České vysoké učení technické v Praze
Fakulta elektrotechnická

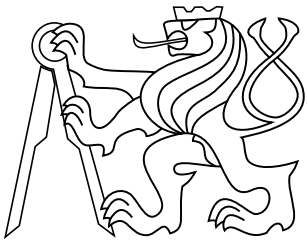
DIPLOMOVÁ PRÁCE

2016 Bc. Adam Berlinger

diplomová práce

Implementace přístrojových funkcí s využitím mikrořadičů STM32

Bc. Adam Berlinger



Květen 2016

Vedoucí práce: doc. Ing. Jan Fischer, CSc.

České vysoké učení technické v Praze
Fakulta elektrotechnická

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Adam Berlinger**

Studijní program: Otevřená informatika
Obor: Počítačové inženýrství

Název tématu: **Implementace přístrojových funkcí s využitím mikrořadičů STM32**

Pokyny pro vypracování:

1. Navrhněte metodiku implementace přístrojových funkcí do mikrořadičů řady STM32 s využitím jejich periferních bloků. Stanovte meze takto implementovaných funkcí.
2. Realizujte přístrojové funkce typu osciloskop, logický analyzátor, generátor impulsů, a funkce typu voltmetr, čítač, zdroj měřicích napětí s využitím mikrořadičů řady STM32 s ohledem na dosažení optimálních parametrů z hlediska rychlosti a přesnosti. Navržené funkce musí být možno využívat jako stavební bloky při návrhu specializovaných přístrojů.
3. Vytvořte vzorový laboratorní přístroj typu impulsní generátor, čítač, logický analyzátor a přístroj typu osciloskop, generátor napětí, vícekanálový voltmetr. Vytvořte též potřebné aplikační programy pro nadřazené PC.

Seznam odborné literatury:

[1] Yiu, J.: Definitive Guide to ARMR CortexR-M3 and CortexR-M4 Processors
[2] STMicroelectronics: RM0316 Reference manual,
[3] STMicroelectronics: DS10362, STM32F30 Data

Vedoucí: doc. Jan Fischer Ing., CSc.

Platnost zadání: do konce letního semestru 2016/2017

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 1. 2. 2016

Poděkování

Tímto bych chtěl srdečně poděkovat doc. Ing. Janu Fischerovi, CSc. za všechen čas, který mi věnoval během konzultací, a odborné rady.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne ....

Abstrakt

Tato práce se zabývá využitím mikrořadičů řady STM32 s jádry ARM Cortex™ M pro realizaci přístrojových funkcí. Jsou realizovány funkce: osciloskop, generátor signálů, voltmetr, zdroj napětí a čítačové funkce. Výsledkem práce je firmware pro několik zařízení/kitů a zobrazovací software pro PC. Firmware komunikuje s PC aplikací pomocí rozhraní USB (Virtual COM port), nebo pomocí rozhraní UART skrze převodník Sériová linka/USB. Aplikace pro PC je napsána v jazyce C/C++ s pomocí frameworku Qt.

Klíčová slova

Osciloskop; Voltmetr; ARM Cortex™ M; STM32; Systém sběru dat; Vestavné přístroje; Qt framework

Abstract

This work describes use of microcontrollers STM32 series, containing ARM Cortex™ M cores, for implementing instrument functions. Osciloscope, waveform generator, voltmeter, voltage source and timer functions are implemented. The result of this work is firmware various devices/kits and simple software for viewing measured data on PC. The firmware comunicates with PC application using USB interface (Virtual COM port), or using UART interface through Serial-to-USB converter. The PC aplication is written in C/C++ language using Qt framework.

Keywords

Oscilloscope; Voltmeter; ARM Cortex™ M; STM32; Data aquisition system; Embedded instruments; Qt framework

Obsah

Zkratky

Zde se nacházejí použité zkratky a jejich stručný význam/překlad

ARM	Advanced RISC Machines — architektura procesorů/mikrořadičů a zároveň jméno firmy, která ji vyvíjí
DDS	Direct digital synthesis — algoritmus pro generování signálů
GPIO	General propouse input/output — vstupní/výstupní piny mikrořadiče, jejichž logickou úroveň lze přímo programově ovládat
HAL	Hardware abstraction layer — vrstva softwaru, která realizuje komunikaci s různým hardwarem pomocí jednotného rozhraní
NVIC	Nested-vector Interrupt Controller — řadič přerušení architektury ARM
OC	Output compare — jednotka čítače která porovnává aktuální hodnotu čítače s předem stanovenou konstantou
PC	Personal computer — osobní počítač
PWM	Pulse width modulation — pulzně šířková modulace
RAM	Random-access memory — operační paměť
CCM	Core-coupled memory — rychlá operační paměť, ke které obvykle přistupuje pouze jádro procesoru/mikrořadiče
SIMD	Single instruction multiple data — instrukce procesoru/mikrořadiče, které jsou schopné zpracovat více dat naráz
USB	Universal serial bus — sériová sběrnice pro přenos dat

1. Úvod

V elektrotechnice se často používá řada měřících a budících přístrojů. Využívají se při vývoji elektroniky, testování, či při výuce. S vývojem moderních mikrořadičů je možné tyto funkce realizovat s využitím vnitřních periferií mikrořadiče, případně s minimálním počtem externích součástek.

Cílem této práce je navrhnout metodiku realizace přístrojových funkcí na mikrořadiči a dále vytvořit vzorové laboratorní přístroje s využitím mikrořadičů řady STM32. Výsledkem je tedy firmware pro řadiče řady STM32 (konkrétně STM32F303 a STM32F042), který je snadno rozšiřitelný o další měřící přístroje a cílové platformy. Dále je potřeba vytvořit ovládací aplikaci pro počítač, která bude pomocí komunikačního rozhraní ovládat jednotlivé přístroje, číst a zobrazovat naměřená data.

2. Rozbor

Cílem této práce je navrhnout metodiku implementace přístrojových funkcí pomocí mikrořadičů a realizovat jednotlivé měřící funkce jako syntetizovatelné bloky pro mikrořadiče STM32. Současné měřící přístroje mají obvykle špičkové parametry, ale jsou poměrně drahé a obvykle realizují pouze jednu konkrétní funkci. Současné mikrořadiče mají již sofistikované periferie (A/D převodníky, čítače apod.), které umožňují implementovat tyto funkce přímo na čipu mikrořadiče.

Takto implementované funkce nedosahují kvalit profesionálních a specializovaných přístrojů, ale pro mnohé aplikace jsou jejich parametry dostačující. Výhodou může být možnost zabudovat mikrořadič přímo do jiného zařízení za účelem diagnostiky a testování. Další výhodou je možnost kombinovat jednotlivé měřící moduly a provádět tak složitější měření. Například kombinací zdroje napětí a voltmetru jsme schopni měřit přechodové děje nebo převodní charakteristiky elektrických obvodů.

Poslední výhodou je pochopitelně cena. Profesionální měřící přístroje jsou obvykle velmi drahé a pro jednotlivce (např. studenta či "nadšence") často cenově nedostupné. Naopak mikrořadiče jsou obvykle levné a mnoho výrobců nabízí poměrně levné vývojové desky, které již obsahují externí součástky nutné pro naprogramování čipu, případně i pro připojení k počítači. Dále jsou také volně dostupná vývojová a ladící prostředí, ať už v nějaké omezené verzi (např. Keil IDE[1]), či se jedná o Open-Source řešení (např. GCC[2], OpenOCD apod.). Nabízí se tak možnost využít takovéto přístroje ve výuce při laboratorních měřeních, nebo jako přístroje, které budou mít studenti k dispozici při řešení návrhu nějakého elektrického obvodu.

Proto v rámci této práce implementuji jak firmware pro mikrořadiče STM32F3 a STM32F0 na vývojových kitech Nucleo, tak PC aplikaci, která bude naměřená data zobrazovat.

Některé funkce je možné dnes realizovat pomocí specializovaných čipů. Příkladem může být generování periodických signálů pomocí algoritmu DDS. Případně lze funkce realizovat pomocí vestavěných modulů. Příkladem můžou být moduly firmy National Instruments [3].

2.1. Rozbor statických funkcí

2.1.1. Voltmetr

Funkce voltmetr slouží k měření elektrického napětí. Předpokládáme, že měřené napětí je stabilní a nemění se v čase. Důraz je kladen na přesnost změřeného napětí. Obvykle se výsledné napětí bere jako průměr z několika vzorků, aby byl potlačen vliv šumu.

2.1.2. Zdroj napětí

Funkce zdroj napětí slouží ke generování předem stanoveného napětí na výstupu. Zdroj napětí společně s voltmetrem můžeme použít k měření převodních charakteristik elek-

[1] Vývojové prostředí od fimy ARM
[2] GNU Compiler Collection — Kompilátor podporující mnoho různých procesorových architektur
[3] `http://www.ni.com/compactrio`

trických obvodů. Důležitým parametrem zdroje napětí je maximální proudový odběr a vnitřní impedance zdroje.

2.2. Rozbor dynamických funkcí

2.2.1. Osciloskop

Funkce osciloskop slouží ke sledování průběhu elektrického napětí. Při použití osciloskopu obvykle nepotřebujeme přesnou aktuální hodnotu napětí, ale především relativní změny (tvar) signálu.

Vzorkování

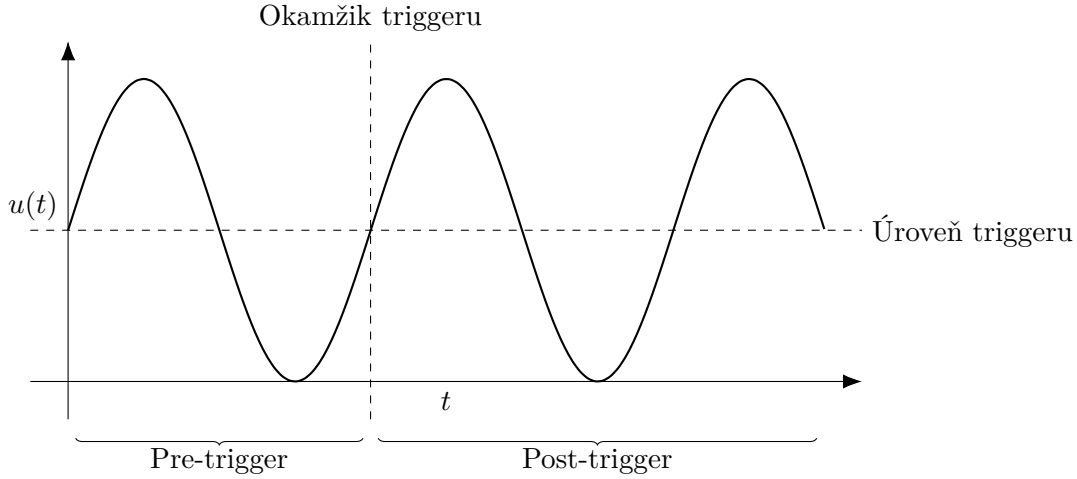
Základním parametrem funkce je vzorkovací frekvence (f_S). Z toho pak vyplývá jak rychlé signály jsme schopni sledovat. Vzorkovací teorém nám určuje maximální frekvenci, kterou jsme schopni měřit:

$$f_{MAX} < \frac{f_S}{2} \tag{2.2.1}$$

Toto omezení znamená, že potřebujeme více jak 2 vzorky na periodu signálu. Abychom ale byly schopni signál analyzovat obvykle potřebujeme více vzorků na periodu.

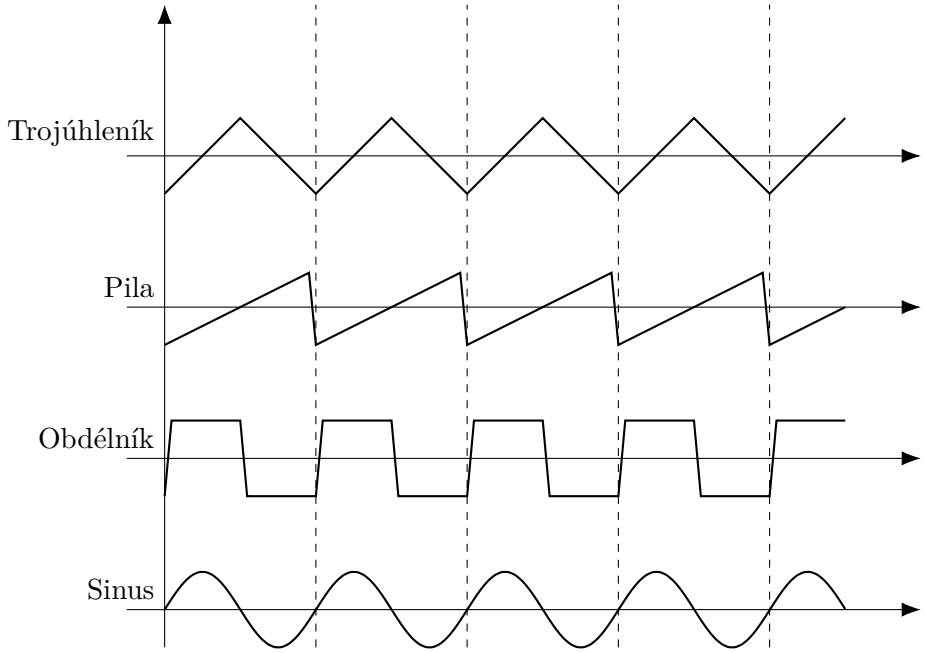
Trigger

Často potřebujeme sledovat průběh signál v nějakém konkrétním úseku, či v reakci na nějakou událost. K tomu slouží tzv. "trigger" (spoušť), který umožňuje zaznamenávat signál v okolí náběžné či sestupné hrany měřeného signálu. Případně může trigger reagovat na hranu odděleného (jiného) digitálního signálu.



2.2.2. Logický analyzátor

Funkce logický analyzátor slouží ke sledování průběhu několika digitálních signálů zároveň. Je to de-facto obdoba osciloskopu pro digitální signály. Velmi často profesionální osciloskopy jsou spojené s logickými analyzátory. Podobně jako u osciloskopu i zde platí vzorkovací teorém (2.2.1). Logický analyzátor se obvykle používá při sledování digitálních komunikačních rozhraní a sběrnic.



Obr. 2.2.1. Průběhy generovaných signálů

2.2.3. Čítač pulsů

Funkce čítač pulsů slouží ke sledování parametrů digitálních signálů. Měříme především frekvenci, střídu, počet pulsů a délku pulsu. Tyto veličiny spolu souvisí a např. z počtu pulsů jsme schopni dopočítat frekvenci.

2.2.4. Generátor pulsů

Funkce generátor pulsů slouží ke generování digitálních signálů:

- PWM, kde nastavujeme frekvenci signálu a střídu ("Duty-cycle").
- Předem stanovený počet pulsů o dané šířce a frekvenci.

Generátor pulsů může sloužit např. pro řízení nějakých akčních členů pomocí PWM, případně pro testování digitálních obvodů jako hodinový signál.

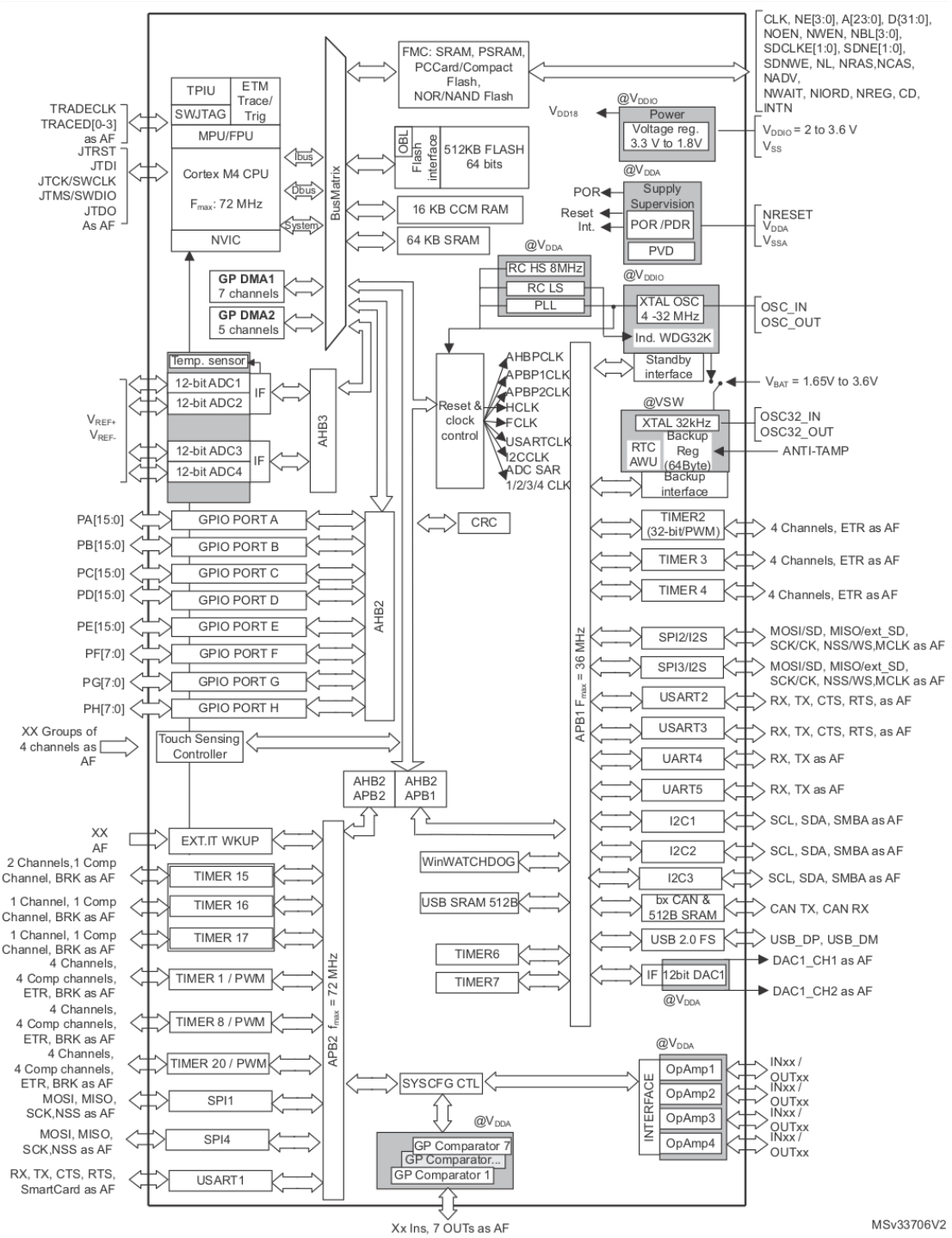
2.2.5. Generátor signálu

Funkce generátor signálu slouží ke generování předem stanoveného periodického průběhu napětí o dané frekvenci. Tímto signálem je obvykle: sinus, obdélník, trojúhelník, "pila", nebo šum (viz Obr. 2.2.1).

2.3. Možnosti mikrořadičů STM32

2.3.1. Jádro ARM Cortex-M

Mikrořadiče STM32 obsahují jádra s architekturou ARM Cortex-M0/M3/M4/M7. Součástí jádra je systémový časovač SysTick a řadič přerušení NVIC (nested vector interrupt controller). Řadič přerušení umožňuje nastavovat jednotlivým přerušením prioritu a umožňuje také, aby přerušení s vyšší prioritou přerušilo vykonávání přerušení s prioritou nižší.



Obr. 2.3.1. Blokové shéma řadiče STM32F303xE, převzato z [1]

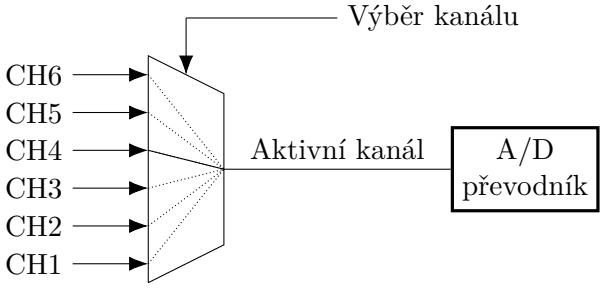
Kód je možné do mikrořadiče nahrát pomocí ladících rozhraní JTAG nebo SWD, nebo pomocí vestavěného bootloaderu skrze komunikační rozhraní (např. UART, SPI nebo USB).

2.3.2. Interní sběrnice a rozdělení paměti

Mikrořadiče STM32 obsahují interní ROM paměť flash a interní paměť SRAM. Obě tyto paměti jsou připojeny přes sběrnice k jádru mikrořadiče a k řadičům DMA. Některé mikrořadiče (STM32F3) obsahují také tzv. paměť CCM, která je připojena přímo k jádru mikrořadiče. V případě mikrořadičů STM32F3 slouží tato paměť k uložení části kódu programu, která má být vykonávána rychle (např. přerušení).

2.3.3. Rozbor A/D převodníků

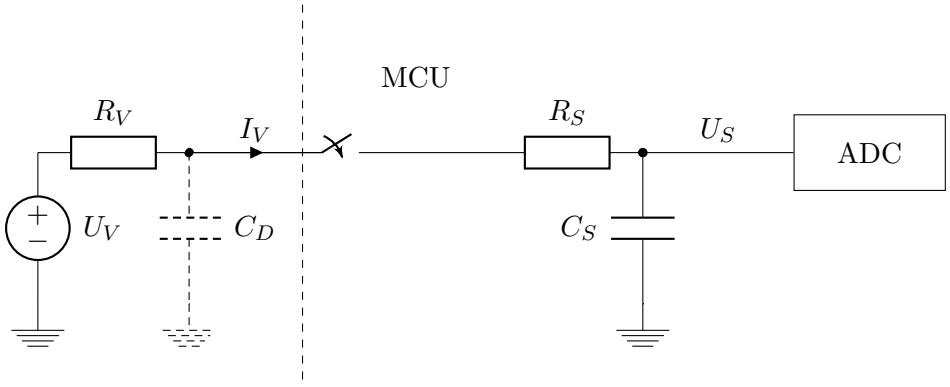
Mikrořadiče řady STM32 obsahují jeden či více vnitřních A/D převodníků s postupnou aproximací. Každý převodník může přepínat mezi několika různými vstupními kanály. Výhodou těchto převodníku je možnost rychle přepínat mezi vstupními kanály a vzorkovat tak více kanálů.



Obr. 2.3.2. Multiplexer A/D převodníku

Vstupní impedance externího signálu

Čtení analogového napětí se skládá ze dvou částí: odběr vzorku a konverze napětí. V první fázi se vzorkovací kondenzátor (C_S) nabije na měřené napětí (U_V). V druhé fázi se uložený náboj převede na digitální hodnotu pomocí postupné aproximace.



Obr. 2.3.3. Zjednodušený model A/D převodníku

Z první fáze vyplývá, že maximální vstupní impedance (R_V) závisí na době odběru vzorku. Pokud bude doba odběru příliš krátká, nestačí se vzorkovací kondenzátor nabít

na měřené napětí. Do výpočtu musíme ještě započítat odpor přepínacího obvodu (R_S).

$$u_S(t) = U_V \times (1 - e^{\frac{-t}{RC}}) \tag{2.3.1}$$

$$\frac{u_S(t)}{U_V} = 1 - e^{\frac{-t}{RC}} \leq \frac{2^N - 1}{2^N} \tag{2.3.2}$$

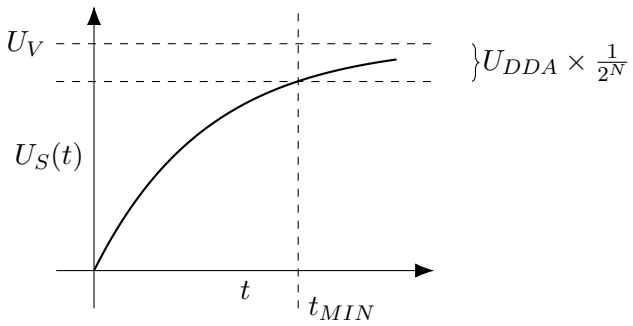
$$e^{\frac{-t}{RC}} \geq \frac{1}{2^N} \tag{2.3.3}$$

$$\frac{-t}{RC} \geq ln(\frac{1}{2^N}) \tag{2.3.4}$$

$$R_V \leq \frac{-t}{C \times ln(\frac{1}{2^N})} - R_S \tag{2.3.5}$$

Teoreticky se vzorkovací kondenzátor nikdy nenabije na plné měřené napětí. Proto musíme vzít v úvahu rozlišení A/D převodníku. Stačí tedy, aby rozdíl mezi měřeným napětím a napětím na vzorkovacím kondenzátoru byl menší než rozlišení A/D převodníku (1 LSB — least significant bit). Někdy může být kritérium přísnější a požadovat např. maximální chybu 1/2 LSB.

Problém vstupní impedance můžeme také vyřešit tím, že na vstup A/D převodníku připojíme paralelně externí kondenzátor na zem (C_D). To umožňuje vyrovnání proudových špiček při odběru vzorku. Zároveň však může omezit maximální frekvenci signálu.



Obr. 2.3.4. Nabíjení vzorkovacího kondenzátoru A/D převodníku

Vnitřní napěťová reference

Mikrořadiče řady STM32 obsahují vnitřní referenční napětí (obvykle 1.2 V), které je nezávislé na napájecím napětí. Tato reference je připojena na vstup A/D převodníku a měřením této reference jsme schopni dopočítat hodnotu napájecího napětí analogové části mikrořadiče. Tuto hodnotu tak můžeme použít k výpočtu skutečného napětí na vstupu A/D převodníku. Někdy může být vhodné měřit hodnotu reference při každém odběru, nebo v pravidelných intervalech. Můžeme tak kompenzovat změny napájecího napětí, pokud není stabilní.

2.3.4. Rozbor čítačů

Mikrořadiče STM32 obsahují několik vnitřních nezávislých čítačů. Obvykle je k dispozici jeden 32-bitový čítač a zbytek čítačů je 16-bitový. Většinu z nich lze použít pro generování, nebo měření externích digitálních signálů. Čítače, které nelze připojit na vstupně/výstupní piny, lze použít pro spouštění vnitřních periferií (např. A/D převodníků).

Čítače mají obvykle hodinový signál o stejné frekvencí jako jádro mikrořadiče. V případě mikrořadičů STM32F3 je možné dokonce taktovat čítače až dvojnásobnou frekvencí jádra z obvodu PLL. Čítač může čítat vzestupně (tedy zvyšuje hodnotu), sestupně (snižuje hodnotu), nebo oběma směry (střídá vzestupně a sestupně). Pokud čítač dosáhne horní meze (nastavené uživatelem) nebo nuly, začne čítat od začátku.

Většina čítačů obsahuje 2 - 4 tzv. "Capture-compare" jednotky. Ty mohou pracovat ve dvou režimech: jako vstupní — tzv. "input-capture" (IC) —, nebo výstupní — tzv. "output-compare" (OC). Ve vstupním módu jednotka reaguje na náběžnou nebo sestupnou hranu a při této události uloží aktuální hodnotu čítače a nastaví příznak ve stavovém registru. Ve výstupní režimu jednotka porovnává hodnotu čítače s předem nastavenou hodnotou a na základě porovnání mění logickou hodnotu na výstupu.

Čítače mají různé možnosti v jednotlivých řadách řadičů, ale základ je vždy stejný. Rozdíly jsou především v pokročilých funkcí. Níže popsané metody je možné implementovat na libovolném řadiči STM32. Čítače lze také propojit s vnitřními periferiemi mikrořadiče. Je tak možné pravidelně spouštět A/D převodník nebo D/A převodník.

Důležitým faktorem pro přesnost čítače je přesnost taktovací frekvence. Mikrořadiče STM32 umožňují připojit externí krystal, který obvykle zaručuje vysokou přesnost taktovací frekvence. Použití vnitřních oscilátorů bohužel takovou přesnost nepřináší, ale v některých aplikacích je možné tyto oscilátory za běhu kalibrovat a zvyšovat tak jejich přesnost. Příkladem může být mikrořadič STM32F042, který umožňuje kalibrovat vnitřní oscilátor z periodických zpráv na sběrnici USB (tzv. "Start-of-frame" pakety).

Další nepřesnost může do měření vnést obvod PLL, který slouží k násobení frekvence hodinového signálu s oscilátoru. Obvod PLL obvykle způsobuje krátkodobé výkyvy frekvence (tzv. jitter).

2.3.5. Rozbor řadiče DMA

Řadič DMA (direct memory access) umožňuje přenášet data mezi periferiemi a pamětí bez účasti jádra mikrořadiče. Použití řadičů DMA značně usnadňuje implementaci přístrojových funkcí, zvláště pak těch měřících. Obvykle tak stačí správně nastavit příslušné periferie a měřená data se automaticky přenášejí do paměti. Pak lze tato data jednoduše poslat do PC aplikace ke zpracování.

Řadič DMA jako kruhový buffer

Řadič DMA na STM32 podporuje snadnou implementaci kruhového bufferu. Umožňuje přenášet data v cyklickém režimu, tedy stále dokola čte nebo zapisuje do stejného úseku paměti. Dále umožňuje vyvolat přerušení při přenosu celého bufferu a jeho poloviny. Pomocí těchto dvou přerušení můžeme jednu polovinu bufferu zpracovávat a druhou přenášet řadičem DMA.

2.3.6. Rozbor komunikačních rozhraní

Mikrořadiče řady STM32 obsahují několik různých komunikačních rozhraní. Téměř všechny obsahují: UART (sériová linka), SPI a I^2C. Některé mikrořadiče pak obsahují rozhraní USB či CAN. Vzhledem k tomu, že chceme připojit mikrořadič k počítači, využijeme především rozhraní UART propojené skrze ST-Link[4] na vývojové desce nebo přímo rozhraní USB při použití samostatného mikrořadiče (např. na kontaktním poli).

[4]Ladící sonda od firmy STMicroelectronics vestavěná ve většině vývojových desek

2.4. Podobné projekty

2.4.1. Projekt Girino

Platforma Arduino je velmi oblíbená, a tak se logicky na internetu objevilo několik projektů realizujících osciloskop na této platformě. Jedním z nich je projekt Girino[5]. Pro použití je potřeba připojit několik externích součástek: RC-článek pro realizaci triggeru a operační zesilovače pro zesílení signálu. Osciloskop je jednokanálový a dosahuje rychlosti 154 kS/s.

2.4.2. Projekt MiniScope

Dalším zajímavým projektem je projekt MiniSkope[6]. Projekt realizuje funkci osciloskop a umožňuje zaznamenávání dat v reálném čase přes rozhraní USB. Projekt obsahuje několik variant jednoduchých zařízení s mikrořadiči STM32 a s mikrořadiči od firmy Atmel. Součástí projektu je PC aplikace a USB ovladač vycházející z knihovny libusb.

[5] `http://www.instructables.com/id/Girino-Fast-Arduino-Oscilloscope/`
[6] `http://tomeko.net/miniscope_v2e/index.php?lang=en`

3. Popis komunikačního protokolu

Komunikační protokol je velmi důležitou součástí celého systému. Je vhodné ho správně navrhnout a zdokumentovat. Pak jsme totiž schopni snadno vyměnit implementaci softwaru na mikrořadiči či PC aplikaci, aniž bychom museli upravovat software na druhé straně komunikačního kanálu.

Na druhou stranu přílišná obecnost protokolu může zkomplikovat implementaci na straně mikrořadiče. Proto se např. většina dat přenáší v nezpracované formě přímo z periferie řadiče. To tedy vyžaduje, aby PC aplikace dostala informace o tom, jak tato data zpracovat, např. v hlavičce zprávy.

3.1. Komunikační rozhraní

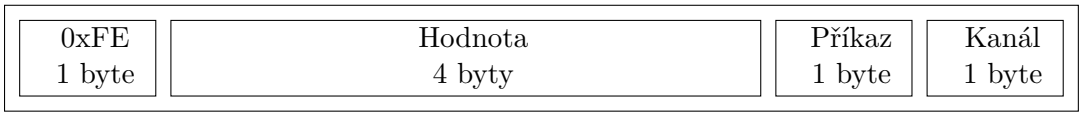
Pro realizaci protokolu předpokládáme komunikační rozhraní, které umožňuje sekvenční posílání jednotlivých bytů (znaků). Takovým rozhraním může být např.: Sériová linka (UART), USB (Virtual COM port), nebo TCP/IP spojení.

3.2. Hlavička zprávy

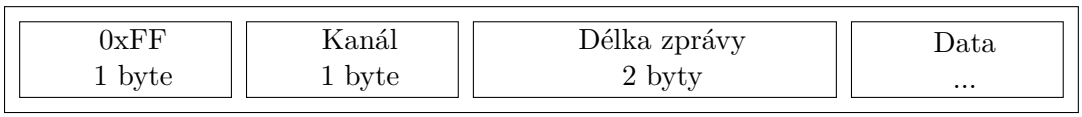
Komunikace se skládá ze dvou druhů zpráv: z příkazů a naměřených dat. Každá zpráva obsahuje číslo kanálu, kterému je určena. Každý měřící modul tak používá kanál s jiným číslem. Číslo kanálu nám tak umožňuje snadno předávat zprávy konkrétním měřícím modulům, jak na straně PC aplikace, tak na straně mikrořadiče. Číslo kanálu 0 slouží k nastavení a čtení globálního nastavení a vlastností mikrořadiče (např. typ zařízení apod.). V podstatě je to podobný koncept jako endpointy v rozhraní USB.

Zpráva typu příkaz obsahuje úvodní byte, číslo kanálu, typ příkazu (specifický pro daný modul) a 32-bitovou hodnotu příkazu (argument funkce). Zpráva naměřených dat obsahuje úvodní byte, který označuje začátek zprávy, číslo kanálu a délku zprávy v bytech. Poté již následují naměřená data.

Pro reprezentaci čísel se používá formát “little-endian”. Vzhledem k tomu, že obě architektury (x86 a ARM) používají little-endian, můžeme se v některých případech vyhnout konverzi formátu při zpracování dat z protokolu.



Obr. 3.2.1. Struktura zprávy typu příkaz

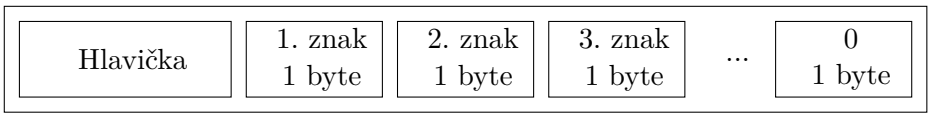


Obr. 3.2.2. Struktura zprávy typu data

Příkaz	Hodnota	Poznámka
G	'N' = název zařízení 'V' = napájecí napětí	Přečti globální parametr

Tabulka 3.4.1. Seznam globálních příkazů

Obr. 3.4.1. Struktura zprávy posílající hodnotu napájecího napětí



Obr. 3.4.2. Struktura zprávy posílající název zařízení, jedná se řetězec zakončený nulou

3.3. Časový limit příchozí zprávy

Při posílání dat může dojít k chybě nebo je komunikace náhle přerušena (např. zavření PC aplikace). Jedna strana tak čeká na příjem dat, které ale nikdy nedorazí. Tuto situaci je možné řešit tím, že nastavíme časový limit, po jehož vypršení budeme pokládat přenos za nepovedený a ukončený. Důležitým ukazatelem tak je doba, která uplynula od přijetí posledního znaku.

3.4. Obecné příkazy

Některé hodnoty, které se však nevztahují ke konkrétnímu měřícímu modulu, je třeba přenést do PC aplikace. Jedná se o napájecí napětí zařízení a jeho název. Pro získání těchto parametrů slouží kanál s číslem 0. PC aplikace odešle příkaz pro čtení příslušné proměnné a mikrořadič odešle zpět hodnotu (zpráva typu data).

4. Realizace funkce voltmetr na STM32

4.1. Využití vnitřních bloků mikrořadiče

4.1.1. Použití A/D převodníků

Pro realizaci voltmetru potřebujeme A/D převodník. Nabízí se využití vnitřních A/D převodníků mikrořadiče. V závislosti na konkrétním mikrořadiči je možné připojit k jednomu A/D převodníku více jak 10 externích kanálů.

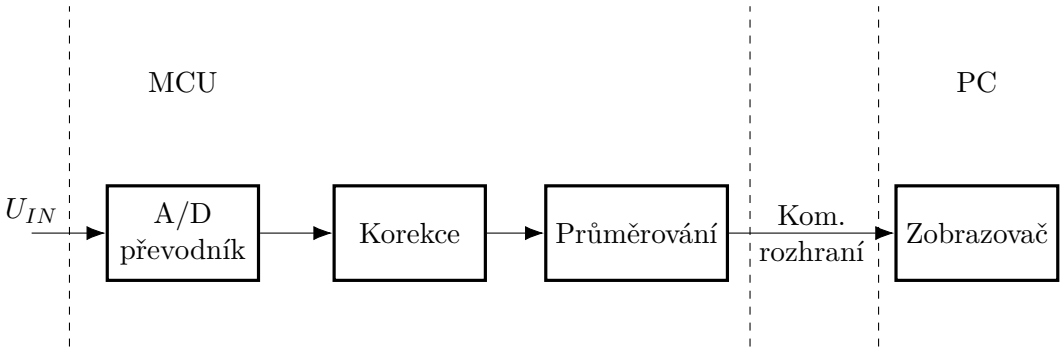
Interní A/D převodníky je možné také zapojit ve společném režimu "dual ADC" případně "tripple ADC". Ale vzhledem k tomu, že nám jde především o přesnost a nikoliv o rychlost, toto zapojení A/D převodníků nevyužijeme.

4.1.2. Korekce vlivu napájecího napětí

A/D převodník převádí napětí v rozsahu 0 až *vstupní pozitivní reference* (U_{REF+}). U mikrořadičů v malých pouzdrech je toto napětí rovné analogovému napájení (U_{VDDA}). Napájení mikrořadiče však může kolísat, případně jej nemusíme dopředu znát. Proto je vhodné naměřená data před zpracováním korigovat.

Mikrořadiče STM32 obsahují tzv. vnitřní referenční napětí (U_{REFINT}), které je připojeno na vstup A/D převodníku. Dále obsahují kalibrační hodnotu (U_{REFINT_CAL}), uloženou v ROM paměti, která reprezentuje hodnotu konverze vnitřní reference při napájecím napětí 3.3 V. To nám umožňuje korigovat změny napájecího napětí. V případě použití je vhodné porovnat změřenou hodnotu s měřením vnitřní reference. Tím jsme schopni získat hodnotu, která je nezávislá na napájecím napětí. Pro 12-bitový A/D převodník spočítáme skutečnou hodnotu napětí na vstupu z hodnoty v datovém registru (ADC_DR) takto:

$$U_{IN} = \frac{3.3\,\mathrm{V} \times U_{REFINT\_CAL} \times ADC\_DR}{U_{REFINT} \times 4095} \tag{4.1.1}$$



Obr. 4.1.1. Blokové schéma voltmetru

Příkaz	Hodnota	Poznámka
S	0 = stop, ostatní = start	Zastaví/spustí osciloskop
A	počet vzorků	Nastaví průměrování

Tabulka 4.3.1. Seznam příkazů pro funkci voltmetr

4.1.3. Vzorkování napětí

Pravidelné odebírání vzorků můžeme řešit hardwarově pomocí čítače, kdy čítač automaticky spouští převod kanálů A/D převodníků, nebo spouštět A/D převodník pomocí softwaru např. v přerušení od systémového časovače (SysTick). Vzhledem k tomu, že měříme pomalu měnící se napětí a není tolik důležitá rychlost, zvolil jsem softwarové řešení. Spolu s měřeným napětím je vzorkována i vnitřní napěťová reference (U_{REFINT}).

Doba trvání odběru vzorku ovlivňuje maximální vzorkovací frekvenci a maximální vstupní impedanci. Vzhledem k tomu, že rychlost měření není důležitá, nastavíme dobu trvání odběru vzorku na velikou hodnotu.

4.1.4. Implementace měření více kanálů

Vícekanálové měření je realizováno pomocí přepínání kanálů A/D převodníku (viz. Obr. 2.3.2). Při zahájení konverze se tak sekvenčně převedou hodnoty ze všech nastavených kanálů.

4.2. Průměrování

Průměrování slouží k potlačení vnitřních šumů a šumů na vstupním napětí. Průměrování můžeme realizovat na straně mikrořadiče, nebo na straně PC aplikace. Vzhledem k tomu, že nároky na průměrování na straně mikrořadiče nejsou příliš velké, realizoval jsem průměrování na mikrořadiči. Výhodou tohoto přístupu je, že snížíme počet posílaných dat do PC. Navíc program by musel před každým odběrem počkat, než předchozí data přijme aplikace. To může v některých případech trvat déle — např. pokud právě data posílá osciloskop, či jiný měřící modul.

4.3. Ukázková implementace na mikrořadiči STM32

4.3.1. Formát komunikace

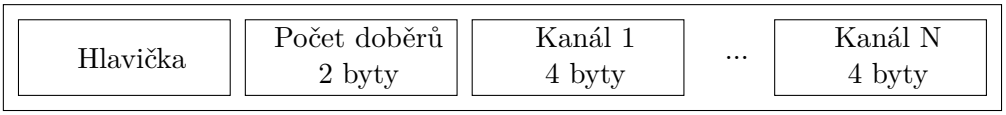
Řídící příkazy

Funkce voltmetr obsahuje pouze dva příkazy. Příkaz start/stop a nastavení průměrování.

Posílání dat

Při posílání dat se nejdříve pošle obecná hlavička zprávy, poté následují dva byty určující počet odběrů (N). Poté už následují hodnoty pro jednotlivé kanály. Přenáší se součet hodnot všech odebraných vzorků pro daný kanál, které jsou korigovány vnitřní referencí. Při zpracování dat na straně PC aplikace je tedy nutné tyto hodnoty vydělit počtem odběrů. Posílaná hodnota tedy vypadá takto:

$$X = \sum_{n=0}^{N-1} \frac{U_{REFINT\_CAL} \times ADC\_DR[n]}{U_{REFINT}[n]} \tag{4.3.1}$$



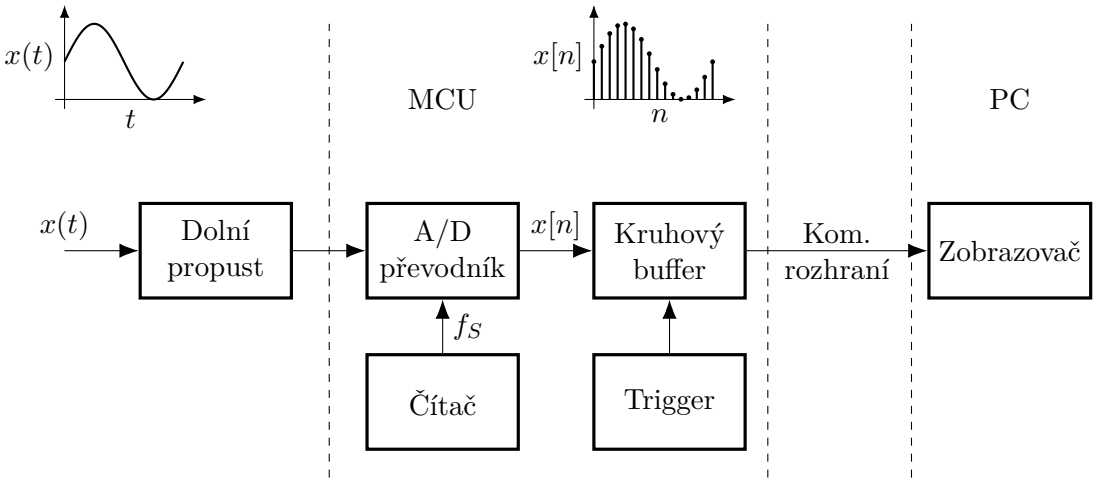
Obr. 4.3.1. Struktura dat posílaných voltmetrem

Na straně PC aplikace se pak vypočítá výsledné napětí (U_{IN}) takto:

$$U_{IN} = \frac{3.3\,\mathrm{V} \times X}{4095} \tag{4.3.2}$$

5. Realizace funkce osciloskop na mikrořadičích STM32

5.1. Využití vnitřních bloků mikrořadiče



Obr. 5.1.1. Blokové schéma osciloskopu

5.1.1. Použití A/D převodníků

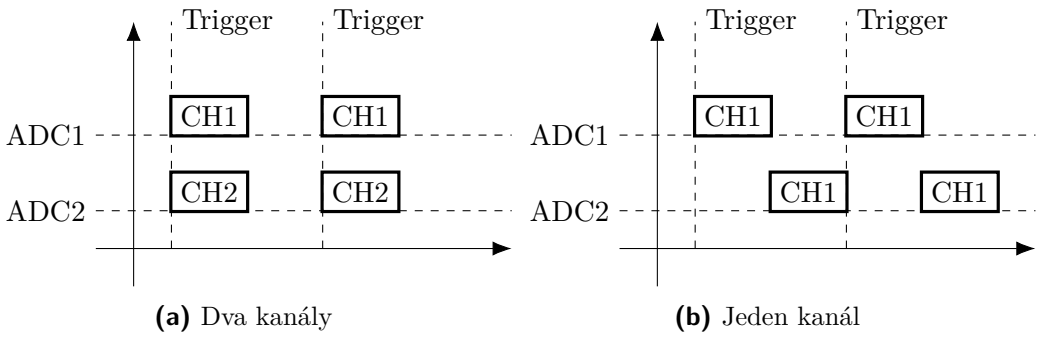
Pro realizaci osciloskopu potřebujeme A/D převodník. Nabízí se využití vnitřních A/D převodníků mikrořadiče. V závislosti na konkrétním mikrořadiči je možné připojit k jednomu A/D převodníku více než 10 externích kanálů.

Interní A/D převodníky je možné také zapojit ve společném režimu "dual ADC" případně "tripple ADC". V těchto režimech pracují 2, nebo 3 A/D převodníky synchronně. Tyto módy umožňují rychlejší vzorkovací kmitočet, pomocí tzv. *Interleaved mode (režim prokládání)* dle Obr. 5.1.2b. Jeden A/D převodník odebírá vzorek a druhý převádí navzorkovanou hodnotu. Další režim umožňuje vzorkovat více kanálů ve stejný okamžik dle Obr. 5.1.2a. Každý A/D převodník tedy vzorkuje jiný kanál, ale jsou spouštěny synchronně.

Kromě interních A/D převodníků můžeme použít externí A/D převodníky, které je možné připojit k mikrořadiči např. pomocí rozhraní SPI nebo I2S. Důvodem k použití externích A/D převodníků může být vyšší přesnost a odolnost vůči šumům.

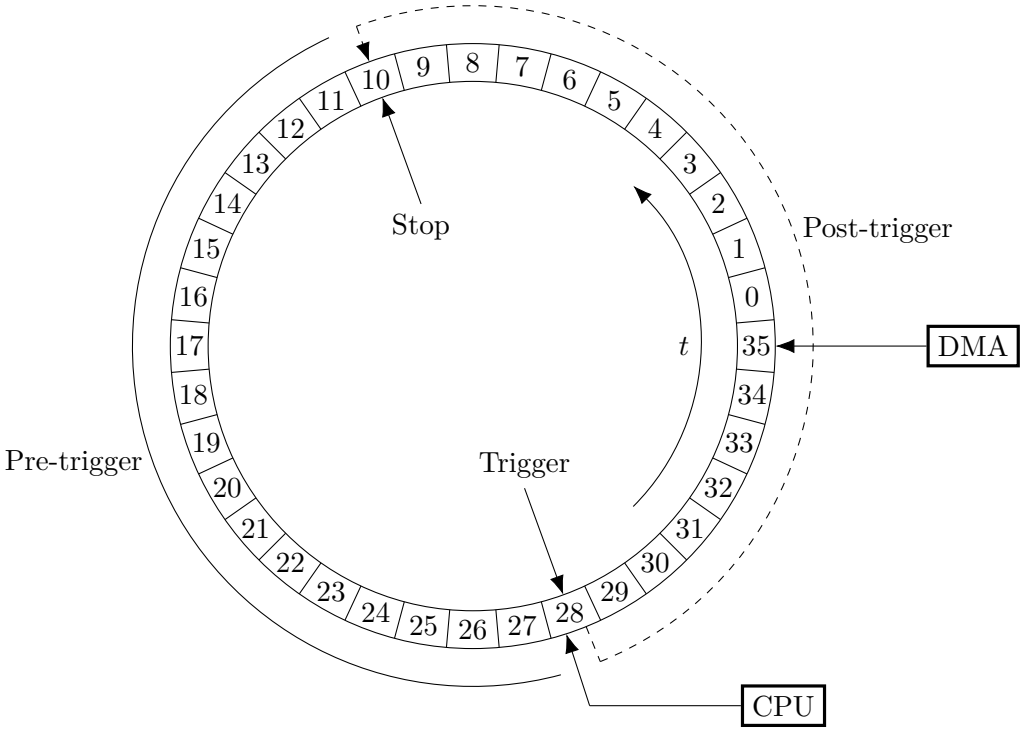
5.1.2. Ukládání vzorků do paměti

Pro ukládání vzorků do paměti je využit řadič DMA (kvůli velkému toku dat softwarové ukládání nepřipadá v úvahu). Ten zajistí rychlé a spolehlivé ukládání vzorků do paměti a možnost průběžného zpracování vzorků v reálném čase.



(a) Dva kanály **(b)** Jeden kanál

Obr. 5.1.2. Vzorkování v módu "Dual ADC"



Obr. 5.1.3. Popis kruhového bufferu

5.1.3. Určení vzorkovací frekvence

Pro realizaci dále potřebujeme nějaký mechanismus, jak A/D převodník spouštět v pravidelných intervalech. V případě interních A/D převodníku můžeme použít vnitřní čítače. To nám umožní nastavit vzorkovací frekvenci jako podíl vstupní frekvence čítače (obvykle frekvence jádra či polovina této frekvence).

Další možností je použít tzv. "Continuous mode", ve kterém A/D převodník se spustí ihned po dokončení předchozí konverze. Použití tohoto módu může být výhodné pro realizaci vysoké vzorkovací frekvence. Pro nízké frekvence je bohužel nepoužitelný, neboť nejsme schopni zajistit dostatečně pomalý běh. Poslední možností je využití externího hodinového signálu. Ten můžeme buď přímo propojit s interním A/D převodníkem, nebo připojit na vstup čítače.

V každém případě je zde jisté omezení vzorkovací frekvence a její přesnosti. V případě použití vnitřních čítačů nebo tzv. "Continuous mode" jsme obvykle omezeni frekvencí

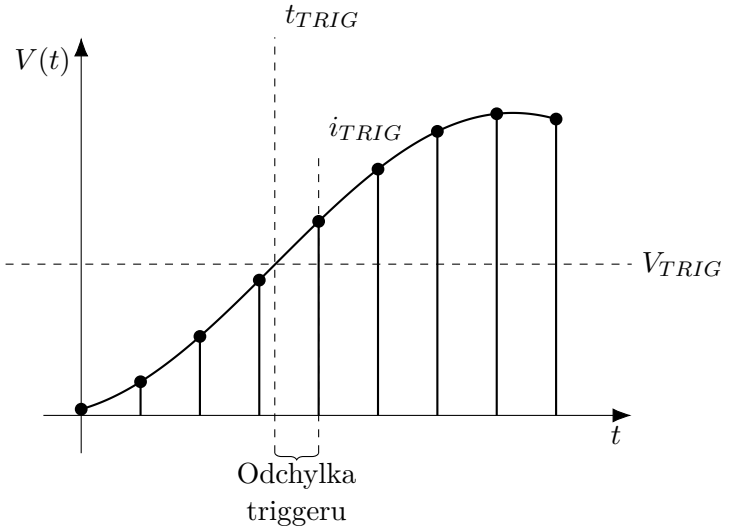
jádra. Vzorkovací frekvence tak obvykle bývá celočíselným podílem frekvence jádra. V případě použití externích hodin dochází tzv. jitteru vzorkovací frekvence, který je způsoben synchronizací mezi mikrořadiče a externím signálem. Jitter se obvykle projeví až při vyšších vzorkovacích frekvencích.

5.1.4. Implementace měření více kanálů

Vícekanálový záznam na osciloskopu můžeme opět provést dvěma způsoby. První způsob je použití jednoho A/D převodníku pomocí přepínání kanálů (viz. Obr. 2.3.2). Vzorky se tedy odebírají sekvenčně a druhý kanál je tak zpožděn oproti prvnímu o čas potřebný pro převod vzorku. Toto zpoždění pak můžeme kompenzovat v zobrazovací aplikaci.

Druhou možností je použít více A/D převodníků např. v tzv. "Dual ADC" módu. Vzorky obou kanálů jsou tak odebrány ve stejný čas. Nevýhodou tohoto přístupu může být potřeba použít pro každý A/D převodník vlastní řadič DMA a vlastní buffer.

5.2. Realizace triggeru



Obr. 5.2.1. Znázornění triggeru na náběžnou hranu

5.2.1. Detekce okamžiku triggeru

Hlavním cílem triggeru je určit přesný časový okamžik, kdy nastala spádová nebo náběžná hrana, tedy kdy napětí kleslo pod stanovenou úroveň respektive překročilo danou úroveň napětí. Tento okamžik se nazývá "trigger" a jeho detekci je možné řešit více přístupy.

Softwarové řešení triggeru

Jedním řešením může být periodické čtení hodnot A/D převodníku jádrem mikrořadiče. Může se jednat o vzorky uložené v paměti, případně o poslední hodnotu uloženou v datovém registru A/D převodníku. Porovnáváním přečtených vzorků s úrovní triggeru tak můžeme zjistit okamžik triggeru.

Nevýhodou tohoto přístupu je, že vytěžuje výkon mikrořadiče zvlášť při použití vysoké vzorkovací frekvence. Toto můžeme řešit tím, že budeme zkoumat každý N-tý vzorek. Tím získáme hrubý odhad pozice triggeru. Z hrubého odhadu pak můžeme dohledat dohledat přesný vzorek, kde trigger nastal. To má za důsledek, že osciloskop nemusí zachytit krátké změny napětí okolo úrovně triggeru. Naopak výhodou tohoto přístupu je možnost nastavit různé parametry triggeru na (téměř) libovolné hodnoty.

Hardwarové řešení triggeru

Druhým přístupem je použití hardwarových prostředků mikrořadiče. Některé mikrořadiče[1] řady STM32 obsahují analogové komparátory, které mohou být připojeny na některé vstupy A/D převodníku. Pomocí druhého vstupu (−) komparátoru jsme schopni nastavit úroveň triggeru. Na druhý vstup (−) můžeme připojit externí signál, výstup D/A převodníku, nebo část $(1, \frac{3}{4}, \frac{1}{2}, \frac{1}{4})$ vnitřní reference (1.2 V). Výstupem komparátoru pak můžeme skrze čítač, nebo přerušení řídit přenos dat.

Nevýhodou tohoto přístupu špatná dostupnost hardwarových prostředků na mikrořadiči. Jen některé vstupy A/D převodníku lze zároveň připojit na vstup komparátoru. Dále je omezen rozsah hodnot, které můžeme nastavit jako úroveň triggeru. Buď musíme použít předem definovaný násobek referenčního napětí mikrořadiče, nebo připojit na druhý vstup D/A převodník.

Vzhledem k hardwarovým omezením jsem se ve své vzorové aplikaci rozhodl použít softwarové řešení triggeru. Hardwarové řešení by však mohlo být v některých konkrétních aplikacích vhodnější a i efektivnější z hlediska výpočetního zatížení mikrořadiče a spotřeby.

Externí trigger

V předchozích odstavcích jsme řešily spuštění (trigger) v reakci na změnu napětí signálu. V některých případech může být vhodné reagovat na změnu jiného externího digitálního signálu. Tento signál se označuje jako externí trigger.

Externí trigger můžeme řešit třemi způsoby. První je použít přerušení od vstupního digitálního pinu. Vzniká tak časová nepřesnost vlivem zpoždění. Druhou možností je použít Input-Capture jednotku čítače a zachytit přesný časový okamžik (nebo vzorek) triggeru. Problém je propojení z čítačem, které omezuje použitelné vstupní piny.

Poslední možností je spojit funkci osciloskopu s logickým analyzátorem (viz Kapitola 6). Poté můžeme trigger řešit softwarově na digitálních signálech. Výhodou je, že jsme schopni externí trigger za běhu přenastavit na jiný digitální vstup.

Nepřesnost časového okamžiku triggeru

Vzhledem k tomu, že vzorkujeme signál v diskrétním čase, trigger nastává v intervalu mezi dvěma vzorky. Pokaždé se ale může jednat o jiný časový okamžik. Abychom mohli určit přesný okamžik triggeru, musíme signál mezi vzorky interpolovat.

Pro interpolaci můžeme využít několik různých postupů. Prvním z nich je lineární interpolace, kdy spojíme dva sousední vzorky úsečkou. Druhým řešením může být použití tzv. sinc filtru.

Díky přesnějšímu určení doby triggeru můžeme odstranit nepříjemné kolísání signálu v časové ose tím, že signál posuneme o zlomek vzorkovací periody. Diskrétní signál je tedy stejný, pouze je jinak zobrazen.

[1] STM32F0, STM32F3, STM32L0, STM32L1 a STM32L4

5.2.2. Realizace pre-triggeru

Při analýze signálu obvykle chceme zobrazit průběh po triggeru i před triggerem (tzv. pre-trigger). Abychom toto mohli realizovat, musíme neustále sledovat měřený signál a ukládat vzorky do kruhového bufferu. Ve chvíli kdy detekujeme na vstupu trigger, musíme včas zastavit ukládání vzorků do paměti tak, abychom měli správný počet vzorků před triggerem.

Opět můžeme použít hardwarové a softwarové řešení. Obě tato řešení mají své klady a zápory. Musíme také vzít v úvahu skutečnost, zda detekci provádíme softwarově či hardwarově. V případě softwarové detekce triggeru může dojít k nepřesnosti způsobené pozdní detekcí triggeru (už nestihneme včas zastavit ukládání / převod).

Vzhledem k možným nepřesnostem zastavení A/D převodníku včas, je potřeba zpětně zkontrolovat, na jakém vzorku se převod skutečně zastavil, abychom byli schopni poslat správná data k dalšímu zpracování.

Hardwarové řešení

Bohužel řadič DMA neumožňuje zastavit přenos na předem stanoveném vzorku. Takové nastavení by vyžadovalo překonfigurovat řadič DMA a během tohoto okamžiku bychom mohli přijít o některé vzorky.

Možným řešením je použít druhý čítač, který bude sledovat kolikátý vzorek čteme v rámci kruhového bufferu. Pomocí output compare kanálu jsme schopni zastavit první čítač (který určuje vzorkovací frekvenci) po přečtení daného počtu vzorků.

Nevýhodou je použití hardwarových prostředků, které mohou být použity k jiným účelům. Také není možné použít libovolné čítače, ale pouze ty, které jdou vzájemně zapojit v obou směrech jako master-slave. Výhodou je větší přesnost zastavení A/D převodníku.

Softwarové řešení

Dalším řešením je pravidelně se dotazovat na počet uložených vzorků a v případě, že počet dosáhne daného čísla, zastavit řadič DMA nebo A/D převodník. Nevýhodou tohoto řešení je vyšší výpočetní zátěž mikrořadiče a nepřesné zastavení.

5.2.3. Správné zobrazení

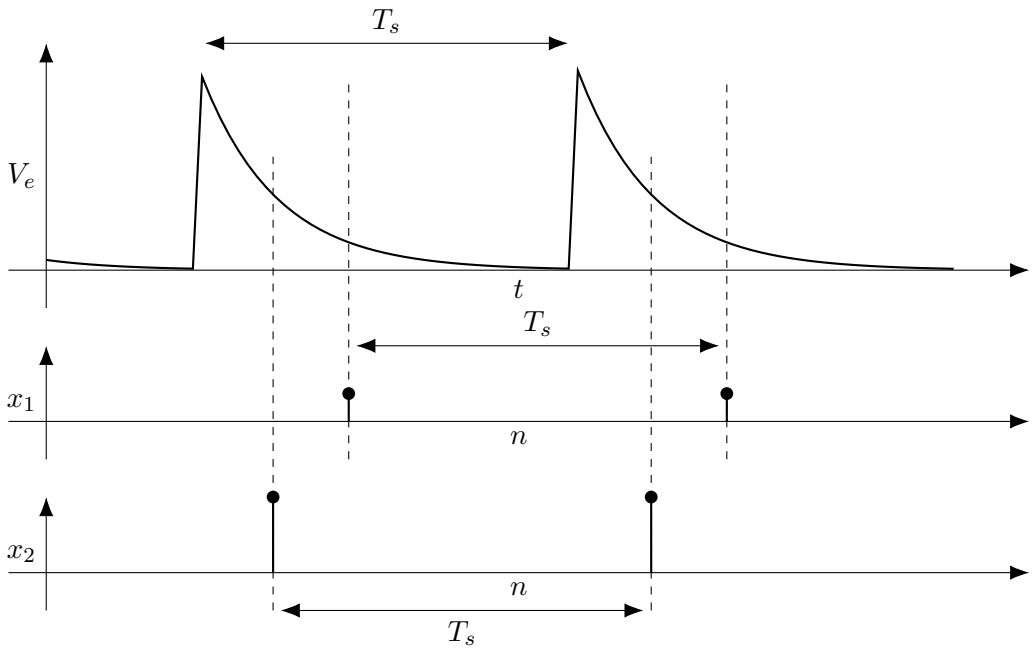
Jak již bylo zmíněno výše, je potřeba dopočítat přesný okamžik triggeru a posunout zobrazený signál.

Dále můžeme signál přepočítat tak, aby v každém průběhu byl trigger na stejné pozici v časové ose. Diskrétní signál je tedy upraven pomocí filtru. To nám usnadní průměrování jednotlivých průběhů a vyhlazení signálu.

5.3. Průměrování

Průměrování se často používá pro potlačení šumů v signálu. Průměrují se tak vzorky z několika průběhů. Problém při průměrování může být nepřesnost triggeru, která způsobí, že neprůměrujeme vzorky ze stejného časového okamžiku. Tento problém můžeme řešit pomocí interpolaci (jak již bylo zmíněno výše).

Průměrování můžeme implementovat na straně PC aplikace nebo na straně mikrořadiče. Výhodou implementace na straně PC je rychlost výpočtu. Implementace na



Obr. 5.4.1. Vliv vnitřního šumu na stejnosměrnou složku signálu

straně mikrořadiče může být výhodná ve chvíli, kdy máme pomalý komunikační kanál s PC aplikací. Pak stačí poslat jenom jeden zprůměrovaný průběh (místo několika jednotlivých průběhů).

5.4. Vnitřní šumy a jejich snížení

Při implementaci osciloskopu se objevilo kolísání střední hodnoty jednotlivých průběhů a to i při různých vzorkovacích kmitočtech. Jednou z možných příčin by mohlo být nějaké vysokofrekvenční rušení, které je přesným násobkem vzorkovací frekvence. Tím dochází k aliasingu a šum se tak promítne do stejnosměrné složky signálu. Toto rušení může být způsobeno např. vnitřními sběrnicemi, jádrem řadiče nebo dalšími periferiemi.

Vzhled povaze rušení je poměrně obtížné takové rušení změřit. Možným řešením je použití externího nezávislého hodinového signálu. Jako zdroj můžeme použít externí generátor, případně jiný mikrořadič. Frekvenci zvolíme tak, aby nebyla soudělná s vnitřními hodinami řadiče. Pomocí spektrální analýzy pak můžeme zjistit frekvenci rušení a odhadnout která část řadiče může takové rušení způsobovat.

Ze změřené frekvence (f_A) — aliasu skutečné frekvence — a vzorkovací frekvence můžeme odhadnout frekvenci původního signálu (f_X) jako následující posloupnost (f_N):

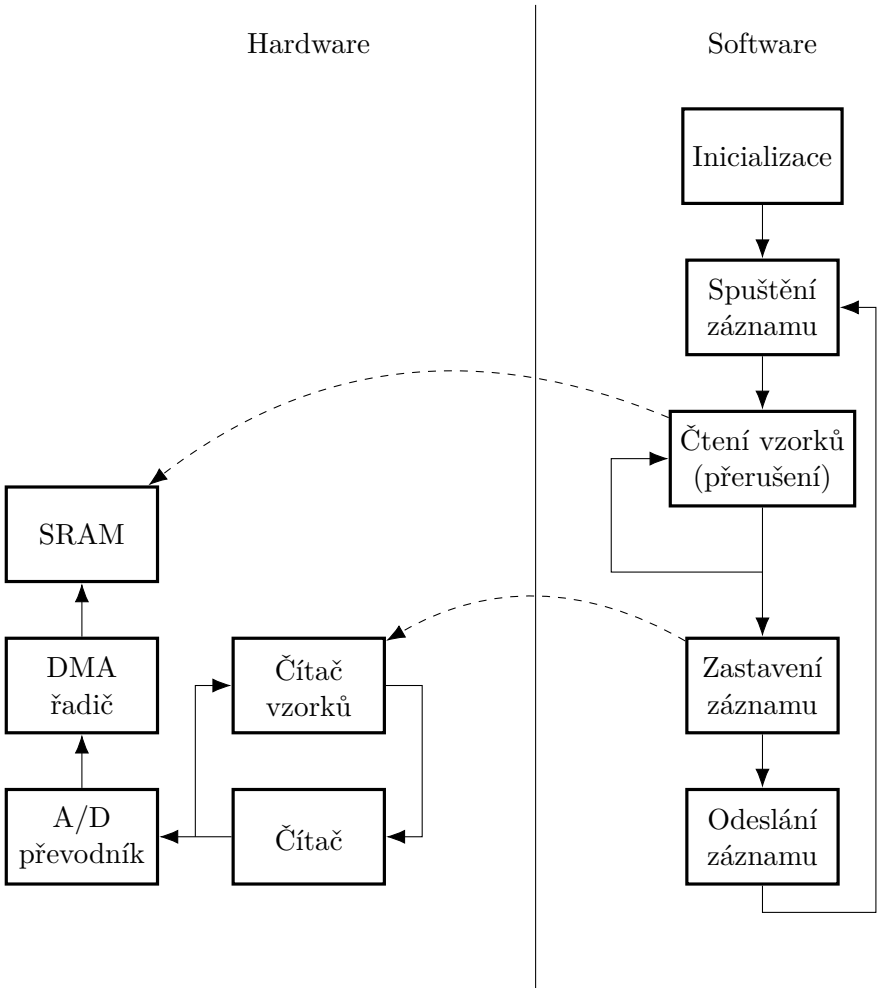
$$f_N[x] = f_S \times x \pm f_A \tag{5.4.1}$$

Abych byli schopni určit f_X jednoznačně, musíme provést více měření s různými f_S a najít průnik jednotlivých posloupností.

V tabulce 5.4.1 jsou naměřené frekvence pro mikrořadič STM32F303RE, který je používá externí hodinový signál 8 MHz a vnitřní hodiny 72 MHz z obvodu PLL (fázový závěs). Z naměřených hodnot vychází, že frekvence šumu odpovídá právě hodnotě 8 MHz. Zdrojem vnitřního šumu je tak pravděpodobně obvod PLL.

f_S(kHz)	f_A(kHz)	Výpočet původní frekvence
750	250	$f_X = 11 \times f_S - f_A = 8\,\text{MHz}$
900	100	$f_X = 9 \times f_S - f_A = 8\,\text{MHz}$
1150	50	$f_X = 7 \times f_S - f_A = 8\,\text{MHz}$

Tabulka 5.4.1. Měření frekvence vnitřního šumu A/D převodníku



5.5. Ukázková implementace na mikrořadiči STM32

Implementace funkce osciloskop je realizovaná pomocí kruhového bufferu. Používá se zde 12-bitové rozlišení A/D převodníku. V paměti jsou vzorky zarovnány na 16-bitů.

Je třeba si uvědomit, že v implementaci pracujeme se třemi délkami bufferu:

- Délka bufferu v bytech — důležité pro přenos dat do PC
- Délka bufferu ve vzorcích — důležité pro nastavení DMA
- Délka kanálu ve vzorcích — důležité pro trigger

Například pokud budeme mít 12-bitový dvou-kanálový osciloskop s velikostí bufferu 1024 bytů, tak délka bufferu ve vzorcích bude 512 vzorků a délka kanálu bude 256 vzorků. V různých částech programu je tedy potřeba počítat s jinou velikostí bufferu.

Příkaz	Hodnota	Poznámka
S	0 = stop, ostatní = start	Zastaví/spustí osciloskop
F	frekvence	Nastaví vzorkovací frekvenci
B	délka bufferu	Nastaví délku záznamu
C	maska kanálů	Zapne/vypne kanály osciloskopu, hodnota daného bitu určuje, zda je daný kanál zapnut
T	napěťová úroveň	Nastaví porovnávací úroveň triggeru
P	0x01 = Náběžná hrana 0x02 = Spádová hrana 0x03 = Obě hrany	Nastaví polaritu triggeru
D	velikost pre-triggeru (‰)	Nastaví velikost pretriggeru jako poměr velikosti bufferu
M	0 = normální mód, 1 = auto-trigger	Zapne/vypne tzv. “auto-trigger”
B	počet vzorků	Nastaví délku záznamu

Tabulka 5.5.1. Seznam příkazů pro funkci osciloskop

5.5.1. Formát komunikace

Řídící příkazy

Funkce osciloskop obsahuje několik řídících příkazů. Většina je určena především pro nastavení parametrů triggeru. Všechny příkazy jsou uvedeny v tabulce 5.5.1.

Posílání dat

Při komunikaci s PC aplikací firmware nejdříve odešle obecnou hlavičkou protokolu obsahující: číslo kanálu a počet bytů. Pak následuje hlavička specifická pro osciloskop, která popisuje počet bufferů, kanálů a velikost vzorku.

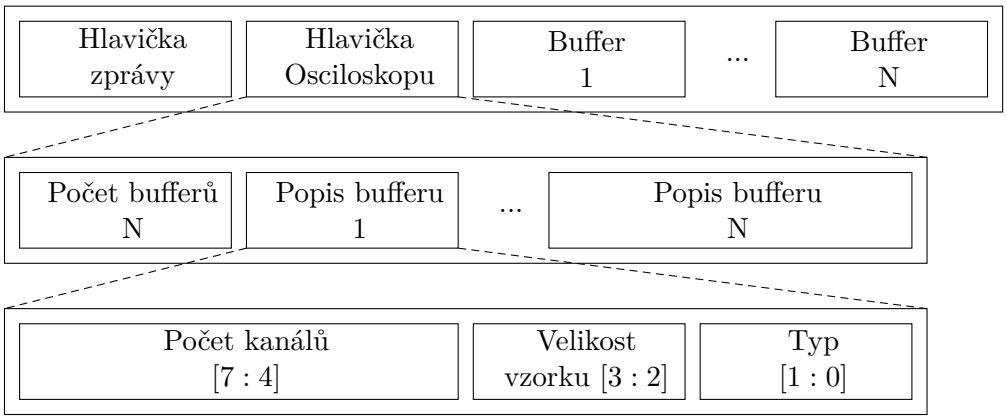
První byte hlavičku osciloskopu určuje počet bufferů. Více bufferů se využije v případě, kdy data pocházejí z více zdrojů a vzorky pro jeden kanál jsou oddělené od vzorků pro druhý kanál. Tohoto principu se používá v případě spojení s logickým analyzátorem, kdy se data digitálních vstupů posílají v odděleném bufferu.

Další byty pak obsahují popis jednotlivých bufferů. Každý buffer je popsán jedním bytem, který obsahuje informaci o typu (digitální nebo analogový), počtu kanálů a počtu bytů na vzorek.

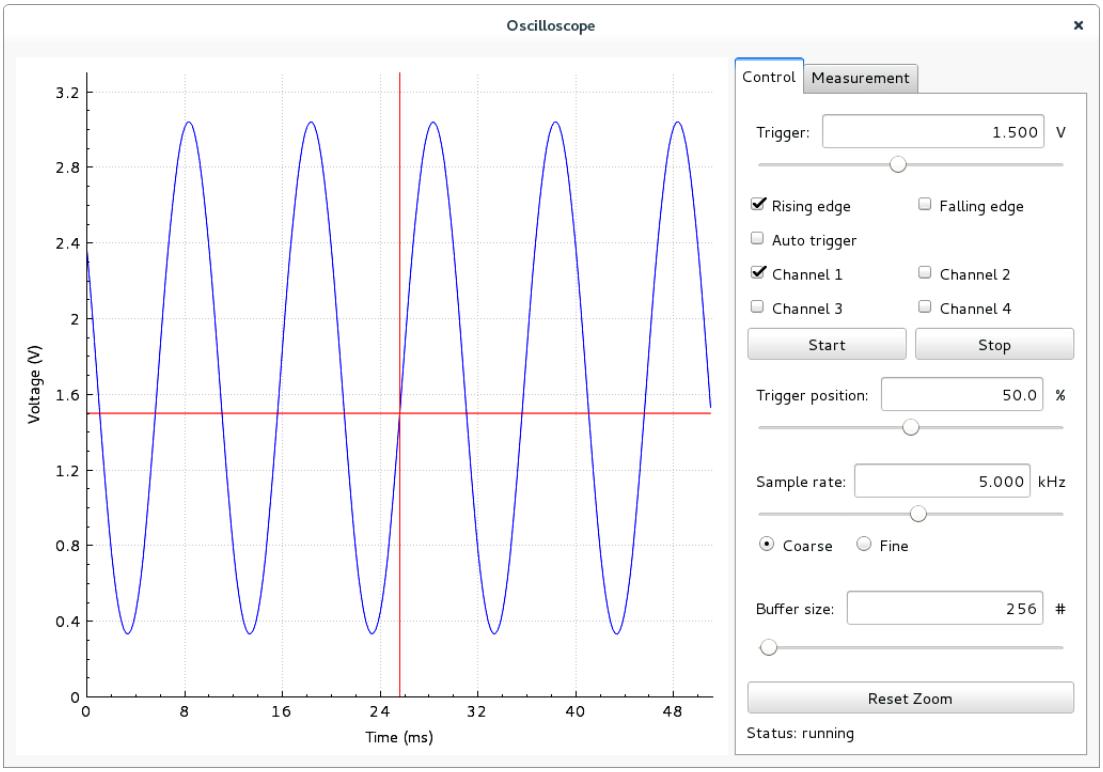
Poté následují obsahy jednotlivých bufferů, které už obsahují binární data z jednotlivých zdrojů.

5.5.2. Dosažené výsledky

Nejzajímavějším parametrem pro osciloskop je maximální vzorkovací frekvence. V tabulce 5.5.2 jsou shrnuty maximální dosažené vzorkovací frekvence při použití jednoho A/D převodníku.



Obr. 5.5.1. Formát dat posílaných z funkce osciloskop



Obr. 5.5.2. Uživatelské rozhraní osciloskopu

Mikrořadič	frekvence jádra (MHz)	Maximální vzorkovací frekvence (kS/s)
STM32F042	48	666.666
STM32F303	72	2000

Tabulka 5.5.2. Maximální dosažená vzorkovací frekvence jedno-kanálového osciloskopu

6. Realizace funkce logický analyzátor na STM32

6.1. Využití vnitřních bloků mikrořadiče

6.1.1. Využití brány GPIO

K realizaci funkce osciloskop můžeme využít digitální brány mikrořadiče, neboli "General purpose Input/Output" (GPIO). Čtením příslušného "Input data" registru (IDR) jsme schopni zaznamenat aktuální stav na 16 vstupně-výstupních pinech.

Nevýhodou čtení IDR je nemožnost zvolit, které piny budeme číst. Vždy tak čteme pevně danou bránu (např. PA0-15). To může být nevýhodou ve chvíli, kdy některé z těchto pinů používáme za jiným účelem, neboť tento pin už nemůžeme použit na čtení vstupu.

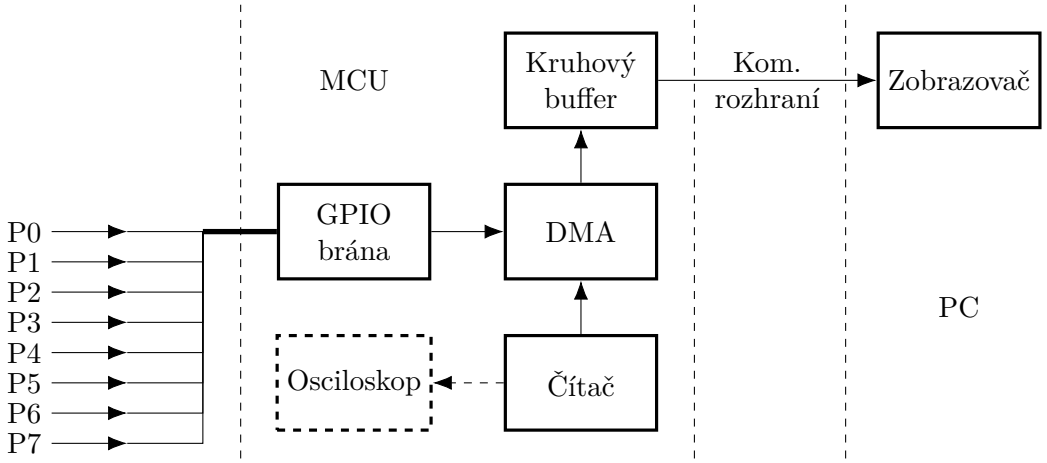
Nepoužité kanály můžeme v zobrazovací aplikaci skrýt a uvolnit tak místo na displeji pro zobrazení jiných kanálů.

6.1.2. Vzorkování dat

Data z IDR registru můžeme číst softwarem, nebo hardwarem pomocí řadiče DMA, který je spouštěn čítačem. Výhodou použití řadiče DMA je větší přesnost vzorkování, které ale stále není úplně přesné, neboť zpoždění mezi signálem od čítače a skutečným přečtení registru IDR závisí na aktuálním vytížení sběrnice. Zpoždění se může pohybovat zhruba v jednotkách cyklů sběrnice.

6.1.3. Počet kanálů

Jak již bylo zmíněno, IDR registr obsahuje 16 vstupních hodnot. Tímto způsobem jsme schopni tedy realizovat 16 kanálů. Pokud bychom chtěli více kanálů, museli bychom



Obr. 6.1.1. Blokové schéma analyzátoru

použít další DMA řadič, další čítač a číst data z další brány.

Data se stejně jako v případě osciloskopu ukládají do kruhového bufferu. V případě, že chceme použít pouze 8 kanálů, můžeme nastaví DMA řadič aby četl pouze dolních 8 bitů registru IDR. Tím ušetříme paměť v kruhovém bufferu.

6.1.4. Trigger

Trigger můžeme realizovat podobně jako v případě osciloskopu. Na rozdíl od osciloskopu ale můžeme chtít spouštět záznam na nějakou kombinaci digitálních vstupů, případně kombinace úroveň jednoho signálu a hrana druhého. Např.: "Chip-select" (úroveň) a hodinový signál komunikačního rozhraní (hrana).

6.1.5. Spojení s osciloskopem

V praxi se často spojuje funkce osciloskopu a logického analyzátoru. Výsledné zařízení tak synchronně zaznamenává logické i analogové signály. Jak je naznačeno na Obr. 6.1.1, je možné propojit čítač logického analyzátoru s osciloskopem. Jeden čítač tak spouští DMA přenos z GPIO brány (analyzátor) a zároveň spouští převod A/D převodníku (osciloskop).

Při konfiguraci délky záznamu či počtu kanálů osciloskopu, je třeba zajistit, aby oba kruhové buffer (pro osciloskop a analyzátor) měly stejný počet vzorků. Buffery se pak do PC aplikace přenášejí odděleně za sebou.

6.2. Ukázková implementace

Jak již bylo zmíněno výše, implementace logického analyzátoru je spojená s implementací osciloskopu (která je popsána v sekci 5.5) a používá stejný princip zpracování dat. Jediným rozdílem je zdroj dat.

7. Realizace čítačových funkcí na mikrořadičích STM32

Velká vstupní frekvence čítačů nám umožňuje poměrně přesné měření časových parametrů digitální signálů. Měřenými parametry jsou frekvence, délka pulsu, střída a počet pulsů. Při měření frekvence digitálních signálů musíme respektovat omezení čítačů. Pomocí čítačů nejsme schopni měřit frekvence větší než polovina frekvence hodinového signálu čítače.

7.0.1. Měření frekvence čítáním pulsů

Základním způsobem jak měřit frekvenci, je měřit počet pulsů (N_P) za nějaký pevný časový úsek (t_G). Délka časového úseku a maximální hodnota M-bitového čítače (N_{MAX}) nám určuje frekvenční rozlišení (Δf) a maximální vstupní frekvenci (f_{MAX}), která dále závisí na počtů bitů čítače (M). Výslednou frekvenci (f) pak dostaneme:

$$f = \frac{N_P}{t_G} \tag{7.0.1}$$

$$\Delta f = \frac{1}{t_G} \tag{7.0.2}$$

$$f_{MAX} = \frac{N_{MAX}}{t_G} = \frac{2^M - 1}{t_G} \tag{7.0.3}$$

Pokud časový úsek nastavíme na 1 sekundu, pak je počet pulsů přímo roven frekvenci v Hertzech. U 16-bitového čítače může být problém s maximální frekvencí. Pokud budeme mít měřený úsek dlouhý 1 s, tak maximální frekvence bude pouze 65.536 kHz, při použití 32-bitového čítače bude už maximální frekvence dostačující (cca. 4 GHz což je mnohem víc než maximální povolená frekvence na vstupním pinu).

V případě použití 16-bitový čítačů lze přetečení čítače detekovat přerušením, nebo pomocí stavového registru. Během měřeného časového úseku tak můžeme zaznamenat počet přetečení a dopočítat tak skutečný počet pulsů. Jedinou podmínkou je, že musíme přetečení být schopni zpracovat dřív, než nastane další. To obvykle není problém.

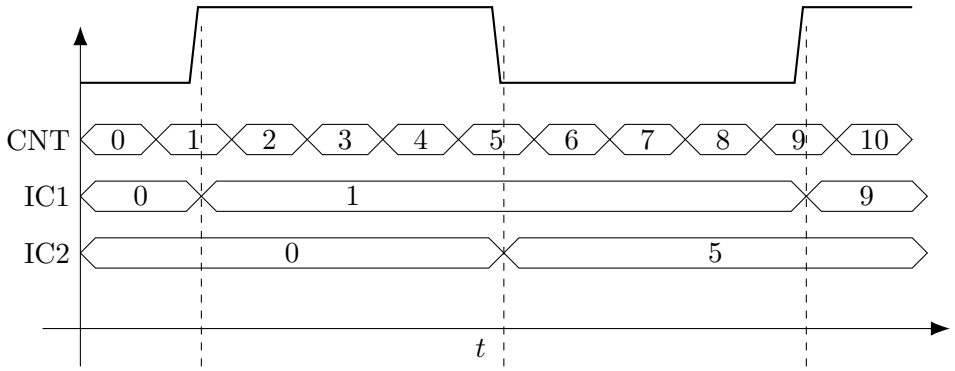
Výhodou toho přístupu je možnost měřit poměrně veliké vstupní frekvence. Nevýhodou je potřeba použití externího hodinového signálu do čítače, který je dostupný pouze na několika málo pinech a čítačích.

7.0.2. Měření frekvence a střídy pomocí tzv. "PWM input mode"

Dalším způsobem jak měřit frekvenci i střídu digitálního signálu je tzv. "PWM input mode". Tento způsob je popsán v referenčním manuálu k mikrořadičům STM32 [2][3].

Pro realizaci potřebujeme 2 "capture-compare" jednotky v režimu "input-capture" (IC). Jedna IC jednotka reaguje na náběžnou hranu vstupního signálu a druhá IC jednotka na spádovou hranu.

Dále je potřeba nakonfigurovat čítač do tzv. reset slave módu. Čítač je pak vynulován při náběžné hraně vstupního signálu, ale zároveň se stále uloží aktuální hodnota čítače pomocí IC jednotky.



Obr. 7.0.1. Princip fungování "Input-capture" (IC) jednotek v režimu "PWM input"

Změřené parametry signálu pak můžeme vyčíst z registrů IC jednotek čítače. První IC registr ($IC1$) obsahuje periodu signálu (v počtech cyklů čítače) a druhý IC registr ($IC2$) pak obsahuje délku kladného pulsu. Při implementaci na STM32 je potřeba dbát na to, že zpoždění resetovacího obvodu čítače je 2 cykly. Proto je potřeba při výpočtech přičíst tyto 2 cykly k hodnotám přečtených z IC registrů. Ve výpočtech jsou tyto hodnoty ($IC1$, $IC2$) již upraveny.

Výslednou frekvenci (f_{IN}), délku pulsu (t_P) a střídu (D) spočítáme ze základní frekvence čítače (f_{TIM}):

$$f_{IN} = \frac{f_{TIM}}{IC1} \tag{7.0.4}$$

$$t_P = \frac{IC2}{f_{TIM}} \tag{7.0.5}$$

$$D = \frac{IC2}{IC1} \tag{7.0.6}$$

Výhodou tohoto přístupu je poměrně snadná a přímočará implementace a také možnost měřit i střídu signálu. Nevýhodou je horší přesnost (bereme v úvahu pouze jeden průběh).

Nastavení měřítka

Pokud jsou vstupní hodiny čítače příliš rychlé (nebo měřený signál pomalý), dojde k přetečení čítače ještě dřív, než stačí zaznamenat další náběžnou hranu. Nejsme tak schopni dopočítat výslednou frekvenci. Naopak pokud budou vstupní hodiny příliš pomalé (měřený signál příliš rychlý), bude měření frekvence nepřesné.

Oba tyto případy jsem schopni na úrovni mikrořadiče detekovat. Přetečení můžeme hlídat pomocí přerušení (případně stavového bitu v registru) a malou přesnost jsme schopni zjistit z naměřených hodnot. Můžeme tak regulovat nastavení před-děličky čítače na základě těchto dvou událostí a měřit tak velký rozsah frekvencí bez nutnosti ručního nastavení rozlišení.

Podobný přístup můžeme použít i další postupy, ale vzhledem k dobrému rozlišení ve výchozím nastavení to není potřeba.

7.0.3. Měření frekvence a střídy pomocí input capture a řadiče DMA

Předchozí způsob můžeme modifikovat, tak abychom byly schopni zaznamenat více hodnot a ty pak průměrovat. Řešením je ukládat hodnoty z IC registrů pomocí řadiče

DMA. Tento způsob je podobný předchozímu způsobu, ale čítač není resetován na náběžnou hranu. Na spádovou, nebo náběžnou hranu je pak spuštěn DMA přenos obou IC registrů do paměti.

Výhodou tohoto přístupu je, že můžeme zaznamenat více period signálu a z toho zprůměrovat požadované parametry. Nevýhodou je větší paměťová náročnost a potřeba DMA řadiče (který by mohl být použit na realizaci jiné funkce). Tento přístup může být vhodný například pro měření stability hodinového signálu. Můžeme tak měřit krátkodobé výkyvy frekvence, nebo střídy (jitter).

7.1. Přesnost měření

Přesnost měření závisí především na přesnosti frekvence čítačů a jejím kolísání. Při použití vnitřních oscilátorů může být nepřesnost poměrně velká. Např. u mikrořadiče STM32F303RE je přesnost oscilátoru $3 - 7\%$ (v závislosti na teplotním rozsahu). Při použití externího krystalu je přesnost výrazně vyšší (obvykle desítky ppm[1]).

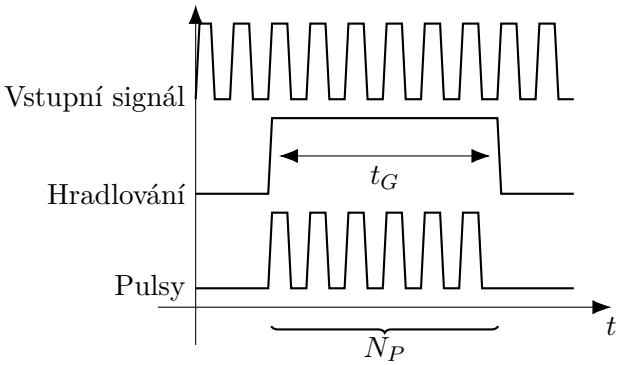
Ještě je třeba vzít v potaz vlastnosti obvodu PLL (fázový závěs). Ten však vytváří pouze dočasné výkyvy frekvence (jitter) a při dlouhodobějším měření je jeho vliv minimální. Význam by to mohlo mít, pokud bychom měřily jitter externího signálu.

7.2. Ukázková implementace na STM32

7.2.1. Čítání pulsů

Pro čítání pulsů potřebujeme 2 čítače v zapojení master-slave. První čítač (slave) počítá pulsy a druhý odpočítává časový okamžik po který pulsy počítáme. První čítač využívá externí hodinový signál skrze tzv. “External clock mode 2” a “gated slave” mód (hradlování vstupního hodinového signálu).

Druhý master čítač má pomocí “Output-compare” jednotky generuje puls o dané délce. Ten je připojen na výstup TRGO (trigger output) čítače. Čítač pracuje v režimu “One-pulse” a po vyslání pulsu se zastaví. Po odeslání naměřených dat a vynulování prvního čítače, se druhý čítač může opět softwarem zapnout.



Obr. 7.2.1. Princip čítání pulsů pomocí hradlování

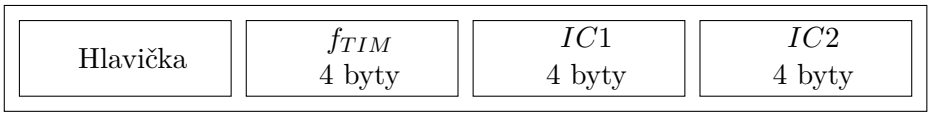
[1] Parts per million — 1000 ppm ≈ 1‰

7.2.2. Mód "PWM input"

Realizace vychází z předchozího popisu režimu "PWM input". Je třeba si uvědomit, že pro realizaci lze použít pouze IC jednotky 1 a 2, protože ostatní jednotky neumožňují resetovat čítač na náběžnou hranu.

7.2.3. Formát komunikace

V režimu "PWM input" se nejdříve pošle základní frekvence čítače v Hz (f_{IN}). Poté následuje délka periody ($IC1$) a délka kladného pulsu ($IC2$) v cyklech čítače (všechny hodnoty jsou reprezentovány 32-bity).



Obr. 7.2.2. Struktura dat v režimu "PWM input"

V režimu čítání pulsu se odešle jedno 32-bitové číslo, které odpovídá počtu pulsů a zároveň naměřené frekvenci (protože doba měření je 1 s).

Čítač přijímá pouze jediný příkaz a to zapni/vypni. Nastavení rozlišení je neměnné, nebo se mění automaticky na základě vstupního signálu.

Příkaz	Hodnota	Poznámka
S	0 = stop, ostatní = start	Zastaví/spustí čítač

Tabulka 7.2.1. Seznam příkazů pro funkci čítač

8. Realizace funkce generátor pulsů na STM32

8.1. Využití vnitřních bloků mikrořadiče

Pro generování digitálních pulsů se používají především vnitřní čítače mikrořadiče. Ty nám umožňují generovat poměrně rychlé signály (obvykle 1/4 nebo 1/2 frekvence jádra). Pro generování použijeme "capture-compare" jednotku v režimu "output compare" (OC). Výstupní frekvenci omezuje kromě frekvence čítače, také budič na výstupním pinu. Mikrořadiče umožňují nastavit různou maximální rychlost (strmost hran) na výstupu. Generovaná frekvence musí být celočíselným podílem vstupní frekvence čítače.

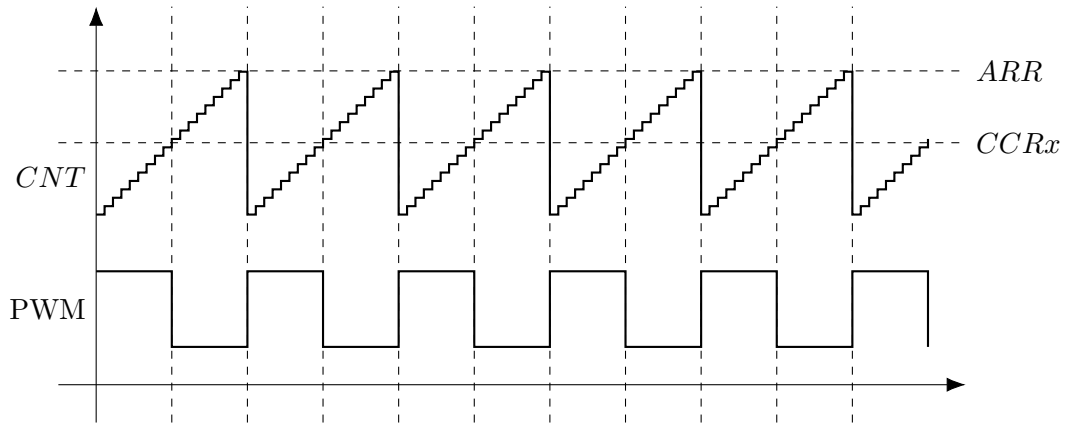
U signálu chceme dále nastavovat střídu. Pokud je signál pomalý, máme větší rozsah nastavitelných hodnot než u signálu pomalého. Pro maximální frekvenci můžeme nastavit pouze střídu 50 %.

8.1.1. Generování pomocí PLL

Pokud bychom chtěli generovat (téměř) libovolnou frekvenci, mohli bychom teoreticky využít vnitřního obvodu PLL. Problémem je, že výstup tohoto obvodu slouží jako hodinový signál mikrořadiče. Mikrořadič by mohl běžet z vnitřního oscilátoru, pak máme ale značně omezenou taktovací frekvenci mikrořadiče. Navíc je potřeba nějaký čas, než se PLL překonfiguruje. Výstupu obvodu PLL je možno připojit na externí pin (tzv. MCO — master clock output).

8.2. Ukázková implementace na mikrořadiči STM32

Parametry výsledného PWM signálu nastavíme pomocí 3 registrů. Registr PSC (prescaler) nastavuje velikost před-děličky čítače. Určuje tedy jak rychle se bude hodnota čítače (CNT) zvyšovat. Druhý registr ARR (auto-reload register) určuje hodnotu, při



Obr. 8.2.1. Princip generování PWM

které se čítače resetuje a začne čítat opět od nuly. Kombinací PSC a ARR získáváme dělící poměr DIV. Posledním registrem je registr dané OC jednotky $CCRx$, který určuje při jaké hodnotě CNT se má signál změnit. Určuje tak střídu signálu. Výslednou frekvenci (f) a střídu (D) získáme:

$$f == \frac{f_C LK}{DIV} = \frac{f_{CLK}}{PSC \times ARR} \tag{8.2.1}$$

$$D = \frac{CCRx}{ARR} \tag{8.2.2}$$

Z toho vyplývá, že čím menší je ARR, tím menší je krok, se kterým můžeme nastavovat střídu signálu.

Pokud tedy chceme generovat danou frekvenci (16-bitovým čítačem), vypočítáme dělící poměr DIV. Tento poměr však musíme rozložit na dvě 16-bitová čísla PSC a ARR. To není triviální úloha. Pro nízké DIV můžeme nastavit $PSC = 1, ARR = DIV$. V případě, že je DIV větší můžeme rozklad provést:

$$PSC = \frac{DIV}{2^{16}} \tag{8.2.3}$$

$$ARR = \left\lfloor \frac{DIV}{PSC} \right\rfloor \tag{8.2.4}$$

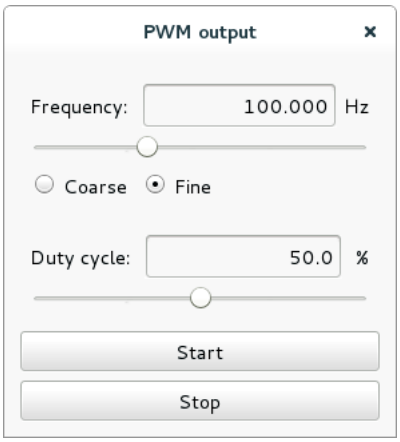
Tento rozklad nevede vždy na optimální výsledek, ale je poměrně jednoduchý.

8.2.1. Komunikace s PC aplikací

Funkce generátor PWM umožňuje vypnout/zapnout generátor a nastavit danou periodu a střídu.

Příkaz	Hodnota	Poznámka
S	0 = stop, ostatní = start	Zastaví/spustí generátor
F	frekvence	Nastaví frekvenci signálu
D	střída (‰)	Nastaví frekvenci signálu

Tabulka 8.2.1. Seznam příkazů pro funkci generátor pulsů



Obr. 8.2.2. PC aplikace

9. Realizace funkce generátor na mikrořadičích STM32

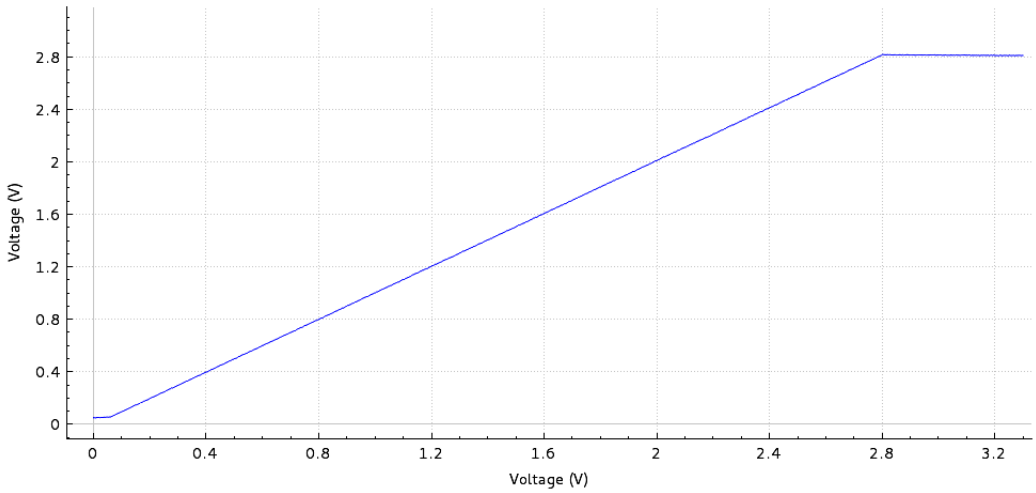
9.1. Využití vnitřních periferií

9.1.1. Použití D/A převodníku

Pro realizaci generátoru potřebujeme D/A převodník. Nabízí se využít vnitřní D/A převodníky mikrořadiče. V závislosti na daném typu je možné připojit až 2 kanály k jednomu D/A převodníku.

Nevýhodou vnitřních D/A převodníků je vysoká vnitřní impedance pokud není zapojen vnitřní zesilovací buffer. Pokud je vnitřní buffer zapnut, dochází ke zkreslení signálu v krajních hodnotách, tedy blízko nulovému a napájecímu napětí. Na Obr. 9.1.1 je znázorněna převodní charakteristika výstupního bufferu při zátěži 1 kΩ proti zemi. Zde se výstup saturuje při vstupním napětí větším než 2.8 V. Měření je provedeno pomocí měřícího modulu popsaném v sekci 11.3.

V některých aplikacích může být výhodné vnitřní buffer vypnout a pro zesílení signálu použít externí zesilovač s lepší charakteristikou. Další možností může být použít externí D/A převodník, který je možno k řadiči připojit např. pomocí rozhraní SPI nebo I2S.

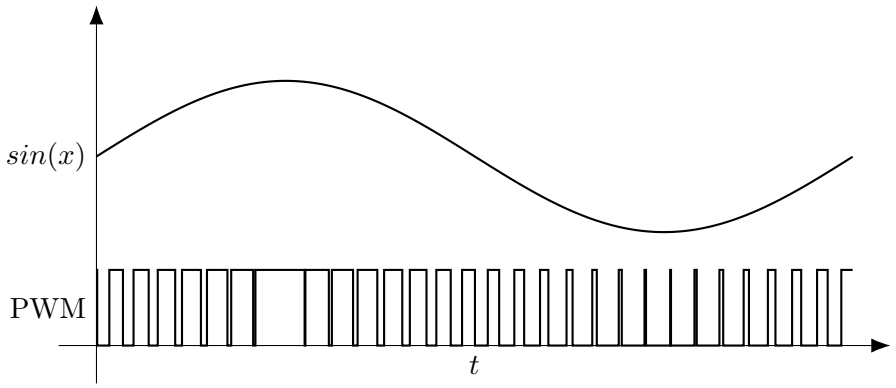


Obr. 9.1.1. Charakteristika D/A převodníku při zátěži 1 kΩ proti zemi

9.1.2. Generování analogových signálů pomocí čítačů

Na některých mikrořadičích nejsou vnitřní D/A převodníky dostupné. Můžeme ale generovat proměnné digitální signály pomocí vnitřních čítačů nebo přímo pomocí GPIO bran. Analogové napětí můžeme nahradit PWM modulací, kde místo napětí měníme střídu signálu. Zapojením externího analogového filtru typu dolní propust pak dostaneme analogový signál. Problémem je, že tento analogový filtr (zvláště filtry 1.řádu)

zkreslí a ztlumí generovaný signál. Tento způsob se ale dá použít pro generování pomalých signálů, případně pro generování statického napětí.



Obr. 9.1.2. Aproximace analogového signálu PWM modulací

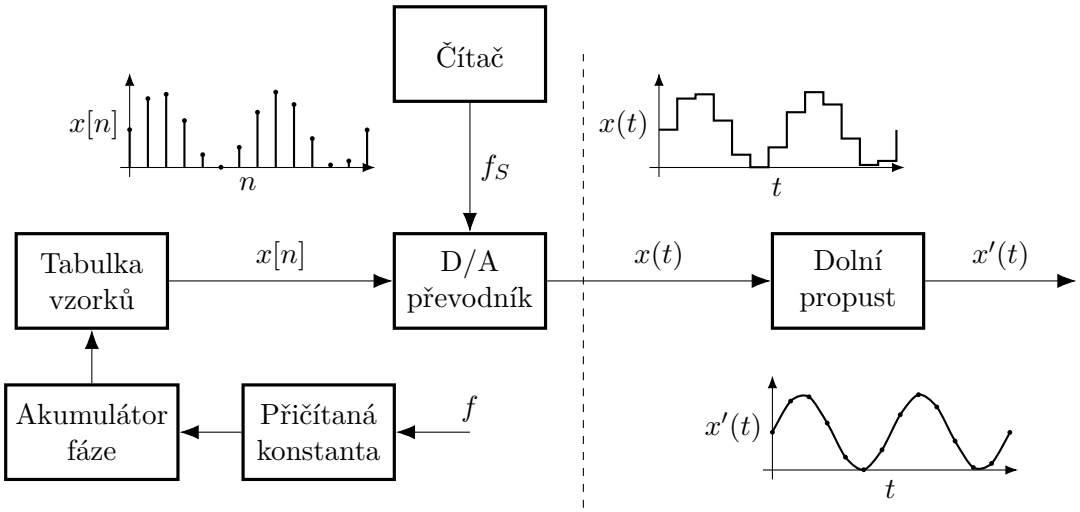
9.1.3. Generování vzorků pro výstup D/A převodníku

Abychom mohli generátor realizovat, potřebujeme kruhový buffer vzorků, které budeme pravidelně posílat na výstup D/A převodníku. Jednou možností je vzorky před-počítat do kruhového bufferu a pak je periodicky posílat. Výstupní frekvence signálu pak bude:

$$f = \frac{f_S}{N} = \frac{f_{MAX}}{N \times DIV} \tag{9.1.1}$$

kde N je velikost bufferu a f_S je vzorkovací frekvence, která je učená jako podíl nějaké maximální frekvence f_{MAX} (obvykle frekvence jádra) a děličkou čítače DIV.

Druhou možností je použít algoritmus DDS (Direct digital synthesis), který umožňuje jemnější nastavení frekvence výstupního signálu (viz. sekce 9.2). Nevýhodou tohoto přístupu je, že se vzorky musejí pravidelně přepočítat. Vzniká tedy vyšší výpočetní zatížení řadiče.



Obr. 9.1.3. Blokové schéma generátoru při použití algoritmu DDS

9.1.4. Určení vzorkovací frekvence

Pro určení vzorkovací frekvence použijeme výstupu jednoho z vnitřních čítačů, kterým je možné spouštět převod D/A převodníku. Obvykle je vhodné použít čítače, které jsou k tomu určeny a neumožňují výstup na piny procesoru.

9.1.5. Vícekanálový generátor

Implementace vícekanálového generátoru je celkem přímočará. V jednom cyklu DDS algoritmu můžeme vygenerovat více signálů naráz. Pro každý signál tak potřebujeme vlastní akumulátor fáze a konstantu, kterou budeme přičítat. Zajímavou možností může být generování dvou fázově posunutých signálů.

9.2. Popis algoritmu DDS

Hlavní prvkem algoritmu DDS je tzv. *akumulátor fáze* (ACC), který je tvořen N-bitovým číslem (v našem případě 32-bitové). Toto číslo reprezentuje aktuální fázi (Φ) v rámci jedné periody:

$$\Phi = \frac{ACC}{2^N} \times 2\pi \tag{9.2.1}$$

v každém kroku (s frekvencí f_S) přičteme k akumulátoru fáze hodnotu ADD, která určuje výslednou frekvenci. Algoritmus počítá s přetečením ACC zpět do nuly. Výsledná frekvence signálu (f_{OUT}) je dána vztahem:

$$f_{OUT} = \frac{f_S \times ADD}{2^N} \tag{9.2.2}$$

Při použití vzorkovací frekvence 1 MHz a 32-bitového akumulátoru fáze dostaneme nejmenší krok pro nastavení frekvence:

$$\Delta f = \frac{f_S}{2^N} = \frac{1\,\mathrm{MHz}}{2^{32}} \approx 233\,\mu\mathrm{Hz} \tag{9.2.3}$$

Dalším prvek algoritmu je tabulka, kde je uložen jedna perioda signálu, který chceme generovat. Velikost tabulky musí být mocnina dvou (2^M). V každém kroku adresujeme tabulku pomocí akumulátoru fáze (ACC), který zaokrouhlíme na horních M-bitů (pomocí bitového posunu). Ve výsledku algoritmus potřebuje jen 3 operace (součet, bitový posun a adresace tabulky) na vzorek.

9.3. Příklady použití

Generátor nemusí být nutně připojen k PC aplikaci. Můžeme program upravit tak, aby šel např. ovládat pomocí tlačítek přímo na zařízení nebo pomocí jiného mikrořadiče/procesoru (skrze UART nebo SPI) v rámci jednoho zařízení.

9.4. Ukázková implementace na mikrořadiči STM32

Ukázková implementace potřebuje pro svou realizace 2 buffery v paměti RAM. V prvním bufferu je uložena tabulka vzorků pro algoritmus DDS. Tato tabulka je vygenerována dynamicky (např. tvar rampa), nebo načtena z paměti ROM (sinusový signál).

V obou případech je ještě tabulka upravena tak, aby vzorky odpovídali dané amplitudě a stejnosměrnému posunutí. Tabulka se tedy vždy přepočítává při změně tvaru signálu, amplitudy a posunutí.

Druhý buffer obsahuje výstupní vzorky algoritmu DDS, které se periodicky posílají na výstup D/A převodníku. Tyto vzorky se pravidelně přepočítávají v reakci na přerušení od řadiče DMA.

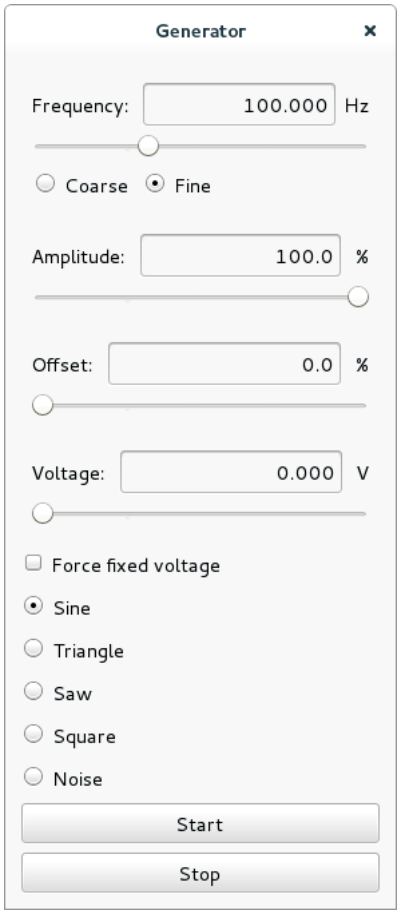
9.4.1. Generátor jako zdroj napětí

Generátor je možné taky použít jako zdroj stejnosměrného napětí. Výhodou sloučení těchto funkcí je, že nemusíme řešit sdílený přístup k periferii (v našem případě D/A převodník). Záleží tedy na uživateli jestli použije funkci generátor nebo zdroj napětí.

9.4.2. Formát komunikace

Funkce generátor podporuje několik příkazů. Většina příkazů nastavuje parametry signálu jako jsou: tvar, amplituda a stejnosměrný posun. Tvar signál je možné měnit za běhu, ale může to způsobit dočasné artefakty na výstupu signálu.

Frekvenci lze také nastavit přímo nastavením přičítané hodnoty do akumulátoru fáze (příkaz “R”). To má výhodu oproti klasickému nastavení frekvence (příkaz “F”), že umožňuje nastavovat velmi jemné rozlišení. Potřebujeme, ale dopředu znát vzorkovací frekvenci generátoru.



Obr. 9.4.1. Uživatelské rozhraní generátoru

Příkaz	Hodnota	Poznámka
S	0 = stop, ostatní = start	Zastaví/spustí generátor
P	0 = sinus, 1 = rampa, 2 = obdélník, 3 = šum, 4 = trojúhelník	Nastaví daný tvar
F	frekvence	Nastaví frekvenci výstupního signálu
A	amplituda: 0 = 0% 2^{16} = 100%	Nastaví amplitudu výstupního signálu
O	posun	Nastaví posun výstupního signálu
V	napětí	Nastaví pevné napětí na výstup
G		Přepne zpět do módu generátor
R	hodnota registru	Nastaví hodnotu registru, který se připočítává k akumulátoru fáze

Tabulka 9.4.1. Seznam příkazů pro funkci generátor

10. Popis ukázkového programu pro STM32

10.1. Struktura projektu

Projekt obsahuje následující složky:

api Obsahuje funkce pro práci s periferiemi.

app Obsahuje hlavní kód aplikace a propojuje ostatní části programu.

modules Obsahuje jednotlivé měřící moduly. Např.: generátor, osciloskop apod.

target Obsahuje kód specifický pro danou platformu.

tools Obsahuje pomocné skripty např. pro generování projektů pro IDE.

Projekt dále obsahuje tyto soubory v kořenovém adresáři:

Makefile Hlavní soubor pro program "make"

build.sh Pomocný skript pro nastavení kompilace. Také umožňuje hromadné spuštění kompilace pro všechny dostupné platformy.

Makefile.user Konfigurační soubor generovaný skriptem "build.sh". Obsahuje např. jméno platformy, pro kterou se bude program kompilovat.

10.2. Nastavení kompilace pro daný mikrořadič

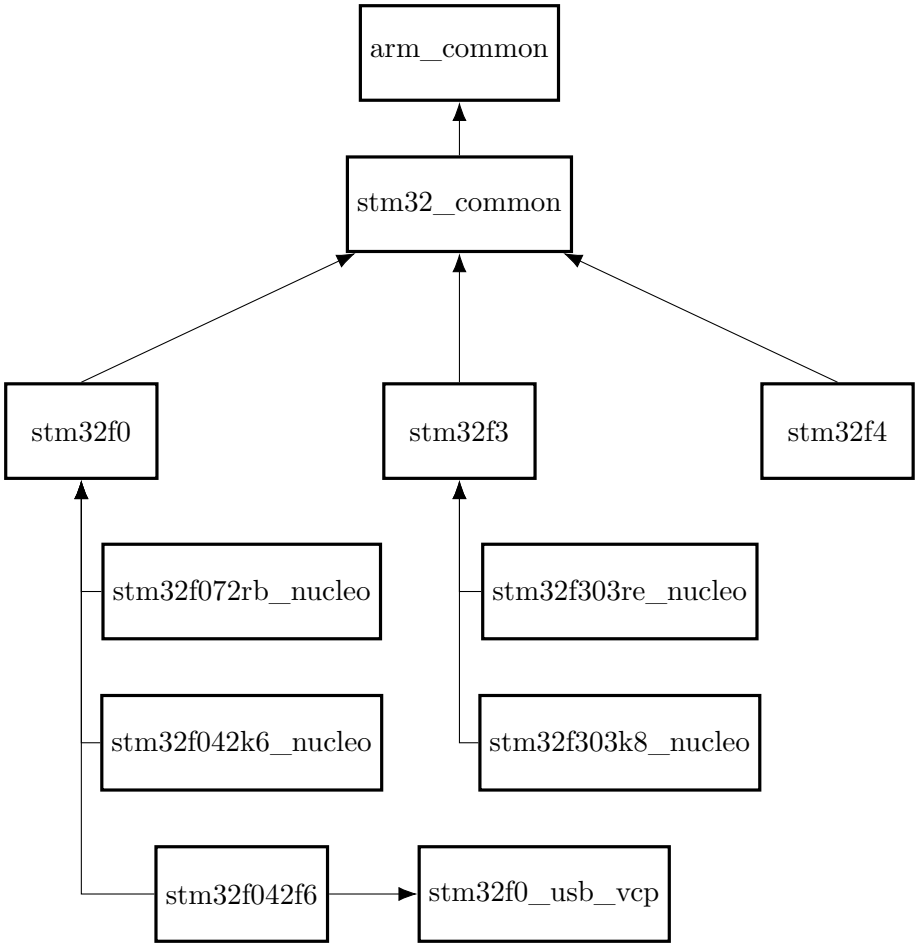
Vzhledem k tomu, že program podporuje různé platformy (mikrořadiče, vývojové kity), je potřeba pro každou platformu správně nakonfigurovat překladač a seznam souborů, které mají být zkompilovány. Jelikož jednotlivé platformy mají některé rysy společné, je program strukturován tak, že platformy se můžou odkazovat na obecnější platformy — tzv. "dědit". Například existují vývojové kity STM32F303K8 Nucleo a STM32F303RE Nucleo, které mají každý jiný řadič i jiné zapojení, ale používají stejnou řadu mikrořadičů. Proto "dědí" od obecné platformy STM32F3.

Některé platformy pak mohou používat další přidané knihovna např. pro ovládání USB. Příkladem je platforma STM32F042F6, která využívá USB knihovnu. Tuto knihovnu připojíme k projektu pouze pokud používáme USB rozhraní, jinak zbytečně zabírá místo v paměti.

"Dědění" konfigurace je realizováno pomocí souborů Makefile a pomocí příkazu "-include". Kdy "potomek" vkládá do svého souboru Makefile soubor Makefile "předka".

10.3. Použité knihovny

Pro implementaci jsem použil knihovnu CMSIS, která obsahuje také definici registrů mikrořadiče, a knihovnu pro ovládání řadiče USB pro STM32F0. Program dále využívá některé funkce standardní knihovny jazyka C.



Obr. 10.2.1. Strom podporovaných platforem a jejich dědičností

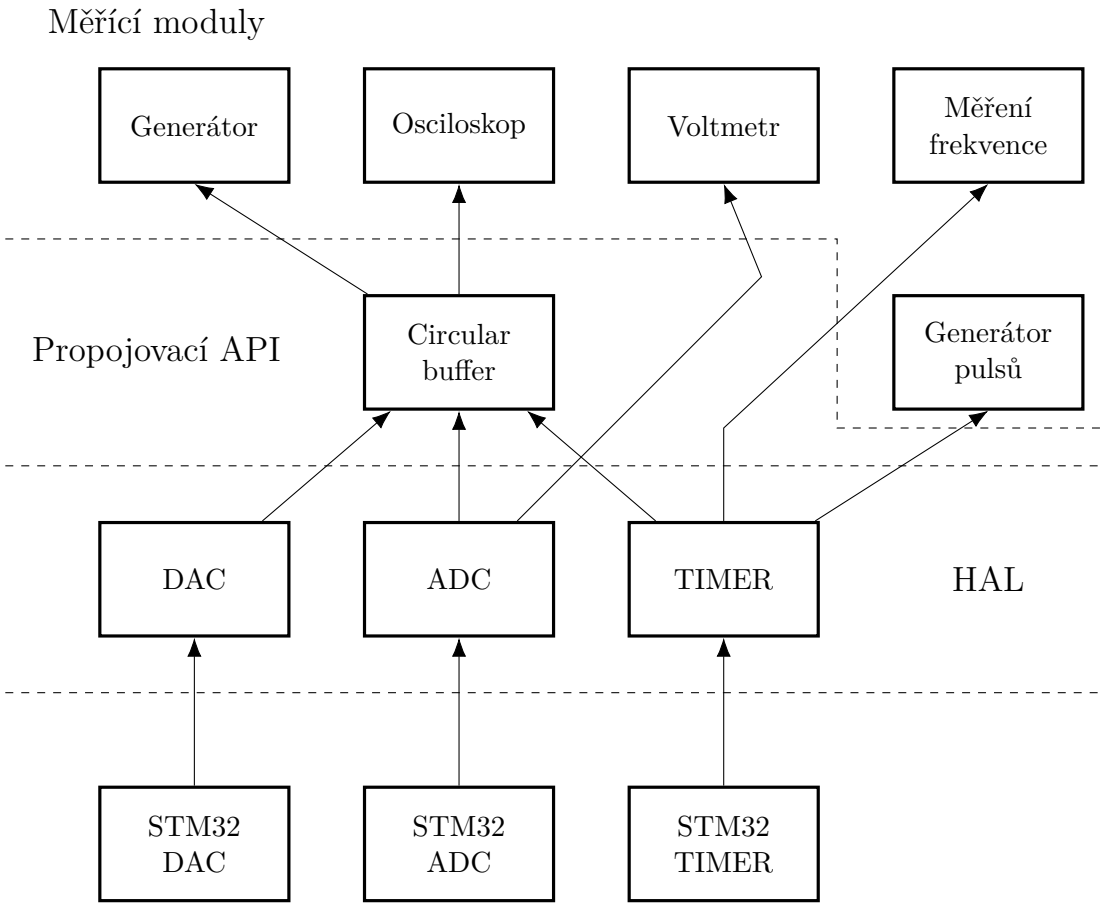
10.4. Implementace ovládání hardwaru

Pro ovládání periferií jsou k dispozici 2 knihovny od výrobce. Jedná se o starší knihovnu "Standard peripheral library" a novější HAL knihovnu, která je součástí generátoru kódu CubeMX. První jmenovaná je již výrobcem označená jako zastaralá a navíc nemá pro různé mikrořadiče stejné aplikační rozhraní (API). Knihovna HAL naopak má jednotné API pro různé mikrořadiče, ale zase má vyšší nároky na paměť.

Nakonec jsem se rozhodl napsat vlastní funkce pro ovládání periferií. Tyto funkce tak přistupují přímo k řídícím registrům jednotlivých periferií. Výjimkou je pouze řadič rozhraní USB, kde jsem použil knihovnu.

10.4.1. Databáze konfigurací

Pro správné nastavení hardwaru je často potřeba znát, které periferie lze spolu propoji (např. na jaké piny lze připojit A/D převodník). Kromě těchto znalostí je třeba také získat informace potřebné pro konfiguraci tohoto propojení (např. o jaký kanál A/D převodníku se jedná). Vzhledem k malému počtu položek v tabulce jsem zvolil sekvenční vyhledávání pro nalezení příslušné konfigurace. Některé moduly tak mohou automaticky vybírat pomocné periferie. Příkladem může být osciloskop, který si sám vybere vnitřní čítač, který bude spoučtě A/D převodník.



Obr. 10.5.1. Vrstvy softwaru

10.4.2. Zamykání periferií

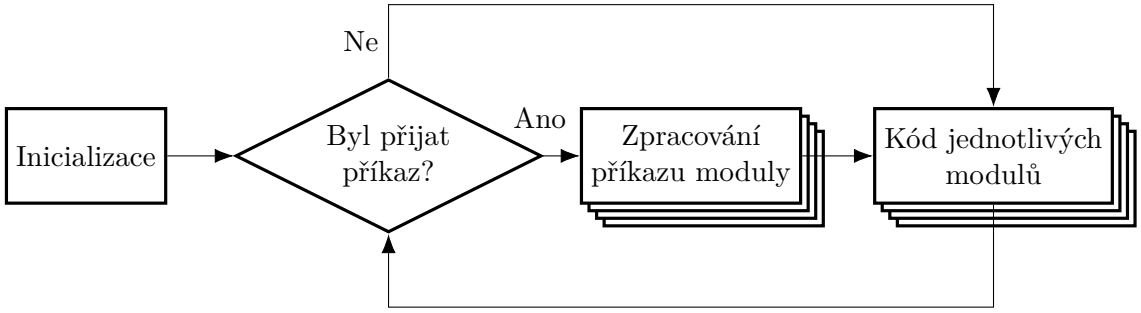
Vzhledem k tomu, že některé programové moduly mohou sami vybírat, který hardware použijí, je třeba zabránit přístupu k jedné periferii z více modulů. Modul pokud chce začne používat danou periferii, tak jí tzv. *zamkne*. Při vyhledávání vhodných periferií se tak vybírají ty periferie, které nejsou zamčené. Zamykání periferií je realizování zápis jedničky do bitového pole na příslušnou pozici, která odpovídá dané periferii.

10.5. Komunikace mezi moduly a hardwarem

Komunikace mezi měřícími moduly a hardwarem probíhá skrze kruhový buffer, který má definované rozhraní, nebo přímo z API konkrétní periferie. Výhodou kruhového bufferu je, že měřící modul lze snadno napojit na jinou periferii mikrořadiče. Příkladem může být osciloskop, který obvykle komunikuje s interním A/D převodníkem, ale v jiné aplikaci bychom chtěli využít externí A/D převodník připojený přes SPI rozhraní. Kruhový buffer je použitý při implementaci funkcí osciloskop a generátor, které jsou složitější a je zde možnost využití externích převodníků.

10.6. Základní smyčka programu

Firmware programu se skládá z inicializace a jednoduché smyčky. Ve smyčce se zkontroluje, zda nepřišel nový příkaz z PC aplikace. Pokud ano, je příkaz předám ke zpracování



Obr. 10.6.1. Vývojový diagram programu

jednotlivým modulům. Poté jsou vykonán periodický kód jednotlivých modulů.

11. Popis ukázkové aplikace pro PC

Ukázková aplikace je napsaná v C/C++ a je postavená na frameworku Qt[1], který slouží nejen pro implementaci uživatelského rozhraní, ale také pro propojení některých měřících modulů pomocí tzv. signálů a slotů. Použití frameworku Qt vyžaduje použití specifických nástrojů pro kompilaci programu jako je např.: qmake.

Dále jsem pro realizaci programu použil tyto knihovny:

- QCustomPlot[2] — knihovna pro vykreslování grafů postavená na frameworku Qt
- fftw[3] — knihovna pro výpočet diskrétní Fourierovy transformace

Obě tyto knihovny jsou volně dostupně pod licencí GNU GPL.

Výhodou použití Qt frameworku a výše zmíněných knihoven je snadná přenositelnost mezi různými operačními systémy (GNU Linux, Windows, OS X) i různými architekturami (x86, ARM).

11.1. Struktura programu

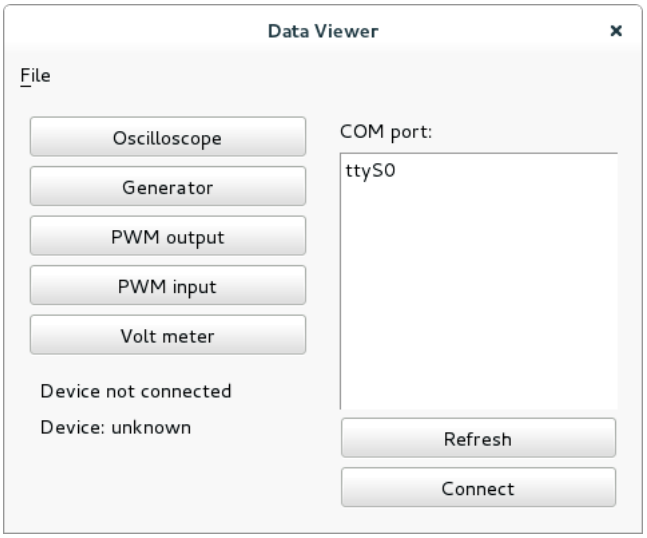
Zdrojový kód je rozdělen do tří složek:

DataUtils Obsahuje pomocné funkce pro příjem a zpracování dat.

FunctionWidgets Obsahuje uživatelské rozhraní pro ovládání měřících modulů.

HelperWidgets Obsahuje pomocné prvky uživatelského rozhraní.

Struktura je podobná jako struktura firmwaru. Obsahuje centrální část, která realizuje komunikaci s mikrořadičem a jednotlivé oddělené měřící moduly. Dále pak ještě obsahuje odvozené měřící moduly (např. měření převodní charakteristiky), které využívají primární měřící moduly (např. voltmetr a zdroj napětí).



Obr. 11.1.1. Úvodní rozhraní PC aplikace

11.2. Komunikace

Komunikace s mikrořadičem probíhá skrze rozhraní USB v režim Virtual COM port. Implementace je postavená na třídě QIODevice. To umožňuje rozšířit aplikaci o další komunikační protokoly (např. TCP/IP).

[1] `http://www.qt.io`
[2] `http://www.qcustomplot.com`
[3] `http://www.fftw.org`

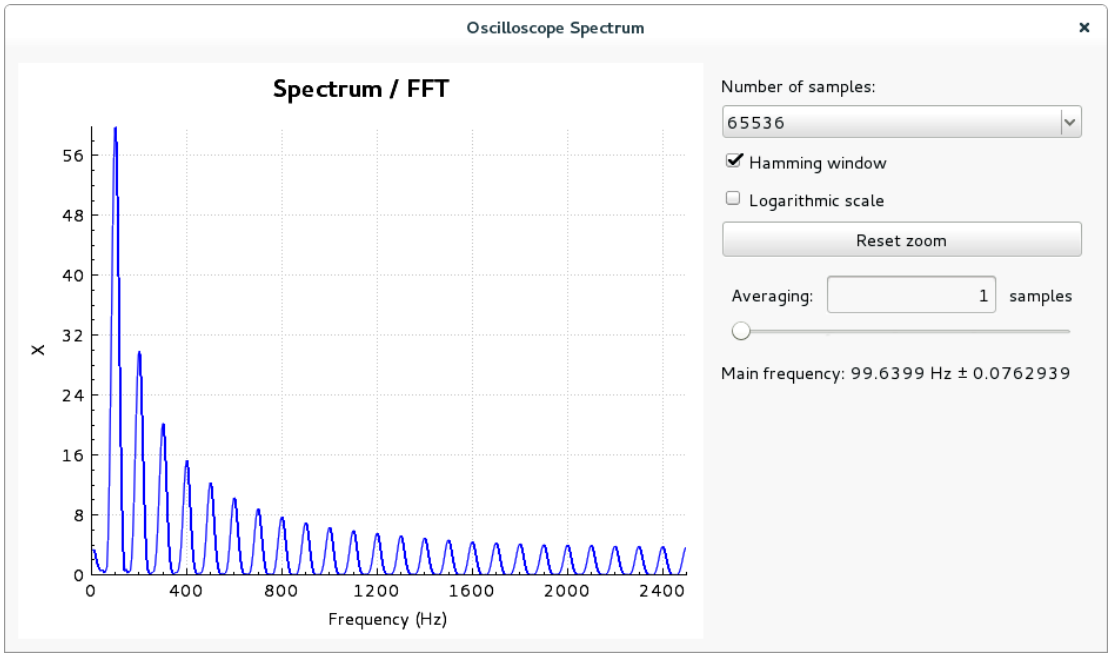
11.3. Rozšiřující měřící moduly

Spektrální analyzátor

Funkce spektrální analyzátor slouží k analýze signálu ve frekvenční oblasti. Pro výpočet se používá algoritmus rychlé diskrétní Fourierovy transformace (FFT). Implementaci zajišťuje knihovna "fftw".

Před samotnou transformací je ještě potřeba vstupní signál upravit. Je vhodné vstupní signál vycentrovat (odečíst stejnosměrnou složku) a vynásobit váhovacím oknem. Váhovací okno odstraní vliv tzv. "prosakování", ale v některých případech může "sloučit" některé spektrální složky. Nakonec ještě můžeme signál doplnit o nuly, čímž získáme detailnější spektrum a můžeme např. přesněji určit výrazné frekvence.

Modul uživateli zobrazí amplitudovou složku spektra. Dále je možné nastavit průměrování spektra. Důležité je počítat průměr z amplitudového spektra. Pokud bychom počítali průměr z komplexního spektra, tak dosáhneme stejného výsledku, jako když zprůměrujeme vstupní signál. To je způsobeno tím, že Fourierova transformace je lineární transformace.



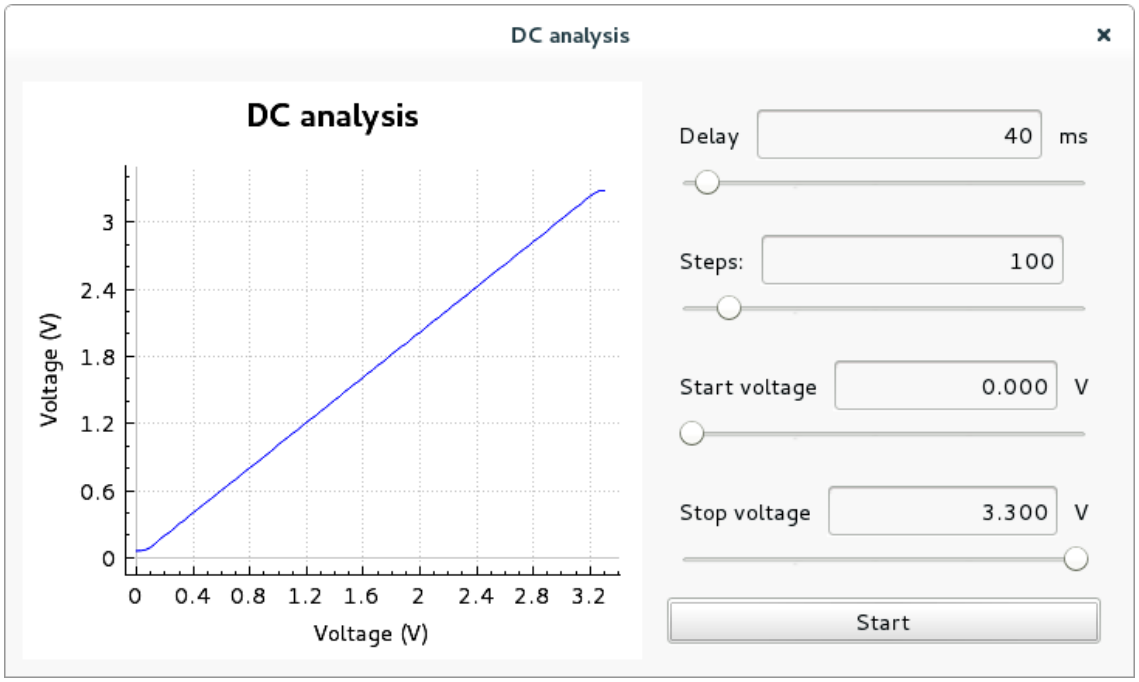
Obr. 11.3.1. Uživatelské rozhraní spektrálního analyzátoru

Měření převodní charakteristiky

Dalším měřícím modulem je měření převodník charakteristiky. To využívá funkcí zdroj stejnosměrného napětí a voltmetru. Přístroj funguje tak, že nastaví napětí na výstup a s odstupem měří napětí na výstupu. V každém kroku se napětí změní o předem nastavený krok. Uživatel si může nastavit v jakém rozsahu napětí bude měřit, kolik kroků se má provést a jaké má být zpoždění mezi nastavením vstupu a měřením výstupu.

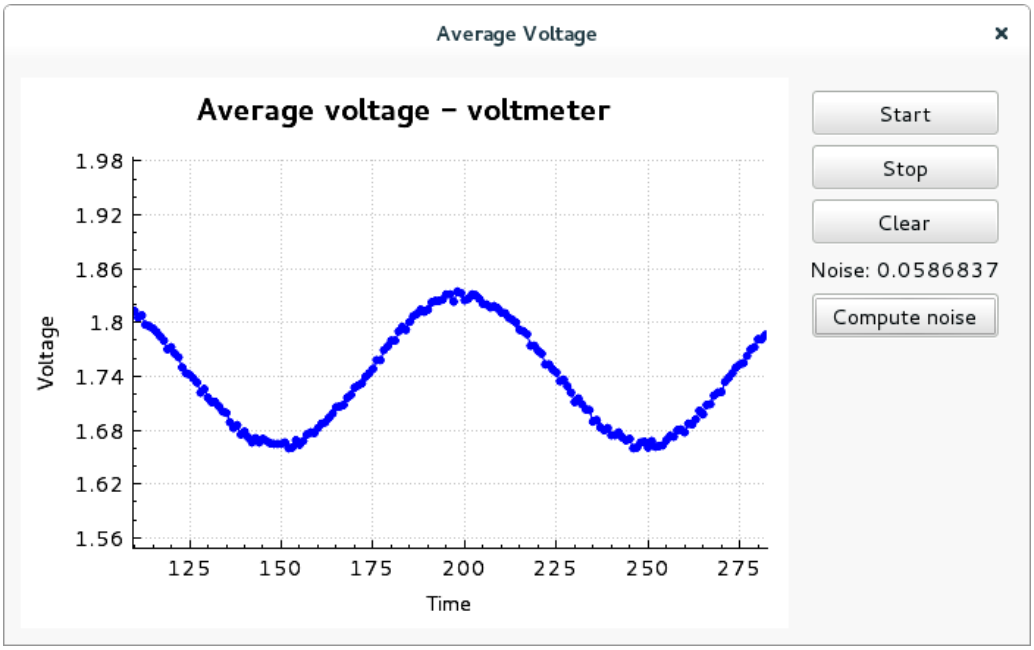
Záznam měření

Modul záznam měření slouží k záznamu jakékoliv měřené hodnoty. Tou můžeme být např. napětí z voltmetru nebo průměrná hodnota napětí z osciloskopu. Díky konceptu



Obr. 11.3.2. Uživatelské rozraní měření převodní charakteristiky

signálu ve frameworku Qt, je možné tento modul snadno napojit na další měřící přístroje.



Obr. 11.3.3. Uživatelské rozraní měření převodní charakteristiky

11.4. Ovládání a manipulace s grafy

Aplikace zobrazuje v měřících modulech okna obsahující grafy s naměřenými hodnotami. Uživatel může tyto grafy posouvat pomocí levého tlačítka myši a přiblížit na

výběr pomocí pravého tlačítka myši. Tímto můžeme změnit i poměr os v grafu. Kolečkem myši je možné graf přiblížit nebo oddálit (obě osy). Na kliknutí pravým tlačítkem myši se objeví kontextové menu které umožňuje:

- Exportovat naměřená data
- Zobrazit/schovat body na křivce
- Vrátit výchozí zobrazení a přiblížení

12. Závěr

Cílem této práce bylo prozkoumat možnosti realizace přístrojových funkcí pomocí řadičů STM32.

Výsledkem je implementace přístrojových funkcí: voltmetr, více kanálový osciloskop s logickým analyzátorem, generátor signálů, který slouží také jako zdroj napětí, pulsní generátor a čítač. Přístroje jsou realizovány pro mikrořadiče STM32F042 a STM32F303 na vývojových kitech Nucleo. Firmware je navržen tak, aby mohl být snadno rozšířen o jiné typy mikrořadičů, či jiné vývojové desky. Také je možné nastavit parametry jednotlivých měřících modulů, případně některé moduly vypnout a dosáhnout tak optimálních parametrů pro danou aplikaci.

Dále jsem vytvořil PC aplikaci sloužící k zobrazování nastavených dat a k ovládání měřících modulů. Aplikace je multi-platformní a snadno rozšiřitelná o nové funkce. Některé zobrazovací komponenty lze snadno použít k dalším účelům.

Příloha A.

Přiložené soubory na CD

Přiložené CD obsahuje text této práce ve formátu PDF. Dále obsahuje tyto složky:

Application zdrojové kódy zobrazovací aplikace pro PC

Application_Dist obsahuje zkompilovanou aplikace pro platformu Windows všechny potřebné DLL soubory

Documents obsahuje katalogové listy a referenční manuály použitých mikrořadičů

Firmware obsahuje zdrojový kód firmwaru

Literatura

[1] STMicroelectronics. *STM32F303xD STM32F303xE Datasheet*. 2015.

[2] STMicroelectronics. *RM0316 Reference manual. STM32F303xB/C/D/E, STM32F303x6/8, STM32F328x8, STM32F358xC, STM32F398xE advanced ARM®-based MCUs*. 2015.

[3] STMicroelectronics. *RM0091 Reference manual. STM32F0x1/STM32F0x2/STM32F0x8 advanced ARM®-based 32-bit MCUs*. 2015.

[4] STMicroelectronics. *AN2834 Application note. How to get the best ADC accuracy in STM32Fx Series and STM32L1 Series devices*. 2013.

[5] STMicroelectronics. *STM32F042x4 STM32F042x6 Datasheet*. 2015.