

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



## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **David**

Jméno: **Petr**

Osobní číslo: **420110**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávací 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

<b>1 Úvod</b>	<b>1</b>
<b>2 Rozbor</b>	<b>3</b>
2.1 PC aplikace Zero Elab Viewer .....	3
2.2 Použitý MCU STM32G431 .....	4
2.3 .....	4
<b>3 Realizace FW</b>	<b>5</b>
3.1 Struktura FW .....	5
3.2 Rozpoznání frekvence externího krystalu HSE .....	7
3.3 Realizace měření frekvence a střídy .....	11
3.4 Realizace generátoru funkcí .....	14
3.5 Realizace osciloskopu .....	23

3.6 Logický analyzátor .....	31
<b>4 Ověření funkčnosti</b>	<b>33</b>
<b>5 Možnosti využití přístroje</b>	<b>35</b>
5.1 Měření V-A charakteristiky .....	35
5.2 Aliasing .....	35
<b>6 Zhodnocení</b>	<b>37</b>
<b>7 Závěr</b>	<b>39</b>
<b>A Literatura</b>	<b>41</b>
<b>B Zadání práce</b>	<b>43</b>



## Obrázky

2.1 STM32G031 zapojen na nepájivém poli převzato z [3] .....	3
3.1 Kategorie zdrojového kodu a jejich posloupnost závislosti .....	6
3.2 Hlavní okno aplikace Zero eLab Viewer zobrazující aktivní moduly dané konfigurace .....	6
3.3 Zkreslení průběhu měřeného signálu v důsledku nestability HSI převzato z [2] .....	8
3.4 Vstupy čítače TIM16 .....	9
3.5 Měření periody vstupního signálu pomocí "Input-capture"a DMA .....	10
3.6 Vyobrazení interního PLL bloku. Převzato z.....	10
3.7 Postup změny zdroje hodinového signálu.....	11
3.8 PWM input režim .....	12
3.9 Diagram propojení čítačů pro upravenou metodu měření frekvence čítáním pulzů. Převzato z [2]	13
3.10 Časový diagram čítání pulzů za proměnlivou dobu odběru .....	14

3.11 Popis funkce kruhového bufferu jako zdroj datových vzorků pro DAC převodník .....	15
3.12 Diagram využití DAC převodníku a dalších periférií pro generování analogového signálu ...	16
3.13 Možnosti nastavení vlastností generovaného signálu v prostředí aplikace Zero Elab Viewer .	16
3.14 Výpočet sinus a cosinu uhlu postupnou aproximací použitou v CORDIC převzato z [7] ....	19
3.15 Naznačení průběhu výpočtu hodnot LFSR registru převzato z [6] .....	21
3.16 Využití LFSR registru pro generování šumu .....	21
3.17 Záznam generovaného signálu šumu se vzorkovací frekvencí 1 Msps .....	22
3.18 Záznam generovaného signálu šumu se vzorkovací frekvencí 250 Ksps .....	23
3.19 Princip funkce Analog Watchdog(AWD) ADC převodníku .....	24
3.20 2-fázové triggerování na nástupnou hranu signálu s využitím AWDG- převzato z [2] .....	24
3.21 Vzájemné propojení dvou čítačů .....	25
3.22 HW konfigurace 1 ADC převodníku pro režim s multiplexováním vstupních kanálů .....	26
3.23 HW konfigurace 2 ADC převodníků pro režim "Independent interleaved" .....	27
3.24 Vzorkování v režimu "Independent Interleaved" .....	27
3.25 HW konfigurace 2 ADC převodníků pro režim "Dual simultaneous" .....	28
3.26 Vzorkování 4 kanálů v režimu "Dual simultaneous" .....	29
3.27 Zpoždění kanálu 2 a 4 za kanálem 1 a 3 při použití Dual simultaneous modu a stroboskopickém vzorkování s ekvivalentní vzorkovací frekvencí 78MHz .....	30
3.28 převzato z [9] .....	30

3.29 Efekt nedodržení maximálního vstupního odporu .....	31
3.30 HW konfigurace pro logický analyzátor .....	31
5.1 .....	36







## Tabulky

3.1 Srovnání paměťové náročnosti jednoduchého programu s využitím různých knihoven.....	7
3.2 .....	8
3.3 Maximálních dosažené odchylky frekvencí výstupních signálů pro různá rozmezí frekvencí ..	18
3.4 Doba trvání přípravy bufferu o délce 1000 vzorků v závislosti na použité metodě .....	20
3.5 Přehled jednotlivých analogových kanálů a jejich na dostupnosti na ADC převodnících .....	26





# Kapitola 1

## Úvod

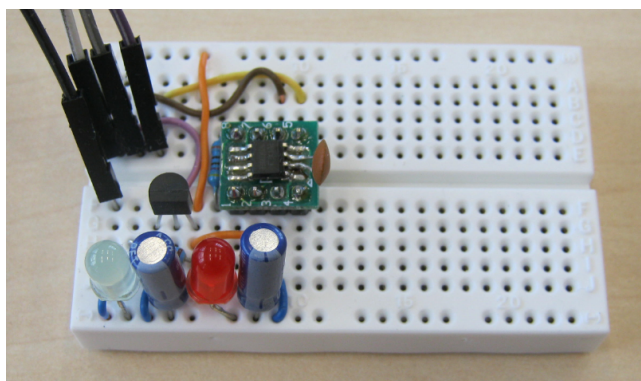


## Kapitola 2

### 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 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 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 přístroje 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 periférií vhodným pro implementaci SDI, jako jsou 2 ADC převodníky či větší počet periférií čítačů(14 vs 8 u G030).

## ■ 2.3

## 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. V zájmu zpětné kompatibility 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 přístroje lze implementovat a jaké budou mít možnosti ovládání či nastavení. Dále jsem se při návrhu zaměřil na tyto body

- **Samostatná využitelnost jednotlivých přístrojových bloků**

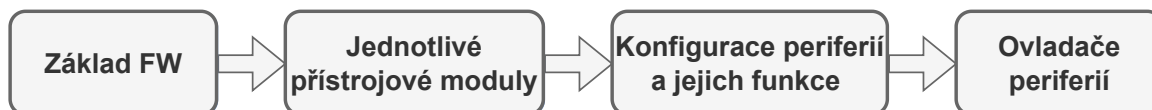
Rozdělit implementaci jednotlivých přístrojů do samostatně fungujících bloků je výhodné z 2 důvodů. Zaprvé lze tak jednodušeji výsledný FW přizpůsobit pro různé mikrokontroléry v závislosti na dostupných perifériích a velikosti FLASH paměti. Druhým důvodem je pak možnost použití kódu i v jiných aplikacích.

- **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**

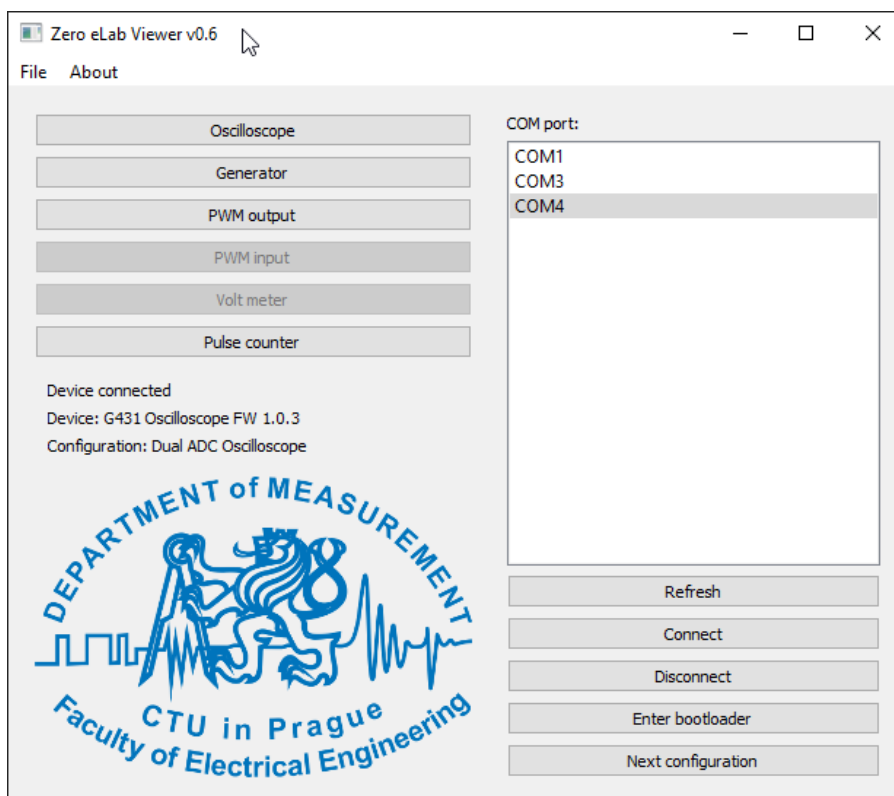
Nástroj STM32CubeMX velmi usnadňuje začátek vytváření FW díky využití grafického rozhraní pro definování počáteční konfigurace MCU. Pokud se zachová struktura generovaného kódu, lze pak nástroj opětovně využít v případě, že chceme jednoduše upravit HW konfiguraci.



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 periférii. 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-



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

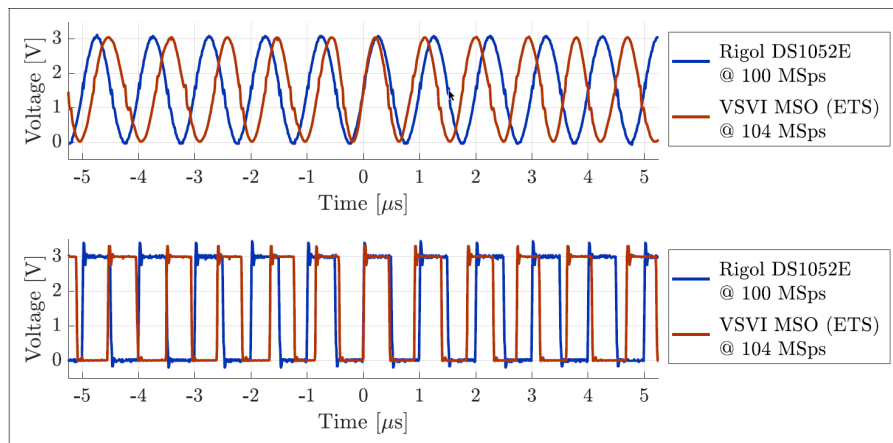
Pro vytvoření představy o rozdílnosti paměťové náročnosti jednotlivých knihoven jsem vytvořil pomocí STM32CubeMX dva minimální projekty, jejichž úkolem bylo nastavení systémových hodin blikání jednou LED. V prvním případě byly periférie RCC(Reset and clock control) a GPIO inicializovány prostřednictvím HAL knihovny a v druhém případě pomocí LL ovladačů a byly použity odpovídající zpoždovací 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 ovladače	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 SYSCCLK. 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  $\pm 0.005\%$  [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]

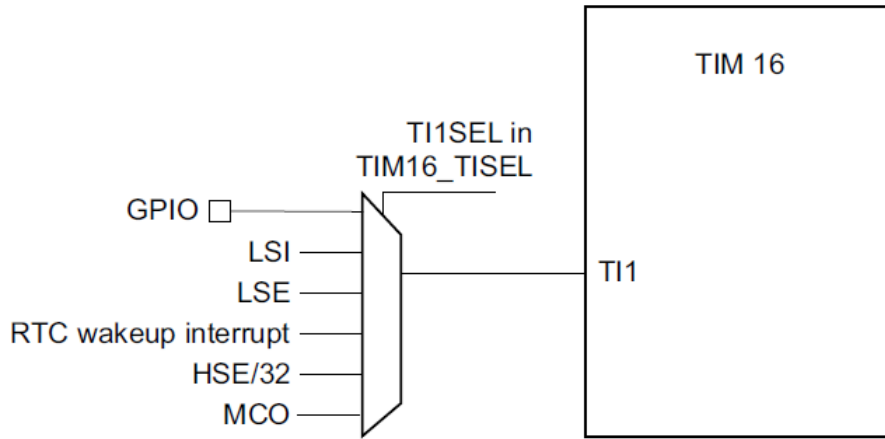
Jako zdroj vysokorychlostního externě získaného hodinového signálu(HSE) lze buď využít krystalu buzeného pomocí MCU(HSE crystal) nebo jiného externího zdroje signálu(HSE bypass) splňující podmínky na maximální frekvenci a střidu daného MCU. Například: pro STM32G431 musí být externí signál v rozsahu 4-48MHz a mít střidu 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 SYSCCLK nebyla na použitém krystalu závislá.

	Absolutní odchylka frekvence	Odchylka frekvence způsobená změnou teploty
STM32G431 HSI 16 MHz	$\pm 1\%$	$\pm 1\%$ ( 0 až $+85^{\circ}\text{C}$ )
Adafruit krystal 16 MHz	$\pm 0.003\%$	$\pm 0.005\%$ ( $-20$ až $+70^{\circ}\text{C}$ )

**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[6]

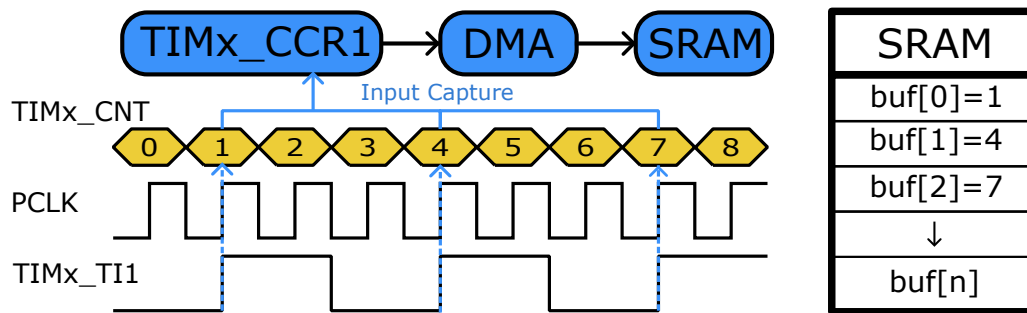
Standardní způsob měření periody vstupního signálu probíhá potom tak, že měříme počet uběhlých cyklů čítače mezi jednotlivými náběžnými hranami nebo sestupnými hranami jako je vyobrazeno na obrázku 3.5. Tento počet cyklů nám pak určuje poměr mezi frekvencí externího hodinového signálu na vstupu  $f_{IN}$  a hodinového signálu, který pro svůj chod využívá periferie čítače  $f_{TIM}$ . Druhou zmíněnou frekvencí ( $f_{TIM}$ ) 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  $f_{TIM}$

$$f_{IN} = \frac{f_{TIM}}{N_{period}} = \frac{f_{TIM}}{buf[n] - buf[n-1]} = \frac{f_{TIM}}{3} \quad (3.1)$$

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

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

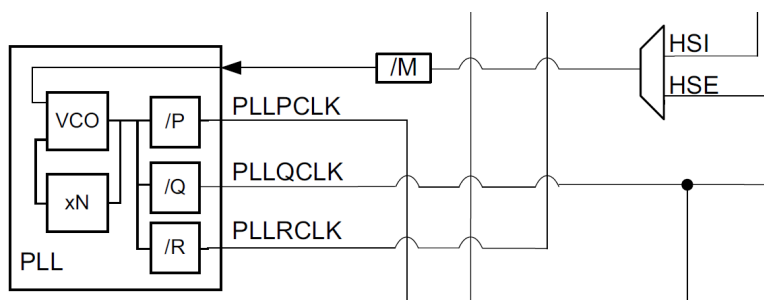
Obdobně pro krystal s frekvencí  $f_{HSE\_16} = 16\text{ MHz}$  bychom změřili, že mezi jednotlivými náběžnými hranami vstupního signálu uběhne  $N_{HSE\_16} = 312$  period vnitřních hodin čítače. Tedy vysoká frekvence vnitřních hodin čítače  $f_{TIM} = 156\text{ MHz}$  a možnost dělení signálu HSE před 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 periférii 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[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.

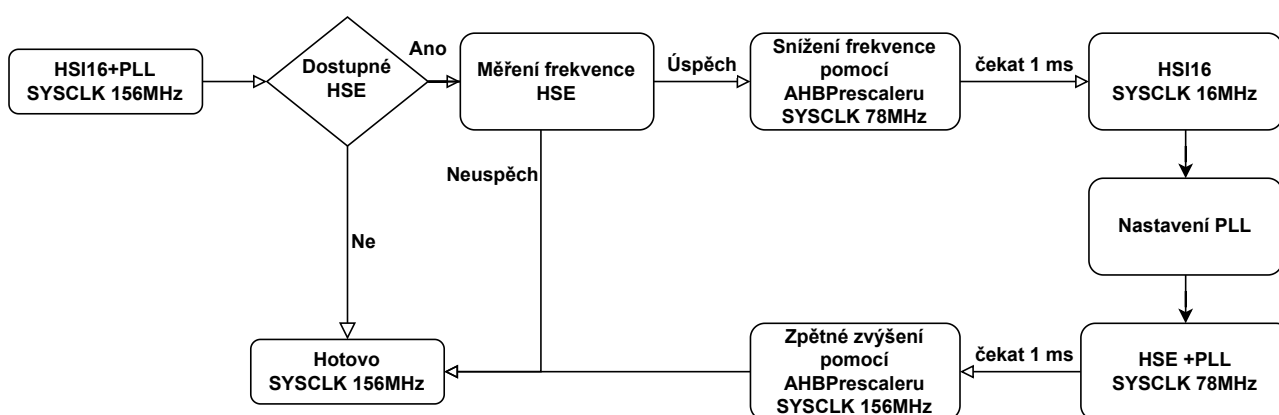


Obr. 3.6: Vyobrazení interního PLL bloku. Převzato z

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 popsán 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( $\text{SYSCLK} > 80\text{MHz}$ ) 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  $1\mu\text{s}$ . V mém řešení program čeká 1ms s využitím připravených funkcí obsahujícím čekání v jednotkách ms.



Obr. 3.7: Postup změny zdroje hodinového signálu

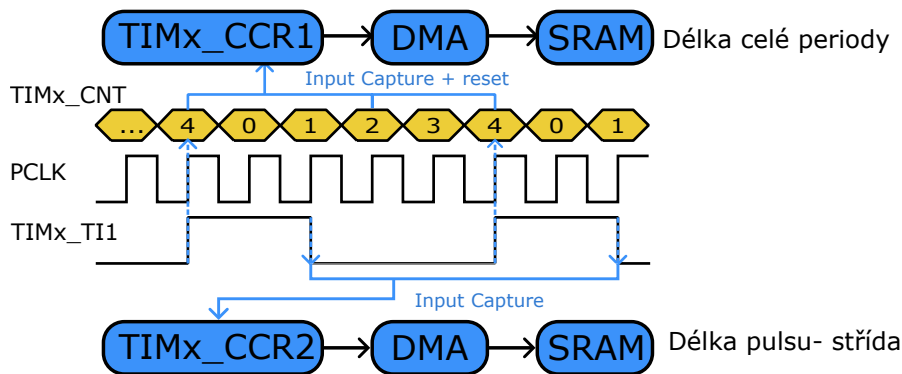
## 3.3 Realizace měření frekvence a stříd

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 periférií 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 stříd 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.

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

### 3.3.1 Metoda měření 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 popsána v [6] 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 3.8.



Obr. 3.8: PWM input režim

Ve své implementaci jsem přejal řešení popsané v práci [1]. Toto řešení adaptivně upravuje frekvenci vstupních hodin čítače pomocí nastavování předděličky 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.

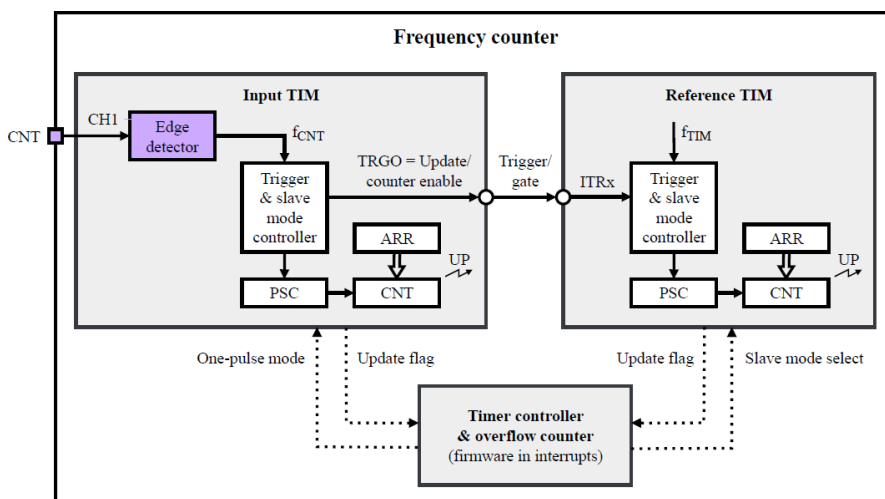
Dále jsem využil DMA převodníku pro zaznamenání 10 po sobě jdoucích period vstupního signálu, abych snížil nepřesnosti měření.

### 3.3.2 Metoda čítání pulsů

Pro měření signálů s vysokými frekvencemi se více hodí alternativní metoda měření, kdy neměříme dobu trvání periody, ale počítáme počet period signálu za nějaký námi stanovený časový úsek  $t_{gate}$ . Nevýhodou této metody bývá, že 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 ve práci [2] navrhl řešení, které se snaží toto omezení řešit.

Řešení spočívá v propojení dvou čítačů. Jako u obvyklého řešení první z čítačů počítá vstupní

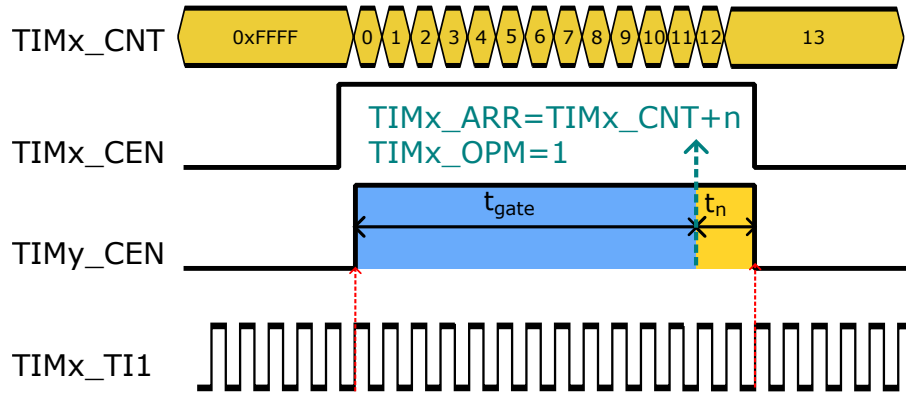
pulzy (Vstupní čítač) a druhý čítač odměřuje dobu která uběhla pomocí interních vstupních hodin čítače PCLK (Referenční čítač). Řídící jednotky čítačů jsou vzájemně propojené tak, aby po spuštění měření první náběžná hrana na vstupu spustila běh referenčního čítače a po uplynutí stanovené doby měření hlídané referenčním čítačem ještě referenční čítač odměřil dobu trvání celých period vstupního signálu a zastavily se oba čítače zároveň. Schématické propojení čítačů je vyobrazeno na obrázku 3.9.



**Obr. 3.9:** Diagram propojení čítačů pro upravenou metodu měření frekvence čítáním pulzů. Převzato z [2]

Časový diagram měření je znázorněn na obrázku 3.10 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. 3.10: Časový diagram čítání pulzů za proměnlivou dobu odběru

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

$$f_{\text{in}} = \frac{N}{T_{\text{meas}}} \quad (3.3)$$

## 3.4 Realizace generátoru funkcí

Součástí zadání byla implementace signálového generátoru. V práci bylo využito vnitřního DAC převodníku 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 pak 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 napětí což dále rozšiřuje repertoár laboratorních úloh.

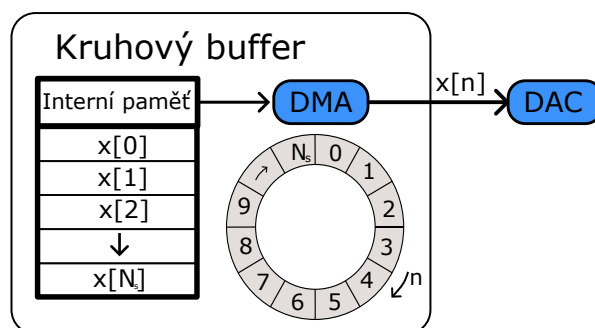
### 3.4.1 Použitý DAC převodník a jeho limity

Přítomný interní DAC převodník nabízí dle [8] 2 výstupní kanály a maximální vzorkovací frekvenci 1 Msps. Rozlišení DAC převodníku je 12 bitů a také umožňuje generování některých signálových průběhů jako je trojúhelníkový signál, signál typu "pila" či šumový signál pomocí nastavení vnitřních registrů čítače. DAC převodník, také disponuje vestavěným výstupním bufferem, který slouží pro snížení výstupní impedance.



### 3.4.2 Generování signálu

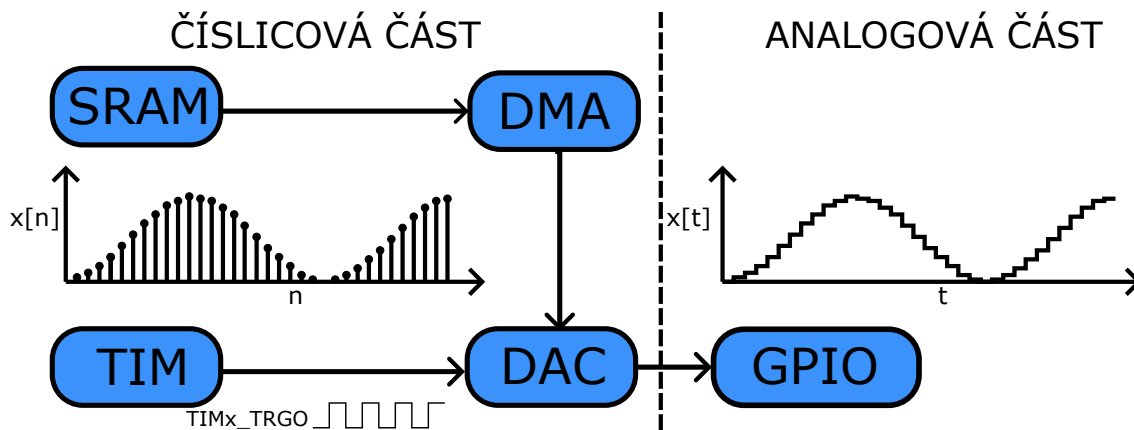
V předchozí části jsem zmiňoval možnost generování signálových průběhů jako pilový nebo trojúhelníkový signál pomocí nastavení interních registrů DAC převodníku. Tato metoda spočívá v automatické výpočtu 12 bitových hodnot pro výstup DAC převodníku prostřednictvím HW registrů. Typicky se k nastavené hodnotě (offsetu) pravidelně přičítá hodnota jiného registru s tím, že si DAC převodník sám řídí generování nových hodnot vzorků. Nevýhodou tohoto přístupu jsou především omezené možnosti ohledně změny tvaru signálu, kdy nám například chybí možnost generování sinusového signálu. Výhodou tohoto přístupu je minimální paměťové nároky, protože číselné hodnoty jednotlivých vzorků jsou vypočtené za běhu.



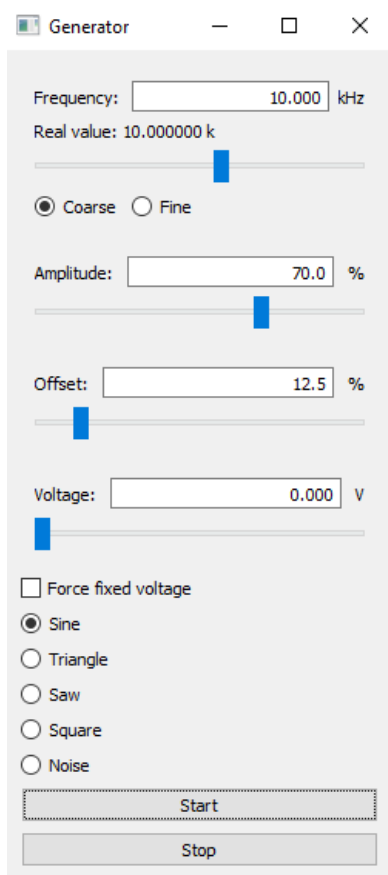
**Obr. 3.11:** Popis funkce kruhového bufferu jako zdroj datových vzorků pro DAC převodník

Alternativou ke generování signálových průběhů s využitím interních registrů je napočítat číselné hodnoty v rozlišení DAC převodníku pro jednotlivé vzorky do paměti mikrokontroléru a následně prostřednictvím DMA řadiče jednotlivé vzorky posílat do DAC převodníku. 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 1KHz.



Obr. 3.12: Diagram využití DAC převodníku a dalších periférií pro generování analogového signálu



Obr. 3.13: 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 periody 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 číslcovou hodnotu kterou dokáže využít bere hodnotu 0xFFFF. 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:

$$O(g) = \langle 0, \text{rozkm} \times 0xFFFF \rangle \quad (3.4)$$

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

$$f(n) = \min((\text{offset} \times 0xFFFF) + g(n), 0xFFFF) \quad (3.5)$$

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

## ■ Optimalizace délky bufferu

Obvyklý postup při generaci signálu s využitím 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 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 dělicím poměrem DIV čítače pro generování zmíněné časové základny. Závislost tohoto dělicího poměru na frekvenci vstupních hodin toho čítače  $f_{PCLK}$ , výstupní frekvenci signálu a počet vzorků  $N_s$  je vyjádřena vzorcem:

$$DIV = \frac{f_{PCLK}}{f_{out}} \quad (3.6)$$

Nicméně zde narážíme na omezené možnosti nastavení výstupní frekvence signálu v podobě dělicího poměru DIV, který může nabývat pouze celočíselných hodnot. Pro zlepšení rozmezí nastavitelných frekvencí výstupního signálu jsem se inspiroval článkem [4], 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 zkracování délky bufferu a minimalizování absolutní odchylky frekvence generovaného signálu.

V rovnicích 3.7 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 dělicího poměru DIV.

$$f_{out} = \frac{f_s}{N_s}, \quad f_s = \frac{f_{PCLK}}{DIV} \quad \implies \quad f_{out} = \frac{f_{PCLK}}{DIV \cdot N_s}, \quad f_s = \frac{f_{out}}{DIV \cdot N_s} \quad (3.7)$$

Dělicí 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 3.8 Při optimalizaci délky bufferu tedy v prvním kroku nehledáme jen dělicí poměr DIV, ale faktorizaci tohoto dělicího 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].

$$DIV = (ARR + 1) \cdot (PSC + 1) \quad (3.8)$$

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 rovnou požadované a druhou faktorizací frekvenci vyšší nebo rovnou. 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 0,01Hz 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 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 dělicího 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 ?? 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 označují reálné výstupní frekvence.

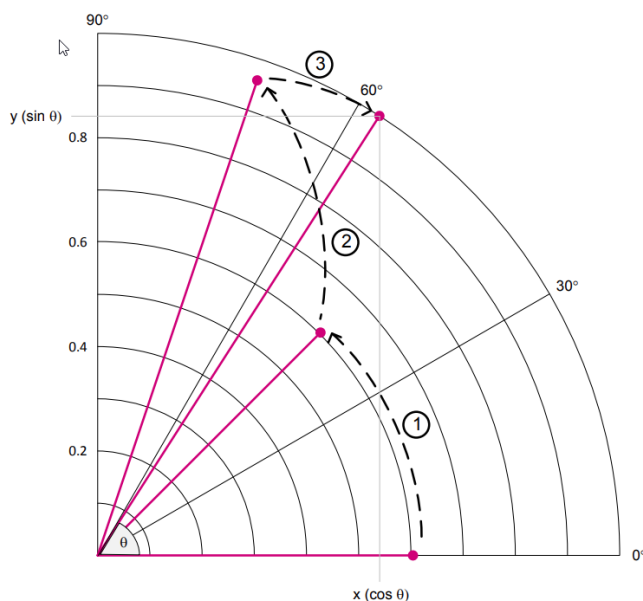
Rozsah frekvencí	Maximální absolutní odchylka			Maximální relativní odchylka		
	$f_{in}$ [Hz]	$f_{out}$ [Hz]	Odchylka [Hz]	$f_{in}$ [Hz]	$f_{out}$ [Hz]	Odchylka [%]
1-500 Hz	477	477.0058	0,0058	314	314.0046	0,0015
500-1000 Hz	977	977.0212	0,0212	977	977.0212	0,0022
1000 - 2500 Hz	2361	2360.811	0,1894	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 dosažených odchylek frekvencí výstupních signálů pro různá rozmezí frekvencí

## Využití CORDIC

Jak již bylo zmíněno číselné hodnoty jedné periody průběhu se ukládají do paměti SRAM. V případě sinusového signálu je potřeba napočítat hodnoty funkce sinus v rozsahu  $2\pi$  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.[10]. Pro získání výsledku používá metody postupné aproximace a případě trigonometrických funkcí algoritmus konverguje pro hodnoty  $-\pi$  až  $\pi$  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 3.14.



**Obr. 3.14:** Výpočet sinus a cosinu uhlu postupnou aproximací použitou v CORDIC převzato z [7]

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.

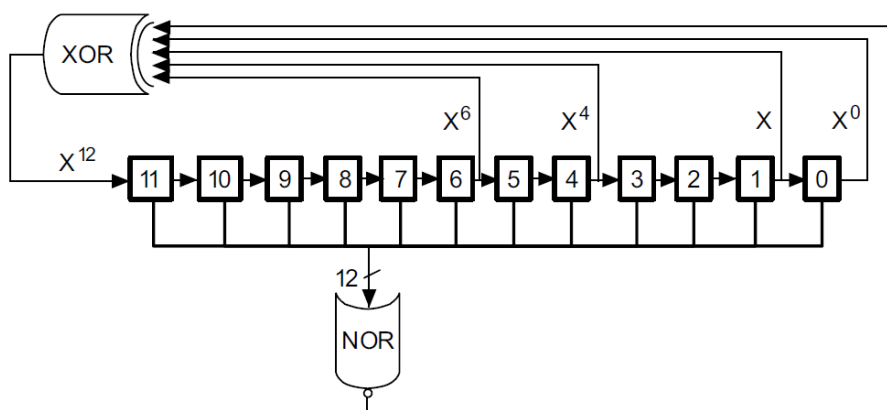
Použitá metoda	Doba plnění bufferu	Čas při SYSCLK = 156MHz
CORDIC	37541 cyklů	0.241 ms
sinf(x) z math.h	85573 cyklů	0.549 ms

**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.

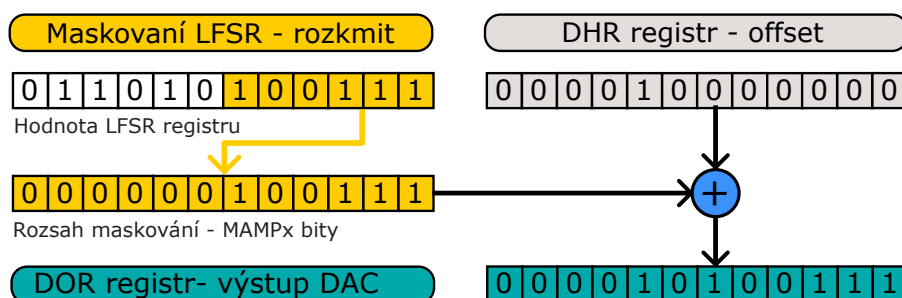
### 3.4.3 Generování šumu

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 přidáváním 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 3.15.



**Obr. 3.15:** Naznačení průběhu výpočtu hodnot LFSR registru převzato z [6]

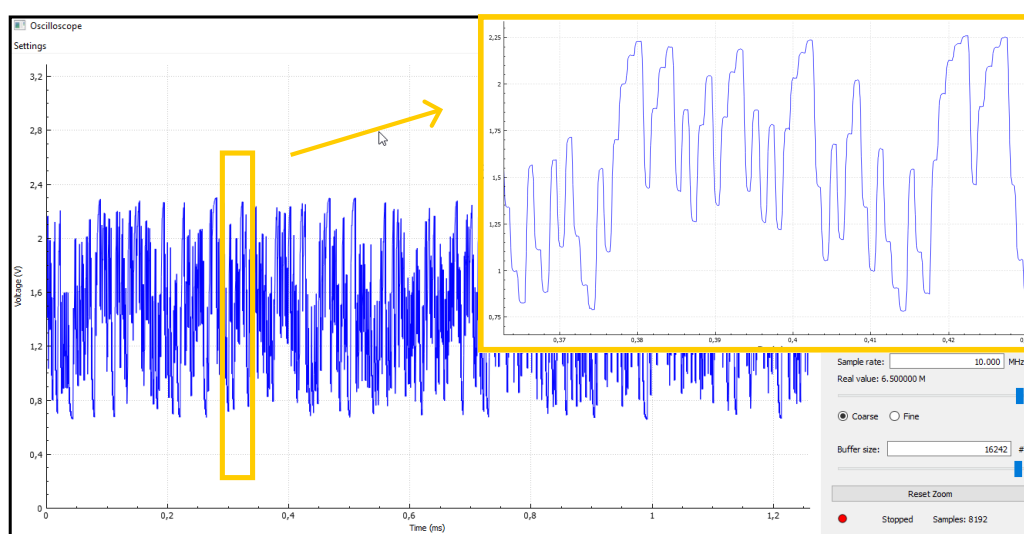
Samotné využití LFSR v DAC převodníku spočívá maskováním bitů tohoto registru pro získání určitého rozmezí(rozkmitu) výstupních hodnot a následným přičtením této hodnoty získané maskováním do DHR(Data-Hold) registru. Hodnota registru DHR určuje nejmenší hodnotu posílanou na výstupu DAC převodníku - tedy 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 3.16. 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. 3.16:** Využití LFSR registru pro generování šumu

Z obrázku 3.16 vyplývá jakým způsobem lze generování šumového signálu nastavovat podobně jako u ostatních typů signálů pouze s drobnými rozdíly. Například hodnota nastavení posuvu generovaného

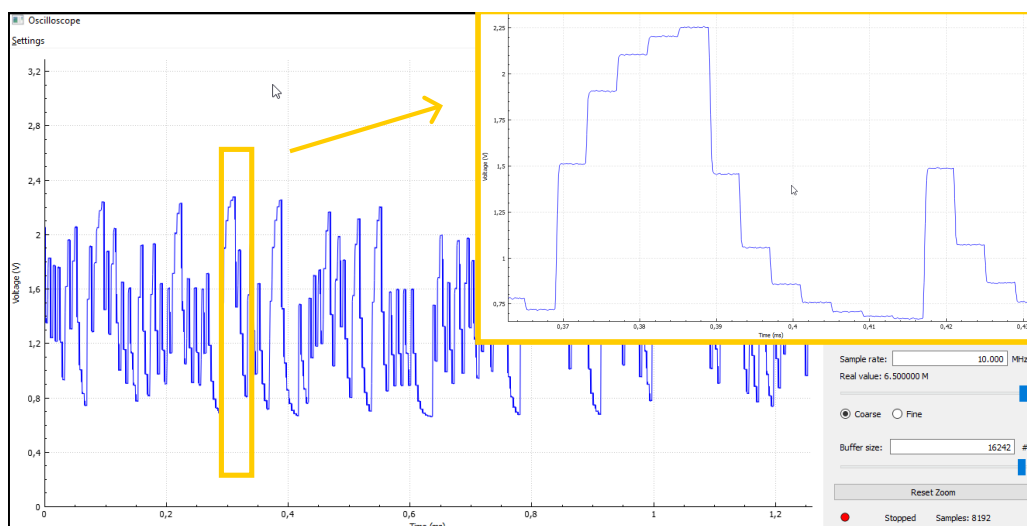
signálu(offset) se dá nastavit v celém rozsahu DAC převodníku. Naopak menší počet různých hodnot lze nastavit pro rozkmit generovaného signálu. Ze strany počítačové aplikace se sice zdá, že ho lze nastavovat v desetinách procenta nicméně tím, že jediný možný způsob přepočtu LFSR registru je maskováním 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. 3.17:** 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 3.17 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 3.18.





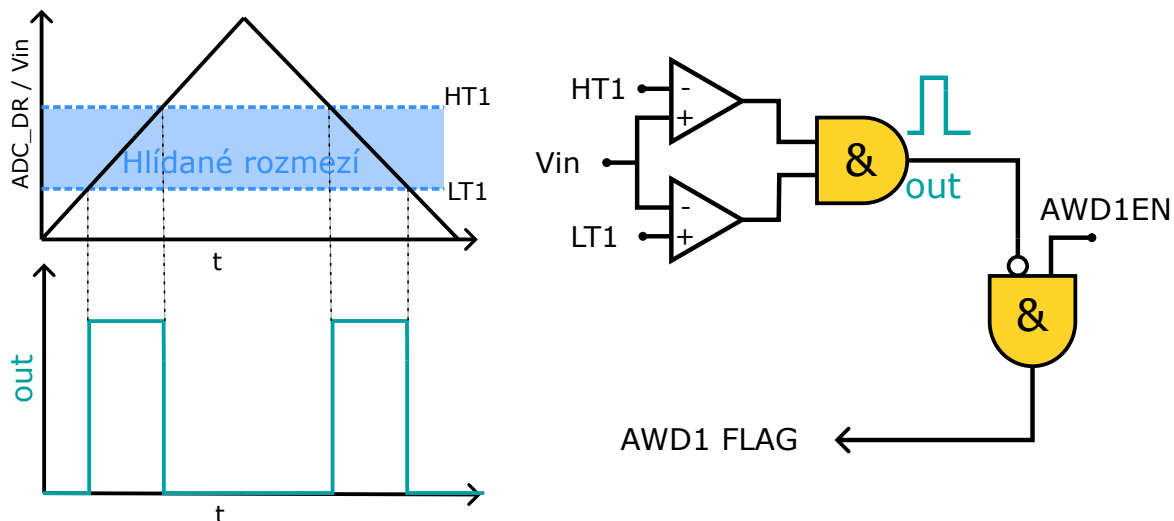
**Obr. 3.18:** Záznam generovaného signálu šumu se vzorkovací frekvencí 250 Ksps

## 3.5 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 převodníků, triggerování pomocí analog watchdog (AWDG) funkce přítomných ADC převodníků, DMA řadiče a 2 vzájemně propojených zřetěžených čítačů.

### 3.5.1 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 napětové 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. 3.19: Princip funkce Analog Watchdog(AWD) ADC převodníku

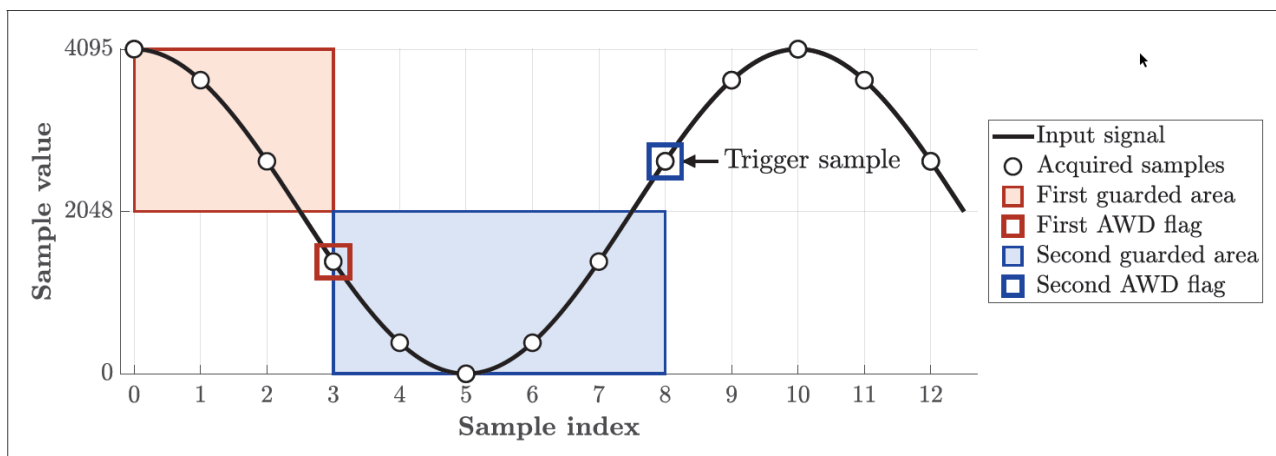
AWDG umožňuje vyvolat přerušení v momentě, kdy se danému kanálu objeví napětí mimo nastavené rozmezí. Abychom mohli určit okamžik, kdy došlo k poklesu pod určitou úroveň (sestupná hrana) nebo naopak k překročení napěťové úrovně (náběžná hrana) je potřeba využití AWDG ve dvou fázích znázorněných na obrázku 3.20.

#### ■ 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. 3.20: 2-fázové triggerování na nástupnou hranu signálu s využitím AWDG- převzato z [2]

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.

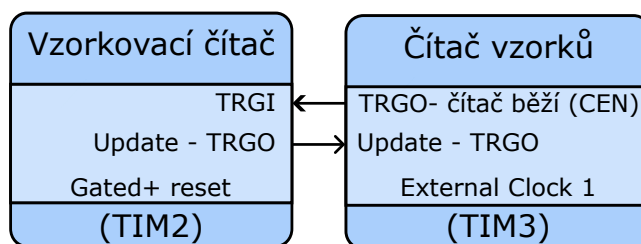
### ■ Vzorkovací čítač

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"kanálů. 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. 3.21: Vzájemné propojení dvou čítačů

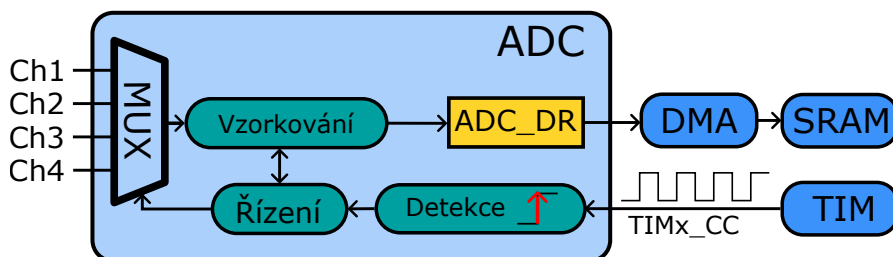
## ■ 3.5.2 Módy vzorkování

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 [2]. 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 převodnicích

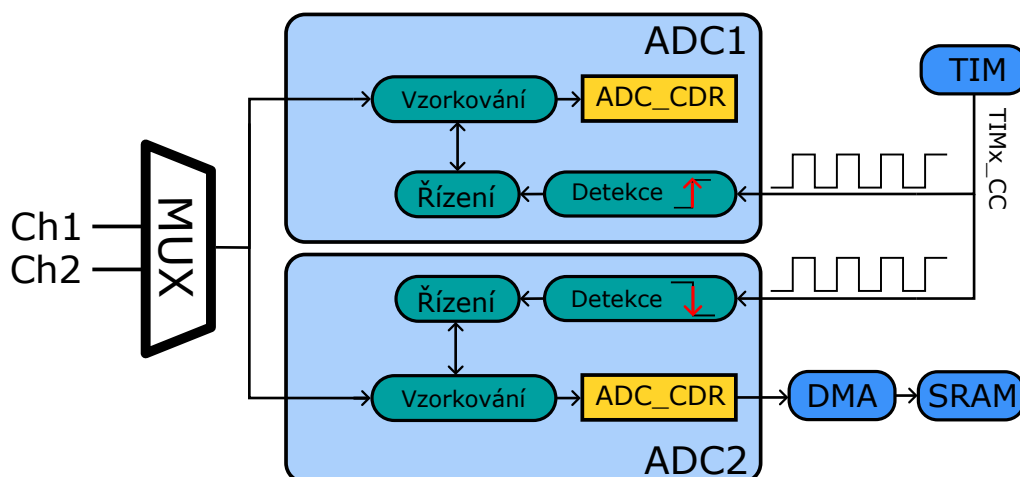
vzorkovací frekvence, respektive zvýšení doby vzorkování pro stejnou vzorkovací frekvenci oproti použití 1 ADC. Výběr použitého módu vzorkování pak záleží na zvolených kanálech, respektive na jejich počtu a na jejich přítomnosti na jednotlivých ADC převodnicích. Například kanál osciloskopu číslo 1 je 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. 3.22:** HW konfigurace 1 ADC převodníku pro režim s multiplexováním vstupních kanálů

### Vzorkování 1 kanálu

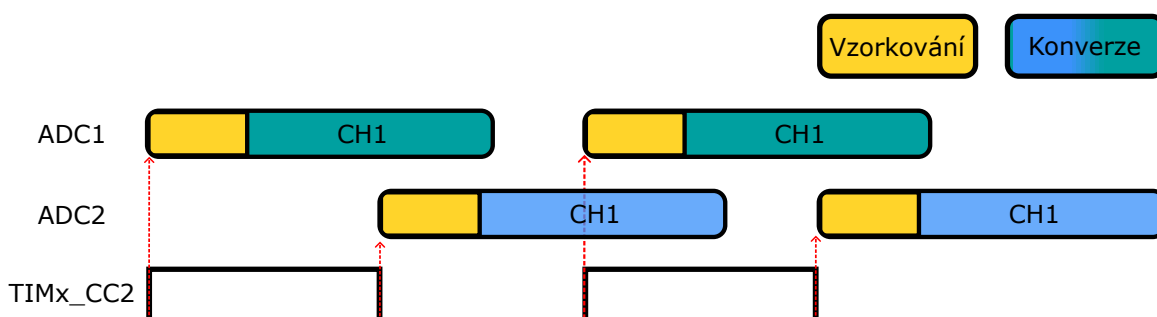
Pokud je zvolený analogový kanál osciloskopu využitím vnitřních multiplexerů je možné přivést zároveň (paralelně) na vstupy obou ADC převodníků, můžeme využít tzv. "Dual -interleaved modu". Tento režim spočívá ve střídavém vzorkování jedním převodníkem a v průběhu konverze odebraného vzorku vzorkováním 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 3.5.3.



Obr. 3.23: 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 3.24

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.



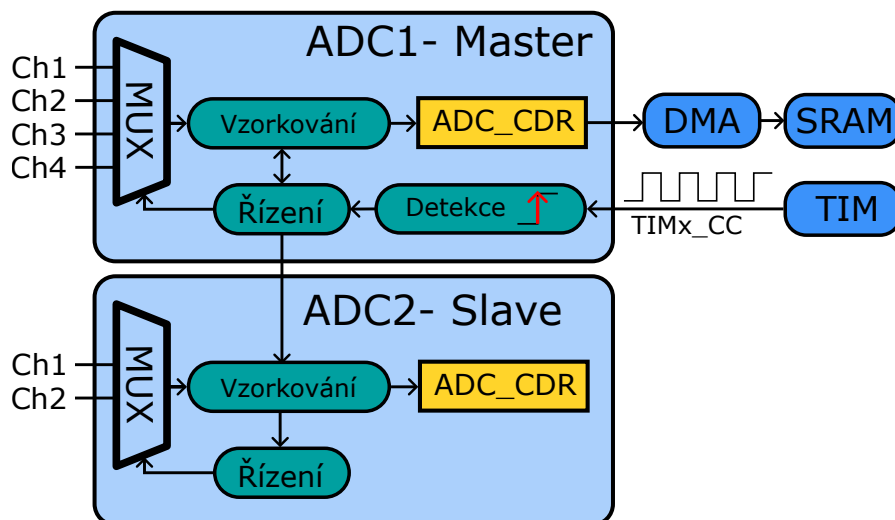
Obr. 3.24: Vzorkování v režimu "Independent Interleaved"

## Vzorkování 2 a více kanálů

Pokud je to možné v případě zvolení 2 a více kanálů, je výhodné použít Dual Simultaneous 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 měl FW implementoval automatické přiřazování kanálů jednotlivým převodníkům, podle jejich dostupnosti. Přiřazování probíhá tak, že nejdříve se přiřadí kanály, které jsou dostupné pouze na jednom z převodníků a pak se posoupnosti 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.

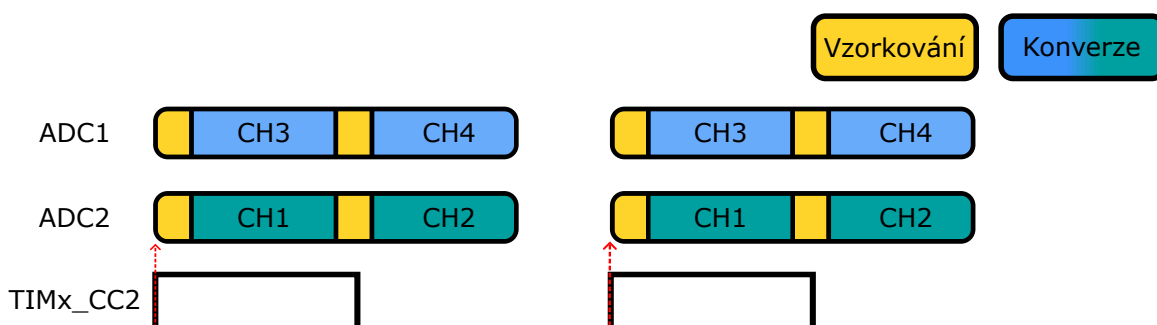


Obr. 3.25: HW konfigurace 2 ADC převodníků pro režim "Dual simultaneous"

Samotné používání 2 ADC převodníků využívá pro STM32 ADC nativního DUAL simultaneous režimu řízení vzorkování. V tomto režimu spouštíme vzorkování pouze řídicího(MASTER) ADC převodníku, který sám řídí podřazený ADC převodník. HW konfigurace včetně zbylých periférií je

ukázána na obrázku 3.25. V tomto režimu se také podobně jako je tomu u interleaved režimů používá společného data registru (CDR), díky čemuž můžeme naměřená data z obou převodníků 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 3.26. 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.

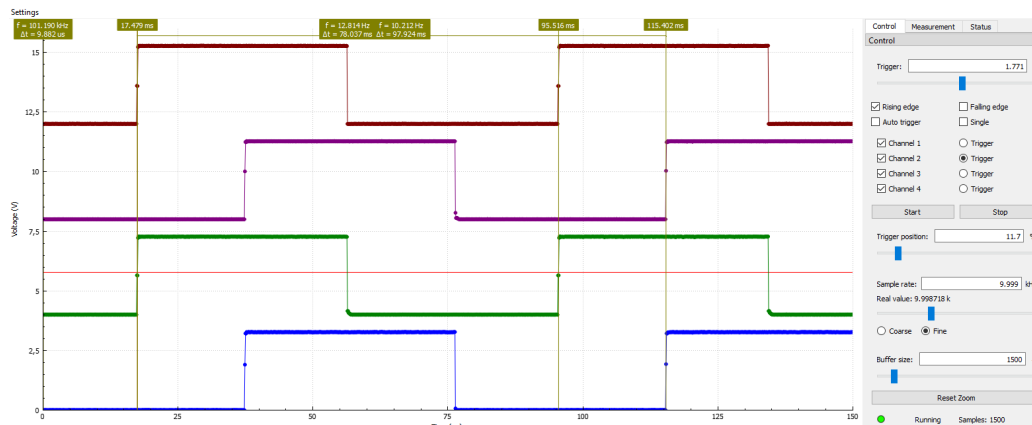


Obr. 3.26: Vzorkování 4 kanálů v režimu "Dual simultaneous"

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 3.27 je vidět příklad vzorkování s ekvivalentní vzorkovací frekvencí 78 MHz. Zpoždění mezi kanály je v tomto případě rovno  $N_{\text{cyklů}} = 653$  cyklů ADC převodníku, což při frekvenci hodin ADC převodníku  $f_{\text{ADC}} = 52 \text{ MHz}$  odpovídá zpoždění  $T_{\text{Delay}} = 12.55 \mu\text{s}$

$$T_{\text{Delay}} = \frac{N_{\text{cyklů}}}{52 \text{ MHz}} = 12.5 \mu\text{s} \quad (3.9)$$

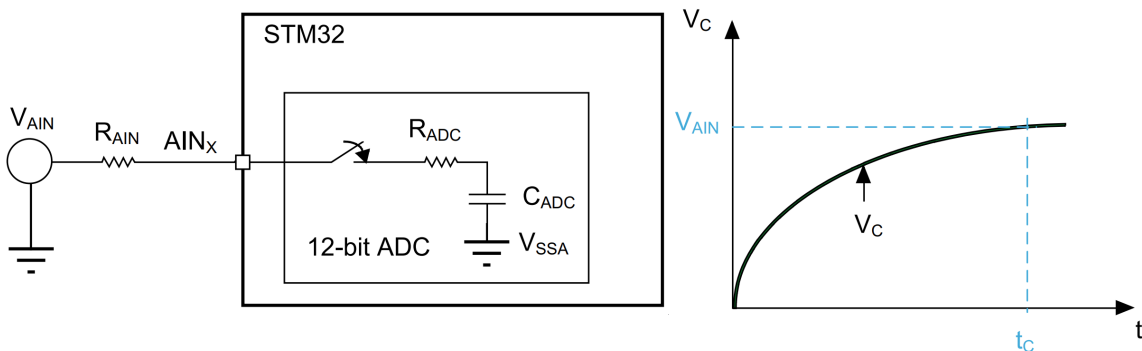
Na vstup byl při tom přiveden obdelníkový signál s frekvencí 10000 KHz, jehož perioda  $T=10 \mu\text{s}$ . Tedy zpoždění signálu je o více než jednu periodu snímaného signálu. Pokud bychom tedy chtěli například měřit zpoždění signálu na nějakých logických obvodech, tak je potřeba vzít existenci tohoto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



**Obr. 3.27:** Zpoždění kanálu 2 a 4 za kanálem 1 a 3 při použití Dual simultaneous modu a stroboskopickém vzorkování s ekvivalentní vzorkovací frekvencí 78MHz

### 3.5.3 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, tak aby se vzorkovací kondenzátor měl čas nabít na vstupní napětí (s tolerancí 1/2 LSB)  $V_{AIN}$  jako je znázorněno na obrázku 3.28. 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  $\pm 0.5\text{LSB}$ .

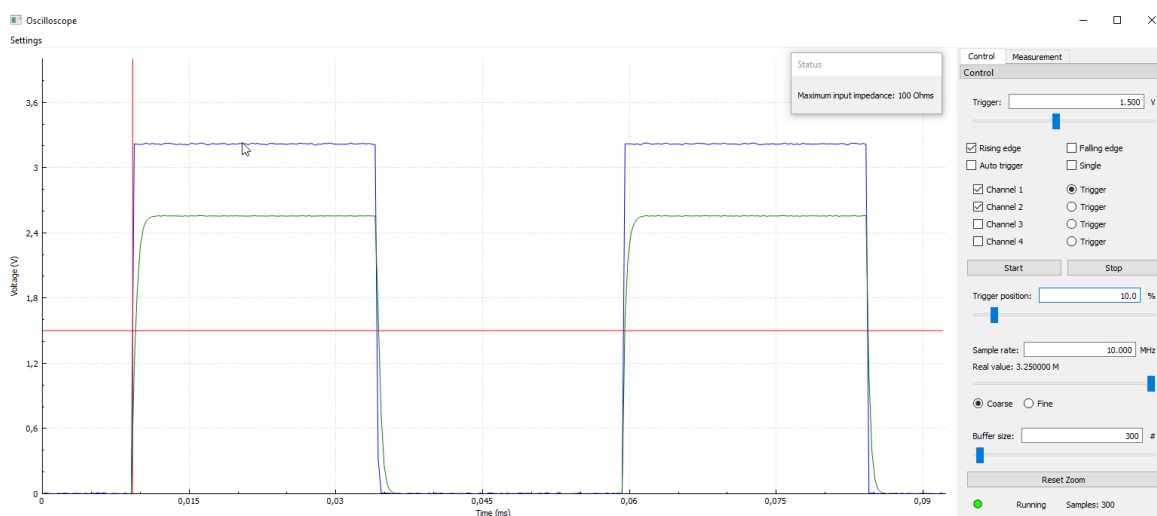


**Obr. 3.28:** převzato z [9]

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 ovlivnit dobu vzorkování změnou frekvence již zmíněného ADC\_CLK. Jako nejlepší řešení se jeví 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. Mikrokontrolér poté informuje uživatele skrz PC aplikaci o maximální impedanci zdroje signálu, aby měření nebylo zkresleno.



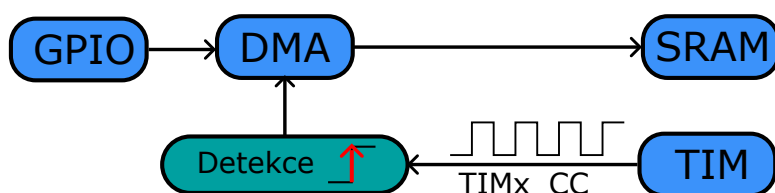
$$t_c = (R_{ADC} + R_{AIN}) \times C_{ADC} \quad (3.10)$$



Obr. 3.29: Efekt nedodržení maximálního vstupního odporu

## 3.6 Logický analyzátor

Logický analyzátor nahrazuje funkci osciloskopu pro digitální kanály. Díky tomu, že rozlišujeme jen 2 napěťové úrovně a nepotřebujeme ADC převodník, tak můžeme přímo vyčítat stav vstupních pinů z registrů mikrokontroléru. Výsledkem čehož můžeme vstupní signály vzorkovat s vyšší vzorkovací frekvencí než je tomu v případě osciloskopu, protože hlavním limitem se v tomto případě stává rychlost systémových hodin. Jednodušší je i HW konfigurace (Obr. 3.30), která spočívá pouze v čítači, který řídí přenos stavu vstupů GPIO brány pomocí DMA řadiče. Teoreticky, tak můžeme odebírat až 16 kanálů zároveň za předpokladu, že jsou sledované vstupy v rámci jedné GPIO brány.



Obr. 3.30: HW konfigurace pro logický analyzátor

V předchozích řešeních FW se využívalo kombinace vzorkování logických a analogových kanálů zároveň. Nicméně toto řešení FW znamenalo omezení maximální vzorkovací frekvence. Rozhodl jsem se tedy v rámci svého řešení řešit logický analyzátor samostatně. Při frekvenci systémových hodin  $f_{SYSCLK} = 156MHz$ , tak bylo možno dosáhnout vzorkovací frekvence až 19,5 Msps. Což je dostatečná vzorkovací frekvence pro řadu aplikací jakou je například analýza sériové komunikace pomocí I2C nebo UART se kterými se studenti v hodinách běžně setkávají.

### ■ 3.6.1 Triggerování digitálních vstupů

Podobně jako u osciloskopu se i u logické analyzátoru hodí moci triggerovat záznam na náběžnou či sestupnou hranu zvoleného vstupního signálu. Pro triggerování můžeme využít konfiguraci EXTI (extended interrupts and events controller), která umožní vyvolat přerušení v momentě kdy dojde k změně napěťové úrovně na zvoleném vstupu. Dále pak pro získání určeného počtu vzorků po triggeru postupujeme jako v případě osciloskopu vhodným nastavením čítače, který počítá odebrané vzorky. V rámci mého řešení je možné triggerovat záznam logického analyzátoru na prvních 4 kanálech.



## Kapitola 4

### Ověření funkčnosti





## Kapitola 5

### Možnosti využití přístroje

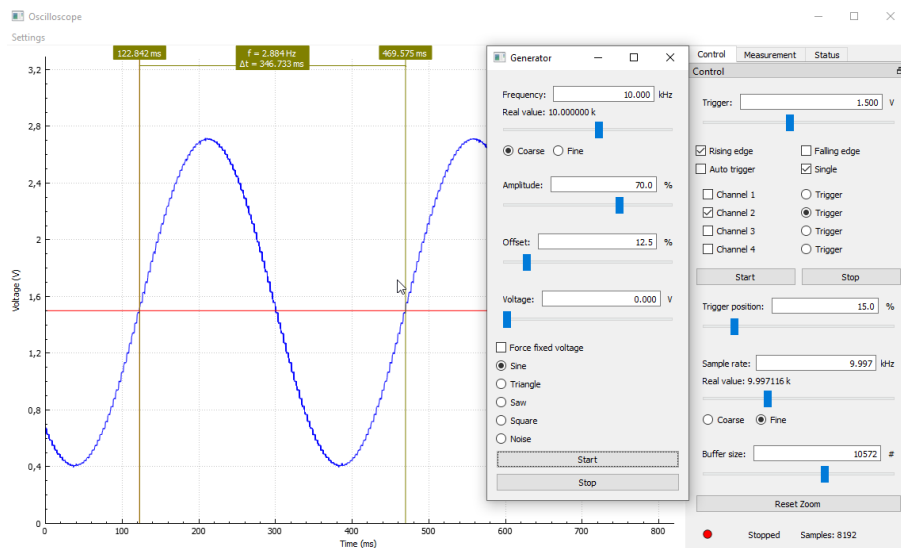


#### 5.1 Měření V-A charakteristiky



#### 5.2 Aliasing

[H]



Obr. 5.1:



## Kapitola 6

### Zhodnocení







## Kapitola 7

### 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].



## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **David**

Jméno: **Petr**

Osobní číslo: **420110**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávací 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