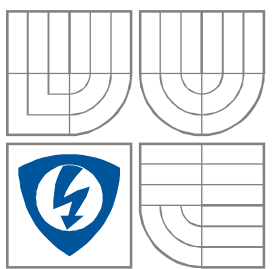


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



**FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH
TECHNOLOGIÍ**

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

KOMUNIKACE PLC S PERIFERIEMI MODELU VÝTAHU

COMMUNICATION BY PLC WITH PERIPHERIES OF MODEL LIFT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

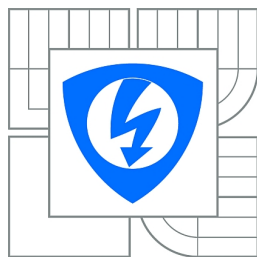
AUTOR PRÁCE
AUTHOR

STANISLAV GORYL

VEDOUČÍ PRÁCE
SUPERVISOR

Ing. RADEK ŠTOHL, Ph.D.

BRNO 2012



VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student: Stanislav Goryl

ID: 125431

Ročník: 3

Akademický rok: 2011/2012

NÁZEV TÉMATU:

Komunikace PLC s periferiemi modelu výtahu

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s modelem výtahu, řídicím PLC a sítí AS-Interface.
2. Navrhněte stavový automat ovládače patra a kabiny.
3. Realizujte software periferií - ovládač patra a ovládač kabiny.
4. Ověřte funkčnost ovládačů pomocí PLC.

DOPORUČENÁ LITERATURA:

Becker, R. a kol.: AS-Interface, řešení pro automatizaci. AS-International Association, 2004, 184 s. ISBN 80-214-2958-5

Logix5000 Controllers General Instructions (Reference Manual). Milwaukee: Rockwell Automation, Inc. 2008.

Dle vlastního literárního průzkumu a doporučení vedoucího práce.

Termín zadání: 6.2.2012

Termín odevzdání: 28.5.2012

Vedoucí práce: Ing. Radek Štohl, Ph.D.

Konzultanti bakalářské práce:

doc. Ing. Václav Jirsík, CSc.

Předseda oborové rady

UPOZORNĚNÍ:

Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

Abstrakt

Tématem bakalářské práce bylo vytvořit firmwarové vybavení pro ovládače pater a kabiny pro model výtahu. Ovládače komunikují s nadřazeným řídicím systémem pomocí průmyslové sběrnice AS-Interface. Firmware je realizován pomocí stavového automatu. V úvodu práce je krátké seznámení se sběrnicevým systémem AS-Interface a nastínění principu přenosu dat. Následuje popis firmwaru pro ovládače. Dále je vytvořena i testovací aplikace pro PLC, která ověřuje funkčnost ovládačů. V poslední části je vytvořen návod jak firmware naprogramovat, včetně konstrukce netypicky zapojeného programovacího kabelu.

Klíčová slova

Komunikace modelu výtahu, AS-Interface, Stavový automat, Ovládač patra, Ovládač kabiny

Abstract

The theme of this bachelor thesis was to create firmware for internal and external control panel for a model of a lift. Control panels communicate with the upper-level control system via the AS-Interface fieldbus. The base of the firmware is the finite state machine. In the introduction of the thesis, there is a brief description of the AS-Interface bus and the outline of the principle of data transfer. Followed out, is the description of firmware for control panels. Furthermore, there was created a test application for PLC which verifies the functionality of control panels. In the last part, there was produced a manual for the firmware programming, including the construction of untypically involved programming cable.

Keywords

Communication of the lift model, AS-Interface, Finite-state machine, External control panel, Internal control panel

Bibliografická citace:

GORYL, S. *Komunikace PLC s periferiemi modelu výtahu*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2012. 60 s. Vedoucí bakalářské práce byl Ing. Radek Štohl, Ph.D.

Prohlášení

„Prohlašuji, že svou bakalářskou práci na téma „Komunikace PLC s periferiemi modelu výtahu“ jsem vypracoval samostatně pod vedením vedoucího diplomové bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **24. května 2012**

.....
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Radku Štohlovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: **24. května 2012**

.....
podpis autora

Obsah

1	Úvod	9
2	Systém AS-Interface.....	10
2.1	Krátce o AS-Interface	10
2.2	Vlastnosti AS-Interface.....	10
2.3	Základní komponenty AS-Interface.....	10
2.3.1	AS-Interface sběrnice.....	10
2.3.2	AS-Interface master	11
2.3.3	AS-Interface slave.....	11
2.3.4	Možné topologie	12
2.4	Systém komunikace.....	12
2.5	Komunikační profil.....	14
3	Komunikace modelu výtahu	16
3.1	Základní zapojení komunikace.....	16
3.2	Popis komunikace.....	17
4	Ovladač patra	18
4.1	Popis panelu patra.....	18
4.2	Stavový popis ovladače patra.....	18
4.3	Popis programu ovladače patra	20
4.4	Popis možného zobrazení na displeji.....	24
5	Ovladač kabiny.....	25
5.1	Popis panelu kabiny	25
5.2	Stavový popis ovladače kabiny	26
5.3	Popis programu ovladače kabiny.....	28
5.4	Popis zobrazení na displeji v kabině.....	35
6	Testovací program pro PLC.....	36
6.1	Komunikace a slavy.....	36
6.2	Důležité proměnné.....	37
6.3	Program pro PLC.....	38
6.4	Snímač dveří.....	47
6.5	Implementace programu pro řízení výtahu	48
7	Návod k programování mikroprocesoru	50
7.1	Programovací kabel	50
7.2	Kompilátor Keil.....	50

7.3	Flip.....	52
8	Závěr.....	55

1 ÚVOD

Úkolem této bakalářské práce je se nejprve seznámit s modelem výtahu, který byl vytvořen studenty. Prostudovat jak má fungovat komunikace mezi PLC a jednotlivými ovládacími panely. Pro tuto komunikaci navrhnout stavový automat. Promyslet firmware pro ovládače kabiny a patra. Dále naprogramovat firmware pro tyto ovládače. Ověření komunikace bude pomocí testovacího programu v PLC.

Komunikaci řídí master PLC Allen-Bradley po sběrnici AS-Interface pomocí modulu firmy Bihl+Wiedemann. Panely jsou do sítě AS-Interface zapojeny přes digitální slavy. Další slavy jsou použity pro snímání zatížení a ovládání motoru výtahu. Ovládací panely jsou řízeny mikroprocesorem AT89C51ED2 a budou programovány v jazyce C. Testovací program, který má ověřit správnost komunikace a bude vytvořen ve vývojovém prostředí RSLogix 5000. Program se nebude zabývat řízením výtahu jako takového, bude pouze zajišťovat jeho komunikaci.

2 SYSTÉM AS-INTERFACE

2.1 Krátce o AS-Interface

V roce 1990 jedenáct velkých společností společně formulovali sběrniceový systém „Actuator/Sensor Interface“ zkráceně AS-Interface[2], je označení pro systém rozhraní aktuátorů a senzorů. Po uvedení na trh v roce 1994 se stal široce používaným fieldbus systémem.

Tato technologie představuje jednoduchý systém pro uživatele. Jedná se o otevřený systém, který se dá použít velice univerzálně. Jednoduše jej lze připojit k PLC (programovatelnému automatu) nebo k nadřazenému systému.

2.2 Vlastnosti AS-Interface

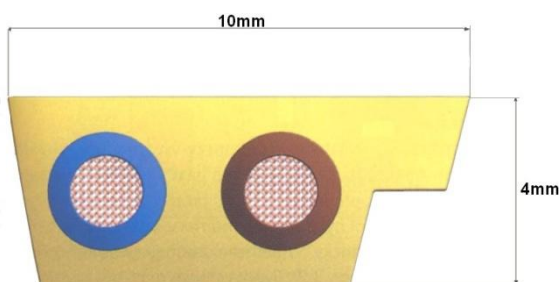
Základní parametry sítě AS-Interface [2]:

- Napájení i data na jednom kabelu
- Master–Slave komunikace
- Maximální počet slave zařízení 62
- Maximální vzdálenost Master-Slave 100m (300m s opakovačem)
- Maximální vzdálenost Slave-Slave 100m (500m s opakovačem)
- Doba cyklu ≤ 5 ms pro 31 zařízení, ≤ 10 ms pro 62 zařízení
- Pracovní teplota $-25 - 75^{\circ}\text{C}$
- Krytí IP20 až IP69
- Použití napětí PELV
- Prakticky libovolná topologie

2.3 Základní komponenty AS-Interface

2.3.1 AS-Interface sběrnice

Standardně se používá dvoužilový profilový kabel, který je nekroucený, nestíněný a bez impedančního ukončení. Kabel se vyrábí ve dvou barvách. Žlutý je pro systém AS-Interface. Tento kabel vede data i napájení (30 V DC, 8 A). Černý, pro přídavné napájení 24V např. pro výstupní slave.



Obrázek 2.1 AS-I sběrnice [2]

Izolace kabelu je vyrobena z různých materiálů (PUR,TPE,EPD,Guma). Kabely jsou odolné proti UV záření.

Díky prořezávací technice spojování je instalace AS-Interface sítě velice rychlá a jednoduchá. Profilování kabelu zabraňuje přepólování.

Pokud nelze z jakéhokoli důvodu použít kabel AS-Interface můžeme použít kabel jiný, který má předem dané parametry. Při použití jiného kabelu může dojít ke zmenšení maximálně dosažené délky sítě.

2.3.2 AS-Interface master

Master obsahuje rozhraní, které se propojí s nadřazeným řídicím systémem (PLC, řídicí jednotkou, na systém průmyslové sběrnice vyšší úrovně).

2.3.3 AS-Interface slave

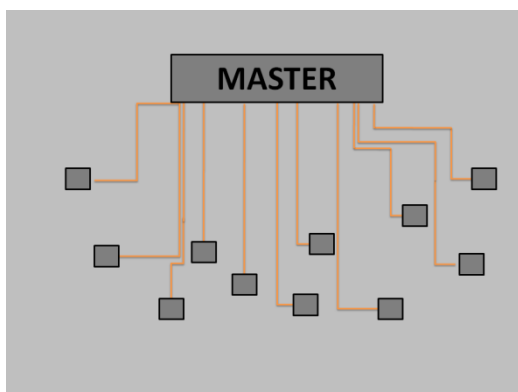
Slave se používá k připojení senzorů do sítě AS-I. V dnešní době se vyrábí velké množství různých slave zařízení. Slavy se vyrábí pro rozvaděčové prostředí (IP20) i pro prostředí mimo rozvaděč (až IP69). Na jednom slavu je umístěno několik vstupů nebo výstupů.

Základní druhy slavů jsou:

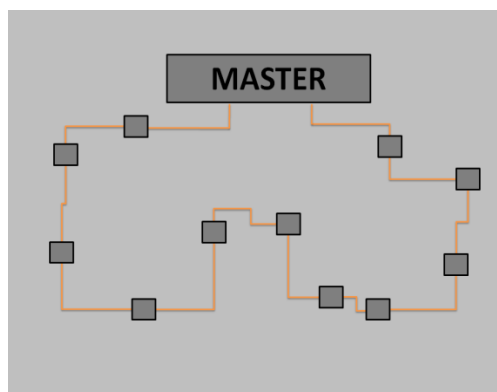
- Digitální vstup
- Digitální výstup
- Analogový vstup proudový (4-20mA)
- Analogový výstup proudový (4-20mA)
- Analogový vstup napěťový (0-10V)
- Analogový výstup napěťový (0-10V)
- Analogový vstup PT100

2.3.4 Možné topologie

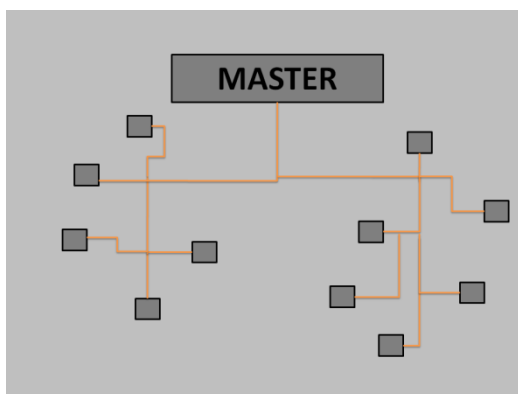
Topologie nám označuje fyzické uspořádání účastníků v síti.



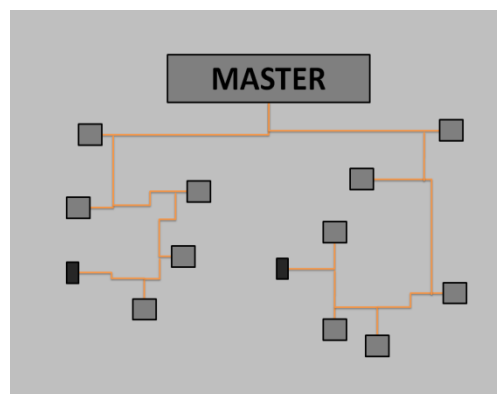
Obrázek 2.2 Hvězda [2]



Obrázek 2.3 Kruh [2]



Obrázek 2.4 Strom [2]



Obrázek 2.5 Lineární s odbočkami[2]

2.4 Systém komunikace

Sběrnici AS-Interface můžeme jako všechny průmyslové sběrnice kategorizovat dle sedmi úrovní referenčního modelu ISO/OSI. AS-Interface používá pouze 3 z těchto sedmi vrstev (fyzickou vrstvu, linkovou vrstvu a aplikační vrstvu).

Vrstva ISO/OSI	Funkce	AS-Interface
7: Aplikační	Poskytování síťových služeb uživateli	Zpráva, cyklus, profily, automatické adresování
6: Prezentační	Transformace síťových formátů na uživatelské formáty	
5: Relační	Přihlášení a odhlášení spojení	
4: Transportní	Příprava dat pro síťové transportní rozhraní	
3: Síťová	Příprava adres, směrování datových cest	
2: Linková	Struktura dat, rámce, zabezpečení, ošetření chyb	Datový telegram, start bit, stop bit, ošetření chyb
1: Fyzická	Mechanické a elektrické spoje pro přenos informací	Vedení, zdroj napájení, oddělení dat, střídavá impulsní modulace, rozvod napájení

Tabulka 2.1 Referenční model ISO/OSI [2]

- **1-Fyzická vrstva**

Je orientovaná na elektrické a mechanické spojení účastníků komunikace a přenáší tok informace mezi nimi.

- **2-Linková vrstva**

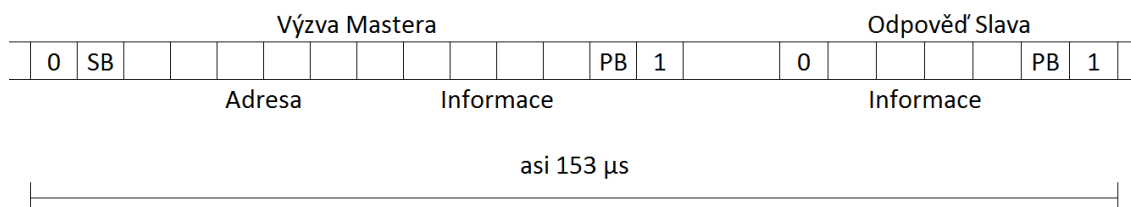
Je nad fyzickou vrstvou a dohlíží na spolehlivý přenos dat. V této vrstvě jsou data formátovaná do rámců, opatřena informací o adrese a informací o zabezpečení dat. Poté je informace postupně vysílána po vedení.

- **7-Aplikační vrstva**

Definuje povely, obsahy dat, posloupnost cyklu AS-I a chování účastníků, např. výměně a připojení slavů za provozu systému.

2.5 Komunikační profil

Princip komunikace AS-Interface je, že master vysílá rámec a slave odpovídá. V jednom cyklu si může master a slave předat maximálně 4 datové bity.



Obrázek 2.6 Master-Slave komunikace [2]

V našem případě potřebujeme přenést více informací než 4 bity. Potřebujeme použít nějaký komunikační protokol, který nám umožní přenést 4 bity v jednom cyklu. Tyto bity složíme a vytvoříme vícebitové slovo.

Můžeme použít jeden ze standardních profilů AS-Interface. Profil S-7.3 slouží k přenosu analogových 16bitových dat a to maximálně pro 4 kanály. Data se přenáší v sedmi cyklech. Přečtení těchto dat trvá asi 50 ms.

Rozlišovací Bity			Datové bity															Doplňkové bity		
E3	E2	E1	D16	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	O	V

Obrázek 2.7 Komunikační profil S7.3[1]

- E3 - E1 – Bity sloužící k určení čísla kanálu (binárně 1 - 4)
- D16 - D1 – Přenášené datové bity
- O – OVERLOAD -0 pokud je hodnota v rozsahu a 1 pokud je mimo
- V – VALID – 0 jedná se o neplatnou hodnotu, 1 platná hodnota

Přenášené 4 bity jsou rozděleny na 1 kontrolní bit K a na 3 datové. V každém jednom cyklu přeneseme data KXXX.

Přenos dat mezi masterem a slavem

Znázornění principu komunikace mezi masterem a slavem, v případě že slave žádá o data mastera.

Master (odesílá data)		Slave (žádá o data)
K X X X	→	1 1 1
K E1 E2 E3	→	1 0 1
K D16 D15 D14	→	1 0 0
K D13 D12 D11	→	0 1 1
K D10 D9 D8	→	0 1 0
K D7 D6 D5	→	0 0 1
K D4 D3 D2	→	0 0 0
K D1 O V	→	1 1 1

Tabulka 2.2 Přenos dat z masteru do slavu [3]

Přenos probíhá pořád cyklicky dokola. V případě, že o data žádá master vypadá komunikace takto.

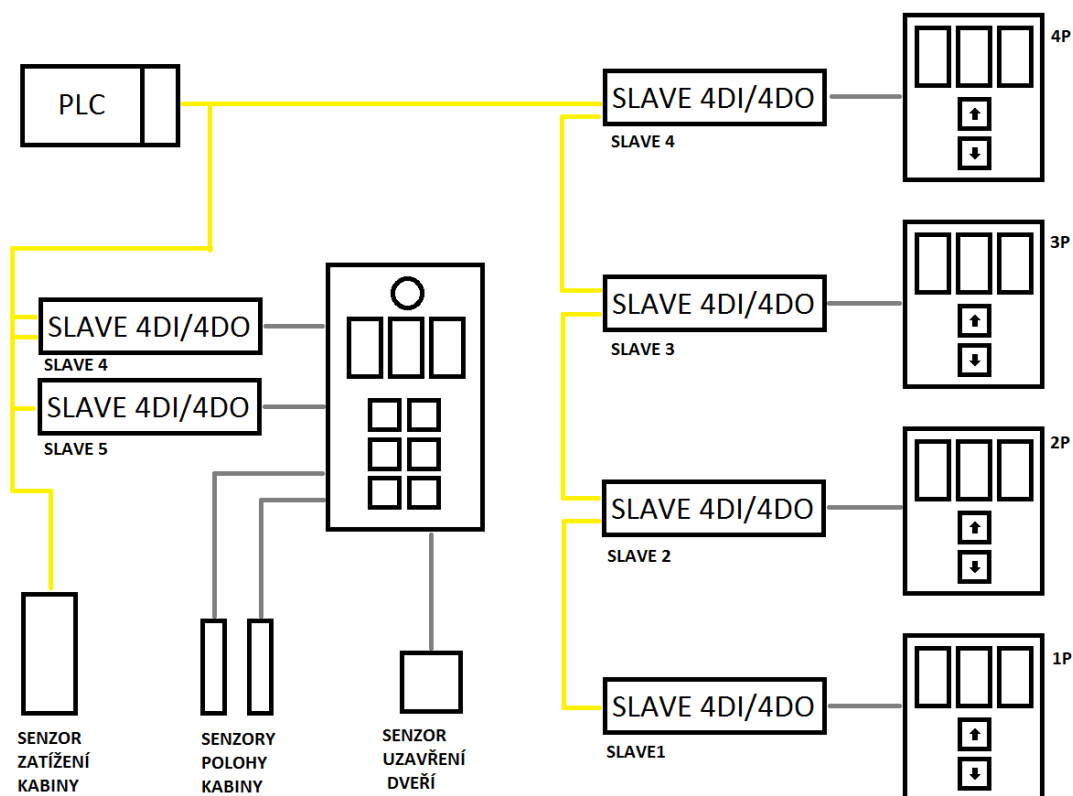
Master (žádá o data)		Slave (odesílá data)
K 1 1 1	→	E1 E2 E3
K 1 0 1	→	D16 D15 D14
K 1 0 0	→	D13 D12 D11
K 0 1 1	→	D10 D9 D8
K 0 1 0	→	D7 D6 D5
K 0 0 1	→	D4 D3 D2
K 0 0 0	→	D1 O V

Tabulka 2.3 Přenos dat se slavu do mastera [3]

3 KOMUNIKACE MODELU VÝTAHU

3.1 Základní zapojení komunikace

Následující obrázek znázorňuje zapojení komunikační sítě modelu výtahu. PLC komunikuje pomocí sběrnice AS-Interface (na obrázku žlutě). Na sběrnici jsou připojeny vstupně-výstupní slavy, které komunikují s ovládacími panely.



Obrázek 3.1 Základní zapojení komunikace

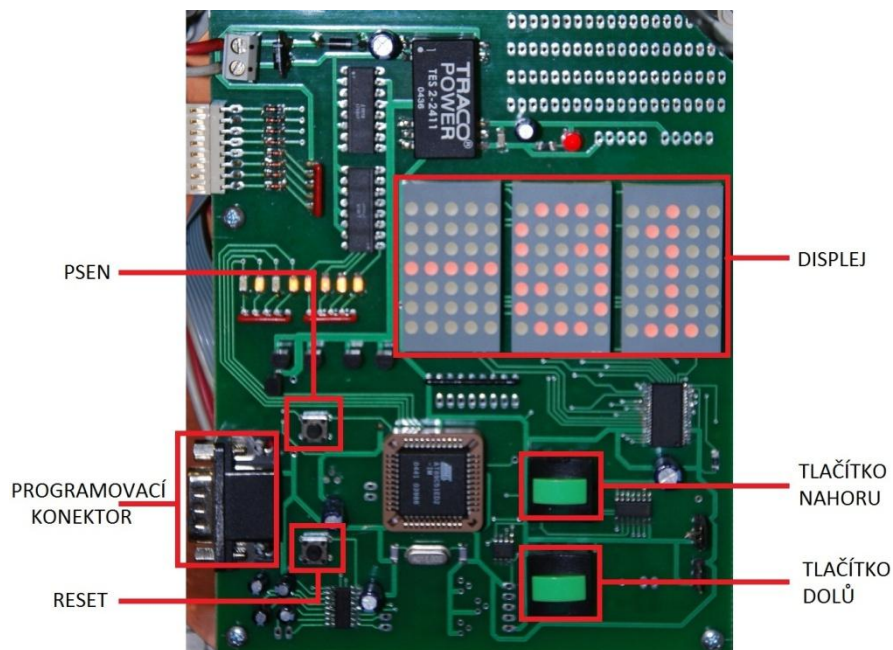
3.2 Popis komunikace

Komunikace je řízena masterem. Master odesílá data jednotlivým slavům. Nejprve jsou obslouženy ovladače pater a poté ovladač v kabině. Master posílá všem ovladačům pater stejné informace skrz jeden 4DI/4DO slave v každém patře. Informaci o tom, ve kterém patře se kabina výtahu nachází, dále pak jestli kabina směřuje (nebo bude směřovat) nahoru či dolů. Další informace jsou o tom, jestli je výtah v pohybu nebo ne, dále pak bit poruchy, paritní bit a bity over a valid.

Komunikace kabiny je realizována trochu jinak, z důvodu toho, že se z ní odesílá mnohem více dat než z ovladačů pater. Proto je použito dvou slavů (oba 4DI/4DO). Slave 5 se stará o zpracování dat příchozích od mastera a vysílá požadavky, aby master věděl jaké data má slavu poslat. Tyto příchozí informace jsou totožné s informacemi posílanými slavům v patře. Slave 6 podle příchozího požadavku mastera odesílá informace o zmáčknutých tlačítkách 1 – 24. Dále jestli jsou sepnuty indukčnostní snímače polohy, poté tlačítko stop, zvonek, snímač dveří a nakonec kontrolní paritní bit. Senzor přetížení kabiny je připojen přímo do sítě AS-Interface.

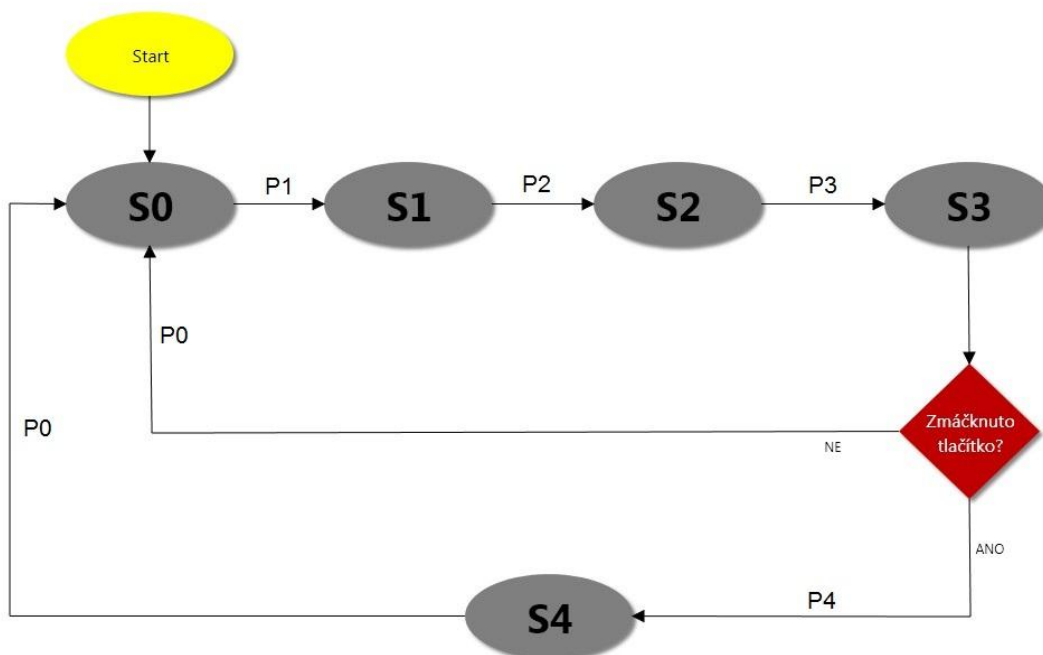
4 OVLADAČ PATRA

4.1 Popis panelu patra



Obrázek 4.1 Obrázek ovladače patra s popisky

4.2 Stavový popis ovladače patra



Obrázek 4.2 Stavový popis ovladače patra

Stavový automat je Moorova typu, jeho výstup je závislý pouze na vnitřním stavu.

Stav	Data (P1)	
	Žádost slavu	Odpověď mastera
S0	(0)111	K D10 D9 D8
S1	(0)010	K D7 D6 D5
S2	(0)001	K D4 D3 D2
S3	(0)000	K D1 O V
S4	(0)101 (zmáčknuto ↑)	-
S4	(0)100 (zmáčknuto ↓)	-
S4	(0)011 (zmáčknuto ↑ i ↓)	-

Tabulka 4.1 Komunikace v jednotlivých stavech ovladače patra obecně

Označení	Podmínka přechodu
P0	Odeslání požadavku 111 na port P0 a příjem kontrolního bitu v log.0
P1	Odeslání požadavku 010 na port P0 a příjem kontrolního bitu v log.1
P2	Odeslání požadavku 001 na port P0 a příjem kontrolního bitu v log.0
P3	Odeslání požadavku 000 na port P0 a příjem kontrolního bitu v log.1
P4	V případě, že bylo zmáčknuto tlačítko, dojde k přechodu do stavu S4

Tabulka 4.2 Tabulka podmínek přechodů (patro)

Než se program dostane do stavu S0 musí se vyslat požadavek 111 (X000XXX) do řídicího systému. Dále program počká na to, než nám řídicí systém odpoví, a to zjistí tak, že dojde ke změně kontrolního bitu K z log. 1 na log. 0 (XXXX1XXX). Nyní se program dostává do stavu S0. Změnou kontrolního bitu se nejen dostává program do stavu S0, ale i signalizuje to, že program obdržel příchozí informaci a může ji tedy zpracovat. Informací jsou 3 bity vpravo od kontrolního bitu. Po zpracování první části informace se vyšle požadavek 010.

Nyní čeká, než se kontrolní bit změní na log. 1. Po změně kontrolního bitu K se program dostane do stavu S1. Znovu dochází ke zpracování informace. Opět to jsou 3 bity z celkové zprávy. Následuje vyslání požadavku 001 na portu P0.

Program počká na změnu kontrolního bitu na log.0, a tím se dostaneme do následujícího stavu. Ve stavu S2 se po uložení informace odešle požadavek 000.

Program znovu čeká na změnu K do log. 1. Po zpracování poslední části informace se provede kontrola parity. Pokud je parita lichá dojde k zobrazení dat na displeji.

Nyní mohou nastat dvě možnosti. První, že bylo zmáčknuto alespoň jedno tlačítko, tím se program přesune do stavu S4. Ve stavu S4 se vyhodnotí, jestli bylo zmáčknuto tlačítko nahoru, dolů nebo oba, a podle toho se masteru pošle požadavek (v tomto případě spíše informace) o tom, která tlačítka byla zmáčknuta. Pokud bylo zmáčknuto

tlačítko nahoru, vysílá se 100, dolů 101 a pokud jsou zmáčknuty oba tlačítka, dochází k odeslání 011. Po stavu S4 se vysílá požadavek 111 a po změně bitu K se program dostane znovu do stavu S0. Druhá možnost je, že nebylo zmáčknuto žádné tlačítko a posílá se požadavek 111 ihned. A následuje znovu stav S0. Komunikace takto probíhá cyklicky dokola.

4.3 Popis programu ovládače patra

Ovladače jednotlivých pater se liší od sebe pouze pozměněním jedné proměnné. Do proměnné „**KTERE_PATRO**“ se uloží číslo patra v intervalu 1-24. Toto je vlastně všechno, co musíme pro změnu programu, pro jednotlivá patra udělat.

Příklad pro třetí patro:

```
unsigned char KTERE_PATRO=3;
```

Pouze u prvního a posledního patra můžeme „zabránit“ zmáčknutí tlačítek, které nemají smysl. To je v prvním patře dolů a v posledním nahoru. Tato úprava nemá smysl v případě, že by tlačítka nebyly v těchto patrech hardwarově vyvedeny.

Příklad pro první patro:

```
void keyboard_interrupt(void) interrupt 0 {
    if (TL_NAH == 0) key_reg |= 2;
    //if (TL_DOL == 0) key_reg |= 1;
}
```

Nyní následuje popis programu. Nejprve jsou definovány makra, abych nemusel pracovat přímo s bity na portech. Poté jsem nadefinoval několik globálních proměnných, které jsou typu **unsigned char** (8 bitů):

- **prichozi_patro** – se ukládá číslo patra, ve kterém se výtah právě nachází, data jsou ve tvaru:

0	0	0	D10	D9	D8	D7	D6
---	---	---	-----	----	----	----	----

- **prichozi_data** – do této proměnné se ukládají informace o pohybu kabiny, o poruše, dále pak paritní bit a bity **Overload** a **Verify**.

0	D5	D4	D3	D2	D1	O	V
---	----	----	----	----	----	---	---

kde: D5 – bit určující směr nahoru
D4 – bit určující směr dolů
D3 – bit určuje, jestli kabina stojí nebo jede
D2 – poruchový bit
D1 – paritní bit

O – Overload (vždy v log. 0)

V – Verify (vždy v log. 1)

- **key_reg** – v této proměnné jsou uloženy záznamy o stisku tlačítka:

Nahoru	0	0	0	0	0	0	1	0
--------	---	---	---	---	---	---	---	---

Dolů	0	0	0	0	0	0	0	1
------	---	---	---	---	---	---	---	---

Nahoru i dolů	0	0	0	0	0	0	1	1
---------------	---	---	---	---	---	---	---	---

Popis jednotlivých procedur použitých v programu.

delaynop	Procedura slouží k vytvoření zpoždění trvajícího přibližně 50 μ s.
Disp_Config	Slouží k nastavení displeje. Touto procedurou můžeme displej zapnout, vypnout. Nebo nastavit blikání.
Disp_Intens	Tato procedura umožňuje nastavit velikost intenzity displeje. Intenzitu můžeme volit od 0 do 15.
Inicializuj	Procedura, která se spouští pouze jednou a to na začátku programu. Nastaví všechny proměnné do počátečního stavu. Nastaví displej včetně jeho intenzity. Povolí přerušení od zmáčknutých tlačítek.
Parita	Tato procedura slouží ke kontrole příchozích dat. Pokud příchozí data mají lichou paritu, tak jsou s vysokou platností správná. Z PLC přichází data, která mají paritu sudou, ale tím že přídavný bit Overload je v log. 0 a bit Verify je v log. 1, celkově vyhodnocují paritu jako lichou.
Zobraz2	Tato poslední procedura slouží k obsluze displeje. Zároveň také zhasíná podsvětlení tlačítek, pokud výtah stojí v patře, pro které byl program vytvořen. Do procedury vstupují globální proměnné prichozi_patro a prichozi_data (zbaveny bitů Overload a Verify). prichozi_data jsou rozdělena na čtyři bity, aby se s nimi mohlo snadněji pracovat. Jsou to bity porucha , jede , up a down . Podle příchozích dat se na displeji objeví příslušné nápisy.

Tabulka 4.3 Popis procedur pro ovladač patra

Na následujícím obrázku je znázorněno jak vypadají data, která odchází z portu a která na port přichází.

	P1		
Odchozí data	Žádost slavu	0000	(Výstup)
Příchozí data	0000	Odpověď mastera	(Vstup)

Obrázek 4.3 Znázornění komunikace patra na portu P1

Průběh komunikace ovládače patra je shrnut v následující tabulce.

Stav	Data (P1)	
	Žádost slavu	Odpověď mastera
S0	(0)111	0 P4 P3 P2
S1	(0)010	1 P1 P0 ↑
S2	(0)001	0 ↓ Jede Por
S3	(0)000	1 Par O V
S4	(0)101 (zmáčknuto ↑)	-
S4	(0)100 (zmáčknuto ↓)	-
S4	(0)011 (zmáčknuto ↑ i ↓)	-

Tabulka 4.4 Komunikace v jednotlivých stavech ovládače patra ve skutečnosti

Celý program běží v takzvané supersmyčce. To znamená, že po prvotní inicializaci (zavoláním procedury **Inicializuj**) se program dostane do nekonečné smyčky. To je docíleno v našem případě vždy splněnou podmínkou v cyklu while („**while (1)**“). Díky tomu je docíleno toho, aby program běžel pořád dokola až do nekonečna. Funkce main obstarává všechnu komunikaci. Komunikace je tvořena posloupností stále se opakujících stavů.

Nyní se pokusím popsat, co se přesně děje v jednotlivých stavech. Komunikace na portech probíhá inverzně (bity jsou negovány). V nultém stavu slave vyšle na port **P1** požadavek **0x8F**, což je ve skutečnosti (01110000), jak bylo popsáno v předchozích kapitolách. Nyní čekáme, až se kontrolní bit **K** změní na logickou nulu. Potom počkáme přibližně **1,5ms**. Data která přišla na port zneguji, aplikuji masku a uložím do proměnné **prichozi_patro**. Aplikováním masky se zbavím zbytečných bitů a pracuji pouze se třemi obdrženými datovými bity. Poté bity v proměnné **prichozi_patro** posunu o tři místa doleva, abych si připravil místo pro další datové bity.

0	0	D10	D9	D8	0	0	0
---	---	-----	----	----	---	---	---

Obrázek 4.4 Proměnná prichozi_patro po stavu 0

Ve stavu prvním se na port posílá **0xDF (010)**. Opět čekáme na příchozí data. Pokud je kontrolní bit v tomto případě roven logické jedničce, opět počkáme **1,5ms**, než dojde k ustálení hodnot. Data zneguji, použiji masku a pomocí logického součtu přidám do

proměnné **prchozi_patro**. Nyní vidím, že hodnota **D5**, která je nyní na pozici posledního platného bitu, nepatří do proměnné **patro**. Proto proměnnou **prchozi_patro** bitově vynásobím hodnotou **0x01**, čímž docílím toho, že můžu do proměnné **prchozi_data** uložit datový bit **D5**. Poté pomocí bitových posunů upravím proměnnou **prchozi_patro** tak, aby v ní byly pouze bity, které ukazují číslo patra. Proměnnou **prchozi_data** opět bitově posunu tak, abych mohl do ní uložit další data.

0	0	0	D10	D9	D8	D7	D6
---	---	---	-----	----	----	----	----

Obrázek 4.5 Proměnná **prchozi_patro** po stavu 1

0	0	0	0	D5	0	0	0
---	---	---	---	----	---	---	---

Obrázek 4.6 Proměnná **prchozi_data** po stavu 1

Ve druhém stavu vysíláme na port hodnotu **0xEF (001)**. Počkáme na změnu **K** v logickou nulu. Po ustálení hodnot příchozí data opět zneguji a použiji masku. Pomocí logického součtu upravené data přidám do proměnné **prchozi_data**. Posledním krokem je znova posunout bity tak, aby se uvolnilo místo pro další

0	D5	D4	D3	D2	0	0	0
---	----	----	----	----	---	---	---

Obrázek 4.7 Proměnná **prchozi_data** po stavu 2

Třetí stav vysílá na port **P1** hodnotu **0xFF**, což je požadavek **000**. Znovu počkáme na změnu logické úrovně kontrolního bitu **K** z logické nuly na logickou jedničku. Po zpoždění data zneguji, opět aplikuji masku a bitově přičtu do proměnné **prchozi_data**.

0	0	0	D10	D9	D8	D7	D6
---	---	---	-----	----	----	----	----

Obrázek 4.8 Proměnná **prchozi_patro** ve stavu 3

0	D5	D4	D3	D2	D1	O	V
---	----	----	----	----	----	---	---

Obrázek 4.9 Proměnná **prchozi_data** ve stavu 3

Nyní jsou obě proměnné úplně a připravené k dalšímu použití. Nyní zkontrolujeme paritu. Pokud jsou data v pořádku (parita je **lichá**), můžu pokračovat dále. Ještě než začnu s proměnnými pracovat upravím si **prchozi_data** bitovým posunem tak, abych se zbavil **paritního bitu** a bitů **Overload** a **Valid**.

0	D5	D4	D3	D2	D1	O	V
---	----	----	----	----	----	---	---



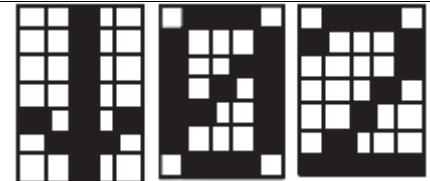


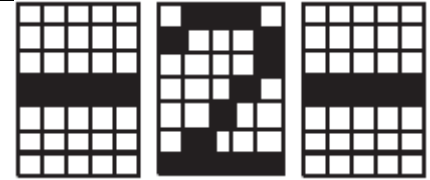
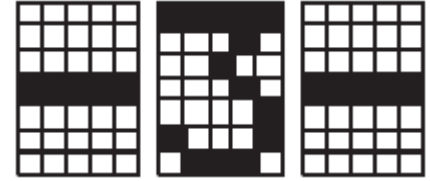
Obrázek 4.10 Proměnná **prchozi_data** ve stavu 3 po úpravě

Nyní můžu zavolat proceduru **Zobraz2**. Procedura podle vstupních dat provede zobrazení na displeji, nebo zruší požadavky na zastavení v daném patře, pokud v něm již stojíme.

Pokud došlo během běhu programu ke zmáčknutí tlačítek, došlo k vyvolání přerušení. Po odskočení do přerušení, přerušení zakážu, a po obsloužení opět přerušení povolím. V případě, že bylo zmáčknuto tlačítko nahoru, bitově přičtu do registru **key_reg** hodnotu 2 (00000010B). Když bylo zmáčknuto tlačítko dolů, přičtu hodnotu 1

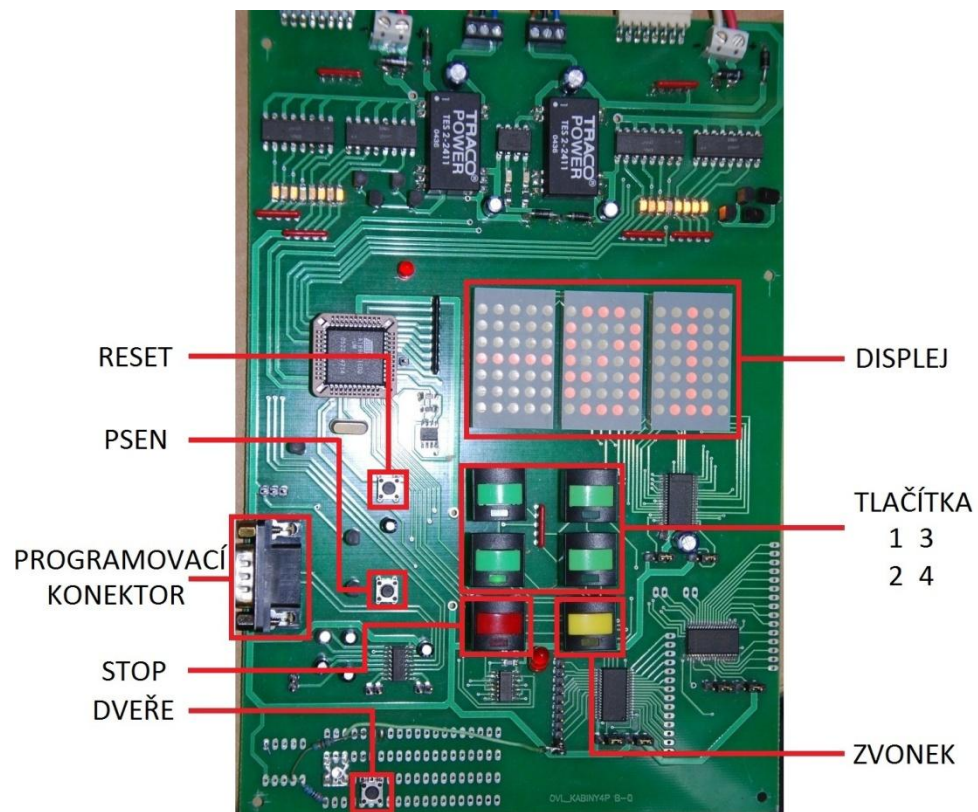
(00000001). Z toho vyplývá, že pokud byly zmáčknuty oba tlačítka, nachází se v **key_reg** hodnota 00000011. V případě, že kabina výtahu stojí v patře, pro které je program určen, procedura **Zobraz2** vynuluje proměnnou **key_reg** a zhasne podsvícení tlačítek. Ve výsledku to znamená, že požadavek přivolání výtahu byl splněn.

4.4 Popis možného zobrazení na displeji

	<p>Znak pomlčka na prvním displeji znamená, že výtah není v pohybu. Druhé dvě číslice určují, v jakém patře se výtah zastavil. V tomto případě výtah stojí v druhém patře.</p>
	<p>Pokud šipka na prvním displeji bliká, znamená to, že se výtah zanedlouho rozjede směrem nahoru. Trvale svítící šipka znamená, že výtah jede směrem nahoru.</p>
	<p>Pokud šipka na prvním displeji bliká, znamená to, že se výtah zanedlouho rozjede směrem dolů. Trvale svítící šipka znamená, že výtah už jede směrem dolů.</p>
	<p>V případě, že z řídicího systému přijde signál porucha v logické 1, vypíše se na displeji okamžitě zpráva „err“.</p>
	<p>Tato chybová hláška se objeví v případě, že řídicí systém pošle číslo patra, které bude větší než 24, což je více než patro nejvyšší.</p>
	<p>Když řídicí systém vysílá signál, že kabina se pohybuje, ale nepřichází údaj o tom, jakým jede směrem.</p>
	<p>Pokud přijímáme z řídicího systému zároveň signál, že kabina jede nebo pojede zároveň nahoru i dolů. Objeví se na displeji tato chybová hláška.</p>

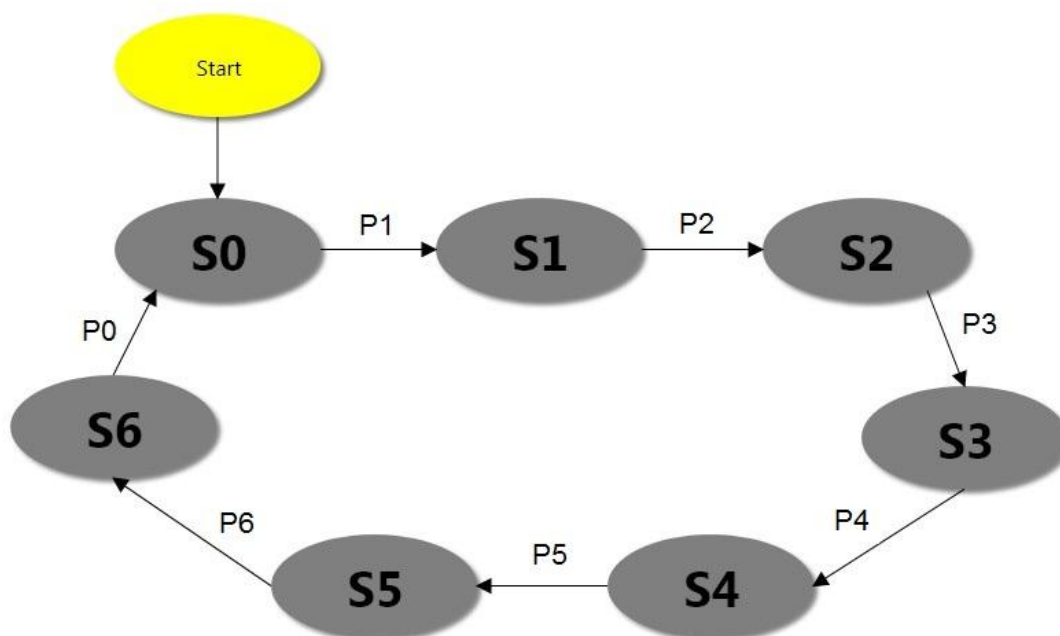
5 OVLADAČ KABINY

5.1 Popis panelu kabiny



Obrázek 5.1 Obrázek ovladače kabiny s popisky

5.2 Stavový popis ovladače kabiny



Obrázek 5.2 Stavový popis ovladače kabiny

Stavový automat ovladače kabiny je také Moorova typu, jeho výstup závisí pouze na vnitřním stavu.

Program kabiny se dá popsat na sedm stále se opakujících stavů, které se starají o spolehlivou výměnu dat mezi kabinou a masterem a naopak. Příchozí data z masteru se přenáší stejným způsobem jako v ovladači patra.

Do masteru potřebujeme přenést 30 datových bitů, proto musíme data rozdělit. Vzhledem k tomu, že v jednom stavu můžeme přenést pouze 3 datové bity a stavů je 7. Nemůžeme data přenést v jednom cyklu. Je zapotřebí použít cykly dva. Data se přenáší podle následující tabulky:

Stav	Příchozí data (P1)		Odchozí data (P0)		Cyklus
	Žádost mastera	Data pro slave	Data pro mastera	Žádost slave	
	Slave 6	Slave 5	Slave 6	Slave 5	
S0	K111	K D1 O V	A E2 E1 E0	B111	E2 E1 E0 = 0 0 0
S1	K101	0 0 0 0	A D16 D15 D14	B101	
S2	K100	0 0 0 0	A D13 D12 D11	B100	
S3	K011	0 0 0 0	A D10 D9 D8	B011	
S4	K010	K D10 D9 D8	A D7 D6 D5	B010	
S5	K001	K D7 D6 D5	A D4 D3 D2	B001	
S6	K000	K D4 D3 D2	A D1 O V	B000	
S0	K111	K D1 O V	A E2 E1 E0	B111	E2 E1 E0 = 0 0 1
S1	K101	0 0 0 0	A D32 D31 D30	B101	
S2	K100	0 0 0 0	A D29 D28 D27	B100	
S3	K011	0 0 0 0	A D26 D25 D24	B011	
S4	K010	K D10 D9 D8	A D23 D22 D21	B010	
S5	K001	K D7 D6 D5	A D20 D19 D18	B001	
S6	K000	K D4 D3 D2	A D17 O V	B000	

Tabulka 5.1 Princip komunikace kabiny výtahu (Výměna dat mezi masterem a slavem)

A – log.1 pokud jsou sepnuty oba snímače polohy
B – log.1 v případě, že bylo zmáčknuto stop tlačítko

V následující tabulce jsou uvedeny podmínky přechodu do následujícího stavu:

Označení	Podmínka přechodu
P0	Přichází požadavek 111 na port P0 (X000XXX) a má následovat S0
P1	Přichází požadavek 101 na port P0 (X010XXX) a má následovat S1
P2	Přichází požadavek 100 na port P0 (X011XXX) a má následovat S2
P3	Přichází požadavek 011 na port P0 (X100XXX) a má následovat S3
P4	Přichází požadavek 010 na port P0 (X101XXX) a má následovat S4
P5	Přichází požadavek 001 na port P0 (X110XXX) a má následovat S5
P6	Přichází požadavek 000 na port P0 (X111XXX) a má následovat S6

Tabulka 5.2 Tabulka podmínek přechodů (kabina)

Po zapnutí program čeká, jestli master vysílá na port požadavek 111, to znamená, že na portu se objeví informace (X000XXX pozn. negovaná). Program se dostává do stavu S0.

Ve stavu S0 se nejprve zpracují data, která přišla z masteru. Dále se začnou připravovat data k odeslání. Odesílají se rozlišovací bity E a zároveň se odesílá požadavek slavu 111 o data z masteru. Proměnná stav se změní na 1. Pokud master vysílá požadavek 101, program se dostane do stavu S1.

V tomto stavu nedochází ke zpracování dat, protože master žádná neposílá. Pouze se připraví odchozí data a přidá se požadavek slavu. Proměnná stav se změní na 2. V případě že master vysílá požadavek 100, program se dostává do stavu S2.

Znovu nepřichází data. Pouze se připraví datové bity pro master k odeslání. Přidá se požadavek o data pro slave. Program nastaví proměnnou stav na hodnotu 3. Vyčká, než bude master žádat 011 a dostaneme se do stavu S3.

Opět žádná data nepřichází, proto nemusíme nic nikam ukládat. Stačí připravit data k odeslání. K datům se přidá požadavek 011. Proměnná stav se navýší o 1 a program čeká, než master vyšle požadavek 010, a tím se dostane do následujícího stavu.

Ve stavu 4 již přichází data pro slave (kabinu), tato data se zpracují podle stejného postupu. Následuje odeslání dat spolu s požadavkem 010. Dojde k inkrementaci proměnné stav. A program opět čeká.

Po žádosti mastera 001 se dostává do stavu S5. Tento stav je předposlední. Udělá to samé co ve stavu předchozím, akorát požadavek slavu je 001.

Do stavu S6 se dostane po žádosti mastera 000. Zpravuje poslední data a poslední data připraví k odeslání. Žádost slavu je v tomto stavu 000.

Nyní proběhl jeden cyklus komunikace. Je patrné, že data pro slave (kabinu) přišly z mastera celé. Dokonce některé datové bity byly nevyužity, zatím co odesílání dat směrem do mastera je v polovině. Proto se změní rozlišovací bit a v dalším cyklu se posílá druhá část informace. Po odeslání celé informace se tento jev opakuje stále dokola.

5.3 Popis programu ovladače kabiny

Program pro ovladač kabiny je o něco složitější než program pro ovládání pater. Program je složitější z důvodu, že se do PLC odesílá více dat než v patrech. Ke komunikaci jsou použity 2 slavy.

	Slave 6 (High)	Slave 5 (Low)	
P1	Žádost mastera	Data pro slave	Příchozí data
P0	Data pro mastera	Žádost slave	Odchozí data

Obrázek 5.3 Znáznornění komunikace kabiny na portech P0 a P1

Data na portech jsou opět negované. Na začátku programu definuji makra, která slouží ke zjednodušení práce s proměnnými. Jednodušší a přehlednější je pracovat s proměnnou DVERE než s P3_7. Potom definuji několik globálních proměnných typu **unsignedchar**(8bitů):

- **stav** – v této proměnné je uloženo číslo stavu, ve kterém se program nachází, po provedení stavu se proměnná změní na číslo stavu, který má následovat.
- **pom_P0** – P0 je portem odesílajícím, jak je patrné z předchozího obrázku. Data, které chceme odesílat musíme postupně připravit, proto máme tuto pomocnou proměnnou. Pro odeslání dat pak jen **pom_P0** přesuneme na port **P0**.
- **key_pat[3]**– tyto 3 byty slouží k zaznamenání zmáčknutí tlačítek pro volbu patra, do kterého chce člověk jet. Zmáčknutím jednoho nebo i více tlačítek se vyvolá přerušení, které nastaví příslušný bit zvoleného patra a v dalším cyklu se požadavek zpracuje. V proměnné nejsou data negované.

24	23	22	21	20	19	18	17
----	----	----	----	----	----	----	----

Obrázek 5.4 key_pat [2]

16	15	14	13	12	11	10	9
----	----	----	----	----	----	----	---

Obrázek 5.5 key_pat [1]

8	7	6	5	4	3	2	1
---	---	---	---	---	---	---	---

Obrázek 5.6 key_pat [0]

- **podsvet [3]**– pokud bylo zmáčknuto tlačítko pro volbu patra proměnnou **key_pat** zneguju a uložím do podsvet přes bitový součin, aby došlo k rozsvícení těch, které ještě nebyly zmáčknuty.
- **prichozi_patro** – se ukládá číslo patra, ve kterém se výtah právě nachází, data jsou ve tvaru:

0	0	0	D10	D9	D8	D7	D6
---	---	---	-----	----	----	----	----

- **prichozi_data** – do této proměnné se ukládají informace o pohybu kabiny, o poruše, dále pak paritní bit a bity **Overload** a **Verify**.

0	D5	D4	D3	D2	D1	O	V
---	----	----	----	----	----	---	---

Kde:

D5 – bit určující směr nahoru

D4 – bit určující směr dolů

D3 – bit určuje, jestli kabina stojí nebo jede

D2 – poruchový bit

D1 – paritní bit

O – Overload (vždy v log. 0)

V – Verify (vždy v log. 1)

- **prichozi_patro** a **prichozi_data** jsou stejné jako v programu patra.

Nyní už jen definuji několik bitů, které slouží k zachování dat:

- **zm_stop** – uchovává informaci, jestli bylo zmáčknuto stop tlačítko
- **zm_zvonek** – uchovává informaci, jestli bylo zmáčknuto tlačítko zvonek

- **zm_dvere** – je v log 1 pokud dveře jsou zavřené (proměnná DVERE je rovna log 0)
- **castzpravy** – díky tomuto bitu víme, ve kterém cyklu se nacházíme a která data se mají odeslat

Popis jednotlivých procedur použitých v programu. Vynechal jsem ty, co jsou uvedeny u ovladače patra a v ovladači kabiny jsou použity také.

delay_ms(číslo)	Procedura slouží k vytvoření zpoždění přibližně trvajícího číslo x 1 ms
MAX6955_Config	Slouží k nastavení obvodu s tlačítky. Zvolíme normální chod a nastavíme skenování 24 tlačítek.
MAX6956_Config	Tato procedura umožňuje měnit nastavení obvodu pro poosvětlení tlačítek.
MAX6956_ActiveLED	Vstupem do této procedury je proměnná podsvet , podle které dojde k rozsvícení těch led, které by svítit měly.
Init	Procedura, která se spouští pouze jednou a to na začátku programu. Nastaví všechny proměnné do počátečního stavu. Nastaví displej včetně jeho intenzity. Povolí přerušení od zmáčknutých tlačítek.
Zobraz_na_3	Tato procedura zjednodušuje výpis na všechny 3 panely displeje. Vstupem do ní jsou 3 znaky, které se mají vypsát.
Disp_Porucha	Tato procedura slouží k zobrazení chybové hlášky při zatížené kabině. V proceduře se nachází lokální proměnná čas, která nastavuje, jak dlouho se bude zobrazovat jednotlivá informace na displeji. Disp_Porucha využívá procedury Zobraz_na_3 .
VynulujPodsvetleni	Pokud kabina stojí v patře, na které byl podán požadavek k zastavení, zavolá se tato procedura. Podle patra se najde příslušný bit v proměnné podsvet a ten se nastaví do log. 1 (nesvítil) a aktualizované podsvet se realizuje procedurou MAX6956_ActiveLED .
Parita_odchozi	I odchozí data musí být kontrolovány pomocí parity. Parita funguje stejně jako parita příchozích dat. Spočítám počet logických jedniček, a pokud je lichý, vracím logickou 1.
snimace_a_stop	V této proceduře se nastaví nepoužité bity, které se posílají na port P0. Jeden z bitů se nastaví, pokud je zmáčknuo stop tlačítko. Druhý se nastaví, pokud jsou sepnuty oba indukční snímače. Slouží k tomu, aby se tyto důležité informace dostaly prakticky okamžitě do PLC. Jinak by se tam dostaly až po uběhnutí dvou celých cyklů.

Hlavní funkce main

Nejprve se podíváme na to, jaké data se posílají do masteru a co přesně znamenají. Přehled je uveden v následující tabulce. Důležité je vědět, která část dat se odesílá. To poznáme v PLC pomocí rozlišovacích bitů E3-E1 a v programu mikroprocesoru to určuje bit **castzpravy**.

STAV	Označení bitu	První část zprávy	Označení bitu	Druhá část zprávy
S0	E3	0	E3	0
	E2	0	E2	0
	E1	0	E1	1
S1	D16	Tlačítko 16	D32	-
	D15	Tlačítko 15	D31	-
	D14	Tlačítko 14	D30	Dveře
S2	D13	Tlačítko 13	D29	Parita
	D12	Tlačítko 12	D28	Zvonek
	D11	Tlačítko 11	D27	Stop
S3	D10	Tlačítko 10	D26	Snímač 2
	D9	Tlačítko 9	D25	Snímač 2
	D8	Tlačítko 8	D24	Tlačítko 24
S4	D7	Tlačítko 7	D23	Tlačítko 23
	D6	Tlačítko 6	D22	Tlačítko 22
	D5	Tlačítko 5	D21	Tlačítko 21
S5	D4	Tlačítko 4	D20	Tlačítko 20
	D3	Tlačítko 3	D19	Tlačítko 19
	D2	Tlačítko 2	D18	Tlačítko 18
S6	D1	Tlačítko 1	D17	Tlačítko 17
	O	Overload	O	Overload
	V	Verify	V	Verify

Tabulka 5.3 Vysvětlení zkratk použitých v komunikaci kabiny

Když už jsme si ujasnili, jak vypadají data která budeme posílat, můžeme se podívat na funkci main. Po zapnutí programu se provede inicializační procedura **Init**, kde se nastaví všechny proměnné na výchozí hodnoty. Po zinicilizování se program dostane do nekonečné supersmyčky, která se provádí pořád dokola. Stejně tomu tak bylo v programu pater.

Program se po zapnutí dostane do stavu **S0**. Což znamená, že od mastera přišel požadavek 111. Nyní podle bitu **castzpravy** určíme, jak mají vypadat bity E3-E1 (000 nebo 001) a tyto data vložíme na příslušné místo v proměnné **pom_P0**.

0	E3	E2	E1	0	0	0	0
---	----	----	----	---	---	---	---

Obrázek 5.7 S0 pomocná pom_P0 po přiřazení bitů E1-E3

Následuje zavolání procedury **snimace_a_stop**. V případě zmáčknutí tlačítka stop, nebo pokud budou oba indukčnostní snímače sepnuty najednou, se do proměnné **pom_P0** přidají log 1.

A	E3	E2	E1	B	0	0	0
---	----	----	----	---	---	---	---

Obrázek 5.8 S0 pomocná pom_P0 po přiřazení bitů A a B

A – log.1, pokud jsou sepnuty oba snímače polohy

B – log.1 v případě, že bylo zmáčknuto stop tlačítko

Nyní ještě musíme zpracovat data, které nám poslal master. Tyto data nerozlišují podle toho jakou hodnotu má proměnná **castzpravy**. Celou informaci jsme schopni přijmout v jednom komunikačním cyklu, zatímco na odesílání informací do PLC potřebujeme cykly dva. Příchozí bity jsou stejné jako v ovladači pater pro cyklus 111. Jsou to bity **D1,O** a **V**. Nyní s výjimkou prvního komunikačního cyklu máme již proměnné **prichozi_data** a **prichozi_patro** úplné.

0	0	0	D10	D9	D8	D7	D6
---	---	---	-----	----	----	----	----

Obrázek 5.9 S0 proměnná prichozi_patro

0	D5	D4	D3	D2	D1	O	V
---	----	----	----	----	----	---	---

Obrázek 5.10 S0 proměnná prichozi_data

Nyní můžeme zkontrolovat paritu příchozích dat pomocí procedury **Parita**. Pokud jsou data v pořádku, upravíme proměnnou **prichozi_data** tak, abychom mohli informaci zobrazit na displeji. Zobrazení provede procedura **Zobraz2**, které je prakticky stejná jako procedura v ovladači patra. Akorát rozlišuje dvě možnosti poruchy, a to pokud porucha nastala v zatížené kabině nebo v kabině prázdné. Poslední krokem je do proměnné **pom_P0** přidat žádost o data pro slave (111). Pak proměnná **pom_P0** vypadá následovně:

A	E3	E2	E1	B	1	1	1
---	----	----	----	---	---	---	---

Obrázek 5.11 S0 proměnná pom_P0 před odesláním

Nyní jen proměnnou **pom_P0** znegujeme a odešleme do PLC. Proměnnou stav nastavíme na 1, abychom věděli jaký stav bude následovat.

Od mastera přichází požadavek 101, jsme ve stavu S1. Podle bitu **castzpravy** do **pom_P0** uložíme informace o zmáčknutí tlačítek 16, 15 nebo 14, které načteme z proměnné **podsvet**. Pozor musíme dát na to, že proměnná **podsvet** má informace uloženy negovaně. Pokud se má odesílat druhá část zprávy, tak do proměnné vložíme informaci o zavření dveří. Přidáme bity, které mají být přednostně předány do PLC

pomocí procedury **snimace_a_stop**. V tomto stavu nám od mastera nepřichází žádné data, proto nemusím nic zpracovávat. Nakonec přidáme žádost slavu o data (101). V proměnné **pom_P0** pak vypadá takto:

A	D16	D15	D14	B	1	0	1
---	-----	-----	-----	---	---	---	---

Obrázek 5.12 S1 proměnná **pom_P0** pro první část zprávy

A	0	0	D30	B	1	0	1
---	---	---	-----	---	---	---	---

Obrázek 5.13 S1 proměnná **pom_P0** pro druhou část zprávy

Znegované data odešleme do PLC a inkrementuju proměnnou **stav**.

Dostáváme se do stavu **S2**. Master žádá 100. V tomto stavu nám stále ještě nepřichází žádné data, a proto jen připravíme data na odeslání. V případě, že bit **castzpravy** bude v log 0, vložíme do **pom_P0** informace o tlačítkách 13, 12 a 11. Pokud bude log. 1, vložíme bity stop, zvonek a parita. Paritní bit pro odchozí paritu musíme zjistit pomocí procedury **Parita_odchozi**. Odchozí parita je **lichá**. Následuje procedura **snimace_a_stop**, jako v každém stavu. Data před odesláním vypadají takto:

A	D13	D12	D11	B	1	0	0
---	-----	-----	-----	---	---	---	---

Obrázek 5.14 S2 proměnná **pom_P0** pro první část zprávy

A	D29	D28	D27	B	1	0	0
---	-----	-----	-----	---	---	---	---

Obrázek 5.15 S2 proměnná **pom_P0** pro druhou část zprávy

Znegované **pom_P0** odešleme a **stav** změním na 3.

Ve stavu **S3** přichází žádost masteru 011. Stále ještě nepřichází žádné data pro slave, a tak pouze připravíme data pro mastera. Podle toho, jestli se posílá první část zprávy nebo druhá, vložíme informace do pomocné **pom_P0**. Pokud se má odesílat první část zprávy, bude v **pom_P0** informace o tlačítkách 10, 9 a 8. V případě, že mají být odesílané data z druhé části, tak vložíme do **pom_P0** informaci o indukčních snímačích a o tlačítku 24.

A	D10	D9	D8	B	0	1	1
---	-----	----	----	---	---	---	---

Obrázek 5.16 S3 proměnná **pom_P0** pro první část zprávy

A	D26	D25	D24	B	0	1	1
---	-----	-----	-----	---	---	---	---

Obrázek 5.17 S3 proměnná **pom_P0** pro druhou část zprávy

Dostáváme se do stavu **S4**. Požadavek mastera je 010. Dle proměnné **castzpravy** do **pom_P0** vložíme informaci o tlačítkách 7, 6 a 5 nebo 23, 22 a 21. Dále zavoláme proceduru **snimace_a_stop**. V tomto stavu nám již master posílá data, které musíme zpracovat. Do proměnné **prichozi_patro** vložíme bity pro slave D10, D9 a D8 a posuneme je bitovým posunem o 3 místa doleva.

A	D7	D6	D5	B	0	1	0
---	----	----	----	---	---	---	---

Obrázek 5.18 S4 proměnná **pom_P0** pro první část zprávy

A	D23	D22	D21	B	0	1	0
---	-----	-----	-----	---	---	---	---

Obrázek 5.19 S4 proměnná pom_P0 pro druhou část zprávy

0	0	D10	D9	D8	0	0	0
---	---	-----	----	----	---	---	---

Obrázek 5.20 S4 prichozí_data po stavu S4

Dále pokračujeme předposledním stavem **S5**. Master žádá kombinaci 001. Do pomocné proměnné vložíme buď informaci o tlačítkách 4, 3, 2, a nebo 20, 19, 18. Zpracujeme příchozí data z PLC stejným způsobem, jak tomu bylo v ovladači patra. Výsledné data pak vypadají takto:

A	D4	D3	D2	B	0	0	1
---	----	----	----	---	---	---	---

Obrázek 5.21 S5 proměnná pom_P0 pro první část zprávy

A	D20	D19	D18	B	0	0	1
---	-----	-----	-----	---	---	---	---

Obrázek 5.22 S5 proměnná pom_P0 pro druhou část zprávy

0	0	0	D10	D9	D8	D7	D6
---	---	---	-----	----	----	----	----

Obrázek 5.23 S5 prichozí_data po stavu S5

0	0	0	0	D5	0	0	0
---	---	---	---	----	---	---	---

Obrázek 5.24 S5 prichozí_data po stavu S5

Po odeslání dat portem **P0** se dostáváme do posledních stavu **S6**. V tomto stavu master žádá poslední kombinaci, a to 000. Nyní odesíláme poslední části zprávy. Podle rozlišovacího bitu **castzpravy** určíme, jestli máme do **pom_P0** vložit informaci o tlačítku 1 nebo tlačítku 17. Bit overload je vždy v log 0 a verify v log 1. Dále zpracujeme poslední část příchozích dat. S **prichozí_patro** se již nic nedělá. Příchozí data se připojí do proměnné **prichozí_data**. Pro provedení stavu **S6** data vypadají následovně:

A	D1	O	V	B	0	0	0
---	----	---	---	---	---	---	---

Obrázek 5.25 S6 proměnná pom_P0 pro první část zprávy

A	D17	O	V	B	0	0	0
---	-----	---	---	---	---	---	---

Obrázek 5.26 S6 proměnná pom_P0 pro druhou část zprávy

0	D5	D4	D3	D2	0	0	0
---	----	----	----	----	---	---	---

Obrázek 5.27 S6 prichozí_data po stavu S6

Po tomto stavu se proměnná **stav** nastaví na 0, aby se program mohl znova dostat do stavu **S0**. Proměnná **castzpravy** se zneuguje. Celý proces se opakuje znova.

5.4 Popis zobrazení na displeji v kabině

Na displeji v kabině se objevují stejné informace, jako na displejích v jednotlivých patrech. Výjimkou je informace vyobrazené při poruše. V případě, že je kabina nezatížená, se na displeji objeví stejný nápis a to:



Obrázek 5.28 Informace Error

Pokud došlo k poruše a PLC nám posílá číslo patra 31. To znamená, že v kabině výtahu někdo je (nebo je něčím zatížená), na displeji se objeví rotující nápis: „**DOSLO K PORUSE PORUCHU RESIME**“. Po ukončení poruchy musíme počkat než nápis „projede“ až na konec. Poté se již začnou na displeji zobrazovat platné informace.

6 TESTOVACÍ PROGRAM PRO PLC

Posledním úkolem bylo ověřit funkčnost ovladačů pomocí PLC. Program neřeší řízení výtahu, pouze zajišťuje komunikaci mezi PLC a ovládacími panely v patrech a panelem v kabině.

6.1 Komunikace a slavy

Komunikace po sběrnici AS-Interface je zajištěna díky modulu firmy Bihl+Wiedemann[8]. Tento modul komunikuje podle následující tabulky:

word	2 ¹⁵	2 ¹⁴	2 ¹³	2 ¹²	2 ¹¹	2 ¹⁰	2 ⁹	2 ⁸	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰
circuit 1:																
0	flags				Slave 1/1A				Slave 2/2A				Slave 3/3A			
	F3	F2	F1	F0	D3	D2	D1	D0	D3	D2	D1	D0	D3	D2	D1	D0
1	Slave 4/4A				Slave 5/5A				Slave 6/6A				Slave 7/7A			
2	Slave 8/8A				Slave 9/9A				Slave 10/10A				Slave 11/11A			
...			
15	Slave 28B				Slave 29B				Slave 30B				Slave 31B			
circuit 2:																
...																

Tabulka 6.1 Řazení dat AS-I mastera[8]

Dále je důležité vědět, jaké adresy mají použité slavy a kde jsou připojeny. Podle toho se musí vytvořit celý program.

SLAVE 1	Zajišťuje komunikaci s ovládacím panelem v prvním patře
SLAVE 2	Zajišťuje komunikaci s ovládacím panelem v druhém patře
SLAVE 3	Zajišťuje komunikaci s ovládacím panelem ve třetím patře
SLAVE 4	Zajišťuje komunikaci s ovládacím panelem ve čtvrtém patře
SLAVE 5	Zajišťuje komunikaci s panelem v kabině
SLAVE 6	Zajišťuje komunikaci s panelem v kabině
SLAVE 7	Senzor zatížení
SLAVE 8	Slouží k ovládání frekvenčního měniče
SLAVE 10	Používá se k nastavování rychlosti motoru

Tabulka 6.2 Přehled použití jednotlivých slavů výtahu

6.2 Důležité proměnné

Programovací prostředí RSLogix 5000 nabízí velké množství proměnných. V této kapitole pouze vypíšu jejich přehled, s vysvětlením co znamenají. V programu je použito proměnných mnohem více, tady zmíním jen ty důležité pro komunikaci.

Jméno	Datový typ	Popis
_I_K_DVERE	BOOL	Proměnná signalizuje zavření dveří v kabině
_I_K_SNIMAC_1	BOOL	První snímač polohy kabiny, proměnná je v log.1 pokud je snímač sepnutý
_I_K_SNIMAC_2	BOOL	Druhý snímač polohy kabiny, proměnná je v log.1 pokud je snímač sepnutý
_I_K_STOP	BOOL	Signalizuje zmáčknutí stop tlačítka v kabině
_I_K_TLACITKA	DINT	Do této proměnné se ukládají všechny zmáčknuté tlačítka v kabině
_I_K_ZATIZENI	INT	Tato proměnná signalizuje zatížení prostoru kabiny kde: 0 - Prázdná kabina, 1 - Zatížená kabina, 2 - Plná kabina, 3 - Přetížená kabina
_I_K_ZVONEK	BOOL	Signalizace zmáčknutí žlutého tlačítka v kabině výtahu
_I_TLACITKO_DOLU	INT	Zde se zobrazují všechny zmáčknuté tlačítka „dolů“ na ovládacích panelech v patrech
_I_TLACITKO_NAHORU	INT	Zde se zobrazují všechny zmáčknuté tlačítka „nahoru“ na ovládacích panelech v patrech
_MOT_RYCHLOST	INT	Do této proměnné se vkládá číslo, jakou rychlostí má výtah jet: 0-5000 stojí, 6000-20000 jede
_MOT_SMER	BOOL	Zde dáme log.1, pokud chceme jet nahoru, popřípadě log.0, když má kabina směřovat nahoru
_MOT_START	BOOL	Nastavením log.1, zapneme chod frekvenčního měniče a uvedeme tím motor do chodu
_O_CisloP	INT	V této proměnné je uloženo číslo patra, které se odesílá do ovládacích panelů

Jméno	Datový typ	Popis
_O_JEDE	BOOL	Proměnná, která říká panelům, že se výtah pohybuje
_O_PORUCHA	BOOL	Proměnná, která říká panelům, že došlo k poruše
_O_SmerD	BOOL	Je v log.1 pokud výtah směřuje dolů
_O_SmerN	BOOL	Je v log.1 pokud výtah směřuje nahoru
_RUN_DEMO	BOOL	Slouží k zapnutí ukázky programu (výtah musí být po inicializaci)
_RUN_INIT	BOOL	Spouští inicializace výtahu. Výtah se dostane do výchozí pozice.

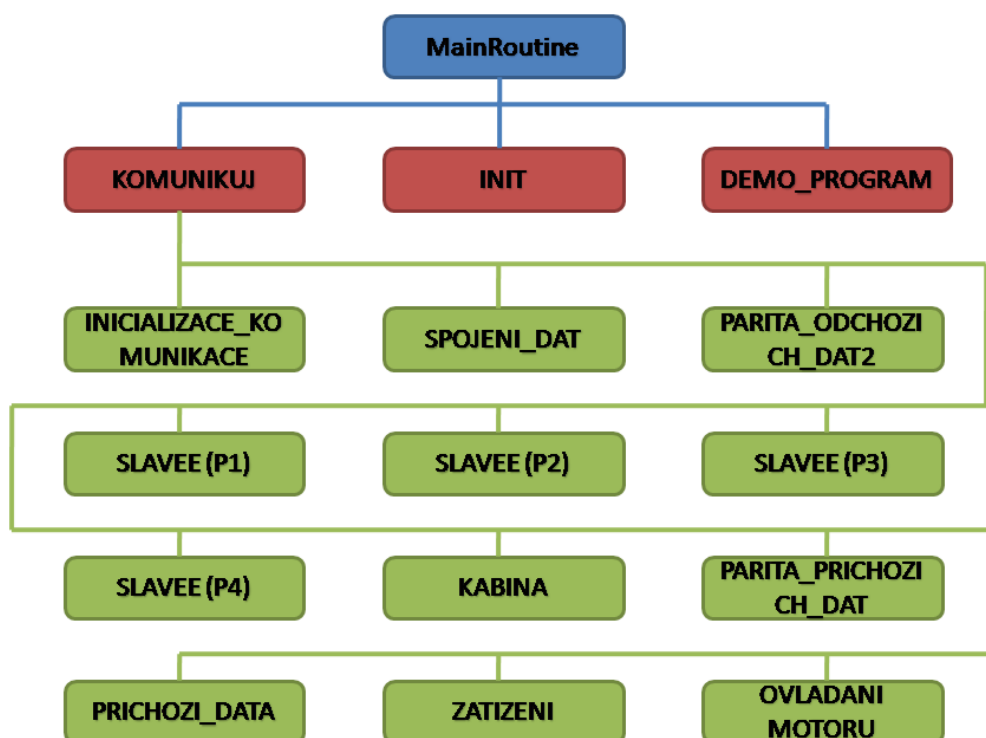
Tabulka 6.3 Seznam důležitých proměnných

6.3 Program pro PLC

Program je vytvářen z ukázkového příkladu komunikace modulu firmy Bihl+Wiedemann[8]. Tento příklad řeší komunikaci se slavy, a proto k nim můžeme přistupovat jako ke globálním proměnným. Jako první jsem si podle tabulky, která je obsažena v dokumentaci k AS-I masteru[8] vytvořil aliasy těchto proměnných, abych k nim nemusel přistupovat přes jejich adresy. Tyto aliasy se nachází v tabulce „ControllerTags“.

Všechny rutiny jsou naprogramovány pomocí Ladder Diagramu. Výjimku tvoří subrutina **PARITA_ODCHOZICH_DAT2** ta je vytvořena v FunctionBlock Diagramu.

Nyní se podíváme na strukturu programu. Hlavní rutinou je rutina „**MainRoutine**“, která je něco jako hlavní funkce. V té jsou umístěny tři subrutiny (funkce). Je to subrutina **KOMUNIKUJ**, která řídí komunikaci se všemi periferiemi. Subrutina **INIT**, která zinicilizuje výtah a subrutina **DEMO_PROGRAM**, která provádí ukázkový program.



Obrázek 6.1 Struktura programu pro PLC

- **MainRoutine**

Hlavní rutina nyní obsahuje pouze tři subrutiny. **KOMUNIKUJ** se provádí vždy. Po nastavení proměnné **_RUN_INIT** do log.1 se zapne subrutina **INIT**, která dostane výťah do jeho výchozí pozice. Po úspěšné inicializaci se nastavím bit **INIT_OK** do log.1, což zabrání znovuspuštění funkce **INIT** a povolí zapnutí poslední funkce. Po úspěšné inicializaci a nastavením bitu **_RUN_DEMO** se začne provádět ukázková subrutina **DEMO_PROGRAM**. Místo ukázkového programu by v této rutině měl být umístěn program ovládající řízení výťahu.

- **DEMO_PROGRAM**

Tato rutina v podstatě ukazuje „možnosti“, které může výťah dělat, co se může na displeji objevit, a demonstuje funkčnost komunikace.

Současný program vysílá 5 s, že pojede nahoru, poté vyjede do čtvrtého patra, kde počká 5 s, následuje oznamování 5 s, že pojede dolů, a sjede do patra druhého, kde zastaví a stojí. Během jízdy lze kabinu zastavit stisknutím tlačítka Stop. Po stisku jednoho ze čtyř tlačítek pro volbu patra se stop „odblokuje“ a výťah pokračuje dál.

- **INIT**

Funkce sloužící k tomu, aby kabina výtahu sjela do výchozí pozice do prvního patra. Kabina má v prvním patře nastavit do proměnné **_O_CisloP** číslo jedna a vypnout frekvenční měnič s motorem.

Kabině nastavím rychlost jízdy **MOT_RYCHLOST** na „9000“ směr jízdy **_MOT_SMER** na „0“ dolů a zapnu motor tím, že proměnnou **_MOT_START** nastavím na „1“. Program pozná, že se kabina nachází v prvním patře tak, že dojde nejprve k sepnutí prvního snímače (**_I_K_SNIMAC_1**) a poté obou. Tato kombinace vede k zastavení kabiny, vypnutí frekvenčního měniče, nastavení rychlosti na 4000 (stojí) a nastavení čísla patra na 1. Jako poslední se nastaví bit **INIT_OK** a tím je inicializace úspěšně provedena.

- **KOMUNIKUJ**

KOMUNIKUJ je subrutina, která řídí a ovládá veškerou komunikaci po sběrnici AS-Interface. Zapouzdřuje několik dalších subrutin, které jsou pro komunikaci nezbytné.

- **SPOJENI_DAT**

Data, která vstupují do subrutiny, jsou spojena do jedné šestnáctibitové proměnné. Jedná se o číslo patra (**_O_CisloP**), směry jízdy (**_O_SmerN** a **_O_SmerD**), o stav kabiny jestli jede (**_O_JEDE**) a o poruchový bit (**_O_PORUCHA**). Výstupní proměnná **MY_DATA_BEZ_PAR** pak vypadá následovně:

0	0	0	0	0	0	CP	CP	CP	CP	CP	N	D	Jede	Por	0
---	---	---	---	---	---	----	----	----	----	----	---	---	------	-----	---

Obrázek 6.2 Proměnná **MY_DATA_BEZ_PAR**

Do subrutiny jsou již integrovány funkční spojení. V případě, že je kabina přetížená (**_I_K_ZATIZENI** = 3) se odesílá číslo patra 0. V kabině dojde k rozsvícení signalizační led diody. Pokud došlo k poruše a kabina je zatížená, vysílá se číslo patra 31, což znamená že se má na displeji objevit rotující nápis „**DOSLO K PORUSE PORUCHU RESIME**“.

▪ PARITA_ODCHOZICH_DAT2

Jediná část programu, která je vytvořena ve FunctionBlock Diagramu. Rozhodl jsem se jej použít, abych si ho mohl vyzkoušet a navíc byl na vyhodnocení parity výhodný. Parita je zjišťována pomocí řady XORů. Bitový XOR není mezi základními funkcemi programovacího prostředí RSLogix 5000, proto jsem si ho vytvořil sám jako Add-On Instrukci. Vyhodnocením dat v **MY_DATA_BEZ_PAR** se přidá paritní bit do proměnné **MY_DATA_S_PAR** na pozici 0 a zbytek bitů se překopíruje.

0	0	0	0	0	0	CP	CP	CP	CP	CP	N	D	Jede	Por	Par
---	---	---	---	---	---	----	----	----	----	----	---	---	------	-----	-----

Obrázek 6.3 Proměnná MY_DATA_S_PAR

▪ SLAVEE

SLAVEE je Add-On instrukce, která obsluhuje komunikaci v daném patře, proto je volána 4x a to vždy s jinými vstupními proměnnými. Jediná proměnná, která se na vstupu nemění, je **MY_DATA_S_PAR**. Kromě toho do procedury vstupují příchozí data z daného slavu (**SLAVE_B1** -**SLAVE_B3**). Výstupem jsou data, která se odesílají (**SLAVE_BO0** - **SLAVE_BO3**) a informace jestli bylo v daném patře zmáčknuto tlačítko nahoru nebo dolů. Komunikace se řídí podle následující tabulky:

Stav	Příchozí data	Odchozí data
	Žádost slavu	Odpověď mastera
S0	(0)111	1 D10 D9 D8
S1	(0)010	0 D7 D6 D5
S2	(0)001	1 D4 D3 D2
S3	(0)000	0 D1 O V
S4	(0)101 (zmáčknuto ↑)	-
S4	(0)100 (zmáčknuto ↓)	-
S4	(0)011 (zmáčknuto ↑ i ↓)	-

Tabulka 6.4 Přehled jednotlivých stavů v rutině SLAVEE

▪ KABINA

Komunikace kabiny je složitější než komunikace v patrech. K posílání dat jsou využity dva slavy. Přenos dat je řízen podle tabulky:

Stav	Příchozí data		Odchozí data	
	Data pro mastera	Žádost slave	Žádost mastera	Data pro slave
	Slave 6	Slave 5	Slave 6	Slave 5
S0	A D17 O V A D1 O V	B000	K111	K D1 O V
S1	A E2 E1 E0 A E2 E1 E0	B111	K101	0 0 0 0
S2	A D16 D15 D14 A D32 D31 D30	B101	K100	0 0 0 0
S3	A D13 D12 D11 A D29 D28 D27	B100	K011	0 0 0 0
S4	A D10 D9 D8 A D26 D25 D24	B011	K010	K D10 D9 D8
S5	A D7 D6 D5 A D23 D22 D21	B010	K001	K D7 D6 D5
S6	A D4 D3 D2 A D20 D19 D18	B001	K000	K D4 D3 D2

Tabulka 6.5 Přehled jednotlivých stavů v rutině KABINA

Data, která se v této subrutině přijmou, jsou uložena do proměnné **K_DATA_PRICHOZI** (32 bitů). Kabina je nejsložitější subrutinou v celém programu.

▪ PARITA_PRICHOZICH_DAT

Do této subrutiny vstupují data (**K_DATA_PRICHOZI**), která byla přijata v subrutině **KABINA**. Zde zkontrolujeme jejich paritu. Pokud je parita lichá tak jsou data v pořádku a bit **PARITA_OK** se nastaví do logické jedničky.

▪ ZATIZENI

Účelem této části programu je převést hodnotu přicházející ze snímače, který je připojený na slave 7, na číslo od 0 do 3. Číslo určuje zatížení kabiny výtahu a je uloženo v proměnné **_I_K_ZATIZENI**. Převod je prováděn podle následující tabulky:

SLAVE 7 bit 0	SLAVE 7 bit 0	_I_K_ZATIZENI	Význam
1	0	0	Prázdná kabina
0	0	1	Zatížená kabina
0	1	2	Plná kabina
1	1	3	Přetížená kabina

Tabulka 6.6 Tabulka pro převod zatížení

▪ OVLADANI_MOTORU

Tato subrutina slouží k ovládání pohybu kabiny výtahu. Motor je ovládám dvěma slavy a to digitálním slavem 8, který nastavuje směr a zapnutí motoru, a pak analogovým slavem 10, kterým se nastaví rychlost kabiny.

Vstupem je proměnná **_MOT_SMER** sloužící k nastavení směru jízdy kabiny. Pokud tento bit je nastaven na log. 1 kabina bude směřovat směrem nahoru. V opačném případě pojede směrem dolů. Proměnná se pojí s bitem 3 na slavu 8. Dále musíme povolit start nastavením bitu 0 na slavu 8, což znamená nastavit **_MOT_START** na log. 1. Dále se musí nastavit rychlost na výstupním slavu 10 (0 -5000 stojí, 6000-20000 jede). Rychlost se nastavuje do proměnné **_MOT_RYCHLOST**. Nastavená hodnota odpovídá velikosti proudu a tím přímo ovlivňuje rychlost motoru. Skutečná rychlost pohybu kabiny se dá spočítat pomocí následujících vztahů.

Výpočet závislosti rychlosti jízdy kabiny na číselné hodnotě ovládacího proudu frekvenčního měniče[9]

Vzorec pro výpočet hodnoty ovládacího proudu frekvenčního měniče:

$$MR = \frac{v \cdot i \cdot p \cdot 16000}{f_{max} \cdot \pi \cdot d \cdot (1 - s)} \quad (1)$$

kde: *MR*..... číselná hodnota ovládacího proudu frekvenčního měniče, která se musí vložit do proměnné **_MOT_RYCHLOST** aby se kabina výtahu pohybovala rychlostí *v*

v..... rychlost jízdy kabiny [mm/s]

i..... převodový poměr [25]

p..... počet pólových dvojic motoru [2]

f_{max} maximální frekvence proudu frekvenčního měniče [35 Hz]

π	Ludolfovo číslo
d	průměr ozubeného kola [mm]
s	skluz motoru [0,1]

Výpočet skluzu motoru:

$$s = \frac{n_s - n}{n_s} = \frac{\frac{60 \cdot f}{p} - n}{\frac{60 \cdot f}{p}} = \frac{\frac{60 \cdot 50}{2} - 1350}{\frac{60 \cdot 50}{2}} = 0,1 \quad (2)$$

kde:	s	skluz motoru
	n_s	synchronní otáčky motoru [ot/min]
	n	asynchronní otáčky motoru [ot/min]
	i	převodový poměr
	p	počet pólových dvojic motoru
	f	frekvence [Hz]

Příklad výpočtu pro rychlost 50mm/s:

$$MR = \frac{v \cdot i \cdot p \cdot 16000}{f_{max} \cdot \pi \cdot d \cdot (1 - s)} = \frac{50 \cdot 25 \cdot 2 \cdot 16000}{35 \cdot \pi \cdot 80 \cdot (1 - 0,1)} = 9052 \doteq 9000 \quad (3)$$

Rychlost jízdy kabiny	Číselná hodnota ovládacího proudu
v [mm/s]	_MOT_RYCHLOST [-]
0	4000
20	6000
40	8000
50	9000
60	10000
70	11000
80	12000
90	13000
100	14000
110	15000

Tabulka 6.7 Tabulka závislost rychlosti na výstupním proudu

K zapnutí frekvenčního měniče je ještě zapotřebí poslat na 3. Bit slavu 8 zapínací impuls. Tento impuls se vygeneruje sám po nastavení proměnné **_MOT_START** na log. 1. Dále je zde propojení s některými daty, která se odesílají do ovládacích panelů.

Pokud bude rychlost menší než 5000 bude to znamenat, že kabina nejede a ani nikam nepojede (dojde k zobrazení pomlčky). V případě, že rychlost bude větší než 6000 mohou nastat čtyři možnosti:

VSTUP			Výstup			
_MOT_START	_MOT_SMER	→	_O_SmerD	_O_SmerN	_O_JEDE	Vysvětlení
0	1	→	0	1	0	Blikající šipka nahoru
0	0	→	1	0	0	Blikající šipka nahoru
1	1	→	0	1	1	Svítící šipka nahoru
1	0	→	1	0	1	Svítící šipka nahoru

Tabulka 6.8 Tabulka zobrazující logické spojení mezi proměnnými ovládajícími motor a proměnnými, které se odesílají do ovládacích panelů

V poslední části je vyřešena zněna (inkrementace, nebo dekrementace) čísla patra po sepnutí obou snímačů. Aby nedošlo k chybné změně je na konci podmínka, že pro další změnu je potřeba, aby byly oba snímače rozepnuty.

▪ INICIALIZACE_KOMUNIKACE

Subrutina sloužící k navázání komunikace s ovladačem kabiny po nahrání programu. Program zjistí, na který stav čeká ovladač kabiny a nastaví tento stav do log. 1. Po provedení program nastaví proměnnou **INICIALIZACE_OK** do log1 a poté se již neprovádí.

▪ PRICHOZI_DATA

Pokud byla parita v pořádku (**PARITA_OK=1**), tak se příchozí data „rozdělí“ do příchozích proměnných.

Příchozí data	Proměnná v programu
K_DATA_PRICHOZI.0	_I_K_TLACITKA.0
K_DATA_PRICHOZI.1	_I_K_TLACITKA.1
K_DATA_PRICHOZI.2	_I_K_TLACITKA.2
K_DATA_PRICHOZI.3	_I_K_TLACITKA.3
K_DATA_PRICHOZI.24	_I_K_SNIMAC_1
K_DATA_PRICHOZI.25	_I_K_SNIMAC_2
K_DATA_PRICHOZI.26	_I_K_STOP
K_DATA_PRICHOZI.27	_I_K_ZVONEK

Tabulka 6.9 Přiřazení příchozích dat k proměnným

V případě, že by měl výtah více pater než čtyři by se muselo přidat propojení K_DATA_PRICHOZI.4 - K_DATA_PRICHOZI.23 pro patra 5 - 24. V našem případě by to bylo zbytečné. Dále je tam problémová proměnná K_DATA_PRICHOZI.29, která signalizuje zavření dveří (**_I_K_DVERE**). Protože s tlačítkem dveří zprvu nebylo počítáno a přidávali ho tam dodatečně, dochází při jeho použití k problémům. Proto jsem tlačítko v programu nechal, ale proměnnou **_I_K_DVERE** bych se dveřmi nepropojoval a místo toho bych do modelu připojil indukční snímač, kterým bych simuloval zavření dveří.

Ještě v této subrutině dochází ke zpracování „rychlých“ informací, označených jako bity **A** a **B**. „**A**“ což je 3. vstupní bit ve slavu 6 (SLAVE6_IN3) pokud je nastaven na log.1 znamená to, že jsou sepnuty oba snímače polohy. To znamená, že kabina se nachází v některém z pater. „**B**“ je 3. vstupní bit slavu 5 (SLAVE5_IN3). V případě, že je nastaven na log. 1 signalizuje, že bylo zmáčknuto stop tlačítko.

6.4 Snímač dveří

Vzhledem k tomu, že tlačítko pro simulaci zavření dveří bylo na ovládací panel kabiny přidáváno dodatečně a při zpracování jeho signálu docházelo k chybám, rozhodli jsme se tlačítko z komunikace vynechat. Na místo simulovaného tlačítka jsme se rozhodli umístit skutečný snímač. Protože snímač má být umístěn v kabině výtahu, musí být připojen na sběrníkový systém AS-Interface přímo, což znamená, že musí mít integrované rozhraní AS-Interface. Snímač by měl být vazební, malého rozměru, aby se dal integrovat do kabiny výtahu. Jeho spínací vzdálenost by neměla být větší než 4 mm. Těmto požadavkům vyhovuje indukční snímač IFC247 od firmy IFM ELECTRONIC [10].

Parametry snímače IFC247:

Typ:	Indukční senzor
Provedení:	Kovové (kovový závit M12)
Spínací vzdálenost:	4 mm
Pracovní vzdálenost:	0...3,25 mm
Krytí:	IP 68
AS-i Verze:	2.1

Tabulka 6.10 Parametry snímače IFC247 [10]

Obsazení datových bitů:

Datový bit	D0	D1	D2	D3
Stav=0	Médium mimo aktivní oblast	Nepoužito	Nepoužito	Nepoužito
Stav=1	Médium v rozsahu aktivní oblasti	Chybná hodnota	Chybná hodnota	Chybná hodnota

Tabulka 6.11 Obsazení datových bitů snímače IFC247 [10]

6.5 Implementace programu pro řízení výtahu

V této kapitole je uvedeno jak dále postupovat v implementaci řídicího algoritmu pro řízení výtahu. Ovládací rutina by měla být umístěna v hlavní rutině MAIN a mělo by dojít k jejímu spuštění, až po úspěšné inicializaci.

Nejprve se podíváme, jaké máme vstupy a výstupy:

Vstupy		
Jméno	Datový typ	Popis
_I_K_DVERE	BOOL	Snímač dveří (Proměnná pro budoucí snímač dveří)
_I_K_SNIMAC_1	BOOL	Příchozí signál ze snímače polohy 1
_I_K_SNIMAC_2	BOOL	Příchozí signál ze snímače polohy 2
_I_K_TLACITKA	DINT	V této proměnné jsou aktuálně zmáčknutá tlačítka, pro volbu patra v kabině výtahu
_I_K_ZATIZENI	INT	Tato proměnná signalizuje zatížení prostoru kabiny kde: 0 - Prázdná kabina, 1 - Zatížená kabina, 2 - Plná kabina, 3 - Přetížená kabina
_I_K_ZVONEK	BOOL	Signál zmáčknutí tlačítka zvonek v kabině
_I_K_STOP	BOOL	Signál zmáčknutí stop tlačítka v kabině
_I_TLACITKO_DOLU	INT	Zde se zobrazují všechny aktuálně zmáčknuté tlačítka dolů, na ovládacích panelech v patrech
_I_TLACITKO_NAHORU	INT	Zde se zobrazují všechny aktuálně zmáčknuté tlačítka nahoru, na ovládacích panelech v patrech
_O_CisloP	INT	Tato proměnná odesílá do ovládacích panelů pozici kabiny, zároveň je aktualizovaná pomocí dat přijatých z kabiny (automatická inkrementace a dekrementace čísla patra)

Tabulka 6.12 Tabulka vstupů

Výstupy		
Jméno	Datový typ	Popis
_MOT_RYCHLOST	INI	V této proměnné je uložena číselná hodnota ovládacího proudu frekvenčního měniče, která ovlivňuje rychlost jízdy kabiny. (0-5000 stojí, 6000-20000 jede)
_MOT_SMER	BOOL	Zde dáme log.1, pokud chceme jet nahoru, popřípadě log.0, když má kabina směřovat dolů
_MOT_START	BOOL	Nastavením log.1, zapneme chod frekvenčního měniče a uvedeme tím motor do chodu
_O_PORUCHA	BOOL	Poruchový bit, který oznamuje ovládacím panelům, že došlo k poruše.

Tabulka 6.13 Tabulka výstupů

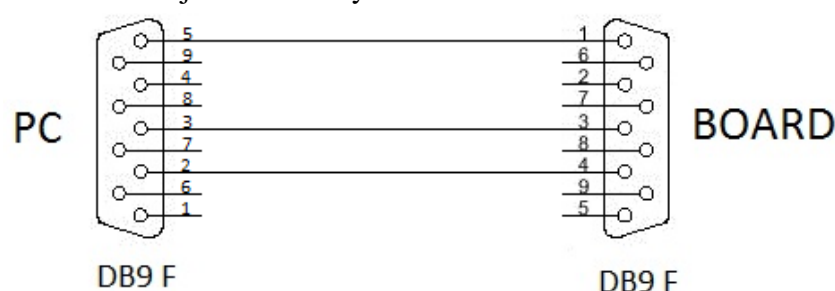
V těchto tabulkách jsou uvedeny veškeré proměnné důležité k vytvoření řídicího algoritmu modelu výtahu. Vstupní proměnné se aktualizují samy díky rutině KOMUNIKUJ, která řídí celou komunikaci. Pro celé ovládání výtahu stačí využít čtyř výstupních proměnných.

7 NÁVOD K PROGRAMOVÁNÍ MIKROPROCESORU

Tato kapitola se zabývá tím, jakým způsobem lze panely v patrech a panel v kabině naprogramovat.

7.1 Programovací kabel

Jako první a nejdůležitější je si vyrobit programovací kabel. Při programování stačí využít pouze 3 vodiče a to Tx, Rx a GND. Zapojení těchto vodičů je trochu netypické, proto zde přidávám náskres jak si kabel vyrobit.

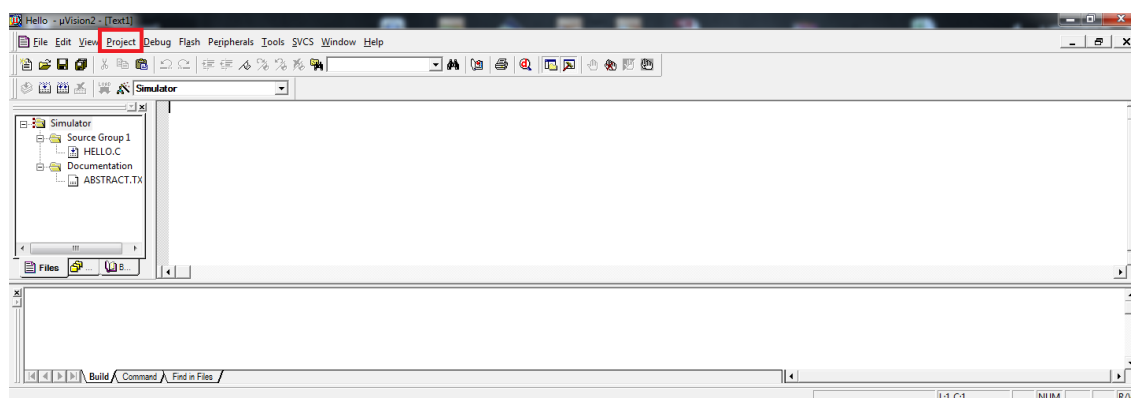


Obrázek 7.1 Schéma zapojení programovacího kabelu pro model výtahu

Konektory použité v zapojení jsou **oba** typu CAN9 female.

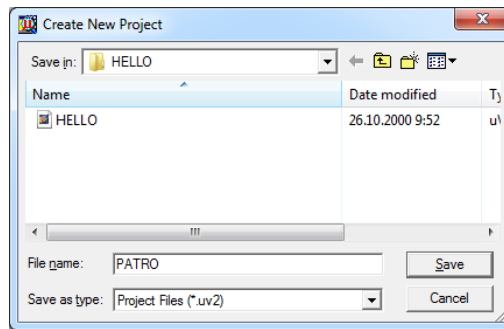
7.2 Kompilátor Keil

Při programování jsem používal program KeilVision 2. Po zapnutí se nám objeví vývojové prostředí programu.



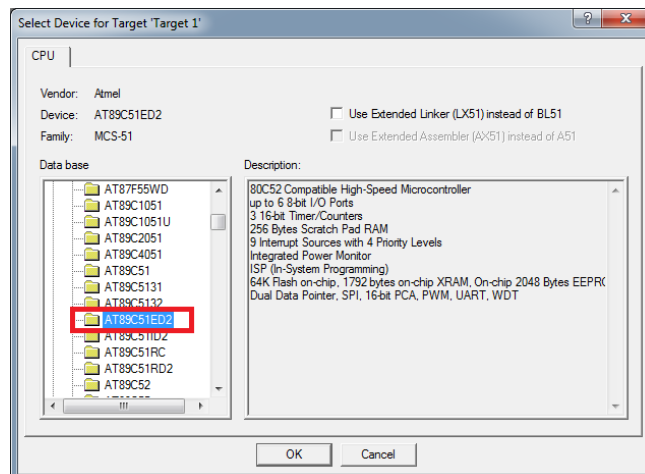
Obrázek 7.2 Programovací prostředí programu KeilVision

Zvolíme „založit nový projekt“, který pojmenujeme např. PATRO.



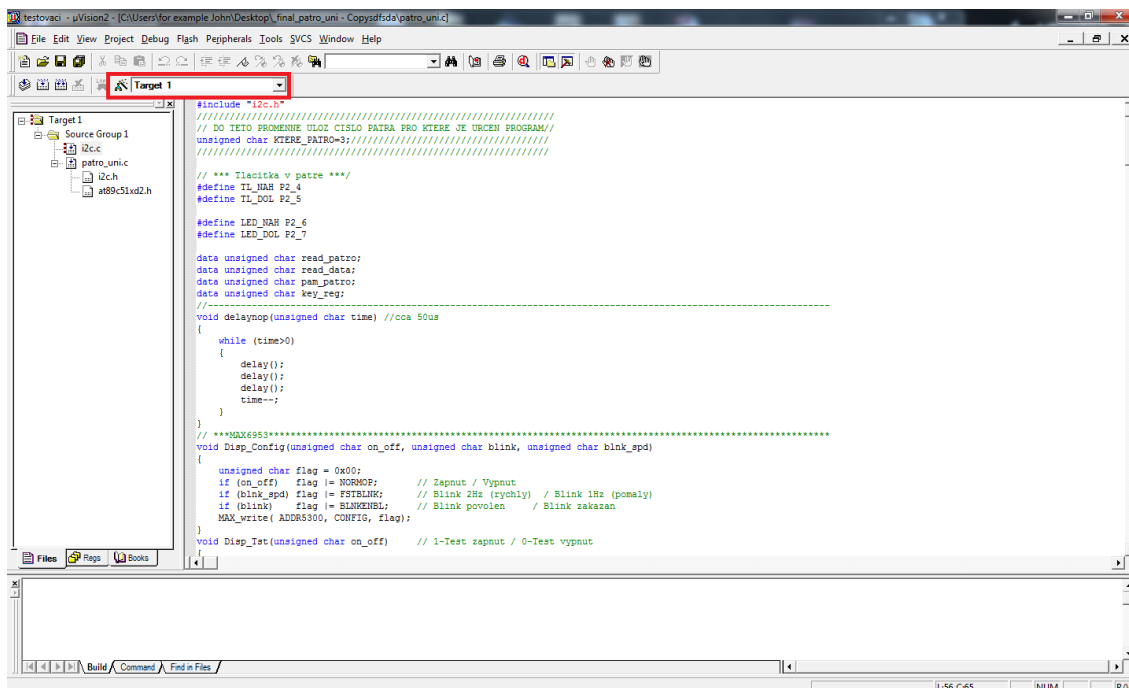
Obrázek 7.3 Založení nového projektu v programu Keil

Vybereme druh programovaného mikroprocesoru. V našem případě se jedná o mikroprocesor AT89C51ED2.



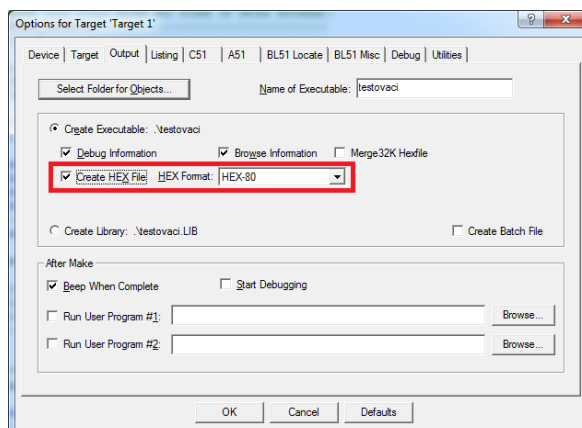
Obrázek 7.4 Vybrání programovaného mikrouprocesoru v programu Keil

Po zvolení mikroprocesoru můžeme začít programovat. Po naprogramování musíme program zkompilovat a nastavit výstupní HEX file, který budeme nahrávat do mikroprocesoru.



Obrázek 7.5 Programování v prostředí Keil

Po zmáčknutí tlačítka „Target“ se nám objeví panel „Options for Target“. Na tomto panelu zvolíme záložku „Output“ ve které zatrhneme pole „Create HEX File“.



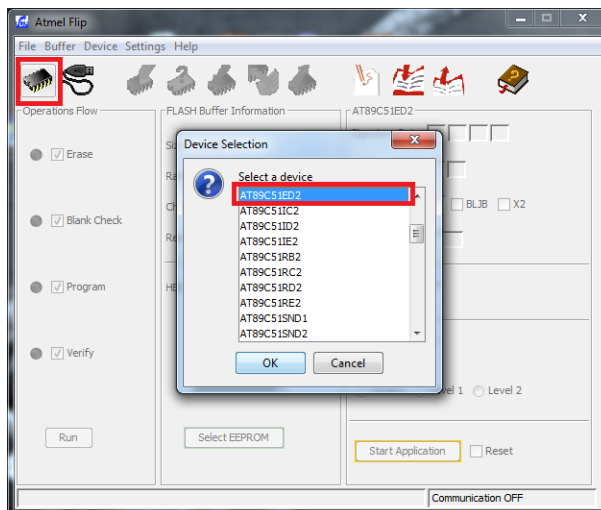
Obrázek 7.6 Vytvoření HEX File v programu Keil

Po znovu zkompileování projektu, který je bez chyby, se nyní objeví požadovaný HEX File.

7.3 Flip

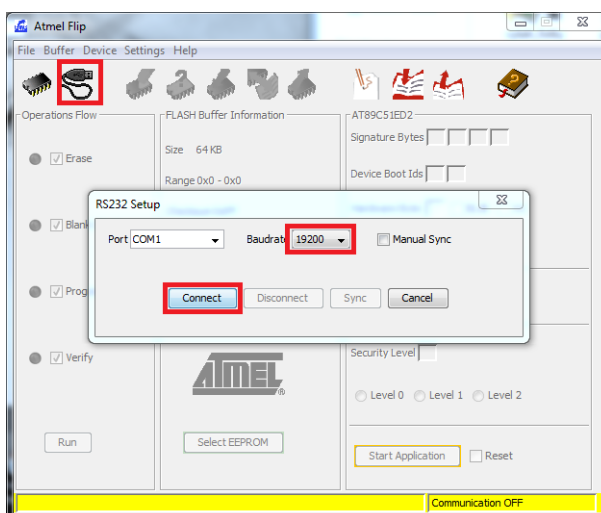
Pokud ji máme vytvořen HEX File, můžeme se pustit do nahrávání programu do mikroprocesoru. K nahrání budeme potřebovat program Flip. Používal jsem verzi

Flip 3.4.3. Po zapnutí programu se nám objeví základní okno. Kde klikneme na ikonu integrovaného obvodu a zvolíme náš programovaný mikroprocesor AT89C51ED2. Volbu potvrdíme.



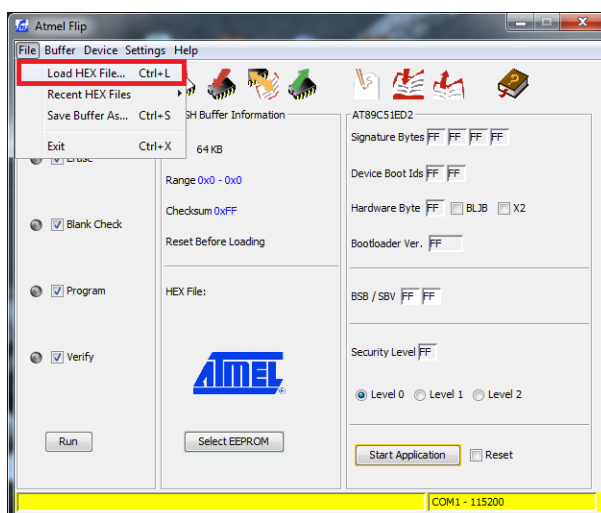
Obrázek 7.7 Výběr mikroprocesoru v programu Flip

Pokračujeme kliknutím na ikonu „se šňůrou“ a zde zvolíme, přes který COM se uskuteční spojení. Dále nastavíme přenosovou rychlost na „19200“, spojení se uskuteční pouze a jen při této rychlosti. Zkontrolujeme, jestli máme programovací kabel připojen správně. Ještě před tím, než zmáčkne tlačítko „Connect“ musíme zapnout „bootloader“, to provedeme podržením tlačítka reset a zmáčknutím tlačítka PSEN. Pokud jsme toto udělali správně, dojde k rozsvícení obou podsvětlení tlačítek v patrech a k rozsvícení podsvětlení tlačítek stop a zvonek v panelu kabiny. Pokud diody svítí, můžeme zmáchnout tlačítko „Connect“ a dojde k připojení k mikroprocesoru.



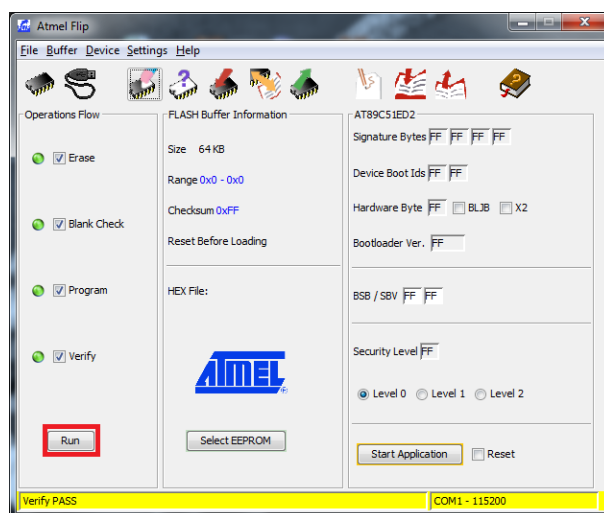
Obrázek 7.8 Nastavení komunikace v programu Flip

Nyní v záložce „File“ zvolíme „Load HEX File....“ a načteme námi vytvořený soubor , který jsme vytvořili pomocí programu Keil.



Obrázek 7.9 Načtení HEX File v programu Flip

Po načtení stiskneme tlačítko „Run“. Dojde k vymazání paměti, ověření vymazání, nahrání programu a ověření nahrání do mikroprocesorové paměti.



Obrázek 7.10 Nahrání programu do mikroprocesoru

Nyní stačí pouze zmáčknout reset na desce a program je spuštěn.

8 ZÁVĚR

V této bakalářské práci jsem se seznámil s komunikací PLC a sítí AS-Interface. Prostudoval jsem, jak by měla komunikace modelu výtahu vypadat. Pro jednotlivá patra a kabinu jsem navrhl stavový automat. Oba stavové automaty jsou Moorova typu. Stavový automat byl pro všechna patra totožný.

Po nastudování jsem se posunul k programování komunikačního algoritmu. Firmware ovládačů je naprogramován v jazyce C. Projekt jsem realizoval ve vývojovém prostředí Keil uVision2. Toto vývojové prostředí je určeno k programování mikroprocesorů. Program jsem nahrával do mikroprocesoru pomocí aplikace Flip 3.4.3.

Zpočátku jsem investoval nemalý čas do bádání nad správným zapojením programovacího kabelu, a proto jsem jeho zapojení a následovný postup pro naprogramování připojil do projektu. Nejdůležitější částí projektu bylo vytvoření firmwaru pro ovládače. Některé procedury jsem převzal z předchozí práce Lukáše Zástěry[3]. Jednalo se o procedury spojené s inicializací displeje, nastavení skenovacích obvodů tlačítek a procedury spojené s ovládáním podsvětlení tlačítek. Ostatní procedury jsem sám naprogramoval. Dále jsem upravil knihovnu i2c.h, protože obsahovala těla funkcí.

Programování ovladače kabiny bylo již složitější než programování ovladače pater, jelikož kabina odesílá mnohem více dat. Po zvládnutí komunikačního problému, bylo nejtěžší promyslet, jak ošetřit informaci o zavřených dveřích kabiny. Dveře jsou simulovány pomocí tlačítka, jehož signál není připojen jako externí přerušení. Při zpracování signálu dveří docházelo k chybám v komunikaci, proto jsem se rozhodl, že se simulované tlačítko dveří v budoucnu nahradí snímačem IFC247 od firmy IFM ELECTRONIC.

Všechny programy jsem se snažil přehledně okomentovat, aby se v nich dalo snadno orientovat.

Po vytvoření firmwaru jsem vytvořil testovací program pro PLC. Tento program měl za úkol nejen komunikovat s ovládači ale i s ostatními periferiemi. Výsledný program obsahuje tři hlavní subrutiny: KOMUNIKUJ, INIT a DEMO_PROGRAM. Celá komunikace je řízena subrutinou KOMUNIKUJ. INIT slouží k prvotní inicializaci výtahu. Po úspěšné inicializaci je možno spustit ukázkovou rutinu DEMO_PROGRAM.

Struktura programu pro PLC je navržena tak, aby bylo možné rutinu DEMO_PROGRAM nahradit řídicím algoritmem, který bude ovládat model výtahu.

Literatura

- [1] AS-Interface (*AS-interface Česká republika*), AS-Interface [on-line], Dostupné na www.as-interface.cz/Seminar/AS-Interface_V3_0.pdf[cit. 2011-12-10].
- [2] BECKER, Rolf , et al. *AS-Interface, Řešení pro automatizaci*. AS-International Association. Schweinfurt : WeppertGmbH, 2002. 184 s.
- [3] ZÁSTĚRA, Lukáš. *AS-Interface slave pro klec a patra modelu výtahu*. Brno, 2006. 88 s. Diplomová práce. VUT v Brně.
- [4] *MAX6953 Matrix LED Display Driver* [datasheet]. USA : Maxim Integrated Products, 2002. Dostupné z WWW: <http://datasheets.maxim-ic.com/en/ds/MAX6953.pdf> [cit. 2011-12-5].
- [5] *MAX6955 Driver with I/O Expander and KeyScan*[datasheet]. USA : Maxim Integrated Products, 2008. Dostupné z WWW: <http://datasheets.maxim-ic.com/en/ds/MAX6955.pdf>[cit. 2011-12-5].
- [6] *MAX6955 28-Port LED Display Driver and I/O Expander*[datasheet]. USA : Maxim Integrated Products, 2003. Dostupné z WWW: <http://datasheets.maxim-ic.com/en/ds/MAX6955.pdf>. [cit. 2011-12-5].
- [7] ŽŮREK, M. *Model výtahu s rozhraním AS-Interface*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 48 s. Bakalářská práce.
- [8] *Bihl+Wiedemann GmbH: AS-i Master/Scanner for ALLEN-BRADLEY ControlLogix*[online]. Poslední aktualizace 26.2.2009. Dostupné z: <http://www.bihlwiedemann.de/cgi-local/byteserver.pl/document/abcoe305.pdf>. [cit. 2012-03-24].
- [9] ŽŮREK, M. *Řízení modelu výtahu*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 63 s. Diplomová práce. VUT v Brně.
- [10] *Ifm electronic Česká republika: IFC247*[online]. Dostupné z: <http://www.ifm.com/products/cz/ds/IFC247.htm>. [cit. 2012-04-25].

SEZNAM OBRÁZKŮ

<i>Obrázek 2.1 AS-I sběrnice [2].....</i>	<i>11</i>
<i>Obrázek 2.2 Hvězda [2].....</i>	<i>12</i>
<i>Obrázek 2.3 Kruh [2].....</i>	<i>12</i>
<i>Obrázek 2.4 Strom [2]</i>	<i>12</i>
<i>Obrázek 2.5 Lineární s odbočkami[2]</i>	<i>12</i>
<i>Obrázek 2.6 Master-Slave komunikace [2].....</i>	<i>14</i>
<i>Obrázek 2.7 Komunikační profil S7.3[1]</i>	<i>14</i>
<i>Obrázek 3.1 Základní zapojení komunikace.....</i>	<i>16</i>
<i>Obrázek 4.1 Obrázek ovladače patra s popisky.....</i>	<i>18</i>
<i>Obrázek 4.2 Stavový popis ovladače patra</i>	<i>18</i>
<i>Obrázek 4.3 Znáznornění komunikace patra na portu P1</i>	<i>22</i>
<i>Obrázek 4.4 Proměnná prchozi _patro po stavu 0</i>	<i>22</i>
<i>Obrázek 4.5 Proměnná prchozi _patro po stavu 1</i>	<i>23</i>
<i>Obrázek 4.6 Proměnná prchozi _data po stavu 1</i>	<i>23</i>
<i>Obrázek 4.7 Proměnná prchozi _data po stavu 2</i>	<i>23</i>
<i>Obrázek 4.8 Proměnná prchozi _patro ve stavu 3.....</i>	<i>23</i>
<i>Obrázek 4.9 Proměnná prchozi _data ve stavu 3</i>	<i>23</i>
<i>Obrázek 4.10 Proměnná prchozi _data ve stavu 3 po úpravě.....</i>	<i>23</i>
<i>Obrázek 5.1 Obrázek ovladače kabiny s popisky.....</i>	<i>25</i>
<i>Obrázek 5.2 Stavový popis ovladače kabiny</i>	<i>26</i>
<i>Obrázek 5.3 Znáznornění komunikace kabiny na portech P0 a P1</i>	<i>28</i>
<i>Obrázek 5.4 key_pat [2]</i>	<i>29</i>
<i>Obrázek 5.5 key_pat [1]</i>	<i>29</i>
<i>Obrázek 5.6 key_pat [0]</i>	<i>29</i>
<i>Obrázek 5.7 S0 pomocná pom _P0 po přiřazení bitů E1-E3.....</i>	<i>32</i>
<i>Obrázek 5.8 S0 pomocná pom _P0 po přiřazení bitů A a B</i>	<i>32</i>
<i>Obrázek 5.9 S0 proměnná prchozi _patro.....</i>	<i>32</i>
<i>Obrázek 5.10 S0 proměnná prchozi _data</i>	<i>32</i>
<i>Obrázek 5.11 S0 proměnná pom _P0 před odesláním.....</i>	<i>32</i>
<i>Obrázek 5.12 S1 proměnná pom _P0 pro první část zprávy</i>	<i>33</i>
<i>Obrázek 5.13 S1 proměnná pom _P0 pro druhou část zprávy.....</i>	<i>33</i>

<i>Obrázek 5.14 S2 proměnná pom_P0 pro první část zprávy.....</i>	<i>33</i>
<i>Obrázek 5.15 S2 proměnná pom_P0 pro druhou část zprávy.....</i>	<i>33</i>
<i>Obrázek 5.16 S3 proměnná pom_P0 pro první část zprávy.....</i>	<i>33</i>
<i>Obrázek 5.17 S3 proměnná pom_P0 pro druhou část zprávy.....</i>	<i>33</i>
<i>Obrázek 5.18 S4 proměnná pom_P0 pro první část zprávy.....</i>	<i>33</i>
<i>Obrázek 5.19 S4 proměnná pom_P0 pro druhou část zprávy.....</i>	<i>34</i>
<i>Obrázek 5.20 S4 prichodzi_data po stavu S4.....</i>	<i>34</i>
<i>Obrázek 5.21 S5 proměnná pom_P0 pro první část zprávy.....</i>	<i>34</i>
<i>Obrázek 5.22 S5 proměnná pom_P0 pro druhou část zprávy.....</i>	<i>34</i>
<i>Obrázek 5.23 S5prichodzi_data po stavu S5.....</i>	<i>34</i>
<i>Obrázek 5.24 S5 prichodzi_data po stavu S5.....</i>	<i>34</i>
<i>Obrázek 5.25 S6 proměnná pom_P0 pro první část zprávy.....</i>	<i>34</i>
<i>Obrázek 5.26 S6 proměnná pom_P0 pro druhou část zprávy.....</i>	<i>34</i>
<i>Obrázek 5.27 S6 prichodzi_data po stavu S6.....</i>	<i>34</i>
<i>Obrázek 5.28 Informace Error.....</i>	<i>35</i>
<i>Obrázek 6.1 Struktura programu pro PLC.....</i>	<i>39</i>
<i>Obrázek 6.2 Proměnná MY_DATA_BEZ_PAR.....</i>	<i>40</i>
<i>Obrázek 6.3 Proměnná MY_DATA_S_PAR.....</i>	<i>41</i>
<i>Obrázek 7.1 Schéma zapojení programovacího kabelu pro model výtahu</i>	<i>50</i>
<i>Obrázek 7.2 Programovací prostředí programu Keilu Vision</i>	<i>50</i>
<i>Obrázek 7.3 Založení nového projektu v programu Keil.....</i>	<i>51</i>
<i>Obrázek 7.4 Vybrání programovaného mikrouprocesoru v programu Keil</i>	<i>51</i>
<i>Obrázek 7.5 Programování v prostředí Keil</i>	<i>52</i>
<i>Obrázek 7.6 Vytvoření HEX File v programu Keil</i>	<i>52</i>
<i>Obrázek 7.7 Výběr mikroprocesoru v programu Flip</i>	<i>53</i>
<i>Obrázek 7.8 Nastavení komunikace v programu Flip</i>	<i>53</i>
<i>Obrázek 7.9 Načtení HEX File v programu Flip</i>	<i>54</i>
<i>Obrázek 7.10 Nahrání programu do mikroprocesoru</i>	<i>54</i>

SEZNAM TABULEK

<i>Tabulka 2.1 Referenční model ISO/OSI [2]</i>	13
<i>Tabulka 2.2 Přenos dat z masteru do slavu [3]</i>	15
<i>Tabulka 2.3 Přenos dat se slavu do mastera [3]</i>	15
<i>Tabulka 4.1 Komunikace v jednotlivých stavech ovladače patra obecně</i>	19
<i>Tabulka 4.2 Tabulka podmínek přechodů (patro)</i>	19
<i>Tabulka 4.3 Popis procedur pro ovladač patra</i>	21
<i>Tabulka 4.4 Komunikace v jednotlivých stavech ovladače patra ve skutečnosti</i>	22
<i>Tabulka 5.1 Princip komunikace kabiny výtahu (Výměna dat mezi masterem a slavem)</i>	27
<i>Tabulka 5.2 Tabulka podmínek přechodů (kabina)</i>	27
<i>Tabulka 5.3 Vysvětlení zkratk použitých v komunikaci kabiny</i>	31
<i>Tabulka 6.1 Řazení dat AS-I mastera[8]</i>	36
<i>Tabulka 6.2 Přehled použití jednotlivých slavů výtahu</i>	36
<i>Tabulka 6.3 Seznam důležitých proměnných</i>	38
<i>Tabulka 6.4 Přehled jednotlivých stavů v rutině SLAVEE</i>	41
<i>Tabulka 6.5 Přehled jednotlivých stavů v rutině KABINA</i>	42
<i>Tabulka 6.6 Tabulka pro převod zatížení</i>	43
<i>Tabulka 6.7 Tabulka závislost rychlosti na výstupním proudu</i>	44
<i>Tabulka 6.8 Tabulka zobrazující logické spojení mezi proměnnými ovládajícími motor a proměnnými, které se odesílají do ovládacích panelů</i>	45
<i>Tabulka 6.9 Přiřazení příchozích dat k proměnným</i>	46
<i>Tabulka 6.10 Parametry snímače IFC247 [10]</i>	47
<i>Tabulka 6.11 Obsazení datových bitů snímače IFC247 [10]</i>	47
<i>Tabulka 6.12 Tabulka vstupů</i>	48
<i>Tabulka 6.13 Tabulka výstupů</i>	49

Seznam příloh na CD

Příloha 1. Firmware pro ovládače jednotlivých pater 1 – 4

Příloha 2. Firmware pro ovládač kabiny

Příloha 3. Testovací program pro PLC