ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ v Praze

Fakulta elektrotechnická
Katedra řídicí techniky

MODEL VÝTAHU

Diplomová práce

2005 Josef Zajíc

Abstrakt

Diplomová práce popisuje realizaci elektronického modelu tří výtahů. Pojezdy kabin výtahů modelu jsou simulovány rozsvěcováním skupin sloupcových LED diod. Model je realizován s využitím mikrokontroléru PIC a je propojen s nadřazeným řídícím systémem pomocí paralelního nebo sériového rozhraní. Řízení pojezdu jenotlivých výtahů a skupinové řízení všech výtahů je realizováno programovatelným automatem PLC WAGO. V práci jsou také popsány použité technologie PLC, PIC , příslušné vývojové prostředky a je diskutována problematika nasazení sériové průmyslové sběrnice LONWORKS.

Abstract

In this diploma thesis is described realization of electronic model of three elevators. Elevator's car track is implemented with LED's bargraphs. Model is based on PIC microcontroller. Model is controlled through parallel or serial interface. Movement control and group control of elevators is implemented with PLC. There are also discussed used technologies PLC and PIC and some aspects of Fieldbus LONWORKS implementation.

Prohlášení

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

Nemám závažný důvod proti užití tohoto školního díla ve smyslu paragrafu 60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejcích s právem autorským a o znění některých zákonů (autorský zákon).

V Praze dne: Podpis:

Poděkování

Chtěl bych poděkovat všem, kdo mi přímo či nepřímo pomáhali při vzniku této práce. Zvláště pak Doc. Ing. Jiřímu Bayerovi, CSc. vedoucímu diplomové práce, Ing. Pavlu Ružičkovi odbornému poradci za cené rady při psaní práce.

Velký dík také patří rodičům a snoubence, protože mi v průběhu studia poskytovali zázemí a veškerou podporu hmotnou i duševní.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod - Rozbor zadání

Cílem diplomové práce je návrh a realizace modelu výtahu, který obsahuje minimum mechanických pohyblivých částí a kabiny jsou simulovány sloupcovými LED diodami. Další funkcí modelu je simulace čidel, která se nachází se ve skutečném výtahu. Řízení modelu na základě těchto čidel je zajištováno programovatelným automatem PLC.

1.1 Upřesnění funkcí realizovaných modelem výtahu

Po dohodě s vedoucím diplomové práce bylo určeno, že model výtahu bude realizovat

- **v kabině výtahu**
  - signalizace otevřených dveří
  - tlačítka pro každé patro a signalizace stisku tlačítka
  - tlačítko STOP
  - signalizace přetížení (bez návaznosti na váhové čidlo)
  - ukazatel směru pohybu kabiny nahoru resp. dolů
  - sedmisegmentový display zobrazující aktuální patro `[P]`,`[1]`,`[2]`,`[3]`
- **v každém patře**
  - tlačítko "směr nahoru" (kromě posledního patra) a signalizace stisku
  - tlačítko "směr dolů" (kromě přízemí) a signalizace stisku
- **čidla detekující polohu kabiny ve výtahové šachtě**
  - signalizace sepnutí čidla polohou kabiny:
    * v patře
    * na zpomalovacím/zrychlujícím bodě pod patrem
    * na zpomalovacím/zrychlujícím bodě nad patrem
    * na horním havarijním dorazu
    * na dolním havarijním dorazu

Signalizace bude provedena LED diodami. Model bude realizován pomocí 8-bitového mikrokontroléru PIC, pojezdy kabin výtahu bude řídit `PLC WAGO`.

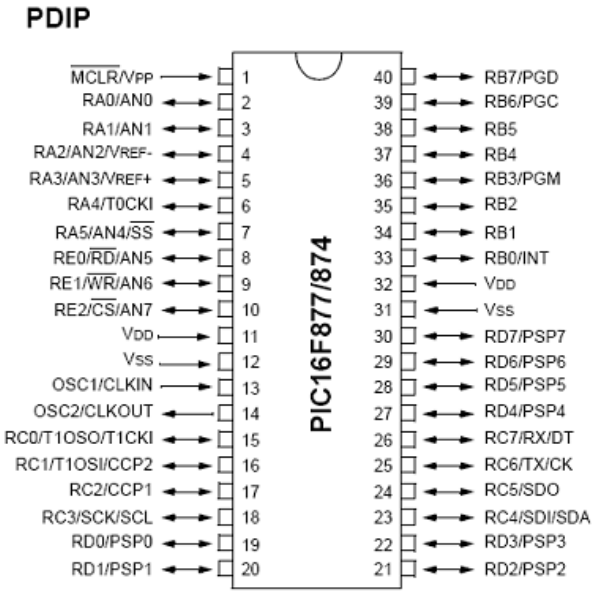
Komunikace bude zajištěna pomocí paralelní sběrnice resp. sériové sběrnice.

2 Přehled uvažovaných a použitých technologií

Tato kapitola seznamuje s nástroji a technologiemi, pomocí kterých byla diplomová práce realizována resp. které byly prostudovány a později nepoužity. Jelikož se jedná o problematiku velice rozsáhlou, bude maximálně zkrácena a doplněna o odkazy na literaturu, kde lze nalézt podrobnosti.

2.1 Mikrokontrolér PIC

Model výtahu je postaven s využitím 8-bitového mikrokontroléru PIC. Mikrokontroléry PIC byly vyvinuty firmou Microchip. Po konzultaci s vedoucím diplomové práce byl zvolen typ `PIC16F877`.



Obrázek 1: Mikrokontrolér PIC 16F877

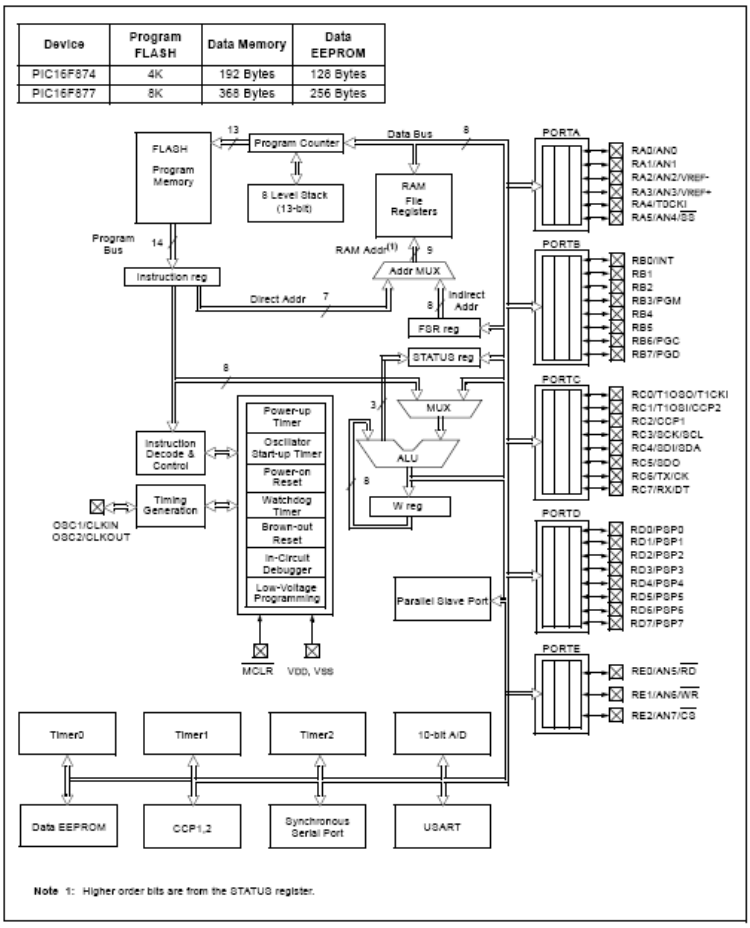
2.1.1 Architektura mikrokontroléru PIC

Společnost Microchip v posledních letech značně rozšířila rodinu mikrokontrolérů řady PIC. Všechny typy se vyznačují příznivou cenou. Převratnou novinkou jsou PIC se zabudovaným rozhraním `USB2` umožňující rychlý přenos dat i pro malé aplikace. Kromě mikrokontrolérů PIC firma vyrábí obvody pro RF přenos, řadiče pro spínání velkých proudů a napětí apod.

PIC16f877 je RISC CPU kontrolér s 35 instrukcemi. Většina instrukcí je jednocyklových. Tento typ mikrokontroléru obsahuje $8K \times 14$ bitů FLASH programové paměti, 368×8 bitů datové paměti.

Základní charakteristiky mikrokontroléru PIC jsou:

- dobře zvládnuté periferie, které umožňují bezpečný chod programu
- zpožděný začátek běhu programu do ustálení napájecího napětí
- zabudovaný Watch Dog timer (WDT)



Obrázek 2: Architektura PIC 16F877

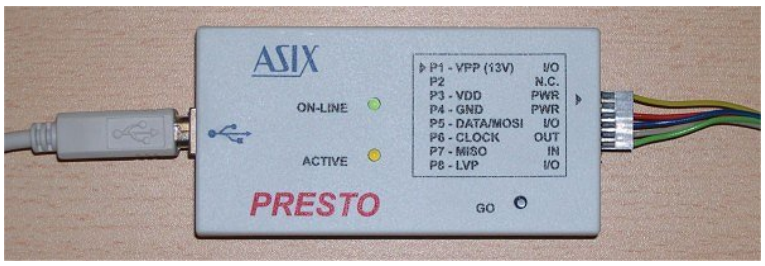
- možnost náhrady vnějšího oscilátoru oscilatorem vnitřním
- aplikačním programem přepisovatelná pamět programu
- široký rozsah napájecího napětí od 2V do 5,5V.
- hardwarově podporovaný sériový port s několika režimy provozu, I^2C, SPI sběrnici
- čítače, časovače, komparátory, PWM generátor, AD převodník,
- osmibitový vstupně výstupní paralelní port s externím řízením toku dat.

Jak je vidět z předcházejícího výčtu, PIC disponuje velkým množstvím rozhraní, které jsou potřebné pro vývoj jednoduché aplikace s přenosem dat.

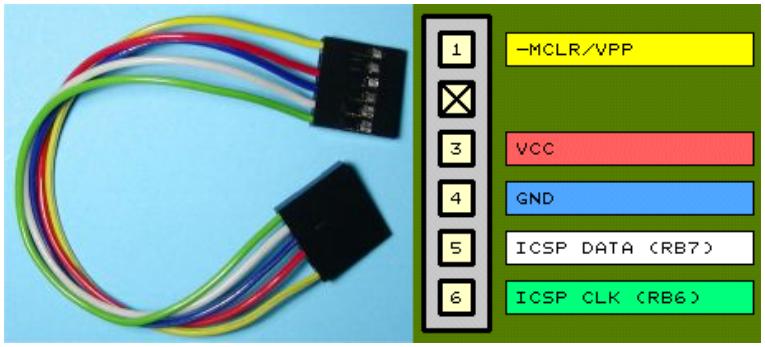
2.1.2 Vývojové prostředky MPLAB, Asix Up

Hlavním kladem tvorby aplikací s PIC jsou dobré vývojové prostředky. Při vývoji bylo použito prostředí MPLAB, které je k dispozici ke stažení zdarma na stránkách firmy Microchip viz.[2] spolu se standardním asemblerem pro PIC `MPASM`.

Při psaní programu v asembleru je doporučeno používání `MAKER`. Program je pak překládán dvakrát. V první fázi makra generují kód a v druhé se tento kód překládá do své `HEX` podoby. `MAKRA` lze použít pro generování tabulek a dlouhých skoků. Nezanedbatelnou výhodou `MAKER` je zlepšení přehlednosti zdrojového kódu. Seznam makropříkazů podporovaných `MPASM` najdete v helpu aplikace `MPLAB` pod záložkou `MPASM`.



Obrázek 3: Programátor presto



Obrázek 4: Programovací kabel ICSP

Nahrávání přeloženého programu do mikrokontroléru bylo prováděno za pomocí programátoru USB PRESTO od firmy ASIX s.r.o. Programátor PRESTO je ovládán z programu ASIX UP, který volně distribuuje firma ASIX. Programování mikrokontrolérů probíhá pomocí sběrnice ICSP. Programátor PRESTO umožňuje programování většiny mikrokontrolérů PIC, ATMEL AVR a sériových pamětí. Nevyžaduje externí napájení.

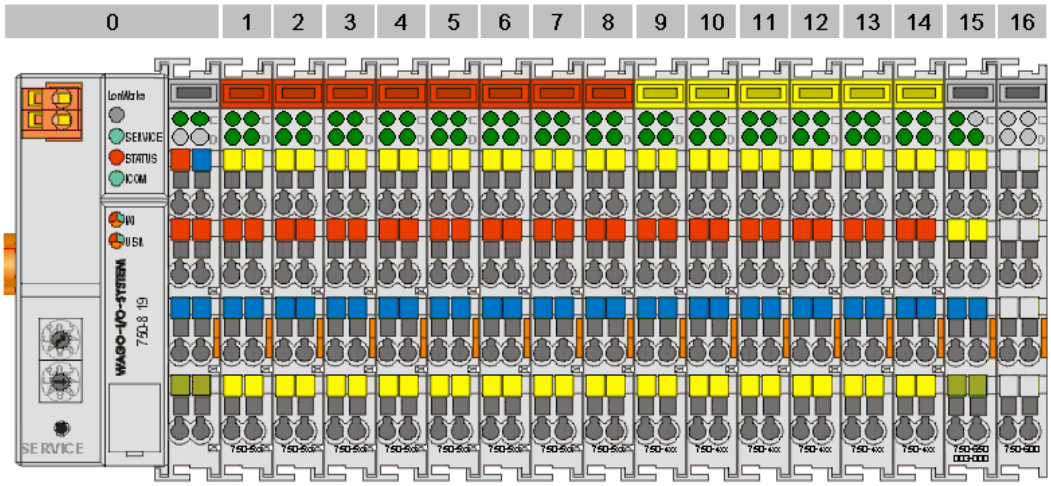
2.2 WAGO PLC

Vznik PLC souvisí s potřebou spolehlivého řízení v průmyslových podmínkách výroby. Pomocí PLC je realizováno programovatelné řízení sekvenčních procesů. Programování PLC je standardizováno normou IEC 61131. Podrobněji v kapitole 2.2.3.

2.2.1 WAGO-I/O-systém 750

Programovatelný automat WAGO-I/O-systém 750 je modulární systém, který kromě standardních modulů PLC podporuje většinu průmyslových i komerčních sítí jako jsou Profibus, CANOpen, LONWORKS, Ethernet a Firewire. PLC WAGO je řešeno pomocí modulů vhodných pro provoz v průmyslových podmínkách a je optimalizováno pro vytváření mnoha různých konfigurací digitálních a analogových vstupů a výstupů.

Protože výtah a jeho obsluha spadá do kategorie tzv. domovní automatizace (Home Automation), byl původně pro řízení výtahu vybrán modul se síťovým připojení LONWORKS. Vybraná sestava dále obsahuje moduly binárních vstupů a výstupů a modul sériového rozhraní RS232. Technická specifikace modulů je k dispozici ve firemní dokumentaci podle sériového čísla uvedeného v tabulce tab. 1.



Obrázek 5: WAGO-I/O-systém 750

Pozice	Funkce	popis	Číslo položky
0	Řadič	Programovatelný Lon řadič	750-819
1-8	Výstup	4-kanálový digitální výstupní modul DC 24V	750-504
9-14	Vstup	4-kanálový digitální vstupní modul DC 24V	750-402
15	RS232	Sériový interface RS232 C	750-650
16	End	Ukončovací modul	750-600

Tabulka 1: Složení WAGO-I/O-systém 750 pro aplikaci výtah.

2.2.2 Nástroje pro programování

Program pro WAGO byl vytvořen pomocí vývojového prostředí WAGO-I/O-I/O-PRO 32. Lze v něm programy psát, simulovat provádění bez přítomnosti PLC, nahrávat do PLC a tam krokovat. Programovací jazyky v tomto vývojovém prostředí se řídí podle normy IEC 61131-3. Výhody a nevýhody jednotlivých jazyků jsou shrnuty v následující kapitole.

2.2.3 Norma IEC 61131-3

1. Kontaktní schémata (Ladder Diagram - `LD`)

   Metoda LD je označení pro klasická liniová nebo také kontaktní schémata. Tato metoda byla všeobecně známá a používaná pro návrh releových ovládacích obvodů. Liniové schéma je ekvivalentní soustavě logických rovnic, přesněji řečeno sekvenci přiřazovacích příkazů přiřazujících logickým proměnným hodnoty definované logickým výrazem na pravé straně tohoto příkazu. U této jednoduché formy zápisu musí automat soustavně vyhodnocovat všechny příkazy. I ty, které v dané chvíli vyhodnocovány být nemusí. Tato potíž může být odstraněna, umožní-li se v sekvenci příkazů skoky, které však jsou plně v režii tvůrce algoritmu. Implementace této metody bývá různě modernizována, princip ovšem zůstává. Jedná se o metodu zastaralou.

2. Funkční bloky (Function Blocks - `FB`)

   Metoda funkčních bloků má stejný původ jako metoda LD, jen používá grafické vyjádření ovládacích schémat z integrovaných prvků realizujících základní logické funkce. Hodnocení významu je shodné jako u metody LD, jen jednotlivé bloky mohou pracovat s menším počtem proměnných.

3. Seznam příkazů (Instruction List - `IL`)

   Metoda IL je návratem do dávné minulosti. Byla užívána u nejprimitivnějších automatů. Z úsporných důvodů byla používána i v nedávné době u jednodušších systémů. Jedná se v podstatě o jazyk symbolických adres, tzv. asembler.

4. Strukturovaný text (Structured Text - `ST`)

   Metoda ST je první z metod, které vyhovují požadavkům doby. Jedná se o jednoduchý programovací jazyk typu autokódu. Podobných jazyků je dnes velmi mnoho. Patří sem Pascal, Visual Basic, všechny tzv. skriptové jazyky, kterými jsou vybaveny mnohé současné programové systémy a které slouží k uživatelskému přizpůsobování jejich funkcí. Příkladem je tvorba dokumentace v systémech třídy CASE. Jazyky si jsou navzájem velmi podobné a jsou velmi jednoduché. Lze se je snadno naučit a jejich používání nevyžaduje dlouhodobou soustavnou praxi. Strukturovaný text by neměl být chápan jako samostatný programovací prostředek. Jeho význam je nutné posuzovat v souvislosti s následující metodou SFC.

5. Sekvenční funkční graf (Sequential Function Chart - `SFC`)

   Metoda SFC je velmi podobná metodě GRAFCET. Metoda sama o sobě může být použita jen k hrubší analýze nebo k hrubému návrhu algoritmů. Funkční náplň jednotlivých kroků v tom případě reprezentuje popis v podobě poznámek přiřazených ke graficky znázorněným krokům. Plnohodnotná metoda vznikne až když je možné zapsat algoritmickou náplň jednotlivých kroků v nějakém programovacím jazyce. U metody GRAFCET to byl jazyk GPL (GRAFCET Programming Language). Jeho rovnocennou náhradou je strukturovaný text. Norma IEC 1131-3 předpokládá, že pro zápis algoritmů lze použít kterýkoli z ostatních jazyků. Případy, kdy k tomu bude rozumný důvod již byly uvedeny. Při zavádění metod programování PLC není důvod volit jinak než kombinaci SFC a ST, i když se vlastně jedná o metodu jednu.

2.3 Sběrnice LONWORKS

2.3.1 Základní data o sběrnici

Sběrnice LONWORKS (v dalším LON) je proprietární technologie realizující tzv. "Field Bus". Byla vyvinuta na počátku 90tých let firmou Echelon ve spolupráci s firmami Toshiba a Motorola.

Je přednostně určena na komunikaci většího množství uzlů při malém objemu dat (nejčastější typ zprávy přenášející 2 byty dat a má délku 12 bytů). Lze ji použít pro různá přenosová média s různou rychlostí přenosu od 4.8kb/s přes 1Mb/s.

Pro řízení provozu na sběrnici je použit upravený protokol CSMA/CA nevyžadující centrální řízení sběrnice a přitom umožňující stanovení priorit uzlů a zpráv.

Sběrnice LON je popsána pomocí klasické sedmivrstvé architektury ISO/OSI. Ta je z větší části implementována standardním programovým vybavením procesorů Neuron od firmy Echelon, které se používají pro realizaci uzlů sběrnice.

Popisů sběrnice a jejího nasazení lze najít v literatuře hodně. Následující text se zaměřuje na rozbor náročnosti realizace uzlu sběrnice.

2.3.2 Vrstvená architektura sběrnice LON

Protokol LON je standardizován jako EIA 709.1. Jednotlivé vrstvy ISO OSI referenčního modelu jsou: fyzická, datová, linková, síťová, transportní, relační, prezentační a aplikační.

Vrstvy 1 až 4 jsou vždy realizovány programovým vybavením Neuron procesoru. Vrstvy 5 až 7 mohou být realizovány Neuron procesorem v případě tzv. "procesor-based" uzlu nebo programovým vybavením jiného procesoru/počítače v případě "host based" uzlu (viz. dále).

2.3.3 Procesor Neuron

Ve skutečnosti se jedná o trojici procesorů v jednom pouzdře společně s pamětí a vstupně-výstupními obvody, které se dělí o obsluhu sběrnice LON a provádění uživatelského programu přeloženého z jazyka Neuron C. Jsou k dispozici dvě varianty (3120 a 3150), lišící se velikostí vnitřní paměti resp. možností adresace externí paměti. Oba typy mají pro komunikaci s periferiemi 11 obousměrných programově ovládaných bran.

2.3.4 Uzel sběrnice LON

Při realizaci zákaznického uzlu pro sběrnici LON je možno zvolit několik odlišných přístupů:

1. Použít periferii s rozhraním LON

   Nejjednodušší případ. Je třeba pouze připojit uzel ke sběrnici a nakonfigurovat ho.

2. Použít kartu s rozhraním LON do PC

   V tomto případě je součástí dodávky i ovladač případně další programové vybavení. Uzel je třeba nakonfigurovat a v případě tvorby aplikace se pro komunikaci se sběrnicí LON použije dodaný ovladač.

3. Použít profesionální převodník sběrnice LON na jinou sběrnici

   Existují hotové produkty od firmy Echelon i od dalších výrobců umožňující připojování na sběrnici LON pomocí sériového, paralelního, ethernet, usb a dalších rozhraní. Součástí dodávky je obvykle ovladač a další programové vybavení pro PC s operačními systémy od firmy Microsoft v různých verzích. Použití na PC s podporovanou verzí OS je v podstatě stejné jako v případě 2). Použití s jinými zařízeními vyžaduje realizovat na daném zařízení ovladač a komunikační program pro obsluhu sběrnice LON, což není zdaleka jednoduché.

4. Vytvoření vlastního uzlu metodou "Chip-hosted" (dále CH uzel)

   V případě CH uzlu je na Neuron procesoru naprogramován kód uživatelské aplikace. Toto řešení má několik významných omezení. Je to velikost kódu (Neuron procesor má adresní prostor 64kB), rychlost provádění (hodinový kmitočet procesoru je 10MHz) a hlavně počet vstupně/výstupních linek. Výhodou je relativní jednoduchost realizace CH uzlu. Zjednodušeně řečeno stačí napsat kód ovládání periferie v jazyce Neuron C a propojit periferii s obvodem Neuron. Hodí se pro připojování jednoduchých zařízení typu žárovka, vypínač a podobně.

5. Vytvoření vlastního uzlu metodou "host-based" (dále HB uzel)

   Tvorba HB uzlu je nejsložitější. Pro komunikaci mezi hostem a Neuronem byla definována varianta LON protokolu na paralelní resp. sériovém rozhraní. Ta je na straně Neuron procesoru realizována tzv. Microprocessor Interface Programem (proto se dále tento protokol označuje jako MIP protokol). Na straně Neuron chipu je situace jednoduchá. Program v Neuron C jen aktivuje MIP program. Na straně aplikačního procesoru je však třeba realizovat nejen vlastní aplikaci, ale i všechny vrstvy MIP (LON) protokolu. Vzorové řešení je k dispozici pro PC. Obsluha fyzické vrstvy je k dispozici v assembleru pro procesor MC68HC11.

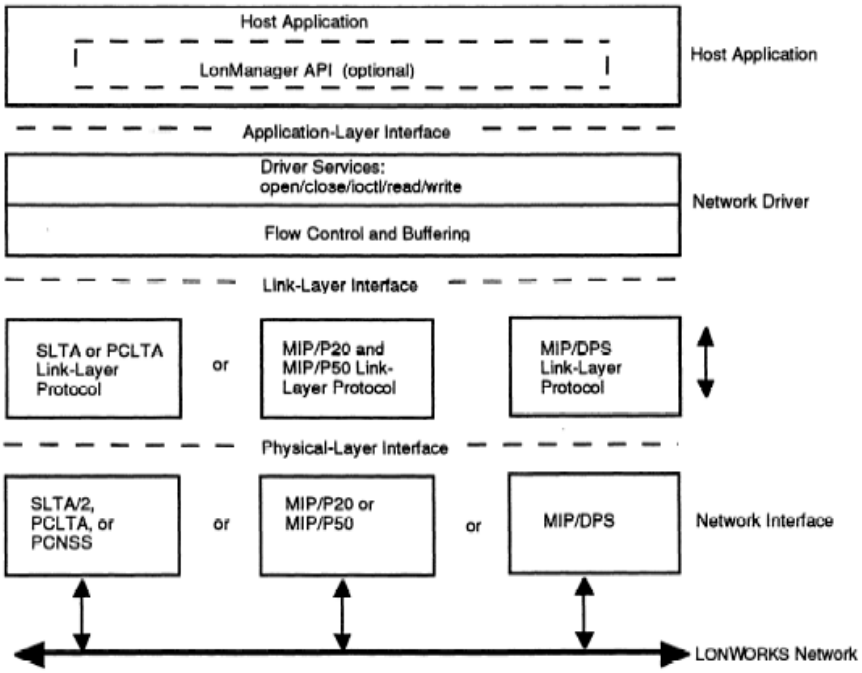
Pro připojení složitého zařízení s mnoha vstupy a výstupy je tedy třeba zvolit přístup 5) resp. 3). Většina dostupných prací se zabývá přístupy 1) až 4). Proto je dále podrobněji rozebrán přístup 5) - host-based uzel používající protokol MIP resp. SLT.

2.3.5 Host-based uzel

Host-based uzel je tvořen procesorem Neuron (dále NP) a aplikačním procesorem/počítačem (dále AP). Na NP běží speciální aplikační program dodaný firmou Echelon, který má dvě funkce. Deaktivuje horní tři vrstvy komunikačního protokolu (relační, prezentační a aplikační) a přenechá jejich funkce AP. Dále podle typu rozhraní mezi NP a AP realizuje spodních pět vrstev komunikačního protokolu mezi NP a AP.

Na AP je třeba instalovat resp. vytvořit ovladač realizující všechny vrstvy komunikačního protokolu kromě vrstvy aplikační. Aplikační vrstva je tvořena vlastní aplikací. Aplikace musí realizovat jistou minimální podmnožinu funkcí sítě LON (např. Change Mode to Online/Offline, Query SNVT), protože nejsou (na rozdíl od Chip-based uzlu) realizovány programem Neuron procesoru.

Stručná charakteristika jednotlivých síťových vrstev následuje.



Obrázek 6: Charakteristika jednotlivých síťových vrstev

1. Fyzická vrstva zajišťuje vlastní přenos dat.

   Je realizována paralelním rozhraním (na straně Neuronu programem MIP/P20 resp. MIP/P50), paralelním přístupem do dvoubránové paměti (program MIP/DPS) nebo sériovým rozhraním (SLTA/2).

   Paralelní rozhraní je typu "master/slave". Aplikační procesor je master a Neuron procesor je slave typu A pro paralelní propojení resp. slave typu B pro propojení prostřednictvím dvoubránové paměti.

   Fyzická vrstva sériového rozhraní je definována EIA RS-232 standardem. Přenos je asynchronní osmibitový, bez parity s jedním stop bitem. Rychlost přenosu je volitelná.

2. Linková vrstva organizuje přístup k mediu.

   Pro paralelní rozhraní je přístup řízen předáváním řídícího slova "token". Po resetu je token vždy na straně master. Vlastník řídícího slova token může zapsat blok dat a tím předat

token implicitně nebo předat token explicitně, nemá-li data k předání. Master může navíc žádat o synchronizaci, na kterou slave odpoví potvrzením.

Pro sériové rozhraní je použit Alert/Ack protokol resp. protokol s vyrovnávací pamětí (buffered protocol). V Alert/Ack protokolu všechny přenosy začínají vysláním bytu ALERT (0x01) na který druhá strana odpoví bytem ALERT ACK (0xFE). Po obdržení ALERT ACK vysílání pokračuje bytem délky datové části paketu a jejím dvojkovým doplňkem. Délka je minimálně 1 (příkaz + délka případných dat). Délku a její dvojkový doplněk přijímač sečte a pokud nedostane 0xFF je paket zahozen. Navíc přijímač ověřuje kontrolní součet vysílaný na závěr.

Data musí přicházet maximálně s prodlevou 100ms (1s při připojení modemem). Implementace SLTA neumožňuje plný "full duplex". Probíhá-li vysílání směrem `"NP(SLTA)->AP(aplikace)"`, je vysílání opačným směrem delší než 16bytů ztraceno.

Je-li na straně AP příjem realizován pomocí přerušení a nedochází ke ztrátě dat, lze použít tzv. "buffered" protokol. V tomto případě příjemce nepotvrzuje přijetí ACK. Formát vysílaných dat je stejný jako v předchozím případě. Podrobnější popis je v `"LTS-20 User's Guide kapitola 6"` viz. [12].

3. Síťová vrstva je prázdná, protože jak paralelní tak sériové rozhraní je "point to point" a proto nepotřebuje adresaci.

4. Transportní vrstva zajišťuje spolehlivý přenos dat. Paralelní a lokálně připojené sériové rozhraní je pokládáno za spolehlivé je pro ně tato vrstva prázdná.

   Pro vzdálené sériové rozhraní je možno zvolit ACK/NACK protokol. Do zprávy je přidáno číslo rámce a přijímač potvrzuje přijetí resp. odmítnutí rámce pomocí ACK resp. NACK.

5. Vrstva relační řídí tok dat tak, aby nedocházelo k jejich ztrátě v důsledku zahlcení protistrany. V této vrstvě jsou definovány příkazy pro řízení LON rozhraní směrem "dolů" (aplikace-LON) a "nahoru" (LON-aplikace). Seznam příkazů je v literatuře [13, Appendix D].

   Příkazy jsou rozděleny do dvou skupin.

   - Příkazy přímo prováděné na NP (programem MIP resp. SLT). Sem patří příkazy niRESET, niFLUSH, `niFLUSH_IGN`, `niFLUSH_CANCEL`, niONLINE, niOFFLINE a niSLEEP. V případě sériového rozhraní rovněž příkazy niPUPXOFF, niPUPXON a niSSTATUS.
   - Příkazy vyžadující vyrovnávací paměť v NP. Jsou to příkazy pro řízení LON sběrnice (niNETMGMT) a příkazy pro přenos dat (niCOMM). Struktura dat odpovídá struktuře aplikační zprávy popsané např. v dokumentaci [13, Appendix C]. Zprávy jsou čtyř typů a jsou popsány v příloze dokumentace [13, Appendix D].

   Tyto příkazy nemohou být rovnou vyslány "dolů" ale vyžadují předběžnou alokaci vyrovnávací paměti typu odpovídajícímu typu zprávy. Alokace se provádí vysláním tohoto příkazu se specifikací typu zprávy, ale bez dat. Po přijetí odpovědi niACK je příkaz zopakován tentokrát s daty. Není-li vyrovnávací paměť k dispozici dojde u sériové verze k odmítnutí požadavku ze strany NP příkazem niNACK . Ovladač musí požadavek zopakovat. U paralelní verze se v tomto případě odloží potvrzení niACK a ovladač musí pozdržet zápis zprávy. Situace je popsána na straně 6-11 v dokumentaci [12] pro sériové a v dokumentaci [16] na straně 5-6 pro paralelní rozhraní.

   Pro směr "nahoru" (k aplikaci) se u paralelního rozhraní předpokládá, že je vždy vyrovnávací paměť k dispozici. U sériového rozhraní může být použito řízení příkazy niPUPXOF a niPUPXON.

6. Vrstva prezentační předává vlastní příkazy a pakety sítě LON aplikaci.

   Prezentační vrstva definuje zprávy předávané mezi aplikací a ovladačem. Jedná se o hodnoty síťových proměnných (tzv. SNVT), explicitní zprávy nebo lokální příkazy síťovému

rozhraní. Explicitní zprávy mohou být aplikační zprávy, zprávy řízení sítě nebo síťové diagnostické zprávy definované v literatuře [14, Appendix B]. Struktura zpráv je definována v literatuře [13, Appendix C] jako hlavičkový soubor v `NI_MSG.H ANSI C`.

7. Aplikační vrstva

   Kromě vlastní aplikace musí tato vrstva implementovat ty služby správy sítě, které nejsou v tomto případě obslouženy NP. Patří sem například:

   - reakce na reset NV. Reset je signalizován přijetím příkazu niRESET. Po něm je NP v tzv. FLUSH stavu, ve kterém ignoruje komunikaci na LON sběrnici. Aplikace muže poslat do NP konfigurační příkazy a pak ukončit FLUSH stav příkazem `niFLUSH_CANCEL`.
   - přepnutí uzlu online/offline
   - zjištění/nastavení konfigurace síťových proměnných (je-li povoleno tzv. Host selection a konfigurace není udržována v NP)
   - dotaz na fyzickou adresu uzlu (wing)
   - dotaz na hodnotu síťové proměnné (query SNVT)

2.3.6 Realizovatelnost HB uzlu

Z uvedeného by mělo vyplynout, že je velmi komplikované, ne-li nemožné vytvořit samostatný uzel sítě LON na aplikačním procesoru pokud procesorem není PC s podporovanou verzí operačního systému. Jedná se o zhruba stejnou náročnost jako naprogramování vlastní implementace TCP/IP stacku. Problém je navíc komplikován nízkou kvalitou dostupné literatury, ve které se mnohokrát opakuje téměř to samé, ale některé podstatné informace chybí. Problematické by bylo rovněž ladění a navíc vlastní implementace uzlu by mohla mít v případě chyby neblahý vliv na celou síť.

Tuto situaci pochopila i firma Echelon a vyvinula zjednodušenou verzi sériového rozhraní implementovaného Neuron programem ShortStack. Hlavní výhodou této zjednodušené verze je to, že obsluha sběrnice je provedená na straně NP a AP využívá jen síťové proměnné. Pokud by se měl implementovat HostBase uzel, připadá v úvahu pouze pomocí ShortStack programu resp. vývojového prostředí. I tak by to byla činnost poměrně náročná.

Proto bylo nakonec upuštěno od řízení modelu sběrnicí LON. Zbývá zde možnost monitorovat chování výtahu pomocí LON sběrnice PCL WAGO z programu běžícího na PC připojeném standardním způsobem na sběrnici LON.

2.4 Rozbor problematiky optimalizace skupinového řízení výtahů

Výtahovým systémem (dále VS) se rozumí skupina dvou nebo více výtahů pracujících pod společným řízením a se společným rozhraním vůči pasažérům. Řídící systém pracuje s plánovacím algoritmem (dále PAVS) a ovládá výtahy příkazy na vyšší úrovni typu: "vytah číslo 1 dojede do 3. patra a otevře dveře pro pasažéry směrem dolů". Jednotlivé výtahy musí mít ještě samostatné řízení pojezdu, zrychlování, zpomalování, otvírání dveří apod.

Problém optimalizace PAVS je velmi složitý, přestože je jednoduše definován: Pasažér přijde v náhodném patře k výtahu a chce být co nejdříve obsloužen. Toto kritérium se nazývá "průměrný čekací čas" (average waiting time, AWT). Někdy se také používá kritérium "průměrný čas obsluhy" (average system time, AST) definované časem od příchodu pasažéra k jeho dojetí směrem nahoru resp. dolů do cílového patra. Jako druhotné kritérium může být vyžadována minimalizace pojezdů (úspora energie).

Minimalizace těchto kritérií je velmi náročné z několika důvodů:

- Stavový prostor systému je velký. Je třeba brát v úvahu pozici, rychlost, směr pohybu a obsazenost kabin. Dále počet pasažérů čekajících v patře.
- Systém má velké množství neurčitosti:
  - Pasažéři přicházejí náhodně a jezdí do náhodných pater. Přitom se nejedná o skutečně náhodné procesy. Můžeme mluvit o špičce směrem nahoru (nástup v přízemí, odjezd do náhodných pater), špičce směrem dolů (nástup v libovolném patře, výstup v přízemí) případně evakuaci (cílem je dopravit co nejvíce lidí). Navíc pasažéři obvykle nepřicházejí jednotlivě, ale ve skupinách.
  - Nelze dopředu zjistit cílové patro ani počet čekajících pasažérů. Vnější tlačítka Nahoru a Dolů dávají jen informaci o požadavku, ne o jejich počtu. Výsledkem může být, že nějaké rozhodnutí se později ukáže jako špatné. Pokud se povolí změna rozhodnutí (tzv. reassignment/přeřazovací politika, obvyklé v západních zemích; opakem je immediate/okamžitá politika, obvyklá v Japonsku), může počet těchto změn rychle růst (jedna změna ovlivní systém a vyvolá jinou změnu).

Na druhé straně úroveň optimalizace se v reálné situaci obtížně měří a jistá míra neoptimálnosti není důvodem k reklamaci výtahu.

V přístupu k optimalizaci lze vysledovat dva přístupy. Akademický a praktický. O akademickém přístupu se pořádají konference a lze o něm najít poměrně hodně článků. Praktický přístup implementovaný ve skutečných systémech je součástí firemního tajemství.

O složitosti problému svědčí údaje z lit.[18] o minimální konfiguraci řídícího systému:

- řídící systém každého výtahu by měl být složen alespoň ze čtyř nezávislých mikroprocesorů, z toho jeden výkonný RISCový s 2MB RAM, 2MB EPROM a multitáskovým systémem ...

Případně další teoretické rozbory jsou uvedeny v lit. [17, 18].

Optimalizace v tomto případě není hlavní částí diplomové práce a je implementována na PLC, které současně musí řídit pojezdy kabin. Proto byl pro optimalizaci navržen zjednodušený algoritmus opírající se více o úvahu než o teoretický rozbor.

3 Rozbor možných řešení modelu výtahu

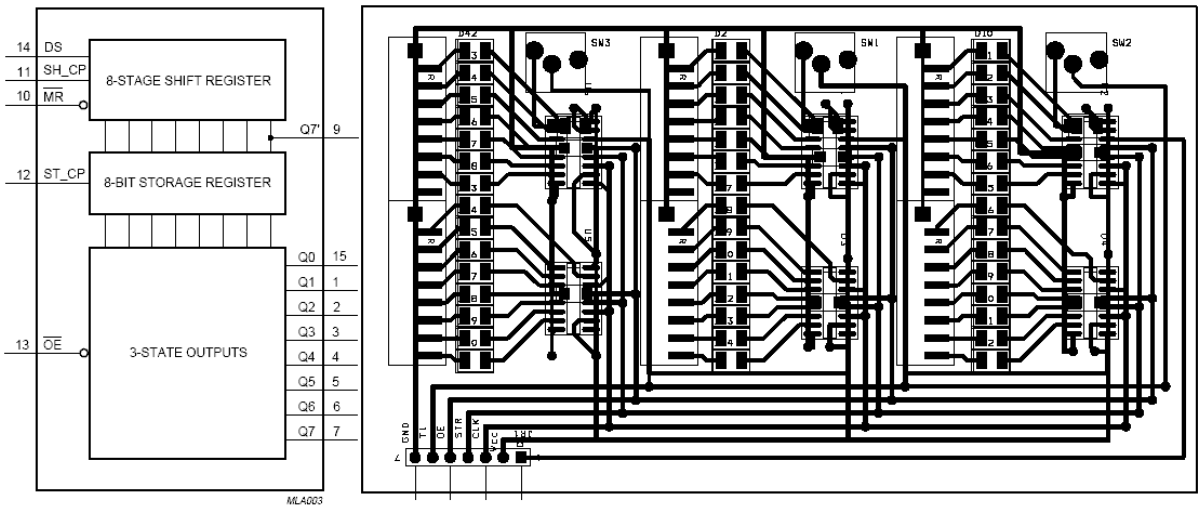
V této kapitole je uvedeno několik variant řešení zobrazovacího modulu výtahu a rozhraní k nadřazeným prvkům (PLC). Nejde o podrobný popis jednotlivých návrhů, ale o rozbor možných přístupů k řešení problému ovládání velkého počtu digitálních prvků.

3.1 Řešení zobrazovacího modulu

Zobrazovací modul(dále ZM), patří k největším částem celého modelu výtahu. Skládá se ze sloupcových LED diod "bargrafů", které simulují polohu kabiny výtahu. ZM obsahuje tlačítka pro zadávání požadavků cestujících a ostatní informační prvky dle zadaní.

3.1.1 Řešení ZM pomocí posuvných registrů 74HC595SMD

První varianta zobrazovacího modulu byla řešena pomocí LATCH obvodů 74HC595 SMD.



Obrázek 7: Latch 74HC595 a deska vzorku výtahu SMD

Zobrazovací modul (ZM) byl navržen technologií SMD. Jako zobrazovací prvky byly použity SMD LED diody velikosti 1206. Každá led dioda byla doplněna sériovým odporem. Pro praktické otestování byl vytvořen pokusný model o 48 diodách. Tento model se podařilo zprovoznit. Jeho funkčnost se zdála uspokojivá i pro složitější aplikaci (víc diod).

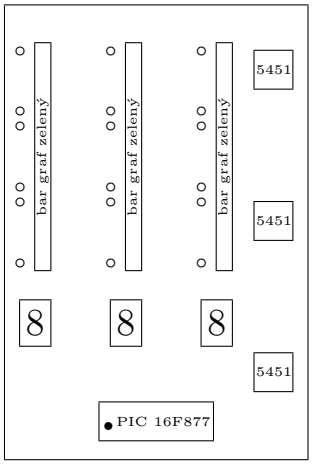
Jelikož trh dnes nabízí více typů specializovaných budičů LED diod, byl tento návrh řešení zamítnut a realizace zobrazovacího modulu se upnula na použití budičů LED M5451Q.

3.1.2 Nerealizovaný zobrazovací modul s multiplexovaným zobrazováním

Multiplexovaný zobrazovací modul (dále MZM) modeluje pohyby tří výtahů. Na MZM v jednom okamžiku svítí LED diody pouze u jediného výtahu. To je zajištěno pomocí spínacích prvků, které jsou propojeny se společnou anodou LED diod. Katody diod jsou připojeny na výstupy budičů M5451Q.

Budiče M5451Q pracují jako posuvné registry. Mají 1 vstup a 35 paralelních výstupů. Multiplex 1:3 znamená úsporu šesti budičů M5451Q. Posuvné registry na MZM řeší problém nedostatku

výstupních portů zvoleného mikroprocesoru. Změnou střídy svícení diod lze řídit jas.



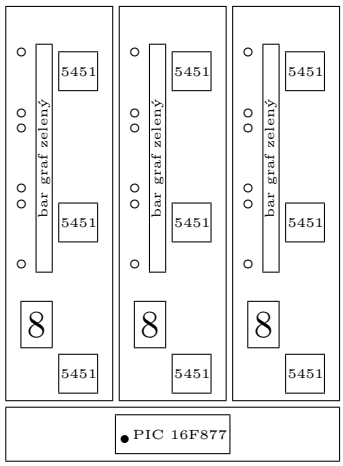
Obrázek 8: Multiplexovaný zobrazovací modul

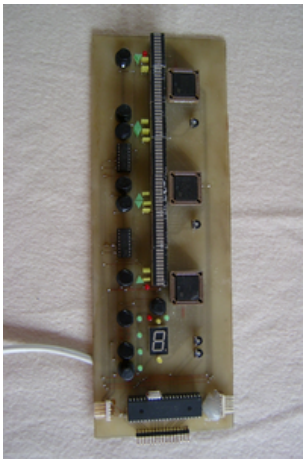
Byl proveden odhad zatížení procesoru PIC multiplexem. Při předpokládaném počtu instrukcí, které se provedou při jednom cyklu rozsvícení resp. zhasnutí všech diod pro všechny tři výtahy, při obnovovací frekvenci $50Hz$ a době provádění jedné instrukce $1\mu s$. Z odhadu vyšlo zatížení procesoru 15%.

Multiplexovaná varianta má jednu velikou nevýhodu. Nelze oživovat jednotlivé výtahy, ale je nutné desku o rozměrech $30 * 30cm$ vyrobit a zprovoznit v celku. Dalším důvodem proč MZM nebyla realizována, byla obava z přepínání proudů cca. 1A při multiplexování a tím vznikajícího rušení.

3.1.3 Nemultiplexovaný zobrazovací modul

Na nemultiplexovaném zobrazovacím modulu, je pomocí obvodů M5451Q ovládána každá LED dioda zvlášť. Obvody M5451Q mají datový vstup, který je synchronizován hodinovým vstupem. Pro ovládání 315 LED je zapotřebí devíti obvodů M5451Q. Mikrokontrolér PIC tak ovládá 9 datových a jeden společný hodinový signál. Pro zjednodušení výroby lze celou NZM rozdělit na





Obrázek 9: Celková sestava výtahu a vzorek nemultiplexovaného zobrazovacího modulu

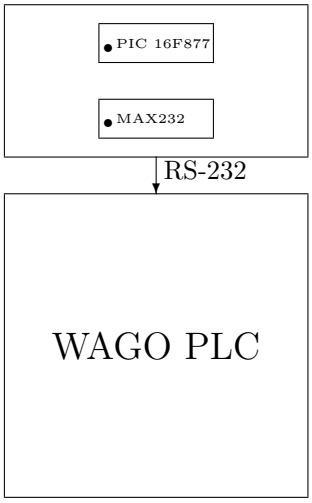
tři identické NZM pro každý výtah zvlášť s jedním řídícím modulem. Řídící modul obsahuje mikrokontrolér PIC a ostatní společné prvky. K přepsání dat všech budičů LED je zapotřebí cca 1000 instrukcí mikrokontroléru neboli $1ms$. Pro ověření funkčnosti NZM byl sestrojen model skládající se pouze z jedné zobrazovacího modulu. Na tomto modelu byl laděn program pro PIC a vyzkoušena funkčnost navrhnutého řešení.

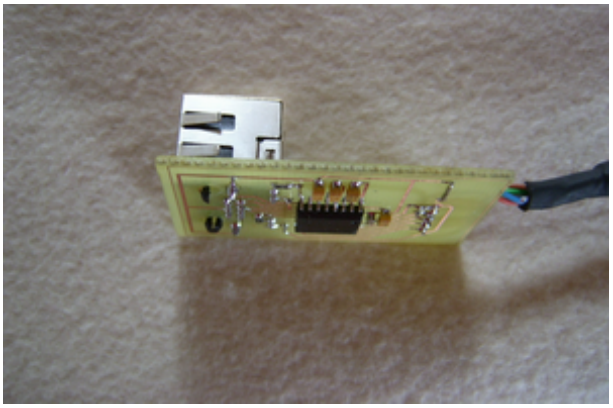
3.2 Řešení rozhraní k nadřazeným systémům

Nadřazeným systémem je v tomto případě myšleno PLC WAGO. Rozhraní předává informace na binární úrovni o stavu jednotlivých čidel výtahu a přijímá řídící signály z nadřazeného systému. Informace předávané na rozhraní jsou popsány v kapitole kapitola 4.3.

3.2.1 Rozhraní sériové sběrnice RS232

Propojení řídící desky s WAGO PLC pomocí sériové sběrnice RS232 umožňuje zařízení propojit pomocí několika vodičů. To je největší výhoda tohoto řešení. Dále tato varianta umožnila ladit řídící desku po připojení k PC. Pomocí počítače s jednoduchým diagnostickým programem tak lze sledovat provoz sběrnice. Program pro PC je popsán v kapitole 4.4. Testovací model výtahu neobsahoval převodník TTL-RS232. Tento problém byl vyřešen pomocí dodatečně navrhnutého převodníku. Deska převodníku je osazena konektorem "RJ45", pomocí něhož se propojuje s PC. Ve finální verzi se tento konektor již nevyskytuje a byl nahrazen standardním konektorem pro RS232 typu CAN9.





Obrázek 10: Rozhraní RS232 a testovací vzorek převodníku TTL/RS232

3.2.2 Návrhy řešení paralelní sběrnice

WAGO PLC disponuje 32 digitálními vstupy a 32 digitálními výstupy. Dále má zabudovaný řadič průmyslové sběrnice LONWORKS a řadič RS232. Zobrazovací modul potřebuje pro ovládání všech 3 výtahů 51 výstupů a 57 vstupů. To je více než je počet vstupů a výstupů u WAGO PLC. Proto byl použit multiplex ovládání výtahů na paralelní sběrnici.

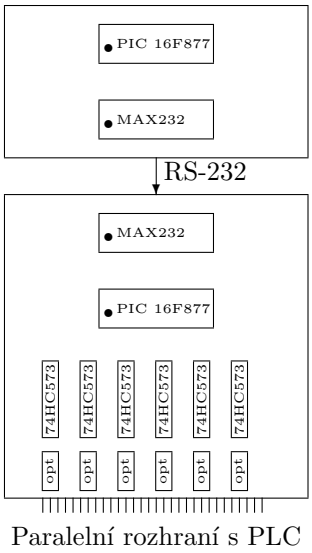
Výtah který je pomocí paralelní sběrnice ovládán je určen dvoubitovou adresou na paralelní sběrnici. Sběrnice má pak 32 vstupů a 32 výstupů stejně jako má WAGO PLC.

I když pomocí multiplexování byl snížen počet vodičů na 1/3, nelze takové množství vstupů a výstupů připojit přímo na mikrokontrolér PIC. Proto byla paralelní sběrnice navržena ve třech variantách.

1. První uvažovaná varianta byla distribuce ovládání zobrazovacího modulu a paralelní sběrnice mezi dva mikrokontroléry PIC.
2. Druhé uvažované řešení bylo zvětšení počtu vstupů a výstupů mikrokontroléru PIC pomocí I^2C sběrnice a expandéru MCP23016.
3. Realizovaným řešením se stalo zvětšení počtu vstupů a výstupů pomocí záchytných obvodů LATCH .

Dva mikrokontroléry PIC propojené prostřednictvím RS232

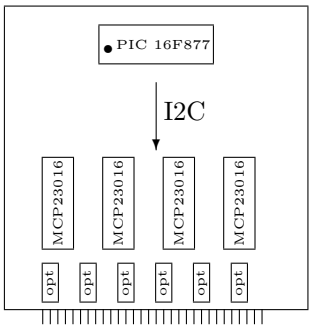
Tato varianta rozděluje problém mezi dva procesory. Jeden by řídil zobrazovací část výtahu, a druhý by řídil provoz na paralelní sběrnici. Po konzultaci s vedoucím DP nebyla tato varianta realizována pro její složitost.



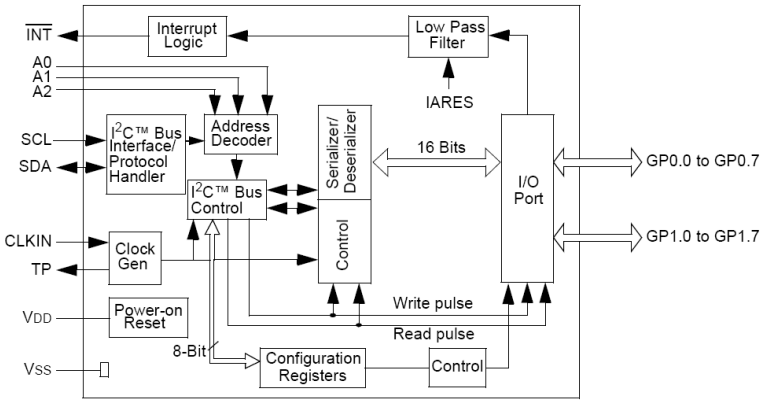
Obrázek 11: Dva mikrokontroléry PIC propojené prostřednictvím RS232

Zvětšení počtu vstupů a výstupů mikrokontroléru PIC pomocí I^2C sběrnice a expandéru MCP23016

Pro toto řešení byl uvažován obvod MCP23016, který má šestnáct vstupně výstupních portů a sběrnici I^2C s rychlostí až 400kHz. PIC 16f877 disponuje periferií I^2C master mode, která umožňuje oboustrannou komunikaci s tímto obvodem. Toto řešení zvoleno nebylo.



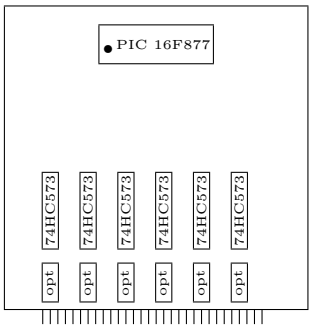
Paralelní rozhraní s PLC



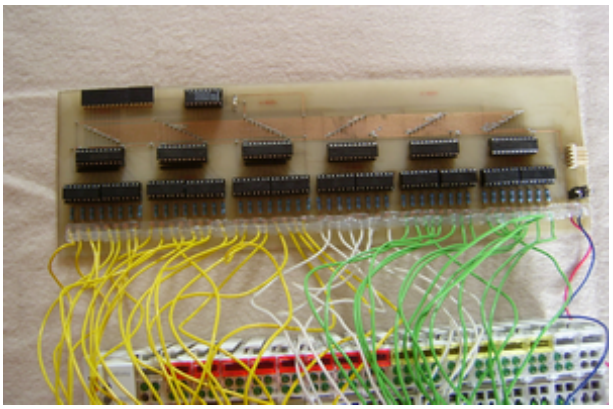
Obrázek 12: Řešení rozhraní pomocí I2C nebo pomocí Latch obvodu 74HC573

Minimalizací vstupů a výstupů pomocí záchytných (Latch) obvodů

Mikrokontrolér PIC disponuje 8 bitovou vstupně výstupní sběrnicí. To byl základ myšlenky multiplexovanou 2x24 bitovou paralelní sběrnici dále multiplexovat na tyto datové vstupy a výstupy. K tomuto účelu byly použity LATCH obvody 74HC573. Tyto obvody zachytí data na paralelní rozhraní v jeden okamžik a následně se postupně připojují na vnitřní 8 bitovou sběrnici a předávají data mikrokontroléru PIC. Polovina latch obvodů je zapojena jako vstupy a druhá polovina jako



Paralelní rozhraní s PLC

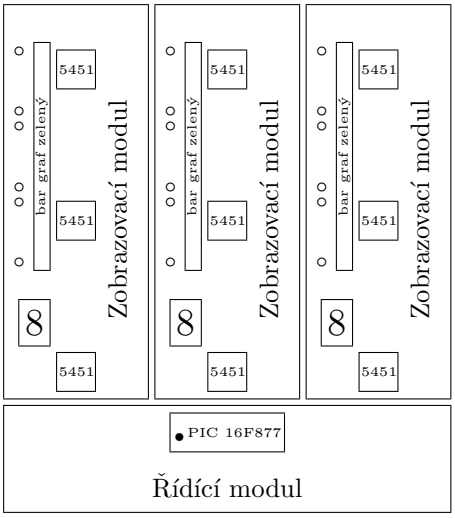


Obrázek 13: Pokusné paralelní rozhraní s LATCH a optočleny

výstupy. Multiplexovaná paralelní sběrnice je od PLC oddělena pomocí optočlenů. Toto řešení bylo realizováno a je podrobněji popsáno v kapitole 4.2. Pro otestování funkčnosti tohoto řešení byl vytvořen vzorek paralelní sběrnice, na kterém se otestoval program pro PIC a zároveň propojení s WAGO PLC.

4 Realizace

Zadání diplomové práce určovalo charakter výsledného modelu. Na doporučení vedoucího DP byla zvolena klasická technologie součástek DIP. Umístění integrovaných obvodů do patic, umožňuje opravovat závady vzniklé během provozu výtahu. Model se skládá ze čtyř modulů `10x30 cm`. Po jejich propojení dostáváme model o rozměrech `30x40 cm`. Mechanické upevnění zajištuje podkladová deska `320x420x8 mm`. Sestavený model je vyfotografován v příloze 6.11.



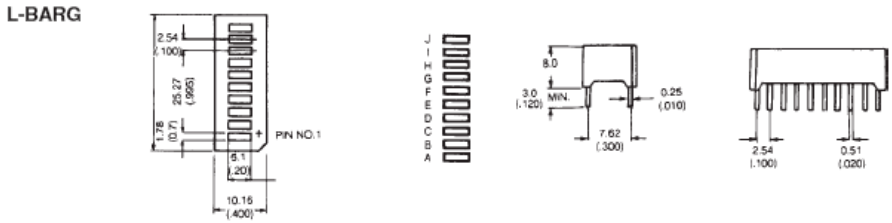
Obrázek 14: Blokové schéma modelu výtah

Zobrazování polohy kabin je umístěno na třech identických zobrazovacích modulech (dále ZM). Mikrokontrolér PIC, paralelní rozhraní k WAGO PLC, stabilizátory a sériové rozhraní RS232 jsou umístěny na řídícím modulu (dále ŘM). V následující kapitole je popsána realizace modulů ZM a ŘM.

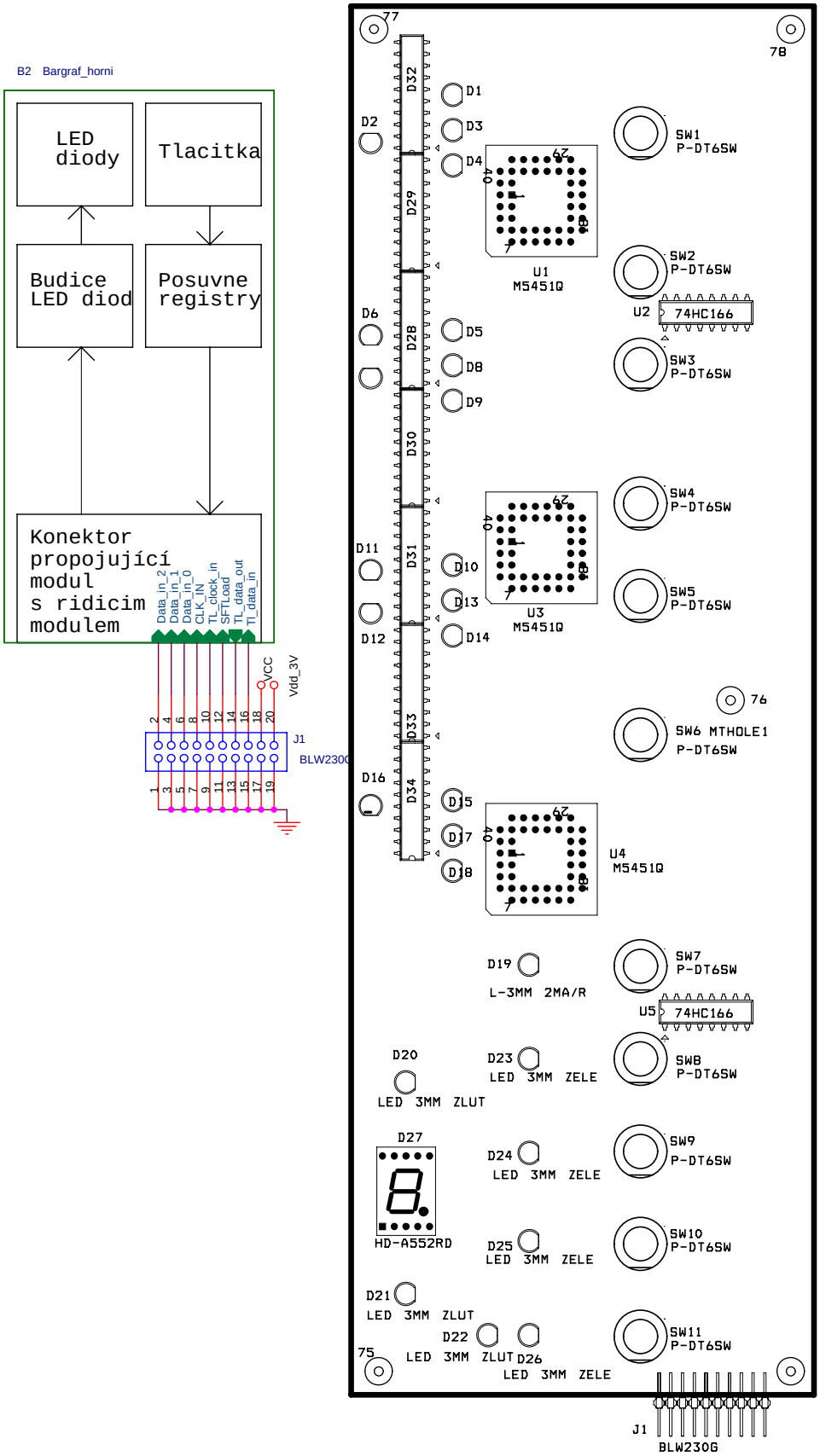
4.1 Zobrazovací modul

Zobrazovací modul je rozdělen na dvě části. Na model výtahové šachty a na ovládací panel kabiny výtahu. Ovládací panel je v jiném měřítku než model výtahové šachty a je umístěn pod ní.

Model výtahové šachty se skládá ze sedmi sloupcových LED diod "bargrafů" (D28-D34) zelené barvy o deseti diodách. Jejich celková délka činí 17,5 cm. Kabina je simulována pěticí svítících LED diod, které se posouvají s krokem 2,25 mm. Při rychlosti posunu $3,52 cm/s$ je frekvence změny LED diod $14 Hz$. Tato rychlost posuvu působí pro lidské oko již plynulým dojmem.



Obrázek 15: Bargraf



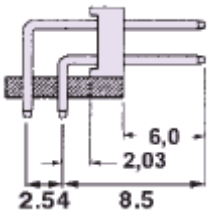
Obrázek 16: Blokové schéma a osazovací plán zobrazovacího modulu

V okolí šachty výtahu jsou umístěny informační LED diody. Zelené diody trojuhelníkového tvaru (např.D11,D12) vedle šachty označují směr vnějšího požadavku cestujícího (tvar diod obr.16 neodpovídá skutečnosti).

Žluté diody obdélníkového tvaru (např.D5,D8,D9), umístěné vpravo od výtahové šachty, zobrazují stav kontaktů signalizujících polohu kabiny. Diody D4,D9 a D14 označují místo `zpomalení/zrychlení` pod patrem, D5,D10 a D15 nad patrem a diody D3,D8, D13 a D17 zastavení v patře. Červené diody D1 a D18 signalizují havarijní kontakty. Rozsvěcovaní diod je ovládáno pohybem kabiny výtahu a tato informace je také k dispozici řídícímu systému.

Vedle šachty výtahu je umístěno šest tlačítek přivolávajících výtah.`(SW1-SW6)` Tato tlačítka jsou rozdělena tak, že v každém patře kromě přízemí a posledního patra jsou umístěna dvě tlačítka určující požadovaný směr jízdy.

Ovládací panel v kabině výtahu je složen také z LED diod a tlačítek. Kromě toho je zde umístěn sedmisegmentový zobrazovač čísla patra (D27). Ten zobrazuje čtyři symboly (`P,1,2,3`). Ovládací panel obsahuje čtyři tlačítka (SW7-SW11) vztahující se k jednotlivým patrům a čtyři diody zelené barvy k nim (D23-D26). Poslední tlačítko je STOP (SW7). Důležitá žlutá dioda označuje otevřené dveře(D22). Červená diodu přetížení kabiny výtahu (D19). Poslední dvě diody (D20,D21) označují momentální směr pohybu kabiny výtahu a jako jediné nejsou ovládány z PLC.



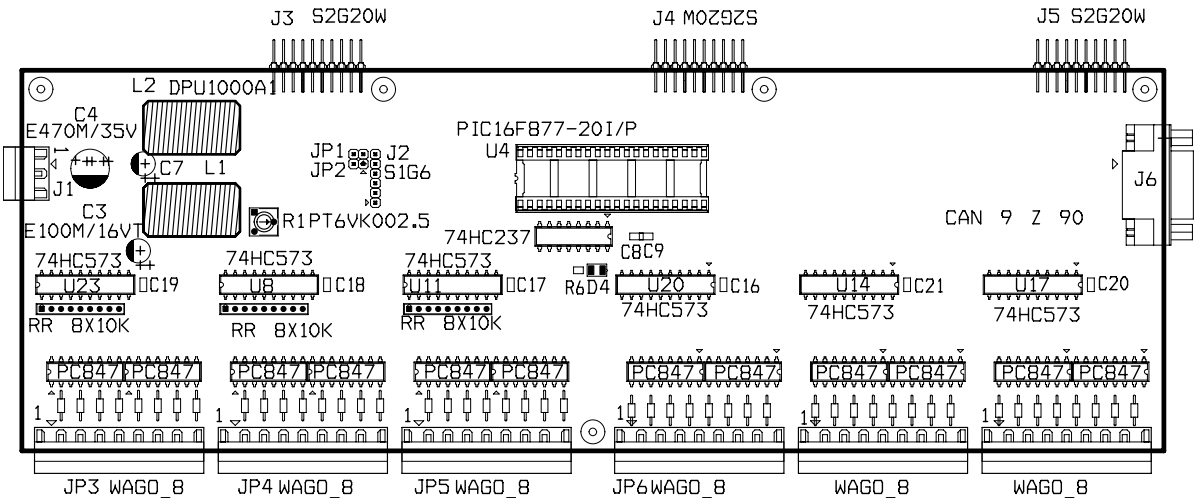
Obrázek 17: Konektor S2G20W

HW ZM je realizován pomocí tří posuvných budičů LED (U1,U3,U4). Každý budič může ovládat až 32 diod. Anody diod jsou společné a napájené regulovatelným napětím 3-5V (Schéma 6.3). Změnou tohoto napětí se dá regulovat jas diod. Načtení dat do budičů se provádí pomocí společného hodinového signálu a oddělených datových signálů (Schéma 6.2). Ke změně stavu LED Diod dochází po načtení posledního bitu dat.

Tlačítka jsou snímána pomocí šesti posuvných registrů 74HCT177. Na signál LATCH je sejmut stav tlačítek. Potom je pomocí signálu CLK a `TL_D` nahrán do PIC. Snímání tlačítek se provádí cyklicky s frekvencí cca 1 kHz.

Číslo pinu na ZM	Název pinu	Popis funkce
2	`Data_in_2`	datový signál pro m5451 spodní
4	`Data_in_1`	datový signál pro m5451 prostřední
6	`Data_in_0`	datový signál pro m5451 horní
8	`CLK_IN`	hodinový signál pro m5451
10	`Tl_clk_in`	hodinový signál pro 74HC166
12	`SFTLoad`	Latch tlačítek pro 74HC166
14	`TL_data_out`	Výstup `QH` sériové data od 74HC166
16	`TL_data_in`	Vstup `SERIN` sériové data pro 74HC166
18	`Vcc`	TTL 5V napájení logické části.
20	`Vdd_3V`	Napětí 3-5V napájení a regulace jasu LED.
Ostatní	`GND`	Společná zem.

Tabulka 2: Zapojení konektoru J1 na zobrazovacím modulu



Obrázek 18: Řídící modul

Všechny datové signály na zobrazovacím modulu jsou přivedeny na propojovací konektor obr. 17. Jeho zapojení naleznete v tab. 2.

4.2 Řídící modul

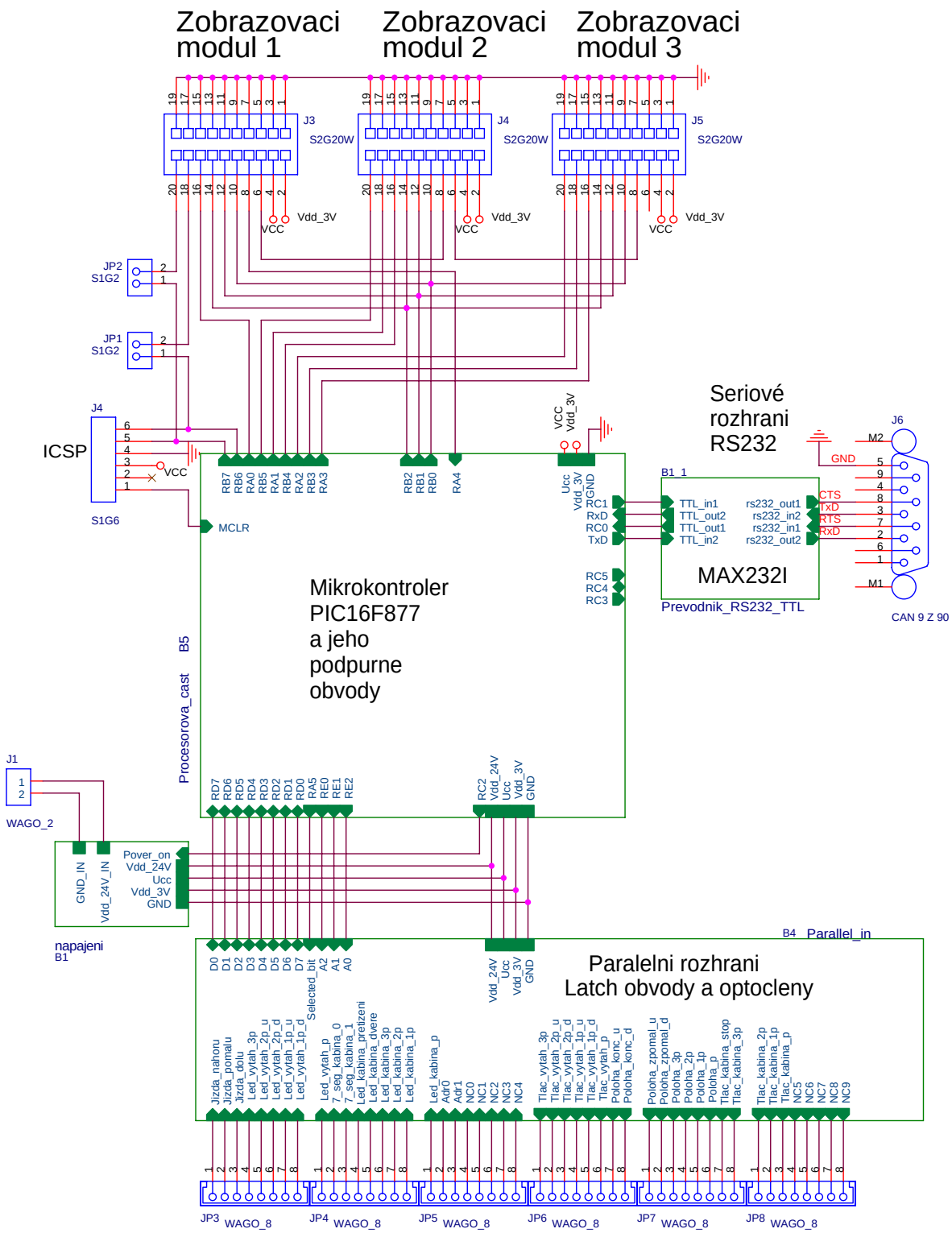
Řídící modul (dále jen ŘM) obsahuje prvky zajišťující inteligenci modelu. Jádrem ŘM je 8 bitový mikrokontrolér `PIC16F877`, jehož hlavním úkolem je ovládání zobrazovacích modulů (Schéma 6.5). Dále tento mikrokontrolér komunikuje s PLC WAGO pomocí paralelního nebo sériového rozhraní.

Na blokovém schématu, obr. 19, je znázorněna struktura řídicího modulu. Schémata jednotlivých subbloků jsou uvedeny v příloze.

Modul je napájen stejnosměrným napětím `24V` při maximálním odběru `500mA`. Napájení je chráněno proti přepólování. U napájecího konektoru (J1) jsou dva spínané stabilizátory. První generuje `+5V` pro TTL logiku a druhý `3-5V` pro LED diody. Napětí tohoto stabilizátoru se dá regulovat pomocí poteciometru (R1). Schéma zapojení stabilizátorů je uvedeno v příloze 6.10.

Paralelní rozhraní mezi ŘM a WAGO PLC je realizováno pomocí optočlenů. Výstupní úrovně logických signálů jsou 0V a 24V. Paralelní rozhraní je tvořeno 24 vstupními a 24 výstupními signály (jejich význam je uveden v další kapitole). Zapojení vstupních resp. výstupních optočlenů je uvedeno na schématu 6.7 resp. 6.8. Buzení optočlenů zajišťuje 6 LATCH obvodů `74HC573`, které jsou napojeny na 8 bitovou vnitřní sběrnici. Selekce aktivního LATCH obvodů je realizována pomocí 74HC237. Sběrnice je napojená na PIC, který cyklicky čte a zapisuje do LATCH obvodů (Schéma 6.6).

Dále je na ŘM osazen SMD obvod MAX232I, který obstarává převod TTL signálů sériové linky na standard RS232. Sériová linka je zakončena konektorem CAN9. To umožňuje připojení řídícího modulu k počítači resp. PLC. Schéma zapojení MAX232I je v příloze 6.9.



Obrázek 19: Blokové schéma řídícího modulu

4.3 Paralelní rozhraní

Paralelní rozhraní slouží k ovládání akčních členů a k detekci stavů tlačítek na modelu výtahu.

Výstupy modelu		Vstupy modelu	
Číslo pinu	Název pinu	Číslo pinu	Název pinu
1	`Jizda_nahoru`	25	`Tlac_vytah_3p_d`
2	`Jizda_pomalu`	26	`Tlac_vytah_2p_u`
3	`Jizda_dolu`	27	`Tlac_vytah_2p_d`
4	`Led_vytah_3p_d`	28	`Tlac_vytah_1p_u`
5	`Led_vytah_2p_u`	29	`Tlac_vytah_1p_d`
6	`Led_vytah_2p_d`	30	`Tlac_vytah_p`
7	`Led_vytah_1p_u`	31	`Poloha_konc_u`
8	`Led_vytah_1p_d`	32	`Poloha_konc_d`
9	`Led_vytah_p`	33	`Poloha_zpomal_u`
10	`7_seg_kabina_0`	34	`Poloha_zpomal_d`
11	`7_seg_kabina_1`	35	`Poloha_3p`
12	`Led_kabina_pretizeni`	36	`Poloha_2p`
13	`Led_kabina_dvere`	37	`Poloha_1p`
14	`Led_kabina_3p`	38	`Poloha_p`
15	`Led_kabina_2p`	39	`Tlac_kabina_stop`
16	`Led_kabina_1p`	40	`Tlac_kabina_3p`
17	`Led_kabina_p`	41	`Tlac_kabina_2p`
18	`Adr0`	42	`Tlac_kabina_1p`
19	`Adr0`	43	`Tlac_kabina_p`
20	`NC`	44	`Data_platna`
21	`NC`	45	`NC`
22	`NC`	46	`NC`
23	`NC`	47	`NC`
24	`NC`	48	`NC`

Tabulka 3: Zapojení paralelního rozhraní na řídícím modulu

V kapitole 4.2 byla popsána vnitřní implemantace rozhraní. V této kapitole bude popsáno řízení modelu pomocí paralelního rozhraní. Úroveň `H` na vodiči rozhraní odpovídá napětové úrovni 24V. Naopak úroveň `L` značí 0V.

Komunikace na paralelním rozhraní je řízena z nadřazeného systému (PLC). PLC nastaví adresu (`Adr0`, `Adr1`) na neplatný výtah (`Adr0=L`, `Adr1=L`) a testuje signál `Data_platna`. Po jeho přechodu do stavu L nastaví výstupy včetně adresy výtahu, pro který jsou určeny. Na to model odpoví nastavením dat a přechodem `Data_platna` do H. Data jsou sejmuta PLC.

1. `Jizda_nahoru` Při úrovni `H` na tomto pinu paralelního rozhraní se kabina pohybuje ve výtahové šachtě směrem nahoru. Pokud kabina dojede na horní konec výtahové šachty, tak již nepokračuje v pohybu a tento stav je signalizován diodou (D1) a na výstupu paralelního rozhraní `Poloha_konc_d`. Pohyb kabiny je signalizován diodou (D20).
2. `Jizda_pomalu` Současně s `Jízda_nahoru` nebo na `Jizda_dolu` se kabina pohybuje poloviční rychlostí.
3. `Jizda_dolu` Funkce opačná k funkci `Jízda_nahoru`. Kabina výtahu se pohybuje směrem dolů a svítí žlutá dioda (D21) pod sedmisegmentovým displejem. Pokud kabina dojela na dolní doraz, tak se pohyb ukončí a tento stav je signalizován na `Poloha_konc_d` a červenou diodou (D18) u spodního konce výtahové šachty.
4. `Led_vytah_3p_d` Zelená LED dioda (D2) trojuhelníkového tvaru ukazující dolů je umístěna ve třetím patře. Dioda slouží k potvrzení stisku tlačítka `Tlac_vytah_3p_d` z programu PLC

směr dolů.

5. `Led_vytah_2p_u` (D6)
6. `Led_vytah_2p_d` (D7)
7. `Led_vytah_1p_u` (D11)
8. `Led_vytah_1p_d` (D12)
9. `Led_vytah_p` (D16)
10. `7_seg_kabina_0`
11. `7_seg_kabina_1` Dva bity určující znak zobrazený na sedmisegmentovém displeji umístěném v kabině výtahu podle následující tabulky:

7_seg_kabina_1	7_seg_kabina_0	číslice
H	H	3
H	L	2
L	H	1
L	L	P

12. `Led_kabina_pretizeni` Červená LED (D19) umístěná v kabině výtahu. V programu PLC má reagovat na stav přetížení výtahu.
13. `Led_kabina_dvere` Žlutá LED (D22) indikující otevřené dveře. V programu PLC zobrazuje stav dveří. Svítí znamená, že jsou dveře otevřeny.
14. `Led_kabina_3p` Zelená LED (D23) umístěná u tlačítka `Tlac_kabina_3p`. Při `H` svítí. V programu PLC stisk potvrzuje `Tlac_kabina_2p` v kabině výtahu.
15. `Led_kabina_2p` (D24)
16. `Led_kabina_1p` (D25)
17. `Led_kabina_p` (D26)
18. `Adr0` Dolni bit adresy výtahu
19. `Adr1` Horní bit adresy výtahu.

Adr1	Adr0	Výtah
H	H	3
H	L	2
L	H	1
L	L	žádný

Adresa výtahu určuje zobrazovací modul, na který je připojeno paralelní rozhraní. Mikrokontrolér PIC zjišťuje stav paralelního rozhraní postupně. Jako první dva bity načte adresu výtahu. Pokud není adresován výtah, stavy na dalších pinech paralelního rozhraní PIC ignoruje a nastavuje `Data_Platna` do L. Zobrazovací modul vlevo má adresu 1 modul uprostřed 2 a vpravo 3.

20. `NC0` Nepřiřazena funkce.
21. `NC1` Nepřiřazena funkce.
22. `NC2` Nepřiřazena funkce.
23. `NC3` Nepřiřazena funkce.
24. `NC3` Nepřiřazena funkce.
25. `Tlac_vytah_3p_d` Kulaté černé tlačítko (SW1) ve třetím patře pro přivolání výtahu a jízdu směrem dolů. Na výstupu je úroveň H pokud je tlačítko stisknuté.
26. `Tlac_vytah_2p_u` (SW2)
27. `Tlac_vytah_2p_d` (SW3)
28. `Tlac_vytah_1p_u` (SW4)
29. `Tlac_vytah_1p_d` (SW5)
30. `Tlac_vytah_p` (SW6)
31. `Poloha_konc_u` Havarijní čidlo horního dorazu ve výtahové šachtě. Sepnutí je opticky signalizováno pomocí červené diody (D1) umístěné u horního konce výtahové šachty. Při správné funkci výtahu by kabina neměla nikdy k čidlu dojet. Všechny čidla spíná a vypíná výtah pohybem v šachtě. V jeden okamžik se přenáší na paralelní sběrnici jen čidla výtahu, který je právě adresován.

32. `Poloha_konc_d` Havarijní čidlo dolního dorazu ve výtahové šachtě. Signalizováno pomocí červené diody (18).
33. `Poloha_zpomal_u` Čidlo detekující kabinu ve zpomalovacím/zrychlovacím bodu umístěném nad patrem. Sepnutí je opticky signalizováno pomocí žluté (D5,D10,D15) obdélníkové diody vpravo od výtahové šachty. Pokud kabina jela až doposud rychle, čidlo označuje bod kdy má zpomalit, aby do patra dojela kabina pomalu. Přepínání rychlostí řídí PLC pomocí signálu `Jízda_pomalu`. Pozor, pokud kabina výtahu jede rychle, čidlo je aktivní po dobu 71,1ms. Po této době kabina čidlo mine.
34. `Poloha_zpomal_d` (D4,D9,D14) Čidlo detekující kabinu pod patrem.
35. `Poloha_3p` Čidlo detekující kabinu ve třetím patře. Sepnutí je opticky signalizováno pomocí žluté obdélníkové diody (D3) mezi čidly `Poloha_konc_u` a `Poloha_zpomal_d`. Čidlo označuje místo, kde má kabina výtahu zastavit. Pozor pokud kabina jede pomalu, čidlo je aktivní po dobu 142ms.
36. `Poloha_2p` (D8)
37. `Poloha_1p` (D13)
38. `Poloha_p` (D17)
39. `Tlac_kabina_stop` (SW7) Tlačítko umístěné nad tlačítky v kabině. V programu PLC realizuje havarijní zastavení.
40. `Tlac_kabina_3p` (SW8) Tlačítko v kabině výtahu. V programu PLC funguje jako zdroj požadavku cestujících. Tento požadavek PLC vizualizuje rozsvícením diody `Led_kabina_3p`. (SW9)
41. `Tlac_kabina_2p` (SW8)
42. `Tlac_kabina_1p` (SW10)
43. `Tlac_kabina_p` (SW11)
44. `Data_platna`
45. `NC0` Nepřiřazena funkce.
46. `NC1` Nepřiřazena funkce.
47. `NC2` Nepřiřazena funkce.
48. `NC3` Nepřiřazena funkce.

Pro všechny výstupy platí podmínka správné adresace. Pokud výtah není zaadresován, tak se data ze zobrazovacího modulu nepřenesou na paralelní sběrnici.

4.3.1 Příklad toku dat mezi modelem a PLC WAGO

Na tomto příkladu je demonstrován způsob cesty signálu od zmáčknutí tlačítka na ZM k jeho detekci na PLC WAG0 a zpětnou cestu od změny signálu na WAG0 PLC po rozsvícení diody na ZM.

Při stisknutí tlačítka (SW11) prvního výtahu se změní úroveň na vstupu registru 74HC166 (U5) na log0. Při následovném scan cyklu tlačítek PIC, je tato hodnota pomocí posuvných registrů 74HC166 přesunuta do paměti PIC (U4) ŘM. V této paměti je uložena informace o stavu tlačítek všech tří výtahů. PIC v jiném cyklu zjišťuje stav paralelního rozhraní. Podle adresy `(adr_0,adr_1)=01` určí, že informace na paralelním rozhrani se týkají výtahu 1. Stav log.1 je přes vnitřní paralelní sběrnici zapsán do LATCH obvodu 74HC573 (U17). Dále přes výstupní optočlen PC847 se změní signál z TTL na 24V. Signál `Tlac_kabina_p` je přiveden přes konektor WAGO k WAG0 PLC. PLC reaguje na tento podnět změnou úrovně na `Led_kabina_p`. Informace se přenáší opět přes paralelní rozhraní, optočlen PC847, LATCH obvod 74HC573 (U11), vnitřní 8 bitovou sběrnicí do mikrokonroléru PIC. Zde je změna uložena do "videopaměti" reprezentující stav všech diod. Voláním procedury refresh jsou data pomocí signálu `CLK_IN` a `Data_in_0,Data_in_1,Data_in_2,` přesunuta z ŘM do ZM. Posledním zapsaným bitem se uvolní LATCH obvodu M5451Q (U4) a rozsvítí se dioda (D26).

4.4 Sériové rozhraní RS232

Sériová komunikace je implementována na mikrokontroléru PIC16F877A. V tomto odstavci jsou uvedeny základní parametry rozhraní. Topologie rozhraní je point to point. Provoz na sběrnici je duplexní s řídícím signálem CTS.

4.4.1 Základní parametry sériové rozhraní

- `Baud Rate:` `9600bD`
- `Parita:` `Žádná`
- `Delka slova:` `8bitu`
- `Stopbit:` `dva bity`
- `Prenos dat pomoci:` `TxD a RxD`
- `HW rízení toku dat:` `CTS rízený mikrokontrolérem.`
- `Signalova zem :` `GND`
- `Další signály nejsou zapojeny.`

4.4.2 Popis komunikačního protokolu

Obecná pravidla pro řídící slovo

Hlavní funkcí programu je správně identifikovat a dekódovat skupinu příchozích bytů po sériové lince tvořících řídící slovo. To se zajišťuje pomocí šablon slov uložených v programové paměti. Tato slova musí splňovat následující podmínky:

1. Slovo nesmí obsahovat znak `"<"`. Tento znak určuje začátek slova, ale není v šabloně uveden.
2. Slovo nesmí obsahovat znak `">"`, který se používá jako ukončovací znak slova.
3. Slova se musí lišit alespoň v jednom znaku (slovo nesmí být částí jiného slova).
4. Slovo nesmí obsahovat znak `"$"` neboť ve vnitřní implementaci se tento znak používá k označení číselné hodnoty.

HW řízení toku dat

Po přijetí posledního znaku slova (`">"`) je slovo uloženo do osmiurovňové fronty FIFO. Pokud je fronta plná z `50%` aktivuje se CTS. K deaktivaci CTS dochází při poklesu obsazení fronty pod `20%`.

4.4.3 Seznam řídících slov pro aplikaci výtah

Značky v seznamu mají následující význam. "⇒" znamená směr komunikace PLC ⇒ PIC. Opačná značka je obecná odpověď PLC ⇐ PIC. Příklady jsou označeny značkou "←". Znak `$` je při přenosu nahrazen jednou šestnáckovou číslicí. Znaky `$$` tedy reprezentují jeden byte dat.

Identifikační skupina

Tato komunikační část jednoznačně určuje vlastnosti zařízení. Lze vyslat najednou i bez řízení toku dat.

⇒ `"<PING>"` Je používán pro test hardwarového spojení s mikrokontrolérem PIC.

⇐ `"<PING OK>"` Pokud PIC nereaguje, PC může zkusit testovat jiné porty.

⇒ `"<IDN NAME>"` Identifikuje zařízení.

⇐ `"<IDN NAME Vytah Vyrobil:Zajic Josef>"`

⇒ "<IDN VERZE>"

⇐ "<IDN VERZE $.$>" Označuje změny v hardwaru desky plošných spojů. Tento udaj je také uveden na DPS.

← "<IDN VERZE 0.0>" Vzorek výtahu s jednou třetinou zobrazovacích prvků.

← "<IDN VERZE 0.1>" Kompletně osazená deska.

⇒ "<IDN MEM>" Identifikuje přídavnou externí paměť připojenou k mikrokontroléru pomocí I2C rozhraní.

⇐ "<IDN MEM "TYP PAMETI">"

← "<IDN MEM ABSENT>" Pamět není implementována.

← "<IDN MEM 24LC128>" Je připojeno 16kByte paměti ovládané pomocí příkazu. "<READ 0x$$$$>" a "<WRITE 0x$$$$ 0x$$>"

⇒ "<IDN SOFT>" Označuje změny ve firmwaru mikrokontroléru.

⇐ "<IDN SOFT $.$>"

← "<IDN SOFT 0.0>" Firmware se základní sadou příkazů upravený pro jediný výtah.

← "<IDN SOFT 0.1>" Tato verze ještě není naprogramována.

Skupina pomocných příkazů pro zápis a čtení z paměti PIC

⇒ "<EEPROM READ 0x$$>" Hodnota nahrazující znaky "$$" určuje adresu odkud se má číst z vnitřní paměti EEPROM.

⇐ "<EEPROM READ OK 0x$$ 0x$$>" První byte odpovědi je adresa a druhý jsou přečtená data z vnitřní paměti EEPROM mikrokontroléru.

→ "<EEPROM READ 0x00>" Přečte byte z pozice a 0x00 a PIC odpoví například slovem.

← "<EEPROM READ OK 0x00 0x01>"

⇒ "<EEPROM WRITE 0x$$ 0x$$>" Příkaz pro zápis do EEPROM. Opět první byte je adresa a druhý jsou data. Po dobu zápisu do paměti mikrokontroléru se další příkazy řadí do vstupní FIFO fronty.

⇐ "<EEPROM WRITE OK 0x$$ 0x$$>" Po zápisu do paměti je byte přečten a tato hodnota je poslána zpět jako odpověď. Pokud se hodnoty zapisujícího příkazu a odpovědi liší, tak se zápis nepodařil.

→ "<EEPROM WRITE 0x00 0xAA>" Zapíše na pozici 0x00 data 0xAA a zápis potvrdí slovem

← "<EEPROM WRITE OK 0x00 0xAA>".

⇒ "<REG 0x$$$$>" Pomocí příkazu REG lze zjistit hodnotu libovolného registru paměti mikrokontroléru. Paměť se zrcadlí po úsecích "0000 .. 01FF". To znamená že "0000 == 0200".

⇐ "<REG OK 0x$$$$ 0x$$>" Vrací hodnotu registru paměti mikrokontroléru.

← "<REG 0x0003>" Vrací dolních osm bitu čítače instrukcí.

← "<REG OK 0x0003 0x45>"

Skupina řídících příkazů

⇒ "<VIDEOMEM WRITE 0x$$ 0x$$>" Zapíše na adresu "videopaměti" jeden BYTE.

→ "<VIDEOMEM WRITE 0x00 0x0F>" Nastaví první až čtvrtou diodu LED v bargraphu prvního sloupce na ON a pátou až šestou diodu na OFF.

⇒ "<VYTAH GO 0x$$$$$$>" Příkaz s daty vstupu na paralelní rozhraní.

⇐ "<VYTAH GO OK 0x$$$$$$>" Odpověd s daty výstupu paralelního rozhraní.

⇒ "<VYTAH SERIAL ON>" Příkaz přepínající model do sériového módu.

⇐ "<VYTAH SERIAL ON OK>" Odpověd potvrzující přepnutí do sériového módu.

⇒ "<VYTAH2 0x$$$$$$$$$$$$$$$$$$$$>" Příkaz s daty vstupu.

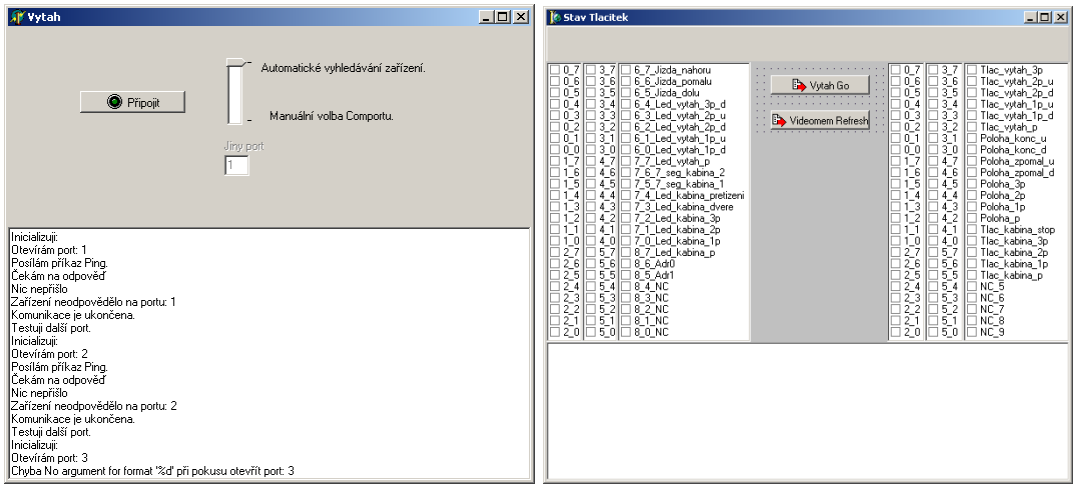
⇐ "<VYTAH3 0x$$$$$$$$$$$$$$$$$$$$>" Data výstupu.

4.4.4 Sériový mód modelu výtahu

Mód je určen pro řízení modelu přes sériové rozhraní RS232. Tento mód se zapíná příkazem "<VYTAH SERIAL ON>". Zapnutí módu způsobí následující akce.

- Paralelní rozhraní je na vstupu ignorováno a na výstupu je trvale Data_platna=L.
- Sériové rozhraní Master Slave se změní na typ Master Master.
- PIC změní chování programu z režimu signalizace úrovní na režim detekce hran. Při každé změně vnitřního stavu je vyslán příkaz "<VYTAH3 0x$$$$$$$$$$$$$$$$$$$$>", který reprezentuje stavy všech výstupů všech výtahů. Příkaz v sobě obsahuje stejnou informaci jako paralelní rozhraní všech výtahů.
- Jsou očekávány příkazy "<VYTAH2 0x$$$$$$$$$$$$$$$$$$$$>". Tyto příkazy mají stejnou informaci jako vstupní brány paralelního rozhraní všech výtahů.

Sériový mód je zrušen resetem modelu.



Obrázek 20: Program Výtah

4.4.5 Program výtah pro sériovou komunikaci

Program výtah je diagnostický prostředek pro model výtahu. Zobrazuje komunikaci na sériovém rozhraní.

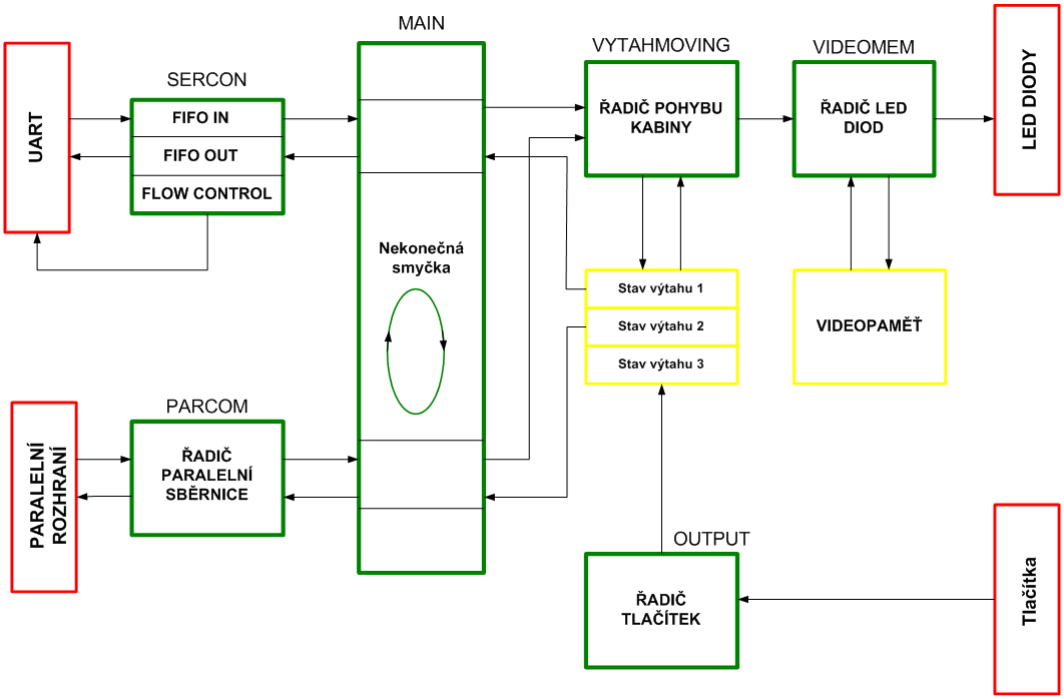
Program zobrazuje dvě okna. První, přihlašovací, se zobrazí po spuštění programu. Druhé, diagnostické, se zobrazí až po uspěšné detekci modelu výtahu.

Do diagnostického okna je model přepnut do sériového módu. Pak lze zadávat modelu příkazy, například o směru pohybu kabiny výtahu. Zároveň jsou zobrazovány stavy čidel na modelu výtahu.

4.5 Program pro mikrokontrolér PIC

Tento odstavec pojednává o realizaci programu pro mikrokontrolér PIC. Ten v konečné verzi obsahuje 5000 instrukcí asembleru.

Program je psán v jazyku MPASM. Jedná se o speciální asembler pro mikrokontrolér PIC. Navíc se tento asembler liší podle řady mikrokontroléru, na který je v konečné fázi vývoje implementován. Zdrojový program je přiložen v příloze.



Obrázek 21: Blokové schéma programu PIC

Největším úskalím v programování mikrokontroléru bylo správné rozdělení celého programu na menší programové celky. Tyto celky umožňují správně rozmístit rozsáhlý kód do stránkované paměti programu.

Program mikrokontroléru se skládá z těchto částí.

- `VYTAH`

  Hlavní program je tvořen nekonečnou smyčku. Z té se volají cyklicky podprogramy. Hlavní smyčka obsahuje také inicializační sekvenci, ve které jsou volány inicializační sekvence jednotlivých podprogramů. V nekonečné smyčce je umístěn WATCH DOG, který hlídá funkci všech podprogramů.

- `Videomem`

  Podprogram pro práci s tvz. "videopamětí". Videopamětí se rozumí bitově adresovatelná část paměti, která spolu s funkci UPLOAD umožňuje ovládání 315 LED diod. Změny stavu diod jsou prováděny pomocí maker nebo funkcí. Makra jsou méně náročná na výsledný strojový čas, ale zabírají větší místo paměti programu. Funkce jsou pomalejší, ale umožňují dynamicky adresovat konkrétní diodu.

- `Vytahmoving`

  Podprogram realizující pohyby kabiny ve výtahové šachtě a to pomocí šesti tabulek obsahující odkazy na podprogramy. Odkazy v tabulkách volají podprogramy, které upravují

hodnoty "videopaměti". Počet řádků v tabulce je stejný jako počet LED diod ve výtahové šachtě. `Vytahmoving` takto posouvá kabinou a zároveň rozsvěcuje čidla výtahové šachty. Rychlost posunu kabiny je odvozena pomocí časovače.

Další funkcí programu je dekódování stavu paralelního rozhraní. Nejde o práci s paralelním rozhraním, ale například o převedení dvoubitově zakódovaného patra na příslušné diody ve videopaměti.

- Output

  Podprogram output pracuje s posuvnými registry 74HC166 pro práci s tlačítky. Program sejme stav všech tlačítek a přidělí je jednotlivým vnitřním proměnným výtahů. Tato informace je pak dále předávána pomocí sériového či paralelního rozhraní.

- Parcom

  Paralelní komunikace na paralelní sběrnici. Jde o přemapovávací algoritmus, který přiřadí vnitřní reprezentaci paralelního rozhraní na vnější fyzické médium.

- Sercom

  Podprogram sercom realizuje další vrstvu nad protokolem RS232. Stará se o řízení provozu pomocí signálu CTS. Obsahuje vyrovnávací paměti. Kontroluje správnost přijatých příkazů. Podprogram je řízen pomocí přerušení od sériového portu. Přijatými příkazy se ovlivňuje chování hlavní nekonečné smyčky.

- EEPROM

  Jednoduchý Podprogram umožňující zápis a čtení do vnitřní paměti EEPROM. Po zápisu je zapsaný bit verifikován. Po dobu zápisu mohou běžet ostatní programy. Zápis je možno vyvolat příkazem ze sériového rozhraní.

- Function

  Obsahuje podpůrné makra pro předešlé programy.

4.6 Ovládání modelu výtahu pomocí PLC

Programování PLC vychází ze základní filozofie chování PLC tzv. skanovací smyčky (PLC scan) viz. lit. [1, kapitola 6.2.2]. V této smyčce se vždy nejprve sejmou vstupní data a převedou do proměnných dostupných z programu. Pak proběhne program PLC a na závěr se hodnoty výstupních proměnných zobrazí na výstupech. Je to vlastně technická realizace synchronního sekvenčního automatu typu Moore. Program musí respektovat tento skanovací cyklus a musí vždy proběhnout dostatečně rychle jinak by došlo aktivací funkce `watch dog` a k vyvolání chybového stavu.

Jak bylo uvedeno v kap. 2.2.3, lze PLC programovat různými způsoby. Vzhledem ke složitosti problému a zlepšení přehlednosti byla zvolena metoda ST (strukturovaný text).

Chování programu (výstupy a nový stav) je dáno vstupy a současným vnitřním stavem. Program by bylo možno řešit přechodovými tabulkami určujícími výstup a nový stav na základě vstupů a současného stavu. Takovéto řešení by nebylo příliš přehledné. Navíc by tabulky byly velice "řídké", protože jenom minimum kombinací vstupů a stavů je přípustné. Proto byla zvolena metoda klasického programování většinou řešící reakci na současný stav a vstupy příkazem CASE.

Program je rozdělen do takzvaných POU (Program Operation Unit). Hlavní POU je typu `Program Unit`. POU typu `Function` vrací právě jednu hodnotu na základě vstupních případně globálních proměnných. POU `Function Unit` provádí kód a může mít vstupní, vstupně/výstupní, výstupní a vnitřní proměné. Všechny POU mohou sdílet globální deklarace konstant, typů a proměnných.

4.6.1 Názvy a funkce POU

Zde jsou uvedeny jednotlivé POU a jejich funkce.

1. `PLC_PRG` je typu `Program Unit`. Hlavní cyklus programu.
2. `Popojed_vytahem` je typu `Function Unit`. Programová jednotka realizující stavový automat řízení výtahu.
3. `Paralel_IO` je typu `Function Unit`. Programová jednotka, která pro konkrétní výtah realizuje čtení paralelní sběrnice a zápis na ní.
4. `Par_Read` Programová jednotka volaná z `Paralel_IO`. Přemapovává vstupy.
5. `Par_Write` Programová jednotka volaná z `Paralel_IO`. Přemapovává výstupy.
6. `Serial_IO` Programová jednotka realizující obsluhu sériové sběrnice.
7. `Optim` Programová jednotka realizující optimalizaci řízení skupiny výtahů.
8. `Odstranvyresene` Programová jednotka použitá při optimalizaci řízení skupiny výtahů. Rozděluje vnější požadavky na ty, které budou vyřízeny současně jezdícími výtahy a na ty, které tuto jistotu nemají.
9. `Vytahobsluhuje` Pomocná funkce pro `Odstranvyresene`.
10. `Virt_doba_prijezdu` Funkce počítající odhad doby dojezdu výtahu do zadaného patra.
11. `Vah_fce_doby_prijezdu` Pomocná funkce pro `Virt_doba_prijezdu`. Heurestika pro odhad doby příjezdu.

4.6.2 Datové struktury řízení výtahu

Program je navržen obecně pro libovolný počet výtahů (`C_pocet_vytahu=3`) a libovolný počet pater (`C_pocet_pater=3`).

Stav výtahu je uložen ve struktuře `T_vytah`. Tyto struktury jsou uloženy do pole `Vytahy:ARRAY[1..C_pocet_vytahu] OF T_vytah`.

Program `Popojed_vytahem(Vytah:=Vytahy[Aktualni_Vytah]);` potom pracuje nad daty jednoho výtahu. Postupným voláním POU `Popojed_vytahem` dojde k obsluze všech výtahů. Postup je tedy obecný pro n výtahů.

Struktura `T_vytah` obsahuje kromě jiných proměnné `V_Stav`, `V_poloha`, `V_zastavit`, `V_patro`, `V_smer`, `V_patro_obratky`. Jednotlivé stavy těchto proměnných určují spolu se vstupy chování výtahu. Při jejich pochopení je konečný automat již nasnadě.

1. `V_Stav` uchovává momentální stav pohybu kabiny výtahu. Kabina se nachází v jednom ze stavů:
   `nekalibrován`, `kalibrace`, `stoji`, `stoji_pred_otevrenim`, `stoji_otevrene_dvere`, `rozjet_nahoru`, `rozjet_dolu`, `jede_nahoru_rychle`, `jede_nahoru_pomalu`, `jede_dolu_rychle`, `jede_dolu_pomalu`.
2. `V_poloha` uchovává momentální polohu kabiny:
   `neznámá`, `na_horním_koncáku`, `nad_patrem`, `stoji_v_patre`, `pod_patrem`, `mezipatro`, `na_dolnim_koncáku`.
   Polohy svázané s patrem jsou upřesněny proměnnou `V_Patro`.
3. `V_zastavit` je pole BOOL proměnných určující, zda výtah při průjezdu patrem zastaví.

4. `V_patro` je aktuální patro, ve kterém se kabina pohybuje.
5. `V_smer` určuje dosavadní směr jízdy:
   `smer_neurcen`, `smer_nahoru`, `smer_dolu`.
6. `V_patro_obratky` je proměnná typu BYTE, která určuje, kam až má výtah dojet před změnou směru.

Struktura `T_vytah` dále obsahuje přemapované vstupní a výstupní proměnné týkající se výtahu. Přemapování se děje v POU `Par_read`, `Par_write`. Je důležité, že všechny výstupní a vstupní proměnné, týkající se pater, jsou dostupné v polích indexovanými číslem patra. Řídící algoritmus řeší problémy jen v rámci jednoho patra a při přejezdu do jiného se mění pouze index. Algoritmus je pak obecný pro n-patrovou budovu.

4.6.3 Hlavní cyklus - POU PLC_PRG

Hlavní cyklus probíhá v POU `PLC_PRG`. V této smyčce se rozhoduje, jedná-li se o sériové nebo paralelní rozhraní. Podle typu rozhraní se snímají vstupní data (to může probíhat během několika skanovacích cyklů). Jsou-li k dispozici vstupní data, volá se obsluha výtahu resp. výtahů (POU `Popojed_vytahem`). Poté se vyvolá optimalizace skupinového řízení výtahů (POU `Optim`) a na závěr se aktivuje výstup dat.

Verze pro paralelní rozhraní vypadá následovně:

```
IF C_paralel_not_serial THEN
    PARALEL_IO;
    IF Paralel_IO_faze=Paralel_IO_Faze_data_ready THEN
        Popojed_vytahem(Vytah:=Vytahy[Aktualni_V44ytah]);
        Optim;
        Paralel_IO_Faze:= Paralel_IO_Faze_start_zapisu;
    END_IF;
    IF Paralel_IO_faze=Paralel_IO_Faze_konec_cyklu THEN
        IF Aktualni_vytah=C_Pocet_vytahu THEN
          Aktualni_vytah:=1;
        ELSE
            Aktualni_vytah:=Aktualni_vytah+1;
        END_IF;
        Paralel_IO_Faze:=Paralel_IO_Faze_start_cteni;
    END_IF
END_IF
```

4.6.4 Řízení pojezdu výtahu - POU Popojed_Vytahem

Tato jednotka realizuje vlastní sekvenční automat chování jednoho výtahu. Po zapnutí je výtah ve stavu nekalibrován a musí se nejprve zkalibrovat (pokud nestojí na čidle některého z pater). Kalibrace se provede pomalou jízdou směrem dolů až do dosažení patra nebo spodního havarijního dorazu. V druhém případě se pomalu jede směrem nahoru do přízemí. Stojící výtah otevře dveře po stisku tlačítka směr nahoru resp. směr dolů v aktualním patře. Dále výtah samostatně vyřizuje požadavky z kabiny aktuálním směrem až do jejich vyčerpání. Pak případně pokračuje v jízdě až do patra `V_patro_obratky`. Pak změní směr a vyřizuje požadavky opačným směrem. Pokud při cestě daným směrem je v patře vnější požadavek na cestu tímto směrem, výtah zastaví a požadavek vyřídí. Výtah se samostatně nerozjíždí vyřizovat vnější požadavky. K tomu musí být vyzván optimalizačním algoritmem. Nejsou-li vnitřní požadavky a je-li patro obrátky rovno aktuálnímu patru, výtah stojí.

Úsek POU realizující počáteční kalibraci vypadá takto:

```
CASE Vytah.V_Stav OF
    nekalibrovan:
        Vytah.Vo_Jizda_dolu:=TRUE;
        Vytah.Vo_Jizda_pomalu :=TRUE;
        Vytah.V_Stav:=kalibrace;
    kalibrace:
        IF Vytah.Vi_poloha_konc_d THEN
            Vytah.Vo_Jizda_nahoru:=TRUE;
            Vytah.Vo_Jizda_dolu:=FALSE;
            Vytah.V_Poloha:=Pod_patrem;
            Vytah.V_Patro:=0;
            Vytah.V_Stav:=jede_nahoru_pomalu;
            Vytah.V_Zastavit[0]:=TRUE;
        ELSE
            FOR i:=1 TO C_pocet_pater DO
                IF Vytah.Vi_poloha_patro[i] THEN
                    Vytah.Vo_Jizda_dolu:=FALSE;
                    Vytah.V_Patro:=i;
                    Vytah.V_Stav:=stoji;
                END_IF
            END_FOR
        END_IF
        Vytah.V_smer:=Smer_neurcen;
        Vytah.V_Patro_obratky:=Vytah.V_patro;
```

Kód psaný v ST lze považovat za samovysvětlující, zvláště jeli doplněn komentáři. Proto nejsou u jednotlivých POU uvedena podrobná bloková schémata.

4.6.5 Optimalizace chování výtahů

Pro optimalizaci pojezdů výtahu je zapotřebí nejdříve vysvětlit několik pojmů, které jsou v této kapitole použity.

- Vnitřní požadavek

  Požadavek vzniká stiskem tlačítka v kabině. Protože v modelu neexistuje podlahové čidlo, zmizí požadavek jen dojezdem do příslušného patra. Obsluha požadavku je zajištěna jednotkou `Popojed_vytahem`.

- Vnější požadavek

  Vzniká stiskem tlačítka přivolání výtahu. Účelem optimalizace je dosáhnout toho, že také vnější požadavek bude splněn v konečném čase. Vnější požadaveky lze rozdělit podle požadovaného směru jízdy na vnější požadavek směrem nahoru a vnější požadavek směrem dolů.

- Spořádaný cestující

  Algoritmus upřednostňuje požadavky takzvaně spořádaných cestujících. Pokud cestující zadá vnitřní požadavek opačného směru než zadal vnější požadavek (přistoupil do kabiny jedoucí jedním směrem a stisknul tlačítko patra v opačném směru), není spořádaný cestující. Proto bude kabina pokračovat v původním směru jízdy a to do doby, než se vyřídí všechny požadavky spořádaných cestujících, případně požadavek skupinového řízení daný patrem obrátky.

- Patro obrátky

  Nastavení patra obrátky způsobí dojezd kabiny výtahu do daného patra, i když nejsou vnitřní požadavky.

- Váhová funkce doby příjezdu

  Pokud máme budovu s více výtahy, je nutné vybrat výtah, který vnější požadavek obslouží. Pro volbu výtahu je použita Váhová funkce doby příjezdu. Jedná se o odhad doby příjezdu kabiny založený na směru jízdy kabiny, vzdálenosti kabiny od požadavku a počtu plánovaných zastávek.

4.6.6 Optimalizace chování jednoho výtahu

Většina starších výtahu optimalizaci vůbec neřeší. Vnější požadavky jsou v lepším případě umístěny do fronty a řešeny podle pořadí v jakém přišly, v horším případě se vyřizuje vždy jen jeden požadavek (když výtah jede, nelze ho přivolat).

Předpokládejme moderní výtah vybavený pro rozlišení směru vnějších požadavků. Pro takový případ lze nalézt optimálnější řízení pojezdů jednoho výtahu. Vnitřní požadavek je stále prioritní, ale kabina cestou přibírá vnější požadavky se stejným směrem. Směr jízdy kabiny se mění minimálně. Pokud se během jízdy vyskytne vnitřní požadavek opačným směrem, znamená to porušení pravidla spořádaného cestujícího. Vnitřní požadavky určují patro obrátky. Pokud jsou vnitřní požadavky vyřízené, výtah zareaguje na neobsloužené vnější požadavky (mimo rozsah pater obrátky). Tento algoritmus selhává v případě, kdy vnitřní požadavky nutí jezdit kabinu mezi dvěma patry. Potom musí být zvýšena priorita vnějších požadavků, což řeší další algoritmus.

Algoritmus se zaručenou konečnou dobou obsloužení vnějšího požadavku. V tomto algoritmu může být patro obrátky zvětšeno podle vnějších požadavků. Prakticky to znamená, že kabina, která splnila poslední vnitřní požadavek v daném směru pokračuje ve směru jízdy až do patra obrátky. Kabina v tomto případě obsluhuje vnitřní a vnější požadavky ve směru jízdy.

4.6.7 Optimalizace skupinového řízení výtahů - POU 0ptim

Realizovaný algoritmus obsluhy výtahu zajišťuje optimalizaci obsluhy "spořádaného cestujícího". Zároveň je zajištěna obsluha příslušných vnějších požadavků jejich sběrem v aktuálním směru. Algoritmus skupinového řízení společný všem výtahům pro zbývající vnější požadavky pomocí váhové funkce určí výtah, kterému obsluhu požadavku vnutí změnou patra obrátky. Tak je zaručeno obsloužení všech vnějšího požadavku v konečné době.

Požadavek se nevnucuje výtahu, který jede směrem od patra, ve kterém požadavek vznikl. To znamená, že může nastat situace, kdy není žádný vhodný výtah k dispozici. V konečné době musí být nějaký výtah "vhodný" (zastaví se a obrátí směr jízdy, nejpozději v posledním patře resp. přízemí).

Vnější požadavek vznikl mezi kabinou a patrem obrátky	Kabina jede k místu vzniku vnějšího požadavku	Směr vnějšího požadavku je stejný jako směr jízdy kabiny	Grafické znázornění situace	Kabina vyřeší vnější požadavek cestou	Zásah algoritmu skupinového řízení
Ano	Ano	Ano	$\Rightarrow\rightarrow\hookleftarrow$	Ano	Netřeba
Ano	Ano	Ne	$\Rightarrow\leftarrow\hookleftarrow$	Ne	Nelze
Ano	Ne	Ano	$\Leftarrow\leftarrow\hookleftarrow$	Nenastane	Nenastane
Ano	Ne	Ne	$\Leftarrow\rightarrow\hookleftarrow$	Nenastane	Nenastane
Ne	Ano	Ano	$\Rightarrow\hookleftarrow\rightarrow$	Ne	Možný
Ne	Ano	Ne	$\Rightarrow\hookleftarrow\leftarrow$	Ne	Možný
Ne	Ne	Ano	$\rightarrow\Rightarrow\hookleftarrow$	Ne	Nelze
Ne	Ne	ne	$\leftarrow\Rightarrow\hookleftarrow$	Ne	Nelze

Tabulka 4: Tabulka případů, kdy chování výtahů řeší alg.skupinového řízení

Tabulka 4 vyjmenovává možné situace a chování algoritmu skupinového řízení. Značky uvedené v tabulce mají následující význam:

$\Rightarrow$ Kabina výtahu a její směr jízdy.
$\rightarrow$ Vnější požadavek a jeho směr.
$\hookleftarrow$ Patro obrátky.

Vhodnost výtahu pro vnucení vnějšího požadavku se posuzuje funkcí odhadující dobu příjezdu kabiny do daného patra a ochotu akceptovat vnější požadavek. Tato funkce zohledňuje:

- Vzdálenost kabiny od aktuálního patra. Čím je vzdálenost větší, tím tato kabina přijede později.
- Počet jistých zastávek na cestě mezi současnou polohou a patrem požadavku. Čím je jich více, tím kabina pojede déle.
- Směr jízdy. Přijíždí-li kabina proti směru požadavku, hrozí nebezpečí, že přistoupí spořádaný cestující a svým požadavkem patro přejede. Toto riziko je tím větší, čím je kabina dále od aktuálního patra.

Váhová funkce je naprogramována v POU `Virt_doba_prijezdu`.

5 Závěr

Výsledkem mojí práce je funkční model skupiny tří výtahů pohybujících se mezi přízemím a třetím patrem řešený plně elektronicky. Model je realizován včetně technické a výrobní dokumentace. Model komunikuje s vnějším světem pomocí paralelního a sériového rozhraní. Základem modelu je mikrokontrolér PIC16F877 s řídícím programem o velikosti přibližně 5k slov.

Dále je realizován program v programovacím jazyce ST pro programovatelný automat PLC zajišťující optimalizované ovládání pojezdů jednotlivých výtahů a skupinové řízení tří výtahů. Tento program může být dále rozvíjen. Například po propojení PLC s výkonějším nadřazeným systémem lze přenést optimalizaci skupinového řízení na tento systém. Případně lze využít nadřazený systém pro sběr statistických údajů o provozu výtahů.

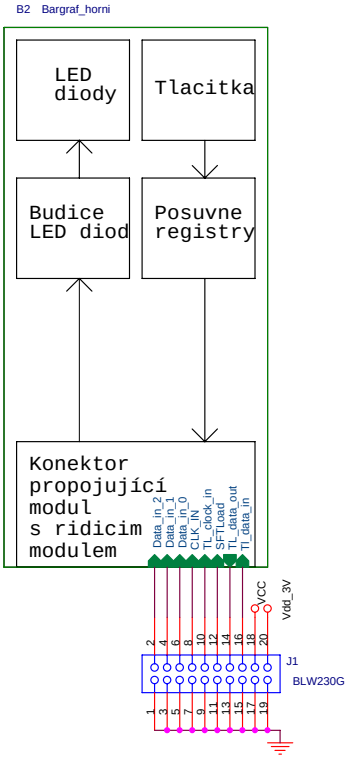
Je vytvořen program pro ovládání modelu prostřednictvím PC. Tento program byl použit při oživování modelu. V budoucnu může sloužit jako základ pro vytvoření programu ovládání modelu z PC jako alternativu ovládání z PLC.

V rámci práce byla provedena analýza různých způsobů komunikace modelu s vnějším světem včetně průmyslové sběrnice LONWORKS.

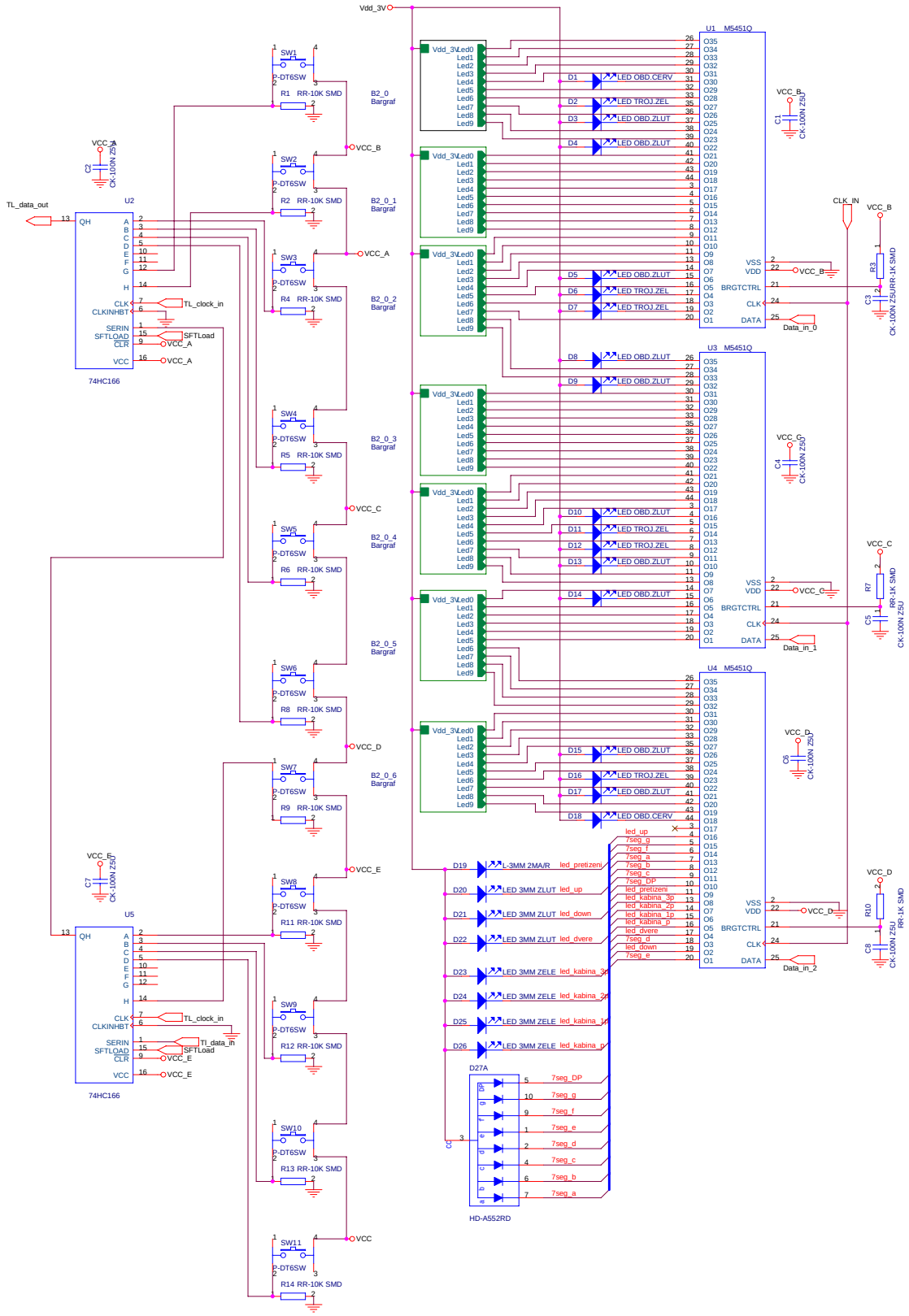
Drobnou úpravou řídícího programu mikrokontroléru PIC by bylo možno zajistit souběžné ovládání modelu prostřednictvím paralelního rozhraní a simulaci stisku tlačítek zadávaných prostřednictvím sériového rozhraní. Tím by bylo možné ověřovat chování ovládacího programu PLC pomocí strojově generovaných požadavků z PC.

6 Přílohy

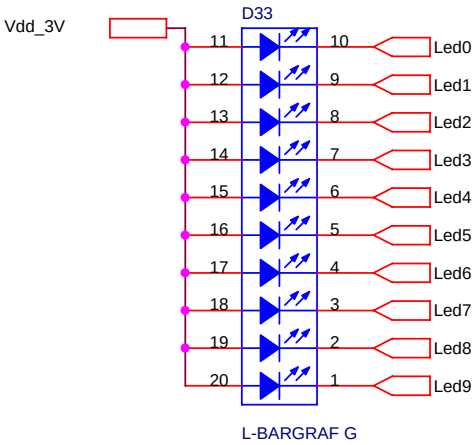
6.1 Blokové schéma zobrazovacího modulu



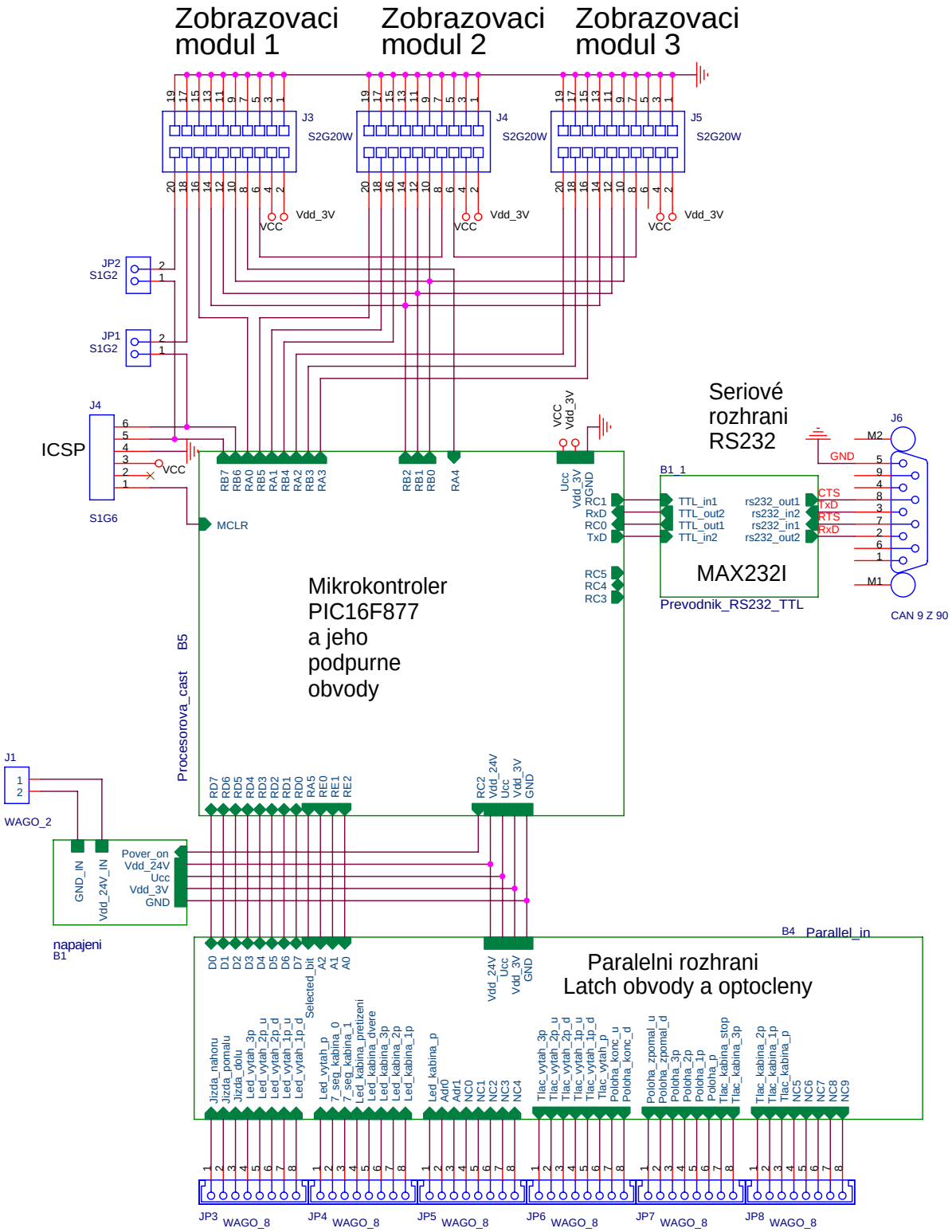
6.2 Podrobné schéma zobrazovacího modulu



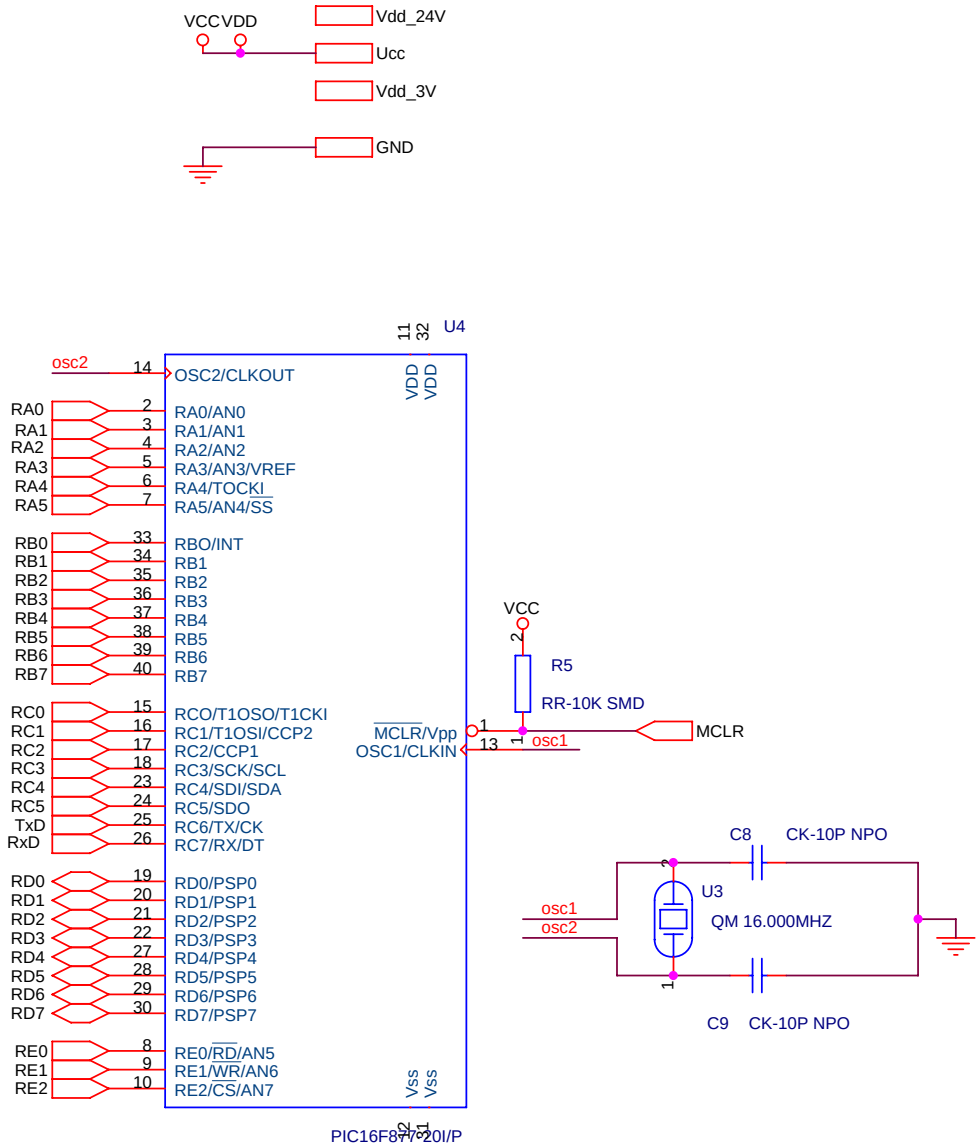
6.3 Zapojení Bargrafu ve schématu zobrazovacího modulu



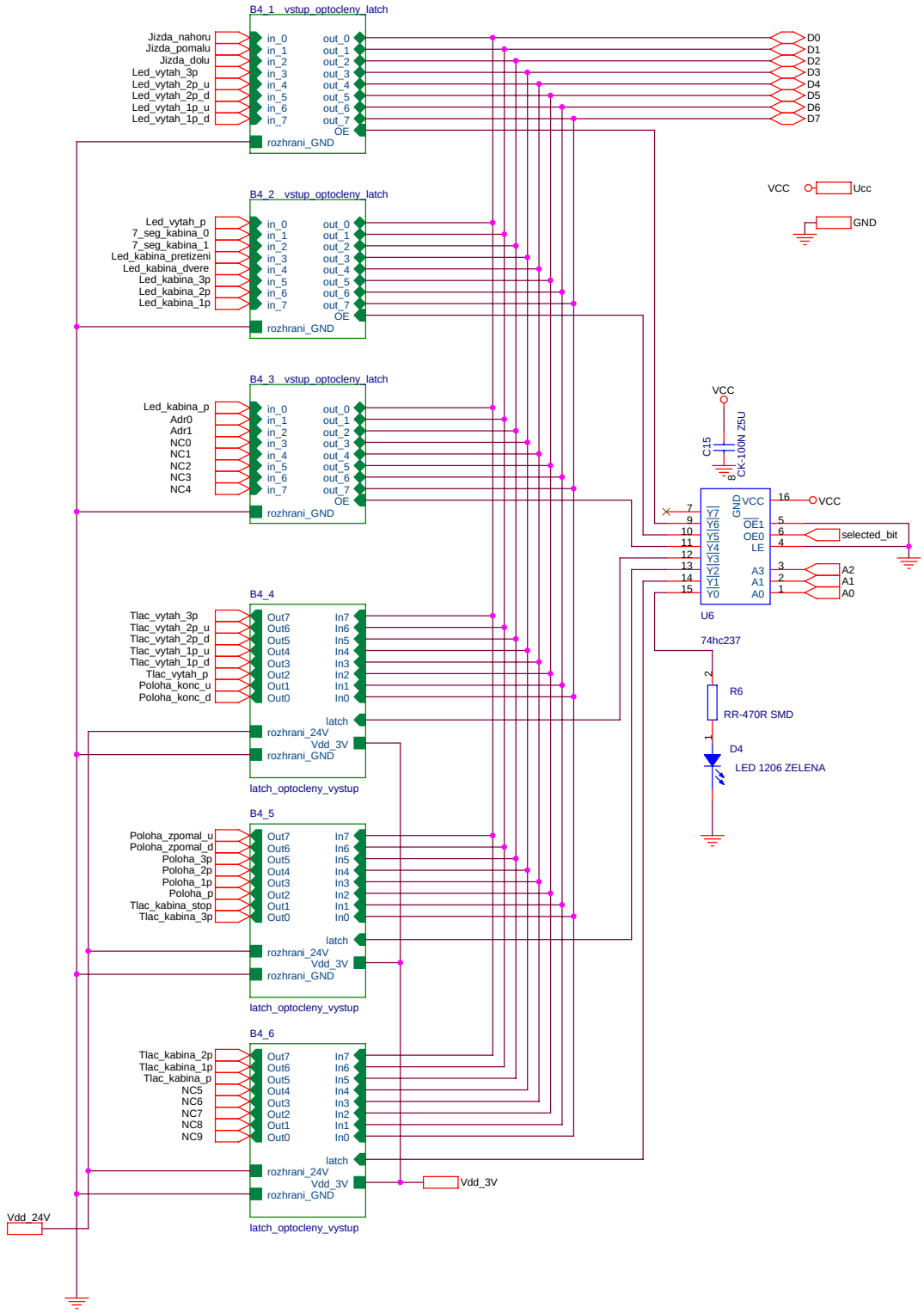
6.4 Blokové schéma řídícího modulu



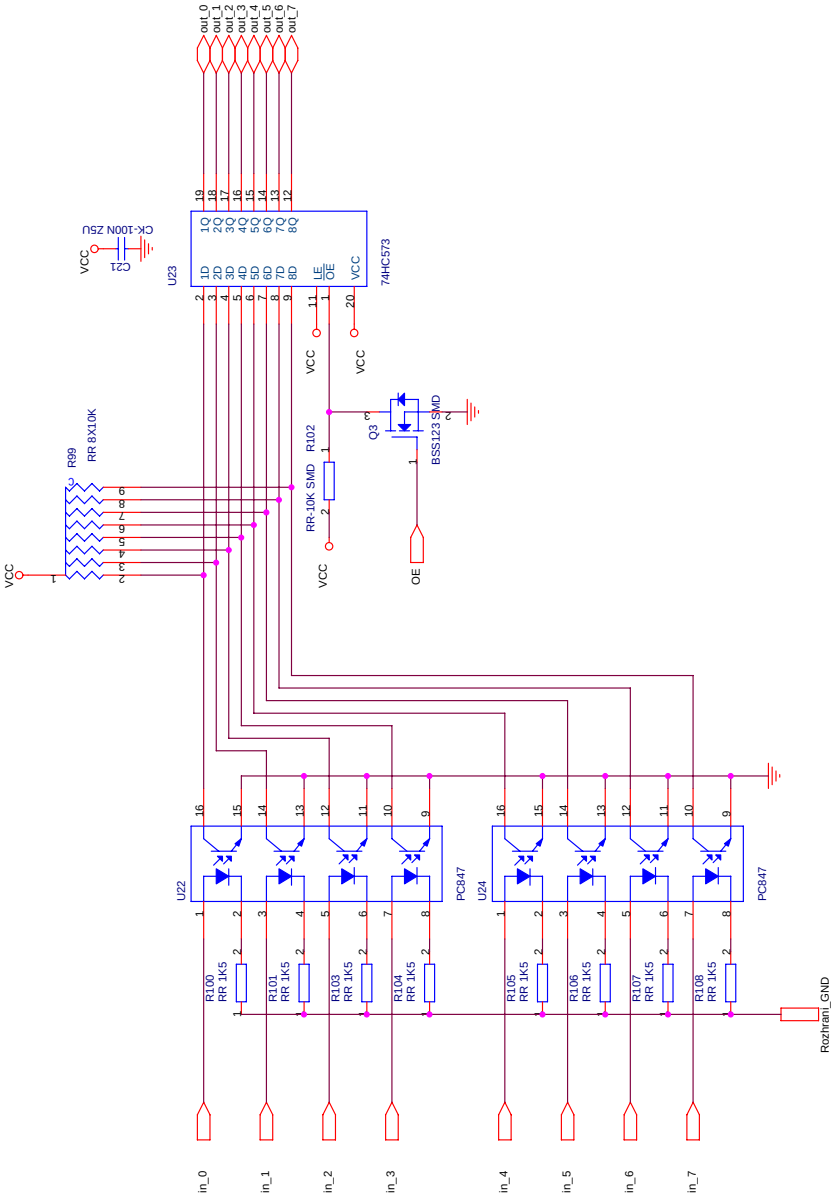
6.5 Podrobné schéma zapojení mikrokontroléru PIC16f877



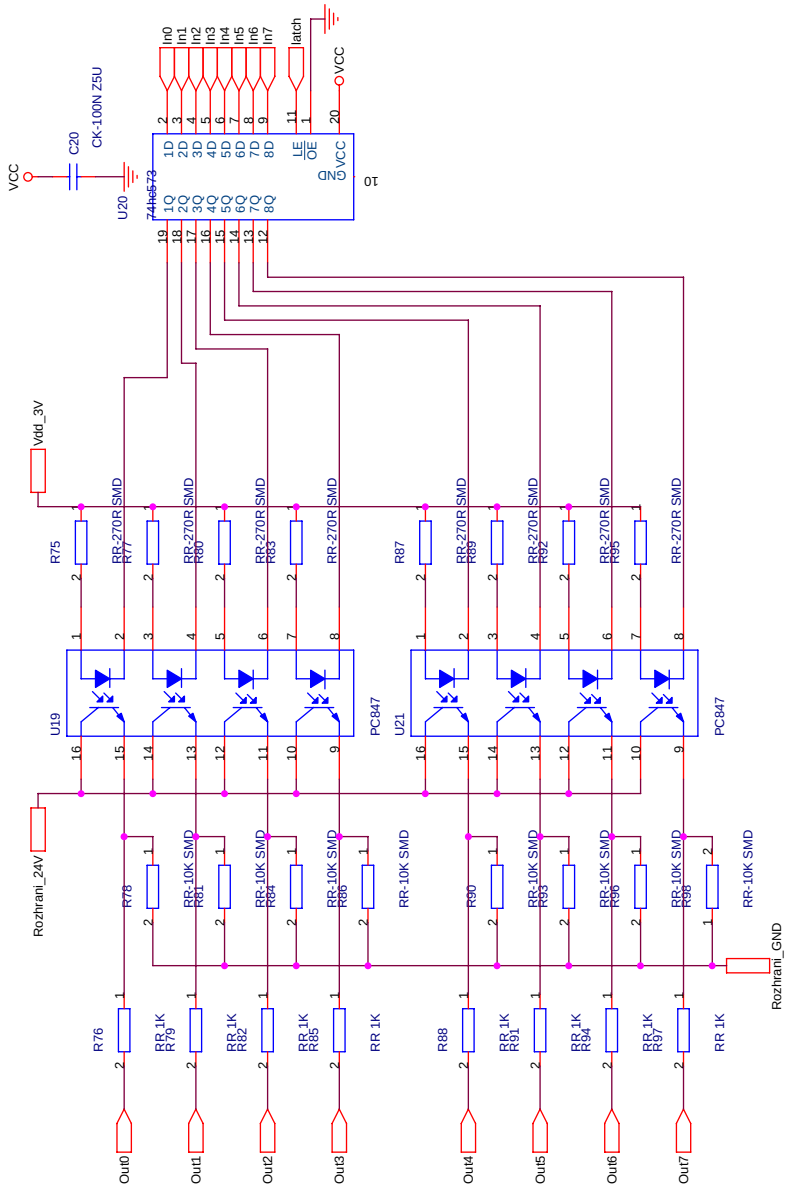
6.6 Blokové schéma paralelního rozhraní na řídícím modulu



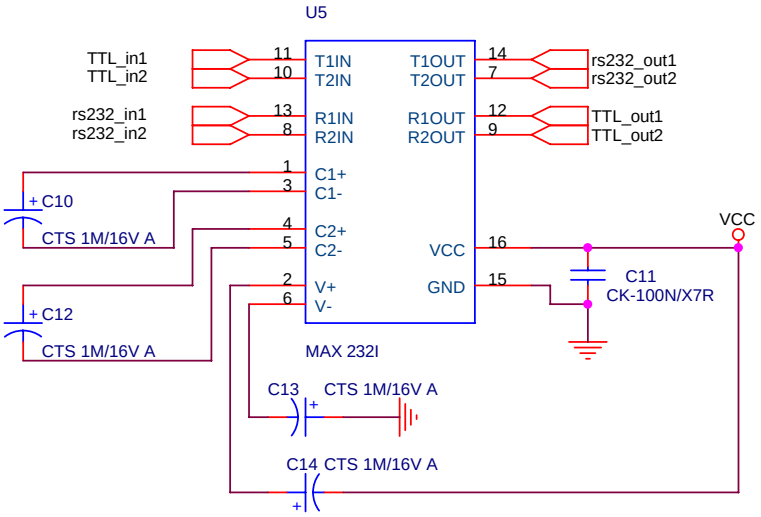
6.7 Schéma zapojení vstupního optočlenu paralelního rozhraní



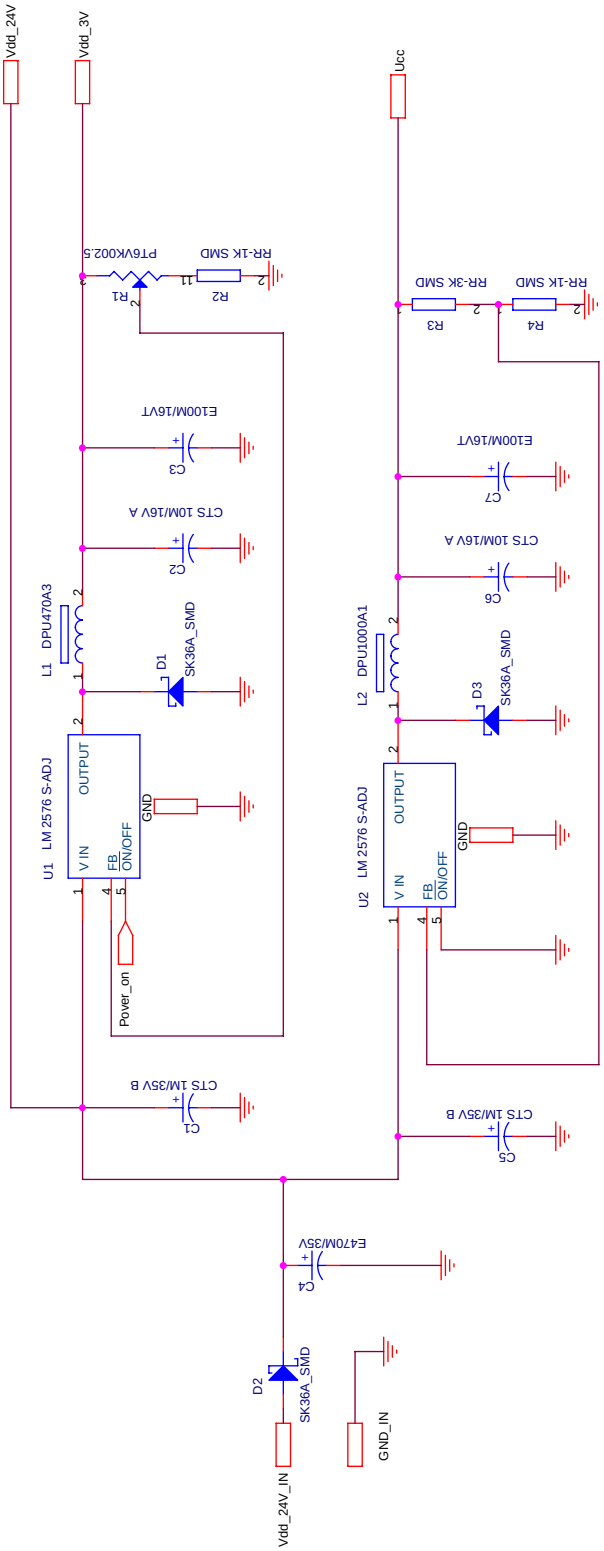
6.8 Schéma zapojení výstupního optočlenu paralelního rozhraní



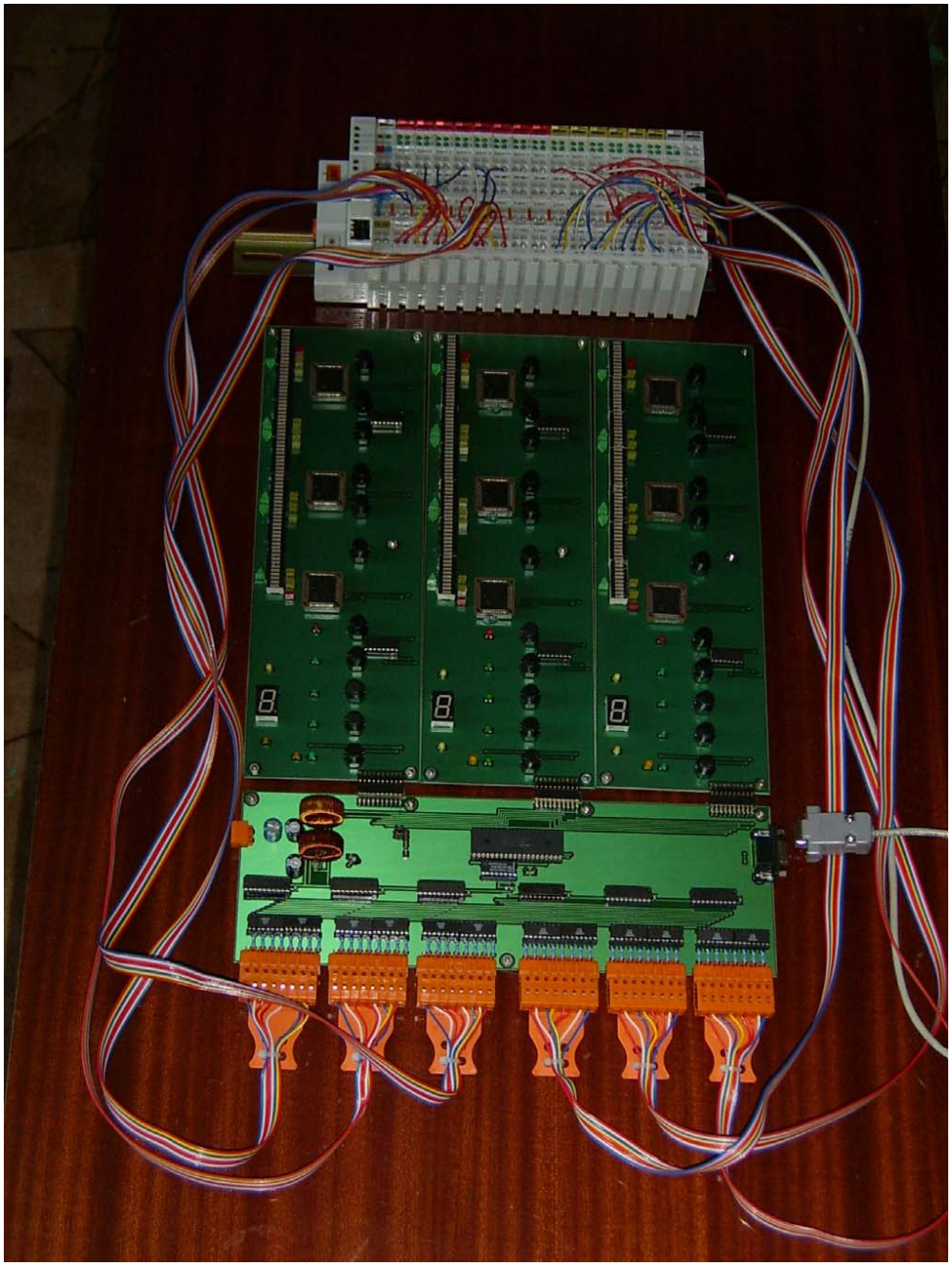
6.9 Podrobné schéma zapojení převodníku MAX232I na řídícím modulu



6.10 Podrobné schéma zapojení napájecích stabilizátorů na řídícím modulu



6.11 Fotografie modelu výtahu s připojeným PLC WAGO na paralelní rozhraní



7 Přehled použitých zkratek a definovaných pojmů

ANSI American National Standards Institute
Americký národní úoad pro normalizace

CSMA/CD Carrier Sense Multiple Access/Collision Detection
Vícenásobný přístup s odposlouchávaním nosného kmitočtu s detekcí kolizí

CH Chip-hosted
Výraz v terminologii LONWORKS definující aplikaci realizovanou na procesoru Neuron

Elevator Group Control Skupinové řízení výtahu

HB Host-based
Výraz v terminologii LONWORKS definující spolupráci procesoru Neuron s jiným procesorem

ISO International Standard Organization
Mezinárodní organizace pro standardizaci

LATCH Záchytný obvod

LON Local Operating Network
Průmyslová komunikační sběrnice definovaná společností Echelon

MAC Media Access Control
Řízení přístupu k médiu

MZM Zobrazovací modul s multiplexovaným zobrazováním

Neuron C Programovací jazyk odvozený od ANSI C. Slouží pro vývoj aplikací, které budou spu1tiny na procesoru Neuron Chip

Neuron Chip Speciální procesor používaný v technologii LONWORKS

OSI Open System Interconnection
Propojení definované normami ISO pro výměnu dat pomocí sedmi vrstev

PLC Programmable Logic Controller
Programovatelné logické řídicí jednotky

POU Program Organization Unit Blok
Programu sloužící k realizaci programu pro PLC

ŘM Řídící modul modelu výtahu

SCPT Standard Configuration Property Type

SNVT Standard Network Variable Type

Videopapěť Videopamětí se rozumí bitově adresovatelná část paměti, která spolu s funkci UPLOAD umožňuje ovládání LED diod

ST Structure Text
Strukturovaný text - programovací jazyk pro PLC

ZM Zobrazovací modul modelu výtahu

8 Seznam adresářů na přiloženém CD

PLC Zdojové soubory pro program v PLC

ASMPIC Zdrojové soubory pro PIC

ORCAD Schémata a podklady pro výrobu plošných spojů

SEZNAM_SOUCASTEK Seznamy součástek tříděné podle distributorů

PRILOHY Články uvedené v Referencích

DATASHEET Dokumentace použitých součástek

Reference

[1] Jiří Bayer,Zdeněk Hanzálek, Richard Šusta Logické systémy pro řízení Vydavatelství ČVUT, Zikova 4, 166 36 Praha 6 v roce 2000 4.6

[2] Data Sheet PIC16F87X 28/40-Pin 8-Bit CMOS FLASH Microcontrollers *Microchip Technology Incorporated. Printed in the USA.* 2001
"http://www.microchip.com/" 2.1.2

[3] Firemní stránky WAGO INOVATIVE CONNECTIONS *WWAGO Kontakttechnik GmbH Postfach 2880 D-32385 Minden Germany.*
"http://www.wago.cz"
"http://www.wagocatalog.com/okv3/index.asp?lid=7&cid=48&str_from_home=first"

[4] Databáze dokumentací integrovaných obvodů
"http://www.alldatasheet.com/"

[5] Firemní stránky Embedded tools and C compilers for developing compact, efficient code
"http://www.htsoft.com/"

[6] Firemní stránky Asix s.r.o. Staropramenná 4, 150 00 Praha 5 - Smíchov
"www.asix.cz"

[7] Vojáček Antonín Sběrnice LonWorks dil 1 Úvod
"http://automatizace.hw.cz/view.php?cisloclanku=2005040501"

[8] Vojáček Antonín Sběrnice LonWorks dil 2 LonTalk protokol
"http://automatizace.hw.cz/view.php?cisloclanku=2005041101"

[9] Podklady pro přednášku Lecture 2 Technology overview od firmy Lonix
"www.lonix.com"
"http://www.lonix.fi/files/Lecture-2_Technology_overview_2003-03-30.pdf"

[10] Norma IEC 61131-3: a standard programming resource
"http://www.plcopen.org"
"http://www.plcopen.org/TC1/Intro_IEC_March04.doc"

[11] Data Sheet Paralel od firmy Echelon "http://www.echelon.com/"
"http://www.echelon.com/support/documentation/bulletin/005-0021-01C.pdf"

[12] Data Sheet Echelon LTS-20 User‘s Guide od firmy Echelon
"http://www.echelon.com/support/documentation/manuals/078-0181-01C.pdf" 2, 5

[13] Data Sheet Host Application Programmer‘s Guide od firmy Echelon
"http://www.echelon.com/support/documentation/manuals/078-0016-01B.pdf" 5, 6

[14] Data Sheet SLTA- 10 Adapter and PSG/3 User's Guide od firmy Echelon
"http://www.echelon.com/support/documentation/manuals/078-0160-01E.pdf" 6

[15] Data Sheet ShortStackUG od firmy Echelon
"http://www.echelon.com/support/documentation/Manuals/078-0189-01C.pdf"

[16] Data Sheet LONWORKS Microprocessor Interface Program(MIP) User's Guide od firmy Echelon
"http://www.echelon.com/support/documentation/manuals/078-0017-01C.pdf" 5

[17] Daniel Nikovski,Matthew Brand,Decision-Theoretic Group Elevator Scheduling MITSUBISHI ELECTRIC RESEARCH LABORATORIES TR2003-61 June 2003
"http://www.merl.com"
"http://www.merl.com/publications/TR2003-061/" 2.4

[18] David L. Pepyne,Christos G. Optimal Dispatching Control for Elevator Systems During Uppeak Traffic 1997
"http://vita.bu.edu/cgc/Published/TCSTprint1197.pdf" 2.4

[19] Cendelín Jiří Historie programovatelných automatů a jejich současné efektivní použití *Fakulta aplikovaných věd Západočeská univerzita v Plzni č.MSM 235200004* "www.automa.cz/automa/2003/au060306.htm"