ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

D I P L O M O V Á P R Á C E

David Šišlák

Řízení výtahu

Katedra řízení
Vedoucí diplomové práce: **Ing. Richard Šusta**

PRAHA 2003

Prohlášení

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

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

V Praze dne **18. ledna 2003**

…….
podpis

Abstrakt

V této diplomové práci jsem navrhl obecný model výtahu včetně pohybu cestujících, který je vhodný pro výuku předmětů řízení. Model se skládá z kabiny výtahu, která může pojíždět ve čtyřech patrech budovy, tlačítek pro přivolávání kabiny do jednotlivých pater a volících tlačítek v kabině. Pohon kabiny má navrženou dynamiku. Pohybující se cestující v modelu mohou být zraněni. Model může být nakonfigurován v několika různých režimech. Poté jsem vytvořil aplikaci s tímto modelem, která komunikuje s řídícím automatem pomocí vstupně výstupní karty. V závěru práce jsem navrhl jednoduché řízení výtahu.

Abstract

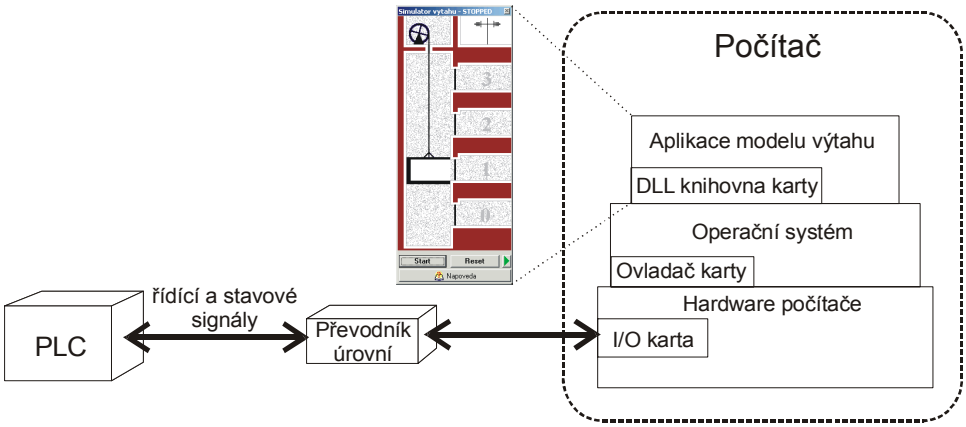
In this diploma thesis I designed the general prototype of elevator including movement of passengers. The elevator is suitable for the training of its control. The prototype consist of a cabin, buttons for calling the cabin into the relevant floor and buttons for choosing the destination. The cabin can move in the range of four floors. The driving mechanism of the cabin has its own dynamics. Passengers traveling in the prototype can be also injured. This prototype can be configured in a variety of modes. Thereafter, I created the software application with the prototype of elevator. The software application communicates with Programmable Logic Controller (PLC) via the input-output expanding card in the computer. Finally, I designed a simple control system of elevator.

Obsah

1 Úvod

Podnětem k mé práci byl požadavek pro vytvoření řiditelného modelu výtahu vhodného pro výuku, kde se studenti seznamují s řízením různých systémů. Nyní se používá v rámci předmětu **řídící systémy,** vyučovaného na katedře řízení ČVUT FEL, fyzický model výtahu, který je připojen na programovatelný logický automat (dále jen PLC). Model se skládá z kabiny výtahu, která může pojíždět nahoru a dolů ve čtyřech patrech budovy, tlačítek pro přivolávání kabiny do jednotlivých pater a volících tlačítek v kabině. Studenti vytvářejí řídící algoritmus výtahu pro PLC tak, aby se model choval podle předepsaných požadavků. Tento koncept systému není vhodný pro důkladné ověření správnosti řídícího algoritmu. Důvodem je pomalý pohyb kabiny daný konstrukci modelu a také problém, že simulaci pohybu přepravovaných osob si provádí student sám; musí přivolávat kabinu a volit cílové patro. Dále nelze snadno měnit parametry modelu a sledovat různé statistické údaje celého systému.

Nový koncept systému výtahu je zobrazen na obrázku 1.1. Pracoviště v laboratořích jsou vybavena počítači s operačním systémem Windows®[1)], do nich budou nainstalovány vstupně výstupní karty (dále jen I/O karta) PCI-1750 společnosti Advantech [10]. PLC bude propojeno s I/O kartou přes převodník úrovní řídících a stavových signálů. Virtuální model výtahu včetně pohybu cestujících je vytvořen jako aplikace, která komunikuje řídícími a stavovými signály přes I/O kartu s řídícím systémem. Na monitoru se zobrazuje přehledně aktuální stav modelu a je umožněno další nastavení parametrů.



Obrázek 1.1. Nový koncept systému výtahu

Výhody virtuálního modelu výtahu proti modelu fyzickému jsou:

- simulace současného pohybu více cestujících,
- možnost zranění cestujícího chybným řízením výtahu,
- snadná změna parametrů modelu,
- automatické sledování správnosti řízení.

[1] Windows® je registrovaná známka Microsoft corporation

Cílem práce je navržení modelu výtahu a vytvoření aplikace s tímto modelem včetně dokumentace. Aplikace bude vytvořena v prostředí Borland C++ Builder™ a bude komunikovat s I/O kartou pomocí ovladačů dodávaných výrobcem karty.

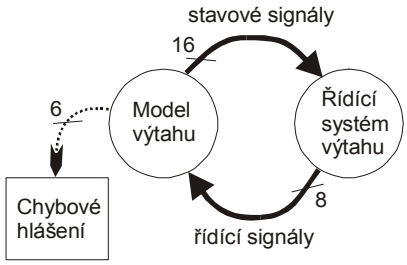
Modelováním systému výtahu se již některé práce zabývaly (viz např. [3], [4] a reference v nich uvedené). V této diplomové práci se nebudu zabývat řízením výtahu, ale budu se soustředit pouze na vytvoření modelu, který je přizpůsoben na výuku svými možnostmi volby parametrů, nastavením chování cestujících v modelu a také možnosti sledování vstupních a výstupních signálů. Aplikace bude také umožňovat generování vstupních signálů přímo v okně aplikace, takže nebude potřeba k ukázce funkce modelu I/O kartu s připojeným PLC.

Při návrhu modelu výtahu jsem vycházel z požadavku nezávislosti simulační na zobrazovací smyčce programu. Aplikace byla navržena jako více vláknová, kde vlákno se simulací má vyšší prioritu v operačním systému než ostatní vlákna aplikace, aby případné pomalé zobrazování z jakýchkoliv důvodů nemělo vliv na rychlost a přesnost simulace.

K práci je přiložen kompaktní disk se zdrojovými a zkompilovanými spustitelnými soubory všech vytvořených aplikací pro Windows®, obsah přiloženého CD je uveden v příloze II.

2 Návrh modelu výtahu

Systém výtahu lze rozdělit na *model výtahu* a *řídící systém*, obrázek 2.1. Tyto dvě části spolu komunikují pomocí *stavových* a *řídících* signálů. *Model výtahu* představuje řízenou část systému se svým chováním a *řídící systém* pomocí informací získaných ze *stavových* signálů řídí tento model. Výstupem *modelu výtahu* jsou navíc chybové hlášení, které informují o chybné řídící sekvenci. „Správné řízení“ modelu co nejdříve odbaví cestující, uspokojí tak korektně jejich požadavky, a současně nesmí dojít ke zranění osob. Dále pak musí splnit všechny bezpečnostní kritéria: kabina nesmí narazit, pohyb kabiny pouze se zavřenými dveřmi a stištěním tlačítka pro zastavení musí kabina co nejdříve zastavit.



Obrázek 2.1. Systém výtahu

V této kapitole se budu zabývat návrhem *modelu výtahu*. Návrh *řídícího systému* není cílem práce. V kapitole 4 uvádím jednoduchý příklad řízení tohoto modelu. Nejdříve musím specifikovat požadavky na model výtahu, definovat vstupní a výstupní signály modelu. Pak popíši logický model výtahu.

2.1 Definice požadavků

Po několika konzultacích s vedoucím diplomové práce jsem nadefinoval všechny požadavky týkající se modelu výtahu.

- **Čtyři podlaží** – kabina výtahu obsluhuje přízemí až třetí patro. Čtyři podlaží jsou zvolena z důvodu omezené velikosti grafické oblasti, ve které se zobrazuje stav modelu. Na každém podlaží mohou čekat dva aktivní cestující. „Aktivní cestující“ je reprezentovaný svým vlastním samostatným modelem. Další cestující mohou čekat ve frontě do doby, než se uvolní místo, pak se teprve stanou aktivními.
- **Nosnost kabiny jedna nebo dvě osoby** – je možno zvolit zda kabina výtahu přepravuje pouze jednu nebo dvě osoby.
- **Dvě rychlosti pojezdu kabiny** – kabina výtahu se může pohybovat pomalejší a rychlejší rychlostí. Tyto rychlosti je možno nastavit.
- **Dynamika pohonu kabiny** – při rozjezdu, zastavení, změně rychlosti nebo směru pohybu kabiny se projevuje dynamika pohonu, tedy zrychlení resp. zpomalení. Tato dynamika se může pro rozjezd a zastavení z pomalejší rychlosti vypnout a změna rychlosti je pak skoková.
- **Délka kontaktu v patře** – poloha kabiny se přenáší z modelu do řídícího systému pomocí dvou kontaktů *K* a *P*, způsob generování je zobrazen na obrázku 2.2. Délka těchto kontaktů jde nastavit.

- **Poloha kabiny po inicializaci modelu** – z důvodu způsobu přenášení polohy do řídícího systému pomocí dvou kontaktů *K* a *P*, potřebuje řídící systém předem znát, ve kterém podlaží se bude kabina nacházet po inicializaci modelu.
- **Přesnost zastavení kabiny pro nástup a výstup** – je možno zvolit, zda musí kabina výtahu stát úplně přesně v úrovni patra nebo zda je povolena určitá tolerance, aby cestující mohli nastoupit nebo vystoupit.
- **Plynulé otevírání a zavírání dveří** – dveře výtahu na jednotlivých podlažích se neotevírají skokově, ale plynule. Tak operace otevření a zavření dveří trvá určitý čas. Rychlost pohybu dveří je možno měnit.
- **Jedno nebo dvě přivolávací tlačítka** – na každém podlaží je podle nastavení jedno nebo dvě tlačítka pro přivolání kabiny. Ve variantě se dvěmi tlačítky cestující při přivolávání specifikují, zda chtějí cestovat nahoru nebo dolů. V této variantě je v přízemí samozřejmě tlačítko pouze pro směr nahoru a ve třetím patře je tlačítko pouze pro směr dolů.
- **Tlačítko STOP** – kabina výtahu obsahuje kromě tlačítek pro volbu cílového podlaží také tlačítko *STOP*, po jehož stištění by měl výtah zastavit. Všichni cestující v kabině poté zvolí své nové požadavky pro přepravu. Je možno zvolit, zda cestující smějí používat tlačítko *STOP* či nikoliv.
- **Délka stisku tlačítka** – je možno nastavit dobu, po kterou cestující bude držet stištěné tlačítko, a také pauzu mezi jednotlivými stisky na *skupině tlačítek*. *Skupinou tlačítek* se rozumí všechny tlačítka v kabině. Další skupiny tlačítek jsou na jednotlivých podlažích. Například: přijdou-li dva cestující na druhé podlaží, nejdříve bude přivolávat první a po dané pauze druhý.
- **Počet aktivních cestujících** – v modelu se může *aktivně pohybovat* současně omezený počet cestujících, daný nastavením. *Aktivním pohybem* se rozumí pohyb osob v modelu, při kterém přivolávají kabinu, nastupují, jsou přepravovaní a vystupují. Při tomto pohybu mohou být zraněni. Další cestující se pohybují pouze *pasivně*. *Pasivní pohyb* je pouze posuv v čekacích frontách na jednotlivých podlažích.
- **Rychlost pohybu osob** – všichni cestující v modelu se pohybují stejnou rychlostí a tato rychlost se může měnit.
- **Možnost zranění cestujících** – při této volbě mohou být cestující v modelu zraněni. Zranění může nastat v těchto případech:
  - zavřením dveří, nachází-li se cestující v prostoru dveří,
  - rozjezdem kabiny v okamžiku nástupu či výstupu,
  - vstupem cestujícího do výtahové šachty otevřenými dveřmi, pokud není přistavena kabina výtahu; cestující je rozdrcen projíždějící kabinou nebo zavírajícími se dveřmi.
- **Nervozita cestujících** – je možno nastavit dobu, kdy cestující, čekající na příjezd kabiny, začne projevovat nervozitu, a také dobu, kdy odejde, protože se nedočkal. Stejná nervozita se projevuje v kabině výtahu, pokud není cestující dopraven do svého cílového patra včas.
- **Systém chybových hlášení** – model poskytuje přehled o všech chybových hlášeních. Pro každý typ hlášení je možno nastavit, zda se jedná o kritickou chybu, která zablokuje model až do nové inicializace. Chybové hlášení jsou vyvolávány v následujících situacích:

- náraz kabiny,
- pohyb kabiny s otevřenými dveřmi,
- odchod cestujícího, který se nedočkal,
- rozdrcení cestujícího dveřmi,
- rozdrcení cestujícího kabinou výtahu,
- nepřijde-li povel k zastavení kabiny do ukončení stisku tlačítka *STOP*.

2.2 Definice vstupů a výstupů

Vstupy modelu jsou *řídící signály* a výstupy jsou *stavové signály*, obrázek 2.1. Tyto signály jsou logické, tedy každý signál může mít dvě úrovně: *aktivní* – odpovídá logické jedničce, *neaktivní* – odpovídá logické nule. Konkrétní přiřazení vstupních a výstupních signálů na konektor použité vstupně výstupní karty PCI-1750 je v kapitole 3.2.

2.2.1 Vstupy modelu

Navržený model má osm logických vstupů:

- **D0**, **D1**, **D2** a **D3** – signály ovládající dveře *D0* v přízemí až *D3* na třetím patře. Pokud je signál aktivní, mají se příslušné dveře otevřít, a pokud je neaktivní, tak se mají zavřít.
- **M+**, **M-**, **Slow** – signály pro ovládání pohonu kabiny výtahu.
- **Res** - aktivováním tohoto signálu je model inicializován, po deaktivaci se model zase spustí.

Význam signálů pro pohon kabiny je popsán v tabulce 2.1. Kolizní situace nastane při současné aktivaci signálů *M+* a *M-*. Tato situace má přesně definované chování pohonu. Pokud je jeden signál aktivován dříve než ten druhý, tak je brán v úvahu pouze signál, který byl aktivován dříve. Po deaktivaci prvního signálu, se pohon chová podle druhého signálu. Pokud jsou oba signály aktivovány současně a nejde rozlišit, který byl dříve, tak je preferován signál *M-*.

M+	M-	Slow	Motor pohonu kabiny
0	0	*x*	zastaven
1	0	1	pomalu nahoru
0	1	1	pomalu dolů
1	0	0	rychle nahoru
0	1	0	rychle dolů
1	1	*x*	kolizní situace, přesně definované chování

Tabulka 2.1. Signály pro ovládání posuvu kabiny výtahu

2.2.2 Výstupy modelu

Navržený model má šestnáct logických výstupů. Signál tlačítka je aktivní pokud je tlačítko stištěné.

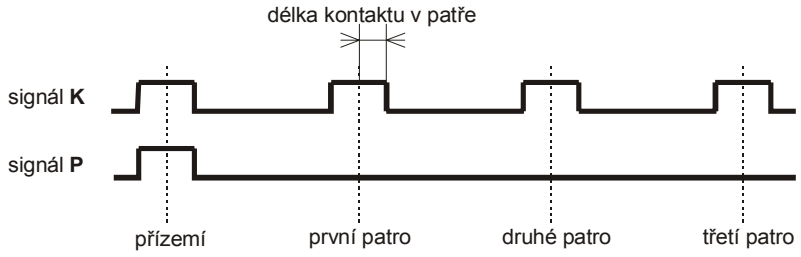
- **TK0**, **TK1**, **TK2** a **TK3** – signály tlačítek voliče v kabině výtahu odpovídající jednotlivým cílovým podlažím.

- **TP0**, **TP1**, **TP2**, **TP3**, **TP1N** a **TP2N** – signály tlačítek přivolávačů výtahu na podlažích.
- **S** – signál tlačítka *STOP* v kabině.
- **A** – signál aktivity modelu. Pokud je tento signál aktivní je model spuštěn, jinak je model zastaven.
- **Zat** – signál zatížení kabiny je aktivní, je-li v kabině alespoň jedna osoba.
- **O** – signál otevřených dveří. Pokud nejsou úplně zavřeny všechny dveře je tento signál aktivní.
- **P** a **K** – signály pro určování polohy kabiny výtahu.

Význam signálů přivolávacích tlačítek pro *jednotlačítkový* a *dvojtlačítkový* režim je popsán v tabulce 2.2. Způsob generování signálů *K* a *P* je zobrazen na obrázku 2.2. Silná čára v horní poloze znamená, že daný signál je aktivní. Délka těchto kontaktů v modelu se dá nastavit. Pokud je povolená tolerance zastavení pro nástup a výstup, mohou cestující nastupovat a vystupovat, pokud je signál *K* aktivní. Je-li vypnutá dynamika pohonu kabiny pro pomalou rychlost pohybu, tak při povelu zastavení a aktivním signálu *K* v poloze kabiny těsně před úrovní patra, není kabina zastavena okamžitě jako jindy, ale dojede přesně až do úrovně patra.

Signál	Jednotlačítkový režim	Dvojtlačítkový režim
TP0	tlačítko v přízemí	tlačítko nahoru v přízemí
TP1	tlačítko v prvním patře	tlačítko dolů v prvním patře
TP2	tlačítko v druhém patře	tlačítko dolů v druhém patře
TP3	tlačítko v třetím patře	tlačítko dolů v třetím patře
TP1N	*nevyužitý*	tlačítko nahoru v prvním patře
TP2N	*nevyužitý*	tlačítko nahoru v druhém patře

Tabulka 2.2. Signály přivolávacích tlačítek na patrech



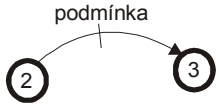
Obrázek 2.2. Signály pro určování polohy kabiny **K** a **P**

2.3 Použité značení ve schématech modelů

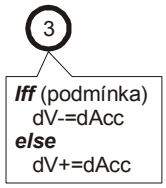
Pro popis chování modelu v následující kapitole používám schémata pro značení funkce automatu. Automat znázorněný na schématu se musí nacházet vždy v jednom ze znázorněných stavů. Pro upřesnění uvádím vysvětlení použitého značení.

- Stav 1 automatu, který má pouze přechody s podmínkou.

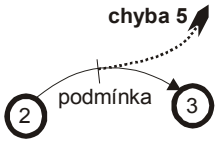
- Stav 1 automatu s časovačem. Příchodem automatu do stavu 1, je vynulován jeho vnitřní čítač. Každým krokem, ve kterém automat zůstane ve stavu 1, je tento čítač zvýšen o jedničku.



- Přechod s podmínkou. Pokud je automat ve stavu 2 a podmínka přechodu je splněna, automat v následujícím kroku přejde do stavu 3. Z každého stavu musí být vždy povolen jeden přechod.

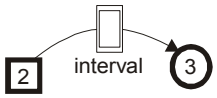


- Příchodem automatu do stavu 3 je vykonán obsah rámečku. V příkladu je podmínka, při jejíž splnění je od *dV* odečteno *dAcc*, při nesplnění podmínky je k *dV* přičteno *dAcc*. Za příchod do stavu se považuje rovněž setrvání ve stavu, způsobeném aktivací *smyčky*.

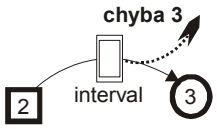


- Přechod s podmínkou, který splněním podmínky generuje současně chybovou zprávu 5.

- Smyčka. Nachází-li se automat v tomto stavu a není splněna žádná jiná podmínka vycházejícího přechodu, tak je aktivována podmínka „*jinak*“ a automat v následujícím kroku zůstane v tomto stavu.



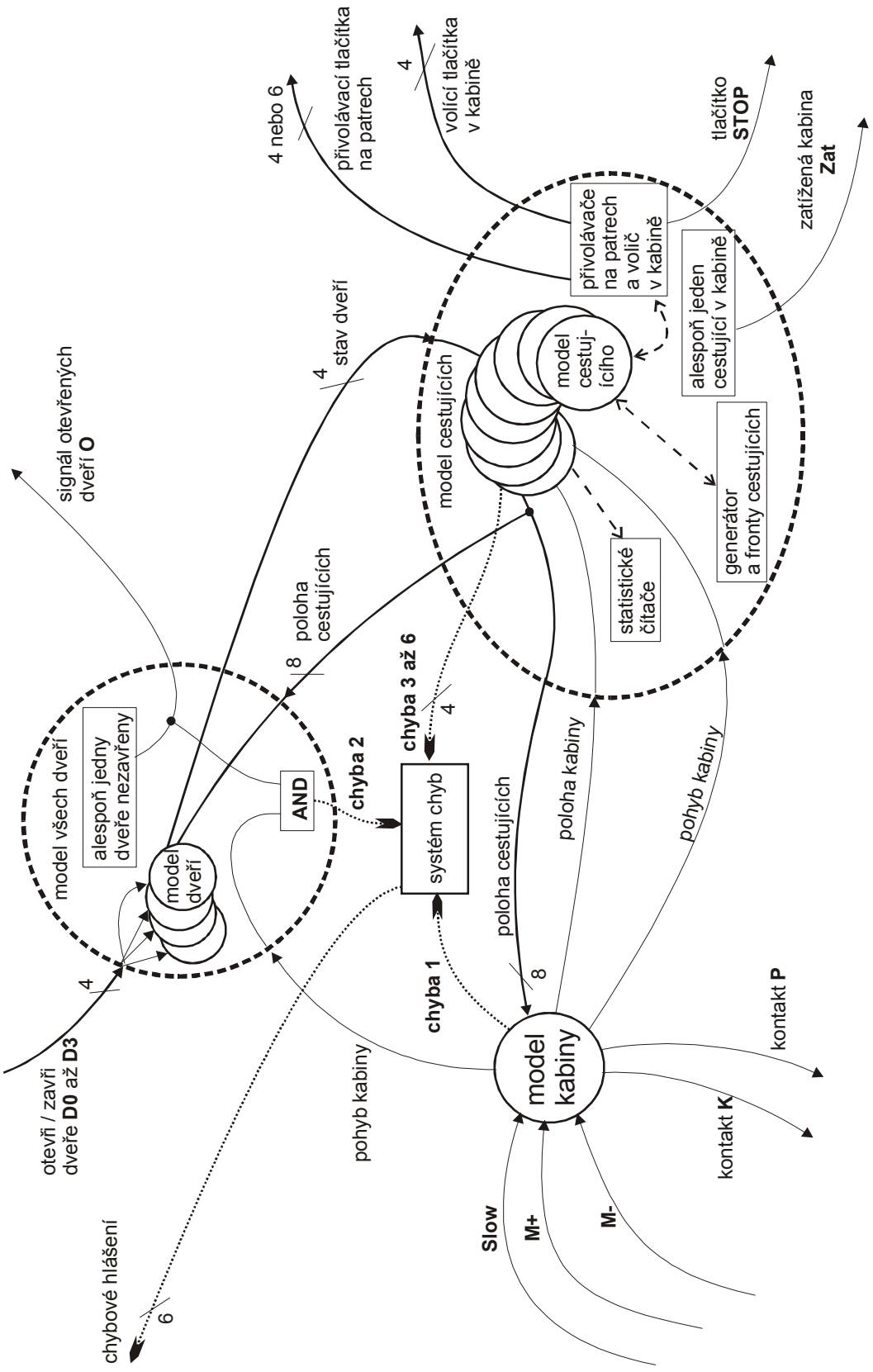
- Časovaný přechod. Pokud je hodnota vnitřního čítače stavu 2 větší než *interval*, tak automat přejde do stavu 3.



- Stejné jako předchozí s tím rozdílem, že při přechodu automatu do stavu 3 je vygenerovaná navíc chybová zpráva 3.

2.4 Model výtahu

Na obrázku 2.1 je znázorněn *model výtahu* jako část systému, která má osm vstupních a šestnáct výstupních logických signálů, a navíc generuje *chybové zprávy*. Popis vzniku všech *chybových zpráv* je v kapitole 2.4.4. *Model výtahu* lze rozkreslit podrobněji, obrázek 2.3.



Obrázek 2.3. Model výtahu, komunikace mezi jednotlivými částmi

Model výtahu se skládá ze čtyř části: modelu kabiny, modelu všech dveří, modelu cestujících a systému chyb. Tyto části mezi sebou komunikují pomocí naznačených vazeb. Logické vstupy a výstupy celého modelu jsou zobrazeny jako signály, které jsou na jednom konci volné. Šipka určuje směr signálu. Pokud je vazba zobrazena silněji a je přeškrtnutá čárou, u které je číslo, znamená to, že daná vazba obsahuje uvedený počet signálů. U výstupu přivolávacích tlačítek na patrech závisí počet signálů na režimu přivolávacích tlačítek, ve kterém je model nakonfigurován.

Celý *model výtahu* je vytvořen jako systém řízený událostmi složený z více automatů. Jednotlivé automaty se nacházejí vždy v jednom ze stavů a přecházejí do jiných stavů podle podmínek ve stejných časových intervalech. Celková rychlost simulace je pak závislá na počtu těchto časových intervalů za sekundu. Tuto rychlost lze nastavovat jako parametr modelu. Jako diskrétní systém je *model výtahu* vytvořen záměrně z důvodu přímočaré implementace programového kódu, kapitola 3.

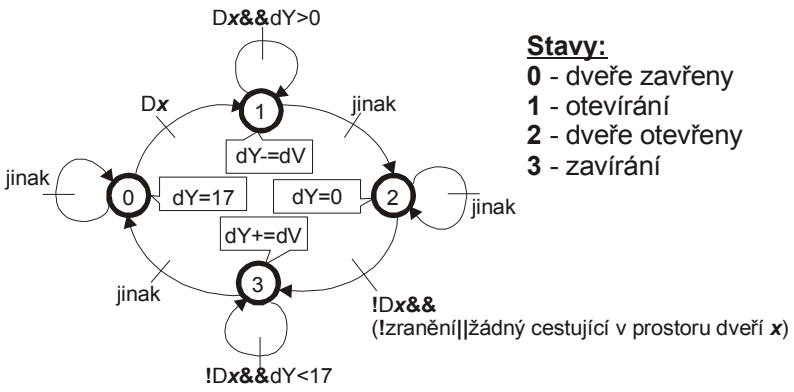
Postupně specifikuji funkce jednotlivých části celého *modelu výtahu*. Popis použitého značení v logických schématech modelů je uveden v kapitole 2.3.

2.4.1 Model všech dveří

Model všech dveří se skládá ze čtyř samostatných *modelů dveří*, které dávají logický výstup *otevřených dveří*. Chování jednoho tohoto modelu je zobrazeno na obrázku 2.4.

Dveře na každém patře jsou reprezentovány samostatným automatem, který bude popsán dále. Vstupem jsou signály pro ovládání dveří *D0* až *D3*, *pohyb kabiny* potřebný pro generování *chyby 2*, *poloha cestujících* pro případné pozdržení zavírání dveří v případě, že se některý cestující nachází v dveřním prostoru. Výstupem je signál *otevřených dveří* a *stavy dveří* potřebné pro model cestujících, kapitola 2.4.3. Tato část generuje *chybu 2*, způsobenou pohybem kabiny s otevřenými dveřmi.

Signál otevřených dveří *O* je aktivní, pokud nejsou všechny dveře úplně zavřeny, tedy nenacházejí-li se všechny jejich automaty ve stavu 0. Jinak je tento signál neaktivní. Pokud je současně aktivní *signál O* a *pohyb kabiny*, tak je vygenerovaná chyba 2.



Obrázek 2.4. Automat simulující chování jedněch dveří

Každý automat simulující chování jedněch dveří, obrázek 2.4, má kromě proměnné *stav* ještě svou proměnnou *dY*, která určuje na kolik jsou dané dveře zavřeny. *dY* je rovno nule pro otevřené dveře a rovno sedmnácti pro úplně zavřené. Rychlost otevírání a zavírání je určená *dV*, čím větší číslo *dV* tím je rychlost větší. *Dx* je vstupní signál *D0* až *D3*, podle toho o který model dveří se jedná.

- **Stav 0** – dveře zavřeny. Dokud nepřejde signál *Dx* do aktivního stavu, zůstávají dveře zavřené a *dY=17*. Aktivováním *Dx* přejde model do stavu 1.
- **Stav 1** – otevírání. V tomto stavu zůstane automat do doby, než se dveře úplně otevřou, tedy *dY<=0*, nebo dostanou signál pro uzavření.
- **Stav 2** – dveře otevřeny. V tomto stavu zůstane automat do doby, než se aktivuje signál *Dx*. Při zakázaném zranění cestujících nesmí být v prostoru dveří žádný cestující, aby mohl automat přejít do stavu 3. V případě, že tam nějaký cestující je, zůstane automat ve stavu 2.
- **Stav 3** – zavírání. V tomto stavu zůstane automat do doby, než se dveře úplně zavřou, tedy *dY>=17*, nebo dostanou signál pro otevření.

Po inicializaci modelu jsou všechny dveře uzavřeny, tedy jejich automaty jsou ve stavu 0. Podmínka výskytu nějakého cestujícího v prostoru dveří se zjišťuje z polohy všech cestujících, která se přenáší pomocí vazby mezi *modelem cestujících* (kapitola 2.4.3) a *modelem všech dveří*, zobrazeno na obrázku 2.3.

2.4.2 Model kabiny

Model kabiny implementuje dvě rychlosti pohonu kabiny, pomalejší a rychlejší. Kabina má svou dynamiku, takže změna rychlosti kabiny není skoková, ale mění se podle zrychlení resp. zpomalení. Tato část modelu má vstupy *M+*, *M-*, *Slow* a *poloha cestujících*; výstupy *K*, *P*, *poloha kabiny* a *pohyb kabiny*. Navíc model generuje *chybu 1*, způsobenou nárazem kabiny na konec výtahové šachty.

- **M+**, **M-** a **Slow** – signály slouží pro řízení pohonu kabiny. Význam signálů pro pohon kabiny je popsán v tabulce 2.1. Kolizní situace nastane při současné aktivaci signálů *M+* a *M-*. Tato situace má přesně definované chování pohonu. Pokud je jeden signál aktivován dříve než ten druhý, tak je brán v úvahu pouze signál, který byl aktivován dříve. Po deaktivaci prvního signálu, se pohon chová podle druhého signálu. Pokud jsou oba signály aktivovány současně a nejde rozlišit, který byl dříve, tak je preferován signál *M-*.
- **K** a **P** – signály pro určování polohy kabiny. Způsob generování je patrný z obrázku 2.2. Silná čára v horní poloze znamená, že je daný signál aktivní. Délka těchto kontaktu v modelu se dá nastavit. Signál *P* je generován pokud je kabina v úrovni přízemí.
- **Pohyb kabiny** – pokud je signál aktivní, je kabina v pohybu (automat je v jiném stavu než 0), jinak stojí (je ve stavu 0). Tento signál je potřebný pro model dveří a model cestujících.
- **Poloha kabiny** – představuje vnitřní proměnnou modelu *dY*, která vyjadřuje aktuální polohu kabiny.
- **Poloha cestujících** – poloha všech cestujících z *modelu cestujících*. Pokud není povoleno zranění cestujících a pohon kabiny dostane povel k rozjezdu v době, kdy se nachází nějaká osoba v prostoru vstupu do kabiny, bude kabina pozdržena proti rozjezdu, než se vstup uvolní.

Model nabízí možnost vypnutí dynamiky motoru pro rozjezd a zastavení z pomalejší rychlosti. Režim dynamiky pro pomalou rychlost určuje nastavení *dynamika*, pro zapnutou dynamiku odpovídá nastavení logické jedničce, jinak nule. Pokud má model vypnutou dynamiku pro pomalou rychlost a dostane povel k zastavení kabiny před úrovní podlaží v době generování signálu *K*, kabina dojede až přesně na úroveň podlaží.

Chování modelu kabiny je zobrazeno na obrázku 2.5 a 2.6. Protože *automat modelu kabiny* má hodně přechodů mezi stavy, je rozdělen do dvou obrázků. První obrázek obsahuje všechny přechody vycházející ze stavů 0, 1, 2, 3 a 8 a druhý obrázek všechny přechody vycházející ze stavů 0, 4, 5, 6 a 7. Stavy 1, 2, 3 a 8 jsou stavy pro pohyb kabiny nahoru a stavy 4, 5, 6 a 7 pro pohyb dolů.

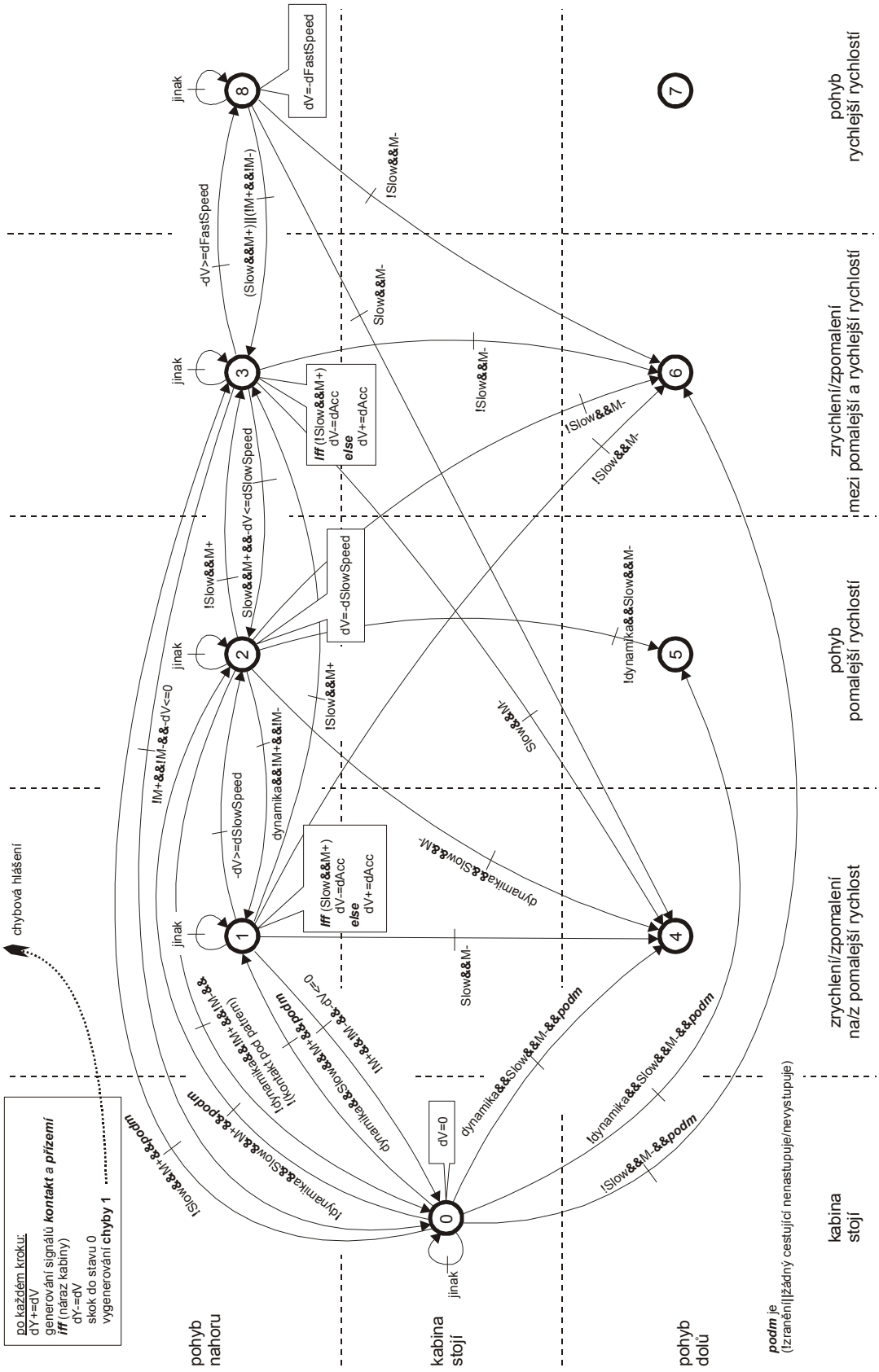
Automat má kromě proměnné *stav* ještě vnitřní proměnnou *dY* (aktuální poloha kabiny) a *dV* (aktuální rychlost pohybu kabiny). *dY* je nulové v nejvyšší poloze, tedy nad třetím patrem, a směrem k přízemí *dY* roste. Záporná rychlost *dV* znamená, že se kabina pohybuje nahoru, a naopak. V každém kroku je k *dY* přičtena hodnota *dV* a podle aktuální polohy kabiny jsou vygenerovány signály *K* a *P*. Poté se provede kontrola, zda nevyjela kabina z povoleného rozsahu. Pokud ano, je okamžitě automat nastaven do stavu 0 a je vygenerovaná *chyba 1*, náraz kabiny.

Nastavovací parametry pro chování modelu kabiny jsou:

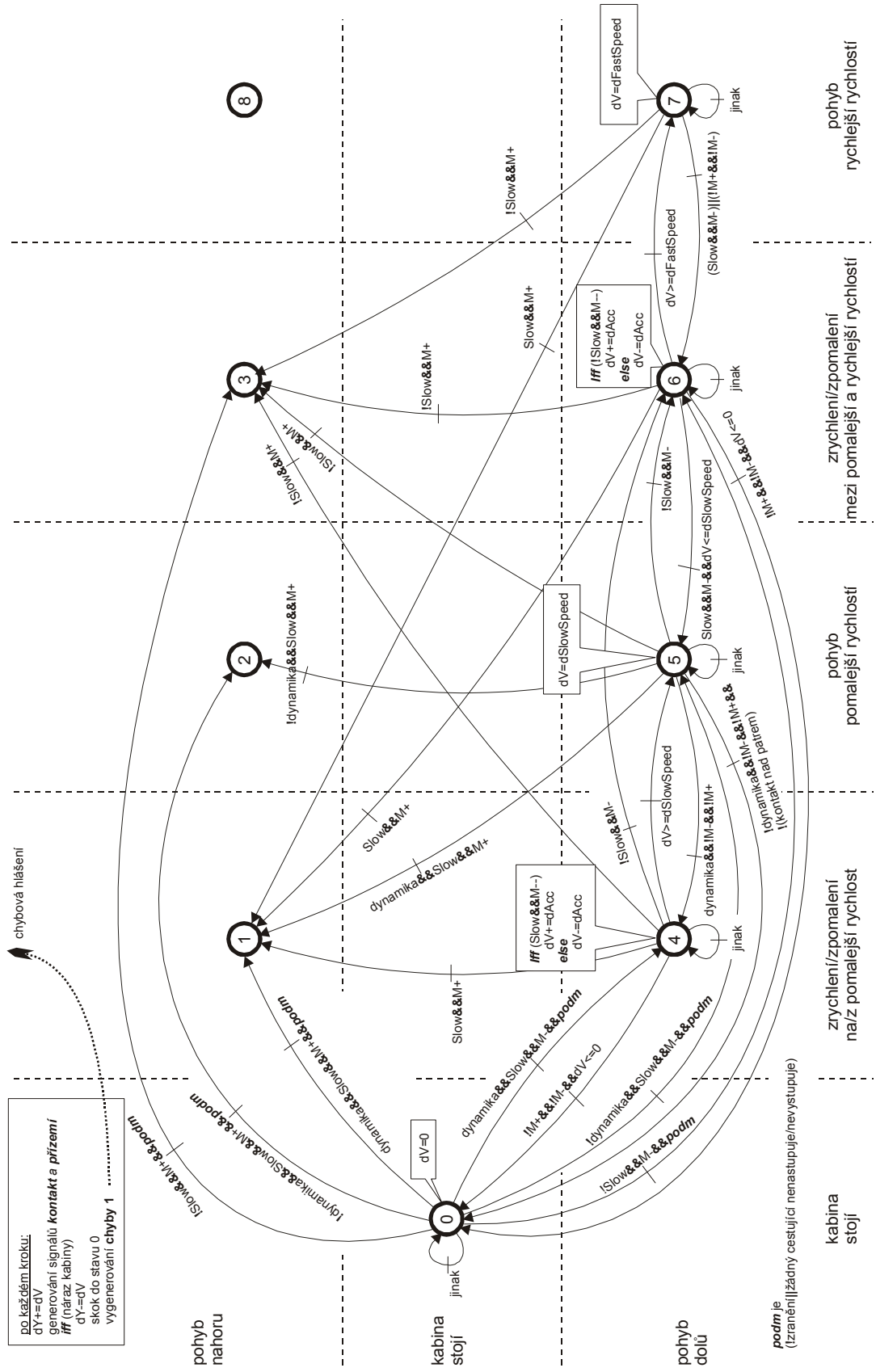
- **dAcc** – zrychlení a zpomalení kabiny při změně rychlosti. Pokud je *dynamika* vypnuta, není toto zrychlení uplatňováno pro změnu z nulové rychlosti na pomalejší a naopak.
- **dSlowSpeed** – pomalejší rychlost pohonu kabiny.
- **dFastSpeed** – rychlejší rychlost pohonu kabiny.
- **Poloha kabiny po inicializaci** – *dY* se bude rovnat úrovni podlaží, ve kterém se má model kabiny nacházet po inicializaci.

Jednotlivé stavy automatu modelu kabiny jsou:

- **Stav 0** – kabina stojí. Pokud není povoleno zranění cestujících a pohon kabiny dostane povel k rozjezdu v době, kdy se nachází nějaká osoba v prostoru vstupu do kabiny, bude kabina pozdržena proti rozjezdu, než se vstup uvolní. Přítomnost nějaké osoby v prostoru vstupu se zjišťuje z polohy všech automatů simulujících chování cestujících.
- **Stav 1** – stav pro zrychlovaní na pomalejší rychlost nahoru a brždění z pohybu nahoru. Do tohoto stavu se automat může dostat v případě, pokud je povolená dynamika pro pomalou rychlost a při změně na pomalý pohyb nahoru z rychlého pohybu dolů, kdy se vždy uplatňuje dynamika pohonu. V tomto stavu se akceleruje nahoru (*dV-=dAcc*) nebo brzdí z pohybu nahoru (*dV+=dAcc*). Pokud rychlost bude zápornější než *–dSlowSpeed*, tak automat přejde do stavu 2, a pokud bude rovna nebo větší než nula, kabina zastaví (stav 0). Z tohoto stavu může automat přejít do jiného stavu také změnou požadavku, například na pohyb opačným směrem anebo na rychlý pohyb nahoru.
- **Stav 2** – pohyb kabiny pomalu nahoru. V tomto stavu je rychlost konstantní a je rovna rychlosti *-dSlowSpeed*. Z tohoto stavu automat přejde, pokud kabina narazí nebo dojde-li ke změně požadavku řízení pohonu.
- **Stav 3** – zrychlování na rychlejší rychlost nahoru a brždění na pomalejší rychlost z pohybu nahoru. V každém kroku zde dochází k akceleraci nahoru (*dV-=dAcc*) nebo brždění z pohybu nahoru (*dV+=dAcc*). Pokud dojde ke změně požadavku řízení pohonu na směr dolů, přejde automat do stavu 4 nebo 6. Dosáhne-li model rychlejší rychlosti nahoru, přejde do stavu 8. Má-li zpomalit na pomalejší rychlost nahoru a rychlost už klesne na hodnotu pomalejší, tak přejde zpět do stavu 2. Při zastavování, dosáhne-li nuly, přejde do stavu 0.



Obrázek 2.5. Automat modelu kabiny, pouze přechody vycházející ze stavů 0, 1, 2, 3 a 8



Obrázek 2.6. Automat modelu kabiny, pouze přechody vycházející ze stavů 0, 4, 5, 6 a 7

- **Stav 4** – stav pro zrychlovaní na pomalejší rychlost dolů a brždění z pohybu dolů. Do tohoto stavu se může automat dostat pouze, pokud je povolená dynamika pro pomalou rychlost a při změně na pomalý pohyb dolů z rychlého pohybu nahoru, kdy se vždy uplatňuje dynamika pohonu. V tomto stavu se akceleruje dolů (*dV+=dAcc*) nebo brzdí z pohybu dolů (*dV-=dAcc*). Pokud rychlost bude větší než *dSlowSpeed*, tak automat přejde do stavu 5, a pokud bude rovna nebo menší než nula, kabina zastaví (stav 0). Z tohoto stavu může automat odejít také změnou požadavku, například na pohyb opačným směrem anebo na rychlý pohyb nahoru.
- **Stav 5** – pohyb kabiny pomalu dolů. V tomto stavu je rychlost konstantní a je rovna rychlosti *dSlowSpeed*. Z tohoto stavu automat přejde do jiného stavu, pokud kabina narazí nebo dojde-li ke změně požadavku řízení pohonu.
- **Stav 6** – zrychlování na rychlejší rychlost dolů a brždění na pomalejší rychlost z pohybu dolů. V každém kroku zde dochází k akceleraci dolů (*dV+=dAcc*) nebo brždění z pohybu dolů (*dV-=dAcc*). Pokud dojde ke změně požadavku řízení pohonu na směr nahoru, přejde automat do stavu 1 nebo 3. Dosáhne-li model rychlejší rychlosti nahoru, přejde automat do stavu 7. Má-li zpomalit na pomalejší rychlost dolů a rychlost už klesne na hodnotu pomalejší, tak přejde zpět do stavu 5. Při zastavování, dosáhne-li nuly, přejde do stavu 0.
- **Stav 7** – pohyb kabiny rychle dolů. V tomto stavu je rychlost konstantní a je rovna rychlosti *dFastSpeed*. Požadavkem zpomalení na pomalejší rychlost dolů nebo na zastavení, přejde automat do stavu 6. Požadavkem na změnu směru pohybu kabiny, přejde automat do stavu 1 nebo 3.
- **Stav 8** – pohyb kabiny rychle nahoru. V tomto stavu je rychlost konstantní a je rovna rychlosti *-dFastSpeed*. Požadavkem zpomalení na pomalejší rychlost nahoru nebo na zastavení, přejde automat do stavu 3. Požadavkem na změnu směru pohybu kabiny, přejde automat do stavu 4 nebo 6.

Po inicializaci modelu je *automat modelu kabiny* ve stavu 0 a *dY* odpovídá úrovní patra, ve kterém se má nacházet kabina výtahu po inicializaci podle nastavení.

2.4.3 Model cestujících

Model cestujících se skládá z více samostatných automatů simulujících chování jednoho cestujícího a ze společných části: statistického čítače, generátoru a front cestujících, přivolávačů na patrech a voliče v kabině, obrázek 2.3. V modelu výtahu se může podle nastavení pohybovat *aktivně* až nastavený maximální počet pasažérů, kde každý je reprezentovaný svým vlastním samostatným automatem. Další cestující mohou čekat ve frontě na příslušném podlaží a jsou vypuštěni, až čas čekání klesne na nulu a uvolní se některý model cestujícího (jeho automat bude ve stavu 0). *Aktivním pohybem* se rozumí pohyb osob v modelu, při kterém přivolávají kabinu, nastupují, jsou přepravováni a vystupují.

Vstupem této části je *stav dveří*, *poloha kabiny* potřebné pro detekci zranění cestujícího, a *pohyb kabiny* pro povolení nástupu a výstupu z kabiny. Výstupem je signál zatížené kabiny *Zat*, je aktivní, když je v kabině alespoň jeden cestující. Dále signály všech přivolávacích tlačítek na patrech, volících tlačítek a tlačítka *STOP* v kabině. Pokud jsou tyto signály aktivní, znamená to, že tlačítko je stištěno. Také jsou přenášeny polohy všech *aktivních cestujících* do *modelu kabiny* (kapitola 2.4.2) a *modelu všech dveří* (kapitola 2.4.1). Navíc model generuje čtyři typy chybových zpráv:

- **Chyba 3** – cestující se nedočkal. Tato chyba je vygenerována v případě povolené nervozity cestujících, když po přivolání výtahu nepřijede kabina a cestující nemůže

nastoupit do určitého časového limitu podle nastavení, anebo když se cestující po volbě cílového patra v kabině výtahu do tohoto limitu nedostane do svého cílového podlaží.

- **Chyba 4** – cestující rozdrcen dveřmi od výtahu. Generována jen pokud je povoleno zranění osob.
- **Chyba 5** – rozdrcení cestujícího mezi kabinou výtahu a patrem. Jen pokud je povoleno zranění osob.
- **Chyba 6** – generována v případě, že pohon kabiny nedostane povel k zastavení do ukončení stisku tlačítka *STOP* v kabině.

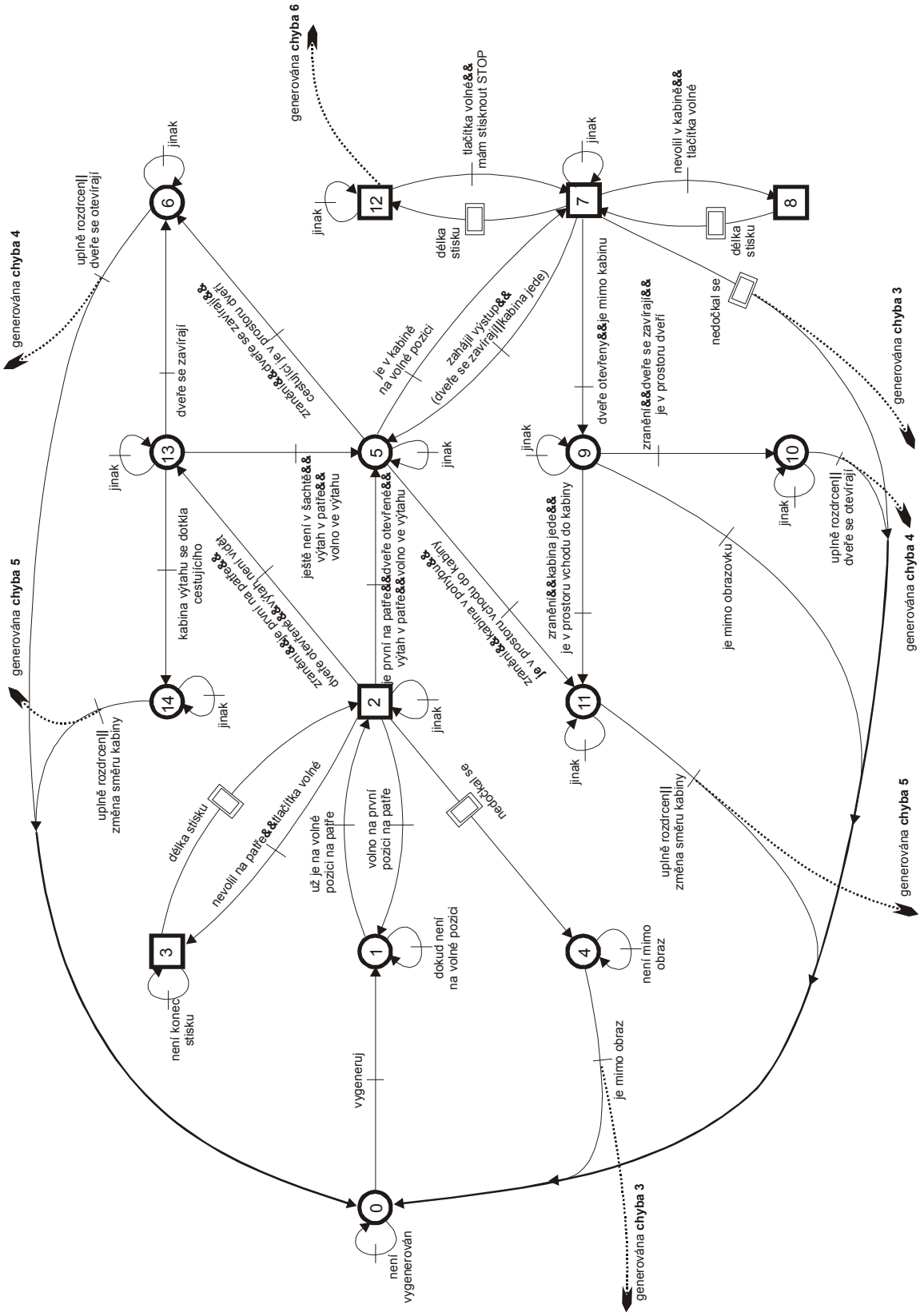
Parametry ovlivňující chování modelu cestujících jsou:

- **Rychlost pohybu cestujícího** – ovlivňuje dobu nástupu a výstupu cestujícího z kabiny, příchod a odchod. Veškerý pohyb cestujícího je prováděn s touto rychlostí.
- **Kapacita kabiny** – počet osob, které mohou být současně přepravovány v kabině výtahu.
- **Povolení tlačítka STOP** – zda je možno používat v kabině tlačítko *STOP*.
- **Přesnost nastavení kabiny pro nástup a výstup** – určuje zda musí kabina výtahu stát přesně v úrovni patra, aby cestující nastupovali a vystupovali. Je-li povolená nepřesnost pro nástup a výstup, cestující nastupují a vystupují v případě, že je aktivní signál *K* (kontakt pro určování polohy kabiny pro řídící systém, kapitola 2.4.2).
- **Limit pro odchod cestujícího** – doba, po které cestující odejde, pokud se po přivolání výtahu nedočká příjezdu kabiny nebo pokud se po volbě v kabině nedočká svého cílového patra.
- **Počet přivolávacích tlačítek** – zda je na každém podlaží pouze jedno tlačítko nebo dva, jedno pro směr nahoru a jedno dolů.
- **Délka stisku tlačítka** – doba, po kterou cestující drží stisknuté nějaké tlačítko.
- **Pauza mezi stisky** – mezera mezi stisky různých cestujících na jedné *skupině tlačítek*. *Skupinou tlačítek* se rozumí všechny tlačítka v kabině. Další skupiny tlačítek jsou na jednotlivých podlažích.
- **Parametry nastavující generátor cestujících** – budou popsány později v kapitole 2.4.3.2.

2.4.3.1 Model cestujícího

Každý *aktivní cestující* v modelu je reprezentován svým vlastním automatem cestujícího zobrazeným na obrázku 2.7. Každý automat má kromě proměnné *stav* ještě další vnitřní proměnné:

- **Poloha** – místo kde se cestující zrovna nachází.
- **Odkud** a **kam cestuje** – tyto proměnné jsou nastaveny ještě předtím, než je cestující vygenerován, tedy jeho automat přejde ze stavu 0 do stavu 1.
- **Přivolával** – zda už přivolával výtah na patře.
- **Volil cílové patro** – zda už volil v kabině cílové patro. Po stisku tlačítka *STOP*, je tato proměnná vynulována a volba cílového patra je provedená znovu.



Obrázek 2.7. Automat simulující chování jednoho cestujícího

- **Čas cestování** – počítadlo, které se každým krokem zvyšuje o jedničku od doby stisku přivolávacího tlačítka kabiny na patře. Používá se pro generování statistických údajů.
- **Zda má stisknout tlačítko STOP** – je nastaveno už při vypuštění cestujícího. Když je cestující v kabině a je někde mezi úrovněmi podlaží, tak stiskne tlačítko *STOP* a zvolí si nové cílové podlaží.

Na každém patře mohou čekat na příjezd kabiny výtahu dva *aktivní cestující*. Další jsou *pasivní* v čekací frontě a nemají přiřazený automat cestujícího. Aby každý cestující věděl, na které pozici se má zastavit, jsou pro každé patro v modelu dvě proměnné společné pro všechny automaty cestujících, odpovídající dvěma pozicím pro čekání na výtah. Každá proměnná odkazuje na žádný nebo jeden *model cestujícího*, který na ni stojí nebo na ni míří. Podobně jsou dvě proměnné pro kabinu výtahu, kde mohou cestovat spolu maximálně dvě osoby.

Každý automat cestujícího také předává různé statistické údaje do části *statistického čítače*, kapitola 2.4.3.4. Jednotlivé stavy automatu cestujícího zobrazeného na obrázku 2.7 jsou:

- **Stav 0** – cestující je neaktivní a čeká, až bude vygenerován *generátorem cestujících*. Při tom mu budou nastaveny jeho parametry *odkud* a *kam cestuje*, příslušně nastavena jeho *poloha*, vynulován příznak *přivolávání* a *volby patra*, nastaveno zda má při své cestě stisknout tlačítko *STOP*. Aktivace cestujícího je provedena přechodem automatu do stavu 1.
- **Stav 1** – cestující se pohybuje na svém příchozím podlaží *rychlostí pohybu* směrem ke dveřím výtahu, až do doby než je na volné pozici pro čekání.
- **Stav 2** – čekání na podlaží. Z tohoto stavu může automat přejít v případě, že:
  - cestující čeká na druhé pozici na patře a první pozice se uvolní,
  - cestující ještě nepřivolával výtah a tlačítka na jeho podlaží jsou volná,
  - otevřou-li se dveře a stojí-li v jeho patře kabina a je v ní volno,
  - je-li povoleno zranění cestujícího v nastavení a dveře jsou otevřeny a kabina výtahu není vidět.

  Tento stav má svůj vnitřní čítač sloužící pro měření doby čekání. Příchodem do tohoto stavu se čítač nuluje a každým krokem setrvávajícím v tomto stavu se zvyšuje o jedničku, pokud je povolena *nervozita cestujícího*.
- **Stav 3** – v tomto stavu cestující drží stištěné tlačítko přivolávání výtahu na svém podlaží, dokud není vnitřní čítač větší, než *doba stisku tlačítka*. Pak je automat vrácen zpět do stavu 2.
- **Stav 4** – cestující odchází pryč, protože se nedočkal výtahu po vypršení *limitu čekání*. Odchází *rychlostí pohybu*, dokud není mimo model. Pak automat přejde do stavu 0 a tato informace je přidána do *statistických čítačů*.
- **Stav 5** – cestující jde do výtahu, dokud není na volné pozici v kabině. Příchodem do tohoto stavu je zaznamenána do statistiky *doba čekání na vstup* do kabiny výtahu dosud ušlá doba cestování, která se počítá od stisku přivolávacího tlačítka do opuštění cílového patra. Z tohoto stavu také může automat přejít, pokud je povoleno *zranění cestujícího* a nachází-li se současně cestující v prostoru dveří, které se zavírají. Také v případě, že se cestující nachází v prostoru vstupu do výtahu a výtah se rozjel.

- **Stav 6** – drcení cestujícího zavírajícími se dveřmi výtahu při nástupu. Až je úplně rozdrcen nebo až se dveře začnou zase otevírat, je vygenerována *chyba 4* a automat přejde do stavu 0.
- **Stav 7** – cestující čeká ve výtahu na cílové patro. Když kabina stojí ve jeho cílovém podlaží a dveře jsou otevřeny, zahájí cestující výstup z kabiny (automat ještě setrvává v tomto stavu do doby, než opustí definitivně kabinu výtahu). V tomto stavu se cestující pohybuje nahoru a dolů s pohybem kabiny, který je přenášen z *modelu kabiny*. Tento stav má svůj vnitřní čítač pro měření doby čekání. Vyprší-li doba čekání přejde automat do stavu 0 a tato informace je přidána do *statistických čítačů*.
- **Stav 8** – v tomto stavu cestující drží stištěné tlačítko volby cílového patra, dokud neuplyne *doba stisku tlačítka*. Pak je automat vrácen do stavu 7.
- **Stav 9** – vystupování cestujícího. Do tohoto stavu se automat dostane, až vystupující cestující opustí kabinu výtahu. Až odejde mimo model, je započtena do statistiky jeho celková doba přepravy a automat přejde do stavu 0.
- **Stav 10** – drcení cestujícího zavírajícími se dveřmi výtahu při výstupu. Až je cestující úplně rozdrcen nebo až se dveře začnou zase otevírat, je vygenerována *chyba 4* a automat přejde do stavu 0.
- **Stav 11** – drcení cestujícího pohybující se kabinou výtahu. Do tohoto stavu se automat dostane, pokud při nástupu nebo výstupu cestujícího se kabina výtahu rozjede. Automat zde setrvá, dokud není cestující úplně rozdrcen anebo kabina nezmění svůj směr pohybu. Pak je vygenerována *chyba 5* a automat přejde do stavu 0.
- **Stav 12** – cestující drží stisknuté tlačítko *STOP* v kabině výtahu. Do tohoto stavu se automat dostane, pokud jsou volné tlačítka voleb v kabině a cestující má stisknout zrovna tlačítko *STOP*. To je určeno, už při jeho vygenerování, kapitola 2.4.3.2, a *polohou kabiny*. Tlačítko *STOP* může být stištěno jen v případě, že se kabina nachází mezi úrovněmi podlaží. Až vyprší doba *stisku tlačítka*, je cestujícímu vygenerováno nové cílové patro a všem *modelům cestujícího*, které jsou zrovna v kabině, je vynulován příznak *volby cílového patra*. To zajistí, že po stištění tlačítka *STOP*, provedou všechny osoby v kabině novou volbu svého cíle. Navíc, pokud do uvolnění tlačítka *STOP* nedostane pohon kabiny povel k zastavení, je vygenerována *chyba 6*.
- **Stav 13** – pohyb cestujícího do výtahové šachty. Cestující se pohybuje směrem do výtahové šachty a zastaví tak, aby kousek vyčníval do prostoru, kde jezdí kabina výtahu. Do tohoto stavu se automat dostane, protože je povoleno *zranění cestujících*, dveře byly otevřeny a kabina tam nebyla. Z tohoto stavu se automat ještě může vrátit do stavu 5, pokud přijede volná kabina.
- **Stav 14** – drcení cestujícího přijíždějící kabinou výtahu. Automat zde setrvá, dokud není cestující úplně rozdrcen nebo kabina nezmění svůj směr pohybu. Pak je vygenerovaná *chyba 5* a automat přejde do stavu 0.

Po inicializaci modelu se všichni cestující nacházejí ve stavu 0 a čekají až *budou generátorem cestujících* vypuštěni. Popis tohoto *generátoru* je v následující kapitole.

2.4.3.2 Generátor a fronty cestujících

Tato část modelu cestujících uvádí *modely cestujících* do *aktivního pohybu*, tedy nastaví jejich parametry *odkud* a *kam se přepravují*, zda mají stisknout tlačítko *STOP* a převedou

jejich automat do stavu 1. Navíc jim nastaví polohu cestujícího, aby odpovídala nástupnímu patru, vynuluje příznak přivolávání a volby cílového patra.

Každý cestující v modelu výtahu cestuje typicky podle jednoho nebo podle kombinace vícero scénářů:

- cestování z přízemí do některého z vyššího patra,
- cestování z některého vyššího patro zpět do přízemí,
- cestování mezi různými vyššími patry.

Parametry nastavení ovlivňující chování *generátoru cestujících* jsou:

- **Maximální počet aktivních cestujících** – určuje kolik *automatů cestujících* může být současně v jiném stavu než ve stavu 0. Pokud nemůže být cestující aktivován, čeká ve frontě, dokud není tato podmínka splněná. Pak je vygenerován.
- **Průměrná četnost příchodů nových cestujících** – nastavení ovlivňující rychlost příchodů nových pasažérů v přízemí. Nastavuje Poissonův generátor intervalů.
- **Povolení cestování zpět do přízemí** – tato volba určuje, zda cestující, který bude přepraven do svého cílového podlaží podle prvního scénáře, bude cestovat zpět do přízemí nebo nikoliv.
- **Průměrná četnost návratů osob zpět** – pokud je povoleno *cestování mezi patry* nebo *zpět do přízemí*, tak toto nastavení ovlivňuje dobu, kdy se cestující vrátí a použije výtah znovu poté, co byl přepravený na některé horní podlaží. Nastavuje Poissonův generátor intervalů.
- **Povolení cestování mezi patry** – tato volba povoluje, po přepravení cestujícího do některého vyššího patra, příchod cestujícího zpět po náhodném intervalu a cestování do jiného patra než do přízemí.
- **Pravděpodobnost cestování mezi patry** – určuje, s jakou pravděpodobností bude cestující, který byl přepraven do vyššího patra, poté cestovat na jiné patro než do přízemí, pokud je to povoleno předchozí volbou.
- **Pravděpodobnost stištění tlačítka STOP** – určuje, s jakou pravděpodobností každý přepravovaný cestující stiskne tlačítko *STOP* v kabině a pak znovu zvolí jiné své cílové podlaží.
- **Vygenerování cestujícího, pokud není v modelu žádný jiný aktivní cestující** – pokud je tato volba povolena, je ihned vygenerován nový cestující v okamžiku, když jsou všechny *automaty cestujících* ve stavu 0.

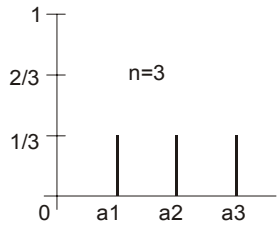
Pokud není povoleno *cestování mezi patry* a *cestování zpět do přízemí*, osoby cestují pouze směrem z přízemí do vyšších pater. Jinak se mohou vracet do přízemí, případně cestovat do jiného podlaží.

Generátor příchodu nového cestujícího v přízemí je reprezentován čítačem, který odpovídá intervalu, po kterém bude vygenerován nový cestující. Po příchodu je naplněn hodnotou odpovídající náhodnému času z generátoru intervalů s Poissonovým rozdělením. Každým krokem automatu je tento čítač snížen o jedničku. Po dosažení nuly, pokud už v přízemí dva *aktivní cestující* nečekají a není-li překročen počet možných aktivních cestujících z nastavení, je v přízemí *vypuštěn cestující*. Jinak je, po dosažení nuly čítačem, do fronty *pasivních cestujících* v přízemí zařazen nulový interval příchodu nového cestujícího. To zajistí, že bude v přízemí *vypuštěn cestující* hned, jak to bude možné. *Vypuštění cestujícího* spočívá v nalezení některého *automatu cestujícího*, který je ve stavu 0; vygenerování mu cílového podlaží náhodně s rovnoměrným rozdělením, nastavení vnitřních proměnných nalezeného modelu a uvedení ho do stavu 1.

Pokud některý cestující opustí své cílové patro mimo přízemí, tedy jeho automat přejde ze stavu 9 do stavu 0, je náhodný interval, za který by se měl daný cestující vrátit zpět, z Poissonova generátoru (pro příchody osob zpět na patře) zapsán do *fronty pasivních cestujících* na daném patře.

Každé patro má svůj vlastní seznam intervalů reprezentovaných čítači, za které by měl být na tomto patře vygenerován další cestující. Tato fronta může být i prázdná. V každém kroku je ve všech seznamech snížen každý čítač o jedničku, dokud není nulový. Není-li ještě *počet aktivních cestujících* v modelu roven nastavenému maximu a je na daném patře volné místo, je tento nulový čítač z příslušného seznamu odstraněn a může být vygenerován cestující v tomto patře. Jde-li o přízemí, je cestující vygenerován s cílovým patrem vybraným náhodně s rovnoměrným rozdělením. Jde-li o vyšší podlaží a je povoleno cestování mezi podlažími, je cestující s pravděpodobností z nastavení vypuštěn do jiného vyššího podlaží vybraného náhodně s rovnoměrným rozdělením. Zbytek cestujících je vypuštěn zpět do přízemí, pokud je to povoleno.

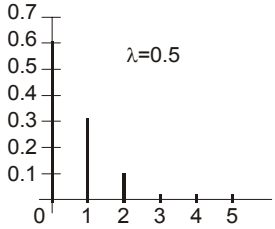
Diskrétní rovnoměrné rozdělení má pravděpodobnost výskytu každého možného stavu stejnou, obrázek 2.8. Toto rozdělení je popsáno vzorci (2.1). Na obrázku je vidět rovnoměrné rozdělení pro *n=3*, tedy například pro volbu cílového podlaží cestujícího generovaného v přízemí odpovídají jevy *a1* až *a3* postupně volbě *prvního* až *třetího* podlaží.



Obrázek 2.8. Diskrétní rovnoměrné rozdělení pro n=3

$$P[X = a_k] = \frac{1}{n}, \quad k = 1, 2, \ldots n.$$
$$\text{n - přirozené číslo, } a_1, a_2, \ldots, a_n \in R \tag{2.1}$$
$$EX = \frac{1}{n}\sum_{k=1}^{n} a_k$$

Pro generování intervalů příchodů cestujících je použito *Poissonovo rozdělení*, obrázek 2.9. Vyjadřuje, s jakou pravděpodobností budou přicházet cestující s danou četností. Parametr *Poissonova rozdělení* λ je roven průměrné hodnotě. *Model cestujících* používá dva tyto generátory s parametry podle nastavení. Jeden pro intervaly příchodů nových cestujících v přízemí a druhý pro intervaly návratů zpět k výtahu každého cestujícího dopraveného do horní podlaží.



Obrázek 2.9. Poissonovo rozdělení pro $\lambda=0.5$

$$P[X = k] = e^{-\lambda} \frac{\lambda^k}{k!}, \quad k = 0, 1, 2, \ldots$$
$$EX = \lambda \tag{2.2}$$

2.4.3.3 Přivolávače na patrech a volič v kabině

Přivolávače na patrech a *volič v kabině* slouží pro generování výstupních logických signálů přivolávacích tlačítek *TP0* až *TP3*, *TP1N*, *TP2N* a voliče v kabině *TK0* až *TK3* a *S* (signál tlačítka *STOP*). Model může být podle nastavení ve dvou režimech počtu přivolávacích tlačítek výtahu. Podle toho se pak používá buď čtyři nebo šest signálů od těchto přivolávačů. Použití signálů je popsáno v tabulce 2.2. Pokud je daný signál v úrovní logické jedničky, znamená to, že dané tlačítko je zrovna stištěno.

Všechny tlačítka v modelu jsou rozděleny na *skupiny*. *Skupinou tlačítek* jsou volící tlačítka v kabině výtahu i s tlačítkem *STOP*. Každé podlaží má také svou skupinu tlačítek. V jedné skupině nemůže být stištěno současně více tlačítek a další stisk v této skupině se může provést až po uplynutí intervalu, *pauzy mezi stisky*, která je volena v nastavení. Například: vejdou-li dva cestující současně do kabiny nejdříve volí první a po určité době druhý.

Každá přivolávací skupina tlačítek je v modelu reprezentovaná svým čítačem, který je v každém kroku modelu snížen o jedničku, pokud je větší než nula. Celkem jich je tedy pět. K těmto čítačům mají přístup všechny *automaty cestujících*. Za volnou skupinu tlačítek se považuje ta, které čítač je roven nule. Chce-li cestující přivolávat výtah, čeká nejdříve, dokud není příslušný čítač od daného podlaží roven nule. Pak jej nastaví na *délku stisku* a drží vybrané tlačítko. Klesne-li tento čítač na nulu, cestující pustí tlačítko a nastaví čítač na hodnotu *pauzy mezi stisky*. Tím je zajištěno, že žádný jiný cestující tlačítko nestiskne dříve, než po uplynutí této doby.

Tlačítko *STOP* v kabině používá stejný čítač jako tlačítka pro volbu cílového patra.

2.4.3.4 Statistické čítače

Model cestujících obsahuje část statistických čítačů, do kterých zapisují údaje všechny *modely cestujících*, obrázek 2.3. Tyto čítače slouží pro sledování kvality řídícího algoritmu. Sledované hodnoty jsou:

- **Celkový počet přepravených osob** – tento čítač je zvýšen o jedničku vždy, když cestující opustí na svém cílovém podlaží model, stav 9 v automatu jednoho cestujícího.
- **Počty přepravených osob na jednotlivé podlaží** – stejné jako předchozí s tím rozdílem, že je zvýšen o jedničku příslušný čítač odpovídající podlaží, kde cestující opustil model.
- **Průměrná kvadratická doba čekání** – tento údaj vypovídá o rychlosti plnění požadavků na přivolání. Vypočte se jako součet druhých mocnin *čekacích dob* všech přepravovaných cestujících vydělený jejich počtem. *Čekací dobou* se rozumí doba, od stisku přivolávacího tlačítka, do okamžiku než je mu umožněno nastoupit do kabiny výtahu. Tato doba se získává z počítadla *času cestování* daného cestujícího v okamžiku, kdy jeho automat přejde do stavu 5.

- **Průměrná přepravní doba** – údaj vypovídající o celkovém odbavování cestujících. Je vypočten jako součet *přepravních dob* všech přepravených osob vydělený jejich počtem. *Přepravní dobou* se rozumí doba, od stisku přivolávacího tlačítka, do okamžiku, než je cestující dopraven na své cílové patro a opustí kabinu výtahu. *Přepravní doba* se získá z *cestovního času* cestujícího v okamžiku přechodu jeho automatu ze stavu 9 do stavu 0.
- **Počet zraněných cestujících** – tento údaj je zvýšen o jedničku *automatem cestujícího*, vždy při opuštění stavů 6, 10, 11 nebo 14.
- **Počet cestujících, kteří se nedočkali** – údaj je zvýšen o jedničku, podobně jako předchozí, opuštěním stavu 4 nebo přechodem ze stavu 7 do stavu 0.

Průměrná kvadratická doba čekání a *průměrná přepravní doba* je ovlivněná nastavením rychlosti pohybu cestujícího, rychlosti pohybu kabiny a také jejím zrychlením a zpomalením. *Údaj o počtu zraněných cestujících* je ovlivněn volbou povolení *zranění*. Podobně *počet cestujících, kteří se nedočkali* je ovlivněn volbou povolení nervozity v modelu a dobou, která tuto nedočkavost určuje.

Inicializací modelu jsou všechny čítače vynulovány.

2.4.4 Systém chyb

Jednotlivé části *modelu výtahu* generují šest různých typů chybových zpráv, tabulka 2.3. Chybová zpráva *1* je generována *modelem kabiny* (kapitola 2.4.2), zpráva *2 modelem všech dveří* (kapitola 2.4.1) a zprávy *3* až *6 modelem cestujících* (kapitola 2.4.3). Všechny tyto chybové zprávy jsou zpracovávány *systémem chyb*, který je zobrazuje uživateli a ukládá do logovacího souboru hlášení pro další analýzu řídícího algoritmu.

Chybová zpráva	Příčina generování chybové zprávy
1	náraz kabiny výtahu na konec šachty
2	pohyb kabiny s otevřenými dveřmi
3	cestující se nedočkal
4	cestující byl rozdrcen zavírajícími se dveřmi
5	cestující byl rozdrcen kabinou výtahu
6	kabina nedostala povel k zastavení do ukončení stisku tlačítka *STOP*

Tabulka 2.3. Chybové zprávy generované modelem výtahu

Každý typ chybové zprávy může podle nastavení modelu navíc vygenerovat *kritickou chybu*. *Kritická chyba* zastaví další běh modelu až do doby, než je model znovu inicializován.

3 Implementace modelu výtahu

Model, navržený v kapitole 2, jsem vytvořil jako aplikaci pro operační systém Windows® v prostředí Borland C++ Builder™. Tato aplikace přehledně zobrazuje pomocí grafiky svůj aktuální stav, vstupní a výstupní signály, umožňuje nastavovat parametry chování modelu výtahu. Vstupní a výstupní signály modelu jsou přenášeny přes vstupně výstupní kartu. Pokud není vybrána žádná karta nebo není vůbec žádná v systému instalována, mohou být všechny vstupní signály generovány ručně pomocí ukazatele myši.

Projekt aplikace simulátoru výtahu se zdrojovými soubory pro *Borland C++ Builder*™ verze 5.0 a také zkompilovaná výsledná aplikace simulátoru je na přiloženém kompaktním disku, obsah disku je uveden v příloze II.

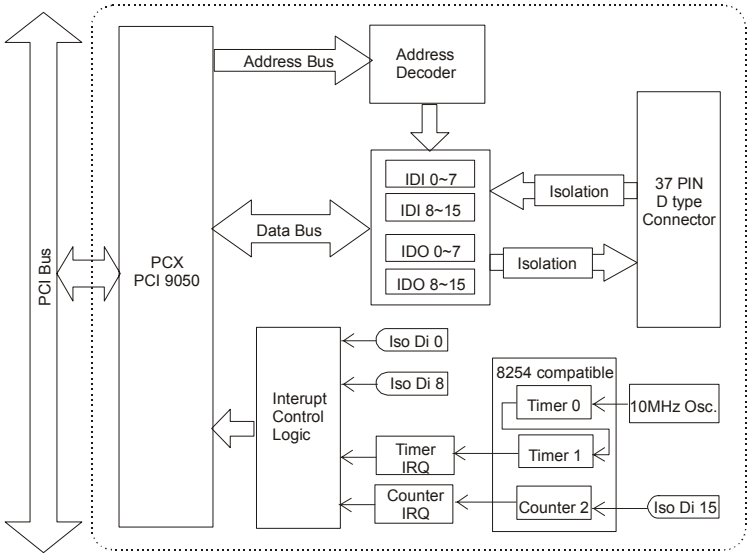
3.1 Vstupně výstupní karta PCI-1750

Pro externí řízení simulátoru výtahu pomocí PLC je použitá I/O karta Advantech PCI-1750, obrázek 1.1. Implementace pomocí univerzální knihovny funkcí obsluhujících kartu pro celou rodinu karet společnosti Advantech nebrání použití jiné vstupně výstupní karty od této společnosti, která má minimálně 8 digitálních vstupů a 16 digitálních výstupů.

Postupně zde uvedu stručný popis možnosti karty, instalaci, popis *DEMO boardu*, výčet použitých funkcí pro komunikaci s kartou a testovací aplikaci PCI-1750.

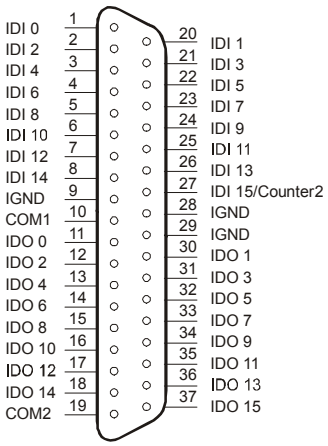
3.1.1 Popis karty

Karta PCI-1750 nabízí šestnáct izolovaných logických vstupů, šestnáct izolovaných logických výstupu a jeden čítač. Přístup k logickým vstupům resp. výstupům je realizován čtením resp. zápisem dvou osmibitových portů. Izolační ochrana karty je 2500 V_{DC}, a proto je karta vhodná pro použití v průmyslových aplikacích. Podrobnější specifikace karty je v lit. [10].



Obrázek 3.1. Blokové schéma karty PCI-1750

Blokové schéma použité vstupně výstupní karty je na obrázku 3.1 a zapojení třicetisedmi pinového konektoru je na obrázku 3.2, popis jednotlivých pinů je v tabulce 3.1. Po zapnutí počítače jsou všechny výstupy karty nastaveny na logickou nulu.



Obrázek 3.2. Zapojení konektoru vstupně výstupní karty

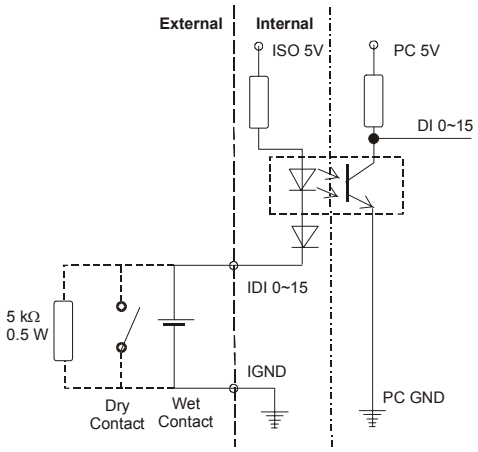
IDI 0 – IDI 15	Izolované digitální vstupy
IDO 0 – IDO 15	Izolované digitální výstupy
IGND	Izolovaná zem
COM1	Společný pin pro připojení induktivní zátěže k výstupům IDO 0 ~ IDO 7
COM2	Společný pin pro připojení induktivní zátěže k výstupům IDO 8 ~ IDO 15
Counter 2	Izolovaný vstup čítače

Tabulka 3.1. Popis pinů konektoru I/O karty

Schéma zapojení jednoho logického vstupu I/O karty je na obrázku 3.3. Vstupy lze připojit ve dvou možných konfiguracích:

- Zapojení se spínačem mezi příslušným vstupním kontaktem a společnou vnitřní zemí, označenou *IGND*. Toto zapojení je pojmenováno jako „*dry contact*“. Pokud je spínač sepnutý, vstup je čten jako logická nula a naopak.
- Na vstupu je připojen zdroj napětí[2)]. Toto zapojení je pojmenováno jako „*wet contact*“. Je-li připojeno 5 ~ 48 V_{DC}, je vstup čten jako logická jednička. Při napětí 0 ~ 2 V_{DC} je čten jako logická nula.

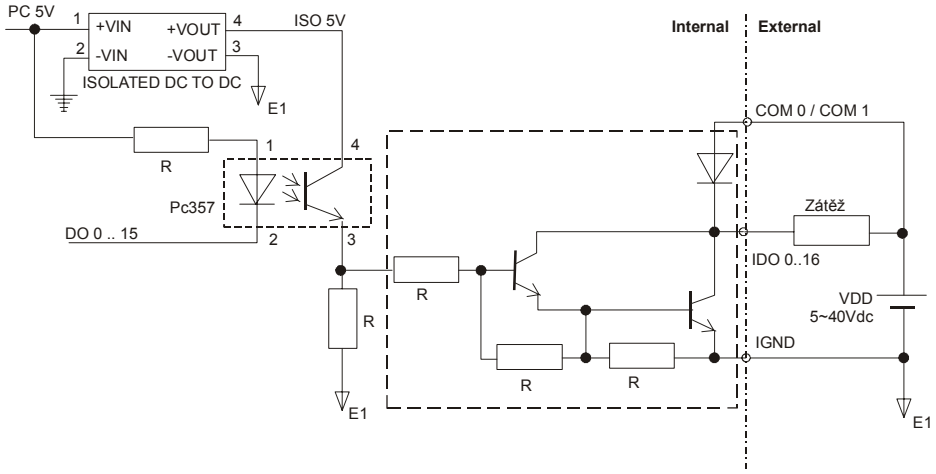
[2] Pokud je vnitřní odpor připojovaného zdroje větší než 5 kΩ, tak nemusí zapojení správně fungovat. Proto se doporučuje ke zdroji připojit paralelně rezistor 5kΩ (0.5 W) k eliminaci vnitřního odporu zdroje.



Obrázek 3.3. Schéma zapojení jednoho logického vstupu I/O karty

Schéma zapojení jednoho logického výstupu karty a připojení externí induktivní zátěže je zobrazeno na obrázku 3.4. Všechny výstupy jsou vybaveny tranzistory v Darlingtonově zapojení. Výstupy jsou vybaveny ochrannou diodou proti přepětí při použití induktivní zátěže. Pro každých osm výstupu je vyveden jeden kontakt pro připojení ochrany. *COM 1* odpovídá výstupům *DO 0 ~ 7* a *COM 2* výstupům *DO 8 ~ 15*. Proud tekoucí z externího zdroje do karty nesmí překročit 200 mA.

Popis přerušovacího systému a čítače karty je v lit. [10].



Obrázek 3.4. Schéma zapojení jednoho logického výstupu I/O karty a připojení externí induktivní zátěže

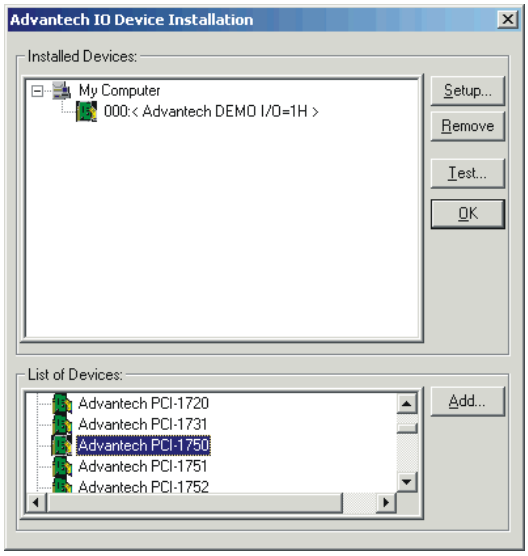
3.1.2 Instalace karty

Instalace vstupně výstupní karty je velice jednoduchá. Karta je vybavena knihovnami, které usnadňují její použití v aplikacích. Podporované operační systémy jsou Windows® 95/98, Windows® NT a Windows® 2000, programovací prostředí:

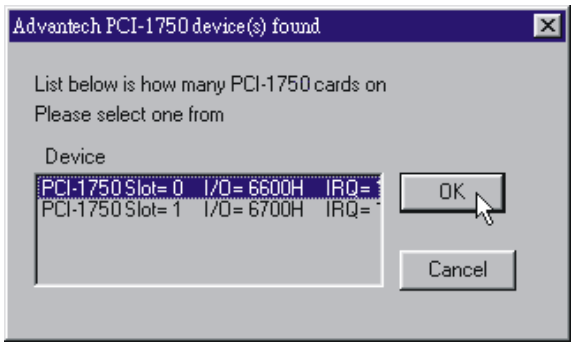
- Microsoft Visual C++ 4.0 (32-bit) nebo vyšší,
- Microsoft Visual Basic 4.0 (32-bit) nebo vyšší,
- Borland Delphi 2.0 (32-bit) nebo vyšší,
- Borland C++ 5.0 (32-bit) nebo vyšší,
- Borland C++ Builder 1.0 (32-bit) nebo vyšší.

Výrobcem dodávaný kompaktní disk [C] také obsahuje ukázkové zdrojové kódy konzolových aplikací vytvořených v prostředí Microsoft Visual Basic®, Microsoft Visual C++® a Borland Delphi™.

Instalace karty probíhá v několika krocích. Nejdříve se nainstaluje Advantech DLL Driver pro příslušnou verzi operačního systému, který mimo jiné obsahuje také ovladač karty. Po vypnutí počítače se karta PCI-1750 vloží do volného PCI slotu. Windows® automaticky nainstaluje potřebný ovladač tohoto zařízení. Nyní se musí spustit konfigurační utilita **devinst.exe** z kořenového adresáře nainstalovaného softwaru Advantech. Pomocí této utility se přidávají a odebírají nainstalované karty v systému, se kterými může pracovat API[3)]. V nabídce **Device/Setup** jsou vidět aktuálně obsluhované karty, obrázek 3.5. Po instalaci je zde automaticky pouze tzv. *DEMO board*, popis v kapitole 3.1.3. Z listu všech podporovaných karet vybereme kartu Advantech PCI-1750 a přidáme ji. Pokud je v systému nainstalováno více karet stejného typu, musíme v dalším kroku přidávání specifikovat konkrétní kartu v daném slotu, obrázek 3.6. Konfigurační utilita dále nabízí možnost otestování výběrem nainstalované karty a stiskem tlačítka **Test**.



Obrázek 3.5. Správa aktuálně obsluhovaných karet API rozhraním



Obrázek 3.6. Výběr příslušné karty daného typu při instalaci

[3] API = Application Interface, programové rozhraní pro použití v aplikacích.

3.1.3 Popis *DEMO boardu*

DEMO board je virtuální karta, která slouží pro testovací účely. Má dva digitální vstupy a dva výstupy na bitech 0 a 1 portu nula, které jsou propojeny. Například nastavením výstupu 0 na logickou jedničku, přečteme ze vstupu 0 také logickou jedničku. Dále má karta čtyři analogové výstupy a osm analogových vstupů. Analogové výstupy jsou propojeny postupně na poslední čtyři analogové vstupy. Popis signálů na analogových vstupech je v tabulce 3.2. Perioda signálů na vstupech 0 až 2 je závislá na rychlosti čtení vstupů, je to 40 čtení.

Analogový vstup	Vstupní signál
0	sinusový signál +5V až –5V
1	obdélníkový signál +5V a –5V
2	trojúhelníkový signál +5V až –5V
3	náhodný signál
4	analogový výstup 0
5	analogový výstup 1
6	analogový výstup 2
7	analogový výstup 3

Tabulka 3.2. Popis analogových vstupů *DEMO boardu*

3.1.4 Vybrané funkce API karty

V této kapitole uvedu popis funkcí API, které jsem použil ve vytvořené aplikaci. Kompletní přehled všech funkcí API včetně popisu je uveden v [11].

```
status = DRV_SelectDevice(HWND hCaller, BOOL GetModule, ULONG *DeviceNum,
                 UCHAR *Description);
```

Zobrazí seznam nainstalovaných I/O karet, ze kterých je možno vybrat.

Vstupy:	**hCaller**	ukazatel na instanci volající aplikace
	GetModule	true – výběr karty ze seznamu z registru false – výběr karty z konfiguračního souboru
Vstup/Výstup:	**DeviceNum**	ukazatel na proměnnou, která bude obsahovat číslo zvoleného zařízení
	Description	místo pro uložení názvu zvoleného zařízení
Výstup:	**status**	0 – bez chyby, jinak kód chyby

```
status = DRV_GetErrorMessage(LRESULT lError,LPSTR lpszErrMsg);
```

Převádí chybový kód na text chyby.

Vstup:	**lError**	kód chyby
Vstup/Výstup:	**lpszErrMsg**	místo pro uložení textu chyby, minimálně 80 znaků
Výstup:	**status**	0 – bez chyby, jinak kód chyby

```
status = DRV_DeviceOpen(ULONG DeviceNum, LONG far * DriverHandle);
```

Vrací parametry příslušející příslušnému zařízení z registru nebo konfiguračního souboru, a alokuje paměť pro jejich uložení, vrátí ukazatel na otevřené zařízení. Tato funkce musí být volaná před voláním dalších funkcí.

Vstup:	**DeviceNum**	číslo otevíraného zařízení
Vstup/Výstup:	**DriverHandle**	ukazatel na proměnnou, do které bude uložen ukazatel na otevřené zařízení
Výstup:	**status**	0 – bez chyby, jinak kód chyby

```
status = DRV_DeviceClose(LONG far * DriverHandle);
```

Uvolňuje alokovanou paměť funkcí *DRV_DeviceOpen.*

Vstup:	**DriverHandle**	ukazatel na zařízení
Výstup:	**status**	0 – bez chyby, jinak kód chyby

```
status = DRV_DioReadPortByte(LONG DriverHandle, LPT_DioReadPortByte
        lpDioReadPortByte);
```

Čte data z digitálního vstupu specifikovaného číslem I/O portu. Čtený byte je specifikován číslem portu od 0 do maxima podle typu karty. Pro kartu PCI-1750 je číslo portu 0 nebo 1.

Vstup:	**DriverHandle**	ukazatel na zařízení
Vstup/Výstup:	**lpDioReadPortByte**	ukazatel na speciální strukturu určenou pro čtení dat
Výstup:	**status**	0 – bez chyby, jinak kód chyby

```
typedef struct tagPT_DioReadPortByte
{
    USHORT      port;
    USHORT far  *value;
} PT_DioReadPortByte, FAR * LPT_DioReadPortByte;
```

Vstup:	**port**	číslo portu, který chceme přečíst
Vstup/Výstup:	**value**	ukazatel na proměnnou, která bude naplněná přečtenou hodnotou

```
status = DRV_DioWritePortByte(LONG DriverHandle, LPT_DioWritePortByte
        lpDioWritePortByte);
```

Zapíše byte na specifikované číslo I/O portu.

Vstupy:	**DriverHandle**	ukazatel na zařízení
	lpDioWritePortByte	ukazatel na speciální strukturu určenou pro zápis dat
Výstup:	**status**	0 – bez chyby, jinak kód chyby

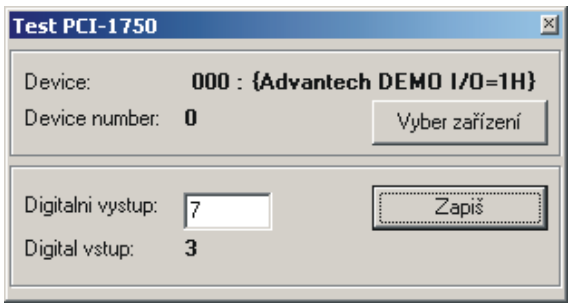
```
typedef struct tagPT_DioWritePortByte
{
    USHORT      port;
    USHORT      mask;
    USHORT      state;
} PT_DioWritePortByte, FAR * LPT_DioWritePortByte;
```

Vstupy:	**port**	číslo portu, na který bude zapsaná hodnota

mask	maska bitů pro zápis; bit, který zde má logickou jedničku bude z **state** přepsán na výstup
state	zde jsou uloženy zapisované bity

3.1.5 Testovací aplikace PCI-1750

Napsal jsem jednoduchou testovací aplikaci pro Windows®, která nabízí možnost zápisu 16-ti bitů na digitální výstup a pravidelného čtení vstupních 8-mi bitů z digitálního vstupu, obrázek 3.7. Výpis všech zdrojových souborů této aplikace je v příloze I. Aplikace používá funkce API karty uvedené v kapitole 3.1.4. Po spuštění aplikace je nutno nejdříve vybrat obsluhované zařízení. Pokud vybereme *DEMO board*, kapitola 3.1.3, zápisem jakéhokoliv dvojslova, přečteme na vstupu vždy hodnotu odpovídající dvěma bitům s nejnižší váhou ze zapsané hodnoty.



Obrázek 3.7. Testovací aplikace vstupně výstupní karty

Pro testy na kartě PCI-1750 jsem vyrobil jednoduchou pomůcku, která má připojeny dva tlačítka mezi vstupy *IDI 0, IDI 1* a *IGND*, mezi výstupy *IDO 0, IDO 1* a *IGND* jsem připojil zkratovou sondu[4)], popis konektoru karty na obrázku 3.2. Použil jsem „dry" variantu zapojení vstupu, obrázek 3.3.

Projekt testovací aplikace pro *Borland C++ Builder*™ verze 5.0 a také zkompilovaný spustitelný soubor je na přiloženém kompaktním disku, obsah disku je uveden v příloze II.

3.2 Přiřazení vstupů a výstupů modelu na konektor PCI-1750

Vstupy a výstupy modelu výtahu definované v kapitole 2.2 jsou přiřazeny na konektor vstupně výstupní karty PCI-1750, obrázek 3.2, podle tabulky 3.3 a 3.4.

Číslo pinu	Jméno	Vstup	Číslo pinu	Jméno	Vstup
3	IDI 4	**D0**	20	IDI 1	**M+**
22	IDI 5	**D1**	1	IDI 0	**M-**
4	IDI 6	**D2**	21	IDI 3	**Slow**
23	IDI 7	**D3**	2	IDI 2	**Res**

Tabulka 3.3. Přiřazení vstupů modelu na konektor vstupně výstupní karty

[4] Zkratová sonda – zařízení, které indikuje zkrat případně pokles odporu pod určitou mez.

Číslo pinu	Jméno	Výstup	Číslo pinu	Jméno	Výstup
17	IDO 12	**TK0**	32	IDO 5	**TP1N**
36	IDO 13	**TK1**	14	IDO 6	**TP2N**
18	IDO 14	**TK2**	33	IDO 7	**A**
37	IDO 15	**TK3**	13	IDO 4	**S**
15	IDO 8	**TP0**	31	IDO 3	**Zat**
34	IDO 9	**TP1**	12	IDO 2	**O**
16	IDO 10	**TP2**	30	IDO 1	**P**
35	IDO 11	**TP3**	11	IDO 0	**K**

Tabulka 3.4. Přiřazení výstupů modelu na konektor vstupně výstupní karty

3.3 Popis aplikace simulátoru výtahu

Aplikace simuluje chování reálného výtahu pro čtyři podlaží pomocí modelu popsaného v kapitole 2. Kompletní aplikace se skládá z několika souborů. Popis souborů je v tabulce 3.5.

Název souboru	Popis
SimulatorVytahu.exe	- spustitelný soubor aplikace pro Windows®
SimulatorVytahu.hlp	- soubor s nápovědou
SimulatorVytahu.cnt	- obsah nápovědy
SimulatorVytahu.ini	- uložená konfigurace všech přednastavení
SimulatorVytahu.log	- záznam chybových hlášení od posledního spuštění
adsapibc.dll	- dynamický modul potřebný pro komunikaci se vstupně výstupní kartou

Tabulka 3.5. Popis souboru distribuce simulátoru výtahu

Aplikace simulátoru využívá čtyř vláken programového kódu:

- **1. vlákno** – hlavní vlákno aplikace.
- **2**. **vlákno** – simulační vlákno. Pokud není model v kritické chybě, je v každé smyčce simulováno postupně chování modelů dveří, modelu kabiny a modelů cestujících. Pro celý běh smyčky se používá jedno nastavení parametrů modelu a jedny hodnoty vstupů, které se načtou na začátku smyčky.
- **3**. **vlákno** – zobrazovací vlákno. Pokud dojde ke změně grafického stavu modelu, je v každé smyčce tohoto vlákna zobrazena na obrazovku grafická část hlavního okna. Zobrazování využívá mezipaměti, aby grafická část nepříjemně neblikala. Všechny grafické prvky se nejdříve vykreslí do této mezipaměti a potom se celý výsledný obraz vykreslí najednou na obrazovku.

- **4**. **vlákno** – vlákno analyzátoru, které zobrazuje signály do grafu. Toto vlákno je spuštěno jen pokud je otevřené okno s analyzátorem, jinak je uspané.

Simulační vlákno má zvýšenou prioritu oproti ostatním vláknům, aby nedocházelo ke zpomalování simulace. Druhé a třetí vlákno je spouštěno v pravidelných intervalech podle nastavení v hlavním okně aplikace.

Model se *inicializuje* po spuštění aplikace, tlačítkem *Reset* nebo vstupním signálem *Res*. *Inicializací* jsou inicializovány všechny logické modely, smazány výstupní hlášení modelu a vymazána historie analyzátoru.

3.3.1 Náhodné generování

3.3.1.1 Náhodné čísla s rovnoměrným rozložením

Pro generováni cílových pater příchozích cestujících, pro rozhodování zda cestující stiskne v kabině tlačítko *STOP* a pro určování zda cestující bude po návratu na patře cestovat do přízemí nebo na jiné patro se používá generátor pseudonáhodných čísel *rand()* z knihovny *stdlib.h*. Tento generátor vrací číslo v rozsahu 0 až *RAND_MAX*. Konstanta *RAND_MAX* je definovaná touto knihovnou.

Pro generování cílového patra cestujícího, který přichází v přízemí se použije následující příkaz:

```
CilovePatro = rand()%3 + 1;
```

Proměnná *CilovePatro* pak obsahuje číslo v rozsahu 1 až 3 odpovídající patru, kam bude daný cestující cestovat.

Pro určování zda cestující stiskne tlačítko *STOP* se porovnává zda vygenerované číslo leží v části celého intervalu odpovídající nastavené pravděpodobnosti stištění tlačítka *STOP* cestujícím.

Aby jednotlivé náhodně generované události měly rovnoměrné rozložení, musí každý generátor těchto událostí využívat svůj nezávislý generátor pseudonáhodných čísel. Pro tyto účely jsem napsal objekt *TRand*, definovaný v souboru *Rand.h*, který s využitím knihovní funkce generuje několik nezávislých řad pseudonáhodných čísel.

3.3.1.2 Intervaly příchodů cestujících

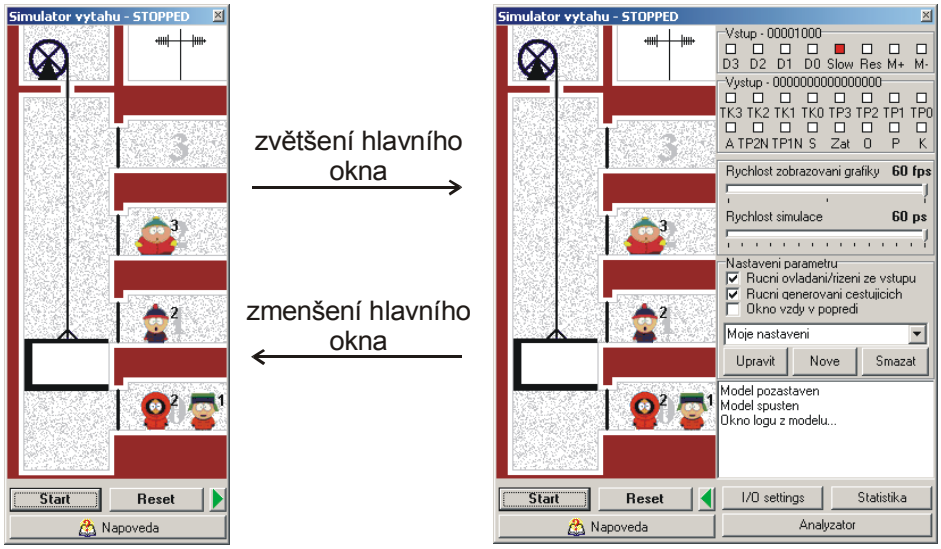
Počet příchodů nového cestujícího v přízemí za určitou časovou jednotku má rozložení podle *Poissonova rozdělení*, kapitola 2.4.3.2. Generátor pro příchod nových cestujících generuje časové intervaly, po kterých přicházejí cestující, tak, aby jejich průměrná četnost odpovídala požadovanému nastavení. Způsob generování intervalů pomocí pseudonáhodného generátoru je popsán v literatuře [4]. Vygenerovaný interval pro příchod nového cestujícího může nabývat neomezené velikosti. Omezil jsem tento interval na maximálně trojnásobek zadané průměrné doby příchodu nového cestujícího, aby nedocházelo k dlouhým prodlevám mezi příchody nových cestujících v přízemí.

3.3.2 Grafický vzhled aplikace a popis funkcí

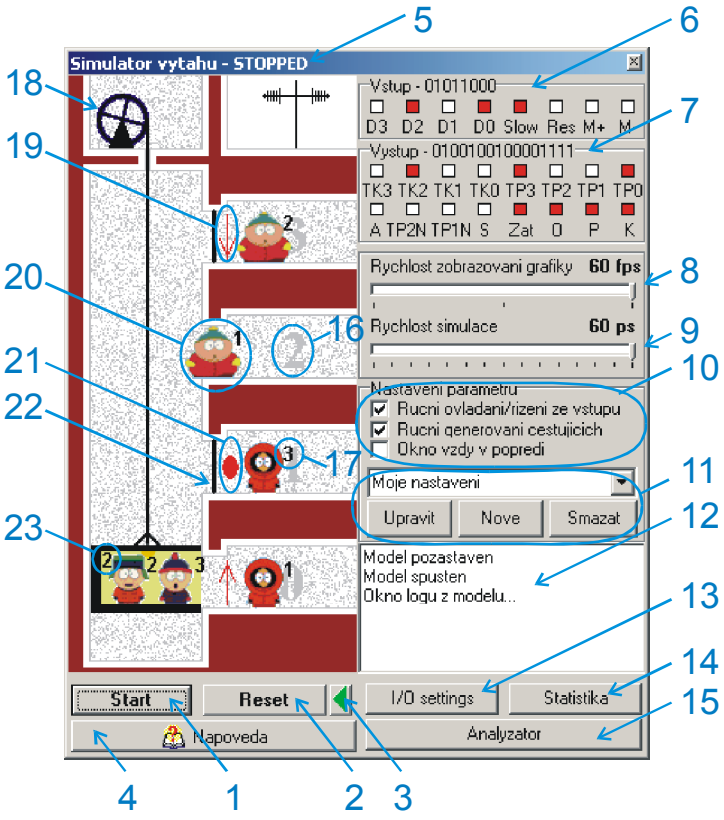
3.3.2.1 Hlavní okno simulátoru

Hlavní okno aplikace může mít dvě velikosti, obrázek 3.8. Přepínání mezi těmito dvěmi rozměry okna se provádí pomocí tlačítka se zelenou šipkou. V případě menšího rozměru okna je graficky zobrazován stav modelu a tlačítka pro spuštění resp. zastavení modelu, inicializaci modelu, nápovědu a zvětšení okna. Pokud je rozměr okna zvětšen, jsou zobra-

zovány navíc aktuální vstupy a výstupy modelu, nastavení pro rychlost zobrazování a simulace, chybové zprávy a další nastavení.



Obrázek 3.8. Grafický vzhled hlavního okna *Simulátoru Výtahu*



Obrázek 3.9. Hlavní okno aplikace s popisem

Popis hlavního okna aplikace, obrázek 3.9:

1 - Tlačítko Start/Stop – spouští a zastavuje simulaci výtahu. Je-li simulace zastavena, model nereaguje na vstupní signály.

2 - Tlačítko Reset – provede inicializaci celého modelu. Všechny požadavky cestujících jsou zrušeny, dveře zavřeny a kabina stojí v inicializačním patře (zvolené v nastavení).

3 - Tlačítko Zvětši/Zmenši rozměr okna – přepíná mezi menší a větší variantou zobrazení hlavního okna aplikace. Menší varianta je vhodná pro současné používání simulátoru výtahu a aplikace pro vytváření programu pro PLC.

4 - Tlačítko Nápověda – zobrazí nápovědu aplikace.

5 - Titulek aplikace – kromě názvu aplikace zobrazuje také stav modelu, zda je spuštěn (*RUN*) nebo zda je zastaven (*STOPPED*).

6 - Vstupní box – zobrazuje stav jednotlivých vstupů modelu výtahu. Pokud je daný čtvereček vybarven červenou barvou, je daný signál aktivní (*odpovídá logické jedničce*). Jinak je signál neaktivní. Vstupy modelu jsou popsány v kapitole 2.2.1.

7 - Výstupní box – zobrazuje stav jednotlivých výstupů modelu. Barevné značení je stejné jako u vstupního boxu. Výstupy modelu jsou popsány v kapitole 2.2.2.

8 - Rychlost zobrazování grafiky – nastavení určující počet překreslení grafické části okna aplikace. Lze nastavit ¼, ½ nebo 1 násobek *rychlosti simulace*.

9 - Rychlost simulace – nastavení určující počet spuštění simulační smyčky. Lze nastavit hodnoty od 4 do 60-ti cyklů za sekundu. Čím je tato hodnota větší, tím je simulace plynulejší. Toto nastavení ovlivňuje nastavení všech parametrů modelu, které se vztahují k jednomu kroku simulace.

10 - Nastavení parametrů:

a - Ruční ovládání/řízení ze vstupu – je-li tato volba zatržena, tak jsou vstupy modelu generovány ručně. Jinak jsou vstupy pravidelně čteny z vybrané nainstalované vstupně výstupní karty. Způsob ručního generování vstupů modelu a ručního generování příchodů cestujících je popsán v kapitole 3.3.2.2.

b - Ruční generování cestujících – je-li tato volba zatržena, tak jsou cestující v modelu generováni pouze ručně. Jinak jsou generování modelem automaticky.

c - Okno vždy v popředí – zatržením této volby bude hlavní okno aplikace simulátoru zobrazováno vždy v popředí před ostatními okny jiných spuštěných aplikací. Tato funkce je vhodná při ladění řídícího programu pro PLC, při současném spuštění více aplikací.

11 - Přednastavení parametrů:

a - Volba přednastavení – vybrané přednastavení, které je zrovna používáno. Na toto přednastavení se pak vztahují volby *b* a *d*.

b - Tlačítko Upravit – vyvolá okno pro změnu parametrů modelu zvoleného přednastavení. Okno změny nastavení je popsáno v kapitole 3.3.2.3.

c - Tlačítko Nové – vloží nové přednastavení a vyvolá jeho editaci.

d - Tlačítko Smazat – vymaže zvolené přednastavení parametrů modelu. Pokud je vybráno poslední zbylé přednastavení, nelze už jej smazat.

12 - Okno logů z modelu – zde jsou vypisována hlášení modelu pro uživatele. Kliknutím do této oblasti je vyvoláno okno s detailním výstupem hlášení, kapitola 3.3.2.4.

13 - I/O settings – vyvolá okno pro výběr vstupně výstupní karty.

14 - Statistika – zobrazí okno se statistikou modelu od poslední inicializace.

15 - Analyzátor – vyvolá okno logického analyzátoru zvolených logických signálů.

16 - Číslo patra – číslo patra. Nulté patro odpovídá přízemí.

17 - Cílové patro – takto je zobrazováno cílové podlaží cestujícího, u kterého je po celou dobu pohybu osoby v modelu zobrazeno vpravo nahoře.

18 - Strojovna výtahu – při pohybu cestujícího se ve strojovně otáčí hlavní nosné kolo.

19 - Přivolávací tlačítko – zobrazeno, pokud některý cestující drží stištěné přivolávací tlačítko na tomto patře. Šipka určuje, které ze dvou tlačítek je stištěno.

20 - Cestující – cestující mají v modelu čtyři různé grafické obrázky, které se cyklicky obměňují. Tento cestující bude při průjezdů kabiny výtahu zraněn, protože vstoupil do šachty výtahu v době, kdy byly dveře otevřeny a kabina výtahu tam nestála.

21 - Přivolávací tlačítko – takto je zobrazováno stištění přivolávacího tlačítka na daném podlaží v režimu jednoho tlačítka na každém podlaží.

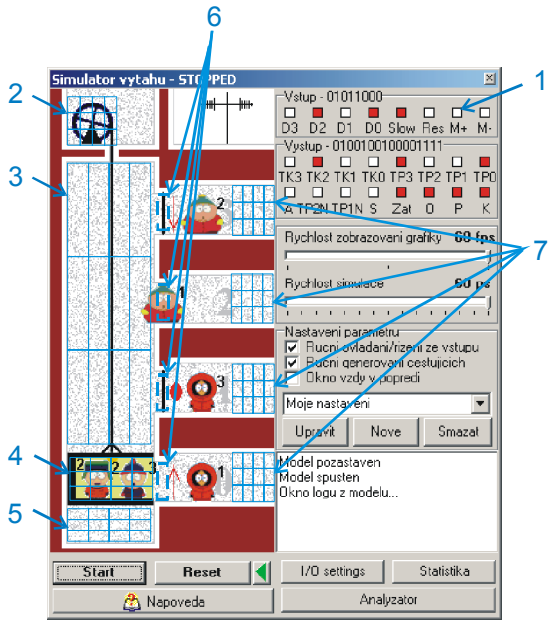
22 - Dveře – zobrazeny úplně zavřené dveře od výtahu. Pokud v dané části grafické oblasti není nic zobrazeno, jsou dveře úplně otevřeny.

23 - Volící tlačítko v kabině – je-li v této části kabiny zobrazeno nějaké číslo, indikuje to stištění odpovídajícího volícího tlačítka cílového patra.

První známka nervozity cestujícího je zobrazována v grafické části cyklickým posunem cestujícího o několik bodů vpravo a vlevo. Stiskne-li některý cestující v kabině tlačítko *STOP*, je přes kabinu zobrazen červený nápis „STOP“. Je-li model ve stavu kritické chyby, je nápis „STOP“ zobrazen vpravo nahoře grafické části a model musí být inicializován.

3.3.2.2 Ruční generování vstupních signálů a příchodů cestujících

Je-li povoleno v hlavním okně ruční generování vstupních signálů, nejsou vstupní signály čteny ze vstupně výstupní karty, ale jsou generovány stiskem levého tlačítka myši nad příslušnou oblastí 1 až 6, obrázek 3.10. Popis jednotlivých oblastí je v tabulce 3.6. Stiskem tlačítka myši v oblasti 7 se vygeneruje na daném podlaží nový cestující s náhodně zvoleným cílovým patrem.



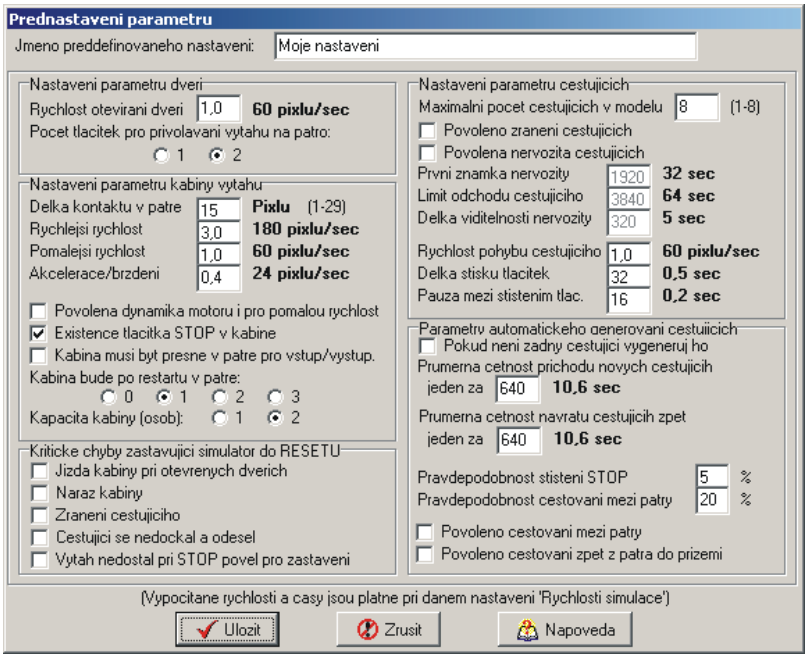
Obrázek 3.10. Ruční generování vstupních signálů a příchodů cestujících

Oblast	Odpovídající akce
1	každý čtvereček odpovídá jednomu vstupnímu signálu; každým stiskem je daný signál invertován
2	stiskem je vždy invertován signál *Slow*
3	poloha nad kabinou; stisk aktivuje signál *M+*
4	v oblasti kabiny; stisk deaktivuje M+ i M-
5	poloha pod kabinou; stisk aktivuje signál *M-*
6	stisk otevírá a zavírá příslušné dveře
7	stiskem je vypuštěn nový cestující na daném podlaží

Tabulka 3.6. Popis oblastí ručního generování vstupních signálů

3.3.2.3 Nastavení parametrů modelu

V tomto okně, obrázek 3.11, se mění parametry vybraného přednastavení z hlavního okna aplikace. Přednastavení může být vytvořeno několik, mohou být libovolně upravovány a mazány. Pouze poslední přednastavení nemůže být smazáno. Aktivní nastavení modelu je nastavení, které je vybráno v hlavním okně. Při ukončení aplikace jsou všechny přednastavení, nastavení rychlosti zobrazování grafiky a simulace uloženy do konfiguračního souboru odkud jsou při spuštění zase načteny. Je-li soubor poškozen nebo chybí, vznikne pouze jedno standardní přednastavení. Konfigurační soubor je uložen ve stejném adresáři jako binární soubor aplikace a jmenuje se „SimulatorVytahu.ini“.

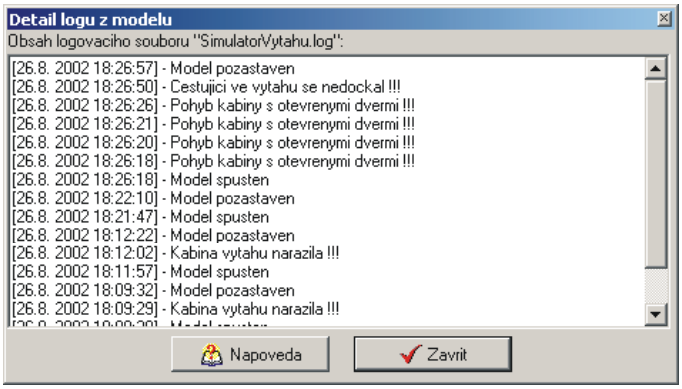


Obrázek 3.11. Okno pro nastavení parametrů modelu

Většina nastavení se nastavuje pro jednu simulační smyčku programu. Skutečná hodnota s přihlédnutím k *rychlosti simulace* je zobrazená hned vedle editovaného pole. Jednotlivé pole odpovídají nastavení modelu výtahu, popsanému v kapitole 2.

3.3.2.4 Detail hlášení z modelu

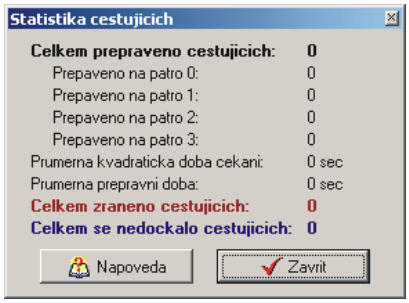
Všechny systémové hlášení a chyby generované modelem výtahu, kapitola 2.4.4, jsou zobrazovány přímo v hlavním okně aplikace a také jsou zaznamenávány do textového souboru „SimulatorVytahu.log“ umístěném v aktuálním adresáři aplikace. Soubor je při spuštění aplikace vždy přemazán. Okno s detailním zobrazením hlášení pro uživatele obsahuje také čas vzniku hlášení, obrázek 3.12. Mezi systémové hlášení patří: aplikace spuštěna, aplikace ukončena, model spuštěn, model zastaven a inicializace modelu.



Obrázek 3.12. Okno detailu hlášení z modelu výtahu

3.3.2.5 Zobrazení statistiky modelu

Aplikace přehledně zobrazuje všechny statistiky popsané v kapitole 2.4.3.4, obrázek 3.13.



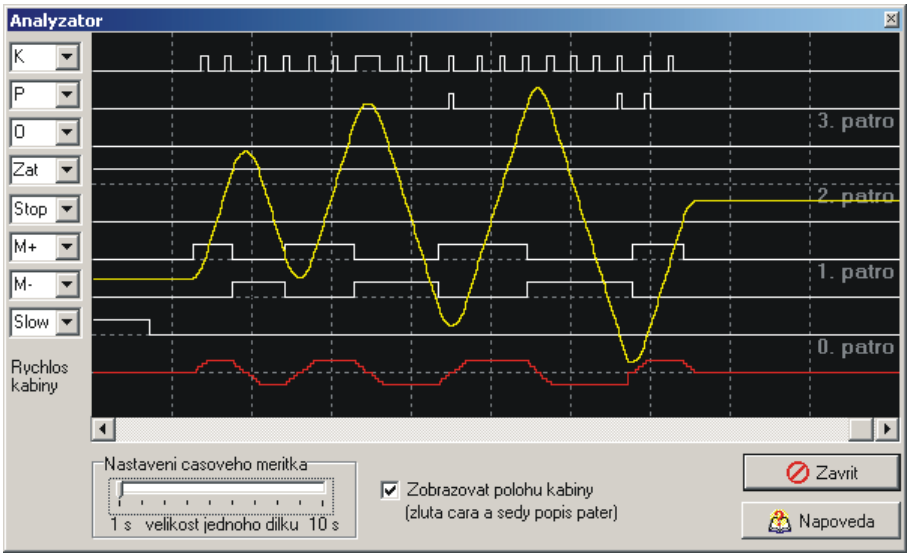
Obrázek 3.13. Okno statistiky běhu modelu

3.3.2.6 Analyzátor logických signálů

Analyzátor slouží pro zkoumání přesného chování modelu výtahu v době odlaďování řídícího programu pro PLC, obrázek 3.14.

Graf analyzátoru je zobrazován průběžně, lze se libovolně vracet pomocí posuvníku v historii, také lze měnit měřítko na časové ose (osa **x**) v intervalu 1 až 10 sekund na jeden dílek. Při inicializaci modelu je historie analyzátoru smazána. Na ose **y** si lze vybírat zobrazované logické signály ze vstupních signálů *M+*, *M-*, *Slow* a z výstupních signálů *K*, *P*, *O*, *Zat*, *Stop*. Nebo vybráním prázdného pole, nemusí být zobrazován žádný signál na dané pozici. Zobrazované logické signály mají v dolní pozici úroveň odpovídající logické nule (neaktivní) a v horní pozici logickou jedničku (aktivní). Dole je vždy zobrazovaná aktuální rychlost pohybu kabiny (červená čára).

Pokud je povoleno zobrazování aktuální polohy kabiny, je poloha zobrazena žlutou čárou a vpravo je zobrazeno měřítko této polohy. Popisek měřítka polohy náleží nejbližší přerušované čáře, která označuje přesnou úroveň polohy patra. Zobrazování polohy kabiny je vhodné pro odlaďování řízení modelu při použité dynamice kabiny.



Obrázek 3.14. Okno analyzátoru se zobrazenými všemi log. signály a také s polohou kabiny

Na obrázku je vidět vliv dynamiky motoru při rychlejším pohonu kabiny. Po změně směru pohonu kabiny, je vidět brždění a pak zrychlování kabiny na opačnou stranu.

U posledního obloučku dole je patrný náraz kabiny na konec výtahové šachty, projevuje se skokovou změnou rychlosti kabiny.

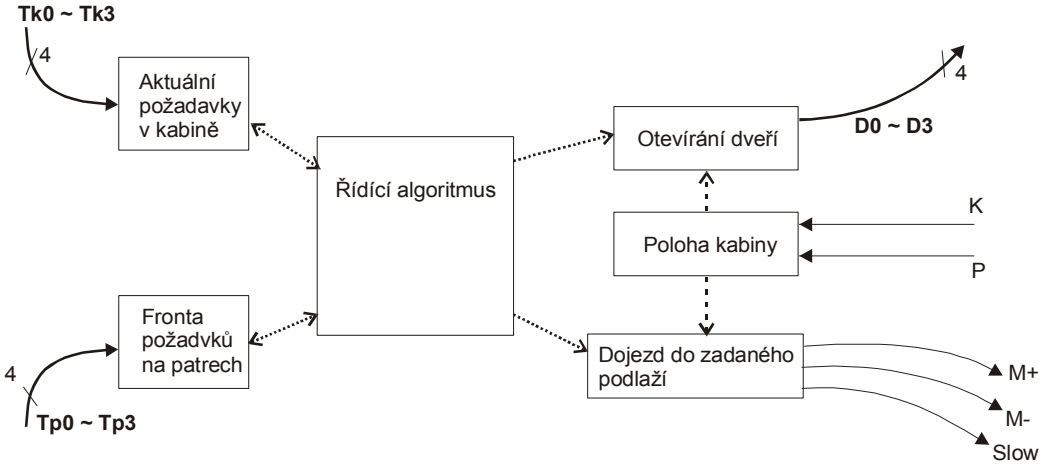
3.3.2.7 Nápověda aplikace

K aplikaci je vytvořena kompletní podrobná nápověda pomocí demoverze programu *ForeHelp* verze 5.0.2a. Projekt, ze kterého byla nápověda vytvořená, je na přiloženém kompaktním disku, obsah v příloze II.

Každé okno aplikace simulátoru výtahu má tlačítko pro vyvolání nápovědy daného okna.

4 Jednoduché řízení výtahu

V této kapitole pouze naznačím *jednoduché řízení* modelu výtahu, které nebudu specifikovat podrobně. Bude řídit model výtahu v jednotlačítkovém režimu, který nemá tlačítko *STOP*. *Systém řízení* výtahu je zobrazen na obrázku 4.1.



Obrázek 4.1. Řídící systém výtahu

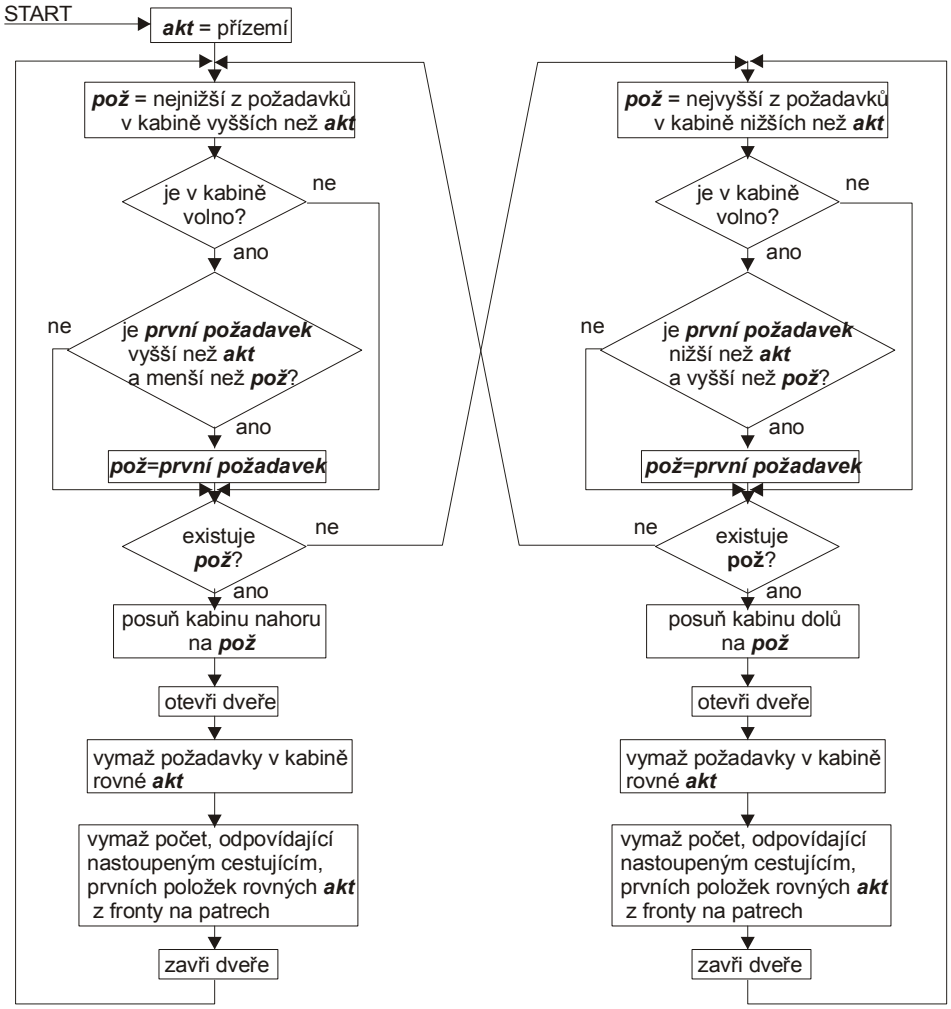
Řídící systém komunikuje s *modelem výtahu* pomocí signálů popsaných v kapitole 2.2. Skládá se z několika samostatných částí:

- **Aktuální požadavky v kabině** – shromažďuje a udržuje požadavky na cílové patro cestujících v kabině. Splněné požadavky řídící algoritmus vždy vymaže.
- **Fronta požadavků na patrech** – přijímá požadavky na přivolání kabiny do jednotlivých pater a řadí je do fronty v pořadí příchodu. Předává *řídícímu algoritmu* na požádání *první požadavek* a umí smazat jeden nebo dva první požadavky pro dané patro.
- **Otevírání dveří** – otevírá a zavírá dveře na podlaží, kde je zrovna kabina výtahu.
- **Poloha kabiny** – část, která ze signálů *K* a *P* vypočítává polohu kabiny.
- **Dojezd do zadaného podlaží** – tato část, na povel z řídícího algoritmu, zajistí dojezd kabiny do žádaného podlaží. Pokud má řízení používat obě rychlosti pohybu kabiny, řízení zajistí tato část.
- **Řídící algoritmus** – specifikuji v následující kapitole.

4.1 Řídící algoritmus

Řídící algoritmus je znázorněn na obrázku 4.2. Tento algoritmus pohybuje kabinou výtahu postupně nahoru a dolů, a v pořadí příchodů cestujících je přepravuje na jejich cílové podlaží.

Proměnná *akt* udržuje aktuální polohu kabiny výtahu. Na začátku je tato proměnná nastavená na polohu výtahu odpovídající přízemí. Levá část zobrazeného algoritmu vyřizuje požadavky pohybem výtahu nahoru a pravá část pohybem dolů. Když je v kabině výtahu volno, tak se algoritmus snaží splnit *první požadavek* na přivolání kabiny.



Obrázek 4.2. Řídící algoritmus

Po otevření dveří výtahu, algoritmus vymaže z požadavků v kabině požadavky směřující do patra, kde stojí kabina. Tím uvolní místo pro příchod nových cestujících. Pokud někdo v tomto podlaží nastoupí, stiskne tlačítko volby cílového patra. Tím algoritmus pozná kolik osob nastoupilo a musí vymazat tento počet přivolávacích požadavků tohoto patra z fronty požadavků na přivolání.

5 Závěrečné zhodnocení

V rámci diplomové práce:

- jsem navrhl řiditelný model výtahu vhodný pro výuku, ve kterém se plynule pohybují cestující,
- navržený model jsem poté vytvořil jako aplikaci pro operační systém Windows®, která komunikuje s řídícím systémem pomocí vstupně výstupní karty PCI 1750,
- vytvořil jsem kompletní nápovědu k aplikaci simulátoru výtahu,
- v závěru práce jsem navrhl jednoduchý řídící algoritmus pro řízení tohoto modelu.

Navržený model je vhodný pro použití při výuce. Studenti si mohou libovolně nastavovat všechny parametry modelu a implementovat řízení pro různé typy výtahů. Vytvořená aplikace přehledně zobrazuje svůj aktuální stav v grafické oblasti. Ruční generování vstupních signálů umožňuje vyzkoušet si chování modelu výtahu bez připojeného PLC. Pro hledání chyby v řídícím programu je velice užitečné použití analyzátoru logických signálů.

Po dohodě s vedoucím diplomové práce jsem k aplikaci nevytvořil ukázkový řídící program pro PLC, protože do uzávěrky diplomové práce jsem neměl k dispozici převodník úrovní mezi vstupně výstupní kartu a PLC.

6 Použité programy

[A] Inprise Corporation. *Borland C++Builder™ version 5.0.*
[B] ForeFront Boulder Corporation. *ForeHelp Demo version 5.0.2a.*
[C] Adantech Corporation. *Advantech DLL Drivers.*

7 Literatura

[1] Rogalewicz Vladimír. *Pravděpodobnost a statistika pro inženýry.* ČVUT, Praha, 1998.
[2] Rogalewicz Vladimír. *Stochastické procesy.* ČVUT, Praha, 1993.
[3] Duenas, Rendón, Sanchez. *The Elevator System: Logical Model.* http://citeseer.nj.nec.com/19798.html. IPTES Project EP5570, 1993.
[4] Steeve Karg. *Elevator Simulator Design.* http://www.angelfire.com/trek/software/elevator.html. University of Phoenix, Denver, 1996.
[5] Herout. *Učebnice jazyka C I. a II.* KOPP, České Budějovice, 1993.
[6] Šusta Richard. *Učební text pro Programovací jazyky pro řízení.* http://dce.felk.cvut.cz/pjr/skripta/jazykc.pdf. ČVUT FEL, Praha.
[7] Šusta Richard. *Programovací jazyky pro řízení ve Windows.* ČVUT, Praha, 1999.
[8] Inprise Corporation. *Nápověda Borland C++Builderu™ 5.0.*
[9] ForeFront Boulder Corporation. *Nápověda ForeHelp 5.*
[10] Advantech Corporation. *PCI – 1750 User's Manual.* Advantech Co., 1998.
[11] Advantech Corporation. *Advantech DLL Drivers User's Manual and Programmer's Reference.* 3rd Edition.

Seznam tabulek

Seznam obrázků

Příloha I. Testovací aplikace PCI-1750

Výpis souboru „PCI1750.cpp“:

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
USERES("PCI1750.res");
USEFORM("HOkno.cpp", Form1);
USELIB("ADSAPIBC.LIB");
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
     try
     {
           Application->Initialize();
           Application->Title = "Test PCI-1750";
           Application->CreateForm(__classid(TForm1), &Form1);
           Application->Run();
     }
     catch (Exception &exception)
     {
           Application->ShowException(&exception);
     }
     return 0;
}
//---------------------------------------------------------------------------
```

Výpis souboru „HOkno.h“:

```
//---------------------------------------------------------------------------
#ifndef HOknoH
#define HOknoH
//---------------------------------------------------------------------------
#include "driver.h"
#include <limits.h>

#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:      // IDE-managed Components
     TTimer *Timer1;
     TGroupBox *GroupBox1;
     TLabel *Label1;
     TLabel *Label2;
     TLabel *lDevice;
     TLabel *lDeviceNumber;
     TButton *Button1;
     TGroupBox *GroupBox2;
     TLabel *Label3;
     TLabel *lIn;
     TLabel *Label4;
     TEdit *eOut;
```

```
     TButton *Button2;
     void __fastcall Button1Click(TObject *Sender);
     void __fastcall FormCreate(TObject *Sender);
     void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
     void __fastcall Timer1Timer(TObject *Sender);
     void __fastcall Button2Click(TObject *Sender);
private:    // User declarations
     LRESULT error;
     UCHAR ucErrMsg[MaxErrMsgLen];
     ULONG ulDevNum;
     UCHAR ucDevName[MAX_DEVICE_NAME_LEN];
     LONG lDriverHandle;
     bool bOpenDevice;
public:           // User declarations
     __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

Výpis souboru „HOkno.cpp“:

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "HOkno.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
     : TForm(Owner)
{ }
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
     ucDevName[0] = 0;
     if ((error = DRV_SelectDevice((HWND)Handle,(BOOL)false,(ULONG
               *)&ulDevNum,(UCHAR *)ucDevName)) != SUCCESS)
     {
          DRV_GetErrorMessage(error,(UCHAR far *)ucErrMsg);
          Application->MessageBox((char *)ucErrMsg, "Driver Message",
               MB_OK);
          exit(0);
     };
     lDeviceNumber->Caption = IntToStr(ulDevNum);
     lDevice->Caption = (char *)ucDevName;
     if ((error = DRV_DeviceOpen(ulDevNum,(LONG far *)&lDriverHandle)) !=
               SUCCESS)
     {
          DRV_GetErrorMessage(error,(UCHAR far *)ucErrMsg);
          Application->MessageBox((char *)ucErrMsg, "Driver Message",
               MB_OK);
          exit(0);
     };
```

```
     bOpenDevice = true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
     ulDevNum = ULONG_MAX;
     bOpenDevice = false;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction &Action)
{
     if (bOpenDevice) {
          DRV_DeviceClose((LONG far *)&lDriverHandle);
     };
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
     if (bOpenDevice) {
          USHORT usValue;
          PT_DioReadPortByte ReadValue;
          ReadValue.port = 0;                //port 0
          ReadValue.value = (USHORT far *)&usValue;
          if ((error = DRV_DioReadPortByte(lDriverHandle,
              (LPT_DioReadPortByte)&ReadValue)) != SUCCESS)
          {
               DRV_GetErrorMessage(error,(UCHAR far *)ucErrMsg);
               Application->MessageBox((char *)ucErrMsg, "Driver Messa-
                   ge", MB_OK);
               exit(0);
          };
          lIn->Caption = IntToStr((int)usValue);
     };
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
     if (bOpenDevice) {
          PT_DioWritePortByte WriteValue;
          WriteValue.port = 0;               //port 0
          WriteValue.mask = 255;
          WriteValue.state = (USHORT) (StrToInt(eOut->Text)&0xff);
          if ((error = DRV_DioWritePortByte(lDriverHandle,
              (LPT_DioWritePortByte)&WriteValue)) != SUCCESS)
          {
               DRV_GetErrorMessage(error,(UCHAR far *)ucErrMsg);
               Application->MessageBox((char *)ucErrMsg, "Driver Messa-
                     ge", MB_OK);
               exit(0);
          };
          WriteValue.port = 1;          //port 1
          WriteValue.mask = 255;
          WriteValue.state = (USHORT) (StrToInt(eOut->Text)&0xff00)>>8;
          if ((error = DRV_DioWritePortByte(lDriverHandle,
              (LPT_DioWritePortByte)&WriteValue)) != SUCCESS)
          {
               DRV_GetErrorMessage(error,(UCHAR far *)ucErrMsg);
```

```
                Application->MessageBox((char *)ucErrMsg, "Driver Messa-
                      ge", MB_OK);
                exit(0);
            };
        };

}
//---------------------------------------------------------------------------
```

Příloha II. Obsah přiloženého kompaktního disku

V následující tabulce je uveden abecední seznam všech kořenových adresářů přiloženého kompaktního disku s popisem obsahu adresářů.

Adresář	Obsah
Help_Source	- zdrojové soubory nápovědy aplikace vytvořené pomocí programu *ForeHelp Demo version 5.0.2a*
Release	- zkompilovaná aplikace simulátoru výtahu; nutno před spuštěním překopírovat celý obsah tohoto adresáře na pevný disk do adresáře, kde je povolen úplný přístup uživateli, pod kterým je aplikace spouštěna
Source	- zdrojové soubory simulátoru výtahu včetně projektu pro *Borland C++ Builder*™ verze 5.0; zdrojové soubory obsahují komentáře
Test_PCI_1750	- zkompilovaná testovací aplikace vstupně výstupní karty PCI-1750
Test_PCI_1750_Source	- zdrojové soubory testovací aplikace pro vstupně výstupní kartu včetně projektu pro *Borland C++ Builder*™ verze 5.0