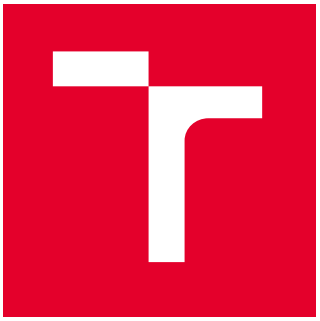


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

SYNCHRONIZACE ČASU PROTOKOLEM IEEE1588

TIME SYNCHRONIZATION BY IEEE1588 PROTOCOL

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — **Bc. Šimon Hříbek**

AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Petr Fiedler, Ph.D.**

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Šimon Hříbek ***ID:*** 220987

Ročník: 2 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Synchronizace času protokolem IEEE1588

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s platformou STM32H747, problematikou synchronizace času protokolem IEEE1588 a principy operačního systému reálného času.

1. Oživte vývojový kit a seznamte se s dostupným operačním systémem.
2. Oživte Ethernetové rozhraní a ověřte jeho funkčnost a funkčnost knihoven pro jeho obsluhu.
3. Navrhněte koncepci jednoduché aplikace na bázi RTOS, která bude využívat IEEE1588 pro opatřování vzorku z AD převodníku časovou značkou.
4. Prověřte možnost využít knihovny IEEE1588 pro realizaci synchronizace na této platformě.
5. Navrhněte koncepci ověření přesnosti synchronizace pomocí osciloskopu.
6. V případě, že se podaří zprovoznit knihovny IEEE1588 proveďte jednoduché ověření, v opačném případě zdokumentujte problémy, které je nutné vyřešit pro využití platformy pro synchronizaci.

DOPORUČENÁ LITERATURA:

1. Dokumentace k vývojovému kitu STM32H747I-DISCO
2. Dokumentace k protokolu IEEE1588 nebo IEC/IEEE 61850-9-3

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 15.5.2024

Vedoucí práce: doc. Ing. Petr Fiedler, Ph.D.

doc. Ing. Petr Fiedler, Ph.D.
předseda rady studijního programu

Abstrakt

Práce se zaměřuje na implementaci časové synchronizace pomocí PTP protokolu na platformu STM32H747 za účelem přiřazovaní přesné časové značky pro získané vzorky pomocí ADC převodníku. Zvolená platforma je konkrétně vývojová deska STM32H747 Discovery Kit. Součástí práce je popis dostupných knihoven potřebných pro implementaci časové synchronizace. Jednou z oblastí je také popis postupu pro zprovoznění ethernetového rozhraní dostupného na vývojovém kitu. Práce se také dotýká volby vhodných metod pro vyhodnocení dosažených výsledků.

Klíčová slova

PTP, časová synchronizace, Precision Time Protocol, STM32, embeded, čas, synchronizace

Abstract

The thesis focuses on implementing time synchronization using the PTP protocol on the STM32H747 platform to assign precise timestamps to acquired samples using the ADC converter. The chosen platform is specifically the STM32H747 Discovery Kit development board. The thesis includes a description of available libraries necessary for implementing time synchronization. One area also covers the procedure for setting up the Ethernet interface available on the development kit. The thesis also addresses the selection of suitable methods for evaluating the achieved results.

Keywords

PTP, time synchronization, Precision Time Protocol, STM32, embedded, time, synchronization

Bibliografická citace

HŘÍBEK, Šimon. Synchronizace času protokolem IEEE1588 [online]. Brno, 2024 [cit. 2024-05-10]. Dostupné z: https://www.vut.cz/studenti/zav-prace/detail/159898. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Petr Fiedler.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	Šimon Hříbek
VUT ID studenta:	220 987
Typ práce:	Diplomová práce
Akademický rok:	2023/24
Téma závěrečné práce:	Synchronizace času protokolem IEEE1588

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

.
V Brně dne: 9. května 2024

Poděkování

Děkuji svému vedoucímu doc. Ing. Petru Fiedlerovi, Ph.D., za jeho rady, ochotu a odbornou a metodickou pomoc při vypracování mé diplomové práce.

V Brně dne: 09.5.2024

Obsah

SEZNAM OBRÁZKŮ

Úvod

Časová synchronizace je přítomna v každé oblasti, liší se pouze v požadavcích dané aplikace. Každý zdroj hodinového signálu je totiž nedokonalý a dochází k většímu či menšímu odchýlení od reálného času. Z toho důvodu bylo potřeba zavést způsoby, jak synchronizovat více zařízení na co nejpodobnější čas. Celosvětově nejpoužívanějším způsobem synchronizace zařízení je synchronizaci používající protokol NTP (Network Time Protocol), který je využíván například při synchronizaci zařízení skrze internet. Způsobů synchronizace času jednotlivých zařízení je celá řada a každý z nich má své výhody a nevýhody.

Oblast měření vyžaduje vysokou časovou přesnost, a proto je potřeba zde zajistit co nejlepší časovou synchronizaci. Při měření a zpracování na rozdílném zařízení často dochází v rámci jedné lokální sítě (LAN) a proto může být vhodnou volbou pro časovou synchronizaci protokol PTP.

Práce je zaměřena na implementaci PTP protokolu na platformě STM32H747.

První část práce se věnuje teoretické části zvolené problematiky. Cílem je dostatečně seznámit čtenáře se základním mechanismem a vlastnostmi časové synchronizace pomocí PTP protokolu, význam jeho jednotlivých zpráv a obecné porovnání s ostatními dostupnými synchronizačními protokoly.

Další část má za úkol zjistit dostupné možnosti knihoven pro podporu PTP protokolu na zvolené platformě. Ne každá platforma má dostupné vhodné prostředky či umožňuje podporu všech požadovaných vlastností.

Součástí práce je také zprovoznění ethernetového rozhraní. Jelikož je výchozí platformou takzvaný Discovery kit, je potřeba pro uvedení této periferie do provozu provést řadu kroků, a to jak na hardwarové úrovni, tak i na úrovni softwaru a práci s pamětí.

Závěrečná část práce se věnuje diskusi dosažených cílů. Jsou zde popsány problémy, které byly v průběhu práce řešeny a také je zde navržena metodika měření dosažených výsledků a základní návrh jejich interpretace.

1. ROZBOR ZADÁNÍ A POPIS CÍLŮ

Účelem této kapitoly je seznámit čtenáře s popisem zadání a cílů této práce. Jednotlivé podkapitoly obsahují interpretaci dílčích bodů zadání.

1.1 Oživte vývojový kit a seznamte se s dostupným operačním systémem.

Pro tuto práci byla zvolena platforma STM32H747. Platforma je více popsána v pozdější kapitole. Cílem práce však není podrobný popis použitého HW. Proto se popis platformy věnuje pouze periferiím relevantním této práci.

Konkrétně jsou v práci popsány následující oblasti: CPU, USART, ethernet, USB, LED, RTC a ADC převodník.

Z logických důvodů je detailněji popsán mikrokontroler STM32H747. Pozornost je věnována tomu, že použitý procesor má dvě jádra. V práci bude použito primárně jádro M4 zatímco jádro M7 využito v podstatě nebude.

STM32H747 má podporu operačního systému reálného času, konkrétně je zde podpora pro freeRTOS. Konfigurace tohoto operačního systému a jeho vlastnosti jsou popsány v kapitolách 4.7 a 6.3.

1.2 Oživte ethernetové rozhraní a ověřte jeho funkčnost a funkčnost knihoven pro jeho obsluhu.

Pro síťové použití bude využito ethernetové rozhraní. Bude využita lwip knihovna/middleware. Implementaci se věnuje samostatná kapitola. Implementace na každé platformě je specifická. Je zde potřeba udělat změny HW, správně nakonfigurovat SW a paměťový prostor.

Pro použití ethernetového rozhraní je použita knihovna HAL. Dále je zde využit middleware lwIP. LwIP je zkratkou LightweightIP. Tomuto bodu zadání se věnuje kapitola pro konfiguraci ethernetu

1.3 Navrhněte koncepci jednoduché aplikace na bázi RTOS, která bude využívat IEEE1588 pro opatřování vzorku z AD převodníku časovou značkou.

Návrh koncepce aplikace pro získávání vzorků z analogově digitálního převodníku je tématem, o kterém pojednává kapitola zaměřená na nastavení freeRTOS. Jedná se o operační systém, který je na zvolené platformě dostupný, a i proto bude využit i v tomto bodě zadání.

Popis AD převodníku, freeRTOS i dalších periferií využitých pro tuto aplikaci jsou popsány v samostatných kapitolách či podkapitolách.

1.4 Prověřte možnost využít knihovny IEEE1588 pro realizaci synchronizace na této platformě.

Cílem práce je také zjistit dostupnost a stav vhodných knihoven potřebných pro implementaci aplikace předchozího bodu zadání. Je potřeba také zjistit postupy potřebné pro implementaci na cílové platformě a podmínky, které jsou pro takovou aplikaci stanoveny výrobcem hardwaru.

1.5 Navrhněte koncepci ověření přesnosti synchronizace pomocí osciloskopu.

Součástí zadání je také tvorba metodiky pro měření dosažené přesnosti časové synchronizace pomocí výše zvoleného protokolu IEEE1588 neboli PTP. Samostatná kapitola této práce se věnuje připraveným metodikám a postupům pro změření dosažených výsledků.

Primárním zaměřením této kapitoly je využití osciloskopu jako měřícího přístroje.

1.6 V případě, že se podaří zprovoznit knihovny IEEE1588 proveďte jednoduché ověření, v opačném případě zdokumentujte problémy, které je nutné vyřešit pro využití platformy pro synchronizaci.

V případě, že bude možné výše navrženou aplikaci implementovat na zvolenou platformu, je možné provést ověření pomocí navržené metodiky měření dosažené přesnosti.

V opačném případě je cílem tohoto bodu zadání zdokumentovat problematické oblasti a navrhnout další kroky, které bude potřeba provést pro vhodnou implementaci žádané aplikace.

1.7 Literární rešerše

Součástí práce je také literární rešerše, která má za úkol zajistit dostatek informací o problematice časové synchronizace pomocí PTP protokolu. Rešerše má za úkol předat čtenáři dostatek informací, aby pochopil fungování a princip časové synchronizace. Práce se problematikou nezaobírá příliš detailně, neboť by se pak jednalo o pouhý přepis standardu IEEE 1588.

Čtenář je v práci seznámen s důležitostí časové synchronizace zařízení. Práce popisuje různé možnosti a protokoly týkající se časové synchronizace.

Další kapitola se věnuje samotnému protokolu PTP. Je zde popsán základní algoritmus časové synchronizace za použití protokolu IEEE 1588v2. Čtenář je také seznámen se základními typy protokolu. Dále jsou popsány jednotlivé zprávy použité při komunikaci. Text práce se také dotýká problematiky topologie sítě vztahů mezi jednotlivými členy. Pozornost je také věnována problematice BMCA algoritmu, který je využíván pro volbu zdroje času – takzvaného GMC zařízení.

Poslední podkapitolou je popis platformy, na které je tento projekt implementován. Jedná se o zařízení o firmy ST, konkrétním modelem je Discovery kit STM32H747LM. Kapitola pojednává o pro tuto práci relevantních záležitostech, a to konkrétně:

- mikrokontroler
- Ethernet
- ST-link
- AD převodník
- UART
- RTC

1.8 Vyhodnocení průběhu práce

Součástí práce je také závěrečná kapitola, která se věnuje zhodnocení práce na dané platformě a popisuje například také přívětivost uživatelského rozhraní a aplikací.

2. ČASOVÁ SYNCHRONIZACE

Časová synchronizace je veliká oblast, která je hojně využívána v mnoha odvětvích průmyslu, například informačních technologii. Tato kapitola přibližuje význam časové synchronizace obecně. Jsou zde popsána její úskalí a také nastíněna některá její řešení a postupy, které jsou, nebo mohou být použity.

2.1 Význam

Časová synchronizace je podstatná v aplikacích, které vyžadují paralelizaci měření a řízení s vysokou přesností. V těchto aplikacích je tedy potřeba zajistit, aby se vnitřní časy daných uzlů lišily co nejméně.

Pro časovou synchronizaci je možné využít několik možných postupů, protokolů a standardů. Mezi možné varianty patří například IRIG-B, NTP nebo PTP. Z těchto zmíněných je nejstarším IRIG-B. Hojně používaný je NTP (Network Time Protocol), který využívá synchronizaci po síti. NTP protokol je velice podobný PTP protokolu, ale jako zdroj hodin používá lokální zdroj hodin. PTP je detailněji popsán ve standardu IEEE 1588.

2.2 Jednoduchý princip synchronizace

Existuje jednoduchý princip, jak synchronizovat čas na více zařízeních. Toto řešení je jednoduché na implementaci i na pochopení, nedosahuje však vysoké přesnosti. Postup je následující: zdrojové zařízení, které je považováno jako zdroj přesného času, pošle zprávu s aktuálním časem. Zařízení, na kterém je cílem upravit čas změní svůj vnitřní čas na základě přijaté zprávy. Tento postup však neošetřuje nepřesnost synchronizace danou samotnou komunikací ani zpracováním zprávy na obou koncích.

Z tohoto důvodu je u aplikací, u kterých je potřeba dosáhnout vysokých přesností, používán hojně PTP standard. Využití PTP protokolu je však limitováno na použití na lokální síti. Pro aplikace, které potřebují využít synchronizaci mimo lokální síť, je možné využít například podobný protokol zvaný NTP či SNTP. Všechny tři zmíněné protokoly jsou založeny na podobném principu, nicméně dosahují jiných přesností, a proto i jejich použití je rozdílné.

2.3 Možné standardy a způsoby

V průběhu let došlo k vytvoření řady synchronizačních standardů, které slouží k časové synchronizaci. Každý standard je něčím specifický a má své využití. Některé jsou vzájemnými alternativami, některé jsou naopak vhodné pouze pro určité použití, zatímco pro jiné účely je jejich využití nevhodné. Následující podkapitoly popisují vybrané běžně používané standardy a protokoly. Práce také obsahuje samostatnou kapitolu, která se

věnuje pro tuto práci podstatnému PTP protokolu. Účelem této kapitoly je seznámit čtenáře se širším kontextem a možnostmi časové synchronizace.

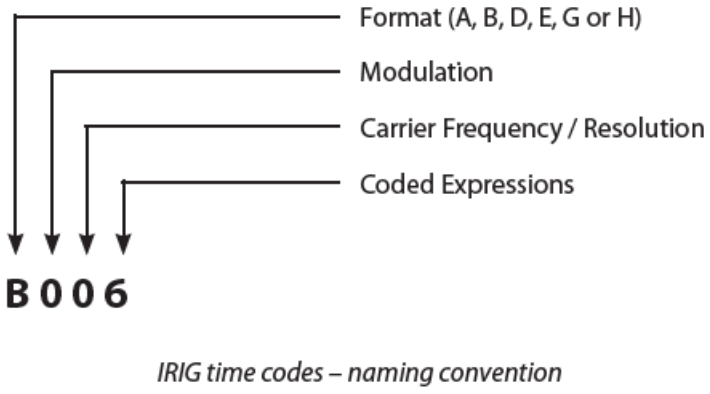
2.3.1 IRIG-B

Tento standard je již poměrně starý, jeho počátky sahají do padesátých let minulého století. Stále je však využíván v průmyslu, hlavně díky své vysoké přesnosti a širokému použití v minulosti. Dnes je stále využíván také v odvětví energetiky či vojenských systémů. Nově vytvořené systémy a aplikace využívají novější protokoly a metody synchronizace. Nicméně z důvodů zpětné kompatibility je IRIG-B na některých místech stále využíván.

I přes jeho stáří je možné pomocí IRIG-B dosáhnout přesností v mezích ms nebo ns. Standard sám o sobě definuje časovou základnu a formát času. Dále je ve standardu definován způsob přenosu časové značky. Jedná se o kódování ve formátu denVRoce, hodina, minuta, sekunda. Je možné ke zprávě přidat další doplňující informace. Nosná frekvence informace je 1 kHz. [14][15]

Největší nevýhodou protokolu IRIG-B je jeho potřeba samostatného vedení a specifického hardwaru. Tyto aspekty zhoršují flexibilitu a škálovatelnost systému. Zároveň potřeba vlastního spojení čistě z důvodu časové synchronizace nezanedbatelně zvyšuje cenu aplikace.

Zkratka IRIG znamená Inter Range Instrumentation Group. Jedná se o název organizace, která tento standard vymyslela. IRIG-B je součástí rodiny více synchronizačních protokolů IRIG (A až H). Tyto verze se liší v parametru bit-rate. Konkrétně použitý IRIG může být také označen konkrétnějšími parametry ve formátu znázorněném na obrázku.



Obrázek 1 Vysvětlení pojmenování různých variant standardu IRIG[15]

2.3.2 Loran – C

Tento standard byl zařazen do této práce, aby byl čtenář obeznámen s tím, že existují i jiné než vojenské či čistě průmyslové standardy a protokoly.

Jedná se o protokol, který je používán především v pobřežních vodách kolem USA.

2.3.3 NTP

Protokol zaměřený na synchronizaci mimo lokální (LAN) síť. Protokol je používán pro synchronizaci se zdrojem času mimo LAN. Princip komunikace je velice podobný jako u PTP.

NTP protokol je používán také při synchronizaci zařízení připojených na světový internet. To z něj dělá jeden z nejhojněji používaných protokolů pro časovou synchronizaci.

2.3.4 SNTP

V podstatě se jedná o modifikaci NTP protokolu. Pro některé aplikace není potřeba používat plnohodnotný NTP protokol se všemi jeho možnostmi, ale stačí využít pouze zjednodušenou verzi.

2.3.5 PTP

Jedná se o protokol, který je založen na lokální synchronizaci. Základní princip fungování tohoto protokolu je velice podobný protokolu NTP. Protokol je podrobněji popsán v samostatné kapitole 3.

3. PRECISION TIME PROTOCOL (IEEE 1588)

Kapitola se zaměřuje na samotný protokol PTP (IEEE 1588). Primárním zdrojem informací byl samotný standard IEEE 1588 ve verzi 2 z roku 2008[1].

3.1 Obecné informace

3.1.1 Verze

Protokol lze dělit na verze (v1, v2, v2.1). Ty reflektují jeho postupné vylepšování v průběhu času. Aktuální verze je verzí v2.1. Práce se však věnuje použití verze PTP v2 z roku 2008, která je nejčastěji používaná a v2.1 je s ní zpětně kompatibilní.

Verze v2.1 přináší výhody především pro aplikace v rozsáhlých sítích, protože jedním z hlavních přínosů nové verze je možnost kombinovat multicast a unicast. Dalším přínosem nejnovější verze je vylepšené zabezpečení. Nicméně dostupné knihovny jsou zaměřeny na implementaci verze z roku 2008, v2, a proto je také použita v této práci. Implementace verze v2 neomezí použití zařízení v síti s dalšími zařízeními používající verzi v2.1.[5]

3.1.2 Profily

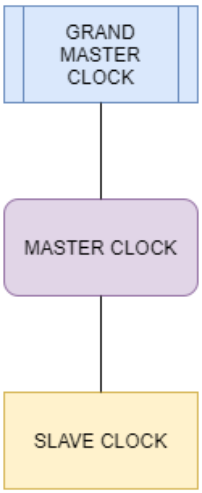
Dalším dělením je použití definovaných profilů. Použití profilů zjednodušuje implementaci především v rozsáhlejších sítích s větším počtem synchronizačních PTP uzlů.

Jednotlivé profily definují vyžadované parametry či funkce, které jsou od použitého zařízení vyžadovány. Toto zjednodušuje správu a rozšiřování aktuální sítě PTP uzlů. Nově připojené zařízení se stejným profilem jako již stávající zařízení bude po připojení do již existující sítě fungovat bez potřeby velikého zásahu do konfigurace.[3][5]

3.2 Základní prvky a názvy v synchronizaci PTP

Pro správné pochopení fungování PTP protokolu je potřeba rozumět významům jednotlivých názvů členů synchronizace. V této kapitole jsou vysvětleny jednotlivé prvky vyskytující se v PTP.

Jedno zařízení může být označeno vícerem následujících názvů, jelikož může zastávat více rolí. Často k tomuto dochází u ED zařízení, které je už z principu samozřejmě také SC zařízením nebo GMC zařízení může být označeno jako MC zařízení.



Obrázek 2 Topologie PTP základ

3.2.1 Grand Master Clock (GMC)

Zařízení, které je bráno jako zdroj správného času. Tento PTP node má všechny své PTP porty nastaveny jako Passive nebo jako Master. Jedná se o zařízení, které je zdrojem, nikoliv příjemcem reálného času.

3.2.2 Master Clock (MC)

Zařízení označeno jako MC je při synchronizaci bráno jako zdroj. Toto zařízení také zahajuje komunikaci. Korektnějším vyjádřením je, že se jedná o stav PTP portu. Pojem MC se používá spíše pro vyjádření vztahu a charakteru komunikace mezi dvěma uzly v síti. Druhým účastníkem a protistranou komunikace je zařízení SC. [3][1]

3.2.3 Slave Clock (SC)

SC je zařízení, které při synchronizaci mění svůj vnitřní čas tak, aby co nejblíže odpovídal času na zařízení MC. [1][3]

3.2.4 Edge device

Jako edge device je označeno zařízení, které je v hierarchii komunikace postaveno nejníže a je komunikačně nejdál od GMC. Jedná se typicky o zařízení, které v souvislosti s PTP pouze používá synchronizaci a mění svůj vnitřní čas. [1]

3.2.5 Ordinary Clock (OC)

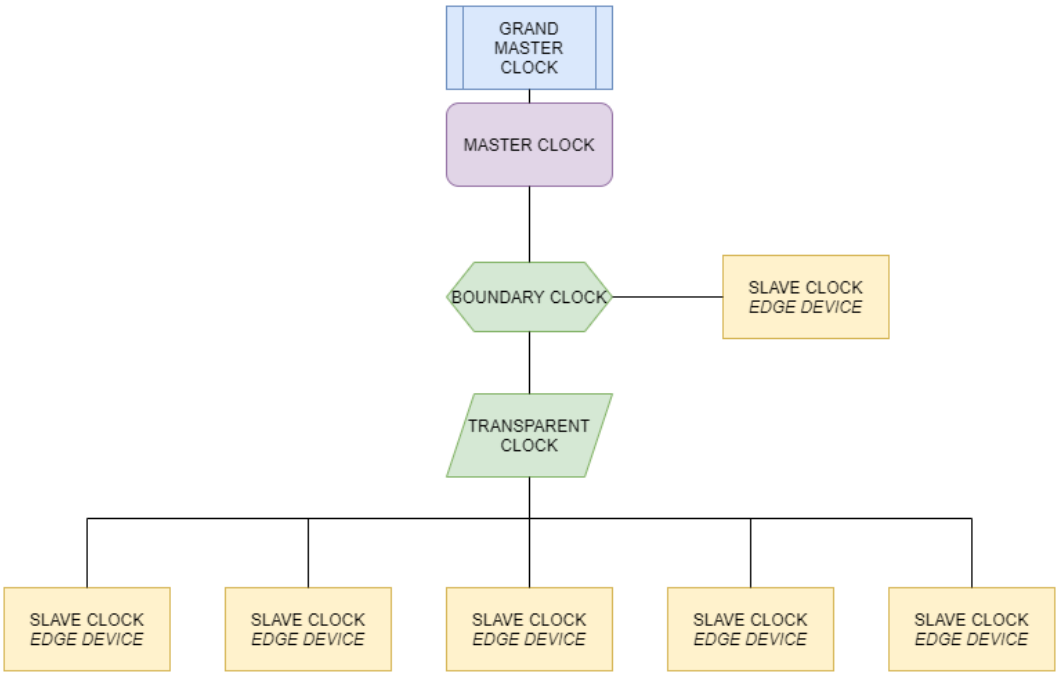
Jako OC je označeno zařízení, které má aktivní pouze jeden PTP port, přesněji pouze jeden port používá pro synchronizaci PTP. Zpravidla se jedná o ED v režimu SC, které vyžaduje synchronizaci. Může se však také jednat o zařízení v režimu MC, které poskytuje čas jinému zařízení. [6][1]

3.2.6 Boundary a Transparency Clock (BC a TC)

Rozsáhlejší síť je vhodné segmentovat, aby nedošlo k přesycení GMC (či MC) zařízení požadavky SC zařízení. K tomu může dojít v případě, že je potřeba čas synchronizovat s vysokou frekvencí na velikém množství zařízení. Také z těchto důvodů je velikost sítě limitována jak počtem zařízení, tak počtem zanoření.

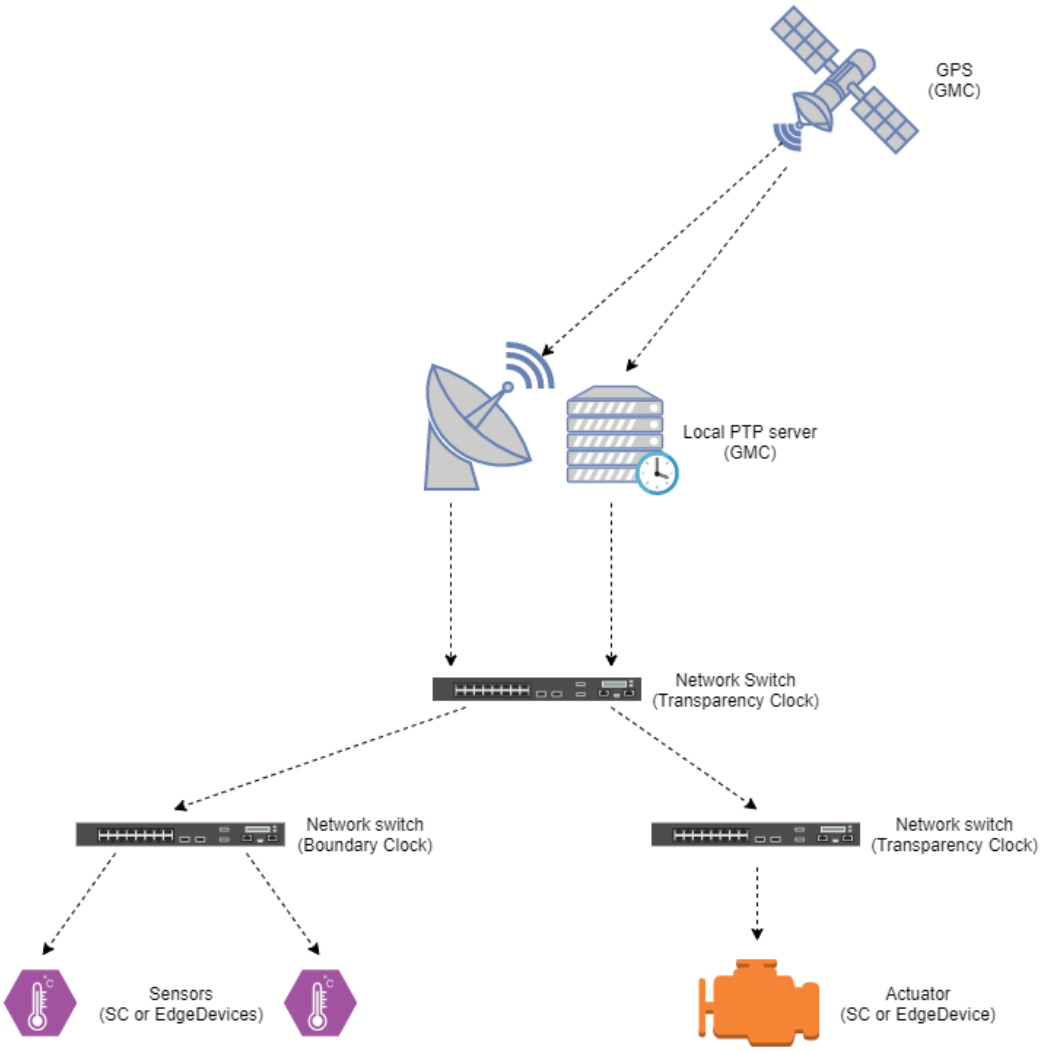
Jedním z řešení tohoto problému je použití BC zařízení, které snižuje zátěž nadřazeného MC zařízení tím, že se stane samo jak MC, tak SC zařízením. Toho je docíleno použitím více PTP portů, tudíž se nejedná o zařízení OC. Fungování BC je takové, že jeden port v režimu SC synchronizuje vnitřní hodiny. Druhý port je pak v režimu MC a umožňuje synchronizaci dalšího podřazeného zařízení SC.

TC zařízení funguje na jiném principu. TC zařízení slouží jako přechodná stanice pro zprávy. Zařízení zná dobu potřebnou pro proces přijetí, zpracování a přeposlání zprávy a je schopno zprávu upravit tak, aby toto zpoždění vyrovnalo/zkorigovalo. Z toho vyplývá, že toto zařízení musí podporovat tuto funkci. TC můžeme dělit na dva typy: E2E a P2P. Jejich rozdíl je v tom, mezi kterými body dochází k dotazu na zpoždění komunikace. Režim E2E zajišťuje komunikaci mezi ED a GMC. U P2P probíhá část komunikace mezi GMC a ED a část pouze mezi TC a ED a TC a GMC. [6][7]



Obrázek 3 Topologie PTP – rozsáhlejší síť

Obrázek 3 popisuje rozsáhlejší síť, kde je možné použít synchronizaci pomocí PTP protokolu. Síť obsahuje jeden zdroj hodin, dva mezistupně v podobě BC a TC zařízení a šestici SC zařízení, které jsou synchronizovány na čas GMC zařízení.

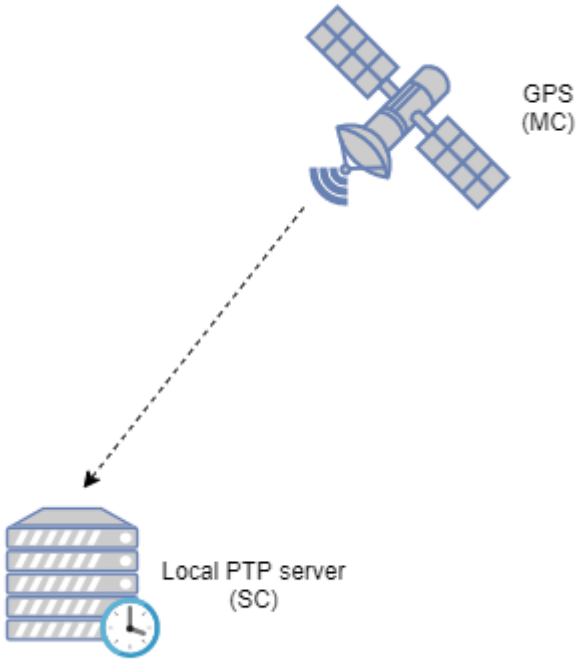


Obrázek 4 Příklad struktury pro PTP synchronizaci

Obrázek 4 popisuje příklad z reálného světa, kde je síť obsahující sadu snímacích zařízení (označeny jako senzory), jeden akční člen (actuator), tři přepínače a dva GMC. Primárním GMC je zvoleno zařízení, které má jako svůj zdroj času satelity GPS. Jako redundantní neboli záložní zdroj času je také přítomen lokální zdroj hodin. Tento lokální zdroj hodin se synchronizuje nezávisle na této síti přímo se systémem GPS.

Tato síť obsahuje na nejnižší úrovní BC zařízení. To zde mohlo být zvoleno proto, aby nebyl hlavní GMC server přetížen synchronizací všech ED zařízení. TC zařízení je přítomno na úrovni nad akčním členem, nebo jako nejvýše postavený přepínač, který umožňuje časovou synchronizaci přepínačů na nižší úrovni. Počet synchronizovaných zařízení je menší a nehrozí tedy přehlcení komunikace GMC.

Na následujícím obrázku je znázorněna komunikace mezi lokálním zdrojem času PTP a GPS.



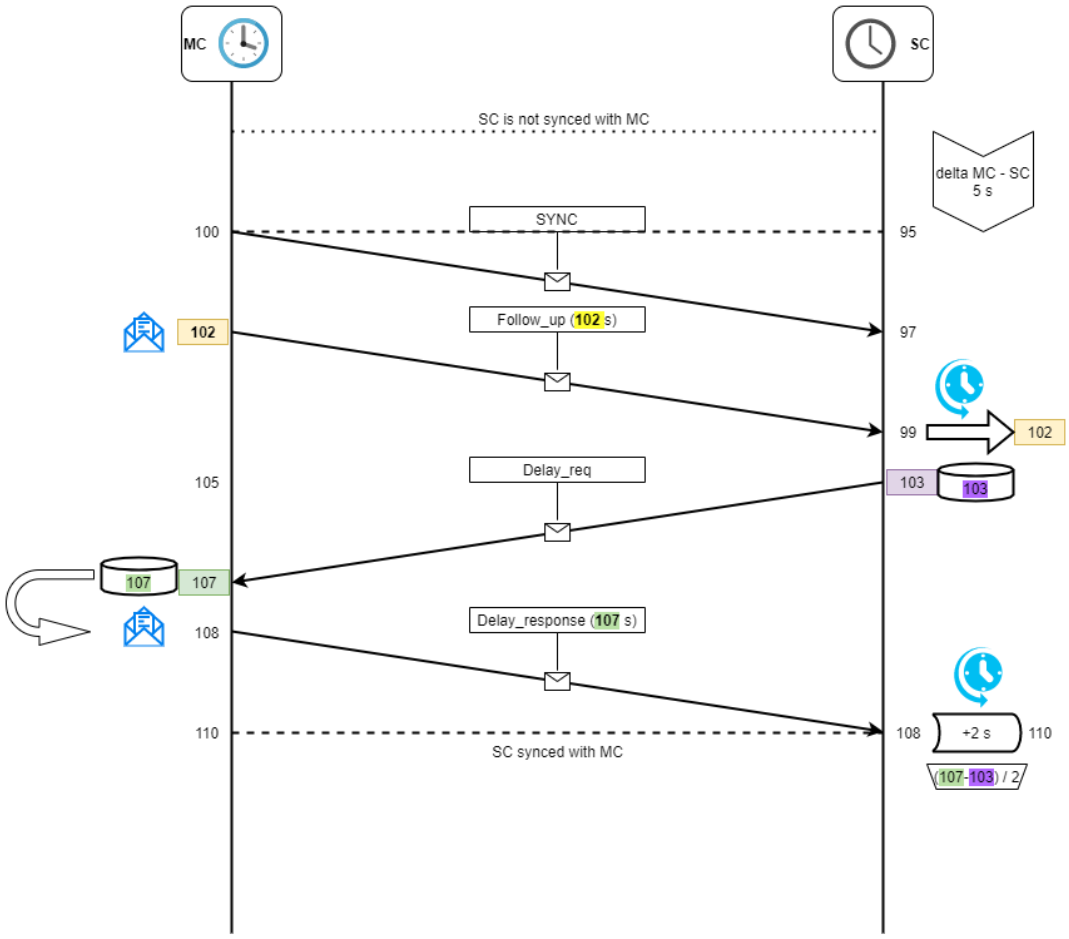
Obrázek 5 Synchronizace lokálního serveru PTP pomocí GPS

Tato synchronizace probíhá z důvodu, že může dojít k výpadku primárního zdroje času (GPS) a je potřeba mít záložní zdroj času (záložní GMC).

3.3 Algoritmus PTP

Synchronizace času mezi M a S zařízeními probíhá s určenou frekvencí. Synchronizace je vždy inicializována M zařízením, které zahájí synchronizaci posláním zprávy *sync*.

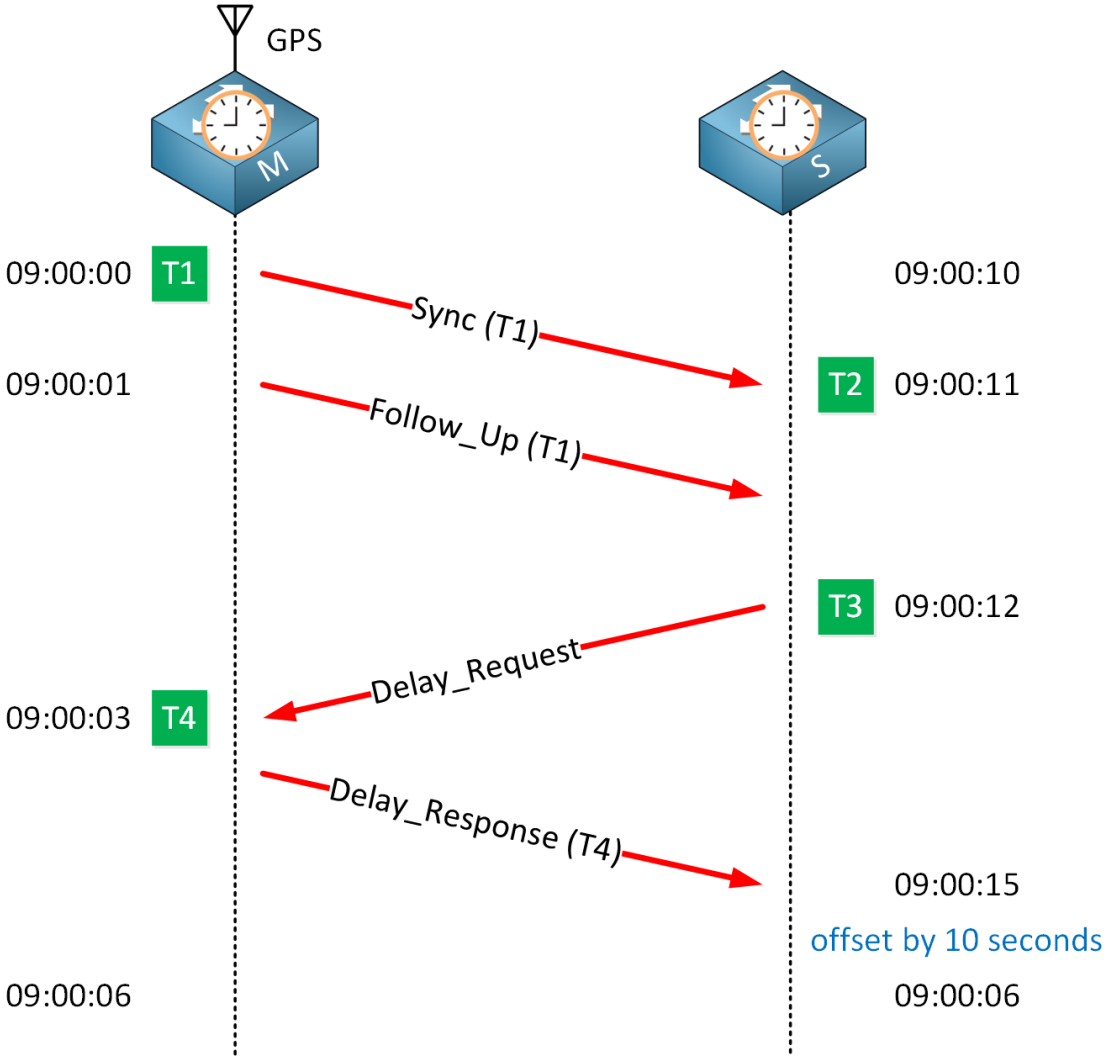
Celý průběh synchronizaci je graficky znázorněn na následujícím obrázku.



Obrázek 6 Příklad PTP algoritmu

Obrázek 6 zobrazuje jednoduchý příklad, jak probíhá synchronizace zařízení S s počátečním časem 95. Dále je v příkladu použito zařízení M s počátečním časem 100. Jako první je poslána zpráva *sync*. Při poslání této zprávy je uložen čas jeho odeslání v zařízení M a v zařízení S je uložen čas jeho přijetí. Poté je poslána zpráva *follow_up* zpráva obsahující časovou značku odeslání zprávy *sync*.

V dalším kroku je poslána zpráva *Delay_req* od zařízení S a je uložen čas jejího odeslání. Následuje zpráva od M pro S *Delay_req*. Tato zpráva obsahuje časovou značku přijetí zprávy *Delay_req*.[1][3][5]



Obrázek 7 PTP algoritmus z pohledu časů použitých ve vzorci [7]

Obrázek 7 popisuje synchronizaci mezi zařízením S a zařízením M z pohledu uložených časových značek (tzv. timestamps). Synchronizace času může být popsána také pomocí následujícího vzorce, který využívá výše zmíněně časy T1, T2, T3 a T4.

$$delay = ((t2 - t1) + (t4 - t3)) / 2 \quad (3.3.1)$$

$$offset = ((t2 - t1) - (t4 - t3)) / 2 \quad (3.3.2)$$

Delay je časový údaj, který popisuje, jak dlouho trvá komunikace mezi MC a SC.
Offset je časový údaj zobrazující, o kolik je čas SC posunut vůči času MC.

3.4 BMCA algoritmus

Ne každá síť obsahuje pouze jedno Grand Master Clock zařízení (dále pouze GMC) a jedno ED/SD. V praxi jsou sítě složitější a obsahují více GMC. Princip synchronizace je založen na tom, že je aktivní pouze jeden GMC. Z technologických nebo jiných důvodů je však možné mít v síti více něž jen jedno GMC. Pokud se v síti nachází více než jedno

GMC, standard tento stav ošetřuje aplikováním BMCA. Tento algoritmus se stará o volbu nejvhodnějšího GMC a o deaktivování ostatních GMC. [1][2]

3.4.1 Parametry GMC pro BMCA

Samotný BMCA používá následující parametry pro volbu GMC:

- priority1
- clockClass
- clockAccuracy
- offsetScaledLogVarience
- priority2
- clockIdentity

3.4.2 priority1

První parametr pro rozhodování BMCA algoritmu. Jedná se parametr, který je volen uživatelem v hodnotě 0 až 255. Čím nižší hodnota, tím vyšší priorita. Jedná se o parametr, který rozhoduje s největší prioritou. [1]

3.4.3 clockClass

ClockClass určuje třídu zdroje času. Tento parametr je určen na základě vlastností a stability zdroje času. Třídy jsou rozděleny dle standardu IEEE 1588 do několika kategorii a nabývají hodnot od 0 do 255, nejsou však obsazeny všechny pozice. [1]

3.4.4 clockAccuracy

Tento parametr určuje přesnost výstupu zdroje času. Dle dosažené přesnosti je tomuto zdroji přiřazena číselná hodnota vyhodnocující přesnost. Hodnota parametru je dána tabulkovou hodnotou ze standardu IEEE1588.

3.4.5 offsetScaleLogVariency

Parametr popisující stabilitu zdroje času.

3.4.6 priority2

Uživatelsky nastavitelný parametr. Funkce parametru je stejná jako u parametr priority1. Stejně tak nabývá hodnot 0 až 255. Čím nižší hodnota, tím vyšší priorita zdroje. [1]

3.4.7 clockIdentity

Nejnižší priorita parametru BMCA algoritmu. Rozhoduje o volbě GMC, pokud jsou veškeré předchozí parametry shodné. [1]

3.4.8 Postup BMCA

Jak již bylo zmíněno, BMCA algoritmus postupně porovnává výše zmíněné parametry PTP uzlu, který se uchází o místo GMC. Nižší stupeň parametru je využit pouze pokud

dojde ke shodě na vyšším stupni. Po vybrání nejlepšího zdroje času dojde k tomu, že vybraný GMC přepne svůj PTP port do stavu Master. Druhý uchazeč přepne svůj PTP port do stavu Pasive.[3]

Je také možné, že se vyskytne vícenásobný výskyt jednoho a toho samého zdroje času. K tomu může dojít díky smyčce v síti. V tu chvíli dojde na porovnání hloubky sítě, respektive se porovnává počet skoků skrze Boundary Clock či Transparent Clock device a zdroj s nižší počtem skoků je vybrán. [1][2][3]

3.5 Zprávy

Kapitola se zabývá popisem jednotlivých zpráv, které jsou při komunikaci používány.

Zprávy jsou rozděleny na dva základní druhy. Prvním je managment. Tyto zprávy slouží pro nastavení komunikace či zařízení. Dále mohou být využity při BMCA algoritmu. Druhým typem jsou již synchronizační zprávy, které jsou využity přímo pro synchronizaci.

V této kapitole jsou popsány nejčastěji využívané zprávy a jejich význam.

3.5.1 Announce

Announce zprávu posílá kandidát o pozici aktivního GMC. Každý GMC podá v announce zprávě informace o sobě – viz kapitola o BMCA. Zpráva slouží pro zahájení BMCA a výběr GMC. Nemá význam pro samotnou synchronizaci. [1]

3.5.2 Sync

Jedná se o první zprávu, která zahajuje synchronizaci. Zprávu posílá zařízení M je adresována zařízení S. Obsahem zprávy je typ zprávy, adresa a také časová značka, která je použita při synchronizaci. [1]

3.5.3 Follow up

Zpráva je poslána zařízením M a adresována pro zařízení S. Opět je přítomna časová značka. Tato zpráva v kombinaci s předchozí zprávou slouží pro změnu vnitřního času S. [1]

3.5.4 Delay request

Tato zpráva zahajuje část algoritmu pro synchronizaci, která se stará o minimalizaci chyby synchronizace způsobené časovým zpožděním mezi zařízením M a S. Toto zpoždění je způsobeno zpožděním při odesílání, přenosem a příjmem zprávy. [1][3]

3.5.5 Delay response

Touto zprávou je dokončena část algoritmu, která minimalizuje chybu synchronizace způsobenou zpožděním komunikace. [1][5][3]

4. VLASTNOSTI VÝVOJOVÉHO KITU

Tato kapitola slouží k seznámení s používaným HW. Pro vypracování byl zvolen vývojový kit STM32H747I-DISCO. Kapitola je zaměřena na popis CPU a relevantních periferií pro téma práce.

Informace popsané v této kapitole vycházejí z informací dostupných ve zdrojích od výrobce vývojového kitu. Primárním zdrojem pro tuto kapitolu byl takzvaný Reference manual rm0399 [9] a User manual UM2411[8].

4.1 CPU

Použitý kit je vybaven dvoujádrovým procesorem STM32H747XIH6U. Je zde přítomno výkonnější jádro Arm Cortex-M7 a druhé, méně výkonné jádro Arm Cortex-M4. Z důvodu přehlednosti bude na jádra dále referováno jako pouze jádro M4 a jádro M7. Výkonnější jádro M7 nebude v rámci této práce využito, proto nebude popsáno. [8]

Tato práce využívá jádra M4. Jádro M7 využito není z důvodu návaznosti na další projekty.

Úkony, které bude jádro M4 obstarávat jsou následující:

- Komunikace pomocí UART
- Časová synchronizace hodin pomocí IEEE 1588 (Precision Time Protocol)
- Čtení ADC převodníku
- Přiřazování časové značky k naměřeným datům

Naměřená data je potřeba také společně s časovou značkou odeslat do připojeného počítače a k tomu zde slouží UART rozhraní.

4.2 Ethernet

Kit obsahuje také standardní konektor RJ-45. Ethernetové rozhraní na kitu deklaruje podporu standardu IEEE802.3-2002 a dostupnou rychlost 10/100 Mb/s. Fyzická vrstva neboli PHY je k procesoru připojena skrze rozhraní RMII. [8][10]

Použitý procesor STM32H747XIH6U má dle oficiální dokumentace na straně integrovaného ethernetového řadiče HW podporu protokolu PTP, IEEE 1588 2008. [10]

Pro použití ethernetu na kitu bylo potřeba změnit pozici pájecích mostů. Ve výchozím nastavení je před použitím ethernetu upřednostňováno zapojení audio rozhraní, je tak potřeba změnit pozice zmíněných mostů.

Samotné rozhraní ethernetu je zde připojeno k fyzickému konektoru RJ45. Výhodou této platformy je možnost využít tzv. middleware, pojmenovaný jako lwIP, který mimo jiné umožňuje také nastavení profilu určeného přímo pro použití s PTP protokolem.

Postup nastavení a fyzických změn je popsán v samostatné kapitole 5.3. Součástí zmíněné kapitoly je také popis potřebných nastavení pro zprovoznění ethernetového rozhraní.

4.3 UART/USART

V rámci projektu bude potřeba využít UART rozhraní pro sériovou komunikaci. Kit podporuje více USART a UART komunikací. Z toho důvodu je možné jednoduše implementovat komunikaci s počítačem skrze USB.

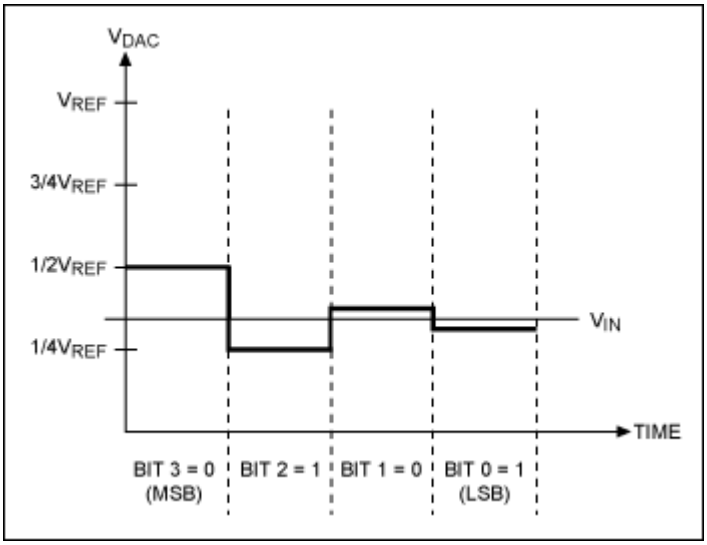
Platforma obsahuje 4 UART a 4 USART rozhraní. Celkem je zde tedy přítomno 8 UART/USART transceiverů. UART 1 je napojen na takzvaný VCOM. To označuje takzvaný virtuální komunikační port, který je propojený s USB rozhraním ovládaným skrze ST-LINK. Zjednodušeně řečeno se jedná o rozhraní, které je propojené pomocí USB s počítačem. To znamená, že skrze toto rozhraní je možné jednoduše pomocí vhodného terminálu komunikovat s připojeným PC.

4.4 ADC

Celý projekt se zaměřuje na co nejpřesnější časové zařazení změřeného vzorku. Z toho důvodu bude na zařízení docházet k převodu analogové hodnoty na hodnotu digitální. Bude zde využit ADC převodník přítomný na zvoleném vývojovém kitu.

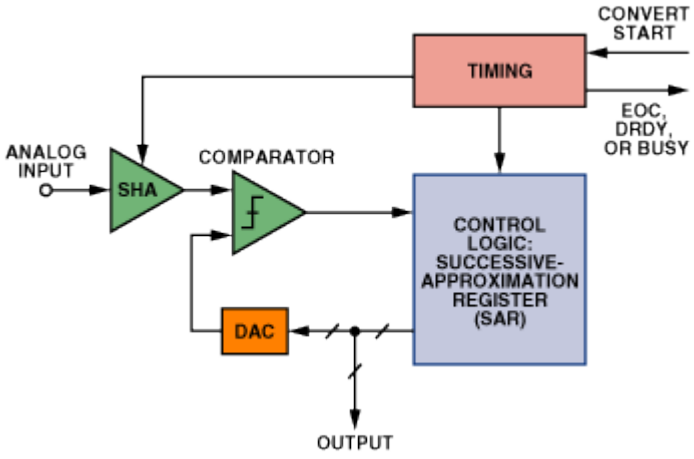
Kit obsahuje 3x 16bitové ADC (analogově digitální převodníky – Analog/Digital Converter). Všechny tři převodníky jsou typu SAR neboli postupné aproximace. Jedná se o typ převodu, kdy se výstupní hodnota postupně přibližuje převáděné hodnotě. Principiálně se tedy jedná o proces, který trvá tolik hodinových cyklů, kolik je bitů převodníku. Princip je znázorněn na obrázcích. [8]

Typ převodníku SAR se také vyznačuje tím, že převedená hodnota je uložena do přítomného registru převodníku. Tento registr má v případě převodníku přítomného na tomto kitu, je 32 bitů. Celý AD (analogově digitální) převodník je připojen k mikrokontroleru pomocí AHB (Advanced Highspeed Bus) – AHB sběrnice pro rychlou komunikaci mezi periferiemi a mikroprocesorem. [8][11][12][13]



Obrázek 8 Příklad fungování AD převodníku typu postupné aproximace.[12]

Obrázek 7 zobrazuje příklad fungování SAR AD převodníku na příkladu, kde je použit 4bitový AD převodník. Dále je na obrázku 8 zobrazeno základní schéma AD převodníku využívající postupnou aproximaci.



Obrázek 9 Základní schéma AD převodníku typu SAR (postupné aproximace) [11]

4.5 RTC

RTC je zkratka, která znamená RealTimeClock, tedy hodiny reálného času. Jinými slovy se jedná o periferii, která umožňuje mít přehled o absolutním čase. Tento rozdíl je podstatný především z důvodů, kdy je v oblasti elektrotechniky referováno k pojmu zdroj hodin, anglicismem označováno jako clock, zdroj hodinového pulsu. V případě RTC je hodinový pulz také využíván, nicméně význam této periferie je založen na čítači, který hodinový pulz převádí na lidem blízkou interpretaci. RTC periferie obsahuje řadu registrů, které slouží pro ukládání aktuálního času.

Základem jsou dva registry. První je RTC_TIME sloužící pro uložení času ve formátu HH:MM:SS. Druhým je registr RTC_DATE sloužící pro ukládání informace o aktuálním dni ve formátu YYYY:MM:DD. Toto jsou dva základní registry, které umožňují jednoduchým způsobem uchovávat a zjišťovat aktuální absolutní čas – nikoliv pouze relativní.

V případě, že je přítomna baterie sloužící pro napájení RTC obvodů, je čas zachován i po odpojení napájení.

RTC je možné také použít pro časové značky. Toto použití je pro tuto diplomovou práci podstatné, a tak mu bude také věnována samostatná kapitola.

Pro aplikace, kde je potřeba dosahovat vyššího rozlišení časové značky, než vteřiny je vhodné používat RTC_SSR registr, který slouží pro ukládání časové značky v jednotkách nižších, než jsou vteřiny. Proto je nazýván SubSecondRegister. Pro čtení tohoto registru je potřeba použít separátní funkci a správně dále pracovat s danou proměnnou. Tato starost však odpadá při správném použití principu DMA, který umožňuje použití časových značek bez plného zapojení procesoru. Přesněji se jedná o to, že dojde k přiřazení časové značky k danému naměřenému prvku a vše je uloženo

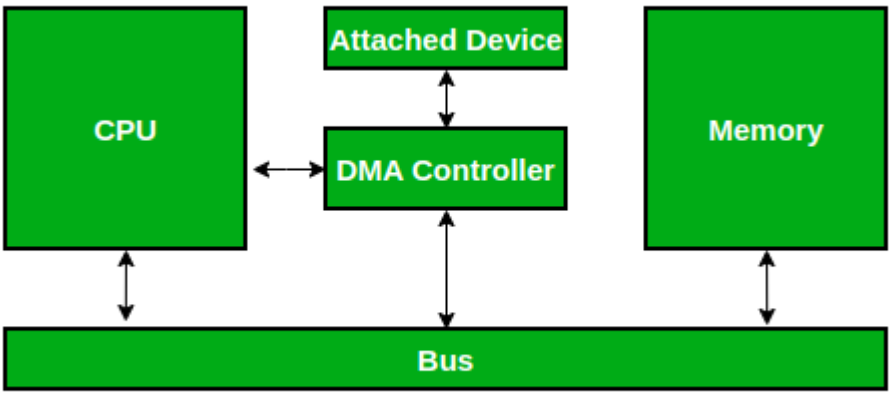
pomocí DMA přímo do registru bez přičinění procesoru, který zpracovává až další případná data.

4.6 DMA (DirectMemoryAccess)

DMA označuje technologii využívanou v počítačích umožňující přístup připojeného zařízení či periferie přímo ke sběrnici či paměti bez účasti procesoru. Tento přístup přináší řadu výhod.

První je snížení zátěže procesoru jednoduchými procesy, které nevyžadují jeho účast. To znamená, že zvolená periferie provádí určenou činnost – respektive přenos dat – bez přičinění procesoru, který se mezitím může věnovat jiné úloze.

Druhou výhodou je zefektivnění přiřazené činnosti, která přestává být vázána na takt a zatížení procesoru, které může být v rozsáhlejších projektech nezanedbatelné. Znamená to jinými slovy to, že i když je procesor plně vytížen, tak může dojít k přenosu podstatných dat do paměti plnému zatížení procesoru navzdory.



Obrázek 10 Schematický obrázek znázorňující použití a princip DMA [15]

4.7 freeRTOS

V počáteční fázi projektu bylo rozhodnuto, že bude při vypracování práce využit operační systém reálného času. Zvolená platforma STM32 podporuje operační systém freeRTOS.

4.7.1 Popis operačního systému

Tento operační systém je založen na jednotlivých procesech, které jsou na sobě zpravidla nezávislé. Často je na procesy referováno jako na procesy. Toto označení je pro osoby znalé více vypovídající než české označení proces.

4.7.2 Popis procesů při používání freeRTOS

FreeRTOS umožňuje rozdělení vytvořených procesů podle priority. Prioritizace rozhoduje o prioritě provádění daného procesu. To znamená, že u procesů, které jsou časově kritické, je možné volbou vysoké priority zajistit, aby byly vykonávány přednostně. Jedná se například o zpracování prioritní komunikace, provádění měření, ovládání připojeného aktuátoru či jiné prioritní aktivity. A naopak procesy, které nejsou časově kritické, jako je například vizuální signalizace nebo UART komunikace sloužící posílání výsledků či jiných dat.

U procesu je možné nastavit kromě jména a priority také vstupní parametry, vstupní funkci procesu a velikost zásobníku pro daný proces.

Použitá platforma umožňuje volbu priority v rozsahu 48 úrovní. Priority jsou primárně děleny do hlavních úrovní a to následujících:

- RealTime
- High
- AboveNormal
- Normal
- BellowNormal
- Low

Každá z výše uvedených úrovní má ještě nastavení úrovně 0–7 a umožňuje tím jemnější nastavení priority daného procesu.

4.8 Použitý vývojový a monitorační SW

Výrobce HW dodává celou sadu vývojových nástrojů. Mezi ně patří STM32CubeIDE, STM32 Programer, STM32CubeMX a další. V této práci byly využity následující: CubeIDE, CubeMX a Programmer.

Celá tato kapitola vychází ze zkušenosti autora z používání zmíněného softwaru a dostupných podkladů od výrobce HW i SW, což je firma STMicroelectronics.

Součástí je také krátký popis dalších dvou programů, které byly při práci využity. Jedná se analytický nástroj Wireshark sloužící pro monitorování síťového provozu a terminálový emulátor pro sériovou komunikaci PuTTY.

4.8.1 CUBE IDE

CubeIDE je vývojové prostředí od výrobce mikrokontroleru STM. Prostředí umožňuje psaní kódu, překlad kódu, debuggování a nahrání kódu do mikrokontroleru. Součástí je také zobrazení využití přiřazených paměťových sektorů. IDE umožňuje jednoduché spojení s vývojovou deskou. V rámci uživatelského rozhraní je možné plynule přecházet do grafického rozhraní pro konfiguraci mikrokontroleru pomocí CubeMX.

4.8.2 CUBE MX

Konfigurace mikrokontroleru probíhá pomocí grafického rozhraní. V rámci CubeMX je možné nastavit zdroj hodinového signálu a nastavení děliček pro jednotlivé sběrnice a periferie. V rámci tohoto prostředí je možné také vypínat či zapínat jednotlivé periferie a provádět jejich následnou konfiguraci.

Součástí CubeMX je také možnost konfigurace takzvaného middleware. Na platformě STM32 je dostupná celá řada dostupných middleware, nicméně pro tento projekt je podstatný především middleware lwIP a freeRTOS. V rámci lwIP je možné nastavit režim ethernetu, IP adresu a další záležitosti ohledně nastavení sítě. FreeRTOS je zde možné nakonfigurovat v podstatě celý. Je možné nakonfigurovat jednotlivé procesy, jejich parametry, velikost stacku a především prioritu daného procesu.

4.8.3 Výhody využití dostupného software

Nespornou výhodou využívání kombinace CubeIDE a CubeMX je provázanost a plynulost přechodu mezi oběma prostředími. Při vývoji projektu je tedy možné plynule přecházet mezi psaním samotného kódu a prováděním nezbytné konfigurace samotného mikrokontroleru či dostupných periferií.

Tato provázanost se naplno projevuje i díky generátoru kódu, který spočívá v tom, že vymezené sektory kódu jsou generovány na základě změn provedených v grafickém konfigurátoru. Takto generovaný projekt vede k tomu, že je možné i v průběhu již rozsáhlého projektu měnit konfiguraci bez manuálního přepisování konfiguračního kódu.

Zmíněný generátor ale není bezproblémový nebo úplně spolehlivý. Nabízí pomoc při tvorbě a úpravě projektu, nicméně je potřeba vygenerovaný kód zkontrolovat a případně upravit tak, aby vše bylo správně.

Posledním zmíněným programem je ST Programmer, pomocí kterého je možné přehrávat nový firmware pro mikrokontroler, přítomný ST-Link, zkontrolovat přítomné registry či nahrát nový kód do připojeného mikrokontroleru.

4.8.4 Ostatní software Wireshark

Wireshark je výkonný nástroj pro monitorování síťového provozu, který nabízí širokou škálu funkcí pro analýzu a ladění síťových komunikací. Jeho uživatelsky přívětivé rozhraní umožňuje monitorovat a filtrovat různé připojení a sledovat síťový provoz na úrovni paketů. Software umožňuje analýzu síťového provozu za použití filtrů adres či například použitých komunikačních protokolů.

Nástroj umožňuje sledování zprávy a její interpretaci na základě použitého protokolu. Pomocí tohoto programu je tedy možné přehledně analyzovat síťovou aktivitu zařízení připojeného na monitorovanou síť.

4.8.5 Ostatní software PuTTY

Jednoduchý nástroj pro komunikaci skrze sériovou linku. Umožňuje čtení i zapisování na zvoleném portu sériové linky. Pro každé připojení je potřeba nakonfigurovat konkrétní připojení – port, rychlost, stop bit atd. Kýžené nastavení je možné uložit pro rychlé načtení. Použití programu je blíže popsáno v kapitole zabývající se konfigurací a obsluhou rozhraní UART.

5. KNIHOVNY ZAMĚŘENÉ NA IMPLEMENTACI PTP (IEEE 1588) A ZPŮSOB JEJICH VYUŽITÍ V TÉTO PRÁCI

Kapitola slouží k seznámení čtenáře s dostupnými knihovnami pro implementaci PTP protokolu. Primárním zaměřením bude knihovna PTPd. Část kapitoly se také věnuje podpoře a funkcím relevantním pro PTP přímo od výrobce HW, STMicroeletronics. Součástí popisu každé knihovny je také způsob její implementace do projektu.

Kapitola čerpá informace z komunitních stránek, kde byly zmíněné knihovny zveřejněny. Dalším zdrojem informací byl také GITHub odkud byly knihovny a jejich deriváty čerpány. [19][20][21]

5.1 Knihovna PTPd

PTPd je projekt, který vznikl v prvních letech tohoto tisíciletí. Původním cílem bylo umožnit provozování PTP časové synchronizace na UNIX based zařízení. Postupně se k této iniciativě přidalo více tvůrčích skupin a časem vytvořili více verzí této knihovny. Knihovna implementuje PTPv2 (2008).

Pro tuto práci byl výchozím materiálem projekt, který implementoval PTPd na platformě příbuzné té, jaká byla zvolena pro tuto práci. Jedná se o platformu STM32F429 Nucleo. Tato platforma (STM32F4) je starší [23] než použitá (STM32H7) a proto se v určitých ohledech liší. Rozdílné jsou také HAL knihovny, které jsou použity ve výchozích materiálech. Tyto často pouze drobné rozdíly však vedou k tomu, že není možné již stávající knihovnu jednoduše použít na cílové platformě.

Výchozí projekt pochází přímo od tvůrců PTPd knihovny na stránce GitHUB[22]. Tento adresář obsahuje ukázkový projekt pro implementaci jak pro synchronizující (v projektu referováno jako Master) zařízení, tak i pro synchronizované (v projektu referováno jako Slave) zařízení. Dalším z rozdílů mezi výchozím a cílovým projektem je fakt, že cílový projekt bude využívat operační systém reálného času freeRTOS, který není použit při výchozím projektu.

Jelikož tento výchozí projekt využíval jiné funkce a odkazoval například na jinak pojmenované registry a funkce než cílová platforma, bylo potřeba výchozí projekt zkombinovat s dalším projektem, který upravil zdrojové a hlavičkové soubory tohoto projektu na platformu STM32H7. Použitý adresář lze opět najít na webové stránce GitHUB pod názvem PTPD FOR STM32H7 AND ATSAME70 [24].

Kombinací zmíněných projektů společně s úpravou na cílovou platformu bylo možné implementovat knihovnu PTPd na cílovou platformu STM32H747. Následující podkapitoly popisují jednotlivé zdrojové či hlavičkové soubory, jejich význam a provedené úpravy, aby bylo možné implementovat je na cílové platformě.

Některé názvy souborů jsou změněny z důvodu sloučení více výchozích souborů dohromady nebo rozdílné názvy ve zdrojových projektech nebo na cílové platformě.

Pří práci byl hojně využíván referenční manuál pro vývojový kit RM0399[9].

5.1.1 ptpd.h a ptpd.c

Hlavičkový a zdrojový soubor plní funkci řídícího souboru pro celý ptpd protokol. Ve výchozím projektu jsou tyto soubory pojmenovány jako ptpd_main.c a ptpd_main.h.

Souboru byl změněn název a přibyla funkce TimerUpdate, která slouží pro aktualizaci časovače. Jiné zásadní změny nebyly potřeba provádět.

5.1.2 bmc.c

Tento soubor slouží pro implementaci BMCA algoritmu. Princip algoritmu je blíže popsán v kapitole 3.4. Ve zkratce se jedná o algoritmus, který rozhoduje o volbě toho, jaké zařízení na sítí bude mít roli takzvaného Master zařízení a roli Slave zařízení. Master zařízení je označení pro synchronizované zařízení, jehož čas je považován za správný. Čas zařízení na Slave je čas, který je pomocí PTP měněn, aby se co nejméně lišil od času na Master zařízení.

V tomto souboru nebylo potřeba provádět žádné změny.

5.1.3 constants.h

Jedná se o hlavičkový soubor obsahující definice parametrů PTP uzlu. Podstatnými definicemi jsou definice vlastností PTP uzlu, jako jsou především:

- parametry pro BMCA (priority1, priority2, clockType a další)
- druh topologie (P2P či E2E)
- interval vysílání zpráv Announce a Sync
- volba boundary či ordinary clock
- a další

Pro správné fungování projektu je nutné správně nastavit výše zmíněné parametry. Pro BMCA algoritmus je potřeba, aby PTP uzly na jedné síti neměly shodné parametry sloužící k vyhodnocení nejlepšího zdroje hodin. Jedná se o parametry (priorita1/2, přesnost hodin, typ hodin). Dále je potřeba nastavit stejný delay mechanismus (E2E či P2P). Toto jsou ty nejdůležitější parametry, které je možné nastavit. Tento soubor neobsahuje veliké množství návazností na jiné soubory, proto nebylo potřeba jej příliš modifikovat.

Vybrané parametry dostupné v souboru constants.h jsou přiloženy v následujícím výstřižku ze zmíněného souboru.

```
/* Implementation specific constants */
#define DEFAULT_DELAY_MECHANISM         E2E
#define DEFAULT_ANNOUNCE_INTERVAL       1 /* 0 in 802.1AS */
#define DEFAULT_DELAYREQ_INTERVAL       3 /* from DEFAULT_SYNC_INTERVAL
to DEFAULT_SYNC_INTERVAL + 5 */
#define  DEFAULT_SYNC_INTERVAL              1 /* -7 in 802.1AS */ //0
```

```
origianly
#define DEFAULT_CLOCK_CLASS             100    // default 248
#define DEFAULT_CLOCK_CLASS_SLAVE_ONLY  248    //slave 255, master 100
#define DEFAULT_CLOCK_ACCURACY          0xFE
#define DEFAULT_PRIORITY1               100 //slave 248, master 100
#define DEFAULT_PRIORITY2               248
#define  DEFAULT_CLOCK_VARIANCE            5000 /* To be determined in
802.1AS */
```

```
/* features, only change to refelect changes in implementation */
#define NUMBER_PORTS      1
#define VERSION_PTP       2
#define BOUNDARY_CLOCK    FALSE
#define SLAVE_ONLY        FALSE
#define NO_ADJUST         FALSE
```

Výše vložené útržky kódu nejsou kompletním seznamem definovaných parametrů. Z důvodu přehlednosti práce jsou zde pro ukázku uvedeny pouze ty stěžejní či nejzajímavější a nejznámější definované parametry.

Soubor dále definuje protokolem využívané enum struktury, které uchovávají definice stavů jednotlivých struktur. Jedná se například o definování dostupných možností domén, síťových protokolů, zdroje času, typu hodin, typu delay mechanismu či také například definice možných PTP stavů, které jsou využity ve stavovém automatu fungování protokolu. Dále jsou zde definovány dostupné druhy PTP zpráv, které protokol umožňuje posílat.

Jsou zde také definovány délky konkrétních typů paketů podle typu PTP zprávy. Toto je opět zobrazeno na krátkém útržku kódu níže.

```
/** \name Packet length
 Minimal length values for each message.
 If TLV used length could be higher.*/
/**\{*/
#define HEADER_LENGTH                 34
#define ANNOUNCE_LENGTH               64
#define SYNC_LENGTH                   44
#define FOLLOW_UP_LENGTH              44
#define PDELAY_REQ_LENGTH             54
#define DELAY_REQ_LENGTH              44
#define DELAY_RESP_LENGTH             54
#define PDELAY_RESP_LENGTH            54
#define PDELAY_RESP_FOLLOW_UP_LENGTH  54
#define MANAGEMENT_LENGTH             48
/** \}*/
```

5.1.4 datatypes.h

Jak již název napovídá, tento hlavičkový soubor obsahuje definice používaných datových struktur.

Jsou zde definovány struktury pro časové značky, délku zpoždění či struktury jednotlivých typů zpráv pro PTP protokol. Jsou zde však také definovány struktury pro identifikaci a nastavení jednotlivých portů nebo dalších dílčích parametrů.

Nejrozsáhlejším datovým typem je struktura sloužící pro definici struktury použité v hlavním programu. Jedná se o strukturu typu PtpClock. Tato struktura obsahuje výše definované dílčí struktury a dohromady popisuje celý PTP uzel. Z toho důvodu je také volána v hlavním souboru programu – main.c, aby došlo k inicializaci a samotnému vytvoření samotného uzlu.

5.1.5 protocol.c

Zdrojový soubor protokol.c obsahuje funkce důležité pro samotné fungování protokolu a synchronizace. Pro správný průběh jsou využívány funkce toState, doState a doInit. Dále jsou přítomny již více konkrétní funkce pro zpracování, nebo naopak odeslání konkrétního typu zprávy.

Funkce doInit slouží, jak již název napovídá, k inicializaci potřebných struktur. Funkci je potřeba volat na počátku programu.

Funkce toState slouží k přechodu mezi stavy stavového automatu. Například při spuštění programu dojde k inicializaci potřených struktur a stav PTP uzlu je inicializační. Je tedy přiřazen výchozí stav definovaný v souboru constants.h INITIALIZING. Po inicializaci dochází k BMCA algoritmu, a tudíž je potřeba na síti rozhodnout o nejlepším zdroji hodin. Podle vyhodnocení BMCA algoritmu dojde k přechodu do stavu danému podle priority zdroje hodin. Konkrétní PTP uzel přejde do stavu MASTER nebo SLAVE a následovně se podle něj chová.

5.1.6 arith.c

Obsah tohoto zdrojového souboru je velice přímočarý. Jedná se o výpočty jednotlivých časových intervalů, používaných při časové synchronizaci. Konkrétně se jedná například o přepočet času v nanosekundách na vnitřní čas. Tento výpočet je závislý na frekvenci, na které operuje použitý regulovaný časovač.

Dále je také přítomen výpočet časové značky ze systémového času.

5.1.7 constants_dep.h

Na poměry projektu krátký hlavičkový soubor, sloužící pro definování síťových a dalších parametrů. Jsou zde například definovány porty pro PTP společně s adresou pro vysílání PTP. U obou adres se jedná o multicast adresu. Liší se v tom, která adresa je použita při E2E (end to end) a která je použita pro P2P (peer to peer) komunikaci. Rozdíl mezi danými způsoby je podstatný především pro výpočet zpoždění přenosu.

```
#define PTP_EVENT_PORT    319
#define PTP_GENERAL_PORT  320
```

```
#define DEFAULT_PTP_DOMAIN_ADDRESS  "224.0.1.129"
#define PEER_PTP_DOMAIN_ADDRESS     "224.0.0.107"
```

5.1.8 datatypes_dep.h

Hlavičkový soubor obsahující definice pro struktury použité jinde v kódu. Je zde například definovaný filtr pro vyhlazení zpoždění, nebo také síťový pbuffer pojmenovaný jako BufQueue.

5.1.9 msg.c

Zdrojový soubor obsahující funkce pro zpracování či vytvoření používaných zpráv pro PTP synchronizaci. Pro každý druh zprávy jsou zde přítomny dvě funkce. Jeden druh funkce je označován jako msgPack[druhzprávy] a druhý jako msgUnpack[druhzprávy].

MsgPack[druhzprávy] slouží pro vytvoření packetu zprávy, který je následně odeslán. Součástí je vždy také potřebná úprava hlavičky zprávy podle jejího druhu.

MsgUnack[druhzprávy] slouží naopak pro rozbalení přijaté zprávy a následné přiřazení přijatých dat do příslušných struktur.

Součástí souboru je také dvojice funkcí sloužících pro zpracování a přípravu hlavičky (slangově často označované anglicismem header) pro používané typy zpráv.

Typy zpráv jsou následující:

- Announce
- Sync
- DelayReq
- DelayResp
- FollowUp
- PDelayReq
- PDelayResp
- PDelayRespFollowUp

Základní typy zpráv jsou více popsány v teoretické části v kapitole 3.5.

5.1.10 net.c

Pro fungování protokolu je potřeba spolupráce protokolu se síťovou vrstvou. Mikrokontrolery STM často využívají lwIP middleware pro konfiguraci síťového rozhraní. V rámci této práce bylo lwIP využito také.

Zdrojový soubor net.c slouží pro obstarání komunikace, předávání zpráv a konfiguraci komunikace. Jelikož se lwIP (lightweight IP) middleware neustále vyvíjí, tak také docházelo ke změnám, jako zanikání starých funkcí a vznikání nových. Bylo proto potřeba zajistit, aby došlo ke správné inicializaci a nastavení této komunikace.

V rámci tohoto souboru je vytvořena celá řada funkcí, které obstarávají odesílání či přijímání packetů na síti. Dále zde probíhá kontrola toho, zda nedošlo k nějaké chybě, nebo zda nečeká zpráva na zpracování.

Samozřejmě jsou zde také přítomny funkce pro inicializaci síťového rozhraní. Součástí této inicializace je také využití tzv IGMP protokol pro zprovoznění skupinových zpráv, takzvaných multicast zpráv. V této oblasti bylo mnoho rozdílných formátů

adres, které se v rámci vývoje na sebe navazujících driverů a knihoven měnily a bylo proto potřeba části kódu z výchozích projektů nakombinovat a následně upravit tak, aby byla zajištěna správná funkčnost. Je zde nakonfigurována například UDP multicast adresa a probíhá také přiřazení adres a vlastností do vytvořené struktury ptpClock, která uchovává informace a vlastnosti daného PTP uzlu.

5.1.11 servo.c, ptpd_dep.c a ptpd_dep.h

Dvojice souboru (zdrojový a hlavičkový) ptpd_dep.c/h a servo.c sloužící pro práci přímo na úrovni registrů. Jedná se o čtení a zápis z registrů z rodiny Media Acces Control, zkráceně MAC.

Nejčastěji používané registry jsou:

- MACSTSR
- MACTSAR
- MACTSCR
- MACSTNR

MACSTSR (System time seconds register) a MACSTNR (System time nanoseconds register) jsou registry obsahující aktuální čas pro MAC. MACSTNR obsahuje subsecond část časové značky, zatímco MACSTNR obsahuje čas ve vteřinách.

MACTSCR (Timestamp control Register) je registr sloužící pro nastavení a řízení časové značky časovače MAC. Registr obsahuje bity pro volbu požadovaných režimů. V případě tohoto projektu bylo potřeba povolit zpracování PTP packetů pomocí UDP IPv4 stejně tak jako zpracování multicast zpráv. Jsou zde také bity, jejichž nastavením se provede konfigurace systémových hodin. Tato aktualizace hodin probíhá tím způsobem, že se změnou bitu TSUPDT (Update TimeStamp) provede úprava systémového času na základě hodnoty ve dvojici registrů, uchovávající relativní rozdíl v časové značce mezi synchronizačním časem a synchronizovaným časem. Jedná se o dvojici registrů MACSTSUR a MACSTNUR. V případě využití bitu TSINIT dojde k inicializaci, respektive přepisu hodnot systémového času na hodnoty uvedené v aktualizačních registrech. Pro používání PTP protokolu je také potřeba povolit funkcionalitu časové značky a její generátor pomocí bitu TSENA (Enable Timestamp).

Jak je již z popisu této kapitole zřejmé, program zde využívá konkrétní registry dané platformy. Proto bylo potřeba v rámci práce použít správné registry a také je správně používat.

5.1.12 startup.c

Zdrojový soubor obsahující funkce pro inicializaci a také ukončení funkce časové synchronizace.

Inicializační funkce slouží pro přechod stavového automatu do stavu PTP_INITIALIZING a úvodní nastavení.

Funkce ptpdpShutdown slouží naopak pro ukončení funkcionality ptpd. Přesněji PTP deamona.

5.1.13 sys_time.c

Zdrojový soubor obsahuje celkem 6 funkcí. Funkce getTime slouží k získání aktuálního systémového času. Funkce setTime naopak slouží pro nastavení nového systémového času.

Funkce updateTime a adjFreq slouží pro aktualizace systémového času. Rozdílem mezi těmito dvěma funkcemi je způsob aktualizace času. První funkce používá takzvanou Coarse, jednorázovou metodu, která upraví čas na základě offsetu. Druhá funkce (adjFreq) používá takzvaný finetuning nebo fine update metodu, která je trochu složitější, ale slouží k přesnější úpravě času.

5.1.14 timer.c

Pro fungování časové synchronizace pomocí PTP protokolu je potřeba také spravovat časovač. Pro správu takového časovače je součástí PTPd knihovny/projektu také zdrojový soubor timer.c, který obsahuje funkce pro inicializaci (initTimer), spuštění a ukončení (timerStart a timerStop) a také pro aktualizaci časovače (TimerUpdate).

5.2 PTP_lib.h a PTP_lib.c

Pro dodatečnou konfiguraci kitu byly dále vytvořeny soubory PTP_lib, které obsahují dodatečnou konfiguraci systémových registrů, která nebyla již provedena v předchozích souborech. Nejedná se už o součást výchozích projektů pro PTP daemon a byly vytvořeny čistě v rámci této práce.

Soubory implementují potřebné kroky pro povolení a konfiguraci generátoru časových značek. Funkce zde vytvořené jsou implementací návodu od výrobce uvedeného v Referenčním manuálu RM0399[9] v kapitole 61.5.4 zabývající se časovými značkami.

5.3 lwIP

lwIP (Lightweight IP) je open-source knihovna pro TCP/IP stack navržená pro vestavné systémy a aplikace s omezenými prostředky, jako jsou typicky mikrokontrolery. Jeho hlavním cílem je poskytnout síťovou komunikaci na zařízeních s omezenou pamětí a výpočetními zdroji, jako na zde použité platformě STM32H747.

Tato knihovna nabízí komplexní implementaci síťových protokolů, včetně IPv4, IPv6, TCP, UDP a dalších, a umožňuje snadnou integraci s různými síťovými rozhraními. Při použití pro tento projekt se jedná především o použití IPv4 a samozřejmě PTP. Knihovna bohužel neobsahuje plnohodnotnou podporu PTP protokolu a je potřeba ji zkombinovat s knihovnou třetí strany, obstarávající podporu kýženého protokolu.

V kontextu tématu práce na platformě STM32H747 pro implementaci časové synchronizace pomocí PTP protokolu lwIP poskytuje základní síťovou infrastrukturu pro přenos časových zpráv a řízení síťového provozu.

6. KONFIGURACE KITU PRO ČASOVOU SYNCHRONIZACI A POUŽITÍ ETHERNETOVÉHO ROZHRANÍ

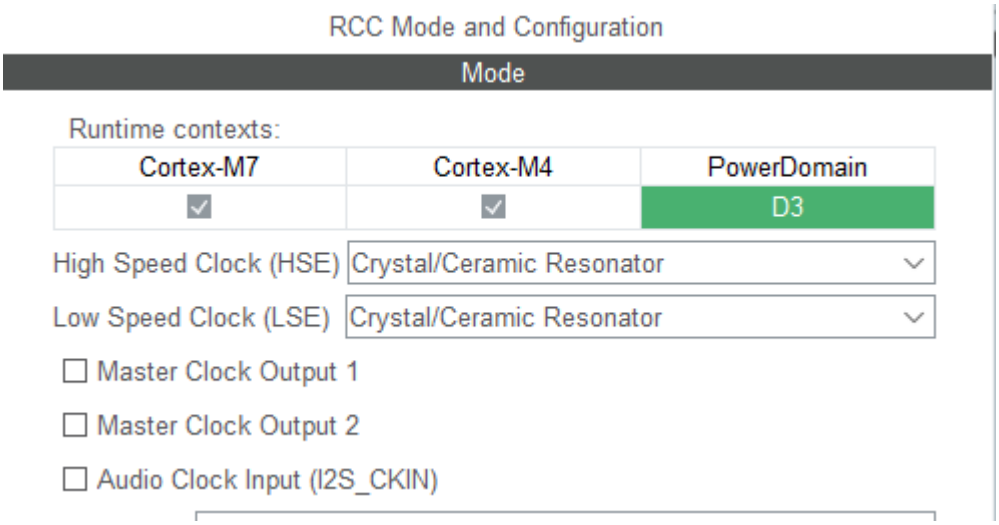
V této kapitole jsou popsány nezbytné kroky pro přípravu kitu na použití knihoven potřebných pro časovou synchronizaci pomocí IEEE 1588 protokolu či jiného obdobného protokolu, jako například NTP protokol.

První podkapitola se věnuje konfiguraci zdroje hodin pro vývojový kit. Druhá podkapitola popisuje kroky potřebné pro zprovoznění ethernet rozhraní přítomného na zvoleném vývojovém kitu. Ve třetí a závěrečné podkapitole je popsáno potřebné nastavení operačního systému freeRTOS a jeho procesů nutných pro správné fungování.

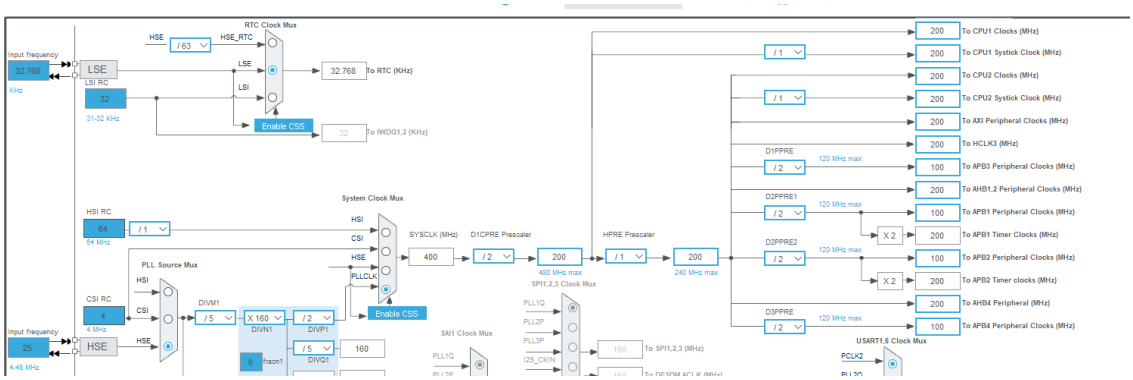
Při konfiguraci, úpravách a nastavení projektu a jeho částí byl vždy využíván referenční manuál RM0399 [9].

6.1 Konfigurace hodin

Jako první bylo potřeba nastavit takt procesoru. Pro správné fungování je potřeba určit zdroj hodin a jejich takt. V rámci tohoto procesu se nastavují také děličky taktu pro jednotlivé periferie, které mají limity taktů, na kterých mohou pracovat.



Obrázek 11 Nastavení zdroje HSC a LSC. V obou případech se jedná o keramický oscilátor. (výstřižek z prostředí MX Cube)



Obrázek 12 Nastavení prescalerů a děliček (výstřižek z STM32cubeMX)

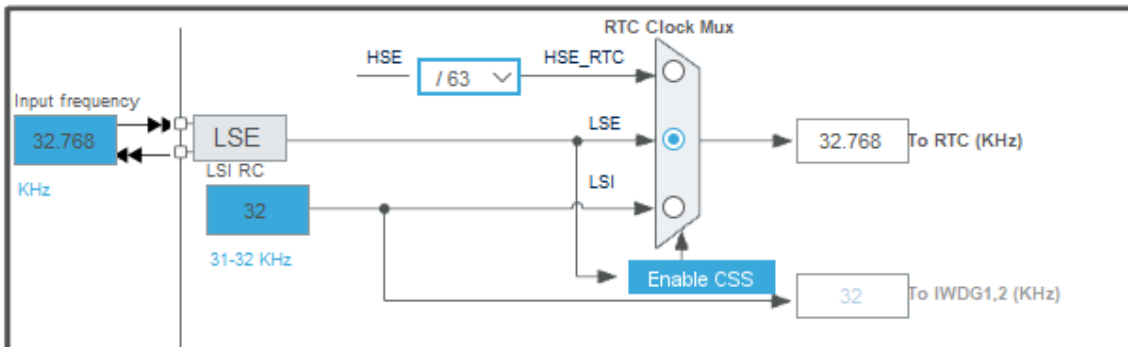
Vzhledem k tomu, že je v této práci primárně využito jádro M4, tak je potřeba dodržet jeho maximální frekvenci, která je 240 MHz. Některé periferie mají hodnotu maximální přípustné frekvence ještě nižší (konkrétně 120 MHz v případě APB sběrnic).

Použité nastavení zdroje taktu je tedy následující:

- Využit krystalový oscilátor s frekvencí 25 MHz
- Pomocí děličky bylo nastaveno pro SYSCLK 400 MHz
- Pomocí děličky byl nastaven takt pro M4 na 200 MHz
- Hodiny pro periferie využívající APB sběrnici byly nastaveny na hodnotu 100 MHz
- Konkrétní nastavení je zobrazeno na obrázku 11

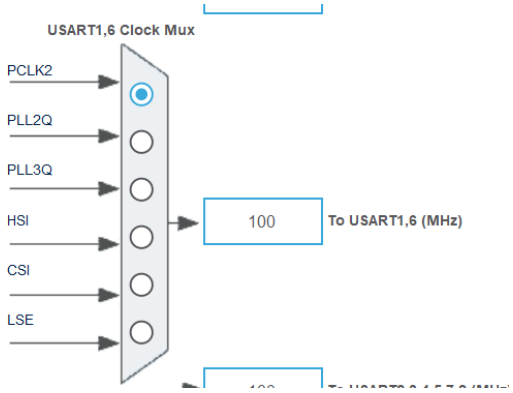
Pozorný čtenář si všimne nastavení zdroje SYSCLK na 400 MHz, nikoliv na maximální hodnotu 480 MHz. S tím také souvisí nižší hodnoty pracovní frekvence navazujících periferií a částí MCU. Toto nastavení je nutné z důvodu limitace maximálního taktu při použití napájení kitu z USB. Z praktických důvodů bylo při vypracování práce zařízení často připojeno pouze k počítači, skrze který byl kit programován. I z toho důvodu nebyla nastavena nejvyšší možná hodnota taktu.

Pro správné fungování RTC periferie, je potřeba nastavit správný zdroj hodin. Kdyby se jedno o projekt a aplikaci, kde stačí určovat čas s přesností na sekundy, tak by bylo vhodné zvolit jako zdroj hodin LSE nastavený na 32,768 kHz z důvodu stability zdroje a jednoduchého převodu na formát rok:měsíc:den respektive hodina:minuta:vteřina.[17]



Obrázek 13 Nastavení zdroje hodin pro RTC periferii (výstřižek z MX Cube)

Poslední stěžejní periferií pro kterou bylo potřeba nastavit zdroj hodin je rozhraní USART pro komunikaci s připojeným počítačem. Zdrojem hodinového pulzu pro používané rozhraní (USART1) byl zvolen PCLK2 a výsledná frekvence je 100 MHz.



Obrázek 14 Zdroj hodin pro USART 1 (výstřižek z MX Cube)

Konfigurace zdroje hodin byla provedena pomocí rozhraní MX Cube. Toto rozhraní díky vizualizaci, automatické kontrole a automatickému návrhu nastavení urychluje a zjednodušuje prvotní nastavení projektu. Nejedná se však o dokonalý nástroj, který nedělá chyby. Z toho důvodu bylo často potřeba provést ruční nastavení a kontrolu parametrů. Nicméně i tato manuální úprava konfigurace je díky grafickému rozhraní MX Cube poměrně jednoduchá.

6.2 Konfigurace ethernetu

Vývojový kit má širokou škálu možných uplatnění. Použití ethernetového rozhraní je pouze jedna z mnoha funkcionalit. Další, pro tento projekt nerelevantní funkcionalitou, je vstup digitálního mikrofonu. Pro použití ethernetového rozhraní je potřeba provést fyzické úpravy vývojového kitu. Tyto úpravy jsou popsány v kapitole 6.2.1. Dále bylo potřeba pro zprovoznění kitu provést také softwarovou konfiguraci rozhraní. Tato konfigurace je specifická pro každou platformu a správnému postupu se věnuje kapitola 6.2.2.

Kromě referenčního manuálu zde bylo využito i doporučeného postupu nastavení a konfigurace ze stránek výrobce desky.[28]

6.2.1 HW změny

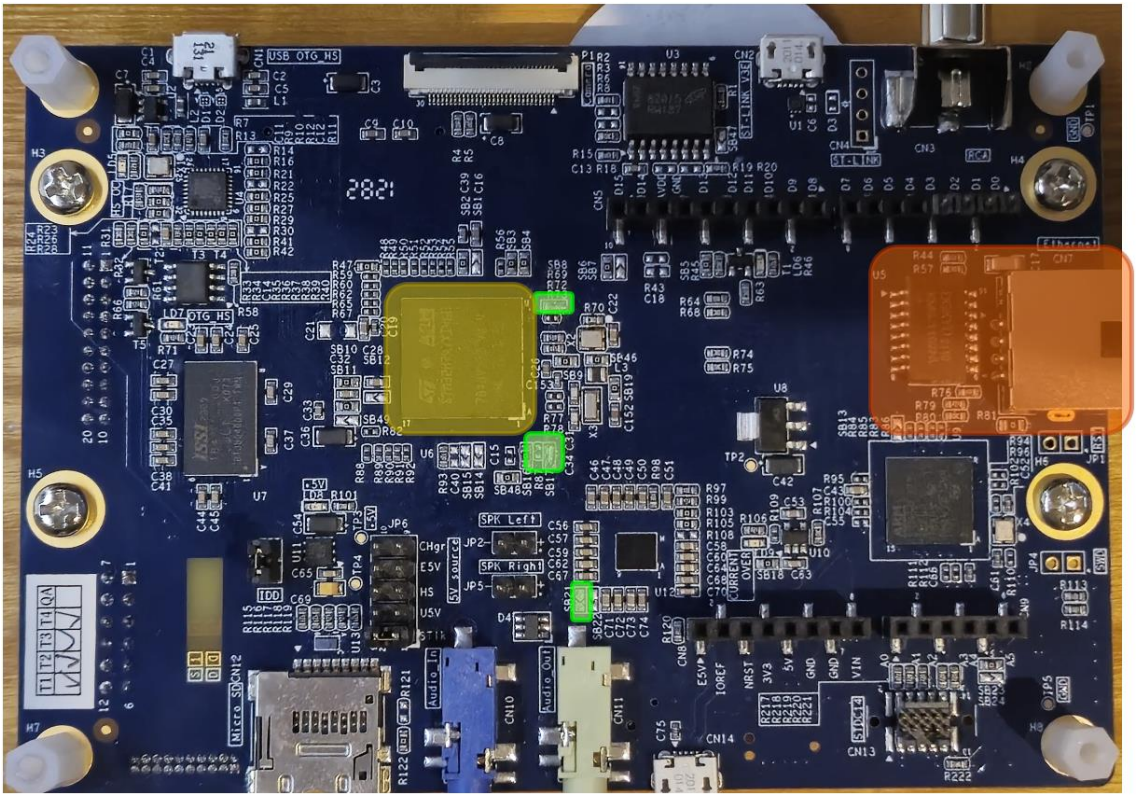
Jak je již nastíněno, vývojový kit je ve výchozím nastavení nastaven na využití mikrofonního vstupu. Z konstrukčních důvodů sdílí tato periferie prvky HW s přítomným ethernetem a jeho přítomným konektorem typu RJ45. Jedná se o konektor označený jako CN7.[8] Proto bylo potřeba upravit dle dokumentace pozici několika pájených přepínačů (anglické označení solder bridge).

Potřebné změny konfigurace jsou popsány v reference manuálu. Konkrétně se jednalo o tyto změny: [9][8]

Rozpojit: SB21 a R87

Spojit: SB17 a SB8

Umístění pájecích mostů je zobrazeno na obrázku Obrázek 15.



Obrázek 15 Popis podstatných prvků na spodní straně vývojového kitu. Žlutá – MCU STM32H747, Zelená – pájecí mosty, Oranžová – konektoru CN7 (RJ45).

6.2.2 SW konfigurace

Pro nastavení ethernetového rozhraní bylo využito opět STM32cubeMX. Aby bylo možné využít ethernet, bylo potřeba provést následující kroky:

1. Přiřazení ethernetového rozhraní k jádru M4. Na vývojové desce je přítomen dvoujádrový systém, a proto je potřeba provést volbu požadovaného jádra. Toto nastavení je možné provést v záložce connectivity.
2. Nastavení režimu ethernetového rozhraní. Rozhraní je možné provozovat ve více režimech. Jelikož je na desce přítomno RMII rozhraní a primárním cílem je připravit desku na použití časové synchronizace pomocí PTP protokolu, je potřeba zvolit režim RMII_PTP_Synchro. Toto nastavení je přítomno v nastavení ETH. Dále bylo potřeba povolit vysokou rychlost výstupu na všech použitých pinech.
3. Povolení lwIP pro jádro M4. V nastavení Middleware and Software Packs je možné obdobně jako u nastavení ethernetu přiřadit balíček lwIP k jádru M4.
4. Nastavení lwIP. Po přiřazení k požadovanému jádru byla provedena řada nastavení.
    a. Aby bylo možné spolehlivě sledovat síťový přenos, je vhodné vypnout DHCP přiřazování adresy a zvolit statickou adresu daného zařízení. To bylo provedeno i zde. Pro zařízení M byla přiřazena adresa 192.168.1.112 a pro zařízení S byla zvolena adresa 192.168.1.111.
    b. Jelikož PTPd využívá posílání zpráv formou multicastu, je potřeba povolit IGMP modul. IGMP je využíván u zpracování skupinových zpráv.
    c. Pro ověření funkčnosti síťového rozhraní je vhodné využít také ICMP modul. Tento modul umožňuje ověření připojení k síti pomocí zaslání zprávy ping.
    d. Z důvodu využití IP adres byl také zapnut LWIP_UDP modul.
    e. Dále bylo nutné zvolit paměťový prostor pro ethernet. Je potřeba zvolit velikost haldy (označováno často jako heap) a počáteční adresu. Pro volbu paměťového prostoru bylo potřeba brát v potaz povolený rozsah pro danou haldu ale také dostupnost dané paměti pro jádro M4. Halda zde byla nastavena na velikost 131048 bytů a pointer na počátek byl nastaven na adresu 0x10020000. Pointer je označen jako LWIP_RAM_HEAP_POINTER.
    f. V případě tohoto vývojového kitu je zde přítomno pouze jedno ethernetové rozhraní, nicméně stejně bylo potřeba přiřadit fyzické rozhraní do programu. Toto je provedeno v záložce Platform settings jako Device_PHY. Dojde k přiřazení driveru LAN8742.

5. Aby byl daný paměťový prostor vyhrazen pouze pro lwIP haldu a byl dostupný pro jádro M4, bylo potřeba změnit linker a tam změnit paměťové oblasti tak, jak je zobrazeno na následujícím útržku kódu z linkeru. Paměťová oblast pro RAM jádro M7 začíná na 0x300000000 a jádro M7 začíná na adrese 0x100000000.

```
/* Specify the memory areas */
MEMORY
{
FLASH (rx)      : ORIGIN = 0x08100000, LENGTH = 1024K
/* ETH_CODE: split memory to data and ethernet buffers */
RAM (xrw)       : ORIGIN = 0x10000000, LENGTH = 128K
ETH_RAM (xrw)   : ORIGIN = 0x10020000, LENGTH = 160K
}
```

```
  /* ETH_CODE: add placement of DMA descriptors and RX buffers */
  .lwip_sec (NOLOAD) :
  {
    . = ABSOLUTE(0x10040000);
    *(.RxDecripSection)

    . = ABSOLUTE(0x10040100);
    *(.TxDecripSection)

    . = ABSOLUTE(0x10040200);
    *(.Rx_PoolSection)
  } >ETH_RAM
```

Výsledné využití paměti je zobrazeno na následujícím obrázku Obrázek 16.

freeRTOS_PTP_CM4.elf - /freeRTOS_PTP_CM4/Debug - May 5, 2024, 9:29:48 PM

Memory Regions | Memory Details

Region	Start address	End address	Size	Free	Used	Usage (%)
FLASH	0x08100000	0x081fffff	1024 KB	877,77 KB	146,23 KB	14.28%
RAM	0x10000000	0x1001ffff	128 KB	57,47 KB	70,53 KB	55.10%
ETH_RAM	0x10020000	0x10047fff	160 KB	13,12 KB	146,88 KB	91.80%

Obrázek 16 Využití a rozložení paměťových oblastí pro jádro M4. (Výstřižek z CubeIDE)

Výše uvedené kroky bylo potřeba provést, aby bylo rozhraní ethernet zprovozněno.

6.2.3 Ověření funkčnosti komunikace pomocí ethernetu

Pro ověření správnosti HW úprav a následné SW konfigurace byl proveden jednoduchý test pomocí ICMP protokolu. Tento protokol obsahuje mimo jiné také oblíbený test spojení v podobě příkazu ping.

Počítač, ze kterého byl příkaz odeslán, má statickou IP adresu 192.168.1.107 a statická IP adresa vývojové desky je nastavena jako 192.168.1.111 (druhá vývojová deska má adresu 192.168.1.112). Na obrázku Obrázek 17 je výstřižek z terminálu, ze kterého bylo provedeno odeslání příkazu. Obrázek 18 zobrazuje výstřižek z programu Wireshark, který ukazuje, jak vypadala výše zmíněná komunikace na síti.

```
PS C:\Users\simon> ping 192.168.1.111

Pinging 192.168.1.111 with 32 bytes of data:
Reply from 192.168.1.111: bytes=32 time=1ms TTL=255
Reply from 192.168.1.111: bytes=32 time<1ms TTL=255
Reply from 192.168.1.111: bytes=32 time<1ms TTL=255
Reply from 192.168.1.111: bytes=32 time<1ms TTL=255
```

Obrázek 17 Výstřižek z terminálu PC pro poslání příkazu ping.

831	335.578536	192.168.1.119	192.168.1.111	ICMP	74	Echo (ping) request id=0x0001, seq=255/65280, ttl=128 (reply in 832)
832	335.579446	192.168.1.111	192.168.1.119	ICMP	74	Echo (ping) reply id=0x0001, seq=255/65280, ttl=255 (request in 831)
835	336.598389	192.168.1.119	192.168.1.111	ICMP	74	Echo (ping) request id=0x0001, seq=256/1, ttl=128 (reply in 836)
836	336.598743	192.168.1.111	192.168.1.119	ICMP	74	Echo (ping) reply id=0x0001, seq=256/1, ttl=255 (request in 835)
839	337.629558	192.168.1.119	192.168.1.111	ICMP	74	Echo (ping) request id=0x0001, seq=257/257, ttl=128 (reply in 840)
840	337.630018	192.168.1.111	192.168.1.119	ICMP	74	Echo (ping) reply id=0x0001, seq=257/257, ttl=255 (request in 839)
843	338.658612	192.168.1.119	192.168.1.111	ICMP	74	Echo (ping) request id=0x0001, seq=258/513, ttl=128 (reply in 844)
844	338.658973	192.168.1.111	192.168.1.119	ICMP	74	Echo (ping) reply id=0x0001, seq=258/513, ttl=255 (request in 843)

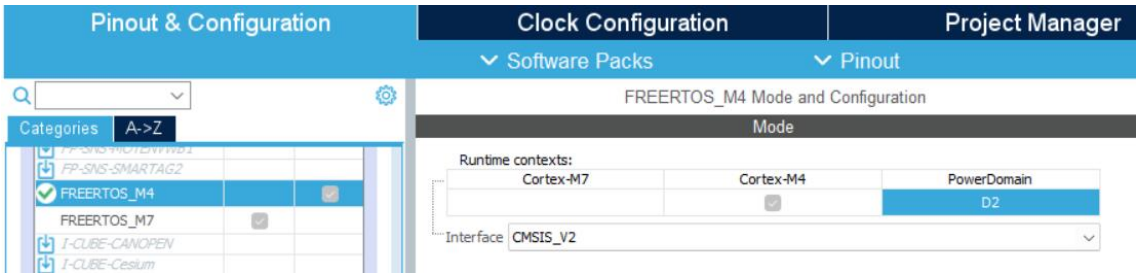
Obrázek 18 Výstřižek z programu Wireshark zobrazující průběh testu ICMP.

6.3 Konfigurace freeRTOS

V programu byla vytvořena řada procesů s různými účely a prioritami. Práce s operačním systémem reálného času může být osobám znalým v oblasti PLC povědomá z důvodu podobnosti logiky programu.

Vlastnosti a parametry kolem freeRTOS jsou popsány v kapitole 4.7. Při tvorbě procesů pro operační systém bylo potřeba přihlédnout také k charakteru daného procesu a jeho požadavkům. Při konfiguraci bylo čerpáno kromě již uvedených zdrojů také od tvůrce ControllersTips. [29][30][31]

Z obecných nastavení operačního systému zde bylo provedeno pouze přiřazení freeRTOS pro jádro M4 a dále zvolena verze interface CMSIS. Byla vybrána verze CMSIS_V2. Konfigurace operačního systému probíhá skrze CubeMX.



Obrázek 19 Přiřazení použití operačního systému pro využívané jádro M4. Výstřižek z prostředí CubeMX.

Nyní následují jednotlivé podkapitoly popisující konfiguraci a význam jednotlivých procesů (tasků) vytvořených pro navrženou aplikaci.

Proces obsluhující ethernet potřebuje mít velikost stacku minimálně 512 bytů.

6.3.1 Proces LED

Prvním a nejpřímočařejším procesem je ovládání signalizační diody. Jedná se o proces, který periodicky přepíná stav LED diody a plní tak funkci jednoduché signalizace stavu zařízení. Stejná LED dioda je využita v případě kritické chyby, která by způsobila vyvolání funkce ErrorHandler, která způsobí vizuální signalizaci chybového stavu rychlým blikání stavové LED diody.

Popisovaný proces sdílí část HW (LED dioda) a část logiky kódu společně s jednoduchým seznamovacím programem vytvořeným v rámci kapitoly 7.2.

Z popisu významu a logiky procesu je zřejmé, že se nejedná o proces náročný na velikost přiřazeného zásobníku ani na časovou dostupnost. Z toho důvodu nebylo potřeba při tvorbě tohoto procesu přiřazovat vysokou časovou prioritu. Spíše naopak je žádoucí, aby tento pro samotnou logiku aplikace nekritický proces zbytečně nevytěžoval prostředky. Z toho důvodu byla tomuto procesu přiřazena nízká priorita.

Definice procesu:

```
/* Definitions for LED_Control */
osThreadId_t LED_Control;
const osThreadAttr_t LED_Control_attributes = {
  .name = "LED_Control",
  .stack_size = 512 * 4,
  .priority = (osPriority_t) osPriorityLow5,
};
```

Vytvoření procesu LED_Control:

```
  /* creation of LED_Control */
  LED_Control     =     osThreadNew(LED_Control_start,     NULL,     &
LED_Control_attributes);
```

6.3.2 Proces UART

Rozhraní sériové komunikace USART může být v této aplikaci využito například pro nastavení parametrů prováděného měření. Dalším možným využitím UART rozhraní může být také nastavení podstatných parametrů PTP uzlu přítomného na měřícím zařízení. To může být například nastavení priority hodin nebo třída hodin pro BMCA. Zmíněné pojmy jsou vysvětleny v kapitole 3. Konkrétní implementace již záleží na konkrétním použití.

Obecně je však potřeba provést nastavení UART rozhraní a procesu, který bude zmíněné rozhraní obhospodařovat. Konfigurace UART rozhraní je popsána v kapitole 7.3.

Komunikace skrze sériovou linku není v žádném z uvažovaných scénářů časově kritická. V průběhu testování byla sériová linka využita pro odesílání aktuálního času a dat z RTC periferie zařízení. Z toho důvodu není při konfiguraci a tvorbě procesu pro jeho obsluhu kritické nastavit vysokou prioritu.

Definice procesu:

```
/* Definitions for UART */
osThreadId_t UARTHandle;
const osThreadAttr_t UART_attributes = {
  .name = "UART",
  .stack_size = 512 * 4,
  .priority = (osPriority_t) osPriorityLow1,
};
```

Vytvoření procesu:

```
  /* creation of UART */
  UARTHandle = osThreadNew(UART_f, NULL, &UART_attributes);
```

6.3.3 Proces RTC

Tento proces vyčítá aktuální časovou značku z RTC periferie. Základní princip RTC periferie je vysvětlen v kapitolách 4.5 a použití v testovacím projektu je popsáno v kapitole 7.4.

V případě vyčítání konkrétního času je již vhodné přiřadit vyšší prioritu. Především pokud je snaha vyčítat milisekundovou hodnotu z registru RTC_SSR. Z toho důvodu je vhodné přiřadit tomuto procesu vyšší prioritu. Na druhou stranu se nejedná o proces obstarávající vzorkování či časovou synchronizaci, proto není potřeba přiřazovat RealTime takovou úroveň priority jako u ADC či PTP procesu.

V případě, že se jedná o jednoduchou aplikaci a proces UART není příliš rozsáhlý, tak je možné zahrnout vyčítání z RTC periferie do již existujícího procesu/tasku. V navržené aplikaci byla zvolena tato cesta, nicméně v průběhu vývoje byl z důvodu přehlednosti a členění kódu RTC proces vytvořen samostatně. Na výsledné fungování aplikace nemá tato záležitost zásadní význam.

6.3.4 Proces ADC

Proces s druhou nejvyšší prioritou je určen pro časově nejkritičtější a nejnáchylnější úlohu. Jedná se o samotné čtení naměřené hodnoty a její převod z analogové hodnoty na vstupních pinech na digitální hodnotu uloženou v registrech periferie.

Popis periferie ADC na platformě STM32H747 je v kapitole 4.4 a 7.5. Konkrétní nastavení periferie není pro konfiguraci procesu stěžejní. Podstatné je, že je potřeba aby byl proces prováděn jako časově-kritická záležitost. Z toho důvodu je pro tento proces zvolena vysoká priorita. Konkrétně byla zvolena priorita RealTime0.

Logika procesu se zabývá získáním vzorku a následovným přiřazením časové značky k danému vzorku. Proces probíhá periodicky a je zvolena perioda vykonání 50 ms.

6.3.5 Proces PTP

Časově nejvíce kritický proces. Pomocí tohoto procesu dochází k aktualizaci vnitřního času daného PTP uzlu. Součástí procesu mimo jeho opakující se část – mimo nekonečnou while smyčku – je také inicializace PTP uzlu. Jedná se o volání inicializační funkce pro PTPd, inicializaci IGMP časovače. Je také volána funkce pro konfiguraci potřebných registrů pro aktivaci a nastavení generátoru časové značky.

Jelikož se jedná o proces s nejvyšší prioritou, byla mu nejvyšší priorita také přiřazena. Konkrétně se jedná o prioritu RealTime7.

Definice procesu PTP:

```
/* Definitions for PTP_Run */
osThreadId_t PTP_RunHandle;
const osThreadAttr_t PTP_Run_attributes = {
  .name = "PTP_Run",
  .stack_size = 512 * 4,
  .priority = (osPriority_t) osPriorityRealtime7,
};
```

Vytvoření procesu PTP:

```
  /* creation of PTP_Run */
  PTP_RunHandle = osThreadNew(PTP_Run_f, NULL, &PTP_Run_attributes);
```

6.3.6 Default task a požadavky na něj kvůli využití ethernetu

Proces označený jako defaultTask je výchozí proces vytvořený automaticky při vytvoření projektu využívající freeRTOS na platformě STM32H7.

Účel tohoto procesu v aplikaci je obsluha ethernetové periferie pomocí lwIP knihovny. Při tvorbě procesu je doporučeno alokovat zásobník o minimální velikosti 512 slov [28].

Definice procesu:

```
/* Definitions for defaultTask */
osThreadId_t defaultTaskHandle;
const osThreadAttr_t defaultTask_attributes = {
  .name = "defaultTask",
  .stack_size = 512 * 4,
  .priority = (osPriority_t) osPriorityHigh7,
```

```
};
```

Tvorba procesu:

```
  /* creation of defaultTask */
  defaultTaskHandle      =      osThreadNew(StartDefaultTask,      NULL,
&defaultTask_attributes);
```

7. SEZNÁMENÍ S PLATFORMOU, POMOCNÉ PROGRAMY A POPIS PERIFERIÍ

Pro práci je podstatné seznámit se s platformou, na které se pracuje. V této kapitole jsou popsány některé jednoduché programy, které byly vypracovány v průběhu seznamování se s platformou a jejím fungováním. Postupy a programy popsané v této kapitole jsou potřebné pro správné nastavení výsledného programu.

Kromě jednoduchých seznamovacích programů, jako je například program pro ovládání LED diod, jsou v této kapitole popsány také dílčí programy vytvořené pro ovládání podstatných periferií v tomto projektu.

7.1 Nastavení kitu

Ještě před tvorbou samotného kódu je potřeba nastavit konfiguraci samotného kitu. Podrobnější popis použité konfigurace se nachází v popisu hlavní aplikace.

Obecně platí, že pro používání kitu bylo potřeba nastavit systémové hodiny. Je potřeba zvolit jak požadované nastavení děliček frekvence, tak také zdroj hodin pro jednotlivé periferie.

Kromě nastavení hodin je také potřeba nakonfigurovat zdroj napájení.

Další nastavení je již specifické pro každou aplikaci. Je totiž potřebné správně nakonfigurovat veškeré periferie, které budou pro aplikaci využity.

Některé periferie, jako je například ethernet, vyžadují také fyzické změny přímo na kitu. Tomu se více věnuje samostatná kapitola 6.1 v dále v této práci.

Obecně se však dá říci, že pokud není řečeno jinak, tak je nastavení popsané v této kapitole aplikováno i ve finálním programu a verzi projektu i přesto, že se tato kapitola zaměřuje na pouhé seznámení se s projektem.

7.2 Ovládání LED pomocí freeRTOS

Pro seznámení se s platformou byl vytvořen jednoduchý program, který ovládá 4 LED diody přítomné na vývojovém kitu.

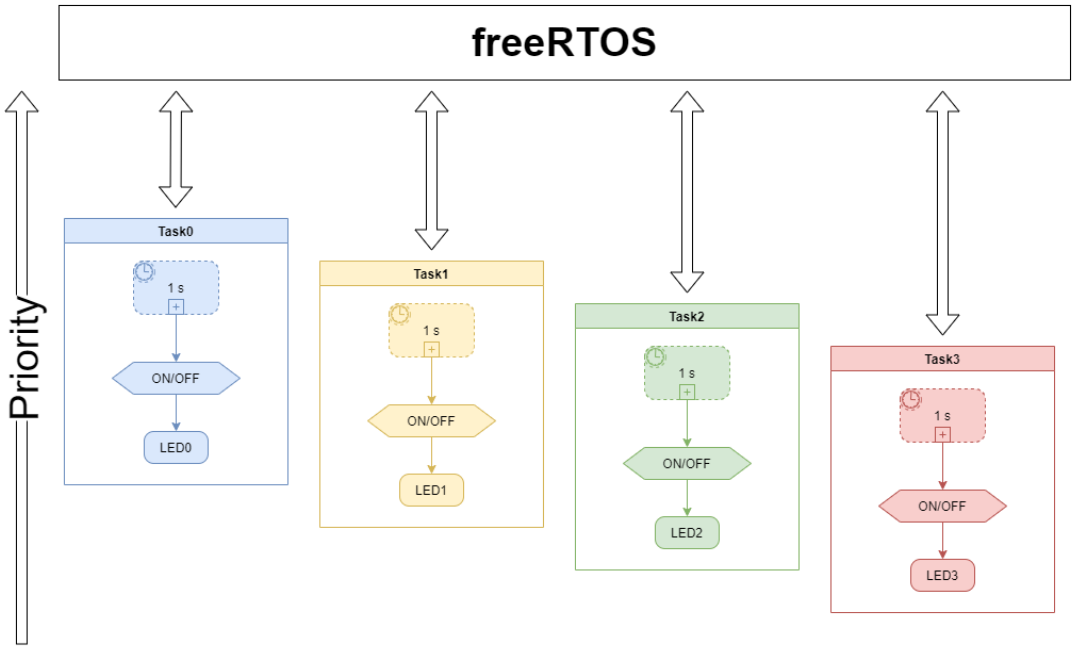
Tento cvičný program slouží pro seznámení se s vývojovým prostředím, HW konfigurací a midleware freeRTOS. Pro fungování celého projektu bude využit právě freeRTOS, jelikož umožňuje prioritizaci jednotlivých úloh (procesů).

Princip fungování tohoto programu je velice přímočarý. V rámci projektu jsou vytvořeny 4 úlohy s různými prioritami. Prioritizace úloh zjednodušuje obhospodařování úloh, které mají rozdílnou prioritu či přesněji časovou naléhavost. Čím je přiřazená priorita pro daný proces vyšší, tím výše je v pořadníku daný proces při souběžném čekání více procesů na prostředky procesoru.

Každá úloha cyklicky zapíná a vypíná přiřazenou LED diodu. Implementace vypínání a zapínání LED funguje za pomoci funkce delay. Tato implementace má spoustu nevýhod a samotný kód není ani zamýšlen jako funkční sám o sobě. Jeho účelem bylo pouze demonstrovat vliv prioritizace úloh.

Každá úloha s periodou 1 s přepíná stav LED diody.

Při spuštění programu, může být pozorováno následující chování. LED dioda ovládána úlohou s nejvyšší prioritou bliká s frekvencí blízkou 1 Hz (pozorováno na krátkém časovém úseku s ručním odečítáním času). S klesající prioritou úlohy také výrazně klesá reálná frekvence blikání řízené LED diody.



Obrázek 20 Jednoduché schéma programu pro ovládání LED diod

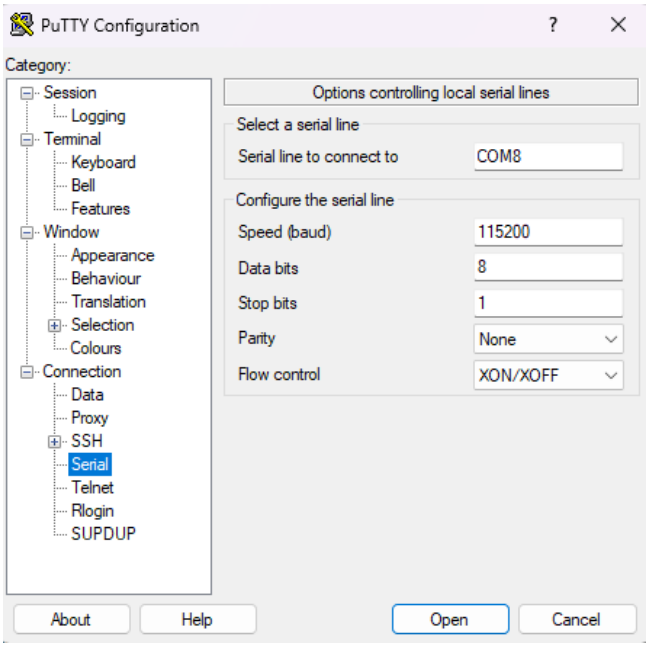
7.3 Komunikace skrze rozhraní UART

Z důvodu seznámení se s platformou byl vytvořen nový proces, který má za úkol poslat jednou za určený časový úsek jednoduchou zprávu pomocí UART rozhraní do připojeného počítače. Pro jednoduchost byla zpočátku zvolena zpráva v podobě konstanty.

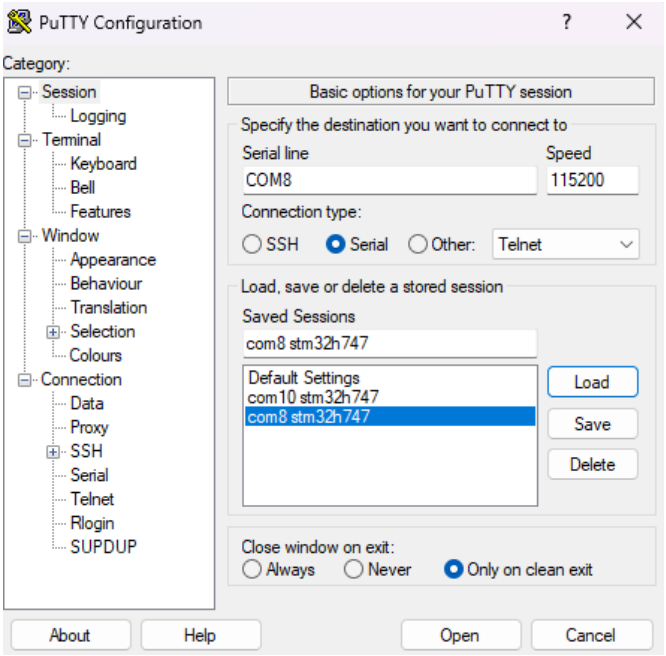
Zpráva je posílána každou vteřinu. Proces vykonávající tento program je označen jako UART. Zároveň byly zachovány předchozí procesy pro blikání LED diodami s jediným rozdílem, že proces UART používá jednu LED diodu pro indikaci odeslání zprávy. LED dioda blikne po odeslání zprávy pomocí UART. Zbylé dvě LED diody jsou ale stále využívány předchozími procesy.

Při implementaci tohoto programu byl využit UART 1, který je připojen k rozhraní ST-LINK, skrze který je pak signál přiveden do počítače ve formě virtuálního komunikačního portu, často označovaného jako VCOM.

Jako protistrana byl na počítači využit program PuTTY, který slouží pro komunikaci po sériové lince na straně počítače. PuTTY umožňuje čtení i posílání zpráv po lince. V rámci programu je možné také provést libovolnou konfiguraci dané komunikace. Pro použití v tomto projektu bylo potřeba provést nastavení zobrazené na obrázcích Obrázek 21 a Obrázek 22.

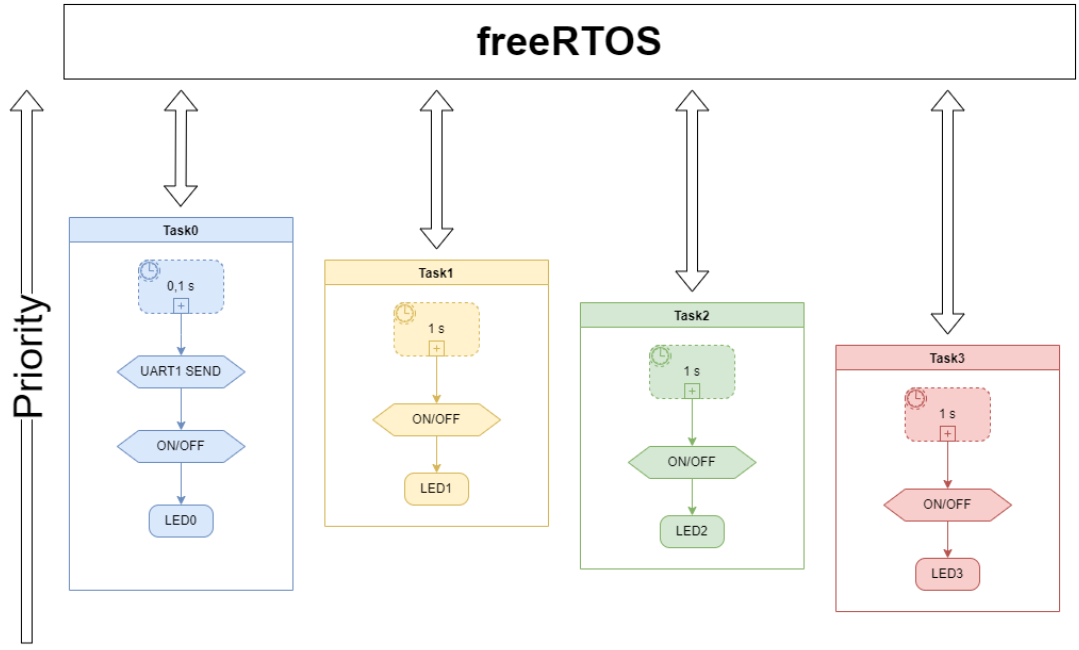


Obrázek 21 Konfigurace připojení k sériové lince pomocí programu PuTTY. Parametry komunikace jsou rychlost 115 200 baud, 8 datových bitů a 1 stop bit. Konfigurace je stejná u obou kitů s jediným rozdílem a to je rozdílná sériová linka v PC (COM8 a COM10).



Obrázek 22 Jelikož byly použity dva vývojové kity, tak byl také vytvořeny dvě rychlé předvolby pro propojení.

Schematické zobrazení zde popisovaného programu je zobrazeno na následujícím obrázku Obrázek 23. Program periodicky vysílá zprávu po sériové lince a přitom periodicky ovládá přítomné LED diody.



Obrázek 23 Diagram programu: seznámení se s platformou – UART

7.4 Práce s integrovaným RTC časovačem

Vývojový kit umožňuje využití RTC periferie. Při seznámení se s platformou bylo této periferie využito v přímočaré úloze, která získává aktuální čas z RTC a následně ji zašle pomocí UART rozhraní.

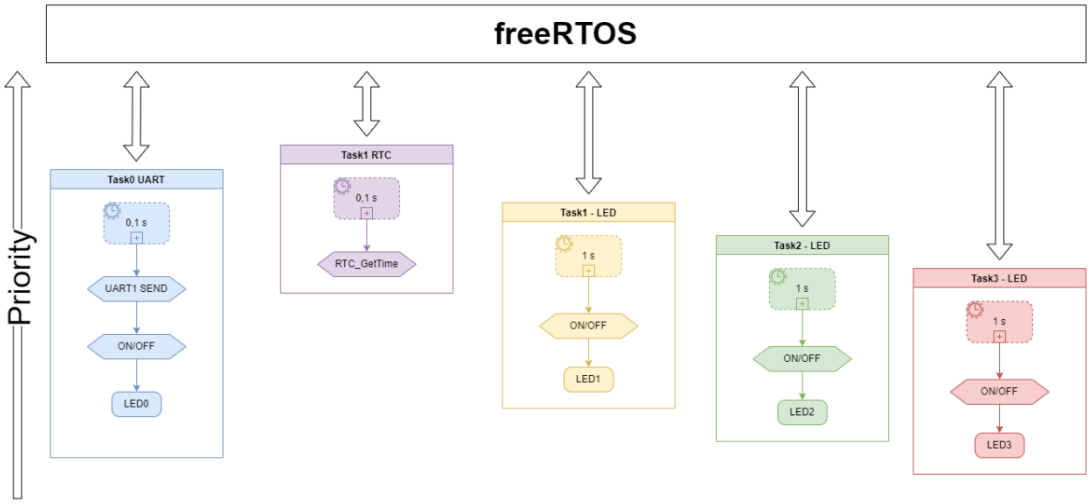
Specifikem práce s RTC registry v případně platformy STM32 je takzvaná ochrana proti přepisu. Ta spočívá v tom, že je potřeba přečíst vždy informace o čase ale i o datu. Respektive je potřeba zavolat funkci čtení obou registrů, aby došlo k obnovení registru čtení – pokud nedojde k zavolání obou funkci GetTime GetDate, tak zůstane stále původní hodnota v registru sloužícímu pro čtení a není přepsána. Zároveň je potřeba, aby proběhlo čtení hodnoty času, aby bylo možné přečíst hodnotu uloženou v registru pro datum.[18]

Pro plné pochopení této problematiky je také potřeba vysvětlit význam zdroje hodin (v technických kruzích referováno jako clock, zdroj hodin nebo zdroj hodinového pulsu) pro RTC. Pro RTC je s velikou oblibou volen jako zdroj hodinového signálu zdroj o frekvenci 32,768 kHz. Toto číslo působí na první pohled jako velice zvláštní a nepraktická volba, nicméně jeho výhoda spočívá v jednoduchosti převodu tohoto hodinového taktu na vteřiny, minuty a hodiny. Dále se jedná o takovou frekvenci, kterou dokáže většina MCU spolehlivě a stabilně udržovat.

Aby bylo možné aktuálně zjištěný čas přenášet mezi procesy je vhodné definovat proměnou pro uložení aktuálního času ve společném paměťovém prostoru. Jinak je proměnná dostupná pouze ve vyhrazeném paměťovém prostoru pro daný proces.

Z důvodu lepšího seznámení se s platformou byl proces UART upraven tak, aby posílal skrze UART rozhraní aktuální čas a datum. Proces byl rozšířen o čtení SSR registru – čtení tzv. subsecond registru. Skrze UART je nyní posílán aktuální absolutní čas ve formátu HH:MM:SS:mSmSmS YYYY:MM:DD.

Nastavení UART rozhraní je zde stejné jako v předchozím programu, tudíž se zde využívá nastavení popsaného v kapitole 7.3. I zde bylo využito nástroje PuTTY pro komunikaci s PC pomocí sériové linky.



Obrázek 24 Schéma programu pro posílání časové značky získané z RTC periferie

7.5 Čtení a zpracování dat z ADC

Princip fungování ADC převodníku a jeho základní popis je obsažen v kapitole 4.4. Tato kapitola není zaměřena na obecný popis, nýbrž na konkrétní implementaci v tomto projektu.

V prvotní fázi bylo potřeba vyzkoušet, jakým způsobem funguje ADC na zvolené platformě. Jednoduchý program, který je napsán pro FREERTOS, funguje na jednoduchém principu. Přečte hodnotu na vstupních pinech, převede danou napěťovou úroveň na digitální číslo a výsledek uloží do svého registru.

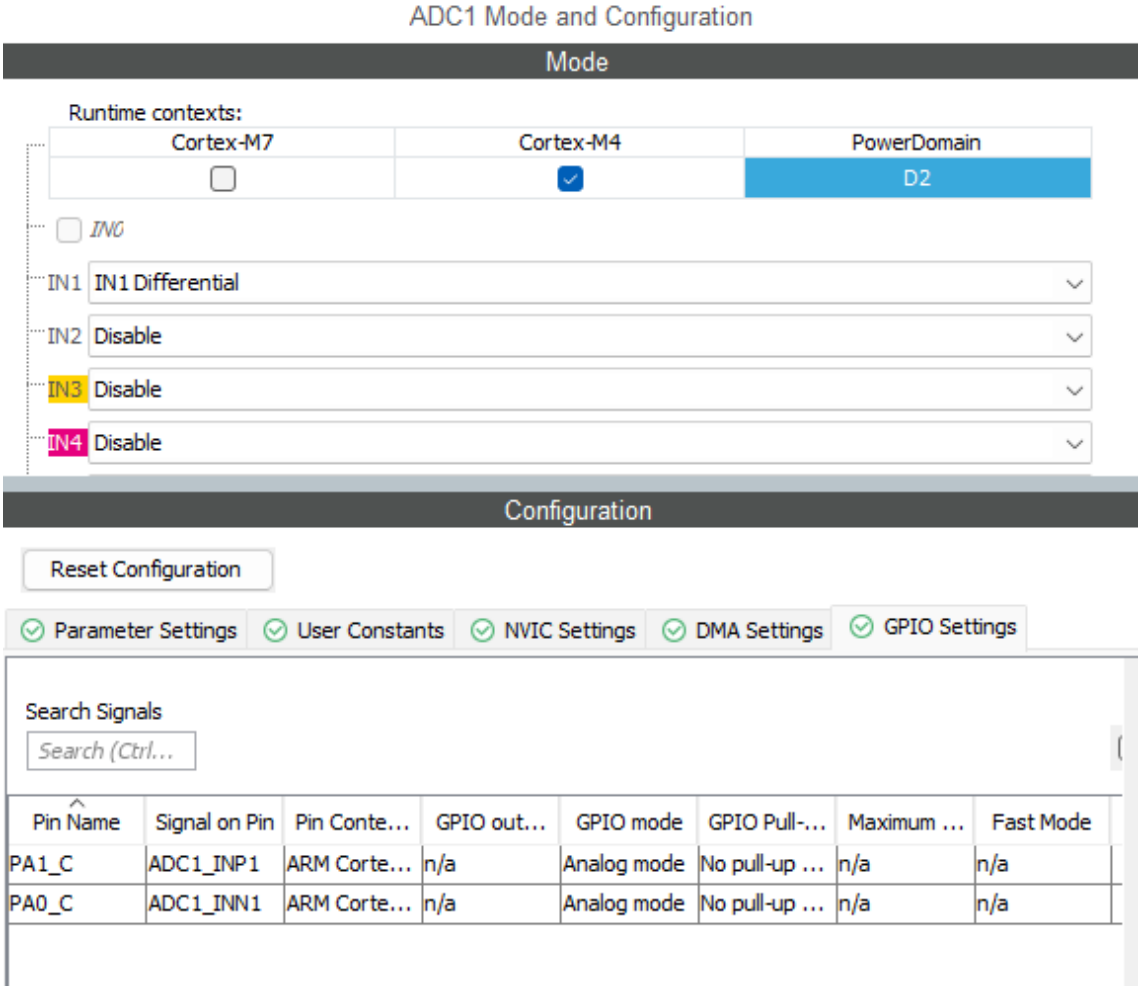
Pro provedení testování bylo využito diferenčního měření. Jedná se o způsob měření napětí na dvou pinech. Toto měření má výhodu většího rozsahu a také menší náchylnosti na případný šum či rušení.

Samotný program je periodický. Je potřeba dbát pozor na následující problémy. Při převodu je potřeba počkat na dokončení konverze, která je indikována změnou stavového registru.

Proces UART byl upraven tak, aby bylo skrze UART posíláno kromě času také hodnota na ADC. Formát zprávy je následující: hodnota/čas/datum.

Samotná konfigurace ADC je následující. Byla využita periferie ADC1. Program používá vstup 1 jako již bylo zmíněno jako diferenční vstup. Jako použité fyzické piny byly zvoleny piny PA0_C a PA1_C.

Výstřižek z konfiguračního prostředí CubeMX je přiložen jako následující obrázek Obrázek 25 .



Obrázek 25 Konfigurace ADC1. Nastavení vstupu, režimu, použitých fyzických pinů a přiřazení k jádru (M7/M4).

8. NÁVRH METODIKY MĚŘENÍ DOSAŽENÉ SYNCHRONIZACE A OČEKÁVANÉ VÝSLEDKY

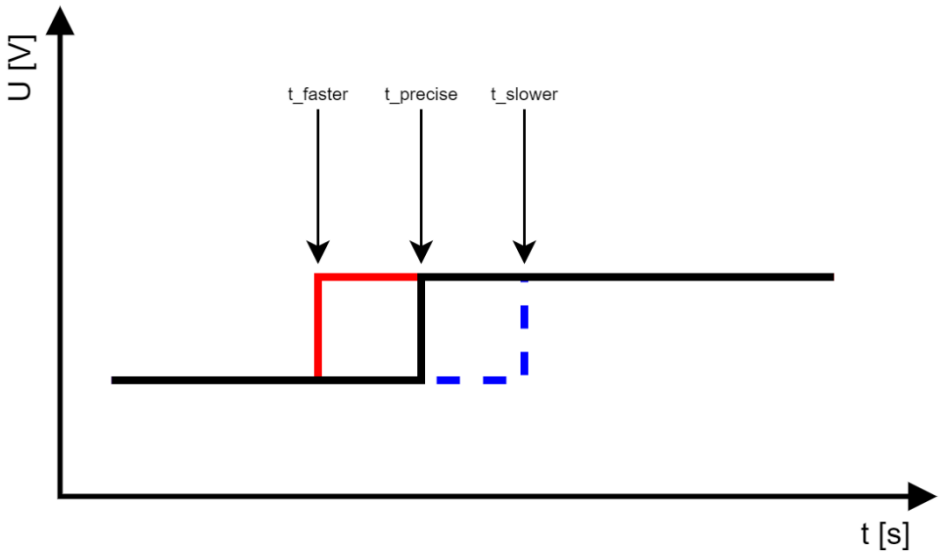
Kapitola se zabývá popisem navržené metodiky měření vlastností časové synchronizace. Jsou zde diskutovány metody, pomocí kterých je možné vyhodnotit dosaženou přesnost a stálost dosažené časové synchronizace.

Popsané metody jsou pouze jedny z mnoha možných přístupů k dané problematice.

8.1 Měření časového rozdílu při hardwarovém signálu

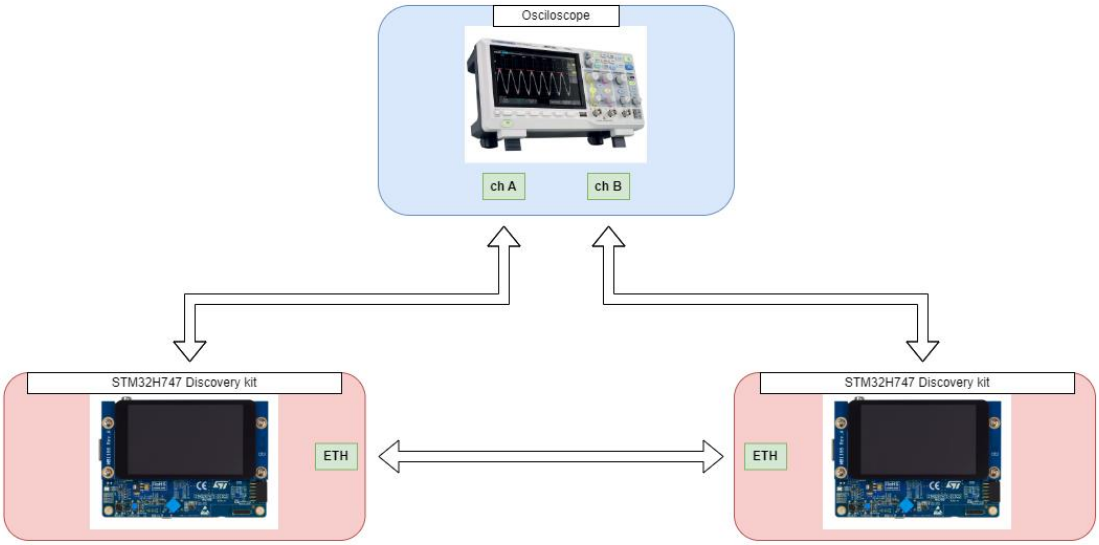
Dosažená přesnost časové synchronizace je změřena jednoduchým způsobem. Jedná se o porovnání a změření času, kdy dojde ke změně stavu na pinu.

Změna stavu pinu dojde ve specifikovaný čas. Měření probíhá na dvou samostatných kitech, na kterých je spuštěn stejný program, který změní v dané časy stavy pinů. Princip měření je znázorněn na obrázku:



Obrázek 26 Princip měření přesnosti dosažené časové synchronizace

Čas označený jako t_precise označuje přesný čas, neboli čas který je zadaný v programu. V optimálním případě by tak nastalo k přepnutí stavu obou kitů ve stejný moment. Časy t_faster a t_slower znázorňují případy, které ve skutečném světě díky nedokonalostem nastanou. Ve skutečnosti dojde k většímu či menšímu časovému posunu na jednu nebo druhou stranu.



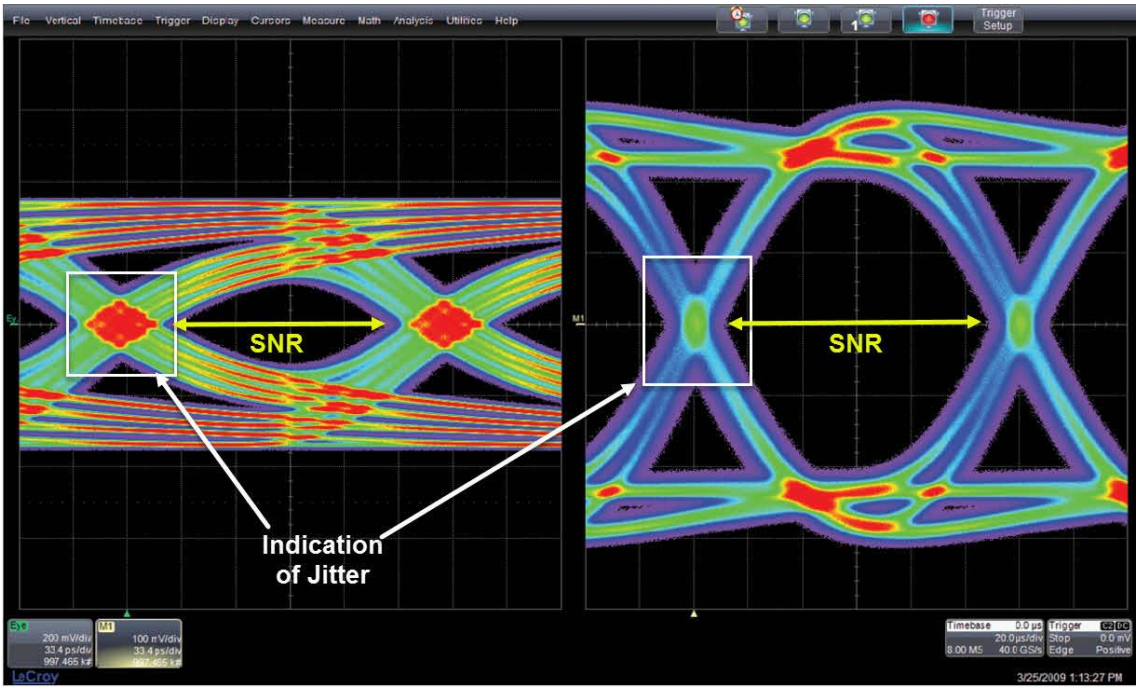
Obrázek 27 Měřící řetězec pro měření rozdílného času sepnutí pinu.

8.2 Měření jitteru pomocí eye diagramu

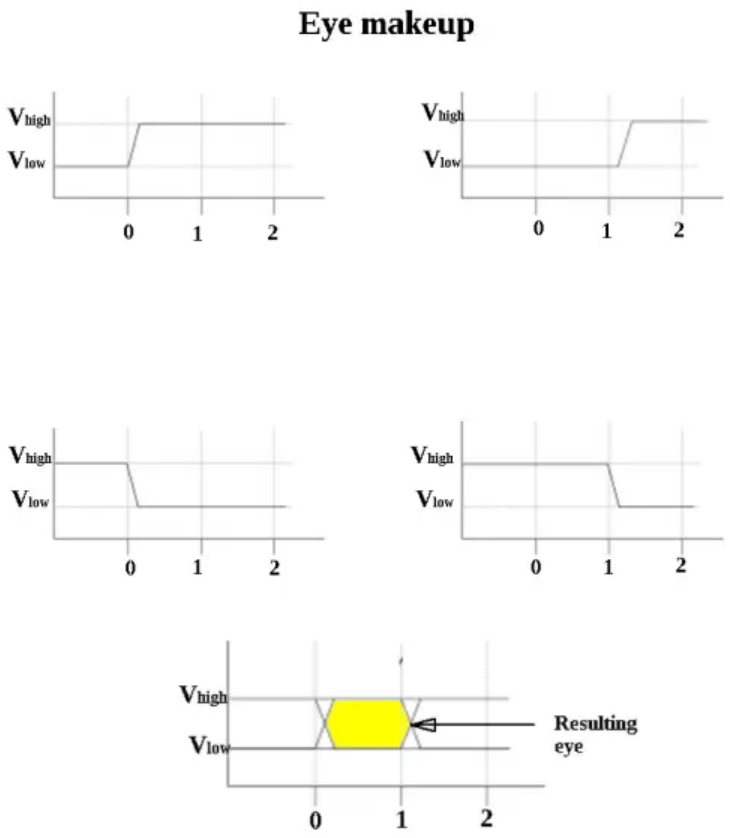
Měření obyčejného posunu signálu vůči sobě i při vyšším počtu měření umožní sice změřit řádovou přesnost synchronizace, nicméně pro změření jitteru neboli stability hodinového signálu, je možné změřit pomocí takzvaných eye diagramů.

Eye diagram je možné měřit tak, že dojde k sekvenci přepnutí bitů, aby se jejich složením přes sebe dal vytvořit kýžený obrazec. Postup je také vizuálně zobrazen na obrázku Obrázek 29. Tento postup se provede opakovaně. Výsledný obrazec je zobrazen na obrázku Obrázek 28 a dá se z něj získat celá řada informací. Mezi nejpodstatnější patří informace o velikosti jitteru, variabilita high a low úrovně, nástupná a sestupná doba pro daný signál a také jeho jeho tvar.[25][26][27]

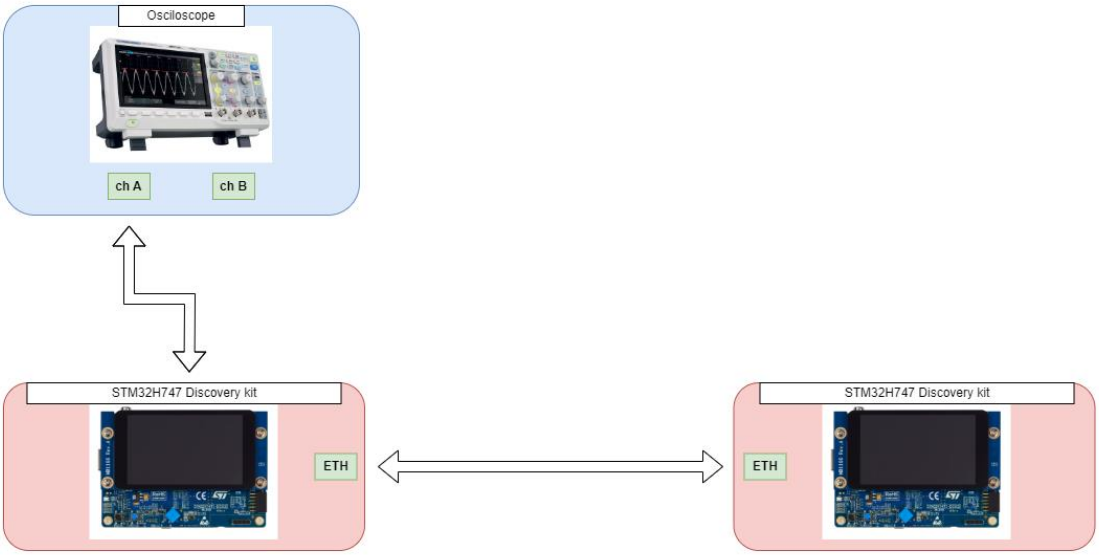
Měření jitteru pomocí eye diagramu je možné provést na jednom vývojovém kitu, nevyžaduje na rozdíl od předchozí metody provádět měření na více zařízení najednou.



Obrázek 28 Vizualizace eye diagramu a zobrazení jitteru. [25]



Obrázek 29 Postup tvorby eye diagramu [27]



Obrázek 30 Měřící řetězec pro eye diagram.

8.3 Pozorování přenášených zpráv při časové synchronizaci

Další způsobem je možnost skrze monitorovací nástroj Wireshark. Zajímavé výsledky by mohlo přinést sledování zpráv obsahující časový offset a porovnat tak reálnou přesnost z měření metodou popsanou v kapitole 8.1 a 8.2 a časový rozdíl, pomocí kterého se snaží samotné zařízení synchronizovat.

8.4 Pozorování PPS výstupu

Posledním navrženým způsobem vyhodnocení dosažené synchronizace je analýza výstupního signálu na PPS výstupu.

PPS označuje jednoduchý signál, který se opakuje s frekvencí 1 Hz. Jedná se vlastně určitým způsobem o PWM signál s úrovní cyklu 10%, jelikož délka pulsu je přesně 100 ms. Tohoto je možné využít při připojení PPS výstupů obou dvou vývojových desek k osciloskopu. Pak je možné již analyzovat více za sebou jdoucích cyklů a změřit jejich přesnou periodu a případně také vyhodnotit skutečnou délku pulsu. Tyto výsledky je možné vyhodnotit u každého kitu samostatně či porovnat výstupní signály obou připojených zařízení.

9. VYHODNOCENÍ PRÁCE NA DANÉ PLATFORMĚ, VZNIKLÉ PŘEKÁŽKY A DISKUZE DALŠÍHO MOŽNÉHO POSTUPU

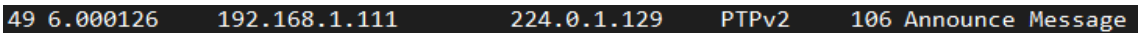
Během práce bylo potřeba řešit řadu problémů, které ovlivnily pokrok projektu. Jedním z hlavních problémů bylo, že výchozí knihovna byla navržena pro odlišnou platformu, než je STM32H747. Navíc od poslední aktualizace této knihovny uplynulo již několik let, což vedlo k nekompatibilitě některých funkcí a struktur, které byly v průběhu vypracování přejmenovány nebo nahrazeny.

Dalším faktorem bylo odstranění oficiálního postupu pro implementaci PTP protokolu ze stránek výrobce mikrokontroleru. I přesto, že je výrobce informován o zájmu zákazníků o podporu PTP, tento problém pro něj není prioritní.

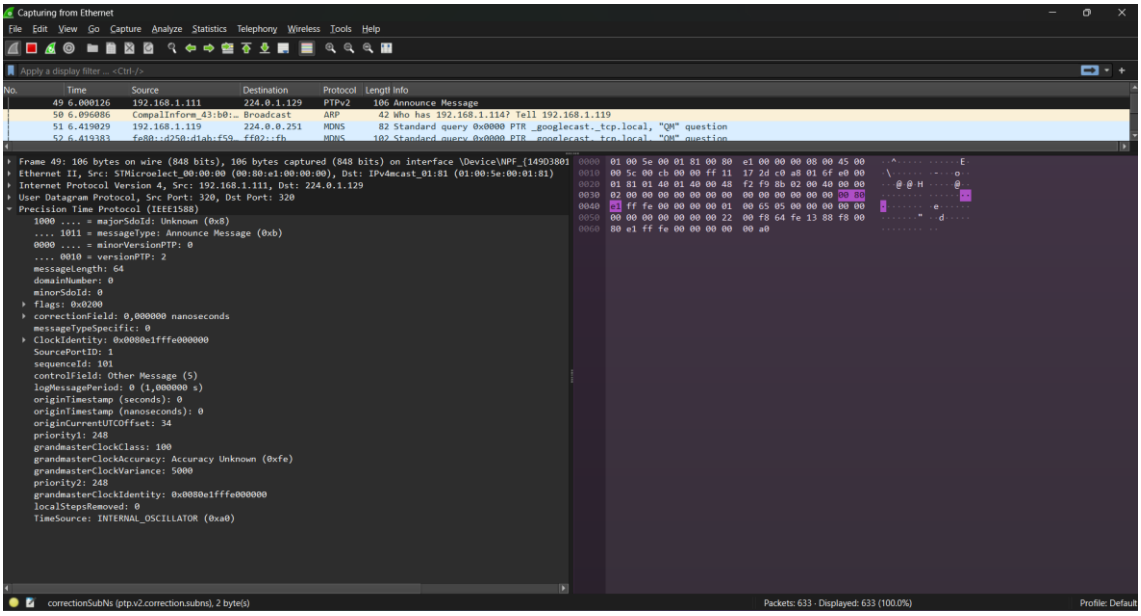
Funkce knihovny HAL, které jsou využívány v implementaci PTPd, jsou navrženy pro jiný mikrokontroler, konkrétně pro STM32F4. Tedy bylo nutné provést přizpůsobení těchto funkcí pro použití na platformě STM32H747.

Aby bylo možné využívat přítomnost ethernetového rozhraní na zvolené platformě, muselo být provedeno několik kroků, z nichž některé nebyly zcela jasné nebo jednoznačně zdokumentované v dostupných manuálech a materiálech. Pro úspěšné zprovoznění ethernetového rozhraní bylo potřeba provést řadu kroků jak na hardwarové úrovni, tak i na softwarové úrovni. Pro zprovoznění bylo potřeba provést komplexní analýzu dané periferie, paměťového prostoru a také fyzických vlastností vývojové desky. Ethernetové rozhraní bylo nakonfigurováno a úspěšně otestováno. Problematice zprovoznění ethernetového rozhraní se věnuje samostatná kapitola.

Pomocí funkčního ethernetového rozhraní bylo možné dosáhnout odeslání dvou typů zpráv PTP protokolu a jejich analýzy pomocí programu Wirewshark, jak je zobrazeno na následujících obrázcích. Další typy zpráv či jejich zpracování nejsou funkční.



Obrázek 31 Zpráva odeslána z vývojového kitu (192.168.1.111) na multicast adresu 224.0.1.129, kterou standardně využívá PTP protokol.



Obrázek 32 Informace o odeslané zprávě typu Announce

```
priority1: 248
grandmasterClockClass: 100
grandmasterClockAccuracy: Accuracy Unknown (0xfe)
grandmasterClockVariance: 5000
priority2: 248
grandmasterClockIdentity: 0x0080e1fffe000000
localStepsRemoved: 0
TimeSource: INTERNAL_OSCILLATOR (0xa0)
```

Obrázek 33 Detail zprávy typu Announce, týkající se parametrů typu zdroje hodin, a sloužící pro BMCA algoritmus.

I přes celkově pozitivní uživatelský zážitek z používání vývojových nástrojů se objevilo několik oblastí, které práci ztěžovaly či neumožňovaly provést všechny potřebné kroky.

Doporučením by mohla být změna platformy na takovou, na které je již ověřena funkčnost a kompatibilita s daným projektem, například STM32F4. Další možností je počkat na oficiální podporu protokolu PTP od výrobce či vývojáře lwIP knihovny. Dále je možné hlouběji přepracovat knihovnu, případně ji vytvořit od základu, aby byla plně kompatibilní s platformou STM32H747.

9.1 Přívětivost práce na platformě

Samotná práce se zvolenou platformou byla celkově obstojná. Stejně jako na každé platformě zde bylo možné najít slabší či silnější body.

Dostupné podklady od výrobce obsahují rozsáhlý popis jednotlivých problémů a problémových oblastí. Nicméně v některých případech není zcela intuitivní

vyhledávání potřebných informací. Orientaci uživatele také nepomáhá označování jednotlivých dokumentů, které neobsahuje na první pohled informaci o tom, ke kterému projektu se daný manuál vztahuje, či jaké oblasti se týká. Přehlednost informací nezjednodušila ani rozsáhlost referenčního manuálu.

Zásadní výhodou této platformy je podpora od samotného výrobce, který dává k dispozici širokou škálu programů a vývojových prostředí. Nejzásadnějším je STM32IDE, což je plnohodnotné vývojové prostředí s častými aktualizacemi a vylepšeními. Toto prostředí bylo v průběhu vypracování projektu velice stabilní a dobře optimalizované. Výhodou je také jeho vizuální podobnost všeobecně známým a často využívaným IDE.

Další výhodou je snaha výrobce o udržování a vylepšování vydaných produktů. V průběhu práce došlo ke dvěma aktualizacím firmwaru pro vývojový kit, přinášejícím opravy chyb či zvýšenou stabilitu.

Celkově se na platformě pracuje dobře, nicméně situace se trochu mění v momentě, kdy je cílem dosáhnout tak specifického cíle, jako je například časová synchronizace využívající protokol PTP. Výrobce se, jak vyplývá z odpovědí zástupců firmy a reakcí ostatních uživatelů na platformě STM32H747, této konkrétní problematice aktivně nevěnuje. Na komunitním fóru padlo opakovaně, že o řadě přání zákazníků a uživatelů ví, nicméně této problematice není přiřazena vysoká priorita, a proto se dá očekávat, že tento požadavek nebude brzy vyřízen.[32][33]

ZÁVĚR

Tato diplomová práce se zabývala analýzou možností implementace časové synchronizace pomocí Precision Time Protocol (PTP či IEEE 1588) na platformě STM32H747. Cílem bylo seznámit se s možnostmi této platformy a ověřit její schopnosti implementovat a obsluhovat časovou synchronizaci pro časové značkování vzorků z ADC.

V úvodní části byla provedena literární rešerše na téma časové synchronizace a PTP protokolu, která poskytla hlubší porozumění problematice a příslušným technologiím. Poté následoval popis dostupných knihoven, zejména PTPd a lwIP, které byly při vývoji aplikace využity. Použité knihovny bylo potřeba upravit pro zvolenou platformu. Knihovna byla implementována, nicméně v průběhu testování se podařilo dosáhnout pouze odeslání úvodních dvou zpráv protokolu PTP, konkrétně zprávy typu Announce a Sync. Zpracování přijatých zpráv či odeslání dalších typů zpráv nebylo úspěšně implementováno.

Další část práce byla zaměřena na konfiguraci periferií, jako UART, RTC a ADC, a také konfiguraci ethernetového rozhraní, které je klíčové pro úspěšnou synchronizaci a komunikaci v síti. Dále byla v této části práce popsána konfigurace freeRTOS, která probíhala pomocí CubeMX a CubeIDE. Součástí konfigurace je také konfigurace paměťového prostoru.

Aby bylo možné začít pracovat na časové synchronizaci, bylo potřeba zprovoznit také síťové rozhraní ethernet. To samo o sobě nebylo zcela přímočarou úlohou, mimo jiné z důvodu potřeby provést hardwarové změny, kterými byla změna konfigurace pájených mostů pro odpojení mikrofonu, který není pro tento projekt podstatný. Odpojením periferie mikrofonu bylo možné připojit periferii ethernet, která je pro tuto aplikaci kritická. Po fyzickém zapojení ethernetu bylo možné provést konfiguraci na úrovni softwaru a paměti. Ethernet byl úspěšně zprovozněn a umožňuje komunikaci pomocí UDP/IP. Součástí zprovoznění bylo také otestování pomocí ICMP protokolu skrze odeslání ping zprávy. Zařízení na ping zprávu správně zareagovalo. V rámci testování zařízení také odeslalo skrze ethernet PTP zprávy typu Announce a Sync, které bylo možné zachytit a analyzovat pomocí nástroje Wireshark.

Samostatná kapitola se věnuje také návrhu metodiky pro měření časové synchronizace na cílové platformě. V rámci práce byly navrženy metody pro měření přesnosti dosažené synchronizace a také způsobem měření v podobě tvorby eye diagramu pro měření velikosti jitteru.

Součástí je také popsaná konfigurace jednotlivých procesů operačního systému freeRTOS. Navržená konfigurace systému a jeho jednotlivých procesů může sloužit jako základ pro aplikaci, která bude přiřazovat časové značky k naměřeným vzorkům z přítomného ADC převodníku a vyhrazený proces zajistí časovou synchronizaci pomocí PTP protokolu.

Závěrečná část práce popisuje problémy, které se vyskytly v průběhu vývoje, a které se podařilo překonat, nebo které přetrvávají. Součástí je také návrh dalších kroků pro implementaci časové synchronizace společně s přiřazováním časové značky k naměřeným vzorkům.

Celkově lze konstatovat, že i přes výzvy a překážky v implementaci časové synchronizace pomocí PTP protokolu na platformě STM32H747, práce přinesla cenné poznatky o možnostech a omezeních této platformy v oblasti časové synchronizace a síťové komunikace. Tyto poznatky mohou sloužit jako cenný zdroj informací pro další výzkum a vývoj v oblasti vestavěných systémů, časové synchronizace a využití této platformy v síťové infrastruktuře.

LITERATURA

[1] IEEE STANDARD ASOCIATION. IEEE 1588, IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. 16.10.2023. [cit. 2023-10-16].

[2] CENA, G.; SCANZIO, S. a VALENZANO, A. Reliable comparison of clock discipline algorithms for time synchronization protocols. In: 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA). 2015, s. 1-9. ISSN 1946-0759. Dostupné z: https://doi.org/10.1109/ETFA.2015.7301461.

[3] Keeping Time with Precision Time Protocol (PTP). Online. In: WAIDSON, Michael. Tektronix, 2019, s. 18. Dostupné z: https://www.videoservicesforum.org/events_archive/2019-04_NABShow2019/Tuesday/1500%20michael%20waidson%20-%20IPShowcase-KeepingTimePTP-Tektronix-NAB%202019.pdf. [cit. 2023-10-04].

[4] WIEBEL, Hans. Synchronization over Ethernet: Standard for a Precision Clock Synchronization Protocol according to IEEE 1588 Synchronous Ethernet according to ITU-T G.8261. PDF. Neznámo. Zurich University of Applied Science, 2008. Dostupné z: http://ines.zhaw.ch/ieee1588. [cit. 2023-10-24].

[5] Introduction to Precision Time Protocol (PTP). Online. NetworkLessons. Neznámý. Dostupné z: https://networklessons.com/cisco/ccnp-encor-350-401/introduction-to-precision-time-protocol-ptp. [cit. 2023-10-04].

[6] PERLE. PTP - Precision Time Protocol - IEEE 1588 V1 and V2 PTP Boundary clock capabilities in Industrial Managed Switches. Online. PERLE. Perle.com. Neznámý. Dostupné z: https://www.perle.com/supportfiles/precision-time-protocol.shtml. [cit. 2023-10-16].

[7] AN-1838 IEEE 1588 Boundary Clock and Transparent Clock Implementation Using the DP83640. PDF. 2013. Texas Instruments, 2013. [cit. 2023-11-15].

[8] UM2411 User manual: Discovery kit with STM32H747XI MCU. PDF. 6. revize. ST, 09.2023n. l.

[9] RM0399 Reference manual: STM32H745/755 and STM32H747/757 advanced Arm®-based 32-bit MCUs. 4. revize. ST, 06.2023n. l. [cit. 2023-11-06].

[10] STMICROELETRONICS. STM32H747xI/G. 2. 2023. Online. STMicroelectronics. 2023. Dostupné z: https://www.st.com/resource/en/datasheet/stm32h747ai.pdf. [cit. 2023-12-05].

[11] Which ADC Architecture Is Right for Your Application? Online. In: ADI Analog Dialog. Neznámý. Dostupné z: https://www.analog.com/en/analog-dialogue/articles/the-right-adc-architecture.html. [cit. 2023-12-01].

[12] Activity: Successive Approximation Register (SAR) ADC. Online. In: Wiki Analog. 2021. Dostupné z: https://wiki.analog.com/university/courses/alm1k/alm-signals-labs/alm-sar-adc-1. [cit. 2023-11-27].

[13] How does Successive Approximation (SAR) ADC Work and Where is it best used? Online. In: Circut digest. 2020. Dostupné z: https://circuitdigest.com/article/how-does-successive-approximation-sar-adc-work-and-where-is-it-best-used. [cit. 2023-12-16].
[14] EIDSON, John C. Measurement, control, and communication using IEEE 1588. London: Springer, c2006. ISBN 18-462-8250-0.
[15] Overview of IRIG-B Time Code Standard. Neznámo. Cyber-Science, 2017. Dostupné z: www.cyber-sciences.com. [cit. 2023-12-25].
[16] Direct Memory Acces in OS. Online. Geeks4Geeks. 2024. Dostupné z: https://www.geeksforgeeks.org/direct-memory-access-in-os/. [cit. 2024-03-08].
[17] The Strengths of 32.768 kHz Oscillators. Online. Jauch Blog. 2020. Dostupné z: https://www.jauch.com/blog/en/the-strengths-of-32-768-khz-oscillators/. [cit. 2024-03-12].
[18] 82. On the RTC readout lock mechanism (and its deficiencies). Online. 2021. Dostupné z: http://www.efton.sk/STM32/gotcha/g82.html. [cit. 2024-03-22].
[19] SOURCEFORGE. Precision Time Protocol daemon. Online. SourceForge. 2005. Dostupné z: https://sourceforge.net/projects/ptpd/. [cit. 2024-04-04].
[20] PTP Deamon. Online. GIT Hub. Neznámý. Dostupné z: https://github.com/ptpd/ptpd. [cit. 2024-04-04].
[21] PTPD. Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 2011. Dostupné z: https://en.wikipedia.org/wiki/PTPd. [cit. 2024-04-12].
[22] PTPD v2 projects for ST NUCLEO-F429ZI development board. Online. GIT Hub. Neznámý. Dostupné z: https://github.com/mpthompson/stm32_ptpd. [cit. 2024-04-22].
[23] STM32 Nucleo-144 development board with STM32F429ZI MCU, supports Arduino, ST Zio and morpho connectivity. Online. STMicroelectronics. 2015. Dostupné z: https://www.st.com/en/evaluation-tools/nucleo-f429zi.html#documentation. [cit. 2024-04-24].
[24] PTPD FOR STM32H7 AND ATSAME70. Online. GIT Hub. 2014. Dostupné z: https://github.com/hasseb/stm32h7_atsame70_ptpd. [cit. 2024-04-30].
[25] What does an eye diagram or eye pattern on an oscilloscope mean? Online. MicrocontrolersTips. 2017. Dostupné z: https://www.microcontrollertips.com/eye-diagram-eye-pattern-oscilloscope-mean/. [cit. 2024-05-01].
[26] What is an Eye Diagram? Online. Altium. 2022. Dostupné z: https://resources.altium.com/p/what-eye-diagram. [cit. 2024-05-01].
[27] The Eye Diagram: What is it and why is it used? Online. Test and measurement tips. 2016. Dostupné z: https://www.testandmeasurementtips.com/basics-eye-diagrams/. [cit. 2024-05-01].
[28] How to create project for STM32H7 with Ethernet and LwIP stack working. Online. Community ST. 2020. Dostupné z: https://community.st.com/t5/stm32-

mcus/how-to-create-project-for-stm32h7-with-ethernet-and-lwip-stack/ta-p/49308. [cit. 2024-02-07].
[29] STM32 ETHERNET [@ControllersTech]. Online. 2021. Dostupné z: YouTube, https://www.youtube.com/watch?v=8r8w6mgSn1A&list=PLfIJKC1ud8ggZKVtytWAlOS63vifF5iJC. [cit. 2024-03-07].
[30] Introduction to Free RTOS in STM32 [@Contro]. Online 2021. Dostupné z: YouTube, https://www.youtube.com/watch?v=muOL9SH0p9g&list=PLfIJKC1ud8gj1t2y36sabPT4YcKzmN_5D. [cit. 2024-03-07].
[31] STM32 Dual Core [@ControllersTech]. Online. 2021. Dostupné z: YouTube, https://www.youtube.com/watch?v=jI1k6p-fduE&list=PLfIJKC1ud8ghx7DRNbhVbWi7n7Ty0YstS. [cit. 2024-03-07].
[32] STM Community [@STM]. Online. 2021. Dostupné, https://community.st.com/. [cit. 2024-03-07].
[33] PTP support in HAL drivers STM32H7. Online. STMicroelectronics. 2020. Dostupné z: https://community.st.com/t5/stm32-mcus-embedded-software/ptp-support-in-hal-drivers-stm32h7/td-p/295602. [cit. 2024-04-24].

Seznam symbolů a zkratek

Zkratky:

AD	Analogově Digitální
ADC	Analog Digital Converter
BC	Boundary Clock
BMCA	Best Master Clock Algorithm
ED	Edge Device
FEKT	Fakulta elektrotechniky a komunikačních technologií
GMC	Grand Master Clock
HW	Hardware
IGMP	Internet Group Managment Procoll
MC	Master Clock
NTP	Network Time Protocol
OC	Ordinary Clock
PTP	Precision Time Protocol
RMII	Reduced Media Independent Interface
RTOS	Real Time Operating Systém
SAR	Succesive Aproximation Register
SC	Slave Clock
SW	Software
TC	Transparency Clock
USART	Universal Synchronous Asynchronous Receiver Transmitter
VUT	Vysoké učení technické v Brně

SEZNAM PŘÍLOH

Příloha A - Projekt obsahující nakonfigurovaný projekt pro freeRTOS – přiloženo v elektronické příloze