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

SLAVE MODUL VYUŽÍVAJÍCÍ KOMUNIKAČNÍ PROTOKOL ETHERCAT

SLAVE ETHERCAT MODULE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Jakub Hadámek
AUTHOR

VEDOUCÍ PRÁCE — Ing. Soběslav Valach
SUPERVISOR

BRNO 2018

Abstrakt

Cílem práce je vytvořit slave modul využívající průmyslovou sběrnici EtherCAT. Práce je zaměřena na seznámení se s protokolem a jeho následnou implementaci do mikrokontroléru XMC4800 od firmy Infineon. Výsledkem je funkční modul s digitálními vstupy a výstupy ovládanými přes EtherCAT, kterému lze přes síť aktualizovat firmware. Jsou zde popsány také výsledky měření jitteru a srovnání s rychlejším Gigabit Ethernetem.

Klíčová slova

EtherCAT, jitter, slave modul, průmyslový Ethernet, mikrokontrolér

Abstract

The aim of this thesis is to create slave device using an EtherCAT fieldbus. It is focused on introduction of protocol and it's implementation on microcontroller XMC4800 from Infineon. The result is a functional slave module with digital inputs and outputs controlled by EtherCAT master. In addition, the firmware of the microcontroller can be updated over the network. It also describes jitter measurements and comparison with faster Gigabit Ethernet.

Keywords

EtherCAT, jitter, slave modul, Industrial Ethernet, microcontroller

Bibliografická citace:

HADÁMEK, J. *Slave modul využívající komunikační protokol EtherCAT*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2018. 52s. Vedoucí bakalářské práce byl Ing. Soběslav Valach.

Prohlášení

„Prohlašuji, že svou bakalářskou práci na téma Slave modul využívající protokol EtherCAT jsem vypracoval samostatně pod vedením vedoucího bakalářské 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 bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské 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: **21. května 2018**

……

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Soběslavu Valachovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: **21. května 2018**

……
podpis autora

OBSAH

OBRÁZKY

TABULKY

1. ÚVOD

V dnešním průmyslovém světě je kladen velký důraz na rychlost komunikace a co nejlepší synchronizaci aplikací. Tyto požadavky splňuje právě sběrnice EtherCAT (Ethernet for Control Automation Technology), vyvinutá pro potřeby automatizace. Její výhodou je mimo jiné to, že je bezplatná a kdokoliv ji po registraci u ETG (EtherCAT Technology Group) může použít pro svůj vývoj.

V této práci se budu zabývat implementací protokolu EtherCAT do mikrokontroléru XMC4800 od firmy Infineon osazený na vývojové desce Relax Kit. K tomu využiji demo aplikací výrobce a programu SSC Tool (Slave Stack Code Tool) od firmy Beckhoff. Kromě toho bude také vyzkoušeno open source řešení SOES (Simple Open EtherCAT Slave) od skupiny Open EtherCAT Society.

Za účelem testování je potřeba v práci implementovat také EtherCAT mastera. Zvoleným řešením je SOEM (Simple Open EtherCAT Master), který je rovněž open source od Open EtherCAT Society a bude použit jako spustitelná aplikace v operačním systému Linux. Dále bude využit program TwinCAT od firmy Beckhoff, jehož prostřednictvím bude otestována aplikace aktualizující firmware v použitém mikrokontroléru.

2. ETHERCAT

EtherCAT (Ethernet for Control Automation Technology) je jedním z mnoha typů průmylového Ethernetu, který byl vyvinut firmou Beckhoff Automation a poprvé představen v roce 2003. Později byl v roce 2007 uznán jako mezinárodní standard. Za jeho další vývoj je zodpovědná skupina ETG.

2.1 Popis

EtherCAT byl vyvinut přednostně pro použití v automatizaci, z čehož vyplývají také jeho primární vlastnosti, jako je velmi krátký komunikační cyklus, lepší využití ethernetového rámce, absence přepínačů a nízká cena. Využívá stejnou fyzickou vrstvu jako Ethernet, avšak oproti němu funguje real-time, neboť veškeré zpoždění sběrnice je díky zpracování rámců „za letu“ („data on the fly“) dáno pouze zpožděním hardwaru.

EtherCAT funguje na principu master-slave, přičemž master je jediný, kdo v síti vysílá nové ethernetové rámce. Ty se po průchodu posledním slavem vracejí zpět a uzavírají tak logický kruh.

K implementaci mastera není potřeba žádný speciální hardware, stačí Ethernet kontrolér nebo obyčejná síťová karta. Masterem se tak může stát jakékoliv zařízení s Ethernet portem. Celá jeho funkce spočívá v softwaru, který může být buď placený od firmy Beckhoff Automation nebo open source určený především pro operační systémy založené na Linuxu. Mezi tyto open source řešení patří EtherLab a SOEM (Simple Open EtherCAT Master). Slave naopak vyžaduje speciální hardware, u kterého jsou kladeny velké nároky na rychlost průchodu signálu.

2.2 Field memory management unit (FMMU)

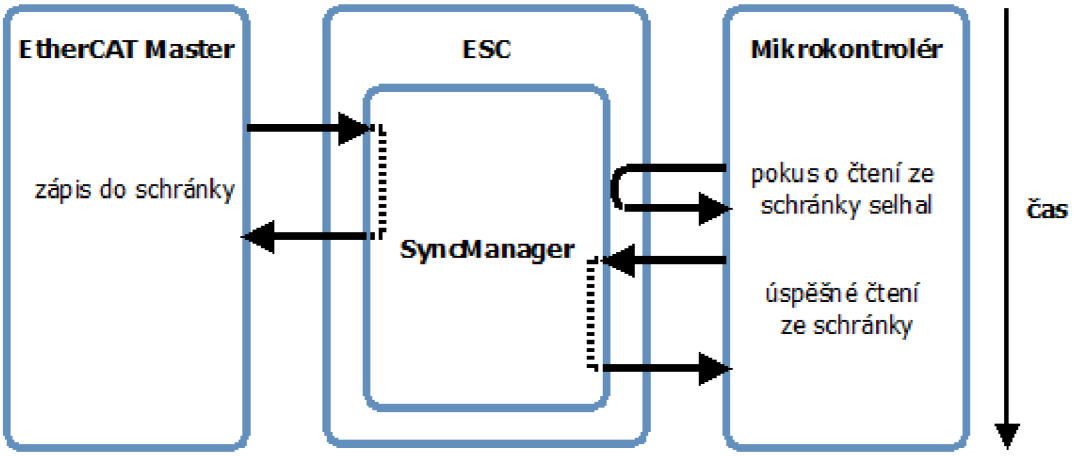
FMMU je jednotka správy paměti používající se pro mapování procesních dat z logického obrazu v masteru do fyzické paměti ve slave zařízení. Procesní data jsou řazena podle úkolů a master pomocí FMMU nastavuje, která zařízení se smí mapovat ve stejném datagramu. [2]

2.3 SyncManager (SM)

SyncManager je mechanismus chránící data v DPRAM (Dual Port RAM) před současným přístupem mastera prostřednictvím EtherCAT sítě a mikrokontroléru prostřednictvím PDI (Process Data Interface). [2] Pokud slave používá FMMU, je SyncManager umístěn mezi FMMU a DPRAM. Může pracovat ve dvou módech:

2.3.1 Mailbox mód

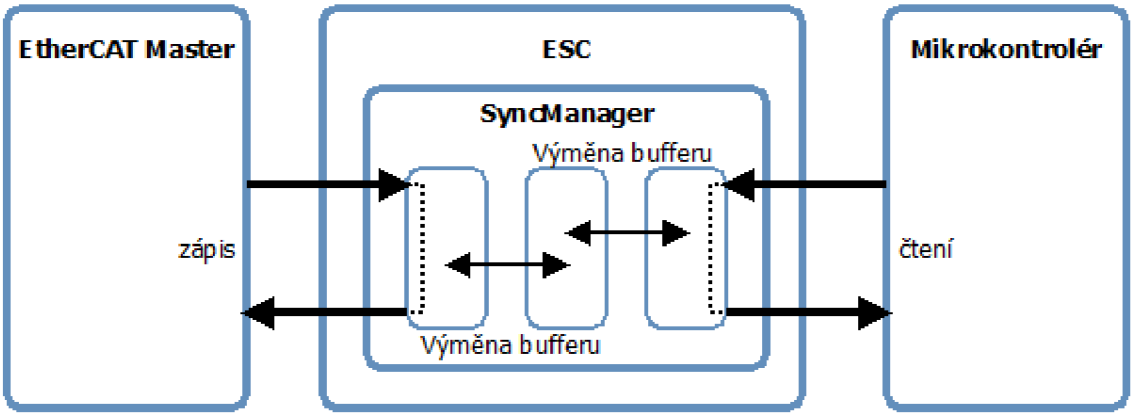
V tomto módu se pro výměnu dat mezi ESC (EtherCAT Slave Controller) a MCU (Microcontroller) využívá handshake. EtherCAT master a MCU může ke schránce přistupovat pouze tehdy, pokud druhý z nich dokončil svůj přístup. Pokud do ní jedna strana zapsala data, je uzamčena dokud ji druhá strana nevyprázdní. Tento mód se používá pro necyklické odesílání dat, při kterém o žádná nechceme přijít. [2]



Obr. 2.1: SyncManager v mailbox módu

2.3.2 Buffer mód

Používá se pro cyklickou výměnu. V tomto režimu je potřeba třikrát větší paměťový prostor, než je velikost procesních dat, neboť jsou použity tři buffery. Master i MCU smí k datům přistupovat kdykoliv bez omezení. V případě, že jsou nová data k dispozici dříve, než se stará stihla odeslat nebo přečíst, je buffer aktualizován a stará jsou zahozena. [2]

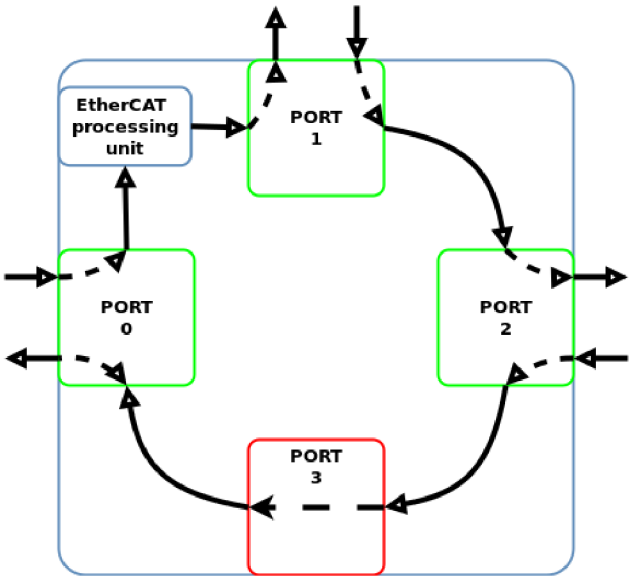


Obr. 2.2: SyncManager v buffer módu

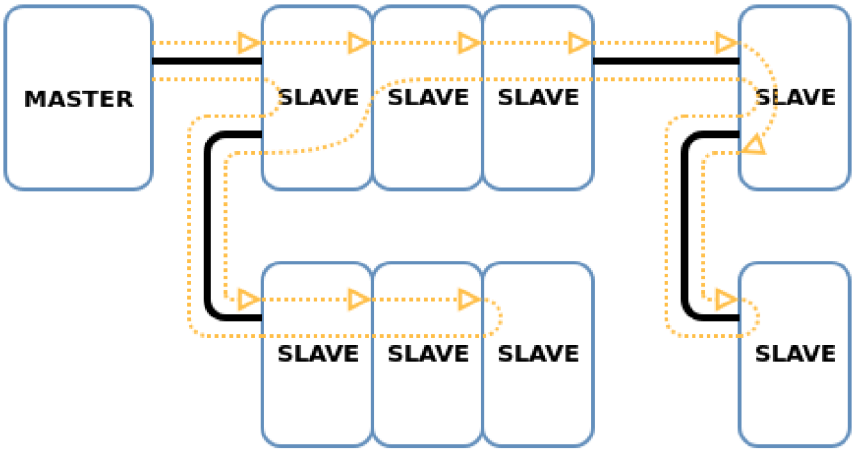
2.4 Topologie

EtherCAT může být fyzicky zapojen v různých topologií, jako například strom, hvězda, kruh, bus nebo line. Všechny ve výsledku uzavírají logický kruh, přičemž je možno připojit až 65 535 zařízení. Do vzdálenosti 100 m se jako fyzická vrstva používá fast Ethernet (100BASE-TX), na větší vzdálenosti pak optické vlákno (100BASE-FX). EtherCAT je vhodný pro centralizované i decentralizované systémy, umožňuje komunikaci master-master, master-slave i slave-slave. Jeho použití je jednodušší než použití Ethernetu, protože není zapotřebí

přepínačů, MAC, ani IP adres. Na obrázcích obr. 2.3 a obr 2.4 můžeme vidět průchod packetu přes ESC a EtherCAT síť. [4]



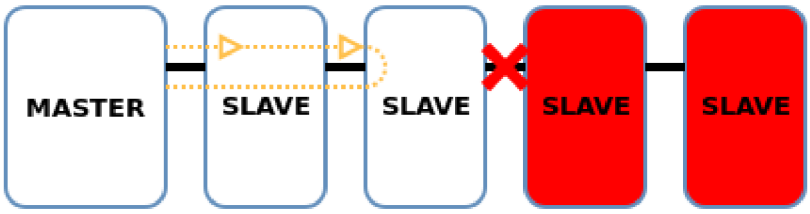
Obr. 2.3: Průchod packetu přes ESC



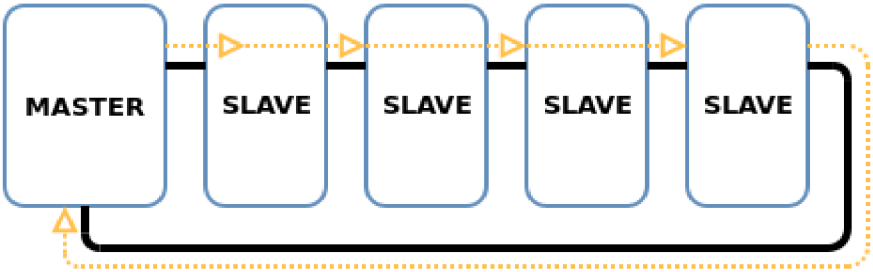
Obr. 2.4: Průchod packetu EtherCAT sítí

2.4.1 Odolnost

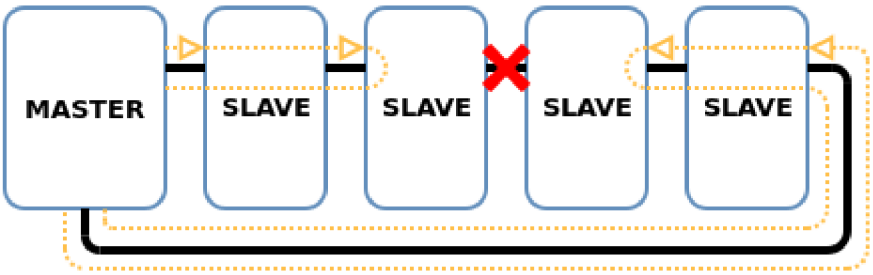
Při přerušení řetězce z důvodu porušení média nebo selhání zařízení jsou další zařízení odstřižena od sítě. K odstranění tohoto nedostatku se používá zapojení, v němž se poslední slave v síti propojí s masterem, čímž se uzavře smyčka. V případě výpadku tak dojde ke ztrátě pouze jednoho komunikačního cyklu, neboť master detekuje chybu a komunikace bude dále pokračovat. K použití tohoto zapojení je zapotřebí pouze dvou ethernetových portů v masteru. Na následujících obrázcích lze vidět princip komunikace odolném režimu. [6]



Obr. 2.5: Průchod packetu EtherCAT sítí při poruše



Obr. 2.6: Průchod packetu EtherCAT sítí v odolném zapojení



Obr. 2.7: Průchod packetu EtherCAT sítí v odolném zapojení při poruše

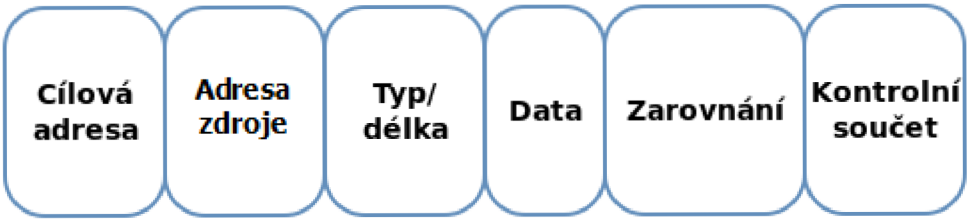
2.5 Protokol

EtherCAT používá ethernetové rámce podle standardu IEEE 802.3. Z tohoto důvodu není na straně mastera zapotřebí speciálního hardwaru, ale postačí obyčejný síťový kontrolér.

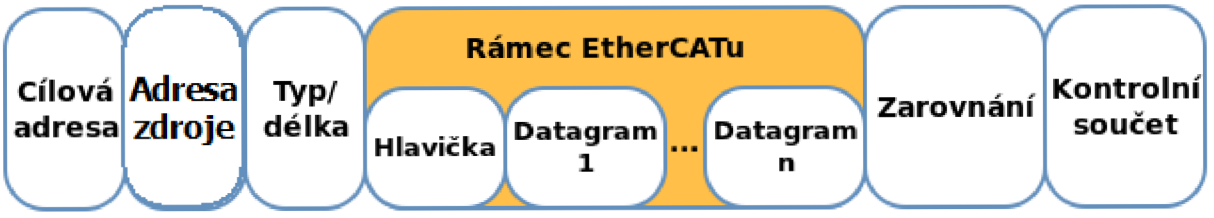
2.5.1 Struktura rámce

Na pozici EtherType, určující jaký protokol je zapouzdřený v datové části rámce je EtherCAT identifikován hodnotou 0x88A4. Samotný rámec EtherCATu je vložen do místa pro přenášená data a skládá se z hlavičky následované jednotlivými datagramy, nazývanými také PDO (Process Data Object). Každý datagram se skládá z vlastní hlavičky, přenášených dat a pracovního čítače WKC (WorKing Counter). Ten je inkrementován každým zařízením, které s danými daty pracovalo. Poté master vyhodnotí, zda se rovná předem vypočtané hodnotě, jíž by se měl WKC rovnat a pokud se nerovná, vyhodnotí to jako chybu přenosu. [2] Na obrázcích níže je ukázáno, jak vypadá rámec Ethernetu a jakým způsobem jsou v něm zapouzdřeny rámce

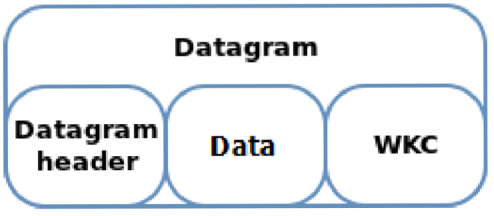
EtherCATu. EtherCAT také umožňuje používání speciálních protokolů, které budou popsány dále.



Obr. 2.8: Rámec Ethernetu



Obr. 2.9: Rámec EtherCATu zapouzdřený uvnitř datové části Ethernet rámce



Obr. 2.10: Datagram EtherCAT rámce

2.5.1.1 Hlavička EtherCAT rámce

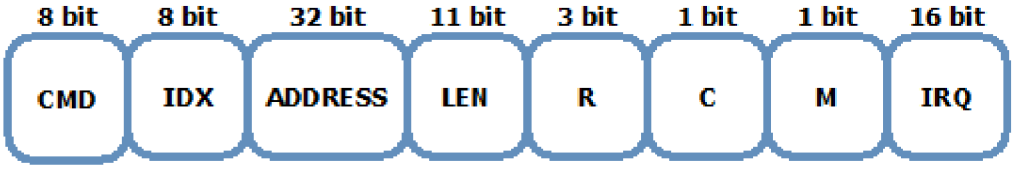
Hlavička EtherCAT rámce je složena z pouze šestnácti bitů.

- LENGTH (11 bitů) – v tomto poli je uložena celková délka EtherCAT datagramů
- RESERVED (1 bit) – rezervováno
- TYPE (4 bity) – typ protokolu

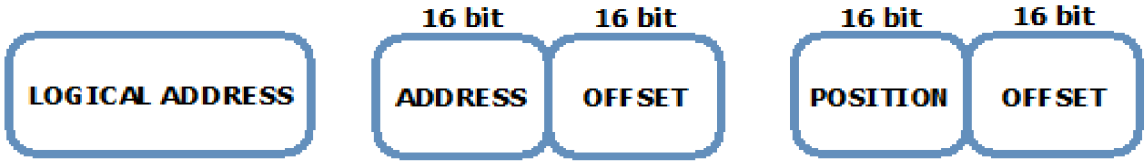
2.5.1.2 Hlavička datagramu

- CMD – příkaz
- IDX – číselný identifikátor používaný masterem k označení duplicitních nebo ztracených rámců používaný masterem
- ADDRESS – adresa zařízení viz následující podkapitola
- LEN – délka následující datové části

- R – rezervováno
- C – označují cirkulující rámec
- M – oznamuje zda se jedná o poslední datagram, nebo bude následovat další
- IRQ – požadavek na událost



Obr. 2.11: Hlavička EtherCAT datagramu



Obr. 2.12: Adresa v hlavičce EtherCAT datagramu

2.5.2 CAN over EtherCAT (CoE)

S COE EtherCAT nabízí stejný komunikační mechanismus, jako v CANopen standardu EN50325-4. CANopen je aplikační vrstva známé sběrnice CAN. Ta byla původně vytvořena pro systémy řízení pohybu, ale dnes se používá v mnohem širším okruhu aplikací jako jsou medicínská technika, automobilový průmysl, železniční doprava nebo automatizované budovy.

2.5.3 Servo Profile over EtherCAT (SoE)

Umožňuje řízení pohybu pomocí univerzálního rozhraní SERCOS (SErial Real time COmmunication Specification), které je od roku 1995 mezinárodním standardem specializovaným na sériovou real-time komunikaci s digitálními servomotory.

2.5.4 Ethernet over EtherCAT (EoE)

Ethernetové rámce mohou být přepraveny přes EtherCAT segment, neboť mají společné fyzické médium. Ethernet zařízení může být připojeno k EtherCAT síti pomocí tzv. switchportu. Rámce jsou tak tunelovány skrz EtherCAT síť, což ji dělá pro ethernetová zařízení kompletně transparentní.

2.5.5 File access over EtherCAT (FoE)

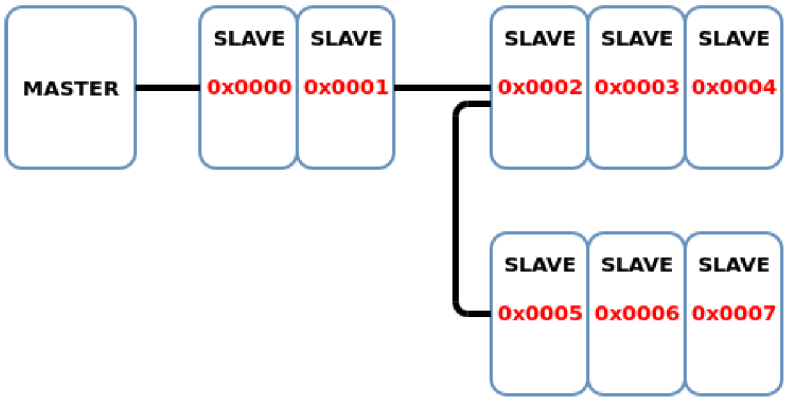
Jednoduchý protokol podobný TFTP umožňující přenos souborů přes EtherCAT síť. Díky tomuto protokolu je možné provést aktualizaci firmwaru po síti, čehož je v této práci využito.

2.6 Adresování

Slave zařízení mohou být síti adresována inkrementálně, pevně, nebo logicky. Všechny tři způsoby jsou popsány níže.

2.6.1 Inkrementální

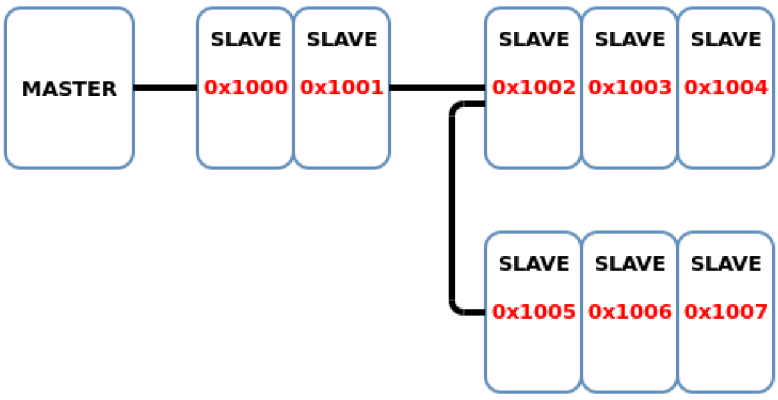
Zařízení jsou adresována postupně v pořadí, ve kterém jsou zapojena. První slave za masterem dostane adresu 0, uloží ji, inkrementuje a pošle dalšímu. Délka adresy je 2 byty.



Obr. 2.13: Inkrementální adresování v EtherCAt síti

2.6.2 Pevné

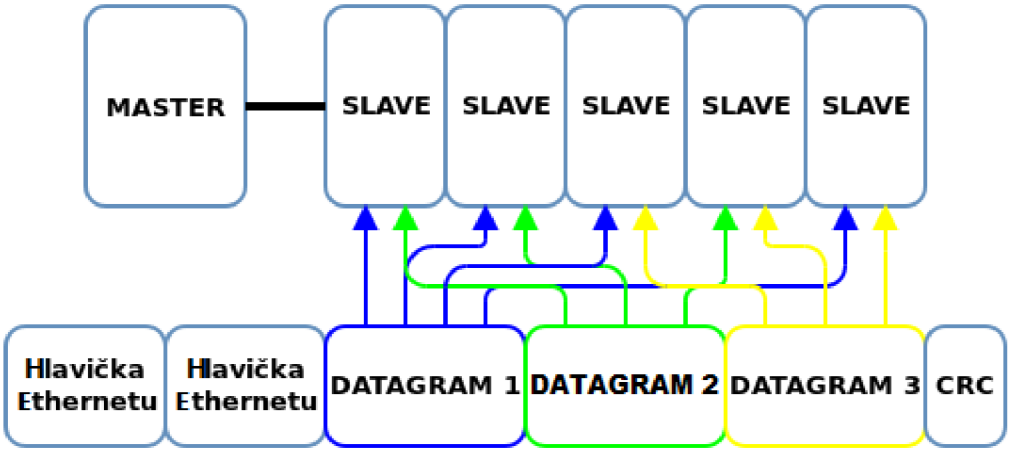
Zařízení má přidělenou adresu, které neodpovídá jeho pořadí v síti. Adresa je ztracena v případě výpadku napětí. Délka adresy je 2 byty.



Obr. 2.14: Pevné adresování v EtherCAT síti

2.6.3 Logické

Slave čte/zapisuje data z/do logického adresového prostoru v masteru o velikosti 4GB. Délka adresy je v tomto případě 32 bitů.



Obr. 2.15: Logické adresování v EtherCAT síti

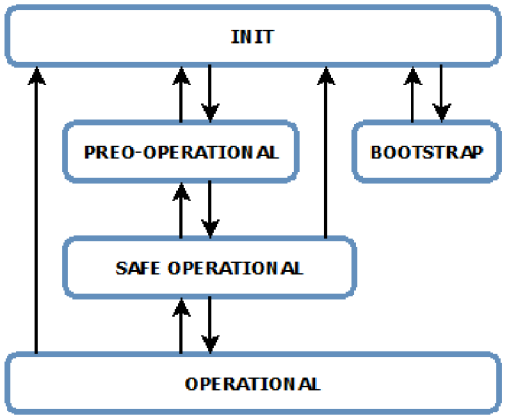
2.7 Synchronizace

V aplikacích, kde je potřeba provádět více operací současně, je kladen velký důraz na synchronizaci jednotlivých prvků. Jako příklad lze uvést více servomotorů provádějící koordinovaný pohyb. EtherCAT nabízí možnost distribuce času, díky čemuž lze procesy synchronizovat s přesností na méně než 1 μs.

Master je schopen vyrovnat zpoždění na sběrnici tak, že zjistí čas potřebný pro průchod mezi jednotlivými zařízeními. Toho se dosáhne vysláním synchronizačního datagramu, do něhož každý slave uloží čas průchodu. Poté vypočte zpoždění každého zařízení a počítá s ním v následujících operacích. Referenční čas je běžně brán z prvního slave zařízení, které má schopnost distribuovat čas. [2]

2.8 EtherCAT state machine

Stavový automat indikuje, které funkce EtherCATu jsou momentálně k dispozici. Požadavky na přechod mezi stavy jsou generovány masterem. V případě, že slave splňuje požadavky na přechod mezi stavy a požadovaný přechod je legitimní, odešle slave potvrzení o přechodu, jinak odpoví hlášením o chybě. [4]



Obr. 2.16: Přechody mezi stavy ve stavovém automatu EtherCATu

Init

Neprobíhá žádná komunikace na aplikační úrovni. Master má přístup pouze k DL (Data Link Layer) registrům. Tento stav je výchozí po zapnutí napájení zařízení.

Preoperational

Je možná pouze mailbox komunikace, nelze přenášet procesní data. Při přechozu z Init do Preoperational stavu master přečte z EEPROM (Electrical Erasable Programable Read Only Memory) paměti ID výrobce, produktové číslo a číslo revize a nastaví DL registry, SM registy a synchronizaci.

Safe Operational

Probíhá mailbox komunikace a lze přenášet i vstupní procesní data, výstupní procesní data nejsou vyhodnocována a výstupy jsou drženy v bezpečném stavu. Při přechodu z Preoperational do Safe Operational stavu master nastaví mapování procesních dat, registry pro SM procesních dat a FMMU registry.

Operational

Vstupní i výstupní procesní data jsou validní. Při přechodu ze Safe Operational do Operational stavu master nastaví validní výstupní hodnoty.

Boot

Volitelný, ale doporučený, pokud je požadována aktualizace firmwaru. Nelze odesílat ani přijímat procesní data, komunikace je možná pouze přes mailbox. V tomto stavu je většinou použit FoE protokol pro přenos firmwaru.

2.9 Fyzická vrstva

EtherCAT je kompatibilní s dnes všude používaným Ethernetem. Z tohoto důvodu také používá stejnou standardizovanou fyzickou vrstvu. EtherCAT je schopen fungovat na standardu 100BASE-TX (metalické vedení – kroucená dvojlinka) a 100BASE-FX (optické vlákno). Tyto standardy fungují na rychlosti 100 Mbit/s a je schopen je splnit stolní PC, notebook, průmyslové PC a jakékoliv další zařízení disponující 100 Mbit síťovým kontrolérem.

3. MASTER

V této kapitole dojde k seznámení s dostupnými řešeními EtherCAT mastera. Dále zde bude uvedeno, které z nich a proč byly v práci použity a jak je lze zprovoznit na jakémkoliv počítači.

Ke zprovoznění mastera není zapotřebí žádný speciální hardware, postačí jakékoliv zařízení s Ethernet portem. Veškerá jeho činnost je dána softwarem. V této práci byl EtherCAT master implementován do notebooku se 100 Mbit síťovou kartou.

3.1 Dostupná řešení

K dispozici je několik placených i open source řešení EtherCAT mastera.

- TwinCAT
- EtherLAB
- Simple Open EtherCAT Master

První zmíněný je TwinCAT od firmy Beckhoff, která je zodpovědná za uvedení EtherCATu na trh. Tento software je založený na vývojovém prostředí Visual Studio od společnosti Microsoft. Lze jej vyzkoušet ve formě dema, které je kromě časového omezení plně funkční. Toho bylo využito pro otestování aktualizace firmwaru přes EtherCAT síť.

Druhou možností je open source řešení EtherLAB vytvořený pro operační systém Linux. Je vytvořen pro starší verzi Linux kernelu, což je jeho velkou nevýhodou. Výhodou je naopak použitelnost ve formě modulu běžícího v kernel prostoru systému. Tím dosahuje většího determinismu než další uvedená aplikace, ale také mu přidává na složitosti. [8]

Třetí možností je také open source řešení Simple Open EtherCAT Master. Ten je k dispozici ve formě knihoven a ukázkových demo aplikací. Na rozdíl od EtherLABu je spustitelný pouze z uživatelského prostoru operačního systému, což jej činí nedeterministickým, avšak jednodušším na implementaci. Ukázkové aplikace jsou zcela vhodné k otestování vytvořeného EtherCAT modulu. [13]

3.2 Aplikace

V této práci byly vyzkoušeny dvě implementace EtherCAT mastera, kterými jsou SOEM a TwinCAT.

3.2.1 SOEM

Spolu s knihovnami jsou k dispozici také demo aplikace realizující zapisování a čtení z EEPROM paměti, diagnostiku sítě a cyklické odesílání procesních dat. Veškeré zdrojové kódy jsou open source a jednoduše dostupné na githubu. [13]

Ke kompilaci ukázkových aplikací je nejprve potřeba mít nainstalován program cmake a gcc kompilátor. Pokud splňujeme tyto požadavky, je možné následujícími kroky zkompilovat zdrojové soubory.

- Vytvořit složku build v domovském adresáři SOEM příkazem *mkdir build*

- Přejít do složky build příkazem *cd build*
- Vygenerovat makefile příkazem *cmake ..*
- Přejít do adresáře SOEM/build/test/Linux/simple-test příkazem *cd test/Linux/simple-test*
- Spustit makefile příkazem *make*, čímž se provede kompilace program

Nyní máme zkompilované demo dodané vývojáři SOEM. Zdrojový soubor k programu se nachází v adresáři SOEM/test/Linux/simple-test. Ten můžeme dle libosti upravovat a měnit tak chování EtherCAT mastera. Za účely testování byla aplikace upravena tak, aby byly výstupy po dobu 30 cyklů ze 100 v logické 1. To se provedlo zapsáním hodnoty 0xff do výstupů outputs v poli struktur ec_slave. Po každé úpravě zdrojového kódu je potřeba aplikaci znova zkompilovat příkazem *make* v adresáři SOEM/build/test/Linux/simple-test. Aplikace musí být spuštěna administrátorem, protože chce využívat raw socketů. Spustí se příkazem *sudo ./simple-test eth0*, kde eth0 musí být nahrazeno názvem rozhraní hostujícího PC. Úpravu programu a jeho spuštění lze vidět na následujících obrázcích.

```
/* cyclic loop */
while(1) {
        ++c;
        if(c%100 < 30)
        {
                for(int a = 1; a <= ec_slavecount; ++a)
                {
                        for( unsigned int i = 0; i < ec_slave[a].Obytes; ++i)
                                ec_slave[a].outputs[i] = 0xff;
                }
        }
        else
        {
                for(int a = 1; a <= ec_slavecount; ++a)
                {
                        for( unsigned int i = 0; i < ec_slave[a].Obytes; ++i)
                                ec_slave[a].outputs[i] = 0;
                }

        }

        ec_send_processdata();
        wkc = ec_receive_processdata(EC_TIMEOUTRET);
```

Obr. 3.1: Úprava SOEM demo aplikace pro změnu výstupních hodnot

```
jakub@jakub-VirtualBox ~/src/bc_thesis/MASTER/SOEM/build/test/linux/simple_test $ sudo ./simple_test enp0s3
[sudo] password for jakub:
SOEM (Simple Open EtherCAT Master)
Simple test
Starting simple test
ec_init on enp0s3 succeeded.
1 slaves found and configured.
Slaves mapped, state to SAFE_OP.
segments : 1 : 18 0 0 0
Request operational state for all slaves
Calculated workcounter 3
Operational state reached for all slaves.
Processdata cycle 77098, WKC 3 , O: ff ff ff ff ff ff ff ff I: 00 00 00 00 00 00 00 00 T:3348356927760258840
```

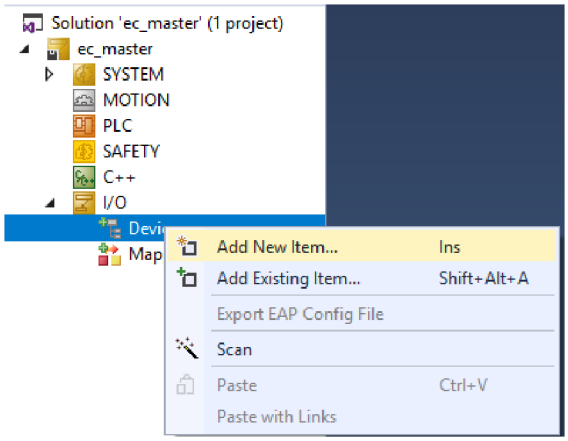
Obr. 3.2: Spuštění EtherCAT mastera

3.2.2 TwinCAT

TwinCAT byl v této práci použit pouze za účelem otestování aktualizace firmwaru slave zařízení přes EtherCAT síť. Důvodem je, že pokus o aktualizaci pomocí SOEM skončil chybou a bylo tak třeba vyloučit chybu mastera. Jak již bylo zmíněno výše, TwinCAT je založený na vývojovém prostředí Visual Studio, a proto je potřeba jej nejprve nainstalovat. Poté je možno nainstalovat samotný TwinCAT, který je k dispozici na stránkách firmy Beckhoff.

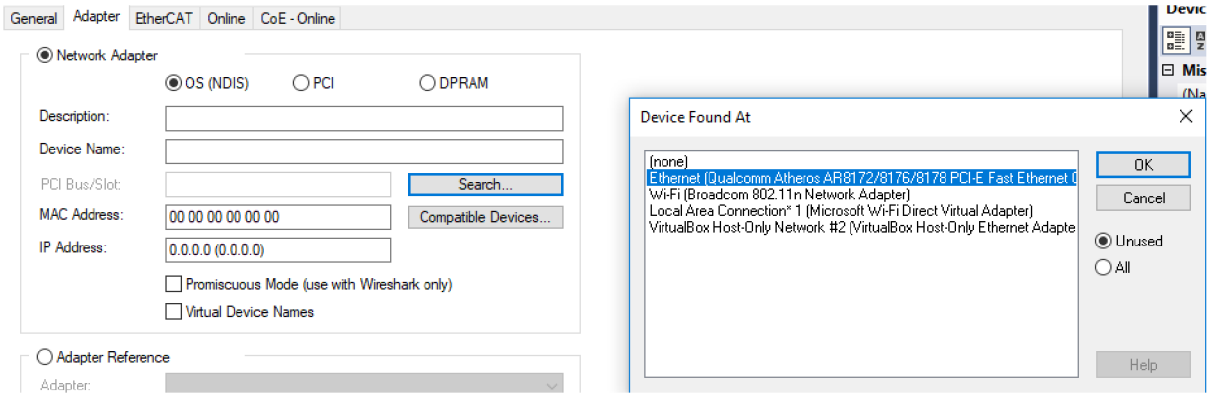
Jeho použití je velice intuitivní a jednoduché. Update firmwaru ve slave modulu se provede následujícími kroky:

1. Založit nový project typu TwinCAT XAE Project. Po jeho otevření pak v levém panelu vybrat záložku I/O (Input/Output), kliknout pravým tlačítkem na Devices a Add New Item. Z nabídnutých možností potvrdit zařízení typu EtherCAT Master.



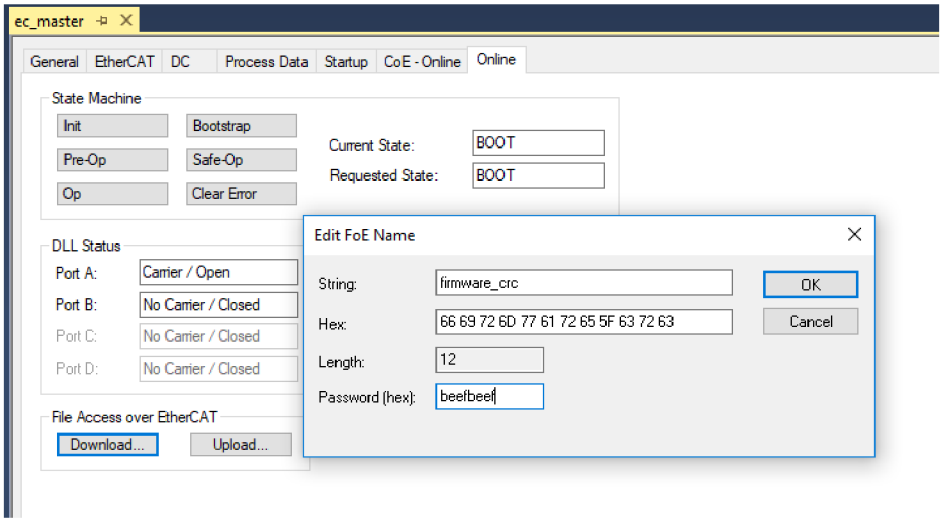
Obr. 3.3: TwinCAT - vytvoření EtherCAT mastera

2. V záložce Devices se poté objeví EtherCAT Master. Kliknutím na něj se zobrazí okno s nastavením, kde se v položce Adapter ukáží možnosti pro výběr správného síťového kontroléru. Kliknutí na Search nabídne možná zařízení, v tomto případě *Ethernet (Qualcomm Atheros....).*



Obr. 3.4: TwinCAT - výběr síťového kontroléru

3. Po vybrání správného kontroléru kliknout pravým tlačítkem na master device a vybrat možnost *Scan*. Master tak provede sken sítě a najde všechna připojená zařízení, která se poté zobrazí v záložce *Devices* pod EtherCAT masterem.
4. Po otevření zvoleného zařízení vyskočí okno s informacemi a nastavením. V kartě online je možné zažádat o přechody mezi stavy. Kliknutím na *Bootstrap* se zažádá o přechod do bootstrap stavu, ve kterém je možno aktualizovat firmware.
5. Kliknout na tlačítko Download a vybrat nový firmware připravený pro update. K úspěšnému dokončení přenosu je potřeba zadat heslo v hexadecimálním tvaru, které je v případě tohoto slave modulu BEEFBEEF.



Obr. 3.5: TwinCAT - deslání nového firmwaru

6. Pro aktualizaci firmwaru musí slave přejít do init stavu. Poté se zařízení restartuje a přepíše starý firmware novým.

4. SLAVE

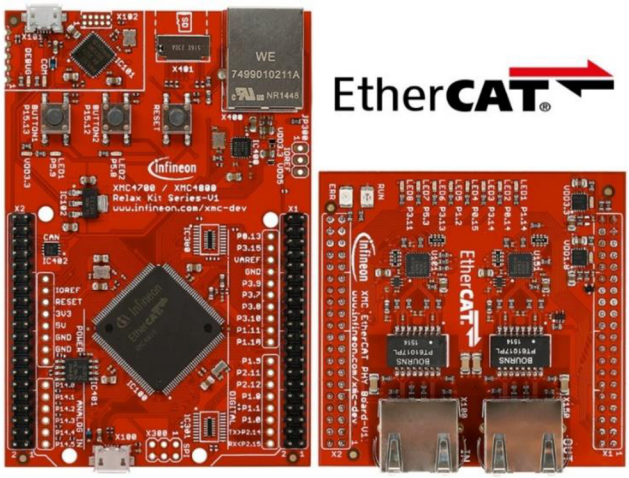
Tato kapitola se bude zabývat implementací protokolu EtherCAT do mikrokontroléru XMC4800 od firmy Infineon. Kromě hardwaru je potřeba také Slave Stack Code, který realizuje EtherCAT slave. Ten lze vygenerovat nástroji Slave Stack Code Tool nebo kombinací SOES (Simple Open EtherCAT Slave) a EtherCAT SDK (Software Development Kit).

4.1 Hardware

Použitým hardwarem je vývojová deska od firmy Infineon osazená mikrokontrolérem XMC4800 postaveným na architektuře ARM Cortex-M4, který má EtherCAT implementován jako vnitřní periferii. [9] Kromě samotného mikrokontroléru nás na desce zajímá také PHY firmy Broadcom. Bylo by ideální znát jeho zpoždění při odesílání a přijímání, ale výrobce tyto parametry bohužel neuvádí. [7] Mikrokontrolér má výstupy připojeny k LED diodám. Můžeme tak přes síť měnit hodnoty výstupů a sledovat výsledek. Kromě toho má také připojena dvě tlačítka, kterými můžeme simulovat vstupy.

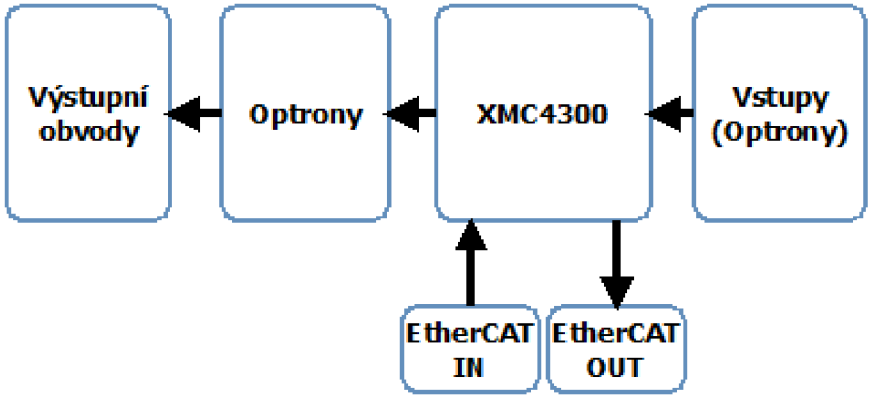
Specifikace vývojového kitu:

- XMC4800 ARM Cortex-M4 144 MHz
- Flash 2 MB
- SRAM 352 KB
- 2x uživatelské tlačítko
- 10x LED
- microSD
- EtherCAT
- 3x 100 Mbit Ethernet
- Napájení přes USB
- CAN



Obr. 4.1: Infineon XMC4800 Relax Kit

Kromě vývojového kitu jsem dostal k dispozici také IO slave modul od firmy DFC Design, který je osazen mikrokontrolérem XMC4300. Ten je stejně jako XMC4800 postaven na architektuře ARM Cortex-M4, avšak na rozdíl od něj nabízí méně periferií a disponuje menším množstvím paměti Flash a RAM. Na následujícím obrázku lze vidět zjednodušené blokové schéma IO modulu.



Obr. 4.2: Zjednodušené blokové schéma IO modulu od firmy DFC Design

4.2 Možnosti řešení

K implementaci EtherCAT protokolu je zapotřebí EtherCAT Slave Stacku, zajišťujícího jeho funkci. Na výběr je ze dvou možností, které jsou v této práci obě použity.

První variantou je Slave Stack Code Tool od firmy Beckhoff. Pro získání tohoto nástroje je potřeba být registrován u skupiny EtherCAT Technology Group. To je umožněno pouze firmám a fyzická osoba nemá možnost jej získat. SSC Tool je nástroj pro generování SSC (Slave Stack Code), ESI (EtherAT Slave Information) a dat pro EEPROM paměť na základě konfigurace, která zde bude popsána. [5]

Druhou možností je Simple Open EtherCAT Slave od skupiny EtherCAT Socienty, která nabízí open source knihovnu použitelnou k vytvoření EtherCAT zařízení. Kromě knihovny a zdrojových kódů je možné využít také nástroje EtherCAT SDK. V EtherCAT SDK lze nadefinovat parametry zařízení a následně vygenerovat aplikaci realizující EtherCAT slave a XML ESI soubor s definovanými parametry. K použití SDK je na rozdíl od knihoven potřeba získat licenci, kterou je společnost pro studijní účely ochotna poskytnout, avšak jen na dobu jednoho měsíce. Po uplynutí této doby je potřeba o ni znova zažádat. [14]

4.3 Information file (ESI)

Jedná se o XML soubor dodávaný výrobcem spolu se modulem, obsahující informace o výrobci a výrobku. Kromě těchto formalit popisuje také parametry zařízení, jako je nastavení

SyncManageru, FMMU, procesních dat, schránek, použitých datových typů a některých registrů. ESI soubor je v této práci generován automaticky na základě nastavených funkcí použitím nástrojů EtherCAT SDK a SSC Tool.

4.4 Implementace

Zadáním práce bylo implementovat protokol EtherCAT do mikroprocesoru. To bylo provedeno zprovozněním aplikace využívající SSC Tool, která realizovala digitální vstupy a výstupy. Kromě toho byly vytvořeny ještě další dvě aplikace. První z nich byla také vytvořena za použití SSC Tool, avšak navíc umožňuje přes EtherCAT síť aktualizovat firmware v mikrokontroléru. Druhá používá kombinace SOES a EtherCAT SDK a umožňuje číst vstupy a ovládat výstupy. Práce byla programována ve vývojovém prostředí DAVE IDE od firmy Infineon.

4.4.1 Procesní data

Jedna z nejčastějších úloh v automatizaci je řízení výstupů a čtení vstupů. To je v EtherCATu prováděno cyklicky v podobě procesních dat. V následujících podkapitolách bude popsána implementace za pomoci nástrojů Slave Stack Code Tool a Simple Open EtherCAT Slave.

4.4.1.1 Slave Stack Code Tool

Projekt je založen na příkladech výrobce čipu. [10] Skládá z několika částí, kterými jsou Slave Stack kód, uživatelská aplikace a APP generované vývojovým prostředím.

Slave Stack kód je uložen v adresáři SSC, kde se kromě zdrojového kódu nachází také konfigurační soubory popisující hardware, generovaný ESI a tabulka s nastavením procesních dat.

Uživatelská aplikace je v souboru main.c, ve kterém se nacházejí funkce:

- nastavující příslušné piny mikrokontroléru
- aktualizující stavy vstupů a výstupů
- main, ve které se při vypnuté synchronizaci periodicky volá funkce realizující EtherCAT slave

APP se nachází v adresáři Dave. Tyto aplikace jsou použity k inicializaci některých funkcí mikrokontroléru, jako jsou například časovače, přerušení, emulování EEPROM paměti a inicializace ESC.

K vygenerování Slave Stacku je použit software od firmy Beckhoff SSC Tool. V něm lze vytvořit popis EtherCAT slave modulu buďto od základu, nebo využít konfiguračního XML souboru generovaného vývojovým prostředím při použití jeho APP. Použil jsem již hotový konfigurační soubor a naimportoval jej. Z tohoto důvodu není potřeba provádět žádné změny v nastavení.

V levé části nástroje SSC Tool jsou k vidění jednotlivé sekce s nastavením určitých funkcí stacku:

- Slave information

Lze nastavit informace o výrobci, sériové čílo, kód produktu a další informace.

- Hardware
  Zde se nastavuje zda má být vygenerován také obsah EEPROM paměti, případně jakou má paměť velikost.
- EtherCAT State Machine
  Umožňuje do stavového automatu přidat boot stav a nastavit timeout pro přechod mezi některými stavy.
- Synchronisation
  Nastavení distribuovaných hodin, minimální a maximální délky jednoho cyklu a zpoždění vstupů/výstupů.
- Application
  Možnost generování ukázkových aplikací. Po rozrolování nastavení adres pro zpracování procesních dat a jejich maximální velikost.
- Mailbox
  Nastavení mailbox komunikace. Lze zde aktivovat protokoly využívající „schránkovou“ komunikaci, jako je například File Over EtherCAT, Ethernet Over EtherCAT, Can Over EtherCAT a další. Dále je zde možno nadefinovat velikosti mailboxů a jejich adresy.

Jakmile jsou parametry nastaveny, přichází na řadu definice vstupních a výstupních procesních dat. To se provádí zápisem to excelovské tabulky, kterou je možné vygenerovat kliknutím na Tool v horním panelu, dále výběrem možnosti Application a Creat new.

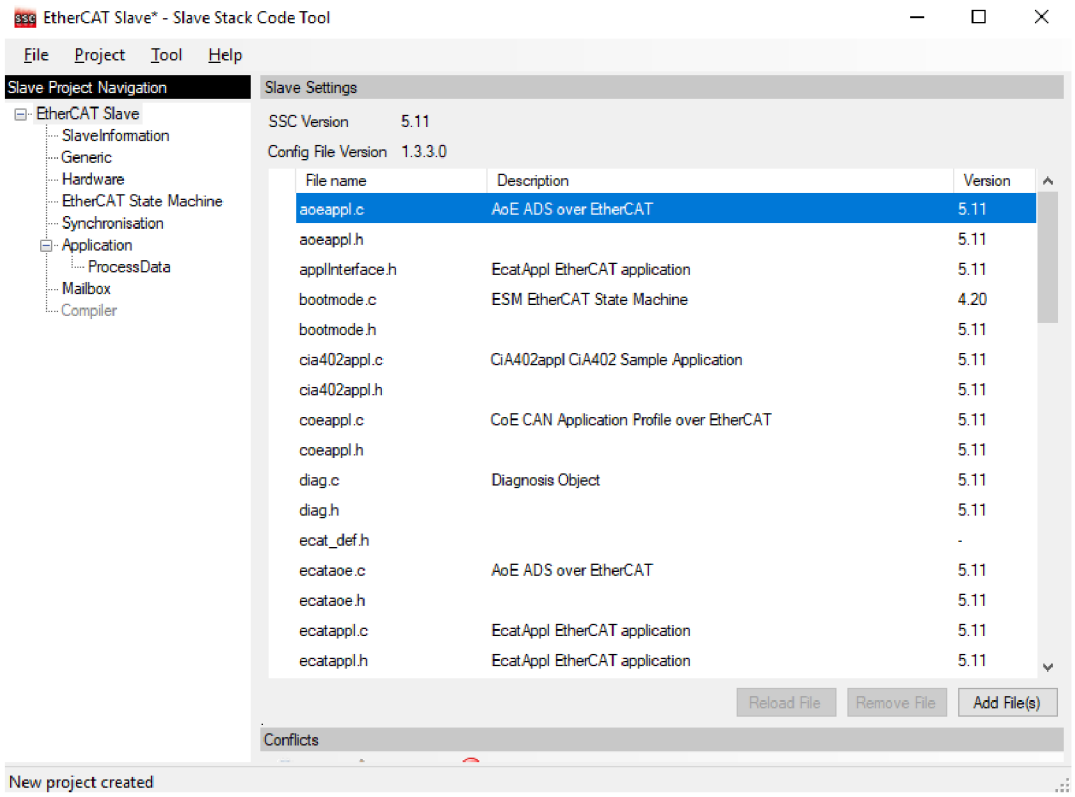
Tabulka je složena z několika sekcí, přičemž nastavení vstupů a výstupů se provádí v sekci Input Data a Output Data. Významy jednotlivých sloupců jsou:

- Index
  Index objektu, který může být pro vstupní data libovolný od 0x6000 do 0x6FFF a pro výstupní data 0x7000 až 0x7FFF.
- ObjectCode
  Udává, zda se jedná o záznam, pole nebo proměnnou. Pole musí být jednoho datového typu, zatímco záznam se může stejně jako struktura skládat z různých datových typů.
- SI (SubIndex)
  Je index proměnné v objektu začínající od jedničky.
- DataType
  Určuje datový typ položky. Datových typů je celá řada, ale uvedeny zde budou jen ty základní. Ty jsou BOOLEAN (1 bit), BYTE (8 bitů), WORD (16 bitů), DWORD (32 bitů). Kompletní seznam lze nalézt v dokumentaci k SSC. [5]
- Name
  Název položky.

0x6nnx	Input Data of the Module (0x6000 - 0x6FFF)									
0x6000	RECORD			INPUTS						
		0x01	BOOL	BUTTON1	0					ro
		0x02	BOOL	BUTTON2	0					ro
0x7nnx	Output Data of the Module (0x7000 - 0x7FFF)									
0x7000	RECORD			OUTPUTS						
		0x01	BOOL	LED1	0					rw
		0x02	BOOL	LED2	0					rw
		0x03	BOOL	LED3	0					rw
		0x04	BOOL	LED4	0					rw
		0x05	BOOL	LED5	0					rw
		0x06	BOOL	LED6	0					rw
		0x07	BOOL	LED7	0					rw
		0x08	BOOL	LED8	0					rw

Obr. 4.3: Konfigurace vstupních a výstupní dat v tabulce generované nástrojem SSC Tool

Po uložení zbývá vygenerovat zdrojové soubory s parametry, vstupy a výstupy nastavenými ve výše zmíněných krocích. To se provede kliknutím na tlačítko „Create new slave files“ v záložce „Project“. Tím se kromě zdrojových kódů vygeneruje také „ESI file“ (EtherCAT Slave Information file), který obsahuje veškeré nastavení slave zařízení a je dodáván výrobcem spolu s modulem.



Obr. 4.4: Slave Stack Code Tool

Ve generovaných kódech je potřeba udělat malé úpravy, které propojí uživatelskou aplikaci se slave stackem.

Jako první se musí ve funkci *APPL_Application* volat funkce *process_app*, která má na starosti nastavování výstupů a čtení vstupů. *APPL_Application* je součástí vygenerovaného slave stacku a je cyklicky volaná buď v hlavní smyčce, nebo v případě zapnuté synchronizace z přerušení.

```
282 \brief    This function will called from the synchronisation ISR
283           or from the mainloop if no synchronisation is supported
284 *////////////////////////////////////////////////////////////////////////////////////////
285 void process_app(TOBJ7000 *OUTPUTS, TOBJ6000 *INPUTS);
286
287 void APPL_Application(void)
288 {
289     process_app(&OUTPUTS0x7000, &INPUTS0x6000);
290 }
```

Obr. 4.5: Volání funkce process_app v cyklicky prohíhající funkci APPL_Application

Dále je zapotřebí kopírovat vstupy z lokální paměti do paměti ESC a výstupy naopak z paměti ESC do lokální paměti. To se provede voláním *memcpy* uvnitř generovaných funkcí *APPL_InputMapping* a *APPL_OutputMapping*. *APPL_InputMapping* kopíruje vstupní procesní data z lokální paměti do paměti ESC pro pozdější odeslání masteru. *APPL_OutputMapping* naopak kopíruje výstupní procesní data z paměti ESC do lokální paměti.

```
260 \brief      This function will copies the inputs from the local memory to the ESC memory
261             to the hardware
262 *////////////////////////////////////////////////////////////////////////////////////////
263 void APPL_InputMapping(UINT16* pData)
264 {
265     memcpy(pData, &(((UINT16 *)&INPUTS0x6000)[1]), SIZEOF(INPUTS0x6000)-2);
266 }

272 \brief    This function will copies the outputs from the ESC memory to the local memory
273           to the hardware
274 *////////////////////////////////////////////////////////////////////////////////////////
275 void APPL_OutputMapping(UINT16* pData)
276 {
277     memcpy(&(((UINT16 *)&OUTPUTS0x7000)[1]), pData, SIZEOF(OUTPUTS0x7000)-2);
278 }
```

Obr. 4.6: Kopírování procesních dat z lokální paměti do paměti ESC a naopak

Po zkompilování programu a jeho následném zapsání do flash paměti mikrokontroléru je implementace kompletní a je možné připojit zařízení do EtherCAT sítě. Aplikace byla kromě vývojového kitu zprovozněna také na IO modulu od firmy DFC Design.

4.4.1.2 Simple Open EtheCAT Slave

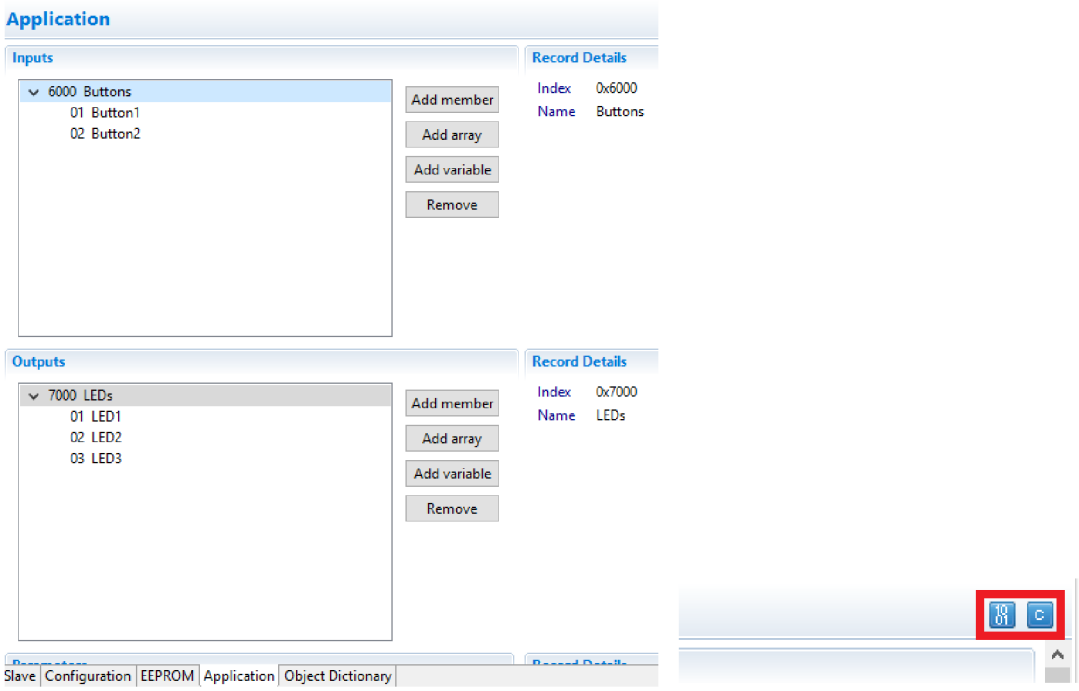
Projekt vychází z příkladů od firmy rt-labs. SOES byl do práce zahrnut, protože jeho zdrojové soubory jsou open source a je tudíž vhodný pro vyzkoušení sběrnice bez členství v ETG. Projekt se skládá ze dvou adresářů soes a xmc48_ecatslave. Složka soes obsahuje zdrojové kódy SOES použité k implementaci EtherCAT slave zařízení. Složka xcm48_ecatslave na druhou stranu obsahuje soubory realizující samotnou implementaci. [15]

EtherCAT SDK je nástroj generující kód pracující s knihovnami SOES. Stejně jako v případě SSC Tool v něm lze nadefinovat parametry zařízení spolu se vstupy a výstupy. Tento nástroj však potřeba nejprve nainstalovat do DAVE IDE. To se provede jednoduše kliknutím na "help" v horním panelu, výběrem možnosti "Install New Software" a následně "Add". Zobrazí se okno do kterého je možné vložit adresu, ze které je možné software nainstalovat. V tomto případě je zapotřebí vložit tento odkaz http://download.rt-labs.com/ethercat/sdk/updates a následně kliknout na možnost "install". K vygenerování souborů je zapotřebí platné licence, kterou je společnost ochotna pro studijní účely na jeden měsíc poskytnout.

Po spuštění programu lze ve spodní části vidět pět záložek:

- Slave
  Informace o výrobci, číslo produktu a revise.
- Configuration
  Nastavení FMMU, SM, DC (Distributed Clocks) a Mailbox komunikace.
- EEPROM
  Nastavení EEPROM paměti.
- Application
  Nastavení vstupních a výstupních dat. Je intuitivnější než u SSC Tool, neboť uživatel si vše může jednoduše navolit. Můžeme vidět na obr. 4.7.
- Object Dictionary
  Zde si může uživatel prohlédnout všechna nastavení ve formátu připomínající XML strom.

Na následujících obrázcích lze vidět nastavování vstupních a výstupní procesních dat a tlačíka pro generování ESI a slave stack kódu.



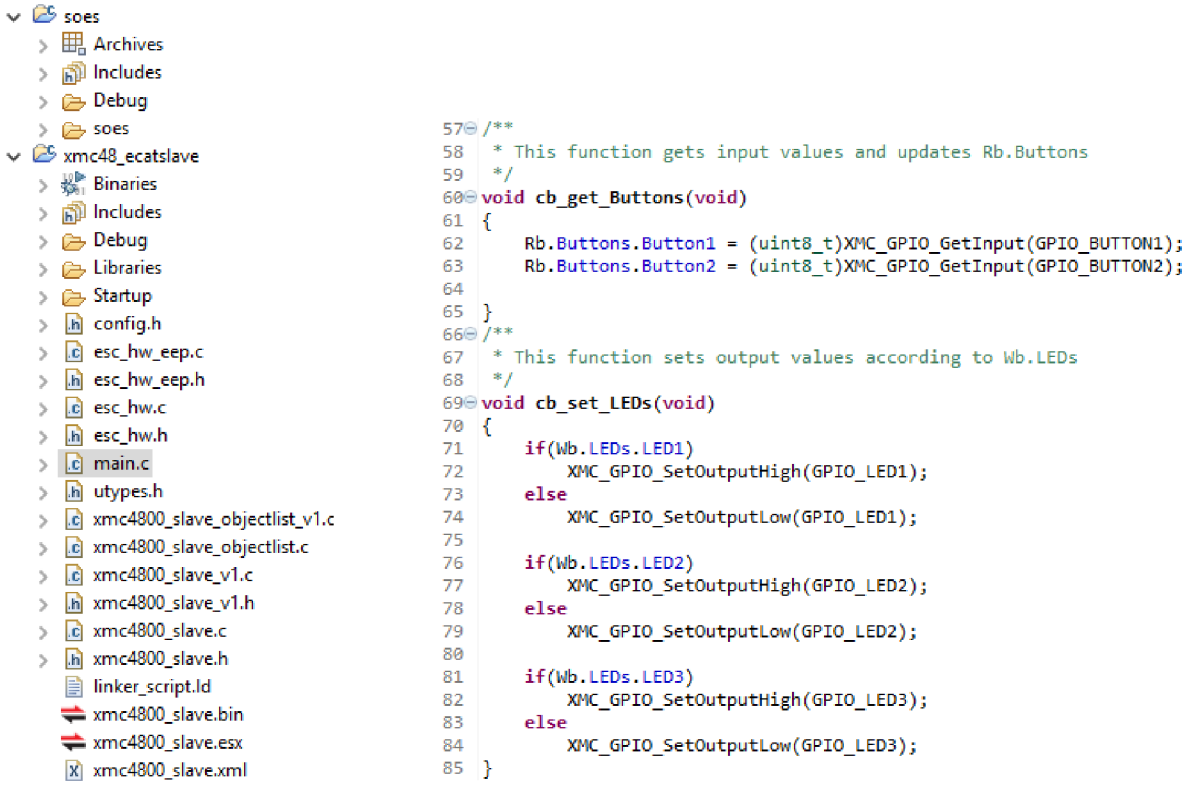
Obr. 4.7: Nastavení vstupů a výstupů v EtherCAT SDK (vlevo) a generování kódu a ESI (vpravo)

Po kliknutí na tlačítka generování kódu a ESI se v adresáři s aplikací vytvoří několik zdrojových souborů.

- **xmc4800_slave.c**
  Implementace EtherCAT slave používající knihovnu SOES
- **xmc4800_slave_objectlist.c**
  Definice použitých objektů včetně vstupních a výstupních procesních dat
- **xmc4800_slave.bin**
  binární obraz ESI souboru pro EEPROM paměť
- **xmc4800_slave.xml**
  ESI soubor s definicí daného zařízení

Mimo tyto jsou v adresáři ještě další soubory main.c, utypes.h a HAL (Hardware Abstraction Layer). V main.c jsou definovány callback funkce pro nastavení výstupů na základě přijatých dat a čtení vstupů z tlačítek. Kromě toho je zde funkce *main*, ve které je cyklicky volána funkce *ecat_slv* z dříve vygenerovaného souboru xmc4800_slave.c realizující EtherCAT slave. V utypes.h jsou definovány datové typy obsahující vstupní a výstupní data. Soubor xmc4800_slave_objectlist.c pak obsahuje objekty používané aplikací včetně vstupů a výstupů.

Další vygenerované soubory obsahující v názvu "_v1" jsou pro starou verzi SOES a při kompilaci jsou ignorovány.



Obr. 4.8: Struktura projektu využívající SOES (vlevo) a callback funkce (vpravo)

4.4.2 Firmware update

Řešení se skládá ze dvou samostatných projektů tvořící bootloader a aplikaci EtherCAT zařízení. Stejně jako v případě procesních dat je využito příkladů výrobce. [11]

Jak již bylo řečeno, EtherCAT nabízí protokol File Over EtherCAT, který umožňuje přenášet soubory po síti. Tohoto se využívá v tzv. bootstrap módu, ve kterém dochází k přenosu nového firmwaru. Do bootstrap stavu kterého se dá přejít pouze z init stavu, viz kapitola stavový automat.

4.4.2.1 Bootloader

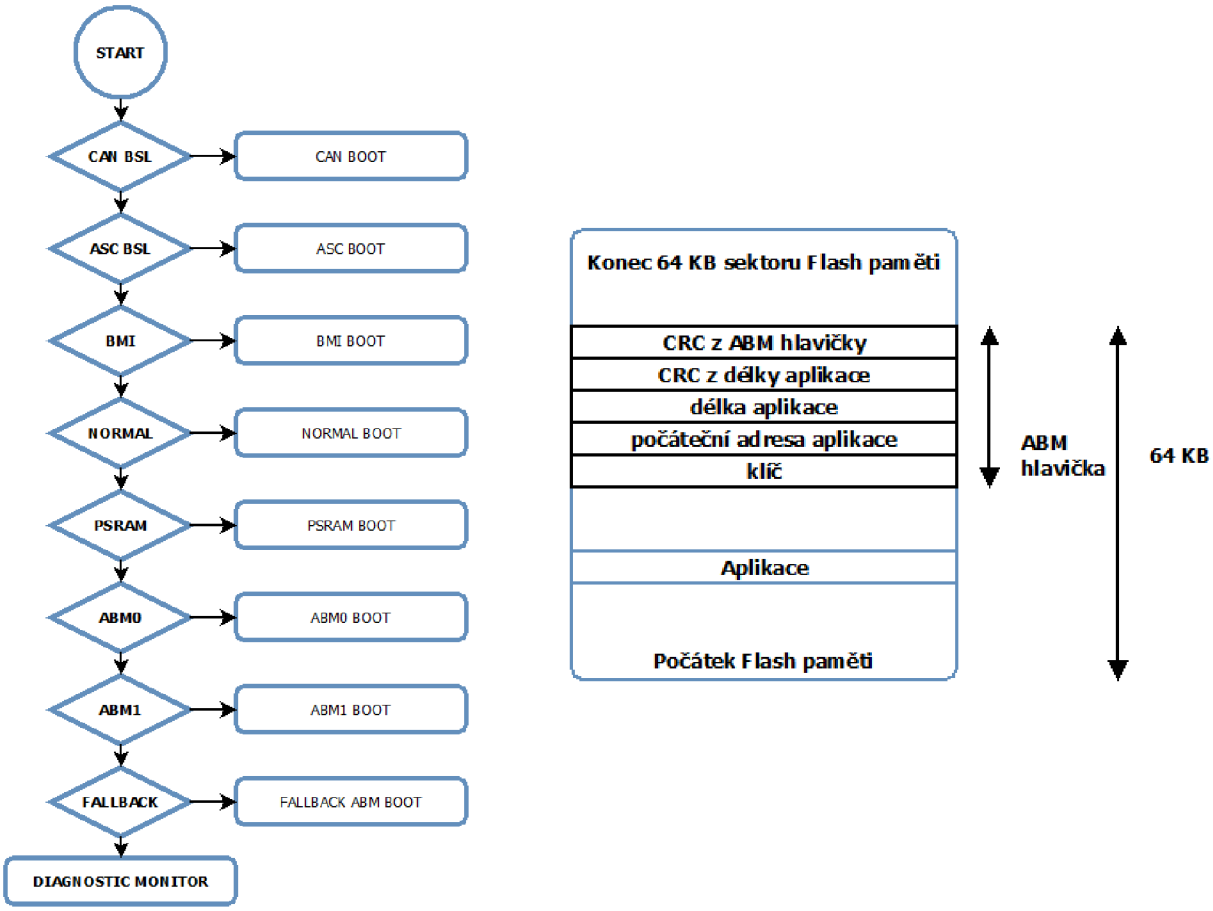
XMC 4800 umožňuje 8 bootovacích módů. Využity jsou dva a to „normal boot mode“ a „alternative boot mode“ (ABM). [9]

Normal boot mode

Aplikace je uložena na začátku flash paměti a začne se vykonávat po připojení napájení, pokud nejsou splněny podmínky pro boot mód s vyšší prioritou.

Alternative boot mode

V ABM módu aplikace začíná na adrese zapsané v ABM hlavičce, která se nachází na přesně určeném místě ve flash paměti. Toto místo je posledních 32 bytů prvního, nebo druhého 64 KB sektoru podle toho, jestli se jedná o ABM0 nebo ABM1. Následující obrázky ukazují prioritu režimů při bootu a uložení ABM hlavičky v paměti. ABM hlavička se skládá z klíče, počáteční adresy aplikace, délky aplikace a kontrolních součtů.



Obr. 4.9: Diagram ukazující prioritu při bootu (vlevo) a popis ABM hlavičky (vpravo)

Čip má k dispozici 2MB flash paměti přístupné ze dvou adresových oblastí (cachovaná od 0x08000000 a necachovaná od 0x0C000000). Bootloader je uložen na začátku paměti, odkud začne mikrokontrolér po resetu napájení vykonávat program. Jeho účelem je při přijetí nového firmwaru uloženého od adresy 0x0C100000 nahradit stávající na adrese 0x0C020000. Nejprve dojde ke kontrole, zda byl nový firmware přijat a nachází se v paměti V případě jeho nalezení se zkopíruje na místo starého a oblast, ve které byl uložen, je vymazána. Poté se nastaví ABM boot a provede software reset. Pokud nový firmware nalezen nebyl, přeskočí se kopírování a dojde rovnou k resetu.

4.4.2.2 Aplikace

EtherCAT aplikace zpracovává stejně jako v předchozím případě procesní data. Procesními daty jsou digitální výstupy připojené k LED diodám a vstupy signalizující stisknutí tlačítka na vývojové desce. Kromě toho však aplikace může v bootstrap módu přijmout také soubor, kterým je nový firmware v binární podobě. Nejprve je potřeba v SSC povolit několik funkcí.

- FOE zapsáním hodnoty 1 do FOE_SUPPORTED v sekci Mailbox.
- Bootstrap mód zapsáním hodnoty 1 do BOOTSTRAPMODE_SUPPORTED v sekci EtherCAT State Machine.

Po vygenerování kódu pak bude EtherCAT State Machine obsahovat také bootstrap mode, který by bez jeho povolení ve stavovém automatu chyběl. Aplikace realizující update fimwaru se skládá ze čtyř funkcí.

```
50 void FWUPDATE_StartDownload(void);
51 void FWUPDATE_StateTransitionInit(void);
52 uint8_t FWUPDATE_GetDownloadFinished(void);
53 uint16_t FWUPDATE_Data(uint16_t *pdata, uint16_t size);
```

Obr. 4.10: Funkce realizující příjem nového firmwaru

FWUPDATE_StartDownload

Je volaná z funkce *FOE_Write*. Nastavuje proměnnou *g_firmware_download_started* do jedničky a symbolizuje tím, že přenos už začal a při dalším volání této funkce vrátí chybu. Zároveň také inicializuje zápis do flash paměti funkcí *FLASHPROG_Init* předáním adresy zápisu a maximální velikosti zapisovaného souboru.

FWUPDATE_Data

Realizuje zapisování nového firmwaru do paměti prostřednictvím funkce *FLASHPROG_Data*. V případě zapsání poslední části firmwaru zkontroluje kontrolní součet, který se nachází v posledních 4 bytech a zapíše délku programu do hlavičky s meta daty. V případě, že všechno proběhlo v pořádku, ukončí se přenos a čeká se na návrat do init stavu.

FWUPDATE_StateTransitionInit

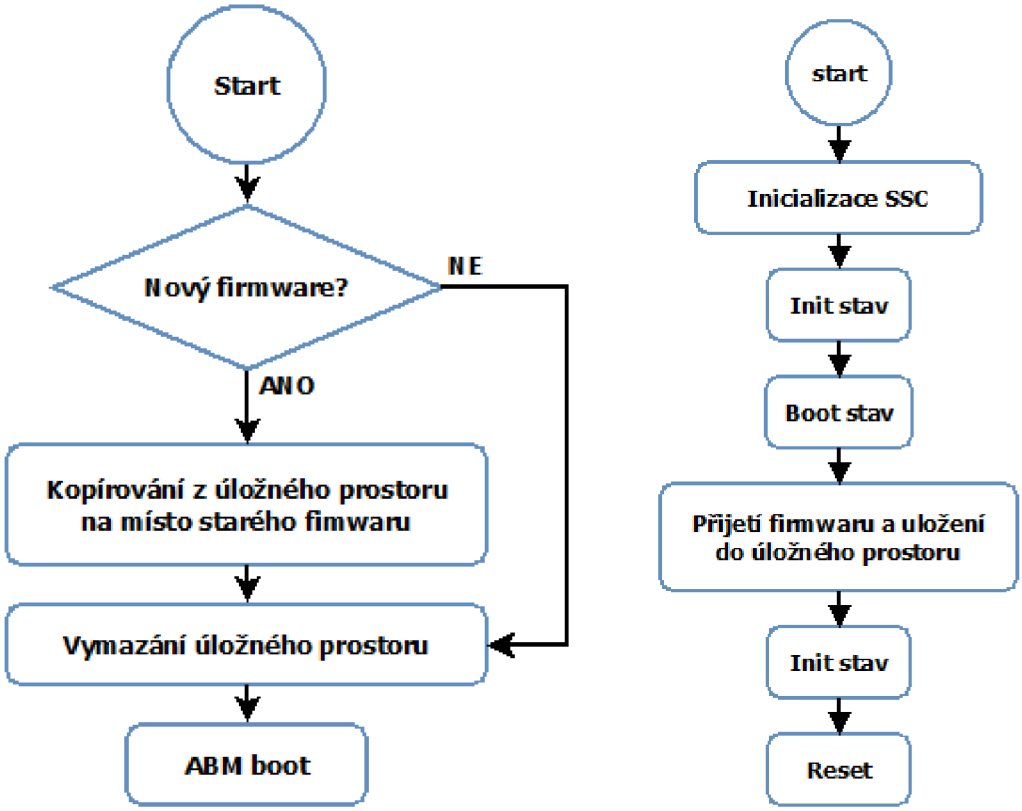
Je volána ve funkci *APPL_StopMailboxHandler*, která je dále volána při přechodu z PREOP nebo BOOT stavu do init stavu. Ve funkci je nastavována proměnná *g_firmware_download_finished* do jedničky. Tato proměnná symbolizuje, že byl přenos firmwaru ukončen a je zapsán v určené oblasti flash paměti.

FWUPDATE_GetDownloadFinished

Pouze vrací hodnotu proměnné *g_firmware_download_finished*.

Při vstupu do bootstrap módu může master začít přenos firmwaru. Na jeho začátku se volá funkce *FWUPDATE_StartDownload* inicializující zápis do flash paměti. V průběhu přenosu je

volána funkce *FWUPDATE_Data*, která na základě dříve provedené inicializace zapisuje přijatá data do zálohového oddílu flash paměti. *FWUPDATE_StateTransitionInit* je volána v případě přijetí souboru a následného přechodu do init stavu. Ve funkci *process_app* (viz předchozí podkapitola) je poté iniciován software reset. Obrázek níže zobrazuje stavový diagram pro aktualizaci firmwaru.



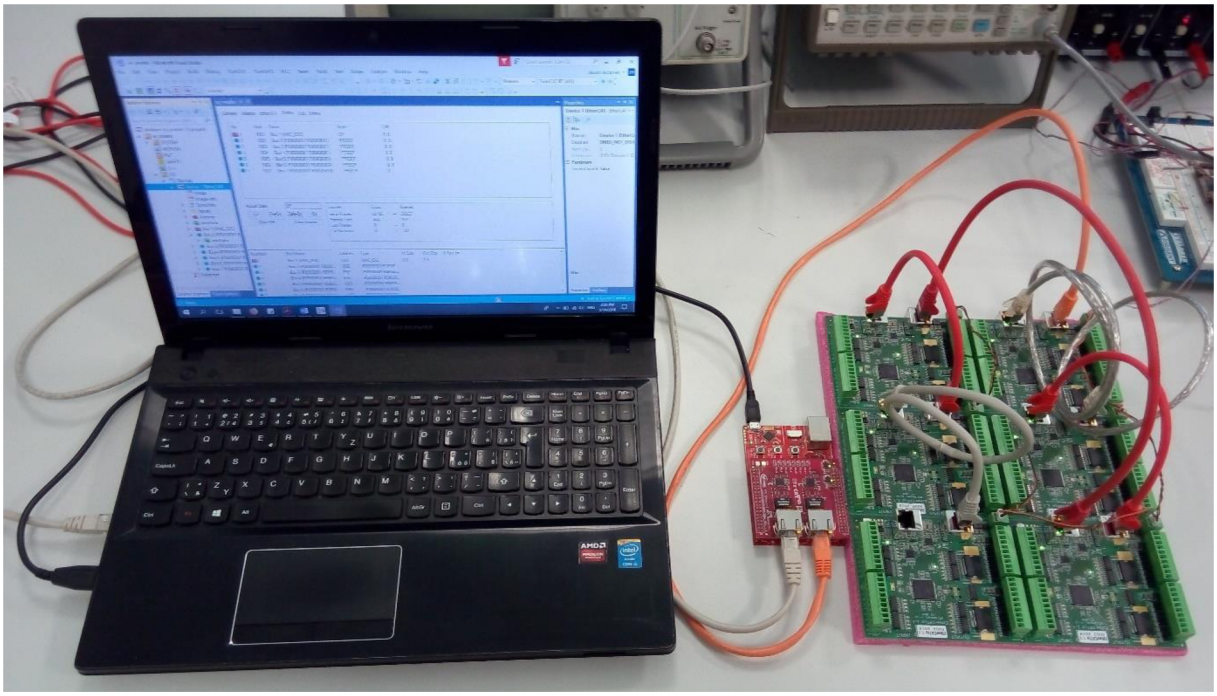
Obr. 4.11: Diagramy popisující princip bootloaderu (vlevo) a aktualizaci firmwaru (vpravo)

Mimo úpravy SSC musí být změněn také tzv. „linker script“, který určuje jak a kam v paměti bude program zapsán.

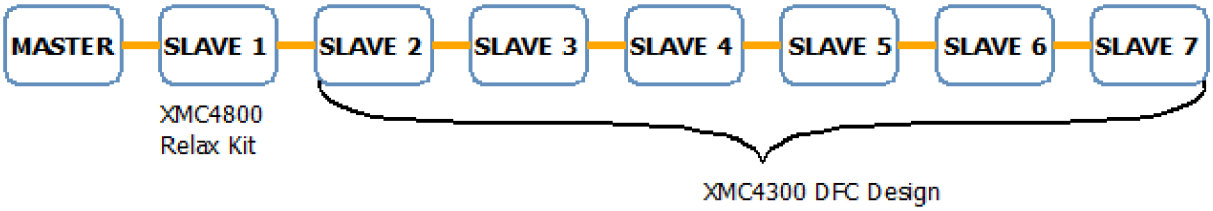
Kromě vývojového kitu s XMC4800 proběhl pokus o implementaci aktualizace firmwaru také do IO modulu od firmy DFC Design, který je osazen mikrokontrolérem XMC4300. Implementace se bohužel nezdařila z důvodu malé flash paměti (256 KB) vzhledem k velikosti kódu (cca 130 KB). [12]

5. JITTER A ZPOŽDĚNÍ

V této kapitole budou prezentovány výsledky měření zpoždění a periodicity v síti. Za účelem měření mi bylo zapůjčeno 6 IO modulů od firmy DFC Design. Následný obrázek a schéma ukazuje zapojenou EtherCAT síť, v níž je masterem notebook se 100 Mbit síťovou kartou. Měření probíhalo na výsutpním pinu mikrokontroléru XMC4800 a výstupech IO modulu. Zpoždění modulu může být tedy lehce ovlivněno obvody zapojenými výstupními obvody.



Obr. 5.1: Soustava, na které probíhalo měření zpoždění a jitteru

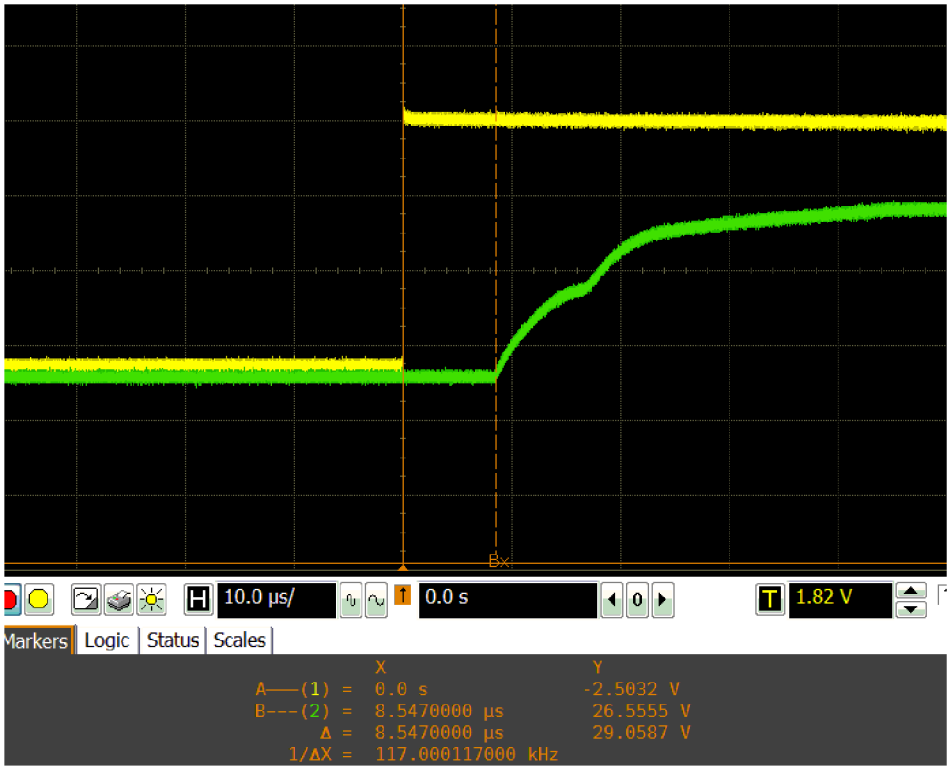


Obr. 5.2: Diagram soustavy, na které probíhalo měření zpoždění a jitteru

5.1 Zpoždění výstupů mezi dvěma zařízeními

Na obr. 5.3 lze vidět zpoždění mezi dvěma slave zařízeními, které jsou zapojeny v síti o celkovém počtu sedmi zařízení. První (žlutý) je vývojový kit použitý v této práci a druhý (zelený) je vstupně výstupní modul (IO modul) vyvinutý firmou DFC Design. Tento modul

využívá na rozdíl od vývojového kitu synchronizace za pomocí distribuovaných hodin. Zpoždění mezi výstupy je tedy dáno časovým rozdílem mezi prvním a posledním zapojeným zařízením, neboť modul čeká až dostane informace o výstupech i poslední slave v síti.



Obr. 5.3: Zpoždění zařízení bez zapnuté synchronizace (žlutý) a zařízení se synchronizací (zelený)

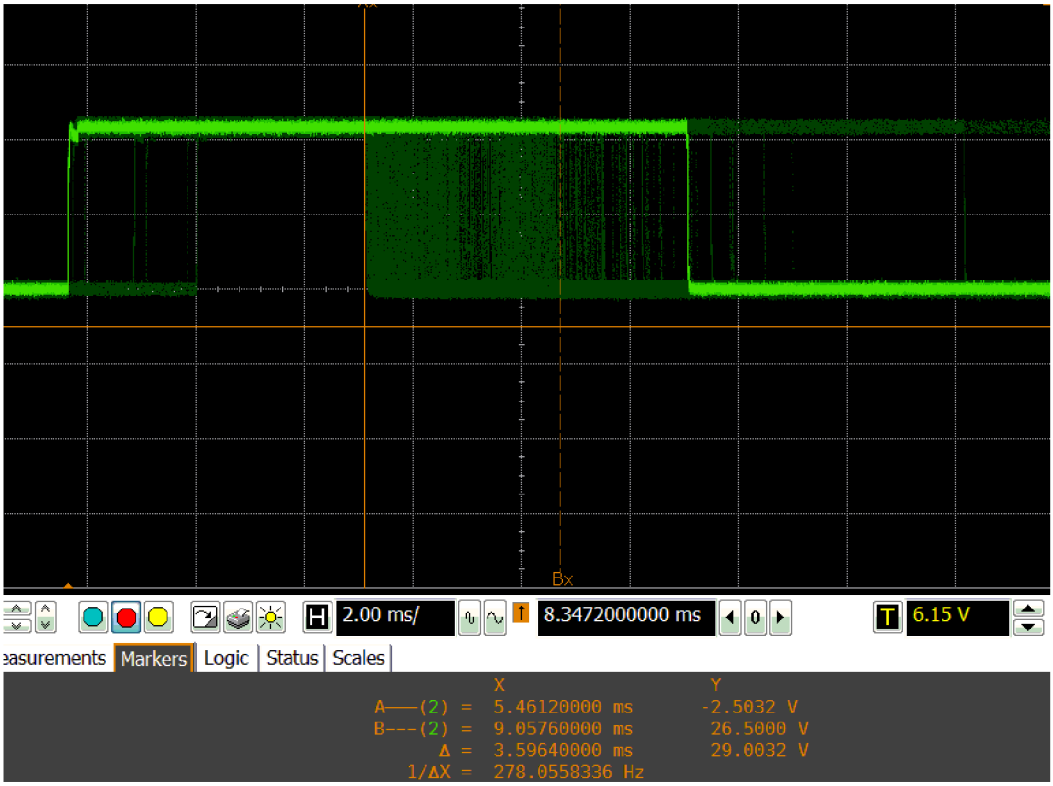
5.2 Jitter

Jitter je odchylka od periodicity signálu. V EtherCAT síti jej můžeme rozdělit na odchylku způsobenou masterem a odchylku způsobenou slavem. Perioda jitteru je rozpětí mezi nejdřívější a nejpozdější měřenou událostí vzhledem k jiné události. Tento parametr se často objevuje ve spojitosti s Ethernetem. Pro představu jím může být například časové rozmezí, ve kterém zařízení v LAN (Local Area Network) síti obdrží zprávu od jiného zařízení v téže síti.

5.2.1 Master

Jitter způsobený masterem je dán mírou determinismu operačního systému zařízení, na kterém master běží. V tomto případě se jedná o notebook s operačním systémem Windows, ve kterém je spuštěno virtuální zařízení s OS Linux Mint. Vzhledem k tomu, že aplikace SOEM je aplikace spustitelná pouze z uživatelského prostoru systému, je očividné, že master bude dosti nederministický. Měření bylo provedeno při uvedení logické jedničky na dobu třiceti cyklů z důvodu lepší viditelnosti výsledků, jitter je tudíž akumulovaný. Na následujícím obrázku jsou

k vidění jeho výsledky. Nejhustěji byl jitter rozložen v rozsahu přibližně 3,6 ms. Je potřeba uvést, že toto měření je pouze orientační a nemá kromě ilustrace velkého nedeterminismu žádný přínos.

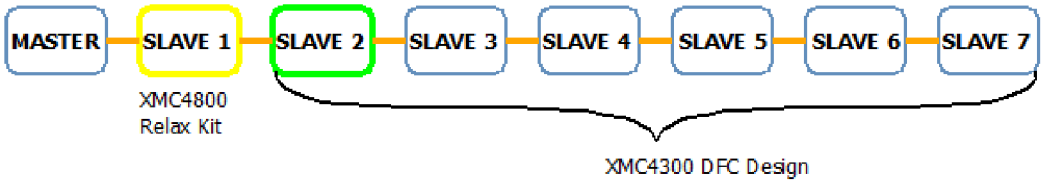


Obr. 5.4: Orientační jitter způsobený masterem

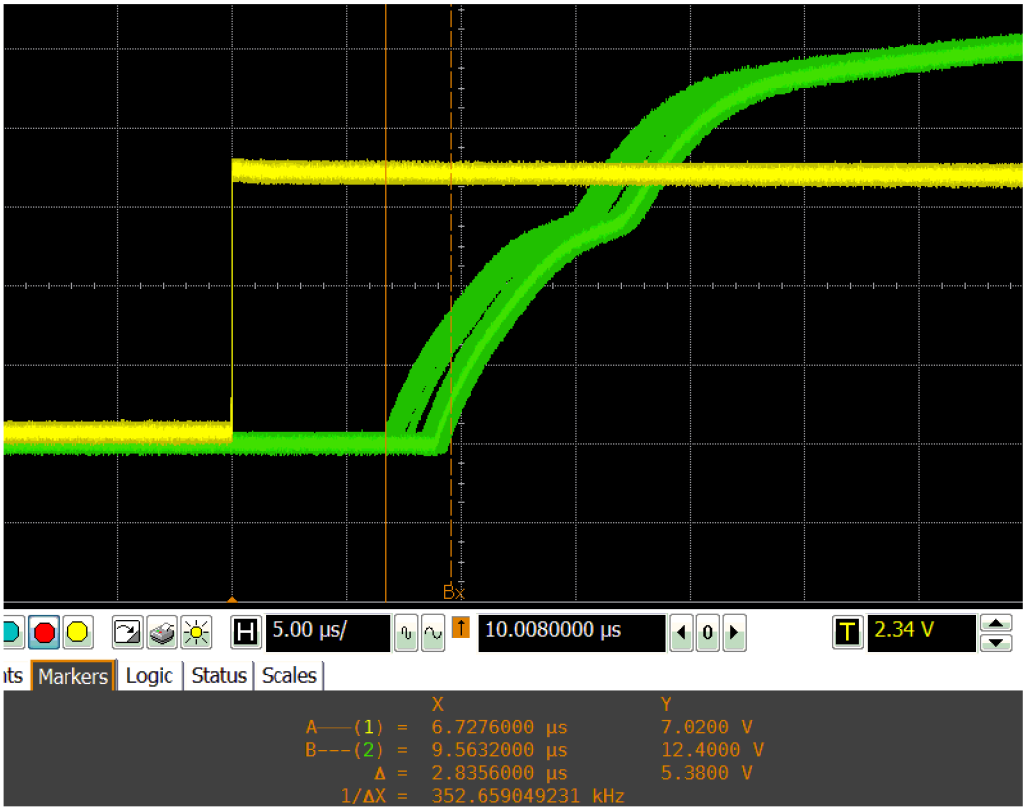
5.2.2 Slave

Jitter způsobený slave zařízením je dán nepřesností hardwaru. Dva stejné mikrokontroléry mohou vlivem výrobních nepřesností generovat rozdílné zpoždění, které se poté v síti dokáže kumulovat. Dalším faktorem je softwarové řešení, které ovšem může mít v případě využití přerušení a synchronizace téměř zanedbatelný vliv.

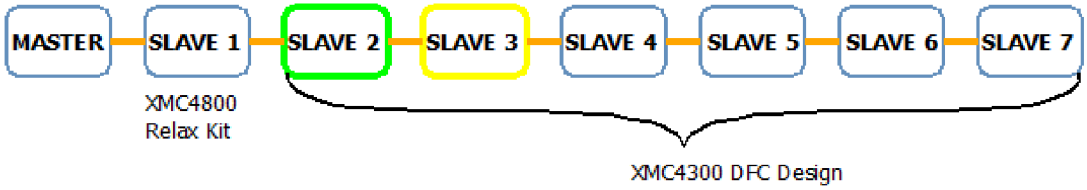
Změřena byla periodičnost zařízení bez synchnronizace a se synchronizací. Bez synchronizace zařízení byl změřen jitter v rozsahu přibližně 2,8 µs a se synchronizací 1,3 µs. Výsledky měření lze vidět na následující obrázcích.



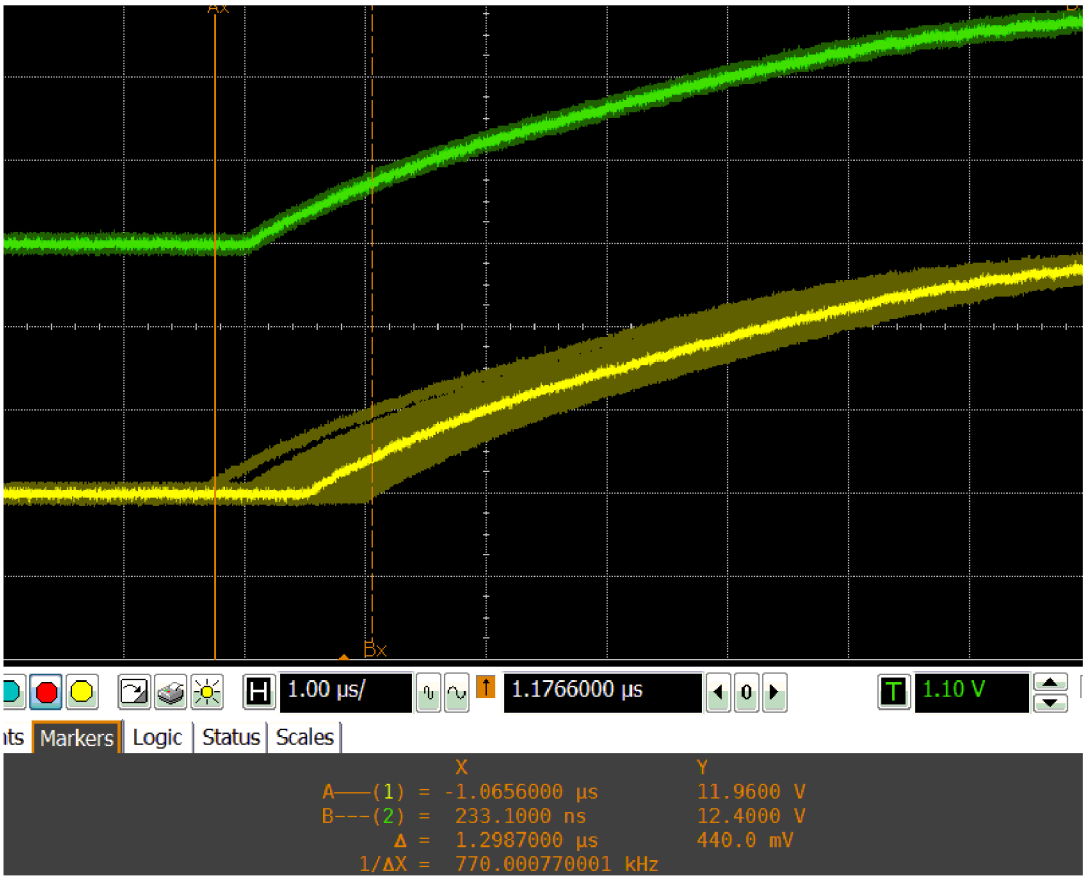
Obr. 5.5: Zapojení měření jitteru zařízení se synchronizací vzhledem k zařízení bez synchronizace



Obr. 5.6: Měření jitteru zařízení se synchnronizací (zelený) vzhledem k zařízení bez synchronizace (žlutý)



Obr. 5.7: Zapojení měření jitteru dvou zařízení se synchronizací



Obr. 5.8: Měření jitteru dvou zařízení se synchronizací

6. POROVNÁNÍ S 1GBIT

EtherCAT dnes využívá jako fyzickou vrstvu standardy 100BASE-TX a 100BASE-FX s přenosovou rychlostí 100 Mbit/s (Fast Ethernet). Není možné využít rychlejšího Gigabit Ethernetu z důvodu nekompatibility s ESC jádrem. Jediná možnost, jak využít vyšší přenosové rychlosti, je implementovat funcki ESC do FPGA (Field Programmable Gate Array). Přesto si můžeme naznačit, jaké výhody a nevýhody by mělo použití Gigabitu.

Maximální délka Ethernet rámce je 1526 bytů. Z toho můžeme jednoduše spočítat, že doba potřebná pro přenesení jednoho takového rámce je 122,08 µs při použití Fast Ethernetu a 12,208 µs při použití Gigabit Ethernetu. [1]

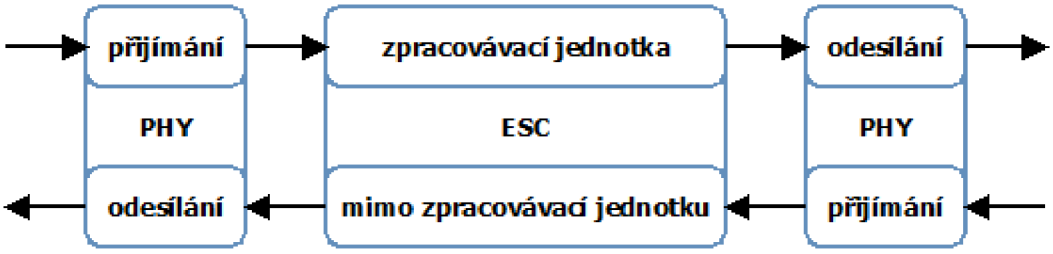
$$\text{doba přenosu packetu při rychlosti 100 Mbps} = \frac{1526 \cdot 8}{10^8} = 122,08\ \mu s$$

$$\text{doba přenosu packetu při rychlosti 1000 Mbps} = \frac{1526 \cdot 8}{10^9} = 12{,}208\ \mu s$$

Při použití Gigabit Ethernetu by tedy doba přenosu jednoho rámce klesla na desetinu. To se ovšem nedá říct o zpoždění v EtherCAT síti, neboť to je dáno také použitým hardwarem a přenosovým médem. Propagační zpoždění přenosového media můžeme vzhledem k jeho délce v tomto případě zanedbat.

Zpoždění hardwaru můžeme rozdělit na zpoždění způsobené:

- Průchodem packetu přes „zpracovávací jednotku“ ESC
- Průchodem packetu přes ESC mimo „zpracovávací jednotku“
- PHY při odesílání
- PHY při přijímání



Obr. 6.1: Průchod packetu přes ESC se dvěma porty a PHY

Infineon XMC4800 s integrovaným ESC používá stejné jádro jako slave kontroléry od firmy Beckhoff. Z tohoto důvodu můžeme říci, že je zpoždění ESC v průměru 305 ns při průchodu zpracovávací jednotkou a 265 ns při průchodu mimo zpracovávací jednotku. [3]

Zpoždění způsobené PHY se liší podle výrobce, přičemž ne každý tyto údaje zveřejňuje v datasheetu k součástce. Vybral jsem tedy dva PHY (100 Mbit a 1000 Mbit) od firmy Texas

Instruments k porovnání doby průchodu při odesílání a přijímání. Zpoždění jsou zobrazena v tabulce. [17][18]

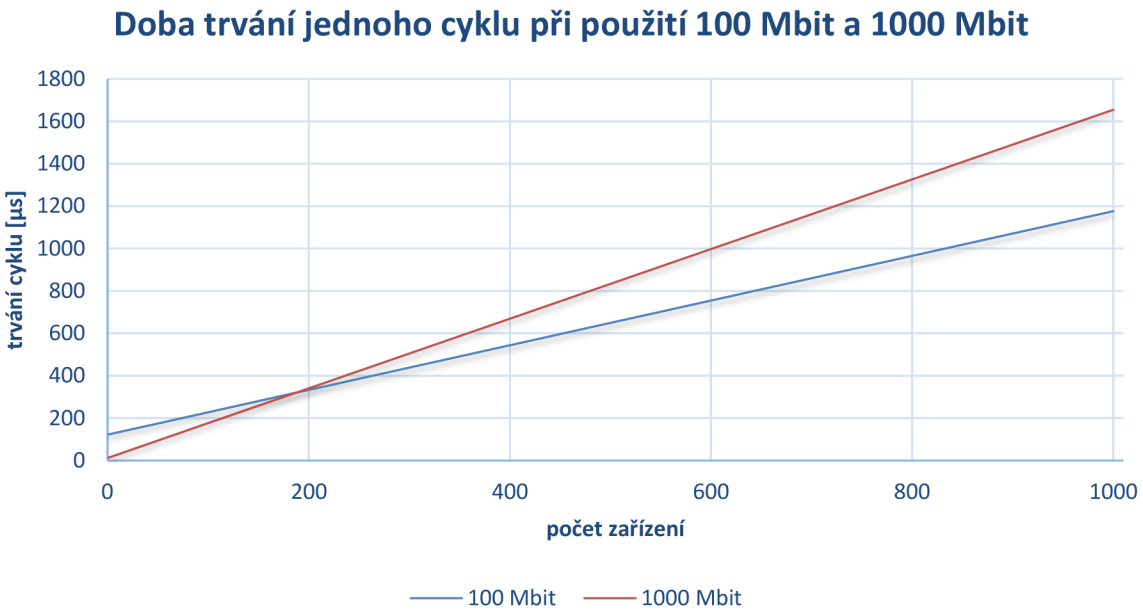
Tab. 1: Zpoždění hardwaru

	Fast Ethernet	Gigabit Ethernet
Zpoždění ESC přes "processing unit" [ns]	305	
Zpoždění ESC mimo "processing unit" [ns]	265	
Zpoždění PHY při odesílání [ns]	48	152
Zpoždění PHY při přijímání [ns]	194	384
Zpoždění zařízení se dvěma porty [µs]	1054	1642
Zpoždění zařízení se třemi porty [µs]	1878	3330
Doba přenosu rámce délky 1526 Bytů [µs]	122,08	12,208

Z těchto hodnot lze na první pohled vidět, že použití Gigabit Ethernetu je výhodné jen částečně. Sníží sice dobu přenosu jednoho rámce na desetinu, ale také výrazně zvýší zpoždění způsobené PHY na více než dvojnásobek. Z toho vyplývá, že při určitém počtu zařízení v síti budou zpoždění při použití Fast Ethernetu a Gigabit Ethernetu stejná a pro ještě vyšší počet bude Fast Ethernet dokonce výhodnější. Celkové zpoždění jednoho slave zařízení se dvěma porty je pak součtem zpoždění daných průchodem ESC a dvěma PHY. V následující tabulce a grafu lze vidět, jak velké zpoždění generuje hardware a jak se s rostoucím počtem zařízení mění trvání jednoho cyklu.

Tab. 2: Porovnání zpoždění Fast Ethernetu a Gigabit Ethernetu

	Fast Ethernet		Gigabit Ethernet	
počet zařízení	Δt [µs]	T [µs]	Δt [µs]	T [µs]
1	1,054	123,134	1,642	13,85
10	10,54	132,62	16,42	28,628
186	196,044	318,124	305,412	317,62
187	197,098	319,178	307,054	319,262
100	105,4	227,48	164,2	176,408
1000	1054	1176,08	1642	1654,208



Obr. 6.2: Závislost doby trvání cyklu při použití 100 Mbit a 1000 Mbit na počtu zařízení

7. ZÁVĚR

V této práci jsem se zabýval popisem protokolu EtherCAT a jeho následnou implementací do ARM mikrokontroléru XCM4800 od firmy Infineon, osazeného na vývojové desce Relax Kit.

V první části práce lze najít popis protokolu EtherCAT, jeho výhod a definici požadavků pro realizaci master a slave zařízení. Je zde popsán obecný princip sběrnice, adresování v síti, protokol, princip synchronizace a fyzická vrstva, na které je EtherCAT schopen fungovat.

Pro otestování komunikace byly použity dvě master aplikace. První je SOEM, která je open source od Open EtherCAT Society a dostupná z jejich GitHub repozitáře. K dispozici jsou jednoduše použitelné demo aplikace pro čtení z EEPROM paměti, zjišťování informací o síti a cyklické odesílání procesních dat. Aplikace simple_test cyklicky odesílající procesní data byla použita z notebooku ve virtuálním zařízení s operačním systémem Linux Mint. Kromě SOEM byl použit také TwinCAT od firmy Beckhoff. Tento software je k dispozici plně funkční ve formě časově omezeného dema. V této práci byl použitý pro otestování aktualizace firmwaru vývojového kitu po EtherCAT síti poté, co selhal pokus o použití SOEM.

Cílem práce bylo implementovat protokol EtherCAT do mikroprocesoru. Zadání bylo splněno zprovozněním aplikace realizující vstupy a výstupy na vývojovém kitu. Kromě toho byla aplikace implementována také do zapůjčeného IO modulu od firmy DFC Design. Důležitým krokem bylo vygenerovat zdrojové kódy slave stacku a ESI soubor definující EtherCAT slave. Použitým nástrojem je SSC Tool od firmy Beckhoff, který je k dispozici stejně jako velká část dokumentace jen členům registrovaným u ETG. Velké poděkování proto patří vedoucímu práce Ing. Soběslavu Valachovi za ochotu mi tento software poskytnout. Kromě toho byl vyzkoušen také EtherCAT SDK od rt-labs, k jehož použití je taktéž potřeba licence. Tu jsem po požádání získal na dobu jednoho měsíce, po jehož uplynutí muselo být o její poskytnutí znova zažádáno. Kromě aplikace realizující digitální vstupy a výstupy byla také zprovozněna aplikace umožňující aktualizaci firmwaru přes EtherCAT síť. Tu jsem se pokusil impementovat také do firmwaru k zapůjčenému IO modulu, který však používá mikrokontrolér XMC4300 s menší velikostí flash paměti (256 KB). Z toho důvodu se vzhledem k velikosti kódu (cca 130 KB) implementace nepodařila.

Následně bylo provedeno měření pro zjištění periodicity a jednotlivých zpoždění v síti se sedmi slave zařízeními. Tato zařízení mi byla dána k dispozici vedoucím práce k ověření funkcionality a výkonu. Měřením jsem zjistil, že master nevyhovuje požadavkům na real-time. To se ovšem vzhledem k použití SOEM spuštěného jako uživatelská aplikace v operačním systému nedalo očekávat. Dále jsem změřil jitter a zpoždění mezi jednotlivými zařízeními. Vývojový kit nemá implementovanou synchronizaci, což jde vidět na obr. 5.3, kde je mezi jeho výstupem a výstupem následujícího zařízení zpoždění asi 8 µs. To odpovídá akumulovanému zpoždění všech zařízení v síti, které se projeví díky jejich synchronizaci už u druhého modulu. Změřený jitter bez použití synchronizace nabývá hodnoty 2,8 µs. Byl změřen také jitter dvou synchronizovaných zařízení, který je menší a má hodnotu 1,3 µs.

Na závěr byla provedena úvaha nad použitím standardu Gigabit Ethernet s rychlostí přenosu 1000 Mbit/s místo stávajícího Fast Ethernetu s rychlostí 100 Mbit/s. Zjistil jsem, že použití Gigabit Ethernetu by bylo výhodné jen do počtu 186 zařízení při využití maximální velikosti ethernetového rámce. Toto číslo je pouze orientační na základě vypočítaných zpoždění a doby průchodu jednoho rámce sítí. Rámec je při této rychlosti sice přenesen za desetinu času, který potřebuje Fast Ethernet, avšak zpoždění způsobené PHY se zvýší na téměř dvojnásobek. Toto zpoždění by se zvyšovalo s každým připojeným zařízením, až by nakonec vyrovnalo čas získaný zvýšením přenosové rychlosti.

Výsledkem práce je funkční slave realizující digitální vstupy a výstupy, kterému je možné po síti aktualizovat firmware. Tím se dají v praxi snížit případné náklady a zredukovat práci potřebnou pro aktualizaci většího počtu zařízení.

LITERATURA

[1] Analog Devices. A Beginner's Guide to Ethernet 802.3 [online]. 2005 [cit. 2018-5-3]. http://www.analog.com/media/en/technical-documentation/application-notes/EE-269.pdf

[2] Beckhoff. EtherCAT Slave Controller Hardware Data Sheet Section I [online]. 2014 [cit. 2018-5-3]. https://download.beckhoff.com/download/document/io/EtherCAT-development-products/EtherCAT_esc_datasheet_sec1_technology_2i2.pdf

[3] Beckhoff. EtherCAT Slave Controller Hardware Data Sheet Section III [online]. 2014 [cit. 2018-5-3]. https://download.beckhoff.com/download/document/io/EtherCAT-development-products/EtherCAT_et1100_datasheet_v1i9.pdf

[4] Beckhoff. EtherCAT Slave Implementation Guide [online]. 2012 [cit. 2018-5-3]. http://www.ethercat.org/pdf/english/ETG2200_V2i0i0_SlaveImplementationGuide.pdf

[5] Beckhoff. Application Note ET9300 (EtherCAT Slave Stack Code) [online]. 2017 [cit. 2018-5-3]. https://download.beckhoff.com/download/document/io/ethercat-development-products/an_et9300_v1i8.pdf

[6] Beckhoff. EtherCAT Introduction [online prezentace]. 2012 [cit. 2018-5-3]. https://www.ethercat.org/pdf/english/EtherCAT_Introduction_0905.pdf

[7] Broadcom. 10/100BASE-TX SINGLE-CHANNEL TRANSCEIVER [online]. 2006 [cit. 2018-5-3]. https://datasheet.octopart.com/BCM5241XA1KMLG-Broadcom-datasheet-11803361.pdf

[8] EtherLAB. EtherLAB[online]. [cit. 2018-5-3]. https://www.etherlab.org/download/flyer.pdf

[9] Infineon. XMC 4800 Reference Manual [online]. 2015 [cit. 2018-5-3]. https://www.infineon.com/dgdl/Infineon-xmc4800-UM-v01_00-EN.pdf?fileId=5546d4624d6fc3d5014ddd85f6d97832

[10] Infineon. XMC4800 EtherCAT APP SSC Slave Example [online]. [cit. 2018-5-3]. https://www.infineon.com/dgdl/Infineon-Getting-Started_XMC4800_Relax_EtherCat_APP_Slave_SSC-GS-v02_02-EN.pdf?fileId=5546d4625a888733015a938b1c0a6274

[11] Infineon. XMC4800 EtherCAT APP SSC Firmware Update Slave Example [online]. [cit. 2018-5-3]. https://www.infineon.com/cms/en/product/promopages/aim-mc/dave_downloads.html

[12] Infineon. XMC 4300 Reference Manual [online]. 2016 [cit. 2018-5-3]. https://www.infineon.com/dgdl/Infineon-XMC4300_ReferenceManual_v1.1.pdf-UM-v01_01-EN.pdf?fileId=5546d462525dbac40152f95e61ad6aff

[13] Open EtherCAT Society. Simple Open EtherCAT Master [online]. 2018 [cit. 2018-5-3]. https://github.com/OpenEtherCATsociety/SOEM

[14] Open EtherCAT Society. Simple Open EtherCAT Slave [online]. 2018 [cit. 2018-5-3]. https://github.com/OpenEtherCATsociety/SOES

[15] rt-labs. SOES EtherCAT Slave Example Getting Started with EtherCAT and Infineon XMC4 [online]. [cit. 2018-5-3]. https://rt-labs.se/content/uploads/EtherCAT_SDK_DAVE.pdf

[16] Rt-labs. Rt-labs [online]. 2018 [cit. 2018-5-3]. https://rt-labs.com/refman/ethercat-sdk/index.php

[17] Texas Instruments. DP83865 Gig PHYTER V 10/100/1000 Ethernet Physical Layer [online]. 2004 [cit. 2018-5-3]. http://www.ti.com/lit/ds/symlink/dp83865.pdf

[18] Texas Instruments. DP83822 Robust, Low Power 10/100 Mbps Ethernet Physical Layer Transceiver [online]. 2018 [cit. 2018-5-3]. http://www.ti.com/lit/ds/symlink/dp83822hf.pdf

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ABM	Alternative Boot Mode
CoE	Can over EtherCAT
DC	Distributed Clocks
DL	Data Link Layer
DPRAM	Dual Port Random Access Memory
EEPROM	Electrical Erasable Programmable Read Only Memory
EoE	Ethernet over EtherCAT
ESC	EtherCAT Slave Controller
ESI	EtherCAT Slave Information
ETG	EtherCAT Technology Group
EtherCAT	Ethernet for Control Automation Technology
FMMU	Field Memory Management Unit
FoE	File over EtherCAT
FPGA	Field Programmable Gate Array
IO	Input Output
LAN	Large Area Network
MCU	Microcontroller Unit
PC	Personal Computer
PDO	Process Data Object
PDI	Process Data Interface
SDK	Software Development Kit
SERCOS	SErial Real time COmmunication Specification
SM	SyncManager
SOEM	Simple Open EtherCAT Master
SOES	Simple Open EtherCAT Slave
SoE	Servo profile over EtherCAT
SSC	Slave Stack Code
WKC	Working Counter

SEZNAM PŘÍLOH