Propojení průmyslového PC s modely technologických procesů

Connection of industrial PC with models of technological processes

Bc. Přemysl Vrba

Diplomová práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Přemysl VRBA**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Automatické řízení a informatika**
Téma práce:	**Propojení průmyslového PC s modely technologických procesů – výuková pomůcka**

Zásady pro vypracování:

1. **Navrhněte propojení daných modelů technologických procesů s počítačem Advantech PPC-L60T s využitím modulů ADAM řady 4000 a rozhraní RS485.**
2. **Vytvořte podpůrné programové vybavení pro daný počítač v jazyce C nebo C++ umožňující tvorbu aplikací pro řízení použitých modelů.**
3. **Zpracujte několik ukázkových úloh pro řízení modelů s využitím vytvořeného programového vybavení.**
4. **Zpracujte uživatelskou dokumentaci k vytvořenému programovému vybavení.**

Rozsah práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **VLACH, Jaroslav, VLACHOVÁ, Viktorie. Počítačová rozhraní přenos dat a řídicí systémy. Praha : BEN – technická literatura, 2002. 176 s. ISBN 80-7300-010-5.**
2. **VLACH, Jaroslav. Řízení a vizualizace technologických procesů. Praha : BEN – technická literatura, 2002. 160 s. ISBN 9788086056661.**
3. **PRATA, Stephen. Mistrovství v C++. Praha : Computer press, 2007. 1120 s. ISBN 9788025117491.**
4. **PPC-L60T VIA Eden Processor based Fanless Panel PC with 6.4 TFT-LCD Users Manual, Advantech, 2005 [cit. 2010-01-18]. Dostupný z URL: http://downloadt.advantech.com/download/downloadsr.aspx?File_Id=1-1KCPDM .**
5. **ADAM 4000 Data Acquisition Modules, Users Manual [online], Advantech, 2008 [cit. 2010-01-18]. Dostupný z URL: http://downloadt.advantech.com/download/downloadsr.aspx?File_Id=GF-22KNC .**
6. **KOHOUT, Luděk. Učební pomůcky EDU-mod [online]. 2008 [cit. 2010-01-19]. Dostupný z URL: http://www.edumat.cz/produkty.php?produkt=edumod .**
7. **PINKER, J. Mikroprocesory a Mikropočítače. Praha : BEN ? technická literatura, 2004. 220 s. ISBN 80-7300-110-1.**
8. **MATOUŠEK, David. Číslicová technika. Praha: BEN ? technická literatura, 2002. 208 s. ISBN 80-7300-025-3.**

Vedoucí diplomové práce: **Ing. Jan Dolinay**
Ústav automatizace a řídicí techniky

Datum zadání diplomové práce: **19. února 2010**

Termín odevzdání diplomové práce: **8. června 2010**

Ve Zlíně dne 19. února 2010

L.S.

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

prof. Ing. Vladimír Vašek, CSc.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Náplní této práce je navržení propojení modelů technologických procesů a zařízení určených pro výuku s průmyslovým počítačem a návrh podpůrné knihovny funkcí programovacího jazyka C pro tvorbu řídicích programů pracujících s těmito modely. Propojení je realizováno pomocí v průmyslu zavedených sběrnic a zařízení, modelem technologického zařízení je model obvyklé automatické pračky s běžnými funkcemi.

Práce se v teoretické části věnuje nastínění historického pozadí a vývoje programovacího jazyka C, dělení komunikačních sběrnic a též předkládá popis některých komunikačních protokolů používaných v praxi. Praktická část je věnována popisu zařízení použitých při realizaci této práce, způsobu jejich použití a nástrojů, které byly pro splnění zadání realizovány. Popsány jsou zde také programové nástroje vytvořené pro snadné programové ovládání modelu a názorné příklady použití modelu a programových nástrojů.

Klíčová slova: C, Advantech, ADAM, knihovna funkcí, model

ABSTRACT

The aim of this thesis is design of the connection of technological process models and devices, given as teaching aids, to industrial computer and designing supporting library of function of C programming language for creating control programs working with these models. Connection is designed with buses and devices established in the industry, the model of technology equipment is a model of washing machine with ordinary features.

In theoretical part of this work there is an outline of historical background and evolution of programming C language, classification of communication buses and description of communication's protocol used in practice. Practical part of this work deals with description of devices used for realizing this work, methods of

using the devices and tools which were designed for fulfillment of the aims of this work. It also describes the programming tools created for easy program control of model and illustrative examples of using the model and program tools.

Keywords: C, Advantech, ADAM, library of functions, model

Poděkování, motto

Děkuji svému vedoucímu Ing. Janu Dolinayi za velmi cenné rady a připomínky při tvorbě této práce.

Věnováno tiché vzpomínce na rodiče.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu
  využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně …….

podpis diplomanta

OBSAH

ÚVOD

Řízení technologických procesů pomocí počítače se v dnešním světě užívá téměř na každém kroku. Počítačem řízené procesy nás obklopují ze všech stran a usnadňují nám život takovým způsobem, že je těžko představitelné co by se stalo, kdyby tohoto řízení nebylo. Tato situace klade čím dál větší nároky na odborníky schopné porozumět těmto procesům a také jejich řídicím orgánům – počítačům, jejich periferiím a jejich řídicím algoritmům. Je proto nutné takové odborníky cíleně vychovávat z řad studentů a za pomoci technických pomůcek jim umožňovat porozumět principům počítačového řízení.

Tvorba jedné takové pomůcky je náplní této práce. Úkolem je navržení propojení modelů technologických procesů a průmyslového počítače typu PC s operačním systémem Windows XP professional pomocí průmyslově používaných a standardizovaných sběrnic, komerčních komunikačních modulů a zařízení. Zároveň je účelem práce vytvořit vhodné nástroje, pomocí kterých si budou studenti moci osvojit základy programování, počítačového řízení technologických zařízení a procesů, algoritmizaci úloh. K tomuto účelu byl určen konkrétní model technologického procesu a také technická zařízení, která slouží k rozšíření průmyslového počítače o možnost řídit tento model. Vybráno bylo zařízení firmy Advantech, konkrétně modul digitálních vstupů a výstupů ADAM4050. Model technologického procesu představuje běžnou domácí automatickou pračku s obvyklými možnostmi a funkcemi vyráběnou jako výuková pomůcka především pro výuku programování PLC automatů firmou EduTech. Pro vytvoření nástrojů pro práci s modelem bylo zvoleno řešení v programovacím jazyce C. Součástí práce je též zpracování ukázkových programů, ve kterých je předvedeno využití vytvořených nástrojů a možností modelu, a technické dokumentace, která popisuje vytvořené programové nástroje. Při realizaci byly zvoleny pro řešení programové části práce nástroje z dílny výrobce operačního systému cílového počítače, firmy Microsoft, a to produkt Microsoft Visual C++ 2008 Express Edition, ve verzi vhodné pro domácí a studentské použití.

I. TEORETICKÁ ČÁST

1 PROGRAMOVACÍ JAZYK C

V úvodu této práce bych chtěl několika větami shrnout historii a původ programovacího jazyka C, který jsem využil pro tvorbu nástrojů pro ovládání zadaného modelu technologického zařízení. [5]

1.1 Historie

Programovací jazyk C vznikal v šedesátých létech minulého století jako nástroj pro přepis operačního systému UNIX pro nový typ počítače. Vzhledem k jeho přednostem se stal velmi rychle populární, byla vytvářena spousta rozšíření pro různé typy zařízení, která ale měla za následek ztrátu důležité vlastnosti jazyka – přenositelnost. Volnou formou pokračoval vývoj jazyka až do 70. let minulého století, kdy byla nakladatelstvím Pretince-Hall vydána kniha The C Programming language autorů Briana Kernighana a Dennise Ritchieho. Tato publikace se stala neoficiální specifikací jazyka. Dnes je tato verze označována jako tradiční C, nebo taky pre-ansi C.

1.2 Normalizace

Standardizace tohoto jazyka probíhala od roku 1982 až 1989, kdy byla přijata norma American National Standard X3.159-1989, známá pod názvem ANSI C. Oproti předchozí neoficiální verzi tato sjednocovala nejasné části, řešené různými kompilátory odlišně a také obsahovala nové prvky, které byly do jazyka zapracovány. Zároveň zachovávala jednoduchost a přenositelnost kódu. Tato norma byla převzata jako vzor též organizací ISO pro normu ISO/IEC 9899-1990. Dnes je možné revidovanou normu najít pod označením ISO/IEC 9899:1999/Cor 3:2007 Programming languages – C například [1].

1.3 Vývoj

Vývoj jazyka C probíhá formou nových knihoven, přepisem stávajících a jejich normativní specifikací tak, aby jej bylo možno používat na rozličných technických zařízeních a přitom byla stále zachována přenositelnost kódu a nezávislost na hardwarovém základě. Jazyk je tak možno používat jak k programování pracovních počítačů a stanic, tak k programování jednoduchých jednočipových počítačů a kontrolérů. Pro udržení kroku se současnými požadavky vznikají nové knihovny a jejich specifikace, které popisují použití moderních technologických postupů a funkcí.

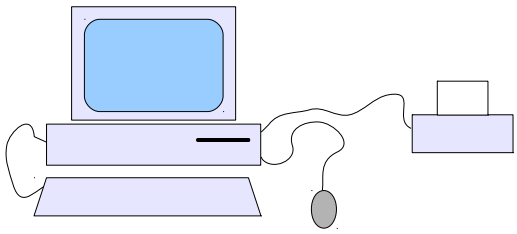
V rámci vývoje jazyka nelze nezmínit ani následovníky, kteří z tohoto jazyka vzešli. Nejdůležitějším potomkem je jistě jazyk s názvem C++, který přináší nové poznatky, postupy a programovací techniky do struktury jazyka C, který tím obohacuje o nové možnosti. Jedná se například o možnost objektového programování, přetěžování operátorů, programování na základě šablon.

2 KOMUNIKAČNÍ SBĚRNICE

Komunikační sběrnice jsou zařízení určená k předávání informací mezi různými technologickými zařízeními. Tato zařízení mohou být realizována jako určitý způsob připojení jednoho zařízení k jinému, případně může být navzájem spojeno více zařízení. Oba způsoby mají své výhody i nevýhody. Způsob, ve kterém jsou propojena pouze dvě zařízení, mezi kterými je třeba přenášet informace, je jednoduchý, ale pokud je takových zařízení více, roste složitost a množství propojujících zařízení. Připojení více zařízení k jednomu určenému k předávání informací klade nároky na schopnosti tohoto propojujícího zařízení. Konstrukčně je vždy propojující zařízení realizováno jako část zařízení, které je připojováno ke komunikačnímu systému tvořenému dvěma nebo více zařízeními vybavenými částmi stejného komunikačního zařízením. Těmto částem říkáme komunikační rozhraní. Tato rozhraní se podílí na řízení toku informací komunikačním kanálem, který tvoří. Podle schopností řídit tok informací k jednomu protějšímu zařízení či schopnostem řídit tok selektivně k jistému zařízení připojenému ke komunikačnímu systému se mohou tyto systémy dělit na dvě skupiny. Systémy, které jsou určené pro spojení jen dvou zařízení je možné zařadit do skupiny systémů s fyzickým rozlišováním, neboť je možné vybavit řídicí prvek několika takovými zařízeními schopnými přenosu informací vždy s jedním protějším zařízením. Systémy, které umožňují přenášet informace mezi konkrétními zařízeními připojenými ke komunikačnímu systému, je možné zařadit do skupiny s rozlišováním podle obsahu přenášených zpráv.

2.1 Systémy s fyzickým rozlišováním

Komunikační systémy s fyzickým rozlišováním připojených zařízení jsou vhodné, pokud je třeba připojit několik málo různých zařízení k jednomu řídicímu prvku. S výhodou může být pro tato zařízení využito různých propojení podle potřeb jednotlivých zařízení. Zařízení jsou navzájem nedostupná, mohou komunikovat pouze s řídicím prvkem, který řídí jejich činnost. Podle vlastností jednotlivých zařízení je možné, aby připojené zařízení v případě potřeby odeslalo datovou zprávu řídicímu prvku kdykoliv, bez ohledu na to, zda probíhá komunikace řídicího prvku s jiným zařízením. Přijetí a současné odesílání dvou různých zpráv je možné a záleží jen na vlastnostech řídicího prvku, jak bude tato situace vyřešena. Příkladem takového komunikačního systému je osobní počítač a jeho běžné periferie.

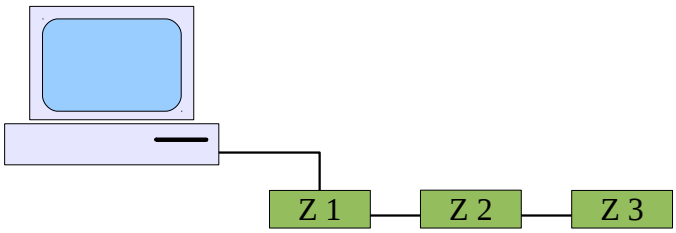


Obrázek 1: Komunikační systém s fyzickým rozlišováním - osobní počítač

2.2 Systémy s rozlišováním podle obsahu zpráv

Systémy s rozlišováním podle obsahu zpráv připojují všechna zařízení k jednomu komunikačnímu kanálu ve kterém probíhá přenos zpráv. Přenosový kanál musí mít definovaný přenosový protokol pomocí kterého lze řídit provoz a tok dat. Provoz může být řízen centrálně jedním řídicím prvkem, nebo může být neřízen. Neřízený způsob provozu pak vyžaduje definici pravidel pro zjišťování kolizí a způsob jejich řešení. Takový systém je například síť ethernet. Výhodou takového systému je schopnost provozu i když některá zařízení připojená do systému nejsou v provozu, nebo následkem závady jejich činnost selhala.

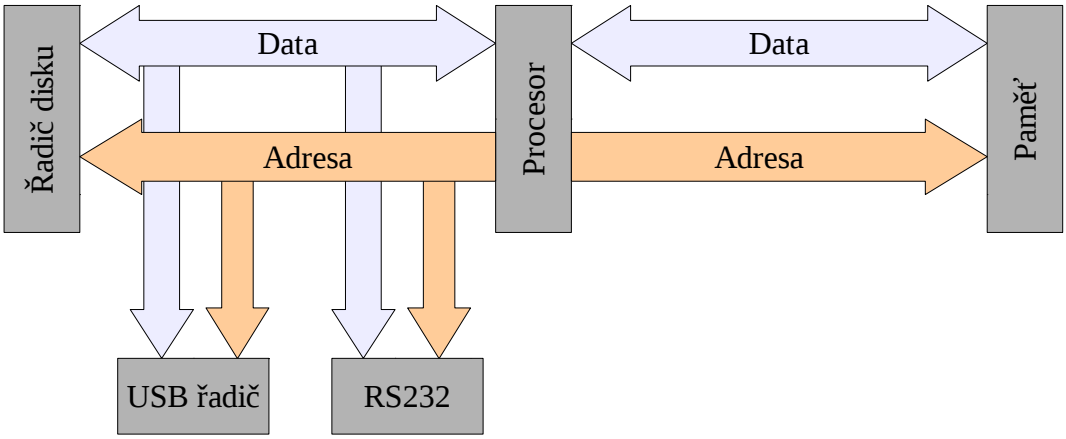
Komunikační protokoly systémů s řízeným provozem nemusí obsahovat pravidla pro detekci a nakládání s kolizemi, jsou tedy jednodušší než protokoly neřízených systémů, ale výpadek řídicího členu tento systém vyřadí z činnosti. Příkladem může být systém RS485 nebo také komunikační systém GPIB používaných v automatizovaných měřicích systémech. Výhodnou tohoto systému je lepší přehled o provozu na společném komunikačním kanálu a rychlejší odezva při komunikaci – odpadá detekce kolizních stavů.



Obrázek 2: Komunikační systém s rozlišováním podle obsahu zpráv

Jiným typem komunikačního systému s rozlišováním podle obsahu zpráv může být systém s adresovou sběrnicí. Řídicí prvek volí kódem na adresové sběrnici zařízení se kterým bude probíhat přenos dat a ostatní zařízení musí své obvody odpojit od datové sběrnice, nebo

uvést do takového stavu, aby nemohlo dojít k ovlivnění přenášených dat. Odesílání a příjem dat je možný pouze mezi jedním vybraným zařízením a řídicím prvkem. Směr přenosu dat může být volen též pomocí signálu přenášeného po zvláštním vodiči. Příkladem takového systému je komunikace procesoru s pamětí, kde jednotlivá zařízení jsou paměťové buňky, nebo komunikace s periferními obvody, ke kterým jsou připojena další zařízení. Adresní kód zařízení, kterému jsou data určena tak může být považován za část přenášených dat.



Obrázek 3: Komunikační systém s rozlišováním podle obsahu zpráv - adresní sběrnice

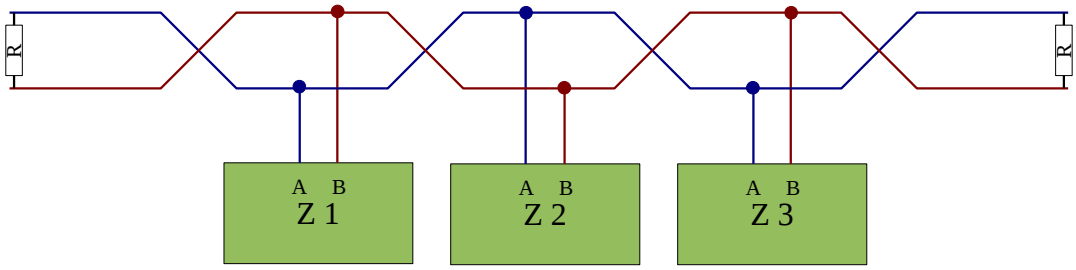
2.3 Komunikační sběrnice RS485

Provedením spadá tato sběrnice mezi systémy s rozlišením podle obsahu zpráv. Jedná se o dvouvodičovou sběrnici, která se používá k vzájemnému propojení více zařízení. Maximální počet připojených zařízení je 32. Největší vzdálenost pro tuto sběrnici je 1200 metrů. Větší vzdálenosti lze překlenout pomocí opakovačů, nebo pomocí převodu na optické vlákno. Způsob komunikace je značně závislý na použitém komunikačním protokolu. Mezi nejznámější komunikační protokoly patří CAN, PROFIBUS, MODBUS.

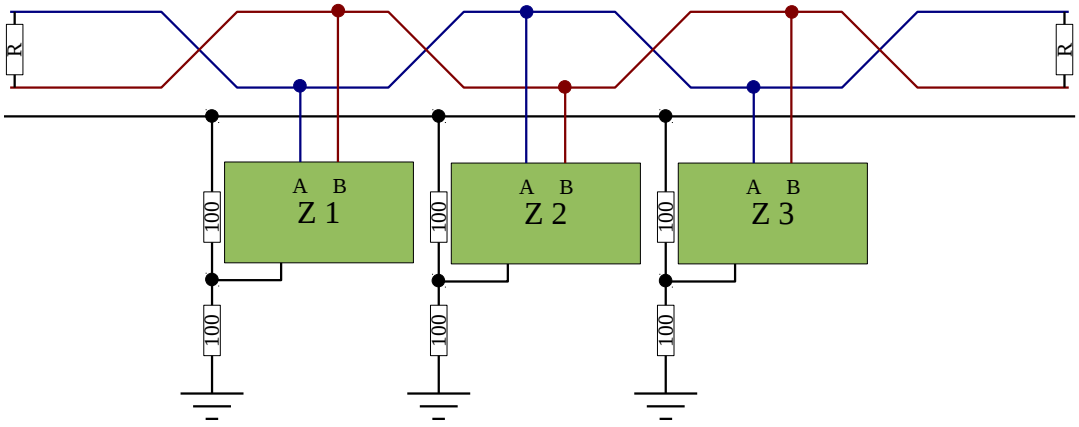
2.3.1 Způsob propojování

Jednotlivá zařízení se připojují mezi vodiče sériově ve formě takzvaných uzlů. Při tom je třeba dbát na označení vodičů, vzhledem ke způsobu detekce logických úrovní není možné připojit zařízení opačně. Pro vedení je doporučována kroucená dvoulinka s charakteristickou impedancí 120 Ω. Pro správný provoz je třeba zajistit zakončení obou konců sběrnice rezistory o hodnotě 120 Ω, čímž se potlačí nežádoucím odrazům signálu na konci vedení.[6] Kroucená dvoulinka též značně omezuje vlivy rušivých signálů a polí na komunikační signál. Není proto doporučováno použití přímých dvoulinek pro vedení na

větší vzdálenosti. I když je sběrnice dvouvodičová, pro správnou detekci signálů je doporučováno použití třetího společného vodiče, například zemnění či společný napájecí vodič. Pokud je použit samostatný třetí vodič, měly by být do série s ním zapojeny odpory o hodnotě 100 Ω, aby se zabránilo nežádoucím proudům vznikajícím při rozdílném zemním potenciálu. [7]



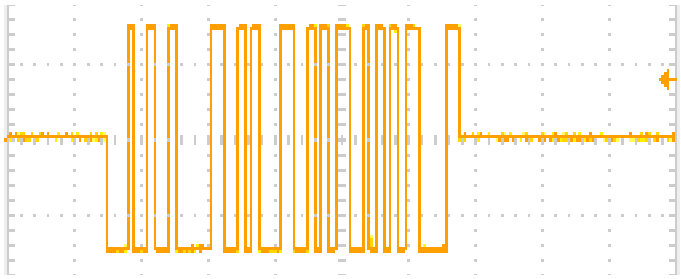
Obrázek 4: Zapojení sběrnice RS485 bez třetího vodiče



Obrázek 5: Zapojení sběrnice RS485 s využitím třetího vodiče

2.3.2 Přenos dat

Pro přenos dat je použito diferenčního způsobu, na rozdíl od sběrnice RS232 či GPIB. Jsou definovány úrovně rozdílového napětí na vodičích. Pokud je napětí na vodiči, který je označován jako B, případně +, větší jak 200 mV, je detekována logická úroveň, pokud je napětí menší, je detekována opačná logická úroveň. Vysílače běžně používají napětí od 2 do 7 V.



Obrázek 6: Záznam zprávy přenášené po RS485

3 KOMUNIKAČNÍ PROTOKOLY

V této části jsou srovnány vlastnosti dvou komunikačních protokolů. Popsány jsou protokoly ADAM a Modbus, které se dají využít pro ovládání modulů řady ADAM4000. Protokol ADAM je podporován všemi moduly, zatímco protokol Modbus podporují jen některé. Ozřejmeny jsou jejich základní formáty přenášených zpráv a možnosti využití.

3.1 Komunikační protokol ADAM

Tato část se zabývá vlastnostmi komunikačního protokolu ADAM, který je použit pro přenos informací mezi převodníky a moduly řady ADAM4000 výrobce Advantech.

Protokol ADAM je čistě textový protokol typu master – slave, kterým lze přenášet data mezi moduly a řídicím členem. Je vytvořen tak, aby jeho použití bylo možné i na komunikačních systémem s rozlišením podle obsahu přenášených zpráv. Pro řízení různých modulů používá protokol jednotnou základní sadu povelů, která je pro jednotlivé typy modulů rozšířena o další povely. Základní sada povelů se dá charakterizovat jako sada systémových povelů pro identifikaci a nastavování modulů, rozšířená sada pak obsahuje povely pro řízení specifických částí modulu. Každá zpráva je ukončena znakem s kódem 13, v syntaxi jazyka C se jedná o znak '\r'.

3.1.1 Ukázka povelů a zpráv protokolu ADAM

První ukázka povelu odesílaného modulu ADAM je povel pro odeslání konfiguračního řetězce. V návodu k použití [4] je uvedena tato syntaxe:

```
$AA2(cr)
```

Význam jednotlivých znaků je tento:

- $: Úvodní znak.
- AA: Dva znaky vyjadřující adresu zařízení v šestnáctkové soustavě, kterému povel patří.
- 2: Povel.
- (cr): Ukončující znak s kódem 13. V tabulce znaků ascii je označen názvem „Carriage return“.

Pokud bude povel odesílán zařízení s adresou 1, bude odeslána následující sekvence znaků:

```
$012(cr)
```

Po přijetí tohoto povelu je tázaným zařízením odeslána zpráva ve formátu

```
!AATTCCFF(cr)
```

nebo

```
?AA(cr)
```

Význam jednotlivých znaků první zprávy je tento:

- !: Úvodní znak s významem přijetí platného povelu.
- AA: Dva znaky vyjadřující adresu zařízení v šestnáctkové soustavě, které zprávu odeslalo.
- TT: Dva znaky určující typ kódu – různé moduly zde uvádí různé specifické informace.
- CC: Kód nastavené komunikační rychlosti.
- FF: Různé příznaky zařízení.
- (cr): Ukončující znak s kódem 13. V tabulce znaků ascii je označen názvem „Carriage return“.

Význam znaků druhé zprávy, kterou je možno od zařízení dostat je tento:

- ?:Úvodní znak s významem přijetí neplatného povelu.
- AA: Dva znaky vyjadřující adresu zařízení v šestnáctkové soustavě, které zprávu odeslalo.
- (cr): Ukončující znak s kódem 13. V tabulce znaků ascii je označen názvem „Carriage return“.

Ukázka druhého povelu, který je možné považovat za základní, je povel pro odeslání typu modulu. Syntaxe tohoto příkazu je:

```
$AAM(cr)
```

Význam jednotlivých znaků v povelu je:

- $ Úvodní znak.
- AA: Dva znaky vyjadřující adresu zařízení v šestnáctkové soustavě, kterému povel patří.
- M: Povel.

- (cr): Ukončující znak s kódem 13. V tabulce znaků ascii je označen názvem „Carriage return“.

Odpověď na tento povel má formát:

```
!AA(Module Name)(cr)
```

Význam jednotlivých znaků zprávy je takovýto:

- !: Úvodní znak s významem přijetí platného povelu.
- AA: Dva znaky vyjadřující adresu zařízení v šestnáctkové soustavě, které zprávu odeslalo.
- (Module Name): Jméno modulu, který zprávu odeslal. Například 4050 pro modul ADAM4050.
- (cr): Ukončující znak s kódem 13. V tabulce znaků ascii je označen názvem „Carriage return“.

Dalším povelem, který je zde ukázán, je povel specifický pro moduly s digitálními vstupy a výstupy, například modul ADAM4050. Jedná se o povel, kterým se cílové zařízení přiměje odeslat aktuální hodnoty registrů pro vstupní a výstupní linky modulu. Syntaxe povelu je:

```
$AA6(cr)
```

Význam jednotlivých znaků povelu je:

- $: Úvodní znak.
- AA: Dva znaky vyjadřující adresu zařízení v šestnáctkové soustavě, které zprávu odeslalo.
- 6: Povel.
- (cr): Ukončující znak s kódem 13. V tabulce znaků ascii je označen názvem „Carriage return“.

Odpověď na tento povel má, za předpokladu, že odpovídající zařízení je typu 4050, tvar:

```
!(dataOutput)(dataInput)00(cr)
```

Význam jednotlivých znaků zprávy je tento:

- !: Úvodní znak s významem přijetí platného povelu.
- (dataOutput): Dva znaky vyjadřující hodnotu výstupního registru v šestnáctkové soustavě.

- (dataInput): Dva znaky vyjadřující hodnotu vstupního registru v šestnáctkové soustavě.
- 00: Znaky nemají žádný význam.
- (cr): Ukončující znak s kódem 13. V tabulce znaků ascii je označen názvem „Carriage return“.

Z uvedených příkladů je vidět, že všechny hodnoty ve zprávách jsou převedeny do textové podoby. Pokud je odesílána číselná hodnota osmibitového registru, je převedena do šestnáctkové soustavy a odeslána jako dvouznakový řetězec. Stejným způsobem jsou kódovány všechny adresy ve zprávách.

3.2 Komunikační protokol Modbus

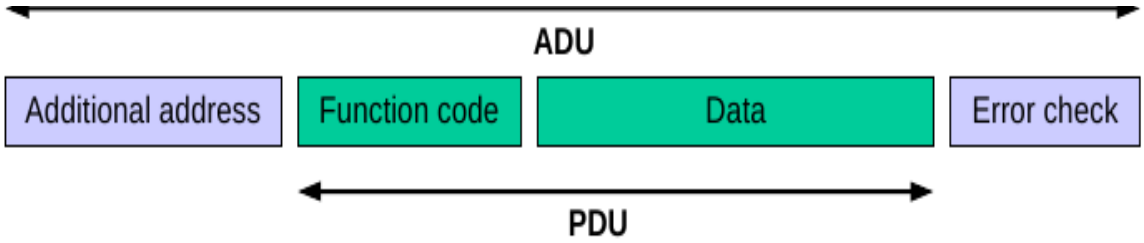
Komunikační protokol Modbus je otevřeným standardem pro realizaci dálkového řízení přístrojů a zařízení. Záměrem při vytváření tohoto protokolu je tvorba jednotného komunikačního protokolu použitelného v různých typech sítí s různými zařízeními, ale s jednotným komunikačním protokolem. Spravován je organizací Modbus Organization, která si vyhrazuje výhradní právo uvádět změny do tohoto standardu.

Z hlediska provozu je protokol Modbus řazen do kategorie server – client protokolů. To znamená, že v síti s komunikačním protokolem Modbus je řídicí člen, který organizuje předávání zpráv jednotlivým zařízením a také od jednotlivých zařízení zprávy přijímá. Protokol je navrhován tak, aby byl použitelný v co nejrozmanitějších typech sítí, případně aby sítě s protokolem Modbus byly navzájem propojitelné. Propojení různých sítí se pak děje pomocí k tomuto účelu sestavených bran. Díky tomu je možné provozovat jednou logickou síť Modbus na několika vzájemně neslučitelných fyzických sítích aniž by toto mělo vliv na chod sítě či vybavenost řídicího členu. Sítě s protokolem modbus je možné provozovat jak na dvouvodičových průmyslových sběrnicích, například RS485, tak na sítí ethernet či bezdrátové síti [3].

3.2.1 Formát přenášených zpráv

Přenášené zprávy jsou protokolem Modbus děleny na dvě hlavní části. Tyto části jsou označeny kódovými názvy PDU a ADU. Část PDU (Protokol Data Unit) je nezávislá na typu sběrnice, pomocí které je přenos dat uskutečňován a má pevný formát. Část ADU

(Application Data Unit) je závislá na typu sběrnice pomocí které se přenos dat uskutečňuje. Část ADU ve svém těle zahrnuje celou část PDU. Strukturu zprávy protokolu Modbus osvětluje následující obrázek, který byl převzat z [3].



Obrázek 7: Struktura přenášené zprávy protokolu Modbus

Jak je z obrázku zřejmé, je část PDU dělena na dvě sekce. Sekci Function code a Data.

Část function code je velká jeden bajt a obsahuje kód akce, která je po adresovaném zařízení požadována. Hodnoty jsou omezeny rozsahem 1 ÷ 127. V tomto rozsahu jsou specifikací protokolu definovány kódy akcí, které je možné pomocí tohoto protokolu vyžadovat. Část data obsahuje podrobnější informace o akci, která se má provést. Její velikost je shora omezena největší možnou délkou části PDU. Ta je 253 bajtů bez ohledu na typ sběrnice, kterou je zpráva přenášena. Dolní mez není pro tuto část není – část může být i prázdná.

Odpovědi adresovaných zařízení mají též definován předepsaný tvar. Struktura této odpovědi je opět dána blokem PDU. Pokud je zpráva pro adresované zařízení platná, dojde k odeslání odpovědi na požadavek. Tato odpověď má definovánu tuto strukturu:

- 1 bajt kód funkce – stejný jako v požadavku
- n bajtů informace závislé na požadované akci.

Délka zprávy je opět omezena maximální délkou bloku PDU.

Pokud je požadavek pro zařízení neplatný, nebo špatně formulován, je zařízením odeslána chybová zpráva. Ta má tento tvar:

- 1 bajt chybového kódu – kód žádané akce zvýšen o 128
- 1 bajt kódu chyby

Hodnoty položky kód chyby jsou protokolem definovány a je možné je nalézt ve specifikaci protokolu.

3.2.2 Ukázka povelu a zprávy protokolu Modbus

Pro ilustraci obsahu přenášených zpráv pomocí protokolu Modbus je zde uvedeno načtení hodnot prvních 10 bitových výstupů zařízení.

3.2.2.1 Požadavek odeslaný tázanému zařízení

Část zprávy	Velikost	Hodnota (hex)
Funkční kód	1 bajt	01
Počátek čtení	2 bajty	00
		00
Počet čtených hodnot	2 bajty	00
		0A

Tabulka 1: Obsah žádosti v protokolu Modbus - příklad

3.2.2.2 Odpověď zařízení na dotaz s žádaným obsahem

Část zprávy	Velikost	Hodnota (hex)
Funkční kód	1 bajt	01
Počet bajtů s informací	1 bajt	02
Stav bitů 0 ÷ 7	1 bajt	FF
Stav bitů 8 ÷ 9	1 bajt	03

Tabulka 2: Obsah odpovědi v protokolu Modbus - příklad

Odpověď vždy obsahuje kompletní žádanou informaci. V tomto případě bylo žádáno odeslání stavů deseti výstupních míst – bitů. Tato informace se ale nevleze do jednoho bajtu, musela tedy být rozdělena na dva bajty. První bajt tedy obsahuje hodnoty výstupů 0 ÷ 7. Pořadí je dáno tak, že nejméně významný bit v bajtu odpovídá výstupnímu místu s nejnižším číslem. Výstupní místa jsou v bajtu naskládána v pořadí 7, 6, 5, 4, 3, 2, 1, 0. Druhý bajt přenáší informaci pouze od dvou výstupních míst. Těmto budou odpovídat dva nejméně důležité bity v odesílaném bajtu. Zbytek bitů v bajtu bude nastaven na 0.

3.2.2.3 Obsah chybové odpovědi

Pokud není možné žádost vyřídit, bude tázaným zařízení odeslána zpráva s následujícím obsahem.

Část zprávy	Velikost	Hodnota (hex)
Funkční kód	1 bajt	81
Kód chyby	1 bajt	01

Tabulka 3: Obsah chybové odpovědi v protokolu Modbus - příklad

II. PRAKTICKÁ ČÁST

4 POUŽITÉ TECHNICKÉ PROSTŘEDKY

Zařízení, které má být použito pro výuku a které je potřeba vzájemně propojit sestává z panelového počítače PPC-L60T firmy Advantech s operačním systémem Windows XP professional, vstupně-výstupního modulu ADAM4050, převodníku RS 232 na RS 422/485 ADAM4520 též od firmy Advantech a modelu technologického zařízení.

4.1 Panelový počítač PPC-L60T

Jedná se o počítač typu PC s dotykovým displejem realizovaným jako jednodílné zařízení. Svým konstrukčním provedením je předurčen pro zabudování jako zobrazovací a ovládací prvek pro technologické zařízení.

Specifikace produktu: procesor: Via Eden, 400 MHz, pracovní paměť: 128 MB, pevný disk: 2,5", 80 GB, 6,4" TFT LCD displej s dotykovým panelem, 1×RS232 port, 1×RS232/RS422/RS485 port, 1×PS/2 port kombinace myši a klávesnice, 1×USB1.1 port, 1×VGA port, 1×síťové rozhraní Ethernet 10/100Base-T, Napájení 24 V_{DC}.

Obrázek 8: Panelový počítač Advantech PPC-L60T

4.2 Převodník ADAM4520

Tento převodník slouží k propojení dvou vzájemně nesourodých počítačových sítí. Svým mechanickým provedením umožňuje montáž na lištu do rozváděcí skříně, provedení vstupně – výstupních míst svorkovnicí se šroubky umožňuje jednoduchou montáž kabelů. Propojení RS232 je provedeno standardním kabelem s devítipinovým konektorem canon. Pro komunikující zařízení ze dvou různých sítí je zcela transparentní.

Specifikace produktu: Vstup: RS 232 (čtyřvodičově), Výstup: RS 485 (dvouvodičově) nebo RS-422 (čtyřvodičově), Podporované komunikační rychlosti (bps): 1200; 2400; 4800; 9600;19,2 k; 38,4 k; 57,6 k, 115,2 k; Napájení 10 ÷ 30 V_{DC}.

Obrázek 9: Převodník RS232/RS485 ADAM4520

4.3 Modul ADAM4050

Tento modul slouží pro realizaci číslicových vstupů a výstupů. Konstrukční provedení umožňuje montáž na lištu rozváděče, přípojná místa modulu jsou realizována svorkovnicemi se šroubky. Digitální vstupy jsou realizovány s přídržným rezistorem o hodnotě 10 kΩ k napětí 5 V, čímž je zajištěna úroveň logická 1, která je rozpoznávána od napětí 3,5 V. Maximální hodnota napětí na vstupu je 30 V. Tato vlastnost modulu umožňuje fungovat i v obvodech s 24V logikou aniž by bylo nutné obvod nějak chránit.

Digitální výstupy modulu jsou realizovány jako tranzistory s otevřeným kolektorem. Maximální napětí, kterému může být výstup vystaven v uzavřeném režimu je 30 V. Maximální proud otevřeným tranzistorem je 30 mA. Vstup je opět možné použít v obvodech s 24V logikou bez zvláštních úprav.

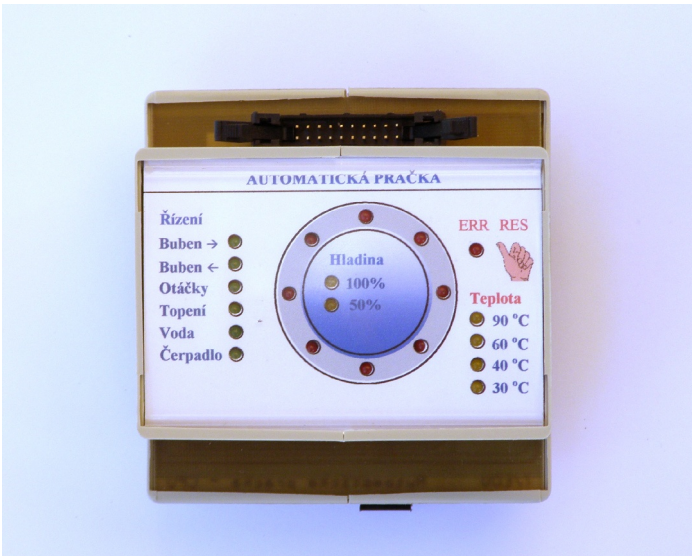
Všechny vstupy a výstupy jsou vztaženy k zápornému pólu napájecího napětí.

Specifikace produktu: Počet digitálních vstupů: 7, Počet digitálních výstupů: 8, Napájecí napětí: 10 ÷ 30 V_{DC}.

Obrázek 10: Modul ADAM4050

4.4 Model pračky

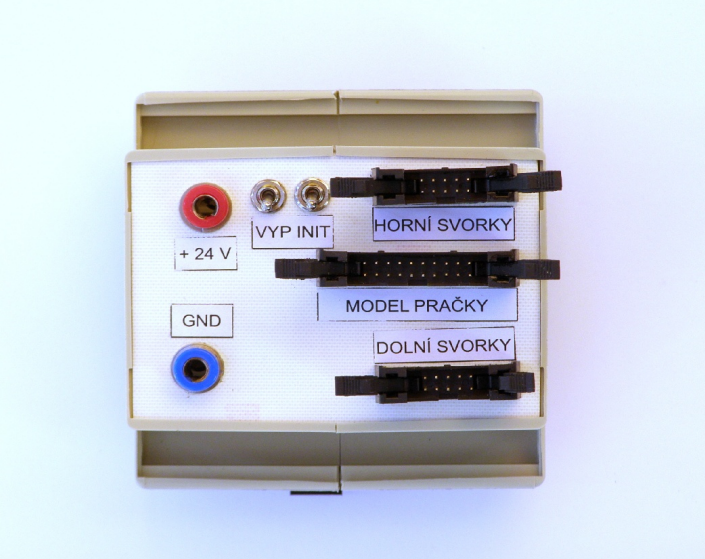
Jedná se o procesorem řízené zařízení v úpravě pro 24V logiku. Mechanické provedení umožňuje montáž na lištu do rozváděče, propojení je provedeno pomocí dvacetižilového plochého kabelu. Vstupy do zařízení jsou realizovány jako pasivní, pro vybuzení do aktivní úrovně je nutno jim dodat energii od ovládacího zařízení. Výstupy jsou realizovány jako tranzistory s otevřeným emitorem s rezistorem o hodnotě 750 Ω mezi kolektorem a napájecím napětím. Tím je umožněno výstupním signálem ovládat připojené pasivní obvody a zařízení hlídající stav modelu. Stav tranzistoru je ovládán světelně – jedná se o část optického oddělovače.



Obrázek 11: Model pračky

4.5 Propojovací blok

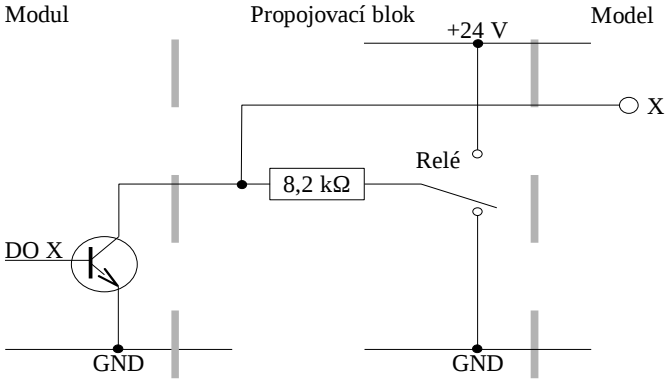
Jedná se o zařízení, které přizpůsobuje parametry ovládacích signálů vycházejících ze vstupně – výstupního modulu ADAM4050 tak, aby model pračky byl schopen jejich stav bezpečně rozeznat, a též upravuje parametry výstupních signálů z modelu takovým způsobem, že je jejich stav zjistitelný vstupními místy modulu ADAM4050. Zároveň slouží pro distribuci napájecího napětí pro model pračky i vstupně – výstupního modulu a převodník. Tím se značně zpřehledňuje propojování jednotlivých částí. Připojení modelu a vstupně – výstupního modulu se provádí plochými kabely, napájecí napětí je přivedeno pomocí kablíků s banánky. Blok obsahuje též dva vypínače, jeden slouží k zapnutí připojených zařízení a druhý pro aktivaci režimu INIT vstupně – výstupního modulu. Aby nebyl po zapnutí napájecího napětí (vypínače) model pračky vystaven matoucímu stavu ovládací povelů vzešlých ze základního stavu výstupů modulu ADAM4050, je propojovací blok vybaven relé, které upravuje vlastnosti některých částí propojovacího bloku tak, že k tomuto nežádoucímu stavu nedochází. Zároveň je tím i blokováno ovládání modelu pračky. Aktivace relé, které je ovládáno jedním výstupem modulu, způsobí přepnutí funkce některých částí propojovacího bloku a tím se povolí ovládání modelu. Signály, které vycházejí z modelu nejsou tímto relé nijak ovlivněny a jejich stav je možno zjistit i bez aktivace relé.



Obrázek 12: Propojovací blok

4.5.1 Zapojení signálů v propojovacím bloku

Signály vystupující ze vstupně výstupního bloku jsou vedeny paralelně na rezistory o hodnotě 8200 Ω a na příslušný ovládací vstup modelu. Rezistory jsou svými opačnými póly



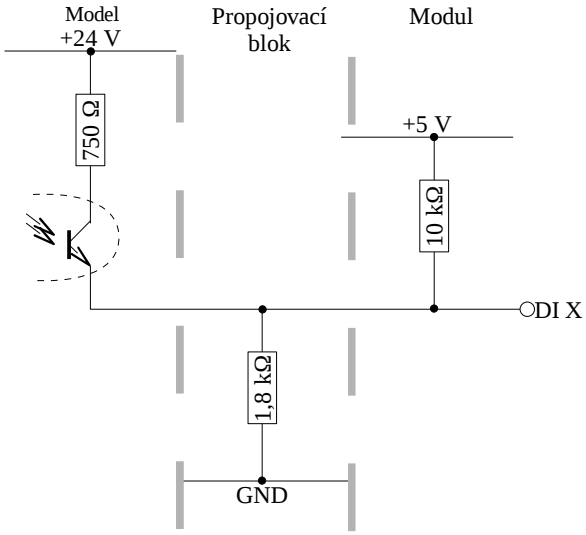
Obrázek 13: Schématické znázornění zapojení ovládacího signálu

spojeny do jednoho bodu a ke kontaktu relé. Relé ve svém základním stavu připojuje tento bod k zápornému pólu napájecího napětí. Vzhledem ke konstrukci výstupů vstupně výstupního modulu není změnou stavu výstupu ovlivněna úroveň vstupu modelu technologického procesu. Přepnutím relé se společný bod rezistorů připojí ke kladnému pólu napájecího napětí. Pokud jsou výstupy vstupně výstupního modulu ve svém základním stavu – zavřené výstupní tranzistory – je na vstupních místech modelu technologického procesu detekována aktivní úroveň. Otevřením tranzistorů výstupů modulu dojde k poklesu tohoto napětí na hodnotu detekovanou jako neaktivní úroveň. Jelikož otevření tranzistorů výstupů modulu je zajištěno vložením logické 1 na příslušné místo v registru modulu, je výsledným efektem propojovacího modulu negace logické úrovně. Na tuto skutečnost je třeba dbát, pokud bude uživatel chtít tento propojovací blok používat společně s nižší úrovní knihovny, zaměřenou na práci se vstupně výstupním modulem.

Ovládací cívka relé je připojena jedním pólem k napájecímu napětí a druhým pólem k výstupnímu místu DO 6 vstupně výstupního modulu. Výstupní místa modulu jsou realizována jako tranzistory s otevřeným kolektorem. Uvedením tohoto výstupního místa do aktivního stavu tento výstupní tranzistor otevře a cívkou začne procházet elektrický proud. Tím dojde k přepnutí stavu relé. Aby nedošlo k poškození výstupního tranzistoru přepěťovými špičkami při přepínání tohoto relé, je k této cívce paralelně připojena ochranná dioda orientována v závěrném směru pro ovládací napětí cívky. Na rozdíl od ovládacích signálů určených pro model technologického zařízení není stav výstupního místa modulu

ovládajícího stav relé negován. Relé je sepnuto při vložení logické 1 na příslušnou pozici registru výstupních míst modulu. Proud procházející cívkou relé je při napětí 24 V přibližně 8 mA. Vzhledem ke schopnosti výstupního tranzistoru modulu převést až 30 mA, je v napájecím proudu relé dostatečná rezerva a k poškození výstupního tranzistoru přetížením by nemělo dojít.

Výstupní místa modelu jsou realizována s tranzistory s otevřeným emitorem. Kolektory těchto tranzistorů jsou připojeny přes rezistory o hodnotě 750 Ω ke kladnému pólu napájecího napětí. Tím je umožněno přímé ovládání pasívních prvků, ale pro přímé připojení k vstupním místem vstupně výstupního modulu je tato konstrukce nevhodná. Propojovací blok proto tyto signály vede přes rezistory o hodnotě 1800 Ω k zápornému pólu napájecího napětí. Tím je při uzavřeném výstupním tranzistoru modelu na vstupním místu modulu připojen napěťový dělič s rezistorem 10 kΩ, který je zabudován uvnitř modulu jako přídržný „pull up“ rezistor, a rezistorem 1800 Ω. Tento dělič je napájen napětím 5 V z modulu, takže napěťová úroveň na vstupním místě je přibližně 0,76 V, což je vstupním místem modulu ještě stále považováno za neaktivní logickou úroveň. Otevřením výstupního tranzistoru modelu aktivací stavového čidla dojde k připojení kladného pólu napájecího napětí do uzlu tohoto děliče a tím se napěťová úroveň vstupního místa modulu zvýší na úroveň 16,1 V. Měřením bylo zjištěno, že tato úroveň je přibližně 15,5 V. To je stále v mezích vstupního místa modulu, které je schopno pracovat se vstupním napětím signálu až 30 V.



Obrázek 14: Schématické znázornění zapojení stavového signálu

4.6 Modul ADAM4017

Tento modul slouží pro realizaci až osmi analogových vstupů. Konstrukční provedení umožňuje montáž na lištu rozváděče, přípojná místa modulu jsou realizována svorkovnicemi se šroubky. Analogové vstupy jsou navzájem odděleny, vyjma vstupů 6 a 7, které mají společný záporný pól. Rozsah všech vstupů je nastavován jednotně. Proudový rozsah vyžaduje externí rezistor 125 Ω.

Specifikace produktu: Počet analogových vstupů: 8, Vstupní rozsahy: 10 V; 5 V, 1 V, 500 mV, 150 mV, 20 mA, Vstupní impedance: 20 MΩ, Převodník: 16 bitů, Přesnost převodu napětí: 0,1 %.

Obrázek 15: Modul ADAM4017

5 PROPOJENÍ ČÁSTÍ

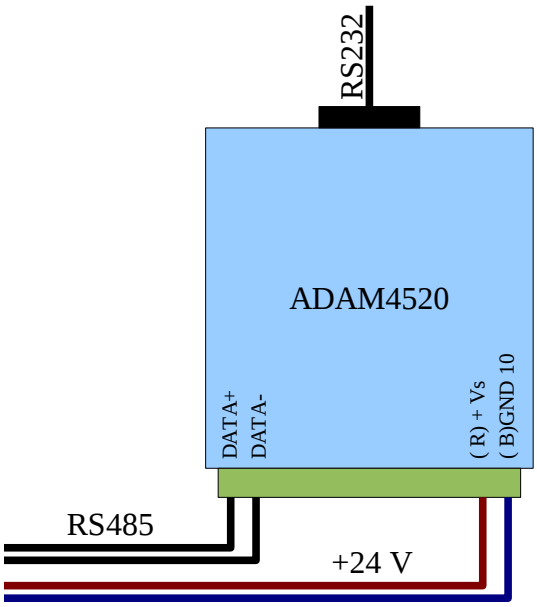
Připojení modelu automatické pračky k počítači tak, aby bylo možno využít služeb knihovny, která byla vytvořena v rámci této práce, musí být provedeno tak, jak je zmíněno v této kapitole. Jediná výjimka v této části se týká převodníku ADAM4520, který nemusí být použit v případě, že je model připojován k již existující komunikační sběrnici typu RS485. Ta však musí být v počítači, který je uvažován jako řídicí jednotka k modelu, přístupná pomocí sériového komunikačního rozhraní. V operačním systému Windows jsou tato rozhraní označována jako porty COMx, kde x je celé číslo vyjadřující pořadové číslo komunikačního rozhraní v operačním systému. Propojení jednotlivých částí je tedy následující.

5.1 Převodník ADAM4520

Tento převodník se k počítači připojuje pomocí sériového komunikačního rozhraní RS232 v dnes obvyklé devítipinové verzi. Kabel, potřebný pro propojení převodníku a počítače, musí být zapojen způsobem pin to pin, neboli jako klasická prodlužovací šňůra. Tomu také odpovídá typ konektorů. Kabel musí mít z jedné strany konektor s dutinkami, z druhé strany musí být konektor vybaven špičkami. K počítači a převodníku se kabel připojuje zasunutím do již připravených rozhraní a zajištěním šroubky, které jsou na konektorech kabelu k tomuto účelu připraveny.

Připojení převodníku na sběrnici RS485 se provede pomocí svorkovnice, na kterou jsou vyvedena všechna ostatní vedení. Sběrnice RS485 zaujímá první dvě pozice zleva při čelním pohledu na převodník a při orientaci svorkovnice směrem dolů. Jedná se o svorky DATA+ a DATA-. Pokud je tento převodník považován za koncové zařízení, mělo by být vedení komunikační sběrnice zakončeno rezistorem s hodnotou 120 Ω. Vedení pro tuto sběrnici, které je dodáno jako součást této práce je tímto rezistorem již vybaveno.

Napájení převodníku musí být realizováno externím zdrojem vhodného stejnosměrného napětí. Toto napájecí napětí se připojuje na dvě svorky úplně vpravo na připojovací svorkovnici převodníku. Svorky jsou označeny jako (R)+Vs a (B)GND 10. Napájení je možné přivést též z modulu ADAM4050 připojením k jeho napájecím svorkám spolu s vodiči kabelu připojujícím tento modul k propojovacímu bloku.



Obrázek 16: Zapojení převodníku ADAM4520

5.2 Modul ADAM4050

Připojení modulu se provádí pomocí dvou svorkovnic se šrouby. Svorkovnice jsou umístěny v horní a dolní části modulu. K datovému vedení se modul připojuje samostatným vedením komunikační sběrnice RS485 ke svorkám (Y) DATA+ a (G) DATA-, které jsou k dispozici na dolní svorkovnici v pořadí 3. a 4. svorka zprava. Pokud je modul použit jako koncové zařízení na komunikační sběrnici, mělo by být vedení této sběrnice zakončeno rezistorem o hodnotě 120 Ω. Vedení pro tuto sběrnici, které je dodáno jako součást této práce je tímto rezistorem již vybaveno.

Napájecí napětí je k tomuto modulu přivedeno pomocí propojovacího kabelu z propojovacího bloku. Pokud je potřeba přivést toto napětí též pro převodník ADAM4520, je toto napájecí napětí k dispozici na první a druhé svorce dolní svorkovnice. Označení svorek je (R)+Vs a (B)GND 10.

Všechny zbývající svorky se připojují pomocí dvou plochých kabelů k propojovacímu bloku. Signály přístupné na horní svorkovnici se připojují pomocí kompletního propojovacího kabelu ke konektoru na propojovacím bloku umístěném nejvýše při čelním pohledu tak, aby červená svorka napájecího napětí byla nahoře. Signály přístupné pomocí dolní svorkovnice a napájecí napětí modulu se připojuje pomocí druhého plochého kabelu,

který má ustřiženy dva vodiče. V propojovacím bloku se tento kabel připojuje do spodního konektoru.

5.2.1 Význam signálů horní svorkovnice

V následující tabulce je uvedeno propojení jednotlivých signálů horní svorkovnice modulu ADAM4050, význam signálu pro model automatické pračky, jeho směr z pohledu modulu, číslo pinu na příslušném konektoru propojovacího bloku a barva vodiče v kabelu.

Označení ADAM4050	Směr	Význam	Číslo pinu	Barva
DI6	Vstup	Nezapojen	1	Hnědá
DI5	Vstup	Hladina 50 %	2	Červená
DI4	Vstup	Hladina 100 %	3	Oranžová
DI3	Vstup	Teplota 90°C	4	Žlutá
DI2	Vstup	Teplota 60°C	5	Zelená
DI1	Vstup	Teplota 40°C	6	Modrá
DI0	Vstup	Teplota 30°C	7	Fialová
DO0	Výstup	Buben vpravo	8	Šedá
DO1	Výstup	Buben vlevo	9	Bílá
DO2	Výstup	Zvýšené otáčky	10	Černá

Tabulka 4: Význam a propojení signálů horní svorkovnice modulu ADAM4050

Vodič pro nezapojený signál DI6 byl ponechán pro možné využití s jiným modelem technologického zařízení či modelu, které by mohlo vyžadovat více vstupních míst z hlediska modulu ADAM4050. Jeho zapojení uvnitř propojovacího bloku ale není realizováno a propojovací blok tak bude muset být pro jiný model upraven.

5.2.2 Význam signálů dolní svorkovnice

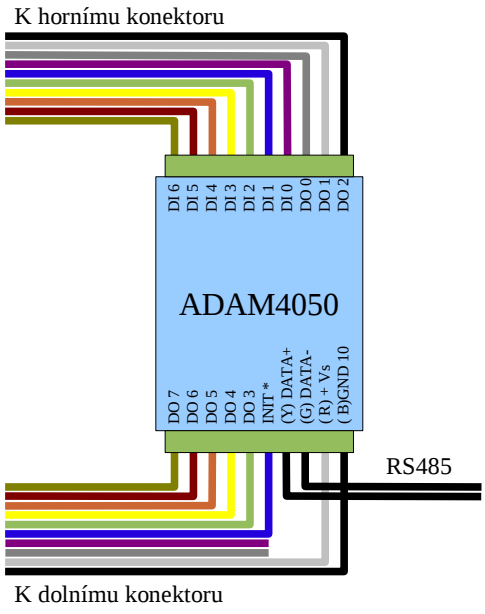
V následující tabulce je uvedeno propojení jednotlivých signálů dolní svorkovnice modulu ADAM4050, význam signálu pro model automatické pračky, jeho směr z pohledu modulu, číslo pinu na příslušném konektoru propojovacího bloku a barva vodiče v kabelu.

Označení ADAM4050	Směr	Význam	Číslo pinu	Barva
DO7	Výstup	Nezapojen	1	Hnědá
DO6	Výstup	Ovládání relé	2	Červená
DO5	Výstup	Vypouštění	3	Oranžová
DO4	Výstup	Napouštění	4	Žlutá
DO3	Výstup	Topení	5	Zelená
INIT*	Vstup	Inicializace	6	Modrá
(Y) DATA+	-	RS485	7	Fialová (vysřižen)
(G) DATA-	-	RS485	8	Šedá (vysřižen)
(R) +Vs	Vstup	Kladný pól napájení	9	Bílá
(B)GND 10	Vstup	Záporný pó napájení	10	Černá

Tabulka 5: Význam a propojení signálů dolní svorkovnice modulu ADAM4050

Vodič pro nezapojený signál DO7 byl ponechán pro možné využití s jiným modelem technologického zařízení či modelu, které by mohlo vyžadovat více výstupních míst z hlediska modulu ADAM4050. Jeho zapojení uvnitř propojovacího bloku ale není realizováno a propojovací blok tak bude muset být pro jiný model upraven.

5.2.3 Nákres zapojení



Obrázek 17: Zapojení modulu ADAM4050

5.3 Propojovací blok

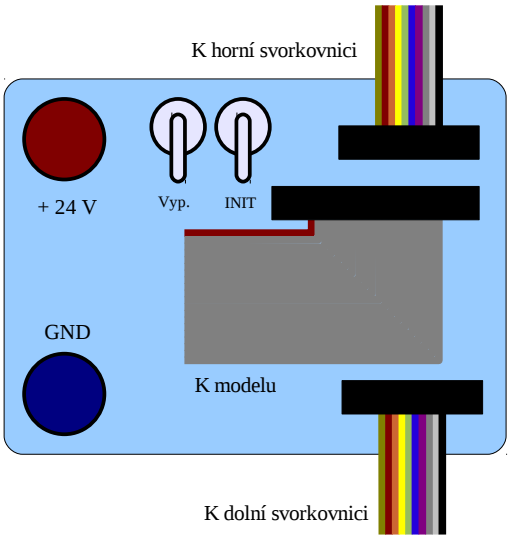
Propojovací blok se zapojuje mezi modul ADAM4050 a model technologického zařízení. Propojení je provedeno pomocí plochých kabelů. Kabely vedoucí od modulu ADAM4050 jsou dva desetižilové, kabel vedoucí k modelu je jeden dvacetižilový. Pozice zapojení jednotlivých kabelů je tato:

Úplný plochý kabel propojující signály z horní svorkovnice modulu ADAM4050 se připojí do horního konektoru propojovacího bloku. Je třeba dbát na správnou orientaci, konektor má zámek zajišťující jednoznačné připojení.

Neúplný plochý kabel propojující většinu signálů z dolní svorkovnice modulu ADAM4050 se připojí do dolního konektoru propojovacího bloku. Konektor je taktéž vybaven zámkem zajišťující jednoznačně orientaci konektoru.

Úplný plochý kabel z příslušenství modelu automatické pračky patří do prostředního konektoru propojovacího bloku. I tento konektor je vybaven zámkem.

Zámky všech konektorů na propojovacím bloku směřují směrem dolů.



Obrázek 18: Zapojení propojovacího bloku

Dále je třeba zapojit zdroj napájecího napětí 24 V stejnosměrných. K tomuto účelu slouží dvě svorky pro banánky. Jsou umístěny v levé části propojovacího bloku. Horní svorka, červená, slouží k přivedení kladného pólu napájecího napětí, dolní svorka, modrá, k přivedení záporného pólu napájecího napětí. Provoz připojeného modulu a modelu je ovládán přepínačem více vlevo, vedle svorky kladného pólu napájecího napětí. V poloze nahoře je přivedeno napájecí napětí na svorky připojených zařízení.

5.4 Model automatické pračky

Model automatické pračky se připojuje pomocí dvacetižilového plochého kabelu k propojovacímu modulu. Kabel není orientován, správná poloha v konektorech je zajištěna zámky. Žádná další propojení není nutno realizovat. Napájení modelu je provedeno pomocí propojovacího bloku přes vypínač s označením Vyp.

5.4.1 Význam jednotlivých vodičů v kabelu

Následující tabulka obsahuje popis jednotlivých signálů obsažených v kabelu, jejich směr vzhledem k modelu, číslo pinu konektoru a označení vstupního či výstupního místa modulu ADAM4050, které tento signál obsluhuje.

Číslo pinu	Směr	Význam	V/V místo
1	Vstup	GND	-
2	Vstup	+24 V	-
3	Vstup	Buben vpravo	DO 0
4	Vstup	Buben vlevo	DO 1
5	Vstup	Zvýšené otáčky	DO 2
6	Nezapojen	-	-
7	Vstup	GND	-
8	Vstup	+24 V	-
9	Vstup	Topení	DO 3
10	Vstup	Napouštění	DO 4
11	Vstup	Vypouštění	DO5
12	Nezapojen	-	-
13	Nezapojen	-	-
14	Nezapojen	-	-
15	Výstup	Hladina 50 %	DI 5
16	Výstup	Hladina 100 %	DI 4
17	Výstup	Teplota 90°C	DI 3
18	Výstup	Teplota 60°C	DI 2
19	Výstup	Teplota 40°C	DI 1
20	Výstup	Teplota 30°C	DI 0

Tabulka 6: Význam a propojení signálů modelu automatické pračky

6 NÁVOD K POUŽITÍ

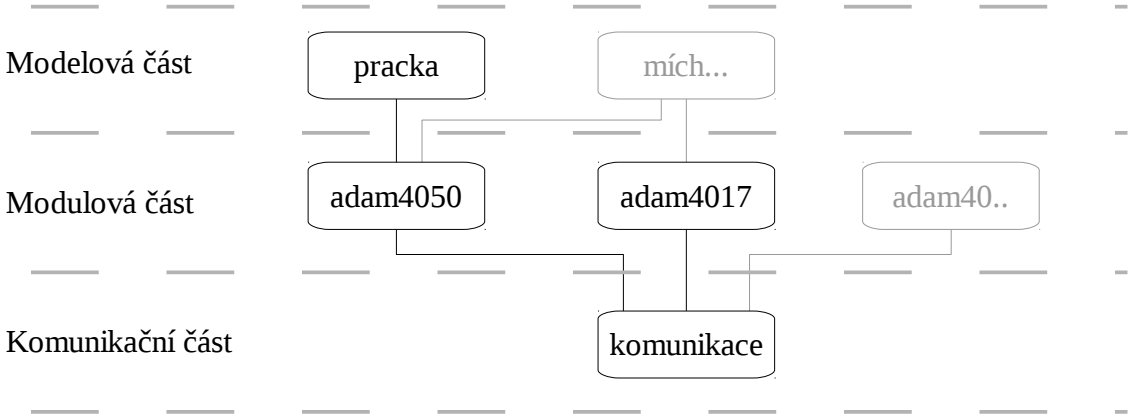
Pro snadné použití při výuce byl zpracován návod k použití. Tento návod je možno rozdělit na několik hlavních částí. První z nich je zaměřena na fyzické připojení modelu a potřebných modulů k počítači, druhá na použití knihovny funkcí poskytující funkce určené k práci s modelem a třetí část je zaměřena na popis celé knihovny a kompletní popis všech funkcí, které poskytují jednotlivé části knihovny vytvořené pro řízení modelu pračky a také funkcí, které je možné použít pro práci s moduly ADAM4017 a ADAM4050. Modul ADAM4017 není pro řízení modelu pračky použit. Obsah návodu k použití je možno popsat těmito body:

- Popis jednotlivých částí, které jsou pro připojení modelu a jeho řízení potřeba
- Podrobný popis zapojení těchto částí
- Jednoduchý příklad použití knihovny k řízení tohoto modelu
- Popis knihovny, která je pro řízení tohoto modelu k dispozici
- Popis další části knihovny popisující funkce pro modul ADAM4017
- Popis propojovacího bloku pro servisní práce

Návod k použití je k dispozici jako samostatný soubor ve formátu pdf, který je přílohou této práce.

7 PROGRAMOVÁ PODPORA

K ovládání modelu pračky je v programovacím jazyce C vytvořena knihovna funkcí. Tato knihovna je řešena jako tříúrovňová. Každá úroveň zajišťuje specifickou část komunikace. Blokové schéma návazností jednotlivých částí knihovny je takovéto:



Obrázek 19: Blokové schéma knihovny

Blokové schéma naznačuje možnost rozšíření knihovny o další části, čímž se knihovna stane univerzálnější.

V této práci byly realizovány tyto části knihovny:

- Modelová část, obsažena v těchto souborech: pracka.h, pracka.cpp
- Modulová část,obsažena v těchto souborech: adam4050.h, adam4050.cpp, adam4017.h, adam4017.cpp
- Komunikační část, obsažena v těchto souborech: komunikace.h, komunikace.cpp

7.1 Komunikační část

Tato část knihovny slouží k sestavení a obsluze spojení pomocí sériové linky RS232 v operačním systému Windows. Mezi její úkoly patří zajistit přidělení prostředků sériového komunikačního rozhraní operačním systémem, nastavit parametry komunikace, obsloužit komunikaci a opět komunikační rozhraní korektně uvolnit pro použití v systému Windows. Nastavení parametrů rozhraní je z velké části dáno pevně, požadavky nutnými pro komunikaci se vstupně – výstupním modulem ADAM4050. V programech používajících tuto knihovnu je možno nastavovat komunikační rychlost rozhraní. Uživatel tedy nemusí znát programátorské postupy, kterými ve svých programech zajistí přidělení prostředků

konkrétního komunikačního rozhraní. Tato část knihovny byla napsána za pomoci příručky [2] volně přístupné v síti Internet.

7.2 Modulová část

Modulová část knihovny je rozdělena na dvě části. Každá část je zaměřena na obsluhu jednoho typu modulu ADAM.

7.2.1 Knihovna funkcí modulu ADAM4050

Tato část slouží k ovládání jednotlivých modulů ADAM. Pro potřeby řízení modelu pračky postačuje jen jeden vstupně – výstupní modul – ADAM4050. Pro tento modul byla zpracována knihovna funkcí, která poskytuje potřebné funkce pro práci s tímto modulem i nad rámec, který je úlohou řízení modelu pračky nutně vyžadován. Tato část ke své činnosti využívá funkcí komunikační části, kterou před uživatelem skrývá. Uživatel se tedy v programech nemusí zabývat prací s obsluhou komunikačního rozhraní. Knihovna poskytuje funkce pro zahájení činnosti s modulem ADAM4050 – včetně ověření, že se jedná o správný typ modulu, nastavení komunikační rychlosti a adresy na lince RS485 v modulu, nastavení hodnot výstupních míst modulu, čtení hodnot ze vstupních míst a funkci pro korektní ukončení práce s modulem. Tato knihovna před uživatelem skrývá komunikační protokol modulu tak, že uživatel nemusí tento protokol znát.

Způsob práce s komunikačním rozhraním v této části knihovny byl volen tak, aby toto rozhraní bylo blokováno programem jen po nezbytně nutnou dobu potřebnou k vyřízení povelů zadaných uživatelem. Komunikační rozhraní je tedy možné použít i v jiném programu, kterým může být řízena jiná úloha s jinými zařízeními na téže lince RS485. Případné kolize při takovémto způsobu použití ale musí uživatel řešit sám.

7.2.2 Knihovna funkcí modulu ADAM4017

Tato část obsahuje funkce pro práci s modulem ADAM4017. Pro svou činnost používá služeb komunikační části knihovny stejným způsobem, jako část zaměřená na práci s modulem ADAM4050. Způsob práce s modulem je podobný jako práce s modulem ADAM4050, funkce řešící podobný problém mají podobné názvy. Využití funkcí této části knihovny je možné při výuce programování číslicových regulátorů, kdy tato část knihovny může zajišťovat čtení analogových hodnot v řízeném procesu či modelu řízeného modelu,

část zaměřená na modul ADAM4050 může s tímto modulem poskytovat funkce pro řízení akčních členů pomocí pulsně šířkové modulace.

7.3 Modelová část

Tato část slouží k přímému ovládání modelu pračky. Pro jiné modely technologických zařízení či procesů, na rozdíl od předcházejících částí, není použitelná. Uživateli poskytuje funkce, pomocí kterých může korektně zahájit práci s modelem, v rámci čehož dojde též k inicializaci a prověření správného propojení a použití správných propojovacích modulů, funkci pro korektní ukončení práce s modelem, a funkce pro ovládání jednotlivých režimů a čtení výstupních stavů modelu. Před uživatelem skrývá celý fyzický řetězec, kterým je model k počítači připojen. Pro správnou činnost této knihovny je tedy nutné dodržet propojovací schéma přesně tak, jak bylo navrženo.

Ke své činnosti využívá funkcí knihovny adam4050 z modulové části.

8 ŘEŠENÍ PROGRAMOVÉ ČÁSTI

V následující části jsou popisovány jednotlivé části knihovny a funkce, které byly pro řízení modelu realizovány. Rozdělení části je stejné, jako v předchozí kapitole, podle jednotlivých částí knihovny.

8.1 Komunikační část

Tato část knihovny využívá služeb operačního systému Windows. Uživateli poskytuje tyto datové typy a funkce:

8.1.1 Datový typ INFOPORT, P_INFOPORT

Datový typ infoport je struktura, která obsahuje informace potřebné pro žádost operačnímu systému o přidělení komunikačního rozhraní, jeho správné nastavení a zjištění, zda již není přidělen. Tato struktura by měla být vytvořena uživatelem před voláním funkce pro inicializaci komunikace pomocí tohoto komunikačního rozhraní. Datové položky, které je v této struktuře potřeba vyplnit jsou:

- nazev
- kod_baudrate

Doporučuji též vložit hodnotu NULL do datové položky hSerial, čímž dojde k její inicializaci na definovanou hodnotu.

Deklarace datového typu

```
typedef struct infoport{
      char nazev[5];
      int kod_baudrate;
      HANDLE hSerial;
      } INFOPORT, *P_INFOPORT;
```

8.1.2 Funkce otevri_port

Funkce slouží ke generování žádosti o přidělení komunikačního rozhraní operačním systémem a nastavení komunikačních parametrů toho rozhraní. Podle výsledků žádosti o přidělení komunikačního rozhraní funkce nastaví potřebné hodnoty ve struktuře INFOPORT předané v poli parametrů a vrátí příslušnou celočíselnou hodnotu.

Vstupní parametr

- port: ukazatel na strukturu INFOPORT

Návratová hodnota

- 0: Funkce selhala. Komunikační rozhraní nelze přidělit, nebo nelze nastavit požadované parametry.
- 1: Funkce proběhla úspěšně.

Deklarace funkce

```
int otevri_port(P_INFOPORT port);
```

8.1.3 Funkce zavri_port

Funkce slouží ke korektnímu ukončení komunikace pomocí komunikačního rozhraní a korektnímu odevzdání příslušných prostředků operačnímu systému. Funkce též nastavuje datovou položku hSerial na NULL ve struktuře předané jako jediný parametr, čímž se zajistí korektní chování při dalším použití komunikačního rozhraní popisovaného touto strukturou. Funkce vrací celočíselnou hodnotu v závislosti na průběhu.

Vstupní parametr

- port: ukazatel na strukturu INFOPORT, která popisuje přidělené komunikační rozhraní, které má být vráceno operačnímu systému

Návratová hodnota

- 0: Funkci byla předána jako parametr struktura, která nepopisuje operačním systémem přidělené komunikační rozhraní.
- 1: Komunikační rozhraní bylo vráceno operačnímu systému.

Deklarace funkce

```
int zavri_port(P_INFOPORT port);
```

8.1.4 Funkce precti_zpravu

Funkce slouží k přijetí zprávy z komunikačního rozhraní popisovaného strukturou předanou v poli parametrů jako položka port. Funkce vrací celočíselnou návratovou hodnotu v závislosti na úspěšnosti čtení z komunikačního rozhraní.

Vstupní parametry

- zprava: ukazatel na začátek paměti kam má být čtená zpráva uložena
- max_delka: Maximální počet znaků, které mohou být do paměti uloženy
- port: ukazatel na strukturu popisující komunikační rozhraní, ze kterého se má zpráva číst

Návratová hodnota

- 0: Čtení zprávy se nezdařilo. Možnou příčinou je neinicializovaná komunikace na komunikačním rozhraní, selhalo čtení, nebo nebyla odeslána žádná zpráva z připojeného zařízení.
- X: kladné celé číslo – zpráva byla načtena. Počet přečtených znaků obsahuje návratová hodnota.

Deklarace funkce

```
int precti_zpravu(char * zprava, unsigned short max_delka, P_INFOPORT port);
```

8.1.5 Funkce odesli_zpravu

Funkce slouží k posílání zpráv pomocí komunikačního rozhraní. Použité komunikační rozhraní, které je popisováno strukturou odkazovanou parametrem port musí být operačním systémem přiděleno – inicializováno pomocí funkce otevri_port, jinak odeslání zprávy nebude úspěšné. Zpráva musí být ukončena znakem '\0'

Vstupní parametry

- zprava: ukazatel na začátek paměti, ve které je zpráva uložena
- port: ukazatel na strukturu popisující komunikační rozhraní, ze kterého se má zpráva číst

Návratová hodnota

- 0: Odesílání zprávy se nezdařilo. Příčina může být v nepřiděleném komunikačním rozhraní, nebo selhalo odesílání zprávy.
- X: počet odeslaných znaků pomocí komunikačního rozhraní

Deklarace funkce

```
int odesli_zpravu(char * zprava, P_INFOPORT port);
```

8.1.6 Funkce zjisti_baudrate

Funkce převádí hodnotu konstanty používanou operačním systémem na odpovídající komunikační rychlost sériového rozhraní.

Vstupní parametr

- baud_code: konstanta operačního systému, jejíž odpovídající rychlost má být zjištěna

Návratová hodnota

- 0: Převod konstanty na hodnotu selhal. Nejpravděpodobnější příčinou je hodnota vstupního parametru neodpovídající platné konstantě operačního systému.
- X: Komunikační rychlost vyjádřená jako počet přenesených logických stavů za jednu sekundu.

Deklarace funkce

```
int zjisti_baudrate(int baud_code)
```

8.1.7 Funkce zjisti_kod_baudrate

Funkce převádí požadovanou komunikační rychlost na kód operačního systému používaný pro nastavení požadované rychlosti. Funkce by měla být použita při nastavování parametru kod_baudrate ve struktuře INFOPORT.

Vstupní parametr

- rychlost: požadovaná rychlost vyjádřená jako počet přenesených logických stavů za jednu sekundu

Návratová hodnota

- 0: Zjištění konstanty odpovídající požadované komunikační rychlosti se nezdařilo. Nejspíše byla požadována rychlost, která není komunikačním rozhraním podporována.

- X: hodnota konstanty, kterou operační systém používá pro nastavení požadované komunikační rychlosti

Deklarace funkce

```
int zjisti_kod_baudrate(int rychlost)
```

8.1.8 Funkce nacti_parametry

Toto je neveřejná funkce komunikační části knihovny. Slouží pro zjištění parametrů komunikačního rozhraní, které je popisováno strukturou, na kterou ukazuje ukazatel v poli parametrů označený jako port.

Vstupní parametry

- Parametry: struktura, do které bude vloženo nastavení komunikačního rozhraní
- port: odkaz na strukturu INFOPORT, která popisuje rozhraní, jehož nastavení má být zjištěno.

Návratová hodnota

- 0: komunikační rozhraní nebylo inicializováno pomocí funkce otevri_port, nebo selhalo čtení parametrů
- 1: Parametry byly zjištěny a vloženy do odpovídající struktury.

Funkční prototyp

```
int nacti_parametry(DCB * parametry, P_INFOPORT port)
```

8.2 Modulová část

Tato část pro práci s komunikačním rozhraním využívá výhradně služeb komunikační části. Je tedy nezávislá na operačním systému. Slouží k ovládání příslušných modulů ADAM připojených pomocí sériového komunikačního rozhraní. Může sestávat z více částí rozdělených do jednotlivých souborů. V rámci této práce byly realizovány dvě části. Jedna pro modul ADAM4050 a druhá pro modul ADAM4017. Popis poskytovaných funkcí je uveden v následujících odstavcích.

8.2.1 Popis funkcí knihovny modulu ADAM4050

Funkce této knihovny jsou přístupné po vložení řádku #include „adam4050.h“ na začátek kódu programu.

8.2.1.1 Datový typ ADAM4050, P_ADAM4050

Datový typ ADAM4050 je struktura, která obsahuje informace potřebné pro komunikační část knihovny, pomocí které probíhá odesílání a čtení zpráv ze zařízení a také informaci o adrese modulu ADAM4050 na komunikační lince RS485. Uživatel by neměl tuto strukturu ve svých programech vytvářet sám, ani měnit údaje uvedené v datových položkách této struktury. Struktura je dynamicky vytvářena na základě úspěšné inicializace komunikace s modulem ADAM4050 a také je paměť alokovaná pro tuto strukturu knihovnou automaticky uvolňována, pokud je práce s modulem ukončena funkcí ADAM4050_zavri. Po uživateli knihovny je vyžadováno pouze uchování odkazu na tuto strukturu ve vhodné proměnné, kterou bude jednoznačně identifikovat konkrétní modul, se kterým chce uživatel pracovat. Tím je umožněno pracovat v jednom programu s více moduly připojenými na stejné nebo i jiné komunikační lince RS485.

Deklarace datového typu

```
typedef struct adam4050{
      P_INFOPORT port;
      unsigned char adresa;
      }ADAM4050, *P_ADAM4050;
```

8.2.1.2 Funkce ADAM4050_inicializace

Tato funkce slouží k inicializaci práce se vstupně – výstupním modulem ADAM4050. Součástí inicializace je prověření funkčnosti komunikační trasy a její nastavení a taktéž ověření, zda cílový modul je skutečně správného typu. Funkci je třeba předat v poli parametrů údaje o názvu komunikačního rozhraní v operačním systému Windows pomocí kterého má být komunikace vedena, požadovaná rychlost komunikace vyjádřená jako celočíselná hodnota vyjadřující počet přenesených logických stavů komunikačním rozhraním za sekundu a logickou adresu modulu ADAM4050, se kterým má být komunikace vedena.

Vstupní parametry

- nazev_portu: pětiznakový název komunikačního rozhraní ukončený znakem '\0', např. „COM1\0“
- baudrate: Požadovaná komunikační rychlost
- adresa: adresa modulu se kterým se bude pracovat

Návratová hodnota

- NULL: Během inicializace práce s modulem došlo k chybě. Příčina může být ve špatných hodnotách předaných parametrů, nebo není požadované komunikační rozhraní k dispozici.
- X: Ukazatel na strukturu ADAM4050. Inicializace modulu proběhla bez chyb. S modulem je možno pracovat.

Deklarace funkce

```
P_ADAM4050 ADAM4050_inicializace(char * nazev_portu, int baudrate, unsigned char adresa);
```

8.2.1.3 Funkce ADAM4050_zavri

Tato funkce slouží ke korektnímu ukončení práce s modulem ADAM4050. Funkce korektně vrátí přidělené komunikační rozhraní zpět operačnímu systému a uvolní paměť alokovanou pro držení informací o modulu a používaném komunikačním rozhraní.

Vstupní parametr

- modul: ukazatel na strukturu ADAM4050, která popisuje modul, se kterým má být práce ukončena.

Návratová hodnota

- 1: práce byla ukončena

Deklarace funkce

```
int ADAM4050_zavri(ADAM4050 * modul);
```

8.2.1.4 Funkce ADAM4050_zjisti_baudrate

Funkce přečte komunikační rychlost nastavenou v modulu. Vhodné pro ověření stavu po pokusu o změnu nastavení modulu.

Vstupní parametr

- modul: ukazatel na strukturu ADAM4050, která popisuje modul, kterému má být zjištěna komunikační rychlost.

Návratová hodnota

- 0: Zjištění komunikační rychlosti se nezdařilo.
- X: Zjištěná komunikační rychlost vyjádřená jako počet logických stavů přenesených komunikačním rozhraním za sekundu.

Deklarace funkce

```
int ADAM4050_zjisti_baudrate(ADAM4050 * modul);
```

8.2.1.5 Funkce ADAM4050_nastav_baudrate

Funkce slouží k nastavení nové komunikační rychlosti v modulu. Modul musí být přepnut do inicializačního režimu, jinak změna nebude modulem akceptována. Nová komunikační rychlost se projeví až po restartu modulu. Změna komunikační rychlosti rozhraní v počítači nebude provedena, je třeba ukončit práci s modulem a znovu inicializovat s novou komunikační rychlostí.

Vstupní parametry

- modul: ukazatel na strukturu ADAM4050, která popisuje modul, kterému má být změněna komunikační rychlost
- baudrate: nová komunikační rychlost, vyjádřena jako celočíselná hodnota počtu logických stavů přenesených komunikačním rozhraním za sekundu.

Návratová hodnota

- 0: změna nastavení komunikační rychlosti v modulu se nezdařila. Příčinou může být neplatná hodnota předaná v parametru baudrate nebo nepřepnutím modulu do režimu INIT

- 1: změna rychlosti byla provedena a modulem akceptována

Deklarace funkce

```
int ADAM4050_nastav_baudrate(ADAM4050 * modul, int baudrate);
```

8.2.1.6 Funkce ADAM4050_nastav_adresu

Funkce slouží k nastavení nové logické adresy modulu ADAM4050 popisovaným strukturou odkazovanou parametrem modul. Pro změnu logické adresy není nutné přepínat modul do režimu INIT. Změna je provedena ihned. Uživatel by měl ukončit práci s modulem a inicializovat práci znovu se změněnou adresou. Je také možné po úspěšném provedení změny adresy v modulu změnit adresu přímo ve struktuře ADAM4050 příkazem:

```
modul->adresa = nova_adresa;
```

Vstupní parametry

- Modul: ukazatel na strukturu ADAM4050, která popisuje modul, kterému má být změněna logická adresa
- nova_adresa: adresa, která bude dosazena místo té stávající

Návratová hodnota

- 0: Změna logické adresy se nezdařila.
- 1:Změna logické adresy byla provedena. Odteď bude modul naslouchat na nové adrese.

Deklarace funkce

```
int ADAM4050_nastav_adresu(ADAM4050 * modul, unsigned char nova_adresa);
```

8.2.1.7 Funkce ADAM4050_zjisti_linky

Funkce slouží k přečtení stavu vstupních a výstupních míst modulu, který je popisován strukturou na kterou odkazuje ukazatel předaný v poli parametrů jako hodnota označena názvem modul.

Vstupní parametry

- modul: ukazatel na strukturu ADAM4050, která popisuje modul u kterého má být stav linek zjištěn

- vystupy: ukazatel na proměnnou typu unsigned char, do které bude umístěn stav všech výstupních linek modulu. Číselné značení linek na modulu odpovídá váze bitu v proměnné.
- vstupy: ukazatel na proměnnou typu unsigned char, do které bude umístěn stav všech vstupních linek modulu. Číselné značení linek na modulu odpovídá váze bitu v proměnné.

Návratová hodnota

- 0: Načtení stavu vstupních a výstupních linek modulu se nezdařilo, hodnoty v proměnných odkazovaných parametry vstupy a výstupy nejsou platné.
- 1: Načtení stavu vstupních a výstupních linek se zdařilo, hodnoty v proměnných odkazovaných parametry vstupy a výstupy jsou platné.

Deklarace funkce

```
int ADAM4050_zjisti_linky(ADAM4050 * modul, unsigned char * vystupy, unsigned char * vstupy);
```

8.2.1.8 Funkce ADAM4050_nastav_vystupy

Funkce slouží k hromadnému nastavení výstupních linek modulu, který je popisován strukturou odkazovanou ukazatelem předaným v poli vstupních parametrů jako hodnota označena názvem modul.

Vstupní parametry

- modul: ukazatel na strukturu ADAM4050, která popisuje modul, kterému mají být nastaveny všechny výstupní linky
- vystupy: hodnota unsigned char, která nese informaci o požadovaném nastavení všech výstupních linkách modulu. Váha bitu odpovídá číselnému značení výstupních linek na modulu.

Návratová hodnota

- 0: Nastavení výstupních linek nebylo úspěšné.
- 1: Nastavení výstupních linek se zdařilo.

Deklarace funkce

```
int ADAM4050_nastav_vystupy(ADAM4050 * modul, unsigned char vystupy);
```

8.2.1.9 Funkce ADAM4050_zapni_bit

Funkce slouží k aktivaci konkrétní výstupní linky. Pokud je model vybaven výstupy s tranzistory s otevřeným kolektorem a na výstup připojený elektrický obvod je vybaven rezistorem v režimu pull up tak, jak je provedeno v propojovacím bloku, bude detekována logická úroveň 0.

Vstupní parametry

- modul: ukazatel na strukturu ADAM4050, která popisuje modul, kterému má být aktivována výstupní linka.
- bit: číslo výstupní linky, která má být aktivována. Hodnota odpovídá číselnému značení linek na modulu.

Návratová hodnota

- 0: Aktivace výstupní linky se nezdařila.
- 1: Aktivace výstupní linky proběhla úspěšně.

Deklarace funkce

```
int ADAM4050_zapni_bit(ADAM4050 * modul, unsigned char bit);
```

8.2.1.10 Funkce ADAM4050_vypni_bit

Funkce slouží k deaktivaci konkrétní výstupní linky. Pokud je model vybaven výstupy s tranzistory s otevřeným kolektorem a na výstup připojený elektrický obvod je vybaven rezistorem v režimu pull up tak, jak je provedeno v propojovacím bloku, bude detekována logická úroveň 1.

Vstupní parametry

- modul: ukazatel na strukturu ADAM4050, která popisuje modul, kterému má být aktivována výstupní linka.
- bit: číslo výstupní linky, která má být deaktivována. Hodnota odpovídá číselnému značení linek na modulu.

Návratová hodnota

- 0: Deaktivace výstupní linky se nezdařila.
- 1: Deaktivace výstupní linky proběhla úspěšně.

Deklarace funkce

```
int ADAM4050_vypni_bit(ADAM4050 * modul, unsigned char bit);
```

8.2.2 Popis funkcí knihovny modulu ADAM4017

Funkce této knihovny jsou přístupné po vložení řádku #include „adam4017.h“ na začátek kódu programu.

8.2.2.1 Datový typ ADAM4017, P_ADAM4017

Datový typ ADAM4017 je struktura, která obsahuje informace potřebné pro komunikační část knihovny, pomocí které probíhá odesílání a čtení zpráv ze zařízení a také informaci o adrese modulu ADAM4017 na komunikační lince RS485. Uživatel by neměl tuto strukturu ve svých programech vytvářet sám, ani měnit údaje uvedené v datových položkách této struktury. Struktura je dynamicky vytvářena na základě úspěšné inicializace komunikace s modulem ADAM4017 a také je paměť alokovaná pro tuto strukturu knihovnou automaticky uvolňována, pokud je práce s modulem ukončena funkcí ADAM4017_zavri. Po uživateli knihovny je vyžadováno pouze uchování odkazu na tuto strukturu ve vhodné proměnné, kterou bude jednoznačně identifikovat konkrétní modul, se kterým chce uživatel pracovat. Tím je umožněno pracovat v jednom programu s více moduly připojenými na stejné nebo i jiné komunikační lince RS485.

Deklarace datového typu

```
typedef struct adam4017{
      P_INFOPORT port;
      unsigned char adresa;
      }ADAM4017, *P_ADAM4017;
```

8.2.2.2 Datový typ ADAM4017_rozsah

Jedná se o datový typ výčet, který obsahuje hodnoty všech rozsahů přístupných v modulu ADAM4017. Tento typ je použit při zjišťování nebo nastavování rozsahu modulu.

Deklarace datového typu

```
typedef enum{
      ADAM4017_10V   = 0x38,
      ADAM4017_5V    = 0x39,
      ADAM4017_1V    = 0x41,
      ADAM4017_500mV = 0x42,
      ADAM4017_150mV = 0x43,
      ADAM4017_20mA  = 0x44
      } ADAM4017_rozsah;
```

8.2.2.3 Datový typ ADAM4017_integrace

Jedná se o datový typ výčet, který obsahuje hodnoty nastavení integračního času převodu analogové hodnoty na digitální. Integrační časy jsou vztaženy k frekvenci rozvodné sítě.

Deklarace datového typu

```
typedef enum{
      ADAM4017_50Hz,
      ADAM4017_60Hz
      } ADAM4017_integrace;
```

8.2.2.4 Funkce ADAM4017_inicializace

Tato funkce slouží k inicializaci práce s modulem analogových vstupů ADAM4017. Součástí inicializace je prověření funkčnosti komunikační trasy a její nastavení a taktéž ověření, zda cílový modul je skutečně správného typu. Funkci je třeba předat v poli parametrů údaje o názvu komunikačního rozhraní v operačním systému Windows pomocí kterého má být komunikace vedena, požadovaná rychlost komunikace vyjádřená jako celočíselná hodnota vyjadřující počet přenesených logických stavů komunikačním rozhraním za sekundu a logickou adresu modulu ADAM4017, se kterým má být komunikace vedena.

Vstupní parametry

- nazev_portu: pětiznakový název komunikačního rozhraní ukončený znakem '\0', např. „COM1\0“
- baudrate: Požadovaná komunikační rychlost
- adresa: adresa modulu se kterým se bude pracovat

Návratová hodnota

- NULL: Během inicializace práce s modulem došlo k chybě. Příčina může být ve špatných hodnotách předaných parametrů, nebo není požadované komunikační rozhraní k dispozici.
- X: Ukazatel na strukturu ADAM4017. Inicializace modulu proběhla bez chyb. S modulem je možno pracovat.

Deklarace funkce

```
P_ADAM4017 ADAM4017_inicializace(char * nazev_portu, int baudrate, unsigned char adresa);
```

8.2.2.5 Funkce ADAM4017_zavri

Tato funkce slouží ke korektnímu ukončení práce s modulem ADAM4017. Funkce korektně vrátí přidělené komunikační rozhraní zpět operačnímu systému a uvolní paměť alokovanou pro držení informací o modulu a používaném komunikačním rozhraní.

Vstupní parametr

- modul: ukazatel na strukturu ADAM4017, která popisuje modul, se kterým má být práce ukončena.

Návratová hodnota

- 1: práce byla ukončena

Deklarace funkce

```
int ADAM4017_zavri(P_ADAM4017 modul);
```

8.2.2.6 Funkce ADAM4017_zjisti_baudrate

Funkce přečte komunikační rychlost nastavenou v modulu. Vhodné pro ověření stavu po pokusu o změnu nastavení modulu.

Vstupní parametr

- modul: ukazatel na strukturu ADAM4017, která popisuje modul, kterému má být zjištěna komunikační rychlost.

Návratová hodnota

- 0: Zjištění komunikační rychlosti se nezdařilo.
- X: Zjištěná komunikační rychlost vyjádřená jako počet logických stavů přenesených komunikačním rozhraním za sekundu.

Deklarace funkce

```
int ADAM4017_zjisti_baudrate(P_ADAM4017 modul);
```

8.2.2.7 Funkce ADAM4017_nastav_baudrate

Funkce slouží k nastavení nové komunikační rychlosti v modulu. Modul musí být přepnut do inicializačního režimu, jinak změna nebude modulem akceptována. Nová komunikační rychlost se projeví až po restartu modulu. Změna komunikační rychlosti rozhraní v počítači nebude provedena, je třeba ukončit práci s modulem a znovu inicializovat s novou komunikační rychlostí.

Vstupní parametry

- modul: ukazatel na strukturu ADAM4017, která popisuje modul, kterému má být změněna komunikační rychlost
- baudrate: nová komunikační rychlost, vyjádřena jako celočíselná hodnota počtu logických stavů přenesených komunikačním rozhraním za sekundu.

Návratová hodnota

- 0: změna nastavení komunikační rychlosti v modulu se nezdařila. Příčinou může být neplatná hodnota předaná v parametru baudrate nebo nepřepnutím modulu do režimu INIT
- 1: změna rychlosti byla provedena a modulem akceptována

Deklarace funkce

```
int ADAM4017_nastav_baudrate(P_ADAM4017 modul, int baudrate);
```

8.2.2.8 Funkce ADAM4017_nastav_adresu

Funkce slouží k nastavení nové logické adresy modulu ADAM4017 popisovaným strukturou odkazovanou parametrem modul. Pro změnu logické adresy není nutné přepínat modul do režimu INIT. Změna je provedena ihned. Uživatel by měl ukončit práci s

modulem a inicializovat práci znovu se změněnou adresou. Je také možné po úspěšném provedení změny adresy v modulu změnit adresu přímo ve struktuře ADAM4017 příkazem:

```
modul->adresa = nova_adresa;
```

Vstupní parametry

- Modul: ukazatel na strukturu ADAM4017, která popisuje modul, kterému má být změněna logická adresa
- nova_adresa: adresa, která bude dosazena místo té stávající

Návratová hodnota

- 0: Změna logické adresy se nezdařila.
- 1:Změna logické adresy byla provedena. Odteď bude modul naslouchat na nové adrese.

Deklarace funkce

```
int ADAM4017_nastav_adresu(P_ADAM4017 modul, unsigned char nova_adresa);
```

8.2.2.9 Funkce ADAM4017_zjisti_adresu

Funkce umožňuje zjistit adresu nastavenou v modulu popisovanému strukturo ADAM4017 předanou v poli parametrů jako položka s názvem modul. Funkce je vhodná pouze při zjišťování nastavení modulu s neznámým nastavením, který byl přepnut do inicializačního režimu a je tedy přístupný na adrese 0 s komunikační rychlostí 9600 baudů. V režimu INIT jsou poskytovány správné údaje o nastavení modulu, i když současně použité INIT vlastnosti jsou jiné.

Vstupní parametry

- Modul: ukazatel na strukturu ADAM4017, která popisuje modul, jehož adresa se má zjistit.
- Adresa: ukazatel na proměnnou typu unsigned char, do které bude vložena zjištěná adresa modulu.

Návratová hodnota

- 0: Zjistit nastavenou adresu se nezdařilo

- 1: Adresa byla zjištěna a vložena na místo v paměti odkazované parametrem adresa

Deklarace funkce

```
int ADAM4017_zjisti_adresu(P_ADAM4017 modul, unsigned char * adresa);
```

8.2.2.10 Funkce ADAM4017_nastav_rozsah

Funkce slouží k nastavení vstupního rozsahu analogového převodníku modulu. Použitelné rozsahy jsou uvedeny jako jednotlivé hodnoty typu ADAM4017_rozsah. Při použití této funkce je třeba dbát na dodržení doby klidu přibližně 7 s, kterou modul ADAM4017 při změně nastavení ovlivňující analogově – digitální převod vyžaduje.

Vstupní parametry

- Modul: ukazatel na strukturu ADAM4017, která popisuje modul, jehož rozsah se má nastavit.
- Rozsah: hodnota rozsahu, který má být nastaven.

Návratová hodnota

- 0: Nastavit požadovaný rozsah se nezdařilo.
- 1: Rozsah byl nastaven.

Deklarace funkce

```
int ADAM4017_nastav_rozsah(P_ADAM4017 modul, ADAM4017_rozsah rozsah);
```

8.2.2.11 Funkce ADAM4017_zjisti_rozsah

Funkce slouží ke zjištění nastaveného rozsahu analogového převodníku modulu.

Vstupní parametry

- Modul: ukazatel na strukturu ADAM4017, která popisuje modul, jehož rozsah má být zjištěn.
- Rozsah: ukazatel na proměnnou typu ADAM4017_rozsah, do které bude vložen zjištěný rozsah.

Návratová hodnota

- 0: Zjistit nastavený rozsah se nezdařilo.

- 1: Rozsah byl zjištěn a vložen do paměti odkazované ukazatelem rozsah.

Deklarace funkce

```
int ADAM4017_zjisti_rozsah(P_ADAM4017 modul, ADAM4017_rozsah * rozsah);
```

8.2.2.12 Funkce ADAM4017_nastav_integracni_cas

Funkce slouží k nastavení požadovaného integračního času. K dispozici jsou hodnoty integračního času uvedené jako možné hodnoty datového typu ADAM4017_integrace. Při použití této funkce je třeba dbát na dodržení doby klidu přibližně 7 s, kterou modul ADAM4017 při změně nastavení ovlivňující analogově – digitální převod vyžaduje.

Vstupní parametry

- Modul: ukazatel na strukturu ADAM4017, která popisuje modul, jehož integrační čas má být nastaven.
- frekvence_site: hodnota kmitočtu rozvodné sítě na kterou má být integrační čas optimalizován.

Návratová hodnota

- 0: Nastavit požadovaný integrační čas se nezdařilo.
- 1: Integrační čas byl nastaven.

Deklarace funkce

```
int ADAM4017_nastav_integracni_cas(P_ADAM4017 modul, ADAM4017_integrace
frekvence_site);
```

8.2.2.13 Funkce ADAM_zjisti_integracni_cas

Funkce slouží ke zjištění nastavené frekvence rozvodné sítě na kterou byl optimalizován integrační čas.

Vstupní parametry

- Modul: ukazatel na strukturu ADAM4017, která popisuje modul, jehož integrační čas má být nastaven.

- frekvence_site: ukazatel na proměnnou typu ADAM4017_integrace, do které má být vložena hodnota kmitočtu rozvodné sítě na kterou byl integrační čas převodu optimalizován.

Návratová hodnota

- 0: Zjistit nastavený integrační čas se nezdařilo.
- 1: Integrační čas byl zjištěn a hodnota vložena do odkazované proměnné.

Deklarace funkce

```
int ADAM4017_zjisti_integracni_cas(P_ADAM4017 modul, ADAM4017_integrace
* frekvence_site);
```

8.2.2.14 Funkce ADAM4017_nacti_vstupy

Funkce slouží k jednorázovému načtení všech osmi analogových vstupů modulu. Před voláním této funkce je nutné, aby uživatel vytvořil v paměti dostatek místa pro uložení zjištěných hodnot například takovýmto způsobem:

```
float kanaly[8];
```

Po té se dá funkce volat následujícím způsobem:

```
ADAM4017_nacti_vstupy(modul, kanaly);
```

Vstupní parametry

- Modul: ukazatel na strukturu ADAM4017, která popisuje modul se kterým se bude pracovat.
- data: ukazatel na začátek paměti, do které může být vložena řada 8 hodnot typu float.

Návratová hodnota

- 0: Načíst hodnoty analogových vstupů se nepodařilo.
- 1: Hodnoty analogových vstupů byly vloženy do odkazované paměti.

Deklarace funkce

```
int ADAM4017_nacti_vstupy(P_ADAM4017 modul, float * data);
```

8.2.2.15 Funkce ADAM4017_nacti_vstup

Funkce slouží k načtení hodnoty vstupního signálu na konkrétním analogovém vstupu. Číslo vstupu je možné volit v rozsahu 0 ÷ 7.

Vstupní parametry

- Modul: ukazatel na strukturu ADAM4017, která popisuje modul se kterým se bude pracovat.
- Kanal: číslo analogového kanálu, jehož hodnota vstupního signálu má být zjištěna.
- Data: ukazatel na proměnnou typu float, do které má být zjištěná hodnota vložena.

Návratová hodnota

- 0: Načíst hodnotu analogového vstupu se nepodařilo, nebo byl požadován převod na neexistujícím kanálu.
- 1: Hodnota analogových vstupu byla vložena do odkazované proměnné.

Deklarace funkce

```
int ADAM4017_nacti_vstup(P_ADAM4017 modul, unsigned char kanal, float * data);
```

8.3 Modelová část

Modelová část knihovny poskytuje funkce pro konkrétní modely technologických zařízení a procesů. Byla zpracována knihovna pro model automatické pračky, který byl předán k řešení této diplomové práce. Aby byla tato část knihovny použitelná, musí být model pračky připojen pomocí propojovacího bloku, který byl postaven v rámci řešení této diplomové práce. Tato část knihovny využívá funkcí modulové části, konkrétně části pro práci s modulem ADAM4050. Další knihovna funkcí, která je v této části použita, je standardní knihovna stdio.h, která by měla být k dispozici i v jiných operačních systémech než je systém Windows. Tato část poskytuje uživateli tyto funkce a datové typy:

8.3.1 Datový typ PRACKA

Datový typ PRACKA je struktura, která obsahuje informace potřebné pro práci s modelem automatické pračky. Jelikož knihovna má být použita pro výukové účely, bylo rozhodnuto, že struktura neponese informace o stavu výstupních hodnot modelu, které by byly

aktualizovány při každé operaci s modelem, ale uživatel – student – si bude muset o zjištění těchto stavů pomocí příslušných funkcí knihovny sám žádat. Struktura tak obsahuje jen jednu položku, kterou je odkaz na strukturu ADAM4050. Neznamená to však, že uživatel bude muset znát tuto strukturu nebo práci s modulovou částí knihovny. Veškerá tato činnost je pro uživatele skryta modelovou částí knihovny. Uživatel tak bude nucen jen vytvořit proměnnou typu P_PRACKA, ve které si bude uchovávat odkaz na strukturu, kterou modelová část knihovny vytvoří při inicializaci práce. Uvolnění paměti alokované pro tuto strukturu proběhne v rámci korektního ukončení práce s modelem.

Deklarace datového typu

```
typedef struct pracka{
      P_ADAM4050 modul;
      }PRACKA, *P_PRACKA;
```

8.3.2 Funkce PRACKA_inicializace

Funkce inicializuje práci s modelem automatické pračky. Součástí funkce je prověření komunikační trasy a nastavení propojovacího bloku do stavu, ve kterém je možné model pračky ovládat.

Vstupní parametry

- nazev_portu: název komunikačního rozhraní v operačním systému Windows, pomocí kterého je model pračky připojen k počítači. Proměnná musí obsahovat 5 znaků. Pátým znakem musí být znak '\0'. Obsah této proměnné může být například „COM1\0“
- baudrate: Požadovaná komunikační rychlost vyjádřená jako součet logických stavů přenesených komunikačním rozhraním za jednu sekundu.
- adresa_ADAM4050: Logická adresa modulu ADAM4050 na komunikační lince RS485, kterým je model pračky ovládán.

Návratová hodnota

- NULL: Inicializace modelu pračky se nezdařila. Chyba může být v sestavení komunikační trasy, v hodnotách parametrů, případně požadované komunikační rozhraní není k dispozici.

- X: Inicializace modelu pračky proběhla bez chyb. Návratová hodnota obsahuje ukazatel na strukturu, ve které jsou uloženy informace potřebné pro práci s modelem.

Deklarace funkce

```
P_PRACKA PRACKA_inicializace(char * nazev_portu, int baudrate, unsigned char adresa_ADAM4050);
```

8.3.3 Funkce PRACKA_zavri

Funkce slouží ke korektnímu ukončení práce s modelem pračky. V rámci funkce bude také deaktivován propojovací blok, model tedy nebude možné ovládat až do jeho další inicializace. Všechny datové struktury, které byly knihovnou automaticky vytvořeny budou zrušeny a paměť uvolněna.

Vstupní parametr

pracka: ukazatel na strukturu PRACKA, která popisuje model, se kterým má být práce ukončena.

Návratová hodnota

- 1: Práce s modelem byla ukončena.

Deklarace funkce

```
int PRACKA_zavri(P_PRACKA pracka);
```

8.3.4 Funkce PRACKA_buben_vlevo

Funkce zapne otáčení pracím bubnem doleva. Vzhledem k funkci modelu a výukovému určení knihovny funkce neřeší kolizi při zapnutí obou směrů otáčení pracovního bubnu. Model má pro tuto situaci zařízeno opravitelné chybové hlášení, které se dá odstranit tím, že bude vypnut jeden směr otáčení. Uživatel si tedy bude muset pamatovat zda je pracím bubnem otáčeno a jakým směrem.

Vstupní parametr

- pracka: ukazatel na strukturu PRACKA, která popisuje model, se kterým se bude pracovat.

Návratová hodnota

- 0: Aktivace směru otáčení pracím bubnem se nezdařila.
- 1: Aktivace směru otáčení pracím bubnem byla úspěšná.

Deklarace funkce

```
int PRACKA_buben_vlevo(P_PRACKA pracka);
```

8.3.5 Funkce PRACKA_buben_vpravo

Funkce zapne otáčení pracím bubnem doprava. Vzhledem k funkci modelu a výukovému určení knihovny funkce neřeší kolizi při zapnutí obou směrů otáčení pracovního bubnu. Model má pro tuto situaci zařízeno opravitelné chybové hlášení, které se dá odstranit tím, že bude vypnut jeden směr otáčení. Uživatel si tedy bude muset pamatovat zda je pracím bubnem otáčeno a jakým směrem.

Vstupní parametr

- pracka: ukazatel na strukturu PRACKA, která popisuje model, se kterým se bude pracovat.

Návratová hodnota

- 0: Aktivace směru otáčení pracím bubnem se nezdařila.
- 1: Aktivace směru otáčení pracím bubnem byla úspěšná.

Deklarace funkce

```
int PRACKA_buben_vpravo(P_PRACKA pracka)
```

8.3.6 Funkce PRACKA_buben_stop

Funkce slouží k zastavení otáčení pracím bubnem modelu pračky. Pokud byly aktivovány oba směry otáčení, dojde k vypnutí obou.

Vstupní parametr

- pracka: ukazatel na strukturu PRACKA, která popisuje model, se kterým se bude pracovat.

Návratová hodnota

- 0: Deaktivace směru otáčení pracím bubnem se nezdařila.
- 1: Deaktivace směru otáčení pracím bubnem byla úspěšná.

Deklarace funkce

```
int PRACKA_buben_stop(P_PRACKA pracka);
```

8.3.7 Funkce PRACKA_otacky_zvysene

Funkce slouží k aktivaci vyšších otáček pracího bubnu – ždímání. Směr otáčení je volen směrem otáčení pracího bubnu.

Vstupní parametr

- pracka: ukazatel na strukturu PRACKA, která popisuje model, se kterým se bude pracovat.

Návratová hodnota

- 0: Aktivace zvýšených otáček pracího bubnu se nezdařila
- 1: Aktivace zvýšených otáček pracího bubnu byla úspěšná.

Deklarace funkce

```
int PRACKA_otacky_zvysene(P_PRACKA pracka);
```

8.3.8 Funkce PRACKA_otacky_normalni

Funkce slouží k vypnutí zvýšených otáček pracího bubnu modelu pračky.

Vstupní parametr

- pracka: ukazatel na strukturu PRACKA, která popisuje model, se kterým se bude pracovat.

Návratová hodnota

- 0: Deaktivace zvýšených otáček pracího bubnu se nezdařila
- 1: Deaktivace zvýšených otáček pracího bubnu byla úspěšná.

Deklarace funkce

```
int PRACKA_otacky_normalni(P_PRACKA pracka);
```

8.3.9 Funkce PRACKA_napousteni_zapnout

Funkce na modelu pračky aktivuje napouštění vody do pracího prostoru.

Vstupní parametr

- pracka: ukazatel na strukturu PRACKA, která popisuje model, se kterým se bude pracovat.

Návratová hodnota

- 0: Zapnutí napouštění vody do pracího prostoru se nezdařilo.
- 1: Zapnutí napouštění vody do pracího prostoru se zdařilo.

Deklarace funkce

```
int PRACKA_napousteni_zapnout(P_PRACKA pracka);
```

8.3.10 Funkce PRACKA_napousteni_vypnout

Funkce na modelu pračky deaktivuje napouštění vody do pracího prostoru.

Vstupní parametr

- pracka: ukazatel na strukturu PRACKA, která popisuje model, se kterým se bude pracovat.

Návratová hodnota

- 0: Vypnutí napouštění vody do pracího prostoru se nezdařilo.
- 1: Vypnutí napouštění vody do pracího prostoru se zdařilo.

Deklarace funkce

```
int PRACKA_napousteni_vypnout(P_PRACKA pracka);
```

8.3.11 Funkce PRACKA_vypousteni_zapnout

Funkce na modelu automatické pračky aktivuje čerpadlo. Není kontrolováno, zda je současně zapnuto také napouštění vody.

Vstupní parametr

- pracka: ukazatel na strukturu PRACKA, která popisuje model, se kterým se bude pracovat.

Návratová hodnota

- 0: Zapnutí vypouštění vody z pracího prostoru se nezdařilo.
- 1: Zapnutí vypouštění vody z pracího prostoru se zdařilo.

Deklarace funkce

```
int PRACKA_vypousteni_zapnout(P_PRACKA pracka);
```

8.3.12 Funkce PRACKA_vypousteni_vypnout

Funkce na modelu automatické pračky deaktivuje čerpadlo.

Vstupní parametr

- pracka: ukazatel na strukturu PRACKA, která popisuje model, se kterým se bude pracovat.

Návratová hodnota

- 0: Vypnutí vypouštění vody z pracího prostoru se nezdařilo.
- 1: Vypnutí vypouštění vody z pracího prostoru se zdařilo.

Deklarace funkce

```
int PRACKA_vypousteni_vypnout(P_PRACKA pracka);
```

8.3.13 Funkce PRACKA_topeni_zapnout

Funkce zapne ohřev vody v pracím prostoru. Není kontrolováno, zda v pracím prostoru modelu pračky nějaká voda je. Tuto kontrolu by měl uživatel knihovny provést sám.

Vstupní parametr

- pracka: ukazatel na strukturu PRACKA, která popisuje model, se kterým se bude pracovat.

Návratová hodnota

- 0: Zapnutí ohřevu vody v pracím prostoru se nezdařilo.
- 1: Zapnutí ohřevu vody v pracím prostoru se zdařilo.

Deklarace funkce

```
int PRACKA_topeni_zapnout(P_PRACKA pracka);
```

8.3.14 Funkce PRACKA_topeni_vypnout

Funkce vypne ohřev vody v pracím prostoru.

Vstupní parametr

- pracka: ukazatel na strukturu PRACKA, která popisuje model, se kterým se bude pracovat.

Návratová hodnota

- 0: Vypnutí ohřevu vody v pracím prostoru se nezdařilo.
- 1: Vypnutí ohřevu vody v pracím prostoru se zdařilo.

Deklarace funkce

```
int PRACKA_topeni_vypnout(P_PRACKA pracka);
```

8.3.15 Funkce PRACKA_voda_50

Funkce slouží ke zjištění stavu čidla úrovně hladiny v pracím prostoru umístěném v polovině výšky pracího prostoru.

Vstupní parametry

- pracka: ukazatel na strukturu PRACKA, která popisuje model, se kterým se bude pracovat.
- stav: ukazatel na proměnnou typu char do které bude uložena informace o stavu čidla

Návratová hodnota

- 0: Zjištění stavu čidla se nezdařilo. Hodnota v proměnné stav není platná.

- 1: Zjištění stavu čidla se zdařilo. Hodnota v proměnné stav je platná.

Deklarace funkce

```
int PRACKA_voda_50(P_PRACKA pracka, char * stav);
```

8.3.16 Funkce PRACKA_voda_100

Funkce slouží ke zjištění stavu čidla zaplnění pracího prostoru vodou. Zaplnění pracího prostoru vodou způsobí též přetečení vody a neopravitelnou chybu modelu pračky. Model je nutno restartovat ručně tlačítkem na panelu modelu.

Vstupní parametry

- pracka: ukazatel na strukturu PRACKA, která popisuje model, se kterým se bude pracovat.
- stav: ukazatel na proměnnou typu char do které bude uložena informace o stavu čidla

Návratová hodnota

- 0: Zjištění stavu čidla se nezdařilo. Hodnota v proměnné stav není platná.
- 1: Zjištění stavu čidla se zdařilo. Hodnota v proměnné stav je platná.

Deklarace funkce

```
int PRACKA_voda_100(P_PRACKA pracka, char * stav);
```

8.3.17 Funkce PRACKA_teplota_30

Funkce slouží ke zjištění stavu čidla teploty prací vody nastaveného na teplotu 30°C.

Vstupní parametry

- pracka: ukazatel na strukturu PRACKA, která popisuje model, se kterým se bude pracovat.
- stav: ukazatel na proměnnou typu char do které bude uložena informace o stavu čidla

Návratová hodnota

- 0: Zjištění stavu čidla se nezdařilo. Hodnota v proměnné stav není platná.

- 1: Zjištění stavu čidla se zdařilo. Hodnota v proměnné stav je platná.

Deklarace funkce

```
int PRACKA_teplota_30(P_PRACKA pracka, char * stav);
```

8.3.18 Funkce PRACKA_teplota_40

Funkce slouží ke zjištění stavu čidla teploty prací vody nastaveného na teplotu 40°C.

Vstupní parametry

- pracka: ukazatel na strukturu PRACKA, která popisuje model, se kterým se bude pracovat.
- stav: ukazatel na proměnnou typu char do které bude uložena informace o stavu čidla

Návratová hodnota

- 0: Zjištění stavu čidla se nezdařilo. Hodnota v proměnné stav není platná.
- 1: Zjištění stavu čidla se zdařilo. Hodnota v proměnné stav je platná.

Deklarace funkce

```
int PRACKA_teplota_40(P_PRACKA pracka, char * stav);
```

8.3.19 Funkce PRACKA_teplota_60

Funkce slouží ke zjištění stavu čidla teploty prací vody nastaveného na teplotu 60°C.

Vstupní parametry

- pracka: ukazatel na strukturu PRACKA, která popisuje model, se kterým se bude pracovat.
- stav: ukazatel na proměnnou typu char do které bude uložena informace o stavu čidla

Návratová hodnota

- 0: Zjištění stavu čidla se nezdařilo. Hodnota v proměnné stav není platná.
- 1: Zjištění stavu čidla se zdařilo. Hodnota v proměnné stav je platná.

Deklarace funkce

```
int PRACKA_teplota_60(P_PRACKA pracka, char * stav);
```

8.3.20 Funkce PRACKA_teplota_90

Funkce slouží ke zjištění stavu čidla teploty prací vody nastaveného na teplotu 90°C.

Vstupní parametry

- pracka: ukazatel na strukturu PRACKA, která popisuje model, se kterým se bude pracovat.
- stav: ukazatel na proměnnou typu char do které bude uložena informace o stavu čidla

Návratová hodnota

- 0: Zjištění stavu čidla se nezdařilo. Hodnota v proměnné stav není platná.
- 1: Zjištění stavu čidla se zdařilo. Hodnota v proměnné stav je platná.

Deklarace funkce

```
int PRACKA_teplota_90(P_PRACKA pracka, char * stav);
```

9 PŘÍKLADY POUŽITÍ KNIHOVNY PRO ŘÍZENÍ MODELU

Tato část ukazuje příklady použití knihovny k řízení modelu automatické pračky za pomoci k tomu vyrobených pomůcek. Ukázány jsou dva příklady. První z nich je velmi jednoduchý příklad pro ukázku základního použití knihovny, druhý příklad slouží k demonstraci kompletního řízení modelu s využitím všech funkcí, které model poskytuje.

9.1 Příklad 1

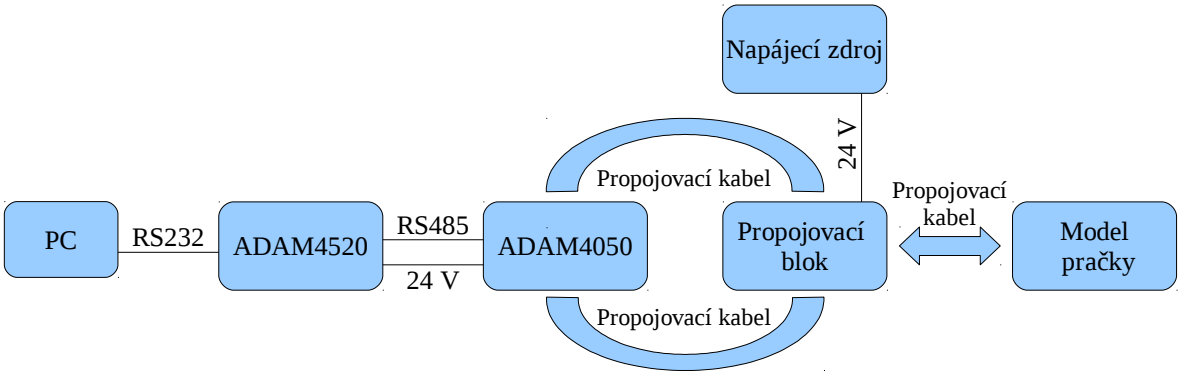
Tento příklad slouží k základní prezentaci práce s knihovnou funkcí pracka.h. Je zde ukázáno, jak s knihovnou pracovat, jak pracovat s modelem a jak práci s modelem ukončit.

9.1.1 Zadání

Napište program pro předvedení základní práce s modelem pračky. Program musí provést správné napuštění vody do pracího prostoru tak, aby nedošlo k jejímu přetečení.

9.1.2 Zapojení modelu

Model bude k počítači připojen podle následujícího obrázku.



Obrázek 20: Blokové schéma zapojení modelu pro příklad 1

9.1.3 Programové řešení příkladu

Vzhledem k určení příkladu jako základní demonstrace pro použití funkcí knihovny pracka.h, nemá tento program velkou váhu při hodnocení prováděné práce modelem. Je zde ale ukázáno jak s knihovnou pracovat a jak pracovat s funkcemi, které knihovna poskytuje. Celý program bude napsán v těle hlavní funkce programu.

9.1.3.1 Hlavní funkce programu

Zde je uveden celý kód bez komentáře či rozboru.

```
#include "pracka.h"
int main(int argc, char * argv[])
{
	P_PRACKA model;
	model = PRACKA_inicializace("COM1\0", 9600, 1);
	if(model == NULL)
	{
		return 0;
	}
	if(!PRACKA_napousteni_zapnout(model))
	{
		PRACKA_zavri(model);
		return 0;
	}
	char snimac;
	do
	{
		if(!PRACKA_voda_50(model, &snimac))
		{
			PRACKA_zavri(model);
			return 0;
		}
	}while (!snimac);
	if(!PRACKA_napousteni_vypnout(model))
	{
		PRACKA_zavri(model);
		return 0;
	}
	PRACKA_zavri(model);
	return 1;
}
```

9.1.3.2 Rozbor programu

Ve všech programech, ve kterých bude použita některá funkce knihovny pracka.h musí být vložen kód této funkce pomocí příkazu #include preprocesoru jazyka C.

```
#include "pracka.h"
```

Hlavní funkce programu má standardní deklaraci.

```
int main(int argc, char * argv[])
{
```

Je nutné vytvořit proměnnou typu P_PRACKA, která bude sloužit jako identifikátor modelu, se kterým se pracuje.

```
P_PRACKA model;
```

První použitá funkce z knihovny musí být inicializační. Tím je zajištěno prověření propojení a nastavení všech částí až po propojovací blok a pokud nebyla zjištěna žádná závada, dojde k jeho aktivaci.

```
model = PRACKA_inicializace("COM1\0", 9600, 1);
```

Následuje testování hodnoty proměnné model, zda neobsahuje hodnotu konstanty NULL. V takovém případě selhala inicializace modelu. Kód zde není uveden. Jestliže inicializace modelu neselhala, Je možné aktivovat některou funkci. V tomto příkladu je to napouštění vody do pracího prostoru. Zároveň je testována návratová hodnota funkce. Ta nese informaci o tom, zda se aktivace napouštění zdařilo, či nikoliv.

```
if(!PRACKA_napousteni_zapnout(model))
```

Pokud se aktivace napouštění nezdařilo, je návratová hodnota volané funkce rovna 0. Je třeba ošetřit chybový stav. V uvedeném příkladě je řešení takovéto. Provedením tohoto bloku bude program ukočen.

```
{
      PRACKA_zavri(model);
      return 0;
}
```

Jestliže aktivace napouštění proběhla bez chyb, bude program pokračovat následujícími řádky. Voda se napouští, je třeba hlídat hladinu vody, aby pračka nepřetekla. Bude třeba číst stav snímače úrovně hladiny 50 %. Informace o stavu tohoto snímače bude ukládána do proměnné vytvořené na tomto řádku.

```
char snimac;
```

Zjišťování stavu čidla bude prováděno cyklicky ve smyčce s testováním podmínky na konci. Hodnota v proměnné snimac po průběhu funkce pro zjištění stavu snímače je 0 nebo 1 v závislosti na stavu snímače. Hodnota této proměnné je použita pro ukončení běhu smyčky. Ošetření chyby při běhu knihovní funkce je vypuštěno.

```
do
```

```
	{
		if(!PRACKA_voda_50(model, &snimac)) { … }
	}while (!snimac);
```

Snímač je aktivní. Smyčka byla ukončena. Zbývá vypnout napouštění vody aby pračka nepřetekla, ukončit práci s modelem a ukončit program. Kód byl již v tomto příkladu jednou vysvětlován, je tedy uveden bez rozboru. Rovněž byla vynechána část pro ošetření chyby při volání knihovní funkce.

```
	if(!PRACKA_napousteni_vypnout(model)) {…}
	PRACKA_zavri(model);
	return 1;
}
```

Konec programu příkladu 1.

9.2 Příklad 2

Tento příklad slouží k ukázce práce s modelem automatické pračky a knihovnou pro jeho řízení.

9.2.1 Zadání

Napište program pro řízení modelu automatické pračky připojeného pomocí modulů ADAM k počítači. Program musí provést tyto operace:

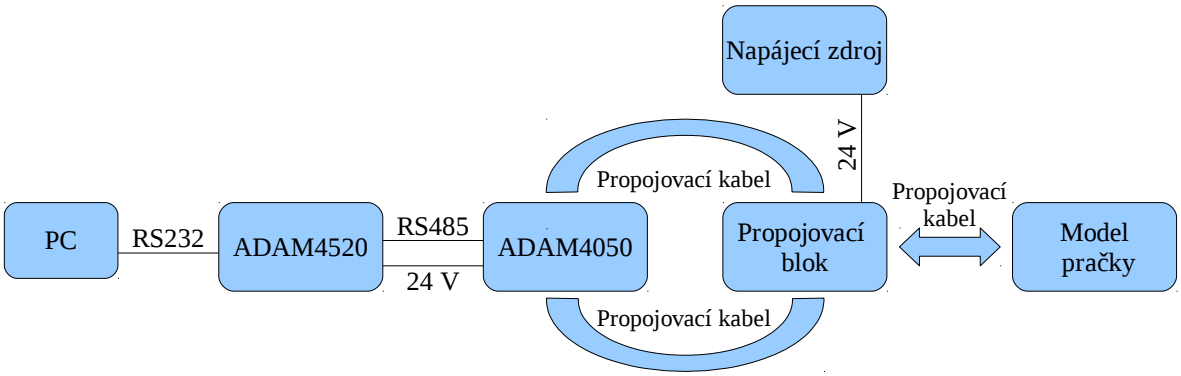
- napustit vodu do pracího prostoru
- ohřát vodu na teplotu 40°C
- provést prací cyklus, při kterém bude udržována nastavená teplota
- vypustit vodu a vyždímat prádlo

Během běhu programu nesmí dojít k indikaci chybového stavu pračky.

K realizaci programu použijte knihovnu pracka.h.

9.2.2 Zapojení modelu

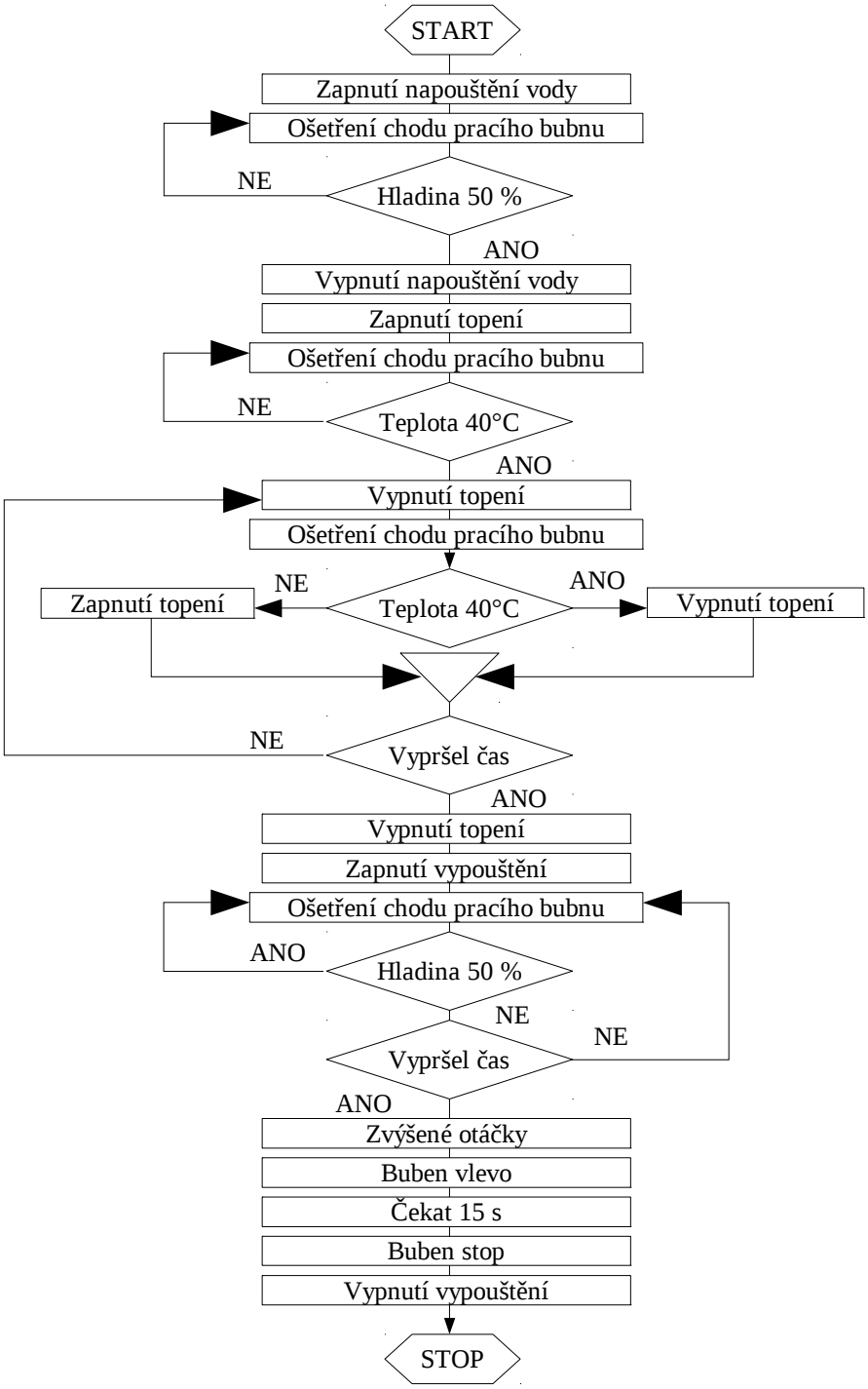
Model bude k počítači připojen pomocí modulů ADAM4050 a ADAM4520. Blokové schéma zapojení je zobrazeno na následujícím obrázku.



Obrázek 21: Blokové schéma zapojení modelu pro příklad 2

9.2.3 Programové řešení příkladu

Tento příklad názorně ukazuje možnosti modelu automatické pračky. Jsou zde využity všechny funkce, které model nabízí. Řízení pračky je provedeno takovým způsobem, aby během praní nedocházelo ke škodlivému řízení chodu motoru pracího bubnu. Základní chod programu tohoto příkladu je možné zapsat tímto vývojovým diagramem.



Obrázek 22: Vývojový diagram programu pro příklad 2

Pro řešení zadaného příkladu je možné postupovat takto. Nejdříve je třeba připojit systémové knihovny. V tomto případě se jedná o knihovnu stdio.h, která poskytuje funkce pro výpis textu na obrazovku. Uvedený příklad používá výpis do standardního chybového kanálu.

```
# include <stdio.h>
```

Dále je třeba připojit knihovnu funkcí pro práci s modelem automatické pračky.

```
#include "pracka.h"
```

9.2.3.1 Pomocná funkce

Pro zjednodušení programu je vhodné si napsat funkci, která pohlídá provoz bubnu pračky, takže nebude docházet k náhlým změnám směru otáčení. Funkce v tomto příkladu se nazývá buben. Má dva vstupní parametry. Model – proměnná typu P_PRACKA, obsahuje odkaz na strukturu popisující model pračky, se kterým se má pracovat, a proměnná ridici_promenna typu int, která obsahuje požadovaný stav pračky.

Deklarace funkce je tato:

```
int buben(P_PRACKA model, int ridici_promenna)
```

Následující kód patří této funkci.

```
{
```

Následující proměnná redukovane_rizeni slouží k jednoduššímu nastavování časových délek jednotlivých pracovních úseků pracího bubnu.

```
    int redukovane_rizeni;
    redukovane_rizeni = ridici_promenna % 31;
```

Ze zadaného požadovaného stavu se počítá modulo všech možných stavů. Hlavní program tedy nebude muset být kvůli změně časování upravován. Volba požadované akce bude vybrána funkcí switch.

```
    switch (redukovane_rizeni)
    {

        case 5 :    return PRACKA_buben_vlevo(model);
        case 15 :   return PRACKA_buben_stop(model);
        case 20 :   return PRACKA_buben_vpravo(model);
        case 30:    return PRACKA_buben_stop(model);
        default:    return 1;
    }
}
```

Z výpisu funkce je vidět, že funkce přímo vrací návratovou hodnotu funkcí z knihovny pracka.h. Všechny funkce v knihovně pracka h, mimo funkci inicializace, vracejí na základě průběhu vlastích instrukcí hodnotu 1, nebo 0. 1 v případě, že funkce proběhla bez chyb,

jinak vrací hodnotu 0. Řešení chybového stavu bude provedeno v hlavní funkci programu. Také zde u jednotlivých větví programu chybí ukončovací break;. V této podobě není ukončovací break; v jednotlivých větvích potřebné.

9.2.3.2 Hlavní funkce – celý kód

Celý kód této funkce zde není uveden. Případný zájemce jej nalezne uveden v přílohách, které jsou nedílnou součásti této práce.

9.2.3.3 Rozbor kódu hlavní funkce – zahájení práce

Hlavní funkce má běžnou deklaraci

```
int main(int argc, char* argv[])
{
```

Pro práci s modelem pračky bude potřeba udržovat ukazatel na strukturu, která obsahuje jeho popis.

```
P_PRACKA pracka;
```

Zahájení práce s modelem. Volaná funkce PRACKA_inicializace prověří komunikační trasu k propojovacímu bloku a v případě úspěchu vrátí ukazatel na strukturu, která obsahuje popis modelu. V opačném případě vrací hodnotu NULL. Parametry předané inicializační funkci znamenají, že model bude ovládán pomocí komunikačního rozhraní COM1 s nastavenou rychlostí komunikace 38400 baudů za sekundu a adresa modulu ADAM4050 na sběrnici RS485, přístupné pomocí tohoto komunikačního rozhraní, je 1. Kód ošetřující neúspěšnou inicializaci zde není uveden.

```
if ((pracka = PRACKA_inicializace("COM1\0", 38400, 1)) == NULL)
{…}
```

Následuje vytvoření dvou proměnných, které budou v programu použity. První je použita pro řízení bubnu, druhá slouží pro získávání stavu čidel modelu.

```
int stav_bubnu = 0;
char stav;
```

9.2.3.4 Rozbor kódu hlavní funkce – napuštění vody

Napouštění vody do pračky se zahájí voláním příslušné funkce. Ošetření chyby při neúspěšném provedení funkce zde není uvedeno.

```
if (!PRACKA_napousteni_zapnout(pracka)) {…}
```

Napouštění vody do pračky bylo právě zahájeno. Následuje kód programové smyčky, ve které je řízena práce bubnu a též zjišťován stav čidla úrovně hladiny 50 %, kterým se smyčka ukončuje. Uvedena je také část, která ukazuje časování programu a řízení práce pracího bubnu. Není zde uveden kód pro ošetření chyb, které mohou vznikat při komunikaci s modelem.

```
do
{
      if (!buben(pracka, stav_bubnu)) {…}
      stav_bubnu++;
      Sleep(500);
      if(!PRACKA_voda_50(pracka, &stav)) {…}
}while (!stav);
```

První část zadání je splněna. Voda je napuštěna. Zbývá vypnout napouštění, aby nedošlo k přetečení pračky.

```
if (!PRACKA_napousteni_vypnout(pracka)) {…}
```

9.2.3.5 Rozbor kódu hlavní funkce – ohřátí vody

Další bod zadání je ohřátí vody na prací teplotu 40°C. Bude proto potřeba zapnout ohřívací těleso a počkat do okamžiku, kdy voda dosáhne požadované teploty. Nejdřív je potřeba zapnout ohřívací těleso.

```
if (!PRACKA_topeni_zapnout(pracka)) {…}
```

Pak opět v cyklu počkat až čidlo teploty prací vody nastavené na hodnotu 40°C začne hlásit dosažení této teploty. Cyklus má podobnou strukturu jako ten předchozí. Je zde uvedena jen část důležitá pro pochopení práce s modelem pomocí funkcí knihovny.

```
do
{
      if (!buben(pracka, stav_bubnu)) {…}
      {…}
      if(!PRACKA_teplota_40(pracka, &stav)) {…}
}while (!stav);
```

Aby nedošlo k velkému překročení teploty prací vody, je třeba vypnout ohřev. Kontrola průběhu funkce byla z ukázky vypuštěna.

```
if (!PRACKA_topeni_vypnout(pracka)) {…}
```

9.2.3.6 Rozbor kódu hlavní funkce – prací cyklus

Nyní bude zahájen prací cyklus. Oproti předchozím cyklům má tento definovanou dobu trvání. Pro jeho realizaci tedy bude použito funkce for. Délka cyklu je nastavena na dobu přibližně 3 minuty. Délka cyklu je závislá na časovací prodlevě, která není v ukázce patrná, ale v úplném kódu je možné tyto časovací prodlevy najít. V cyklu je zjišťován stav snímače teploty prací vody 40°C a podle jeho stavu je zapínáno nebo vypínáno ohřívání vody. Časování a ošetření chyb při komunikaci s modelem je z ukázky kódu vypuštěno.

Nejdříve je potřeba vytvořit proměnnou, kterou bude cyklus for řízen. Její název je prani.

```
int prani;
```

Zahájení cyklu praní s požadovanou dobou praní asi 3 minuty.

```
for(prani = 0; prani < 360; prani++)
{
      if (!buben(pracka, stav_bubnu)) {…}
      {…}
      if(!PRACKA_teplota_40(pracka, &stav)) {…}
      if(stav)
      {
            if(!PRACKA_topeni_vypnout(pracka)) {…}
      }
      else
      {
            if(!PRACKA_topeni_zapnout(pracka)) {…}
      }
}
```

Prací cyklus byl proveden. Jelikož se v jeho těle měnil stav ohřívacího tělesa a nyní není jisté, v jakém je stavu, je proto vhodné toto těleso vypnout.

```
if(!PRACKA_topeni_vypnout(pracka)) {…}
```

9.2.3.7 Rozbor kódu hlavní funkce – vypuštění vody

Nyní proběhne vypuštění vody z pracího prostoru pračky. Model pračky neposkytuje informaci, zda je v pracím prostoru nějaká voda, pokud jí není aspoň do poloviny pracího prostoru. Aby byla vypuštěna všechna voda, je potřeba nechat vypouštění zapnuto déle, než je doba indikovaná čidlem pro detekci úrovně hladiny 50 %. Pro tento účel bude vytvořena proměnná s názvem prodluz, která bude sloužit k prodloužení vypouštěcího cyklu o dobu

potřebnou k vypuštění všechny vody z pracího prostoru. Hodnota uložená v této proměnné určuje počet základních časovacích cyklů, o které se vypouštěcí cyklus prodlouží.

```
int prodluz = 20;
```

Před samotným vypouštěcím cyklem bude zahájeno vypouštění vody.

```
if(!PRACKA_vypousteni_zapnout(pracka)) {…}
```

Zde vypouštěcí cyklus začíná. Jeho tvar je podobný prvním dvěma cyklům, kromě ukončovací podmínky, která je závislá na hodnotě proměnné prodlužující běh tohoto cyklu.

```
do
{
      if (!buben(pracka, stav_bubnu)) {…}
      {…}
      if(!PRACKA_voda_50(pracka, &stav)) {…}
      if(!stav)
            prodluz--;
}while (prodluz);
```

Vypouštěcí cyklus je dokončen. Pračka je vypuštěna.

9.2.3.8 Rozbor kódu hlavní funkce – ždímání prádla

Zbývá poslední úkol zadání. Vyždímat prádlo. Jelikož během ždímání se z prádla uvolňuje voda, bude vypouštění ponecháno zapnuto i při ždímání.

Ždímací část programu, na rozdíl od předcházející, nepoužívá k řízení bubnu společné proměnné stav_bubnu a funkce buben. Tím bude zajištěno, že se buben bude otáčet tak, jak má a nebude se během ždímání zastavovat či měnit směr. Proto je potřeba jej nejdříve zastavit.

```
if(!PRACKA_buben_stop(pracka)) {…}
```

Po zastavení bubnu následuje prodleva jedné sekundy, než bude buben opět uveden v provoz.

```
Sleep(1000);
```

Zde dojde k roztočení bubnu směrem vlevo. Nejdříve při normálních otáčkách.

```
if(!PRACKA_buben_vlevo(pracka)) {…}
```

Normální otáčky pracího bubnu budou trvat jednu sekundu.

```
Sleep(1000);
```

Pak budou otáčky pracího bubnu zvýšeny.

```
if(!PRACKA_otacky_zvysene(pracka)) {…}
```

Zvýšené otáčky pracího bubnu budou trvat přibližně patnáct sekund. Během této doby bude prádlo vyždímáno a přebytečná voda vypouštěna. Vypouštěcí čerpadlo je stále v provozu.

```
Sleep(15000);
```

Po této době budou otáčky sníženy na normální hodnotu.

```
if(!PRACKA_otacky_normalni(pracka)) {…}
```

A pak se opět ještě jednu vteřinu počká, aby se otáčky pracího bubnu plynule snížily.

```
Sleep(1000);
```

Nakonec bude prací buben zastaven.

```
if(!PRACKA_buben_stop(pracka)) {…}
```

9.2.3.9 Rozbor kódu hlavní funkce – ukončení procesu

Vypouštěcí čerpadlo bude ponecháno v chodu o dvě vteřiny déle, čímž dojde k vypouštění i posledních zbytků vody z pracího prostoru.

```
Sleep(2000);
```

Nakonec bude i vypouštěcí čerpadlo vypnuto a praní je u konce.

```
if(!PRACKA_vypousteni_vypnout(pracka)) {…}
```

Následuje korektní ukončení práce s modelem a zakončení programu s návratovou hodnotou, která bude znamenat úspěšné splnění úkolu.

```
PRACKA_zavri(pracka);
return 1;
}
```

Konec programu.

Kompletní kód programu je uveden v přílohách.

ZÁVĚR

Cílem diplomové práce bylo vytvořit knihovnu funkcí pro zjednodušení práce s různými moduly řady ADAM4000 firmy Advantech a navrhnout a realizovat propojení modelu technologického zařízení a průmyslového počítače za pomoci těchto modulů. K takto připojenému modelu též vytvořit vhodné nástroje pro programové řízení v uživatelských programech.

Knihovna funkcí pro práci s moduly řady ADAM4000 byla napsána pro moduly ADAM4050 a ADAM4017, které byly pro tuto část práce vybrány.

Pro realizaci připojení byl vytvořen propojovací blok, který nahrazuje původní připojovací zařízení dodávané k modelu a zároveň zjednodušuje sestavení celé propojovací trasy. Knihovna funkcí byla doplněna o funkce, které zjednodušují řízení tohoto modelu připojeného pomocí vytvořeného propojovacího bloku. Pro ukázku práce s knihovnou byly vytvořeny příklady řízení, kterými je demonstrováno základní použití i komplexní řízení činnosti modelu. Byl též vypracován návod k použití, který popisuje všechny důležité kroky při připojování modelu k počítači i způsob použití knihovny funkcí. Tím bylo dosaženo všech cílů stanovených v zadání této práce.

ZÁVĚR V ANGLIČTINĚ

Goal of this work was to build a library of functions for simplification of work with various modules of the Advantech ADAM4000 series and to design and realize connection of models of technological devices and industrial computer by these modules. For model, connected to industrial PC by this way, build proper tools for program control in user's programs.

The library of functions for working with modules of series ADAM4000 was written for modules ADAM4050 and ADAM4017 which were selected for this part of the work.

For realization of the connection of washing machine model to industrial PC an adapter unit was built, which replaces the original one supplied with the model and simplifies the assembly of the whole connection. The library was supplemented with functions for controlling this model connected with this adapter unit. For illustration of work with this library two examples were created. One shows the basic usage of the library and the second gives an example of complete control of the model. For description of connecting model to PC and using library of functions User's manual was written. By this all aims, defined for this thesis, were reached.

SEZNAM POUŽITÉ LITERATURY

[1] International Organization for Standardization: ISO/IEC 9899:1999 Programming languages -- C, [online], International Organization for Standardization, 1999. [citace: 10.2.2010]. Dostupný z: <http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=29237>

[2] Robertson Bayer: Windows Serial Port Programming, [online], 2008. [citace: 10.2.2010]. Dostupný z: <http://www.robbayer.com/files/serial-win.pdf>

[3] Modbus Organization: MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b, [online], Modbus Organization, 2006. [citace: 10.2.2010]. Dostupný z: <http://www.modbus-ida.org/docs/Modbus_Application_Protocol_V1_1b.pdf>

[4] ADAM 4000 Series data Acquisition Modules User's Manual, [online], Advantech Co., Ltd, 2008. [citace: 10.2.2010]. Dostupný z: <http://downloadt.advantech.com/download/downloadsr.aspx?File_Id=GF-22KNC>

[5] Pavel Černohorský: Historie a vývoj jazyka C (od C až po C#), [online], Fakulta informatiky Masarykovy univerzity, 2003. [citace: 10.2.2010]. Dostupný z: <http://www.fi.muni.cz/usr/jkucera/pv109/2003p/xcernoh1.htm>

[6] František Hruška: Technické prostředky informatiky a automatizace, Univerzita Tomáše Bati ve Zlíně, 2007. s. ISBN: 978-80-7318-535-0

[7] Jan Staněk, Jan Řehák: RS 485 & 422, [online], HW server s.r.o., 1998. [citace: 10.2.2010]. Dostupný z: <http://hw.cz/Teorie-a-praxe/Dokumentace/ART821-RS-485-422.html>

[8] Pavel Herout: Učebnice jazyka C, Kopp nakladatelství, 2005. 271s. ISBN: 80-7232-220-6

[9] VLACH Jaroslav, VLACHOVÁ Viktorie: Počítačová rozhraní přenos dat a řídicí systémy, Praha: BEN - technická literatura, 2002. 176s. ISBN: 80-7300-010-5

[10] VLACH Jaroslav: Řízení a vizualizace technologických procesů, Praha: BEN - technická literatura, 2002. 160s. ISBN: 9788086056661

[11] PRATA Stephen: Mistrovství v C++, Praha: Computer press, 2007. 1120s. ISBN: 9788025117491

[12] PPC-L60T VIA Eden Processor based Fanless Panel PC with 6.4" TFT-LCD Users Manual, [online], Advantech Co., Ltd, 2005. [citace: 10.2.2010]. Dostupný z: <http://downloadt.advantech.com/download/downloadsr.aspx?File_Id=1-1KCPDM>

[13] KOHOUT Luděk: Učební pomůcky EDU-mod, [online], , 2008. [citace: 10.2.2010]. Dostupný z: <http://www.edumat.cz/produkty.php?produkt=edumod>

[14] PINKER, J.: Mikroprocesory a Mikropočítače, Praha: BEN – technická literatura, 2004. 220s. ISBN: 80-7300-110-1

[15] MATOUŠEK, David: Číslicová technika, Praha: BEN – technická literatura, 2002. 208s. ISBN: 80-7300-025-3

[16] MARTÍNEK Radislav: Senzory v průmyslové praxi, Praha: BEN – technická literatura, 2004. 200s. ISBN: 80-7300-114-4

[17] NEVŘIVA Pavel: Analýza signálů a soustav, Praha: BEN – technická literatura, 2002. 670s. ISBN: 80-7300-004-0

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

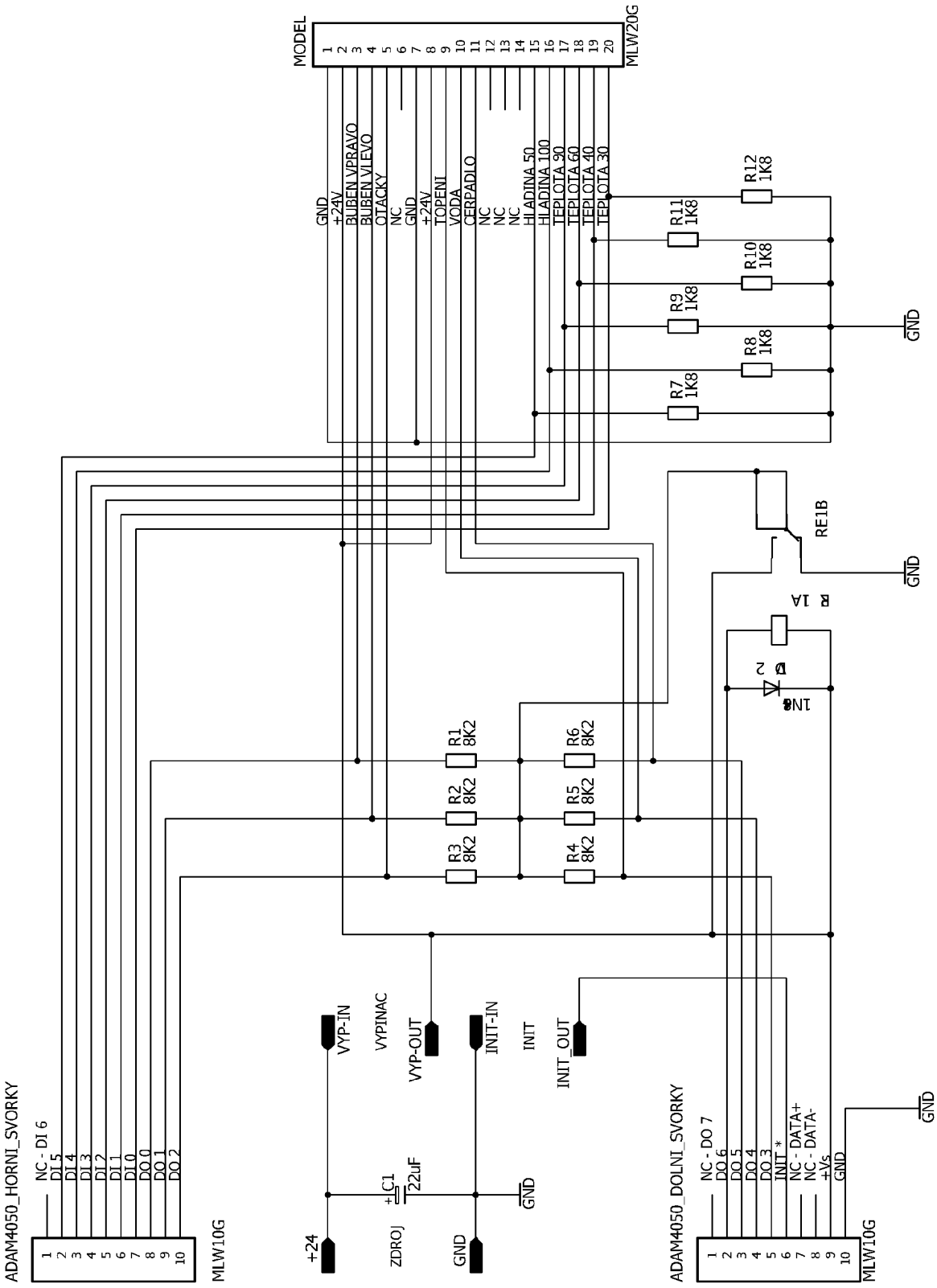
ADAM	Komunikační protokol používaný pro řízení modulů firmy Advantech.
ADAM4017	Typové označení modulu analogových vstupů.
ADAM4050	Typové označení vstupně výstupního modulu.
ADAM4520	Typové označení převodníku RS232 – RS485 / RS422.
ADU	Application Data Unit. Označení specifické části zprávy protokolu Modbus.
ANSI	American National Standard Institute – http://www.ansi.org
C	Označení programovacího jazyka.
C++	Označení programovacího jazyka.
CAN	Přenosový protokol vyvinutý firmou Robert Bosh GmbH specifikovaný normou ISO 11898.
COMx	Označení sériových komunikačních rozhraní v operačním systému Windows. Písmeno x ve zkratce zastupuje číslici, která udává pořadové číslo komunikačního rozhraní v operačním systému.
GPIB	General Purpose Interface Bus. Komunikační sběrnice určená pro připojování měřicích a testovacích systémů k PC.
ISO	International Oraganization for Standardization – http://www.iso.org
Modbus	Přenosový protokol používaný na sběrnicích typu RS485. Popsán v IEC PAS 62030.
NULL	Konstanta v jazyce C. Vyjadřuje hodnotu ukazatele, který nemá platnou adresu – neukazuje na použitelnou oblast.
PC	Personal Computer – Osobní počítač.
PDU	Protocol Data Unit. Označení specifické části zprávy protokolu Modbus.
PLC	Programmable Logical Controller. Programovatelný logický automat.
PROFIBUS	Přenosový protokol standardizovaný normou IEC 61158/EN 50170.
RS232	Komunikační linka sériového přenosu dat schopná propojit pouze dvě zařízení.
RS485	Dvouvodičová (třívodičová) komunikační sběrnice pro průmyslová prostředí.
UNIX	Víceúlohový víceuživatelský operační systém firmy AT&T.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: SCHÉMA ZAPOJENÍ PROPOJOVACÍHO BLOKU



PŘÍLOHA P II: ROZPIS SOUČÁSTEK PROPOJOVACÍHO BLOKU

Součástky propojovacího bloku

Označení	Hodnota	Typ
Krabička WEB1001		U-87.080.0053.0
ADAM4050_DOLNI_SVORKY	PSL10	PSL10
ADAM4050_HORNI_SVORKY	PSL10	PSL10
MODEL	PSL20	PSL20
VYPINAC		P-B070B
INIT		P-B070B
C1	22uF	E-22U/50
R1	8K2	TR245S
R2	8K2	TR245S
R3	8K2	TR245S
R4	8K2	TR245S
R5	8K2	TR245S
R6	8K2	TR245S
R7	1K8	TR245S
R8	1K8	TR245S
R9	1K8	TR245S
R10	1K8	TR245S
R11	1K8	TR245S
R12	1K8	TR245S
RE1		RELEM3S24T
VD2	1N4148	LL4148
+24V		Svorka přístrojová
GND		Svorka přístrojová

Součástky propojovacího kabelu (2×)

Kabel plochý desetižilový		
Konektor		PFL10

Součástky napájecího kabelu

Dvoulinka 1,5 m		
banánek 4×		

Součástky datového kabelu

Dvoulinka		
Rezistor 2×	120	

PŘÍLOHA P III: ZDROJOVÝ KÓD PŘÍKLADU 2

```
/*
    Priklad_2.cpp : Ukazkovy priklad pouziti knihovny pro ovladani modelu
                    pracky. Napsano jako soucast diplomove prace na tema

                    Propojení průmyslového PC s modely technologických procesů

    Napsal: Bc. Premysl Vrba
    Dne:    10.4.2010
*/

/* Pouzita bude systemova knihovna */
# include <stdio.h>

/* Priklad pouziva knihovnu funkci pracka.h, ktera poskytuje funkce pro praci
   s modelem automaticke pracky.
*/

#include "pracka.h"

/* Pomocna funkce programu
   Slouzi k rizeni smeru bubnu pracky. */

int buben(P_PRACKA model, int ridici_promenna)
{
    /* Rizeni smeru bubnu pracky je dano hodnotou ridici promenne. */
    /* Hodnota rizeni bude upravena na potrebny rozsah. */
    int redukovane_rizeni;
    redukovane_rizeni = ridici_promenna % 31;

    switch (redukovane_rizeni)
    {

        case 5 : /* Zapnout smer doleva */
                 return PRACKA_buben_vlevo(model);
        case 15 : /* Vypnout otaceni bubnem */
                 return PRACKA_buben_stop(model);
        case 20 : /* Zapnout smer doprava */
                 return PRACKA_buben_vpravo(model);
        case 30: /* Vypnout otaceni bubnem */
                 return PRACKA_buben_stop(model);
        default: /* v poradku */
                 return 1;
    }
}

/* Hlavni funkce programu */
int main(int argc, char* argv[])
{
/* Promenna, ve ktere bude ulozen ukazatel na strukturu nesouci informace
   o modelu pracky */
P_PRACKA pracka;

/* Zahajeni prace s modelem.
   Pripojeni je provedeno pomoci komunikacniho rozhrani COM1
   komunikacni rychlost modulu ADAM4050 je nastavena na 38400 baudu/s
   adresa modulu ADAM4050 na sbernici RS485 je 1
*/
    if ((pracka = PRACKA_inicializace("COM1\0", 38400, 1)) == NULL)
/* pokud se nezdari praci s modelem zahajit, vypis chybove hlaseni
       a ukonci program */
    {
        fprintf(stderr, "ERR: Nepodarilo se zahajit praci s modelem.\n");
```

```
        return 0;
    }
/* Prace s modelem byla zahajena. */

/* Promennou stav_bubnu bude rizen chod bubnu. */
    int stav_bubnu = 0;

/* Promenna stav bude slouzit k ukladani informace o stavech cidel modelu. */
    char stav;

/* Zapni napousteni vody do pracky */
    if (!PRACKA_napousteni_zapnout(pracka))
/* Nepodarilo se zapnout napousteni vody do pracky. */
    {
        fprintf(stderr, "ERR: Nepodarilo se zapnout napousteni pracky.\n");
/* Je treba korektne ukoncit praci s modelem */
        PRACKA_zavri(pracka);
/* a ukoncit program */
        return 0;
    }

/* Cyklus napousteni vody do pracky */
    do
    {
/* Nastav smer bubnu */
        if (!buben(pracka, stav_bubnu))
/* funkce obsluhujici buben vraci chybovou hodnotu - selhalo nastavovani
   bubnu */
        {
            fprintf(stderr, "ERR: Selhalo nastaveni smeru bubnu.\n");
            PRACKA_zavri(pracka);
            return 0;
        }

/* Zvys hodnotu ridici promenne bubnu. */
        stav_bubnu++;

/* Vyckej pul sekundy */
        Sleep(500);

/* Nacti stav cidla urovne hladiny 50 % */
        if(!PRACKA_voda_50(pracka, &stav))
/* Nezdarilo se zjistit stav cidla urovne hladiny. Ukonci program. */
        {
            fprintf(stderr, "ERR: Selhalo cteni cidla urovne hladiny 50 %.\n");
            PRACKA_zavri(pracka);
            return 0;
        }
/* Telo cyklu je u konce. */
/* Pokud bude stav == 0, cidlo neni aktivni, podminka bude TRUE a telo
   cyklu bude opakovano. */
    }while (!stav);

/* Voda je napustena. Vypni napousteni. */
    if (!PRACKA_napousteni_vypnout(pracka))
/* Nepodarilo se vypnout napousteni vody do pracky. */
    {
        fprintf(stderr, "ERR: Nepodarilo se vypnout napousteni pracky.\n");
        PRACKA_zavri(pracka);
        return 0;
    }

/* Zapni ohrev vody na 40 C */
    if (!PRACKA_topeni_zapnout(pracka))
/* Nepodarilo se zapnout ohrev vody. */
```

```
    {
        fprintf(stderr, "ERR: Nepodarilo se zapnout ohrev vody.\n");
        PRACKA_zavri(pracka);
        return 0;
    }

/* Cyklus ohrevu vody */
    do
    {
/* Nastav smer bubnu */
        if (!buben(pracka, stav_bubnu))
/* funkce obsluhujici buben vraci chybovou hodnotu - selhalo nastavovani
   bubnu */
        {
            fprintf(stderr, "ERR: Selhalo nastaveni smeru bubnu.\n");
            PRACKA_zavri(pracka);
            return 0;
        }

/* Zvys hodnotu ridici promenne bubnu. */
        stav_bubnu++;

        /* Vyckej pul sekundy */
        Sleep(500);

/* Nacti stav cidla teploty vody 40 C */
        if(!PRACKA_teplota_40(pracka, &stav))
/* Nezdarilo se zjistit stav cidla teploty vody 40 C. Ukonci program. */
        {
            fprintf(stderr, "ERR: Selhalo cteni cidla teploty vody 40 C.\n");
            PRACKA_zavri(pracka);
            return 0;
        }
/* Telo cyklu je u konce. */
    }while (!stav);
/* Voda je ohrata. Vypni ohrev. */
    if (!PRACKA_topeni_vypnout(pracka))
/* Nepodarilo se vypnout ohrev. */
    {
        fprintf(stderr, "ERR: Nepodarilo se vypnout ohrev vody.\n");
        PRACKA_zavri(pracka);
        return 0;
    }

/* Praci cyklus. */
/* Voda je napustena a ohrata. Ted bude probihat praci cyklus,
   ve kterem bude kontrolovan stav cidla teploty vody a pokud
   teplota poklesne, bude voda dohrata.
   Tento cyklus ma pevny pocet kroku.
*/

/* Promenna, ktera bude slouzit k pocitani cyklu prani. */
    int prani;
/* Jeden cyklus bude trvat priblizne pul sekundy. Praci cyklus bude mit
   360 cyklu = asi 3 minuty. */
    for(prani = 0; prani < 360; prani++)
    {
/* Nastav smer bubnu */
        if (!buben(pracka, stav_bubnu))
/* funkce obsluhujici buben vraci chybovou hodnotu - selhalo nastavovani bubnu
*/
        {
            fprintf(stderr, "ERR: Selhalo nastaveni smeru bubnu.\n");
            PRACKA_zavri(pracka);
            return 0;
```

```
        }

/* Zvys hodnotu ridici promenne bubnu. */
        stav_bubnu++;

/* Vyckej pul sekundy */
        Sleep(500);

/* Nacti stav cidla teploty vody 40 C */
        if(!PRACKA_teplota_40(pracka, &stav))
/* Nezdarilo se zjistit stav cidla teploty vody 40 C. Ukonci program. */
        {
            fprintf(stderr, "ERR: Selhalo cteni cidla teploty vody 40 C.\n");
            PRACKA_zavri(pracka);
            return 0;
        }

/* Otestuj stav cidla */
        if(stav)
/* voda je ohrata - vypni topeni */
        {
            if(!PRACKA_topeni_vypnout(pracka))
/* Chyba pri vypinani topeni */
            {
                fprintf(stderr, "ERR: Selhalo vypinani ohrevu vody.\n");
                PRACKA_zavri(pracka);
                return 0;
            }
        }
        else
/* voda je studena - prihrej */
        {
            if(!PRACKA_topeni_zapnout(pracka))
     /* Chyba pri zapinani topeni */
            {
                fprintf(stderr, "ERR: Selhalo zapinani ohrevu vody.\n");
                PRACKA_zavri(pracka);
                return 0;
            }
        }
/* Telo cyklu prani je ukonce. */
     }

/* Pro jistotu vypni topeni */
     if(!PRACKA_topeni_vypnout(pracka))
/* Chyba pri vypinani topeni */
     {
         fprintf(stderr, "ERR: Selhalo vypinani ohrevu vody.\n");
         PRACKA_zavri(pracka);
         return 0;
     }

/* Cyklus vypousteni vody. */
/* Model neposkytuje informaci o tom, zda je v pracim prostoru nejaka voda
   pod urovni 50 %. Vypusteni veskere vody je tedy dilo casovani a znalosti
   rychlosti vypousteni vody a mnozstvim vody v pracim prostoru. */

/* Promenna, ktera bude ridit prodlouzeni cyklu prani.
   Jeji hodnota vyjadruje pocet pulvterin. */
     int prodluz = 20;

/* Zapni vypousteni vody z pracky */
     if(!PRACKA_vypousteni_zapnout(pracka))
/* Nepodarilo se zapnout vypousteni vody */
     {
```

```
        fprintf(stderr, "ERR: Nepodarilo se zapnout vypousteni vody.\n");
        PRACKA_zavri(pracka);
        return 0;
    }

    do
    {
/* Nastav smer bubnu */
        if (!buben(pracka, stav_bubnu))
/* funkce obsluhujici buben vraci chybovou hodnotu - selhalo nastavovani bubnu
*/
        {
            fprintf(stderr, "ERR: Selhalo nastaveni smeru bubnu.\n");
            PRACKA_zavri(pracka);
            return 0;
        }

/* Zvys hodnotu ridici promenne bubnu. */
        stav_bubnu++;

/* Vyckej pul sekundy */
        Sleep(500);

/* Nacti stav cidla urovne hladiny vody 50 %. */
        if(!PRACKA_voda_50(pracka, &stav))
/* Nezdarilo se zjistit stav cidla urovne hladiny vody 50 %. */
        {
            fprintf(stderr, "ERR: Selhalo cteni cidla urovne hladiny 50 %.\n");
            PRACKA_zavri(pracka);
            return 0;
        }
        if(!stav)
/* voda je vypustena - odecti pulvterinu */
        prodluz--;

/* Dokud neni v prodluz 0, opakuj. */
    }while (prodluz);

/* Cyklus vypousteni je ukonce. Ted je treba vyzdimat. */
/* Zastav buben - predchazeni cuknuti bubnem. */
    if(!PRACKA_buben_stop(pracka))
/* Nepodarilo se vypnout buben */
    {
        fprintf(stderr, "ERR: Nepodarilo se vypnout otaceni bubnem.\n");
        PRACKA_zavri(pracka);
        return 0;
    }

/* pockej sekundu */
    Sleep(1000);

/* Zapni otaceni doleva. */
    if(!PRACKA_buben_vlevo(pracka))
/* Nepodarilo se zapnout otaceni bubnem doleva */
    {
        fprintf(stderr, "ERR: Nepodarilo se zapnout otaceni bubnem doleva.\n");
        PRACKA_zavri(pracka);
        return 0;
    }

/* Pockej sekundu */
    Sleep(1000);

/* Zapni zvysene otacky */
    if(!PRACKA_otacky_zvysene(pracka))
```

```
/* Nepodarilo se zapnout zvysene otacky */
    {
        fprintf(stderr, "ERR: Nepodarilo se zapnout zvysene otacky.\n");
        PRACKA_zavri(pracka);
        return 0;
    }

/* Pockej 15 sekund. */
    Sleep(15000);

/* Vypni zvysene otacky */
    if(!PRACKA_otacky_normalni(pracka))
/* Nepodarilo se vypnout zvysene otacky */
    {
        fprintf(stderr, "ERR: Nepodarilo se vypnout zvysene otacky.\n");
        PRACKA_zavri(pracka);
        return 0;
    }

/* Pockej sekundu. */
    Sleep(1000);

/* Zastav buben */
    if(!PRACKA_buben_stop(pracka))
/* Nepodarilo se vypnout buben */
    {
        fprintf(stderr, "ERR: Nepodarilo se vypnout otaceni bubnem.\n");
        PRACKA_zavri(pracka);
        return 0;
    }

/* Pockej dve sekundy. */
    Sleep(2000);

/* Konecne zastav cerpadlo. */
    if(!PRACKA_vypousteni_vypnout(pracka))
/* Nepodarilo se vypnout vypousteni */
    {
        fprintf(stderr, "ERR: Nepodarilo se vypnout vypousteni vody.\n");
        PRACKA_zavri(pracka);
        return 0;
    }

/* Ukonci praci s modelem */
    PRACKA_zavri(pracka);

    return 1;
}
```