsou od sebe vzdáleny tři periody interních hodin PCLK. Frekvence vstupního signálu f_{IN} je tedy třetinová oproti frekvenci f_{TIM}

$$f_{\mathrm{IN}} = \frac{f_{\mathrm{TIM}}}{N_{\mathrm{period}}} = \frac{f_{\mathrm{TIM}}}{\mathrm{buf[n]} - \mathrm{buf[n\text{-}1]}} = \frac{f_{\mathrm{TIM}}}{3} \tag{3.1}$$

V případě měření frekvence HSE o frekvenci $f_{\mathrm{HSE\_8}} = 8\mathrm{MHz}$ přivedený na zmíněný vstup čítače s vestavěnou předděličkou HSE/32 a frekvencí vnitřních hodin $f_{\mathrm{TIM}} = 156$ MHz. Tak můžeme očekávat následující počet period mezi jednotlivými náběžnými hranami:

$$N_{\mathrm{period}} = \frac{f_{\mathrm{TIM}}}{f_{\mathrm{IN}}}, \quad f_{\mathrm{IN}} = \frac{f_{\mathrm{HSE}}}{32} \implies N_{\mathrm{HSE\_8}} = \frac{32 f_{\mathrm{TIM}}}{f_{\mathrm{HSE\_8}}} = 624 \tag{3.2}$$

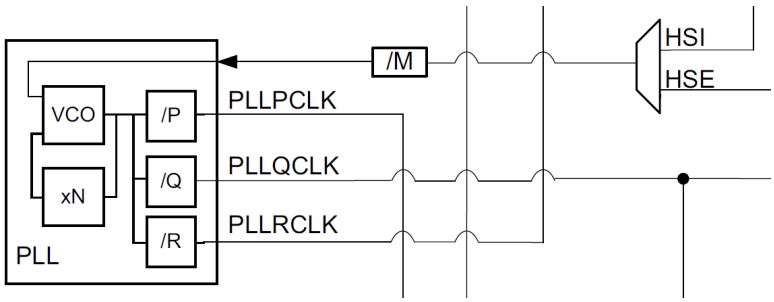
Obdobně pro krystal s frekvencí $f_{\mathrm{HSE\_16}} = 16$ MHz bychom změřili, že mezi jednotlivými náběžnými hranami vstupního signálu uběhne $N_{\mathrm{HSE\_16}} = 312$ period vnitřních hodin čítače. Tedy vysoká frekvence vnitřních hodin čítače $f_{\mathrm{TIM}} = 156$ MHz a možnost dělení signálu HSE před samotným vstupem do periferie čítače poskytuje dostatečnou rozlišovací schopnost mezi jednotlivými krystaly, abychom je od sebe dokázali rozlišit.



Obr. 3.5: Měření periody vstupního signálu pomocí "Input-capture"a DMA

3.2.2 Využití PLL

V systému je zapotřebí několika zdrojů hodinových signálů pro které se výhodně použije obvod PLL. Může být použit jako zdroj hodinového signálu pro periferii USB, ADC převodníků a v především slouží jako zdroj systémových hodin v případě, že chceme dosáhnout vyšších frekvencí systémových hodin než získáme z HSE nebo HSI. PLL umožňuje dělení a násobení vstupní frekvence. Jako vstup pak lze použít HSI nebo HSE ve stanovené rozsahu. Například 2.66-16MHz pro stm32G431[11]. 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.

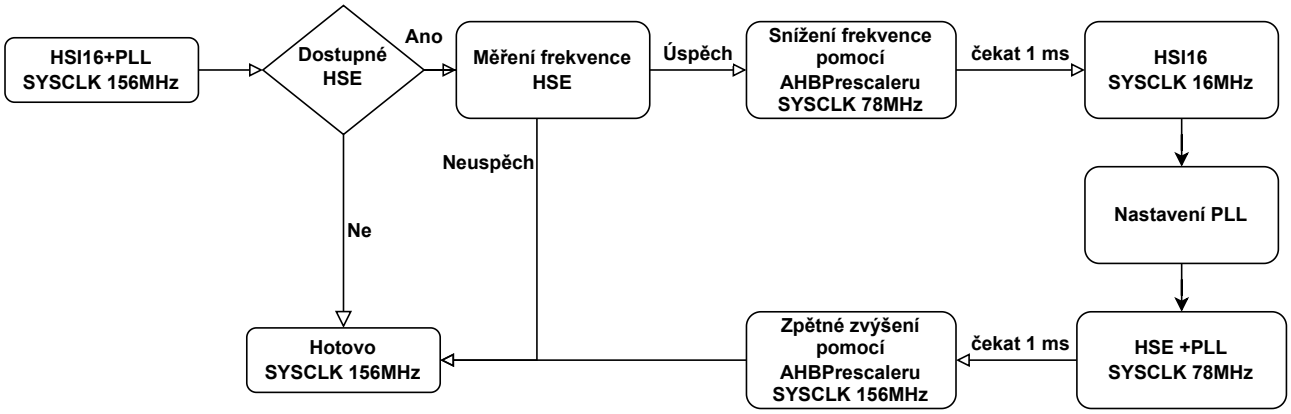


Obr. 3.6: Vyobrazení interního PLL bloku. Převzato z

Díky této možnosti více výstupů z obvodu PLL, které mohou mít nastavené různé děličky, lze například nastavit vstupní hodiny ADC převodníku nezávisle na systémových hodinách. Toho lze vhodně využít pro optimalizaci provozních frekvencí. Například maximální frekvence vstupních hodin ADC při využití více než 1 převodníku je 52 MHz. Jelikož při odvozování frekvence ADC ze systémových hodin máme omezené možnosti předděličky, tak maximální frekvence systémových hodin by pak mohla být pouze 104 MHz. Nicméně s využitím PLL můžeme nastavit vstupní hodiny ADC převodníku na požadovaných 52 MHz a zároveň nastavit frekvenci systémových hodin na 156MHz.

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 [9] 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 generátoru funkcí

Součástí zadání byla implementace signálového generátoru. V práci je využit vnitřní převodník DAC pro generování jednokanálového analogového signálu s různými průběhy. Většina přístrojů do této doby implementovaných v rámci katedry měření zatím umožňovala generování pouze obdélníkové signálu s využitím PWM generátorů, které neumožňovaly některé druhy měření analogových obvodů. Zejména chyběla možnost generace sinusového signálu, který se v elektronice často využívá - například pro stanovení amplitudové a fázové frekvenční charakteristiky RC článku. Dále generátor umožňuje generování nastavitelného stejnosměrného napětím, což dále rozšiřuje repertoár laboratorních úloh.

3.3.1 Použitý DAC převodník a jeho limity

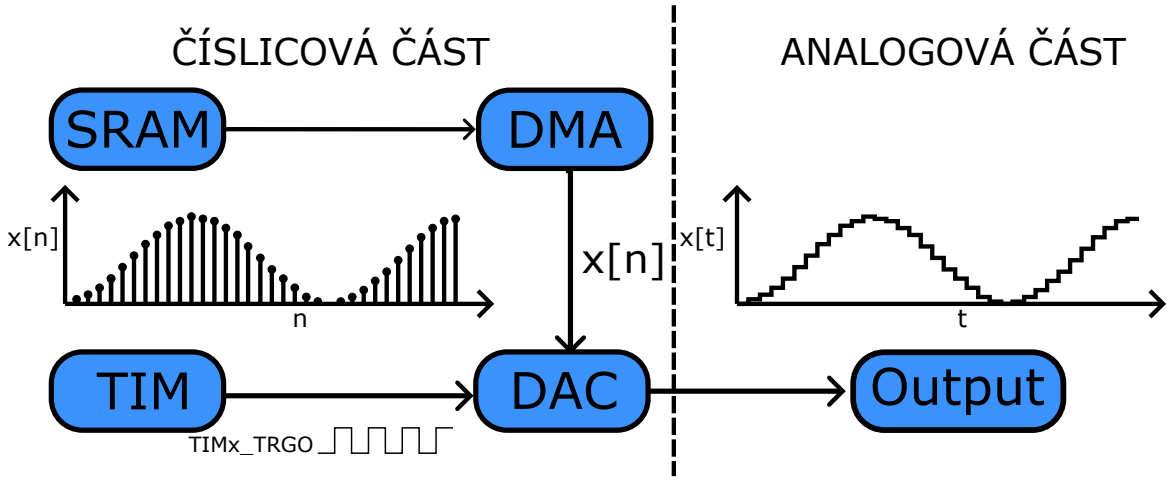
Použitý interní DAC převodník nabízí dle [11] 2 výstupní kanály s maximální vzorkovací frekvenci 1 Msps a rozlišením 12 bitů. Převodník také s pouhým využitím vnitřních obvodů - tedy nezávisle na programu umožňuje generování některých signálových průběhů jako jsou trojúhelníkový signál, signál

tipu "pila"či šumový signál. DAC převodník má také na svém výstupu integrovaný výstupní zesilovač, který slouží pro snížení výstupní impedance.

3.3.2 Generování signálu

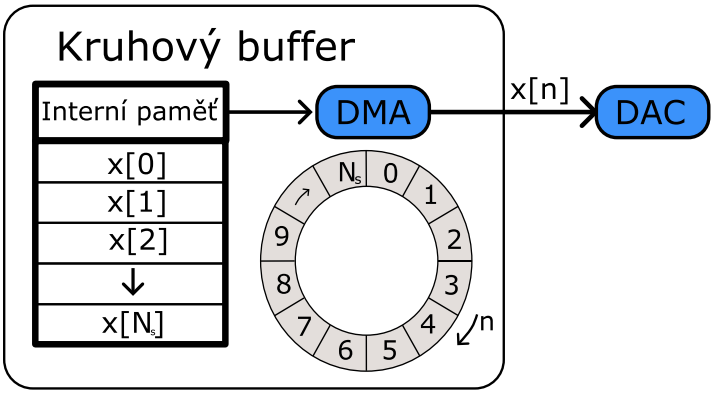
V předchozí části jsem zmiňoval možnost generování signálových průběhů, jako jsou pilový nebo trojúhelníkový signál, pouze nastavením interních registrů DAC převodníku. Tato metoda spočívá v automatickém výpočtu 12 bitových hodnot pro výstup DAC převodníku prostřednictvím HW registrů. Typicky se k nastavené hodnotě(offsetu) pravidelně přičítá hodnota jiného registru s tím, že si DAC převodník sám řídí generování nových hodnot vzorků. Výhodou tohoto přístupu jsou minimální paměťové nároky, protože číslicové hodnoty jednotlivých vzorků jsou vypočtené za běhu a nejsou uloženy někde v paměti. Tohoto způsobu generování nových číslicových hodnot pro DAC převodník bylo v této práci využito pro generování pseudošumu, které popisuji v kapitole 3.3.3

Nevýhodou využití interních registrů pro generaci signálových průběhů jsou především omezené možnosti ohledně změny tvaru signálu, kdy nám například chybí možnost generování sinusového signálu. Alternativou je pak uložit číslicové hodnoty v rozlišení DAC převodníku pro jednotlivé vzorky do paměti mikrokontroléru a následně prostřednictvím DMA řadiče jednotlivé vzorky posílat do DAC převodníku jako je znázorněno v obrázku 3.8.

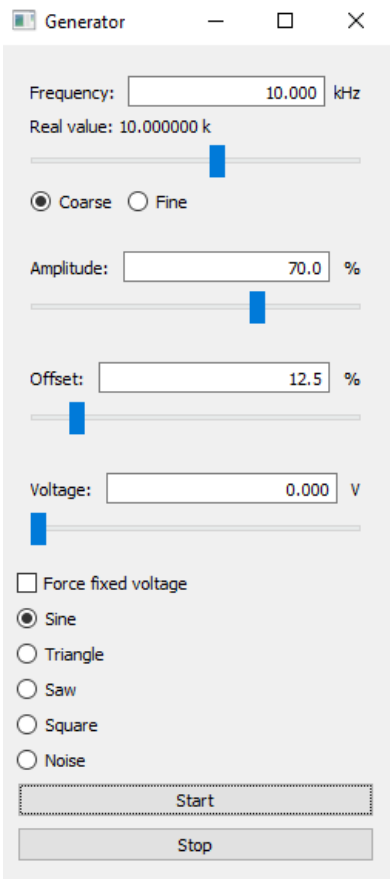


Obr. 3.8: Diagram využití DAC převodníku a dalších periferií pro generování analogového signálu

Jedním z možných způsobů je uložení pevných hodnot vzorků 1 periody signálu do paměti FLASH. Hodnoty uložené ve Flash paměti, se ale za běhu složitě upravují a jako výhodnější řešení z důvodů flexibility se zdálo řešení s ukládáním hodnot vzorků do paměti SRAM. Hodnoty vzorků jedné periody můžou být tak před samotným generováním vždy přepočítány s ohledem na nastavené parametry. Pro generaci relativně hladkých průběhů signálů jsem předpokládal, že bude dostačovat maximální délka bufferu 1000 vzorků. S ohledem na maximální vzorkovací frekvenci DAC převodníku to umožňuje generování signálů s plným počtem vzorků do výstupní frekvence f_{out}=1 kHz.



Obr. 3.9: Popis funkce kruhového bufferu jako zdroj datových vzorků pro DAC převodník



Obr. 3.10: Možnosti nastavení vlastností generovaného signálu v prostředí aplikace Zero Elab Viewer

- **Výstupní frekvence signálu** Požadovanou výstupní frekvenci signálu lze nastavovat v rozsahu 1 Hz - 250 KHz. V závislosti na nastavení výstupní frekvence signálu a omezení maximální vzorkovací frekvence DAC převodníku - se automaticky stanoví počet vzorků na jednu periodu signálu v rozsahu 4 - 1000 vzorků.
- **Rozkmit** signálu se v prostředí aplikace zadává v procentech celkového 12- Bitového rozsahu DAC převodníku, který tedy jako maximální číslicovou hodnotu kterou dokáže využít bere hodnotu 0xFFF. Rozkmit určuje, v jakém rozsahu se bude obor hodnot generovaného signálu pohybovat. Pokud označíme funkci určující tvar signálu $g(n)$ s tím, že n označuje číslo vzorku, tak můžeme psát:

$$O(g) =< 0, \text{rozkmit} \times \text{0xFFF} > \tag{3.3}$$

- **Offset** nebo také posuv signálu se opět udává v procentech rozsahu převodníku. Jedná se o nejmenší úroveň generovaného signálu respektive o hodnotu, která posouvá vypočtené hodnoty funkcí g. Díky tomu můžeme popsat funkci výpočtu n-tého vzorku bufferu :

$$f(n) = \min\left((\text{offset} \times \text{0xFFF}) + g(n), \text{0xFFF}\right) \tag{3.4}$$

- **Tvar průběhu signálu** Prostředí aplikace umožňuje zvolit různé signálové průběhy - sinusový, trojuhelníkový, signál typu pila(rampa) nebo obdélníkový.

Optimalizace délky bufferu

Obvyklý postup při generaci signálu s vyžitím DMA řadiče a paměti RAM je stanovit stálý počet vzorků N_s na jednu periodu generovaného signálu v závislosti na velikosti dostupné paměti. Dále na základě požadované výstupní frekvence generovaného signálu f_out , což je frekvence s jakou se generovaný signál má opakovat, se určí časová základna pro generování vzorků DAC převodníkem. Ta je definována dělícím poměrem DIV čítače pro generování zmíněné časové základny. Závislost tohoto dělícího poměru na frekvenci vstupních hodin toho čítače f_PCLK, výstupní frekvenci generovaného signálu f_out a počtu vzorků N_s je vyjádřena vzorcem:

$$\mathrm{DIV} = \frac{f_\mathrm{PCLK}}{f_\mathrm{out}.} \tag{3.5}$$

Nicméně zde narážíme na omezené možnosti nastavení f_out pomocí dělícího poměru DIV, který může nabývat pouze celočíselných hodnot. Pro zlepšení rozmezí nastavitelných frekvencí výstupního signálu jsem se inspiroval článkem [6], kde autor využívá toho, že nepracuje s konstantní délkou záznamové paměti, ale adaptivně se zkracuje pro jemné nastavení výstupní frekvence signálu. Metoda spočívá v iterativním zkracováním délky bufferu a minimalizování absolutní odchylky frekvence generovaného signálu.

V rovnicích 3.6 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.

$$f_\mathrm{out} = \frac{f_\mathrm{s}}{N_\mathrm{s}}, \quad f_\mathrm{s} = \frac{f_\mathrm{PCLK}}{\mathrm{DIV}} \quad \Longrightarrow \quad f_\mathrm{out} = \frac{f_\mathrm{PCLK}}{\mathrm{DIV} \cdot N_\mathrm{s}}, \quad f_\mathrm{s} = \frac{f_\mathrm{out}}{\mathrm{DIV} \cdot N_\mathrm{s}} \tag{3.6}$$

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, jako je vidět ve vzorci 3.7. Při optimalizaci délky bufferu tedy v prvním kroku nehledáme jen dělící poměr DIV, ale faktorizaci tohoto dělícího čísla, která nám minimalizuje rozdíl mezi cílenou 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 [4].

$$\mathrm{DIV} = (\mathrm{ARR} + 1) \cdot (\mathrm{PSC} + 1) \tag{3.7}$$

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 [6]. 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 0,01Hz nebo zkrácení výsledné délky bufferu na 1/4 původní délky.

Při testování funkčnosti optimalizace se potvrdila pozorování uvedené [6] a to, že touto metodou lze dosáhnout poměrně velké přesnosti při frekvencích do 500 Hz. S tím, jak dosáhneme frekvence 1000 Hz, tak se začne maximální velikost bufferu snižovat a s tím se začne snižovat schopnost optimalizace frekvence signálu. Algoritmus jsem testoval za následujících podmínek:

- Zadávané hodnoty pro požadované výstupní frekvence v rozsahu 1-10000 Hz s krokem po 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.

Algoritmus jsem nechal určovat vhodné nastavení délky bufferu a dělícího poměru pro výstupní frekvence signálu s krokem po jednom Hz a ukládal maximální absolutní a relativní odchylku od požadované výstupní frekvence. Pro jednotlivé intervaly uvedené v tabulce 3.3 jsem pak určil maximální odchylky a související frekvence pro které se tedy reálná výstupní frekvence nejvíce lišila od požadované. Označení f_{in} je pro požadované výstupní frekvence a f_{out} pak označuji reálné výstupní frekvence.

	Maximální absolutní odchylka			Maximální relativní odchylka		
Rozsah frekvencí	f_{in}[Hz]	f_{out} [Hz]	Odchylka [Hz]	f_{in}[Hz]	f_{out}[Hz]	Odchylka [%]
1-500 Hz	477	477.0058	0,0058	314	314.0046	0,0015
500-1000 Hz	977	977.0212	0,0212	977	977.0212	0,0022
1000 - 2500 Hz	2361	2360.811	0,1894	1951	1951.171	0,0087
2500 - 5000 Hz	4683	4683.841	0,8407	4683	4683.841	0,0179
5000 - 10000 Hz	9968	9936.306	31,694	9968	9936.306	0,3179

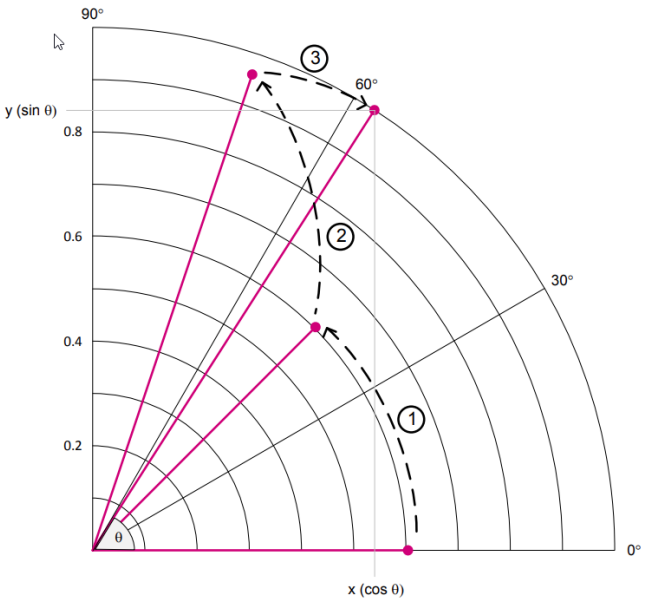
Tab. 3.3: Maximálních dosažené odchylky frekvencí výstupních signálů pro různá rozmezí frekvencí

Využití CORDIC

Jak již bylo zmíněno číslicové hodnoty jedné periody průběhu se ukládají do paměti SRAM. V případě sinusového signálu je potřeba napočítat hodnoty funkce sinus v rozsahu 2π radiánů pro získání kompletní jedné periody. Pro výpočet funkce sinus v rámci kódu mikrořadiče existují různé způsoby. Běžnou

variantou bývá využití standardní matematické knihovny `math.h` a její funkci sinf.h. Nicméně existují efektivnější způsoby výpočtu hodnot takovýchto funkcí například založené na HW implementaci jakou je například u STM32G431 speciálně dedikovaná periferie CORDIC.

CORDIC(COordinate ROtation DIgital Computer) je HW akcelerátor designovaný pro urychlení kalkulací vybraných matematických funkcí jako jsou trigonometrické a hyperbolické funkce.[13]. 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 naznačen na obrázku 3.11.



Obr. 3.11: Výpočet sinus a cosinu uhlu postupnou aproximací použitou v CORDIC převzato z [10]

Při implementaci signálového generátoru své práci jsem se snažil zjistit možný přínos využití této metody výpočtu funkce sinus. Jak již bylo zmíněno při generování signálu je zapotřebí napočítat hodnoty celého bufferu a tedy je potřeba dané matematické operace opakovat až 1000krát. Zdálo se tedy vhodné ověřit časovou náročnost výpočtu a zároveň ověřit potenciální přínos využití HW orientovaného řešení.

Pro porovnání časové náročnosti výpočtu hodnot bufferu jsem zvolil podmínky, které jsou stejné s následnou implementací FW pro použití s Zero Elab aplikací. Tedy jsem porovnával za jakou dobu buffer o 1000 vzorcích přichystá MCU s využitím CORDIC a kolik času bude zapotřebí s využitím funkce `sinf(x)` ze zmíněné standardní knihovny. Měřený úsek byl 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 kódu se zaměřením na rychlost (-OSpeed). V tabulce 3.4 jsou pak srovnané dosažené výsledky.

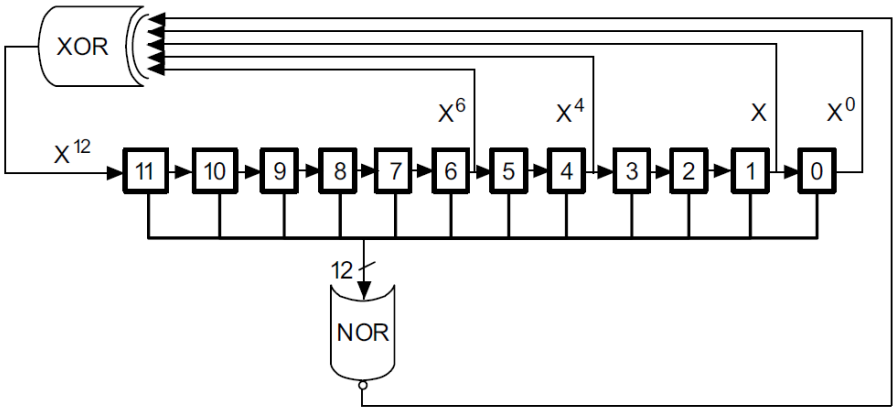
Použitá metoda	Doba plnění bufferu	Čas při SYSCLK = 156MHz
CORDIC	37541 cyklů	0.241 ms
sinf(x) z math.h	85573 cyklů	0.549 ms

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

Z tabulky je vidět, že výpočet pomocí CORDIC byl v průměru asi o 120% rychlejší. Nicméně obě doby výpočtu v případě využití systémových hodin s frekvencí 156 MHz jsou pod 1 ms a zřejmě tedy tyto výpočty hodnot bufferu pro DAC převodník nebudou mít žádný dopady na plynulost běhu aplikace a uživatel dobu výpočtů hodnot nijak nezaznamená.

3.3.3 Generování šumu

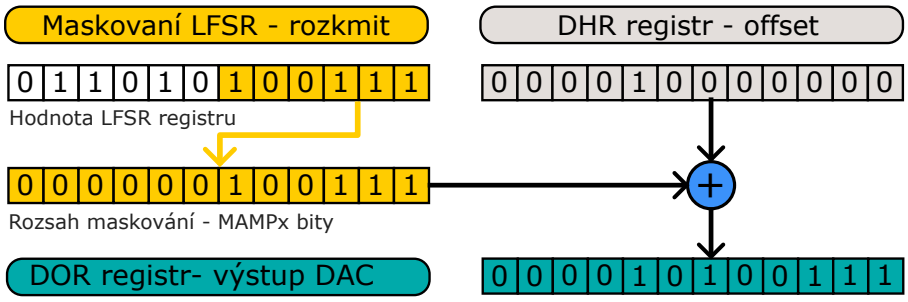
Pro některé experimenty v elektronice se může hodit možnost generování nastavitelného šumového signálu. Generování šumu může být využito při testování fungování obvodu za přítomnosti šumu nebo existují postupy umožňující efektivní zvýšení rozlišovací schopnosti DAC převodníků založené právě přidávání šumu do měřeného signálu. 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 3.12.



Obr. 3.12: Naznačení průběhu výpočtu hodnot LFSR registru převzato z [9]

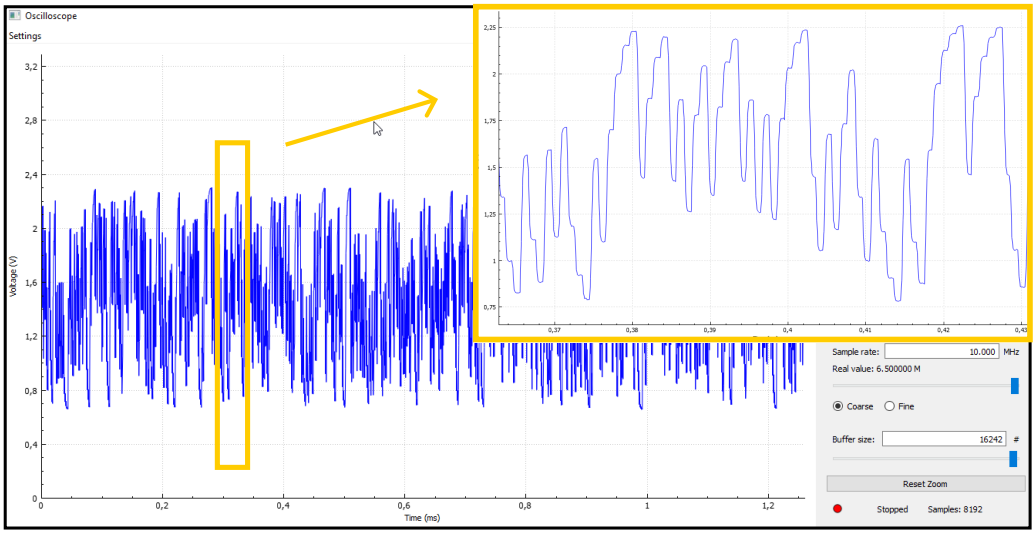
Samotné využití LFSR v DAC převodníku spočívá maskováním bitů tohoto registru pro získání určitého rozmezí(rozkmitu) výstupních hodnot a následným přičtením této hodnoty získané maskováním do DHR(Data-Hold) registru. Hodnota registru DHR určuje nejmenší hodnotu posílanou na výstupu DAC převodníku - tedy 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 3.13

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. 3.13: Využití LFSR registru pro generování šumu

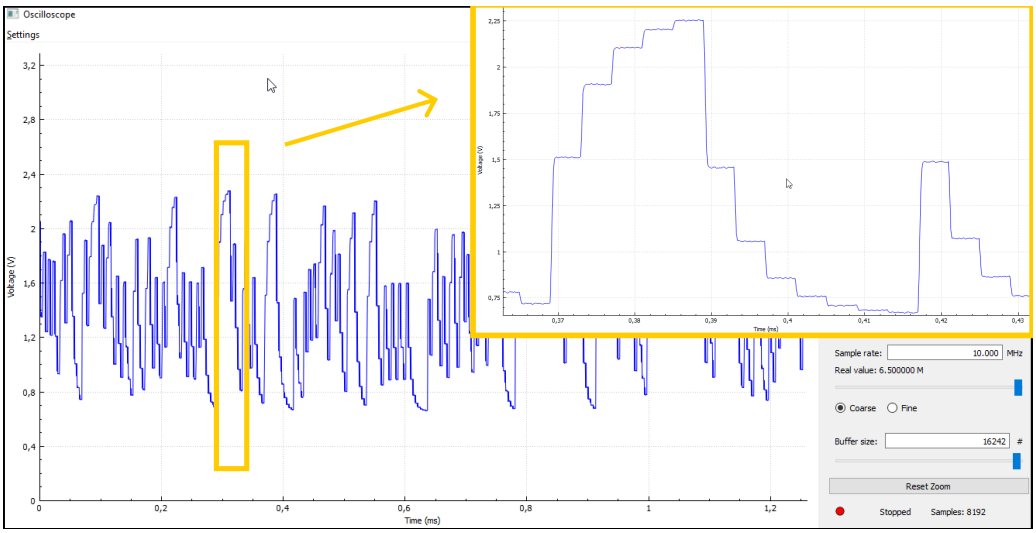
Z obrázku 3.13 vyplývá jakým způsobem lze generování šumového signálu nastavovat podobně jako u ostatních tipů signálů pouze s drobnými rozdíly. Například hodnota nastavení posuvu generovaného signálu(offset) se dá nastavit v celém rozsahu DAC převodníku. Naopak menší počet různých hodnot lze nastavit pro rozkmit generovaného signálu. Ze strany počítačové aplikace se sice zdá, že ho lze nastavovat v desetinách procenta nicméně tím, že jediný možný způsob přepočtu LFSR registru je maskováním bitů LFSR, tak lze velikost rozkmitu nastavit pouze ve 12 krocích. S tím že krok následující je polovinou kroku předchozího. Pro 50.1% až 100% rozkmit(a nulový offset) je na výstupu plný rozkmit šumového signálu, dalším krokem pak je rozkmit 50% - maskujeme nejvyšší bit a a tak dále...



Obr. 3.14: Záznam generovaného signálu šumu se vzorkovací frekvencí 1 Msps

Dalším nastavitelným parametrem je frekvence, která v tomto případě představuje vzorkovací frekvenci se kterou jsou nové vzorky posílány na výstup. Vzorkovací frekvence lze nastavit v rozsahu 1 Hz a 1 MHz. Nicméně vzorkovací frekvence 1MHz není pro šumový signál optimální, protože se zde začne projevovat konečná rychlost přeběhu DA převodníku zakulacováním hran jako je vidět na obrázku 3.14 Pro měření kde by se mohla projevit filtrace způsobená tímto jevem omezené rychlosti přeběhu, je lepší využít nižší vzorkovací frekvence, kde již tento problém vizualizovaný zakulacováním

hran není patrný a neovlivní průběh měření. Detail generování signálu s nižší vzorkovací frekvence (250 Ksps) je vidět na obrázku 3.15.



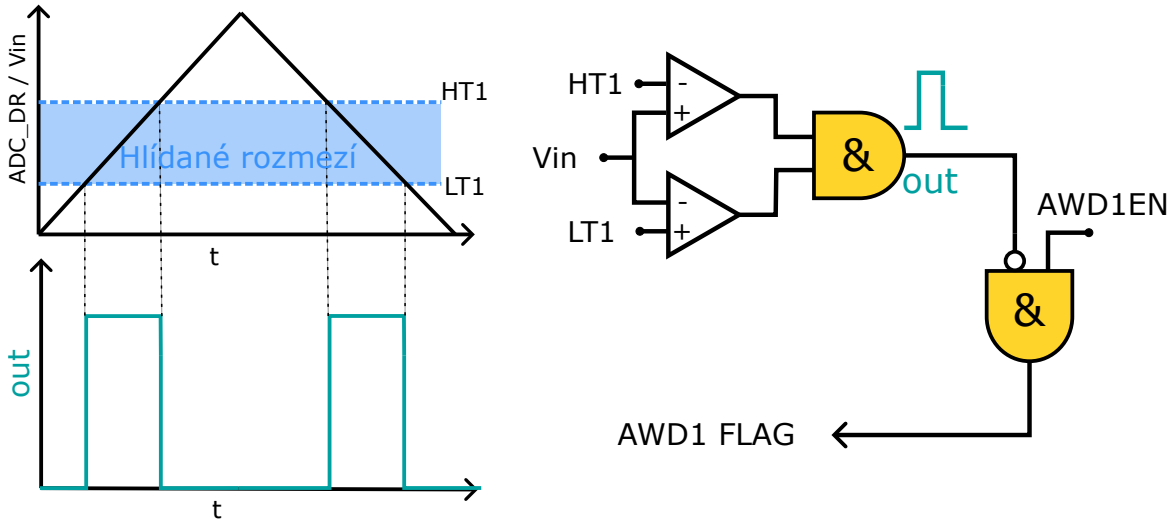
Obr. 3.15: Záznam generovaného signálu šumu se vzorkovací frekvencí 250 Ksps

3.4 Realizace osciloskopu

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

3.4.1 Řešení triggerování osciloskopu

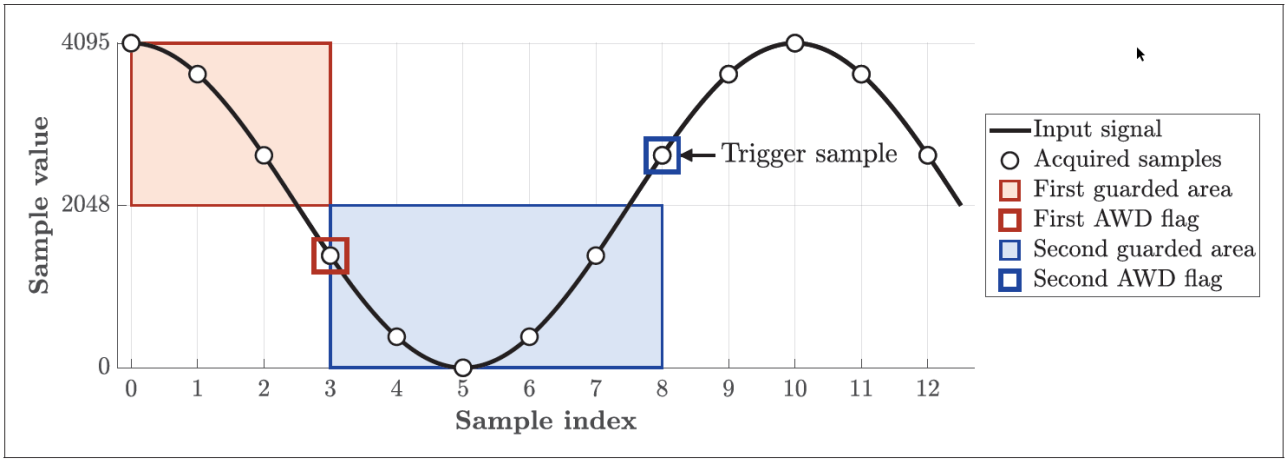
Určení okamžiku náběžné nebo sestupné hrany vstupního signálu je podstatnou funkcí osciloskopu pro zobrazování jak přechodných jevů tak periodických průběhů signálu. Existuje více přístupů, jak takovou funkci implementovat. Jednou z nejjednodušších variant je SW orientované řešení, kdy jsou naměřená 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 [2]. 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. 3.16: Princip funkce Analog Watchdog(AWD) ADC převodníku

Blok 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 3.17.

- **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.

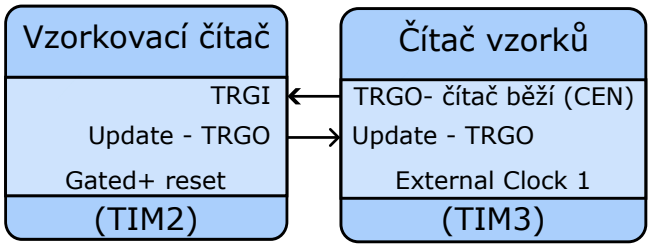


Obr. 3.17: 2-fázové triggerování na nástupnou hranu signálu s využitím AWDG- převzato z [4]

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 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.



Obr. 3.18: Vzájemné propojení dvou čítačů

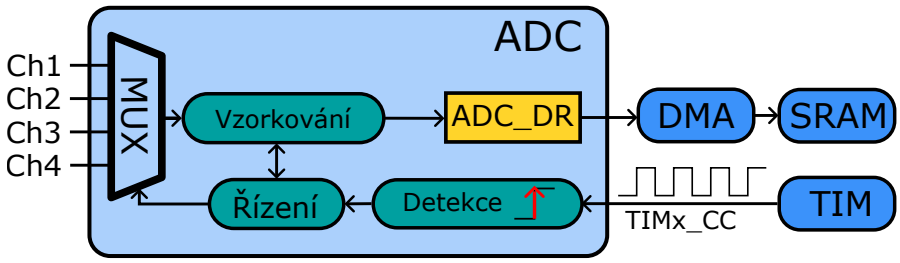
3.4.2 Módy vzorkování

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 variabilitu, jak vzorkovat daný set analogových kanálů. Podrobněji se jednotlivým módům věnuje práce [4]. STM32G431 má k dispozici 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

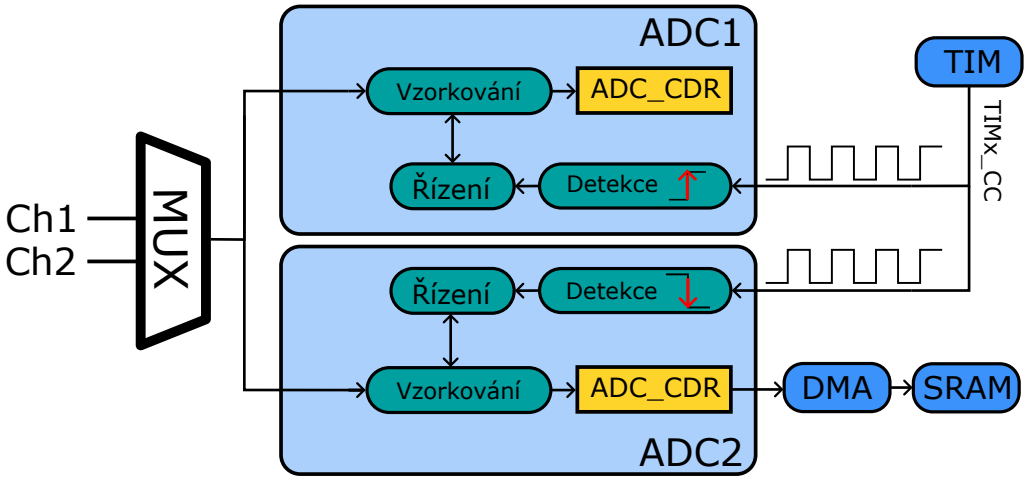
vzorkovací frekvence, respektive zvýšení doby vzorkování pro stejnou vzorkovací frekvenci oproti použití 1 ADC. Výběr použitého módu vzorkování pak záleží na zvolených kanálech, respektive na jejich počtu a na jejich přítomnosti na jednotlivých ADC převodnících. Například kanál osciloskopu číslo 1 je 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. 3.19: HW konfigurace 1 ADC převodníku pro režim s multiplexováním vstupních kanálů

Vzorkování 1 kanálu

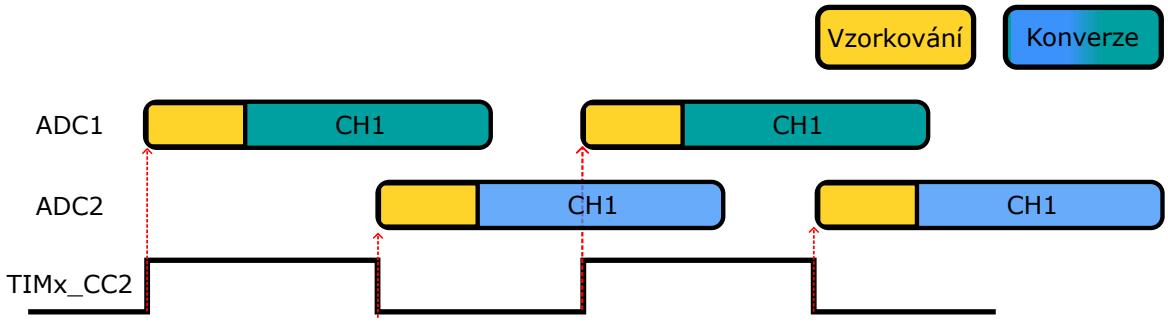
Pokud je zvolený analogový kanál osciloskopu využitím vnitřních multiplexerů je možné přivést zároveň (paralelně) na vstupy obou ADC převodníků, můžeme využít tzv. "Dual -interleaved modu". Tento režim spočívá ve střídavém vzorkování jedním převodníkem a v průběhu konverze odebraného vzorku vzorkováním 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. Výsledně lze pak odebrat vzorek až každých 8 cyklů. Což dovoluje téměř zdvojnásobit maximální vzorkovací frekvenci. Druhou výhodou je možnost zvolení delší doby vzorkování při měření s nižší vzorkovací frekvencí více v kapitole 3.4.3.



Obr. 3.20: HW konfigurace 2 ADC převodníků pro režim "Independent interleaved"

Nevýhodou pro STM32 ADC nativního 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 stejný. 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 obdélníkový signál se střídou 50% , ADC1 začíná vzorkovat na hranu náběžnou a ADC2 na hranu sestupnou. Každý cyklus čítače tedy znamená odebrání 2 vzorků. Průběh odběru vzorků je znázorněn na obrázku 3.21

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 registrů jako v případě obvyklého 'dual-interleaved' módu. 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.



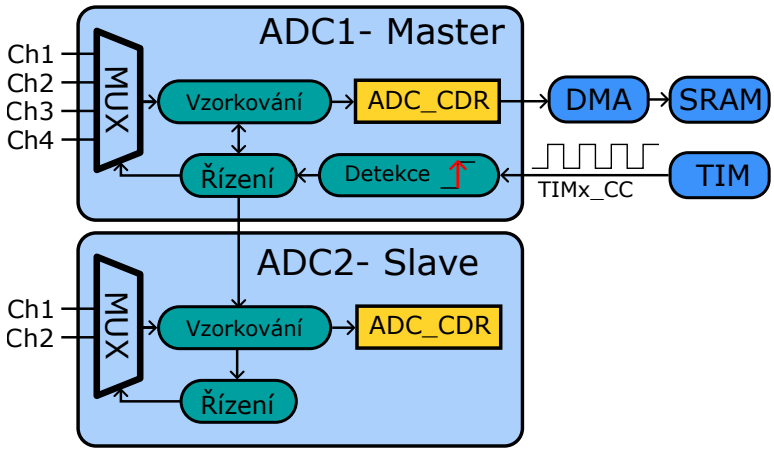
Obr. 3.21: Vzorkování v režimu "Independent Interleaved"

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 zároveň. V praxi nás často zajímá napětí na dvou uzlech obvodu ve stejnou chvíli. Pokud měříme pouze jedním ADC převodníkem, tak mezi odběry vzorků je měřitelné zpoždění o které bychom museli naměřená data poté korigovat.

Dostupnost fyzických propojení mezi piny a analogovými kanály jednotlivých ADC převodníků se mezi různými mikrokontrolery dost liší to i v závislosti na použitém pouzdře. Abych udělal firmware jednodušeji portovatelný na další zařízení, tak jsem mém FW 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 snímaly stejný počet kanálů. Vznikají tak různé konfigurace na základě zvolených kanálů.

Při analogových kanálech, tak jak jsou popsány v tabulce 3.5,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 snímány kanály CH1+CH3 a CH2+CH4. Po navzorkování bufferu je pak případně nezvolený kanál vyfiltrován z bufferu před posláním naměřených dat do počítače.

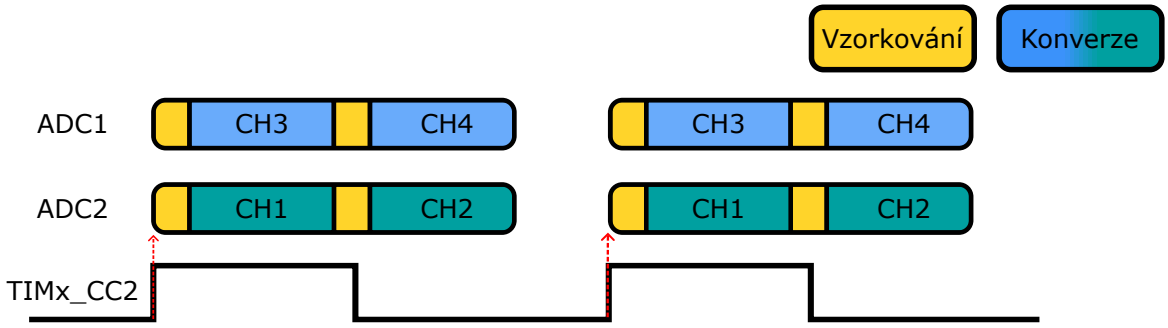


Obr. 3.22: HW konfigurace 2 ADC převodníků pro režim "Dual simultaneous"

Samotné používání 2 ADC převodníků využívá pro STM32 ADC nativního DUAL simultaneous režimu řízení vzorkování. V tomto režimu spouštíme vzorkování pouze řídícího(MASTER) ADC převodníku, který sám řídí podřazený ADC převodník. HW konfigurace včetně zbylých periferií je ukázána na obrázku 3.22. V tomto režimu se také podobně jako je tomu u interleaved režimnu používá

společného data registru(CDR), díky čemuž můžeme naměřená data z obou převodníku přenést pomocí DMA řadiče najednou.

Pokud sbíráme data z 4 analogových kanálů najednou je zapotřebí použít vnitřních multiplexerů ADC převodníku jako je tomu v případě více kanálů na jednom ADC převodníku. Průběh vzorkování pro 4 kanály je ukázán obrázku 3.23. Je patrné, že se kanály CH2 a CH4 budou nutně zpožďovat za kanály CH1 a CH3. Toto zpoždění je závislé na době vzorkování, kterou v rámci své práce upravuji na základě zvolené vzorkovací frekvence, tak aby doba odběru vzorku byla pokud možná nejdelší.

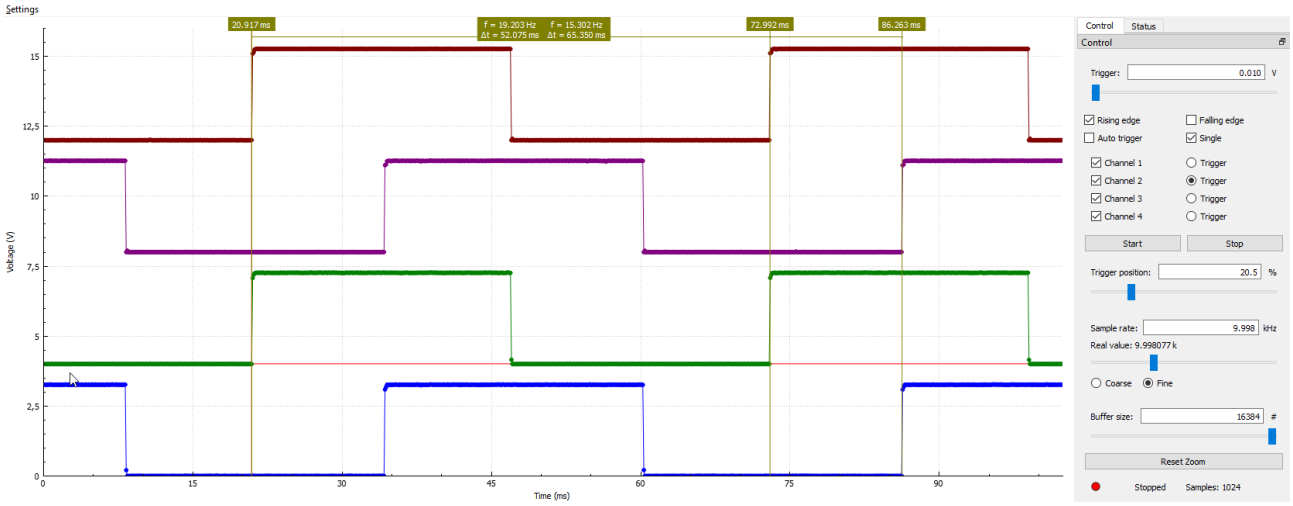


Obr. 3.23: Vzorkování 4 kanálů v režimu "Dual simultaneous"

V běžném režimu vzorkování si zpoždění odběru vzorku pravděpodobně nevšimneme, protože zpoždění bude typicky menší než doba která uběhne mezi 2 po sobě jdoucími odběry. Nicméně zpoždění je velmi patrné při vzorkování v ekvivalentním čase. Na obrázku 3.24 je vidět příklad vzorkování s ekvivalentní vzorkovací frekvencí 52 MHz. Zpoždění mezi kanály je v tomto případě rovno $N_{\mathrm{Delay}} = 653$ cyklů ADC převodníku, což při frekvenci hodin ADC převodníku $f_{\mathrm{ADC}} = 52$ MHz odpovídá zpoždění $T_{\mathrm{Delay}} = 12.55\mu$s

$$T_{\mathrm{Delay}} = \frac{N_{\mathrm{Delay}}}{f_{\mathrm{ADC}}} = \frac{653}{52\mathrm{MHz}} = 12.5\mu\mathrm{s} \tag{3.8}$$

Na vstup byl při tom přiveden obdélníkový signál s frekvencí 10000 KHz, jehož perioda T=10μs. Tedy zpoždění signálu je o více než jednu periodu snímaného signálu. Pokud bychom tedy chtěli například měřit zpoždění signálu na nějakých logických obvodech, tak je potřeba vzít existenci tohoto zpoždění v potaz a nebo použít kanály, které jsou vzorkované zároveň.

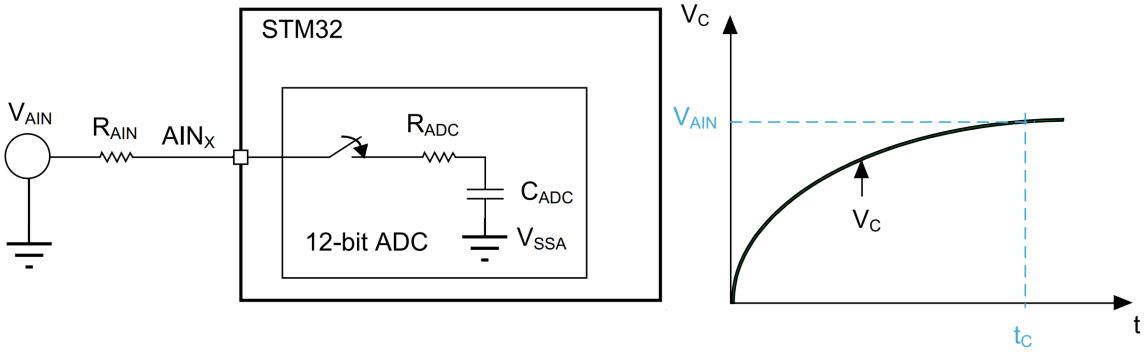


Obr. 3.24: Zpoždění kanálu 2 a 4 za kanálem 1 a 3 při použití Dual simultaneus modu a stroboskopickém vzorkování s ekvivalentní vzokovací frekvencí 52MHz

3.4.3 Nastavení délky vzorkování a vliv na maximální vstupní odpor

Impedance analogového zdroje signálu, respektive sériový odpor(R_{AIN}) mezi zdrojem a pinem, ovlivňuje proud nabíjející vzorkovací kondenzátor. Časová konstanta nabíjení t_c potom určuje minimální dobu vzorkování $T_s > tc$, tak aby se vzorkovací kondenzátor měl čas nabít na vstupní napětí(s tolerancí 1/2 LSB) V_{AIN} jako je znázorněno na obrázku 3.25.

$$t_c = (R_{\mathrm{ADC}} + R_{\mathrm{AIN}}) \times C_{\mathrm{ADC}}, \quad T_s = \frac{N_{\mathrm{s}}}{f_{\mathrm{ADC}}} \tag{3.9}$$



Obr. 3.25: převzato z [12]

Doba vzorkování T_s je určena počtem cyklů doby vzorkování N_{s} hodinového signálu ADC (ADC_CLK) převodníku, jehož frekvenci označuji f_{ADC}. Typicky je pro daný ADC převodník omezený výběr hodnot N_{s}, které jsou k dispozici. V případě STM32G431 se jedná o výběr 8 různých hodnot viz tabulka 3.6. Kromě zvolení hodnoty N_{s}, lze ovlivnit dobu vzorkování T_s také změnou frekvence f_{ADC}.

Při řešení osciloskopu jsem zvolil frekvenci ADC_CLK jako maximální doporučenou frekvenci při používání 2 ADC převodníků zároveň $f_{\mathrm{ADC}} = 52\mathrm{MHz}$ dle technické specifikace uvedené v [11]. N_s je ve FW přizpůsobovaná aktuální vzorkovací frekvenci f_{SAMP} takovým způsobem, aby se všechny aktuálně zvolené kanály stihly navzorkovat a zkonvertovat před tím, než přijde z čítače časové základny pokyn k dalšímu odběru vzorků.

Výběr vhodného počtu cyklů doby vzorkování N_s pro danou vzorkovací frekvenci f_s probíhá následovným způsobem. Máme nějaký počet dostupných ADC cyklů pro odběr a konverzi vzorků $N_{\mathrm{available}}$ a celkový potřebný počet cyklů pro odběr vzorků N_{total} pro které platí:

$$N_{\mathrm{total}} = (N_s + N_c) \times N_{\mathrm{ch}}, \quad N_{\mathrm{available}} = \frac{f_{\mathrm{ADC}}}{f_{\mathrm{SAMP}}} \tag{3.10}$$

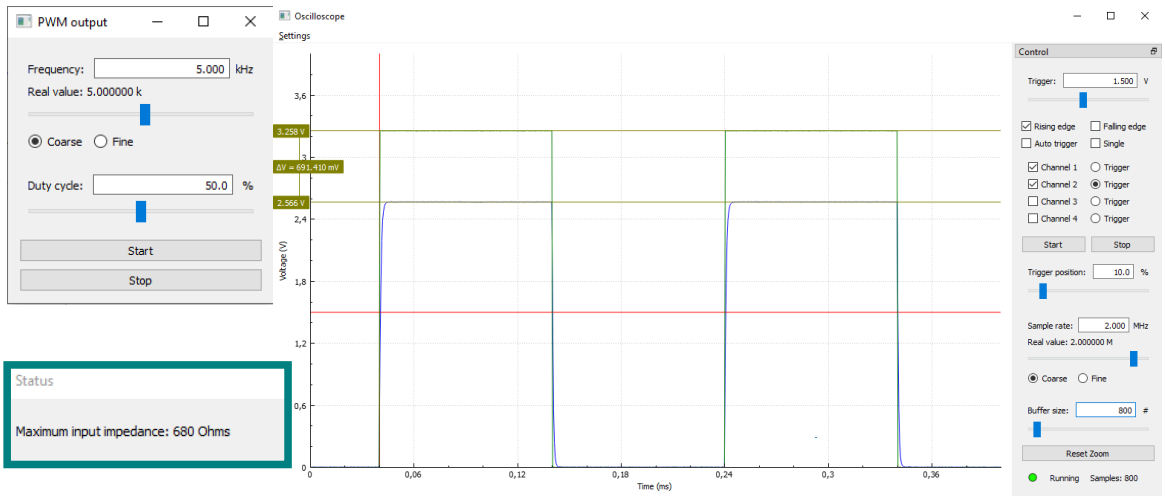
kde $N_c = 12.5$ označuje potřebný počet cyklů ADC převodníku pro konverzi 12-bitové naměřené hodnoty a N_{ch} označuje kolik kanálů vzorkuje 1 ADC převodník v případě multiplexování kanálů. Následně volíme hodnotu N_s ze seznamu dostupných hodnot tak, aby byla bylo dosaženo maximální možné doby vzorkování T_s aniž by došlo k překročení dostupného počtu cyklů N_{total}. Doba vzorkování ovlivňuje měření a pro uživatele může být výhodné tuto informaci znát. Z tohoto důvodu po zvolení počtu cyklů MCU informuje uživatele skrz PC aplikaci formou informace o maximální impedanci zdroje signálu aby byla dodržena podmínka uvedená v 3.9 a doba vzorkování T_s delší než velikost časové konstanty nabíjení vnitřního vzorkovacího kondenzátoru t_c.

Zmíněný vzorec je vhodné použít pokud známe přesné vnitřní parametry ADC převodníku. Ty v mém případě nebyli známy a z dat v datasheetu STM32G431 vyplívá, že například hodnota vnitřního odporu ADC převodníku R_{ADC} se mění v závislosti na konfiguraci. Pro určení maximální vstupního odporu pro danou dobu vzorkování T_s jsem tedy místo vzorce použil hodnoty uvedené v datasheetu[11]. V mé implementaci byli využity tzv."Fast channels", což jsou vnitřní kanály 1 až 5 ADC převodníku, které umožňují vzorkování vstupů s daným vnitřním odporem v kratším čase. V tabulce 3.6 pak uvádím přehled hodnot maximálního vstupního odporu v souvislosti se zvolenou dobou vzorkování. Na základě této tabulky a informace z aplikace o maximálním vstupním odporu pak uživatel může určit použitou dobu vzorkování.

Max R_{AIN} [Ω]	100	330	680	1500	2200	4700	12000	39000
N_{s}	2.5	6.5	12.5	24.5	47.5	92.5	247.5	640.5
T_s [ns]	48	125	240	471	913	1779	47660	12317

Tab. 3.6: Přehled hodnot maximálního vstupního odporu společně s odpovídající dobami vzorkování

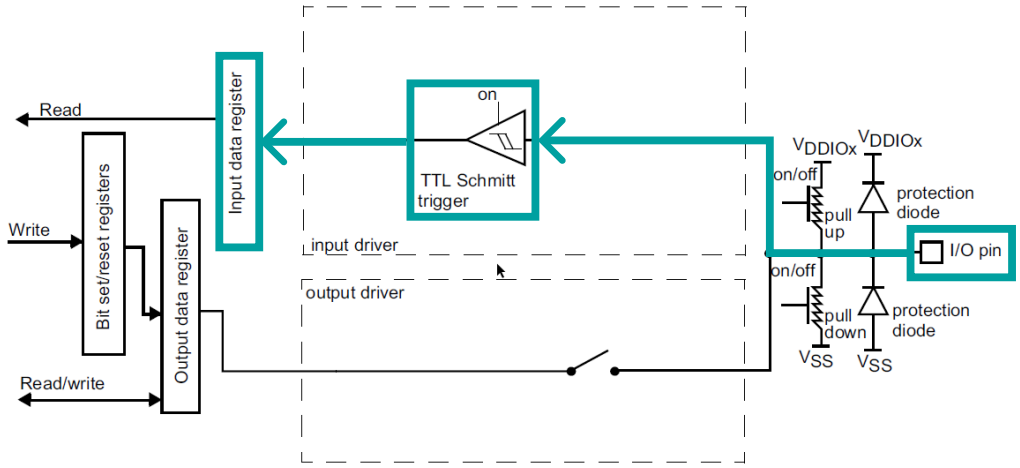
Na obrázku 3.26 je pak vyobrazen příklad, kdy pomocí impulzního generátoru generuji PWM signál s frekvencí f_{pwm} =5 kHz přivedený na vstupy ADC převodníků. Jeden z kanálů je připojen na přímo a druhý kanál přes odpor 47 kΩ. Pro vzorkovací frekvenci $f_{\text{SAMP}} = 2$MHz dostaneme informaci o maximálním vstupním odporu $R_{\text{AIN}} = 680\Omega$. Odpovídající dle tabulky 3.6 době vzorkování $T_s = 240$ ns. Na vstupním kanálu číslo 1, který je spojen se impulzním generátorem přes zmíněný odpor R=47 kΩ je vidět pokles napětí $\Delta V = 691$ mV způsobený tím, že se vzorkovací kondenzátor se nestíhá dobíjet nedodržením podmínky 3.9.



Obr. 3.26: Efekt nedodržení maximálního vstupního odporu

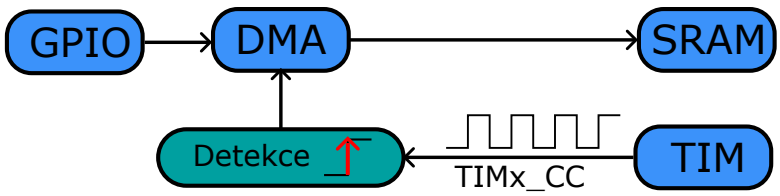
3.5 Logický analyzátor

Logický analyzátor nahrazuje funkci osciloskopu pro sledování logických signálů, jakými jsou například signály sériových sběrnic typu UART nebo I2C. V případě osciloskopu máme velmi omezený počet kanálů, které můžeme vzorkovat zároveň daný počtem dostupných ADC převodníků. Tento problém sice můžeme řešit multiplexováním vstupních kanálů, jak bylo popsáno kapitole 3.4.2 nicméně to má zase velký vliv na maximální vzorkovací frekvenci. Zároveň nutnost vzorkování a následné konverze dále zpomaluje celé vyčítání stavu logického kanálu.



Obr. 3.27: Vnitří struktura pinu v "Input"konfiguraci. Převzato z [9]

Díky tomu, že u logických kanálů rozlišujeme jen 2 napěťové úrovně, tak ADC převodník pro určení stavu kanálu nepotřebujeme a stačí využít vnitřní struktury vstupních pinů (Obr. 3.27). Součástí této struktury je registr vstupních dat(GPIO_IDR), který zachycuje stav I/O pinu každou periodu hodin AHB sběrnice. Tento registr pak obsahuje stav všech pinů náležící k jedné GPIO bráně a za předpokladu, že jsou sledované vstupy v rámci jedné GPIO brány, tak můžeme vyčíst stav všech vstupů zároveň jedním přístupem do registrů.



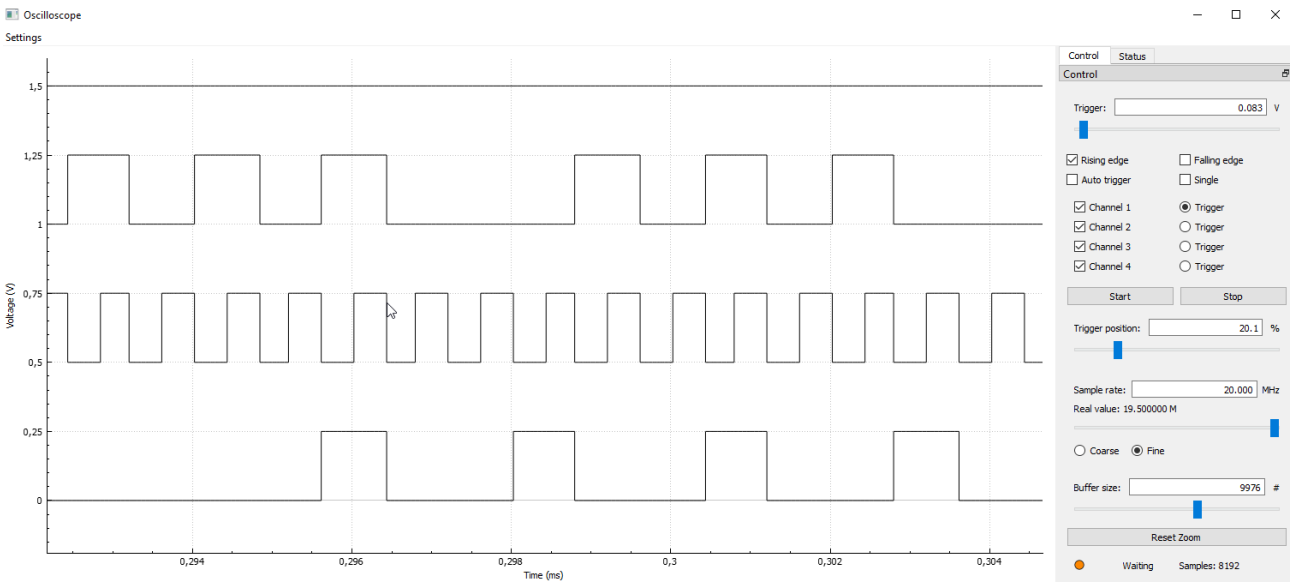
Obr. 3.28: HW konfigurace pro logický analyzátor

Pro synchronní vyčítání registru GPIO_IDR můžeme efektivně využít DMA řadiče ve spojení s čítačem podobně jako v případě osciloskopu. Čítač pak určuje časovou základnu pro odběr vzorků řídí přenosy hodnot registru do bufferu v SRAM paměti. Tato propojení periferií je znázorněno na obrázku 3.28.

Maximální vzorkovací frekvence, kterou jsme schopni snímat vstupy logického analyzátoru je limitována rychlostí DMA přenosu. DMA přenos typicky trvá 6 cyklů hodin AHB sběrnice [3]. S tím, že přenos může trvat déle v případě souběžných přístupů do SRAM paměti nebo v případě vytíženosti AHB sběrnice dalšími procesy. Z tohoto důvodu jsem pro přenos počítal $N_{\text{DMA\_cyklů}} = 8$ cyklů, abych snížil vytížení sběrnice a nedocházelo ke kolizím. Maximální vzorkovací frekvence f_s logického analyzátoru je tedy určena při dané frekvencí hodin AHB sběrnice f_{HCLK} jako:

$$f_s = \frac{f_{\text{HCLK}}}{N_{\text{DMA\_cyklů}}} = \frac{156\ \text{MHz}}{8} = 19.5\ \text{MHz} \tag{3.11}$$

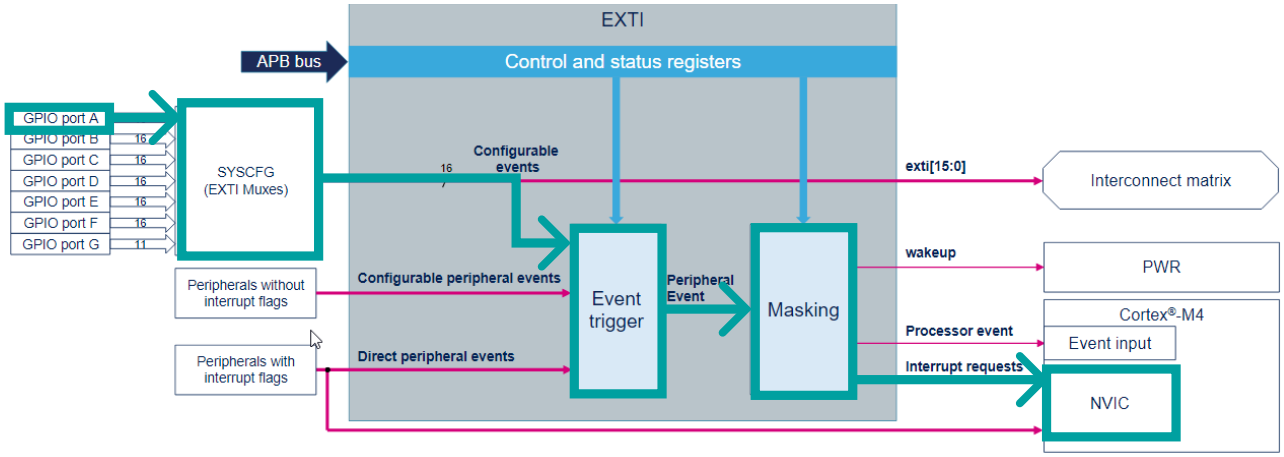
Maximální vzorkovací frekvence až 19,5 MSps, je dostatečná pro řadu využití jako je například analýza sériové komunikace pomocí I2C, UART nebo SPI, se kterými se studenti v hodinách běžně potkávají.



Obr. 3.29: Záznam SPI komunikace s baudrate = 125 KBits/s

3.5.1 Triggerování logického analyzátoru pomocí externího přerušení - EXTI

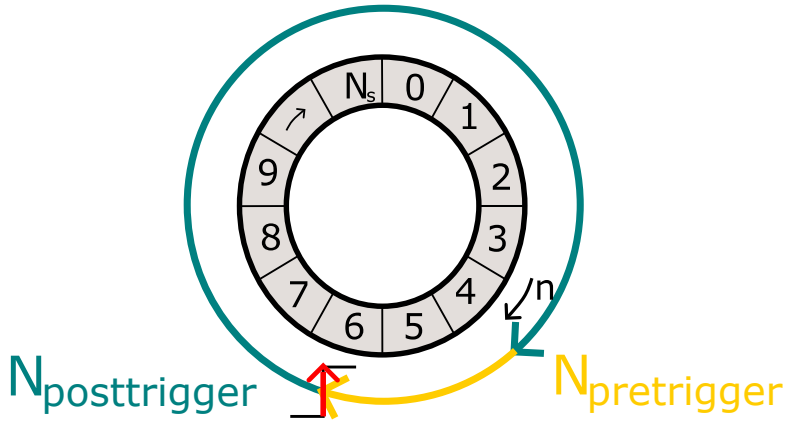
Podobně jako u osciloskopu je i u logického analyzátoru potřeba triggerování záznamu některým ze zvolených signálů. Narozdíl od osciloskopu se v případně logických signálů nemusí vyhodnocovat úrovně napětí, ale rozhodující úlohu hrají náběžné nebo spádové hrany. Pro tento účel lze velmi účelně využít dostupnou řídící jednotku pro přerušení a události - EXTI (extended interrupts and events controller). Tato řídící jednotka slouží jako prostředník mezi periferiemi a řídicí jednotkou přerušení (NVIC). Díky tomu stejný signál, který využíváme jako vstupní může tak zároveň sloužit jako signál přerušení. V momentě, kdy na zvoleném vstupním signálu dojde ke změně napěťové úrovně - tedy k náběžné nebo sestupné hraně(podle volby) vstupního signálu, tak dojde k vyvolání přerušení běhu programu.



Obr. 3.30: EXTI blokový diagram - konfigurace vyvolání přerušení daným pinem - převzato z [8]

Po vyhodnocení události přerušení pak dojde přenastavení čítače, který počítá odebrané vzorky, tak aby odebral zbývající počet vzorků $N_{\text{posttrigger}}$ a následně zastavil měření. $N_{\text{posttrigger}}$ se odvíjí od celkové délky bufferu N_s a zadané hodnoty pre-triggeru v procentech.

$$N_s = N_{\text{pretrigger}} + N_{\text{posttrigger}}, \quad N_{\text{pretrigger}} = \text{Pretrigger}[\%] \times N_s, \quad N_{\text{posttrigger}} = N_s \times (1 - \text{Pretrigger}[\%]) \tag{3.12}$$



Obr. 3.31: Rozdělení kruhového bufferu na data před a po přerušení

V rámci mého řešení takto lze triggerovat záznam logického analyzátoru pomocí jednoho ze 4 různých vstupních kanálů s tím, že lze zvolit chceme-li triggerovat na náběžnou hranu, sestupnou hranu nebo obě, kdy jakákoli hrana, která dorazí na vstup zvoleného signálu vyvolá přerušení. Díky implementaci pretriggeru lze pak sledovat co se odehrávalo na vstupech před vyvoláním přerušením.

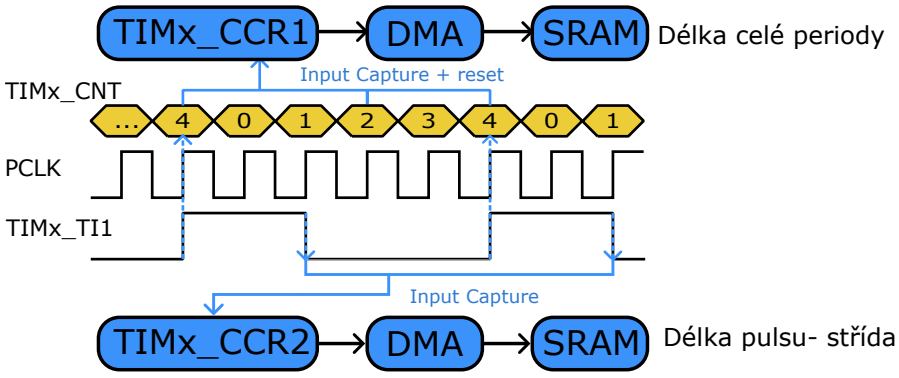
3.6 Realizace funkce čítač - měření frekvence a střídy

Přístroj typu čítač s funkcí měření frekvence vstupního signálu se hodí v mnoha případech. Pokud například chceme využít osciloskopu v režimu vzorkování v ekvivalentním čase, tak potřebuje znát přesnou frekvenci vstupního signálu, abychom mohli správně zvolit vzorkovací frekvenci osciloskopu. Využitím vnitřních periferií čítačů pro měření frekvence vstupních signálů navíc můžeme měřit větší rozsah frekvencí než například interpretací naměřených dat z osciloskopu, kde jsme limitováni vzorkovací frekvencí ADC převodníku. Kromě měření frekvence lze vnitřních čítačů využít i například pro měření střídy vstupního signálu. V rámci této práce jsem adaptoval 2 různé postupy pro měření frekvence s tím že mezi způsoby měření si v aplikaci můžeme vybrat.

Reciproční metoda měření frekvence, která spočívá v měření délky periody se hodí více pro pomalejší signály a v rámci mé implementace navíc umožňuje měření střídy signálu. Druhou metodou, kterou jsem se v této práci zabýval je metoda čítáním pulzů. která se naopak hodí více pro

3.6.1 Reciproční metoda měření frekvence určením délky periody - PWM input režim

Upravenou verzi metody měření délky periody vstupního signálu popsané v kapitole 3.2.1, můžeme využít pro stanovení vlastností PWM signálu. Kromě určení délky periody tak lze čítač využít i k určení střídy PWM signálu. Metoda je popsaná v [9] a označována jako PWM input režim. Režim spočívá v konfiguraci čítače v režimu, kdy zaznamenává hodnotu CNT registru střídavě pro náběžnou i sestupnou hranu s tím, že v případě příchodu náběžné hrany zároveň čítání vyresetuje, tak jako je znázorněno na obrázku 3.32.



Obr. 3.32: Princip funkce měření v tzv. PWM input režimu

Ve své implementaci jsem přejal řešení popsané v práci [2]. Toto řešení adaptivně upravuje frekvenci vstupních hodin čítače pomocí nastavování před-děličky signálu, aby se zvyšovalo rozlišení, se kterým je vstupní signál měřen. Pokud je totiž signál příliš pomalý ve srovnání se vstupními hodinami čítače,

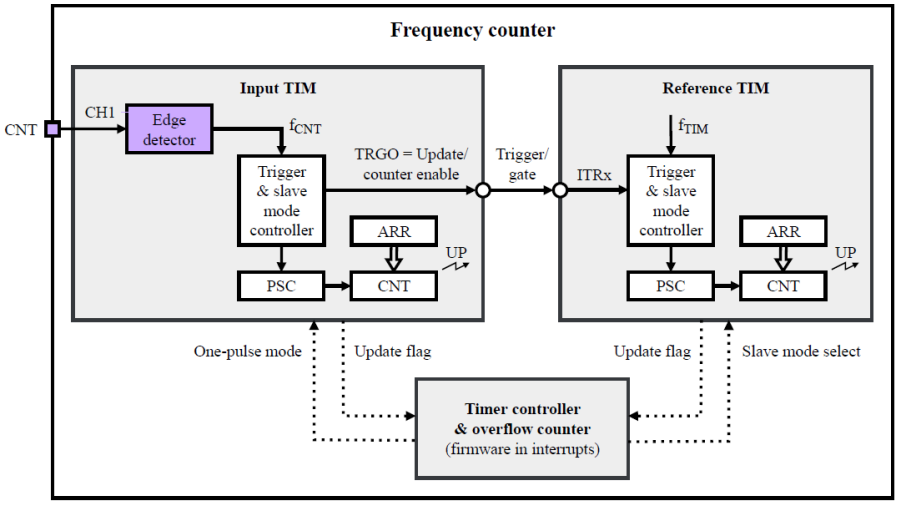
tak dojde k přetečení čítače, ještě před tím než dojde k zaznamenání délky pulzu, čímž by to došlo k chybnému odečtu délky periody. A naopak pokud se bude frekvence vstupních hodin blížit k frekvenci vstupního signálu, je potřeba frekvenci vstupních hodin čítače zvýšit, abychom zvýšili rozlišení měření.

Dále jsem využil DMA převodníku pro zaznamenání 10 po sobě jdoucích period vstupního signálu, abych snížil nepřesnosti měření.

3.6.2 Metoda měření frekvence čítáním pulsů za časový interval

Pro měření signálů s vysokými frekvencemi se více hodí alternativní metoda měření, kdy neměříme dobu trvání periody, ale počítáme počet period signálu za nějaký určený časový interval t_{gate}. Nevýhodou této metody bývá, že u pevně stanoveného časového úseku měření není obvykle na začátku a na konci zaznamenaná celá perioda vstupního signálu, což je zdrojem nepřesnosti měření. Pro řešení tohoto v problému autor v práci [4] navrhl řešení, které se snaží toto omezení řešit.

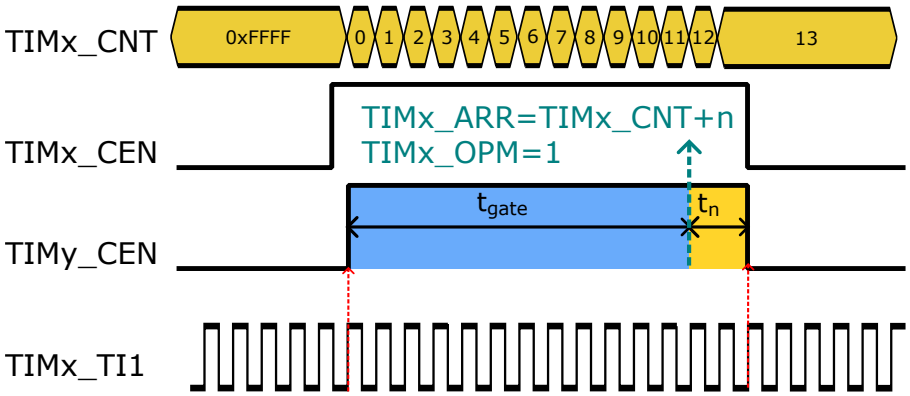
Řešení spočívá v propojení dvou čítačů. Jako u obvyklého řešení první z čítačů počítá vstupní pulzy(Vstupní čítač) a druhý čítač odměřuje dobu, která uběhla, pomocí interních vstupních hodin čítače PCLK(Referenční čítač). Řídicí jednotky čítačů jsou vzájemně propojené tak, aby po spuštění měření první náběžná hrany na vstupu spustila běh referenčního čítače a po uplynutí stanovené doby měření hlídané referenčním čítačem ještě referenční čítač odměřil dobu trvání celých period vstupního signálu a zastavily se oba čítače zároveň. Schematické propojení čítačů je vyobrazeno na obrázku 3.33.



Obr. 3.33: Diagram propojení čítačů pro upravenou metodu měření frekvence čítáním pulzů. Převzato z [4]

Časový diagram měření je znázorněn na obrázku 3.34 s tím, že TIMx označuje vstupní čítač a TIMy čítač referenční. Průběh měření se dá popsat následujícími kroky:

- Měření je zahájeno zapnutím vstupního čítače. TIMx_CEN = 1;
- První nástupní hrana vstupního signálu vyresetuje oba čítače a zapne referenční čítač, který začne odměřovat uběhlý čas pomocí interních hodin PCLK.
- Po uplynutí stanovené doby t_{gate} se nastaví kolik n vstupních period signálu se ještě má odměřit pro doběhnutí měření. Toho docílíme nastavením registru ARR vstupního čítače na aktuální hodnotu CNT registru zvýšenou o n a nastavením vstupního čítače do One pulse režimu (OPM).
- Díky nastavenému OPM režimu se po doběhnutí vstupního čítače do hodnoty ARR oba čítače automaticky zastaví.



Obr. 3.34: Časový diagram čítání pulzů za proměnnou dobu odběru

Výše popsaným způsobem získáme dobu měření $T_{\text{meas}} = t_{\text{gate}} + t_{\text{n}}$ a počet period vstupního signálu N, které se za tu dobu objevily na vstupu. Čas měření jsme schopní měřit v rozlišení daném frekvencí interních hodin čítače. V mém případě $f_{\text{TIM}} = 156$ MHz, což znamená že dobu měření stovujeme v krocích asi 6.4 ns. Výslednou frekvenci vstupního signálu f_{in}, pak získáme tímto přepočtem:

$$f_{\text{in}} = \frac{N}{T_{\text{meas}}} \tag{3.13}$$

Kapitola 4

Ověření funkčnosti a možnosti využití přístroje

Vzniklý FW měl fungovat s PC aplikací, která se již v používá společně STM32F042 v hodinách laboratoří průmyslové elektroniky(LPE), které se vyučují na ČVUT FEL a bylo tedy zapotřebí ověřit funkčnost všech implementovaných modulů. Funčnost nového firmware byla testována studenty již v průběhu letního semestru 2022/23, kdy tato práce vznikala. Současně funkci všech částí FW demonstruji v této kapitole prostřednictvím reálných záznamů běhu PC aplikace spolupracující s mým FW běžícím na mcu STM32G431

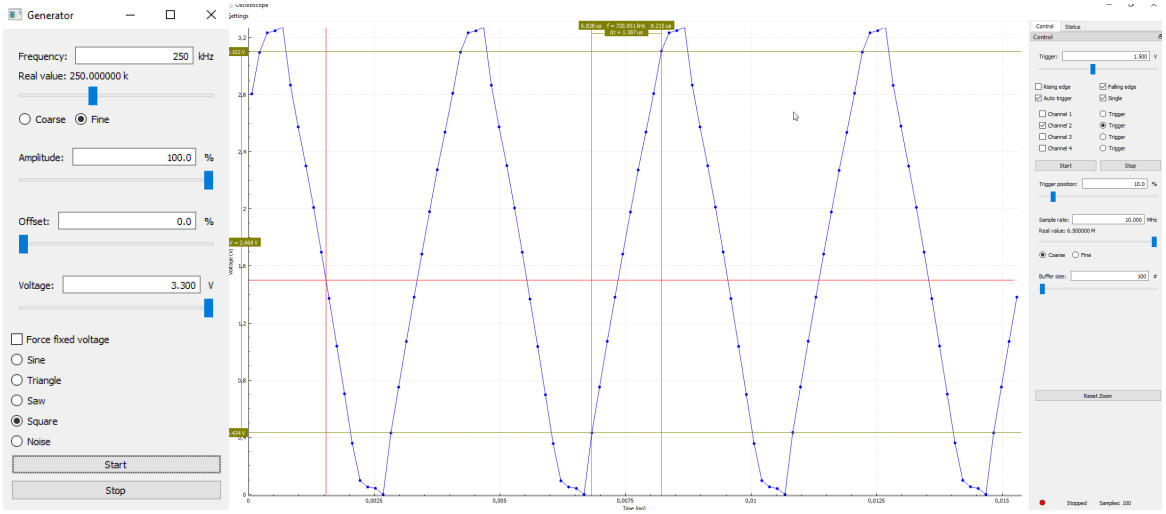
4.1 Signálový Generátor

4.1.1 Rychlost přeběhu výstupního zesilovače DAC převodníku

Maximální použitelná výstupní frekvence generátoru souvisí s omezenou rychlostí přeběhu(SR) výstupu DAC převodníku. Na obrázku 4.1, je vidět záznam měření, kdy jsem generoval obdélníkový signál s frekvencí 250KHz- tedy pouze 4 vzoky na periodu a byl zvolen v plný rozkmit výstupního signálu. Z obrázku je zřejmá omezená rychlost přeběhu, kterou můžeme rovnou změřit pomocí v aplikaci integrovaných kurzorů a dopočítat dle:

$$\mathrm{SR} = \frac{\Delta V}{\Delta t} = \frac{2.668[\mathrm{V}]}{1.387[\mu\mathrm{s}]} = 1.92[\mathrm{V}/\mu\mathrm{s}] \tag{4.1}$$

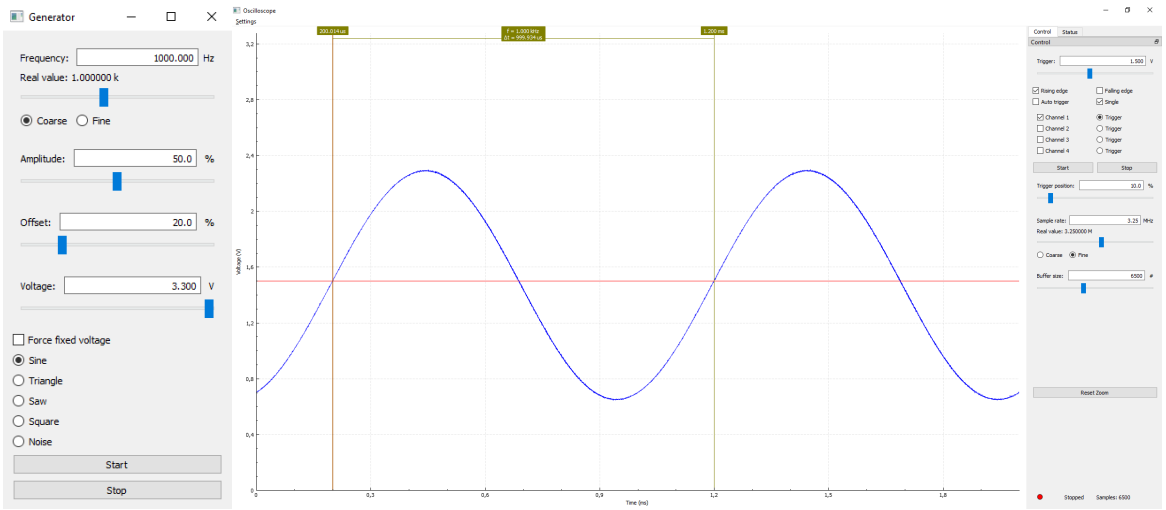
Z tohoto je zřejmé, že rychlost přeběhu výstupního zesilovače je potřeba brát v potaz při volbě výstupní frekvence generovaného signálu a pokud potřebujeme generovat signál s vysokou výstupní frekvencí je dobré zvážit nižší rozkmit, aby signál nebyl tolik zkreslený.



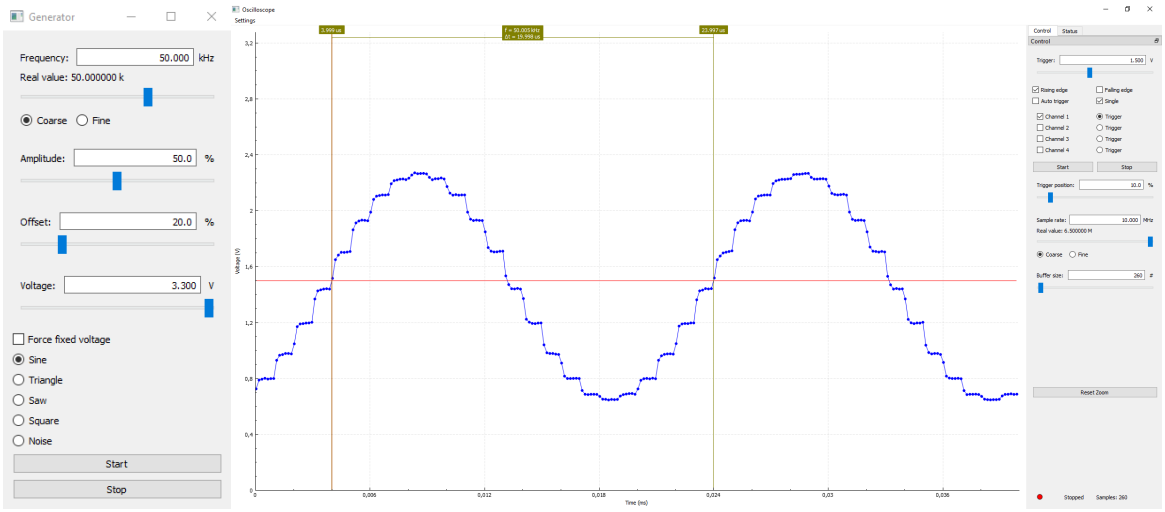
Obr. 4.1: Záznam obdélníkového signálu, f_{out}=250 kHz(4 vzorky na periodu) vzorkovaný rychlostí 6.5 MSps. Pozorování omezené rychlosti přeběhu výstupního zesilovače DAC převodníku

4.1.2 Vliv počtu vzorků an periodu

V kapitole 3.3.2 se zmiňuji o maximální vzorkovací frekvenci DAC převodníku a jeho vlivu na počet vzorků na periodu pro danou výstupní frekvenci signálu. Na obrázcích 4.2 a 4.3 je vidět rozdíl v počtu vzorků na periodu při zvýšení výstupní frekvence signálu z 1000 Hz na 50 kHz



Obr. 4.2: Záznam signálu sinus o frekvenci 1000 Hz s 1000 vzorky na 1 periodu vzorkovaný rychlostí 3.25 Msps



Obr. 4.3: Záznam signálu sinus o frekvenci 50 kHz s 20 vzorky na 1 periodu vzorkovaný rychlostí 6.5 Msps

4.2 Osciloskop

4.2.1 Ověření funkce v režimu ETS

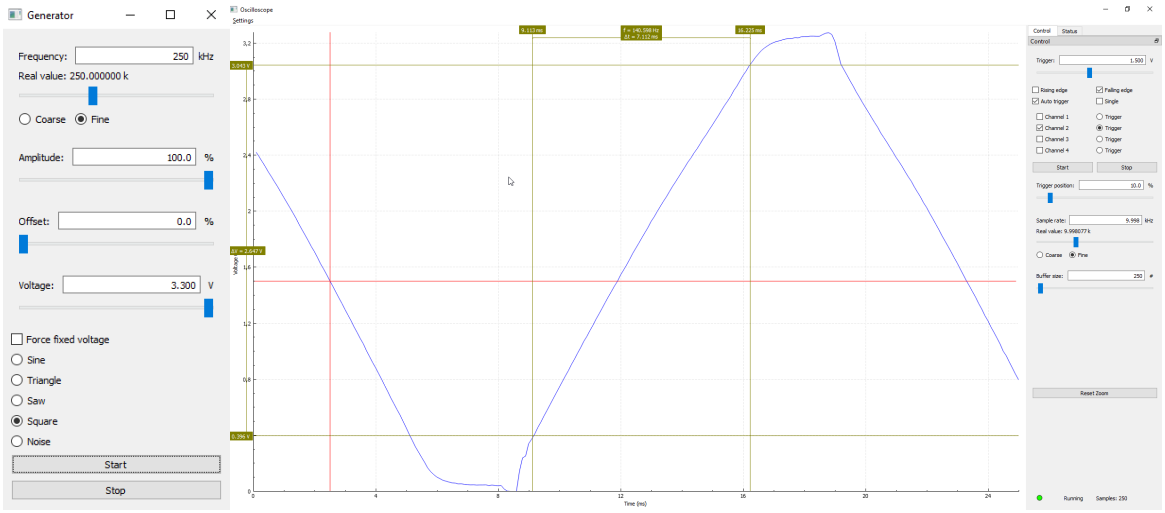
Aby bylo možné změřit přeběh použil jsem vzorkování v ekvivalentním vzorkování pro dosažení maximální vzorkovací frekvence 52 MSps. Tedy odečtený čas měření ΔT_{SAMP} je třeba nejdříve vhodně přepočíst koeficientem(k_{AEQ}), abychom získali reálnou dobu přeběhu ΔT_{SEQ}. Koeficient (k_{AEQ}) je závislý na frekvenci signálu $f_{\mathrm{SIG}} = 250$ kHz, frekvenci vzorkování $f_{\mathrm{SAMP}} = 9998.077$Hz a celočíselném koeficientu k_s.

$$k_{\mathrm{AEQ}} = \frac{f_{\mathrm{SIG}}}{f_{\mathrm{SIG}} - k_s \cdot f_{\mathrm{SIG}}}, \quad k_s \approx \frac{f_{\mathrm{SAMP}}}{f_{\mathrm{SAMP}}} \approx \frac{250\mathrm{kHz}}{9.998\mathrm{kHz}} = 25, \quad \Delta T_{\mathrm{SEQ}} = \frac{\Delta T_{\mathrm{SAMP}}}{k_{\mathrm{AEQ}}} \tag{4.2}$$

$$k_{\mathrm{AEQ}} = \frac{250\mathrm{kHz}}{250\mathrm{kHz} - 25 \cdot 9.998\mathrm{kHz}} = 5200.208, \quad \Delta T_{\mathrm{SEQ}} = \frac{7.112\mathrm{ms}}{5200.208} \tag{4.3}$$

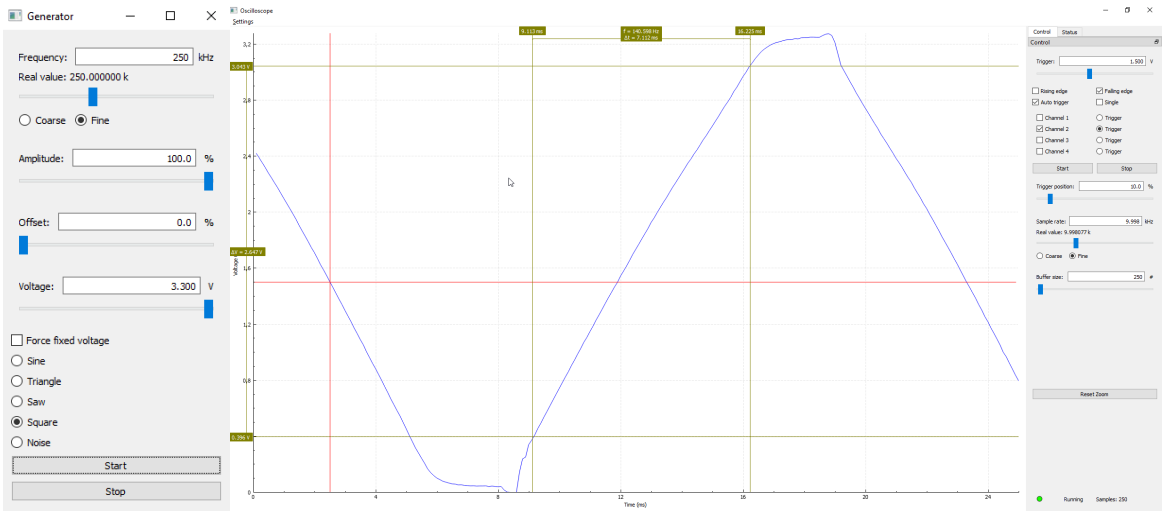
4.3 Logický analyzátor

4.4 Voltmetr se záznamem



Obr. 4.4: Záznam voltmetru. Na kanálu č.1 signál sinus o frekvenci 1 Hz, Na kanálu č.2 PWM o frekvenci 2 Hz, Kanál č. 3 uzemněný

4.5 Čítač



Obr. 4.5: Záznam signálu obdelník o frekvenci 250 kHz(4 vzorky) vzorkovaný rychlostí 52 Msps - omezeni rychlosti prebehu

4.6 Voltmetr se záznamem

4.6.1 Měření V-A charakteristiky

4.7 Aliasing

[H]

4.8 Parametry přístroje

Pro budoucí referenci v této části shrnuji v této části dosažených parametrů přístroje s STM32G431.

- **Základní vlastnosti**
  - Frekvence systémových hodin a hodin čítačů - 156 MHz
  - Frekvence hodin ADC převodníků - 52 MHz
  - Možnost funkce pouze interním oscilátorem HSI nebo volitelně možnost připojit krystal s frekvencí 8 , 12 nebo 16 MHz
- **Osciloskop**
  - 4 kanály,záznam až 1x8192 vzorků(nebo 2x 4096, 4x 2048), rozlišení 12 bitů
  - Rychlost záznamu až 1x 6.5 MS/s (nebo 2x 3.25 MS/s, 4 x 1.677 kS/s)
  - Ve stroboskopickém módu - až 52 MS/s - rozlišení s intervalem 19,2 ns.
  - Nastavení vzorkovací frekvence po 1Hz (1-9999Hz), poté s proměnným intervalem až do maximální frekvence.
  - Triggerování libovolného kanálu na náběžnou hranu, sestupnou hranu nebo obě zároveň s nastavitelnou úrovní napětí triggerování
  - Možnost funkce pretrigger zadávaného v % počtu vzorků
  - Možnost auto-trigger
- **Logický analyzátor**
  - 8 kanály,záznam až 8x8192 vzorků
  - Rychlost záznamu až 19.5 MS/s
  - Ve stroboskopickém módu - až 78 MS/s - rozlišení s intervalem 12,8 ns.
  - Nastavení vzorkovací frekvence po 1Hz (1-9999Hz), poté s proměnným intervalem až do maximální frekvence.
  - Triggerování libovolného kanálu na náběžnou hranu, sestupnou hranu nebo obě zároveň
  - Možnost funkce pretrigger zadávaného v % počtu vzorků
  - Možnost auto-trigger
- **Signálový generátor**
  - Maximální vzorkovací frekvence 1 Msps
  - Rozmezí počtu vzorků na 1 periodu - 4 až 1000 vzorků
  - Optimalizace délky bufferu podle nastavené frekvence signálu
  - Definované tvary signálu: sinus, trojuhelník, pila, obdelník
  - Nastavitelný rozkmit a offset v procentech nápájecího napětí
  - Možnost generování pseudošumu
  - Možnost generování pevného napětí
- **Impulsní generátor PWM** , nastavení střídy 0 až 100 %; nastavení frekvence 1 Hz až 78 MHz
- **Voltmetr** tři kanály, 0 až + 3,3 V, 100 odměrů/s, průměrování z 1 až 256 odměrů, možnost funkce záznamu

Kapitola 5

Závěr

Příloha A

Literatura

[1] Lpe 2023 - laboratoře průmyslové elektroniky v roce 2023. [online], 2022. [cit. 2023-05-22.

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

[3] DanielleCollins. An2548 using the stm32f0/f1/f3/cx/gx/lx series dma controller. [online], 12 2022. [cit. 2020-05-21].

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

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

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

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

[8] STMicoelectronics. Stm32g4 extended interrupts and events controller. [online]. [cit. 2020-05-18].

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

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

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

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

[13] 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

© ČVUT v Praze, Design: ČVUT v Praze, VIC