

Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra měření

Firmware pro měřicí přístroj s mikrořadičem STM32G431

Bc. Petr David

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

Velmi děkuji doc. Ing. Janu Fischerovi, CSc. za čas který mi věnoval v rámci konzultací a že mě svoji podporou nenechal pochybovat o dokončení této práce. Dále bych chtěl poděkovat své ženě a dceři, které mi byly po boku po celou dobu vzniku práce a projevovaly pochopení, když jsem se jim zrovna nemohl věnovat.



Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu. V Praze, 26. května 2023

.....
Bc. Petr David



Abstrakt

Práce řeší návrh přístroje SDI s mikrokontrolérem STM32G431, který ve spolupráci s existující PC aplikací Zero eLab Viewer emuluje funkce řady laboratorních přístrojů. Přístroj SDI využívá synchronního vzorkování vstupních signálů pomocí dvou ADC převodníků pro řešení modulu osciloskopu, ale zároveň umožňuje sledování logických signálů v rámci implementovaného modulu logického analyzátoru. Dále se text zabývá možnostmi využití DAC převodníku pro generování různých signálových průběhů a využitím vnitřních periférií čítačů pro měření frekvence vstupních signálů či implementaci modulu impulzního generátoru. Práce také ukazuje možnosti využití vzniklého SDI přístroje pro výukové účely a vysvětluje limity interních periférií použitého MCU.

Klíčová slova: SDI, STM32, mikrokontrolér, oscilokop, signálový generátor, čítač

Vedoucí: doc. Ing. Jan Fischer, CSc.



Abstract

This thesis is focused on the design and implementation of an SDI instrument with microcontroller STM32G431, that together with the existing PC application Zero eLab Viewer, emulates a number of functionalities of laboratory instruments. Finished SDI utilizes synchronous sampling of input signals with two ADC converters for implementing a function of the oscilloscope and also offers the possibility of monitoring logical signals with an implemented module of the logic analyzer. Further text provides insight into the possibilities of using DAC converters for generating a number of different signal waveforms and utilizing internal timer peripherals for measuring the frequency of input signals or for the implementation of pulse generators. This thesis also shows the possibilities of using the designed SDI instrument for educational purposes and explains the limitations that internal peripherals of used MCU have.

Keywords: SDI, STM32, microcontroller, oscilloscope, signal generator, timer

Title translation: Firmware for measuring instrument based on microcontroller STM32G431



Obsah

1 Úvod	1
2 Rozbor	3
2.1 PC aplikace Zero Elab Viewer	4
2.2 Vlastnosti použitého MCU STM32G431	5
2.3 Komunikační protokol aplikace Zero eLab Viewer	6
3 Realizace FW	7
3.1 Struktura FW	7
3.2 Rozpoznání frekvence externího krystalu HSE	8
3.3 Realizace modulu signálového generátoru	12
3.4 Realizace modulu osciloskopu	19
3.5 Realizace modulu voltmetru	26
3.6 Realizace modulu logického analyzátoru	28
3.7 Realizace modulu čítače - měření frekvence a střídy	31
4 Ověření funkčnosti přístroje a možností jeho využití	35
4.1 Signálový generátor	36
4.2 Modul osciloskopu	37
4.3 Ověření funkce modulu logického analyzátoru	39
4.4 Ověření funkce modulů čítače a impulzního generátoru	39
4.5 Ověření funkčnosti modulu voltmetru se záznamem	40
5 Zhodnocení dosažených výsledků práce	43
6 Závěr	45
A Literatura	47

Obrázky

2.1 Modul s STM32G431 na nepájivém poli používaný v rámci výuky LPE. Převezato z [4].....	3
2.2 Modul SDI s STM32F042 s předchystaným FW pro Zero Elab Viewer	3
2.3 STM32G031 zapojen na nepájivém poli. Převezato z [3]	4
2.4 Rozložení bytů ve zprávě typu příkaz(CMD)	6
2.5 Rozložení bytů ve zprávě typu data	6
3.1 Kategorie zdrojového kodu a jejich posloupnost závislosti	7
3.2 Hlavní okno aplikace Zero eLab Viewer zobrazující aktivní moduly dané konfigurace	8
3.3 Zkreslení průběhu měřeného signálu v důsledku nestability HSI. Převezato z [2].....	9
3.4 Vstupy čítače TIM16	10
3.5 Měření periody vstupního signálu pomocí "Input-capture"a DMA	10
3.6 Vyobrazení interního PLL bloku. Převezato z [8]	11
3.7 Postup změny zdroje hodinového signálu.....	11
3.8 Diagram využití DAC převodníku a dalších periférií pro generování analogového signálu	12
3.9 Možnosti nastavení vlastností generovaného signálu v prostředí aplikace Zero eLab Viewer ..	13
3.10 Popis funkce kruhového bufferu jako zdroj datových vzorků pro DAC převodník	14
3.11 Výpočet sinus a cosinu uhlu postupnou aproximací použitou v CORDIC. Převezato z [10]... ..	16
3.12 Naznačení průběhu výpočtu hodnot LFSR registru. Převezato z [8]	17
3.13 Využití LFSR registru pro generování šumu	17
3.14 Záznam generovaného signálu šumu se vzorkovací frekvencí 1 Msps.....	18
3.15 Záznam generovaného signálu šumu se vzorkovací frekvencí 250 Ksps	18
3.16 Princip funkce Analog Watchdog(AWD) ADC převodníku	19
3.17 2-fázové triggerování na nástupnou hranu signálu s využitím AWDG. Převezato z [2]	20
3.18 Vzájemné propojení dvou čítačů	20
3.19 HW konfigurace 1 ADC převodníku pro režim s multiplexováním vstupních kanálů.....	21
3.20 HW konfigurace 2 ADC převodníků pro režim "Independent interleaved"	22
3.21 Vzorkování v režimu "Independent Interleaved"	22
3.22 HW konfigurace 2 ADC převodníků pro režim "Dual simultaneous".....	23
3.23 Vzorkování 4 kanálů v režimu "Dual simultaneous"	24
3.24 Zpoždění kanálu 2 a 4 za kanálem 1 a 3 při použití Dual simultaneous modu a stroboskopickém vzorkování s ekvivalentní vzorkovací frekvencí 52MHz	24
3.25 Vliv odporu zdroje analogového signálu. Převezato z [13]	25
3.26 Efekt nedodržení maximálního vstupního odporu	26
3.27 Využití vnitřní reference jako vstupu pro ADC převodník. Převezato z [8].....	27
3.28 Ovládací okno funkce voltmetru	27
3.29 Vnitřní struktura pinu v "Input"konfiguraci. Převezato z [8]	28
3.30 HW konfigurace pro logický analyzátor	28
3.31 Záznam SPI komunikace s baudrate = 125 KBits/s	29
3.32 EXTI blokový diagram - konfigurace vyvolání přerušování daným pinem. Převezato z [7].....	29
3.33 Rozdělení kruhového bufferu na data před a po přerušování	30
3.34 Princip funkce měření v tzv. PWM input režimu	31

3.35 Diagram propojení čítačů pro upravenou metodu měření frekvence čítáním pulzů. Převzato z [2]	32
3.36 Časový diagram čítání pulzů za proměnnou dobu odběru	33
4.1 Pinout vzniklého SDI	35
4.2 Záznam signálu sinus o $f_{\text{SIG}} = 1000$ Hz s 1000 vzorky na 1 periodu vzorkovaný osciloskopem rychlostí 3.25 Msps	36
4.3 Záznam signálu sinus o frekvenci 50 kHz s 20 vzorky na 1 periodu vzorkovaný rychlostí 6.5 Msps	36
4.4 Záznam obdélníkového signálu, $f_{\text{SIG}}=250$ kHz(4 vzorky na periodu) vzorkovaný rychlostí 6.5 MSps. Pozorování omezené rychlosti přeběhu výstupního zesilovače DAC převodníku	37
4.5 Vzorkování 4 analogových kanálů s vzorkovací frekvencí $f_{\text{SAMP}} = 1.677$ MSps	38
4.6 Záznam signálu obdélník s $f_{\text{SIG}} =250$ kHz vzorkovaný osciloskopem v ETS režimu rychlostí 52 MSps	38
4.7 Záznam SPI komunikace s rychlostí baudrate 125 kBit/s triggerován NSS signálem sběrnice .	39
4.8 Záznam PC aplikace při generování PWM signálu impulzním generátorem a ověření výstupní frekvencí modulem čítače a logického analyzátoru	40
4.9 Záznam voltmetru. Na kanálu č.1 signál sinus o frekvenci 1 Hz, Na kanálu č.2 PWM o frekvenci 2 Hz, Kanál č. 3 signál z potenciometru	40
4.10 Využití funkce DC charakteristiky pro zpracování VA charakteristiky LED diody	41

Kapitola 1

Úvod

V rámci výuky na katedře měření v předmětech, které se zaměřují na oblast elektroniky a měření, 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 SDI nedosahují parametrů profesionálních přístrojů, tak existuje mnoho aplikací, kde jejich výkon ve smyslu maximální vzorkovací frekvence či přesnosti určení napětí je dostatečný a mohou v těchto případech profesionální přístroje nahradit.

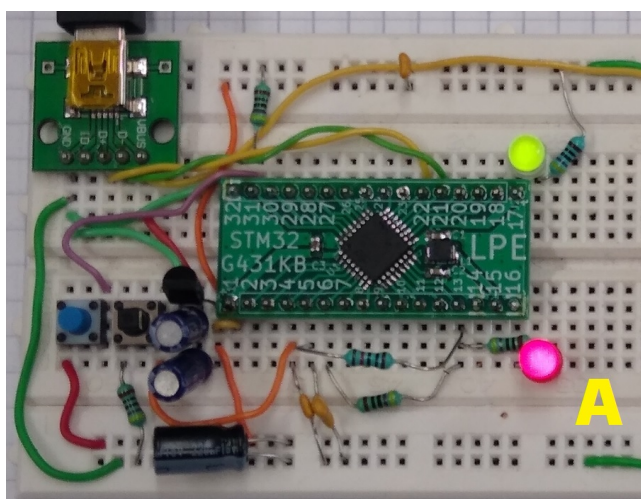
Konkrétně v předmětu laboratoře průmyslové elektroniky (B3B38LPE1) se právě používá SDI běžící na MCU STM32F042 v kombinaci s aplikací Zero eLab Viewer pro různé experimenty, jako je měření časové konstanty RC článku, zapojování různých obvodů s operačními zesilovači a mnoho dalších. SDI se v rámci výuky 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 na katedře měření. Ovládání této aplikace se ale prokázalo méně intuitivní, než je tomu v případě Zero eLab Viewer. Zároveň studenti stále v těchto případech využívají hotový modul s STM32F042. Vznikla tak myšlenka využít výkonného STM32G431 pro implementaci nového a výkonnějšího přístroje spolupracujícího s původní aplikací Zero eLab Viewer, kterou studenti a vyučující dobře znají.

Kapitola 2

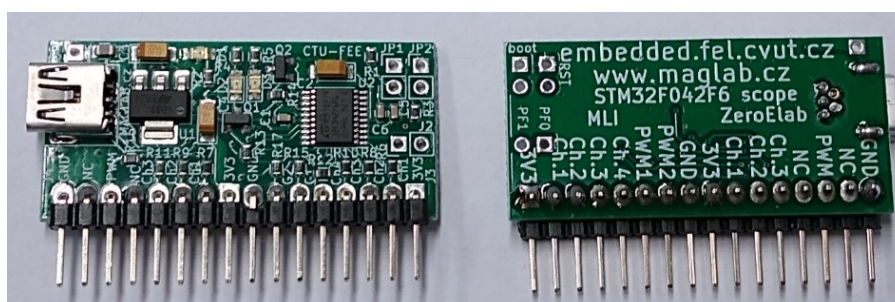
Rozbor

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 periférií než doposud používaná STM32F042. Otevřely se tím možnosti pro implementace SDI s lepšími parametry, jako je například SDI vzniklé v rámci práce [2] spolupracující s aplikací VSVI. Ten oproti dosud používanému řešení nabízí 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 [4]

Nicméně jiné ovládání typu TUI (Text-based User Interface) oproti klasickému grafickému rozhraní a také vyšší komplexnost tohoto řešení způsobily 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, tedy v případě přechodu od jedné implementace SDI k druhé by pak odpadala nutnost se orientovat v novém uživatelském rozhraní.

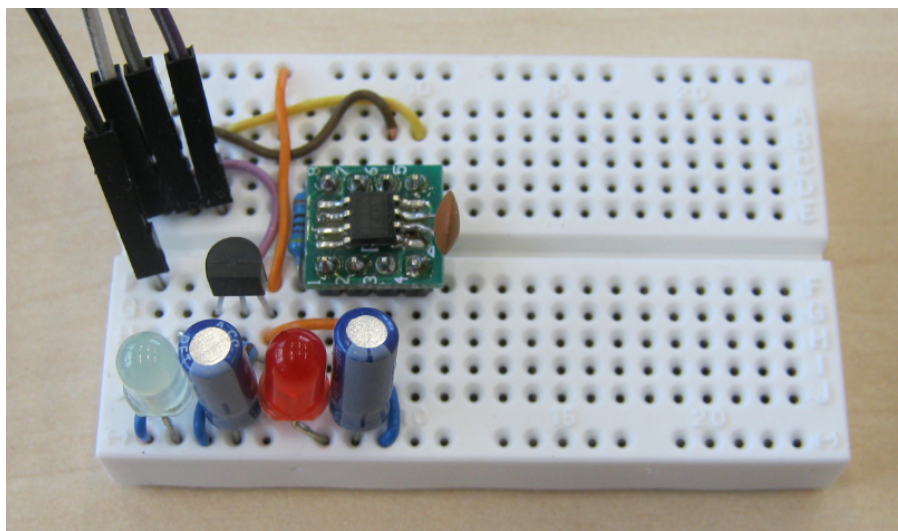


Obr. 2.2: Modul SDI s STM32F042 s předchystaným FW pro Zero Elab Viewer

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 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 signálový generátor. 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 [1] pan Ing. Adam Berlinger vytvořil aplikaci Zero eLab Viewer fungující jako grafické rozhraní pro komunikaci se softwarově definovaným přístrojem (SDI) založeném na STM32F042. Aplikace společně s FW umožňuje použít MCU jako dostupnou náhradu za laboratorní přístroje, jako jsou voltmetr, osciloskop, impulsní generátor, signálový generátor nebo logický analyzátor.



Obr. 2.3: STM32G031 zapojen na nepájivém poli. Převzato z [3]

Aplikace se od té doby stále používá ve výuce praktické elektroniky, následně ještě vznikla řada dalších implementací SDI FW pro jiné MCU. Nejnověji to jsou například verze pro mikrokontrolér ATmega 328, která je určena pro uživatele se zkušenostmi s ARDUINO, nebo zatím nejmenší (a nejlevnější) verze SDI přístroje využívající 8-pinovou variantu mikrokontroléru STM32G030.

■ Existující SDI pro spolupráci s aplikací Zero eLab Viewer a jejich parametry

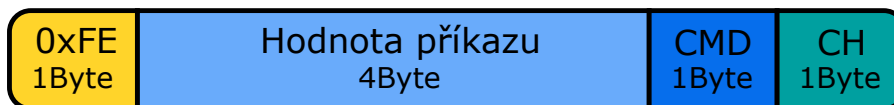
- SDI s STM32F042
 - Osciloskop - záznam až 1152 vzorků, rozlišení 12 bitů,
 - Rychlost záznamu až 1x 600 kS/s
 - Ve stroboskopickém módu - až 48 MS/s - rozlišení s intervalem 15,6 ns.
 - Impulsní generátor PWM , nastavení střídání 0 až 100 %; nastavení frekvence 1 Hz až 24 MHz
 - Voltmetr tři kanály, 0 až + 3,3 V, 100 odměřů/s, průměrování z 1 až 256 odměřů, možnost funkce záznamu
- SDI s 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 střídání 0 až 100 %; nastavení frekvence 1 Hz až 32 MHz
 - Voltmetr tři kanály, 0 až + 3,3 V, 100 odměřů/s, průměrování z 1 až 256 odměřů, možnost funkce záznamu
- SDI s ATMega 328
 - 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 střídání 0 až 100 %; nastavení frekvence 0.5 Hz až 4 MHz
 - Voltmetr tři kanály, 0 až +5 V, rozlišení 10 bitů, 100 odměřů/s, průměrování z 1 až 256 odměřů, možnost funkce záznamu

■ 2.2 Vlastnosti použitého MCU STM32G431

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

2.3 Komunikační protokol aplikace Zero eLab 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, které slouží k oboustrannému předávání pokynů. Jedná se o jednoduché zprávy, ve kterých pole CH určuje, kterému modulu je příkaz určen a 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: typický příkaz, který je pro jednotlivé moduly společný je zapnutí/vypnutí běhu daného modulu. Hodnota příkazu pak rozšiřuje možnost předaných parametrů spojených s daným příkazem. Druhým typem zpráv jsou data, která míří z MCU do aplikace v PC specifikovaná číslem kanálu (modulu), z jakého data pocházejí, délkou data, po které 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 vznikl 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 byly 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 přístroje 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**

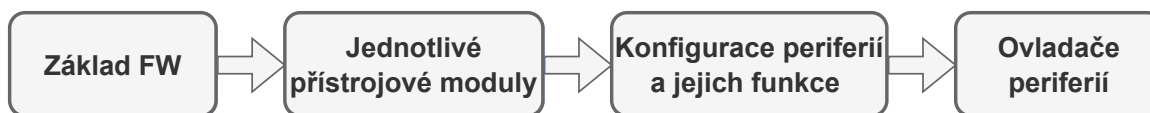
Rozdělit implementaci jednotlivých přístrojů do samostatně fungujících bloků je výhodné z 2 důvodů. Zaprvé lze tak jednodušeji výsledný FW přizpůsobit pro různé mikrokontroléry v závislosti na dostupných perifériích a velikosti FLASH paměti. Druhým důvodem je možnost použití částí kódu jako příklady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

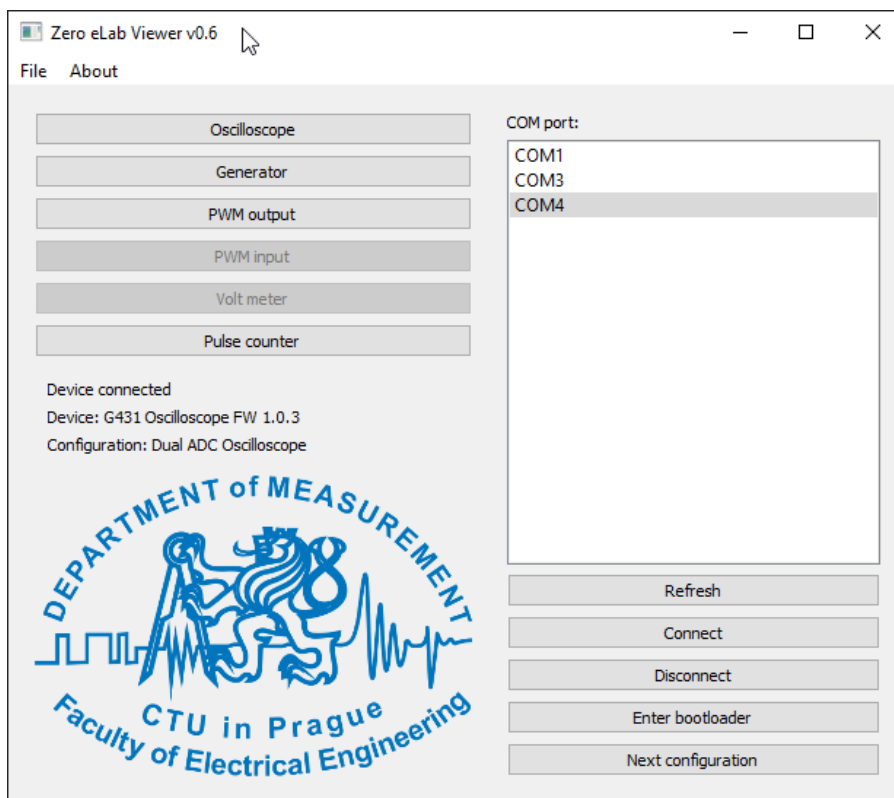
Nástroj STM32CubeMX velmi usnadňuje začátek vytváření FW díky využití grafického rozhraní pro definování počáteční konfigurace MCU. Pokud se zachová struktura generovaného kódu, lze pak nástroj opětovně využít v případě, že chceme jednoduše upravit HW konfiguraci.



Obr. 3.1: Kategorie zdrojového kódu a jejich posloupnost závislosti

- **3.1.1 Využití možnosti přepínání mezi jednotlivými konfiguracemi modulů**

Především z důvodů omezených hardwarových prostředků na různých mikrokontrolérech je využito možnosti přepínání různých přístrojových konfigurací. Konfigurací v tomto případě myslíme set nástrojů(modulů), které mohou být použity současně. Přístrojové moduly pak mohou sdílet periférie 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 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 portfolia 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 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 parametrů a uživatel 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 diodou. V prvním případě byly periferie RCC (Reset and clock control) a GPIO inicializovány prostřednictvím HAL knihovny a v druhém případě pomocí LL ovladačů. Dále jsem použil odpovídající zpovědovací funkce společně s přepínáním napěťové úrovně na výstupu pinu. Výsledkem bylo, že po zkompileování verze s HAL využívala asi o 46 procent více FLASH paměti.

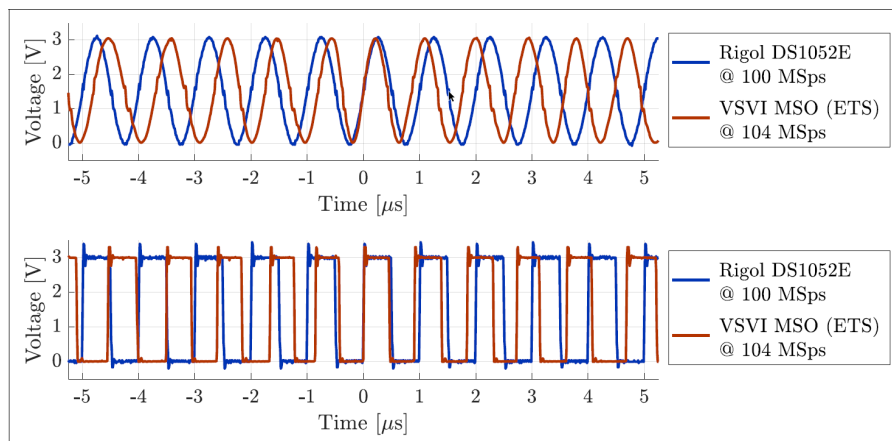
	FLASH	RAM
LL ovladače	3,04 kB	1,53 kB
HAL ovladače	5,66 kB	1,55 kB

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 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 obzvláště 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 $\pm 0.005\%$ [6]. 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]

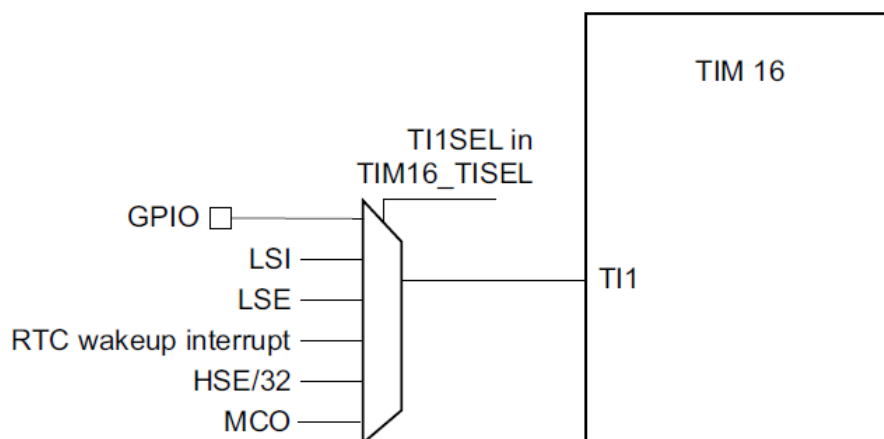
Jako zdroj vysokorychlostního externě získaného hodinového signálu (HSE) lze buď využít krystalu buzeného pomocí MCU (HSE crystal) nebo jiného externího zdroje signálu (HSE bypass) splňující podmínky na maximální frekvenci a střihu hodinového signálu pro dané MCU. Například pro STM32G431 musí být externí signál v rozsahu 4-48MHz a mít střihu 40-60%. Ve výuce laboratorních měření na katedře měření jsou k dispozici krystaly různých výstupních frekvencí převážně pak 8 MHz, 12 MHz a 16 MHz. 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 nastavením 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	$\pm 1\%$	$\pm 1\%$ (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: Srovnání odchylek frekvence HSI a externího krystalu

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[8]

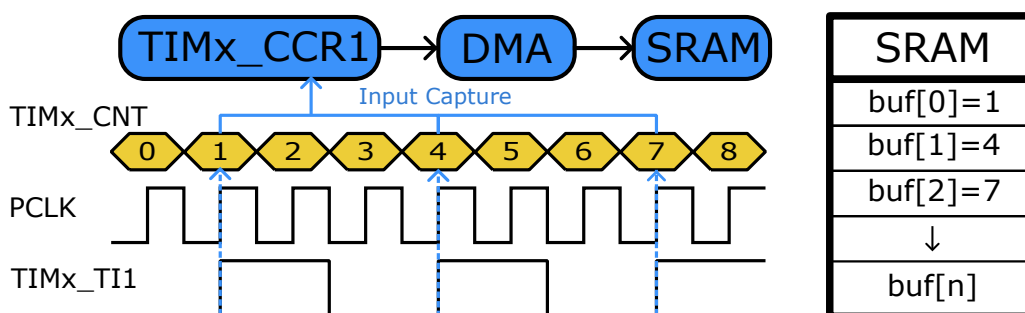
Standardní způsob měření periody vstupního signálu probíhá tak, že měříme počet uběhlých cyklů čítače mezi jednotlivými náběžnými hranami nebo sestupnými hranami jako je vyobrazeno na obrázku 3.5. Tento počet cyklů nám pak určuje poměr mezi frekvencí externího hodinového signálu na vstupu f_{IN} a hodinového 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.

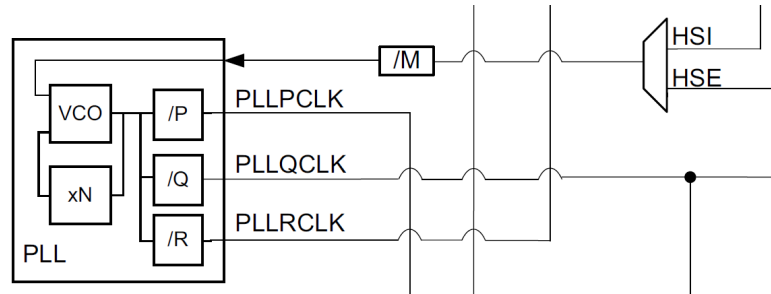


Obr. 3.5: Měření periody vstupního signálu pomocí "Input-capture" a DMA

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 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.

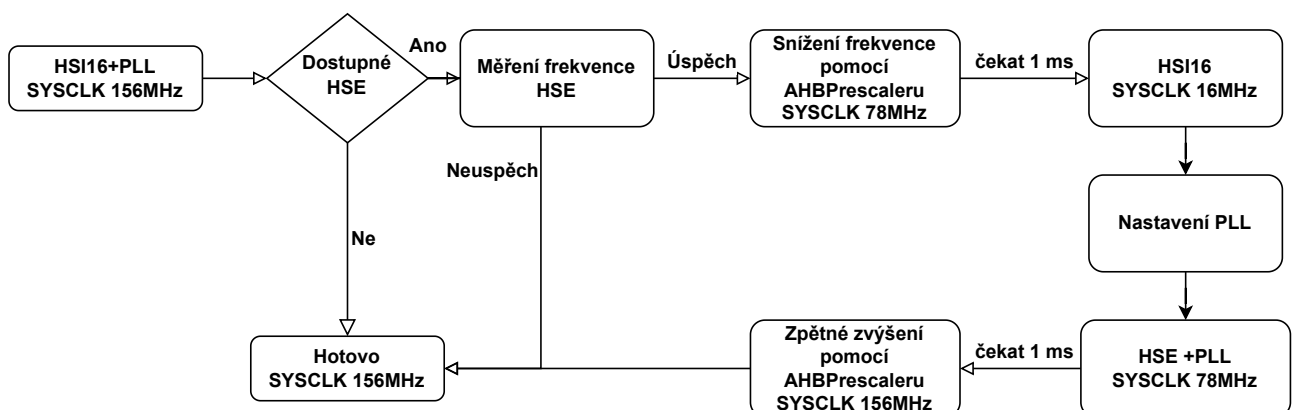


Obr. 3.6: Vyobrazení interního PLL bloku. Převzato z [8]

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 [8] 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 modulu signálového generátoru

Součástí zadání byla implementace signálového generátoru. V práci je využít vnitřní převodník DAC pro generování jedнокanálového 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.

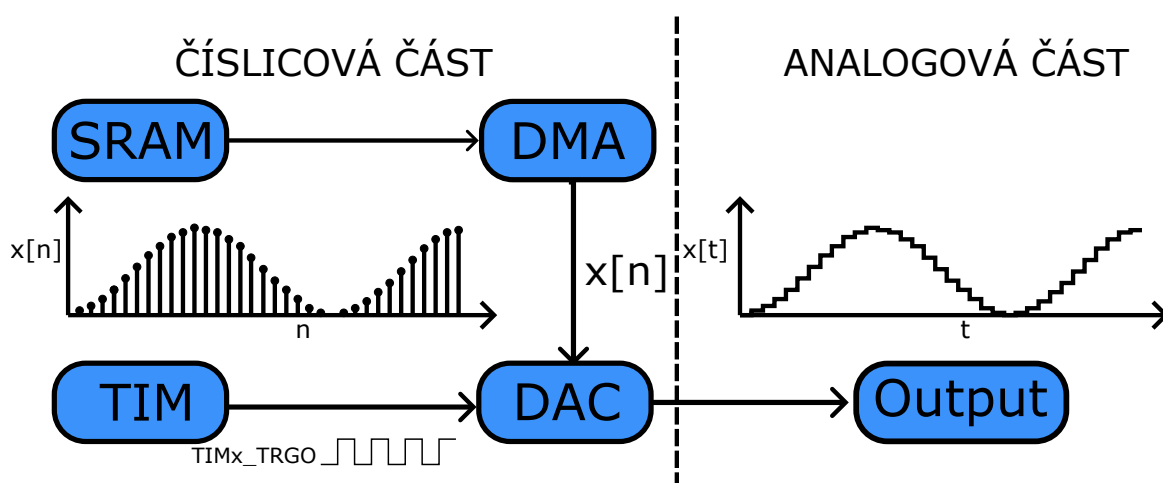
3.3.1 Použitý DAC převodník a jeho limity

Použitý interní DAC převodník nabízí dle [11] 2 výstupní kanály s maximální vzorkovací frekvencí 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 typu "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 číslkové hodnoty jednotlivých vzorků jsou vypočtené za běhu a nejsou uloženy někde v paměti. Tohoto způsobu generování nových číslkových hodnot pro DAC převodník bylo v této práci využito pro generování pseudošumu, které popisují v kapitole 3.3.3

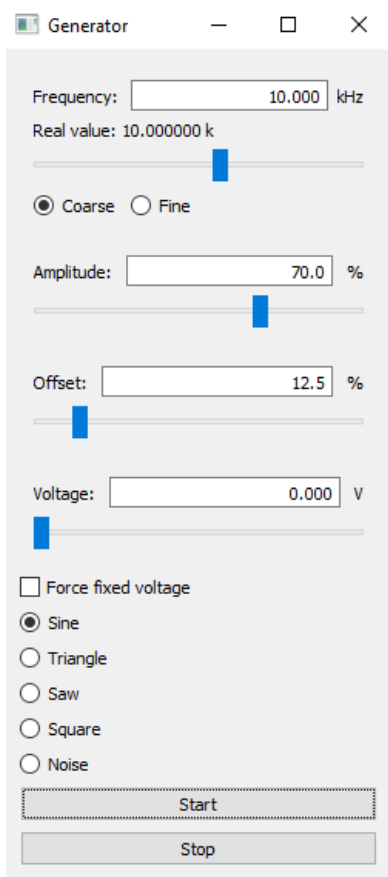
Nevýhodou využití interních registrů pro generaci signálových průběhů jsou především omezené možnosti ohledně změny tvaru signálu, kdy nám například chybí možnost generování sinusového signálu. Alternativou je uložit číslkové hodnoty v rozlišení DAC převodníku pro jednotlivé vzorky do paměti mikrokontroléru 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 periférií 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 generovaného signálu $f_{SIG}=1$ kHz.



Obr. 3.9: Možnosti nastavení vlastností generovaného signálu v prostředí aplikace Zero eLab Viewer

- **Výstupní frekvence generovaného signálu** Požadovanou frekvenci lze nastavovat v rozsahu 1 Hz - 250 KHz. V závislosti na nastavení této frekvence generovaného 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 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:

$$O(g) = \langle 0, \text{rozkm} \times 0xFFFF \rangle \quad (3.3)$$

- **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 :

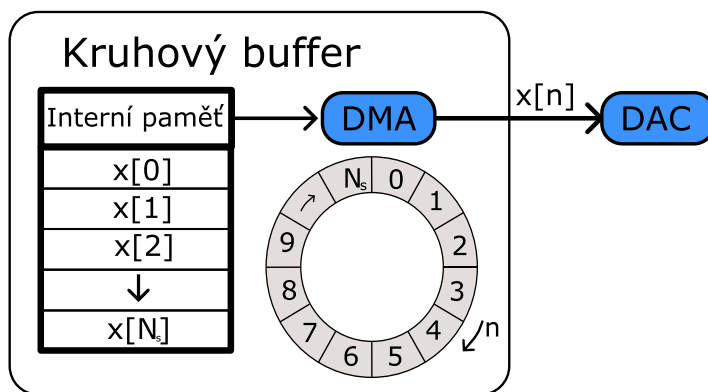
$$f(n) = \min((\text{offset} \times 0xFFFF) + g(n), 0xFFFF) \quad (3.4)$$

- **Tvar průběhu signálu** Prostředí aplikace umožňuje zvolit různé signálové průběhy - sinusový, trojúhelníkový, 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 v kruhovém bufferu (obr. 3.10) na jednu periodu generovaného signálu v závislosti na velikosti dostupné paměti. Dále na základě požadované výstupní frekvence generovaného signálu f_{SIG} , 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} , frekvenci f_{SIG} a počtu vzorků v N_s je vyjádřena vzorcem:

$$DIV = \frac{f_{PCLK}}{f_{SIG} \cdot N_s} \quad (3.5)$$



Obr. 3.10: Popis funkce kruhového bufferu jako zdroj datových vzorků pro DAC převodník

Nicméně zde narážíme na omezené možnosti nastavení f_{SIG} v podobě dělicího poměru DIV, který může nabývat pouze celočíselných hodnot. Pro zlepšení rozmezí nastavitelných frekvencí výstupního generovaného signálu jsem se inspiroval článkem [5], kde autor využívá toho, že nepracuje s konstantní délkou záznamové paměti, ale adaptivně ji zkracuje pro přesnější nastavení výstupní frekvence signálu. Metoda spočívá v iterativním zkracování délky bufferu a minimalizování absolutní odchylky frekvence generovaného signálu.

V rovnicích 3.6 uvádím vztahy pro výpočet reálné výstupní frekvence generovaného f_{SIG} , 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.

$$f_{\text{Sig}} = \frac{f_s}{N_s}, \quad f_s = \frac{f_{\text{PCLK}}}{\text{DIV}} \quad \implies \quad f_{\text{SIG}} = \frac{f_{\text{PCLK}}}{\text{DIV} \cdot N_s}, \quad f_s = \frac{f_{\text{SIG}}}{\text{DIV} \cdot N_s} \quad (3.6)$$

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 3.7. Při optimalizaci délky bufferu v prvním kroku nehledáme jen dělicí poměr DIV, ale faktorizaci tohoto dělicího 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 [2].

$$\text{DIV} = (\text{ARR} + 1) \cdot (\text{PSC} + 1) \quad (3.7)$$

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.

Po určení vzorkovací frekvence jsem dále pokračoval optimalizací délky bufferu, kde jsem postupoval podle pseudo-kódu uvedeného v [5]. 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 dosažená, 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 uvedená [5] 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 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 3.3 jsem pak určil maximální odchylky a související frekvence pro které se 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

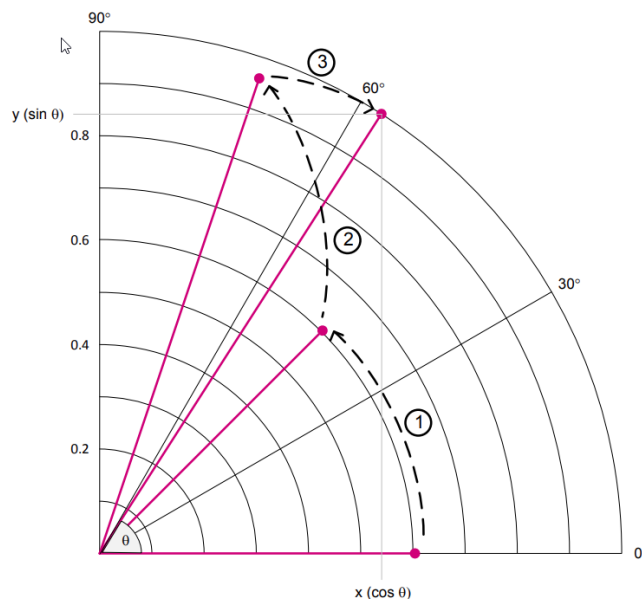
Tab. 3.3: Maximálních dosažených odchylek frekvencí výstupních signálů pro různá rozmezí frekvencí

■ 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 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 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.[14]. Pro získání výsledku používá metody postupné aproximace a případně 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 jsem se ve své práci snažil zjistit možný přínos využití této metody pro výpočet hodnot 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 Viewer 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.

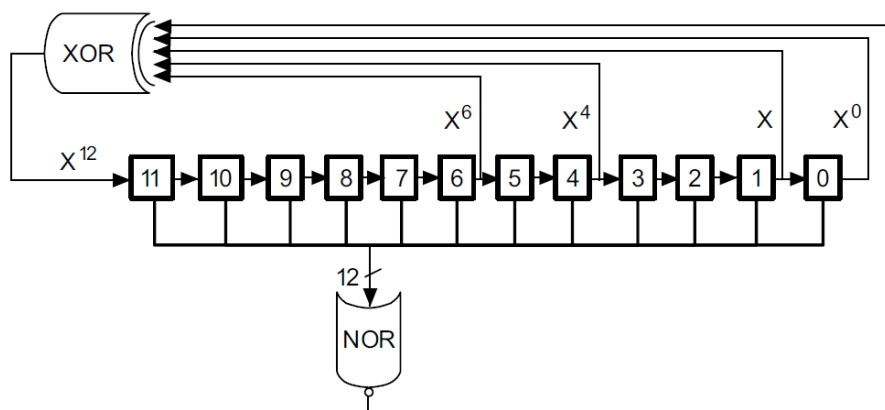
Použitá metoda	Doba plnění bufferu	Čas při SYSCLK = 156MHz
CORDIC	37541 cyklů	0.241 ms
<code>sinf(x)</code> z <code>math.h</code>	85573 cyklů	0.549 ms

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 žádné dopady na plynulost běhu aplikace a uživatel dobu výpočtů hodnot nijak nezaznamená.

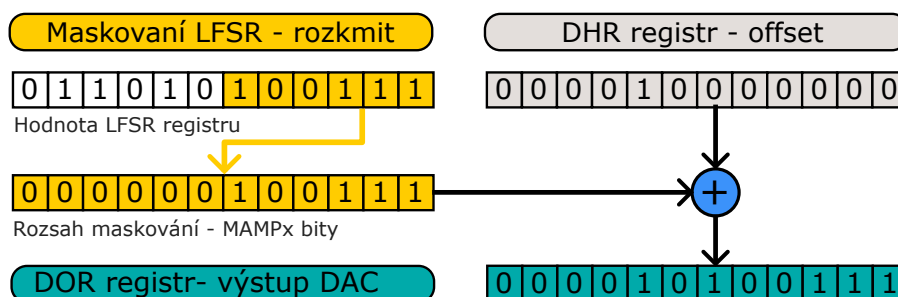
3.3.3 Generování šumu

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 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 popsáno na obrázku 3.12.



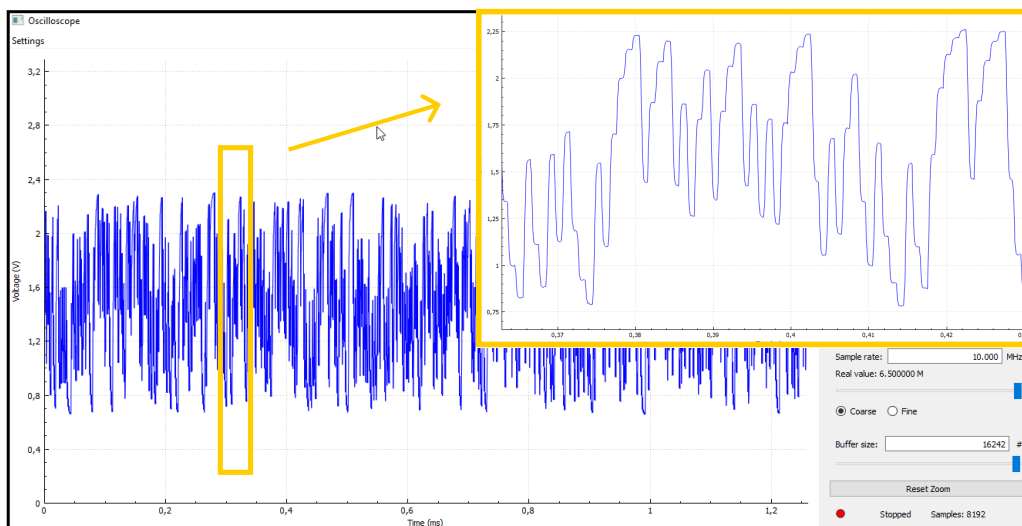
Obr. 3.12: Naznačení průběhu výpočtu hodnot LFSR registru. Převzato z [8]

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 na rozdíl od pravého bílého šumu nemá Gaussovské (normální) rozdělení, ale rovnoměrné.



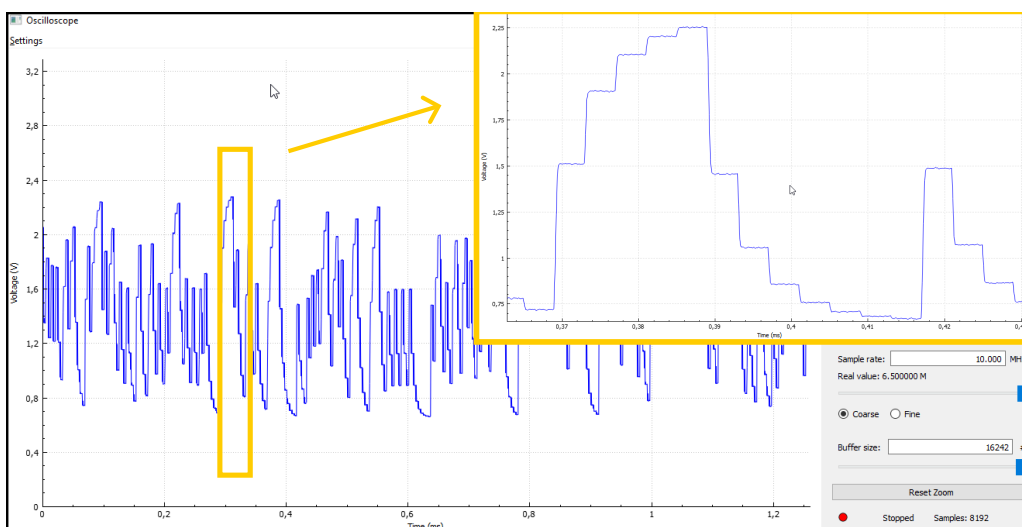
Obr. 3.13: Využití LFSR registru pro generování šumu

Z obrázku 3.13 vyplývá, jakým způsobem lze generování šumového signálu nastavovat podobně jako u ostatních 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. 3.14: 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 frekvencí (250 Ksps) je vidět na obrázku 3.15.



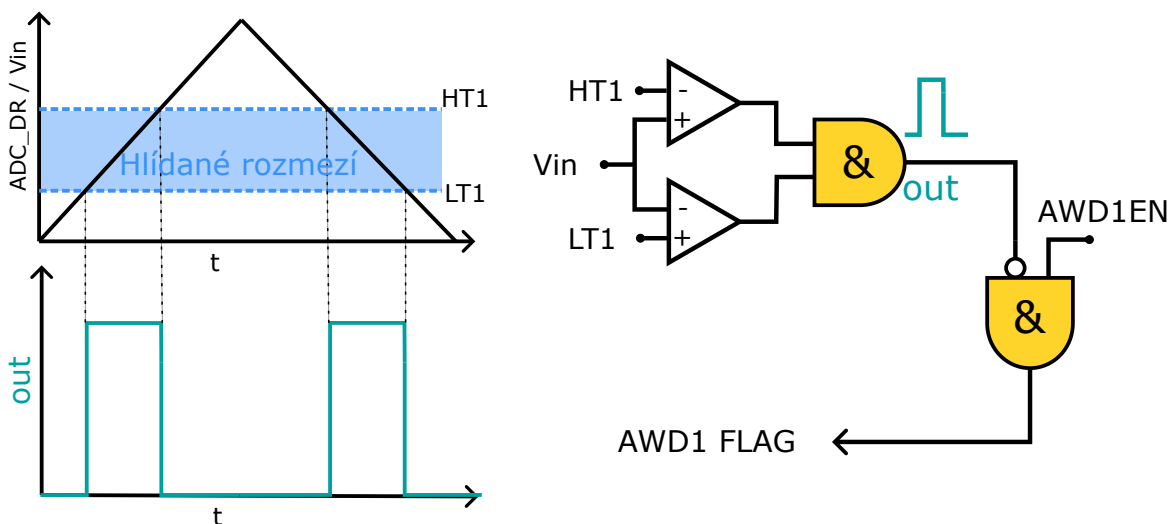
Obr. 3.15: Záznam generovaného signálu šumu se vzorkovací frekvencí 250 Ksps

3.4 Realizace modulu osciloskopu

Modul osciloskopu je nejkomplexnější z modulů implementovaných v této práci a též využívající nejvíce HW prostředků. K jeho implementaci jsem využil obou převodníků 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 [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. 3.16: Princip funkce Analog Watchdog(AWD) ADC převodníku

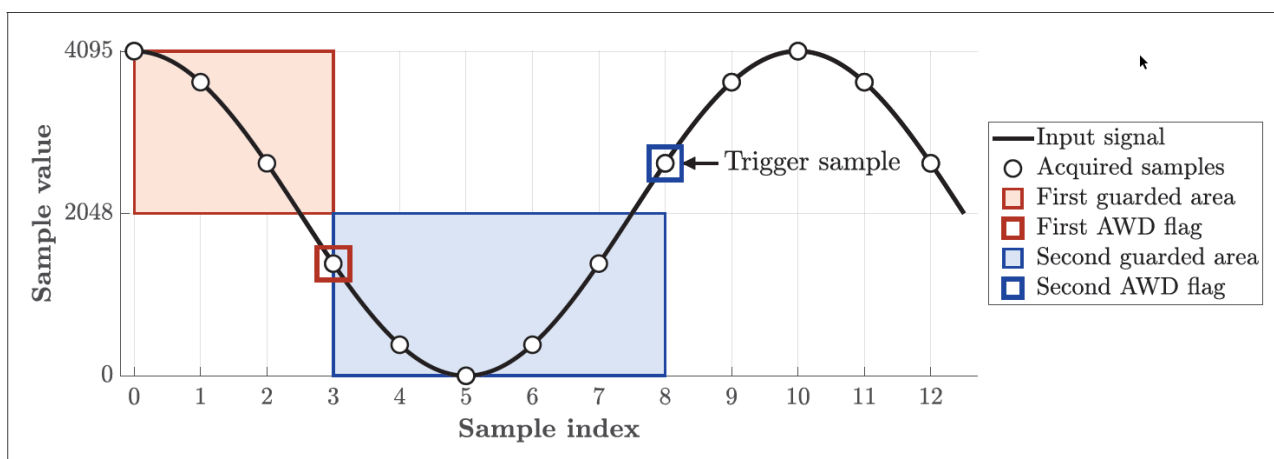
Blok AWDG umožňuje vyvolat přerušení v momentě, kdy se danému kanálu objeví napětí mimo nastavené rozmezí. Abychom mohli určit okamžik, kdy došlo k poklesu pod určitou úroveň (sestupná hrana) nebo naopak k překročení napěťové úrovně (náběžná hrana), je potřeba využití AWDG ve dvou fázích znázorněných na obrázku 3.17.

■ Fáze 1

Při hledání okamžiku nástupní hrany v první fázi nejdříve nastavíme rozmezí nad střeženou napěťovou úrovní. Jakmile 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 [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.

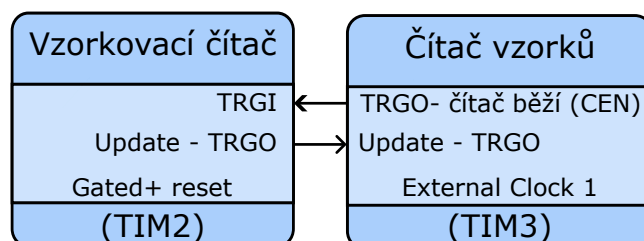
■ Vzorkovací čítač

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 v takzvaném 'one pulse' modu, tak se oba čítače zastaví po odběru stanoveného počtu vzorků.



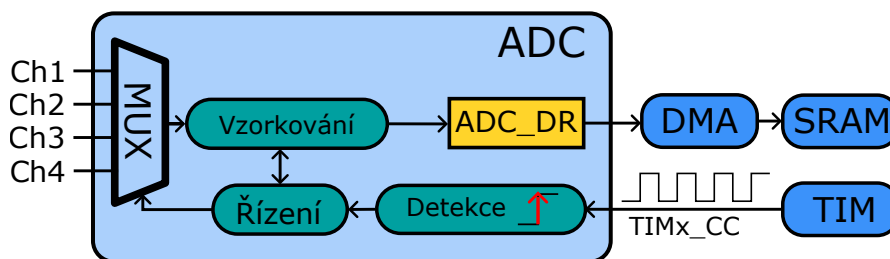
Obr. 3.18: Vzájemné propojení dvou čítačů

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

3.4.2 Módy vzorkování

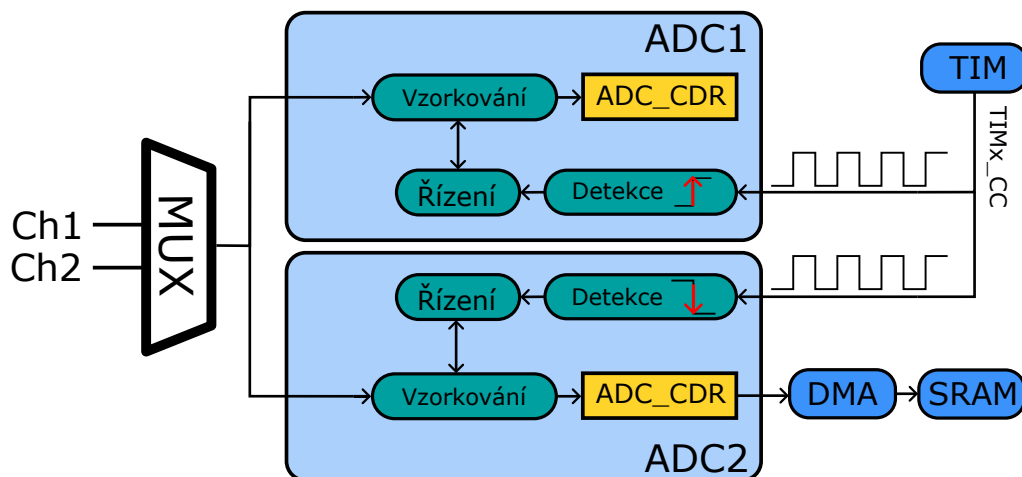
Pokud máme k dispozici pouze jeden ADC převodník, musíme v případě měření signálu na více kanálech vstup ADC převodníku přepínat mezi jednotlivými kanály. V případě využití dvou a více ADC převodníků máme daleko větší variabilitu, jak vzorkovat daný set analogových kanálů. Podrobněji se jednotlivým módům věnuje práce [2]. STM32G431 má k dispozici 2 ADC, které lze využít pro zvýšení 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(kanál 1), 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 analogového kanálu

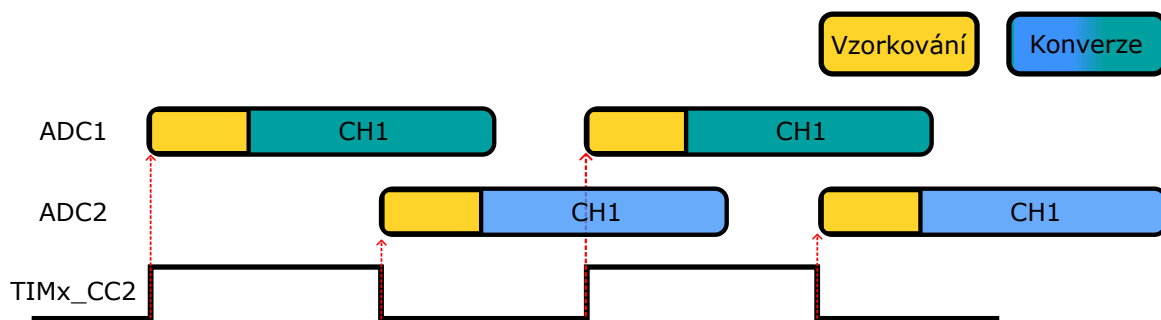
Pokud je zvolený analogový kanál osciloskopu s využitím vnitřních multiplexerů, je možné přivést zároveň (paralelně) na vstupy obou ADC převodníků, můžeme využít tzv. "Dual -interleaved modu". Tento režim spočívá ve střídavém vzorkování jedním převodníkem a v průběhu konverze odebraného vzorku vzorkování 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 stanoveným počtem 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 obdélkový 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 spuštěn událostí dokončení konverze na ADC2.



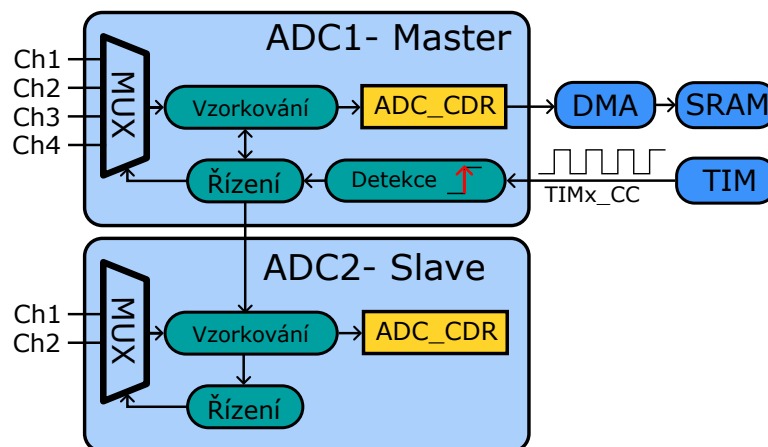
Obr. 3.21: Vzorkování v režimu "Independent Interleaved"

Vzorkování 2 a více analogových 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 mikrokontroléry dost liší a to i v závislosti na použitém pouzdře. Abych udělal firmware jednodušeji portovatelný na další zařízení, tak jsem mému 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 posloupností kanálů doplní kanály dostupné na obou převodnících tak, aby pokud možno oba ADC převodníky snímaly stejný počet kanálů. Vznikají tak různé konfigurace na základě zvolených kanálů.

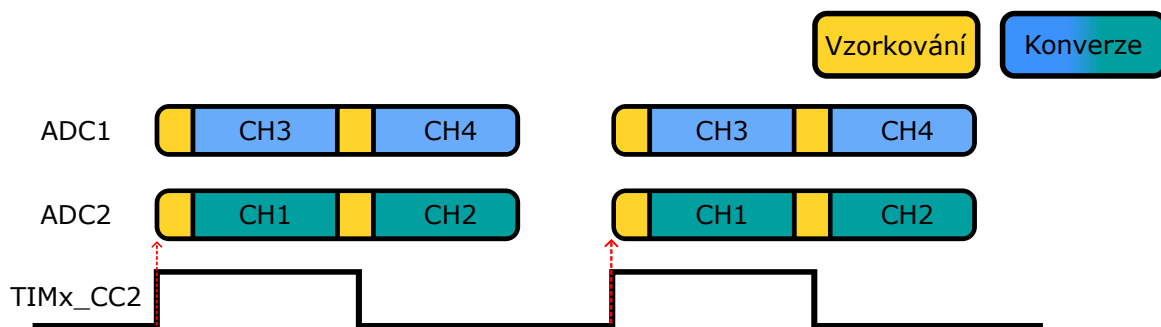
Při analogových kanálech, tak jak jsou popsány v tabulce 3.5, je při zvolení jakýkoliv 2 kanálů CH1-CH4 zajištěno snímání ve stejný čas, kromě kombinace CH3+Ch4, které jsou oba dostupné pouze na ADC1 a tedy musí být vzorkovány multiplexováním vstupu ADC převodníku. To má za následek v případě této kombinace 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 řídicího (MASTER) ADC převodníku, který sám řídí podřazený ADC převodník. HW konfigurace včetně zbylých periférií je ukázána na obrázku 3.22. V tomto režimu se také podobně, jako je tomu u interleaved režimu, 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 3.23. 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 upravují na základě zvolené vzorkovací frekvence, tak aby doba odběru vzorku byla pokud možná nejdelší.

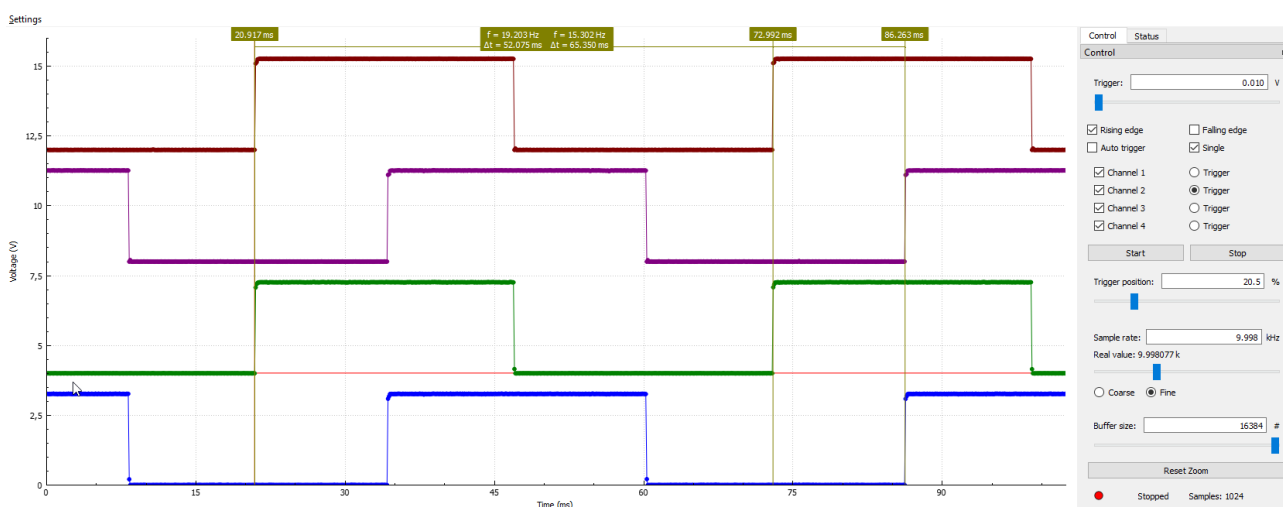


Obr. 3.23: Vzorkování 4 kanálů v režimu "Dual simultaneous"

V běžném režimu vzorkování si zpoždění odběru vzorku pravděpodobně nevšimneme, protože zpoždění bude typicky menší než doba která uběhne mezi dvěma 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 $N_{\text{Delay}} = 653$ cyklů ADC převodníku, což při frekvenci hodin ADC převodníku $f_{\text{ADC}} = 52 \text{ MHz}$ odpovídá zpoždění $T_{\text{Delay}} = 12.55 \mu\text{s}$

$$T_{\text{Delay}} = \frac{N_{\text{Delay}}}{f_{\text{ADC}}} = \frac{653}{52 \text{ MHz}} = 12.5 \mu\text{s} \quad (3.8)$$

Na vstup byl při tom přiveden obdélníkový 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 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.

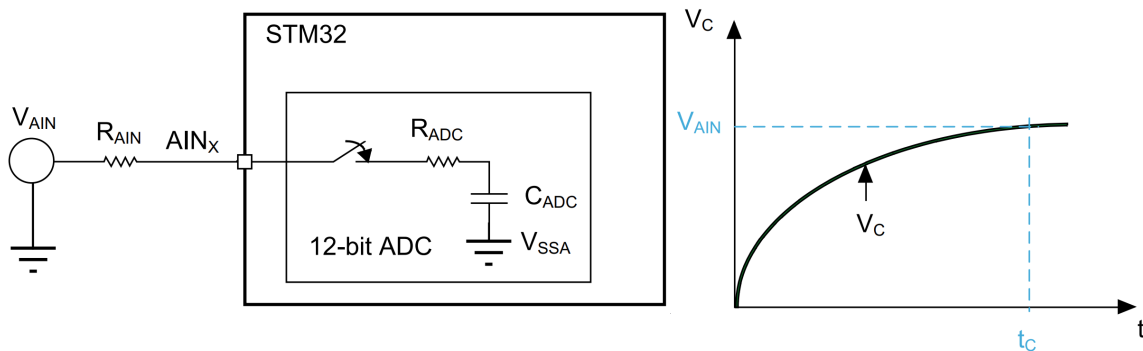


Obr. 3.24: Zpoždění kanálu 2 a 4 za kanálem 1 a 3 při použití Dual simultaneous modu a stroboskopickém vzorkování s ekvivalentní vzorkovací frekvencí 52 MHz

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 $T_s > t_c$, 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.

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



Obr. 3.25: Vliv odporu zdroje analogového signálu. Převzato z [13]

Doba vzorkování T_s je určena počtem cyklů doby vzorkování N_s hodinového signálu ADC (ADC_CLK) převodníku, jehož frekvenci označují 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 $f_{ADC} = 52\text{MHz}$ 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 následným způsobem. Máme nějaký počet dostupných ADC cyklů pro odběr a konverzi vzorků $N_{available}$ a celkový potřebný počet cyklů pro odběr vzorků N_{total} pro které platí:

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

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 .

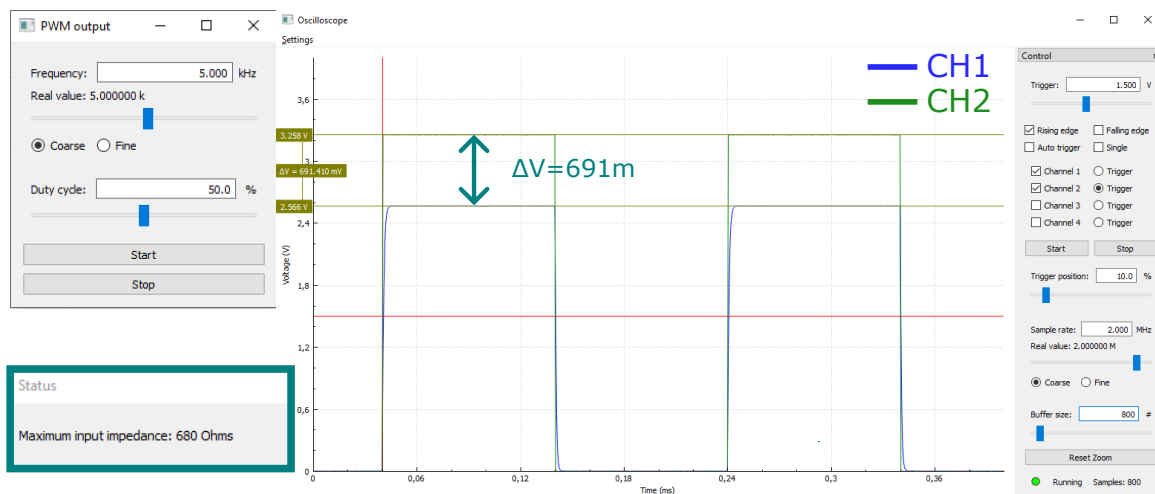
Zmíněný vzorec je vhodné použít pokud známe přesné vnitřní parametry ADC převodníku. Ty v mém případě nebyly známy a z dat v datasheetu STM32G431 vyplývá, že například hodnota vnitřního odporu ADC převodníku R_{ADC} se mění v závislosti na konfiguraci. Pro určení maximální vstupního odporu pro danou dobu vzorkování T_s jsem tedy místo vzorce použil hodnoty uvedené v datasheetu[11]. V mé implementaci byly 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 odpovídajícími dobami vzorkování

Na obrázku 3.26 je vyobrazen příklad PWM signálu s frekvencí $f_{pwm} = 5$ kHz přivedeného na 2 vstupy modulu osciloskopu. Kanál CH1 je se zdrojem pwm signálu propojen přes rezistor s odporem $R=47$ k Ω a CH2 je pak připojen na přímo. Pro vzorkovací frekvenci $f_{SAMP} = 2$ MHz dostaneme informaci o doporučeném maximálním vstupním odporu $R_{AIN} = 680$ Ω , který dle tabulky 3.6 odpovídá době vzorkování $T_s = 240$ ns. Z obrázku je patrný rozdíl napětí $\Delta V = 691$ mV mezi kanály CH1 a CH2 způsobený tím, že se u měření kanálu CH1 vzorkovací kondenzátor nestíhá dobíjet kvůli nedodržení podmínky na maximální vstupní odpor 3.9.



Obr. 3.26: Efekt nedodržení maximálního vstupního odporu

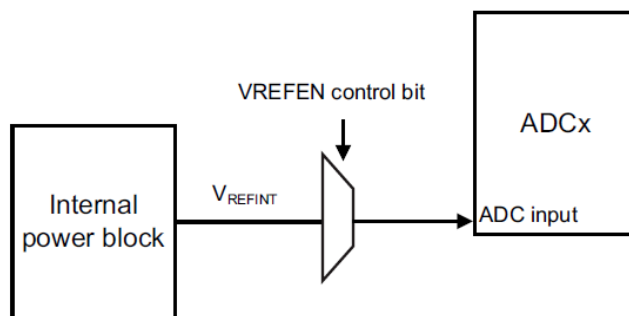
3.5 Realizace modulu voltmetru

Na straně MCU se realizace funkce voltmetru příliš neliší od funkce osciloskopu. Pomocí 2 ADC převodníků vzorkujeme 4 analogové kanály v režimu Dual-Simultaneous popsaného v kapitole 3.4.2. V tomto režimu se vzorkují vždy 2 vstupní kanály zároveň a na rozdíl od osciloskopu u voltmetru nemůže uživatel PC aplikace měnit vzorkovací frekvenci, která je na straně fw napevno definovaná $f_{SAMP} = 100$ Hz. Zvolenými kanály, které jsou vzorkovány, jsou také napevno zvoleny kanály CH1 až CH3 a kanál vnitřní reference popsané v tabulce 3.5.

Pokud na vstup ADC převodníku přivedeme napětí vnitřní reference V_{REFINT} (obr. 3.27), můžeme využít kalibrační hodnoty V_{REFINT_CAL} změřené při výrobě a uložené v paměti MCU pro určení velikosti referenčního napětí V_{REF+} . Velikost referenčního napětí je pak dána známou hodnotou referenčního napětí při kalibraci V_{REF_CAL} a aktuální 12-bitovou hodnotou změřenou ADC převodníkem na kanálu vnitřní reference .

$$V_{REF+} = \frac{V_{REF_CAL} \times V_{REFINT_CAL}}{V_{REFINT_DATA}} \quad (3.11)$$

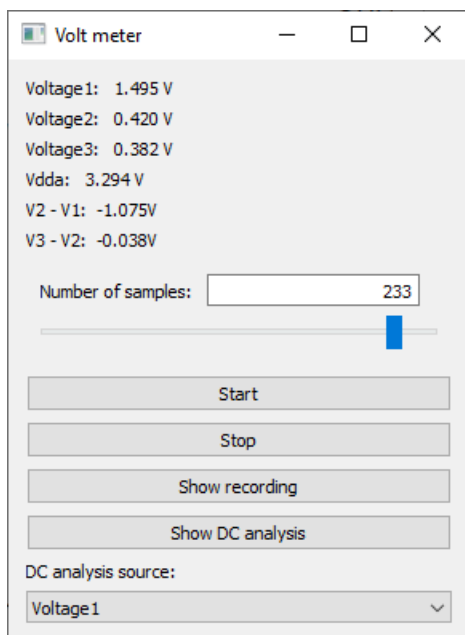
Znalost referenčního napětí pak slouží pro určení absolutní hodnoty napětí na vstupu z dat získaných ADC převodníkem ADC_DATA . V [8] je uveden vzorec pro určení absolutní hodnoty napětí na vstupu



Obr. 3.27: Využití vnitřní reference jako vstupu pro ADC převodník. Převzato z [8]

V_{CH} v závislosti na velikosti referenčního napětí:

$$V_{CH} = \frac{V_{REF+} \times ADC_DATA}{4095} \quad (3.12)$$



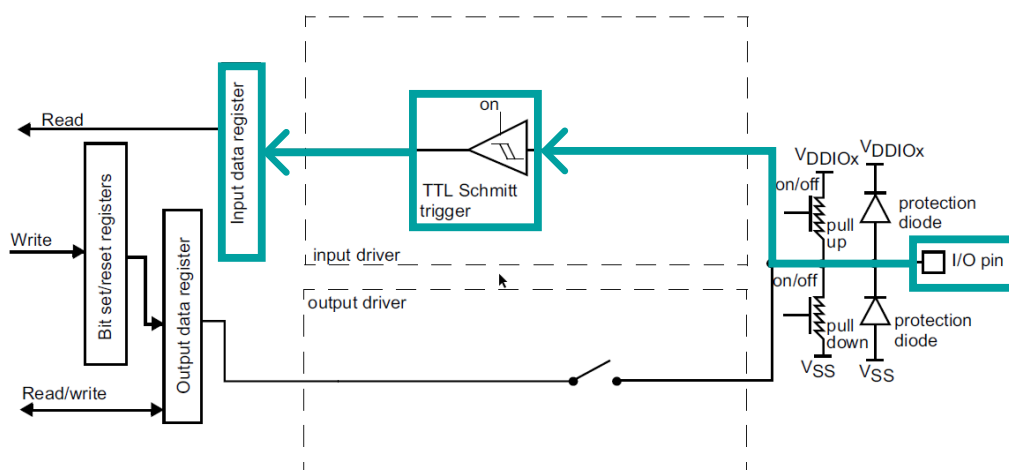
Obr. 3.28: Ovládací okno funkce voltmetru

Kromě zpracování dat z ADC převodníku pro získání absolutních hodnot napětí s ohledem na možné změny referenčního napětí, modul Voltmetru na straně MCU řeší průměrování nastavitelným počtem vzorků. Lze zvolit 1 - 256 vzorků k průměrování změřené hodnoty napětí. Na jednotlivých kanálech jsou pak sčítány a výsledná suma poslána do PC aplikace na jejíž straně se pak řeší dělení sumy nastaveným počtem vzorků.

PC aplikace pak dále implementuje další funkcionality jako zobrazení průběhu napětí na vstupech prostřednictvím tlačítka záznamu (**Show recording**) a změření DC charakteristiky, která kombinuje funkci voltmetru se signálovým generátorem, který slouží ke krokovému nastavení stejnosměrného napětí. Díky tomu je možné postupně v čase měnit nastavení napětí na vstupu nějakého obvodu a zaznamenat změny napětí na výstupu měřeného obvodu pomocí voltmetru.

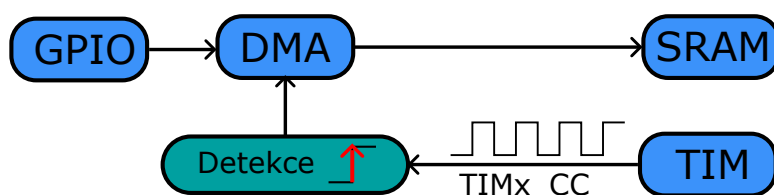
3.6 Realizace modulu logického analyzátoru

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. 3.29: Vnitřní struktura pinu v "Input" konfiguraci. Převzato z [8]

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. 3.29). 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. 3.30: HW konfigurace pro logický analyzátor

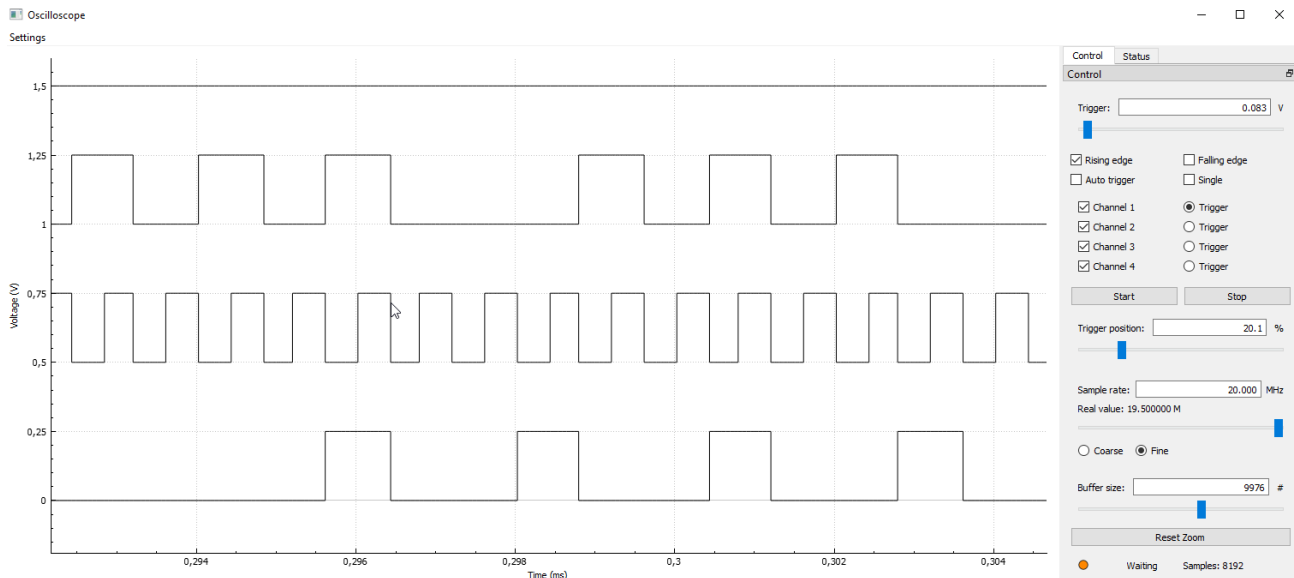
Pro synchronní vyčítání registru GPIO_IDR můžeme efektivně využít DMA řadiče ve spojení s čítačem podobně, jako v případě osciloskopu. Čítač pak určuje časovou základnu, pro odběr vzorků řídí přenosy hodnot registru do bufferu v SRAM paměti. Tato propojení periférií je znázorněno na obrázku 3.30.

Maximální vzorkovací frekvence, kterou jsme schopni snímat vstupy logického analyzátoru je limitována rychlostí DMA přenosu. DMA přenos typicky trvá 6 cyklů hodin AHB sběrnice [12]. 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 $N_{DMA_cyklů} = 8$ cyklů, abych snížil vytížení sběrnice a nedocházelo ke kolizím.

Maximální vzorkovací frekvence f_s logického analyzátoru je tedy určena při dané frekvenci hodin AHB sběrnice f_{HCLK} jako:

$$f_s = \frac{f_{HCLK}}{N_{DMA_cyklů}} = \frac{156 \text{ MHz}}{8} = 19.5 \text{ MHz} \quad (3.13)$$

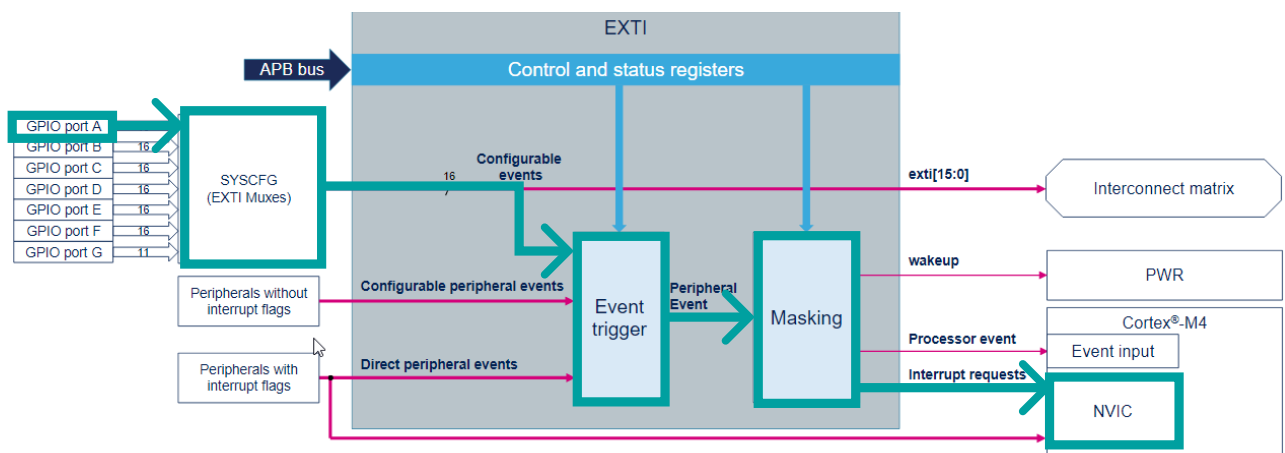
Maximální vzorkovací frekvence až 19,5 MSps, je dostatečná pro řadu využití, jako je například analýza sériové komunikace pomocí I2C, UART nebo SPI, se kterými se studenti v hodinách běžně setkávají.



Obr. 3.31: Záznam SPI komunikace s baudrate = 125 KBits/s

3.6.1 Triggerování logického analyzátoru pomocí externího přerušení - EXTI

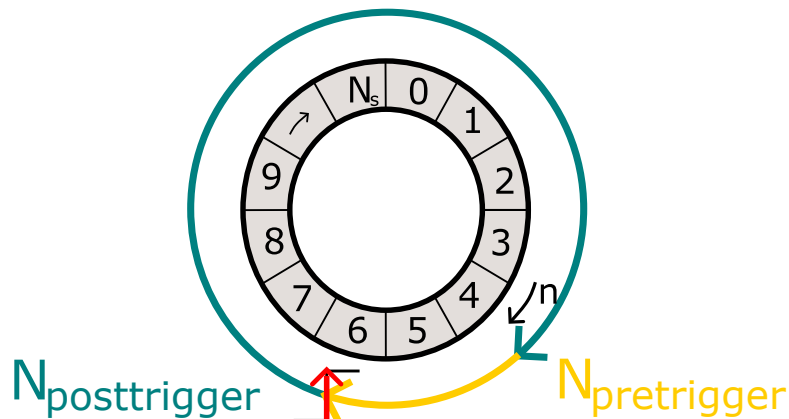
Podobně jako u osciloskopu je i u logického analyzátoru potřeba triggerování záznamu některým ze zvolených signálů. Narozdíl od osciloskopu se v případně logických signálů nemusí vyhodnocovat úroveň napětí, ale rozhodující úlohu hrají náběžné nebo spádové hrany. Pro tento účel lze velmi účelně využít dostupnou řídicí jednotku pro přerušení a události - EXTI (extended interrupts and events controller). Tato řídicí jednotka slouží jako prostředník mezi periferiemi a řídicí jednotkou přerušení (NVIC). Díky tomu stejný signál, který využíváme jako vstupní, může 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. 3.32: EXTI blokový diagram - konfigurace vyvolání přerušení daným pinem. Převzato z [7]

Po vyhodnocení události přerušení pak dojde k 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.

$$N_s = N_{\text{pretrigger}} + N_{\text{posttrigger}}, \quad N_{\text{pretrigger}} = \text{Pretrigger}[\%] \times N_s, \quad N_{\text{posttrigger}} = N_s \times (1 - \text{Pretrigger}[\%]) \quad (3.14)$$



Obr. 3.33: 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 přerušení.

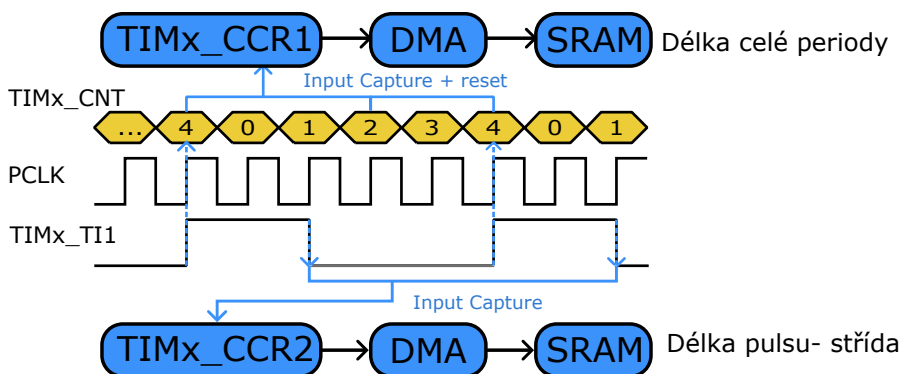
3.7 Realizace modulu čítače - měření frekvence a střídy

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 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 střídy vstupního signálu. V rámci této práce jsem adaptoval dva 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. V rámci mé implementace navíc umožňuje měření střídy signálu. Druhou metodou, kterou jsem se v této práci zabýval, je metoda čítáním pulzů. Ta se naopak hodí více pro vyšší frekvence řádů MHz.

3.7.1 Reciproční metoda měření frekvence určením délky periody - PWM input

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, lze čítač využít i k určení střídy PWM signálu. Metoda je popsána v [8] 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 3.34.



Obr. 3.34: Princip funkce měření v tzv. PWM input režimu

Ve své implementaci jsem se inspiroval řešením popsaným v práci [1]. Toto řešení adaptivně upravuje frekvenci vstupních hodin čítače pomocí nastavování před-děličky (TIMx_PSC) 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 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.

Měření by se dalo popsat následujícím způsobem. Po spuštění čítače se v registru TIMx_CNT zaznamenává počet cyklů vstupních hodin čítače (PCLK) vydělený nastavenou před-děličkou (v obrázku 3.34 je před-dělička rovna 1). V případě příchodu sestupné hrany na vstup čítače se zaznamená aktuální hodnota TIMx_CNT registru do "Capture-Compare" registru čítače TIMx_CCR2 pro následné počítání střídy a v případě příchodu náběžné hrany se zaznamená aktuální hodnota TIMx_CNT registru do registru čítače TIMx_CCR2 pro následné určení periody a vyresetuje se čítač. Pokud čítač dopočítá do své maximální hodnoty TIMx_CNT registru (V případě použitého 16-bitového čítače je to hodnota 65535), znamená to, že vstupní signál je příliš pomalý a FW zvýší před-děličku o jedna a měření se vyresetuje. Pokud změřená doba periody je naopak po měření nižší než půlka maximální zmíněné

hodnoty, tak to znamená, že můžeme hodnotu předděličky vydělit dvěma a další změřená perioda by se stále měla vejít do rozsahu 16 bitového TIMx_CNT registru.

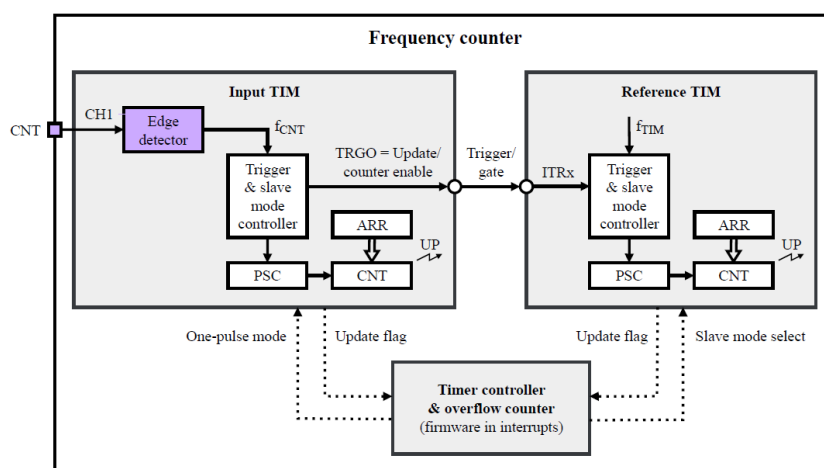
Pro přenos hodnot zaznamenaných do registrů TIMx_CCR1 a TIMx_CCR2 je použito DMA řadiče. Ten zaznamenává 10 po sobě jdoucích hodnot do 2 kruhových bufferů. Jeden pro hodnoty registrů TIMx_CCR1 (hodnoty pro určení střídy) a druhý pro hodnoty registrů TIMx_CCR2 (Hodnoty pro určení délky periody). Na straně MCU se pak spočítají aritmetické průměry hodnot uložených ve zmíněných kruhových bufferech CCR1 a CCR2, které jsou následně poslány do PC aplikace společně s frekvencí vnitřních hodin čítače f_{TIM} , čímž označují frekvenci vstupních hodin čítače f_{PCLK} vydělenou hodnotou předděličky PSC. Frekvence vstupního signálu f_{in} a střída D je pak interpretována na straně PC aplikace. Použitím vzorců:

$$f_{in} = \frac{f_{TIM}}{CCR1}, \quad D = \frac{CCR2}{CCR1} \cdot 100\% \quad (3.15)$$

3.7.2 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 problému autor v práci [2] navrhl způsob, kterým se snaží toto omezení řešit.

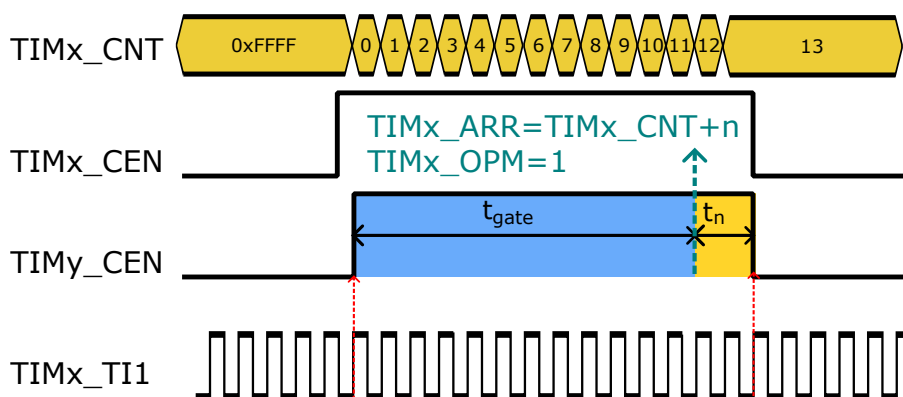
Řešení spočívá v propojení dvou čítačů. Jako u obvyklého řešení první z čítačů počítá vstupní pulzy (Vstupní čítač) a druhý čítač odměřuje dobu, která uběhla, pomocí interních vstupních hodin čítače PCLK (Referenční čítač). Řídící jednotky čítačů jsou vzájemně propojené tak, aby po spuštění měření, první 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. Schematické propojení čítačů je vyobrazeno na obrázku 3.35.



Obr. 3.35: Diagram propojení čítačů pro upravenou metodu měření frekvence čítáním pulsů. Převzato z [2]

Časový diagram měření je znázorněn na obrázku 3.36 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. 3.36: Časový diagram čítání pulzů za proměnnou dobu odběru

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

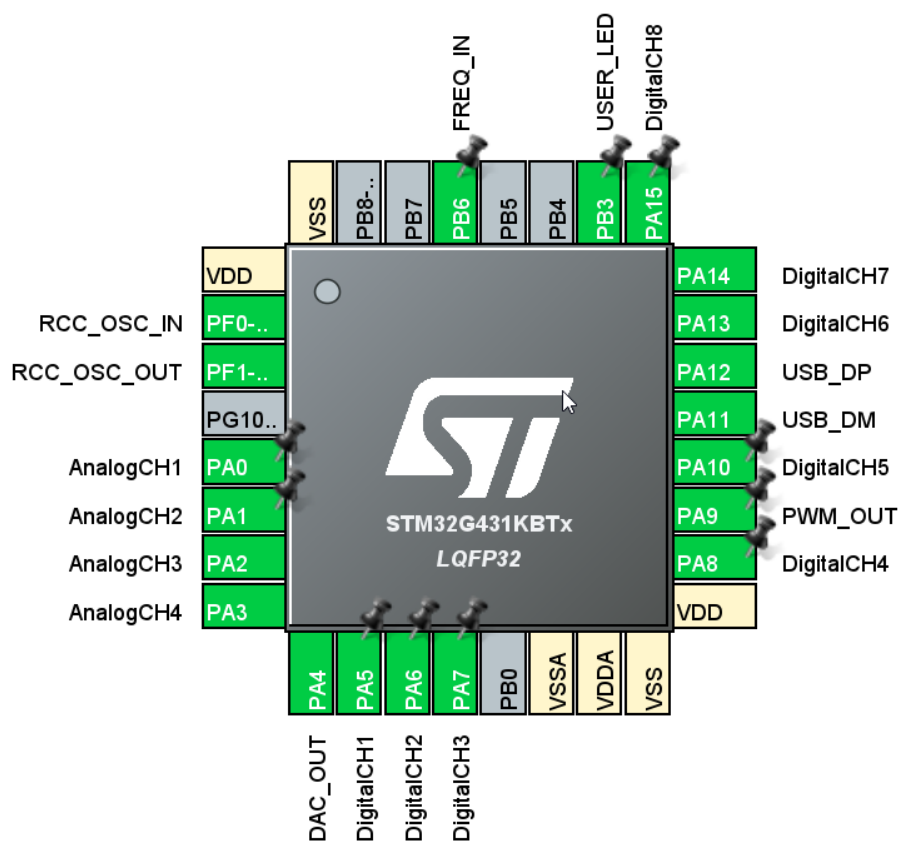
$$f_{in} = \frac{N_p}{T_{meas}} \quad (3.16)$$

Maximální frekvence vstupního signálu je limitována vnitřním synchronizačním blokem čítače, který má za úkol synchronizovat externí signál s vnitřními hodinami čítače. Dle [9] musí být frekvence vstupního signálu alespoň 3krát menší než frekvence vnitřních hodin čítače $f_{TIM} = 156$ MHz. A tedy v rámci mé implementace můžeme měřit frekvenci externích signálů s rozsahem frekvencí 1 Hz až 52 MHz.

Kapitola 4

Ověření funkčnosti přístroje a možností jeho využití

Vzniklý FW měl spolupracovat s PC aplikací, která se již používá společně s STM32F042 v předmětu laboratoře průmyslové elektroniky (LPE), vyučovanému na ČVUT FEL a bylo tedy zapotřebí ověřit funkčnost všech implementovaných modulů. Funkčnost nového firmwaru byla testována studenty již v průběhu letního semestru 2022/23, kdy tato práce vznikala. Současně funkci všech částí FW demonstrují v této kapitole prostřednictvím reálných záznamů běhu PC aplikace spolupracující s mým FW běžícím na MCU STM32G431. Přehled vstupních a výstupních pinů pro zapojení na je zobrazen na obrázku 4.1 .



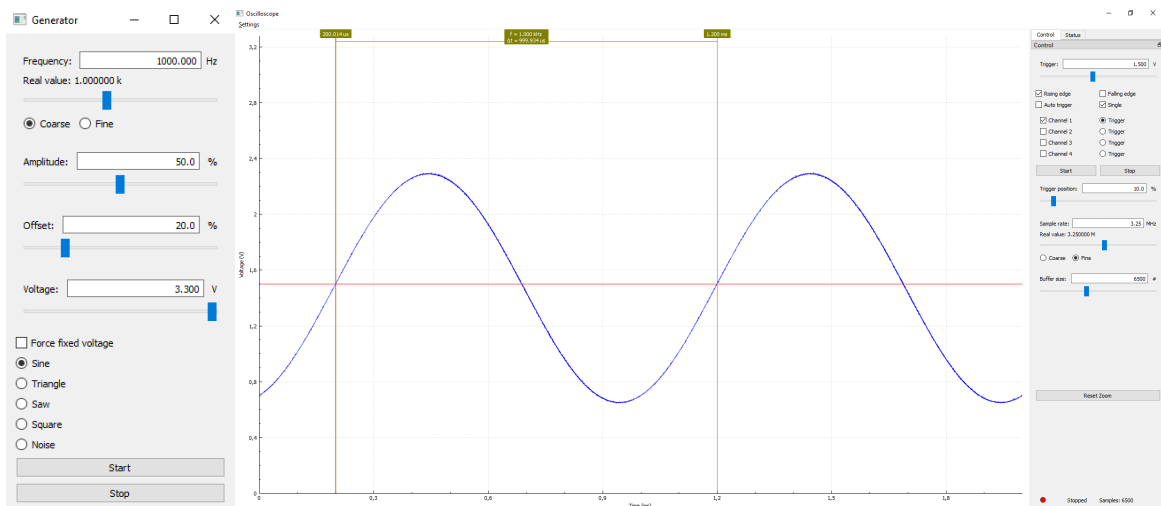
Obr. 4.1: Pinout vzniklého SDI

4.1 Signálový generátor

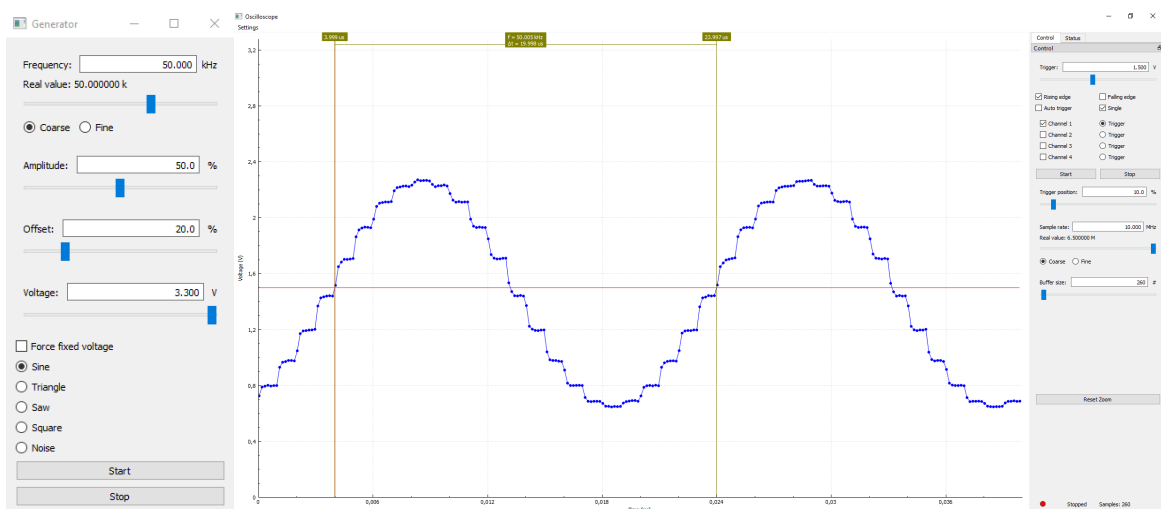
Modul signálového generátoru umožňuje generovat signál s výstupní frekvencí 1 Hz až 250 kHz s tím, že maximálního počtu vzorků na periodu může být dosaženo akorát pro frekvence do 1000 Hz kvůli omezené maximální vzorkovací frekvenci interního DAC převodníku. Je také potřeba počítat s omezenou rychlostí přeběhu výstupního zesilovače a pro signály s vysokými výstupními frekvencemi zvážit nižší rozkmit. Pomocí generátoru lze generovat obdélníkový signál se střídou 50%, sinusový signál, trojúhelníkový signál a také signál tvaru pila. Dále modul umožňuje generování pseudo-šumového signálu a stejnosměrné napětí nastavitelné úrovně.

4.1.1 Vliv počtu vzorků na 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 $f_{\text{SIG}} = 1000$ Hz s 1000 vzorky na 1 periodu vzorkovaný rychlostí 3.25 Msp/s



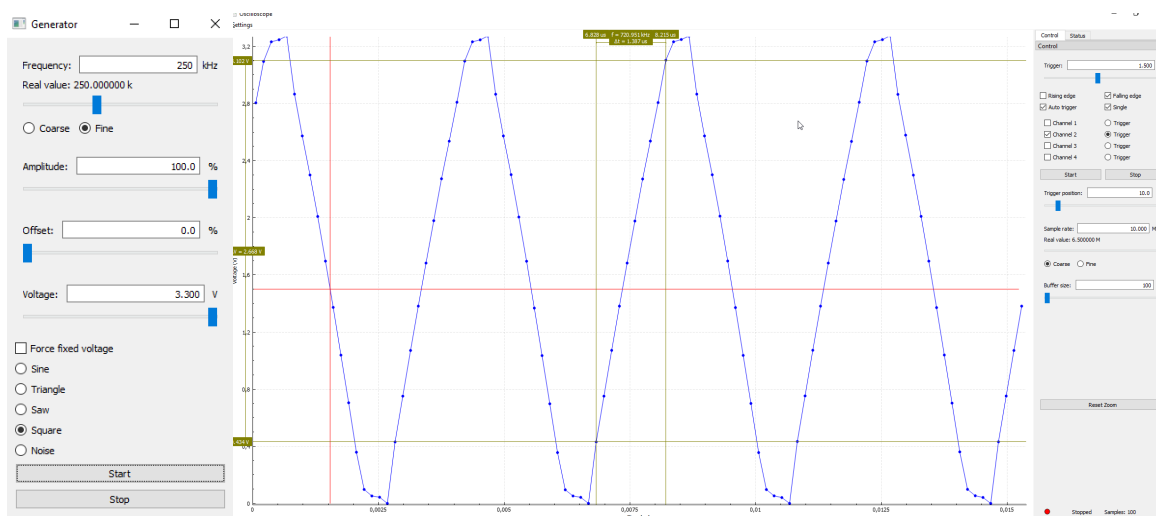
Obr. 4.3: Záznam signálu sinus o frekvenci 50 kHz s 20 vzorky na 1 periodu vzorkovaný rychlostí 6.5 Msp/s

4.1.2 Rychlost přeběhu výstupního zesilovače DAC převodníku

Maximální použitelná výstupní frekvence generátoru souvisí s omezenou rychlostí přeběhu (SR) výstupu DAC převodníku. Na obrázku 4.4 je vidět záznam měření při generaci obdélníkového signálu s frekvencí 250 kHz (tedy pouze 4 vzorky na periodu), kde byl zvolen plný rozkmit výstupního signálu. Z obrázku je zřejmá omezená rychlost přeběhu, kterou můžeme rovnou změřit pomocí v aplikaci integrovaných kurzorů a dopočítat dle:

$$SR = \frac{\Delta V}{\Delta t} = \frac{2.668 \text{ V}}{1.387 \mu\text{s}} = 1.92 \text{ V}/\mu\text{s} \quad (4.1)$$

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 volbu nižšího rozkmitu, aby signál nebyl tolik zkreslený.



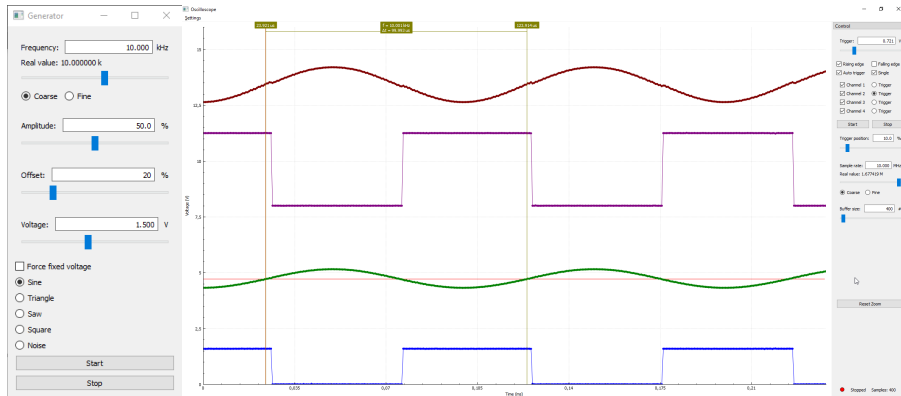
Obr. 4.4: Záznam obdélníkového signálu, $f_{SIG}=250$ kHz (4 vzorky na periodu) vzorkovaný rychlostí 6.5 MSps. Pozorování omezené rychlosti přeběhu výstupního zesilovače DAC převodníku

4.2 Modul osciloskopu

Modul osciloskopu využívá pro určení stavu jednotlivých analogových kanálů až 2 ADC převodníky zároveň. Mezi jednotlivými režimy měření jako jsou režimy dual simultaneous, independent interleaved nebo například režim využívající pouze jednoho ADC s multiplexováním vstupních kanálů, FW sám vybírá nejvhodnější konfiguraci v závislosti dostupnosti jednotlivých kanálů na ADC převodnících. Při vzorkování v reálném čase může být zvolena maximální vzorkovací frekvence až $f_{SAMP} = 6.5$ MS/s při zvolení pouze vzorkování pouze 1 analogového kanálu a to kanálu CH1 nebo CH2, které jsou oba dostupné z obou ADC převodníků. Maximální vzorkovací frekvence pro všechny 4 zvolené kanály zároveň je 1.677 Ms/s.

4.2.1 Ověření funkce v běžném režimu vzorkování v reálném čase

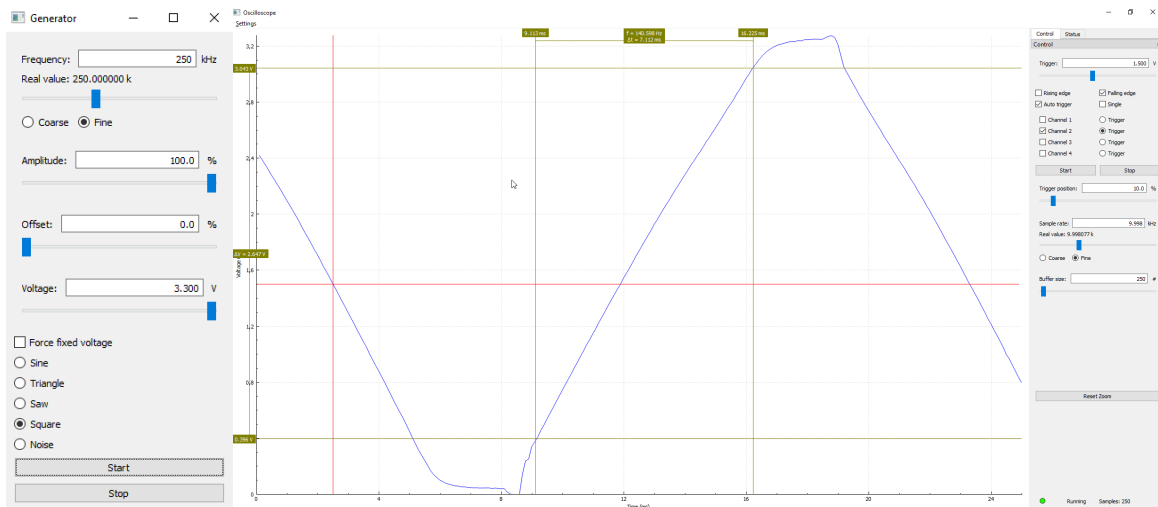
Pro jednoduché ověření funkce osciloskopu jsem na vstupy dvou napěťových děličů přivedl signál z mých dvou dostupných modulů generátorů. Na vstup prvního z děličů byl přiveden sinusový signál s frekvencí 10 kHz generovaný pomocí signálového generátoru a na druhý dělič byl přiveden signál z impulzního generátoru, také o frekvenci 10 kHz. Osciloskopem jsem pak měřil zároveň vstupy a výstupy zmíněných napěťových děličů. Záznam osciloskopu na obrázku 4.5 byl triggerován kanálem číslo 2 (zelený). Změřené průběhy odpovídaly očekávání. Správnost nastavení vzorkovací frekvence bylo ověřeno měřením délky periody vstupních signálů pomocí kurzorů.



Obr. 4.5: Vzorkování 4 analogových kanálů s vzorkovací frekvencí $f_{\text{SAMP}} = 1.677 \text{ MSps}$

4.2.2 Ověření funkce v režimu ETS

Pro některé situace maximální vzorkovací frekvence, při měření osciloskopem v reálném čase, není dostatečná. Nicméně pokud měříme periodický signál se známou frekvencí, můžeme použít režim vzorkování v ekvivalentním čase (ETS). Jako příklad uvádím měření průběhu výstupního zesilovače, který jsme v kapitole 4.1.2 měřili vzorkováním v reálném čase, ale s použitím ETS můžeme tento periodický signál změřit s vyšší vzorkovací frekvencí a to až 52 MSps. Na obrázku 4.6 je opět vidět záznam obdélníkového signálu s frekvencí $f_{\text{SIG}} = 250 \text{ kHz}$ generovaného modulem signálového generátoru s odměřením doby průběhu ΔT_{SAMP} a rozdílem napětí ΔV .



Obr. 4.6: Záznam signálu obdélník s $f_{\text{SIG}} = 250 \text{ kHz}$ vzorkovaný osciloskopem v ETS režimu rychlostí 52 MSps

Pro výpočet SR je třeba hodnotu ΔT_{SAMP} nejdříve přepočítat koeficientem (k_{AEQ}), abychom získali reálnou dobu průběhu ΔT_{SEQ} . Koeficient (k_{AEQ}) je závislý na frekvenci signálu $f_{\text{SIG}} = 250 \text{ kHz}$,

frekvenci vzorkování $f_{\text{SAMP}} = 9998.077 \text{ Hz}$ a celočíselným koeficientu k_s .

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

$$k_{\text{AEQ}} = \frac{250 \text{ kHz}}{250 \text{ kHz} - 25 \cdot 9.998 \text{ kHz}} = 5200.208, \quad \Delta T_{\text{SEQ}} = \frac{7.112 \text{ ms}}{5200.208} = 1.367 \mu\text{s} \quad (4.3)$$

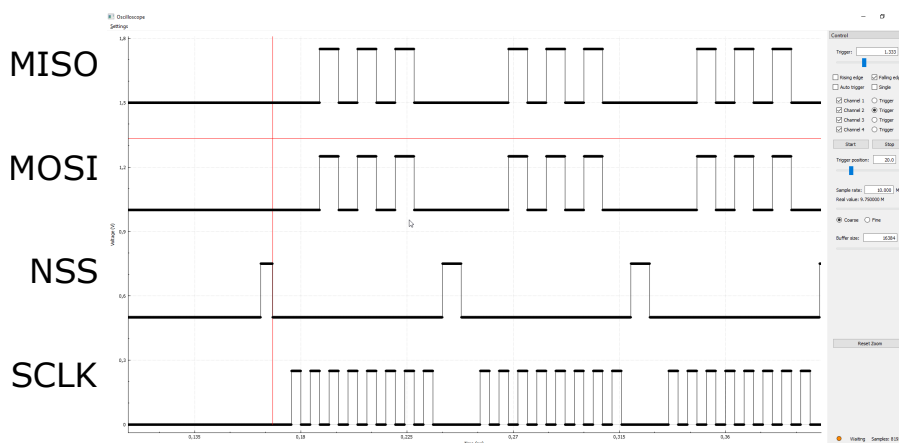
Pak již můžeme dosadit do vzorce pro spočítání rychlosti přeběhu SR:

$$\text{SR} = \frac{\Delta V}{\Delta T_{\text{SEQ}}} = \frac{2.647 \text{ V}}{1.367 \mu\text{s}} = 1.93 \text{ V}/\mu\text{s} \quad (4.4)$$

Takto změřená a vypočtená doba přeběhu vyšla stejně jako doba přeběhu změřená při vzorkování v reálném čase, nicméně jsme ukázali využitelnost ETS a jakým způsobem se dá s takto změřeným průběhem pracovat.

4.3 Ověření funkce modulu logického analyzátoru

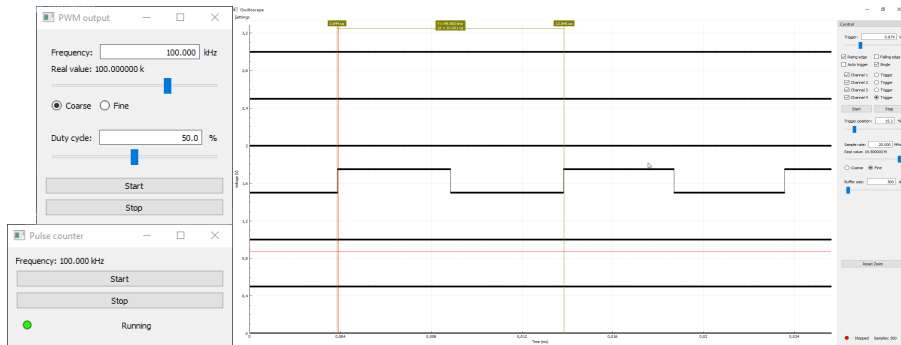
Funkčnost správné funkce logického analyzátoru jsem ověřil záznamem SPI komunikace mezi 2 deskami s rychlostí komunikace 125 kBit/s. Záznam byl úspěšně zatriggrován vstupním kanálem zaznamenávajícím NSS signál SPI sběrnice ohraničující přenos jednotlivých bytů. Ze záznamu můžeme pohodlně dekodovat první byty probíhající komunikace.



Obr. 4.7: Záznam SPI komunikace s rychlostí baudrate 125 kBit/s triggerován NSS signálem sběrnice

4.4 Ověření funkce modulů čítače a impulzního generátoru

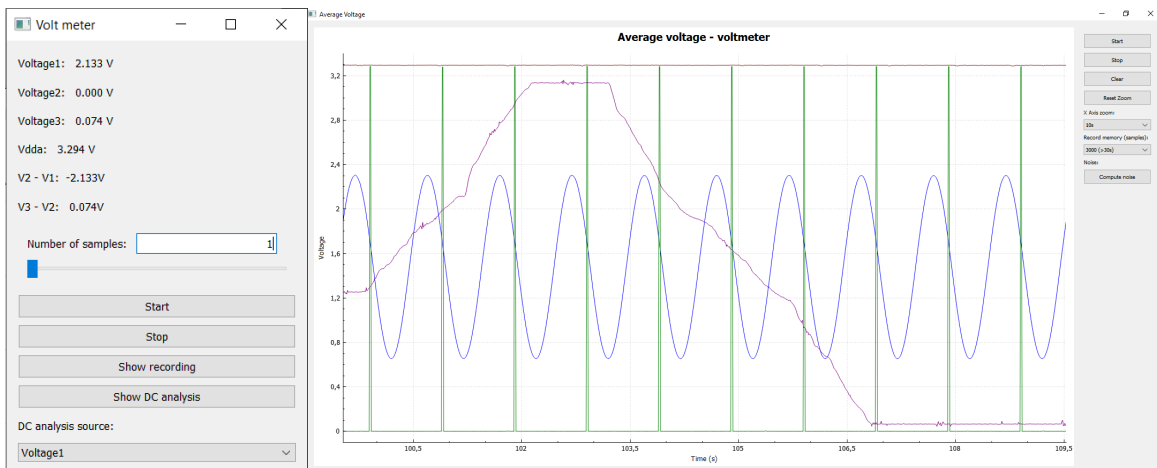
Funkci čítače jsem ověřil pouhým přivedením PWM signálu o frekvenci 100 kHz na vstup čítače v režimu počítání pulzů (pulse counter). Pro kontrolu jsem ještě frekvenci výstupního signálu ověřil odměřením délky periody pomocí kurzorů v modulu logického analyzátoru. Na obrázku 4.8 je vidět, že nastavená frekvence na impulzním generátoru 100 kHz, byla změřena modulem čítače a též v modulu logického analyzátoru délka odměřené periody vstupního signálu odpovídala vstupnímu signálu o frekvenci 100 kHz.



Obr. 4.8: Záznam PC aplikace při generování PWM signálu impulzním generátorem a ověření výstupní frekvencí modulem čítače a logického analyzátoru

4.5 Ověření funkčnosti modulu voltmetru se záznamem

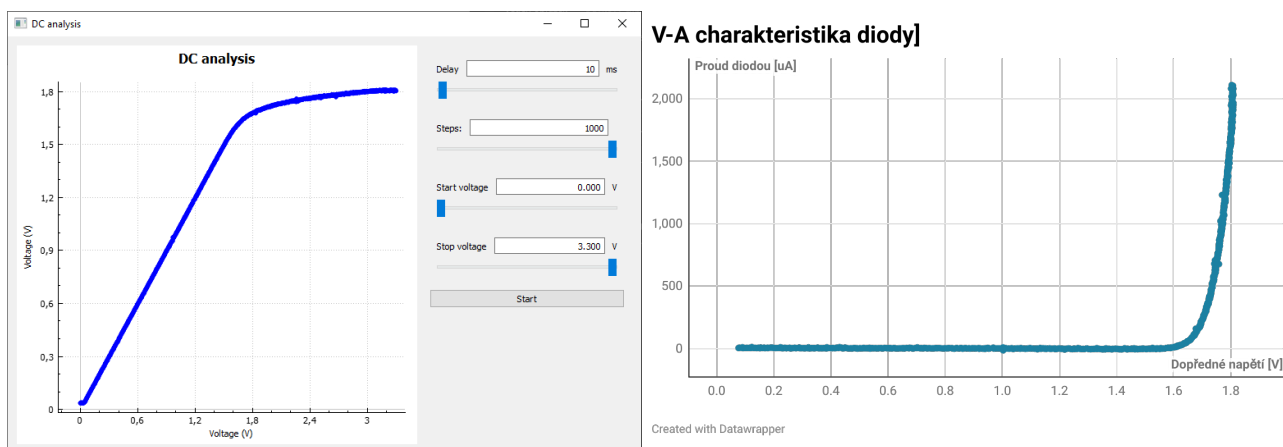
Funkce záznamu voltmetru je vhodná pro pozorování pomalých signálů. Na rozdíl od modulu osciloskopu neumožňuje triggerování záznamu, ale jen zachycuje odměřená data v plynoucím čase. Na obrázku 4.9 je vidět záznam voltmetru, který má na vstupech kanálu 1 a 2 pomalé signály generované vnitřními generátory a na třetím kanálu je záznam napětí měněného ručně pomocí potenciometru.



Obr. 4.9: Záznam voltmetru. Na kanálu č.1 signál sinus o frekvenci 1 Hz, Na kanálu č.2 PWM o frekvenci 2 Hz, Kanál č. 3 signál z potenciometru

4.5.1 Využití možnosti měření DC charakteristiky

Modul voltmetru ve spolupráci s modulem generátoru signálu, který je využit pro nastavování úrovně stejnosměrného napětí, umožňuje stanovení stejnosměrné převodní charakteristiky. Měření spočívá v postupném zvyšování napětí na výstupu generátoru a zaznamenávání změn napětí na vstupu voltmetru. Na obrázku 4.10 je vidět záznam takového měření pro určení charakteristiky LED diody. Zde na sériové spojení rezistoru a LED diody přivádíme napětí z generátoru a měříme dopředné napětí na LED diodě. Výsledná změřená data, pak mohou být uložena v csv souboru a následně zpracována pro určení velikosti proudu tekoucí LED diodou při daném dopředném napětí.



Obr. 4.10: Využití funkce DC charakteristiky pro zpracování VA charakteristiky LED diody

Kapitola 5

Zhodnocení dosažených výsledků práce

V rámci této práce byl navržen přístroj s STM32G431 dle zadání. Jeho výsledné parametry, které byly ověřeny, jsou shrnuty v této kapitole.

Základní vlastnosti přístroje

- Frekvence systémových hodin a hodin čítačů - 156 MHz
- Frekvence hodin ADC převodníků - 52 MHz
- Komunikace s PC pomocí periferie USB
- Možnost funkce pouze interním oscilátorem HSI nebo volitelně možnost připojit krystal s frekvencí 8 , 12 nebo 16 MHz

Vlastnosti jednotlivých přístrojových funkcí

- **Čítač - měření frekvence**
 - dostupné 2 režimy měření frekvence vstupního signálu.
 - Režim čítání pulzů vhodný pro měření frekvencí 1Hz až 52 MHz
 - Režim PWM input umožňuje kromě stanovení frekvence měřením délky periody také měření střídy vstupního signálu.
- **Impulsní generátor PWM**
 - Nastavení střídy 0 až 100 %
 - Nastavení frekvence 1 Hz až 78 MHz
- **Voltmetr**
 - Tři vstupní kanály
 - Rozmezí vstupních napětí 0 až + 3,3 V,
 - Vzorkovací frekvence 100 odměrů/s,
 - Průměrování z 1 až 256 odměrů
 - Možnost funkce záznamu a měření DC charakteristiky
- **Osciloskop**
 - 4 kanály, záznam až 1x8192 vzorků(nebo 2x 4096, 4x 2048), rozlišení 12 bitů
 - Rychlost záznamu až 1x 6.5 MSps (nebo 2x 3.25 MSps, 4 x 1.677 kSps)
 - Ve stroboskopickém módu - až 52 MSps - 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

Maximální vzorkovací frekvence 6.5 MS/s může být použita pouze na jednom z kanálů CH1 nebo CH2 a je jí dosaženo díky interleaved režimu.

Vzorkování 2 kanálů zároveň funguje u všech kombinací 2 kanálů, kromě kanálů CH3 a CH4, které musí být vzorkovány v režimu multiplexování vstupního kanálu, z čehož plyne určité zpoždění kanálu CH4 za kanálem CH3. Velikost zpoždění je dáno dobou vzorkování a konverze CH1.

Při vzorkování 3 nebo 4 kanálů se kanály CH2 a CH4 zpožďují za kanály CH1 a CH3.

Při zvolené funkci auto-trigger a zvolené nějaké hrany triggeru dojde k vypsání bufferu, pokud nedojde k triggeru za 10 načtení celé délky bufferu

Při zvolené funkci auto-trigger bez zvolené hrany triggeru dojde k okamžitému vykreslování bufferu

■ Logický analyzátor

- 8 kanálů, 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 zvoleným kanálem na náběžnou hranu, sestupnou hranu nebo obě zároveň
- Možnost funkce pretrigger zadávaného v % počtu vzorků
- Možnost auto-trigger

Na vstupních pinech logických kanálů jsou defaultně nastaveny interní pull-down rezistory pro stanovení pevné úrovně v případě nepřipojení vstupů k žádnému zdroji signálu.

Jako zdroj triggeru může být použit jeden ze vstupních signálů na kanálu CH1 až CH4

Při zvolené funkci auto-trigger a zvolené nějaké hrany triggeru dojde k vypsání bufferu, pokud nedojde k triggeru za 10 načtení celé délky bufferu

Při zvolené funkci auto-trigger bez zvolené hrany triggeru dojde k okamžitému vykreslování bufferu

■ 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, trojúhelník, pila, obdélník
- Nastavitelný rozkmit a offset v procentech napájecího napětí
- Možnost generování pseudošumu s nastavitelným rozkmitem a offsetem
- Možnost generování pevného napětí v rozsahu 0-3.3V

V závislosti na zvolení výstupní frekvenci generovaného signálu je potřeba uvážit snížení použitého rozkmitu kvůli omezené rychlosti přeběhu $SR = 1.92 \text{ V}/\mu\text{s}$

Generátor optimalizuje počet vzorků na periodu signálu pro dosažení výstupní frekvenci generovaného signálu, která je blíže zadané hodnotě

Kapitola 6

Závěr

Úkolem této práce bylo navrhnout a 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 nutné analyzovat protokol, který zmíněná PC aplikace používá pro komunikaci s SDI a dále pak implementovat funkce jednotlivých přístrojových modulů. Vytvořený přístroj SDI emuluje funkce osciloskopu a logického analyzátoru pro snímání a záznam logických i analogových signálů a také umožňuje generaci signálů pomocí modulů impulsního a signálového generátoru. Dále SDI obsahuje funkci čítače pro měření frekvence signálu a voltmetr se záznamem pro měření pomalých průběhů.

Podrobné vlastnosti a parametry vzniklého přístroje jsou uvedeny v předchozí kapitole 5. Funkčnost přístroje byla již ověřena používáním ve výuce předmětu B3B38LPE1 na ČVUT FEL již během letního semestru 2023. Zkušenosti ukazují, že tento přístroj by mohl být využit nejen v tomto předmětu, ale i ve výuce dalších předmětů na ČVUT FEL, případně i na středních školách technického zaměření. Domnívám se, že úkoly vyplývající ze zadání této práce byly splněny v plném rozsahu.

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. 2023-05-09].
- [4] FISCHER, J. Lpe 2023 - laboratoře průmyslové elektroniky v roce 2023. [online], 2022. [cit. 2023-05-22].
- [5] HLADIK, J. Single chip software defined instrumentation for educational purposes. [online], 05 2017. [cit. 2023-05-08].
- [6] RIGOL TECHNOLOGIES, I. Datasheetds1000e, ds1000d series digital oscilloscopes. [online], 12 2015. [cit. 2023-04-13].
- [7] STMICROELECTRONICS. Stm32g4 extended interrupts and events controller. [online]. [cit. 2023-03-18].
- [8] STMICROELECTRONICS. Reference manual RM0440 stm32g4 series advanced arm®-based 32-bit mcus. [online], 1 2017. [cit. 2023-04-30].
- [9] STMICROELECTRONICS. Application note AN4776 general-purpose timer cookbook for STM32 microcontrollers. [online], 7 2019. [cit. 2023-02-05].
- [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. [online], 10 2021. [cit. 2023-05-06].
- [12] STMICROELECTRONICS. An2548 using the stm32f0/f1/f3/cx/gx/lx series dma controller. [online], 12 2022. [cit. 2020-05-21].
- [13] STMICROELECTRONICS. Application note AN2834 how to get the best adc accuracy in stm32 microcontrollers. [online], 01 2022. [cit. 2023-05-02].
- [14] STMICROELECTRONICS. How to use the cordic to perform mathematical functions on stm32 mcus. [online], 02 2023. [cit. 2023-05-05].