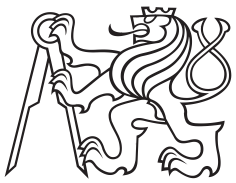
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra Řídící techniky

# Portace real-time systému RTEMS na rodinu mikrokontrolérů TMS570

Přemysl Houdek

Leden 2016

Vedoucí práce: Ing. Pavel Píša Ph.D.

# Poděkování / Prohlášení

Mé poděkování patří panu Ing. Pavlu Píšovi, Ph.D. za odborné vedení diplomové práce, ochotu a trpělivost a také za cenné rady, připomínky a čas bez kterého by se tato práce nedala zkompletovat.

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 11.1.2016

.....

# Abstrakt / Abstract

Tato diplomová práce si klade za cíl navrhnout, implementovat a začlenit podporu bezpečnostního mikrokontroleru TMS570 od firmy Texas Instruments do operačního systému RTEMS. A dále integrovat do výsledného projektu síťovou knihovnu LwIP.

Výsledná implementace byla integrována do hlavního vývojového stromu projektu RTEMS. Síťová komunikace pracující s podporou knihovny LwIP operuje stabilně.

**Klíčová slova:** RTEMS, LWIP, portace, ETHERNET, ARM, generování hlavičkových souborů

This master theasis aims to design, implement and integrate safety microcontroler TMS570 support for real time operating system RTEMS. ETHERNET LwIP library should be integrated into final project.

The implementation has been accepted into RTEMS's mainline. Network communucation is running without crashes.

**Keywords:** RTEMS, LWIP, porting, ETHERNET, ARM, header files generation

# Obsah /

# Obrázky

# Kapitola 1
# Úvod

Tato diplomová práce si klade za cíl navrhnout, implementovat a začlenit podporu mikrokontroleru TMS570 od firmy Texas Instruments do operačního systému RTEMS. [1]

Do operačního systému je potřeba začlenit obsluhu základních periferii. Pro zprovoznění operačního systému RTEMS na nové hardwarové platformě je nutná minimálně implementace obsluhy periferie časovače a implementace obsluhy sériových portů. Aby mohl být výsledný implementovaný ovladač použit a začleněn do hlavního vývojového stromu projektu RTEMS, musí ovladač odpovídat požadavkům komunity na čitelnost a správnost formátování. Pro zvýšení uplatnitelnosti podpory nové rodiny mikrokontrolérů je taktéž potřeba vybavit portaci hlavičkovými soubory, které jsou nutné pro využití v dalších perifériích mikrokontroléru TMS570. K začlenění hlavičkových souborů popisujících periferie a jádro mikrokontroléru TMS570 je nutné, aby hlavičkové soubory odpovídaly zvyklostem projektu RTEMS jak licencí, tak formou. Dalším úkolem je vytvoření podpory síťového rozhraní ETHERNET pro mikrokontrolér TMS570 s operačním systémem RTEMS. V neposlední řadě je důležité výsledný implementovaný kód otestovat na vývojovém kitu TMS570LS31xHDK a důsledně zdokumentovat.

Mikrokontrolér TMS570 od firmy Texas Instruments patří do rodiny bezpečnostních mikrokontrolérů Hercules. Mikrokontrolér byl vyvinut, aby splňoval bezpečnostní normy ISO 26262 ASIL D a IEC 61508 SIL 3. Proto je mikrokontrolér TMS570 doporučen pro použití v automobilovém, železničním a leteckém průmyslu. Hlavní motivací této diplomové práce je snížit čas potřebný k tvorbě nové aplikace a zároveň rozšířit aplikační spektrum mikrokontroléru. Dosažení zmíněného cíle je předpokládáno rozšířením operačního systému RTEMS o ovladače a podpůrný kód (BSP) pro mikrokontrolér TMS570. Operační systém RTEMS byl zvolen z důvodu jeho vhodnosti pro použití v bezpečnostně kritických aplikacích jako je vesmírný a medicínský průmysl.
Text této diplomové práce lze použít pro základní seznámení s mikrokontrolérem TMS570. Dále zde vývojář může naleznout návod, jak přidat podporu pro nový mikrokontrolér do operačního systému RTEMS nebo jak napsat ovladač do knihovny LwIP a jakých věcí se při jeho tvorbě vyvarovat.

Teoretická část diplomové práce je rozdělena na popis použitých hardwarových (kapitola 2) a programových (kapitola 3) částí. V kapitole věnované hardwarovým platformám může čtenář najít informace o použitém mikrokontroléru TMS570LS (kapitola 2.1), včetně popisu některých, pro tuto práci důležitých periferii. Časovač (kapitola 2.1.3) je nutný jako zdroj aktuálního času a především jako zdroj časování v aplikacích a přidělování procesorového času plánovačem. Dále je zde popsán model řízení přerušení (kapitola 2.1.5) a periferie seriových portů (kapitola 2.1.4) vhodných pro základní komunikaci. Do hardwarového popisu patří ještě

popis ETHERNETové (kapitola 2.3) periferie společně s popisem na mikrokontroléru integrovaném řadiče MAC (kapitola 2.3.1) a externího fyzického rozhraní. V neposlední řadě jsou zde základní informace o vývojovém kitu TMS570LS HDK (kapitola 2.5).

Popis programových částí (kapitola 3) je rozdělen do tří hlavních témat. Jako první je zde krátký úvod do operačního systému RTEMS (kapitola ??). Po seznámením s operačním systémem následuje obecný popis knihovny LwIP (kapitola 3.1). V poslední části programového úvodu do problematiky je velmi lehký nástin fungování ETHERNETU (kapitola 3.2) ze softwarového pohledu.

Realizační část diplomové práce (kapitola 4) sleduje pořadí dílčích úkolů nutných k dosažení výsledné implementace. Během přípravy projektu bylo nutné, aby byly připraveny podklady pro hlavičkové soubory (kapitola 4.1). Tyto soubory v sobě obsahují kompletní popis použité hardwarové platformy, tedy mikrokontroléru TMS570. Systém generace hlavičkových souborů (kapitola 4.2) je popsán v další kapitole.

Po úspěšném vygenerování hlavičkových souborů bylo potřeba zvážit programový model vyvíjené aplikace a připravit vše potřebné pro start operačního systému (kapitola 4.3) RTEMS. Před počátkem vlastní implementace je nutné připravení souborů k začlenění TMS570 do RTEMS (kapitola 4.4) projektu. Když je vše připraveno, lze začít s počáteční integrací základních periferií do projektu RTEMS (kapitola 4.5). Jako první část implementace se obvykle volí výstup ze sériových portů (kapitola 4.5.1). Tento výstup pak slouží dále pro ladění a komunikaci s jádrem a testy běžícími na hardware. Po úspěšném zobrazení znaků přijatých na sériovém portu následuje prostor pro integraci časovače 4.5.2. Podpora časovače je nezbytná pro počáteční start a běh operačního systému. Zbylá integrace modulu VIM (kapitola 4.5.4) a PINMUX (kapitola 4.5.3) už je pouze logický důsledek přípravy na implementaci sítové knihovny.

Integrace síťového rozhraní je rozdělena do tří stejně důležitých částí. Jedná se o přizpůsobení vybraného protokolového zásobníku operačnímu systému RTEMS (kapitola 4.6.1), přizpůsobení hardwarové platformě TMS570 (kapitola 4.6.2) a konečně přizpůsobení knihovně LWIP (kapitola 4.6.3).

Výsledky dosažené v této diplomové práci byly průběžně testovány (kapitola 4.7) a ačkoliv se při testování objevilo mnoho záludných problémů, vedly k jednoznačně pozitivnímu závěru (kapitola 5).

# Kapitola 2
# Popis hardwarových částí práce

## 2.1 TMS570LS

### 2.1.1 Microkontroléry firmy Texas Instruments

Texas Instruments nabízí rodinu mikrokontrolérů TMS570, které jsou vhodné pro použití v bezpečnostně-kritických aplikacích. Obvody TMS570 mají snahu zjednodušit vývoj cílových systémů, neboť jsou již primárně navrženy pro dosažení Safety Integrity Level 3 (SIL 3) standardu IEC61508. V současné době jsou dostupné například moduly TMS570LS, zahrnující dvě 32-bitová jádra CORTEX–R4, pracující v režimu `lock-step`. Taktované jsou na 160 MHz (250 DMIPS) a při práci využívají přidruženou jednotku pro práci s plovoucí čárkou (FPU), vhodnou pro rychlé 32-bitové a 64-bitové výpočty a operace s plovoucí desetinnou čárkou (IEEE 754). Instrukce s plovoucí čárkou a celočíselné instrukce je tak možné zpracovávat paralelně, s cílem dosažení vyššího výpočetního výkonu.

Jádro CORTEX–R4 obsahuje dvaatřiceti bitovou ARM a šestnácti nebo dvaatřiceti bitovou Thumb2 instrukční sadu. Jádro dovoluje aplikaci přepínat mezi oběma instrukčními sadami, podle aktuálních požadavků programového kódu. Tím je zajištěn optimální kompromis mezi výpočetní rychlostí a velikostí programového kódu. Obvody z řady TMS570 jsou v současné době k dispozici s integrovanou programovou pamětí typu Flash s kapacitou 1 až 2 MB.

Důležitou složkou mikrokontrolérů TMS570 je tzv. High End Timer (NHET). Jedná se o přidružený časovací koprocesor, který lze programovat pomocí speciálních instrukcí.

Pro snížení zatížení mikrokontroléru je možné čtení i zápis provádět pomocí DMA nebo HTU (High End Timer Transfer Unit), s využitím NHET DMA řadiče.

Pro konverzi analogových signálů z připojených senzorů jsou v obvodech TMS570LS dostupné dva analogově-digitální převodníky typu MibADC (multi-input-buffered analog-digital converter) s 12-bitovým rozlišením a 24 vstupními kanály. Pro snížení zatížení jádra obvodu jsou oba moduly MibADC vybaveny svou vlastní paměťovou oblastí RAM, ve které je možné uchovávat až 64 výsledků posledních konverzí. Výsledky jsou ukládány pro každý modul samostatně a to v předem stanovených časových intervalech, čtených pomocí CPU nebo DMA.

Pro koordinaci systému a pro připojení k nadřazeným obvodům systému je často integrována sběrnice typu FlexRay. Pro modul FlexRay je důležité, aby podporoval čtení dat bez interakce CPU, podobně jako pracuje DMA, o což se stará jednotka FlexRay Transfer Unit (FTU). Kromě zmíněného umožňuje připojení k SPI či LIN/SCI modulu. [2]

### 2.1.2 Bezpečnost mikrokontroléru

Rodina obvodů TMS570LS na jediném čipu kombinuje dvě shodná jádra typu CORTEX–R4,která zpracovávají shodný tok instrukcí. Operace a výsledky obou výpočtů jsou po každém instrukčním cyklu CPU porovnávány s cílem identifikovat potenciálně problémová místa a adekvátně na ně reagovat. Obě jádra CORTEX–R4 mají implementovány geometrické a časové rozdíly, sloužící ke společnému odhalení možných hardwarových chyb. Druhé jádro je uvedeno zrcadlově a při zpracovávání kódu využívá zpoždění v řádu několika cyklů.

Výhodou synchronní architektury je především vysoké diagnostické pokrytí, neboť hardware, porovnávající práci obou jader, pracuje trvale v každém cyklu CPU. V případě detekce chyby je systém během několika cyklů hodin schopen chybu vyhodnotit a spustit program od začátku nebo přejít do nouzového režimu. Toto vede k transparentnosti hardwarových chyb.

V mikrokontroléru je integrován modul automatické detekce a nápravy ojedinělých chyb (ECC), který diagnostikuje jak chyby v programové paměti, tak i v datové paměti mikrokontroléru TMS570 a zajišťuje příslušné reakce. Tento modul umožňuje opravit chyby v jednom bitu a detekovat chyby ve dvou bitech datového slova.

Obvody integrují dva typy self-test monitorovacích modulů CPU (LBIST) a pracovních dat (PBIST). Modul LBIST kontroluje jádro mikropočítače během inicializace. Naproti tomu modul PBIST umožňuje testování datové paměti (RAM) různými uživatelem volitelnými algoritmy. Uvedené integrované moduly BIST jsou určeny k nahrazení a zjednodušení odpovídajících softwarových testovacích procedur.

Všechny periferní moduly, které mají integrovánu vlastní paměť (NHET, MibADC, FlexRay, DCAN a MibSPI) využívají hardwarovou paritní logiku. Avšak i tyto oblasti RAM mohou být otestovány s využitím modulu PBIST.

Pro monitorování přístupu k určité oblasti přidělené paměti, nebo oblasti se zvláštními přístupovými právy, je možné využít jednotku ochrany paměti MPU (Memory Protection Unit).

Pro otestování staticky uložených dat je k dispozici 64-bit CRC jednotka, umožňující provoz na pozadí s využitím DMA přenosů.

Většina periferních modulů je vybavena schopností detekce základních typů chyb, jako je například vnitřní test analogově-digitálního převodníku.

Všechny uvedené hlavní moduly, určené pro detekci chyb, jsou spojeny s modulem signalizace poruchových stavů. Ten umožňuje centralizovanou volbu priorit, analýzu a signalizaci všech zjištěných chyb dalším obvodům. [3]

### 2.1.3 Časovač

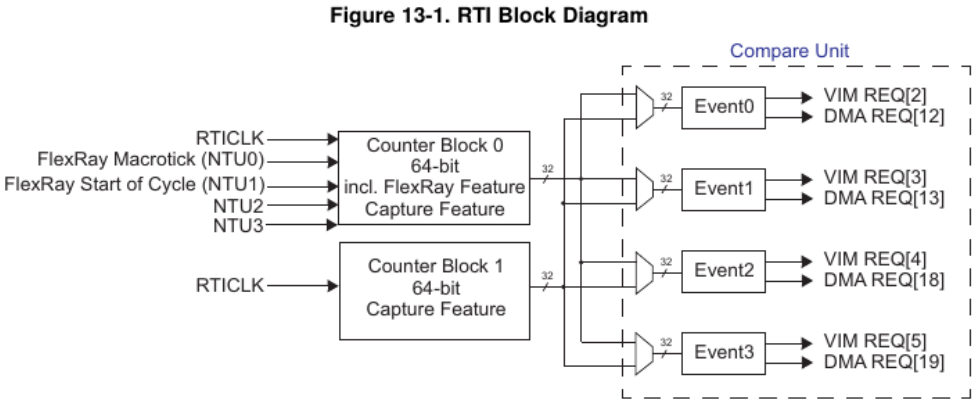
Periferie časovače RTI poskytuje operačnímu systém nástroj, jak měřit časová kvanta a podle nich provádět přepínání kontextu. Periferie obsahuje několik čítačů, které mohou definovat základní časové základny pro plánovač operačního systému. Mimo jiné dovoluje periferie časovače využití některého z čítačů jako nástroj pro měření doby běhu celého programu nebo jeho úseků. Při používání komunikační periferie FlexRay, dovoluje modul časovače synchronizovat čas ze sběrnice FlexRay.

Při poruše sběrnice dokáže modul časovače automaticky přejít na vnitřní zdroj hodinového signálu mikrokontroléru tak, aby běh operačního systému zůstal zachován.

Součástí periferie časovače je hlídací (watchdog) obvod. Úkolem této části mikrokontroléru je po prvním nastavení hlídat běh kódu a vynutit restart mikrokontroléru, pokud vykonávání hlídané smyčky kódu trvá delší nebo kratší dobu, než je specifikováno.

Tento modul je navržen tak, aby odpovídal požadavkům normy automobilového průmyslu OSEK.

Obrázek 2.1 ukazuje základní blokový diagram RTI periferie.




**Obrázek 2.1.** Blokový diagram RTI periferie

Z obrázku je vidět, že periferie obsahuje dva nezávislé čítače pro generování rozdílných časových základen. První čítač mimo jiného pracuje i se zmíněným časovým signálem z periferie FlexRay (pokud není uspaná). Dále dokáže první čítač dohlížet na časový signál z modulu FlexRay. V případě chyby se čítač číslo jedna dokáže okamžitě přepojit na interní hodinový signál. Zbylá funkcionalita obou čítačů je stejná.

Porovnávací jednotka porovnává čítače s naprogramovanými hodnotami a generuje čtyři různé žádosti o přerušení nebo o začátek přenosu DMA. Vstup do každé z těchto čtyřech porovnávacích jednotek může být zvolen buď z čítače jedna nebo z čítače dva.

### 2.1.4 Sériové porty

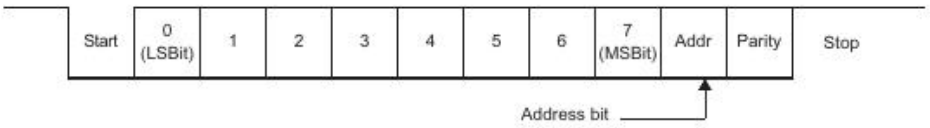
Obsluhu sériových portů v mikrokontroléru TMS570 podporují dvě periferie. V obou je obsluha sériových portů stejná, ale periferie SCI/LIN poskytuje vývojáři rozšířenou funkcionalitu o protokol LIN. Základní funkcí obou periferií je tedy protokol universální asynchronní sériové komunikace (UART), který využívá shodné rozložení registrů, a proto stačí pouze jeden ovladač. LIN je protokol vycházející z protokolu UART, který podporuje komunikaci typu jeden hlavní a více vedlejších zařízeních s možností adresovat více zařízení v síti najednou. Periferie podporuje jednocestnou i oboucestnou komunikaci, dva kanály přerušení, 1-8bitů přenesených dat, volitelný devátý bit pro adresování, možnost sudé nebo liché parity a samozřejmě jeden až dva konečné bity za znakem.

Vzhledem ke stále rostoucím požadavkům na energetickou úspornost může přijít vhod možnost probouzení mikrokontroléru přijetím zprávy. Periferie pro obsluhu sériového portu (SCI) se skládá ze tří hlavních bloků:

- Přijímač – postupně posouvá přijaté bity v posuvném registru. Při dokončení přenosu budou data v registru SCIRD.

- Vysílač – Podle registru SCITD postupně, bit po bitu, nastavuje úroveň signálu na výstupním pinu mikrokontroléru.
- Generátor modulační rychlosti – Programovatelný generátor závislý na hodinovém kanálu VBUSP.

Na obrázku 2.2 lze naleznout formát komunikace jednoho znaku.




**Obrázek 2.2.** Komunikace po sběrnici UART

Znázorněný adresový bit je velmi důležitý při usnadnění komunikace více zařízení po sériové lince.

### 2.1.5 Řízení přerušení

Kvůli snadnější obsluze přerušení byl firmou Texas Instruments do mikrokontroléru zakomponován modul řadiče vektorových přerušení (VIM). Periferie, které generují žádosti o přerušení, posílají zprávy do modulu VIM. Po jejich zpracování rozhodne modul, jestli je žádost povolená. Po kladném vyhodnocení modul VIM pošle znamení jádru mikrokontroléru (CORTEX–R4), aby bylo provedeno přerušení. Následně modul VIM pomáhá s jeho řízením čtyřmi rozlišnými způsoby.

Nejstarší přístup pracuje tak, jak by to bylo běžné v jádře CORTEX–R4. Řízení je předáno na adresu vektoru přerušení v tabulce výjimek jádra. Úkolem obslužné rutiny je přečíst registr IRQINDEX, kam modul VIM uložil index právě aktivní žádosti. Tento přístup je kompatibilní se staršími mikrokontroléry jako je TMS470.

Poněkud modernější přístup automaticky připraví vektor přerušení. Nicméně je potřeba tyto vektory vyplnit před zapnutím přerušení do paměti asociované s modulem VIM. Jakmile modul VIM obdrží žádost o přerušení, připraví do registru IRQVECREG adresu rutiny přerušení, kterou si načetl z předem vyplněné tabulky. Obecná obslužná rutina přerušení, která se nachází v tabulce výjimek, může pouze provést skok na adresu vyplněnou modulem VIM.

Třetí automatický přístup lze použít pouze pro obyčejná přerušení (IRQ) a ne pro přerušení s rychlou odezvou (FIQ). Postup přerušení je analogický k předchozímu přístupu s jedinou výjimkou. Modul VIM požádá jádro mikrokontroléru, aby neprovádělo skok na obvyklou adresu v tabulce výjimek, ale aby bylo předání řízení na adresu určenou modulem VIM. Tím je zaručeno, že již první procesorovým jádrem vykonaná instrukce po přijetí žádosti, jejíž první instrukce obslužné rutiny specifické pro danou periferii. Tato funkcionalita se zapíná v jádře CORTEX–R4 pomocí bitu VE v konfiguračním registru.

Poslední přístup předpokládá využití programem řízených přerušení. Aplikace se spoléhá pouze na sebe a funkcionalitu vektorů spravovaných modulem VIM odmítá. V takovémto případě se od aplikace očekává, že kromě zrušení žádosti o přerušení v dané periferii (časovač), zruší aplikace ještě žádost o přerušení v modulu VIM. Tohoto lze dosáhnout přečtením registru IRQVECREG nebo zápisem logické jedničky na příslušné umístění v registru INTREQ.

## 2.2 ARM CORTEX–R4

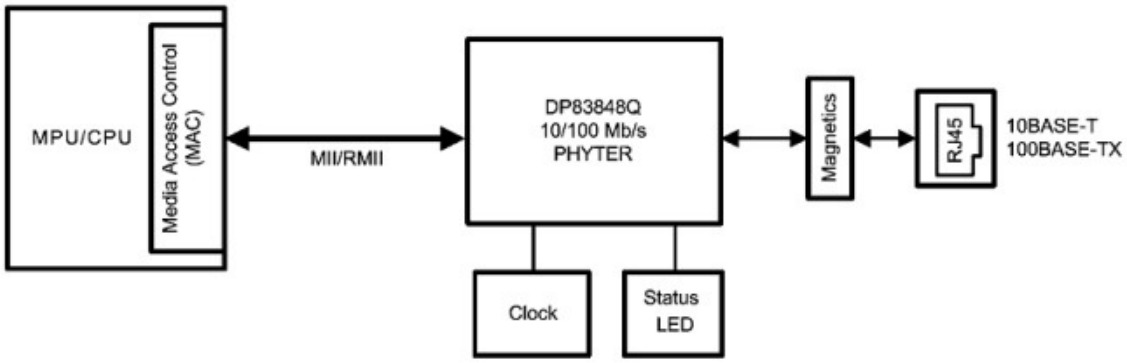
Procesory ARM je dnes možné najít téměř ve všech oblastech spotřební elektroniky, jako jsou mobilní telefony, PDA, kalkulačky, herní konzole, routery, multimediální přehrávače, roboty a spousta dalších. Mezi jejich hlavní přednosti patří poměrně velký výpočetní výkon, malá spotřeba a malá velikost kódu. Těchto vlastností je s výhodou využíváno nejen ve vestavěných zařízeních.

Bohužel podrobnější informace o jádře CORTEX–R4 patří pod licenci firmy ARM Holdings Ltd a proto se teoretickému fungování jádra CORTEX–R4 tato práce nezabývá. [4]

## 2.3 ETHERNET a PHY

Adaptér pro ethernet nebo rychlý ethernet se ve vestavěných systémech rozděluje na dvě základní části a to na část MAC (Media Access Control), která řeší problémy na vyšší úrovni a je součástí mikrokontrolérů, a na rozhraní fyzické vrstvy (Physical Layer Interface – PHY), která řeší fyzickou část připojení.

Propojení MAC s PHY je provedeno přes synchronní paralelní rozhraní, jak je uvedeno na obrázku 2.3.




**Obrázek 2.3.** Schéma zapojení adaptéru pro Ethernet

Pro připojení PHY a MAC jsou k dispozici dvě varianty rozhraní. První variantou rozhraní je rozhraní MII (Media Independent Interface) pracující se 4-bitovou šířkou sběrnice a s časováním 25 MHz. Druhá varianta RMII (Reduced Media Independent Interface) pracuje s poloviční šířkou sběrnice, ale s dvojnásobným kmitočtem.

MII (RMMI) dovoluje přenášet datagramy rychlostí 100Mbit/s.

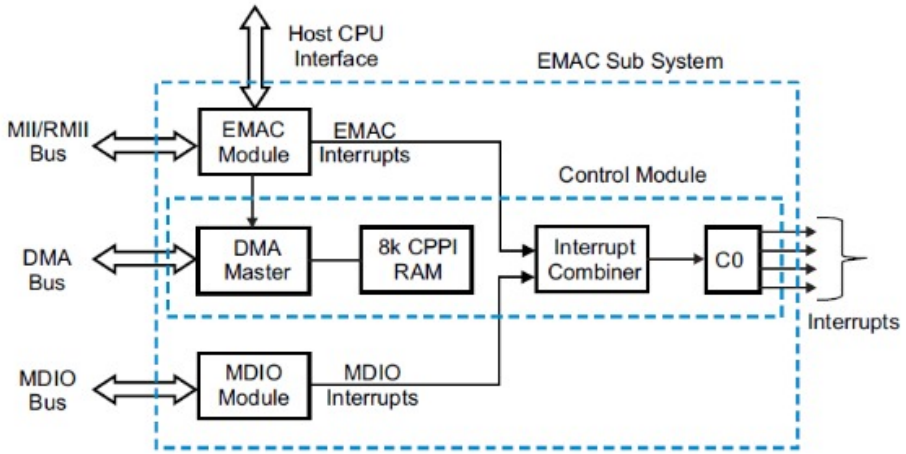
### 2.3.1 MAC

Část MAC, která je součástí mikrokontroléru je mimo jiné v souladu s normou IEEE 802.3 – 2002 Ethernet MAC, s normou IEEE 1588-2008 standard for precision networked clock synchronization (přesná síťová synchronizace s hodinami) a s RMII specifikací z konsorcia RMII.

MAC podporuje přenosovou rychlost, podle normy IEEE 802.3 obsahuje kompatibilní rozhraní MII pro komunikaci s externím obvodem realizujícím fyzické rozhraní FAST ETHERNET PHY, podporuje jak full-duplex, tak half-duplex provoz včetně CSMA/CD protokolu v half-duplex provozu, podporuje IEEE802.3x řízení toku pro full-duplex provoz.

U mikrokontroléru TMS570LS3137 je MAC řešen třemi základními moduly a to kontrolním modulem (EMAC kontrol module), EMAC modulem (EMAC module) a MDIO modulem (MDIO module).

Jak je patrno z obrázku 2.4, je řídící modul hlavním rozhraním mezi EMAC modulem, který poskytuje rozhraní mezi jádrem mikrokontroléru, sítí a MDIO modulem, který zajišťuje řízení PHY.

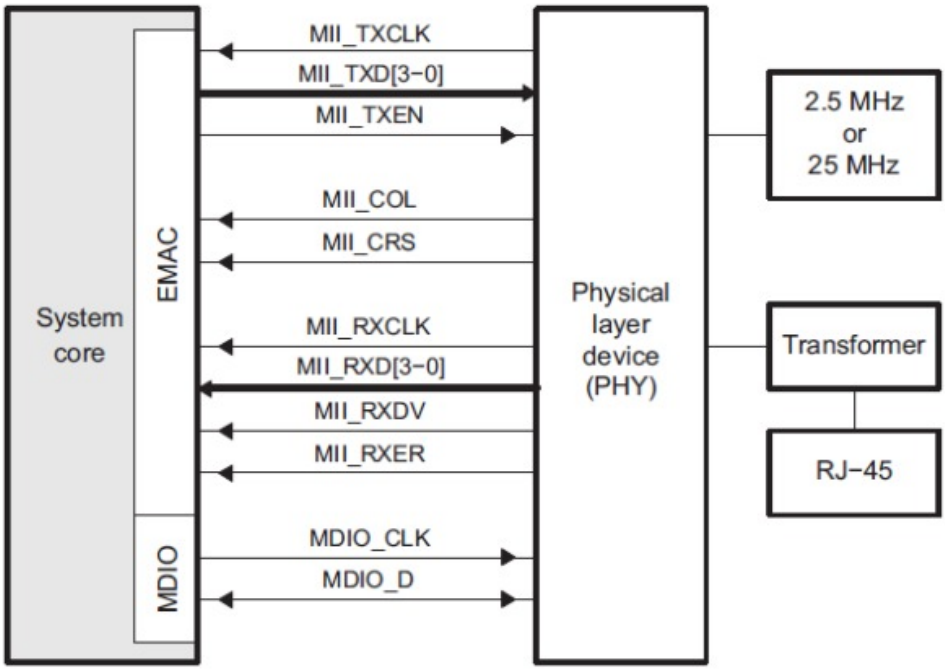



**Obrázek 2.4.** moduly MAC

Součástí kontrolního modulu je řízení přerušení a 8KB integrovaná vyrovnávací paměť rezervovaná pro popisovače (descriptors), která je mapovaná do adresního prostoru.

Obrázek 2.4 dále zobrazuje rozhraní mezi EMAC řídícím modulem a mikrokontrolérem (CPU). Spojení DMA sběrnice s EMAC řídícím modulem umožňuje čtení a zápis do vnitřní i vnější paměti bez využití CPU. Přerušení z modulu EMAC a MDIO jsou spojeny do čtyř signálů a přivedeny do vektorového řízení přerušení (VIM – Vectored Interrupt Manager).

Obrázek 2.5 zobrazuje vlastní připojení EMAC modulu a MDIO modulu s PHY. Detailní popis jednotlivých signálů je uveden v [X - SPNU499B–November 2012–Revised August 2013]



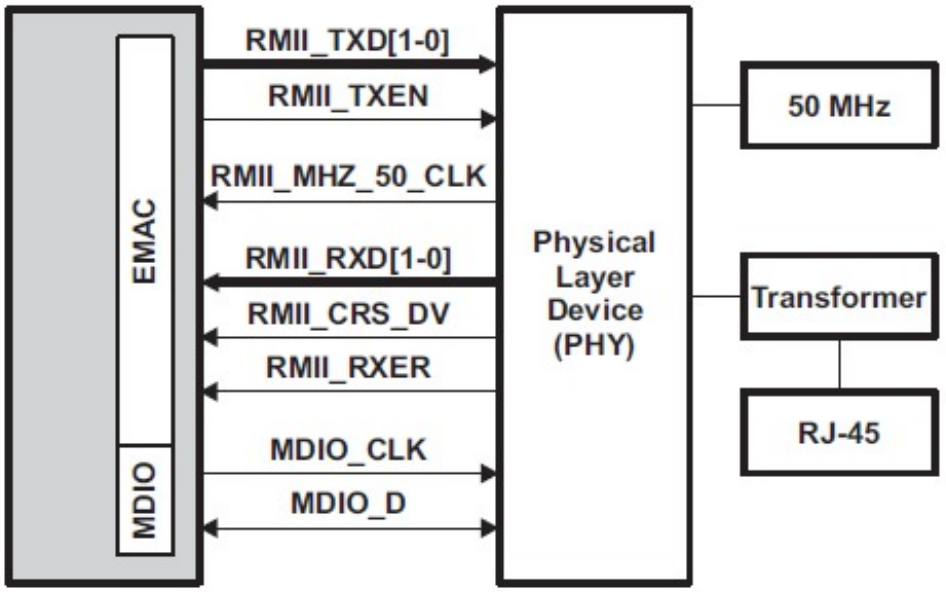

**Obrázek 2.5.** EMAC a MDIO Signály pro MII Interface

MII_TXCLK jsou kontinuální hodiny, které poskytují referenční časování pro Tx přenos. Jmenovitá frekvence je 2,5MHz pro 10Mbit/s a 25MHz pro přenos 100Mbit/s. MII_TXD[3-0] je vysílací paralelní sběrnice. MII_TXEN je signál uvolnění vysílání. Jako rozšíření se může použít i signál MII_TXER (z MAC do PHY) chyba vysílání.

MII_RXCLK jsou opět kontinuální hodiny avšak pro Rx přenos. MII_RXD[3-0] je sběrnice pro příjem datových rámců. MII_RXDV je signál, že jsou k dispozici platná data a MII_RXER je signál chyby příjmu.

MII_COL informuje o kolizi a MII_CRS informuje o komunikaci na sběrnici. Tyto signály mají význam jen v half-duplex režimu.

Redukovaná varianta MII (RMII) je na obrázku 2.6.




**Obrázek 2.6.** EMAC a MDIO Signály pro RMII Interface

### 2.3.2 MDIO modul

MDIO modul (Management data input output) umožňuje aplikaci získat přístup do až 32 registrů PHY zařízení pomocí dvouvodičové sběrnice (hodiny, data). MDIO rozhraní podporuje až 32 PHY zařízení. Aplikace může vybrat jeden libovolný registr v kterémkoliv PHY a posílat řídící data a přijímat informace o stavu. Avšak pouze jeden registr v jednom PHY zařízení může být adresován v daném okamžiku.

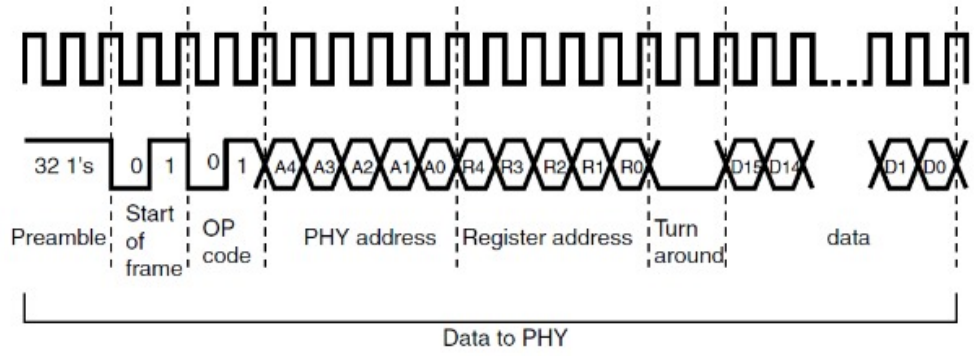
Maximální frekvence hodinového signálu MDIO_CLK je 2,5MHz. MDIO_D signál je oboustranný a umožňuje jak čtení, tak zápis. Na obrázku 2.7 je zobrazena datová struktura zprávy (frame).

| | Management frame fields | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Preamble (32 bits) | Start | Operation | PADDR | RADDR | TA | Data (16 bits) | Idle |
| Read | 1... 1 | 01 | 10 | ppppp | rrrrr | Z0 | dddddddddddddddd | Z |
| Write | 1... 1 | 01 | 01 | ppppp | rrrrr | 10 | dddddddddddddddd | Z |

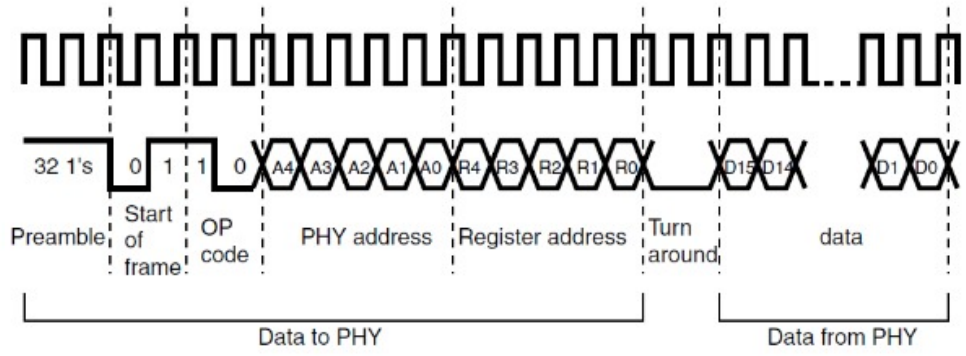
**Obrázek 2.7.** Struktura MDIO zprávy

Preamble je inicializační pole a používá se k synchronizaci s PHY zařízením. PADDR je 5bitová adresa PHY zařízení a RADDR je 5bitová adresa registru. TA bity slouží k oddělení datového pole, aby se zabránilo k připojení během transakce čtení. Pro transakci čtení nastavuje MAC pro první i druhý TA bit vysokou impedanci na datovém vodiči (MDIO_D). PHY zařízení musí po dobu prvního bitu TA zachovat vysokou impedanci (je nastaven jako vstup) a po dobu druhého bitu TA být nastaveno na logickou nulu. Pro transakci zápisu MAC zařízení nastavuje TA bity na hodnotu „10“ a PHY zařízení se musí nastavit do vysoké impedance pro oba bity TA.

Na obrázku číslo 2.8 a 2.9 jsou zobrazeny časové průběhy přenášených dat při transakci zápisu (2.8) a při transakci čtení (2.9).




**Obrázek 2.8.** MDIO časování – transakce zápis




**Obrázek 2.9.** MDIO časování – transakce čtení

## 2.4 DP83848

DP83348 je modul PHY od společnosti Texas Instruments. Tento obvod podporuje obě zmíněné možnosti připojení (MII a RMII).

Standardní MII má sadu registrů od adresy 0h do adresy 7h

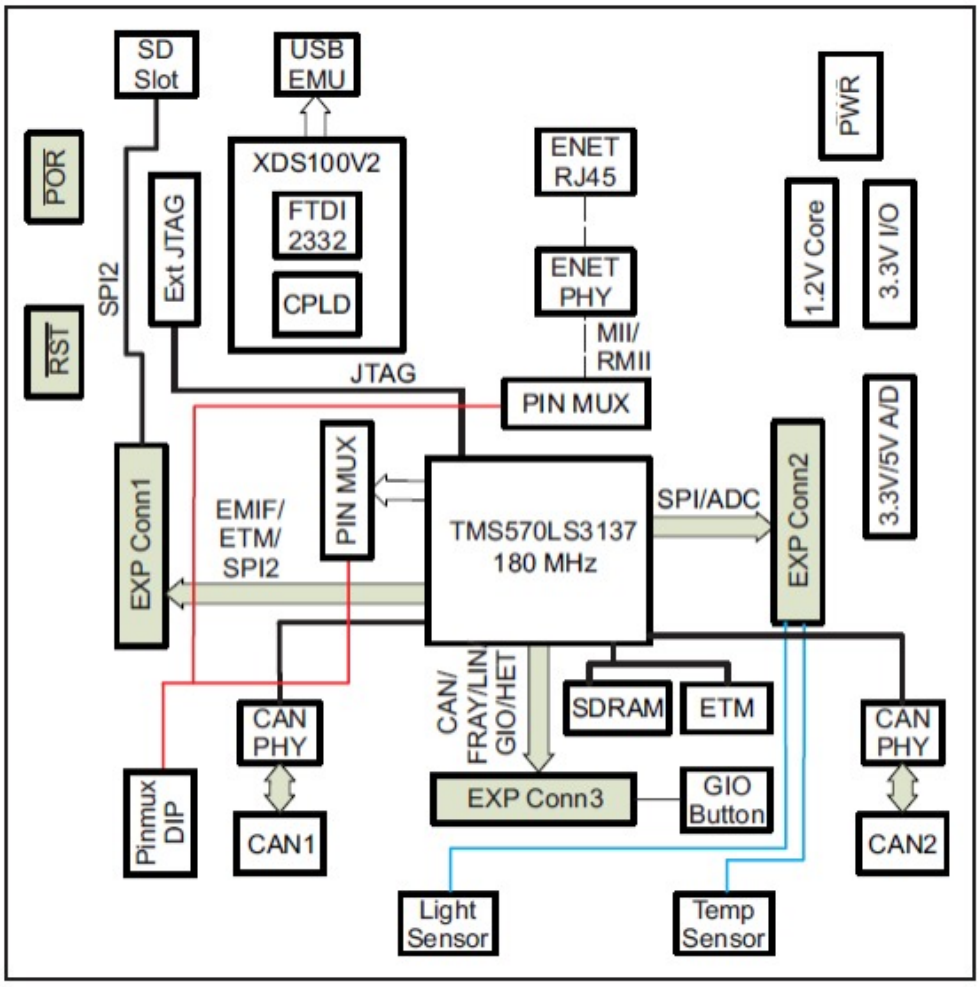
## 2.5 Vývojový kit TMS570

Vývojový kit TMS570LS31 společnosti Texas Instrumens poskytuje hardwarovou platformu k ověření funkčnosti mikrokontrolérů rodiny TMS570LS a je určen zejména k usnadnění vývoje zákaznického hardware a software a tím snížení doby uvedení na zákaznický trh.

### 2.5.1 Popis kitu

Podrobné informace k vývojovému kitu společnosti Texas Instrumens včetně elektrického schématu nebo výrobních podkladů k plošnému spoji jsou na jejich internetových stránkách. [5]

Vývojový kit je založen na mikrokontroléru TMS570LS31. Na obrázku 2.10 je blokové schéma vývojového kitu.




**Obrázek 2.10.** Blokové schéma vývojového kitu TMS570LS31 HDK

Vývojový kit je osazen mikrokontrolérem Hercules TMS570LS3137 v BGA pouzdře se 337 piny. Dále je zde 10/100Mbit síťové rozhraní s PHY DP83640, externí JTAG konektor, UART, 8MB SDRAM, slot pro SD kartu přes SPI sběrnici, napájení od +5V do +12V @ 130mA.

# Kapitola 3
# Popis programových částí práce

## 3.1 LwIP

Knihovna LwIP poskytuje implementaci protokulu TCP/IP vhodnou pro aplikace s omezeným množstvím paměti.

Knihovna LwIP je dostupná pod BSD licencí a je vhodná do aplikací využívajících jazyk C. Cílem projektu bylo poskytnout plnou podporu TCP/IP při co nejmenším využití operační paměti. Knihovna je vhodná pro mikrokontroléry s více jak deseti kilobyty operační paměti a čtyřicetikilobyty paměti programu.

Knihovna LwIP je schopna funkce na různých operačních systémech pro mikrokontroléry. Stejně tak byla přizpůsobena rozličným hardwarovým platformám. [6]

## 3.2 ETHERNET

Protože je implementace Ethernetu jedním z požadavků, poskytuje následující kapitola alespoň stručnou teorii týkající se tohoto tématu. Jsou zde rozebrány i protokoly vyšších vrstev, které s Ethernetem souvisí.

### 3.2.1 Co je ethernet

Ethernet je širokopásmový komunikační systém určený k přenosu datagramů (packetů) přes lokální síť.

Počátky Ethernetu sahají do 70. let minulého století, kdy s jeho vývojem započala firma Xerox. Na této prvotní verzi se spolu s firmou Xerox dále podílely firmy DEC a Intel. První standardizace byla provedena organizací IEEE pod označením 802.3 (následně pak organizací ISO jako 8802.3).

Jako přenosové médium se dnes používají převážně kabely s kroucenou dvojlinkou a optické kabely s přenosovou rychlostí od 10Mbit/s až po 100Gbit/s.

### 3.2.2 ISO/OSI model

Nejpoužívanějším referenčním modelem v dnešní době je referenční model ISO/OSI popsaný v následující kapitole.

Základem chápání ethernetu je jeho rozdělení do jednotlivých vrstev. K tomu slouží referenční model ISO/OSI. Jedná se o referenční model otevřené komunikace, odtud potom zkratka OSI znamenající Open Systems Interconnection. Jelikož samotný referenční model je velmi rozsáhlý a vydal by za sepsání několika samostatných svazků, uvádím v této práci pouze nezbytný přehled.

### 3.2.3 Vrstvy

Referenční model ISO/OSI definuje následující vrstvy:

fyzická vrstva, linková vrstva, síťová vrstva, transportní vrstva, relační vrstva, prezentační vrstva, aplikační vrstva

Díky rozdělení ethernetu do jednotlivých vrstev je možné stanovit podmínky, za kterých je možné spolehlivě komunikovat mezi sebou po sériové sběrnici. Model je tvořen sedmi vrstvami, z nichž každá plní svoji předem definovanou funkci a službu. V referenčním modelu je vysvětleno jak se zpracovává odesílaná a přijímaná zpráva postupně po vrstvách od odesílatele až k příjemci. K samotnému přenosu informace dochází přes fyzický spoj. Účastníci komunikace, kteří spolu komunikují na aplikační vrstvě běžně nepotřebují a nedostávají žádné informace o funkci nižších vrstev. Každá z vrstev má definována pravidla, která řídí komunikaci mezi účastníky (zahájení, provedení, ukončení přenosu).

Vazba každé vrstvy je omezena na jednu nižší vrstvu a jednu vyšší vrstvu (pokud existují). Každá vrstva poskytuje své funkce nejbližší vyšší vrstvě přes softwarové rozhraní realizované jako komunikační protokol. To lze chápat jako soubor pravidel pro komunikaci.

### 3.2.4 Fyzická vrstva

Fyzická vrstva je základní vrstvou referenčního modelu ISO/OSI. Fyzická vrstva je tvořena

logickou sběrnicí, po které jsou datové pakety přenášeny směrem ke všem účastníkům komunikace. Datové pakety jsou však určeny pouze těm, jejichž adresa je uvedena v adresovém poli přenášeného rámce. Fyzická vrstva definuje rozložení pinů, použité konektory, napěťové úrovně, vlastnosti a specifikace kabelů, elektrické vlastnosti přenosového média i jeho mechanické vlastnosti.

### 3.2.5 Spojová vrstva

Jelikož síť je obecně využívána mnoha zařízeními od různých výrobců, je každé ethernetové rozhraní těchto zařízení označeno unikátní adresou nazývanou MAC (z anglického „Media Access Control“), často označovanou také jako fyzická adresa. MAC adresa je přiřazována síťové kartě bezprostředně při její výrobě. Ethernetová MAC adresa se skládá z 48bitů převážně zapisovaných ve formátu hexadecimálních čísel 00-00-00-00-00-ab. První polovinu MAC adresy přiřazuje centrální správce adresního prostoru, druhou polovinu MAC adresy zajišťuje přímo výrobce.

Spojová vrstva zajišťuje přenos dat v rámci jedné lokální sítě právě pomocí fyzických adres zařízení. Jednotlivé bity přenášeného rámce se přenášejí po fyzickém médiu, samotnému přenosu informačních bitů pak předchází startovací posloupnost. Startovací posloupnost, označovaná také jako preambule (sekvence střídajících se jedniček a nul), slouží k synchronizaci vysílací stanice a všech přijímacích stanic. Datový rámec obsahuje adresu příjemce, odesílatele, typ zprávy, samotná data a kontrolní součet.

Spojová vrstva dále definuje přístupovou metodu k přenosovému médiu (kabelu). Jelikož je přenosové médium sdíleno několika stanicemi, které mohou ve stejnou chvíli začít vysílat, je třeba definovat pravidla přístupu k tomuto přenosovému médiu. Nejznámější přístupovou metodou je CSMA/CD (Carrier Sense Multiple Access/Collision Detection). Každý z účastníků komunikace má v tomto případě stejné právo využít sdílené přenosové médium v jakémkoli okamžiku, kdy je médium nevyužito. Pokud se však dvě zařízení rozhodnou odeslat data ve stejný

okamžik dojde ke kolizi. Pokud stanice detekují kolizi, vyšle signál JAM, kterým ohlásí i ostatním stanicím, že došlo ke kolizi a po náhodném čase vysílání opakuje, pokud je sdílené médium volné k použití. Modernější varianty ethernetu však od sdíleného média a tedy od přístupové metody CSMA/CD ustupují a využívají přepínače s plně duplexním režimem provozu.

### 3.2.6 Síťová vrstva

Úkolem síťové vrstvy (network layer) je zajistit především síťovou adresaci, směrování a předávání dat (datagramů). Síťová vrstva je pak schopna přenášet data v jedné síti nebo mezi vícero sítěmi i technologicky rozdílnými.

Předávání dat mezi jednotlivými sítěmi je obstaráno systémem bran a směrovačů. Na síťové vrstvě pracuje množství protokolů. Nejdůležitější z nich jsou Internet Protocol (IP), Internet Control Message Protocol (ICMP), NWLink, IPX

### 3.2.7 IP protokol

Internet Protocol je zodpovědný za směrování datagramů (paketů) ze zdrojového zařízení do cílového zařízení přes jednu nebo více IP sítí. Datagram se skládá z řídících dat a z uživatelských dat. Řídící data zajišťují informace k doručení datagramů (adresu zdroje a cíle, kontrolní součty, informace o pořadí atd.). Při zátěži sítě nebo přerušení některé její části se může snadno stát, že diagramy vůbec nedorazí na místo určení. Taktéž mohou dorazit vícekrát a IP protokol v základní verzi neručí ani za pořadí doručovaných diagramů.

### 3.2.8 ICMP

ICMP(anglicky Internet Control Message Protocol) se používá pro odeslání chybových zpráv, nebo za účelem diagnostiky sítě či směrování datagramů.

### 3.2.9 Transportní vrstva

Účelem transportní vrstvy je zajistit spolehlivý přenos dat s požadovanou kvalitou spojení pro uživatelské počítačové programy. Úkolem transportní vrstvy je doručit data k požadovanému aplikačnímu procesu na cílovém zařízení. Mezi dvěma zařízeními může v jeden okamžik vzniknout několik spojení.

Identifikační adresa aplikačního procesu je tvořena zdrojovou adresou, cílovou adresou a číslem portu a je známa pod pojmem Socket.

Transportní vrstva obsahuje větší množství protokolů. Zde uvedeme jen dva protokoly a to TCP a UDP.

### 3.2.10 TCP (Transmission Control Protocol)

Jeden z nejvíce využívaných protokolů v transportní vrstvě je TCP (Transmission Control Protocol). TCP protokol zajišťuje vytvoření spojení mezi dvěma zařízeními v síti, přes které obě zařízení mohou obousměrně komunikovat. TCP protokol garantuje spolehlivý (bezchybný) přenos datagramů a to i ve správném pořadí.

### 3.2.11 UDP (User Datagram Protocol)

UDP je jednodušší protokol založený na odesílání nezávislých zpráv. Na rozdíl od TCP protokolu nevyžaduje UDP protokol sestavení přenosového kanálu. Nepožaduje zajištění spolehlivého přenosu datagramů, ani jejich správné pořadí příjmu. Hlavní výhodou UDP protokolu je jeho malá náročnost. UDP protokol je zejména vhodný pro nasazení u časově kritických aplikací a u aplikací, které pracují systémem otázka – odpověď (DNS, SNMP, DHCP atd.).

### 3.2.12 Prezentační vrstva (presentation layer)

Úkolem prezentační vrstvy je transformovat data do tvaru, které používají aplikace.

### 3.2.13 Aplikační vrstva ( application layer)

Účelem aplikační vrstvy je poskytnout aplikacím přístup ke komunikačnímu systému a umožnit tak jejich spolupráci.

Na rozdíl od prezentační vrstvy, která se nezabývá vlastním významem přenášených dat, jsou právě tyto informace aplikační vrstvou rozeznávány.

Stejně jako na nižších vrstvách i na aplikační vrstvě se můžeme setkat s různými protokoly. Tyto protokoly mohou být vázány ke konkrétní aplikaci, ale existují i aplikační protokoly, které jsou v některých případech nezbytné pro správný chod sítě a s aplikací běžící na aplikační vrstvě nemají nic společného. Případem takového protokolu je například DHCP protokol (Dynamic Host Configuration Protocol). Dalšími známějšími protokoly aplikační vrstvy jsou protokoly HTTP, IMAP, SSH, POP3 a další.

# Kapitola 4
# Realizace

## 4.1 Příprava hlavičkových souborů

Jedním z kritických problémů při zadání této diplomové práce byla nedostupnost hlavičkových souborů popisujících periferie mikrokontroleru a registry jádra. Jediné existující hlavičkové soubory byly pod licencí TI, která byla nekompatibilní s licencí operačního systému RTEMS. Důsledkem nekompatibility licencí byla nemožnost přijmout hlavičkové soubory od firmy Texas Instruments do oficiálního vývojového stromu RTEMS. Mezi cíle této diplomové práce tedy patří návrh a vytvoření vlastních hlavičkových souborů pod licencí kompatibilní s projektem RTEMS.

Použití veřejně dostupného popisu registrů, který je součástí referenčních manuálů, bylo vyhodnoceno jako jediný právně a licenčně bezrizikový způsob získání potřebných souborů. Pro vytvoření hlavičkového souboru shodného s hlavičkovým souborem od firmy Texas Instruments stačilo vyextrahovat jméno registru, popis a jeho umístění v paměti mikrokontroleru. Náplní diplomové práce bylo kromě vygenerování shodných hlavičkových souborů i návrh přehlednějších souborů a struktur, které by mohly vést ke zrychlení vývoje a většímu programátorskému komfortu při používání nových hlavičkových souborů. Pro účely přípravy podrobnějších hlavičkových souborů musely být vyextrahovány další informace z referenčního manuálu. Mezi tyto informace patří například bitová pole registrů, jejich možné hodnoty, informace a popisy.

Referenční manuál popisující periferie mikrokontroléru TMS570 obsahuje přibližně dva tisíce stránek. Mezi hlavní kritéria generátoru hlaviček patřila rychlost extrakce dat a tvorba nových hlavičkových souborů. Dále pak korektnost vyextrahovaných dat. Hlavičkové soubory jsou používané programátory s předpokladem, že jsou správné. Hledání i jen jediné drobné chyby může zpomalit vývoj nové aplikace o několik týdnů.

V rámci práce byla vyzkoušena efektivita, časová náročnost a korektnost následujících několika přístupů.

### 4.1.1 Ruční extrakce

Nejprve se nabízí extrakce dat z referenčního manuálu ručním kopírováním.

Mezi výhody tohoto přístupu patří absolutní kontrola nad zpracovávanými daty. Mezi nevýhody patří časová náročnost a nemožnost snadného opakování. Vytvoření jednoho hlavičkového souboru znamená mnoho hodin monotónního kopírování.

### 4.1.2 Plně automatický přístup

Dalším možným přístupem je extrakce dat z referenčního manuálu za pomoci pouze naprogramovaného kódu. Výhodou tohoto přístupu je snadná opakovatelnost a relativně jednoduché převedení na generování hlaviček pro jiný mikrokontroler od firmy Texas Instrument.

Referenční manuál se pomocí otevřených nástrojů převedl z formátu PDF do formátu čistého textu. Tento text se následně mohl zpracovávat skriptovacím jazykem Python. Naprogramovanému skriptu se podařilo automaticky načíst z obsahu seznam všech periferií a ty následně vyhledat a spárovat v dalším textu. Vygenerovat informace nutné k sestavení původních hlavičkových souborů od firmy Texas Instruments se povedlo ze dvou třetin. Hlavním důvodem chyb byl nesourodý styl textu referenčního manuálu a tedy potřeba, aby se skript automaticky přizpůsoboval obsahu a formě textu. Řešení takového problému není triviální. Taktéž možnost vygenerovat informace o bitových polích je z čistého textu nereálná, protože v něm už neexistují informace o poloze textu a tabulek. Posledním problémem tohoto přístupu je velká závislost na původním referenčním manuálu, který k mikrokontroléru TMS570 existuje již ve třech verzích. Ze třetí nejmodernější verze referenčního manuálu nejde tímto způsobem vyextrahovat ani jeden registr.

### 4.1.3 Makra

Cílem dalšího přístupu byla snaha zůstat u automatického generování hlavičkových souborů tak, aby se snadno mohla měnit jejich cílová podoba. Avšak přístup k extrakci dat z referenčního manuálu byl odlišný.

Hlavní myšlenkou bylo zůstat u formátu PDF, který zachovává všechny informace o poloze textu. Způsob jak s těmito informacemi pracovat vyžaduje využití člověka nebo použití pokročilých rozpoznávacích metod pro zpracování obrazu.
Rozpoznávání obrazu bylo vyzkoušeno pomocí otevřeného projektu Poppler a několika dalších nepříliš udržovaných utilit.

Po těchto experimentech byl nakonec použitý postup kombinující automatizovaný výběr do schránky a další zpracování v jazyce Python. Formát výstupních dat ze scriptu je snadno modifikovatelný. Script využívá jazyka Autohotkey a běží v operačním systému. Pracuje jako stavový automat a po stisku příslušného tlačítka provede nad referenčním manuálem sérii krátkých operací. Pomocí schránky vykopíruje data z referenčního manuálu, doplní je o požadované náležitosti a zapíše na disk. Takto zpracovaná data mají vždy stejný a snadno nastavitelný formát, což z nich dělá ideální vstup do dalšího programu generujícího hlavičkové soubory.
Výhodou tohoto přístupu je použití snadno modifikovatelných skriptů, které zajišťují velkou adaptabilitu na jiné styly textu či jiné referenční manuály. Celkový čas extrakce dat z největších periferií mikrokontroléru TMS570 se pohybuje okolo jedné hodiny. Nevýhodou je, že se stále jedná o ruční práci, takže se může objevit chyba. Výsledná vygenerovaná data je tedy nutno překontrolovat.
Tato metoda byla použita pro její nejlepší poměr mezi časem stráveným extrakcí dat z referenčního manuálu, korektností vyextrahovaných dat a dobou potřebnou k vývoji metody.

Ukázka výstupního formátu je uvedana níže.

```
"name" : "CRC",
"full name" : "Cyclic Redundancy Check",
"offset" : ["0xFE000000"],
"registers" : [{
    "name" : "CTRL0",
    "info" : "CRC Global Control Register",
    "lenght" : "32",
    "adress" : "0",
    "offset" : "0",
    "fields" : [ {
        "bit_number" : "8",
        "bit_Field_Name" : "CH2_PSA_SWREST",
        "info" : "Channel 2 PSA Software Reset.
      },
```

### 4.1.4 Budoucí rozšíření

Naposledy zmíněný postup je snadno replikovatelný a použitelný v praxi. Avšak pro případy potřeby extrahovat data k vícero mikrokontrolérům byl navržen ještě jeden komfortnější přístup. Základní myšlenka vychází z faktu, že pro moderní počítače není problém zpracovat i větší obrázky v téměř okamžitém čase. Komfortní postup extrahování dat by mohl být následující:

- naučit program rozpoznávat znaky z pdf. Ideální varianta je statické učení pomocí textového modulu.
- označit umístění pdf na monitoru
- listovat pdf souborem a ukazovat tabulky, které je potřeba vyextrahovat

Tento postup je oproti klasickému rozpoznávání textu pomalejší, ale stále je natolik rychlý, aby uživatel nepoznal zpoždění. Oproti klasickému rozpoznávání textu je těžší naučit se nový font. Naproti tomu tento algoritmus převádí s naprostou přesností. Tento způsob je výrazně jednoduší naimplementovat.

Zbývá jen rozhodnout, jakým způsobem bude algoritmus pracovat s vyextrahovanými daty. Základním řešením je exportovat data do souboru ve formátu tabulky oddělené středníkem. Pokročilým řešením by bylo přidání abstrakční vrstvy sloužící ke zpracovávání dat.

### 4.1.5 Optimální řešení

Ideální je dohoda s výrobcem čipu, aby data, která stejně během vývoje čipu a jeho podpory používá, poskytl. V tomto případě se i přes snahu nepovedlo od výrobce data obstarat, ale po několika dalších vydáních svých nástrojů pro podporu mikrokontrolérů Texas Instruments změnil licenci poskytnutých souborů. Přesto jsem přesvědčen, že vytvořené hlavičkové soubory v rámci této diplomové práce jsou v tuto chvíli podrobnější a stylově čistější.

## 4.2 Generování hlavičkových souborů

Generátor je napsaný v jazyce Python s použitím objektového přístupu.

Generátor je rozdělený na dvě části. První část je specifická pro tento projekt. Druhá část je obecná a mohla by sloužit ke generování libovolných hlavičkových

souborů akceptovatelných RTEMS komunitou. Generátor je takto rozdělen kvůli úspoře času a i možnosti později upravit data v generované databázi. Hlavním důvodem rozdělení generovacího procesu na dvě části je obava o zbytečnou komplexitu generovacího algoritmu. V případě, kdy by generovací proces rozdělen nebyl, musel by být schopen zpracovat jakoukoli formu vstupních dat. To by mohlo vést k implementaci plné výjimek a přepínačů, která by se postupně nabalovala při každém přizpůsobení novému mikrokontroléru. Proto byl vytvořen datový standard, který je generovací část algoritmu schopna převést do hlavičkového souboru. Tento standardní formát se snaží vytvořit první čistící část generátoru.

### 4.2.1 Datové formáty generátoru

Při rozhodování o datové reprezentaci zpracovávaných dat byl brán v potaz formát XML a formát JSON. Formát JSON byl zvolený jako vhodnější na základě jeho snadné úpravy v libovolném textovém editoru a na základě jeho dostupné a snadno použitelné podpory v jazyku Python. Mezi další výhody JSON patří jednoduchost a zároveň velké možnosti uspořádání dat.

### 4.2.2 První část generátoru - standardizace dat

Vstupem do první části jsou data vyextrahovaná z referenčního manuálu, například pomocí skriptů jazyka Autohotkey. Tato data již mají strukturu typu JSON avšak odpovídají spíše formátu referenčního manuálu než formátu vyžadovaného generovací částí. Příklad této triviální úpravy je přiložen mezi zdrojovými soubory.

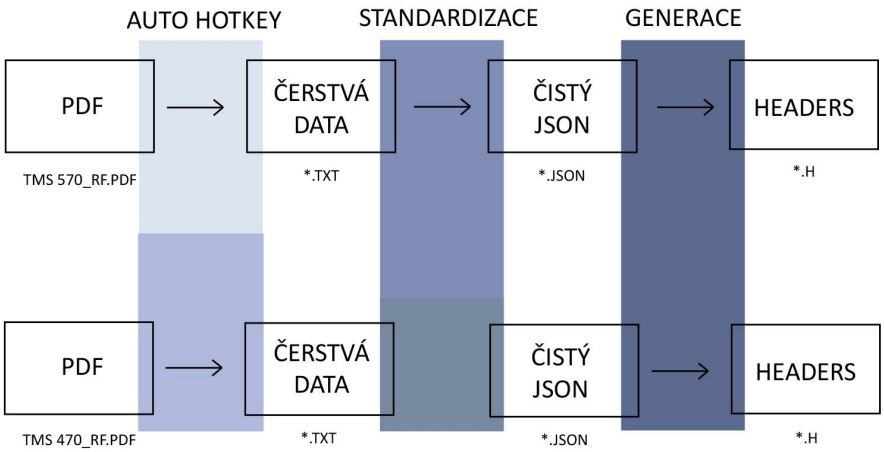
Ukázka, pro porovnání proti vstupnímu formátu je uvedana níže.

```
  "name" : "CRC",
      "full name" : "Cyclic Redundancy Check",
      "offset" : ["0xFE000000"],
      "registers" : [ {
        "name" : "CTRL0",
        "info" : "CRC Global Control Register",
        "lenght" : "32",
        "adress" : "0x0",
        "fields" : [ {
          "start_bit" : "8",
          "bit_lenght" : "1",
          "bit_Field_Name" : "CH2_PSA_SWREST",
          "info" : "Channel 2 PSA Software Reset."
          },
```

### 4.2.3 Druhá část generátoru - generace

Standardní formát vstupu do této části nedělá generaci složitou. Program byl navržen tak, aby bylo možno jednoduše měnit a opravovat výstupní formát hlavičkových souborů až do momentu, kdy RTEMS komunita odsouhlasí možné začlenění do projektu.

Obrázek 4.1 ukazuje celý postup generování hlavičkových souborů, při zachování stávajícího přístupu.




**Obrázek 4.1.** Sekvence generování hlavičkových souborů

Z obrázku by mělo být vidět, že narozdíl od skriptu autohotkey se proces standardizace mezi různými mikrokontroléry nemusí téměř měnit. Dále jsou zde znázorněny formáty souborů, což může velmi pomoci v orientaci mezi přiloženými soubory.

### 4.2.4 Hlavičkové soubory

Formát hlavičkových souborů, který nakonec schválila RTEMS komunita, obsahuje pro každou periferii jednu hlavní strukturu popisující všechny registry periferie. Posuny registrů jsou automaticky přepočítány a doplněny o rezervovaná místa. [7] Dále pak obsahuje vnitřní popis všech bitových polí každého registru a příslušná makra pro jejich čtení a nastavování. Při použití těchto maker může programátor dosáhnout vyšší čitelnosti a přenositelnosti aplikačního kódu. Pro každý registr a bitové pole je zde krátký popisující text. Možné fyzické hodnoty bitového pole se do hlaviček prozatím nedostaly.

Níže je uvedena ukázka struktury periferie hlavičkového souboru. Tato struktura je velmi podobná té, kterou přikládají výrobci mikrokontrolérů ke svým výrobkům.

```
typedef struct{
  uint32_t CTRL0;             /*CRC Global Control Register*/
  uint8_t reserved1 [4];
  uint32_t CTRL1;             /*CRC Global Control Register 1*/
  uint8_t reserved2 [4];
  uint32_t CTRL2;             /*CRC Global Control Register 2*/
  ...
  uint32_t RAW_DATAREGH2;     /*Channel 2 Raw Data High Register*/
  uint8_t reserved11 [128];
  uint32_t BUS_SEL;           /*Data Bus Selection Register*/
} tms570_crc_t;
```

Příklad smysluplného popisu vnitřních polí registrů, který v hlavičkových souborech firmy Texas Instruments chybí, je přiložen níže.

```
/*----------------------TMS570_CRC_CTRL2----------------------*/
/* field: CH2_MODE - Channel 2 Mode Selection */
#define CRC_CTRL2_CH2_MODE(val) BSP_FLD32(val, 8, 9)
#define CRC_CTRL2_CH2_MODE_GET(reg) BSP_FLD32GET(reg,8, 9)
#define CRC_CTRL2_CH2_MODE_SET(reg,val) BSP_FLD32SET(reg, val, 8, 9)

/* field: CH1_TRACEEN - Channel 1 Data Trace Enable.
 * When set, the channel is put into data trace mode. */
#define CRC_CTRL2_CH1_TRACEEN BSP_BIT32(4)

/* field: CH1_MODE - Channel 1 Mode Selection */
#define CRC_CTRL2_CH1_MODE(val) BSP_FLD32(val, 0, 1)
#define CRC_CTRL2_CH1_MODE_GET(reg) BSP_FLD32GET(reg,0, 1)
#define CRC_CTRL2_CH1_MODE_SET(reg,val) BSP_FLD32SET(reg, val, 0, 1)
```

Makra typu BSP_FLD32 jsou definována přímo v projektu RTEMS. Makro BSP_FLD32 v prvním případě umístí hodnotu val na osmý a devátý bit dvaatřiceti bitového čísla. Hlavní výhoda těchto maker tkví ve zjednodušení pro vývojáře. Vývojář si nemusí hledat, že se jedná o 8 a 9 bit. Ani nemusí hledat a ve svém kódu udržovat nepřehledné konstanty nebo definice.

Příklad pro vyčtení módu druhého kanálu v periferii CRC může vypadat například takto:

```
CRC_CTRL2_CH2_MODE_GET(TMS570_CRC->CTRL2)
```

## 4.3 Start operačního systému

Tato sekce je zaměřena na popis průběhu startu operačního systému RTEMS a alternativy, kterými je možné inicializaci a start systému realizovat.

Předání řízení operačnímu systému RTEMS je realizováno skokem na symbol _ARMV4_Exception_interrupt. Tento skok nemůže být proveden hned po startu mikrokontroléru, protože je potřeba mikrokontrolér připravit na běh aplikace a tato funkcionalita v operačním systému RTEMS zatím není obsažena. Jedná se například o zkontrolování integrity paměti ale i o možné předpřipravení periferií. O paměťový prostor mikrokontroléru se tedy většinou dělí dva nezávislé programy. Typicky se jedná o low_level_init a aplikaci, do které je v tomto případě připojen i operační systém RTEMS. Základním nedostatkem tohoto modelu je neschopnost sám sebe upravit nebo přeprogramovat. V praktických aplikacích je často požadavek, aby se do mikrokontroléru dala nahrát aktualizace stávající aplikace. Ideálním případem je nahrání kódu bez použití externího debugeru, protože zařízení již může být připojeno na nedostupném místě v jiném komplexním systému.

Z těchto důvodů je obvyklejší aplikační model zavaděč a aplikace. Zavaděč je nahrán na adresu nula a obsahuje základní funkcionalitu potřebnou k rozběhnutí mikrokontroléru (low_lv_init). Kromě této funkcionality bývá zavaděč schopen komunikovat po vybrané komunikační sběrnici, po které dokáže přijímat příkazy nebo aktualizace kódu.

Jádro CORTEX-R podporuje dvě možnosti umístění tabulek výjimek. První možné umístění tabulky se nachází na počátku adresového prostoru a proto se mikrokontrolér po startu či resetu ocitne na adrese 0. Druhá možnost jádra CORTEX–R4 je používat tabulku výjimek z adresy 0xFFFF0000. Používat druhou tabulku výjimek na adrese 0xFFFF0000 je bohužel v mikrokontroléru TMS570 nemožné,

protože adresa 0xFFFF0000 je rezervovaná pro periferie a není možné z ní vykonávat kód. Toto je velmi důležité, protože operační systém RTEMS dokáže zpracovat všechny výjimky z tabulky výjimek a je vhodné, aby to také prováděl. Avšak protože je nutné kód zavaděče vykonat jako první, nachází se zavaděč pravděpodobně v prostorech okolo tabulky výjimek a tabulka výjimek odkazuje na zavaděč.

Měnit obsah tabulky výjimek je značně komplikované, protože se tabulka nachází v paměti typu flash. Počet zápisů do paměti typu flash je omezen. Jednoduchým řešením tohoto problému je rozšířit zavaděč tak, aby skoky z tabulky výjimek přeposílal do aplikace. Nevýhodou tohoto řešení je i několika skokové zpoždění reakční doby na přerušení nebo jinou výjimku. Na mikrokontroléru TMS570 se může tento problém vyřešit pomocí periferie POM, která může tabulku výjimek překrýt virtuální stránkou paměti a tím tabulku přesměrovat bez nutnosti přepisovat flash paměť. Mechanizmus je však určen především pro ladění a pokud je využitý, tak není možné povolit ochrany proti chybám v paměti. Proto je potřeba podle aplikace a oblasti použití volit vhodnou alternativu. Pro ušetření omezeného počtu zápisových cyklů Flash a jednoduchost byla po dobu vývoje používaná především alternativa s využitím překryvu realizovaného POM a běh kódu z externí paměti SDRAM.

## 4.4 Návrh a začlenění TMS570 do RTEMS

V této části diplomové práce lze naleznout popis adresářové struktury projektu RTEMS.

Z projektového adresáře RTEMS je stále ještě patrné, že první implementace byla zaměřena na použitá jazyka ADA než jazyk C. V kořenovém adresáři je možné mimo konfiguračních a instalačních skriptů objevit i testovací příklady.

Tato diplomová práce se zabývá přidáním podpory a tedy podpůrných kódů pro mikrokontrolér TMS570 (BSP). Jádro mikrokontroléru ARM již bylo projektem RTEMS podporováno a složka pro BSP založené na arm jádře je umístěna v rtems/c/src/lib/libbsp/arm. Pokud by bylo potřeba přidat zcela novou architekturu, musela by se vytvořit příslušná složka v rtems/c/src/lib/libbsp, rtems/c/src/lib/libcpu ale hlavně i v rtems/cpukit/score/cpu. Složka libcpu obsahuje definice a podporu různých modelů jader mikrokontrolérů. Jako typický příklad lze uvést obsluhu paměti CACHE nebo způsob přímého přístupu do paměti (DMA). Obojí je charakteristické právě pro jádro mikrokontroléru. Tímto způsobem mohou dva mikrokontroléry se stejným jádrem sdílet stejný kód, přestože pocházejí od konkurenčních výrobců.

Pro založení nového BSP je typicky nejvhodnější použít existující, dobře udržované BSP a to zkopírovat. Odpovědností vývojáře je postupně upravovat soubor README, do které patří informace o aktuálním stavu BSP. Dále by měl soubor README obsahovat návod, jak nejjednodušeji BSP použít v reálné aplikaci a co všechno může být konfigurováno.

V souboru configure.ac je možnost nastavit programové definice, které jsou později předány do zkompilované aplikace. Pro správnou funkci operačního systému je zde potřeba nastavit například minimální velikost vlákna nebo hodinovou frekvenci mikrokontrolérů. Jak je vidět, všechny informace v této složce budou silně platformově závislé.

Pokyny, jak zkompilovat právě vytvářené BSP, obsahuje soubor Makefile.am. Zde je nutné nezapomenout na přidání skriptu pro linker pomocí:

```
EXTRA_DIST += startup/linkcmds.tms570ls3137_hdk_sdram
```

Zbylá organizace složky už záleží na uvážení vývojáře. V této práci bylo dbáno, aby byla rozdělena do složek podle logické funkcionality s jednou společnou složkou pro hlavičkové soubory. Cestu ke každému zdrojovému a hlavičkovému souboru je potřeba popsat v Makefile.am.

## 4.5 Integrace základních periférií do RTEMS

### 4.5.1 Sériové porty

Prvním krokem při přidávání podpory nové platformy, portaci operačního systému nebo testování nového programovacího jazyka obvykle bývá počáteční test typu Ahoj světe. Jednou z nejjednodušších možností, jak z programu uvnitř mikrokontroléru TMS570 pozdravit okolní svět, je použití periferie SCI (sériového komunikačního rozhraní).

Základní způsob komunikace s SCI periferií je velmi jednoduchý. Stačí na příslušné místo v paměťovém prostoru, kde je mapovaný registr TD (pošli data) v periferii SCI, zapsat písmeno, které se odešle na sběrnici. Dále je pak před každým dalším zápisem nutno kontrolovat příznak, jestli je v periferii místo na zapsání dalšího znaku. Příznaky přenosu můžeme naleznout v registru FLR v periferii SCI. V tomto případě stačí kontrolovat příznak TX EMPTY (prázdný výstupní buffer).

Po restartu mikrokontroléru jsou parametry sériové komunikace nastaveny na použití jednoho stopbitu a bez použití parity a s délkou přenášených dat 1 bit. Toto nastavení nemusí odpovídat v aplikací požadovanému nastavení a proto je potřeba přidat podporu mechanizmu, kterým si aplikace může zažádat o nastavení požadovaných parametrů. Počet stopbitů a paritu najdeme v registru GCR1. Délku přenášených a očekávaných dat můžeme hledat v registru FORMAT a nastavení rychlosti v registru BRS. Dále je ještě důležité nastavení přerušení v registru SETINT.

Po zvládnutí práce s periferií nastává čas pro integraci její podpory do operačním systému RTEMS. V případě, že v konfiguraci aplikace je nastaven příznak informující o potřebě použití konzole, operační systém RTEMS při inicializaci volá funkci console_initialize. V této funkci je potřeba inicializovat datové struktury subsytému termios a zaregistrovat sériové porty, které zařízení podporuje. V tomto případě se jedná o periferii SCI a periferii LIN, která taktéž dokáže komunikovat po sériové lince. Registrace se provádí funcí rtems_termios_device_install. Funkce má dva důležité vstupy. Odkaz na strukturu registrovaného portu, kterou bude operační systém předávat dále a tabulku funkcí, které operační systém bude volat. Jaký může mít tabulka charakter je zobrazeno níže.

```
const rtems_termios_device_handler tms570_sci_handler_interrupt  = {
  .first_open = tms570_sci_interrupt_first_open,
  .last_close = tms570_sci_interrupt_last_close,
  .poll_read = NULL,
  .write = tms570_sci_interrupt_write,
  .set_attributes = tms570_sci_set_attributes,
```

```
    .mode = TERMIOS_IRQ_DRIVEN
};
\
```

Důležitý parametr je mode, který udává, zda má operační systém pracovat s konzolí, jako řízenou přerušením nebo nikoliv. Pro potřeby této diplomové práce byly tyto struktury vyplněny oběma způsoby a uživatel si může způsob řízení konzole vybrat v konfiguračním souboru. Dalšími parametry jsou především funkce inicializace a deinicializace při prvním respektive posledním použití, čtení znaku, poslání znaku a nastavení parametrů přenosu.

Funkce nastavení parametrů obsahuje pouze práci s registry popsanými výše. Parametry pro nastavení portu do funkce předává operační systém pomocí struktury termios. Během vykonávání funkce je vhodné port zamknout použitím rtems_termios_device_lock_acquire.

Funkce pro otevření a zavření portu jsou zodpovědné za nastavení přerušení v periferii SCI a za registrování přerušení v operačním systému RTEMS.

Pro kompletní funkčnost ovladače je přidána podpora funkce printk. Jedná se o funkci, kterou obvykle používá jádro operačních systémů pro tisknutí ladících informací, chyb a podobných kritických dat. Funkce printk by neměla používat přerušení, protože může být potřeba v době, kdy přerušovací subsystém není ještě konfigurovaný. Mezi další použití funkce printk patří ladění uvnitř přerušení nebo informace o kritickém stavu operačního jádra. Podpora funkce printk je přidána do druhého souboru a oddělena od ovladače sériových portů. Do jádra systému RTEMS se vkládá pomocí definice BSP_output_char.

```
BSP_output_char_function_type BSP_output_char = tms570_uart_output;
BSP_polling_getchar_function_type BSP_poll_char = tms570_uart_input;
```

### 4.5.2 Časovač

Časovač je periferie, která je zásadní pro správné fungování operačního systému. Operační systém pomocí časovače řídí přidělování času úlohám a inicializuje přepínání úloh.

V mikrokontroléru TMS570 je hlavní časovací jednotka pro podporu operačních systémů nazývána RTI. Periferie RTI obsahuje dva nezávislé 64bitové čítače, podporu synchronizace s komunikací na sběrnici FlexRay a mnohé další.

Po časovači požadujeme, aby nezávisle a periodicky produkoval událost, kterou operační systém použije pro řízení svého běhu. V mikrořadiči TMS570 tohoto docílíme následujícím způsobem. Jednomu z čítačů určíme maximum, do kterého bude čítat předtím, než se vynuluje a začne čítat znova. Doba mezi dvěma vynulováními by mohla odpovídat například intervalu jedné mikrosekundy. Dále budeme pracovat s počtem, kolikráte již čítač dočítal do maxima (počet mikrosekund). Podle přání operačního systému nastavíme záchytný registr na dobu trvání jednoho časového kvanta. Záchytný registr vygeneruje událost v případě, že počet mikrosekund byl naplněn.

Pro účely portace stačí nastavit pouze předděličku čítače 0 v registru CPUC0. Tím zajistíme kulatou stabilní frekvenci čítání. Nastavíme registr TBCTRL tak, aby čítač počítal impulzy z výstupu předděličky. Nastavením registrů COMP0 a UDP0 docílíme periodického generování událostí. Registr COMP0 obsahuje číslo, při kterém čítač vygeneruje událost. Při vygenerování události se registr COMP0 automaticky zvýší o hodnotu registru UDP0. Jako poslední se nastaví povolení přerušení v registru SETINTENA. Čítač se spustí registrem GCTRL.

### 4.5.3 Pinmux

Periferie Pinmux slouží jako prostředník mezi téměř všemi periferiemi mikrokontroleru a výstupními piny.

V dnešní době je počet vstupně výstupních pinů mikrokontrolérů důležitý parametr. Počet pinů je většinou spjatý s počtem zařízení, které bude mikrokontrolér schopen obsluhovat či kontrolovat. Bohužel pouzdro PQFP s fyzickými piny pro mikrokontrolér je jedna z podstatných složek ceny mikrokontroléru. Naproti tomu pouzdra BGA většinou vyžadují vícevrstvé tištěné spoje a tím efektivně prodražují návrh a cenu desky. Z tohoto důvodu se firmy snaží naleznout kompromis mezi počtem pinů mikrokontroléru a zároveň udržení co nejvyšší použitelnosti mikrokontroléru. Pro účely zvýšení efektivity a vytíženosti pinů byla vytvořena transformační síť, která dovoluje výhradní kontrolu nad pinem a jeho přiřazení jedné z několika spjatých periferií. Periferie pinmux ovládá tuto síť.

V souboru pinmux.c je umístěna implementace obsluhy této periferie. Prozatím není provázána s jádrem operačního systému, ale protože pro vývojáře může mít tato periferie kritickou důležitost, jsou funkce obsluhující tuto periferii vyexportovány a připraveny k použití. Taktéž by podpora této periferie byla nutná pro případný start operačního systému RTEMS jako primárního kódu přímo přes reset vektor bez použití zavaděče.

### 4.5.4 VIM

Vektorový modul přerušení (VIM) je zásadní pro asynchronní obsluhu vnějších událostí. Modul VIM rozhoduje o prioritě jednotlivých přerušení, informuje o právě zpracovávaném přerušení a řídí možnosti probuzení po události.

Z teorie je zřejmé, že jádro mikrokontroléru CORTEX–R4 podporuje práci s přerušeními. CORTEX-4R rozděluje přerušení na rychlá (FIQ) a normální (IRQ). Pro oba druhy přerušení je rezervována speciální adresa, na které mikrokontrolér začne vykonávat rutinu přerušení. Nicméně, kromě zmíněného systému podporuje jádro CORTEX–R4 i možnost delegování kontroly nad přerušeními jinému modulu (VIM) v mikrokontroléru.

Pokud je v jádře CORTEX–R4 zapnuto přímé přesměrování přerušeních do modulu VIM (v registru CPSR), modul sám určí, které přerušení nastalo. Přerušení je automaticky vyhodnoceno a vykonávání programu se přesune na příslušnou obslužnou rutinu vyplněnou v tabulce přerušení bez skoku na standardní obsluhy přerušení IRQ a FIQ v tabulce výjimek jádra CORTEX–R4.

Přidání podpory přerušení do operačního systému se tedy musí řešit pro oba případy, s podporou řízení v modulu VIM i bez podpory.

Bez podpory modulu VIM je přizpůsobení operačnímu systému RTEMS velmi snadné a čisté. Operační systém je navržen tak, aby všechna přerušení procházela přes jeho jádro. Tímto postupem dokáže jádro RTEMS softwarově určovat a nastavovat hodnotu priority přerušení. Taktéž tento postup dává plánovači znatelně větší kontrolu nad plánováním vykonávaného kódu. Připojení se provede zapsáním instrukce skoku na funkci _ARMV4_Exception_interrupt do tabulky výjimek jádra CORTEX–R4 (adresa 24 - IRQ a 28 - FIQ).

Při zapnutí aktivního řízení přerušení z modulu VIM a aktivní událostí přerušení přejde mikrokontroler přímo na adresy vyplněné ve vektorové tabulce modulu VIM. Toto řešení je vhodné pro co nejrychlejší obsluhu přerušení bez zbytečné

zátěže, ale má nevýhodu, že nebudou provedené případné požadavky na přeplánování vzniklé během obsluhy přerušení. Návrat z obslužné rutiny nebo driveru je skokem přímo do kódu přerušené úlohy. Systém nemá nad obsluhou přerušení dohled. A proto pro zajištění správné práce plánovače RTEMS bylo nutné všechny specializované vektory nasměrovat opět na obecné vstupní body operačního systému RTEMS. Tedy, adresy ve vektorové tabulce modulu VIM nahradit skokem na jedinou rutinu _ARMV4_Exception_interrupt. Funkcionalita modulu VIM se tedy při kombinaci s operačním systémem RTEMS může zdát do značné míry redundantní.

Po vykonání náležitostí v rutině _ARMV4_Exception_interrupt je ze systému předán běh zpět do ovladače mikrokontroléru. Funkce bsp_interrupt_dispatch musí rozhodnout o původu přerušení a předat řízení funkci bsp_interrupt_handler_dispatch. Který kanál vyvolal přerušení, můžeme vždy najít v registru IRQINDEX z modulu VIM. Přerušení od jednotlivých periferii v microkontroléru TMS570 vždy vedou přes modul VIM a proto by měla být hodnota IRQINDEX aktuální i při nepoužívání řízení přerušení modulem VIM.

## 4.6 ETHERNET

Výběr sítové komunikační knihovny byl konzultován s komunitou vývojářů systému RTEMS. Do hlavního výběru byla uvažována vnitřní knihovna RTEMSu, knihovna BSD a knihovna LwIP. Komunita RTEMS před časem vložila velké úsilí do vlastní knihovny, která je přímo zavedena v jádře operačního systému. Bohužel se ukázalo, že udržování takhle komplikované knihovny je nad komunitní síly, proto je nyní tendence podporovat ostatní komunikační knihovny od třetích stran. Knihovna BSD se kvůli svým paměťovým nárokům nehodí pro použití v aplikacích s mikrokontrolérem TMS570. Vnitřní knihovna RTEMS nebyla vybrána pro její možné brzké zrušení z vývojové větve RTEMS. V rámci této diplomové práce byl stvořen ovladač pro knihovnu LwIP. Ovladač do knihovny LwIP by měl splňovat následující tři funkce. Přizpůsobení hardwarové platformě, přizpůsobení operačnímu systému a přizpůsobení knihovně LwIP.

### 4.6.1 Přizpůsobení operačnímu systému

Přizpůsobení operačnímu systému je nejjednodušší část. Knihovna LwIP vnitřně pracuje s obvyklými nástroji operačních systémů, jako jsou vlákna, semafory, fronty a podobné. V případě, že tato část nebude implementována, LwIP použije vlastní implementaci dříve zmíněných softwarových konstruktů. Tato implementace bude sice funkční, ale neefektivní nebo založená na režimu opakovaného dotazování.

Při vkládání podpory operačního systému je důležitá dvojice souborů sys_arch.c a sys_arch.h. Zde je vhodné naimplementovat synchronizační a další systémové funkce vyžadované knihovnou LwIP s využitím mechanizmů a služeb podporovaných cílovým operačním systémem a hardwarovou platformou. V tomto případě s využitím semaforů, mutexů a front systému RTEMS. Z důvodu rozdílného zacházení s těmito strukturami nebylo vhodné provádět redefinici jedna ku jedné. RTEMS při předávání zmíněných konstruktů nepoužívá přímý odkaz na konstrukt, ale používá vnitřní identifikátor. Z hlediska bezpečnosti je přístup operačního systému RTEMS znatelně lepší, protože uživatelské aplikaci nedává možnost pracovat

se zmíněnými konstrukty operačního systému bez volání vyexportovaných systémových funkcí. Pro párování konstruktů operačního systému RTEMS s konstrukty LwIP bylo potřeba vytvořit příslušné struktury (např. port_mutex_t).

Navíc proti požadavkům LwIP byla do hlavičkového souboru přidána funkce pro práci s datovými bariérami. Této funkce bude zapotřebí při dalším přizpůsobování v případech, kdy mohou vznikat datové hazardy. Dále by se měly do souboru umístit funkce pro označování a práci s kritickými sekcemi. Obecně se kritické sekce vyznačují tím, že do nich nesmí vstoupit dvě vlákna najednou. Při použití se může například jednat o práci s nedělitelným hardwarem nebo o konzistenci datových struktur. Jednoduchou implementací kritických oblastí je zakázání přerušení a tím i přeplánování běhu vláken.

Implementace dříve zmíněných funkcí a funkcí pro práci s konstrukty operačního systému je implementována v souboru sys_arch.c. Protože kód bude součástí veřejného projektu či různých aplikací, je vhodné dbát na úpravu kódu, kontrolovat návratové hodnoty z funkcí a držet se zásad psaní čistého kódu. Ve většině případech si aplikační rozhraní knihovny LwIP a rozhraní operačního systému RTEMS odpovídala a proto je ve funkcích pouze převod volání z jednoho aplikačního rozhraní do druhého. Výjimku tvoří funkce čekání na semafor a frontu. V operačním systému RTEMS se nepracuje s časem, který funkce strávila čekáním na uvolnění nebo data. Proto tato funkčnost musela být implementována pomocí funkce rtems_clock_get_uptime_nanoseconds. Poslední odlišností je žádost knihovny LwIP čekat při vkládání do fronty do doby, než bude ve frontě místo. Tato funkcionalita je realizována pomocí přidaného čítacího semaforu, který znázorňuje volné místo ve frontě.

Naposledy je ještě vhodné zmínit existenci souboru cc.h, který obsahuje definice datových typů podle použitého kompilátoru a C knihovny. Tento soubor je dále důležitý pro definování návratových hodnot z funkcí knihovny LWIP. Pro účely této práce byly standartní chybové kódy použity z operačního systému, tak aby nedocházelo ke kolizím ve jménech. Tohoto bylo dosaženo pomocí definice LWIP_PROVIDE_ERRNO. Definice návratových hodnot jsou v RTEMSu v souboru sys/errno.h. Dále jsou v souboru cc.h definice potřebné pro kompilátor.

### 4.6.2 Přizpůsobení hardwarové platformě

V této části je hlavním cílem inicializovat periferie MDIO a EMAC a připravit je na příjem a odesílání dat. Periferie MDIO se stará o komunikaci s PHY. Hlavní úlohou periferie EMAC je příjem a odesílání síťových rámců.

Práce s periferii MDIO je velmi přímočará. Je ponecháno výchozí nastavení a při startu inicializace se pouze kontroluje, zda je spojení aktivní. Kontrola přerušení a případná reinicializace spojení je ponechána k budoucí implementaci.

Periferie EMAC je na rozdíl od MDIO poněkud obsáhlejší. Z důvodu zrychlení odezvy a snížení výpočetních nároků byla zvolena práce s přerušeními. EMAC obsahuje osm vstupních a osm výstupních kanálů. K implementaci základní komunikační funkcionality byl vybrán pár kanálů na pozici nula. Veškerá komunikace tedy prochází touto dvojicí kanálů, je zpracovávána periferií EMAC a předána do knihovny LwIP.

Funkce potřebné pro nastavování registrů periferie jsou umístěny v souborech ti_drv_emac.h a ti_drv_emac.h a jejich příslušných zdrojových souborech. Tyto krátké funkce jsou převzaté z kódu generovaného nástrojem HalCoGen od firmy Texas Instruments. Současná verze již licenčně umožňuje sloučení s projektem

RTEMS. Nicméně jsou tyto funkce celkově přepsané, aby odpovídaly novým definičním hlavičkovým souborům mikrokontroleru.

Inicializace obsahuje možnost nastavení promiskuitního režimu, určení výchozího kanálu pro příjem všesměrových zpráv, povolení přerušení pro daný kanál, výběr mezi jednocestnou a oboucestnou komunikací a nakonec povolení přerušení na periferii EMAC. Po tomto nastavení je periferie EMAC připravena k použití a při vyplnění adresy prázdného a připraveného spojitého seznamu do registru RX_HDP začne periferie EMAC zapisovat přijatá data.

### 4.6.3 Přizpůsobení LwIP knihovně

Tato část ovladače spojuje knihovnu LwIP s hardwarovým rozhraním mikrokontroleru TMS570. Cílem je vytvořit datové struktury pro přenos dat mezi zmíněnými rozhraními a příslušnou funkcionalitu pro jejich obsluhu.

Knihovna LwIP na vstupu očekává zřetězený seznam typu pbuf, který obsahuje všechny nově přijaté pakety určené ke zpracování. Na výstupu očekává knihovna LwIP funkci, která podobný zřetězený seznam odešle.

Periferie EMAC příchozí komunikaci ukládá do softwarem připraveného zřetězeného seznamu, který není kompatibilní se strukturou pbuf, používanou knihovnou LwIP. Pro účely transformace mezi jednotlivými formáty byly vytvořeny struktury emac_tx_bd a emac_rx_bd. Tyto struktury jsou kompatibilní s hardwarovými požadavky periferie EMAC a zároveň v sobě nesou odkaz na spjatou strukturu pbuf. Tento způsob značně ulehčuje práci s vyhledáváním pbufu a následnou transformaci. Cenou za rozšíření této struktury o odkaz na víc je snížení maximálního množství dostupných struktur o čtvrtinu a zhoršení přenositelnosti kódu na jiný mikrokontrolér s jiným typem ETHERNET periferie.

Pro účely přenášení dat je v systému při inicializaci ovladače zaregistrováno další vlákno a semafor. Při přerušení od periferie EMAC je nastaven semafor, na který čeká zmíněné vlákno. Tímto způsobem je dosaženo minimální doby strávené v přerušení a zvýšení suverenity operačního systému nad řízením běhu aplikace. V tomto vlákně se z periferie EMAC postupně vyčítá příchozí paket. Pro identifikaci paketu se používají příznaky začátek paketu (SOF) a konec paketu (EOF) nastavované periferií EMAC při fyzickém přijmu dat. Při nalezení obou příznaků se všechny části paketu, reprezentované strukturami typu emac_rx_bd, mezi oběma příznaky vyjmou z příchozí fronty. Pomocí odkazu uvnitř struktury se dohledají příslušné struktury pbuf a zřetězí se do spojového seznamu tak, aby struktury pbuf tvořily logicky stejnou informaci jako v původních strukturách emac_rx_bd. Odkaz na tento nově zformovaný řetězec se odešle ke zpracování LwIP knihovně pomocí vstupní funkce.

Struktury emac_tx_bd a emac_rx_bd jsou vytvořeny ve speciální části adresového prostoru, který odpovídá periferii EMAC, a tím periferii EMAC zajišťují snadnější přístup a operace nad zmíněnými strukturami. Pro tuto část ovladače je velmi kritické, jak s danými strukturami bude zacházeno a jak budou reprezentovány. Z teorie je zřejmé, že registr RX_HDP musí ukazovat na první volnou strukturu, která je navíc součástí ukončeného zřetězeného seznamu těchto struktur.

Vnitřní reprezentace zřetězených struktur emac_rx_bd byla volena z několika možností. Jako nejefektivnější možnost se jeví každému komunikačnímu kanálu přiřadit skupinu struktur a tuto skupinu rozdělit na dvě části. Jedna zřetězená část již má alokovaný a prázdný buffer připravený k příjmu. Začátek této fronty je obsahem registru RX_HDP příslušného kanálu. Druhá zřetězená část obsahuje

volné struktury, které se používají pro doplnění první fronty v momentě, kdy se začnou přijímat data. Výhoda i nevýhoda tohoto přístupu je patrná při použití více komunikačních kanálů současně. V případě, kdy jeden komunikační kanál bude výrazněji vytěžován, může nastat kritická situace a struktury dojdou. Výhodou je, že každý kanál má předem rezervovaný počet struktur a nemůže se stát, že jeden kanál si během vytížení přivlastní všechny dostupné struktury ostatních kanálů. Druhý přístup se od prvního liší tím, že nepoužívá předem daný počet struktur na komunikační kanál a fronta volných struktur je společná pro všechny. Poněkud odlišný způsob vnitřní reprezentace zmiňovaných zřetězených struktur tkví v udržování zřetězeného zacykleného seznamu všech struktur. V něm si udržujeme odkaz na první připravenou a první volnou strukturu. Protože je potřeba udržovat o několik odkazů méně, je tento způsob méně paměťově náročný. Avšak klade mnohonásobně větší nároky na strojový čas, protože je neustále potřeba udržovat konzistenci a smysluplnost zmíněného cyklického seznamu. Implementace ovladače v této práci používá první způsob a pro jeho realizaci jsou vytvořeny struktury rxch a txch. Deklarace všech struktur jsou v souboru tms570_emac.h. Odesílání paketů z knihovny LwIP je řešeno analogicky k přijímání dat. Jediná výjimka je, že obslužný kód běží v strojovém čase obslužného vlákna knihovny LwIP a ne ve vlastním samostatném vlákně.

Systém předávání interních dat v ovladači a parametrů mezi ovladačem a knihovnou LwIP je realizován pomocí struktury netif. Tato struktura je v kompetenci knihovny LwIP a ze zásad objektového přístupu není vhodné ji měnit jinak, než přes vyexportované nastavující funkce. To však neplatí pro její položku state, kam se při inicializaci ovladače může nahrát vnitřní struktura tms570_netif_state. Tato struktura obsahuje záhlaví již zmíněných zřetězených seznamů struktur emac_tx_bd a další interní informace jako jsou odkazy na periferie EMAC a MDIO. Smyslem takovéto práce s daty je pokus o udržování objektového přístupu, v tomto případě se jedná o větší kontrolu nad strukturou předávaných dat.

Kontrola nad alokací struktur pbuf je kompletně v režii knihovny LwIP. Vzhledem k faktu, že struktury pbuf se používají po celé knihovně LwIP a že jedna a tatáž instance struktury je předávána mezi mnoha vrstvami knihovny či vlákny operačního systému, je struktura pbuf sdílená datová oblast a je potřeba k ní přistupovat jako ke kritické. Při správném přizpůsobení operačnímu systému a vhodném nastavení knihovny LwIP je uvolňování a alokování struktury pbuf považováno za bezpečné i z jiného vlákna nebo dokonce kontextu přerušení. Avšak stále je potřeba dbát na správné uvolňování již použitých pbuf struktur po odesílání paketu. Stejně tak kritické je správně uvolňovat paměť po špatné alokaci struktury pbuf. Velký pozor by měl být věnován i místu odkud se pbuf_alloc volá. Knihovna LwIP obsahuje možnost při uvolnění místa v paměti pro struktury pbuf zavolat uživatelskou funkci (např. tms570_eth_memp_avaible). Pokud by tato zpětná funkce volala přímo pbuf_alloc, může snadno dojít k rekurzivnímu zacyklení, protože zpětná funkce může být volána i z vnitřku špatně použitého pbuf_alloc. Z tohoto důvodu není v této práci zpětné funkci přiřazena vysoká priorita. Jediná možnost, jak si být jistý korektností použití pbuf_alloc, je počítání místa, které je ještě volné k alokaci.

## 4.7 Testování

Testování v rámci této diplomové práce probíhalo v několika cyklech. Během implementování byly periodicky spouštěny testy poskytnuté operačním systémem RTEMS. Tyto testy jsou ve složce testsuit/samples. Pro účely testování síťové komunikace byl vyvinut další software běžící na osobním počítači. Program se snažil zjistit celkovou propustnost, rychlost a správnost sítové komunikace.

Během testování a implementace bylo mnoho času stráveno nad řešením problémů.

Kód programu se špatně nahrával do paměti procesoru. Jednalo se pouze o nahrávání a běh z externí paměti. Z vnitřní paměti flash stejný kód běžel bezchybně.

Chyba byla způsobena pravděpodobně špatnou funkčností instrukce pro nezarovnaný přístup z ladícího a nahrávacího programu.

Chyba se opravila programováním a kontrolováním kódu vždy po 32-bitových slovech.

Při běhu aplikace tuto chybu odhalovalo samotné jádro, protože zpravidla na konci programových sekcí byly neplatné instrukce, případně skoky na nesmyslná místa v paměti.

Implementace podpory časovače a konzole probíhala v několika opakováních, protože jejich podpora se v operačním systému měnila. Toto není nic kritického a je to běžné v rozsáhlých projektech. Nakonec smyslem otevřených projektů je, aby různí vývojáři sami opravovali stávající porušenou funkcionalitu při přechodu na nové verze.

Podstatná rána byla implementaci uštědřena při testování funkčnosti LwIP knihovny. Po zahájení komunikace modulem EMAC začal operační systém registrovat přicházející přerušení s neexistujícím indexem nula. Stále není jisté, co je příčinou tohoto chování. Experimentálně bylo vyzkoušeno, že modul EMAC opravdu vnitřně oznámí modulu VIM, že žádá o přerušení. Modul VIM přerušení uzná, přeruší jádro CORTEX–R4 a řízení se předá do obslužné rutiny. Bohužel, když se rutina dotazuje, které přerušení je aktivní, tak modul VIM žádné přerušení nehlásí.

Prvním řešením bylo přidání výjimky do tabulky přerušení a pro kanály odpovídající příchozímu i odchozímu rámci v periferii EMAC přidat vlastní obslužnou rutinu. Takováto rutina si na zásobník uložila dodatečné informace a předala řízení klasické obslužné rutině přerušení. Toto řešení se ve výsledné implementaci nepoužilo, protože bylo potřeba pracovat s nestandartními výjimkami v operačním systému.

Čistějším řešením bylo přepsat implementaci na použití jiného modelu řízení přerušení. Ukázalo se, že tato chyba neplatí pro model řízení, kdy se periferie VIM nepoužívá.

Dalším poněkud nečekaným problémem při testování sítové komunikace bylo náhodné zamrzávání aplikace při obvyklém umístění paměti pro struktury pbuf. Místo pro struktury pbuf je staticky alokováno a původně se umísťovalo do stejné paměti jako programovaný kód, tedy externí paměti. Vzhledem k velikosti tohoto místa by bylo téměř vždy vhodné, aby se umístilo do větší externí paměti.

Řešením tohoto problému byla výjimka v ovladači LwIP, která určovala přesné místo alokace struktury do vnitřní datové paměti. Knihovna LwIP je na tuto alternativu dobře připravena a proto se jednalo hlavně o konfigurační problém.

Zběžná testování nové revize mikrokontroléru TMS570 ukazují, že se některých ze zmíněných chyb mohlo vyvarovat použitím novější revize mikrokontroléru. Tím by odpadly problémy například s nahráváním programu do externí paměti nebo by již nebylo nutné přemísťovat struktury pbuf do vnitřní paměti.

# Kapitola 5
# Závěr

Náplní této diplomové práce bylo navrhnout, implementovat a začlenit podporu mikrokontroléru TMS570 od firmy Texas Instruments do otevřeného projektu operačního systému RTEMS.

Nejprve byly vygenerovány hlavičkové soubory. V době realizace diplomové práce byly jediné existující hlavičkové soubory pod licencí neakceptovatelnou otevřeným projektem RTEMS. Proto bylo vymyšleno a vyzkoušeno několik způsobů, jak efektivně zpracovávat text a generovat nové hlavičkové soubory. Dalším výstupem je taktéž generátor hlavičkových souborů podle požadavků a zvyklostí komunity RTEMS a datový formát informací o periferiích, který zachovává veškeré podstatné informace a zároveň je snadno čitelný a upravitelný člověkem i automatem.

V dalším kroku byly hlavičkové soubory použity k implementování podpory základních periferii mikrokontroléru TMS570 do operačního systému RTEMS. Kromě základního časovače a sériového komunikačního rozhraní, které bylo požadováno zadáním, byla přidána podpora modulu zpravujícího vektorové přerušení (VIM) a mapování periferií na piny mikrokontroléru (PINMUX). Detailní postup vkládání podpory zmíněných periferii do operačního systému RTEMS je v práci zdokumentován a může sloužit jako návod při vytváření podpory pro jiný mikrokontrolér.

Začlenění podpory mikrokontroléru TMS570 do vývojového stromu RTEMS proběhlo úspěšně. Začlenění proběhlo až po několika kolech schvalování, kdy se několik vývojářů vyjadřovalo a diskutovalo o kvalitě a úpravě kódu, bezpečnosti implementace a celkové integrity s projektem RTEMS.

Po několika dalších komunitních rozhovorech byl navrhnut příklad preferovaného hlavičkového souboru. Do hlavní vývojové větve se tímto dostaly hlavičkové soubory popisující ostatní periferie mikrokontroléru TMS570 v příslušném dohodnutém formátu. Formát hlavičkových souborů je úplnější než ten poskytovaný firmou Texas Instruments. A ačkoliv jsou nyní hlavičkové soubory generované firmou Texas Instruments dostupné pod RTEMS kompatibilní licencí, hlavičkové soubory vzniklé v rámci této práce jsou pro vývojáře snadnější na použití a jejich používáním vzniká přehlednější a čistější kód.

Poslední implementační částí diplomové práce byl ovladač pro síťovou knihovnu LwIP. Této knihovně byla dána přednost před knihovnou BSD a původní vestavěnou síťovou knihovnou v projektu RTEMS. V textu této práce je základní návod k implementaci LwIP ovladače a popis jeho částí kritických pro stabilní běh. Dokumentace knihovny LwIP je do značné míry minimalistická a proto může tato diplomová práce sloužit i jako jednoduchý úvod do mechanizmu knihovny LwIP. Kromě návodu na tvorbu ovladače byl velký důraz kladen na korektní umístění a volání funkcí, efektivitu, stabilitu a přenositelnost.

Implementace byla během práce i po skončení úspěšně testována obecnými testy již obsaženými v projektu RTEMS. Pro otestování ETHERNETového rozhraní a

integrované podpory TCP/IP v LwIP byla napsána aplikace na počítač, která se snažila co nejvíce zatížit testovanou desku TMS570LS31xHDK. Během testování se vyskytlo několik zásadních problémů, ale po vyladění implementace se výsledná podoba chovala vždy podle očekávání a stabilně i při dlouhodobé zátěži.

# Literatura


[1] RTEMS Real Time Operating Systém RTOS, leden 2016.
https://www.rtems.org/ .

[2] Gerhard Wenderlein. Mikrokontroléry pro pohon hybridních a elektrických automobilů ,červen 2013.
http://pandatron.cz/?1722&mikrokontrolery_pro_pohon_hybridnich_a_elektrickych_automobilu/
.

[3] Texas Instruments. Overview for Hercules TMS570 MCUs ,leden 2016.
http://www.ti.com/lsds/ti/microcontrollers_16-bit_32-bit/c2000_performance/
safety/tms570/overview.page/ .

[4] ARM Ltd. Cortex-R4 Processor, leden 2016.
http://www.arm.com/products/processors/cortex-r/cortex-r4.php .

[5] Texas Instruments. TMS570LS31x HDK Kit, leden 2016
http://processors.wiki.ti.com/index.php/TMS570LS31x_HDK_Kit .

[6] lwIP Wiki,leden 2016.
http://lwip.wikia.com/wiki/LwIP_Wiki/ .

[7] RTEMS. Hlavičkové soubory mikrokontrolétů TMS570,listopad 2015.
https://github.com/RTEMS/rtems/tree/master/c/src/lib/libbsp/arm/tms570/
include/ti_herc/ .

# Příloha A
# Zadání práce

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

## ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Přemysl Houdek**

Studijní program: Otevřená informatika
Obor: Počítačové inženýrství

Název tématu: **Portace real-time systému RTEMS na rodinu mikrokontrolérů TMS570**

Pokyny pro vypracování:

1. Navrhněte a začleňte do operačního systému RTEMS základní podporu mikrokontrolérů řady TMS570 firmy Texas Instruments.2. Zprovozněte základní periferie mikrokontroléru (časovač, sériové porty).3. Při návrhu dodržujte správci projektu RTEMS specifikované požadavky na formátování a doporučené programové konstrukce tak, aby bylo možné začlenění práce do hlavního vývojového stromu projektu RTEMS.3. Připravte hlavičkové soubory i pro další periferie mikrokontroléru s využitím skriptů a manuálů výrobce.4. Implementujte a začleňte do projektu podporu síťového rozhraní ETHERNET mikrokontroléru TMS570.5. Zdokumentujte implementovaný kód a testování systému na vývojovém kitu TMS570.

Seznam odborné literatury:

[1] Texas Instruments: TMS570LS31x/21x 16/32-Bit RISC Flash Microcontroller Technical Reference Manual, 2013[2] RTEMS project documentation and On-Line Library: Applications C User's Guide, POSIX API User's Guide

Vedoucí: Ing. Pavel Píša, Ph.D.

Platnost zadání: do konce letního semestru 2015/2016

L.S.

V Praze dne 23. 1. 2015