České vysoké učení technické v Praze

FAKULTA ELEKTROTECHNICKÁ

Katedra řídicí techniky

DIPLOMOVÁ PRÁCE

Podpora External Mode v mikrokontroleréch

Praha, 2010

Autor:Ondřej Smítka

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne _____ _____

podpis

Poděkování

Na tomto místě bych rád poděkoval Ing. Liborovi Wasniowskému, Ph.Dr., vedoucímu mojí diplomové práce , za jeho cenné rady, odbornou pomoc a snahu vyhovět mým dotazům. Můj dík patří také mé rodině a přátelům za jejich psychickou pomoc a pevné nervy.

Abstrakt

Tato diplomová práce se zabývá podporou External mode v mikrokontrolérech. External mode je speciálním komunikačním módem Simulinku pomocí něhož můžeme sledovat a řídit průběh simulace v Simulinku, i když je výpočet modelu spuštěn na mikrokontroléru.

V práci jsou popsány možnosti využití External Mode při návrhu řídícího systému, vytvoření podpory pro MCU, popis architektury a komunikačního protokolu External Mode. External Mode podporuje komunikaci prostřednictvím sériové linky RS-232, která byla použita při portování podpory na mikrokontrolér 56F8367. Jádrem práce je vytvoření podpory External Mode na mikrokontrolér prostřednictvím komunikace po sběrnici CAN. V závěru práce je nastíněna možnost podpory jiného komunikačního protokolu (JTAG/OnCE) pro External Mode jako podklad pro další projekt.

Abstract

This thesis deals with the support of External Mode in microcontrollers. External mode is a special mode of communication with Simulink that allows us to monitor and control the course of simulation in Simulink, even if the calculation model is executed on a microcontroller.

The thesis describes the possibility of using External Mode in the design of the control system, build support for MCU, describes architecture and communication protocol of the External Mode. External Mode supports communication via serial RS-232, which was ported on 56F8367 microcontroller. The core work is to build support for External Mode to microcontroller using communication via CAN bus. The support of the External Mode via different communication protocol (JTAG/OnCE) is outlined in the end of the work. This description is intended to be basic for the next project.

zadani

Obsah

1 Úvod 1
2 External Mode 3
2.1 Podpora External Mode na MCU 4
2.1.1.RTW 5
2.1.1.1 Target 5
2.2 Použití a nastavení External Mode 6
2.2.1.Výběr simulace a ovládání z panelu Simulinku 6
2.2.2.Řídící panel External Mode 8
2.2.3.Vložené a nevložené parametry 12
2.3 Architektura Klient / Server 13
2.3.1.Použití sériového komunikačního protokolu 15
2.3.1.1 Nastavení vstupních parametrů MEX-souboru 15
2.3.2.Zdrojové soubory External Mode 16
2.3.2.1 Zdrojové soubory pro host 17
2.3.2.2 Zdrojové soubory pro target 18
2.3.2.3 Další soubory 20
2.4 Komunikační protokol External Mode 20
2.4.1.Struktura rámců 20
2.4.2.Druhy rámců 22
2.4.3.Použití a význam rámců při komunikaci 24
2.4.3.1 Popis významu rámců 26
2.4.3.2 Inicializace komunikace 27
2.4.3.3 Průběh komunikace 29
3 Portování podpory External Mode pro MCU s komunikací po RS-232 31

3.1 Postup portování transportní vrstvy....31
3.2 Mikrokontrolér Freescalle 56F8367....32
3.2.1.Práce s mikrokontrolérem....32
3.2.1.1 Processor Expert (PE)....32
3.2.1.2 Vytvoření projektu....33
3.2.1.3 Přidané beany....33
3.2.1.4 Přidání zdojových souborů a nastavení....34
3.2.1.5 Nastavení preprocessoru....36
3.2.2.Komunikační funkce na CPU....37
3.2.3.Modifikace zdrojových souborů External Mode....38
3.2.4.Demonstrační úloha....39
4 Implementace podpory External Mode pro MCU s komunikací CAN....40
4.1 Nutné předpoklady implementace....40
4.2 Sběrnice CAN....41
4.3 Target....43
4.3.1.1 Změna rámců External Mode po sběrnici CAN....43
4.3.2.CAN na MCU....45
4.3.2.1 FlexCAN interface....45
4.3.2.2 FlexCAN bean....46
4.3.2.3 Přidání zdrojových souborů do projektu PE....48
4.3.3.Zdrojové soubory External Mode target....49
4.4 Host....51
4.4.1.USB-CAN převodník....53
4.4.1.1 Instalace USB-CAN převodníku....53
4.4.1.2 Připojení převodníku na sběrnici....53
4.4.1.3 Softwarová podpora pro Windows....54
4.4.2.Změna zdrojových souborů External Mode (host)....56
4.4.3.MEX-soubor....58
4.4.3.1 Tvorba MEX-souboru....59
4.4.3.2 MEX-soubor pro sériovou podporu External Mode....60
4.4.3.3 MEX-soubor pro CAN podporu External Mode....60
4.4.3.4 Aktualizace souborů v Matlabu....61
4.4.3.5 Vstupní parametry pro MEX-soubor transportní vrstvy CAN....62
4.4.3.6 Debug MEX-souborů....62

4.4.4.Registrace transportní vrstvy v softwaru Simulinku 63
4.4.5.Demonstrační úloha a testování výkonových parametrů 65
4.4.5.1 Demonstrační úloha 65
4.4.5.2 Testování výkonových parametrů 67
5 Možnosti použití JTAG/ONCE komunikace.....69
5.1 JTAG/OnCE pro Freescale 56F800.....70
5.1.1.JTAG.....70
5.1.2.OnCE modul 72
5.2 JTAG pro External Mode 72
5.2.1.Porovnání komunikačních protokolů 72
5.2.2.Podpora JTAG/OnCE hardwaru 73
5.2.3.Host 74
5.2.4.Target 74
6 Závěr 75
Použitá literatura 77

Seznam příloh

A Obsah přiloženého CD 79

Seznam obrázků

Obrázek 1: Generování kódu „Převzato z [4]“.....5
Obrázek 2: Simulace v External Mode.....6
Obrázek 3: Připojení k targetu.....7
Obrázek 4: Spuštění simulace 8
Obrázek 5: Změna parametrů.....9
Obrázek 6: Výběr signálů a spouštění.....10
Obrázek 7: External Mode uládání dat.....11
Obrázek 8: Možnosti v subsystémech 12
Obrázek 9: Architektura External Mode.....14
Obrázek 10: Nastavení parametů pro MEX-soubor.....16
Obrázek 11: Kód External Mode.....16
Obrázek 12: Formát Header rámce.....21
Obrázek 13: Formát Data rámce.....21
Obrázek 14: Inicializace komunikace.....24
Obrázek 15: Průběh simulace.....25
Obrázek 16: Příklad sériové komunikace.....26
Obrázek 17: Nastavení Beanu AS1.....34
Obrázek 18: PE - přidané uživatelské soubory.....36
Obrázek 19: Nastavení preprocesoru v CodeWarrioru.....37
Obrázek 20: Struktura CAN rámce „Převzato z [9]“.....42
Obrázek 21: Target - změny související s novou komunikační vrstvou CAN.....43
Obrázek 22: Upravené rámce External Mode pro CAN.....44
Obrázek 23: FlexCAN interface „Převzato z [7]“.....45
Obrázek 24: Nastavení beanu CAN1.....46

Obrázek 25: Host - změny související s novou transportní vrstvou CAN.....52
Obrázek 26: USB-CAN informační panel.....53
Obrázek 27: Stavový diagram softwaru DLL „Přezato z [10]“.....54
Obrázek 28: Výběr transportní vrstvy CAN v nastavení.....64
Obrázek 29: JTAG.....70
Obrázek 30: JTAG signál.....71

Seznam tabulek

Tabulka 1: Zdojové soubory pro sériovou RS-232 komunikaci.....39
Tabulka 2: Zapojení konektoru FlexCAN 1.....45
Tabulka 3: Zapojení pinů CAN konektoru DB-9.....54
Tabulka 4: Modifikované soubory External Mode pro CAN.....65
Tabulka 5: Porovnání výkonových parametrů.....68
Tabulka 6: Komunikační protokoly pro External Mode.....73

Kapitola 1

1 Úvod

Tato diplomová práce se zabývá vytvořením podpory **External Mode** v mikrokontrolérech. External Mode definuje softwarové možnosti komunikačního spojení mezi modelem v Simulinku a kódem modelu, který je spuštěn na cílovém hardwaru.

V dnešní době při návrhu a vývoji každého regulačního systému je potřeba kontrolovat správnost řídícího algoritmu a možnost rychlé změny nastavení parametrů (regulátoru). V základní fázi vývoje je softwarovou simulací ověřena základní správnost návrhu algoritmu na modelu soustavy. V dalším kroku návrhu regulačního systému je již řídící algoritmus přímo testován na fyzickém hardwaru.

Řídící algoritmus je namodelován v Simulinku ve formě blokového schématu. Z tohoto blokového schématu je možné nástrojem RTW (Real Time Workshop) [1] vygenerovat analogický model, který je vyjádřen ve formě zdrojového kódu nejčastěji v jazyce C. Vygenerovaný kód řídícího algoritmu je možné spustit na fyzickém hardwaru. Tento návrh je možné testovat na modelu soustavy, která

může být spuštěna na simulátoru nebo na reálné soustavě.

V této fázi návrhu se již využívá podpory External Mode. Pomocí External Mode můžeme ovládat výpočet algoritmu na fyzickém HW přímo ze Simulinku. Dále můžeme monitorovat a zaznamenávat jednotlivé výstupy subsystému algoritmu a přímo je zobrazovat v modelu v Simulinku. To vše je možné provádět, i pokud je algoritmus spuštěný, vše probíhá v reálném čase. Další nepostradatelnou vlastností při vývoji algoritmu je změna parametrů. Pokud jsou změněny nějaké parametry modelu v Simulinku, jsou nahrány pomocí External Mode do vykonávaného algoritmu na fyzickém HW. Vše je prováděno v reálném čase za běhu algoritmu, bez nutnosti nového kompilování algoritmu.

Cílem této diplomové práce je implementace podpory External Mode pro mikrokontrolér 56F8367. External Mode podporuje komunikaci po sériové lince RS- 232, která bude naportována na použitý mikrokontrolér. Hlavním cílem diplomové práce bude rozšíření External Mode o komunikaci prostřednictvím sběrnice CAN. Tato část bude obsahovat analýzu funkce External Mode. Dále bude implementována podpora pro komunikaci po sběrnici CAN, která bude vycházet z podporované komunikace po sériové lince RS-232. V další části práce bude ověření výkonových vlastností vytvořené komunikace po sběrnici CAN. V závěru práce budou analyzovány možnosti použití JTAG komunikace v External Mode. Práce tím bude ukazovat nutné požadavky, které musí být splněny pro implementaci jiného komunikačního protokolu.

Popis problematiky External Mode včetně jeho použití a popisu architektury je obsahem kapitoly 2. Kapitola 3 popisuje portování sériové podpory External Mode na mikrokontrolér 56F8367. Implementace podpory pro komunikaci po sběrnici CAN je popsáno v kapitole 4. Součástí této kapitoly je vytvoření demonstrační úlohy a ověření výkonových parametrů. Zároveň tato kapitola uvádí postup pro implementaci jiného komunikačního protokolu. Kapitola 5 naznačuje možnosti implementace JTAG komunikace pro External Mode.

Kapitola 2

2 External Mode

External Mode [3] je speciálním komunikačním módem podporovaným v Simulinku, pomocí něhož můžeme spouštět a sledovat výpočet simulací na externím procesoru v reálném čase.

External Mode obsahuje dva separované systémy *host* a *target*, kteří mezi sebou komunikují pomocí komunikačního protokolu. Host je počítač na kterém je spuštěn Matlab se Simulinkem, ve kterém je navržen algoritmus řídícího systému (model). Target je počítač, na kterém je spuštěn vygenerovaný kód z modelu vytvořeném v Simulinku.

External Mode je založen na *client/server* architektuře, kde Simulink je *client* a target je *server*. Host (Simulink) odesílá zprávy se žádostí na target jako je změna parametrů nebo upload dat. Target při vykonávání spuštěného algoritmu zpracovává požadavky a odesílá zpět výsledky do Simulinku.

Hlavní přednosti External Mode:

- Změna a ladění parametrů v reálném čase. Pokud jsou

v External Mode změněny parametry v blokovém diagramu v Simulinku, okamžitě jsou odeslány do spuštěného programu na targetu. To přináší mnoho výhod, není nutné znova kompilovat program a přináší rychlé ladění parametrů.

- Zobrazování a logování výstupů v kterémkoliv bloku systému i podsystému v Simulinku. Tím můžeme monitorovat výstupy, i když jsou tyto hodnoty vypočívány na targetu, bez nutnosti psaní speciálního kódu pro monitorování. Můžou se definovat podmínky, za kterých jsou data odesílána z targetu do Simulinku. Podmínky pro odesílání dat můžou být například, pokud vybraný signál protíná nulu v kladném (záporném) směru nebo manuální spouštění.

Základem External Mode je komunikační kanál mezi Simulinkem a kódem spuštěném na targetu. Tento kanál je vytvořen na nízkoúrovňové komunikační vrstvě, která zajišťuje vysílání a přijímání zpráv. Simulink i kód vygenerovaný z modelu je nezávislý na této komunikační vrstvě, každá část je oddělena v samostatném modulu. Tato koncepce je vhodná pro různé Targety (pojem Target s velým T je definován v kapitole 2.1.1.1) a užití různých komunikačních vrstev.

2.1 Podpora External Mode na MCU

Pro potřeby simulace je nejčastěji používán simulační nástroj Simulink [2], ve kterém je možné vytvořit a simulovat model řídícího algoritmu. Tento model je možné simulovat v Simulinku nebo za pomoci External Mode na fyzickém mikrokontoléru (MCU). Pokud je model spuštěn na fyzickém mikrokontroléru, potom jsou vypočtené výsledky zasílány prostřednictvím External Mode do Simulinku a zde zobrazovány.

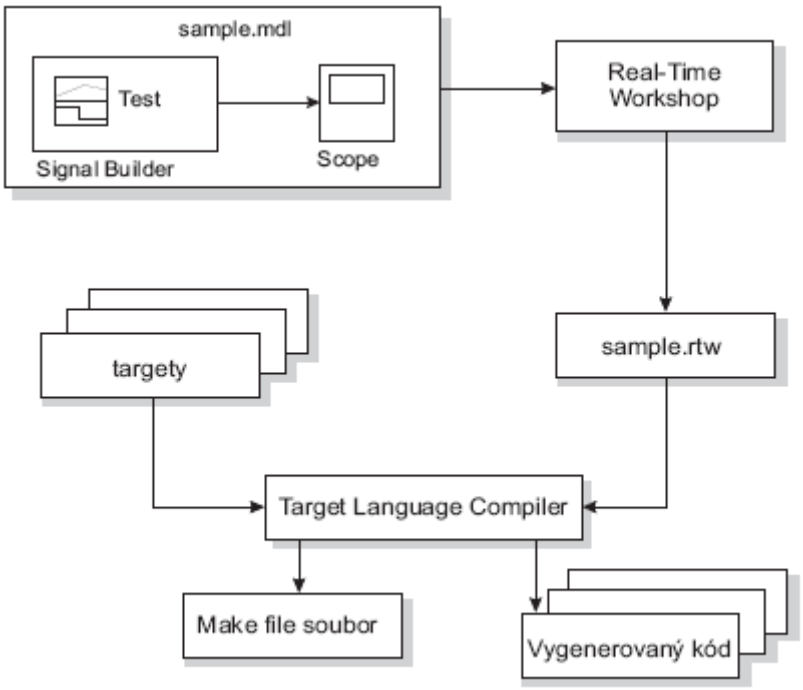
Aby bylo možné model spouštět na MCU, je nutné použít nástroj Real Time Workshop (RTW), který umožní z modelu vygenerovat kód a přeložit ho zvoleným překladačem pro konkrétní MCU. Tím můžeme zajistit, aby výsledný spustitelný soubor, který reprezentující model mohl být spuštěn mimo software Simulinku a to

dokonce i na jiném počítači (hardware) a operačním systému.

2.1.1. RTW

Tato kapitola popisuje nejjednoduší způsob pro vygenerování C kódu z modelu v Simulinku. Postup generování kódu je vidět na obrázku 1. Nástrojem RealTime Workshop [3] je z modelu *sample.mdl* vytvořen soubor *sample.rtw*, který je spolu se zvoleným Targetem pomocí *Target Language Compileru [4]* vygenerován výsledný kód. *Target Language Compiler* umožňuje měnit strukturu generovaného kódu. Pro nás je důležitější vědět, že některé Targety neumožňují podporu External Mode. Význam pojmu target a zjištění podpory pro External Mode je popsána v kapitole 2.1.1.1.

2.1.1.1 Target



Obrázek 1: Generování kódu „Převzato z [4]“

Target je objekt Simulinku, kterým je v simulačním programu reprezentován cílový procesor. Targetem je specifikován CPU jako je použitý programovací jazyk, použitý operační systém nebo třeba nástroje pro práci s vygenerovaným kódem

na cílovém hardwaru.

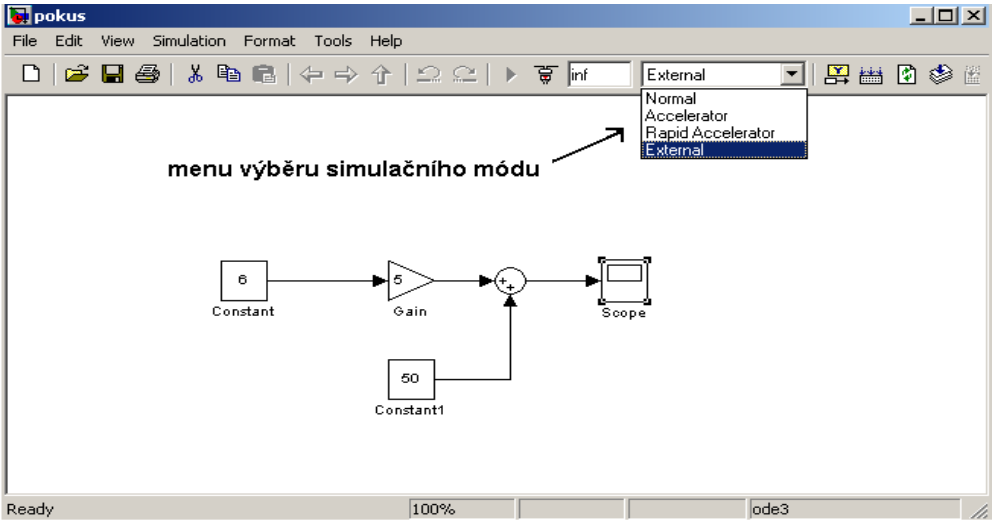
External Mode není podporován ve všech Targetech produkovaných firmou Mathworks. Zde uvádím několik Targetů, které podporují External Mode: ERT, GRT, GRT malloc a Tornado Target nebo Real Time Windows Target. Jestli je External Mode podporován lze zjistit dvěma způsoby. První z nich je zjištění podpory v uživatelských příručkách. Druhý je zjištění podpory z hlavičky systémového TLC souboru. Pokud není External Mode podporován, je u *EXTMODE* uvedeno *no_ext_comm*.

2.2 Použití a nastavení External Mode

Tato část ukazuje jak použít a ovládat External Mode v Simulinku. Také jsou zde ukázány možnosti, které External Mode nabízí (viz [3]).

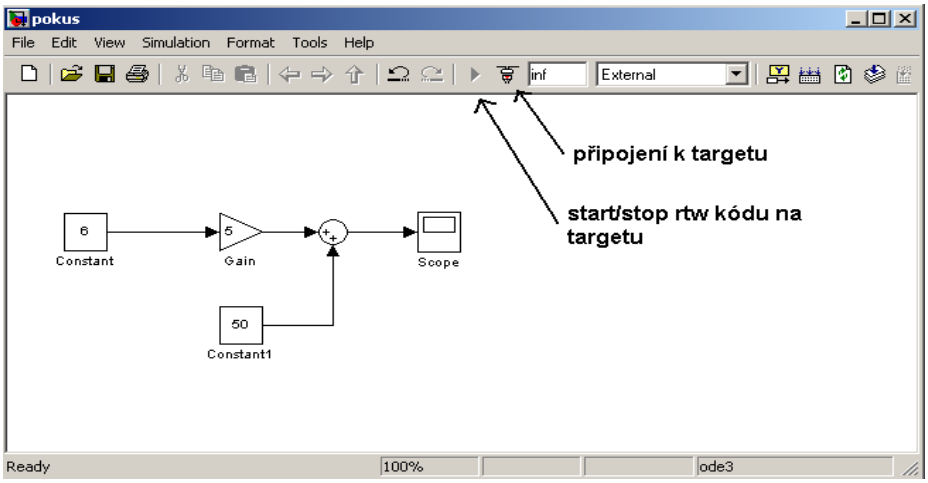
2.2.1. Výběr simulace a ovládání z panelu Simulinku

Pro komunikaci s targetem v External Mode je potřeba vybrat v Simulinku volbu *External*, aby simulace probíhala pomocí External Mode. Existují dva způsoby výběru typu simulace ten nejjednoduší ukazuje obrázek 2.



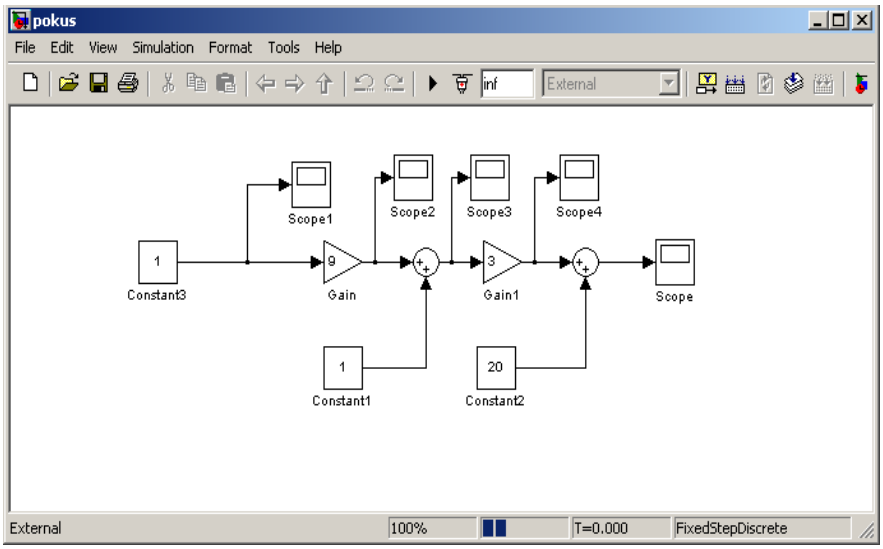
Obrázek 2: Simulace v External Mode

Pro připojení k targetu je nutné, aby byl spuštěn vygenerovaný kód. V této fázi se ještě neprovádí výpočet, ale target pouze čeká na startovací zprávu k připojení od Simulinku. K targetu se můžeme připojit volbou v Simulinku v menu *Simulation -> Connect to Target* nebo pomocí ikony (*Connect to Target*), která je umístěna vlevo od menu výběru simulačního módu (viz obrázek 3). V této fázi je tlačítko *start rtw kódu* neaktivní, protože ještě není navázána komunikace s targetem.



Obrázek 3: Připojení k targetu

Po úspěšném připojení k targetu je spuštěn výpočet kódu modelu. Jestli je host k targetu správně připojen poznáme změnou stavu z *Ready* na *External* v levo dole v okně modelu nebo pokud je aktivní tlačítko *start real-time code* (viz obrázek 4).



Obrázek 4: Spuštění simulace

Spuštění kódu na targetu můžeme provést ze Simulinku volbou *Simulation -> Start Real-time Code* nebo pomocí ikony *start real-time code*. Pokud je kód na targetu spuštěn, zobrazí se místo ikony *Start real-time code* ikona *Stop real-time code*. Dále si můžeme ve spodní liště okna modelu všimnout spuštění času simulace (viz obrázek 4).

Zastavení simulace je možné provést z menu *Simulation -> Stop Real-time Code* nebo stisknutím tlačítka *Stop real-time code*, host odešle na target příkaz k zastavení vykonávání programu a odpojí se od targetu. Po stisku tlačítka *Disconnect from target* odešle host zprávu na ukončení spojení a odpojí se od targetu, i když je na něm vykonáván program (viz obrázek 4).

2.2.2. Řídící panel External Mode

Řídící panel External Mode (viz obrázek 5) zajišťuje centrální ovládání External Mode. V tomto panelu můžeme změnit různé nastavení, které External Mode nabízí. Zde jsou shrnuty hlavní možnosti:

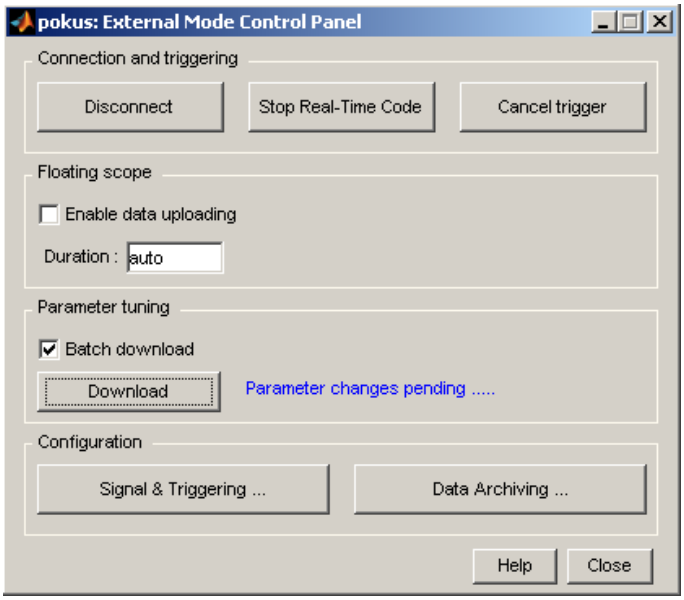
- **Connect/disconnect** - host/target připojení a odpojení.
- **Start/Stop Real-time code** - spuštění a zastavení vykonávaného programu na targetu.
- **Arm Trigger** – manuální aktivování/deaktivování příjmu dat z targetu.

- **Floating scope** – udává jestli a v jakých intervalech se mají přijímat data do *Floating scope* bloků.
- **Parameter tuning** – změna parametrů v modelu a následně na targetu.
- **Configuration** – *Signal & Triggering* nastavuje, které signály budou odesílány a nastavuje možnosti spouštění odesílání dat.
- **Data Archiving** - slouží k ukládání dat z targetu.

Změna parametrů (tunable parametry)

Pokud není zvolena volba ***Batch download***, tak se změna hodnoty parametru v jednotlivém bloku odešle na target, až potom co potvrdíme uložení hodnoty v okně modelu tlačítkem *OK* nebo *Apply button*.

Jestliže je aktivována volba ***Batch download***, tak se souhrnně změny v blocích ukládají v lokální paměti modelu. V tomto případě se všechny změny odesílají na target, až po stisku tlačítka *Download*. Na řídícím panelu (viz obrázek 5) se objeví nápis *Parametr changes pending...,* který je zobrazen tak dlouho, dokud nepřijde zpráva z targetu o úspěšném nahrání parametrů. Tato funkce je výhodná pokud potřebujeme změnit více parametrů ve stejný časový okamžik.

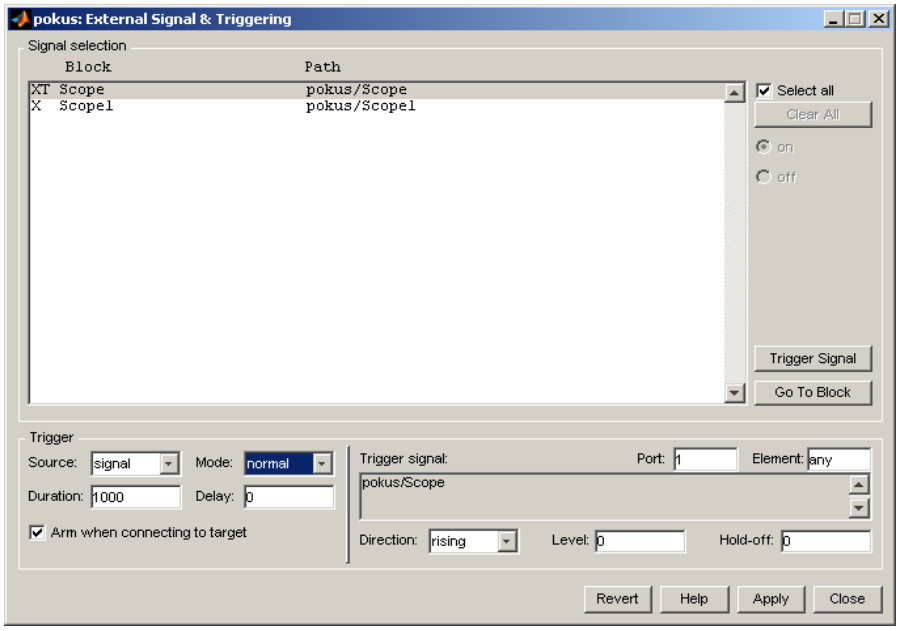


Obrázek 5: Změna parametrů

Nastavení signálu a spouštění

Pokud v řídícím panelu klikneme na tlačítko *Signal & Triggering*, zobrazí se nám dialogové okno (viz obrázek 6), které zobrazuje všechny signály z modelu, které je možno pomocí External Mode sledovat. Je možnost vybrat konkrétní signály, které se budou odesílat z targetu do Simulinku. Existují také bloky, které jsou nekompatibilní pro External Mode a tyto signály tu nebudou zobrazeny. Ve spodní části dialogového okna *Signal & Triggering* se nastavují možnosti spouštění:

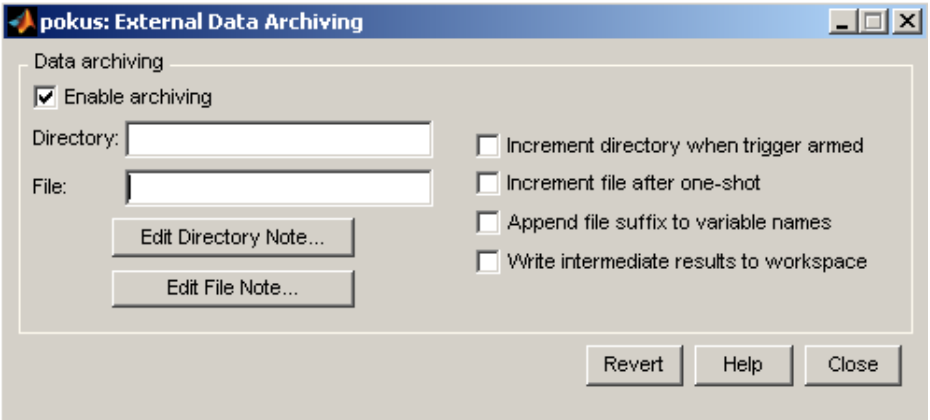
- **Source** - zdroj pro spouštění může být manuální, kdy jsou data odesílána do Simulinku okamžitě, pokud co je aktivní *Arm triger.* Pokud je vybrán jako zdroj *signal*, potom jsou data odesílána až po splnění podmínek (např. signál prochází úrovní triggeru a definovaným směrem a další). Signál triggeru se vybere z okna signálů a potvrdí se tlačítkem *Trigger Signal*.
- **Duration** -udává počet základních kroků, ve kterých External Mode odesílá data, potom co byla splněna podmínka triggeru.
- **Mode** – v *normal* módu je po splnění podmínky trigger pořád aktivní. Na rozdíl od volby *one-shot,* kdy jsou data odeslána pouze jednou.
- **Delay –** reprezentuje časové zpoždění mezi spuštěním triggeru a odesláním dat. Toto zpoždění může být kladné i záporné (pretriggering).



Obrázek 6: Výběr signálů a spouštění

Ukládání dat

Pokud v řídícím panelu klikneme na tlačítko *Data archiving* zobrazí se nám dialogové okno (viz obrázek 7), ve kterém můžeme nastavit možnosti archivování přijatých dat z targetu.



Obrázek 7: External Mode uládání dat

Zaškrtnutím volby *Enable archiving* aktivujeme automatické ukládání dat v External Mode. Volba *Directory* specifikuje adresář kam se mají data ukládat. *File* je název souboru, kam budou data ukládána. External Mode umožňuje u názvu adresáře i souboru přidání koncovky, která je závislá na spuštění triggeru. Pokud nezaškrtneme volbu *Increment directory when trigger armed* nebo *Increment file after one-shot*, data jsou stále ukládána do jednoho souboru, tak že se přepíší hodnoty starých dat novými. V opačném případě se vytvářejí soubory a adresáře, které se liší koncovkou po každém spuštění.

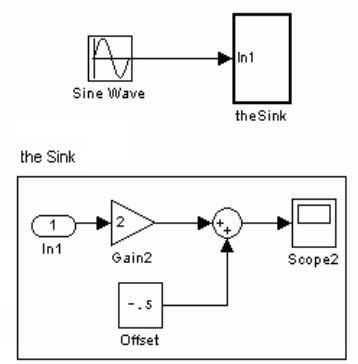
Použitelné bloky pro External Mode

V External Mode můžeme použít následující typy bloků pro příjem a zobrazování signálů přijatých z targetu.

- Floating Scope a Scope bloky
- Spectrum Scope and Vector Scope bloky ze skupiny Signal Processing Blockset™ product
- Bloky ze skupiny Gauges Blockset™ product
- Zobrazovací bloky
- Uživatelské S-Function bloky

- Možnost vytvoření vlastního bloku
- XY Graph bloky

Z bloků je možné vytvářet systémy a subsystémy. V těchto subsystémech můžeme využívat funkce External Mode stejně jako u samostatných bloků. Příklad subsystém je možné vidět na obrázku 8.



Obrázek 8: Možnosti v subsystémech

2.2.3. Vložené a nevložené parametry

V této kapitole je popsán rozdíl, když je kód modelu generován s vloženými parametry a bez vložených parametrů.

Nevložené parametry

Pokud je kód generovaný z RTW, tak že hodnoty proměnných jsou nahrazeny symbolickými jmény parametrů namísto konstant, tak se tyto parametry nazývají nevložené. Tato volba se využívá pro možnost změny parametrů (tunable parametry) v modelu na targetu.

Vložené parametry

Pokud v nastavení Real-Time Workshopu (*Optimization panel -> Configuration Parameters*) zvolíme volbu *Inline parameters,* potom je generován kód s numerickými hodnotami konstant modelu namísto symbolických jmen parametrů. V tomto případě je vygenerovaný kód jednodušší a více efektivnější. Možnost generování kódu s vloženými parametry je pro nás mohem výhodnější, ale na druhou stranu nejdou měnit parametry. RTW software nám dovoluje i při volbě

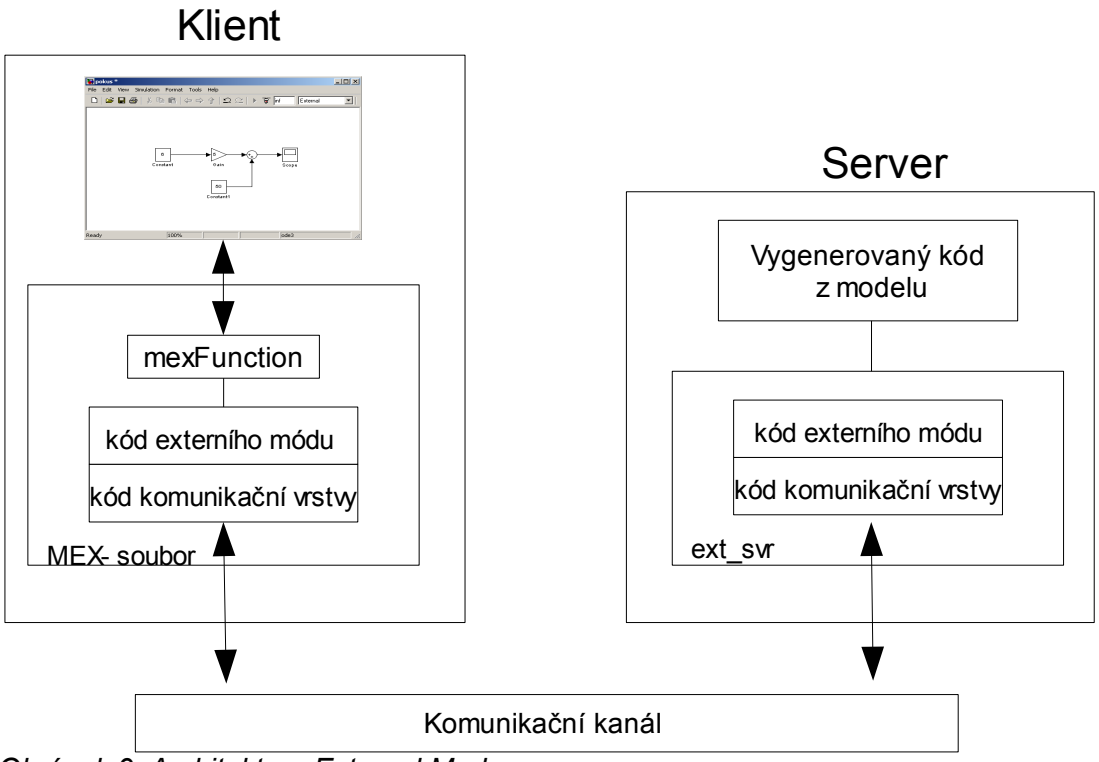
Inline parameters nastavit i některé parametry jako proměnné (*tunable*), které lze následně pomocí External Mode měnit. Toto nastavení se provádí v nastavení *Real-Time Workshopu -> Optimization panel -> Configuration Parameters* volbou *Configure*

2.3 Architektura Klient / Server

External Mode obsahuje zdrojový kód pro implementaci obou stran klienta i serveru. Komunikace v External Mode je podporována užitím TCP/IP nebo sériové RS-232 komunikace. Další možností je implementace vlastního komunikačního protokolu, protože Simulink i kód spuštěný na modelu, jsou nezávislý na komunikační vrstvě. Zdrojový kód External Mode s použitou komunikační vrstvou reprezentuje také samostatný modul. S tímto modulem komunikuje host (Simulink) a target pomocí definovaného rozhraní. V rozhraní je pouze stanoven jednotný formát přenosu dat, který je nezávislý na použitém komunikačním protokolu. Tuto koncepci je možné vidět na obrázku 9.

V další části se budeme zabývat popisem sériové komunikační vrstvy, protože ta bude využita dále při portování na mikrokontrolér Freescale 56F8367 (viz [5]).

Aby mohla být komunikace správně přeložena, je nutné aby server (target) byl vytvořen s definovanou podmínkou EXT_MODE. Podmínka EXT-MODE je definována v souboru název_modelu.mk, tento soubor je vytvořen při generování kódu z modelu. Tuto podmínku je nutné nastavit pro CodeWarrior (viz kapitola 3.2.1.5).



Obrázek 9: Architektura External Mode

Klient a server komunikují pomocí obousměrného kanálu, kterým se přenášejí rámce s daty. Tyto rámce je možné rozdělit do několika skupin Header ,Data nebo speciální rámce. Header rámce definují typ úlohy (spuštění, změna parametrů, loggování dat) a Data rámce většinou dopňují Header rámce o konkrétní data (hodnoty signálů, nastavení triggeru a další).

Host většinou zasílá požadavky. Target obvykle vykonává tyto požadavky. Odpovědí může být potvrzení o vykonání, požadovaná data nebo potvrzení přijetí. Zde je ukázáno pár příkladů zpráv:

- Připojení / potvrzení připojení
- Spuštění simulace / potvrzení spuštění simulace
- Změna parametrů / potvrzení změny parametrů
- Odesílání dat / potvrzení odesílání dat
- Ukončení simulace / potvrzení ukončení simulace

2.3.1. Použití sériového komunikačního protokolu

Aby mohl být použit sériový komunikační protokol RS-232 v External Mode je potřeba splnit následující podmínky:

- Vykonávání programu targetu i hosta na platformě Windows
- Ověření jestli je použit správný *external interface MEX-soubor* pro sériovou komunikační vrstvu. Targety podporované firmou MathWorks specifikují jméno external interface MEX-souboru v matlabroot/ toolbox/ simulink/simulink/extmode_transports.m. Název rozhraní lze v tomto souboru měnit, potom se tento název zobrazuje v nastavení *Interface-> Configuration Parameters*. Pro sériovou vrstvu se název shoduje s názvem MEX-souboru *serial_win32*.
- Zkontrolovat jestli v nastavení makefile je nastaveno propojení na správný soubor pro sériovou podporu serveru, kde jsou definovány důležité kompilační nastavení.
- Následně můžeme vygenerovat kód
- Spuštění programu (kódu) na targetu
- Nastavení v Simulinku volbu simulace External a připojení k targetu

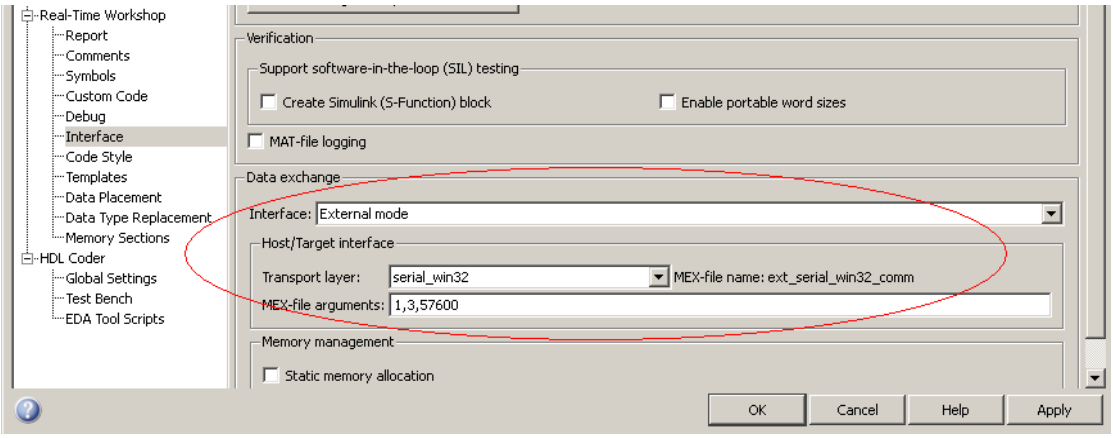
2.3.1.1 Nastavení vstupních parametrů MEX-souboru

Nastavení vstupních parametrů MEX-souboru se nachází v *Tools->Real-Time Workshop->Options->Configuration Parameters* (viz obrázek 10). Zde můžeme nastavit parametry, které jsou předány External Mode interface MEX-souboru (ext_win32_comm). Pro sériovou komunikaci lze nastavit následující parametry:

- **Verbosity level -** nastavení úrovně výpisu informací, které jsou vypisovány na konzoli v Matlabu během přenosu. Lze nastavit hodnotu 0 pro žádné informace nebo hodnotu 1 pro detailní výpis.
- **ID sériového portu –** číslo sériového portu (například 1 pro COM1).
- **Přenosová rychlost –** číselná hodnota přenosové rychlosti, pro sériovou komunikaci je možné nastavit (1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600) kbit/s.

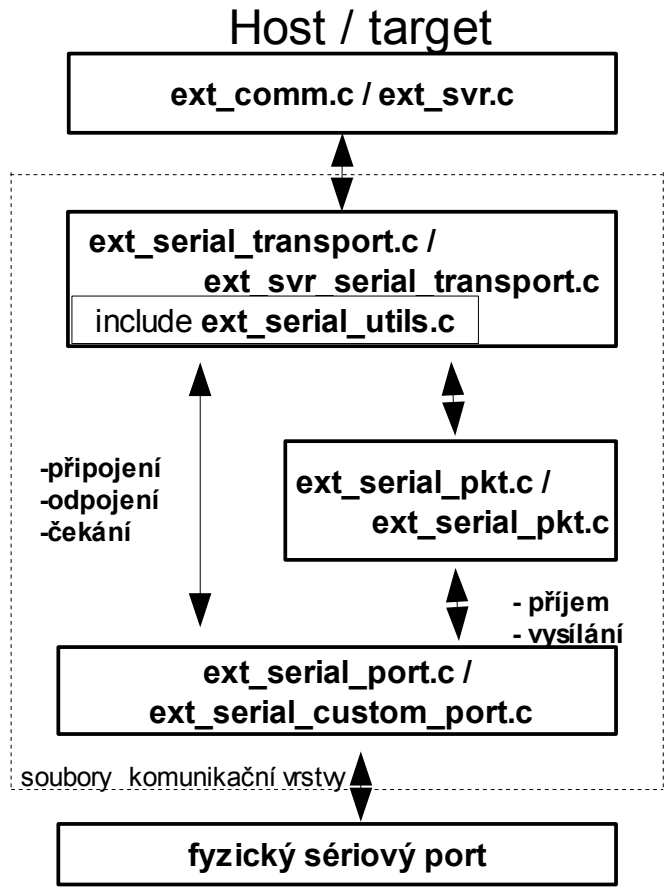
Parametry se nastavují pod položkou *MEX-file arguments* a zapisují se do

řádku podle pořadí jak byly jmenovány (viz obrázek 10).



Obrázek 10: Nastavení parametů pro MEX-soubor

2.3.2. Zdrojové soubory External Mode



Obrázek 11: Kód External Mode

Na obrázku 11 je vidět struktura zdrojových souborů na straně hosta i targetu (zdrojový soubor hosta / zdrojový soubor targetu). Obrázek 11 ukazuje použití

souborů pro podporu sériové komunikace, ale použití je obdobné pro použití TCP/IP nebo vlastní komunikační vrstvy.

Pokud jsou zdrojové soubory targetu zcela odlišné od hostu objevuje se v názvu zkratka *svr* (server).

2.3.2.1 Zdrojové soubory pro host

Zdrojové soubory pro MEX-soubor jsou umístěny v adresáři *matlabroot/rtw/ext_mode*. Následně uvádím popis těch nejdůležitějších:

common/ext_comm.c

Tento soubor je jádrem komunikace External Mode. Ext_comm.c komunikuje se Simulinkem pomocí sdílené datové struktury *ExternalSim*. Pro komunikaci s targetem používá volání funkcí komunikační vrstvy. Hlavně zajišťuje spojení a odpojení s targetem, nahrání parametrů, čekání na příchod zpráv od targetu.

common/ext_main.c

Soubor je vstupním rozhraním do MEX-souboru. Tento soubor je vkládán do souboru ext_comm.c. Simulink volá tento soubor pomocí standartních mex funkcí.

serial/ext_serial_transport.c

Tento soubor implementuje funkce pro komunikační vrstvu. Ext_serial_transport.c využívá pomocných funkcí definovaných v souboru ext_serial_utils.c, který je umístěn v adresáři matlabroot/rtw/c/src/ext_mode/serial.

serial/ext_serial_utils.c

Obsahuje funkce a datové struktury pro komunikaci, propojení s MEX -souborem a generovaným kódem pro moduly host i target s jakoukoliv komunikační vrstvou.

common/ext_convert.c and ext_convert.h

Zde jsou obsaženy funkce pro převod formátu dat ze Simulinku na target a naopak. Dále se provádějí jiné konverze dat. Jednou z mnoha důležitých je převod formátu mezi *little endian* a *big endian*. Tyto funkce jsou volány pomocí ext_comm.c nebo přímo Simulinkem (pomocí pointeru).

common/extsim.h

Tento soubor definuje datovou strukturu ExternalSim a přístupová makra. Tato struktura je využívána pro komunikaci mezi Simulinkem a ext_comm.c.

common/extutil.h

Soubor obsahuje pouze podmínky pro kompilaci přístupových maker

common/ext_transport.h

Zde jsou definovány funkce, které musí být implementovány transportní vrstvou.

Pro implementaci sériového protokolu jsou definovány následující soubory:

serial/ext_serial_pkt.c and ext_serial_pkt.h

Funkce v tomto souboru zajišťují vkládání přijatých dat a výběr dat pro odeslání z objektu sériového paketu

serial/ext_serial_win32_port.c

Je interface mezi programem External Mode a fyzickým sériovým portem.

2.3.2.2 Zdrojové soubory pro target

Tyto soubory jsou částí interface na straně serveru a jsou linkovány k výsledné aplikaci na targetu. Jsou umístěny v adresáři matlabroot/rtw/c/src/ext_mode/.

common/ext_svr.c

Ext_svr.c je analogií k ext_comm.c na straně Simulinku, ale vykonává více úloh. Je vlastně jako rozhraní mezi soubory External Mode na targetu a aplikací (vygenerovaným kódem). Zajišťuje řadu úloh jako je navázání a ukončení spojení, změna přijatých parametrů modelu a zapsání těchto parametrů do modelu na targetu, vyčtení vypočtených dat z bufferu a odeslání těchto dat do Simulinku a mnohé další.

serial/ext_svr_serial_transport.c

Tento soubor implementuje funkce transportní vrstvy. Ext_serial_transport.c používá ext_serial_utils.c, který je umístěn v adresáři matlabroot/rtw/c/src/ext_mode/serial a obsahuje pomocné komunikační funkce.

common/updown.c

Updown.c ovládá vygenerovaný model na targetu. Při přijetí nových parametrů, updown.c nastavuje tyto nové hodnoty do proměnných modelu. Dále se používá pro odesílání dat, obsahuje funkce pro získání dat z vstupů a výstupů modelu a následné zapsání těchto dat do vysílacího zásobníku.

dt_info.h and model.dt

Tyto soubory obsahují informace o převodech datových typů pro přístup do více typů datových struktur prostřednictvím různých počítačových architektur. Tyto informace se používají k datovým konverzím mezi host a target formátem.

common/updown_util.h

Tento soubor obsahuje pouze podmínky pro kompilaci přístupových maker.

Pro implementaci sériového protokolu jsou definovány následující soubory:

serial/ext_serial_pkt.c and ext_serial_pkt.h

Funkce v tomto souboru zajišťují vkládání přijatých dat a výběr dat

pro odeslání z objektu sériového paketu

serial/ext_serial_custom_port.c

Je interface mezi programem External Mode a fyzickým sériovým portem.

2.3.2.3 Další soubory

common/ext_work.c

Tento soubor obsahuje funkce a data specifické pro definovaný Target.

common/ext_share.h

Obsahuje definici zpráv s jejich číselnými kódy a definuje žádosti od modulu hosta i targetu.

ext_svr_transport.h

Zde jsou definovány funkce, které musí být implementovány komunikační vrstvou.

2.4 Komunikační protokol External Mode

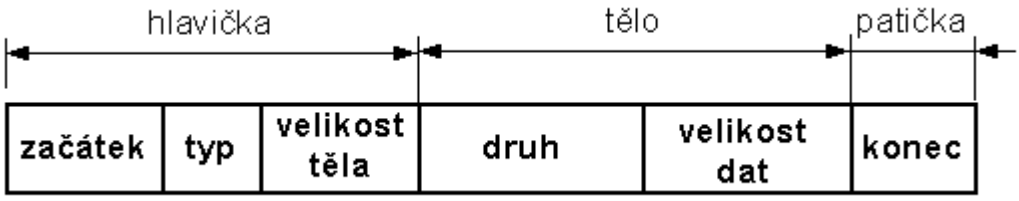
V External Mode host odesílá požadavky na target. Po přijetí zprávy target provádí vykonávání požadavku. Příkladem požadavku je změna parametrů nebo odeslání dat. Komunikace v External Mode je založena na *klient/server* architektuře, kde klient je Simulink a target je server.

2.4.1. Struktura rámců

External Mode definuje přenos dat mezi hlavním souborem External Mode (ext_comm.c/ext_svr.c) a hlavním souborem komunikačního protokolu (ext_transport.c/ext_svr_transport.c) (viz obrázek 11). Přenášená data můžeme rozdělit na dva typy rámců. Prvním je Header rámec (viz obrázek 12) definuje

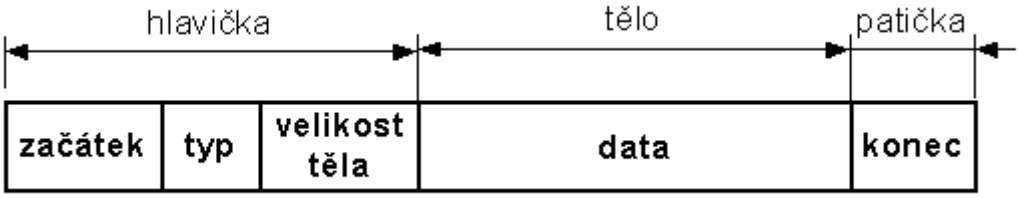
žádost (odpověď), která bude prováděna. Druhým je Data rámec (viz obrázek 13), jehož typ a velikost dat je uvedena Header rámcem. Při použití různé transportní vrstvy se může měnit struktura rámce, vždy je stejné *tělo rámce* a *velikost těla.* Ostatní části rámce se mohou měnit podle použitého komunikačního protokolu.

Header rámec



Obrázek 12: Formát Header rámce

Data rámec



Obrázek 13: Formát Data rámce

Popis jednotlivých polí rámců:

- **začátek** – začátek rámce, použití závisí na zvoleném komunikačním protokolu. Pro sériovou komunikační vrstvu je definována velikost 2B (7E 7E)

- **typ** - definice typu rámce, použití závisí na zvoleném komunikačním protokolu.

- **velikost rámce** – tato hodnota udává velikost datové části těla aktuálního rámce.

- **druh** – toto pole je pouze v Header rámci a udává druh Header rámce, vždy o velikosti 4B, více informací o druhu rámce je popsáno v kapitole

2.4.2.

- **velikost dat** - toto pole je pouze v Header rámci a udává velikost dat (těla) Data rámce, je vždy 4B. Pokud jeho hodnota není nulová, tak je po Header rámci odeslán Data rámec, ve kterém jsou odesílána data (hodnoty parametrů). Pokud je hodnota pole nulová, tak není odeslán Data rámec a jedná se o žádost nějakých dat ze serveru nebo změnu v nastavení.

- **data** – pole je pouze v Data rámci a v poli *data* přenáší informaci, která byla většinou specifikovaná v Header rámci. Například pokud je Header rámcem v poli *druh* specifikována činnost EXT_SETPARAM a pole *velikost dat* není nula, tak je odeslán Data rámec. V Poli Data tohoto rámce jsou hodnoty jednotlivých parametrů z modelu.

2.4.2. Druhy rámců

Druhy rámců jsou definovány v matlabroot\rtw\c\src\ext_mode\common\ext_share.h. Pořadové číslo rámce je uvedené v závorce v hexadecimální soustavě. Toto číslo je udané v Header rámci osmý byte. Význam rámců má různé použití a je možné je rozdělit do několika skupin.

- **rámce k targetu**
  - *rámce pro připojení*
    - EXT_CONNECT (***00***)
    - EXT_DISCONNECT_REQUEST (***01***)
    - EXT_DISCONNECT_REQUEST_NO_FINAL_UPLOAD (***02***)
    - EXT_DISCONNECT_CONFIRMED (03)
  - *rámce pro nahrání a získání parametrů*
    - EXT_SETPARAM (***04***)
    - EXT_GETPARAMS (***05***)
  - *rámce pro upload dat*

    - EXT_SELECT_SIGNALS (***06***)
    - EXT_SELECT_TRIGGER (***07***)
    - EXT_ARM_TRIGGER (***08***)
    - EXT_CANCEL_LOGGING (***09***)

  - *rámce pro řízení modelu*
    - EXT_MODEL_START (***0A***)
    - EXT_MODEL_STOP (***0B***)
    - EXT_MODEL_PAUSE (***0C***)
    - EXT_MODEL_STEP (***0D***)
    - EXT_MODEL_CONTINUE (***0E***)

  - *rámce požadavku dat*
    - EXT_GET_TIME (***0F***)

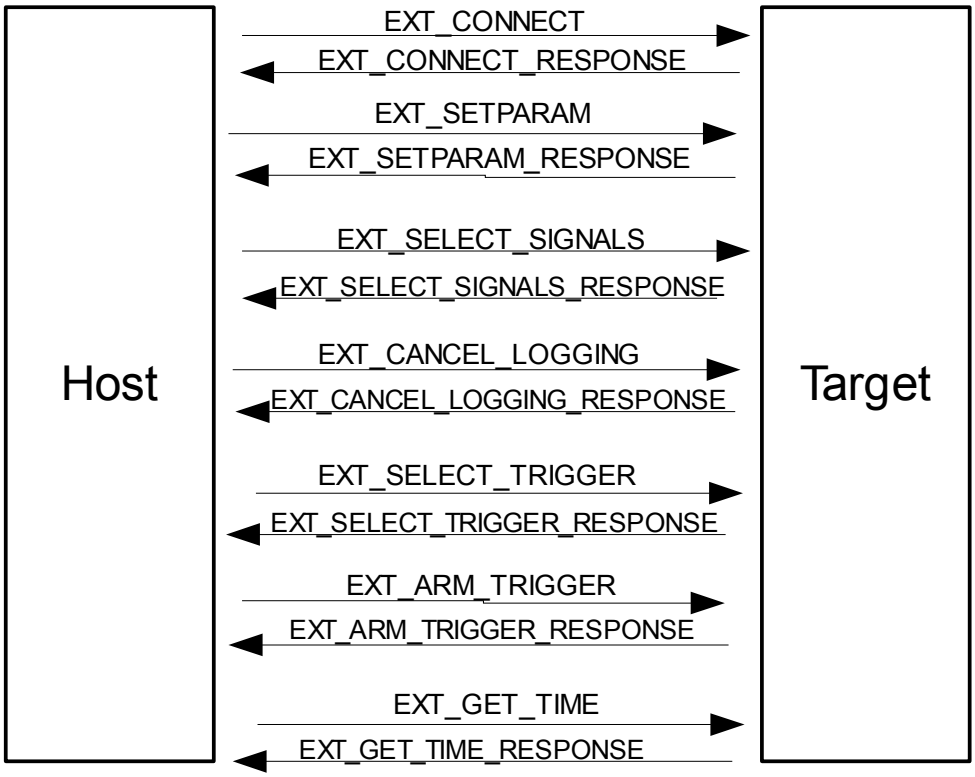
- **rámce z targetu**

  - *odpovědi*
    - EXT_CONNECT_RESPONSE (***10***)
    - EXT_DISCONNECT_REQUEST_RESPONSE (***11***)
    - EXT_SETPARAM_RESPONSE (***12***)
    - EXT_GETPARAMS_RESPONSE(***13***)
    - EXT_MODEL_SHUTDOWN (***14***)
    - EXT_GET_TIME_RESPONSE (***15***)
    - EXT_MODEL_START_RESPONSE (***16***)
    - EXT_MODEL_PAUSE_RESPONSE (***17***)
    - EXT_MODEL_STEP_RESPONSE (***18***)
    - EXT_MODEL_CONTINUE_RESPONSE (***19***)
    - EXT_UPLOAD_LOGGING_DATA (***1A***)
    - EXT_SELECT_SIGNALS_RESPONSE (***1B***)
    - EXT_SELECT_TRIGGER_RESPONSE (***1C***)
    - EXT_ARM_TRIGGER_RESPONSE (***1D***)
    - EXT_CANCEL_LOGGING_RESPONSE(***1E***)
    - EXT_TERMINATE_LOG_EVENT(***1F***)

- EXT_TERMINATE_LOG_SESSION(***20***)

2.4.3. Použití a význam rámců při komunikaci

Nejdříve je spuštěn kód External Mode na targetu, který čeká až host zašle startovací rámec pro vytvoření spojení. Tento startovací rámec je vyvolán volbou *Connect to target (*viz kapitola 2.2*)*, tím dojde k odeslání rámce EXT_CONNECT. Server na to odpoví dvěma rámci. První rámec obsahuje potvrzení EXT_CONNECT_RESPONSE. Druhá zpráva obsahuje c*hecksum* a informace o formátu dat. Po příjmu druhého rámce Simulink vezme *checksum* z targetu a porovná ji s aktuální v modelu ze Simulinku. Simulink si uloží hodnoty o formátu dat, aby věděl v jakém formátu bude data odesílat a přijímat. Dále můžou být odesílány z hostu další zprávy, které závisí na konkrétním nastavení External Mode (např. EXT_SETPARAM,EXT_ARM_TRIGGER, EXT_SELECT_SIGNALS, EXT_SELECT_TRIGGER, EXT_CANCEL_LOGGING a další (viz obrázek 14).

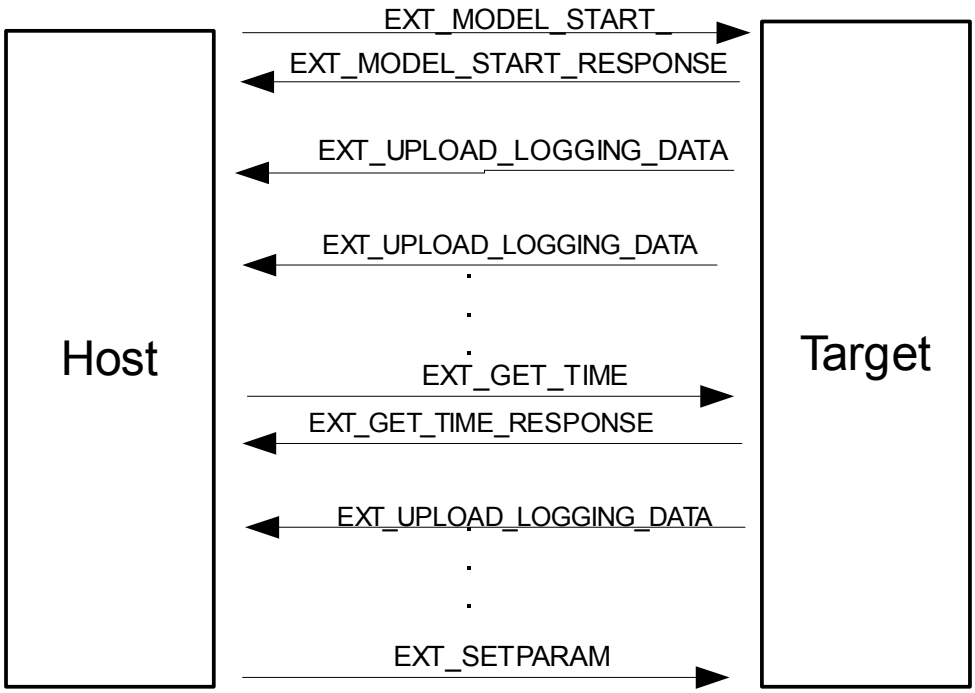


Obrázek 14: Inicializace komunikace

Pokud vše proběhne bez problémů je server a host připojen. V této fázi host

zasílá zprávy se žádostí o čas na targetu (EXT_GET_TIME), target mu zasílá odpovědi na tyto zprávy a čeká na spuštění simulace modelu (EXT_MODEL_START). Po příjmu tohoto rámce target odpoví rámcem (EXT_MODEL_START_RESPONSE). Target vykonává kód modelu a zároveň zpracovává úlohy. Úlohy zpracovávané jsou například přijímání, odesílání, nahrání parametrů, zjištění simulačního času a další.

Důležitou částí je odesílání vypočtených výstupů modelu do Simulinku rámcem (EXT_UPLOAD_LOGGING_DATA). Tento rámec je odesílán periodicky, dle zvolené periody při generování modelu (viz kapitola 3.2.1.4). Pokud bylo při inicializaci nastaveno odesílání dat pomocí triggeru (EXT_ARM_TRIGGER) a jsou splněny podmínky triggeru (EXT_SELECT_TRIGGER) jsou odesílány signály definované pomocí (EXT_SELECT_SIGNALS). Tento případ je možné vidět na obrázku 15. Jestliže nebylo při inicializaci nastaveno odesílání dat (EXT_ARM_TRIGGER), potom data (signály) nejsou odesílány a target čeká na příchod tohoto rámce.



Obrázek 15: Průběh simulace

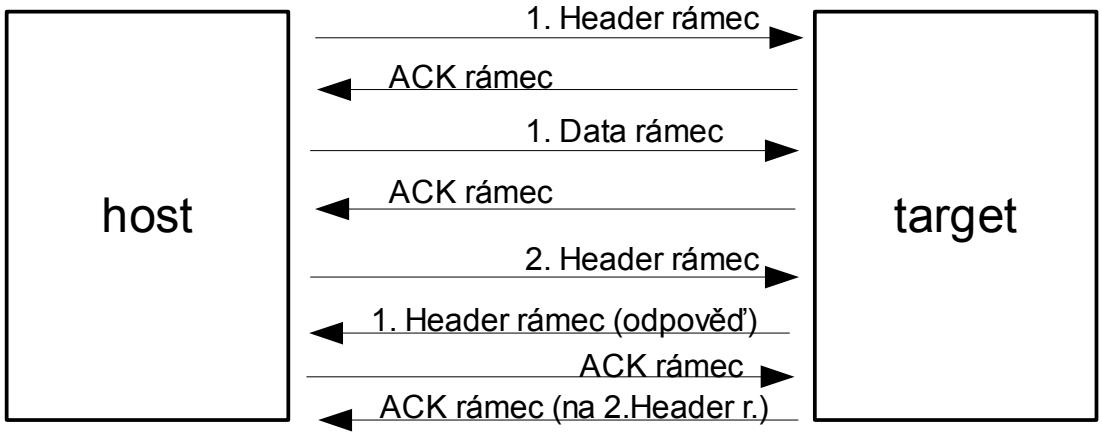
Konec simulace je ukončeno dosáhnutím cílového času simulace, zasláním ukončovacího rámce (EXT_DISCONNECT_REQUEST) nebo zastavení simulace

(EXT_MODEL_STOP). Target informuje host o ukončení vykonávání zprávou (EXT_DISCONNECT_REQUEST_RESPONSE) a ukončí spojení s hostem. Host také ukončí spojení a uzavře External Mode.

2.4.3.1 Popis významu rámců

V této kapitole je popsáno použití a význam jednotlivých rámců při komunikaci mezi Simulinkem a targetem. Sériová komunikace v External Mode mezi hostem a targem je založena na potvrzení každého došlého rámce. Každý Header a Data rámec je potvrzen speciálním (ACK) rámcem, což je možné vidět na obrázku 16.

Na obrázku 16 si je možné všimnout situace, kdy host odesílá 2. Header rámec, tak nedostane hned potvrzení ACK rámcem. Jestliže má target k odeslání nějaký Header nebo Data rámec, nejdříve ho odešle a počká na jeho potvrzení o doručení. Potom až odesílá potvrzení na 2. Header rámec. Tím je zajištěno, že průběžně zasílají Header nebo Data rámce oba host i target. Tím nikdy nedojde k tomu, že jeden zasílá Header nebo Data rámce a ten druhý pouze potvrzuje přijetí ACK rámcem.



Obrázek 16: Příklad sériové komunikace

Speciální ACK rámec

Tento rámec se skládá pouze z hlavičky a patičky s pevnou délkou 9 byte a slouží jako potvrzení přijetí Header nebo Data rámce. V sériové komunikaci v External Mode je definován následně:

7E 7E 02 00 00 00 00 **03 03**

Význam jednotlivých byte:

- **7E 7E** – odpovídá 1-2 byte, je pevný začátek všech rámců, dále už tyto byte nebudeme popisovat u dalších rámců
- **02** – udává typ rámce (ACK), jinak u ostatních rámců je hodnota 01 (normální rámec)
- 00 00 00 00 – tato podtržená čtveřice byte udává velikost datové části, pro ACK rámec je nulová
- **03 03** – stadartně u všech rámců stejná hodnota patičky, dále už tyto byte nebudeme popisovat u dalších rámců

2.4.3.2 Inicializace komunikace

Začátek komunikace začíná inicializací, kdy target čeká na startovací rámec. Průběh inicializace je možné vidět na obrázku 14, dále je popsán význam jednotlivých byte důležitých rámců.

Rámec EXT_CONNECT

Je to první rámec od Simulinku, který inicializuje spojení. Je to Data rámec a má následující tvar:

7E 7E 01 08 00 00 00 65 78 74 2D 6D 6F 64 65 03 03

- 08 00 00 00 – velikost datové části (těla) je 8B
- 65 78 74 2D 6D 6F 64 65 - podtržené datové pole obsahuje hodnoty, které v ASCII odpovídají názvu *ext_mode*.

Rámec EXT_CONNECT_RESPONSE

Po příjmu rámce EXT_CONNECT od Simulinku odešle target následující tři rámce. První rámec má formát:

7E 7E 01 08 00 00 00 **10** 00 00 00 **08** 00 00 00 03 03

Tento rámec udává v prvních 4B datové části druh rámce (EXT_CONNECT_RESPONSE) a v druhých podtržených 4B udává počet bitů v 1B na targetu. Dále ještě tímto rámcem Simulink získá typ *endianu* na targetu.

Druhý rámec má tvar:

7E 7E 01 08 00 00 00 **10** 00 00 00 **54** 00 00 00 03 03

Tento rámec je Header rámec, druh rámce (EXT_CONNECT_RESPONSE) a podtržený 4B udávají velikost následujícího Data rámce o velikosti 84 byte (54 hex.).

Třetí rámec informuje o velikosti datových typů na targetu a má tvar:

7E 7E 01 54 00 00 00 F1 3A 62 1F EB BD 94 37 ED 6B 9C 77 DC 10 07 BE 00 00 00 00 **01** 00 00 00 **0E** 00 00 00 **04** 00 00 00 **01** 00 00 00 **01** 00 00 00 **02** 00 00 00 **02** 00 00 00 **02** 00 00 00 **02** 00 00 00 **01** 00 00 00 **04** 00 00 00 **02** 00 00 00 **02** 00 00 00 **04** 00 00 00 **04** 00 00 00 03 03

Tento Data rámec má délku 84B. V prvních 16B datového pole jsou čtyři 4 byte kontrolní součty. Dále následuje čtveřice byte, která popisuje parametr *intCodeOnly* (zda je na targetu použit pouze typ integer). První čtveřice podtržených byte udává parametr *targetStatus* a další čtveřice byte udává počet datových typů na targetu. V našem případě 14 typů (0E hex.), takže dalších čtrnáct čtveřic byte udává velikost jednotlivých typů na targetu.

Rámec EXT_SET_PARAM

7E 7E 01 08 00 00 00 **04** 00 00 00 **28** 00 00 00 03 03

Jedná se o Header rámec, po kterém následuje Data rámec o velikosti 40B.

7E 7E 01 28 00 00 00 **03** 00 00 00 00 00 **00** 00 01 00 00 00 **00 00 80 3F** 00 00 **01** 00 01 00 00 00 **00 00 A0 40** 00 00 **02** 00 01 00 00 00 **00 00 A0 41** 03 03

První 4B z data pole udávají počet parametrů, v našem případě 3. Každá podtržená čtveřice byte, udává začátek jednotlivých parametrů a jejich pořadové

číslo. Zvýrazněná čtveřice byte udává číselnou hodnotu jednotlivých parametrů.

Rámec EXT_SELECT_SIGNALS

Tento rámec je využíván jen pokud existují tzv. tunable parametry. V případě nastavených tunable parametrů je ve vygenerovaném kódu obsažen soubor model_data.c, kde model je název modelu. Více informací v kapitole 2.2.2. První je odeslán Header rámec, který udává že další odeslaný Data rámec bude mít velikost datové části 44B.

7E 7E 01 08 00 00 00 **06** 00 00 00 **2C** 00 00 00 03 03

Další vyslaný rámec od Simulinku je:

7E 7E 01 **3C** 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 02 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 01 00 00 00 00 00 00 00 **C4 8C** 00 00 03 03

Podtržené 4 byte je hodnota bufferu na targetu, o jehož alokaci Simulink žádá. Simulink žádá o alokaci bufferu velikosti 36036 byte. Protože velikost paměti na použitém procesoru je menší, musela být modifikována funkce *UploadLogInfoInit* v souboru *updown.c*. Úpravou je funkci UploadBufInit předána hodnota 2048B představující velikost bufferu více v souboru updown.c.

2.4.3.3 Průběh komunikace

Úspěšná inicializace mezi Simulinkem a targetem se projeví aktivováním ikony *Start real-time code* (viz kapitola 2.2.1). V této fázi čeká target na rámec EXT_MODEL_START, potvrzení odešle rámcem EXT_MODEL_START_RESPONSE, kterým je spuštěn výpočet modelu na targetu. Od této doby je spuštěn výpočet na targetu a podle zvolené periody výpočtu, jsou odesílány vypočtené hodnoty z targetu rámcem EXT_UPLOAD_LOGGING_DATA (pokud jsou aktivovány podmínky triggeru a je povolen upload). Každý přijatý rámec je potvrzen ACK rámcem. Mezitím Simulink žádá rámcem EXT_GET_TIME aktuální čas simulace, pokud nedostane odpověď od targetu pozastaví se zobrazování simulace, bez chybové hlášky a dále čeká

dokud nepřijme rámec s odpovědí (EXT_GET_TIME_RESPONSE). Následně může Simulink zasílat libovolné žádosti na target (dle žádostí uživatele nebo běhu simulace). Průběh je možné vidět na obrázku 15.

Rámec EXT_UPLOAD_LOGGING_DATA

7E 7E 01 **30** 00 00 00 **1A** 00 00 00 **28** 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 **CD CC 4C 3E** 00 00 00 00 /***00 00 80 3f*** *00 00 10 41* ***00 00 20 41*** *00 00 f0 41* ***00 00 48 42*** 03 03

Tento ráme je typu Data o velikosti těla rámce 48B (30 hex). První modře podtržená čtveřice byte udává typ rámce EXT_UPLOAD_LOGGING_DATA (1A). Následuje popis jednotlivých důležitých byte:

28 00 00 00 – udává velikost byte, které budou následovat. Tato velikost zároveň udává počet přenášených hodnot signálů.

CD CC 4C 3E - v této čtveřici byte je přenášen čas simulace

Dále od červeného lomítka následuje přenos hodnot jednotlivých signálů, které byly požadovány od hosta. Každá čtveřice byte udává hodnotu jedoho signálu Tento případ ukazuje přenos pěti signálů. Rámec se zvětšuje s každým signálem o 4B.

Kapitola 3

3 Portování podpory External Mode pro MCU s komunikací po RS-232

Cílem této kapitoly je naportovat sériovou RS-232 komunikaci External Mode na mikropočítač Freescale 56F8367. Tato kapitola vychází z diplomové práce Processor In the Loop simulace [6], kde byl External Mode použit pro PEERT Target.

3.1 Postup portování transportní vrstvy

Pro External Mode je podporována sériová komunikace RS-232 pro win32 aplikace, proto je nutné modifikovat zdrojové soubory na straně targetu pro 16-bit mikrokontrolér (MCU) následujícím způsobem:

Všechny zdrojové soubory External Mode (viz obrázek 11) budou principielně zachovány. Bude nutné vytvořit nový soubor *ext_serial_port.c,* ve kterém budou

implementovány komunikační funkce pro sériovou linku na MCU. Šablona pro tyto funkce je uložena v matlabroot\rtw\c\src\ext_mode\custom\ext_serial_custom_port.c. Jelikož kód External Mode je 32-bitový a vygenerovaný kód na targetu 16-bitový bylo nutné modifikovat některé soubory a funkce. Tyto modifikace jsou popsány v kapitole 3.2.3.

3.2 Mikrokontrolér Freescalle 56F8367

MC56F8367EVM je konstruovaná jako všestranná vývojová deska s procesorem 56F8367, umožňující vytvořit programové a hardwarové produkty v reálném čase.. Výkon 16- bitových procesorů 56F8367 v kombinaci s 128K x 16-bit externí programovou/datovou statickou pamětí RAM (SRAM), 128K x 16- bit externí data/program SRAM, sériový RS-232 portem, CAN porty, periferními konektory umožňuje tvorbu rozsáhlých aplikací v reálném čase.Vývojová deska obsahuje JTAG konektor, který se využívá pro nahravání a laďení programu. Více informací o této vývojové desce můžete nalézt v [5] a [7].

3.2.1. Práce s mikrokontrolérem

Pro práci s mikrokontrolérem 56F8367 je využito vývojového prostředí Metrowerks CodeWarrior s nástrojem Processor Expert. Tyto vývojové nástroje jsou podrobně popsány v [8].

3.2.1.1 Processor Expert (PE)

Processor Expert je nástroj podporující mikrokontroléry Freescale určené pro rychlý vývoj aplikací s využitím programovacího jazyka C. PE má k dispozici informace o procesorech a jejich periferiích obsažených v tzv. *Embedded Beans (Bean)*, proto je možné provádět nastavení i pokud není ještě fyzicky k dispozici samotný procesor. Podle těchto beanů PE generuje kód s využitím nastavení hardwarových vlastností procesoru.

Nastavení jednotlivých beanů je velmi jednoduché díky grafickému uživatelskému rozhraní. *Bean* je tvořen vlastnostmi (*properties*), metodami (*methods*) a událostmi (*events*), což je vidět na obrázku 17.

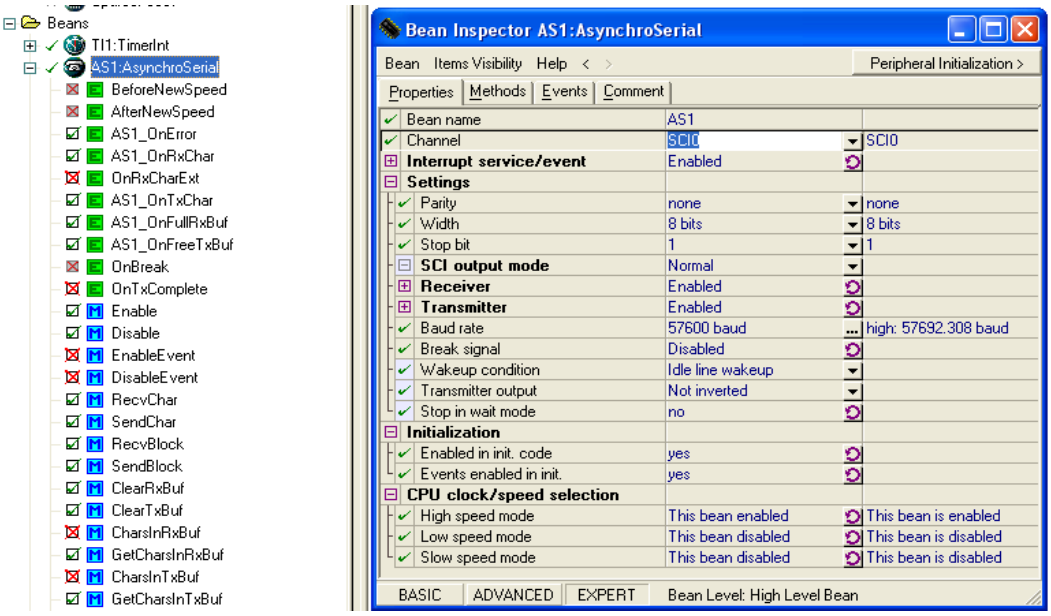
Pomocí vlastností se nastaví parametry hadware bez konkrétní znalosti detailů procesoru a jeho registrů. Všechna nastavení se nechají kdykoliv měnit v případě změny zapojení výstupních periferií. Metody umožňují uživatelský přístup k periferiím. Tyto metody se nechají vkládat přímo do uživatelského kódu. Díky událostem od jednotlivých periferií, můžeme okamžitě reagovat na proběhlou změnu stavu.

3.2.1.2 Vytvoření projektu

Ve vývojovém prostředí Metrowerks CodeWarrior vybereme nový projekt (*File- >New->DSP56800 New Project Wizard*). Po zadání jména projektu vybereme skupinu a typ mikroprocesoru a jazyk v jakém bude vytvořena základní funkce main(). Ve vytvořeném projektu lze nastavovat vlastnosti procesoru, přidávat jednotlivé beany (informace o vlastnostech periferie) a přidávat vlastní zdrojové soubory. Více informací lze najít v [5].

3.2.1.3 Přidané beany

Pro komunikaci v External Mode po sériové lince je nutné přidat *bean AS1:Asynchro serial*. Na obrázku 17 je vidět přidání beanu mezi ostatní. Nastavení Beanu, použité funkce a události, které lze použít pro obsluhu sériového kanálu je vidět na obrázku 17. Pro External Mode využívající komunikaci prostřednictvím sériové linky je maximálně možné nastavit rychlost 57600 baud.



Obrázek 17: Nastavení Beanu AS1

Název beanu AS1 by měl zůstat zachován, protože použité metody tohoto beanu jsou použity v kódu External Mode. Pro práci byly použity následující metody:

- **AS1_RecvChar** – přečte jeden byte z bufferu sériové linky.
- **AS1_SendBlock** – metoda odešle nastavený počet byte na sériovou linku.
- **AS1_GetCharsInRxBuf** – metoda zjistí počet byte v přijímacím bufferu

Dalším přidaným beanem je časovač TI1:TimerInt. Tímto časovačem bude nastavena doba, po které se spustí výpočet algoritmu modelu. Nastavení se provádí v záložce *Properties*, kde je důležité nastavit dobu a prioritu přerušení. Nastavení hodnoty přerušení je nutné nastavit na hodnotu *minimal priority*, aby byla zajištěna bezchybná komunikace mezi Simulinkem a mikrokontrolérem.

3.2.1.4 Přidání zdojových souborů a nastavení

Do vytvořeného projektu je potřeba přidat zdrojové soubory External Mode a vygenerovaného modelu. Popis zdrojových souborů External Mode je popsán v kapitole 2.3.2 .

Zdrojové soubory External Mode

- ext_svr.c
- updown.c
- ext_work.c
- ext_serial_pkt.c
- ext_svr_serial_transp.c
- ext_serial_PE_port.c

Zdrojové soubory vygenerované z modelu

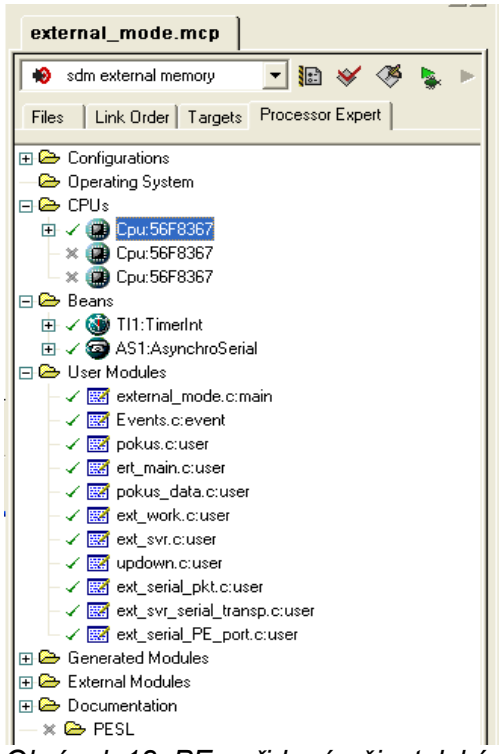
Popis generování souborů z modelu v Simulinku je popsáno v kapitole 2.1.1 nebo v literatuře [1]. Název souboru je vždy vytvořen z názvu modelu. Přidané soubory do projektu, které byly vygenerovány z modelu s názvem *Pokus* jsou vidět na obrázku 18. Dále jsou popsány hlavní soubory:

- **název_modelu.c** – obsahuje funkce pro inicializaci, spuštění, krok a ukončení výpočtu modelu. Dále je zde uvedená perioda spouštění.
- **název_modelu_data.c** – v tomto souboru jsou uloženy hodnoty v proměných, které odpovídají nastavením jednotlivých bloků v modelu v Simulinku.
- **ert_main.c** (název targetu_main.c) – tento soubor je šablona, která ukazuje jak volat *step()* funkci (v název_modelu.c). Je zde popsáno volání funkce *step()* pomocí přerušení nebo potřebná nastavení pro uložení vstupů a výstupů. Nastavení přerušení je provedeno časovačem TI1:TimerInt (viz kapitola 3.2.1.3).

Dále je potřeba nastavit cesty pro různé jiné soubory, které jsou potřeba pro funkci External Mode. Tyto cesty se nastaví ve vlastnostech procesoru v záložce *Build options -> User paths.* Pro sériovou komunikaci jsou nastaveny následně:

- ...\název-modelu_ert_rtw – adresář, který obsahuje kód vygenerovaný z modelu s podporou targetu ERT
- matlabroot\simulink\include
- matlabroot\rtw\c\libsrc
- matlabroot\rtw\c\src\ext_mode\common
- matlabroot\extern\include

- matlabroot\rtw\c\src

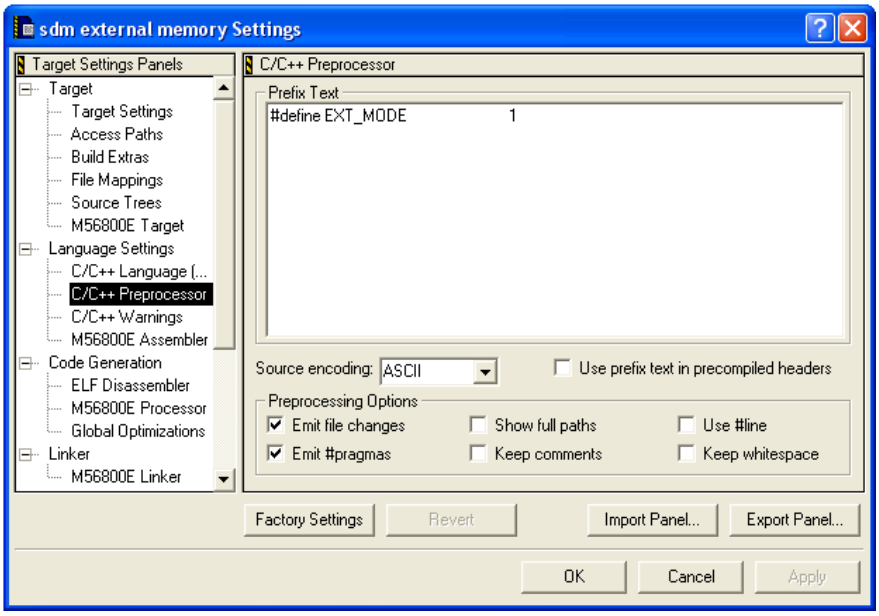


Obrázek 18: PE - přidané uživatelské soubory

V záložce *Build options* je potřeba nastavit velikost zásobníku (*Stack size*) na hodnotu 2000B a haldy (*Heap size*) na 8000B. Tím se předejde problémům s alokací paměti na CPU.

3.2.1.5 Nastavení preprocessoru

Aby mohly být soubory External Mode správně přeloženy, je nutné nastavit v CodeWarrioru příznak EXT_MODE. V nastavení v CodeWarrioru je nutné nastavit flag EXT_MODE na hodnotu 1, to lze zajistit příkazem *#define EXT_MODE 1*. Nastavení preprocesoru v CodeWarrioru je možné vidět na obrázku 19.



Obrázek 19: Nastavení preprocesoru v CodeWarrioru

3.2.2. Komunikační funkce na CPU

Tyto komunikační funkce využívají *bean Asynchroserial* a jeho funkce, které obsluhují fyzický sériový port. Jsou implementovány v souboru ext_serial_PE_port.c, který je vytvořen ze šablony umístěné v souboru matlabroot\rtw\c\src\ext_mode\custom\ext_serial_custom_port.c V tomto souboru je nutné implementovat těla následujících funkcí:

- **ExtSerialPortCreate** – vytvoří objekt typu *ExtSerialPort*. Tento objek slouží pro přístup k fyzickému sériovému portu.
- **ExtSerialPortConnect** – tato funkce zajišťuje připojení objektu *ExtSerialPort* k fyzickému sériovému portu.
- **ExtSerialPortDisconnect** – funkce odpojí objekt *ExtSerialPort od* fyzického sériového portu.
- **ExtSerialPortSetData** – funkce odešle jeden byte na sériovou linku.
- **ExtSerialGetRawChar** – přijme jeden byte ze sériové linky.
- **ExtSerialPortDataPending** – tato funkce vrací počet přijatých byte čekajících na přečtení ze sériové linky.

3.2.3. Modifikace zdrojových souborů External Mode

Modifikace souborů transportní vrstvy

Soubor ext_svr_serial_transport.c je hlavní soubor použité komunikační vrstvy v External Mode. Do tohoto souboru je vložen soubor ext_serial_utils.c, protože byl problém s překladem vygenerovaného kódu v CodeWarrioru. Touto úpravou vznikl nový soubor *ext_svr_serial_transp.c.*

Čekání na první rámec od Simulinku

Vygenerovanému kódu External Mode je možné předat vstupní parametry z příkazové řádky při spuštění aplikace. Tím se předává informace, zda se má aplikace spustit ihned nebo nebo až po příjmu prvního rámce. Při nahrání programu v CodeWarrioru, nelze při spuštění tyto parametry aplikaci předávat.

Proto musela být modifikována funkce *ExtWaitForStartPktFromHost* v souboru ext_svr_serial_transp.c, vložením kódu:

UD -> WaitForStartPkt = 1;

Potom již spuštěná aplikace pro simulaci v External Mode čeká na příjem prvního rámce od Simulinku.

Zjištění velikosti datových typů

Při inicializaci komunikace mezi targetem a hostem je zjišťována velikost datových typů na targetu. Jelikož kód transportní vrstvy External Mode pro sériovou linku je 32 – bitový, tak je nutné upravit části kódu pro 16 – bitový mikroprocesor.

Musel být změněn kód ve funkci *ProcessConnectPkt* v souboru ext_svr.c. Byl zde zakomentován původní příkaz a použita funkce memcpy pro překopírování obsahu paměti, kde jsou uloženy velikosti datových typů.

```
//(void)memcpy(&tmpBuf[7], dtSizes, sizeof(uint32 T)*nDataTypes);
(void)memcpy(&tmpBuf[7], dtSizes, sizeof(uint T));
(void)memcpy(&tmpBuf[8], dtSizes+1, sizeof(uint T));
```

```
(void)memcpy(&tmpBuf[9], dtSizes+2, sizeof(uint T));
…
```

Touto úpravou se již správně překopírují 16 bitové hodnoty *dtSizes* do 32bitového bufferu *tmpBuf*.

3.2.4. Demonstrační úloha

Byla vytvořena demonstrační úloha pro External Mode s přenosem pomocí sériové komunikace. Popis potřebných změn při portování na mikrokontrolér 56F8367 byly popsány v této kapitole 3. Podrobný popis kroků spuštění demonstrační úlohy bude popsán v kapitole 4.4.5, která se zabývá implementací podpory prostřednictvím CAN sběrnice. Tabulka 1 zobrazuje přehled zdrojových souborů, které musely být modifikovány. Zdrojové soubory External Mode pro komunikační vrstvu včetně modelu a PE projektu jsou uloženy na přiloženém CD v adresáři external_mode\serial.

ext_svr.c	Tyto soubory je nutné přehrát místo souborů, které jsou uloženy v matlabroot\rtw\c\src\ext_mode\common
updown.c	
ext_serial_PE_port.c	Tyto soubory je nutné přehrát místo souborů, které jsou uloženy v matlabroot\rtw\c\src\ext_mode\serial
ext_svr_serial_transp.c	

Tabulka 1: Zdojové soubory pro sériovou RS-232 komunikaci

Kapitola 4

4 Implementace podpory External Mode pro MCU s komunikací CAN

Cílem této kapitoly je vytvořit podporu komunikace External Mode prostřednictvím sběrnice CAN.

4.1 Nutné předpoklady implementace

Pro vytvoření podpory komunikace CAN v External Mode bude nutné změnit zdrojové soubory komunikační vrstvy obou stran host i target. Budou vytvořeny nové soubory pro komunikační vrstvu CAN z předvytvořených šablon. Tyto soubory budou pojmenovány názvem komunikace CAN (např. ext_custom_port.c bude změněn na ext_can_port.c) Některé soubory budou pouze modifikovány v porovnání se soubory sériové RS-232 komunikační vrstvy.

Nejvíce změn bude v souborech *ext_can_port.c* a *ext_can_pkt.c*

(viz obrázek 11). V těchto souborech jsou funkce, které sestavují (rozkládají) data z rámců a tyto rámce následně odesílají (přijímají) z fyzického portu. Dále jsou zde prováděny další úlohy jako je připojení, odpojení, čekání a mnohé další.

Celá implementace podpory komunikace CAN bude rozdělena na dvě podčásti. V první z nich bude popsána modifikace *targetu.* Jelikož již v kapitole 3 bylo popsáno portování podpory sériové komunikace na mikrokontrolér 56F8367, bude se tato část týkat pouze změn oproti sériové podpoře provedené na MCU. V druhé podčásti bude popsána implementace External Mode na straně Simulinku, která se bude skládat z následujících podčástí:

- Úprava zdrojových souborů pro komunikační protokol CAN
- Převodník pro USB-CAN komunikaci
- Tvorba MEX – souboru pro komunikační vrstvu CAN
- Registrace podpory komunikačního protokolu v Simulinku
- Demonstrační model a testování výkonových parametrů

4.2 Sběrnice CAN

Sběrnice CAN [9] byla navržena tak, aby umožnila provádět distribuované řízení systémů v reálném čase s přenosovou rychlostí až 1Mbit/s a s vysokým stupňem zabezpečení přenosu proti chybám. Jedná se o protokol typu multi-master, kde každý uzel sběrnice může být master. Pro řízení přístupu k médiu je použita metoda s náhodným přístupem a příposlechem komunikace, která řeší kolize na základě prioritního rozhodování.

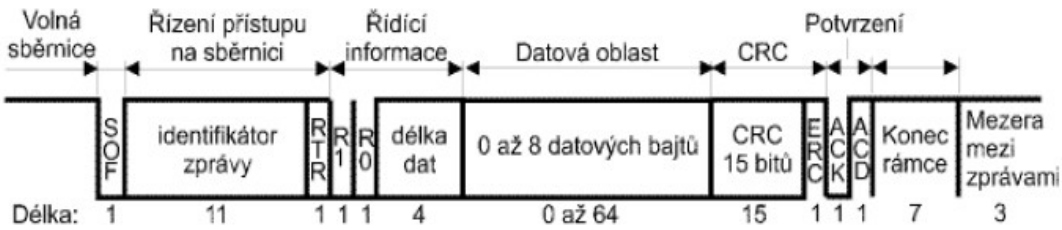
Zprávy vysílané po sběrnici CAN neobsahují žádnou informaci o cílovém uzlu a jsou přijímány všemi uzly připojenými ke sběrnici. Každá zpráva je uvozena identifikátorem, který udává význam přenášené zprávy a její prioritu. Nejvyšší prioritu má zpráva s identifikátorem 0. Tím je zajištěno na sběrnici CAN, že při kolizi dvou zpráv dostane přednost zpráva s vyšší prioritou. Z identifikátoru je možné zjistit ID zprávy a podle toho rozhodnout, jestli má zprávu daný uzel přijímat (*Acceptance Filtering*).

Fyzická vrstva

Fyzická vrstva definuje jak jsou po sběrnici přenášeny signály. Základním požadavkem na fyzickém přenosovém médiu protokolu CAN je, aby realizovalo funkci logického součinu. Standart protokolu CAN definuje dvě vzájemně komplementární hodnoty bitů na sběrnici *dominant* a *recessive*. Vysílají-li všechny uzly sběrnice recessive bit, pak je na sběrnici úroveň *recessive*. Pokud alespoň jeden uzel vysílá dominant je na sběrnici úroveň *dominant*. Pro eliminaci odrazů na vedení je nutné, aby byla sběrnice na obou koncích přizpůsobena zakončovacími odpory o velikosti 120Ω. Více informací lze najít v [9].

Datové rámce

Specifikace protokolu CAN definuje čtyři typy zpráv. Pro External Mode, který je založen na klient/server architektuře budeme potřebovat pouze typ zprávy pro odeslání dat. Tento typ je pro CAN definován jako datový rámec (Data frame). Struktura datového rámce je na obrázku 20 Význam jednotlivých částí důležitých pro použití přenosu v External Mode:

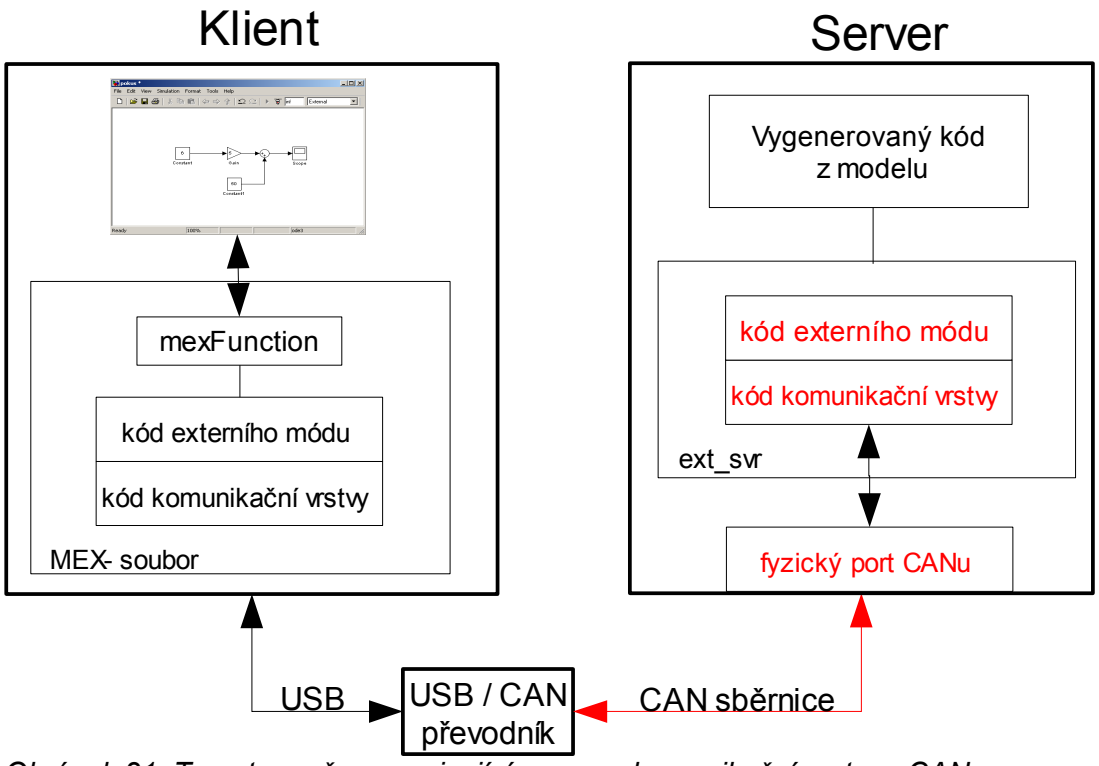


Obrázek 20: Struktura CAN rámce „Převzato z [9]“

- **Identifikátor zprávy** (ID) – podle něho se identifikuje zpráva na sběrnici
- **Délka dat** – udává počet přenášených datových byte v rámci, povolené jsou 0-8 byte.
- **Datová oblast** – udává datové byte zprávy, maximálně lze odeslat 8 byte.

4.3 Target

Nutné modifikace pro komunikační vrstvu CAN jsou červeně zvýrazněny na obrázku 21.



Obrázek 21: Target - změny související s novou komunikační vrstvou CAN

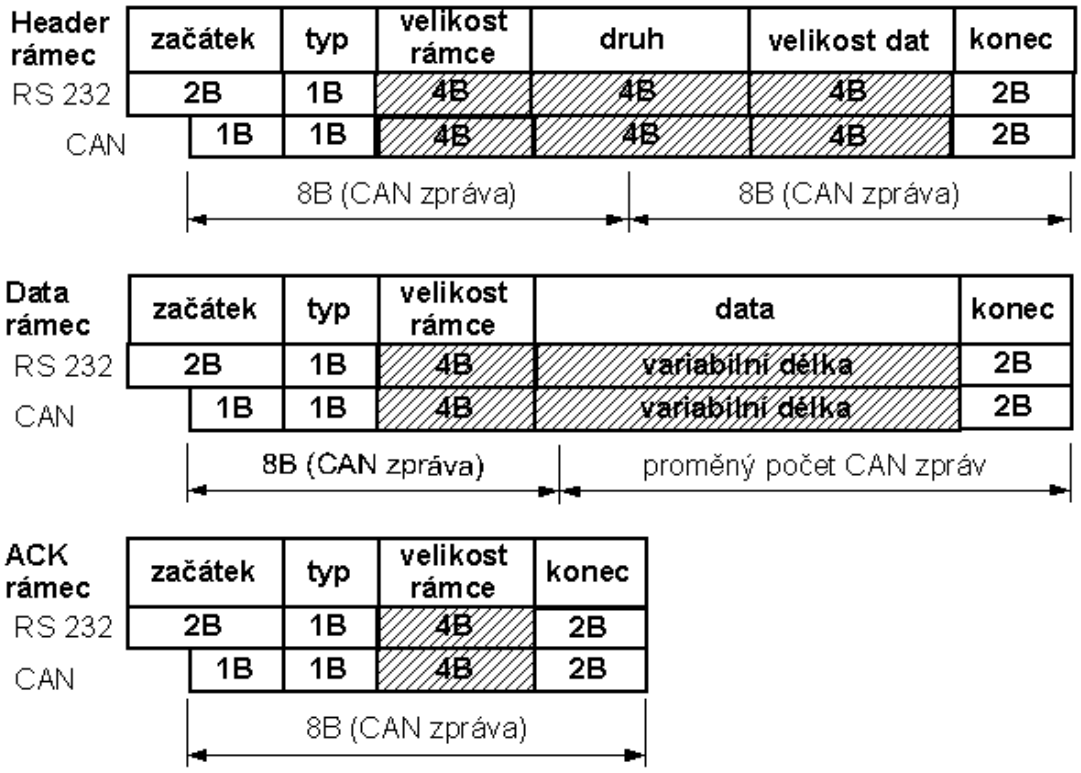
4.3.1.1 Změna rámců External Mode po sběrnici CAN

Jedna CAN zpráva může přenášet maximálně 8 datových byte. Pokud by byly použity stejné rámce External Mode jako pro sériovou komunikaci, docházelo by k nadbytečnému přenosu CAN zpráv. Struktura rámců External Mode je popsána v kapitole 2.4.2 a použitá struktura rámců pro sériovou komunikaci (viz kapitola 2.4.3.1). Rámce External Mode pro CAN komunikační vrstvu byly upraveny (viz obrázek 22) následujícím způsobem:

- **Header rámec** - tento rámec má určenou velikost přenášených dat

(vyšrafovaná oblast). Ostatní části rámce mohou být modifikovány (viz kapitola 2.4.2). Rámec byl upraven, aby mohl být odeslán pomocí dvou CAN zpráv.

- **Data rámec** – tento rámec má proměnou délku přenášených byte. Tento rámec byl změněn v závislosti na rámci EXT_UPLOAD_LOGGING_DATA, kterým jsou přenášeny hodnoty vypočítávaných signálů na targetu k Simulinku. Rámec pro přenos jednoho signálu má délku 41B, proto byl upraven začátek rámce z 2B (7E 7E) na 1B (7E) (viz obrázek 22).
- **Speciální (ACK) rámec** – tento potvrzovací rámec má délku 9B, proto byl stejně změněn začátek rámce jako u Header a Data rámce, aby tento rámec mohl být odeslán v jedné CAN zprávě.



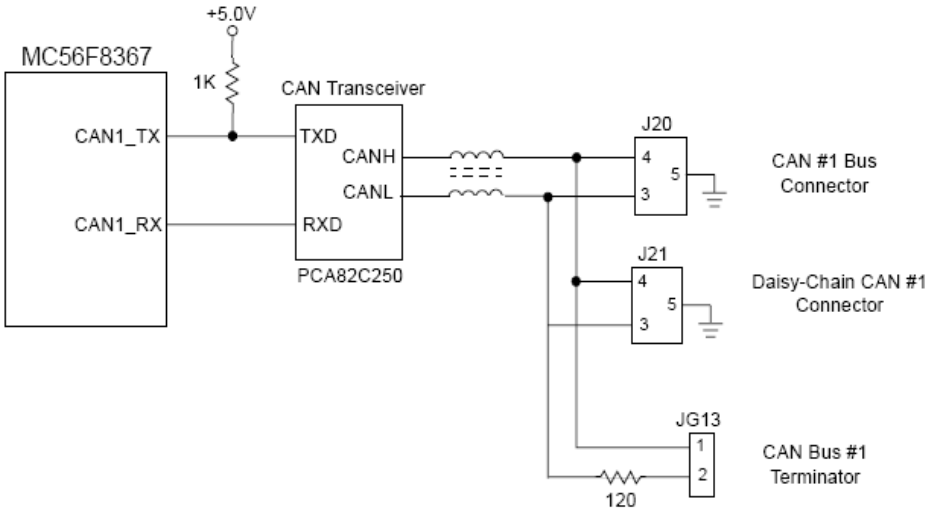
Obrázek 22: Upravené rámce External Mode pro CAN

4.3.2. CAN na MCU

Práce ve vývojovém prostředí pro mikrokontrolér 56F8367 byla popsána v kapitole 3.2.1. V této kapitole se budeme zabývat nastavením beanu a změnami pro komunikaci CAN oproti předchozí sériové RS-232 komunikaci na použitém mikrokontroléru.

4.3.2.1 FlexCAN interface

Mikroprocesor 56F8367 obsahuje dva FlexCAN interface. Při práci je využit první interface FlexCAN #1, jeho schéma včetně konektorů je možno vidět na obrázku 23. Mikroprocesor využívá CAN převodník PCA82C250 definovanou fyzickou vrstvou podle Phillips high-speed s maximální rychlostí 1.0Mbit/s (viz [7]).



Obrázek 23: FlexCAN interface „Převzato z [7]“

Pro připojení byl použit konektor J20, jeho zapojení je v tabulce 2. Dále si můžeme z obrázku všimnout, že propojkou JG13 můžeme připojit zakončovací odpor sběrnice o velikosti 120 Ω.

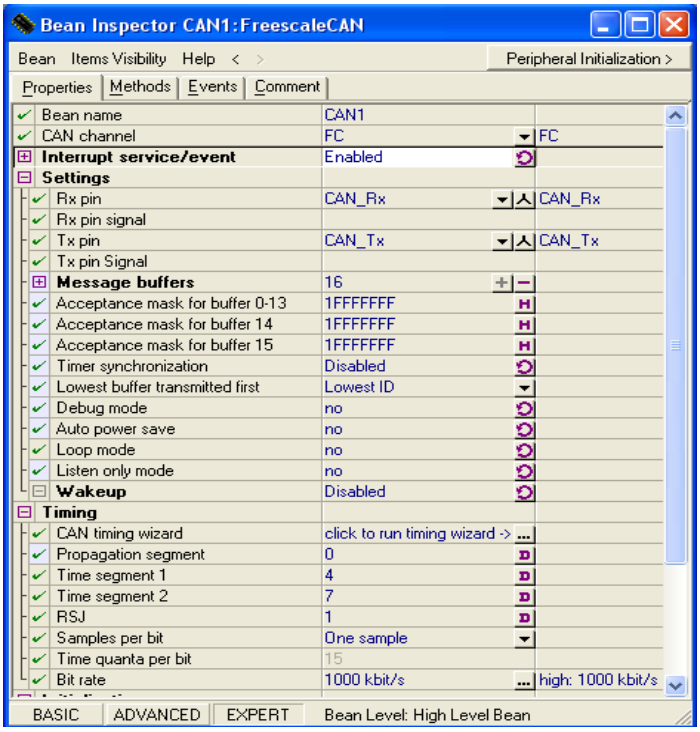
Pin	Signál	Pin	Signál
1	nezapojen	2	nezapojen
3	CANL	4	CANH
5	GND	6	nezapojen
7	nezapojen	8	nezapojen
9	nezapojen	10	nezapojen

Tabulka 2: Zapojení konektoru FlexCAN 1

4.3.2.2 FlexCAN bean

Pro komunikaci v External Mode postřednictvím CAN sběrnice je nutné přidat do projektu bean CAN: *CAN1:FreescaleCAN*. Nastavení vlastností je široké, uvedu pouze ty nejdůležitější, které byly při práci využity.

CAN1 obsahuje 16 bufferů pro zprávy, které se mohou nastavit jako vysílací nebo přijímací. Dále se u jednotlivých bufferů nastavuje typ zpráv, které se mají přijímat (standard, extendet). Identifikační číslo (ID) zprávy se nastavuje pouze u přijímacího bufferu. ID odesílané zprávy lze měnit ve funkci pro odeslání. Přerušení pro vysílání a příjem bylo nastaveno se střední prioritou na hodnotu *medium priority*. Rychlost přenosu je možné nastavit pod položkou *Bit rate*, byla nastavena na maximální možnou pro sběrnici CAN na hodnotu 1.0Mbit/s. Nastavení beanu CAN1 je vidět na obrázku 24.



Obrázek 24: Nastavení beanu CAN1

Jméno beanu musí být zachováno, protože je součástí názvu metod, které jsou použity ve zdrojovém kódu External Mode.

Použité metody (funkce) beanu CAN1

- **CAN1_SendFrame** (BufferNum, MessageID, FrameType, Length, Data, waitForRTR) – tato funkce odešle CAN rámec z bufferu (*BufferNum*) s ID definovaným v parametru *MessageID.* Počet odesílaných byte udává parametr *Length*, na adresu datových byte ukazuje ukazatel *Data*. Typ odesílaného rámce je definován v *FrameType* a poslední hodnotou se nastavuje možnost na čekání potvrzovacího rámce (*waitForRTR*).

- **CAN1_ReadFrame** (BufferNum, MessageID, FrameType, FrameFormat, Length, Data) – tato funkce přijme rámec z bufferu (*BufferNum*) zprávu s ID (*MessageID*) o délce (*Length*), data jsou uložena na adresu ukazatele *Data*. Typ přijímaného rámce je definován v *FrameType* a formát rámce je uveden v (FrameFormat). Tato funkce podle zvoleného ID přijme pouze zprávy se stejnými parametry *ID* a *FrameType*, který jsou definovány v nastavení bufferu nebo je možné je měnit pomocí metod.

- **CAN1_GetStateTX** (void) – vrátí počet CAN zpráv, které čekají na odeslání.

- **CAN1_GetStateRX** (void) – vrátí počet CAN zpráv, které čekají na přijetí.

- **CAN1_SetAcceptanceCode** (BufferNum, ID) – tato metoda nastaví zvolenému přijímacímu bufferu ID číslo CAN zprávy, potom tento buffer přijímá zprávy s tímto ID.

Použité události (events) beanu CAN1

- **CAN1_OnFreeTxBuffer** (BufferMask) – událost je vyvolána, pokud byla úspěšně odeslána zpráva. Pomocí parametru *BufferMask* lze zjistit, jaký buffer vyvolal událost.

- **CAN1_OnFullRxBuffer** (void) – událost je vyvolána, pokud je nějaký nastavený přijímací buffer plný, pomocí masky lze zjistit, který buffer vyvolal událost.

FlexCAN bean obsahuje možnost nastavení 16 bufferů pro příjem nebo odeslání zpráv. Prvních osm bufferů (0-7) bylo nastaveno pro příjem a zbývajících osm pro odesílání (8-15).

Při odesílání je rámec External Mode rozdělen po 8B do jednotlivých CAN zpráv. Tyto zprávy jsou odesílány pomocí nastavených osmi vysílacích bufferů. Jelikož není definována konstantní velikost rámce External Mode, proto je nutné nastavit vysílacím bufferům, aby se odesílala nejdříve zpráva s nejnižším ID (Lowest buffer transmited first – Lowest ID) a jednotlivé CAN zprávy jsou potom odesílány od nejnižšího ID. Počáteční ID je nastaveno na hodnotu 100. Po každém odeslaném rámci se ID nastaví na výchozí hodnotu. Na druhé straně host přijímá CAN zprávy od nejnižšího ID.

Přijímacím osmi bufferům je nastaveno ID zprávy, které mají přijímat. Nejnižšímu přijímacímu bufferu je nastaveno ID s hodnotou 200 a dalším bufferům postupně vždy o jedno číslo vyšší. Pokud je přijata zpráva je následně tomuto bufferu pomocí metody CAN1_SetAcceptanceCode nastaveno ID o jedno vyšší než je poslední nastavené pro přijímací buffery. Na druhé straně host odesílá CAN zprávy s rostoucím ID

4.3.2.3 Přidání zdrojových souborů do projektu PE

Popis přidání zdrojových souborů a jiná nastavení jsou popsány v kapitole 3.2.1.4 pro sériovou komunikaci External Mode. Pro novou komunikační vrstvu CAN, jsou ze šablon vytvořeny nové soubory:

- ext_can_pkt.c
- ext_svr_can_transp.c
- ext_can_PE_port.c

Popis modifikací souborů pro transportní vrstvu CAN bude popsáno v kapitole 4.3.3. Nastavení preprocesoru je stejné jako u sériové komunikace a je popsáno v kapitole 3.2.1.5.

4.3.3. Zdrojové soubory External Mode target

Nutné změny v souborech External Mode při portování na mikrokontrolér 56F8367, byly popsány (viz kapitola 3.2.3). Jsou stejné pro jinou komunikační vrstvu, protože jsou závislé na typu mikrokontroléru.

V této kapitole budou popsány změny, které souvisí se změnou komunikační vrstvy na targetu. V kapitole 2.3.2 jsou popsány zdrojové soubory External Mode. Na obrázku 11, je vidět struktura použití jednotlivých souborů pro sériovou transportní vrstvu. Pro změnu transportní vrstvy je nutné změnit zdrojové soubory označené v rámečku s popisem *soubory komunikační vrstvy*.

Zde je uveden postup pro změnu komunikační vrstvy na targetu:

ext_svr_can_transport.c a ext_can_utils.c

Prototypy funkcí Ext*, které definují volání funkcí transportní vrstvy jsou umístěny v *matlabroot*/rtw/c/src/ext_mode/common/ext_svr_transport.h. Pro sériovou implementaci jsou umístěny v souboru *matlabroot*/rtw/c/src/ext_mode/serial/ext_svr_serial_transport.c. Nutné body pro implementaci target komunikační vrstvy:

- Úprava šablony *matlabroot*/rtw/c/src/ext_mode/custom/ext_svr_custom_transport.c
  - Změna jména souboru transportní vrstvy, byla provedená změna na ext_svr_can_transport.c
  - Implementace těl funkcí *VISIBLE* FUNCTIONS, které volá hlavní soubor External Mode (ext_svr.c). V tělech jednotlivých funkcí je implementováno volání komunikačních funkcí, které přistupují k fyzické vrstvě protokolu. Musíme implementovat všechny funkce definované v souboru ext_svr_transport.h.
  - Definice datové struktury ExtUserData. Pokud tato struktura není definována je vytvořena tato struktura s prázdným polem.
  - Definice volby EXT_BLOCKING, kdy můžeme použít spojení s targetem jako jedno nebo vícevláknovou aplikaci.

- Ze šablony ext_custom_utils.c pro definování uživatelských symbolů a funkcí, byl vytvořen soubor ext_can_utils.c. Pokud jsou tyto funkce používány pro target, je nutné vložit do zdrojového souboru transportní vrstvy.
- Úprava šablon makefile pro podporu nové transportní vrstvy. Důležité je definovat volbu EXT_MODE.

Komunikace prostřednictvím sériové linky RS-232 a CANu je velmi podobná, protože po sběrnici CAN jsou odesílány zprávy po 8B oproti sériové lince po 1B. Proto nově vzniklé soubory jsou modifikací zdrojových souborů sériové transportní vrstvy External Mode.

ext_can_PE_port.c

Ze šablony matlabroot\rtw\c\src\ext_mode\custom\ext_serial_custom_port.c byl vytvořen soubor ext_can_PE_port.c. V tomto souboru byly vytvořeny komunikační funkce využívající bean CAN1. Metody a události beanu CAN1 obsluhují fyzický CAN port a jsou popsány v kapitole 4.3.2.2. V tomto souboru byly implementovány těla následujících funkcí:

- **ExtCanPortSetData** – odešle jeden rámec External Mode na sběrnici. Tento rámec je odeslán po jednotlivých CAN zprávách po 8 byte. Více informací o struktuře rámce External Mode (viz kapitola 4.3.1.1) a o odesílání CAN zpráv (viz kapitola 4.3.2.2).
- **ExtCanGetRawData** – Přijme jeden rámec External Mode. Tento rámec se může skládat z více CAN zpráv, tak funkce čeká dokud nepřijme kompletní rámec External Mode.
- **ExtCanPortDataPending** – Tato funkce vrací počet CAN zpráv čekajících na příjem ze sběrnice CAN.

ext_can_pkt.c

Tento soubor vznikl ze souboru ext_serial_pkt.c, který slouží jako šablona. V tomto souboru jsou hlavně funkce, které zajišťují sestavování a rozebírání rámce External Mode z přijatých (odeslaných) CAN zpráv. CAN zprávy jsou příjímány a odeslány pomocí mfunkcí definovaných v souboru ext_can_PE_port.c (pro target),

ext_can_port.c (pro host). Hlavní odlišnosti oproti sériové lince jsou v tom, že rámec External Mode je odesílán (přijímán) pomocí CAN zpráv. CAN zprávy mohou obsahovat 8 data byte oproti sériové lince, kde je informace posílána po jednotlivých byte. Zde jsou uvedeny hlavní funkce, které obsahuje tento soubor:

- **SetExtCanPacket** (ExtCanPacket *pkt, ExtCanPort *portDev) – v této funkci je vytvořen odesílaný rámec External Mode z objektu datového paketu a následně předán funkci *ExtCanPortSetData*, která ho odešle na sběrnici.
- **GetExtCanPacket** (ExtCanPacket *pkt, ExtCanPort *portDev) – po příjmu dat, jsou tyto data vkládána do objektu datového paketu. Nejdříve se dekóduje v hlavičce počet datových byte a následně se tyto byte ukládají do objektu datového paketu.

4.4 Host

Tato kapitola bude obsahovat popis implementace podpory External Mode (strana host) pro komunikaci po sběrnici CAN. Aby bylo možné komunikovat po sběrnici CAN z PC je použit USB-CAN převodník. External Mode na straně host bude komunikovat s driverem USB-CAN převodníku.

Implementace podpory External Mode na straně host (Simulinku) bude obsahovat více změn. Hlavní body následujích změn jsou zde popsány:

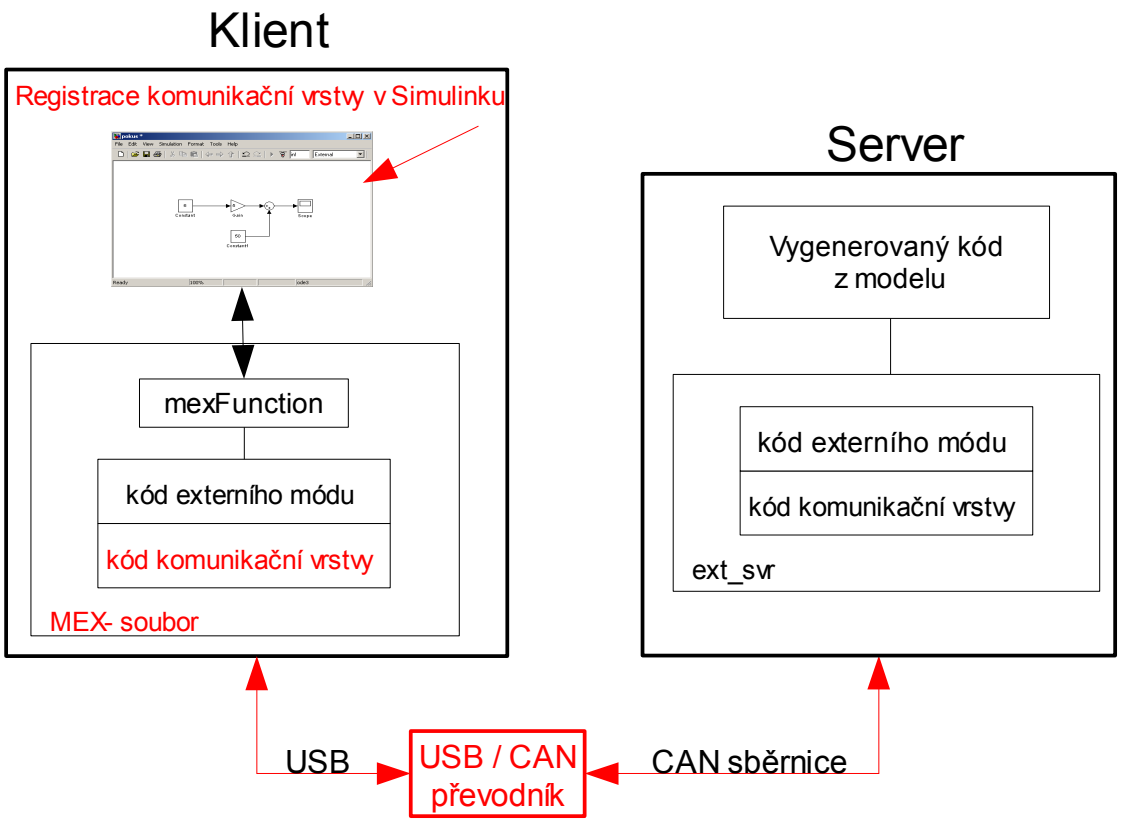
1. Úprava zdrojových šablon External Mode. Hlavní šablona souboru pro podporu External Mode je uložena v adresáři *matlabroot*/rtw/c/src/ext_mode/custom/ext_custom_transport.c. V této šabloně je potřeba definovat funkce pro volání nové komunikační vrstvy Tuto šablonu je potřeba změnit následujícím způsobem:
    - Přejmenování šablony ext_custom_transport.c jménem nové transportní vrstvy.

- Změna funkcí definovaných v souboru ext_custom_transport.c.jako je (připojení, odpojení, odesílání, příjem, čekání a definování uživatelských dat). Pomocí těchto funkcí jsou volány funkce komunikační vrstvy, které jsou definovány v souboru *ext_custom_port.c* (viz obrázek 11).

2. Vytvoření uživatelského MEX-souboru užitím funkce *mex* z Matlabu, pro komunikační vrstvu CAN, která bude využívat USB-CAN převodník.

3. Registrování nové transportní vrstvy External Mode v softwaru Simulinku.

Nutné změny pro vytvoření nové komunikační vrstvy jsou zobrazeny červeně na obrázku 25.



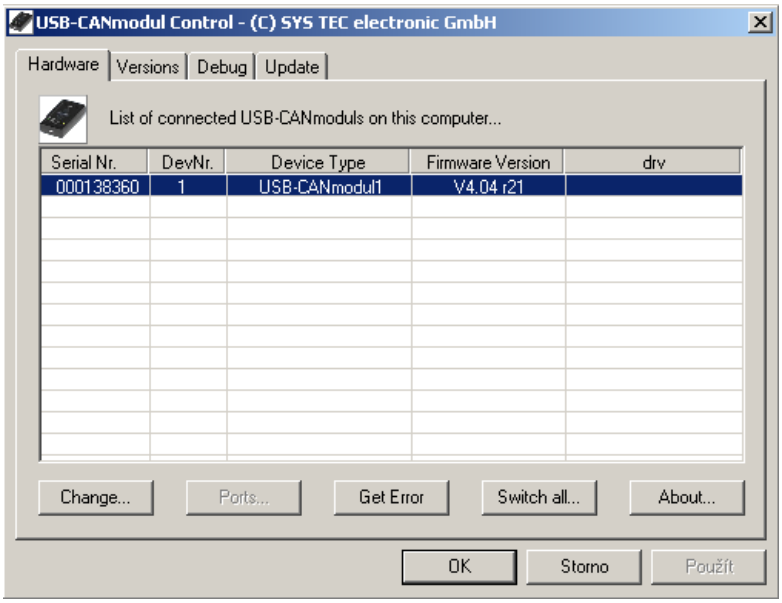
Obrázek 25: Host - změny související s novou transportní vrstvou CAN

Nejdřive bude popsán převodník USB-CAN, protože použitím jeho funkcí budou implementovány těla funkcí External Mode pro komunikaci po sběrnici CAN.

4.4.1. USB-CAN převodník

4.4.1.1 Instalace USB-CAN převodníku

Pro instalaci je nutné dodržovat postup popsaný (viz kapitola 1 v literatuře [10]). Je nutné nejdříve spustit sotwarovou instalaci a potom až připojit fyzický USB konektor USB-CAN převodníku do PC. Po instalaci je možné získat informace o zařízení. Tyto informace je možné nalézt v *Start -> Ovládací panely -> USB-CAN modul control* (viz obrázek 26). Zde je možné zjistit a nastavit číslo zařízení, to je vhodné k rozlišení v případě, že používáme více nainstalovaných převodníků na jednom počítači. Dále je možné nastavit jestli má být převodník využíván jednou nebo více aplikacemi, debug informace nebo třeba zjistit verzi driveru.



Obrázek 26: USB-CAN informační panel

4.4.1.2 Připojení převodníku na sběrnici

Použitý převodník USB-CAN od firmy SYS TEC Elektronic typ 3204000 obsahuje jeden CAN kanál s *high-speed* vysílačem Philips 82C251. Převodník obsahuje pro připojení na sběrnici CAN konektor DB-9 CAN. Zapojení pinů je možné vidět v tabulce 3.

Tento použitý typ převodníku nemá připojen zakončovací odpor o velikosti 120Ω. Proto je nutné při připojování převodníku ke CAN sběrnici přidat tento zakončovací odpor. Odpor se připojuje mezi piny CANL a CANH.

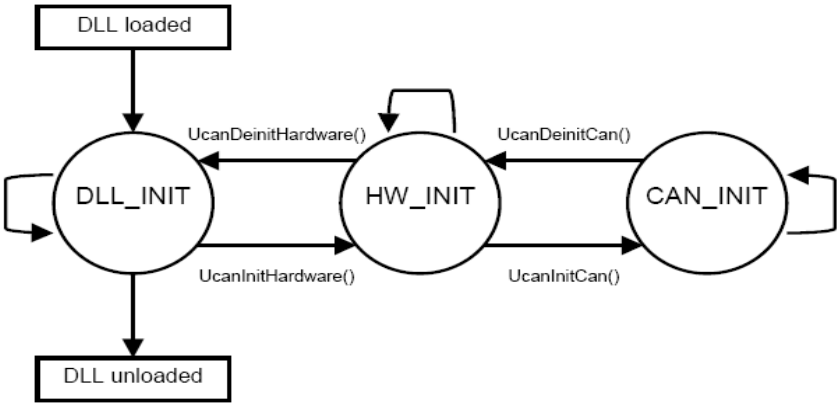
Pin	Signál	Pin	Signál
1	nezapojen	2	CANL
3	GND	4	nezapojen
5	CAN stíněný	6	GND
7	CANH	8	nezapojen
9	VCC		

Tabulka 3: Zapojení pinů CAN konektoru DB-9

4.4.1.3 Softwarová podpora pro Windows

Při instalaci převodníku je vytvořen adresář umístěný v C:\Programs_Files\SYSTEC-electronic\USB-CANmodul Utility Disk. V tomto nainstalovaném adresáři jsou k dispozici demo úlohy, dokumentace a hlavně soubory (USBCAN32.LIB, USBCAN32.H) a knihovna funkcí (USBCAN32.DLL) pro práci s tímto převodníkem.

Pro každý USB-CAN modul, který využívá knihovnu DLL jsou generovány tři stavy (viz obrázek 27).



Obrázek 27: Stavový diagram softwaru DLL „Přezato z [10]“

Po startu aplikace a načtení DLL přecházíme do prvního stavu DLL_INIT. V tomto stavu voláním knihovních funkcí *UcanInitHardware()* změníme stav na HW_INIT. V stavu HW_INIT můžeme volat mnoho funkcí, které nastavují různé

možnosti komunikace, ale nelze odesílat (přijímat) zprávy. Voláním funkce *UcanInitCan()* přecházíme do stavu CAN_INIT. V tomto stavu je již možné posílat (přijímat) zprávy.

Zpět se můžeme dostat voláním funkcí *UcanDeinitCan()* do stavu HW_INIT a dále funkcí *UcanDeinitHardware()* do výchozího stavu DLL_INIT. V tomto stavu je teprve možné ukončit program.

Přehled všech možných funkcí, které lze použít v jednotlivých stavech popisuje tabulka 11 v literatuře [10]. Dále budou uvedeny důležité funkce včetně popisu, které byly použity při implementaci CAN komunikační vrstvy.

DLL_INIT

- **UcanInitHardware** (*pUcanHandle_p,bDeviceNr_p, fpCallbackFkt_p) – tato funkce inicializuje USB-CAN modul. V prvním parametru je definována adresa pointeru na *handle*. V druhém parametru se zadává číslo USB-CAN zařízení (viz obrázek 26). Poslední parametr obsahuje adresu *callback* funkce.

- **UcanDeinitHardware** (UcanHandle_p) – opak funkce UcanInitHardware, vstupní parametr je handle, který byl získán při inicializaci.

HW_INIT

- **UcanInitCanEx** (UcanHandle_p,* pInitCanParam_p) – tato funkce inicializuje CAN interface s nastavením parametrů a tím přejde do stavu CAN_INIT. První parametr je handle získaný z funkce UcanInitHardware, druhým parametrem je ukazatel na adresu struktury, která obsahuje inicializační data.

- **UcanDeinitCan** (UcanHandle_p) – tato funkce se odpojí od CAN interface, tím přejde ze stavu CAN_INIT do stavu HW_INIT. Je opakem funkce UcanInitCanEx**.**

- **UcanGetStatus** (UcanHandle_p,* pStatus_p) – tato funkce vrátí chybový

stav USB-CAN modulu, vstupní parametr je handle a ukazatel pro uložení chybového stavu. Tato funkce má výhodu v tom, že po přečtení chyby ji vymaže, bez nutnosti volání funkce pro vymazání (reset).

CAN_INIT

- **UcanReadCanMsg** (UcanHandle_p,* pCanMsg_p) – funkce přečte Can zprávu z přijímacího bufferu. Druhý parametr ukazuje na adresu, kde je uložena přijatá CAN zpráva.

- **UcanWriteCanMsg** (UcanHandle_p,* pCanMsg_p) – funkce odešle Can zprávu do vysílacího bufferu. Druhý parametr ukazuje na adresu struktury, která obsahuje data odesílané CAN zprávy.

- **UcanGetMsgPending** (UcanHandle_p, bChannel_p, dwFlags_p,* pdwCount_p) – tato funkce vrací počet CAN zpráv, které jsou umístěny v bufferu. Parametr *bChannel* určuje, který kanál se kontroluje. Parametr *dwFlags_p* určuje jaký buffer se bude kontrolovat (vysílací, přijímací). Poslední parametr je ukazatel na data, kam se ukládá počet čekajících zpráv.

Detailní informace o použitých funkcích včetně jejich popisu, definice parametrů nastavení a návratových hodnot je popsáno (viz kapitola 2.3.2.1 v literatuře [10]).

4.4.2. Změna zdrojových souborů External Mode (host)

V této kapitole budou popsány změny, které souvisí se změnou komunikační vrstvy na straně Simulinku (host).Tyto změny jsou velmi podobné změnám provedeným na straně targetu pro CAN transportní vrstvu popsanou v kapitole 4.3.3 V kapitole 2.3.2 jsou popsány zdrojové soubory External Mode. Na obrázku 11 je vidět struktura použití jednotlivých souborů pro sériovou komunikační vrstvu pro CAN je stejná. Pro změnu komunikační vrstvy je nutné změnit zdrojové soubory označené v rámečku s popisem *soubory komunikační vrstvy*.

ext_can_transport.c a ext_can_utils.c

Prototypy funkcí Ext*, které definují volání funkcí transportní vrstvy jsou umístěny v *matlabroot*/rtw/c/src/ext_mode/common/ext_svr_transport.h. Pro sériovou implementaci jsou umístěny v souboru *matlabroot*/rtw/c/src/ext_mode/serial/ext_serial_transport.c. Nutné body pro implementaci target transportní vrstvy:

- Úprava šablony *matlabroot*/rtw/c/src/ext_mode/custom/ ext_custom_transport.c
  - Změna jména souboru transpotrní vrstvy, byla provedená změna na ext_can_transport.c
  - Implementace těl funkcí *VISIBLE* FUNCTIONS, které volá hlavní soubor External Mode (ext_svr.c). V tělech jednotlivých funkcí je implementováno volání komunikačních funkcí, které obsluhují fyzickou vrstvu protokolu. Musíme implementovat všechny funkce definované v souboru ext_transport.h.
  - Definice datové struktury ExtUserData. Pokud tato struktura není definována je vytvořena tato struktura s prázdným polem.
  - Definice volby EXT_BLOCKING, kdy můžeme použít spojení s targetem jako jedno nebo vícevláknovou aplikaci.
- Ze šablony ext_custom_utils.c pro definování uživatelských symbolů a funkcí, byl vytvořen soubor ext_can_utils.c.
- Úprava šablon makefile pro podporu nové transportní vrstvy. Důležité je definovat volbu EXT_MODE.

ext_can_port.c

Ze šablony matlabroot\rtw\c\src\ext_mode\custom\ext_custom_port.c byl vytvořen soubor ext_can_port.c. V tomto souboru byly vytvořeny komunikační funkce využívající funkcí USB-CAN převodníku. Tyto funkce obsluhují fyzický CAN port zmíněného převodníku a jsou popsány v kapitole 4.4.1.3. V tomto souboru byly implementovány těla následujících funkcí:

- **ExtCanPortCreate** – vytvoří objekt typu *ExtCanPort*. Tento objek slouží pro přístup k fyzickému portu.
- **ExtCanPortConnect** – tato funkce zajišťuje připojení objektu *ExtCanPort*

k fyzickému portu (driveru USB-CAN převodníku) včetně jeho nastavení.

- **ExtCanPortDisconnect** – tato funkce odpojí objekt *ExtCanPort od* fyzického portu (driveru USB-CAN převodníku).
- **ExtCanPortSetData** – odešle jeden rámec External Mode.
- **ExtCanGetRawData** – přijme jeden rámec External Mode. Tento rámec se může skládat z více CAN zpráv, tak čeká dokud nepřijme celý rámec External Mode.
- **ExtCanPortDataPending** – tato funkce vrací počet CAN zpráv čekajících na příjem ze sběrnice CAN.

ext_can_pkt.c

Tento soubor je stejný pro stranu host i target. Soubor je popsán v kapitole 4.3.3.

Systém odesílání rámců External Mode

Popis rámců External Mode pro CAN komunikační vrstvu je popsáno v kapitole 4.3.1.1. Tyto rámce jsou odesílány (přijímány) pomocí CAN zpráv, systém odesílání je popsáno v kapitole 4.3.2.2. S použitím USB-CAN převodníku na straně hostu není potřeba nastavovat v přijímacím bufferu nastavovat ID zpráv, které se mají přijímat. Přijímají se CAN zprávy s jakýmkoliv ID, protože jsou zprávy ukládány do fronty v převodníku a pomocí driveru je postupně čteme. Odesílací buffer je řešen podobně.

Z přijatých CAN zpráv od nejnižšího ID od hodnoty 100 je následně skládán rámec External Mode. Stejně je rámec External Mode odesílán pomocí CAN zpráv od ID 200. Nastavení ID čísla zprávy lze změnit.

4.4.3. MEX-soubor

Matlab API umožňuje interakci Matlabu s okolními aplikacemi (viz [11]). Matlab API lze také použít k vytváření funkcí pro Matlab napsaných v jazyku C, C++ nebo Fortran (tzv. MEX-files) a tyto funkce následně jednoduše volat z Matlabu. MEX- soubor obsahuje funkce, které se skládají ze dvou částí. První, tzv. Gateway

interface, zajišťuje komunikaci s Matlabem a je obdobou funkce *main()* v jazyce C. Druhá část je vlastní kód. Výhodou mex funkcí je jejich vyšší rychlost oprotí m- funkcím.

V External Mode tyto MEX-soubory obsahují zdrojový kód External Mode a příslušný zdrojový kód použité komunikační vrstvy. Popis zdrojových souborů External Mode je popsáno v kapitole 2.3.2. Na obrázku 9 jsou vidět obsažené části MEX-souboru pro External Mode. Matlab (Simulink) dle zvoleného interface volá různý MEX-soubor, se kterým komunikuje prostřednictvím vstupní mex funkce (mexFunction). Výběr nové komunikační vrstvy (MEX-souboru) je popsán (viz kapitola 4.4.4).

4.4.3.1 Tvorba MEX-souboru

MEX-soubor vznikne překladem a linkováním zdrojových souborů napsaných v jazyce C, C++, Fortran pomocí funkce *mex* definované v Matlabu. Po spuštění následujícího příkazu v příkazové řádce

mex -setup

můžeme vybrat nebo změnit překladač, který má být použit při tvorbě MEX- souboru. Při práci byl použit překladač *Microsoft Visual C++ 2005,* aby mohl být použit debug MEX-souboru (viz kapitola 4.4.3.6).

Použití mex funkce je následující:

*mex -(**možnosti nastavení**) -(**názvy souborů**)*

možnosti nastavení – specifikují hodnoty a proměnné, které jsou dále předávány ve formě argumentů překladači, linkeru a jiným nástrojům. Zde uvedu jen argumenty, které byli použité při tvorbě MEX-souboru pro External Mode.

- -g - s tímto parametrem je vytvořen MEX-soubor, který obsahuje přidané informace pro debugger. Pokud není tento parametr použit, tak je defaulně nastavena optimalizace kódu (volbou -o).

- -I<cesta> - přidá cestu pro hledání hlavičkových souborů, které jsou definovány ve zdrojových souborech pomocí direktivy *#include.*

- -l<název> - přidání knihovny funkcí. Při použití na Windows je z <název>, vytvořeno <název>.lib nebo lib<název>.lib.

- -L<directory> - přidá do seznamu cest, cestu ke knihoně specifikované příkazem -l.

- -output <název> - touto možností lze vybrat název MEX-souboru a lze zadat cestu kam se má soubor uložit.

- -v – Verbose mód. Vypíše v konzoli hodnoty důležitých proměných, vypíše každý kompilační krok a také konečný krok linkeru.

Názvy souborů - soubory mohou být kombinací zdrojových souborů, object souborů a knihovních souborů.

4.4.3.2 MEX-soubor pro sériovou podporu External Mode

MEX-soubor vytvořený pro sériovou komunikaci se jmenuje *ext_serial_win32.mexw32*. Tento soubor je uložen v adresáři *matlabroot*/toolbox/rtw/rtw/. Dále je uveden příkaz pomocí něhož můžeme vytvořit MEX-soubor pro sériovou komunikaci RS-232:

mex rtw\ext_mode\common\ext_comm.c rtw\ext_mode\common\ext_convert.c rtw\ext_mode\serial\ext_serial_transport.c rtw\c\src\ext_mode\serial\ ext_serial_pkt.c rtw\c\src\ext_mode\serial\ext_serial_win32_port.c -Irtw \c\src\ext_mode\common -Irtw\c\src\ext_mode\serial -Irtw\ext_mode\common -output toolbox\rtw\rtw\ext_serial_win32_comm -DWIN32

4.4.3.3 MEX-soubor pro CAN podporu External Mode

Byl vytvořen MEX-soubor s názvem ext_can_systec.mexw32. Tento soubor je potřeba uložit do adresáře *matlabroot*/toolbox/rtw/rtw/, aby mohl být spuštěn Matlabem (Simulinkem). Registrace výběru komunikační vrstvy (MEX-souboru) popisuje kapitola 4.4.4.

Pro tvorbu MEX-souboru byly použity soubory popsané v kapitole 4.4.2 a knihovna funkcí popsaná v kapitole 4.4.1.3. Při tvorbě MEX-souboru byl problém při linkování knihovny USB-CAN převodníku s ostatními zdrojovými soubory. Tento problém nastal, pokud byla knihovna vkládána pomocí příkazu *-l<názevknihovny>* a *-L<adresář knihovny>*.

mex rtw\ext_mode\common\ext_comm.c ….zdrojové soubory....-output toolbox\rtw\rtw\ext_can_comm -DWIN32 -L rtw\ext_mode\can -lUSBCAN32

Problém byl odstraněn pokud byla knihovna přidána jako ostatní zdrojové soubory, bez použití speciálních argumentů na přidání knihovny. Příklad ukazuje vytvoření MEX-souboru, pokud byly všechny zdrojové, hlavičkové soubory a knihovna umístěny v jednom adresáři. Potom příkaz pro vytvoření MEX-souboru je následující:

mex ext_can_port.c ext_comm.c ext_convert.c ext_can_transport.c ext_can_pkt.c USBCAN32.lib -output 'C:\Program Files\MATLAB\R2008a\toolbox\rtw\rtw\ext_can_systec' -DWIN32

Následující příkaz vytvoří MEX-soubor pro transportní vrstvu CAN, pokud jsou zdrojové soubory External Mode uloženy (viz tabulka 4).

mex rtw\ext_mode\common\ext_comm.c rtw\ext_mode\common\ext_convert.c rtw\ext_mode\can\ext_can_transport.c rtw\c\src\ext_mode\can\ext_can_pkt.c rtw\c\src\ext_mode\can\ext_can_port.c USBCAN32.lib -Irtw \c\src\ext_mode\common -Irtw\ext_mode\common -Irtw\c\src\ext_mode\can -Irtw \ext_mode\can -output 'C:\Program Files\MATLAB\R2008a\toolbox\rtw\rtw\ext_can_systec -DWIN32

Tento příkaz je uložen v souboru mex_can_systec.m.

4.4.3.4 Aktualizace souborů v Matlabu

Pokud je uložen nebo změněn jakýkoliv soubor v adresáři matlabroot/toolbox nebo je přidán nebo odebrán soubor je nutné spustit v Matlabu funkci:

rehash toolbox

Tato funkce nemá žádné vstupní parametry a aktualizuje seznam adresářů a souborů v matlabroot/toolbox. Porovná časové stopy nahraných funkcí s funkcemi, které jsou uloženy na disku. Pokud jsou časové stopy funkcí na disku novější, tak Matlab aktualizuje svůj seznam funkcí.

4.4.3.5 Vstupní parametry pro MEX-soubor transportní vrstvy CAN

Nastavení vstupních parametrů MEX-souboru se nachází v *Tools->Real-Time Workshop->Options->Configuration Parameters* (viz obrázek 10). Zde můžeme nastavit parametry, které jsou předány External mode interface MEX-souboru (ext_can_systec). Pro CAN komunikaci lze nastavit následující parametry:

- **Verbosity level -** nastavení úrovně výpisu informací, které jsou vypisovány na konzoli v Matlabu během přenosu. Lze nastavit hodnotu 0 pro žádné informace nebo hodnotu 1 pro detailní výpis.
- **ID zařízení –** číslo připojeného USB-CAN převodníku (viz obrázek 26).
- **Přenosová rychlost –** číselná hodnota přenosové rychlosti, pro CAN komunikaci je možné nastavit (10, 20, 50, 100, 125, 250, 500, 800, 1000) kbit/s.

4.4.3.6 Debug MEX-souborů

Pokud je MEX-soubor vytvořen pomocí volby -g (viz kapitola 4.4.3.1) je možný debug MEX-souboru za pomoci zdrojových souborů, ze kterých byl MEX-soubor vytvořen. Tím není možné spustit tento soubor na jiném počítači, protože MEX- soubor má vazbu na zdrojové soubory z nichž byl vytvořen. Následně bude popsán postup pro debug MEX-souboru v prostředí Microsoft Visual Studio.

Ve vývojovém prostředí Microsoft Visual Studio je možný debug MEX-souboru pomocí vstupních zdrojových souborů MEX-souboru. Je možné krokovat zdrojový kód, nastavovat breakpointy a sledovat proměnné. Dále uvádím stručný popis nastavení:

- Výběr překladače – při tvorbě MEX-souboru je nutné vybrat překladač

Microsoft Visual C++ 2005. Tento výběr se provede napsáním do příkazové řádky v Matlabu příkaz *mex -setup*, čím se vybere překladač.

- Vytvoření MEX-souboru – je nutné vytvořit MEX-soubor s volbou *-g*.
- Spustit Visual Studio – nesmí se zavírat Matlab okno.
- V menu ve Visual Studiu zvolit *Tools -> Attach to Process.*
- Z dialogového okna vybrat proces MATLAB a potvrdit Attach.
- Otevřít zdrojový soubor použitý v MEX-souboru volbou *File -> Open -> File.*
- Nastavení breakpointů – lze provést pravým klikem myši nebo z menu *Breakpoint -> Insert Breakpoint*.
- Poslední krok spočívá ve spuštění MEX-souboru v Matlabu.

4.4.4. Registrace transportní vrstvy v softwaru Simulinku

Transportní vrstva External Mode se vybírá pro model v Simulinku v záložce *Tools -> RealTime Workshop -> Options,* tento výběr je možné vidět na obrázku 28. Změna je nutná provést pro Target, podle něhož je generován zdrojový kód v modelu (viz kapitola 2.1.1.1). Registraci je možné provést pro uživatelské Targety nebo Targety podporované firmou MathWorks.

Pro Targety podporované firmou MathWorks se registrace nové transportní vrstvy provádí v souboru *matlabroot*/toolbox/simulink/simulink/extmode_transports.m. Tento soubor je uložen (viz tabulka 4). Při práci byl používán Target ERT, který podporuje External Mode pro ukázku uvádím příklad registrace nové transportní vrstvy do tohoto Targetu (změny jsou podtrženy).

ERT Target

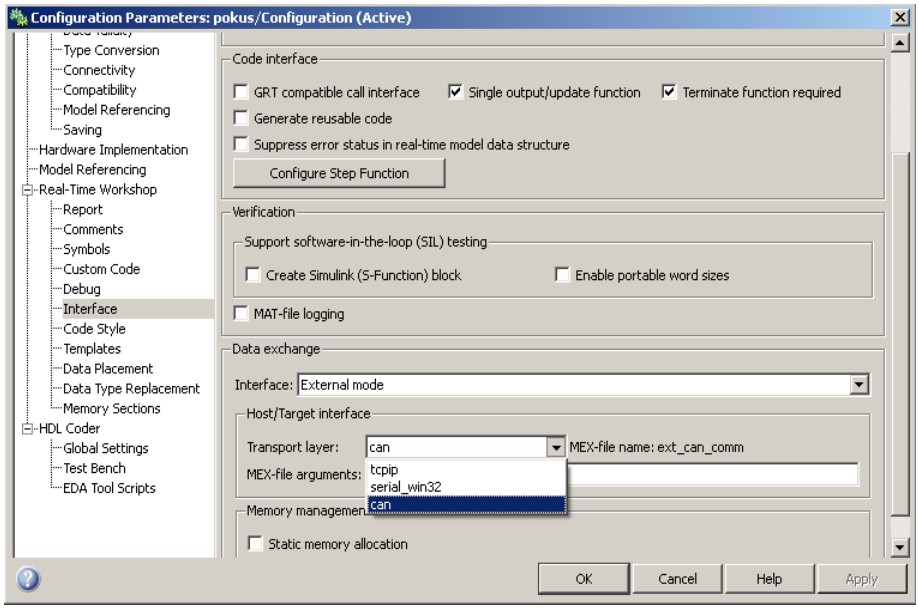
```
strcmp(sysTargFile, 'ert.tlc');
```

```
transports = {
'tcpip1'          % Index 0 in the template makefiles
'serial_win32'    % Index 1
'can'             % Index 2
                 };
mexfiles = {
'ext_comm',                  % MEX-File for 'tcpip'
'ext_serial_win32_comm', % MEX-File for 'serial_win32'
'ext_can_systec'             % MEX-File for 'can'
                 };
```

ert.tlc – je jméno Targetu, do kterého je nová komunikační vrstva registrována

transport – je jméno komunikační vrstvy, které je zobrazované v nastavení *Transport layer* v panelu *Configuration Parameters -> Interface*. Zobrazení pro komunikační vrstvu CAN je možné vidět na obrázku 28.

mexfile – je jméno MEX- souboru vytvořeného pro konkrétní komunikační vrstvu. Tento název je zobrazován vedle názvu transportní vrstvy v nastavení *Transport layer* (viz obrázek 28).



Obrázek 28: Výběr transportní vrstvy CAN v nastavení

4.4.5. Demonstrační úloha a testování výkonových parametrů

Byla vytvořena demonstrační úloha pro External Mode s přenosem pomocí CAN komunikace. Popis potřebných změn při portování na mikrokontrolér 56F8367 byly popsány v kapitole 3 a také v kapitole 4.3. Tabulka 4 zobrazuje přehled zdrojových souborů, které musely být modifikovány. Zdrojové soubory External Mode i modelu včetně PE projektu jsou uloženy na přiloženém CD v adresáři external_mode\can.

Target

ext_svr.c updown.c	Tyto soubory je nutné přehrát místo souborů, které jsou uloženy v matlabroot\rtw\c\src\ext_mode\common
ext_svr_can_transp.c ext_can_pkt.c ext_can_PE_port.c	Tyto soubory je nutné nahrát do adresáře can, který se vytvoří v adresáři matlabroot\rtw\c\src\ext_mode

Host

ext_can_transport.c ext_can_utils.c ext_can_port.c	Tyto soubory je nutné nahrát do adresáře can, který se vytvoří v adresáři matlabroot\rtw\ext_mode
mex_can_systec.m	Tento m-file obsahuje příkaz pro vytvoření MEX-souboru ext_can_systec
extmode_transports.m	Soubor obsahující registraci komunikační vrstvy v Simulinku , tento soubor je potřeba nahrát do adresáře matlabroot/toolbox/simulink/simulink/

Tabulka 4: Modifikované soubory External Mode pro CAN

4.4.5.1 Demonstrační úloha

Byla vytvořena demonstrační úloha, aby ukázala použití výsledného řešení realizace External Mode s komunikací po sběrnici CAN na praktickém příkladu. Dále poskytuje návod vedoucí k vytvoření příkladu, který je sepsán proto, aby sloužil pro lidi, kteří budou tuto práci využívat. Také může být jako zdroj pro vyjasnění nesrovnalostí, které mohou vzniknout nedostatečným prostudováním problematiky. Těd budou postupně popsány kroky, teré je nutné udělat, aby byla

úspěšně spuštěna demostrační úloha:

Host (Simulink)

1. Instalace USB-CAN převodníku (viz kapitola 4.4.1).
2. Upravené zdrojové soubory External Mode (viz kapitola 4.4.2) nahrát do adresářů dle tabulky 4.
3. Vytvoření MEX-souboru (viz kapitola 4.4.3.1), příkaz je také uložen na přiloženém CD v adresáři external_mode\can\mex v m-file souboru mex_can_systec.m. Tento příkaz je nutné spustit v příkazovém řádku Matlabu.
4. Registrace transportní vrstvy CAN do softwaru Simulinku (viz kapitola 4.4.4).
5. Spuštění demo modelu s názvem pokususb.mdl, který je uložen v adresáři external_mode\can\model.
    - výběr simulačního módu (viz kapitola 2.2.1)
    - nastavení zvoleného Interface- External mode v nastavení v okně modelu *Tools->Real-Time Workshop->Options->Configuration Parameters,* výběr transportní vrstvy (viz obrázek 28) a nastavení MEX parametrů (viz kapitola 4.4.3.5)
    - nastavení kroku simulace Fixed Step s definovanou periodou a nastavení zvoleného Targetu (např. ERT)
    - generování kódu z modelu

Target (MCU)

1. Upravené zdrojové soubory External Mode nahrát do adresářů dle tabulky 4.
2. Spustit projekt v Metrowerks Code Warrior, demonstrační projekt se jménem external_mode.mcp je uložen na CD v adresáři external_mode\can\PE\external_mode_can.
    - nastavení FreescaleCAN portu na MCU
    - nastavení cesty k zdrojovým souborům vygenerovaného

modelu nebo překontrolování cest k ostatním použitým zdrojových souborů, případné nastavení jejich cesty pomocí *access paths* toto nastavení je popsáno v kapitole 4.3.2

- nastavení příznaku EXT_MODE v preprocesoru (viz kapitola 3.2.1.5)
- následuje přeložení projektu pomocí make a spuštění

Target je spuštěn, ale čeká na startovací rámec od Simulinku. Popis spuštění a ovládání simulace v Simulinku je popsána (viz kapitola 2.2).

4.4.5.2 Testování výkonových parametrů

Testování výkonových parametrů External Mode s komunikací po sběrnici CAN bylo porovnáváno se sériovou komunikací RS-232. V Simulinku byl vytvořen model sestavený z jednotlivých bloků, které charakterizují regulační algoritmus. Dále byly přidány monitorovací bloky (Scope), pomocí kterých lze sledovat signály mezi libovolnými bloky modelu.

Průběh komunikace mezi Simulinkem a targetem je popsán v kapitole 2.4.3.3 a je ho možné vidět na obrázku 15. Pokud je při inicializaci nastaven upload zvolených signálů je po odeslání odpovědi (EXT_MODEL_START_RESPONSE) spuštěn podle nastavené periody výpočet signálů modelu. Hodnoty jednotlivých signálů jsou ukládány do paměti a následně odesílány směrem k Simulinku dle možností komunikačního kanálu.

Testování výkonových parametrů bylo provedeno pro případ, kdy Simulink neodesílá žádné žádosti na target (EXT_SETPARAM) pouze zasílá žádosti o aktuální čas simulace (EXT_GET_TIME). Target odesílá upload rámec (EXT_UPLOAD_LOGGING_DATA) s hodnotami jednotlivých signálů. Základní délka tohoto rámce pro přenos jedneho vypočteného signálu je 41B (pro sériovou komunikaci) pro CAN komunikační vrstvu byl upraven na délku 40B (viz kapitola 4.3.1.1), aby tento rámec mohl být odeslán pomocí pěti CAN zpráv. Další modifikace rámců pro komunikační vrstvu CAN je popsáno (viz kapitola 4.3.1.1). S každým dalším signálem se rámec zvětšuje o 4B.

Tabulka 5 ukazuje dosažené maximální vzorkovací periody (krok výpočtu modelu) pro komunikaci po sběrnici CAN (1Mbit/s) a sériovou linku RS-232 (57 600 bit/s). Test byl proveden s proměným počtem přenášených signálů (int32).

	Maximální krok výpočtu modelu	
Počet přenášených signálů	RS-232 T [ms]	CAN T [ms]
1	9,1	0,7
2	10	0,78
3	10,9	0,81
4	12	0,89
5	13,1	0,92

Tabulka 5: Porovnání výkonových parametrů

Kapitola 5

5 Možnosti použití JTAG/ONCE komunikace

V této kapitole budou teoreticky popsány možnosti použití JTAG/OnCE pro External Mode. Poznatky v této kapitole slouží pro informativní přehled, jakým směrem by mohla pokračovat podpora External Mode.

Joint Test Action Group (JTAG) je název, který vznikl z původního účelu pro testování desek plošných spojů pomocí *boundary scanu*. Později normalizované jako *IEEE 1149.1 Standard Test Access Port and Boundary-Scan* Architecture (viz [12]).

JTAG se dnes široce používá pro ladící účely integrovaných obvodů. Mnoho vestavěných procesorů na trhu podporují JTAG, když mají dostatek pinů. Debug programy komunikují s čipy prostřednictvím JTAG a umožňují provádět krokování programu nebo vkládání breakpointů. Dále se používá pro produkty, které nemají jinou sběrnici pro nahrání programu a ladění.

5.1 JTAG/OnCE pro Freescale 56F800

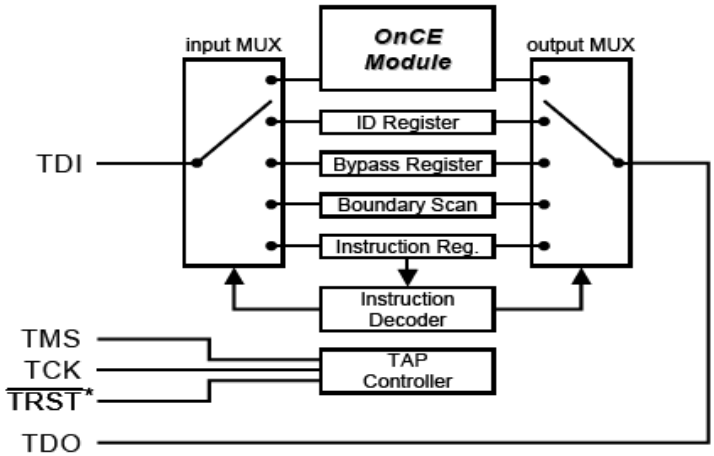
Toto rozhraní umožňuje zápis a čtení flash paměti jednotek procesoru pomocí JTAG/OnCE rozhraní. Toto rozhraní obsahuje dva moduly:

- OnCE (*On-Chip Emulation*) modul pro emulaci
- JTAG konektor (nazýván jako Test access port (TAP))

Toto rozhraní umožňuje nahrání programu včetně funkcí pro debug. To je výhodou, protože není nutné kupovat drahý kabel, který umožňuje funkce emulace, více informací (viz [13]).

5.1.1. JTAG

K JTAG rozhraní je vyhrazen přístup pomocí pěti pinů. Základem je *TAP controller*, který obsahuje 16-stavový automat. Instrukčním dekodérem je dekódována instrukce, která je následně vykonávána. Freescalle implementoval OnCE rozhraní (*OnCE module*) pro debug aplikace. Schéma JTAG rozhraní je možné vidět na obrázku 29.



Obrázek 29: JTAG

JTAG (TAP) vykonává nejčastěji následující funkce:

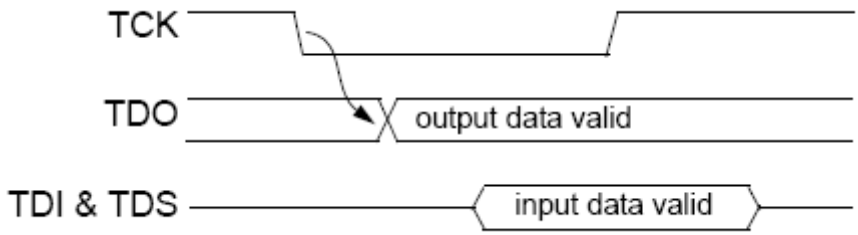
- posun instrukce dovnitř a ven a následné dekódování
- příjem a vysílání dat
- zápis do JTAG/OnCE registrů

JTAG piny

Ve specifikaci rozhraní JTAG je definován minimální počet signálů. Je potřeba minimálně čtyři signály TDI, TDO, TCK a TMS.

- **TDI** - vstup sériového datového toku do JTAG a OnCE modulu.
- **TDO** - výstup sériového datového toku do JTAG a OnCE modulu.
- **TMS** - vstupní sekvence dat pro funkci TAP stavového automatu.
- **TCK** - signál hodin pro posun dat, maximální frekvence pro TCK je 1/8 maximální frekvence pro 56F80x (tj. 5MHz pro sběrnici procesoru 40MHz).

Časování JTAG signálů je možné vidět na obrázku 30. Při sestupné hraně na TCK je aktualizována hodnota na TDO. Pro TDIa TMS je vzorkována hodnota při náběžné hraně.



Obrázek 30: JTAG signál

JTAG vykonává poslední přijatou instrukci (*TAP controlerem*) dokud není přijata a zpracována nová instrukce (např. instrukce příjem a ukládání dat). Pokud se přijímá instrukce na pinu TDI, tak na pin TDO je vysílán informační stav.

Přenos dat z a od JTAGu

Přenos dat je spojen s vykonáváním instrukce. Data se posunují z a do JTAG registrů nebo do OnCE modulu. Typ (adresa registru) a délka datového registru závisí na druhu instrukce.

5.1.2. OnCE modul

OnCE modul byl navrhnut pro ladění aplikačního sotwaru. Tento blok je samostatný a přístup k němu je možný prostřenictvím pinů JTAG rozhraní. Modul OnCE umožňuje sledovat a měnit hodnoty registrů, paměti a periferií, které jsou na čipu.

Freescale má OnCE (On-Chip Emulation), který je možné spustit ve speciálním režimu pomucí speciální instrukci v TAP stavovém automatu. OnCE instrukční registr drží ladění příkazů pro operace, jako je krok programu, breakpoint a přístup k registrům a paměti.

Pro vstoupení (propojení dat) do OnCE modulu musí být TAP stavovým automatem vykonávána instrukce *ENABLE_ONCE,* tím je JTAG modul master a povoluje spojení. OnCE modul poskytuje plnou rychlost, nepřerušovanou emulaci a debug. OnCE modul má implementovaný vlastní stavový automat. OnCE modul má definovány dva typy příkazů zápis (čtení) instrukcí a dat. V debug módu je možné zapisovat a číst data na adresu registru, která je definována pro zápis (OPDBR) a pro čtení (OPGDBR).

5.2 JTAG pro External Mode

5.2.1. Porovnání komunikačních protokolů

V této kapitole budou porovnány základní vlastnosti jednotlivých komunikačních protokolů možných pro External Mode (viz tabulka 6).

Komunikační protokol	+ klady	- zápory
Sériová linka RS-232	+ jednoduchá implementace	- nízká rychlost přenosu
	+ podporována externím módem pro win32	- zatěžuje MCU
TCP/IP	+ podporován externím módem pro Win. i Linux	- nutný převodník (pro některé MCU)
		- zatěžuje MCU
CAN	+ přenosová rychlost	- zatěžuje MCU
	+ dostupný převodník (např. USB-CAN včetně ovladačů)	
	+ možné použít na stávající sběrnici	
	+ více CAN portů na MCU	
SPI	+ vysoká přenosová rychlost	- zatěžuje MCU
	+ jednotuchá implementace	- často nejsou vyvedeny piny nebo jsou obsazeny
	+ master - slave komunikace host(klient) může být Master	- nutný převodník (USB-SPI,Ethernet-SPI)
		- méně používaný než CAN
JTAG	+ nazatěžuje MCU	- složitá implementace
	+ vysoká přenosová rychlost	- nutný převodník
	+ standartní interface pro MCU	

Tabulka 6: Komunikační protokoly pro External Mode

5.2.2. Podpora JTAG/OnCE hardwaru

Existuje celá řada JTAG hardwaru pro účely, jako je výroba, testování, ladění vysokorychlostních systémů nebo vývoj mikrokontrolérů. JTAG je většinou používán vývojáři softwaru pro ladění a aktualizace firmware.

Na trhu jsou dostupné USB-JTAG/OnCE převodníky pro mikrokontrolér Freescale řady 56Fxxx. První je přímo od Freescale více v literatuře[14], ale nikde v informacích jsem nenašel, že by tento převodník byl dodáván s driverem. Druhý USB-JTAG/OnCE převodník (viz [15]) nabízí i drivery pro práci s tímto převodníkem.

Pro použití JTAGu pro External Mode je nutné vždy realizovat rozhraní mezi komunikačním protokolem z PC a JTAG/OnCE. Pro External Mode , proto uvádím následující možnosti:

- **USB-JTAG/OnCE**

- možno koupit převodník, nevýhoda vysoká cena a nejistá podpora ovladačů
- vytvořit vlastní převodník, nutnost vytvořit podporu ovladačů, nutná implementace podpory pro host

- **Eternet-JTAG/OnCE**
  - nutnost vytvořit převodník, výhoda - External Mode podporuje TCP/IP

5.2.3. Host

Pokud bude vybrána koncepce rozhraní Ethernet-JTAG/OnCE, tak stranu host External Mode nebude potřeba modifikovat. V případě koncepce USB- TAG/OnCE se bude muset vytvořit podpora komunikačního protokolu pro host. Tato změna komunikačního protokolu je popsána v kapitole 4.4 pro podporu CAN sběrnice.

5.2.4. Target

Implementace podpory na straně targetu pro sériovou linku jsou popsány v kapitole 3 nebo pro komunikaci prostřednictvím sběrnice CAN v kapitole 4.4. Důležitá změna bude v implementaci funkcí komunikační vrstvy (viz obrázek 11). Tyto funkce nebudou přistupovat k fyzickému portu, ale k paměti kam se budou ukládat rámce s daty. External Mode je založen na klient/server architektuře, kde klient je host. Tím host bude odesílat (zapisovat data do paměti) na target a zároveň číst (data z paměti), což umožuje přenos JTAG (viz obrázek 30).

Další možností je zjištění z targetu na jaké adrese jsou ukládány hodnoty proměných. Pomocí JTAG/OnCE rozhraní potom přímo přístupovat do paměti a číst přímo tyto hodnoty.

Kapitola 6

6 Závěr

Tato diplomová práce se zabývá podporou External Mode v mikrokontrolérech. External Mode se využívá při návrhu řídícího algoritmu, kdy je možné návrh řídícího algoritmu testovat na cílovém hardwaru, pro který je řídící systém vyvíjen.

Kód řídícího algoritmu je spuštěn na cílovém mikrokontroléru. Prostřednictvím komunikačního kanálu External Mode jsou vypočtená data přenášena mezi modelem v Simulinku a mikrokontrolérem v reálném čase. Díky External Mode můžeme v Simulinku tyto data zobrazovat. Z prostředí Simulinku je možné ovládat spouštění, měnit parametry bloků modelu a nastavovat parametry v modelu na mikrokontroléru a to vše za běhu vykonávání algoritmu.

V úvodu práce byly prostudovány možnosti využití External Mode pro mikrokontroléry (MCU) včetně automatického generování kódu z modelu za pomoci nástroje Real-Time Workshop (RTW). Dále byla prostudována *klient/server* architektura External Mode a komunikační protokol, který bude využit při implementaci podpory nové komunikační vrstvy.

Následně byla naportována podpora External Mode na mikrokontrolér Freescale 56F8367 prostřednictvím podporované sériové komunikace RS-232. . Tato část obsahuje změny související s použitím External Mode na 16 bitovém mikrokontroléru 56F8367. Při práci se zmiňovaným mikrokontrolérem byl použit nástroj Processor Expert, který výrazně urychluje vývoj vestavěných aplikací.

Hlavním cílem práce bylo implementovat podporu External Mode s komunikací po sběrnici CAN. Tato část obsahuje impementaci podpory komunikační vrstvy na straně host (Simulink) i target (MCU). Na straně Simulinku byl použit převodník USB-CAN prostřednictvím jeho driveru komunikujeme se sběrnicí CAN. Následně byl vytvořen MEX-soubor pro tuto komunikační vrstvu CAN prostřednictvím něhož Matlab umožňuje interakci s okolními aplikacemi a potom byla zaregistrována CAN komunikační vrstva v softwaru Simulinku. Na straně targetu byla implementována podpora komunikační vrstvy CAN sběrnice. Také byla upravena struktura rámce External Mode pro sběrnici CAN, aby nedocházelo k přebytečnému přenosu zpráv, které neobsahují užitečnou informaci. V závěru této části byly ověřeny výkonové parametry při použití sběrnice CAN oproti sériové komunikaci.

V závěru práce byly teoreticky popsány možnosti použití JTAG/OnCE pro External Mode. V této části jsou také porovnány výhody a nevýhody jednotlivých komunikačních vrstev pro External Mode.

Výsledné řešení umožňuje vygenerovat řídící algoritmus z modelu v Simulinku a spustit ho v mikrokontroléru. S použitím External Mode s komunikací po sběrnici CAN můžeme monitorovat výpočet řídícího algoritmu na MCU s periodou výpočtu modelu 700us a výsledky zobrazovat v Simulinku v reálném čase. Tuto podporu lze využít při jakémkoliv druhu simulace, při kterém je řídící algoritmus spuštěn na fyzickém hardwaru.

Použitá literatura

[1] Real-Time Workshop R User's Guide [online]. poslední revize 2009-05-08 [cit.2009- 05- 08]. Dostupné na World Wide Web: <http://www.mathworks.com/access/helpdesk/help/pdf_doc/rtw/rtw_ug.pdf>.

[2] The MathWorks, Inc. [online]. poslední revize 2009-05-08 [cit. 2009-05-08]. Dostupné na World Wide Web: <http://www.mathworks.com>.

[3] External Mode, Real-Time Workshop User's Guide [online]. poslední revize 2009-05-08 [cit. 2009-05-08]. Dostupné na World Wide Web: <http://www.mathworks.com/access/helpdesk/help/pdf_doc/rtw/rtw_ug.pdf>.

[4] The MathWorks Inc. Real-Time Workshop - Target Language Compiler . Version 6, 2005.

[5] Freescale Semiconductor [online]. poslední revize 2009-05-08 [cit. 2009- 05- 08]. Dostupné na World Wide Web: <http://www.freescale.com>.

[6] RAPAVÝ, Martin. Processor in The Loop simulace: Diplomová práce. Praha:

ČVUT, Fakulta elektrotechnická, Katedra řídící techniky, 2007. 125 l., 41 l. příl. Vedoucí diplomové práce Libor Waszniowsky.

[7] Freescale Semiconductors. MC56F8367 Evaluation Module User Manual, 10/2004, <http://www.freescale.com>

[8] Processor Expert [online]. poslední revize 2009-05-08 [cit. 2009-05-08]. Dostupné na World Wide Web: <http://www.processorexpert.com>.

[9] CAN, Controller Area Network [online]. poslední revize 2009-10-12 [cit. 2009-10-12]. Dostupné na World Wide Web: <http://fieldbus.feld.cvut.cz/can>.

[10] SYS TEC Electronics, USB-CAN modul- L-487e_21.pdf

[11] The MathWorks Inc. MATLAB External interfaces. Version 7, 2005.

[12] IEEE 1149.1, *Standard Test Access Port and Boundary-Scan Architecture* , *http://standards.ieee.org/reading/ieee/std_public/description/testtech/1149.1-1990_desc.html*

[13] Freescale Semiconductors.JTAG_OnCE AN1935, http://www.freescale.com

[14] Freescale, USB TAP [online]. poslední revize 2009-12-12 [cit. 2009-12-12]. Dostupné na World Wide Web: <http://www.freescale.com/files/soft_dev_tools/doc/user_guide/CWH-UTP-UG.pdf>

[15] Domain Technologies, Inc. *SB – USB* [online]. poslední revize 2009-12-12 [cit. 2009-12-12]. Dostupné na World Wide Web: <http://www.domaintec.com/usbemu.html>

[16] Burkhard Mann, C pro mikrokontroléry, BEN-technická literatura 2004

Příloha A

Obsah přiloženého CD

K této práci je přiloženo CD, na kterém jsou uloženy zdrojové soubory. Adresářová struktura je následující.

- ./ doc – tato diplomová práce ve formátu pdf
- ./ src
  - / serial – tento adresář obsahuje zdrojové soubory External Mode, PE projektu a demonstrační úlohy pro komunikaci v External Mode prostřednictvím sériové komunikace RS-232
  - /can -tento adresář obsahuje zdrojové soubory External Mode, PE projektu a demonstrační úlohy pro komunikaci v External Mode prostřednictvím sběrnice CAN