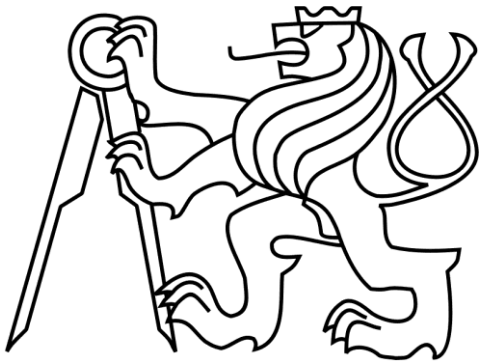
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra měření



Bakalářská práce

Návrh a programové vybavení synchronizační jednotky

Jakub Ešner

vedoucí práce: doc. Ing. Jan Fischer CSc.

Praha, 2012

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
Katedra měření

Akademický rok **2011-12**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	**Jakub Ešner**
Studijní program:	**Kybernetika a robotika**
Obor:	**Senzory a přístrojová technika**
Název tématu česky:	**Návrh a programové vybavení synchronizační jednotky**
Název tématu anglicky:	**Design and Programming of the Synchronization Unit**

Pokyny pro vypracování:

Navrhněte a vytvořte programové vybavení pro jednotku distribuovaného systému synchronizovaného s využitím protokolu PTP IEEE 1588. Pro tuto jednotku řízenou procesorem s jádrem ARM Cortex-M3 v provedení STM32F207 sestavte program pro synchronizovaný sběr dat a ovládání rychlých binárních výstupů, dále program pro synchronizaci a ovládání dalších podřízených jednotek zajišťujících sběr dat a ovládání akčních členů. Experimentálně ověřte správnost funkce programu v jednotce.

Seznam odborné literatury:

[1] Yiu, J.: The definitive Guide to the ARM Cortex- M3. Elsevier, 2007
[2] STMicroelectronics: RM0033 Reference manual, Doc ID 15403 Rev 3
[3] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)

Vedoucí bakalářské práce:	doc. Ing. Jan Fischer, CSc.
Datum zadání bakalářské práce:	6. prosince 2011
Platnost zadání do[1]:	1. února 2013

L.S.

Prof. Ing. Vladimír Haasz, CSc.
vedoucí katedry

Prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 6. 12. 2011

[1] Platnost zadání je omezena na dobu dvou následujících semestrů.

Abstrakt

Tato práce se zabývá návrhem programového vybavení pro distribuovaný systém synchronizovaný protokolem PTP IEEE-1588. Systém je složený z Ethernetových modulů s procesorem z rodiny STM32 s jádrem ARM Cortex-M3. Součástí práce je také návrh konfiguračních příkazů, ovládací aplikace na PC a následná demonstrace funkčnosti celého systému.

Abstract

The aim of this thesis is to design software for distributed system synchronized by protocol PTP IEEE-1588, which is composed of Ethernet modules based on microcontrollers with ARM Cortex-M3 core. It will also include design of configuration commands and control application for PC and finally demonstrate functionality of the whole system.

Čestné prohlášení autora práce

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

V Praze dne …… Podpis ……

Poděkování

Rád bych tímto poděkoval vedoucímu práce, doc. Ing. Janu Fischerovi CSc. za vedení práce a odborné konzultace. Dále bych rád poděkoval svým rodičům, Blance Ešnerové a Jaroslavu Ešnerovi, za neocenitelnou morální a materiální podporu v průběhu celého mého studia.

Obsah

Seznam obrázků

1 Úvod

1.1 Úvod do problematiky

V dnešní době se můžeme v oblastech průmyslové automatizace setkávat s čím dál tím větší oblibou distribuovaných systémů. U modulů takovýchto systémů je potřeba řešit otázku přesné časové synchronizace spolu se zachováním minimálních pořizovacích nákladů. Jedním způsobem, jak docílit nízké ceny, je velkosériová výroba, které lze dosáhnout vytvořením universálních modulů, které dokážou změnou nastavení zastat funkci senzoru i aktuátoru. Dalším způsobem je využít hojně rozšířených a cenově dostupných komponent.

Synchronizace lze dosáhnout více způsoby. Čistě softwarový přístup nedosahuje požadované přesnosti. Systémy, které vyžadují dodatečné vodiče zajišťující synchronizaci, potřebují atypické kabely, což zvyšuje pořizovací náklady.

Kompromisem mezi oběma přístupy je využití protokolu časové synchronizace, která je založena na výměně zpráv a funguje na běžném komunikačním rozhraní s hardwarovou podporou v mikroprocesorech a řadičích daného rozhraní. Za tímto účelem byl navržen protokol PTP (precision time protocol), který je od roku 2002 standardizován pod označením IEEE-1588 [1].

1.2 Cíl práce

Cílem této práce je navrhnout a implementovat program pro modul distribuovaného systému, navrhnout ovládací aplikaci nadřazeného systému (PC) umožňující správu distribuovaného systému a navrhnout formát konfiguračních příkazů.

Program modulu zajistí přesné ovládání binárních výstupů, synchronizovaný sběr dat, zpracování asynchronních událostí a převod analogových signálů

integrovaným A/D převodníkem. Bude využito protokolu Precision Time Protocol (PTP), který synchronizuje jednotlivé moduly distribuovaného systému.

Dále bude vytvořena aplikace pro nadřazený systém (PC), která má být schopna přehledně a rychle spravovat distribuovaný systém. Pro účely demonstrace bude obsahovat rozhraní pro hromadné ovládání a konfiguraci systému.

Pro možnost základní konfigurace bez nutnosti nahrávat do procesoru nový firmware budou implementovány příkazy standardu „Standard Commands for Programmable Instruments“ [2] (SCPI). Tyto příkazy umožní nastavení odesílání asynchronních zpráv, obsahujících převážně naměřené hodnoty s možností odesílat data do jiných modulů, což je základ k vytvoření systému bez přítomnosti nadřazené řídící jednotky (zpravidla PC).

Závěrem práce bude změření kvality časové synchronizace mezi moduly a provedení jednoduchých experimentů demonstrujících funkčnost AD převodníku a ovládání synchronních binárních výstupů.

2 Rozbor problematiky

2.1 Způsoby časové synchronizace

Způsoby časové synchronizace lze zjednodušeně rozdělit do tří kategorií. Čistě softwarová synchronizace, synchronizace využívající standardního rozhraní a vodičů s hardwarovou podporou v řadičích a synchronizace využívající přídavných synchronizačních vodičů.

Nejjednodušším způsobem synchronizace je čistě softwarový přístup, mezi které patří například Network Time Protocol (NTP), se kterým se nejčastěji setkáme v prostředí internetu, kde jeho přesnost synchronizace 1-10ms vyhovuje drtivé většině požadavků. Tohoto způsobu synchronizace využívají i zařízení LXI [3] (LAN eXtensions for Instrumentation) kategorie C.

Dalším možným řešením je použití řadičů s hardwarovou podporou synchronizace. Na tomto principu staví protokol PTP, se kterým se můžeme setkat u zařízení LXI-B. Výhodou je dosažení přesnosti synchronizace s chybou menší než 1us a přitom zachování příznivé ceny.

Pokud by ani tento způsob synchronizace nedostačoval, tak je potřeba zvolit metodu využívající přídavné synchronizační vodiče. Této synchronizace využívají zařízení LXI-A, se kterými se ale moc nesetkáme. Důvodem je, že použití samotného protokolu PTP pro většinu aplikací dostačuje a není potřeba používat atypické a tím pádem drahé kabely.

2.1.1 Protokol PTP

Protokol PTP definovaný normou Institute of Electrical and Electronics Engineers (IEEE-1588) je založen na výměně zpráv, obsahujících časové značky (time stamps), na jejichž základě se určí nejen aktuální rozdíl časů (offset) mezi nadřazeným (master) a podřízeným (slave) modulem, ale také rychlost jeho změny (drift). Na základě těchto informací provádí podřízená (slave) jednotka korekci svých hodin. Celý synchronizační cyklus se opakuje s periodou jedné sekundy.

2.2 TCP/IP stack

TCP/IP stack je nezbytnou součástí zařízení, které využívá Ethernetové rozhraní. Takovýto stack zajišťuje navázání spojení, komunikaci a následné ukončení spojení. Vzhledem k relativně malé paměti SRAM procesoru je použitá jeho odlehčená verze, nazývaná LwIP stack [4] (lightweight TCP/IP stack). Tento stack podporuje jak UDP (User Datagram Protocol), tak TCP (Transmission Control Protocol) transportní vrstvu. Zásadní je však podpora IGMP (Internet Group Management Protocol), díky kterému umožňuje zasílat *multicastové* zprávy, využívané pro zasílání synchronizačních PTP zpráv.

UDP na rozdíl od TCP poskytuje službou nespolehlivou, což znamená, že nepotvrzuje úspěšně doručené pakety. Výhodou je možnost hromadného odesílání paketů. První možností je *broadcast*, který odešle data všem v síti. Druhou možností je *multicast*, který posílá data jen těm účastníkům sítě, kteří se do daného *multicastu* zaregistrovali. Pro účely časové synchronizace je tato metoda ideální, protože nezatěžuje komunikační kanál a dorazí pouze modulům, které mají o synchronizační údaje zájem.

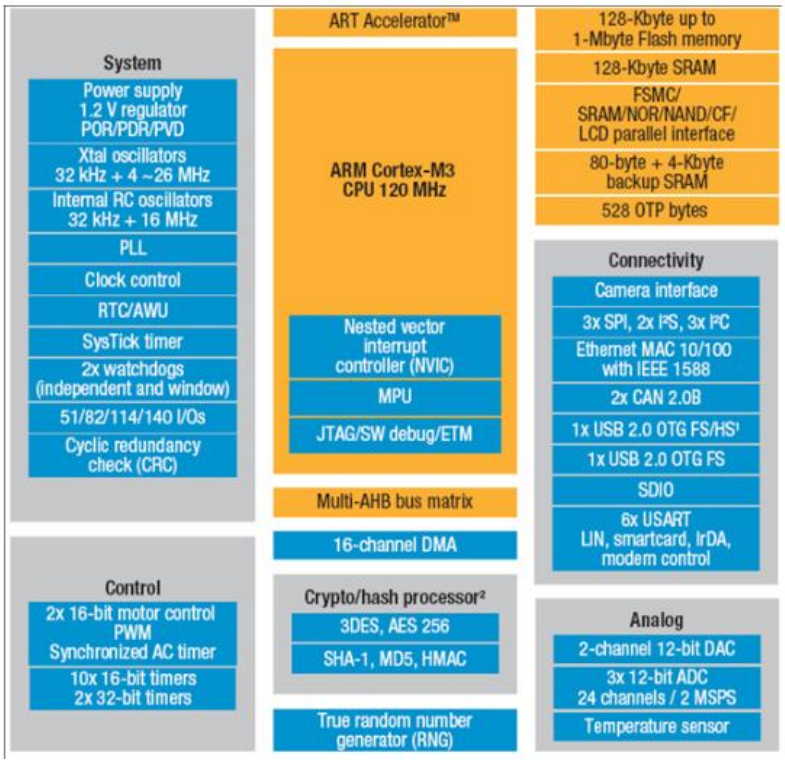
TCP poskytuje spolehlivou službu, která kvůli potvrzování každého úspěšně odeslaného paketu vyžaduje daleko větší režii. Tuto službu používá například síťový protokol Telnet (Telecommunication Network), používaný ke vzdálené správě. Pro účely nastavování modulů bylo tedy zvoleno TCP spojení na portu 5025, který se je zpravidla využíván laboratorními přístroji. Protože není dopředu známa adresa počítače, ze kterého bude probíhat konfigurace, a není žádoucí zatěžovat Ethernetový modul jejím hledáním, tak bude modul TCP serverem.

Velikou výhodou tohoto stacku je, že ho podporuje značné množství výrobců mikrokotrolerů, stejně jako firma ST Microelectronics, která poskytuje jeho implementaci spolu s příklady použití jak na UDP, tak na TCP transportní vrstvě [5].

2.3 Popis HW použitého modulu

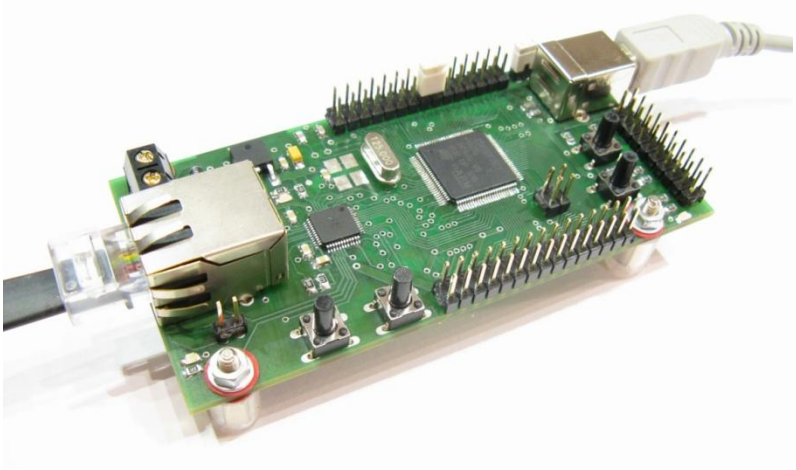
Použitý Ethernetový modul navržený v rámci práce [6] je vidět na obr. 2.2. Jádrem tohoto modulu je 32bitový mikrokontroler ARM z rodiny Cortex-M3 [7] firmy ST Microelectronics. Bude použit mikrokontroler nové generace STM32F2xx, konkrétně STM32F207VGT6 [8], který v porovnání s předcházející generací STM32F1xx přináší kromě vylepšených periferií [9] i zvýšení maximální pracovní frekvence na 120 MHz. Tento model je vybaven 128kB paměti SRAM (Static Random Access Memory) a 1MB paměti Flash.

Mikrokontroler obsahuje značné množstvím periferií [10], ze kterých bude využita především Ethernetová MAC vrstva s hardwarovou podporou protokolu PTP [11], jednotka časovače/čítače (TIMx), Universal synchronous/asynchronous receiver/transmitter (USART), A/D převodník (ADC), jednotka obsluhy přerušení (NVIC) a DMA (direct memory access) řadič. Blokové schéma použitého procesoru je znázorněno na obrázku obr. 2.1 blokové schéma mikrokontroleru stm32f2xx.



Obr. 2.1 Blokové schéma mikrokontroleru STM32F2xx

Procesor poskytuje velice široké možnosti nastavení jednotlivých periferií, které lze nakonfigurovat tak, aby spolupracovaly, a tím pádem nebyla nezbytná obsluha programem. Jejich nastavení je sice komplikované a vyžaduje podrobně prostudovat jejich možnosti, na druhou stranu však jejich použití šetří strojový čas procesoru a především vede k docílení deterministického chování s minimální latencí, což je pro v tomto případě, kde se pracuje s časem s rozlišením 20 nanosekund, více než žádoucí.



Obr. 2.2 Ethernetový modul

2.4 Konfigurační příkazy SCPI

Pro konfiguraci distribuovaného systému je potřeba použít nějaký formát konfiguračních zpráv a ovládacích příkazů. Za tímto účelem byl roku 1990 definován standard SCPI [2] (Standard Commands for Programmable Instrumens), který definuje především strukturu příkazů, jejich syntaxi, datové formáty a povinné obecné příkazy (Common Commands).

Příkazy se dělí na povinné příkazy dané SCPI normou, které by měly být obsaženy v každém zařízení. Takovéto příkazy začínají hvězdičkou (*). Druhou kategorií jsou tzv. specifické příkazy (Device Dependet Commands), které již nejsou pro všechny typy zařízení stejné.

Příkazy se dále dělí podle charakteru na dotazy a povely. Dotazy poznáme tak, že končí otazníkem (?). Příkazy nastavují, nebo ovládají zařízení. Často obsahují

konfigurační parametry. Tyto parametry jsou zpravidla čísla v desítkové soustavě převedené do ASCII textového řetězce.

Pro zjednodušení implementace SPCI příkazů byl použit *SPCI command procesor* [12]. Tato knihovna spravuje seznam příkazů, vyhledává v něm a následně vykoná předem definovanou funkci. Knihovna také obsahuje funkce dekódující data z textových řetězců.

Protože je takovýto modul distribuovaného systému nestandardním zařízením, tak pro něj neexistují SCPI příkazy. Některé přístroje jsou však svou funkcí podobné a tudíž mohly být jejich příkazy s mírnými úpravami použity.

2.4.1 Inspirace od LXI-B PTP Trigger Boxu Agilent E5818A

Největší inspirací byl PTP Trigger Box Agilent E5818A [13]. Kromě základních příkazů diagnostikující stav PTP synchronizace byly převzaty funkce Alarm (3.2.1), Ext (3.3) a Lanset (4).

Alarm je zdrojem událostí, které v předem specifikovaný čas vyvolá událost. Alarm umožňuje, díky možnosti nastavení počtu opakování se specifikovaným intervalem, generovat události periodicky.

Ext je zdrojem externích událostí, u kterého se konfiguruje pouze typ hrany, která vyvolá událost.

Poslední zajímavou funkcí je tzv. *Lanset*, který umožňuje odesílat asynchronní, resp. neočekávané, zprávy na nastavenou adresu v síti při dané události a s předem nastaveným prefixem. Prefix je textový řetězec předcházející posílaným datům, který pomůže příjemci rozpoznat, o jaký typ zprávy jde.

3 Práce s přesným časem

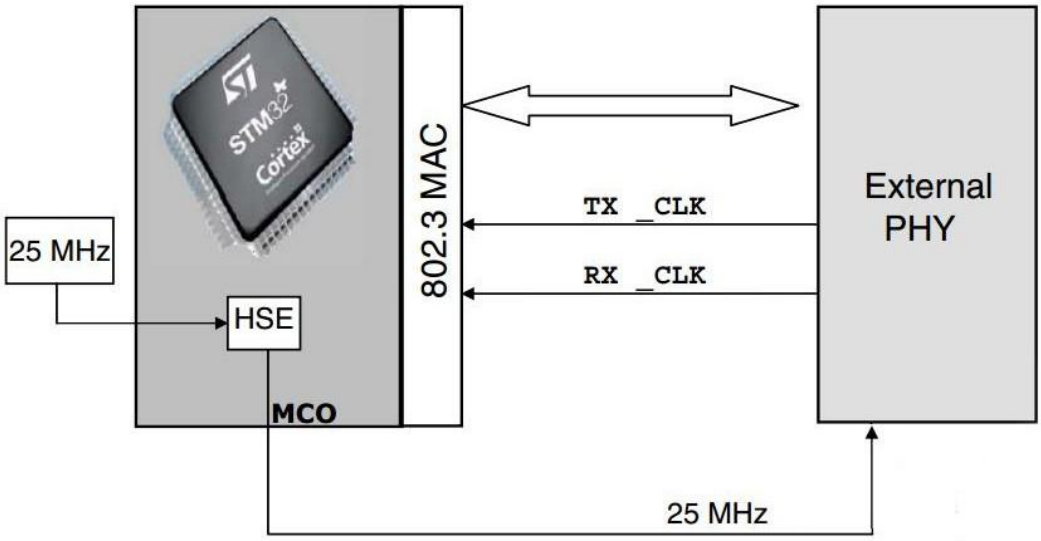
Cílem této kapitoly je popsat způsoby práce s časem získaným protokolem časové synchronizace PTP. Dále v textu bude označován jako přesný čas.

3.1 Frekvence procesoru a fyzické vrstvy

Na tomto modulu je fyzická vrstva Ethernetu (PHY) taktována externím zdrojem hodinového signálu (MCO), kterým je procesor. Díky možnosti konfigurace MCO výstupu v procesoru lze zvolit mezi dvěma různými zdroji požadované frekvence.

K propojení mikrokontroleru a fyzické vrstvy bylo použito klasické MII (Media Independent Interface) rozhraní, které vyžaduje frekvenci 25 MHz. Tuto frekvenci lze na MCO výstup přesměrovat přímo z krystalu (HSE), jehož frekvence je v případě použitého modulu právě 25 MHz. Toto řešení zobrazené na obr. 3.1 umožňuje nastavit libovolnou pracovní frekvenci procesoru v rozsahu 25-120 MHz.

Pokud by bylo potřeba uvolnit některé výstupy procesoru využívané rozhraním MII, mohlo by se použít RMII (Reduced Media Independent Interface) rozhraní, které ale vyžaduje frekvenci 50 MHz, což znamená, že by výstup MCO musel generovat frekvenci 50 MHz, které lze docílit dělením systémové frekvence jádra procesoru, což by znamenalo omezit ji na 100 MHz.



Obr. 3.1 Zdroj hodinového signálu fyzické vrstvy Ethernetu

3.2 Zdroje přesného času v procesoru

Procesor STM32F207 [8] je vybaven hardwarovou podporou protokolu časové synchronizace PTP, která se nachází v periferii Ethernetu. V této periferii se nachází tímto protokolem synchronizovaný čas, který se ale dá využít jen omezeně.

3.2.1 Časovač TIM2 - ALARM

První možností je použití událostí v předem nadefinovaný čas (Target Time Trigger). Tento způsob připomíná funkci budíku, a proto bude po vzoru PTP Trigger Boxu [13] dále v textu označovat jako „Alarm“. Na obr. 3.2 je znázorněno propojení Ethernetové periferie s blokem Alarmu. Událostí může být jak přerušení od Ethernetové periferie, tak hrana na výstupu z periferie (ne však z procesoru) ITR1, kterou lze vnitřně přesměrovat na vstup časovače TIM2 jako externí trigger. Použití přerušení od Ethernetu není vhodné, protože může kvůli obsluze příchozích paketů LwIP stackem trvat relativně dlouho a časově choulostivý požadavek na obsluhu by musel počkat na jeho dokončení.

Proto je vhodné využít časovače TIM2 a jeho přerušení, které může mít větší prioritu, a tak může být požadavek události o obsluhu vyřízen nezávisle na aktuálním stavu Ethernetové komunikace.

Takováto událost může být generována periodicky, což umožní vytvořit přesný zdroj frekvence. Maximální frekvence je však relativně nízká z důvodu nutnosti nastavovat každou následující událost programem. Tato frekvence je omezena na 100kHz, při které je procesor nastavováním dalších časů vytížen zhruba na 40%. Experimentálně bylo dosaženo frekvence 205KHz, při které již ale vzniká reálné riziko, že přestane korektně fungovat Ethernetová komunikace, jež má nižší prioritu.

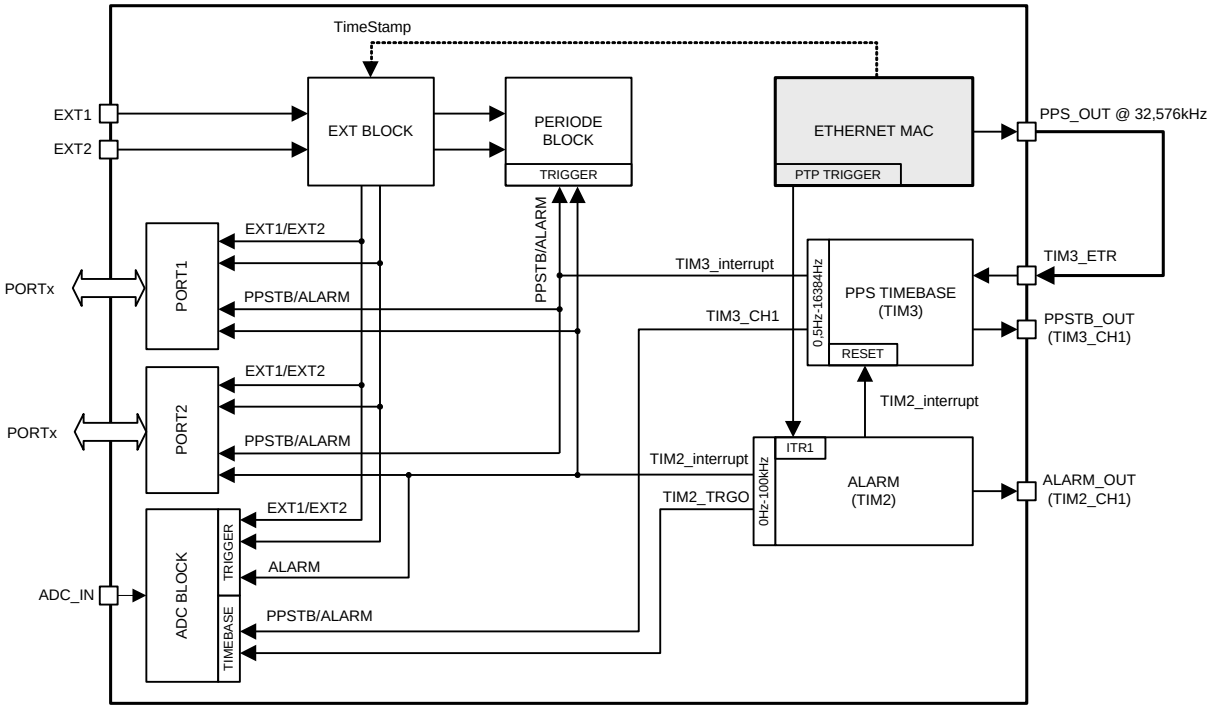
Díky použití časovače TIM2 je možné při každé události vygenerovat impuls na výstupu procesoru, který může být použit kromě testovacích účelů i k triggrování jiných zařízení. Aby bylo možno na jednu hranu generovat impuls, tak je časovač v režimu PWM2, kterému je programově udržovaná střída 25 % tak, aby byly na osciloskopu generované impulsy dobře viditelné. Výstup, kde je možné najít tyto impulzy, je na prvním kanálu časovače 2 (TIM2_CH1), konkrétně na výstupu PA5.

3.2.2 Časovač TIM3 – PPS TIMEBASE

Druhou možností je použít výstup poskytující přesnou frekvenci o periodě jedné sekundy PPS (Pulse Per Second), kterým je procesor vybaven zejména pro účely testování. Tento výstup je možné nastavit na generování frekvence až 32,576kHz. Tento zdroj přesné, průběžně synchronizované frekvence lze zpětně zavést do procesoru jako externí trigger časovače (TIMx_ETR). Časovač bude v tomto případě plnit funkci programovatelné děličky. Výstup tohoto časovače bude vzhledem k jeho původu a předpokládanému použití jako časové základny A/D převodníku dále nazývat „PPS Timebase“, nebo zkráceně „PPSTB“, který můžeme také vidět na obr. 3.2.

Tímto způsobem sice získáme přesnou frekvenci, ale abychom ji mohli použít ke generování synchronních událostí, tak je potřeba zajistit restartování časovače ve stejný čas. Jediným vhodným způsobem je použití Alarmu, který odstartuje časovače

ve všech modulech ve stejný okamžik. Výstup tohoto časovače můžeme sledovat na prvním kanálu časovače 3 (TIM3_CH1) konkrétně na výstupu PC6.



Obr. 3.2 Blokové schéma časové distribuce v procesoru

3.3 Externí trigger

Externí trigger je zdrojem událostí vyvolaných při detekci hrany na vstupu procesoru (EXT), která vyvolá přerušení (EXTI). Hned jak program skočí do přerušení, tak uloží časovou značku (TimeStamp) přesného času, provede předem nastavené úkony a přerušení ukončí. Vzhledem ke snaze o zachycení času, vždy se stejným a minimálním zpožděním od detekované hrany k samotnému uložení časové značky, má toto přerušení nejvyšší nastavitelnou prioritu.

Jak je vidět na obrázku obr. 3.2, tak se tento zdroj nevyužívá pouze k triggrování různých úkonů, ale i k měření periody. Toto měření využívá rozdíl časů získaných právě z uložených časových značek.

Vstupy, které slouží jako zdroj externího triggeru, jsou pouze dva. Důvodem je snaha minimalizovat pravděpodobnost kolize, kdy by obě dvě přerušení vyžadovala obsluhu ve stejný čas a jedno z nich by muselo počkat. Byly zvoleny

takové vstupy, u kterých lze nastavit různou prioritu a tudíž specifikovat, který je důležitější. Pokud by ale bylo potřeba více externích triggrovacích vstupů, tak není problém zvětšit jejich počet.

U těchto vstupů lze, vzhledem k podpoře procesoru, nastavit typ hrany, na kterou reaguje. Může reagovat na náběžnou hranu (POS), spádovou hranu (NEG), nebo na obě dvě (BOTH).

3.4 Ovládání binárních vstupů/výstupů

Vzhledem k potřebě zajistit jednoznačnou dobu vykonání dané operace s bránou procesoru, tak je vhodné pracovat vždy právě s jednou z nich, protože se s ní pracuje jako s 16bitovým registrem. K práci s bránou budou použity registry *GPIOx_ODR* a *GPIOx_IDR*.

Ne vždy však chceme zapsat data na celou bránu procesoru, a tudíž je nutné před zápisem nových dat do registru *GPIOx_ODR* data přečíst, maskovat podle předem nakonfigurované masky a následně zapsat zpět. Masky jsou potřeba dvě. Jedna fixní, která umožní práci pouze s přístupnými vstupně/výstupními linkami (*GPIO*) na modulu. Druhá maska je uživatelsky konfigurovatelná, a to proto, aby se na stejné bráně mohlo provádět více nezávislých operací.

Aby se dala daná úloha s Portem[1] jednoduše konfigurovat, tak je nutné specifikovat její možnosti. S Portem bude možné provést operaci čtení (READ), zápisu (WRITE) a bitové inverze (TOGGLE).

Dále je třeba specifikovat, kdy se má daná operace provést. Jak je vidět na obrázku obr. 3.2, tak můžeme volit ze všech tří výše zmíněných triggrovacích zdrojů. Každý z nich využívá přerušení o relativně vysoké prioritě tak, aby bylo docíleno vždy stejného zpoždění (latence) mezi generovanou událostí (zdrojem přerušení) a vykonáním dané operace.

Počet implementovaných Portů ovládající bránu procesoru je vzhledem k využití struktur programovacího jazyka C lehce rozšiřitelný. Pro ověření funkčnosti jsou implementovány Porty dva.

[1] Port je blok zajišťující práci s bránou procesoru

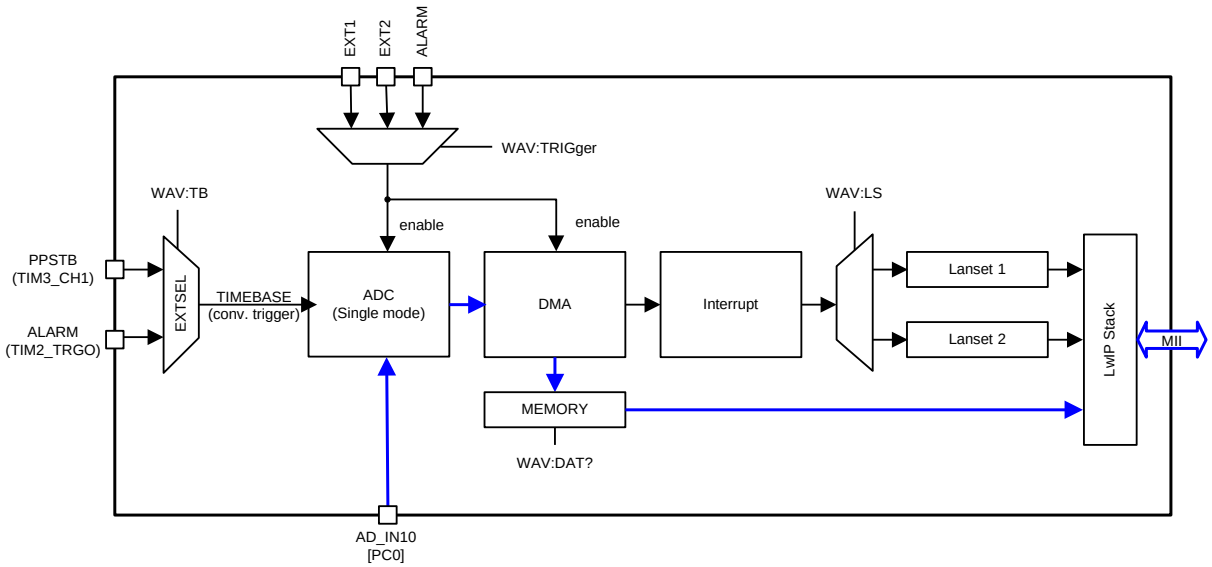
3.5 A/D převodník

Použitý procesor obsahuje integrovaný A/D převodník s rozsáhlými konfiguračními možnostmi. Při práci s přesným časem je nejdůležitější vlastností převodníku možnost triggrovat ho výstupem časovačů, díky čemuž můžeme jako časovou základnu použít výše zmíněný Alarm (TIM2_TRGO), nebo PPSTB (TIM3_CH1), což minimalizuje ztrátu kvality zdrojové frekvence.

Maximální rychlost vzorkování dána maximální rychlostí Alarmu, popsaném v kapitole 3.2.1, je 100 kSa/s. Při této vzorkovací frekvenci není vhodné po každém převodu vyvolávat přerušení, kopírovat naměřená data a zatěžovat tak procesor. Za tímto účelem je procesor vybaven DMA řadičem, který dokáže zkopírovat naměřená data z datového registru A/D převodníku na předem specifikované místo v paměti. DMA řadič po předem nastaveném počtu operací vyvolá přerušení, ve kterém může být odeslána asynchronní zpráva informující uživatele o dokončení A/D převodu.

Na obrázku obr. 3.3 můžeme vidět jak vstupy poskytující frekvenci pro vzorkování sloužící jako časová základna (*WAV:TB*), tak i vstupy zapínající proces celého převodu (*WAV:TRIG*).

Na obr. 3.3 je modrou barvou znázorněn průchod měřeného signálu celým systémem, který jde ze vstupu procesoru (PC0/AD_IN10) na vstup A/D převodníku (ADC3). Po dokončení každého převodu zkopíruje DMA řadič nově naměřená data do bufferu v paměti SRAM. Po dokončení posledního převodu řadič DMA vyvolá přerušení, kde se vypne ADC i DMA, změní se stavová informace o převodu a v případě nastavení asynchronní zprávy se odešle zpráva o dokončeném převodu.



Obr. 3.3 Schematické uspořádání A/D převodníku

3.5.1 Odesílání naměřených dat

Kdykoli Ethernetový modulu přijme dotaz na naměřená data ve formě SCPI příkazu *WAV:DAT?* a není právě vykonáván A/D převod, tak se dotazovanému odešlou data. První bajt v odesílaném řetězci obsahuje číslo 19, které v ASCII tabulce spadá do tzv. přístrojových příkazů (device control). Tento bajt informuje příjemce (aplikaci na PC), že jde o data z A/D převodníku. Dalších 5 bajtů v řetězci obsahuje číslo v ASCII formátu, které určuje počet bajtů následujících v tomto řetězci, obsahujících odměřená data.

4 Odesílání asynchronních zpráv

Velice zajímavým způsobem vyřešila firma Agilent zasílání asynchronních zpráv. Asynchronní zprávy jsou všechny zprávy, které přijdou bez předchozího dotazu. Funkce zajištující zasílání těchto zpráv se nazývá „Lanset“ a s mírnými modifikacemi je buď použita k odesílání naměřených výsledků, anebo časových značek. Na obrázku obr. 4.1 jsou znázorněny možné zdroje asynchronních zpráv.

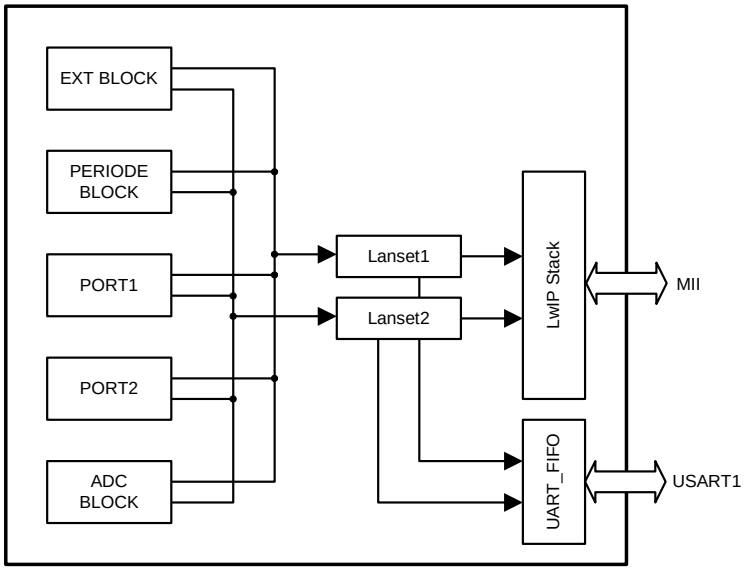
Pro účely demonstrace jsou Lansety implemetovány pouze dva, ale v případě potřeby je možné jejich počet navýšit. Asynchronní zprávy využívají stejně jako konfigurační příkazy TCP transportní vrstvu a port 5025. Důvodem je, že ve většině případů je zasílaná asynchronní zpráva klientovi, který má otevřené konfigurační spojení, a tudíž není potřeba vytvářet nové spojení. Pokud však při konfiguraci Lansetu nemá cílový klient aktivní spojení, vytvoří se nové, což ale vyžaduje, aby na straně příjemce běžel TCP server. V rámci vývoje bylo vyzkoušeno a ověřeno navázání spojení s jiným modulem, což je základem distribuovaného systému nezávislého na řídící jednotce (PC).

Lanset umožňuje odeslat zprávu všem, ale kvůli použité TCP transportní vrstvě se nejedná o broadcast nebo multicast, ale nezávisle odesílané pakety. V okamžiku odesílání zprávy se zpráva odešle všem aktivním klientům, kterými jsou zpravidla PC, ze kterých se distribuovaný systém ovládá, nebo jen monitoruje.

Klíčovou vlastností Lansetu je zasílání tzv. prefixů, které předcházejí každé zprávě. Může to být textový řetězec o délce až 16 bajtů, nebo libovolná jednobajtová hodnota, kterou může být třeba ACII symbol spadající do přístrojových příkazů (device control), které se používají v ovládací aplikaci na PC k určení typu zprávy.

Další vlastností Lansetu je možnost za prefix připojit data, kterými může být například čas detekce externího trigger-u EXT, změřená perioda nebo binární hodnota přečtená z portu procesoru.

Velice praktickou asynchronní zprávou je periodické odesílání stavu časové synchronizace, která může být v ovládací aplikaci na PC přehledně zobrazena.



Obr. 4.1 Schéma zdrojů asynchronních zpráv

5 Struktura SCPI příkazů

Struktura SCPI příkazů celkem přesně odpovídá výše popsaným možnostem Ethernetového modulu.

Z takzvaných povinných obecných příkazů jsou implementovány příkazy **IDN?* a **RST*. Dotaz **IDN?* se zařízení dotáže na jeho identifikační údaje, kterými je v tomto případě typ zařízení a verze firmware. Příkaz **RST* restartuje procesor modulu. Po přijetí tohoto příkazu je procesor restartován stejným způsobem, jako když uživatel stiskne restartovací tlačítko na modulu.

Některé příkazy lze zapsat dvěma způsoby, zkráceně, kdy se použijí pouze veliká písmena, anebo nezkráceně, kdy se použije celý příkaz. Například pokud chceme použít příkaz *WAVeform:DAT?*, tak můžeme volit mezi *WAV:DAT?* a *WAVEFORM:DAT?*.

Na obrázku obr. 5.1 je vidět příklad použití jednoho z příkazů, s využitím programu PuTTY. Tento obrázek zároveň ukazuje možnost komunikace s modulem bez použití ovládací aplikace.

Seznam implementovaných příkazů spolu s jejich popisem je k nalezení v příloze C Implementované SCPI příkazy.

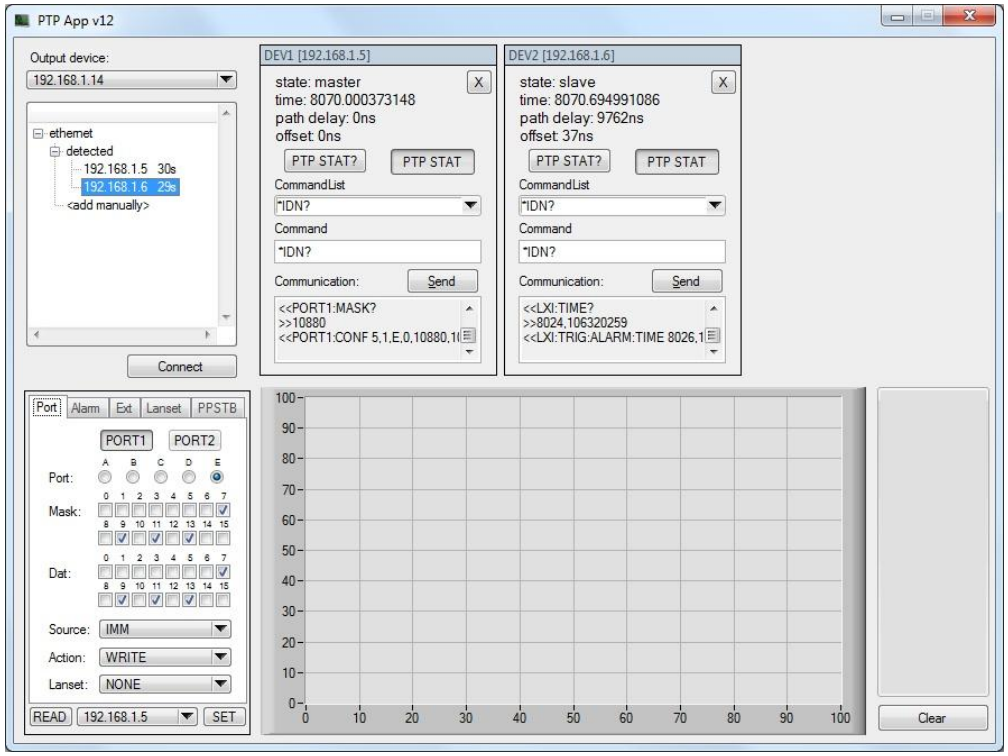
Obr. 5.1 Ovládání modulu programem PuTTY

6 Ovládací aplikace na PC

Ovládací aplikace na PC je napsaná v jazyce C, ve vývojovém prostředí LabWindows CVI 2010. Úkolem této aplikace je přehledně a rychle ovládat distribuovaný systém složený z více modulů.

Aplikace zobrazena na obr. 6.1 je od počátku strukturovaná tak, aby ji bylo možno později bez problému rozšířit o další komunikační rozhraní. Vzhledem k tomu, že Ethernetový modul podporuje UART (sériová komunikace) a USB (Universal Serial Bus) rozhraní, tak je vhodné, aby i ovládací aplikace tato rozhraní podporovala.

Aplikace je vícevláknová, a to především kvůli tomu, aby v případě vytížení procesoru Ethernetovou komunikací nebylo hlavní vlákno obsluhující grafické rozhraní ovlivněno. Aplikace je striktně rozdělena na dvě vlákna. První je tzv. hlavní, ve kterém běží grafické rozhraní. Druhé vlákno spravuje veškerou komunikaci a zpracovává příchozí zprávy.



Obr. 6.1 Ovládací aplikace na PC

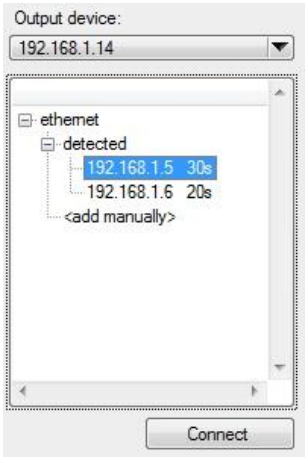
6.1 Připojení k modulu

První věcí, kterou bude muset uživatel po spuštění programu udělat, je připojení k nějakému modulu v distribuovaném systému. Problém ale nastává v okamžiku, kdy bude uživatel vyzván k zadání IP adresy, kterou nemusí vždy znát. Z tohoto důvodu by zařízení mělo podporovat nějaký způsob nalezení takovéhoto modulu.

Všechny laboratorní přístroje vybavené Ethernetovým rozhraním (LXI zařízení) by měly implementovat metodu autodetekce. Podle desáté kapitoly LXI specifikace [3] existují dva způsoby autodetekce. Prvním způsobem je využití dnes již starého, avšak výrobci stále podporovaného protokolu VXI-11(VMEbus Extension for Instrumentation). Druhým způsobem je využití mDNS (Multicast DNS) protokolu [14].

V případě zařízení pracujícího s protokolem PTP, je k autodetekci možné použít multicastové spojení, na kterém probíhá komunikace zajišťující PTP synchronizaci.

K zahájení naslouchání je potřeba určit, na které síti se bude naslouchat, což je potřeba řešit pouze v případě, že je na daném PC více síťových karet, a tím pádem více lokálních adres. Dále je potřeba znát multicastovou adresu, která je pro PTP *224.0.1.129* a v poslední řadě port. PTP používá zprávy dvou kategorií tzv. *Event* a *General*. V tomto případě je potřeba zvolit *Event*, které náleží port 319, na němž jsou posílány pakety nejen od *master*, ale i od *slave* modulů.



Obr. 6.2 Připojení k modulům

Po úspěšné registraci do multicastu začnou přicházet pakety, ze kterých se získá zdrojová IP adresa. Při každém příchozím paketu se obnoví seznam detekovaných adres spolu se zbývajícím časem. Tento čas informuje uživatele o činnosti modulu a po uplynutí 30sekundového intervalu bez přijetí jediného synchronizačního paketu z dané IP adresy se ze seznamu adres adresa odebere.

Připojení k modulu je programu věnovaná část zobrazená na obr. 6.2, která umožňuje volbu síťového adaptéru „Output device“, zvolení nalezené adresy, nebo zadané adresy a následné připojení.

6.2 Relace modulu

Aplikace je strukturována tak, že nezáleží na počtu připojených modulů. Každé aktivní spojení má vlastní ovládací panel, který obsahuje čtyři části obr. 6.3.

První část je věnovaná zobrazení stavu časové synchronizace, která obsahuje kromě místa pro přijatou stavovou zprávu dvě ovládací tlačítka, jedno odesílá dotaz na aktuální stav (*LXI:TIME:STAT?*), druhé zapíná/vypíná periodické zasílání asynchronních stavových zpráv (*LXI:TIME:STAT:AUTO ON/OFF*).

Druhá část nabízí seznam se všemi implementovanými příkazy, který má za účel usnadnit psaní příkazů a dotazů.

Obr. 6.3 Panel modulu

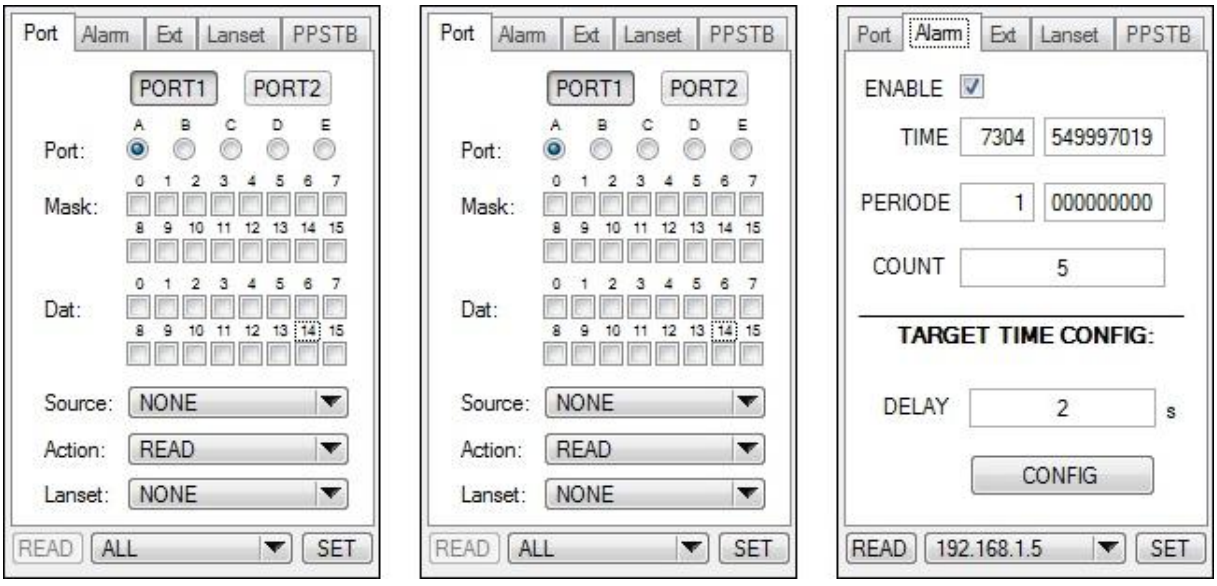
Další část obsahuje textové pole, do kterého uživatel píše zprávu, kterou chce odeslat. Samotné odeslání zprávy se provede stiskem klávesy Enter, anebo stisknutím tlačítka „Send“.

Poslední část pouze zaznamenává komunikaci mezi daným modulem a počítačem.

6.3 Konfigurátor

Konfigurátor je část programu, která má za úkol rychle a přehledně nakonfigurovat distribuovaný systém. Na obrázku obr. 6.4 je vidět, že je rozdělen do záložek odpovídajících jednotlivým funkcím.

V dolní části jsou tři ovládací prvky, které jsou pro všechny záložky společné. Pokud je zvolena konkrétní IP adresa, tak je povoleno stisknout tlačítko „READ“, které vyčte nastavení zařízení s danou adresou specifické pro aktuální záložku. Následně lze provést úpravy, zvolit „ALL“ a stisknout „SET“, což nastaví všechny moduly, ke kterým je program připojen.



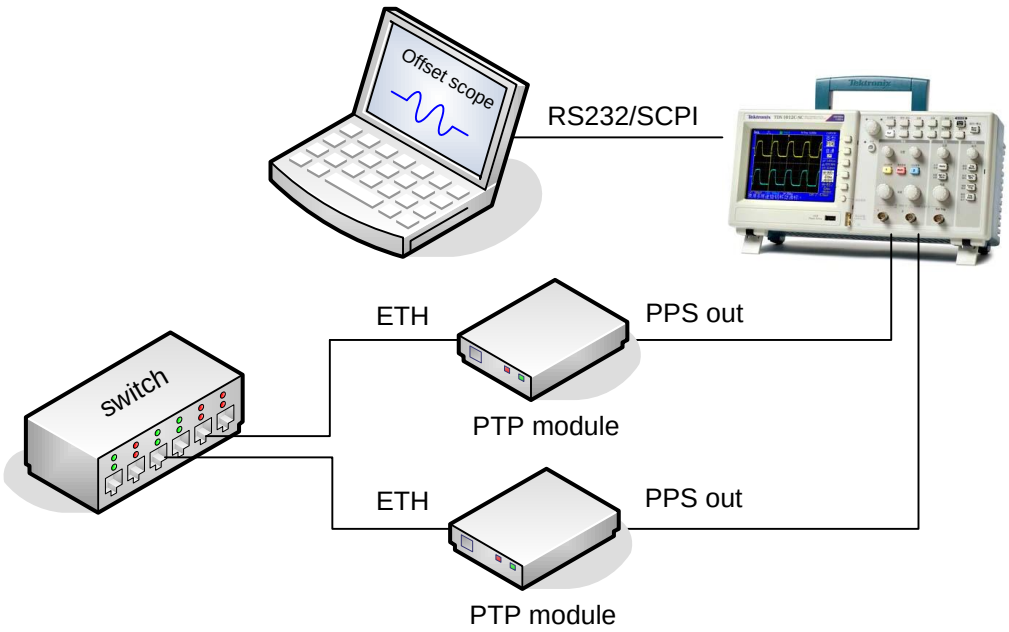
Obr. 6.4 Panel Konfigurátoru

Mimo základního nastavení dokáže Konfigurátor vykonávat i jednoduché operace. Jednou z nich je třeba funkce „Target Time Config“ v kartě „Alarm“, která se zeptá všech připojených modulů na jejich PTP čas a stav PTP synchronizace. Z těchto časů vyloučí časy, které patří modulům, které nejsou synchronizované, přičte zadané zpoždění (delay), které je přednastaveno na 2 s a nastaví všem připojeným modulům TargetTime, což je čas, ve který má Alarm vygenerovat událost. Tato funkce ušetří hodně času, především u většího počtu modulů.

7 Měření kvality časové synchronizace

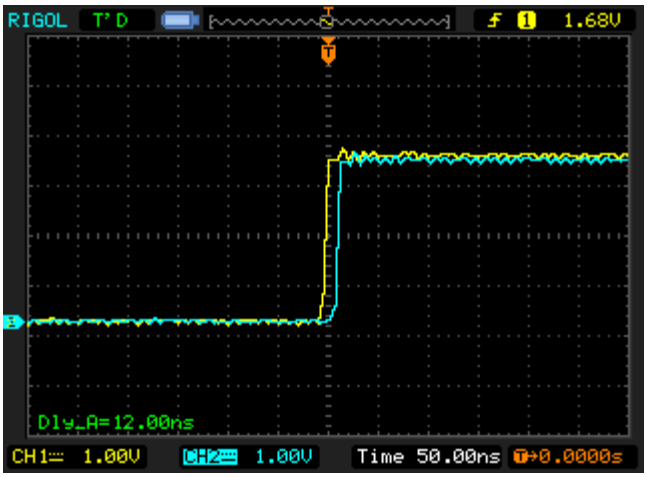
Pro měření časové synchronizace je mikrokontroler vybaven výstupem PPS (Pulse Per Second), který v základním nastavení generuje frekvenci o periodě jedné sekundy. Jak bylo zmíněno výše (3.2.2), tak je frekvence výstupu PPS nenastavena na 32,768 kHz.

Pokud jsou dostupné alespoň dva Ethernetové moduly a digitální osciloskop, tak lze zapojit měření podle obrázku obr. 7.1. Pokud jsou moduly dva, tak není potřeba použít síťový přepínač (switch) a lze moduly propojit přímo. Pokud jsou moduly připojeny přímo bez použití switche, tak dosáhneme lepších výsledků (obr. 7.5). Jak je vidět na obrázku obr. 7.2, tak je osciloskop nastaven na měření zpoždění dvou náběžných hran.



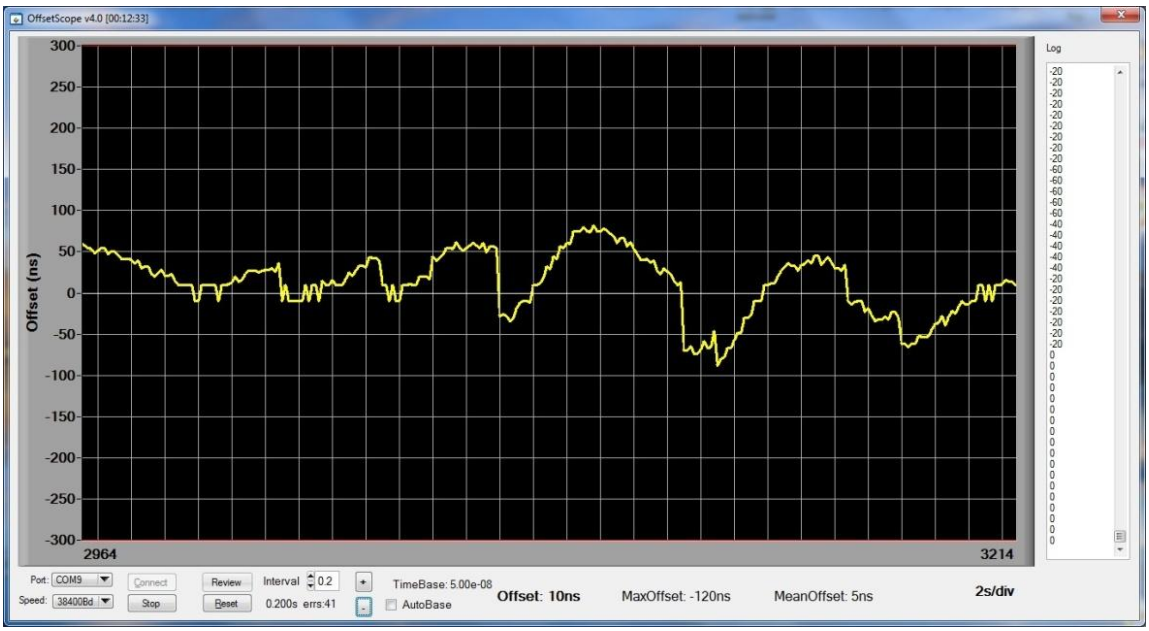
Obr. 7.1 Měření kvality časové synchronizace distribuovaného systému

Naměřená data je však vhodné nějakým způsobem dále zpracovat, aby se dala určit kvalita synchronizace. Za tímto účelem byla vytvořena aplikace na PC, která pomocí sériové linky (RS232) a příkazů SCPI automatizuje měření a získané výsledky ukládá do souboru a vykresluje do grafu. Aplikace provádějící měření je zachycena na obrázku obr. 7.3.



Obr. 7.2 Obrazovka osciloskopu při měření synchronizace

Problémem takovéhoto měření je nutnost zvolit fixní časovou základnu s předpokládaným maximálním offsetem. Pro účely měření byla zvolena časová základna 50 ns/dílek, což na použitém osciloskopu poskytlo rozsah +/- 300 ns.

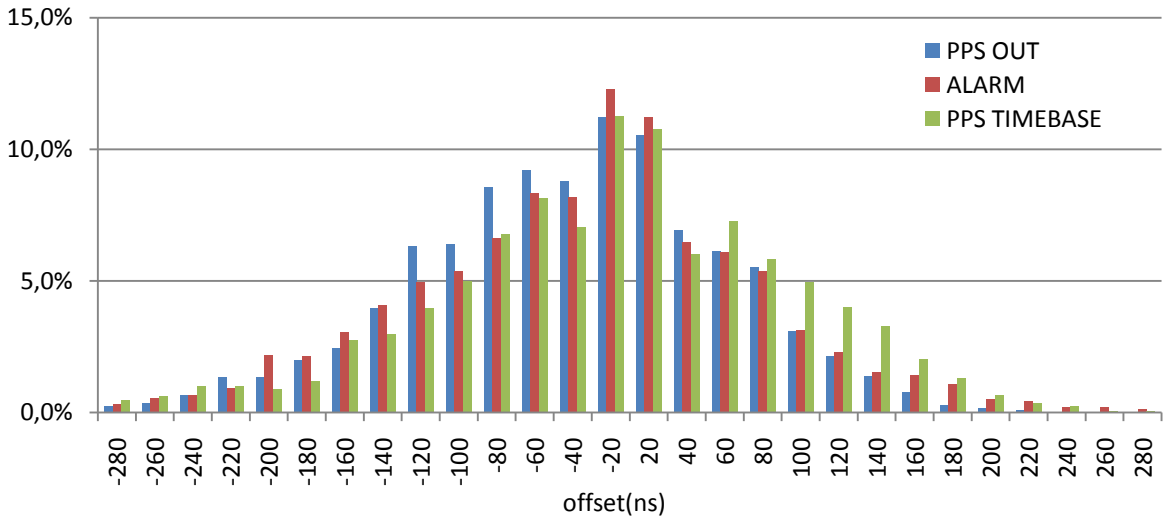


Obr. 7.3 Aplikace zachytávající offset na PC

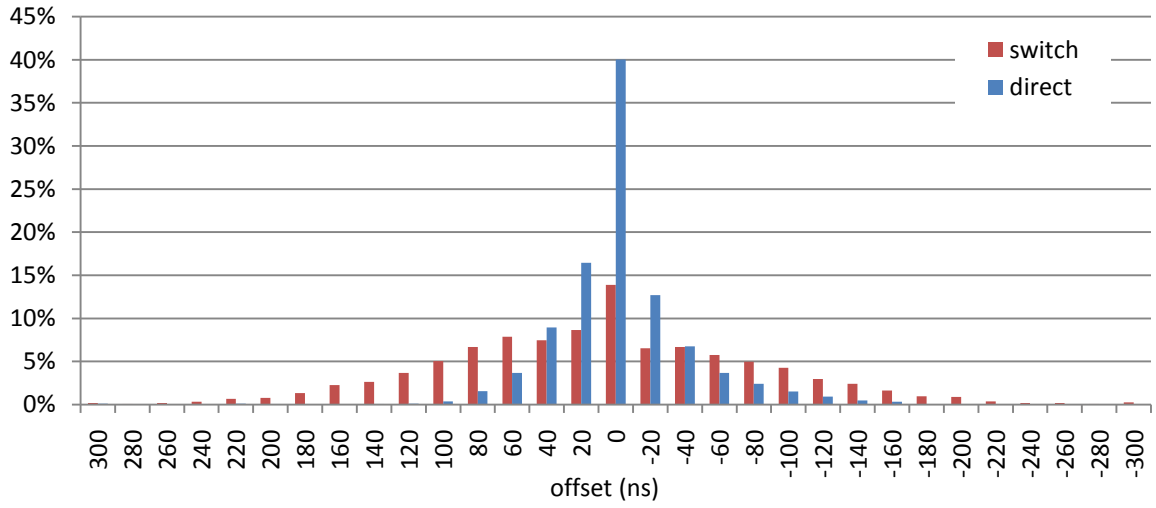
Na použitém osciloskopu bylo možné provádět měření každých 200ms, což je vzhledem k sekundovému intervalu synchronizace PTP protokolu zcela dostačující. Měřící aplikace umožňuje automatickou změnu rozsahu, kterou lze však použít pouze v případě pomalého měření. Důvodem je, že po změně časové základy nemá osciloskop nějakou dobu naměřená data. Pokud na ně v tomto okamžiku přijde

dotaz, tak je vrácena hodnota z konce nového rozsahu, což aplikace na PC vyhodnotí jako potřebu přepnout na vyšší rozsah.

Graf na obrázku obr. 7.4 je výsledkem měření ověřujícího minimální degradaci času použitím časovačů, na kterém je důležitý minimální rozdíl mezi měřením na výstupu časovačů (Alarm a PPSTB) a na výstupu PPS OUT. Data byla měřena výše popsaným způsobem, nejdříve na PPS výstupu a následně na výstupech časovačů TIM2(Alarm) a TIM3 (PPSTB). Výsledkem každého měření bylo 4 000 hodnot odečítaných v 300ms intervalu. Je třeba zdůraznit, že každý modul měl jako zdroj frekvence obyčejný krystalový oscilátor, který se se svou přesností pohybuje kolem 30 ppm (parts per million).



Obr. 7.4 Vliv použitých metod na kvalitu časové synchronizace (zapojení se swithcem)



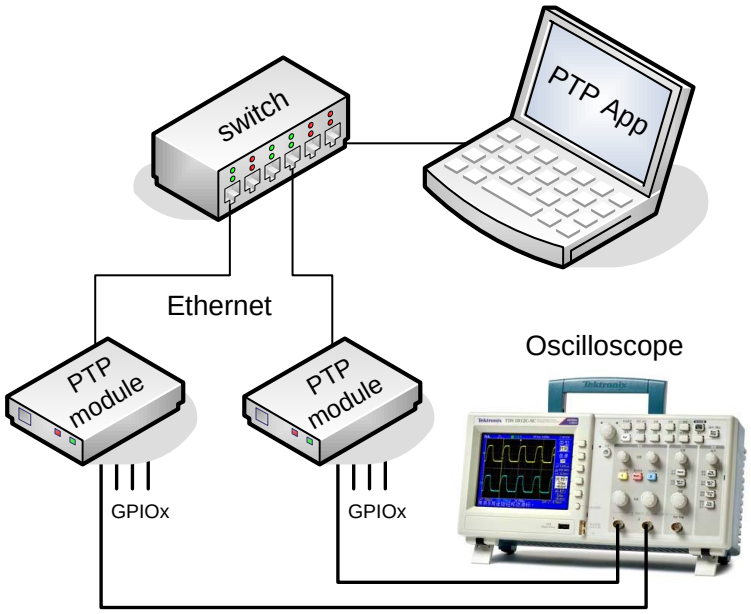
Obr. 7.5 Vliv použití switche na kvalitu časové synchronizace

8 Demonstrace časové synchronizace

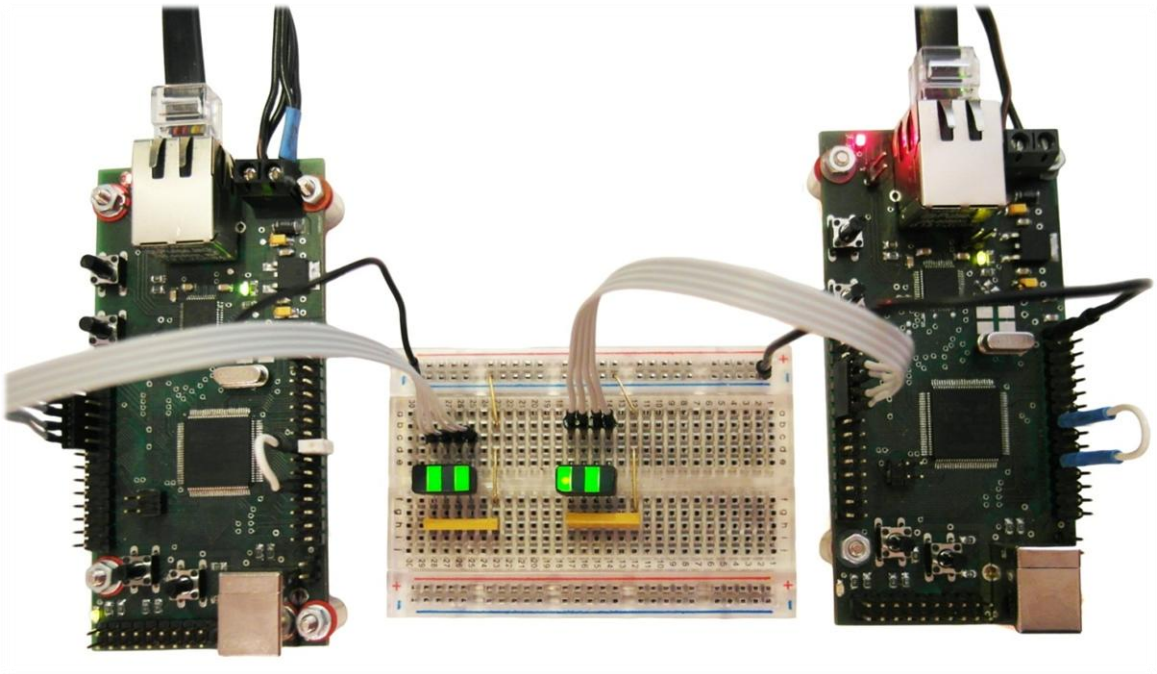
Pro ověření funkčnosti vytvořeného programového vybavení byly vytvořeny dvě základní demonstrace. První synchronně ovládá binární výstupy a druhá synchronně měří analogové signály.

8.1 Ovládání binárních výstupů

Jednou z možností ovládání binárních výstupů je použití Konfigurátoru v aplikaci na PC (kapitola 6.3), který můžeme vidět na obrázku obr. 6.4. Na ovládané výstupy můžeme pro ověření funkčnosti připojit osciloskop, tak jak je znázorněno na obrázku obr. 8.1. Pro základní ověření funkčnosti bez informace o kvalitě synchronizace mezi moduly můžeme na výstupy připojit LED diody, tak jak je zachyceno na obrázku obr. 8.2.



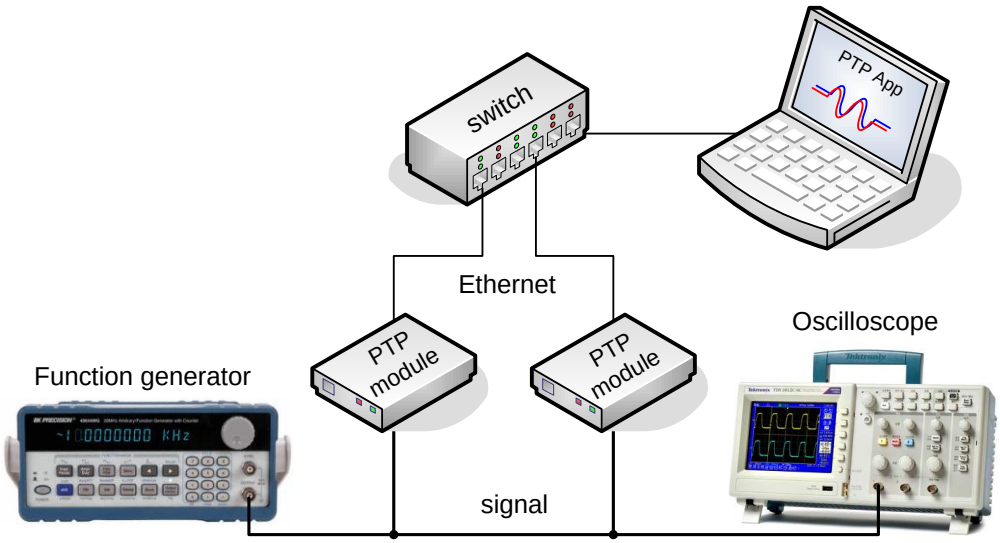
Obr. 8.1 Blokové schéma ovládání binárních výsupů



Obr. 8.2 Fotografie demonstrující ovládání binárních výstupů

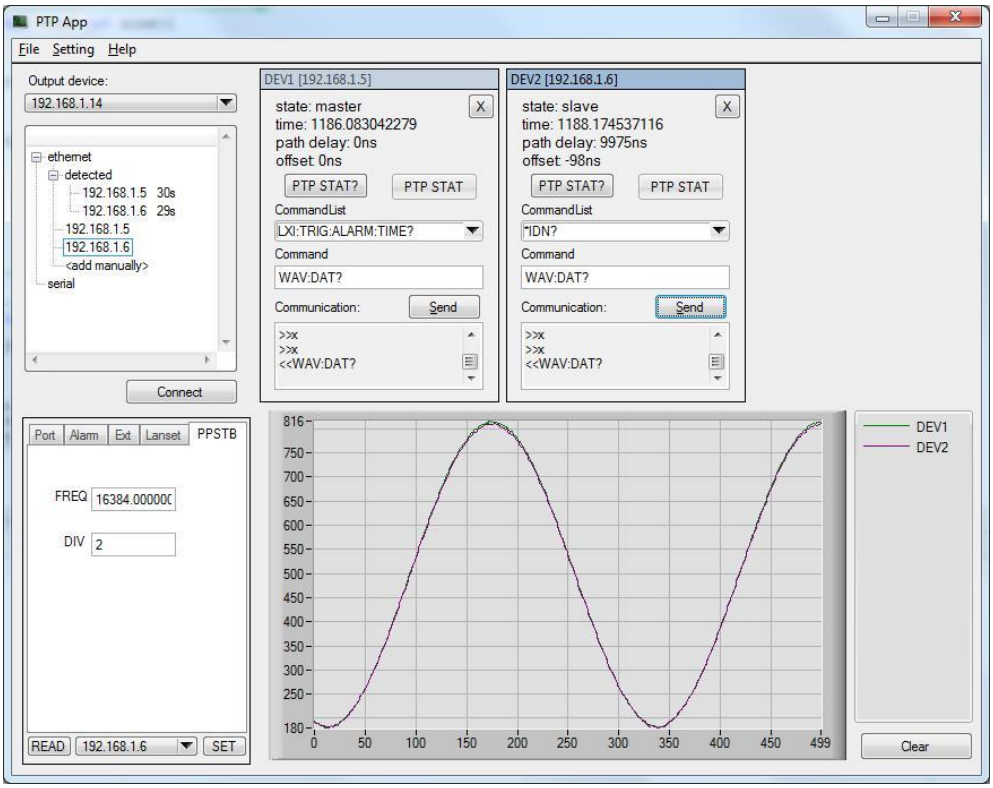
8.2 Měření analogových signálů

Druhou demonstrací je měření analogového signálu dvěma moduly, tak jak je znázorněno na obrázku obr. 8.3. Osciloskop zde slouží pouze pro ověření správného nastavení offsetu a amplitudy funkčního generátoru. Pokud totiž zapojujeme měřený signál přímo na modul, tak se musí generovaný signál pohybovat mezi 0V a 3,3V.



Obr. 8.3 Blokové schéma měření analogových signálů

Výsledek tohoto měření je zobrazen v ovládací aplikaci obr. 8.4. Z tohoto obrázku lze vyčíst, že byla nastavena časová základna měření na 16384Hz. Dále je vidět že moduly odesílaly znak "x", který byl asynchronní zprávou. Tato zpráva byla odesílána vždy po dokončení měření. Následně byl odeslán požadavek na odeslání naměřených dat, které byly po přijetí vykresleny do společného grafu.



Obr. 8.4 Výsledek měření analogových signálů v aplikaci na PC

9 Závěr

Hlavním cílem této práce bylo navrhnout programové vybavení distribuovaného systému využívajícího časové synchronizace PTP IEEE-1588. Pro základní účely demonstrace má být tento systém schopen synchronně ovládat binární vstupy/výstupy, které má být možno nastavovat z nadřazeného počítače k tomu vytvořenou aplikací.

Jedním z prvních kroků při řešení této práce bylo osazení a oživení dvou Ethernetových modulů, na kterých se následně vyvíjelo požadované programové vybavení. Po seznámení s procesorem ARM Cortex STM32F207 [8], použitým na Ethernetovém modulu distribuovaného systému, byla navržena metoda využívající synchronizovaný čas pomocí časovačů procesoru (3.2). Následně byla vytvořena knihovna pracující s externím triggerem (3.3), knihovna ovládající binární výstupy (3.4) a navržena metoda, která využívá přesného času k vzorkování integrovaného A/D převodníku (3.5).

Pro možnost komunikace nadřazeného systému (PC) s Ethernetovým modulem bylo zvoleno použití SCPI příkazů (2.4 a 5). Použité příkazy byly většinou převzaty od již existujících zařízení, jako je osciloskop, stolní multimeter, anebo PTP Trigger Box firmy Agilent [13], od kterého byla převzata metoda odesílání asynchronních zpráv (4).

Při návrhu ovládací aplikace na PC bylo snahou vytvořit takový program, který umožní rychlé a přehledné nastavení distribuovaného sytému (6). Aplikace byla napsána v programovacím jazyce C, ve vývojovém prostředí LabWindows, která je strukturována tak, aby nezáleželo na počtu připojených modulů. Aplikace obsahuje sekci schopnou přehledně zobrazit a nastavit jednotlivé funkce modulu (6.3).

Závěrem této práce bylo měření kvality časové synchronizace, za jehož účelem byla navržena aplikace na PC, která měření automatizuje (7). Výsledky těchto měření ukázaly, že použité metody zvětšily odchylku od synchronizovaného času minimálně.

10 Literatura

[1] **IEEE Std. 1588.** Standard for a Precission Clock Synchronization Protocol for Networked Measurement and Control Systems. 2008.

[2] **SCPI.** Standard Commands for Programmable Instruments. *http://www.ivifoundation.org.* [Online] 1999. [Citace: 25. 5 2012.]

[3] **LXI Consorcium.** *LXI Device Specification.* 2011. Revision 1.4.

[4] **Swedish Institute of Computer Science.** Lightweight TCP/IP stack. *http://savannah.nongnu.org/projects/lwip/.* [Online] [Citace: 24. 5 2012.]

[5] **ST Microelectronics.** *Application note AN3384.* Doc ID 018698 Rev 2.

[6] **BAŘTIPÁN, Adam.** *Bakalářská práce - Moduly s rozhraním Ethernet pro systém sběru dat.* 2011.

[7] **J., Yiu.** *The definitive Guide to the ARM Cortex-M3. Elsevier.* 2007.

[8] **ST Microelectronics.** *STM32F207xx Datasheet.* Doc ID 15818 Rev7.

[9] **ST Microelectronics**. *Application note AN3427.* Doc ID 019001 Rev 1.

[10] **ST Microelectronics**. *RM0033 - STM32F207xx Reference Manual.* Doc ID15403 Rev 4.

[11] **ST Microelectronics**. Application note AN3411. Doc ID 018905 Rev 1.

[12] **SARNOVSKÝ, Michal.** SCPI Command processor. v 1.00.

[13] **Agilent.** E5818A LXI Class-B Trigger Box. *Programming manual.*

[14] **Multicast DNS.** *http://www.multicastdns.org/.* [Online] [Citace: 25. 5 2012.]

Seznam použitých symbolů a zkratek

PC	Personal Computer
MCU	Microcontroller Unit
ARM	Advanced RISC Machine
SRAM	Static Random Access Memory
IP	Internet Prtocol
TCP	Transmission Control Protocol
UDP	User Data Protocol
DNS	Domain Name System
mDNS	Multicast DNS
LwIP	Lightweight TCP/IP stack
PPM	Parts Per Million
RISC	Reduced Instruction Set Computer
PHY	Ethernet Physical Transciever
MAC	Media Access Control
MII	Media Independent Interface
RMII	Reduced Media Independent Interface
MCO	Microcontroller Clock Output
HSE	High Speed External
PLL	Phase Lock Loop
USB	Universal Serial Bus
UART	Universal synchronous/asynchronous receiver/transmitter
TIM	Timer
ADC	Analog to Digital Converter
DMA	Direct Memory Access
NVIC	Nested Vector Interrupt Controller
EXTI	External Interrupt
ETR	External Trigger
GPIO	General Purpose I/O
IEEE	Institute of Electrical and Electronics Engineers
NTP	Network Time Protocol
PTP	Precision Time Protocol
LXI	LAN eXtensions for Instrumentation
VXI	VMEbus Extension for Instrumentation
ASCII	American Standard Code for Information Interchange
SCPI	Standard Commands for Programmable Instruments
PPS	Pulse Per Second
PPSTB	PPS Timebase

Přílohy

A Získané zkušenosti při vývoji

A.1 Funkce printf

Pokud nám přestane fungovat program takovým způsobem, že se nevykoná instrukce skoku ze *startup* souboru do funkce *main*, problém může být ve funkci pritf. Pokud je totiž někde v programu použita tato funkce a není řádně přesměrována, tak se chyba může projevovat tímto způsobem.

A.2 Vyčítání PTP času z periferie Ethernetu

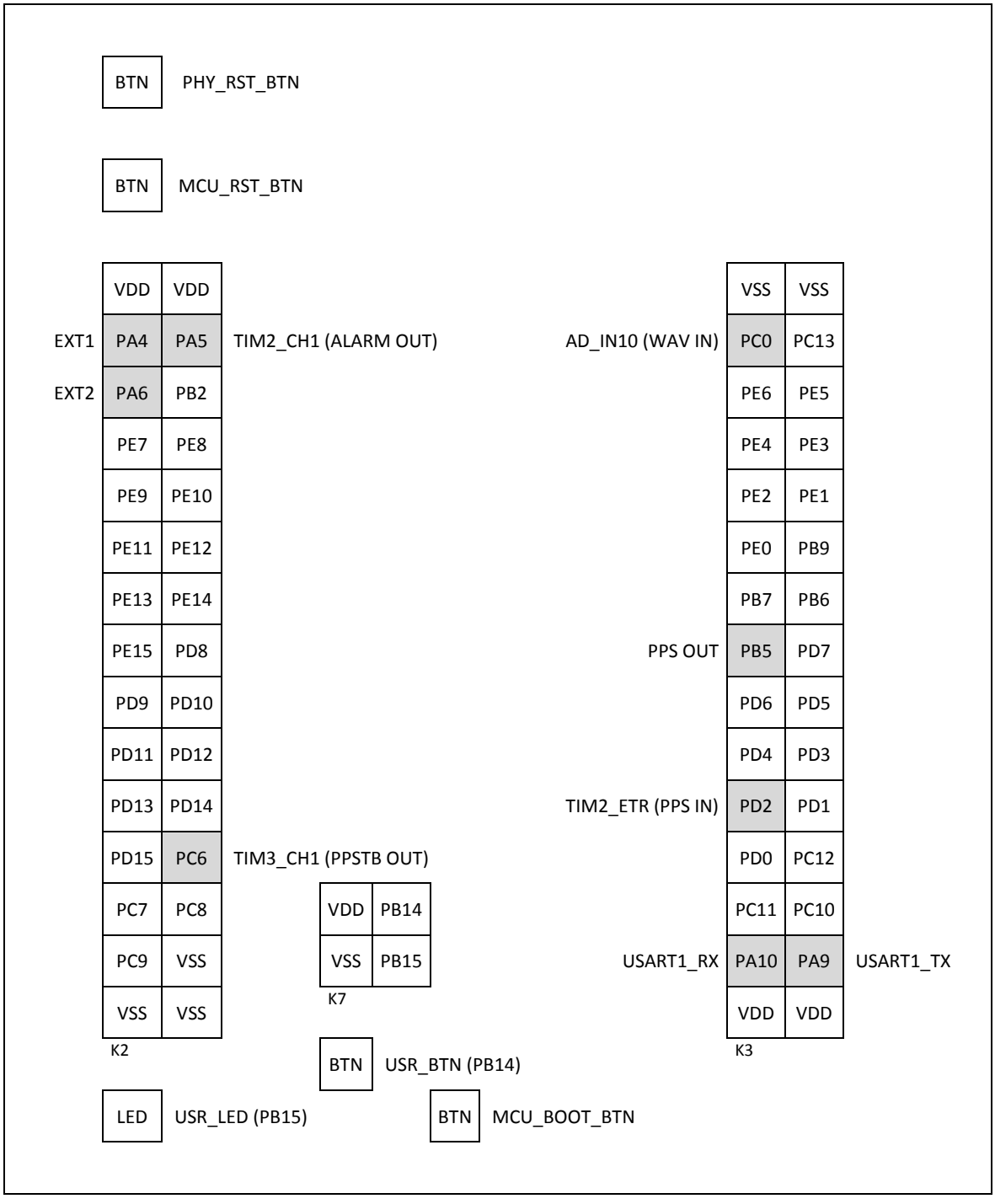
Pokud není zaručeno vyčítání PTP časových registrů v nejvyšší prioritě, tak je vhodné číst tyto registry dvakrát. Stačí přečíst subsekundový, sekundový a pak zase subsekundový registr. Pokud je subsekundová hodnota při druhém čtení menší než při prvním, je třeba přičíst sekundu k hodnotě ze sekundového registru.

A.3 Boot strap pull-up rezistory PHY

Pokud fyzická vrstva Ethernetu neustále přepíná mezi 10Mbit/s a 100Mbit/s režimy, řešením může být přidání pull-up rezistorů. Při bootování fyzické vrstvy se totiž provádí její nastavení v závislosti na logické úrovni na tzv. *„strap“* pinech. Pokud připojíme paralelně k LED diodám pull-up rezistory 2,2kΩ, mělo by nestandardní chování přestat. Dvě diody jsou integrované v Ethernetovém konektoru, třetí je umístěna vedle něj.

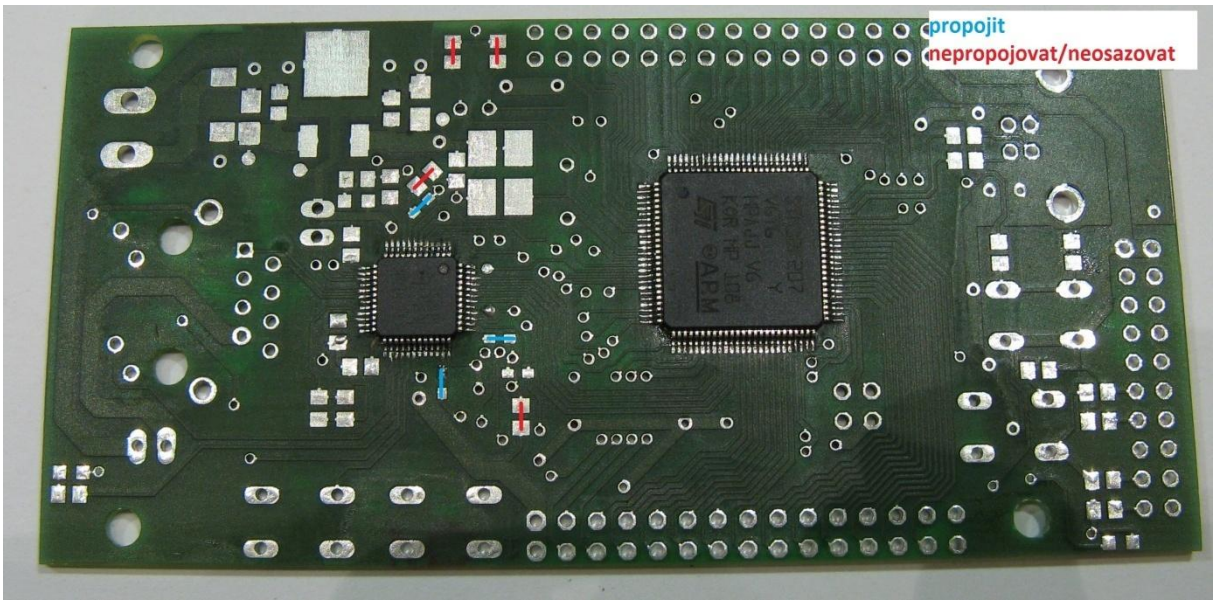
B Dodatkové materiály k modulu

B.1 Použité výstupy modulu

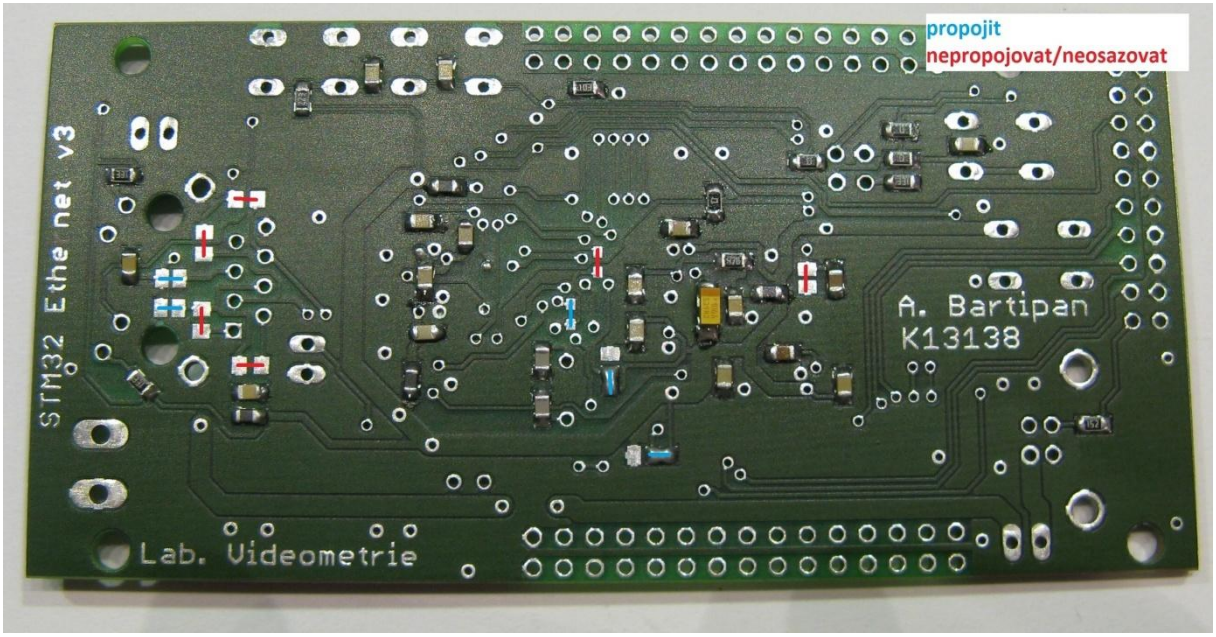


B.1 Pájecí propojky

Níže uvedené obrázky ukazují jaké pájecí propojky propojit, pokud je osazován procesor STM32F207 komunikující s fyzickou vrstvou Ethernetu MII rozhraním.



Obr. B.1.1 Pájecí propojky – TOP



Obr. B.1.2 Pájecí propojky – BOTTOM

C Implementované SCPI příkazy

C.1 Seznam implementovaných SCPI příkazů

*IDN?
*RST
SYSTem:NAME?

MEASure:PERiod1|2:CONFigure <trig>,<ls>
MEASure:PERiod1|2:TRIG?
MEASure:PERiod1|2:TRIG <0(IMM)|1(ALARM)|2(PPSTB)>
MEASure:PERiod1|2:DONE?
MEASure:PERiod1|2:PER?
MEASure:PERiod1|2:STOP

WAVeform:CONFigure <trig>,<tb>,<poin>,<ls>
WAVeform:TRIGger?
WAVeform:TRIGger <0(NONE)|1(IMM)|2(EXT1)|1(EXT2)|2(ALARM)>
WAVeform:TB?
WAVeform:TB <0(ALARM)|1(PPSTB)>
WAVeform:POINts?
WAVeform:POINts <value>
WAVeform:LS?
WAVeform:DAT?
WAVeform:DONE?
WAVeform:STOP

PORT1|2:CONFigure <trig>,<opp>,<port>,<ls>,<dat>,<mask>
PORT1|2:TRIG?
PORT1|2:TRIG
<0(NONE)|1(IMM)|2(EXT1)|3(EXT2)|4(ALARM)|5(PPSTB)>
PORT1|2:OPP?
PORT1|2:OPP <0(READ)|1(WRITE)|2(TOGGLE)>
PORT1|2:DAT?
PORT1|2:DAT <dat>
PORT1|2:MASK?
PORT1|2:MASK <mask>

LXI:TIME?
LXI:TIME:OFFset?
LXI:TIME:SYNC?
LXI:TIME:MASTER?
LXI:TIME:SLAVE?
LXI:TIME:STATus?
LXI:TIME:STATus:AUTO <0(DISABLE)|1(ENABLE)>

LXI:TRIGger:ALARM:CONFigure
<start_sec>,<start_nsec>,<per_sec>,<per_nsec>,<count>
LXI:TRIGger:ALARM:ENABle?
LXI:TRIGger:ALARM:ENABle <0(DISABLE)|1(ENABLE)>
LXI:TRIGger:ALARM:COUNt?
LXI:TRIGger:ALARM:COUNt <count>
LXI:TRIGger:ALARM:PERiod?
LXI:TRIGger:ALARM:PERiod <per_sec>,<per_nsec>
LXI:TRIGger:ALARM:TIME?
LXI:TRIGger:ALARM:TIME <start_sec>,<start_nsec>

LXI:TRIGger:EXTx:ENABle?
LXI:TRIGger:EXTx:ENABle <0(DISABLE)|1(ENABLE)>
LXI:TRIGger:EXTx:SLOPe?
LXI:TRIGger:EXTx:SLOPe <0(POS)|1(NEG)|2(BOTH)>

LXI:TRIGger:PPSTB:DIV?
LXI:TRIGger:PPSTB:DIV <value>
LXI:TRIGger:PPSTB:RST?
LXI:TRIGger:PPSTB:RST <0(NONE)|1(IMM)|2(ALARM)>

LXI:LANSet1|2:CONFigure: <dest>,<addr>,<prefix>,<addDat>,<useUART>
LXI:LANSet1|2:DEST?
LXI:LANSet1|2:PREFIX
LXI:LANSet1|2:ADDDAT?
LXI:LANSet1|2:ADDDAT <0(NO)|1(YES)>
LXI:LANSet1|2:UART?
LXI:LANSet1|2:UART <0(DISABLE)|1(ENABLE)>

C.2 Popis vybraných SCPI příkazů

MEASure:PERiod1|2

Měření periody využívá vstupu externího triggeru (LXI:TRIGger:EXT1|2). To je důvodem, proč jsou tyto bloky dva. PERiod1 měří period na vstupu EXT1 a PERiod2 měří period na vstupu EXT2.

CONFigure <src>,<ls>

<src> určuje při jaké události bude měření zahájeno:

0 (IMM)	měření bude zahájeno okamžitě po konfiguraci
1 (ALARM)	měření bude zahájeno Alarmem
2 (PPSTB)	měření bude zahájeno PPSTB

<ls> určuje který LANSet bude použit pro odeslání výsledku:

0	nebude použit žádný LANSet
1	bude použit LANSet1
2	bude použit LANSet2

příklad: „MEAS:PER1:CONF 0,0“

DONE?

Dotaz jestli je měření hotové. Použití tohoto příkazu je vhodné především v případě, že není použit LANSet.

PER?

Dotaz na poslední naměřený výsledek.

STOP

Příkaz, který okamžitě zastaví právě probíhané měření.

WAVeform

CONFigure <trig>,<tb>,<poin>,<ls>

<trig> určuje při jaké události bude převod zahájen:

0 (NONE)	převodník zůstane neaktivní
1 (IMM)	převod bude zahájen okamžitě
2 (EXT1)	převod bude zahájen externím zdrojem EXT1
3 (EXT2)	převod bude zahájen externím zdrojem EXT2
4 (ALARM)	převod bude zahájen ALARM-em

<tb> určuje zdroj časové základy převodníku:

0	jako časová základna bude použit ALARM
1	bude použita PPS časová základna

<poin> určuje počet vzorků převodu:

2-512

<ls> určuje který LANSet bude použit pro odeslání výsledku:

0	nebude použit žádný LANSet
1	bude použit LANSet1
2	bude použit LANSet2

příklad: „WAV:CONF 1,1,200,1“

DONE?

Dotaz jestli je měření hotové. Použití tohoto příkazu je vhodné především v případě, že není použit LANSet.

STOP

Příkaz, který okamžitě zastaví právě probíhané měření.

PORT1|2

CONFigure <trig>,<opp>,<port>,<ls>,<dat>,<mask>

<trig> určuje při jaké události bude daná operace <opp> provedena:

0 (NONE)	port zůstane neaktivní
1 (IMM)	operace bude provedena okamžitě
2 (EXT1)	operace bude provedena při událostí EXT1
3 (EXT2)	operace bude provedena při událostí EXT2
4 (ALARM)	operace bude provedena při události ALARM-u
5 (PPSTB)	operace bude provedena při události z PPSTB

<opp> určuje operaci s portem:

0 (READ)	budou přečtena data z portu procesoru
1 (WRITE)	data <dat> budou zapsána na port procesoru
2 (TOGGLE)	na portu procesoru bude vykonána bitová inverze

<port> určuje bránu procesoru:

A-F

<ls> určuje který LANSet bude použit pro odeslání přečtené hodnoty po operaci READ:

0	nebude použit žádný LANSet
1	bude použit LANSet1
2	bude použit LANSet2

<dat> obsahuje data, která mají být zapsána

0-65535

<mask> obsahuje masku, specifikuje tedy se kterými I/O linkami bude port pracovat

0-65535

příklad: „PORT1:CONF 1,1,E,0,8704,10880“

LXI

TIME?

Dotaz na aktuální čas modulu.

LXI:TIME

OFFset?
Dotaz na aktuální offset mezi modulem a *master* modulem v systému.

SYNC?
Dotaz jestli je modul synchronizován. Je vrácena „1“ pokud je offset <1us.

MASTER?
Dotaz jestli je modul master.

SLAVE?
Dotaz jestli je modul slave.

STATus?
Dotaz na aktuální stav synchronizace.

STATus:AUTO <0|1>
Příkaz zapínající/vypínající periodické asynchronní zasílání informace o aktuálním stavu synchronizace.

LXI:TRIGger:ALARM

CONFigure <start_sec>,<start_nsec>,<per_sec>,<per_nesec>,<count>

> **<start_sec> a <start_nsec>** určují čas kdy má Alarm vygenerovat další/první událost.
>
> **<per_sec> a <per_nsec>** určují periodu mezi jednotlivými událostmi. Minimální povolená perioda je 10us (10000ns).
>
> **<count>** určuje počet opakování. Pokud je zadána nula, událost bude generována bez přestání.
>
> **příklad**: „LXI:TRIG:ALARM 100,0,1,500000000,0
> „LXI:TRIG:ALARM 1,500000000,0,10000,10

LXI:TRIGger:PPSB

RST <0(NONE)|1(IMM)|2(ALARM)>

0 (NONE)	PPSTB nebude restartována
1 (IMM)	PPSTB bude restartována okamžitě po přijetí příkazu
2 (ALARM)	PPSTB bude restartována první událostí Alarmu a následně bude změněn status RST na NONE.

LXI:TRIGger:LANSet1|2

CONFigure <dest>,<addr>,<prefix>,<addDat>,<useUART>

<dest> určuje cílovou IP adresu, která je použit pouze v případě, že DEST

0 (NONE)	Lanset nebude odesílat data do Ethernetu
1 (SINGLE)	Data budou odeslána na adresu specifikovanou v parametru <addr>
2 (ALL)	Data budou odeslána všem připojeným klientům.

<addr> určuje cílovou IP adresu
<prefix> textový řetězec, který může být až 16znaků dlouhý.
<addDat> povoluje připojování dat za prefix
<useUART> povoluje použití UARTu

příklad: „LXI:TRIG:LANS1 2,“”,“prefix1“ 0,0
„LXI:TRIG:LANS2 1,“192.168.1.2“,“prefix2“ 1,1