Implementace komunikačního protokolu MP-Bus pro řídicí systémy firmy AMiT

Bc. František Grebeníček

Diplomová práce
2006

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav automatizace a řídicí techniky
akademický rok: 2005/2006

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. František GREBENÍČEK**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Automatické řízení a informatika**
Téma práce:	**Implementace komunikačního protokolu MP-Bus pro řídicí systémy firmy AMiT**

Zásady pro vypracování:

Využití řídících systémů českého výrobce firmy AMiT jako master pro komunikaci se servopohony švýcarské firmy Belimo pomocí komunikačního protokolu MP-Bus této společnosti.
Úkolem je nastudovat tento komunikační protokol a implementovat jej pro použití s řídícími systémy firmy AMiT. V první fázi se bude jednat o vytvoření jednoduché uživatelské komunikace pomocí volně programovatelného parametrizačního programu PSP3 firmy AMiT, pro zadání polohy, čtení polohy a čtení teploty. V další fázi pak půjde o vytvoření funkčních modulů protokolu MP-Bus pro program PSP3, pomocí nichž bude možné klapkové a ventilové servopohony firmy Belimo ovládat tak, aby bylo možno využít všech jejich funkcí, definicí adresy zařízení počínaje a indikací havarijních stavů konče.
V rámci této diplomové práce bude také vytvořena dokumentace s popisem vytvořených funkčních modulů protokolu MP-Bus.

Rozsah práce:

Rozsah příloh:

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

Seznam odborné literatury:

AMiT, spol. s r. o.: PSP3 – Obsluha programu, firemní technická dokumentace, 2003
AMiT, spol. s r. o.: PSP3 – Knihovny funkčních modulů, programátorská příručka (1. a 2. část), firemní technická dokumentace, 2005
AMiT, spol. s r. o.: PSE3-REMOTE – Knihovna základních funkčních modulů pro komunikaci po sériové lince, firemní technická dokumentace
AMiT, spol. s r. o.: Katalogové listy řídích systémů firmy AMiT, firemní technická dokumentace
Belimo Automation AG: MP-bus documentation V2.4, firemní technická dokumentace, 2004
Belimo Automation AG: MFT(2): Direct coupled actuators with and without safety function, firemní technická dokumentace
Belimo Automation AG: ZIP-232-MP: Interface module for control cabinet installation, firemní technická dokumentace
www.amit.cz
www.belimo.org

Vedoucí diplomové práce: **Ing. Petr Chalupa, Ph.D.**
Ústav řízení procesů

Datum zadání diplomové práce: **14. února 2006**

Termín odevzdání diplomové práce: **26. května 2006**

Ve Zlíně dne 14. února 2006

L.S.

prof. Ing. Vladimír Vašek, CSc.
pověřený děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Jádrem celé této diplomové práce je tzv. uživatelská komunikace mezi řídicími systémy firmy AMiT a servopohony firmy Belimo, která probíhá prostřednictvím komunikačního protokolu MP-Bus této společnosti. Vytvořené uživatelské komunikace jsou dvě, z nichž jedna slouží pro nastavení adresy servopohonu a druhá k samotnému ovládání servopohonů. Kromě toho jsou v této práci blíže popsány samotné servopohony, rozebrán komunikační protokol MP-Bus a jeho nejdůležitější příkazy, a čtenář je uveden do problematiky programování řídicích systémů firmy AMiT. Na závěr je potom navrhnuto jakým směrem pokračovat dál při vytváření softwarové podpory k tomuto protokolu.

Klíčová slova: Belimo, AMiT, MP-Bus, MFT, PSP3, funkční bloky, uživatelská komunikace, klapkové servopohony, ventilové servopohony, VAV jednotky

ABSTRACT

Heart of this diploma thesis is so-called user communication between AMiT company control systems and Belimo company actuators by means of MP-Bus communication protocol. There was created two user communications for this thesis. One is for actuators addressing and second is for own actuators controlling. In this thesis, there is also look at own actuators and theirs functions, MP-Bus communication protocol description and its commands, and introduction to AMiT control systems programming. On the end there are mentioned some suggestion what else could be done in software support for this protocol in the future.

Keywords: Belimo, AMiT, MP-Bus, MFT, PSP3, function blocks, user com, damper actuators, valve actuators, VAV units

Na tomto místě bych velmi rád poděkoval svému vedoucímu diplomové práce Ing. Petru Chalupovi, Ph.D., za odborné vedení, které mi poskytl během řešení mé diplomové práce. Poděkování patří také mému zaměstnavateli, jenž mi poskytl prostor a technické prostředky pro realizaci této práce, a také mým spolupracovníkům za spoustu cenných rad a připomínek a zejména za pomoc při pronikání do tajů programování řídicích systémů. Určitě bych také neměl zapomenout na firmu Belimo, která mi poskytla nezbytný hardware a potřebnou technickou dokumentaci.

OBSAH

ÚVOD

Tato práce bude pojednávat o postupu při vytváření uživatelské komunikace mezi řídicími systémy firmy AMiT a servopohony firmy Belimo, která umožňuje ovládat základní funkce těchto servopohonů a je realizována prostřednictvím komunikačního protokolu MP-Bus firmy Belimo. Podrobněji pak rozebereme také tento protokol, který umožňuje připojit na jednu sběrnici až osm servopohonů, a jeho příkazy potřebné k ovládání vybraných funkcí pohonů. V první části si však nejprve blíže popíšeme vlastní servopohony.

O společnosti Belimo

Švýcarská společnost Belimo je předním světovým výrobcem elektrických servopohonů pro topenářská, vzduchotechnická a klimatizační zařízení s více než 30-ti letou tradicí a zastoupením ve více než 45-ti zemích světa. Dvě třetiny v Evropě montovaných klapkových servopohonů pochází právě od této firmy. V Americe je to pak takřka polovina.

Její sortiment se dá rozdělit do následujících skupin:

- servopohony vzduchotechnických klapek
- pohony požárních a odkuřovacích klapek
- regulace průtoku vzduchu
- regulační armatury a pohony s otočným nebo zdvihovým pohybem
- systémy okenního větrání

Jako vedoucí společnost ve svém segmentu trhu je Belimo průkopníkem v oblasti inovací. Příkladem toho je digitální inteligence pohonů, kdy se použitím mikročipů dají pohony individuálně nastavovat pro danou aplikaci a umožňují sběrnicovou komunikaci pomocí komunikačního protokolu MP-Bus (firemní protokol společnosti Belimo), o kterém bude pojednávat tato diplomová práce. Ovládání přitom zůstává stejně tak jednoduché jako u tradičních pohonů s konvenčním způsobem ovládání. [1]

O společnosti AMiT

Společnost AMiT spol. s r.o. je ryze český výrobce řídicích systémů a elektroniky pro průmyslovou automatizaci, který byl založen roku 1992 v Praze. Tato společnost disponuje vlastním vývojovým týmem schopným reagovat na individuální požadavky zákazníků a

vedle výroby standardních produktů se zabývá také zakázkovým vývojem a výrobou elektroniky. Hlavní část jeho produkce tvoří:

- kompaktní řídicí systémy
- modulární řídicí systémy
- průmyslové terminály
- moduly vzdálených vstupů a výstupů
- komunikační převodníky
- převodníky fyzikálních veličin
- zakázkový vývoj a výroba

Více než polovina aplikací jeho řídicích systémů je v oblasti tepelného hospodářství a energetiky, přibližně čtvrtina v řízení technologií a desetina řídicích systémů se uplatní při řízení strojů a zařízení. [2]

Ze stručného popisu obou společností je patrné, že se jejich produkty velmi často setkávají při aplikacích v oblastech jako je technické zabezpečení budov (TZB) a tepelné hospodářství. V té chvíli pak vzniká potřeba ovládání servopohonů firmy Belimo např. právě řídicími systémy firmy AMiT.

Toto ovládání lze samozřejmě řešit konvenčním způsobem z analogového či digitálního výstupu řídicího systému, ale je tu také druhá možnost, kterou je sběrnicové řízení pomocí komunikačního protokolu MP-Bus. A právě to stálo na počátku této práce a bylo důvodem pro navázání užší spolupráce mezi těmito dvěma společnostmi.

I. TEORETICKÁ ČÁST

1 SERVOPOHONY BELIMO UMOŽŇUJÍCÍ MP-BUS PROVOZ

Při popis servopohnů Belimo, o kterých pojednávají následující kapitoly, bylo čerpáno z materiálů [3]-[5].

Kromě konvenčního způsobu řízení umožňuje firma Belimo u svých produktových řad označovaných jako MFT (do této skupiny spadá i regulátor průtoku vzduchu NMV-D2M), MFT(2) a nově také MP, digitální řízení pomocí komunikačního protokolu MP-Bus. Pokud nebude uvedeno jinak budeme se v dalším textu zabývat právě těmito typy servopohonů.

Konvenčním způsobem řízení se zde rozumí spojitý provoz (základní výrobní nastavení), tříbodové ovládání, provoz otevřeno-zavřeno či PWM. Vybraný typ řízení se dá navolit pomocí ručního konfiguračního zařízení MFT-H nebo PC-Tool. V tomto „klasickém“ provozu jsou servopohony řízeny signálem 0..10 V DC a pohybují se do polohy zadané řídícím signálem. U všech konvenčních typů řízení je možné paralelní připojení dalších pohonů, je však třeba zajistit dostatečný příkon zdroje.

Přepnutí z konvenčního provozu na sběrnicový provoz se děje automaticky, jakmile je pohonu prostřednictvím MP-Busu přidělena MP adresa. Při sběrnicovém provozu dostávají pohony svůj digitální řídící signál z nadřazeného regulátoru přes sběrnici MP-Bus a pohybují se do zadané polohy. Prostřednictvím MP-Busu může být vzájemně propojeno až 8 zařízení (i různých) podporujících technologii MFT, MFT(2) či MP. Tyto zařízení jsou pak přímo připojeny na nadřazené systémy (DDC regulátory či řídicí systémy) nebo mohou být přes speciální uzly začleněny do jiných sítí (např. pomocí zařízení UK24LON do sítě LonWorks). Z různých regulačních článků se tak při MP-Bus provozu jednoduše stanou decentrální funkční jednotky. Toto řešení tak umožňuje vyšší uživatelský komfort, bezpečnost a zabezpečení, ale také snížení spotřeby energie díky optimálnějšímu provozu s možností synchronizace více zařízení, minimalizaci nákladů na údržbu, a také se velmi výrazně snižují náklady na kabeláž. Tento způsob řízení pak také umožnil integrovat do pohonů další funkce.

Tyto servopohony navíc také umožňují přímé připojení čidel a spínačů, kdy digitalizují analogové signály čidel a integrují je do MP-Busu, což dále eliminuje nároky na kabeláž. Servopohony MFT umožňují připojení aktivního čidla (s výstupem 0..32 V DC) nebo spínací kontakt. Servopohony MFT(2) a MP pak navíc umožňují i vyčítání pasivních odporových čidel. Další rozdíl oproti MFT je pak i možnost změnit dobu přeběhu pohonu. Servo-

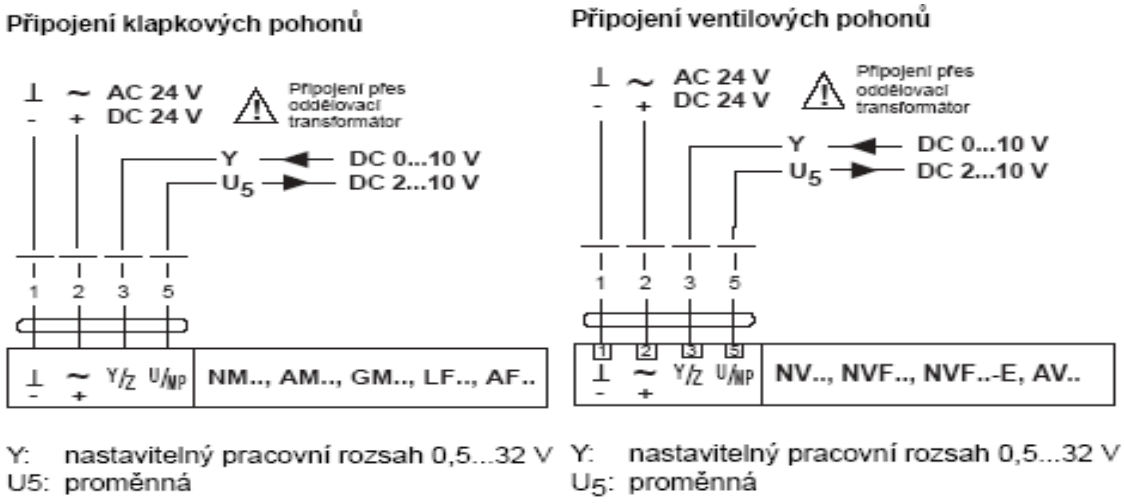
pohony MP mají oproti MFT a MFT(2) pohonům přímo na sobě servisní zásuvku pro připojení parametrizačního zařízení MFT-H, stavovou a napájecí LEDku a tlačítka pro adresaci a adaptaci.

Pohony mohou být z výroby dodány se standardním nastavením nebo naprogramované dle přání zákazníka. Kdykoliv pak může být nastavení změněno pomocí mobilního přístroje MFT-H nebo z počítače pomocí PC-Tool. Změnou parametrů jako je účinnost, doba přestavení, zpětné hlášení polohy, pracovní rozsah, elektrické omezení pracovního úhlu a kroutící moment/síla (vyjma pohonů s havarijní funkcí), lze jednotlivé pohony přizpůsobit požadované aplikaci. Pohony se navíc při uvádění do provozu sami adaptují, kontrolují za provozu, jsou chráněny proti přetížení a nepotřebují žádný koncový spínač (automaticky zůstanou stát na dorazu).

Zvláštní skupinou pohonů jsou pak pohony se zpětným pružinovým chodem (s havarijní funkcí), které uvádějí klapku do provozní polohy současně s natažením zpětné pružiny. To zabezpečí, že se při přerušení napájení klapka sama vrátí do bezpečné polohy. U zdvihových pohonů opatřených pružinou hřídel dle typu buď zajíždí nebo vyjíždí.

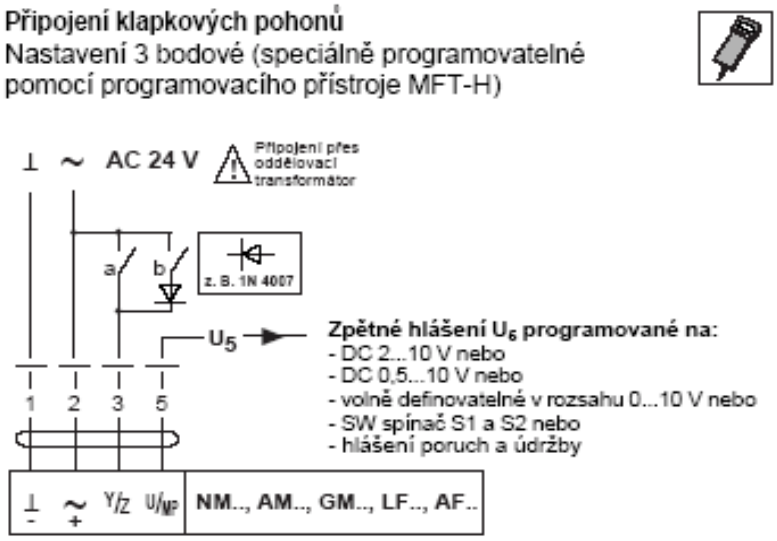
1.1 Konvenční způsoby řízení

U konvenčního spojitého řízení 0..10 V DC (základní nastavení z výroby) pomocí vstupu pohonu Y zadávám pracovní rozsah v rozmezí 0..10 V DC (nastavitelný pracovní rozsah je 0,5..32 V DC) a pomocí signálu U5 vyčítám z pohonu zpětné hlášení v rozmezí 2..10 V DC. Zapojení pro ventilové i klapkové pohony je shodné.



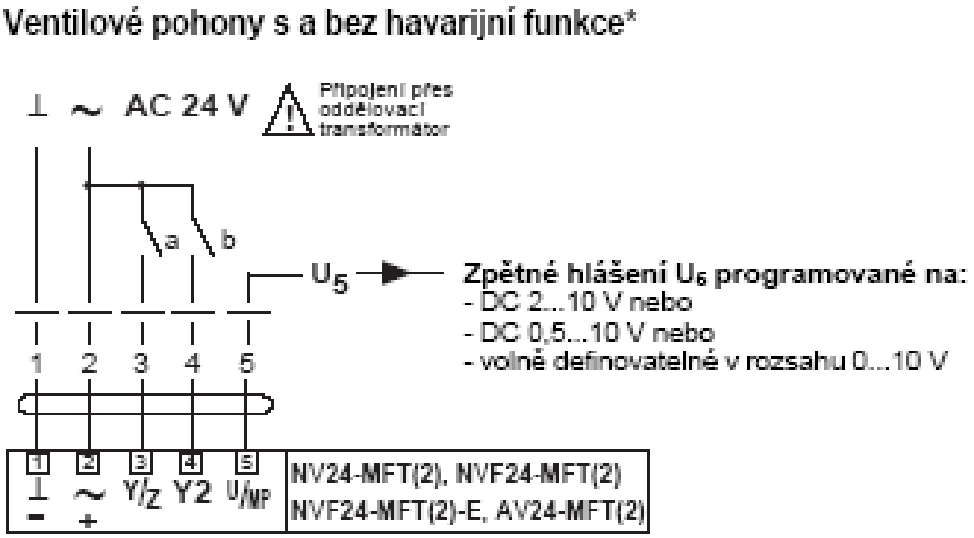
Obr. 1. Připojení pohonů při spojitém řízení

Tříbodové řízení musí být naprogramováno. U klapkových pohonů jde o jednovodičové připojení přes svorku Y s diodou. Vodič U5 slouží opět ke zpětnému hlášení a signál na tomto vodiči může být naprogramován na rozmezí 0..10 V DC, 0,5..10 V DC, volně definovaný v rozsahu 0..10 V, jako softwarový spínač S1 a S2 nebo jako hlášení poruch a údržby.



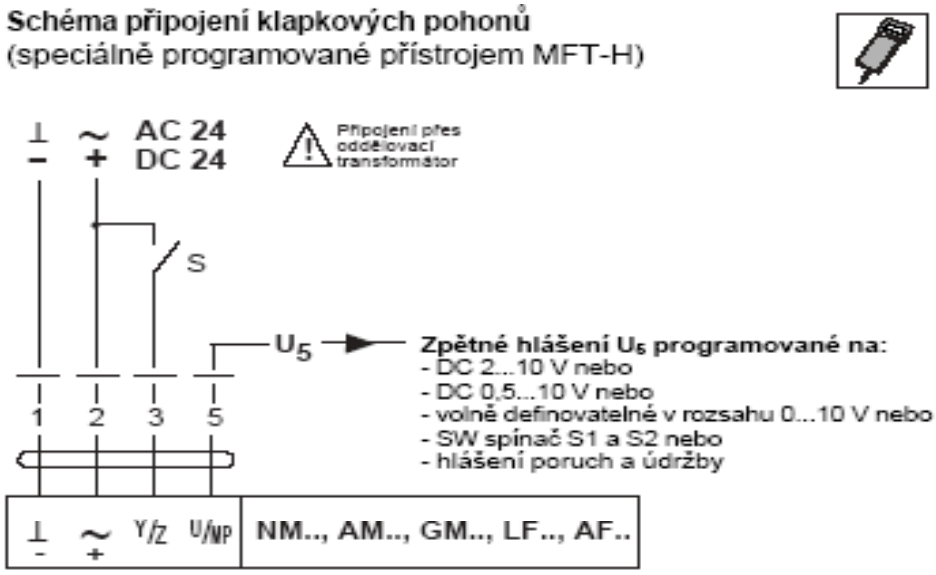
Obr. 2. Připojení klapkových pohonů při 3 bodovém řízení

U ventilových pohonů je potom možné i jednodušší čtyřvodičové připojení (je tu navíc vstup Y2) a zpětné hlášení U5 lze naprogramovat na 0..10 V DC, 0,5..10 V DC nebo jako volně definovatelné v rozsahu 0-10 V.



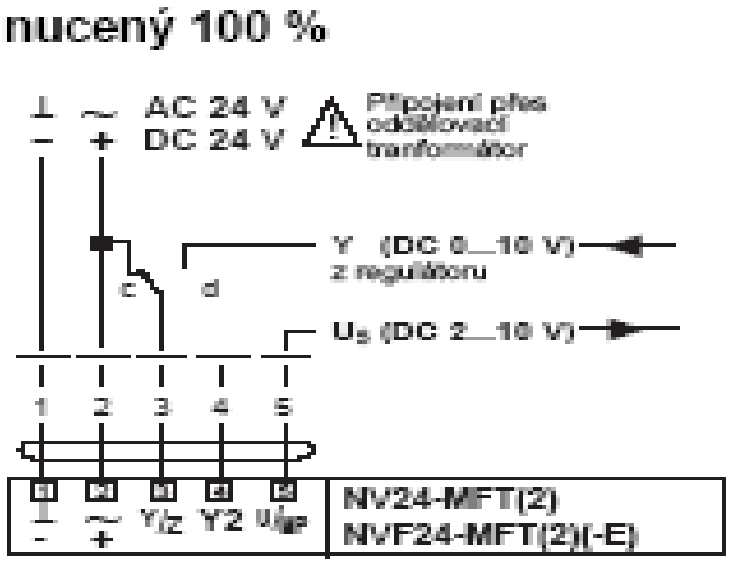
Obr. 3. Připojení ventilových pohonů při 3 bodovém řízení

Ovládání otevřeno-zavřeno musí být opět naprogramováno. U klapkových pohonů je přepínač umístněn na vodiči Y a signál na vodiči U5 nám dává zpětné hlášení v podobě 2..10 V DC, 0,5..10 V DC, volně definovatelné v rozsahu 0..10 V DC, jako softwarový spínač S1 a S2 nebo jako hlášení poruch a údržby.



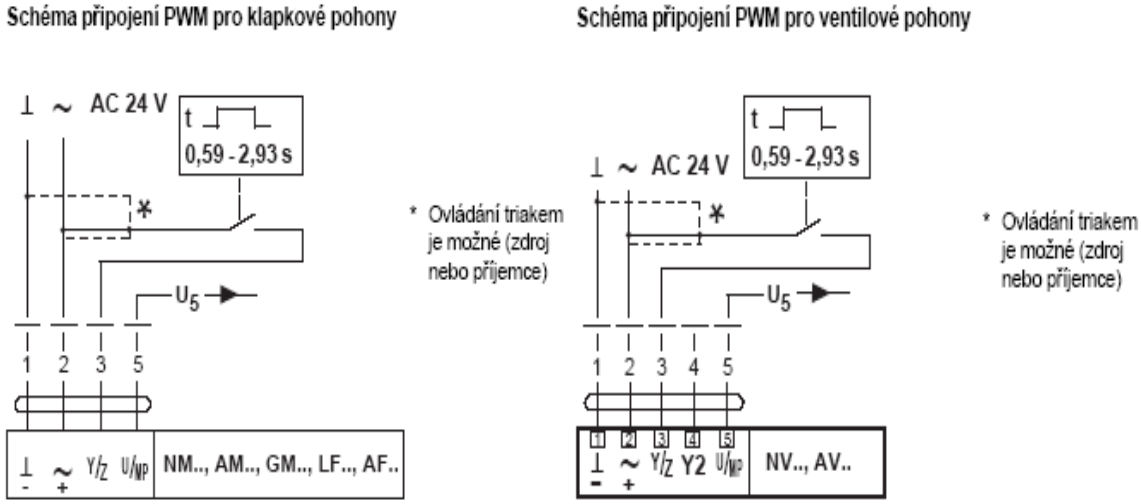
Obr. 4. Připojení klapkových pohonů při řízení otevřeno-zavřeno

U ventilových pohonů se jedná o tzv. nucené řízení 100%, které se používá např. pro zapínání protimrazové ochrany. Přepínač je umístněn na vodiči Y a na vodiči U5 je modulován signál se zpětným hlášením v rozmezí 2..10 V DC.



Obr. 5. Připojení ventilových pohonů při nuceném řízení

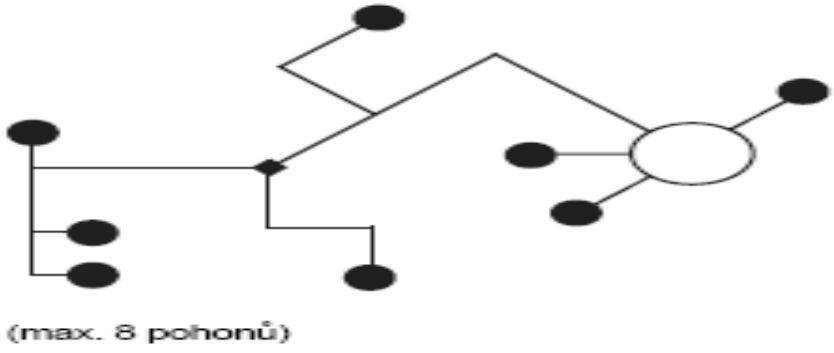
Ovládání pomocí PWM, které musí být opět naprogramováno, je pro klapkové i ventilové pohony stejné. U tohoto způsobu řízení se měří délka impulsu poslaného na pohon, který se na základě doby jeho trvání pohybuje do příslušné pozice. Řídicí signál jde opět na Y a povolené rozsahy PWM jsou s ohledem na použitý regulátor 0,02..5 s, 0,59..2,93 s, 0,1..25,5 s nebo s proměnným PWM mezi 0,02..50 s.



Obr. 6. Připojení pohonů při PWM řízení

1.2 MP-Bus řízení

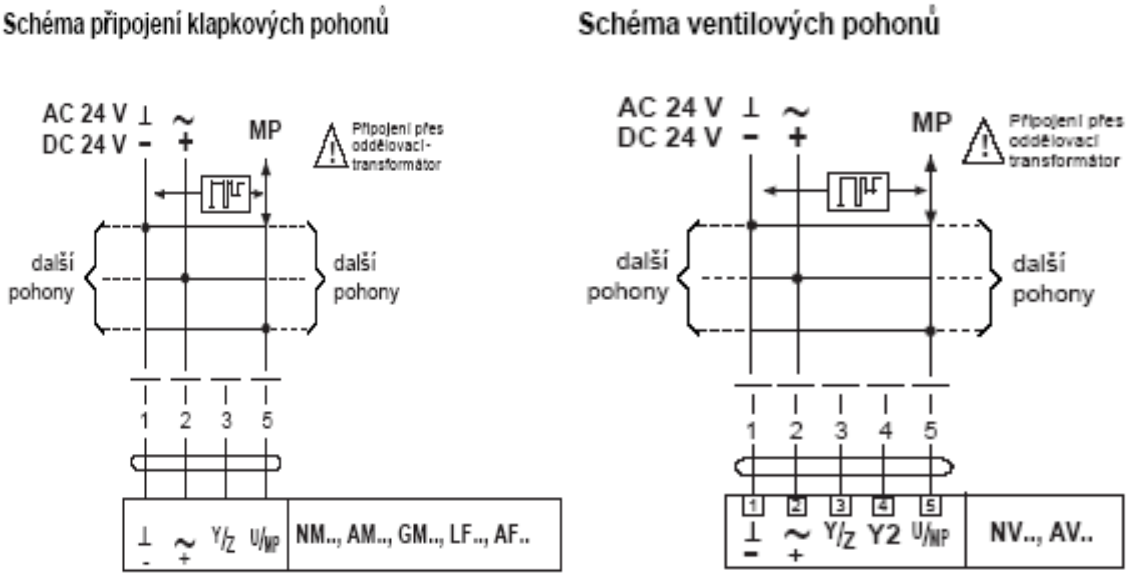
Až 8 pohonů může být vzájemně propojeno pomocí MP-Busu v jedné síti. Toto propojení může být 3 žilové při napájení přes sběrnici nebo 2 žilové při lokálním napájení. Nejsou zapotřebí žádné speciální kabely ani ukončovací odpory. Délka vodiče je limitována množstvím připojených pohonů, průřezem vodiče a způsobem napájení (DC či AC přes sběrnici nebo AC lokálně).



Obr. 7. *Povolené topologie sítě*

Pro topologii vodičů nejsou pro klapkové a ventilové pohony žádná omezení (tvary jako hvězdicové, stromečkové či kruhové jsou přípustné). Je možné také libovolně kombinovat různé typy pohonů v rámci jedné sítě.

Pohony dostanou svůj řídicí signál digitálně přes svorku U5 z nadřazeného MP-Bus masteru a pohybují se do zadané polohy. Zpětné hlášení pak opět probíhá přes svorku U5. Přepínání z konvenčního provozu na sběrnicový provoz se děje automaticky, jakmile je pohonu prostřednictvím MP-Busu přiřazena MP adresa v rozmezí 1..8.



Obr. 8. Schéma připojení pohonu k MP-Busu

Na každý MP-Bus příkaz reaguje pohon odpovědí. Průměrná doba od vyslání příkazu do přijetí odpovědi je cca 150 ms. Algoritmus tohoto cyklu musí stanovit výrobce digitálního regulátoru.

1.3 Připojení externích čidel nebo spínačů pro provoz MP-Bus

Na každý jeden pohon je možno připojit jedno čidlo (pasivní nebo aktivní) či spínací kontakt. Pohon pak vlastně slouží jako analog / digitální měnič pro zprostředkování signálu čidla přes MP-Bus do nadřazeného systému. Ten musí znát fyzickou adresu zařízení, na kterém je čidlo umístněno a jeho typ, aby mohl správně interpretovat jeho signál.

Signál z čidla je vyčítán na svorce Y a napojení čidla je doporučeno provést samostatným kabelem resp. minimálně základní vedení čidla udržet co nejdéle oddělené od napájení (pro

zamezení vyrovnávacích proudů). Pro pasivní čidla je pak potřeba zvolit co největší průřez vodiče (1..1,5 mm^2) neboť ohmický odpor vodiče ovlivňuje přesnost měření.

Pasivní čidla

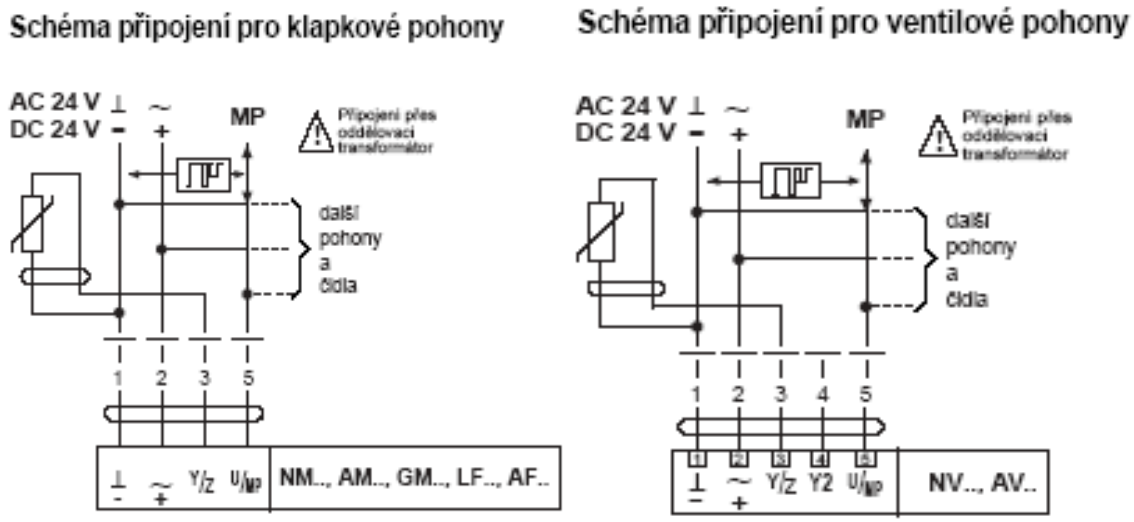
Měřící rozsahy pasivních čidel připojených na vstup pohonu Y ukazuje následující tabulka.

Tab. 1. Připojitelná pasivní čidla

Typ čidla	Měřitelné rozsahy teplot	Měřitelné rozsahy odporu
Ni1000	-28°C..98°C	850 Ω..1600 Ω
Pt1000	-35°C..155°C	850 Ω..1600 Ω
NTC (1k..10k vždy při 25°C)	-10°C..160°C	100 Ω..60 000 Ω

Přesnost měření pro Pt1000 a Ni1000 je ±0,3 %. U NTC se tolerance liší dle zvoleného rozsahu měření a typu NTC.

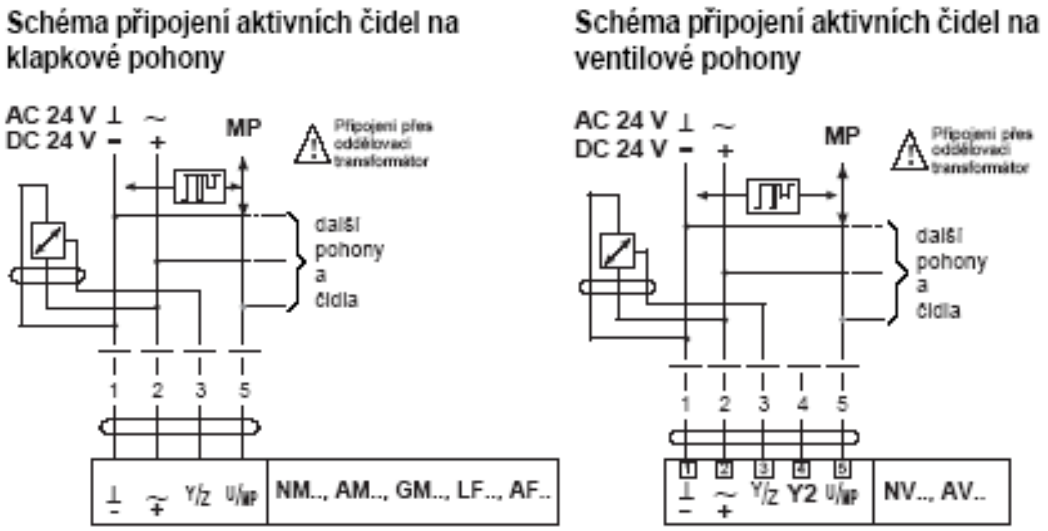
Pozn.: pohony řady MFT nepodporují vyčítání pasivních čidel.



Obr. 9. Schéma připojení pasivních senzorů

Aktivní čidla

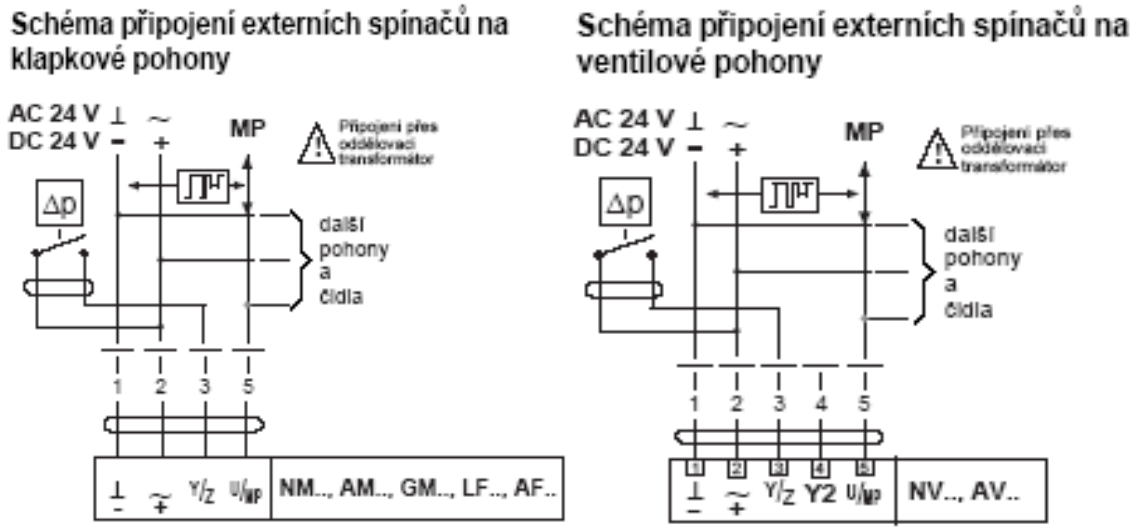
Je možno připojit např. čidla teploty či vlhkosti s výstupem 0..32 V DC. Rozlišení je typicky 30 mV.



Obr. 10. Schéma připojení aktivních senzorů

Spínací kontakt

Musí být schopen čistě spínat proud od 16 mA při 24 V. Bod startu pracovního rozsahu musí být naprogramován na hodnotu min. 0,6 V.



Obr. 11. Schéma připojení spínacího kontaktu

1.4 Adresování MP zařízení

1.4.1 Poloautomatické adresování

Každý pohon v MP-Bus musí být jednoznačně identifikovatelný. K tomu potřebuje mít vlastní jedinečnou adresu od 1 do 8.

Postup při poloautomatickém adresování MP s kvitováním:

1. Na MP-Bus masteru (např. UK24LON) se nastaví požadovaná MP adresa 1..8
2. MP-Bus master se uvede do stavu připravenosti k naadresování pohonu aktivací příslušné funkce (na UK24LON je to stisknutím tlačítka SET)
3. Provede se kvitace na pohonu, čímž je mu přidělena MP adresa, předem nadefinovaná na MP-Bus master

Postup při kvitování

U klapkových pohonů se dle typu buď stiskne tlačítko pro ruční ovládání (u typů bez havarijní funkce) nebo se přepínač L/R přepne během 5 s sem a tam (u typů s havarijní funkcí).

U ventilových pohonů se stiskne tlačítko S2. Požadavek na kvitování je zde signalizován blikající LED diodou (střídavě červená/zelená).

1.4.2 Adresování pomocí sériového čísla

Každý pohon je při výrobě opatřen štítkem s individuálním sériovým číslem, které je navíc uloženo také v paměti pohonu a je možné jej vyčíst příslušným MP příkazem. Toto jedinečné sériové číslo je pak možno využít k identifikaci zařízení a přidělení MP adresy.

Příklad sériového čísla: 09939-31234-064-008

Význam jednotlivých číslic :

09939 rok a týden výroby

31234 den v týdnu a pořadové číslo ve dni

064 typ pohonu

008 číslo zkušebny

Každý pohon má navíc jeden samolepící štítek se sériovým číslem, který slouží k nalepení do dokumentace, na místo kde byl fyzicky namontován. Tak lze zpětně vystopovat jeho umístnění.

Při uvádění do provozu je možné využít sériového čísla k aktivaci pohonu a přidělení MP adresy také pomocí parametrizačních nástrojů PC-Tool nebo MFT-H.

1.5 Základní poloha

Klapkové pohony

Po prvním zapnutí napájecího napětí, tedy při uvedení do provozu nebo po stisknutí tlačítka na pohonu, jede pohon do základní polohy. Po této operaci pak jede pohon do polohy zadané řídicím signálem.

Smysl otáčení může být změněn pomocí přepínače nebo PC-Tool či MFT-H, buď na levý doraz (poloha Y=0, pohon k ní jde proti směru hodinových ručiček) nebo na pravý doraz (poloha Y=0, pohon k ní jde po směru hodinových ručiček). Základní poloha tedy může být invertována oproti výrobnímu nastavení. Pokud se tak stalo pomocí přeprogramování konfiguračním zařízení, poloha nastavená na mechanickém přepínači nemusí odpovídat skutečnosti.

Klapkové pohony s havarijní funkcí

Po prvním zapnutí napájecího napětí si pohony nejprve automaticky určí svoji havarijní polohu (inicializace nulového bodu). Tato operace trvá cca 15 s a pohon po tuto dobu stojí.

Ventilové pohony

Viz následující kapitola o adaptaci.

1.6 Adaptace pracovního úhlu nebo zdvihu

Adaptace pracovního úhlu u klapkových pohonů

Adaptace neprobíhá automaticky, ale lze nastartovat pomocí PC-Tool nebo MFT-H. Stanoví se zde mechanický pracovní úhel (horní a spodní zarážka), který je uložen do procesoru. Doba přestavení a pracovní rozsah jsou přizpůsobeny MIN a MAX hodnotě, nastavené rozsahem regulace. Měřící signál U5 odpovídá efektivnímu mechanickému pracovnímu úhlu.

Adaptace může být vyvolaná i ručně. U klapkových pohonů bez pružiny se tak děje pokud se 2x stiskne tlačítko pro ruční ovládání a u pružinových pokud se během pěti sekund 2x přepne přepínač L-R a R-L.

Adaptace zdvihu u ventilových pohonů (u ventilů se dvěma mechanickými dorazy)

Po prvním zapnutí napájecího napětí je provedena automatická adaptace zdvihu. Tato adaptace je provedena mezi dvěma mechanickými zarážkami ve ventilu a tento zdvih je uložen do paměti jako 100% zdvihu. Řídicí signál i doba přestavení jsou pak přizpůsobeny tomuto 100% zdvihu. Tuto funkci lze také aktivovat tlačítkem S2, které je umístněno pod krytem pouzdra. Adaptaci lze také aktivovat pomocí PC-Tool nebo MFT-H.

Hlášení případných poruch lze kvitovat tlačítkem S2, po jehož stisknutí se provede adaptace.

Pozn.: u ventilů bez druhé mechanické zarážky lze efektivní zdvih stanovit softwarově. Adaptace pomocí tlačítka S2 je pak znemožněna (přesto je prováděn test synchronizace na uzavírací bod).

1.7 Pracovní rozsah

Základní pracovní rozsah je nastaven pro 2..10 V DC. Mezi těmito body je přímka, která odpovídá pracovnímu úhlu nebo zdvihu.

Pracovní rozsah jde i nastavit pomocí PC-Tool nebo MFT-H a může být také nastaven z výroby. Bod startu může být v rozmezí 0,5..30 V DC a koncový bod v rozmezí 2,5..32 V DC. Musí být ale splněna podmínka, že koncový bod je minimálně 2 V nad startovním.

1.8 Zpětné hlášení

Zpětné hlášení se provádí pomocí spojitého napěťového signálu na vodiči U5, který se pohybuje mezi 2..10 V DC při 0,5 mA. Závislost signálu U5 na pracovním úhlu nebo zdvihu v % je pak stejná jako pro pracovní rozsah 2..10 V DC.

Tento signál jde také nastavit pro bod startu mezi 0,5..8 V DC a koncový bod mezi 2,5..10 V DC s nutností dodržet rozdíl minimálně 2 V mezi koncovým bodem a bodem startu.

1.9 Hlášení údržby či poruchy

Signál na U5 může také přenášet hlášení potřeby údržby či poruchy. Lze definovat následující kritéria, která vydávají signálem U5 hlášení pro údržbu či poruchu:

- Poměr běhu a stání pohonu (Stop & Go ratio) – přílišné používání pohonu
- Mechanické přetížení - nebyla dosažena požadovaná poloha
- Dráha přestavení – dolní nebo horní poloha překročena o zvolenou hodnotu

Dle toho, zda je u uvedených kritérií definována údržba nebo porucha (pomocí PC-Tool nebo MFT-H), vydává výstup pohonu U5 příslušný příkaz:

- Výstupní úroveň při normálním provozu (bez hlášení poruchy či údržby): 3 V DC
- Výstupní úroveň pro hlášení údržby: 5,5 V DC
- Výstupní úroveň pro hlášení poruchy: 8,5 V DC

Podmínka fungování hlášení poruch a údržby u klapkových pohonů je, že u mechanicky omezeného pracovního úhlu (<95º) musí být nejprve provedena jeho adaptace. U ventilových pohonů svítí při aktivním hlášení poruch červená kontrolka LED pod krytem pouzdra a poruchu lze kvitovat pouze adaptací, která se vykoná po stisknutí tlačítka S2.

Hlášení poruch a údržby mohou být přenášena také pomocí MP-Busu příslušnými příkazy.

1.10 Softwarový spínač

Signál U5 může fungovat také jako softwarový spínač. V tomto případě bude signál U5 rozkládán na tři různé úrovně napětí, které budou signalizovat stav aktivace dvou volitelných softwarových spínačů S1 a S2. Ty lze nastavit v rozsahu 1% až 99% pracovního úhlu nebo zdvihu. Poloha S1 musí být minimálně o 10% menší než S2. Tyto hodnoty se opět nastavují pomocí PC-Tool nebo MFT-H, případně také MP příkazy.

1.11 Smysl otáčení a směr zdvihu (volba uzavíracího bodu)

U všech klapkových pohonů se dá měnit smysl otáčení pomocí přepínače R-L.

Směr zdvihu jde nastavit u všech ventilových pohonů pomocí MFT-H a PC-Tool. U typů NV a NVF jde navíc měnit nastavení pomocí posuvných spínačů S3.1 (směr zdvihu) a S3.2

(volba uzavíracího bodu). Nastavení směru zdvihu a volba uzavíracího bodu pomocí posuvných spínačů S3 je podrobněji popsána v (Tab. 2) v kapitole 1.17.

1.12 Doba přestavení

Doba přestavení nastavená z výroby je pro klapkové pohony bez havarijní funkce 150 s. U pohonů s havarijní funkcí je to 150 s pro motorické přestavení a cca 20 s pro přestavení pružinou při teplotě mezi –20..50 ºC a max. 60 s při –30 ºC.

Doba přestavení lze měnit pomocí PC-Tool či MFT-H. Je však třeba upozornit, že se tím změní kroutící moment resp. řídicí síla a hladina hluku. Povolená nastavení jsou v rozmezí 75..300 s (u typu GM to je 120..300 s).

U ventilových pohonů je z výroby nastaveno 150 s pro typy NV a NVF. Pro zdvih 10(20) mm, pak lze nastavit dobu přestavení 55(95)..1200(2200) s. U typu AV je pak nastaveno 320 s a tato hodnota lze měnit v rozmezí 170..800 s.

1.13 Síly a momenty

Pro klapkové pohony bez havarijní funkce lze kroutící moment omezit na 75 %, 50 % či 25 %. Na stejné hodnoty lze omezit i přestavnou sílu u ventilových pohonů bez havarijní funkce. U pohonů s havarijní funkcí nelze kroutící moment resp. přestavnou sílu měnit.

1.14 Omezení pracovního úhlu a zdvihu

Pracovní úhel u klapkových pohonů je max. 95º a dle příslušného typu pohonu ho lze omezit mechanicky pomocí vestavěných zarážek nebo pomocí příslušenství ZDB-xx. Možné omezení se dle typu pohonu pohybuje v rozmezí 20..100 %.

Omezení zdvihu u ventilových pohonů pak lze provést pomocí vestavěných mechanických zarážek.

Pracovní úhel či zdvih lze omezit také elektronicky pomocí konfiguračních nástrojů tak že:

- poloha max (koncový bod elektrického pracovního rozsahu) může být nastaven mezi 0..100 % z celého rozsahu
- Poloha min (bod startu elektrického pracovního rozsahu) může být 0..100 % z max

- Poloha mid (střední poloha mezi 0 % = min, 100 % = max) musí být mezi 0..100 % z regulačního rozsahu (min..max)

Omezení pracovního úhlu se hojně využívá u zapojení pohonů s nuceným řízením.

Upozornění: pro správnou interpretaci zpětného hlášení na U5 po mechanickém omezení pracovního úhlu je potřeba nejprve provést adaptaci.

1.15 Ruční přestavení

Klapkové pohony

Ruční přestavení u klapkových pohonů bez havarijní funkce se provádí samovratným tlačítkem, kdy po dobu jeho stisknutí je z činnosti vyřazeno převodové ústrojí.

U klapkových pohonů s havarijní funkcí lze změnit pracovní úhel pouze u typu AF. Tato změna se provádí pomocí ruční kličky a klapku lze fixovat v kterékoliv poloze. Odblokování se pak provádí manuálně nebo automaticky vložením napájecího napětí.

Ventilové pohony

Ruční přestavení u ventilových pohonů typu NV a NVF se provádí na pohonu pomocí imbus klíče. Toto ruční přestavení je jištěno proti přetížení. Hřídel zdvihu pak zůstává v manuální poloze do doby než je pohon napojen na napájecí napětí.

1.16 Zdvih a citlivost

Normální nastavená citlivost z výroby je pro klapkové pohony 1º, ale může být přeprogramována (utlumena) pomocí PC-Tool či MFT-H na 2º. Vratná hysterze je pak nastavena na 2,5º a jde změnit na 5º.

Jmenovitý zdvih u ventilových pohonů je 20 mm (40 mm pro typ AV) a dá se nastavit mezi 10..20 mm (20..40 mm pro typ AV). Normální citlivost je u těchto pohonů 0,2 mm a jde změnit (utlumit) na 0,4 mm. Vratná hysterze je nastavena na 0,5 mm a jde změnit na 1 mm.

1.17 Popis ventilových pohonů

Pod krytem pohonu se nalézají svorky pro kabelové připojení, ovládací elementy S1, S2, S3 a kontrolka LED H1. Řídící signál je zpracováván mikroprocesorem a přes driver dále předáván bezkomutátorovému motoru BÜLOMO. Nastavením posuvného přepínače S3

nebo stisknutím tlačítka S1 a S2 lze pohon jednoduše přímo na místě přizpůsobit potřebám dané aplikace (při změně výrobního nastavení).

Tab. 2. Popis funkcí ovládacích prvků S

S1 Tlačítko test	Ventil projíždí zdvih při maximální době přestavení a kontroluje adaptovaný zdvih, zda obě dvě koncové hodnoty (0 % a 100 %) byli dosaženy
S2 Adaptace	Možný vyjetý zdvih (mezi dvěma mechanickými zarážkami ve ventilu) je stanoven jako 100 % a uložen v mikroprocesoru. Řídicí signál a doba přestavení budou tomuto zdvihu přizpůsobeny
S3.1 Směr zdvihu	Směr zdvihu je vůči řídicímu signálu invertován
Poloha Off (výrobní nastavení)	0 % řídicího signálu odpovídá 0 % zdvihu = 0 % U5
Poloha On	100 % řídicího signálu odpovídá 0 % zdvihu = 0 % U5
S3.2 Volba uzavíracího bodu	Uzavírací bod vzniká při vyjeté či zajeté hřídeli zdvihu. Měřící signál ve zvoleném uzavíracím bodě odpovídá 0 %
Poloha Off (výrobní nastavení)	uzavírací bod hřídele zdvihu zajeté do pohonu
Poloha On	uzavírací bod hřídele zdvihu vyjeté z pohonu

Tab. 3. Kontrolka LED H1

Zeleně svítí	Pohon pracuje bezchybně
Zeleně bliká	Běží test nebo adaptace se synchronizací
Červeně svítí	Porucha. Bude provedena nová adaptace
Červeně bliká	Po každém výpadku napětí (>2 s). Při dalším uzavírání ventilu bude automaticky ve zvoleném uzavíracím bodě provedena synchronizace a LED přejde z červeného blikání na trvale zelené světlo
Střídavě červené/zelené světlo	Adresování přes řídicí sytém a stisknutí tlačítka S2

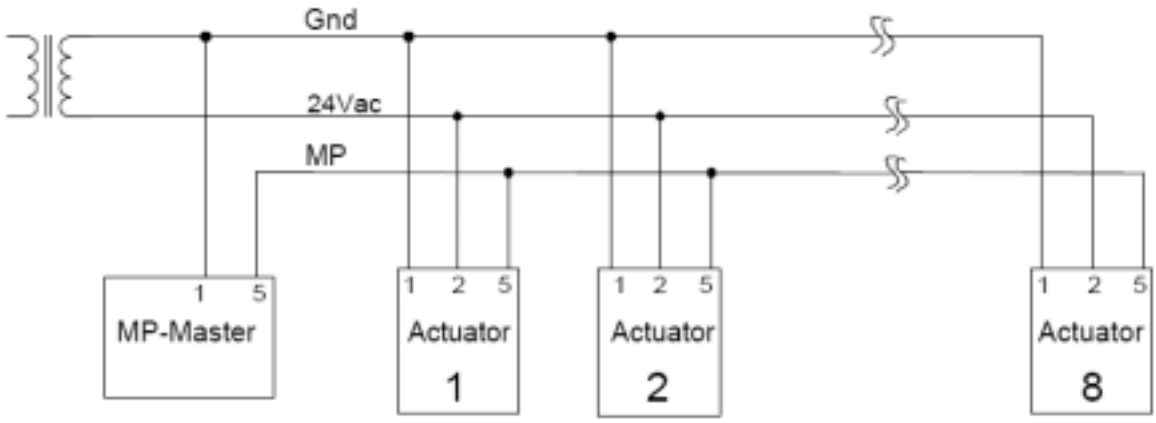
2 MP-BUS

Informace o tomto protokolu byli čerpány z dokumentu [6].

2.1 Stručný popis

- Určeno pro jeden master (např. konfigurační nástroj či řídicí systém) a 1…8 slavů (např. servopohony)
- Sériová master-slave komunikace; slave pouze odpovídá na příkazy mastera
- Datový přenos je obousměrný, poloduplexní, signál je modulovaný na vodiči „U5“ a vztažený vůči zemi
- Parametry komunikace: 1200 Bd, 1 start bit, 8 datových bitů, 1 stop bit, bez parity
- Komunikační protokol je chráněn kontrolním součtem (16 bitů)
- Komunikační protokol je připraven na budoucí rozšíření díky rezervovaným adresám
- Dva komunikační módy:
    - PP mód (Point-to-Point): 1x master, 1x slave, pouze propojení (není to sběrnicový mód) bez adresace
    - MP mód (Point-to-Multipoint): 1x master, až 8x slave, sběrnicový mód s adresami pro každý slave
- Pokud se nekomunikuje, může být v PP módu analogový signál na vodiči „U5“ stále aktivní (např. 0..10 V). Během komunikace je analogový signál přerušen
- V MP módu se vstup „U5“ chová jako digitální a analogový signál nepodporuje.
- V MP módu se rozlišují tři způsoby komunikace:
    - Addressed: Adresovaný slave provede a odpoví na příkaz okamžitě
    - Broadcast: Každý slave na síti provede příkaz od mastera, ale neodpovídá na něj
    - OnEvent: Příkaz vykoná a odpoví na něj pouze slave detekující definovanou událost

2.1.1 Příklad MP sítě



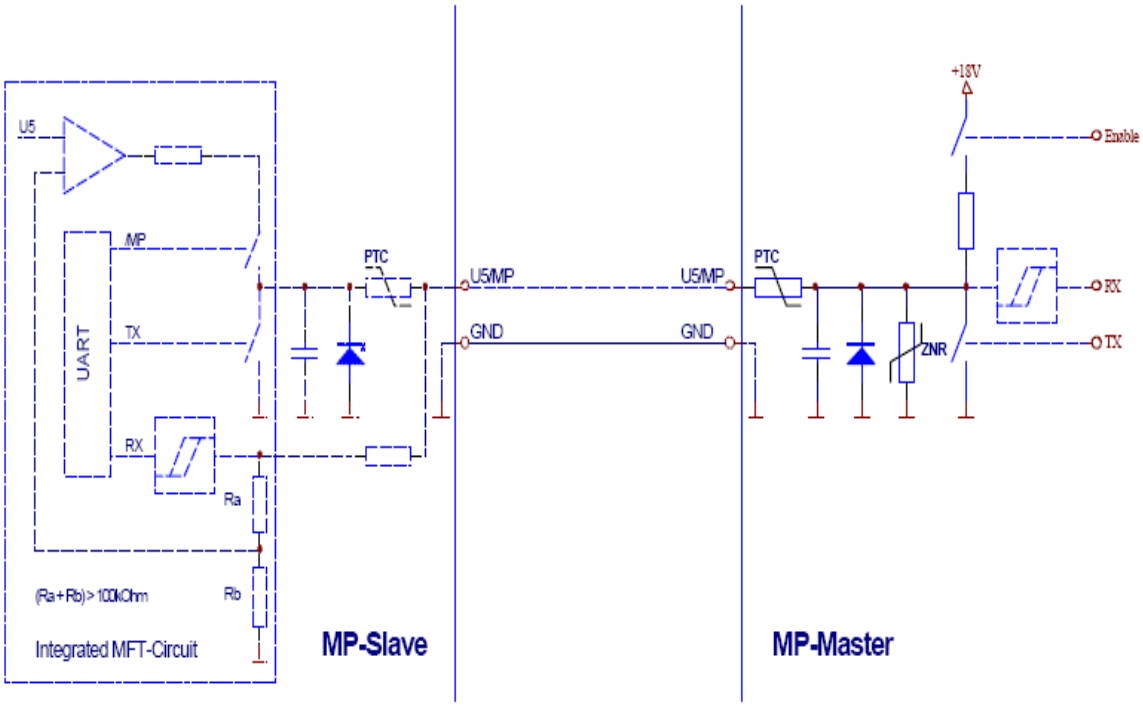
Obr. 12. Příklad MP sítě

Napájení: 24 V DC/AC; při napájení pomocí 24 V DC může být vedení delší.

MP-Master: např. řídicí systémy, UK24-LON.

Akční členy: max. 8 zařízení podporujících komunikaci MP-Bus.

2.2 Popis hardwarové vrstvy



Obr. 13. Principielní schéma zapojení MP-Busu

Obousměrný poloduplexní přenos probíhá na vodiči U5, který je normálně používán jako analogová zpětná vazba např. na určení aktuální pozice či aktuálního průtoku vzduchu. Pokud je slave v PP módu, může být tato funkce stale využívána. Pokud je ovšem přenášen příkaz, analogový signál je přerušen. Když je slave v MP módu, musí být analogový výstup ve vybuzeném stavu, jinak by proudová zátěž na sběrnici byla příliš vysoká. Analogový signál pak není vůbec podporován

2.3 Detailní popis komunikačního protokolu

Protokol využívá tři vrstvy OSI: první, druhou a sedmou. Toto rozdělení ovšem ne zcela odpovídá dělení dle OSI.

Tab. 4. Komunikační vrstvy

Komunikační vrstva	**Hodnoty funkcí**	**Příklady funkcí**
první vrstva (fyzická)	Bit	Softwarová implementace sériového rozhraní, rozpoznání start bitu a stop bitu, přenos signálu po lince
druhá vrstva (protokol)	Byte	Generace a kontrola start bytu, hlídání timeotu, generace a kontrola příčné i podélné parity
sedmá vrstva (aplikační)	Příkaz	Vykonávání příkazů a generace odpovědí

2.3.1 Specifikace první vrstvy

Komunikační parametry

Komunikační rychlost: 1200 Bd

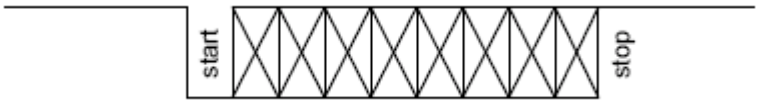
Parita: žádná

Struktura: 1 start bit, 8 datových bitů, 1 stop bit

Pořadí bitů: LSB první

Struktura rámce

Rámec se skládá ze start bitu, datových bytů a stop bitu.



Obr. 14. Struktura rámce

hodnota 0 = dolní úroveň napětí (vztažená k zemi napájení)

hodnota 1 = horní úroveň napětí

2.3.2 Specifikace druhé vrstvy

Datový tok z masteru na slave se nazývá příkaz a tok v opačném směru odpověď. Slave nikdy nepošle odpověď bez předešlého obdržení příkazu od mastera.

Příkazy a odpovědi jsou přenášeny jako tok bytů bez mezer mezi nimi. To znamená, že čas mezi stop bitem předcházejícího bytu a start bitem následujícího bytu musí být menší než $\mathbf{t_{gap}}$. Po resetu nebo chybě komunikace musí slave začít odpočítávat timeout odpovídající času nejméně 10 bitů, aby měl jistotu, že rozpozná následující platný start byte.

2.3.2.1 Komunikační módy

Jak bylo napsáno výše, existují dva komunikační módy: PP (Point-to-point) a MP (Point-to-Multipoint). Komunikační mód zařízení je dán nastavenou (naprogramovanou) adresou. Jestliže nastavená adresa není PP adresa, zařízení se automaticky přepne do MP módu. Existují tři způsoby komunikace v MP módu: Addressed, Broadcast a OnEvent. Tyto komunikační typy jsou definovány pomocí start byte kódu příkazu posílaného mastrem.

PP mód

Základní komunikační mód všech slavů od Belima s výrobním nastavením. Příkazy poslané v tomto módu budou vykonány a bude na ně zodpovězeno kterýmkoliv slavem, dokonce i když má nastavenu MP adresu. Také starší typy slavů, které nepodporují MP-Bus komunikaci, vykonají a odpoví na takovýto příkaz. Pokud neprobíhá komunikace a slave je v PP módu, vodiči U5 může být využíván jako analogová zpětná vazba.

MP mód

Tento komunikační mód je možný pouze u servopohonů Belimo nové generace řady MFT, MFT(2) a MP. Komunikace se přepne do MP módu po nastavení MP adresy na slavu. K tomu slouží např. speciální příkaz PP_Set_MP_Address. Nastavení PP adresy na slavu způsobí přepnutí zpět do PP módu. V MP módu jsou podporovány následující módy:

- Addressed komunikace: Adresovaný slave vykoná a odpoví příkaz okamžitě. Ostatní slavy ho obdrží také, ale nevykonají ho a ani na něj neodpoví. Zařízení, která nemají implementován MP-Bus nemohou vykonat ani odpovědět na takovýto příkaz.
- Broadcast komunikace: Slavy s implementovaným MP-Busem neodpovídají na tuto komunikaci, ale rovnou vykonají příkaz (pokud můžou). Není tedy možné využít příkazy pro vyčítání a není možné ani ověření správného přijetí příkazu. Slavy bez implementovaného MP-Busu na broadcast příkazy neodpovídají.
- OnEvent komunikace: Slavy s implementovaným MP-Busem vykonají a odpoví na příkaz, pokud detekují definovanou událost. Ta závisí na příkazu a musí být definovaná v seznamu příkazů. Tento mód je v současnosti použit při adresovací procedůře. Slavy bez implementovaného MP-Busu na OnEvent příkazy neodpovídají.

Poznámka: Protože nejde ověřit správné doručení příkazu při Broadcast a OnEvent komunikaci, je doporučeno příkazy dvakrát až třikrát opakovat.

2.3.2.2 Příkaz slavu

Délka příkazu: min. 4 byty, max. 10 bytů

Použitelné byty: max. 7 bytů

Tab. 5. Struktura příkazu

Počty bytů	1	1	0..6 příp. 1..6 při rozšířeném příkazu	1	1
	stb	cc	Parametry	cp	lp

Počty bytů 4..10 příp. 5..10 při rozšířeném příkazu

PP nebo MP příkazový paket

Tab. 6. Popis struktury příkazu

Byte	**Zkratka**	**Význam**	**komentář**
1	Stb	start byte	viz tabulka popisující start byty (Tab. 7)
2	Cc	kód příkazu (1.užitečný byte)	viz tabulka Seznam MP příkazů (Tab. 16)
3		Parametr (2.užitečný byte)	první parametr či rozšíření kódu příkazu
...		...	...
N+1		Parametr (N-tý užitečný byte)	poslední parametr
N+2	Cp	příčná parita	paritní bity z předešlých bytů
N+3	Lp	podélná parita	EXOR všech předešlých bytů

Start byte příkazu

Tab. 7. Start byte kódy (budou rozpoznány i slavem nepodporujícím MP mód)

Mód	**Adresa**	**Start byte**							
		Bit 7; MSB	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0; LSB
PP	--	0	N	N	N	0	0	0	1
MP	Broadcast	0	N	N	N	1	0	0	1
	OnEvent	1	N	N	N	0	0	0	1
	BusMaster	1	N	N	N	0	1	0	1
PP	Odpověď	x	x	x	x	0	0	0	1
MP	Odpověď	x	x	x	x	1	1	0	1
Rezervováno pro budoucí použití. Nepoužívat!		0	x	x	x	0	1	0	1
		1	x	x	x	1	0	0	1

N: počet užitečných bytů následujících po start bytu (vyjma paritních) v binárním kódu

x: nemá význam

Tab. 8. Doplňující start byte kódy (budou rozeznány jen slavy podporujícími MP mód)

Mód	**Adresa**	**Start byte**							
		Bit 7; MSB	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0; LSB
MP	1	0	N	N	N	0	0	0	0
	2	0	N	N	N	0	1	0	0
	3	0	N	N	N	1	0	0	0
	4	0	N	N	N	1	1	0	0
	5	1	N	N	N	0	0	0	0
	6	1	N	N	N	0	1	0	0
	7	1	N	N	N	1	0	0	0
	8	1	N	N	N	1	1	0	0
Rezervováno pro budoucí použití! Nepoužívat!		X	x	x	x	x	x	1	0
		X	x	x	x	x	x	1	1

N: počet užitečných bytů následujících po start bytu (vyjma paritních) v binárním kódu

x: nemá význam

Příkazy a parametry

První použitelný byte po start bytu obsahuje příkazový kód označený číslem mezi 0..255. Kód 255 je rezervován pro rozšíření příkazů, jestliže by 255 příkazů nebylo dostačujících. Seznam většiny příkazů je uveden v (Tab. 16). Pro starší akční členy NMV-D2 a AM24-M1 je k dispozici speciální seznam příkazů.

Příčná parita

Příčná parita se počítá ze sudé parity start bytu, příkazového bytu a všech bytů parametrů. Paritní bity se vkládají od MSB příčně paritního bytu. Jeho inicializační hodnota je 0.

Podélná parita

Je přenášen na konci každého telegramu. Je exklusivním součtem všech předešlých bytů.

Příklad výpočtu paritních bytů

Příklad odezvy na příkaz s kódem 00, poslaný na MP adresu 1, s hexa parametrem AA:

Tab. 9. Výpočet paritních bytů

Označení bytu	**Hexadecimální hodnota**	**Binární hodnota**	**Sudá parita**
start byte	20h	00100000 →	1
kód příkazu	00h	00000000 →	0
Parametr	AAh	10101010 →	0
Příčná parita	20h	**001**00000 ←	
podélná parita	Aah	**10101010**	

2.3.2.3 Odpověď od mastera

Slave generuje odpověď pouze pokud obdrží správný příkaz poslaný na jeho PP nebo MP adresu. Stejně jako příkaz, tak i odpověď obsahuje start byte, příčnou paritu a podélnou paritu. V případě že se jedná o příkaz, kterým master nepožaduje zaslat zpět parametr, posílá slave zpět alespoň start byte spolu s příčnou a podélnou paritou. Takto slave oznámí masterovi, že správně obdržel poslední příkaz, a že je nyní připraven obdržet další.

Délka odpovědi: min. 3 byty, max. 10 bytů

Použitelné byty: min. 0 bytů, max. 7 bytů

Tab. 10. Struktura odpovědi

Počty bytů	1	0..7	1	1
	stb	Parametry	cp	lp

Počty bytů 3..10

PP nebo MP odpovědní paket

Tab. 11. Popis struktury odpovědi

Byte	**Zkratka**	**Význam**	**Komentář**
1	Stb	start byte	viz tabulka popisující start byte (Tab. 12)
2		parametr (1.užitečný byte)	první parametr
...		...	...
N+1		parametr (N-tý užitečný byte)	poslední parametr
N+2	Cp	příčná parita	paritní bity z předešlých bytů
N+3	Lp	podélná parita	EXOR všech předešlých bytů

Start byte odpovědi

Nové pohony podporující komunikaci po MP-Busu mají jiné odpovědní start byte kódy něž starší pohony bez podpory MP-Bus (NMV-D2, AM24-M1).

Tab. 12. Start byte kódy pro slave

MP zařízení	**Obsah**	**Start byte**							
		Bit 7; MSB	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0; LSB
Ne	parametr	0	N	N	N	0	0	0	1
Ne	chybový kód	1	N	N	N	0	0	0	1
Ano	parametr	0	N	N	N	1	1	0	1
Ano	chybový kód	1	N	N	N	1	1	0	1

N: počet užitečných bytů následujících po start bytu (vyjma paritních) v binárním kódu

Parametry odpovědi

Normálně, pokud se neobjeví chyby komunikace, sestávají parametry odpovědi z hodnot jako je teplota, doba přeběhu apod.. V případě chyby příkazu (chyby na sedmé vrstvě) bude odpovědní start byte obsahovat chybový kód. V prvním bytu parametru pak bude uveden typ chyby. Více v tabulce s kódy chyb (Tab. 13) .

Příčná a podélná parita

Parity jsou generovány stejně jako na straně mastera. Viz kapitola 2.3.2.2.

2.3.2.4 Reakce na chybu

Chyby na první vrstvě

Chyby komunikace na první vrstvě jsou typicky chyby rámce způsobené chybou na vodiči, úroveň napětí mimo specifikaci, časováním bitů mimo specifikaci, chybějícím stop bitem, méně nebo více než osmi datovými bity.

Reakce na chybu je, že slave nepošle odpověď. Kritérium pro vyhodnocení chyby je maximální timeout odpovědi $\mathbf{t_{amax}}$. V tomto případě musí master opakovat telegram.

Chyba na druhé vrstvě

Chyby komunikace na druhé vrstvě jsou typicky chyby kontrolního součtu způsobené změnou hodnoty bitu vlivem rušení přenosu, chybou v generátoru parity u masteru, příkazem s více než sedmi užitečnými byty, skutečným počtem bytů jiným než specifikovaným ve start bytu, chybným start bytem, špatnou MP adresou.

Kritérium pro vyhodnocení chyby je maximální timeout odpovědi $\mathbf{t_{amax}}$. V tomto případě musí master opakovat telegram.

Chyba na sedmé vrstvě

Chyby komunikace na sedmé vrstvě jsou typicky chyby příkazu způsobené zamítnutým přístupem (chybějící přihlášení), neznámým příkazem, chybou v průběhu vykonávání příkazu slavem.

Reakce na chybu je, že slave pošle odpověď, kde ve start bytu je nastaven chybový bit a jako parametr je poslán jednobytový chybový kód. V případě odmítnutí přístupu se master musí nejprve přihlásit, a až poté poslat příkaz znovu. V případě neznámého příkazu je nutno zkontrolovat, zda je připojen správný typ pohonu.

Tab. 13. Kódy chyb na sedmé vrstvě

Kód chyby	**Chyba**
0 – 10	Rezervováno

11	Neznámý příkaz
12	Příkaz nepovolen (např. z důvodu nepřihlášení)
13	Chyba v průběhu vykonávání příkazu
14 – 255	Rezervováno pro budoucí použití

2.4 Elektrické charakteristiky a požadavky na časování

PP nebo MP linka je chráněna proti zkratu k GND nebo 24 V AC/DC.

Obecné podmínky měření

Následující tabulky jsou platné pro MP-Bus složený z 1 masteru a 8 slavů, kteří jsou na master připojeni napřímo při použití standardní kabeláž o délce 1 m a průřezu 0,75 mm^2. Parametry jsou měřeny přímo na konci vodiče U5 připojeného zařízení. Není-li uvedeno jinak, zařízení je připojeno do MP-Bus.

Tab. 14. Požadavky na časování

Parametr	Značka	Min	Typicky	Max	Jednotka
Čas potřebný pro přenos bitu	$\mathbf{t_{bit}}$	846	833,3	821	µs
Čas potřebný pro přenos rámce	$\mathbf{t_{Fr}}$	8,46	8,33	8,21	ms
Budící čas	$\mathbf{t_r}$	60	80	200	µs
Čas poklesu	$\mathbf{t_f}$	2	60	80	µs
Čas mezi dvěma byty	$\mathbf{t_{gap}}$			5	ms
Master timeout / opakovací interval příkazu[1)]	$\mathbf{t_{amax}}$	600			ms
Zpoždění odpovědi[2)]	$\mathbf{t_{admin}}$	$\mathbf{t_{Fr}}$			ms
Zpoždění příkazu[3)]	$\mathbf{t_{cdmin}}$	$\mathbf{t_{Fr}}$			ms

[1)] minimální čas od startu prvního bytu příkazu do rozpoznání že schází odpověď a opakování příkazu

[2)] čas mezi posledním stop bitem příkazu a start bitem start bytu odpovědi

[3)] čas mezi stop bitem odpovědi a start bitem následujícího start bytu příkazu

Tab. 15. Elektrické charakteristiky

Parametr	Značka	Min	Typicky	Max	Jednotka
Signálové úrovně na MP-Busu					
Horní úroveň napětí pro příkaz a odpověď	$\mathbf{V_h}$	11	12,5	18	V
Dolní úroveň napětí pro příkaz	$\mathbf{V_{lc}}$		1	2,5	V
Dolní úroveň napětí pro odpověď	$\mathbf{V_{la}}$		4	4,5	V
Specifikace pro master					
Napájení mastera	$\mathbf{V_{ms}}$	17	18	19	V
Zkratovací proud mezi U5 a GND[1)]	$\mathbf{I_{hmax}}$			10	mA
Kapacitní odpor[2)]	**C**		10	12	nF
Dolní práh	$\mathbf{V_{thml}}$	7	8	8,5	V
Horní práh	$\mathbf{V_{thmh}}$	8,5	9	9,5	V
Hysterze	$\mathbf{V_{hystm}}$	0,8	1	1,5	V
Specifikace pro slave					
Elektr. odpor v MP módu bez komunikace[3)]	$\mathbf{R_{is}}$	90	100		kΩ
Kapacitní odpor mezi U5 a GND[4)]	$\mathbf{C_{u5}}$		1	1,2	nF
Dolní práh[5)]	$\mathbf{V_{thsl}}$	4,5	5	5,5	V
Horní práh[5)]	$\mathbf{V_{thsh}}$	5,5	6	6,5	V
Hysterze	$\mathbf{V_{hysts}}$	0,8	1	1,5	V

[1)] horní úroveň pro samotný master

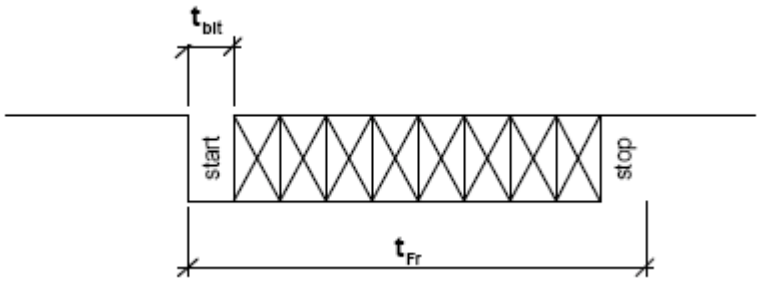
[2] samotný master

[3] jeden slave při 12,5 V. Slavy založené na technologii X-Bela mají minimální vstupní odpor 50 kΩ

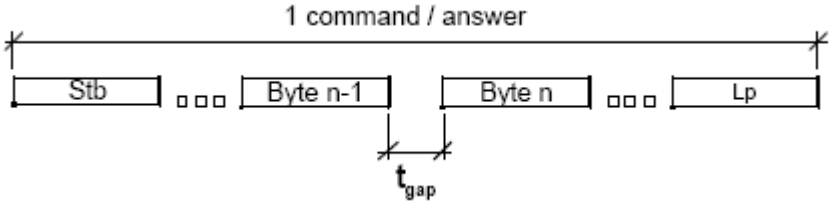
[4] jeden slave

[5] PP zařízení bez podpory MP-Bus mohou mít o 0,5 V nižší minimální práh tj. 4 resp. 5 V

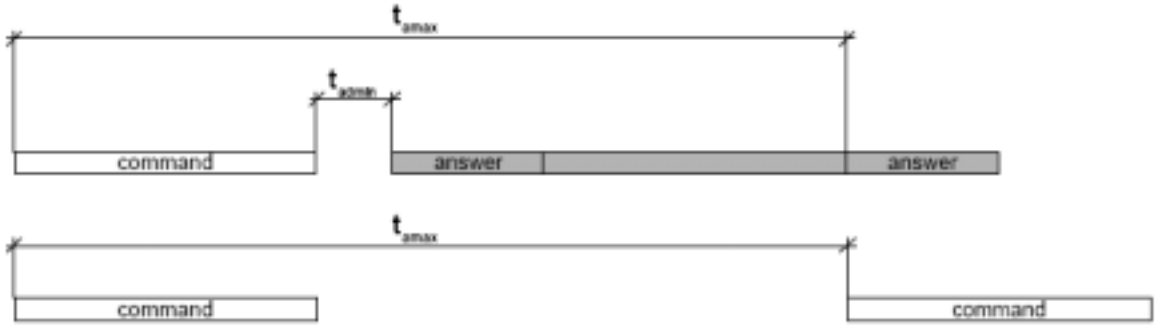
Vysvětlující příklady



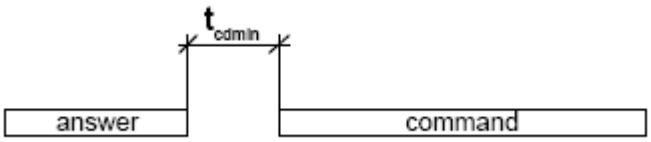
Obr. 15. Čas potřebný k přenesení jednoho bitu a rámce



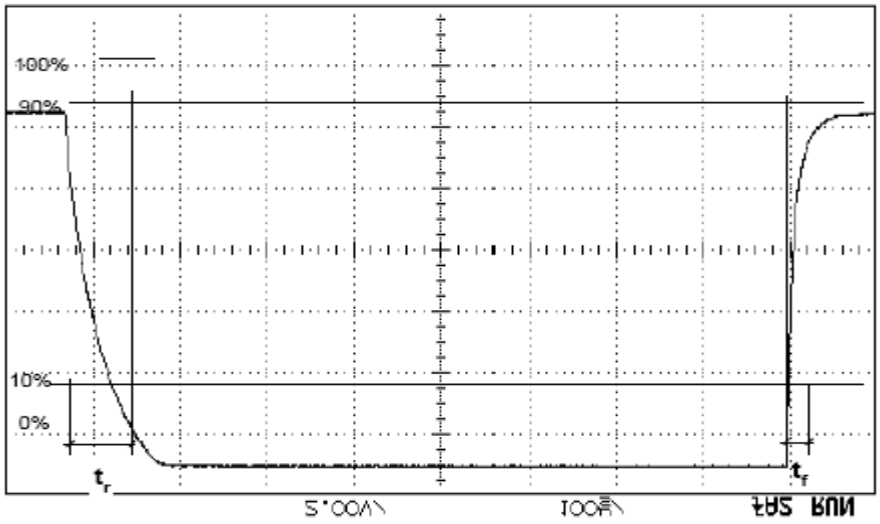
Obr. 16. Čas mezi dvěma byty



Obr. 17. Zpoždění odpovědi, master timeout nebo opakovací interval



Obr. 18. Zpoždění příkazu



Obr. 19. Budící čas a čas poklesu

Obr. 20. Signálové úrovně na sběrnici pro jeden připojený slave (při více slavech bude horní úroveň napětí nižší)

2.4.1 Dodatek

2.4.1.1 Příklad příkazu

Následujícím příkazem nastavíme na pohonu s MP adresou 3 polohu na 50%. Příkaz pro nastavení této polohy je MP_Set_Relative s kódem 37. Hodnota pro nastavení nové pozice je 5000, kdy jednotka je 0,01 %. Užitečné byty jsou 3 (1 byte pro příkaz a 2 byty pro parametr).

Generace start bytu:

Adresa 3 :	0b00001000 sečteno s
Počet užitečných bytů:	0b00110000
Celý start byte:	0b00111000 = 0x38

Generace MP příkazového paketu:

Kompletní start byte:		0b00111000 = 0x38
Kód příkazu :	37 =	0b00100101 = 0x25
Parametr:	5000 = 0x1388 =	0b00010011 = 0x13
		0b10001000 = 0x88
Příčná parita:		0b01110000 = 0x70
Podélná parita:		0b11110110 = 0xF6

Detekce MP odpovědního paketu:

Start byte:	0x0D = 0b00001101
Příčná parita:	0x80 = 0b10000000
Podélná parita:	0x8D = 0b10001101

MP-Bus telegram:

Master posílá:	0x38 0x25 0x13 0x88 0x70 0xF6
Slave odpovídá:	0x0D 0x80 0x8D

2.4.1.2 Tok protokolu

Zde jsou ukázky různých případů toku přenosu. Na levé straně jsou akce PP masteru a na pravé straně jsou odezvy ze slavu. Čas běží od shora dolů.

Chybová komunikace

PP-Master			***PP-Slave***	
Vrstva 7	Vrstva 2 + 1	<—>	Vrstva 1 + 2	Vrstva 7
Kódy příkazů			**Čekání na start bit**	**Čekání na příkaz**
posílá příkaz a následně čeká na odpověď	posílá příkazové byty	—>	Neobdrží příkaz (např. kvůli přerušenému vedení) nebo chyba na vrstvě 1 • chybějící stop bit • chybná parita nebo chyba na vrstvě 2 • neplatný start byte • chybějící byte (nastává timeout) Čeká na start bit	
	Chybějící timeout odpovědi			
	Opětovný přenos stejného příkazu (jsou nastaveny max. tři opakování)	—>	Obdrží všechny příkazové byty bezchybně	Obdrží příkaz Objeví se chyba na 7 vrstvě • neznámý příkaz • přístup zamítnut Je generována odpověď s chybovým kódem a poslána druhé vrstvě
	Neobdrží příkaz kvůli chybě přenosu	<—	Posílá odpověď s chybovým hlášením	Čeká na příkaz

	Chybějící timeout odpovědi		Čeká na start bit	
	Posílá znovu stejný příkaz (jsou nastaveny max. tři opakování)	—>	Obdrží všechny příkazové byty bez chyb	Obdrží příkaz
				Objeví se chyba na 7 vrstvě • neznámý příkaz • přístup zamítnut Je generována odpověď s chybovým kódem a poslána druhé vrstvě
Obdrží odpověď	Obdrží odpověď s chybovým hlášením	<—	Přenáší odpověď s chybovým hlášením	Čeká na příkaz
Reakce na chybu způsobenou aplikačním programem			Čeká na start bit	

Bezporuchová komunikace

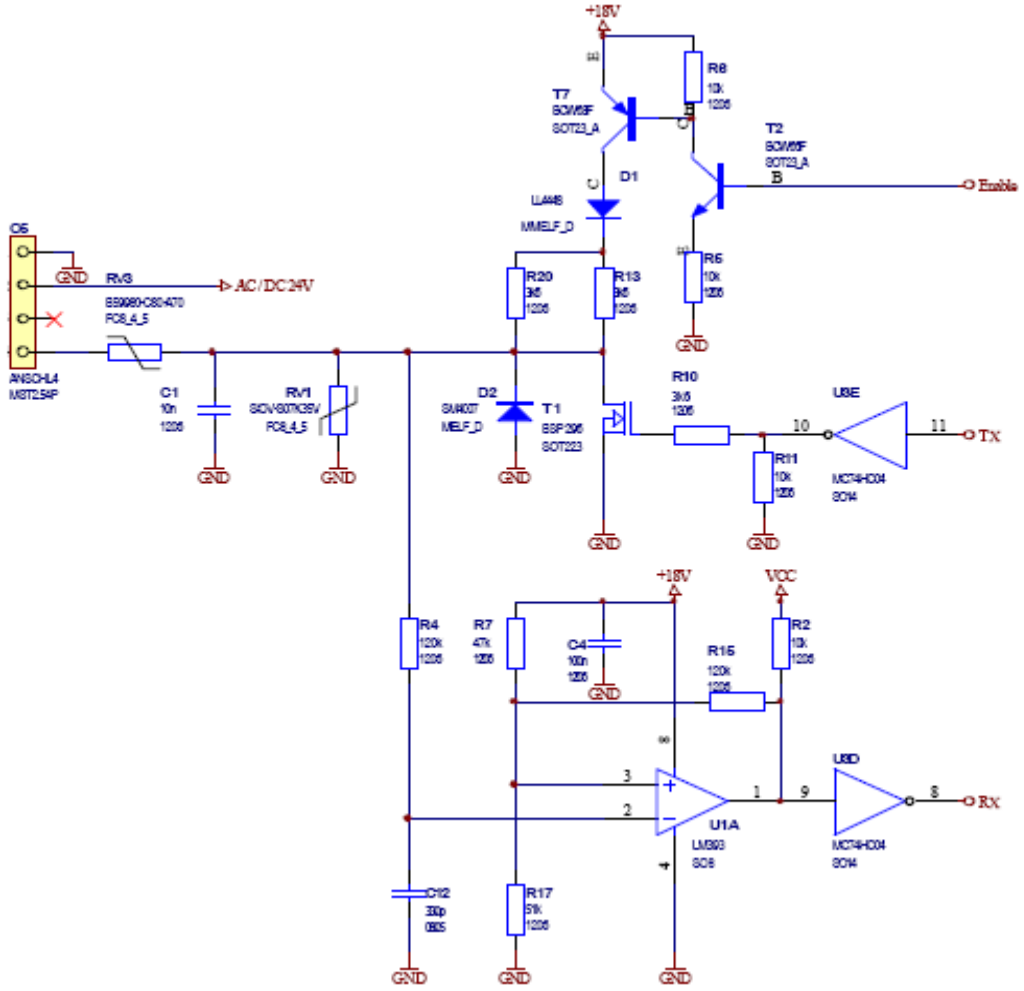
PP-Master ***PP-Slave***

Layer 7	Layer 2 + 1	<—>	Layer 1 + 2	Layer7
Připravuje následující příkaz				
Pošle příkaz a poté čeká na odpověď	Posílá příkazové byty	—>	Obdrží všechny příkazové byty bez chyb	Obdrží příkaz
				Vykoná příkaz Generuje odpověď a posílá ji druhé vrstvě
Obdrží odpověď	Obdrží odpovědní byty	<—	Posílá odpověď a poté čeká na další start bit	Čeká na příkaz

2.4.2 Zapojení obvodu MP masteru

Následující obvod je používán v zařízení UK24LON od firmy Belimo, které je chráněno proti zkratu se zemí nebo 24 V AC / ± 42 V DC a je plně otestováno. Toto zařízení splňuje

všechny výše zmíněné specifikace. UK24LON budiž bráno jako předloha pro všechny další master zařízení.



Obr. 21. Schéma zapojení MP mastera UK24LON od Belima

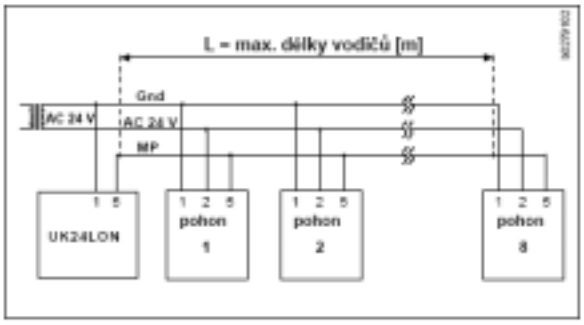
2.4.3 Maximální délka kabeláže

Na následujícím obrázku je uveden přehledem maximálních délek vedení pro centrální napájení z jednoho zdroje AC nebo DC nebo z lokálního zdroje AC. Tyto hodnoty platí pro všechny MP mastery splňující specifikaci MP-Bus.

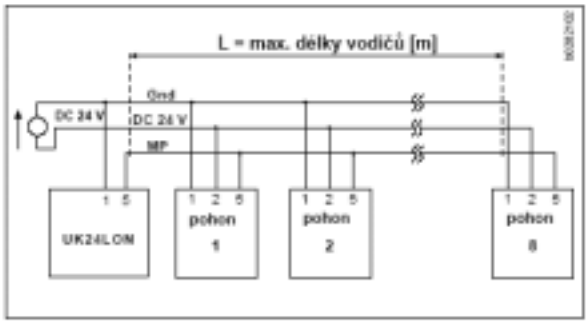
Připojení MP-Bus

- síť se zkládá z žilového vodiče (komunikace MP a napájení 24 V).
- možnost připojení max. 8 pohonů MFT(2) na síť.
- zapotřebí není ani speciální kabel ani koncový odpor.
- délky vodičů jsou limitovány (výpočet viz níže)
  - součtem dat výkonnů připojených pohonů MFT(2),
  - druhem napájení (AC 24 V přes Bus nebo DC 24 V přes Bus),
  - průřezem vodiče.

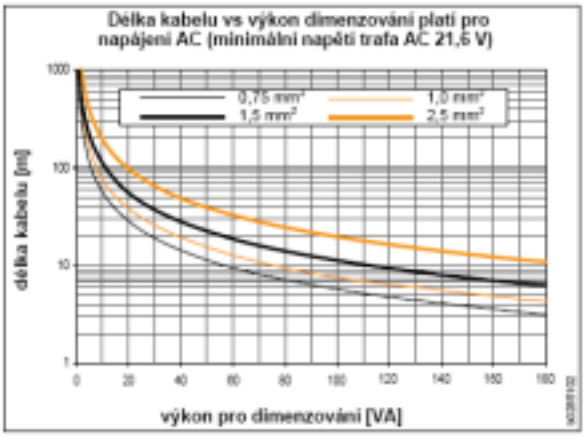
Maximální délky vodičů při napájení AC 24 V



Maximální délky vodičů při napájení DC 24 V

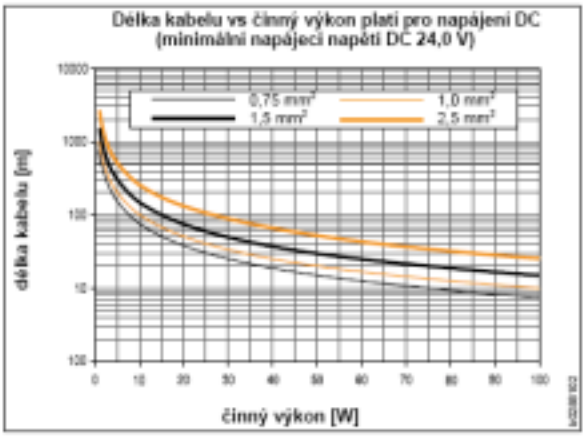


Celkový výkon pro dimenzování pohonu MFT(2) [VA]



⚠ U NVF24-MFT(2) je třeba výkon pro dimenzování násobit koeficientem 2.

Celkový výkon pro dimenzování pohonu MFT(2) [W]



Délka kabelu vs činný výkon platný pro napájení DC (minimální napájecí napětí DC 24 V)

Určování maximální délky vodiče

y pro dimenzování [VA] použitých pohonů MFT(2) se sečtou a v diagramu se odečtou potřebné délky vodiče.

Příklad:
Na MP-Bus jsou připojeny: 1 kus NM...,
1 kus AM..., 1 kus AF... a 1 kus NV...

Celkový výkon pro dimenzování:
3 VA + 5 VA + 10 VA + 5 VA = 23 VA

Z křivky se odečte:

- pro kabel s vodičem ∅ 0,75 mm² vychází: **kabel délky 25 m**
- pro kabel s vodičem ∅ 1,0 mm² vychází: **kabel délky 33 m**
- pro kabel s vodičem ∅ 1,5 mm² vychází: **kabel délky 50 m**
- pro kabel s vodičem ∅ 2,5 mm² vychází: **kabel délky 85 m**

Určování maximální délky vodiče

Příkony [W] použitých pohonů MFT(2) se sečtou a v diagramu se odečtou potřebné délky vodiče.

Příklad:
Na MP-Bus jsou připojeny: 1 ks NM...,
1 ks AM..., 1 ks AF... a 1 ks NV...

Celkový výkon pro dimenzování:
1,3 W + 2,5 W + 6,0 W + 3,0 W = 12,8 W

Z křivky se odečte:

- pro kabel s vodičem ∅ 0,75 mm² vychází: **kabel délky 60 m**
- pro kabel s vodičem ∅ 1,0 mm² vychází: **kabel délky 80 m**
- pro kabel s vodičem ∅ 1,5 mm² vychází: **kabel délky 115 m**
- pro kabel s vodičem ∅ 2,5 mm² vychází: **kabel délky 200 m**

Maximální délky vodičů při lokálním napájení AC 24 V (přímo na místě)

Upozornění: Pokud jsou pohony lokálně napájeny samostatným transformátorem s AC 24 V, pak lze podstatně prodloužit délku vodiče. Nezávisle na údajích o výkonu na UK24LON napojených pohonů činí délky vodičů viz tabulka.

vodič ∅ mm²	L = max. délky vodičů [m]
0,75	800
1,0	
1,5	
2,5	

2.5 Seznam MP příkazů

Do současnosti bylo pro MP-Bus vytvořeno 92 příkazů s tím, že některé z nich už nejsou u nových typů servopohonů podporovány a byli nahrazeny příkazy novými. Část příkazů je zase určena jen pro konkrétní typ servopohonu. Seznam příkazů pro daný typ servopohonu spolu s parametry těchto příkazů, je pak uložen v paměti tohoto servopohonu. V následující tabulce jsem se pokusil sestavit seznam příkazů s jejich stručným popisem, které byli popsány ve vývojové dokumentaci poskytnuté firmou Belimo. Jejich bližší popis pak lze nalézt v dokumentaci [7].

Tab. 16. Seznam MP příkazů

Název a dekadický kód příkazu	Popis funkce
Obecné příkazy	
MP_Get_SeriesNo (50)	Vyčtení sériového čísla
MP_Set_MP_Address (38)	Nastavení MP Adresy
MP_Get_MP_Address (13)	Vyčtení MP adresy
MP_Set_Relative (37)	Nastavení pozice
MP_Get_Relative (41)	Vyčtení aktuální a nastavené pozice
MP_Get_VRelative (57)	Vyčtení aktuálního a nastaveného průtoku vzduchu
Doplňující příkazy	
MP_AD_Convert (04)	Vyčtení hodnoty aktivního nebo pasivního senzoru
MP_Y_U_in_mV (44)	Předchůdce MP_AD_Convert pro VAV jednotky
MP_Get_Vist_Nonlin (76)	Vyčtení hodnoty D2 senzoru u VAV jednotek (pohony NMV-D2, NMV-D2M a VRD2)
MP_Set_Override_Control (14)	Nastavení funkce pro nadřazené řízení
MP_Get_Override_Control (75)	Vyčtení stavu nadřazeného řízení (slouží i na vyčtení stavu externího přepínacího kontaktu)
MP_Motor_Control (36)	Předchůdce MP_Set_Override_Control pro VAV jednotky

Název a dekadický kód příkazu	Popis funkce
MP_Get_State (10)	Vyčtení aktuálního stavu pohonu
MP_Get_Settings (12)	Vyčtení nastavení z EEPROM
MP_Set_Switch (40)	Nastavení pozic pro přepnutí dvou softwarových přepínačů
MP_Get_Switch (28)	Vyčtení nastavení přepínacích pozic a pozice softwarového přepínače
MP_Set_Ext_Event (15)	Nastavení událostí přiřazených mechanickému přepínači nebo tlačítku
MP_Get_Ext_Event (69)	Vyčtení nastavených událostí přiřazených mechanickému přepínači nebo tlačítku
MP_Get_String_Adr (71)	Vyčtení počátečních adres identifikačních řetězců
MP_Peek (01)	Vyčtení identifikačního řetězce z paměti
MP_Poke (02)	Zápis identifikačního řetězce do paměti
MP_Get_MEM_Info (20)	Vyčtení informací z paměti (určení verze firmwaru, počáteční adresy a hodnoty EEPROM)
MP_Get_Config_Table_Info (18)	Zjištění počtu implementovaných příkazů a adresa umístnění konfigurační tabulky se seznamem příkazů a parametrů
Konfigurační příkazy	
MP_Set_Direction (07)	Nastavení směru pohybu pohonu
MP_Get_Transit_Time (32)	Vyčtení nominální doby přeběhu a nominálního rozsahu
MP_Set_Transit_Time (46)	Nastavení nominální doby přeběhu
MP_Set_Operating_Range (58)	Nastavení nominálního operačního rozsahu
MP_Set_Min_Mid_Max (61)	Nastavení minimální, střední a maximální pozice

Název a dekadický kód příkazu	Popis funkce
MP_Get_Min_Mid_Max (59)	Vyčtení minimální, střední a maximální pozice
MP_Set_Min_Max (31)	Nastavení minimálního a maximálního průtoku vzduchu u VAV jednotek
MP_Get_Min_Max (30)	Vyčtení nastavení nominálního, maximálního a minimálního průtoku vzduchu u VAV jednotek
MP_Login (78)	Přihlášení
MP_Logout (60)	Odhlášení
MP_Set_Password (79)	Nastavení hesla
MP_Start_Adaption (33)	Spuštění adaptace, synchronizace nebo testovacího přeběhu
MP_Set_Sync (65)	Nastavení směru synchronizace
MP_Set_Ext_Functions (83)	Nastavení doplňujících funkcí
MP_Get_Ext_Functions (89)	Vyčtení nastavení doplňujících funkcí
Servisní příkazy	
MP_Get_Stress (11)	Vyčtení doby zapnutí, doby běhu a poměru doby běhu k době zapnutí
MP_Set_Malfunction_Maintenance_Mask (34)	Nastavení událostí pro selhání a potřebu údržby
MP_Get_Malfunction_Maintenance_Mask (24)	Vyčtení nastavených událostí pro selhání a potřebu údržby
MP_Get_Malfunction_Maintenance_State (26)	Vyčtení statusu pro identifikaci selhání či potřeby údržby
MP_Reset_Malfunction_Maintenance_State (29)	Vymazání signalizace selhání či potřeby údržby
MP_Echo (00)	Vrátí se poslaná hodnota

Název a dekadický kód příkazu	Popis funkce
Příkazy pro pohony požárních a odkuřovacích klapek	
MP_Start_Testrun_Fire (86)	Spuštění synchronizace, požárního testovací přeběhu, testu průchodnosti komínové klapky a adaptace
MP_Set_Min_Adaptionrange (87)	Nastavení minimálního adaptačního úhlu
MP_Get_Min_Adaptionrange (88)	Vyčtení nastavení minimálního adaptačního úhlu

3 SOUVISEJÍCÍ BELIMO PŘÍSLUŠENSTVÍ

3.1 Parametrizační nástroje

Belimo servopohony vybavené technologií MFT mají z výroby nastaveny parametry se základními hodnotami vhodnými pro běžné aplikace. Na přání zákazníka však také mohou být přímo ve výrobě nastaveny dle zvolených požadavků. Pro změnu nastavení servopohonů po jejich dodání, pak slouží parametrizační nástroje MFT-H nebo PC-Tool. Ty kromě přenastavení hodnot různých parametrů umožňují také vyčítat aktuální a nastavené hodnoty parametrů, kontrolovat správnou funkčnost zařízení, provádět funkční testy apod. [1]

3.1.1 MFT-H

Jedná se o ruční parametrizační zařízení, které opět umožňuje provádět parametrizaci a servisní zásahy u většiny servopohonů s technologií MFT (vyjma MF a MP ventilových pohonů).[8]

Obr. 22. Ruční parametrizační zařízení MFT-H

3.1.2 PC-Tool

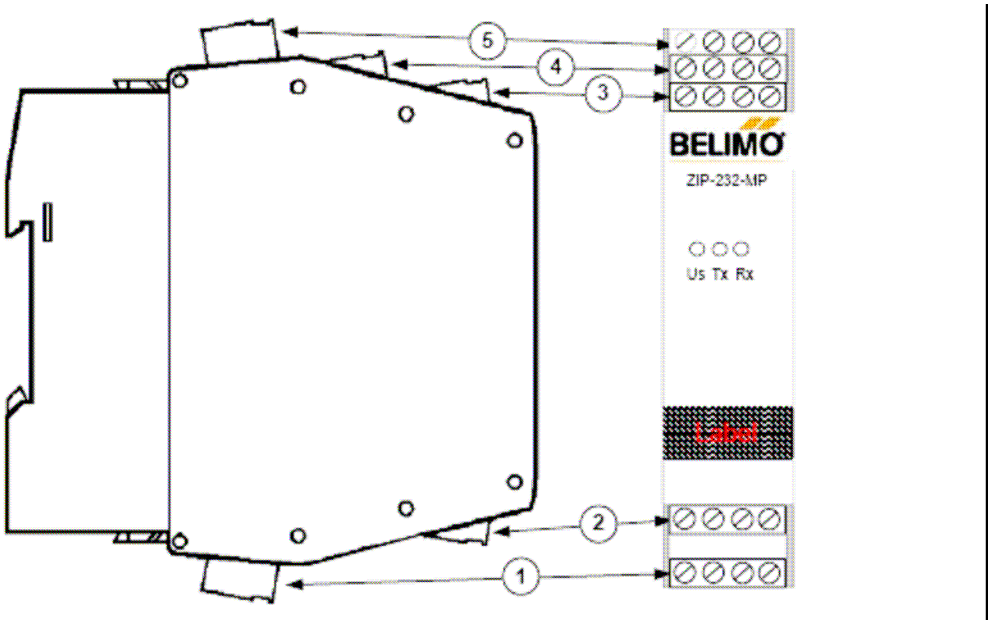
Je to program určený k instalaci na PC, který obsahuje různé moduly umožňující přenastavení parametrů připojených servopohonů s technologií MFT. Servopohony MP a MFT(2) pak umožňují také přímé přenastavování svých parametrů prostřednictví sítě MP-Bus, kdy si PC-Tool samo identifikuje všechna připojená MP a MFT(2) zařízení.

Pomocí PC-Tool se dá provádět parametrizaci a servisní diagnostika pohonů, jako např. nastavení pozice, vyčtení aktuální pozice a vyčtení hodnoty připojeného senzoru. Tyto hodnoty je pak možno zobrazit i ve formě grafu, znázorňujícího jejich průběh v čase.

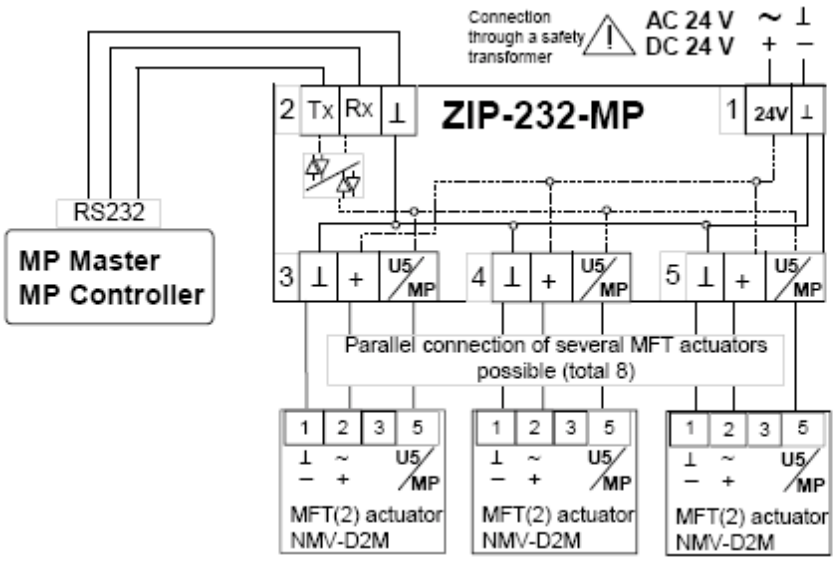
Servopohony se z PC parametrizují pomocí převodníku z rozhraní RS232 na MP-Bus. K tomu slouží převodníky ZIP-RS232 a ZIP-232-KA.[9]

3.2 ZIP-232-MP

Převodník mezi rozhraním RS232 a MP-Bus vhodný k montáži na DIN lištu. Je určen k připojení MP-Bus masterů (řídicí systémy, PC,…), které mají rozhraní RS232 a mají implementován protokol MP-Bus, ale nemají vlastní hardwarové rozhraní schopné zpracovat signály ze sběrnice MP-Bus.



Obr. 23. Vzhled převodníku ZIP-232-MP



Obr. 24. Schéma zapojení převodníku ZIP-232-MP

Tento převodník má na sobě tři LED diody:

Us – signalizuje napájení

Tx – signalizuje tok dat z převodníku na pohon

Rx – signalizuje tok dat z pohonu na převodník

Dále pak má napájecí svorku 24 V AC/DC, svorku rozhraní RS232 (pouze signály Rx, Tx a GND) a tři svorky rozhraní MP-Bus s možností dohromady připojit až osm MP zařízení.[10]

3.3 ZN230-24MP

Jednotka pro lokální napájení servopohonů připojených na MP-Bus.[11]

4 ŘÍDICÍ SYSTÉMY FIRMY AMIT

Většinu použitých informací o řídicích systémech firmy AMiT jsem čerpal z webových stránek této společnosti [2]. Na těchto stránkách jsou uvedeny všechny potřebné informace týkající se nejenom vyráběných řídicích systémů, ale jsou zde také zdarma ke stažení softwarové nástroje potřebné k jejich programování.

4.1 Modulární a kompaktní řídicí systémy

Hlavní část produkce firmy AMiT spočívá v nabídce modulárních a kompaktní řídicích systémů. Tyto řídicí systémy jsou volně programovatelné v návrhovém prostředí PSP3 a všechny disponují rozhraním RS232, které je nezbytné pro nahrání operačního systému. Dle typu řídicího systému jsou k dispozici také další rozhraní jako RS485, Ethernet, CAN nebo M-Bus. Rozhraní RS232 může být také v případě potřeby převedeno externím převodníkem na rozhraní RS485, Ethernet, M-Bus, MP-Bus apod.

4.1.1 Možnosti rozšíření počtu vstupů a výstupů

U modulární koncepce se k základní jednotce (CPU) může připojit až 16 vstupně/výstupních či komunikačních modulů. Počet vstupů a výstupů lze pak dále rozšířit pomocí expanzních vstupně/výstupních modulů komunikujících pomocí protokolu ARION (protokol firmy AMiT) na lince RS485. Další možnost rozšíření pak nabízí distribuovaný systém DIOCAN, vycházející z koncepce sběrnice CAN, který je kompatibilní s protokolem CANOpen.

U kompaktních systémů existuje možnost rozšíření počtu jejich vstupů a výstupů pomocí expanzních modulů komunikujících s řídicím systémem po rozhraní RS485 protokolem ARION. Ten dovoluje připojit až 63 takových modulů.

4.1.2 Komunikace

Řídicí systémy mohou být také propojovány mezi sebou nebo být napojeny na nadřazený systém (např. dispečerské pracoviště). K tomu slouží komunikační protokoly DB-Net a DB-Net/IP (protokol na bázi TCP/IP). Jedná se o komunikaci typu multimaster-multislave na bázi linky RS485 nebo průmyslového Ethernetu. Tyto protokoly umožňují mimo jiné také možnost dálkového přenosu aplikace.

Velkou předností programového vybavení řídicích systémů firmy AMiT jsou tzv. lokální archivy, které umožňují archivovat měřená a vypočtená data i v okamžiku, kdy není v činnosti dispečerské pracoviště anebo došlo k poruše komunikace. Automatický zpětný přenos dat zabezpečí, že ani v těchto případech uživatel nepřijde o důležité údaje.

Řídicí systémy firmy AMiT lze mezi sebou nebo s nadřazeným systémem propojovat těmito základními způsoby:

- komunikační linka RS485
- komunikační linka RS232 (pouze bod-bod)
- komunikační sběrnicí CAN
- průmyslovým Ethernetem (DB-Net/IP)
- Intranetem, Internetem (DB-Net/IP)
- modemovým přenosem (telefon, radio, GSM, GPRS)

Propojení linkou RS485 představuje standardní průmyslovou komunikaci. Na jedné lince RS485 lze připojit až 32 stanic, celková délka segmentu bez opakovače linky nesmí přesáhnout 1200m. Vhodným použitím opakovačů lze však bez problémů tuto vzdálenost prodloužit až na 6000m, přičemž lze vytvářet i větvené struktury.

Tyto řídicí systémy mají přímou podporu pro komunikaci protokoly:

- ARION (expanzní moduly DM-xx)
- Modbus
- S-Bus
- M-Bus
- ASIMP (propojení regulátorů REFACO)
- CANopen
- sériová tiskárna
- Díky možnosti doprogramování uživatelské komunikace v prostředí PSP3 je pak možno komunikovat i jinými protokoly (např. MP-Bus)

4.2 Programování řídicích systémů firmy AMiT

Všechny řídicí systémy firmy AMiT jsou volně programovatelné v návrhovém prostředí PSP3.

4.2.1 PSP3

Programové prostředí PSP3 je určeno pro tvorbu uživatelských aplikací pro všechny standardní řídicí systémy nabízené firmou AMiT. Program PSP3 představuje "balík" vzájemně provázaných programových modulů, pomocí nichž uživatel vytvoří nejen vlastní aplikaci řídicího systému, ale například i vzhled terminálových obrazovek, definici chybových hlášení, získá velice užitečnou ladící podporu a může bez problémů komunikovat s řídicími systémy. Mezi základní moduly patří:

- PSP - správce projektů. Tento modul zastřešuje všechny následující programové moduly
- PSE - tvorba a ladění řídicích algoritmů řídicího systému
- LCDSHELL - tvorba a ladění vzhledu terminálových obrazovek
- DTE - tvorba formátování hlášení provozního deníku

Prostředí je navrženo tak, aby umožnilo využívat všech výpočetních a algoritmizačních možností řídicích systémů bez hlubších znalostí programování. Výsledkem je prostředí, ve kterém se programování blíží slovnímu popisu úlohy, přičemž tvůrce aplikace je doslova nucen k dobré strukturalizaci problému. Pro speciální aplikace (především pro řízení strojů), pak lze využít možnost programování pomocí kontaktního schématu či instrukcemi logického automatu, aby bylo vyhověno již zažitým postupům některých tvůrců aplikací. Tento parametrizační program je včetně všech potřebných manuálů k dispozici zdarma na [2].

PSE

Programový modul PSE poskytuje všechny nástroje pro tvorbu a ladění vlastních algoritmů řídicích systémů. Filozofie programu vychází ze zásady jednotného způsobu programování všech řídicích systémů firmy AMiT.

Uživatel má několik způsobů návrhu algoritmu a sám může zvolit podle povahy problému ten nejvhodnější. Základním a nejvíce blízkým slovnímu popisu je použití tzv. funkčních

modulů, což jsou v podstatě již předdefinované funkce. Programátor pak pouze vyplňuje vhodné parametry funkcí a zajišťuje vazby na měřená či vypočtená data. V současné době je k dispozici více než 200 funkčních modulů zařazených do knihoven s různým zaměřením (např. pro tepelné soustavy, energetiku, propojení modemy, maticový počet, ovládání tiskárny, komunikační protokoly apod.). Pro uživatele se zkušenostmi v programování klasických PLC je k dispozici podpora návrhu pomocí kontaktního plánu a instrukcí logického automatu dle normy IEC 1131. Předností programu PSE je, že umožňuje použití uvedených koncepcí parametrizace současně.

Program PSE neplní pouze funkci parametrizační, ale jeho velkou předností jsou i další podpůrné funkce, které řeší komplexní otázku nasazení a ladění algoritmů. Velmi ceněnou vlastností je jeho automatická tvorba dokumentace. Tímto způsobem je zajištěn soulad mezi dokumentací a skutečnou činností řídicího systému, což je základní podmínkou pro udržovatelnost a pozdější modifikovatelnost programového vybavení. PSE dále podporuje on-line sledování a editaci technologických proměnných (včetně kontaktního plánu) a to přímo v rámci komunikační sítě DB-Net resp. DB-Net/IP. Mezi další podpůrné funkce patří i možnost vytvoření vlastních proměnných, používání pomocných označení signálů, analýza návrhu, ladění a zavádění programu přímo po síti DB-Net apod..

LCDSHELL

Vzhledem k častému používání průmyslových terminálů je prostředí PSP3 doplněno i o nástroj pro tvorbu uživatelských obrazovek terminálu. Jinými slovy nástroj pro definici výsledného obsahu zobrazovaných údajů na displeji a způsobu jakým lze tyto údaje editovat. Pomocí nástroje LCDSHELL lze definovat chování všech textových terminálů a displejů řídicích systémů. Program nejen napomáhá snadno a přehledně definovat data, určující zobrazování požadovaných údajů na displeji, ale dovoluje i simulaci chování terminálu již při samotném návrhu bez nutnosti jeho fyzického připojení k řídicímu systému.

DTE

Automatickou součástí programového vybavení všech řídicích systémů firmy AMiT je tzv. provozní deník. Vedle standardních, systémem generovaných chybových a provozních hlášení, má uživatel možnost definovat si vlastní "technologická" hlášení, která vyplývají z charakteru řízené anebo regulované technologie. A právě pro definici a formátování hlášení provozního deníku slouží modul DTE. Program se používá v případě, kdy je potřeba tisk-

nout provozní deník na tiskárně připojené k řídicímu systému nebo zobrazovat deník na terminálu, případně zpracovávat údaje z deníku na vizualizační stanici či dispečinku.

II. PRAKTICKÁ ČÁST

5 POPTÁVKA PO IMPLEMENTACI

Produkty obou výrobců se spolu velmi často setkávají v aplikacích na poli technického zabezpečení budov či tepelného hospodářství s tím, že zde nejsou v pozici konkurentů ale jsou naopak předurčeny spolu spolupracovat. Z toho tedy vzešel logický požadavek na vzájemnou spolupráci mezi oběma společnostmi.

Servopohony Belimo jsou totiž typickým zástupcem akčního členu (slavu), který potřebuje být řízen, a naproti tomu řídicí systémy AMiT představují typického mastera určeného ke zpracování naměřených údajů a řízení periférií, jako jsou právě servopohony Belimo.

Dlouhou dobu tato spolupráce existovala jen na úrovni realizačních firem (aplikátorů), kdy při konvenčním způsobu řízení servopohonů „stačí“ jen připojit vodiče jdoucí ze servopohonu na správné vstupy a výstupy řídicího systému a v programu s nimi pracovat jako s normální vstupy a výstupy. Každý servopohon tak zabírá svůj díl těchto vstupů a výstupů, a každý pohon si také v případě asynchronního provozu žádá své vlastní vodiče.

Pokrok však nejde zastavit, a tak se dnes už i u periférií, které dříve vystačili pouze s ovládáním pomocí analogových či digitálních signálů, objevuje snaha je vybavit větší „inteligencí“. A tak se děje to, že jsou akční členy vybavovány mikroprocesory a začleňují se do sběrnicových systémů jako je LonWorks či EIB. To pak usnadňuje současné řízení více akčních členů najednou. Úplně zde pak odpadá potřeba vstupů a výstupů na systému, jelikož řízení se děje po komunikačních rozhraních a v neposlední řadě se velmi významně snižují nároky na kabeláž a to díky možnosti mít všechny prvky sériově či paralelně zapojeny v jedné síti.

Touto cestou šla i firma Belimo, která představila svůj vlastní komunikační protokol MP-Bus, který umožňuje připojit k jednomu masteru (např. řídicímu systému) až osm slavů, typicky servopohonů. Přes tuto sběrnici se pak servopohony dají nejen ovládat a vyčítat z nich např. pozici, ale umožňuje mnohem víc od nastavení různých parametrů, přes kompletní diagnostiku, až po možnost vyčtení hodnoty externího senzoru připojeného k pohonu.

Z toho je tedy zřejmé, že takovýto „nový přístup“ k ovládání periferií se stává pro běžného aplikátora systémů měření a regulace poměrně komplikovaný a na scénu musí vstoupit výrobce řídicích systémů. Ten by pak měl svým zákazníkům zajistit takové softwarové a případně i hardwarové prostředky, aby pro ně bylo používání tohoto typu řízení co nejjed-

nodušší. Pokud jde o hardwarové prostředky, tak lze využít produktů firmy Belimo jako je převodník ZIP-232-MP. Proto je třeba se v první fázi zaměřit zejména na vytvoření takového softwarového produktu, který by co nejvíce zjednodušil práci aplikátora.

Jelikož si je firma Belimo vědoma nutnosti spolupráce s výrobci řídicích systémů apod., tak kromě toho že poskytuje vlastní přístupové uzly do sítí jako je LonWorks či EIB, se nesnaží protěžovat žádný vlastní regulátor, tak jako to dělají např. výrobci kotlů, ale naopak dává výrobců řídicích systému k dispozici popis tohoto protokolu i s technickou podporu při jeho implementaci. Mezi nejznámější výrobce, kteří už implementovali tento protokol do svých řídicích systémů patří např. firma SAIA Burges, která má vlastní komunikační modul schopný ovládat až osm servopohonů a vyčítat až osm senzorů.

Podobnou cestou by se ráda vydala také firma AMiT, a proto oslovila českou pobočku Belima, která jí poskytla nutný hardware k testování a kontakt na příslušné oddělení ve švýcarské centrále. To pak poskytlo veškerou potřebnou technickou dokumentaci v anglickém jazyce.

No a práce mohla začít.

6 POUŽITÝ HARDWARE

6.1 Řídicí systém

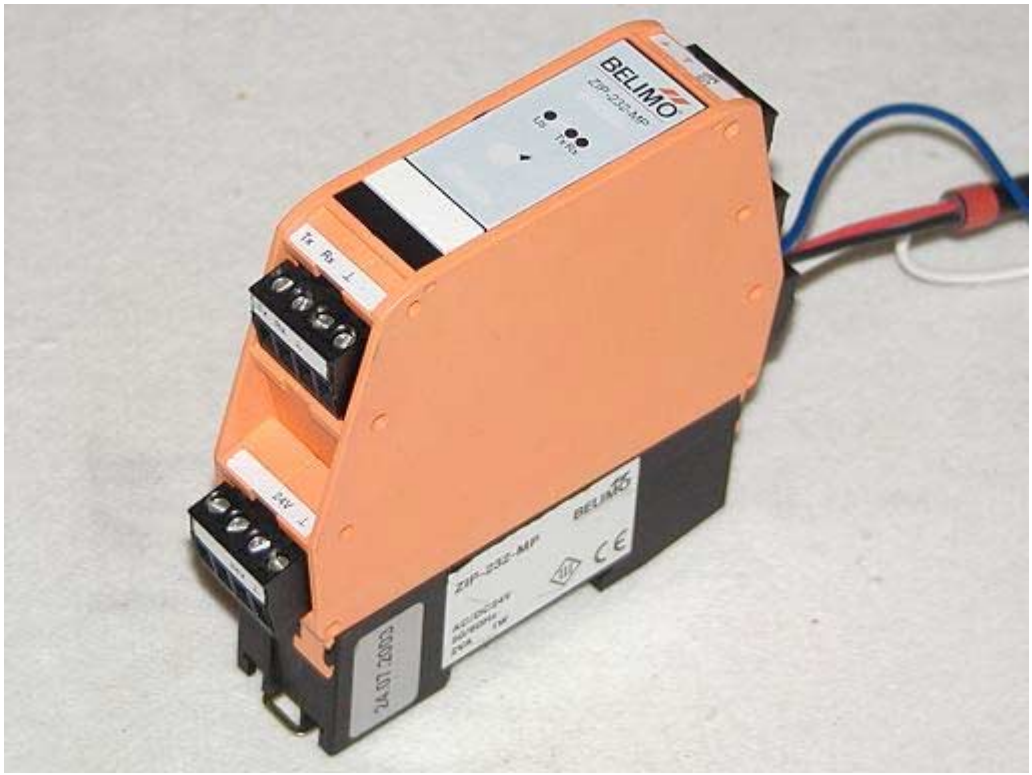
Jako řídicí systém byl zvolen malý kompaktní řídicí systém AMiNi-E, který má tři komunikační rozhraní (RS232, RS485 a Ethernet), 8 digitálních vstupů, 8 digitálních výstupů a 4 univerzální analogové vstupy. Tyto informace jsou více méně zbytečné, protože ve svém důsledku je pro komunikaci prostřednictvím MP-Busu potřeba pouze jedno rozhraní RS232. Vzhledem k tomu, že je toto rozhraní na všech řídicích systémech firmy AMiT, jelikož pouze přes něj jde nahrát operační systém, a vzhledem k tomu, že všechny řídicí systémy mají použit stejný mikroprocesor a programují se ve stejném programovacím prostředí, mohl být stejně tak dobře použit kterýkoliv jiný řídicí systém. Z tohoto důvodu je také možno použít vytvořenou uživatelskou komunikaci pro kterýkoliv z řídicích systémů firmy AMiT. Je jen potřeba vždy zvolit správnou platformu (správný operační systém) a správný komunikační port.

Obr. 25. Kompaktní řídicí systém AMiNi-E

6.2 Převodník

Jelikož má sběrnice MP-Bus vlastní signálovou úroveň (hardwarovou vrstvu), bylo potřeba použít převodníku z MP-Busu na rozhraní RS232 řídicího systému. K dispozici jsem měl převodník firmy Belimo ZIP-232-MP, který převádí signálové úrovně MP-Bus na signálové úrovně rozhraní RS232 a naopak.

Toto zařízení má na sobě svorku napájení, svorku RS232 (pouze signály Rx, Tx a GND) a tři svorky pro připojení až osmi zařízení podporujících komunikaci MP-Bus. Kromě toho na sobě má tři LED diody, které signalizují připojení napájení Us, tok dat z převodníku k servopohonu Tx a tok dat ze servopohnu na převodník Rx.



Obr. 26. Převodník ZIP-232-MP

6.3 Servopohony

K dispozici jsem měl dva typy servopohonů. Klapkový servopohon NM24-MFT2 a ventilový servopohon NV24-MFT2.

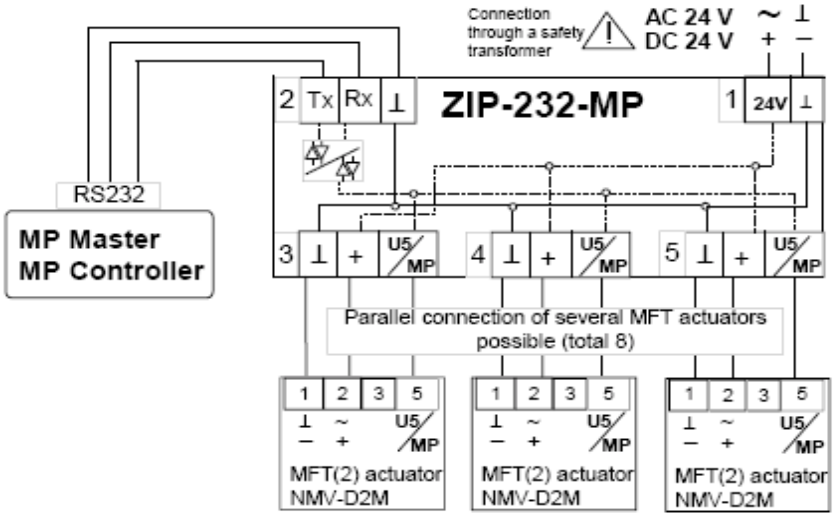
Obr. 27. Klapkový servopohon NM24-MFT2

Obr. 28. Ventilový servopohon NV24-MFT2

7 PROPOJENÍ ZAŘÍZENÍ

7.1.1 Připojení řídicích systémů k převodníku ZIP-232-MP

Nejvíce práce pak bylo s připojením převodníku ZIP-232-MP na konektor RS232 řídicího systému AMiNi-E, kde je rozhraní RS232 opatřeno konektorem RJ45 (stejné jako je u Ethernetového portu). Na straně převodníku jsou pak tři svorky pro signály Tx, Rx a GND. Jelikož jsem takový propojovací kabel neměl k dispozici, udělal jsem si jej sám z kabelu, který měl na jednom konci konektor RJ45. Jeho druhý konec jsem pak po odizolování příslušných vodičů Rx, Tx a GND připojil na tyto svorky u rozhraní RS232 převodníku.



Obr. 29. Schéma připojení řídicích systémů a servopohonů k převodníku ZIP-232MP

7.1.2 Připojení servopohonů k převodníku ZIP-232-MP

Připojení servopohonů na převodník ZIP-232-MP bylo snadné jelikož jsou oba servopohony opatřeny metrovým kabelem. Pro klapkový pohon je čtyř žilový s vodiči pro:

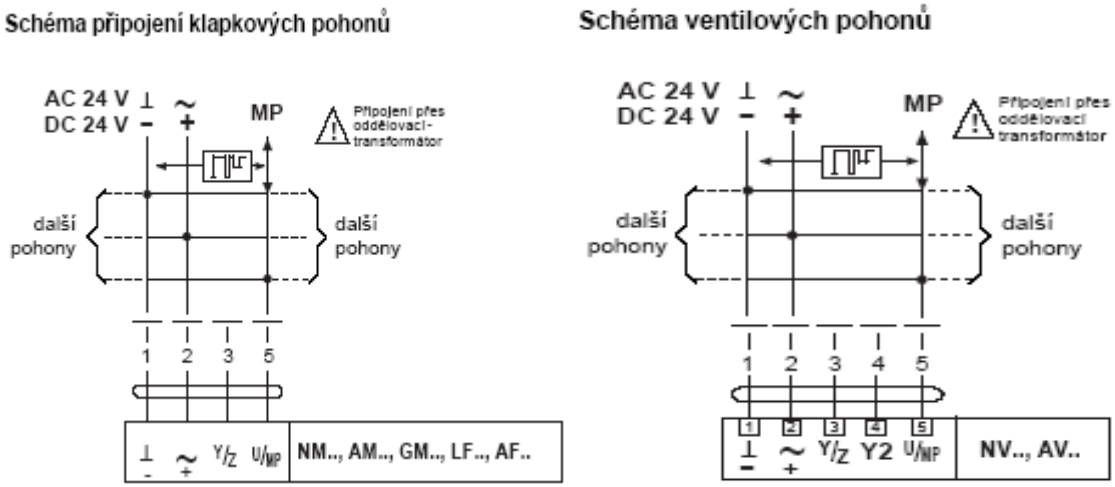
- Napájení 24 V AC/DC
- GND
- Y/Z – v MP módu je používaný pro připojení externího spínače nebo senzorů

- U5/MP – v MP módu na něm probíhá komunikace tímto protokolem. Tento signál je vztažen k zemi napájení.

U ventilových pohonů je pak k dispozici ještě pátý vodič:

- Y2 – v MP režimu nemá žádný význam

Následující obrázky přibližují schéma připojení pohonů na sběrnici MP-Bus. V mém případě byla metrová délka kabelu servopohonu dostačující, a tak jsem jej připojil dle uvedených schémat přímo k převodníku. Jelikož na převodníku ZIP-MP-232 jsou celkem tři svorky pro připojení MP zařízení, každý servopohon jsem dal na vlastní svorku.



Obr. 30. Schéma připojení pohonů na sběrnici MP-Bus

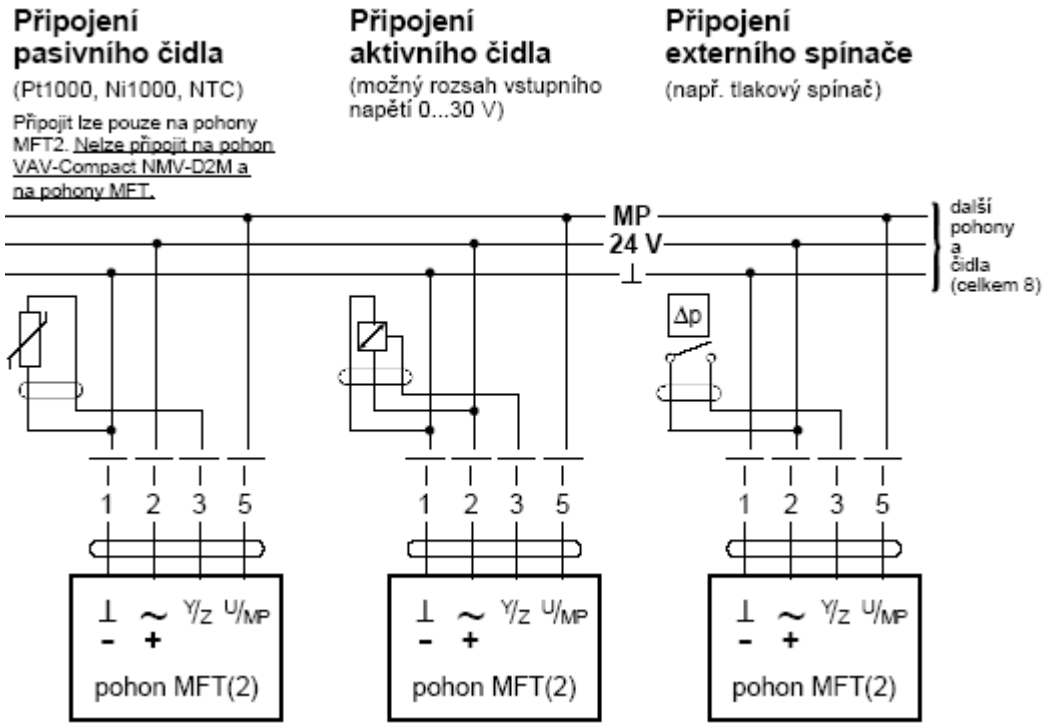
Pozn.: Sběrnice MP-Bus nevyžaduje žádné speciální kabely nebo zakončovací odpory a nemá ani žádné omezení pokud jde o topologii, takže je možné zapojení do hvězdy, kruhu, trojúhelníku i jejich kombinace.

7.1.3 Připojení napájení k převodníku ZIP-232-MP

Napájení převodníku je realizováno ze stejného zdroje jako napájení řídicího systému, v našem případě 24 V DC. Přes převodník je pak realizováno napájení celé sběrnice MP-Bus a tím i servopohonů.

7.1.4 Připojení senzorů

K pohonům bylo připojeno také jedno pasivní odporové čidlo Ni1000 k ověření funkčnosti MP příkazu pro vyčítání hodnoty externích čidel. Zapojení čidel a spínačů je patrné z následujícího obrázku.



Obr. 31. Schéma připojení externích čidel a spínačů

8 POSTUP PRÁCE

V této práci se budeme zabývat vytvořením takových softwarových prostředků, které by co nejvíce zjednodušili práci aplikátorům řídicích systémů firmy AMiT, kteří by chtěli ovládat Belimo servopohony prostřednictvím sběrnice MP-Bus.

Cesty k tomu jsou dvě, z nichž první je vytvoření uživatelské komunikace v programu PSP3, která by postihla všechny funkce potřebné k ovládání pohonů při využití stávajících funkčních bloků a programováním dle zásad pro program PSP3. Druhou možností pak je přímo vytvořit funkční bloky do programu PSP3, které by pokrývali všechny potřebné funkce. Tato cesta je však nepoměrně složitější, jelikož vyžaduje nejenom hlubší znalost některého programovacího jazyka (např. C++), ale také znalost způsobu psaní funkčních bloků pro program PSP3.

Jelikož nejsem programátorem a ani neznám způsob jakým se vytváří funkční bloky pro program PSP3, přenechám raději jejich vytvoření zkušeným softwarovým vývojářům firmy AMiT, kterým jsem schopen poskytnout relevantní informace potřebné k jejich tvorbě. V rámci této práce tedy budu zadaný úkol řešit cestou tvorby uživatelské komunikace v programu PSP3.

8.1 Vlastní postup

Vlastní postup práce vypadal přibližně takto:

- Všechno začalo studiem podkladů, především pak specifikace samotného protokolu, zejména s cílem zjištění struktury datagramu
- Poté následovalo studium jednotlivých příkazů. Zejména těch potřebných k adresování servopohonu, zadání polohy, vyčtení aktuální polohy a vyčtení hodnoty připojeného senzoru či čidla. Jsou to tedy hlavně příkazy ze skupiny obecných příkazů viz (Tab. 16)
- Následně přišla na řadu práce s tužkou a čtverečkovaným papírem, kdy bylo potřeba sestavit datagramy k těmto příkazům, tj. sestavit správný start byte, užitečné byty a vypočíst podélnou a příčnou paritu
- Pak jsem na port RS232 řídicího systému připojil převodník na MP-Bus a připojil k němu servopohon

- V dalším kroku přišlo na řadu nastudování samotného programování v programu PSP3 a příkladu jednoduché uživatelské komunikace, která je standardní součástí instalace programového prostředí PSP3.
- Vytvoření jednoduchého programu pro otestování komunikace a základních příkazů
- Prosté dosazení připravených datagramů a ověření reakce pohonu na jednotlivé příkazy
- Dekódování přijatých odpovědních datagramů a ověření jejich správnosti

Takto byli získány podklady pro vytvoření dalších, mnohem složitějších verzí uživatelské komunikace. V této komunikaci se pak vyčtené hodnoty interpretují přímo v příslušných jednotkách např. hodnota odporu pasivního čidla jako teplota ve ºC či zdvih servopohonu (0..10 000) jako % z jmenovitého rozsahu. Kromě tohoto „polidštění“ interpretace naměřených hodnot byli přidány i funkce, které maximálně automatizují provádění některých operací, jako je generování paritních bytů, seskládání celých telegramů, hlídání překročení timeoutů, identifikaci chybného odpovědního rámce apod.

9 UŽIVATELSKÁ KOMUNIKACE

V této části se budu věnovat popisu samotné uživatelské komunikace vytvořené v programu PSP3, která využívá tzv. rozhraní User COM za použití funkčních bloků COMxxx. Bohužel na detailnější vysvětlení způsobu programování a použitých funkčních bloků nebude v této práci dostatek prostoru, a proto musím čtenáře odkázat na webové stránky společnosti AMiT [2]. Na těchto stránkách naleznou jak samotný program, tak i manuály s popisem tohoto programu a způsobu programování, včetně popisu všech funkčních bloků.

9.1 Použité MP příkazy

Jak již bylo napsáno v předchozí kapitole, bylo nejprve potřeba udělat podrobnější analýzu příkazů, které budou následně využívány zákazníky při ovládání servopohonů. Jsou to příkazy potřebné pro nastavení MP adresy zařízení, pro zadání a vyčtení polohy pohonu a pro vyčtení hodnoty externího čidla či spínače. Jedná se tedy především o příkazy ze skupiny obecných příkazů. Po zapracování těchto příkazů do uživatelské komunikace jsem pak schopen přímo z řídicího systému např. ovládat pohon a vyčítat z něj informaci o poloze a hodnotě externího snímače.

U ostatních parametrů servopohonů, jako je kroutící moment, doba přestavení, pracovní rozsah apod., se potom předpokládá využití defaultního nastavení, nastavení ve výrobě dle přání zákazníka nebo nastavení pomocí parametrizačního nástroje PC-Tool či MFT-H.

Jak již bylo uvedeno v kapitole 2.3.2.2 pojednávající o podobě příkazu posílaného slavu, je MP příkaz složen ze start bytu, kódu příkazu, až šesti bytů parametrů a celý rámec je zabezpečen příčnou a podélnou paritou. Velikost příkazů je tak mezi 4 až 10 byty.

Na příkazový datagram poslaný z masteru v addressed módu odpovídá slave zpětným zasláním příkazového datagramu. Teprve za ním pak následuje odpovědní datagram. Ten je složen ze start bytu, až sedmi parametrů a dvou paritních bytů. Pokud se jedná o reakci na příkaz, na který slave nemá odpovídat žádným parametrem, posílá se jen start byte a dva paritní byty. Odpověď tedy následuje až za podélně paritním bytem příkazu a její velikost je od 3 do 10 bytů.

V tabulce (Tab. 17) jsou uvedeny použité příkazy i s jejich parametry. Jejich bližší popis si pak uvedeme v následujících kapitolách.

Tab. 17. Seznam použitých MP příkazů

Kód a název příkazu	**Parametry**	
	Posílané	**Obdržené**
50 : MP_Get_SeriesNo	Žádné	[1,2] rok a den [3,4] den a sériové číslo [5] skupina zařízení a zásuvný modul [6] typ zařízení [7] testovací pracoviště
38 : MP_Set_MP_Address	[1,2] rok a týden [3,4] den a sériové čís. [5] testovací pracoviš-tě [6] MP start byte	Žádné
37: MP_Set_Relative	[1,2] nastavení pozice [1/10 per mil]	Žádné
41: MP_Get_Relative	Žádné	[1,2] aktuální hodnota [1/10 per mil] [3,4] nastavená pozice [1/10 per mil]
59: MP_Get_Min_Mid_Max	Žádné	[1,2] Min [1/10 per mil] [3,4] Mid [1/10 per mil] [5,6] Max [1/10 per mil]

04: MP_AD_Convert	[1] funkce [2] volný = posílá hAA	[1,2] naměřená hodnota
75: MP_Get_Override_Control	Žádné	[1] "ZF" [2] "ZS"
14: MP_Set_Override_Control	[1] "ZF"	Žádné
26: MP_Get_Malfunction_Maintenance_State	Žádné	[1] Selhání / Servis

9.2 Vyhodnocení chyby na sedmé vrstvě

V uživatelské komunikaci se vyhodnocují dvě situace signalizující chybu v odpovědním paketu. První nastane pokud se v odpovědním start bytu objeví v jeho dolní polovině jiná hodnota než 0x_D a druhou pak je chyba na sedmé vrstvě tj. chyba typu neznámý příkaz, příkaz nepovolen (např. z důvodu nepřihlášení) či chyba v průběhu vykonávání příkazu. Tato chyba je signalizována nastavením jedničky na sedmém (nejvyšším) bitu start bytu, samozřejmě za podmínky, že dolní polovina bytu je 0x_D. Celý byte pak vypadá následovně: 1xxx1011. Příčina chyby je pak uvedena v následujícím parametr bytu jehož význam je popsán v (Tab. 13).

9.3 Použití uživatelské komunikace

Uživatelské komunikace budou dvě, z nichž jedna bude sloužit pouze pro zadání MP adresy servopohonu a druhá bude obsluhovat samotný servopohon.

Při tvorbě těchto dvou programů jsem sestavoval testovací datagramy a ověřoval správnou funkčnost na klapkovém servopohonu NM24-MFT(2) se sériovým číslem 00508-10023-062-076, který měl přidělenu MP adresu číslo 3. Proto i uvaděné příklady budou právě pro tento servopohon.

Program pro nastavení MP adresy je univerzální pro všechny typy pohonů, zatímco program pro obsluhu servopohonů je schopen ovládat „pouze“ klapkové a ventilové servopohony. Jeho úprava pro ovládání pohonů VAV jednotek by však nebyla složitá a jednalo by se pouze o nahrazení některých příkazů jejich ekvivalentem pro tento typ pohonů. Tyto příkazy, ač nejsou zapracovány do uživatelské komunikace, jsou vždy uvedeny u popisu příslušných příkazů pro ventilové a klapkové pohony. Bohužel jsem však nemohl ověřit funkčnost těchto příkazů, jelikož jsem neměl k dispozici žádný pohon tohoto typu.

9.4 Uživatelská komunikace pro nastavení MP adresy zařízení

Přiřazení MP adresy je prvním krokem k tomu, aby zařízení vůbec začalo fungovat v MP módu. Jakmile má totiž zařízení přiřazenu MP adresu, přestává na vodiči U5 fungovat analogový provoz a přechází se na provoz digitální pomocí protokolu MP-Bus.

Jelikož každý pohon v síti MP-Bus musí být jednoznačně identifikovatelný, je třeba mu přidělit vlastní jedinečnou adresu od 1 do 8. Pohon lze identifikovat i pomocí sériového čísla, což je ovšem vzhledem k jeho délce značně nepraktické. Naproti tomu MP adresa je krátká a výstižná. Při adresování je pak třeba zajistit, aby se v jedné síti neopakovali dvě stejné adresy, jinak by došlo k chybě komunikace.

MP adresu je možno zařízení přidělit třemi způsoby. Prvním je možnost ji na přání zákazníka nastavit přímo ve výrobě, kdy je poté uvedena i na výrobním štítku. Druhou možností je její nastavení pomocí konfiguračních nástrojů PC-Tool či MFT-H. Třetí možností je potom její nastavení z MP masteru. Tato možnost byla popsána v kapitole 1.4 věnované adresování MP zařízení.

Pro nakonfigurování MP adresy z masteru, což je náš případ, existují dvě možnosti. První tzv. poloautomatická nelze v našem případě využít, jelikož námi použitý hardware (řídicí systém AMiNi-E a převodník ZIP-232-MP) nedisponuje funkcí umožňující nastavení adresy, jako je tomu např. u zařízení UK24LON, na kterém se zadá požadovaná adresa a poté se potvrdí tlačítkem SET. Proto se zde budeme zabývat druhým způsobem přidělení MP adresy a to pomocí ručního zadání seriového čísla.

Pro zadání MP adresy je třeba dvou MP příkazů, které si blíže popíšeme v následujících dvou kapitolách. V příkladech uvedených u těchto příkazů je pro vyčtení sériového čísla využit addressed komunikační mód a pro zadání MP adresy broadcast komunikační mód.

Tyto módy jsou zvoleny záměrně, protože jsou nezbytné pro námi zvolený postup přidělení MP adresy.

9.4.1 Popis příkazu pro vyčtení sériového čísla (MP_Get_SeriesNo)

Tab. 18. Rozbor příkazu pro vyčtení sériového čísla

Kód a název příkazu	**Parametry**	
	Posílané	**Obdržené**
50 : MP_Get_SeriesNo	Žádné	[1,2] rok a týden [3,4] den a sériové číslo [5] skupina zařízení a typ zásuvného modulu [6] typ zařízení [7] testovací pracoviště

Každý servopohon má své specifické sériové číslo uložené v jeho EEPROM a vytištěné na štítku pohonu. Z tohoto jedinečného čísla se dají vyčíst data vztahující se k výrobě pohonu a zároveň slouží jako sekundární adresa, pomocí které lze pohonu přidělit MP adresu.

Sériové číslo uvedené na štítku mnou použitého pohonu bylo 00508-10023-062-076. V následujících tabulkách jsou vysvětleny hodnoty jednotlivých bytů parametrů, které vyčtu z tohoto pohonu pomocí příkazu MP_Get_SeriesNo.

Tab. 19. Rozbor sériového čísla vyčteného pomocí příkazu MP_Get_SeriesNo

Byte 1,2			**Byte 3,4**		**Byte 6**	**Byte 7**
Prefix	Rok	Týden	Den v týdnu	Pořadí ve dni	Typ zařízení	Testovací pracoviště
0	05	08	1	0023	062	076
Možné parametry	00 – 99	01 – 53	0 - 6 = Po-Ne	1 - 9999	0 - 254	0 - 254

Byte 5, jehož hodnota není uvedena na výrobním štítku, mi pak udává kategorii zařízení a typ zabudovaného modulu. To umožňuje identifikovat speciální funkce pohonu.

Tab. 20. Význam pátého bytu

Dolní polovina bytu (hexadecimálně)	
Hodnota	**Kategorie**
0	Volný
1	Universální vstupně / výstupní modul
2	Přímo připojitelný pohon / pohon se zpětnou pružinou, vzduch
3	Zdvihový pohon lineární
4	Zdvihový pohon rotační
5	Okenní ventilační systém
6	Požární a odkuřovací klapkové pohony
7	Volný
8	VAV jednotky pro řízení průtoku vzduchu
9 ... F	Volný
Horní polovina hexadecimálního bytu	
Hodnota	**Zabudovaný modul**
0	Žádný
1	Modul SR-95
2	Phasecut modul (fázový dělič)
3	Y modul
4	Modul měření teploty
5	LON modul
6	Modul 4..20mA
7	BAE-2 modul
8	LEN modul

9	Hartmann modul
A...F	Volný (neznámý modul)

Byte 6 označuje typ zařízení a má význam při finálním testování, kdy se podle něj určuje jaké limitní hodnoty budou uloženy do pohonu.

Tab. 21. Význam šestého bytu

Hodnota	Typ zařízení
1	NMV-D2
2	NMV-D2M
3	VRP-M
16	VRD2
17	VRD2-L
18	VRD2M
24	GM-MFT
60	LM_MFT
62	NM-MFT
63	NMQ-MFT (s rychlým časem přeběhu)
64	AM-M1
65	AM-MFT
66	AMQ-MFT
67	AM-MFT shodný s typem AM-M1 – jeho náhrada pro USA
68	AM-MFT shodný s typem AM-M1 – jeho náhrada pro EU
80	LF-MFT
82	NF-MFZT
84	AF-MFT

85	AF24-SR (=> nástupce staršího AF24-SR s implementovaným MP-Busem, ve kterém jsou všechny příkazy chráněny heslem)
86	AF24-V
87	TF24-MP
88	TF24-SR, TF24-MF
100	BLF-MFT-Top
104	BF-MFT-Top
105	BFG-MFT-TL
106	BF-SR
112	ZEVO
120	BE-MFT
128	NV-MFT (800N)
129	NVF-MFT (800N)
130	NVF-MFT-E (800N)
131	NVS-MFT (1200N)
132	AV-MFT
136	Okenní pohony
252	Senzory jiných výrobců
253	UST3 (Univerzální I/O modul na MP-Bus)
254	MFT-H

Sestavení datagramu pro vyčtení sériového čísla z mého zařízení na adrese 3

Poslaný datagram na vyčtení sériového čísla obsahuje pouze čtyři byty: start byte, kód příkazu a dva paritní byty. Odpovědní paket pak dává celkem sedm bytů parametrů, které nám udávají rok, týden a den výroby, pořadí výrobku ve dni, kategorii výrobku, jestli a jaký je v něm zabudovaný speciální modul, typ pohonu a číslo pracoviště na kterém byl pohon testován. Datagram jsem vytvořil dle postupu, který je uveden v kapitole 2.4.1.1.

Generce MP příkazového paketu:

Start byte: 0b00011000 = 0h18

pro adresu 3 a jeden užitečný byte

Kód příkazu: 0b00110010 = 0h32

Příkaz číslo 50

Příčná parita: 0b10000000 = 0h80

Podélná parita: 0b10101010 = 0hAA

MP-Bus telegram:

Master tedy posílá hexadecimální hodnoty: 18 32 80 AA

Slave pak vrací hexadecimální hodnoty: 18 32 80 AA 7D 01 FC 27 27 32 3E 4C E2 22

Rozbor MP odpovědního paketu:

Start byte: 0h7D = 0b01111101

bude doručeno 7 užitečných bytů

[1,2] 0h01FC = 0b00000001 11111100 = 0d508

vyrobeno v 8 týdnu roku 2005

[3,4] 0h2727 = 0b00100010 00100010 = 0d1023

což bylo úterý a byl to 23 výrobek toho dne

[5] 0h32 = 0b00110010

jde o přímo připojitelný pohon se zabudovaným Y modulem

[6] 0h3E = 0b00111110 = 0d62

jedná se o typ servopohonu NM-MFT

[7] 0h4C = 0b01001100 = 0d76

servopohon testován na testovacím pracovišti číslo 76

Příčná parita: 0hE2 = 0b11100010

Podélná parita: 0h22 = 0b00100010

Vyčtené data tedy odpovídají údajům uvedeným na výrobním štítku 00508-10023-062-076 a navíc jsme se z pátého bytu dozvěděli, že toto zařízení patří do kategorie přímo připojitelných pohonů a je opatřeno Y modulem.

9.4.2 Popis příkazu pro nastavení MP adresy (MP_Set_MP_Address)

Tab. 22. Rozbor příkazu pro nastavení adresy

Kód a název příkazu	**Parametry**	
	Posílané	**Obdržené**
38 : MP_Set_MP_Address	[1,2] rok a den [3,4] den a sériové číslo [5] testovací pracoviště [6] MP start byte kód	Žádné

Jelikož každý pohon v síti MP-Bus musí být jednoznačně identifikovatelný, je třeba mu přidělit vlastní jedinečnou adresu od 1 do 8. V našem případě chceme tuto MP adresu nastavovat pomocí tzv. manuálního adresování na základě znalosti sériového čísla, které je uvedeno na výrobním štítku pohonu a lze též vyčíst z EEPROM paměti servopohonu, což jsme si vysvětlili v předešlé kapitole.

Sériové číslo ze štítku je třeba zadat spolu se start byte kódem zvolené MP adresy do příkazu MP_Set_MP_Address a poslat ho po MP-Bus síti pomocí broadcast komunikačního módu. Tím je pohonu s daným sériovým číslem přidělena nastavená MP adresa.

Tab. 23. Start byte kódy adres

Adresa	PP	MP:1	MP:2	MP:3	MP:4	MP:5	MP:6	MP:7	MP:8
Start byte kód	H01	H00	H04	H08	H0C	H80	H84	H88	8HC

Sestavení datagramu pro zadání MP adresy 3 v broadcast komunikačním módu

Poslaný datagram pro zadání MP adresy obsahuje šest bytů s parametry, z nichž prvních pět odvodíme z výrobního čísla uvedeného na štítku pohonu. Tyto byty musí korespondovat s byty [1,2], [3,4] a [7], které jsme vyčetli pomocí příkazu MP_Get_SeriesNo v předešlé kapitole. Posledním šestým bytem je pak MP start byte kód pro vybrané číslo stanice viz (Tab. 23). Jelikož je tento příkazový paket posílán v broadcast režimu, slave tento příkaz pouze vykoná, ale na rozdíl od addressed módu na něj neodpoví.

Generce MP příkazového paketu:

Start byte: 0b01111001 = 0h79

broadcast komunikační mód a sedm užitečných bytů

Kód příkazu: 0b00100110 = 0h26

příkaz číslo 38

[1,2] 0b00000001 111111000 = h01FC

první část výrobního čísla (odpovídá bytům [1,2])

[3,4] 0b00100010 001000100 = h2727

druhá část výrobního čísla (odpovídá bytům [3,4])

[5] 0b01001100 = 0h4C

poslední část výrobního (odpovídá bytu [7])

[6] 0b00001000 = 0h08

MP start byte pro adresu 3

Příčná parita: 0b11000111 = 0hC7

Podélná parita: 0b00100001 = 0h21

MP-Bus telegram:

Master tedy posílá hexadecimální hodnoty: 79 26 01 FC 27 27 4C 08 C7 21

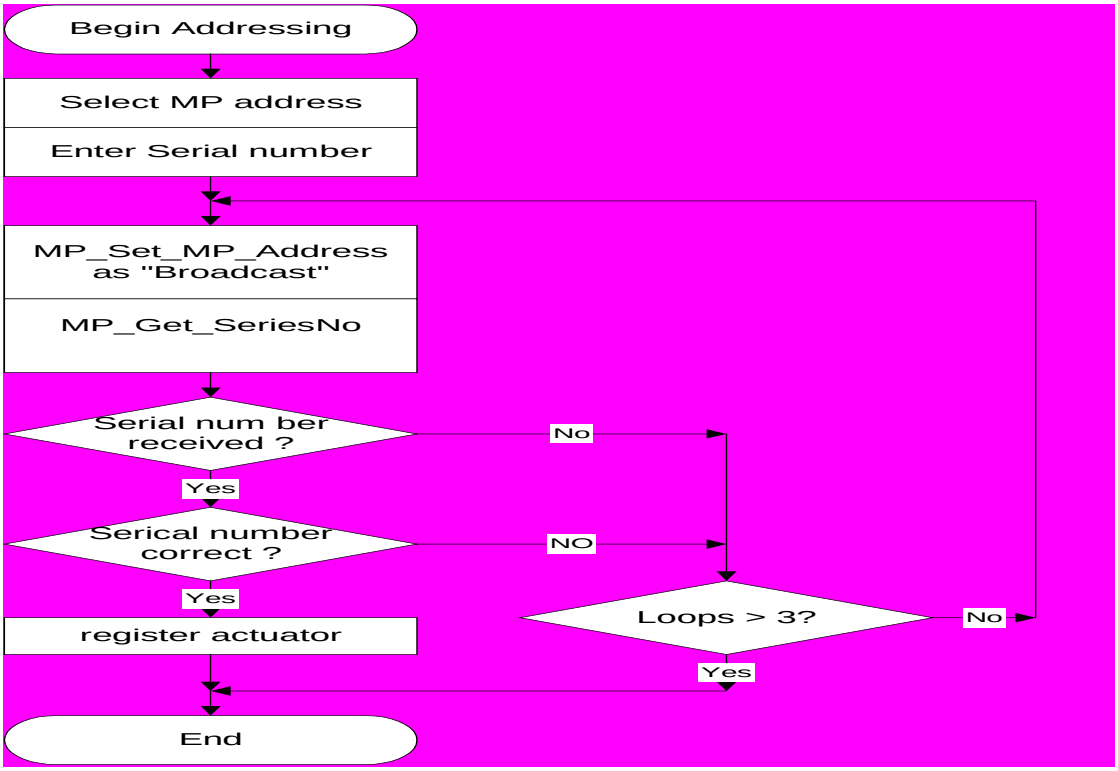
Tímto telegramem se tedy na stanici s výrobním číslem 00508-10023-062-076 nastaví adresa 3.

9.5 Samotná uživatelská komunikace pro nastavení MP adresy

Jak již bylo uvedeno, jde v této uživatelské komunikaci o adresování s ručním zadáním známého sériového čísla a při tomto postupu není třeba zmáčknutí tlačítka přímo na pohonu, takže při znalosti sériového čísla není ani potřeba být fyzicky u tohoto pohonu.

Tato komunikace využívá dvou příkazů, které byli blíže popsány v předešlých dvou kapitolách, a to příkazu pro zadání MP adresy a vyčtení sériového čísla. Princip této adresace spočívá v zadání celého sériového čísla, tak jak je uvedeno na štítku pohonu a start byte kódu pro vybranou MP adresu. Tyto hodnoty se pak použijí jako parametry příkazu MP_Set_MP_Addres, který slouží k nastavení MP adresy. Jelikož se tento příkaz posílá v broadcast módu a slave na něj tudíž neodpovídá, je potřeba zajistit kontrolu správného přidělení adresy. To se provádí příkazem pro vyčtení sériového čísla MP_Get_SeriesNo, který se posílá na předešlým příkazem přidělenou adresu. Vyčtené sériové číslo se potom porovnává se sériovým číslem, které bylo zadáno do parametrů příkazu MP_Set_MP_Address.

Při sestavování algoritmu pro tuto komunikaci jsem vycházel z návodu uvedeného v dokumentaci [6].



Obr. 32. Doporučený algoritmus pro nastavování MP adresy

Využití

V případě, kdy budu za pomocí parametrizačního zařízení měnit před instalací parametry pohonu, tak pravděpodobně současně zadám i MP adresu a nebudu tedy tuto uživatelskou komunikaci vůbec potřebovat. Pokud však budu používat pohony s továrním nastavením, a nebudu tak potřebovat konfigurační zařízení k žádnému jinému účelu než k adresaci, nebo budu chtít změnit adresy již nasazených pohonů (zejména v případě dálkové zprávy třeba prostřednictvím internetu), může mi přijít tato komunikace velmi vhod.

Postup při zadávání MP adresy pomocí uživatelské komunikace (více v příloze P I)

Nejprve zadám v procesu ProcINIT do buněk připravené matice „SC“ sériové číslo, tak jak je uvedeno na štítku. Do jednotlivých buněk dám vždy jednu část sériového čísla, které je rozděleno na části pomlčkami. První buňka se pak bere jako byty [1,2], druhá buňka jsou byty [3,4], třetí buňka se ignoruje a čtvrtá buňka je byte [5]. Do proměnné „Adresa“ potom zadám zvolenou MP adresu jako číslo od 1 do 8. Pro deadresaci, čili přechod do PP módu, zadám hodnotu 0. Takto zadané číslo se samo přepočte na hodnotu start byte kódu, který se pak použije jako byte [6]. Takto získám všechny parametr byty potřebné pro příkaz MP_Set_MP_Address. Ty se pak přidají k pevně zadanému start bytu a příkazovému bytu, které se nemění. V posledním kroku se pak dopočítají paritní byty. Takto sestavený datagram se pošle na pohon a nastaví se mu adresa.

Jelikož byl příkaz poslán v broadcast komunikačním módu, tak na něj nepřijde žádná odpověď. Proto se využije druhého příkazu MP_Get_SeriesNo, kterým se vyčte sériové číslo z pohonu, který byl právě naadresován. Celý datagram pro tento příkaz se sestaví sám. Start byte kód je totiž stejný jako byte [6] z předešlého příkazu, který máme v proměnné „Adresa“ a pouze se k němu přičte hexadecimální hodnota 0x10. Potom se už jen přidá kód příkazu a dopočítají se paritní byty. Posledním krokem je porovnání parametrů z odpovědního datagramu s parametry zadanými jako sériové číslo do příkazu MP_Set_MP_Address. Kromě toho si ještě program hlídá také chybu na sedmé vrstvě (viz kapitola 9.2).

Pokud proběhne vše správně, nastaví se proměnná OK na hodnotu 1.

Celý proces adresování probíhá jen jednou, kdy v programu nastavím požadované hodnoty, provedu generaci programu, nahraji jej do řídicího systému a ten při prvním běhu programu nastaví požadovanou adresu.

Celý program i s dalším popisem lze nalézt v příloze P I.

Datagramy

Pokud použiji stejné hodnoty jako u příkladů z předešlých dvou kapitol, budou posílané a vyčítané datagramy vypadat následovně:

Příkaz MP_Set_MP_Address: 79 26 ***01 FC 27 27 4C*** **08** C7 21

Příkaz MP_Get_SeriesNo: **18** 32 40 6A

Odpověď na příkaz MP_Get_SeriesNo: 18 32 40 6A 7D ***01 FC 27 27*** 32 3E ***4C*** E2 22

9.6 Uživatelská komunikace pro obsluhu servopohonu

V této části se podrobněji podíváme na uživatelskou komunikaci pro samotné ovládání servopohonů.

Zatímco předešlá uživatelská komunikace pro nastavení MP adresy pohonu nemusí být uživatelem vůbec využita, jelikož existují i jiné možnosti jak MP adresu nastavit, tak tato komunikace bude užitečná všem, kteří chtějí ovládat servopohony Belimo po MP-Bus pomocí řídicích systémů firmy AMiT. Součástí této komunikace je především zadávání polohy servopohonu, vyčtení aktuální a nastavené pozice pohonu, vyčtení hodnoty externího čidla či spínače a navíc se také hlídají případné poruchové hlášení.

V následujících kapitolách se podrobněji podíváme na jednotlivé příkazy, které jsou použity pro ovládání pohonů pomocí této uživatelské komunikace.

9.6.1 Popis příkazu pro nastavení pozice pohonu (MP_Set_Relative)

Tab. 24. Rozbor příkazu pro nastavení pozice pohonu

Kód a název příkazu	Parametry	
	Posílané	**Obdržené**
37: MP_Set_Relative	[1,2] nastavení pozice [1/10 per mil]	Žádné

Tento příkaz slouží k zadání požadované pozice servopohonu ve formě procent z nominálního rozsahu (úhlu u klapkových pohonů, zdvihu u ventilových pohonů a objemového průtoku u VAV jednotek). Požadovaná pozice z rozmezí 0..100 % se zadává jako číslo mezi 0...10 000. Pokud je však pracovní rozsah pohonu změněn, zadáním min a max ve výrobě či pomocí parametrizačního zařízení, tak se rozmezí 0..100 % přizpůsobuje tomuto zvolenému min a max, kde max odpovídá 100 %.

Následná akce po zadání příkazu pak závisí na vybraném operačním módu, tzn. zda řídíme dle pozice nebo dle objemového průtoku.

Sestavení datagramu pro zadání pozice pohonu na adrese 3

Následující datagram říká pohonu na adrese 3, aby se nastavil na 60 % svého rozsahu, tzn. že musím poslat hodnotu 6 000, která se zadává do dvou parametr bytů příkazu. Na tento příkaz slave neodpovídá žádnými parametry, ale pouze potvrdí správnost přijetí zasláním start bytu a paritních bytů.

Generce MP příkazového paketu:

Start byte: 0b00111000 = 0h38

pro adresu 3 a tři užitečné byte

Kód příkazu: 0b00100101 = 0h25

příkaz číslo 37

[1,2] 0b00010111 01110000 = 0h1770

nastavení pohonu na 60 % nominálního rozsahu

Příčná parita: 0b10110000 = 0hB0

Podélná parita: 0b11001010 = 0hCA

MP-Bus telegram:

Master tedy posílá hexadecimální hodnoty: 38 25 17 70 B0 CA

Slave pak vrací hexadecimální hodnoty: 38 25 17 70 B0 CA 0D 80 8D

Tímto telegramem jsme tedy nastavili servopohon na 60 % rozsahu mezi min a max.

9.6.2 Popis příkazu pro vyčtení pozice pohonu (MP_Get_Relative/MP_Get_VRelative)

Tab. 25. Rozbor příkazu pro vyčtení pozice pohonu

Kód a název příkazu	**Parametry**	
	Posílané	**Obdržené**
41: MP_Get_Relative	Žádné	[1,2] aktuální pozice = [1/10 per mil] [3,4] požadovaná pozice = [1/10 per mil]
57: MP_Get_Vrelative (linearizovaný relativní VAV signál)	Žádné	[1,2] aktuální hodnota objemového průtoku Vflow =[1/10 per mil] [3,4] požadovaná hodnota Vflow = [1/10 per mil]

Existují dva příkazy pro vyčtení pozice. První z nich MP_Get_Relative slouží k vyčtení pozice pohonu v % (úhlu u klapkových a zdvihu u ventilových pohonů), zatímco druhý příkaz MP_Get_VRelative prezentuje hodnotu objemového průtoku u VAV jednotek.

U VAV jednotek je potřeba použít příkazu MP_Get_VSettings, kterým nevyčítám polohu, ale objemový průtok. Je zde potřeba zadat pouze jiný příkazový byte (0h39). V odpovědním paketu pak nevyčítám aktuální a zadanou pozici, ale objemový průtok. Upozornění: relativní aktuální hodnota zde může být větší než 10 000. Jak již bylo uvedeno v kapitole 9.3, budu se zde věnovat pouze rozboru příkazu MP_Get_Relative.

Jako odpověď na příkaz MP_Get_Relative obdržím nastavenou a aktuální pozici pohonu. Nastavenou pozici vyčtu z bytů [3,4] a aktuální pozici z bytů [1,2]. Takto vyčtené hodnoty v rozmezí 0..100 % odpovídají nominálnímu rozsahu (úhlu nebo zdvihu) a ne rozsahu mezi min a max jako při zadávání pozice. Tento rozdíl v interpretaci hodnot má při změněném pracovním rozsahu za následek, že vyčtená hodnota není rovna hodnotě zadané.

Platí totiž následující vztahy:

- **Min / Vmin**
  0..10 000 odpovídá 0%..max (neodpovídá nominálnímu rozsahu)
- **Mid**
  0..10 000 odpovídá min..max (neodpovídá nominálnímu rozsahu)
- **Max / Vmax**
  Povolený rozsah:
  - servopohony: 0.. 10 000 odpovídá 0..100%
  - VAV jednotky: 0..12 000 odpovídá 0.. 120 %
- **Řízení na konstantní objemový průtok**

  Když je Vmin = 100% = Vmax nezáleží pak na nastavené hodnotě a objemový průtok je regulován stále na Vmax
- **Vmax je v 1/10 per mil z Vnom**

Příklad pro přepočet vyčteného nominálního rozsahu na rozsah mezi min a max

Příkazem MP_Set_Relative byl zadán úhel 60 % tj. 6 000 při nastavené hodnotě min = 20 % a max = 80 %.

Vyčtená hodnota příkazem MP_Get_Relative pro daný rozsah je 5 440 a ne zadaných 6 000.

Vysvětlení:

Je to dáno tím, že zatímco max = 80 % tj. 8 000 se počítá ze 100 % tj. 10 000, hodnota min = 20 % už se počítá jako 20 % z 8 000 (max) a je tedy 1 600.

Zadaná poloha 60 % odpovídá 60 %-ům ze zvoleného rozsahu min a max, zatímco vyčtená hodnota z pohonu odpovídá nominálnímu rozsahu, kde 0..100 % odpovídá 0..10 000. Vyčtená pozice odpovídá následujícímu výpočtu:

Nastavená pozice [3,4] = aktuální pozice [1,2] = (MP_Set_Relative [1,2] / 10 000) x {max – ((min x max) / 10 000)} + ((min x max) / 10 000)

Pokud chci, aby vyčtená data byla přizpůsobena zadanému rozsahu min a max, tak je musím přepočíst dle vztahu:

% z rozsahu min a max = (1000000 x aktuální pozice [1,2] – 100 x min x max) / (100000 x max – min x max)

Uvedený vzorec je pro výpočet aktuální pozice. Pro nastavenou pozici je třeba dosadit byty [3,4].

Sestavení datagramu pro vyčtení aktuální a nastavené pozice z pohonu na adrese 3

Posílaný příkaz MP_Get_Relative neobsahuje žádný parametr a pouze jím žádám o zaslání odpovědního paketu. V něm pak dostávám hodnotu aktuální pozice v bytech [1,2] a zadané pozice v bytech [3,4]. Stejně jako v příkladu pro zadávání pozice použiji i zde hodnotu 60 % tj. 6 000. Předpokladem je, že pracovní rozsah není omezen, a že pohon už se přesunul na zvolenou pozici a aktuální pozice je tak rovna pozici zadané.

Generace MP příkazového paketu:

Start byte: 0b00011000 = 0h18

pro adresu 3 a jeden užitečný byte

Kód příkazu: 0b00101001 = 0h29

příkaz číslo 41

Příčná parita: 0b10000000 = 0h80

Podélná parita: 0b10110001 = 0hB1

MP-Bus telegram:

Master tedy posílá hexadecimální hodnoty: 18 29 80 B1

Slave pak vrací hexadecimální hodnoty: 18 29 80 B1 4D 17 70 17 70 A0 ED

Rozbor MP odpovědního paketu:

Start byte: 0h4D = 0b01001101

budou doručeny 4 užitečné byty

[1,2] 0h1770 = 0b00010111 01110000

aktuální pozice je 6 000

[3,4] 0h1770 = 0b00010111 01110000

zadaná pozice byla 6 000

Příčná parita: 0hA0 = 0b10100000

Podélná parita: 0hED = 0b11101101

Vyčtené data mi udávají stejnou hodnotu pro aktuální i zadanou polohu a shodují se také s polohou zadanou a tím odpovídají mnou zadaným podmínkám.

9.6.3 Popis příkazu pro vyčtení nastavené hodnoty min a max (MP_Get_Min_Mid_Max)

Tab. 26. Rozbor příkazu pro vyčtení hodnoty min a max

Kód a název příkazu	**Parametry**	
	Posílané	**Obdržené**
59: MP_Get_Min_Mid_Max	Žádné	[1,2] Min [1/10 per mil] [3,4] Mid [1/10 per mil] [5,6] Max [1/10 per mil]

30: MP_Get_Vsettings (pro VAV jednotky)	Žádné	[1,2] Vnom [m3/h] [3,4] Vmax [1/10 per mil] [5,6] Vmin [1/10 per mil]

Jak již bylo zmíněno v předchozí kapitole, je možno na pohonu omezit jeho pracovní rozsah zadáním hodnot pro min a max (dolní a horní mez). Tomuto rozsahu se pak přizpůsobuje i nastavená doba přeběhu, což dovoluje paralelní běh pohonů.

Takto přenastavený pracovní rozsah však má vliv na interpretaci vyčítaných údajů o pozici pohonu, což jsme si ukázali na příkladu v předešlé kapitole. Proto si zde popíšeme datagram, kterým se vyčítají hodnoty min a max, která se potom použijí do vzorce na přepočet.

Sestavení datagramu pro vyčtení zadaného min a max z pohonu na adrese 3

Příkazem MP_Get_Min_Mid_Max si vyčtu hodnotu nastavené minimální, střední a maximální pozice pohonu. V mém případě se bude jednat o vyčtení továrního nastavení, kdy min = 0, mid = 5 000 a max = 10 000.

Generace MP příkazového paketu:

Start byte: 0b00011000 = 0h18

pro adresu 3 a jeden užitečný byte

Kód příkazu: 0b00111011 = 0h3B

příkaz číslo 59

Příčná parita: 0b10000000 = 0h80

Podélná parita: 0b10100011 = 0hA3

MP-Bus telegram:

Master tedy posílá hexadecimální hodnoty: 18 3B 80 A3

Slave pak vrací hexadecimální hodnoty: 18 3B 80 A3 6D 00 13 88 27 10 92 53

Rozbor MP odpovědního paketu:

Start byte: 0h6D = 0b01101101

bude doručeno 6 užitečných bytů

[1,2] 0h0000 = 0b00000000 00000000 = 0d0

hodnota minima

[3,4] 0h1388 = 0b00010011 10001000 = 0d5000

střední hodnota

[5,6] 0h2710 = 0b00100111 00010000 = 0d10000

hodnota maxima

Příčná parita: 0h92 = 0b10010010

Podélná parita: 0h53 = 0b01010011

Z pohonu jsem vyčetl hodnotu nastaveného min = 0, mid = 5000 a max = 10000, které odpovídají zvolenému továrnímu nastavení.

9.6.4 Popis příkazu pro vyčtení hodnoty externího čidla (MP_AD_Convert)

Tab. 27. Rozbor příkazu pro vyčtení hodnoty externího čidla

Kód a název příkazu	**Parametry**	
	Posílané	**Obdržené**
04: MP_AD_Convert	[1] Funkce [2] Volný = posílá se hAA	[1,2] naměřená hodnota
76: MP_Get_Vist_Nonlin Jen pro NMV-D2 , NMV-D2M a VRD2	Žádné	[1,2] nelineární naměřená hodnota senzoru D2 [3,4] linearizovaná hodnota senzoru D2

Prostřednictvím MP-Busu mohou být vyčítány různé analogové signály včetně D2 senzorů, BAE-2, napájení apod.. Možné rozsahy připojitelných senzorů jsou:

- Aktivní: 0....32000 mV (pro signály aktivních senzorů v rozmezí 0...32 V)

- Pasivní: 850 - 1600 Ohm (pro odporové senzory s rozsahem 850 - 1600 Ohm, např. Ni1000, Pt1000, Mo1000)

- Pasivní: 0,1 – 60 kOhm (pro odporové senzory s rozsahem 0,1 - 60 kOhm, např. NTC termistor 1 – 20 kOhm)

- Kontakt: pro ZAP/VYP kontakty – provádí se příkazem MP_Get_Override_Control viz následující kapitola

MP-Bus dovoluje vyčítat až dva aktivní či pasivní senzory. Podmínkou je, aby byli stejného typu.

Měření probíhá pomocí do pohonu zabudovaného Y modulu a senzor je k pohonu připojen přes svorku Y. Vyčtení signálu trvá 100 ms a je opakováno nejméně každých 1,5 s. Aktivace měření teploty je automatická, kdy se Y modul aktivuje při prvním požadavku na vyčtení hodnoty pomocí příkazu MP_AD_Convert. Jelikož čas ustálení hardwareového filtru je relativně dlouhý, je důležité aby následující vyčtení proběhlo až po tomto ustálení.

Je třeba si dát pozor jestliže vyčtení hodnoty senzoru pomocí příkazu MP_AD_Convert následuje po příkazu MP_Get_Override_Control (nadřazené řízení). Oba tyto příkazy totiž využívají stejný odpor 1k5 a pro správné měření je potřeba nejdříve vypnout napájení.

V posílaném příkazu MP_AD_Convert musím dle typu vyčítaného senzoru dosadit do prvního bytu parametru správnou hodnotu dle následující tabulky.

Tab. 28. Hodnota funkce posílané v prvním bytu parametru

Vyčítaná funkce (0..255)	**Rozlišení signálu**	**Měřená jednotka**
0: napětí D2 senzoru u vnitřních VAV jednotek	300 µV	[100 µV]
1: proud tekoucí D2 senzorem u vnitřních VAV jednotek	300 µV	[100 µV]
2: rozlišení modulu (pro modul SR95 a 20mA)	32 mV	[mV]
3: dodávané napájení (vnitřní hodnota)	32 mV	[mV]
4: Hodnota vstupu Y	32 mV	[mV]

5: analogový výstup (nastavení U5, není měřeno)	32 mV	[mV]
6: rozlišení modulu (pro moduly Y, BAE2, Phasecut a LON)	300 μV	[100 μV]
7 -15: Volný		
16: Senzor 1 na vstupu Y (Pt1000, Mo1000, Ni1000)		Ω
17: Senzor 1 na vstupu Y (NTC termistor 1k..20k)		Ω
18: Senzor 2 (Pt1000, Mo1000, Ni1000)		Ω
19: Senzor 2 (NTC termistor 1k..20k)		Ω

Pasivní senzory tedy mohou být vyčítány funkcemi 16..19 a aktivní senzory funkcí 4. Toto čtení pak deaktivuje případné nadřazené řízení (příkaz MP_Get_Override_Control), které se opět aktivuje zapnutím napájení.

Hodnoty signálů se potom vyčítají následovně:

- Hodnota signálu aktivního senzoru 0..32000 mV je čtena pomocí funkce 4 a posílaný parametr má hexadecimální hodnotu 04 AA. Z pohonu pak obdržím dva byty s hodnotou napětí v mV.
- Vyčítání odporů s rozsahem 850..1600 Ohm se provádí funkcí 16 pro první senzor a 18 pro případný druhý senzor. Posílaný hexadecimální parametr pro jeden senzor pak je 16 AA a vyčítaná dvou bytová mi udává odpor v ohmech.
- Vyčítání odporů s rozsahem 0,1..60 kOhm (NTC 1 kOhm – 20 kOhm) probíhá pomocí funkce 17 pro první senzor a 19 pro případný druhý senzor. Posílaný hexadecimální parametr pro jeden senzor pak je 17 AA a nazpět potom vyčítám dva byty, ve kterých je hodnota odporu v ohmech.

Sestavení datagramu pro vyčtení teploty z pasivního senzoru Ni1000 připojeného na pohonu s adresou 3 při teplotě 0 °C

Příkazem MP_AD_Converter posílám na pohon žádost o vyčtení zvoleného typu senzoru. V našem případě je to pasivní senzor Ni1000. Dle tabulky (Tab. 28) tedy zadám jako první byte parametru hexadecimální hodnotu 16.

Generace MP příkazového paketu:

Start byte: 0b00111000 = 0h38

pro adresu 3 a tři užitečné byty

Kód příkazu: 0b00000100 = 0h04

příkaz číslo 4

[1] 0b00010000 = 0h10

použitý senzor má odpor v rozsahu 850....1600 Ohm

[2] 0b10101010 = 0hAA

vždy se dosazuje hexadecimální hodnota AA

Příčná parita: 0b01110000 = 0h70

Podélná parita: 0b11110110 = 0hF6

MP-Bus telegram:

Master tedy posílá hexadecimální hodnoty: 38 04 10 AA 70 F6

Slave pak vrací hexadecimální hodnoty: 38 04 10 AA 70 F6 2D 03 E8 80 8D

Rozbor MP odpovědního paketu:

Start byte: 0h2D = 0b00101101

budou doručeny 2 užitečné byty

[1,2] 0h03E8 = 0b00000011 11101000 = 0d1000

hodnota odporu v ohmech

Příčná parita: 0h00 = 0b00000000

Podélná parita: 0hC6 = 0b11000110

V odpovědi se mi vrací vyčtená hodnota odporu 1 000 Ω, která se přepočítá na teplotu pomocí převodní tabulky. Tato hodnota odporu odpovídá teplotě 0ºC.

9.6.5 Popis příkazu pro nastavení funkce vyčtení polohy externího kontaktu (MP_Set_Override_Control)

Tab. 29. Rozbor příkazu pro nastavení funkce vyčtení polohy externího kontaktu

Kód a název příkazu	Parametry	
	Posílané	Obdržené
14: MP_Set_Override_Control	[1] "ZF"	Žádné

Tímto příkazem se nastavuje operace, která se má vykonat při nadřazeném řízení jako reakce na stav vstupu Y, jenž je ovládán přes nadřazený obvod (externí přepínač). Může se jednat o složitější obvody s pěti polohovým přepínačem či pouze o signál z okenního spínače, kdy se při otevřeném okně dá povel k uzavření pohonu. K vyčítání této funkce může být použito jak stejnosměrné, tak i střídavé napájení.

Nadřazené řízení lze nastavit jak hardwarovým spínačem tak i softwarově a to pomocí příkazu MP_Set_Forced_Control (70).

Jednotlivé akce které se dají přiřadit pohonu jako reakce na příslušnou událost, jsou uvedeny v následující tabulce. Hodnota „ZF" se potom posílá jako jediný parametr příkazu.

Tab. 30. Hodnoty parametru „ZF"

"ZF"	Akce provedená pohonem	Hodnoty vyčtené příkazy MP_Get_Relativ / MP_Get_VRelativ
0	Žádná	MP nebo analogové nastavení pozice
1	Otevření	32000
2	Zavření	33536
3	přechod do max nebo Vmax (dle operačního módu)	Dle vypočtené minimální, střední a maximální pozice

4	přechod do min nebo Vmin (dle operačního módu)	Dle vypočtené minimální, střední a maximální pozice
5	přechod do mid nebo Vmid (dle operačního módu)	Dle vypočtené minimální, střední a maximální pozice
6	-	
7	Přechod na 100% adaptovaného rozsahu nebo nominálního objemového průtoku Vnom (dle operačního módu)	10000
8	Rychlé zavření	33536
9	Rychlé otevření	32000
10	Stop	Odpovídá minimu
11	Rychlé zavření (u požárních klapek)	33536
12	Rychlé otevření (u kouřových odvětrávacích klapek)	32000

Jelikož v našem případě budeme pouze chtít číst polohu externího kontaktu a nebudeme chtít využívat funkce nadřazeného řízení, zadáme do posílaného parametru hodnotu 0.

Sestavení datagramu pro nastavení funkce vyčítání polohy externího kontaktu na pohonu s adresou 3

Na pohon se posílá jediný parametr, jehož hodnota je pro případ vyčítání stavu externího kontaktu rovna 0.

Generace MP příkazového paketu:

Start byte: 0b00101000 = 0h28

pro adresu 3 a dva užitečné byte

Kód příkazu: 0b00001110 = 0h0E

Příkaz číslo 14

[1] 0b00000000 = 0h00

aktivace vyčítání stavu kontaktu

Příčná parita: 0b01000000 = 0h40

Podélná parita: 0b01100110 = 0h66

MP-Bus telegram:

Master tedy posílá hexadecimální hodnoty: 28 0E 00 40 66

Slave pak vrací hexadecimální hodnoty: 28 0E 00 40 66 0D 80 8D

Na pohonu byla nastavena funkce pro vyčítání hodnoty externího kontaktu.

9.6.6 Popis příkazu pro vyčtení polohy externího kontaktu (MP_Get_Override_Control)

Tab. 31. Rozbor příkazu pro vyčtení polohy externího kontaktu

Kód a název příkazu	**Parametry**	
	Posílané	**Obdržené**
75: MP_Get_Override_Control	Žádné	[1] "ZF" [2] "ZS"

Jak již bylo uvedeno, tak při nadřazeném řízení prostřednictvím vstupu Y, bývá různým událostem na tomto vstupu přiřazena různá akce jako je otevření, zavření, nastavení max, mid, min pozice apod.

My však chceme tímto vstupem pouze vyčítat stav externího přepínacího kontaktu a nebudeme chtít vykonávat žádnou následnou nadřazenou akci. Proto jsme v příkazu MP_Set_Override_Control zadali do parametru "ZF" nulovou hodnotu. Tento parametr pak obdržíme jako první byte odpovědi na příkaz MP_Get_Override_Control. Druhý obdržený byte „ZS" nás potom informuje o stavu, ve kterém se kontakt nachází viz (Tab. 32).

Tab. 32. Hodnota parametru „ZS“ v závislosti na stavu vstupu Y

Hodnota parametru “ZS“	**Nadřazený signál na vstupu Y**
0	Žádný (0,5....32 V DC)
1	Otevřený obvod (počáteční hodnota Y musí být při SR/SRS řízení větší než 0,5 V)
2	0 V nebo GND (limity hysterze 120 mV a 220 mV)
3	24 V AC
4	Kladná půl vlna (24 V AC s diodou)
5	Záporná půl vlna (24 V AC s diodou)
6	Trvání pulsu menší než 0 % (u PWM řízení).
7	Trvání pulsu delší než zadané maximum (u PWM řízení)

Pro naše účely, kdy chceme pouze zjistit stav přepínacího kontaktu, se nám tabulka s přehledem všech možných stavů velmi zjednoduší a vypadá následovně.

Tab. 33. Hodnota parametru „ZS“ v závislosti na poloze kontaktu

Hodnota parametru “ZS“	**Poloha přepínače**
1-2	otevřen
Ostatní hodnoty	zavřen

Na stav přepínače si pak sami můžeme navázat akci kterou chceme při rozepnutém kontaktu vykonat jako např. uzavření ventilu či odstavení celé technologie.

Upozornění: pokud dojde k výpadku proudu, je to vyhodnoceno stejně jako otevření kontaktu.

Sestavení datagramu pro vyčtení polohy externího kontaktu připojeného na pohonu s adrese 3, který je ve stavu otevřeno

Na pohon posílám pouze příkaz bez parametrů, kterým si žádám zaslání informace o poloze externího kontaktu pověšeného na pohonu.

Generace MP příkazového paketu:

Start byte: 0b00011000 = 0h18

pro adresu 3 a jeden užitečný byte

Kód příkazu: 0b01001011 = 0h4B

příkaz číslo 75

Příčná parita: 0b00000000 = 0h00

Podélná parita: 0b01010011 = 0h53

MP-Bus telegram:

Master tedy posílá hexadecimální hodnoty: 18 4B 00 53

Slave pak vrací hexadecimální hodnoty: 18 4B 00 53 2D 00 01 80AC

Rozbor MP odpovědního paketu:

Start byte: 0h2D = 0b00101101

budou doručeny 2 užitečné byty

[1] 0h00 = 0b00000000

čtení stavu kontaktu aktivní

[2] 0h01 = 0b00000001

obvod je otevřený (kontakt je rozepnut)

Příčná parita: 0h80 = 0b10000000

Podélná parita: 0hAC = 0b10101100

Na žádost o vyčtení nastavení nadřazeného řízení je nám v prvním bytu odpovězeno, že jde o vyčítání stavu externího kontaktu a druhým bytem obdržíme informaci, že je otevřen.

9.6.7 Popis příkazu pro vyčtení poruchových stavů (MP_Get_Malfunction_Maintenance_State)

Tab. 34. Rozbor příkazu pro vyčtení poruchových stavů

Kód a název příkazu	Parametry	
	Posílané	Obdržené
26: MP_Get_Malfunction_Maintenance_State	Žádné	[1] Selhání / Servis

Servopohon může detekovat několik různých poruchových stavů, hlásících selhání pohonu či potřebu servisního zásahu.

Tyto chyby se dají vyčíst pomocí příkazu MP_Get_Malfunction_Maintenance_State, kterým vyčtu jeden byte, jehož jednotlivé bity jsou příznaky jednotlivých poruch. Pokud se mi některý byt nastaví na hodnotu 1, ohlašuje mi tím poruchu. Dle její závažnosti se pak provádí zvolená akce, jako je odstavení pohonu, jeho adaptace apod..

Tab. 35. Význam signalizovaných poruchových stavů

Bit	Příčina chyby
0=1	Nadměrné používání
1=1	Překročen řídící rozsah
2=1	Přetížení pohonu, nastavený bod nebyl dosažen
3=1	Aktuální kroutící momente > povolená zátěž (jen u pohonů bez havarijní fce)
4=1	Selhání související s bezpečností (pouze u požárních a kouřových pohonů)
5=1	Chyba při testu komínových klapek (pouze u požárních a kouřových pohonů)
6=1	Příliš vysoká teplota v potrubí (pouze u požárních a kouřových pohonů)
7=1	Poplach z kouřového detektoru (pouze u požárních a kouřových pohonů)

Sestavení datagramu pro vyčtení poruchových stavů z pohonu na adrese 3

Na pohon posílám pouze příkaz bez parametrů a jako odpověď mi přijde byte, ve kterém jsou signalizovány případné poruchové stavy.

Generace MP příkazového paketu:

Start byte: 0b00011000 = 0h18

pro adresu 3 a jeden užitečný byte

Kód příkazu: 0b01001011 = 0h1A

příkaz číslo 26

Příčná parita: 0b10000000 = 0h80

Podélná parita: 0b10000010 = 0h82

MP-Bus telegram:

Master tedy posílá hexadecimální hodnoty: 18 1A 80 82

Slave pak vrací hexadecimální hodnoty: 18 1A 80 82 1D 00 00 1D

Rozbor MP odpovědního paketu:

Start byte: 0h1D = 0b00011101

bude doručen 1 užitečný byte

[1] 0h00 = 0b00000000

není signalizována žádná porucha

Příčná parita: 0h00 = 0b00000000

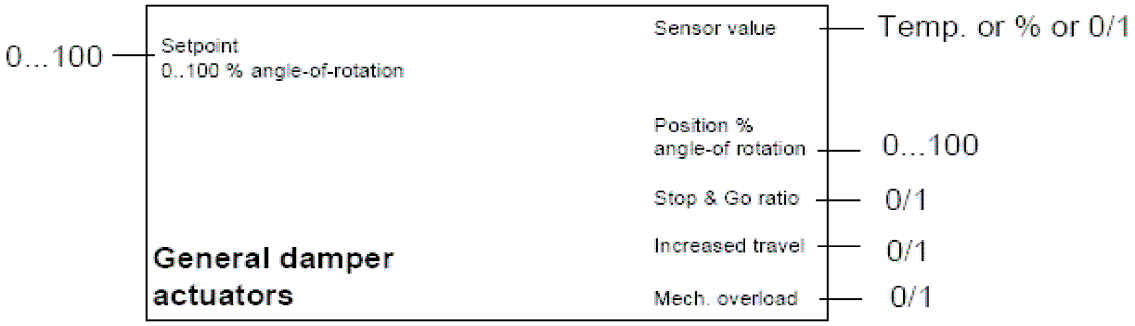
Podélná parita: 0h1D = 0b00011101

Z parametru odpovědi jsme se dozvěděli, že na pohonu není signalizován žádný poruchový stav.

9.7 Samotná uživatelská komunikace pro obsluhu servopohonu

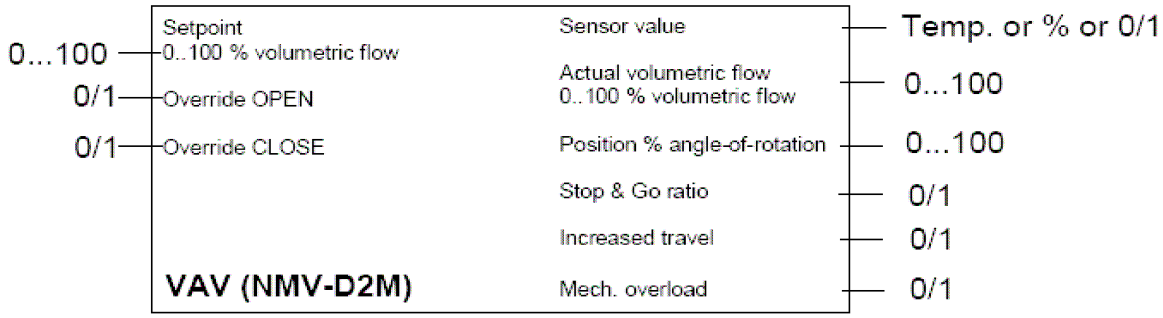
Tato uživatelská komunikace je podstatně složitější, než byla komunikace pro nastavení MP adresy. Pokusil jsem se v ní totiž postihnout všechny důležité funkce, které by mohl uživatel potřebovat při obsluze pohonů. Proto je zde také použito podstatně více příkazů. V tomto programu se navíc také kontroluje mnohem více věcí, kdy se např. hlídají timeouty, jelikož celý tento program běží na rozdíl od předešlého programu periodicky.

Při návrhu této komunikace jsem vycházel zejména z doporučení pro vytvoření funkčních bloků, vhodných k ovládání různých typů servopohonů, jenž jsem nalezl v dokumentu [13].



Obr. 33. Funkční blok pro klapkové pohony

Obr. 34. Funkční blok pro ventilové pohony



Obr. 35. Funkční blok pro VAV jednotky

Využití

Tato uživatelská komunikace je připravena na obsluhu nejdůležitějších funkcí u ventilových a klapkových pohonů. Pokud bychom chtěli touto uživatelskou komunikací ovládat i VAV jednotky, museli bychom provést pouze drobné úpravy programu, kdybychom příkaz pro vyčtení hodnoty min a max a příkaz pro vyčtení polohy nahradili jejich ekvivalentem pro VAV jednotky.

Navíc by bylo vhodné doplnit ovládání VAV jednotky dle uvedeného doporučení (viz *Obr. 35*) o funkci pro nadřazené řízení Otevřeno/Zavřeno, které se nastavuje příkazem MP_Set_Override_Control, do jehož parametru „ZF“je třeba zadat hodnoty z následující tabulky.

Tab. 36. Hodnota parametru „ZF“ při řízení otevřeno/zavřeno u VAV jednotek

Byte [ZF]	**pozice**
3	jde na Vmax
4	jde na Vmin

Popis funkcí programu (více v příloze P II)

V procesu Proc00 jsou zadávány všechny parametry pro příkazy, které chceme vykonávat, spolu s adresami pohonů, které mají tyto příkazy provádět. Vlastní příkazy se potom vykonávají v podprogramech, které jsou volány pomocí funkcí Call. V nich se provádí sestavování celých datagramů, jejich posílání na pohon a vyhodnocení přijatých odpovědních paketů. Proces Proc00 jsem připravil pro dva servopohony s MP adresou 2 a 3.

Pro servopohon s adresou tři (1. zařízení) jsem připravil následující příkazy:

- Odeslání zadané polohy – dva byty parametrů s žádanou polohou zadávám do matice „ZPoloha1“.
- Vyčtení aktuální polohy – vyčtená aktuální poloha je v matici „BResult“ na řádku tři v nultém sloupci. Tato poloha je zároveň přepočítávána pro zadané meze min a max, do proměnné „Prepocteny“.

- Vyčtení mezí – vyčtení hodnot omezujících pracovní rozsah na zadané meze min a max, které využívám při přepočtu aktuální polohy. Tyto vyčtené meze jsou uloženy v matici „BResult“ na řádku tři ve sloupcích dva a tři.
- Vyčtení hodnoty čidla – zde je ještě potřeba zadat o jaký typ čidla se jedná do matice „BParam“. Výsledek je pak uložen v matici „BFResult“ na řádku tři v nultém sloupci. V tomto příkladu se jedná o vyčítání hodnoty pasivního čidla Ni1000 a výsledná hodnot je přepočtena přímo na stupně celsia.
- Čtení stavu zařízení – slouží k vyčtení poruchových hlášení pohonu, které jsou uloženy v matici „BResult“ na třetím řádku ve čtvrtém sloupci. Tento příkaz si pak uživatel zpracuje dle vlastní potřeby a k jednotlivým bitům tohoto bytu si přiřadí operace prováděné při nastavení příznaku chyby na 1.

Pro servopohon s adresou dvě (2. zařízení) jsem připravil následující příkazy:

- Odeslání zadané polohy – dva byty parametrů s žádanou polohou zadávám do matice „ZPoloha2“.
- Vyčtení aktuální polohy – vyčtená aktuální poloha je v matici „BResult“ na řádku tři v nultém sloupci. Pohon nemá zadány žádné meze min a max.
- Nastavení parametru pro čtení polohy externího kontaktu – po zadání tohoto příkazu lze vstupem servopohonu Y vyčítat spínací kontakt.
- Vyčtení polohy spínacího kontaktu – vyčtená hodnota je uložen v matici „BResult“ na řádku dvě v pátém sloupci. Pokud je hodnota tohoto bytu 1 nebo 2, tak je kontakt otevřený.

Celý program spolu i s dalším popisem je umístěn v příloze P II.

ZÁVĚR

V této práci jsem se čtenáře pokusil blíže seznámit s klapkovými a ventilovými servopohony firmy Belimo, které jsou světově nejrozšířenějším typem pohonů pro oblast technického zabezpečení budov a tepelného hospodářství. Zabýval jsem se zde nově vyráběnými pohony, jenž umožňují sběrnicový provoz pomocí komunikačního protokolu MP-Bus společnosti Belimo, který je v této práci podrobně popsán, včetně popisu jeho nejdůležitějších příkazů.

Pomocí příkazů tohoto komunikačního protokolu pak lze tyto pohony nastavovat pro danou aplikaci, vyčítat z nich údaje o jejich stavu a nastavení, ovládat je, získat hodnotu z jejich externího senzoru, apod.

Účelem celé práce pak bylo „naučit“ mezi sebou komunikovat tyto servopohony a řídicí systémy firmy AMiT, právě pomocí komunikačního protokolu MP-Bus, kde řídicí systém vystupuje jako master ovládající až osm servopohonů, které se chovají jako slave.

Tato komunikace je realizována pomocí tzv. uživatelské komunikace, která byla vytvořena v programovacím prostředí PSP3 určeném k programování řídicích systémů firmy AMiT. Uživatelské komunikace vznikly dvě. První z nich slouží pro zadání MP adresy a je určená k adresaci všech zařízení podporujících MP-Bus komunikaci. Druhá je pak určena pro samotné ovládání servopohonů zadáním požadované polohy a umožňuje také vyčítání polohy a hodnoty z externího čidla či spínače připojeného na pohonu. Signalizuje nám také poruchová hlášení, jako je nadměrné používání pohonu, přetížení pohonu či překročení pracovního rozsahu. Tato komunikace je připravena pro obsluhu klapkových a ventilových servopohonů.

Tyto uživatelské komunikace odpovídají doporučením pro tvorbu funkčních bloků určených k ovládání servopohonů z řídicích systémů či regulátorů pomocí sběrnice MP-Bus, které jsou uvedeny v technické dokumentaci poskytnuté firmou Belimo.

Dalším logickým krokem, který by měl navazovat na tuto práci, by mělo být právě vytvoření takovýchto funkčních bloků pro programovací prostředí PSP3 a to v některém z programovacích jazyků jako je např. C++. Tyto funkční bloky by měly stejně jako mnou navrhnuté uživatelské komunikace umožnit adresaci a ovládání pohonů, včetně vyčtení hodnoty externího čidla či spínače, polohy a poruchových hlášení.

Na závěr bych také rád zmínil, že vytvořené uživatelské komunikace nachází již v současnosti praktické uplatnění u zákazníků společnosti AMiT. Již v průběhu vzniku této práce totiž byly poskytnuty několika zákazníkům, kteří je využili k ovládání servopohonů firmy Belimo prostřednictvím protokolu MP-Bus.

SEZNAM POUŽITÉ LITERATURY

[1] www.belimo.ch

[2] www.amit.cz

[3] MFT2: Product information

[4] Belimo Bus Solutions

[5] MFT2: Informace o výrobku

[6] MP / PP Specification V1.22b

[7] BELIMO MFT Actuators - Developers documentation

[8] Operating instructions MFT-H

[9] PC-Tool V3.1

[10] Interface module for cubicle mounting ZIP-232-MP

[11] MP-Bus Power Unit ZN-MP

[12] Basic Technology of MP-Bus Systems

[13] Additional Information on the MP-Bus System

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

%	Procento
°C	Jednotka teploty
A	Jednotka proudu
AC	Střídavé napájení
Bd	Jednotka přenosové rychlosti - Baud
C	Kapacitní odpor
Cc	Kód příkazu
Cp	Příčná parita
C_{u5}	Kapacitní odpor mezi U5 a GND
DC	Stejnosměrné napájení
DDC	Direct digital controler (digitální regulátor)
GND	Zem napájení
I_{hmax}	Zkratovací proud mezi U5 a GND
Lp	Podélná parita
LSB	Nejméně významný bit
m	Jednotka délky
MFT	Multi-funkční-technologie
mm^2	Jednotka plochy
MSB	Nejvíce významný bit
R_{is}	Elektrický odpor v MP-módu bez komunikace
Rx	Svorka pro příjem signálu / signalizace příjmu
s	Jednotka času
Stb	Start byte
t_{admin}	Zpoždění odpovědi

t_{amax}	Master timeout / opakovací interval příkazu
t_{bit}	Čas potřebný pro přenos bitu
t_{cdmin}	Zpoždění příkazu
t_f	Čas poklesu
t_{Fr}	Čas potřebný pro přenos rámce
t_{gap}	Čas mezi dvěma byty
t_r	Budící čas
Tx	Svorka pro vysílání signálu / signalizace vysílání
Us	Signalizace napájení u převodníku ZIP-22-MP
V	Jednotka napětí
VAV	Jednotky pro řízení průtoku vzduchu
V_h	Horní úroveň napětí pro příkaz a odpověď
V_{hystm}	Hysterze prahového napětí masteru
V_{hysts}	Hysterze prahového napětí na slavu
V_{la}	Dolní úroveň napětí pro odpověď
V_{lc}	Dolní úroveň napětí pro příkaz
V_{ms}	Napájení mastera
V_{thmh}	Horní práh napětí signálu mastera
V_{thml}	Dolní práh napětí signálu mastera
V_{thsh}	Horní práh napětí sleva
V_{thsl}	Dolní práh napětí sleva
Ω	Jednotka odporu

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: UŽIVATELSKÁ KOMUNIKCE PRO NASTAVENÍ MP ADRESY

```
ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿
³  Copyright (c) 2006, AMiT, spol. s r.o., Aplikace mikroprocesorove techniky³
ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ´
³                                                                              ³
³  Projekt : MP-Bus                                                            ³
³  Verze   : V01.00                                                            ³
³  Stanice : PS04 - Uzivatelska komunikace pro nastaveni MP adresy             ³
³  Verze   : V01.00                                                            ³
³  Id1     :                                                                   ³
³  Id2     :                                                                   ³
³                                                                              ³
³  Soubor  : COM.TXT                                                           ³
³  Autor   : Frantisek Grebenicek                                              ³
³                                                                              ³
³  Datum   :  1. 9.'06    (generovano editorem procesni stanice PSE)           ³
³                                                                              ³
ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ´
³                                                                              ³
³  Volby   : Pasivni, bez mapovani WIDu                                        ³
³  Hardware: AMINI                                                             ³
³                                                                              ³
ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ



ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Poznamky k procesni stanici
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ


Postup pri adresovani zarizeni:

1) V procesu INIT v sekci zvane "Zadavani parametru" zadavame
postupne vsechny ctyri casti serioveho cisla adresovaneho zarizeni,
ktere je uvedene na jeho stitku. Na nasledujicim radku zadame
adresu (1-8), kterou chceme tomuto zarizeni priradit.

2) Program Ulozime, Prelozime, Nahrajeme.

3) Po cca 20s je adresovani hotovo.

4) Pokud je promenna OK = 1, adresovani probehlo v poradku.

5) Pro dalsi pohony opakujeme postup od bodu 1 do bodu 4.

ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Vypis konfigurace procesnich vstupu a vystupu
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
```

```
 Typ  Log.kanal  Komentar
 ÄÄÄ  ÄÄÄÄÄÄÄÄÄÄ  ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
 DIO      0
 DIO-AC   1
 DAIO     2
 DAIO-AC  3
 DOO      0
 LED      1
 AIO      0
 Ni1000   1

ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Vypis databazovych promennych
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ

Por. Jmeno         WID    Typ a rozmer  Druh  St. Komentar
ÄÄÄ  ÄÄÄÄÄÄÄÄÄÄÄÄÄ  ÄÄÄÄÄ  ÄÄÄÄÄÄÄÄÄÄÄÄÄÄ  ÄÄÄ  ÄÄ  ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
  1  Adresa        04041  I              i..   4
  2  BChyba        04037  I                    4
  3  BIndex_In     04018  I                    4  index
  4  BIndex_Out    04020  I                    4  index
  5  BInTlg        04029  MI[   1, 256]        4
  6  Bixxb         04023  I                    4
  7  Bixxc         04026  I                    4
  8  BNacteno      04019  I                    4  pocet
  9  BOutLen       04027  I                    4
 10  BOutTlg1      04034  MI[   1, 256]        4
 11  BOutTlg2      04031  MI[   1, 256]  i..   4
 12  BOutTlg3      04033  MI[   1, 256]  i..   4
 13  BParCross     04024  I                    4
 14  BParLength    04028  I                    4
 15  BPomocna      04036  I                    4
 16  BPraporek     04032  I                    4
 17  BPrep         04040  MI[   9,   1]  i..   4
 18  BPrijBuf      04000  MI[   1, 256]        4  Prijimaci bufer
 19  BVyslBuf      04001  MI[   1, 256]        4  Vysilaci bufer
 20  BZapisovat    04030  I                    4
 21  OK            04042  I                    4
 22  SC            04038  MI[   1,   4]  i..   4

ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Vypis alias-promennych
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ

Por. Alias         Jmeno            Komentar
ÄÄÄ  ÄÄÄÄÄÄÄÄÄÄÄÄÄ  ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ  ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Vypis tabulky procesu
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ

Proces    Typ    Perioda Ofset/Hrana Modulu Komentar
ÄÄÄÄÄÄÄÄÄ  ÄÄ   ÄÄÄÄÄÄÄÄ   ÄÄÄÄÄÄÄÄ  ÄÄÄÄÄÄ ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
Proc00             2.0 s      4.0 s    16   Vysilani prikazu
```

```
Proc01            10.0 s     10.0 s    13   Prijem a kontrola spravnosti
ProcINIT                               20   Hlavni modul komunikace
Lib101                                 15   Vypocet parity

ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Proc00             2.0 s      4.0 s    16   Vysilani prikazu
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ

  1         If         BPraporek.1, :NONE
  2           Let      BOutLen = 2
  3           MtxCopy  BOutTlg1, NONE, NONE, BOutTlg3, NONE, NONE, NONE, NONE, NONE.0
  4           Call     101
  5           Let      BZapisovat = BOutLen
  6           ComWrite  :01000, BOutTlg1, 0, BOutLen, BIndex_Out, BZapisovat
  7           Let      BPraporek = 0
  8         EndIf
  9         If         BPraporek.0, :NONE
 10           Let      BOutLen = 8
 11           MtxCopy  BOutTlg1, NONE, NONE, BOutTlg2, NONE, NONE, NONE, NONE, NONE.0
 12           Call     101
 13           Let      BZapisovat = BOutLen
 14           ComWrite  :01000, BOutTlg1, 0, BOutLen, BIndex_Out, BZapisovat
 15           Let      BPraporek = 2
 16         EndIf

ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Proc01            10.0 s     10.0 s    13   Prijem a kontrola spravnosti
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ

  1         ComRead    :01000, BInTlg, 0, BIndex_In, BNacteno
  2         REM        "Chyba, kdy nejvyssi bit 1. byte odpovedi je 1"
  3            Let        BPomocna = BInTlg[0,14]
  4         If         BPomocna.7, :NONE
  5         Let      BChyba = BInTlg[0,15]
  6         EndIf
  7         REM        "Chyba, kdy nizsi pulbyte 1. bytu odpovedi je ruzny od 0xD"
  8            Let          BPomocna.0 = Not((BInTlg[0,14].0) And Not(BInTlg[0,14].1) And
(BInTlg[0,14].2) And (BInTlg[0,14].3))
  9         If         BPomocna.0, :NONE
 10         Let      BChyba = 255
 11         EndIf
 12         REM        "Kontrola, jestli adresace probehla v poradku"
 13       Let        OK.0 = (BInTlg[15,0]==BOutTlg2[2,0]) And (BInTlg[16,0]==BOutTlg2[3,0])
And      (BInTlg[17,0]==BOutTlg2[4,0])      And      (BInTlg[18,0]==BOutTlg2[5,0])      And
(BInTlg[21,0]==BOutTlg2[6,0])

ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
ProcINIT                               20   Hlavni modul komunikace
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ

  1 :01000  ComInit    0x0001, 0, 1200, 8, Zadna, 1, :NONE, :NONE, :NONE, :NONE, BPrijBuf,
BVyslBuf
  2         Let        BPraporek = 1
  3         REM        "======================================================"
  4         REM        "===== Zadavani parametru ====="
```

```
          REM        "=== Seriove cislo zarizeni ==="
          Let        SC[0,0] = 418
          Let        SC[0,1] = 10040
          Let        SC[0,2] = 128
          Let        SC[0,3] = 52
          REM        "=== Adresa zarizeni ==="
          Let        Adresa =  2
          REM        "========================================================"
          REM        "Dekodovani zadaneho serioveho cisla"
          StrFormat  BOutTlg2, 2, NONE, "", BinIntBE, -1, 2, 0, SC[0,0], NONE.0
          StrFormat  BOutTlg2, 4, NONE, "", BinIntBE, -1, 2, 0, SC[0,1], NONE.0
          Let        BOutTlg2[0,6] = SC[0,3]
          REM        "Prepocet adresy"
          Let        BOutTlg2[0,7] = BPrep[Adresa,0]
          Let        BOutTlg3[0,0] = BOutTlg2[0,7] + 0x10
          REM        "======================================================"
```

íí
Lib101   15   Vypocet parity
íí

```
          Let        BParCross=0
          REM        "Cross parity"
          For        Bixxb, 0.0, 7.0, 1.0, :NONE
            Let      BParCross.Bixxb=0
            For      Bixxc, 0.0, 7.0, 1.0, :NONE
              Let    BParCross.Bixxb=BParCross.Bixxb XOR BOutTlg1[0,Bixxb].Bixxc
            EndFor
          EndFor
          Let        BParCross=BParCross << (8-BOutLen)
          Let        BOutTlg1[0,BOutLen]=BParCross
          Let        BOutLen=BOutLen+1
          REM        "Lenght Parity"
          Let        BParLength=BOutLen
          ChkCreate  BOutTlg1, 0, BParLength, BOutLen, XOR-8, BOutLen
          Nop
```

PŘÍLOHA P II: UŽIVATELSKÁ KOMUNIKCE PRO OVLÁDÁNÍ POHONU

```
ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ł  Copyright (c) 2006, AMiT, spol. s r.o., Aplikace mikroprocesorov, technikył
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ´
ł                                                                            ł
ł  Projekt : MP-Bus                                                          ł
ł  Verze   : V01.00                                                          ł
ł  Stanice : PS04 - Uzivatelska komunikace pro obsluhu servopohonu           ł
ł  Verze   : V01.00                                                          ł
ł  Id1     :                                                                 ł
ł  Id2     :                                                                 ł
ł                                                                            ł
ł  Soubor  : COMBELIM.TXT                                                    ł
ł  Autor   : Frantisek Grebenicek                                            ł
ł                                                                            ł
ł  Datum   :  1. 9.'06    (generovano editorem procesni stanice PSE)         ł
ł                                                                            ł
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ´
ł                                                                            ł
ł  Volby   : Pasivni, bez mapovani WIDu                                     ł
ł  Hardware: AMiNi                                                           ł
ł                                                                            ł
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ


ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Poznamky k procesni stanici
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ

Uzivatelska komunikace BELIMO - priklad
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Popis protokolu:
----------------
- RS232, 1200Bd, suda parita, 1 stopbit
- komunikace master - slave s potvrzovanim

Popis demonstracniho programu:
----------------
Vlastni demonstracni program je ukryt v periodickem procesu Proc00.
Program ukazuje, jakym zpusobem je vhodne komunikovat s vice zarizenimi
Belimo. Pri kazdem behu procesu je inkrementovana promenna "Rozstrel", ktera
rika, jakou akci pri tom kterem behu procesu budeme provadet. Hodnoty ziskane
ze zarizeni neni nutno po ukonceni behu podprogramu bezprostredne zapisovat
do jinych promennych, zustavaji az do dalsiho prepsani novejsimi hodnotami
v maticich vysledku BResult a BFResult.

Obsluha:
----------------
```

Zadana poloha 1. ventilu se zadava do promenne, ZPoloha1‘, zadana poloha 2. ventilu se zadava do promenne, ZPoloha1.
Pro kazdou jednotlivou komunikaci je naprogramovan vlastni podprogram. Tyto podprogramy zacinaji Lib500. V podprogramech od Lib590 do Lib599 jsou ukryty casti kodu, ktere pouzivaji podprogramy od Lib500.

Pokud je treba poslat podprogramu parametr(y), ulozi se tyto do matice BParam. Parametry je nutno ukladat do matice bezprostredne pred vyvolanim konkretniho podprogramu. Pokud je volano stridave vice podprogramu, vzdy musi byt parametry zadany znovu. To je dano tim, ze vsechny podprogramy pouzivaji stejnou matici parametru.
Pokud ma podprogram vracet vysledky, jsou tyto ulozeny v matici BResult pro integer promenne a BFResult pro float promenne. V techto maticich kazdy radek odpovida jednomu zarizeni, podle toho, jakou ma adresu .
Do promenne BAddr se VZDY pred konkretni komunikaci musi zadat fyzicka adresa zarizeni, s nimz bude komunikovano. Pokud se komunikuje pouze s jednim zarizenim, je mozne zadat adresu BAddr pouze jednou pri startu programu.

Lib500 - Zapis pozadovane polohy

Vstupy : BAddr, BParam[0,0]-Zadana poloha v rozmezi 0-10000
Vystupy: -

Lib501 - Cteni aktualni polohy

Vstupy : BAddr
Vystupy: BResult[BAddr,0]-Aktualni poloha, BResult2[BAddr,1]-Zadana poloha
---
Lib502 - Cteni teploty

Vstupy : BAddr, BParam[0,0]-Typ cidla (pro Ni1000 = 0x10)
Vystupy: BFResult[BAddr,0]-Aktualni teplota cidla ve stupnich celsia
---
Lib503 - Cteni minima, maxima

Vstupy : -
Vystupy: BResult[BAddr,2]-Minimum, BResult[BAddr,3]-Maximum

Lib504 - Cteni stavu zarizeni

Vstupy : -
Vystupy: BResult[BAddr,4]-Stav zarizeni

Lib505 a Lib506

Volanim Lib505 se nastavi funkce cteni pripojeneho externiho kontaktu.

Volanim Lib506 se vycte stav tohoto kontaktu.
Vstupy: -
Vystup: BResult[BAddr,5]-Stav kontaktu

Vysvetleni vyznamu promennych:

BCntErrorD - Pocitadlo komunikaci, kdy se ve startbytu odpovedi
vyskytne neco jineho nez 0b0___1101 - tedy chyba.
BCntErrorNEQ - Pocitadlo komunikaci, kdy nesedi predpokladana a skutecna
delka ramce
BCntErrorTMO - Pocitadlo komunikaci, kdy vyprsi TimeOut

Vypis konfigurace procesnich vstupu a vystupu

Typ	Log.kanal	Komentar
AIO	0	

Vypis databazovych promennych

Por.	WID	Jmeno	Typ a rozmer	Druh	St.	Komentar
1	04003	ZPoloha1	I		4	Zadana poloha 1. zarizeni
2	04004	ZPoloha2	I		4	Zadana poloha 2. zarizeni
3	04800	BAddr	I		4	Adresa modulu s nimz se komunikuje
4	04801	BAnswrLength	I		4	Pozadovana delka odpovedi
5	04802	BCntErrorD	L		4	Pocitadlo chyb v komunikaci
6	04803	BCntErrorNEQ	L		4	Pocitadlo chyb v komunikaci
7	04804	BCntErrorTMO	L		4	Pocitadlo chyb v komunikaci
8	04805	Belimo	I		4	Stavova promenna
9	04806	BErrType	I		4	kod chyby
10	04809	Bfxx	F		4	Pomocna promenna typu float
11	04810	BInLen	I		4	Delka prijimaneho telegramu
12	04811	BInTlg	MI[1, 20]		4	Prijimany telegram
13	04812	Bixxa	I		4	Pomocna promenna typu Int
14	04813	Bixxb	I		4	Pomocna promenna pro vypocet parity
15	04814	Bixxc	I		4	Pomocna promenna pro vypocet parity
16	04815	Bixxd	I		4	Pomocna promenna typu Int
17	04816	BMessLength	I		4	Pocet znaku nasledujicich za StartBytem
18	04817	BOutLen	I		4	Delka vysilaneho telegramu
19	04818	BOutLenBuf	I		4	Delka ulozeneho vysilaneho telegramu
20	04819	BOutTlg	MI[1, 20]		4	Vysilany telegram
21	04820	BOutTlgBuf	MI[1, 10]		4	Buffer pro ulozeni vyslanych dat
22	04821	BParam	MI[1, 3]		4	Parametry funkce
23	04823	BParCross	I		4	Promenna pro vypocet krizove parity
24	04824	BParLength	I		4	Pocitadlo pro ladeni
25	04825	BPrijBuf	MI[1, 256]		4	Prijimaci bufer
26	04826	BReqNdx	I		4	Cislo provadeneho pozadavku
27	04827	BReqResult	I		4	Vysledek pozadavku
28	04828	BReqStatus	I	i..	4	Stav pozadavku
29	04836	BT_Operace	I		4	Typ komunikace
30	04840	BVyslBuf	MI[1, 256]		4	Vysilaci bufer

```
 31  04841  B_L2_Event    I                    4  Udalost do automatu L2
 32  04842  B_L2_Status   I               i..  4  Stav automatu L2
 33  04843  B_R2T         MF[  15,   2]  i..  4  Pro prepocet odporu na teplotu, Ni1000
 34  04847  Rozstrel      I                    4  Pro cyklicke zasilani ruznych dotazu
 35  04850  BResult       MI[  10,   6]        4  Vracene hodnoty typu INT
 36  04851  BFResult      MF[  10,   3]        4  Vracene hodnoty typu Float
 37  04853  Prepocteny    F                    4  Akt. poloha prepoctena na meze

ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Vypis alias-promennych
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ

Por. Alias         Jmeno            Komentar
ÄÄÄ ÄÄÄÄÄÄÄÄÄÄÄÄÄ ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
  1  @Ano          Belimo.0         Testovaci bit
  2  @Chyba1       Belimo.1         Testovaci bit
  3  @Chyba2       Belimo.2         Testovaci bit
  4  @ComFree      Belimo.7         Priznak volne linky
  5  @PlatnaData   Belimo.6         Priznak pripravenosti odpovedi
  6  @Test1        Belimo.3         Testovaci bit
  7  @Test2        Belimo.4         Testovaci bit

ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Vypis tabulky procesu
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ

Proces    Typ    Perioda Ofset/Hrana Modulu Komentar
ÄÄÄÄÄÄÄÄÄ ÄÄ   ÄÄÄÄÄÄÄÄ  ÄÄÄÄÄÄÄÄ ÄÄÄÄÄÄ ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
Proc00            0.2 s      3.0 s    60   Komunikace s Belimy
ProcINIT                              10   Inicializace komunikace
Lib500                                29   Belimo-Zapis pozadovane polohy
Lib501                                24   Belimo-Cteni aktualni polohy
Lib502                                22   Belimo-Cteni teploty
Lib503                                24   Belimo-Cteni Minima a Maxima
Lib504                                24   Belimo-Vycitani stavu zarizeni
Lib505                                25   Belimo-Nastaveni cteni ext. kontaktu
Lib506                                24   Belimo-Cteni stavu externiho kontaktu
Lib590                                73   _Belimo-Stavovy automat
Lib592                                 2   _Belimo-Preruseni od prijmu
Lib593                                 2   _Belimo-Preruseni od timeoutu
Lib594                                44   _Belimo-Konec prijmu,dekodovani odpoved
Lib597                                13   _Belimo-Detekce a zaznam chyb
Lib599                                15   _Belimo-Vypocet CRC

ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Proc00            0.2 s      3.0 s    60   Komunikace s Belimy
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ

  1         Switch     Rozstrel, :NONE
  2           Case     0, :NONE
  3             REM    "Zadana poloha pro 1. zarizeni"
  4             Let    BAddr = 3                                  Zarizeni ma adresu 3
  5             Let    BParam[0,0] = ZPoloha1                     Vlozeni zadane polohy do
vhodne promenn
```

```
 6            Call   500
 7          EndCase
 8          Case     1, :NONE
 9            REM    "Cteni aktualni polohy 1. zarizeni"
10            Let    BAddr = 3
11            Call   501
12            REM    "Vysledek je ulozen v matici BResult na ra
                     dku 3 ve sloupci 0"
13            REM    "Prepocet na pripadne jine meze"
14            Let    Prepocteny = 1000*(1000000*Float(BResult[B
                     Addr,0])-100*BResult[BAddr,2]*BResult[BAdd
                     r,3])/(100000*BResult[BAddr,3]-BResult[BAd
                     dr,2]*BResult[BAddr,3])
15          EndCase
16          Case     2, :NONE
17            REM    "Vycteni hodnoty cidla na 1.zarizeni"
18            Let    BAddr = 3
19            REM    "Urceni typu cidla, zde Ni1000"
20            Let    BParam[0,0] = 0x10
21            Call   502
22            REM    "Vysledek je ulozen v matici BFResult na r
                     adku 3 ve sloupci 0"
23          EndCase
24          Case     3, :NONE
25            REM    "Zadani polohy pro 2. zarizeni"
26            Let    BAddr = 2
27            Let    BParam[0,0] = ZPoloha2
28            Call   500
29          EndCase
30          Case     4, :NONE
31            REM    "Cteni aktualni polohy 2. zarizeni"
32            Let    BAddr = 2
33            Call   501
34            REM    "Vysledek je ulozen v matici BResult na ra
                     dku 2 ve sloupci 0"
35          EndCase
36          Case     5, :NONE
37            REM    "Cteni mezi 3. zarizeni"
38            Let    BAddr = 3
39            Call   503
40            REM    "Vysledek je ulozen v matici BResult na ra
                     dku 3 ve sloupcich 2(min) a 3(max)"
41          EndCase
42          Case     6, :NONE
43            REM    "Cteni stavu 3. zarizeni"
44            Let    BAddr = 3
45            Call   504
46            REM    "Vysledek je ulozen v matici BResult na ra
                     dku 3 ve sloupci 4"
47          EndCase
48          Case     7, :NONE
49            REM    "Cteni stavu externiho kontaktu 2. zarizen
                     i"
50            Let    BAddr = 2
```

```
 51            Call   506
 52            REM    "V˜sledek je ulo‘en v matici BResult na ©
                      dku 2 ve sloupci 5"
 53          EndCase
 54          Case     8, :NONE
 55            REM    "Nastaveni parametru pro cteni externiho k
                      ontaktu 2.zarizenim"
 56            Let    BAddr = 2
 57            Call   505
 58          EndCase
 59        EndSwitch
 60        Let        Rozstrel = If(Rozstrel>=7,0,Rozstrel+1)

íííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííí
ProcINIT                               10   Inicializace komunikace
íííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííí

  1 :50001  ComInit    0x0000, 0, 1200, 8, ’ dn , 1, :50010,
                       :NONE, :NONE, :NONE, BPrijBuf, BVyslBuf     Komunikacni jadro
  2 :50002  Tmo        :50011, 10, 60                              Timeout
  3 :50010  SubInst    592                                          Instance podprog. preru-
seni od prijmu
  4 :50011  SubInst    593                                          Instance podprog. preru-
seni od timeoutu
  5         Let        B_L2_Status = 0x0001                        Pocatecni stav automatu
  6         Let        BReqStatus = 0x0001                         Pocatecni stav pozadavku
  7         Let        @ComFree = 1
  8         Let        @PlatnaData = 0
  9         StopWatch  1, 0x0001, NONE
 10         Let        Rozstrel = 8

íííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííí
Lib500                                 29   Belimo-Zapis pozadovane polohy
íííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííí

  1         Let        BT_Operace = 1
  2         REM        "Zapis pozadovane polohy"
  3         REM        "Zadana poloha je prvnim a jedinym paramet
                       rem"
  4         REM        "Vycisteni matice pro odchozi telegram"
  5         For        Bixxa, 0.0, 20.0, 1.0, :NONE
  6           Let      BOutTlg[0,Bixxa]=0
  7         EndFor
  8         REM        "Sestaveni prvniho znaku telegramu"
  9         Let        BOutTlg[0, 0].0 = 0                         1. byte musi mit bit 0=0
 10         Let        BOutTlg[0, 0].1 = 0                         1. byte musi mit bit 1=0
 11         Let        BOutTlg[0, 0].2 = (BAddr-1).0               Nastaveni 1. bytu zpravy
...
 12         Let        BOutTlg[0, 0].3 = (BAddr-1).1               ... ve kterem se nachazi
...
 13         Let        BOutTlg[0, 0].7 = (BAddr-1).2               ... adresa
 14         Let        BReqNdx=1
 15         Let        @Ano = BReqStatus == 1
 16         REM        "Nastaveni delky zpravy a zadane delky odp
                       ovedi"
```

```
 17         REM        "... a zakodovani do telegramu"
 18         Let        BMessLength = 3                         Delka odesilane zpravy
 19         Let        BAnswrLength = 9                        Pozadovana delka odpove-
di
 20         Let        BOutTlg[0, 0].4 = BMessLength.0         Nastaveni 1. bytu zpravy
...
 21         Let        BOutTlg[0, 0].5 = BMessLength.1         ... ve kterem se nachazi
...
 22         Let        BOutTlg[0, 0].6 = BMessLength.2         ... take delka zpravy.
 23         Let        BOutTlg[0, 1] = 0x25                    Kod prikazu
 24         StrFormat  BOutTlg, 2, BOutLen, "", BinIntBE, -1, 2,
                       -1, BParam[0,0], NONE.0                     Prilepeni zakodovane
ZadaneHodnoty
 25         Call       599                                       Podprogram pro vypocet
CRC
 26         Let        B_L2_Event = 0x0002                     Generuj udalost Send
 27         If         @ComFree, :NONE
 28           Call     590
 29         EndIf

íííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííí
Lib501                              24   Belimo-Cteni aktualni polohy
íííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííí

  1         Let        BT_Operace = 2
  2         REM        "Cteni aktualni polohy"
  3         REM        "vystupy jsou Aktualni poloha a Pozadovana
                        poloha"
  4         For        Bixxa, 0.0, 20.0, 1.0, :NONE
  5           Let      BOutTlg[0,Bixxa]=0
  6         EndFor
  7         Let        BOutTlg[0, 0].0 = 0                     1. byte musi mit bit 0=0
  8         Let        BOutTlg[0, 0].1 = 0                     1. byte musi mit bit 1=0
  9         Let        BOutTlg[0, 0].2 = (BAddr-1).0           Nastaveni 1. bytu zpravy
...
 10         Let        BOutTlg[0, 0].3 = (BAddr-1).1           ... ve kterem se nachazi
...
 11         Let        BOutTlg[0, 0].7 = (BAddr-1).2           ... adresa
 12         Let        BReqNdx=1
 13         Let        BMessLength = 1                         Delka odesilane zpravy
 14         Let        BAnswrLength = 11                       Pozadovana delka odpove-
di
 15         Let        BOutTlg[0, 0].4 = BMessLength.0         Nastaveni 1. bytu zpravy
...
 16         Let        BOutTlg[0, 0].5 = BMessLength.1         ... ve kterem se nachazi
...
 17         Let        BOutTlg[0, 0].6 = BMessLength.2         ... take delka zpravy.
 18         Let        BOutTlg[0, 1] = 0x29                    Kod prikazu
 19         Let        BOutLen=2                               Aktualni delka zpravy
 20         Call       599                                       Podprogram pro vypocet
CRC
 21         Let        B_L2_Event = 0x0002                     Generuj udalost Send
 22         If         @ComFree, :NONE
 23           Call     590
 24         EndIf

íííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííí
Lib502                              22   Belimo-Cteni teploty
íííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííííí
```

```
  1         Let        BT_Operace = 3
  2         REM        "Cteni teploty cidla"
  3         REM        "Vstupem je typ cidla,vystupem je float ho
                       dnota teplota [stC]"
  4         For        Bixxa, 0.0, 20.0, 1.0, :NONE
  5           Let      BOutTlg[0,Bixxa]=0
  6         EndFor
  7         Let        BOutTlg[0, 0].0 = 0                       1. byte musi mit bit 0=0
  8         Let        BOutTlg[0, 0].1 = 0                       1. byte musi mit bit 1=0
  9         Let        BOutTlg[0, 0].2 = (BAddr-1).0             Nastaveni 1. bytu zpravy
...
 10         Let        BOutTlg[0, 0].3 = (BAddr-1).1             ... ve kterem se nachazi
...
 11         Let        BOutTlg[0, 0].7 = (BAddr-1).2             ... adresa
 12         Let        BReqNdx=1
 13         Let        BMessLength = 2                           Delka odesilane zpravy
 14         Let        BAnswrLength = 10                         Pozadovana delka odpove-
di
 15         Let        BOutTlg[0, 0].4 = BMessLength.0           Nastaveni 1. bytu zpravy
...
 16         Let        BOutTlg[0, 0].5 = BMessLength.1           ... ve kterem se nachazi
...
 17         Let        BOutTlg[0, 0].6 = BMessLength.2           ... take delka zpravy.
 18         Let        BOutTlg[0, 1] = 0x04                      Kod prikazu
 19         Let        BOutTlg[0, 2] = BParam[0,0]               Typ cidla
 20         Let        BOutLen=3                                 Aktualni delka zpravy
 21         Call       599                                         Podprogram pro vypocet
CRC
 22         Let        B_L2_Event = 0x0002                       Generuj udalost Send
```

```
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Lib503                                24   Belimo-Cteni Minima a Maxima
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
```

```
  1         Let        BT_Operace = 4
  2         REM        "Cteni minima a maxima"
  3         REM        "vystupy jsou Minimum a Maximum"
  4         For        Bixxa, 0.0, 20.0, 1.0, :NONE
  5           Let      BOutTlg[0,Bixxa]=0
  6         EndFor
  7         Let        BOutTlg[0, 0].0 = 0                       1. byte musi mit bit 0=0
  8         Let        BOutTlg[0, 0].1 = 0                       1. byte musi mit bit 1=0
  9         Let        BOutTlg[0, 0].2 = (BAddr-1).0             Nastaveni 1. bytu zpravy
...
 10         Let        BOutTlg[0, 0].3 = (BAddr-1).1             ... ve kterem se nachazi
...
 11         Let        BOutTlg[0, 0].7 = (BAddr-1).2             ... adresa
 12         Let        BReqNdx=1
 13         Let        BMessLength = 1                           Delka odesilane zpravy
 14         Let        BAnswrLength = 13                         Pozadovana delka odpove-
di
 15         Let        BOutTlg[0, 0].4 = BMessLength.0           Nastaveni 1. bytu zpravy
...
 16         Let        BOutTlg[0, 0].5 = BMessLength.1           ... ve kterem se nachazi
...
 17         Let        BOutTlg[0, 0].6 = BMessLength.2           ... take delka zpravy.
 18         Let        BOutTlg[0, 1] = 0x3B                      Kod prikazu
 19         Let        BOutLen=2                                 Aktualni delka zpravy
```

```
 20         Call       599                                    Podprogram pro vypocet
CRC
 21         Let        B_L2_Event = 0x0002                    Generuj udalost Send
 22         If         @ComFree, :NONE
 23           Call     590
 24         EndIf

ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Lib504                                24   Belimo-Vycitani stavu zarizeni
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ

  1         Let        BT_Operace = 5
  2         REM        "Cteni stavu zarizeni"
  3         REM        "Vystupem je byte, ktery popisuje stav zar
                       izeni"
  4         For        Bixxa, 0.0, 20.0, 1.0, :NONE
  5           Let      BOutTlg[0,Bixxa]=0
  6         EndFor
  7         Let        BOutTlg[0, 0].0 = 0                    1. byte musi mit bit 0=0
  8         Let        BOutTlg[0, 0].1 = 0                    1. byte musi mit bit 1=0
  9         Let        BOutTlg[0, 0].2 = (BAddr-1).0           Nastaveni 1. bytu zpravy
...
 10         Let        BOutTlg[0, 0].3 = (BAddr-1).1           ... ve kterem se nachazi
...
 11         Let        BOutTlg[0, 0].7 = (BAddr-1).2          ... adresa
 12         Let        BReqNdx=1
 13         Let        BMessLength = 1                        Delka odesilane zpravy
 14         Let        BAnswrLength = 9                        Pozadovana delka odpove-
di
 15         Let        BOutTlg[0, 0].4 = BMessLength.0         Nastaveni 1. bytu zpravy
...
 16         Let        BOutTlg[0, 0].5 = BMessLength.1         ... ve kterem se nachazi
...
 17         Let        BOutTlg[0, 0].6 = BMessLength.2        ... take delka zpravy.
 18         Let        BOutTlg[0, 1] = 0x1A                   Kod prikazu
 19         Let        BOutLen=2                              Aktualni delka zpravy
 20         Call       599                                      Podprogram pro vypocet
CRC
 21         Let        B_L2_Event = 0x0002                    Generuj udalost Send
 22         If         @ComFree, :NONE
 23           Call     590
 24         EndIf

ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Lib505                                25   Belimo-Nastaveni cteni ext. kontaktu
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ

  1         Let        BT_Operace = 6
  2         REM        "Nastaveni cteni externiho kontaktu"
  3         REM        "Vystupem je byte, ktery popisuje stav zar
                       izeni"
  4         For        Bixxa, 0.0, 20.0, 1.0, :NONE
  5           Let      BOutTlg[0,Bixxa]=0
  6         EndFor
  7         Let        BOutTlg[0, 0].0 = 0                    1. byte musi mit bit 0=0
  8         Let        BOutTlg[0, 0].1 = 0                    1. byte musi mit bit 1=0
```

```
  9         Let        BOutTlg[0, 0].2 = (BAddr-1).0             Nastaveni 1. bytu zpravy
...
 10         Let        BOutTlg[0, 0].3 = (BAddr-1).1             ... ve kterem se nachazi
...
 11         Let        BOutTlg[0, 0].7 = (BAddr-1).2             ... adresa
 12         Let        BReqNdx=1
 13         Let        BMessLength = 2                           Delka odesilane zpravy
 14         Let        BAnswrLength = 10                         Pozadovana delka odpove-
di
 15         Let        BOutTlg[0, 0].4 = BMessLength.0           Nastaveni 1. bytu zpravy
...
 16         Let        BOutTlg[0, 0].5 = BMessLength.1           ... ve kterem se nachazi
...
 17         Let        BOutTlg[0, 0].6 = BMessLength.2           ... take delka zpravy.
 18         Let        BOutTlg[0, 1] = 0x0E                      Kod prikazu
 19         Let        BOutTlg[0, 2] = 0x00                      Typ cidla
 20         Let        BOutLen=3                                 Aktualni delka zpravy
 21         Call       599                                         Podprogram pro vypocet
CRC
 22         Let        B_L2_Event = 0x0002                       Generuj udalost Send
 23         If         @ComFree, :NONE
 24           Call     590
 25         EndIf

ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Lib506                                24   Belimo-Cteni stavu externiho kontaktu
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ

  1         Let        BT_Operace = 7
  2         REM        "Cteni stavu externiho kontaktu"
  3         REM        "Vstupem je byte, ktery povoluje ‡ten˘ hod
                       noty extern˘ho kontaktu na vstupu pohonu Y
                       "
  4         For        Bixxa, 0.0, 20.0, 1.0, :NONE
  5           Let      BOutTlg[0,Bixxa]=0
  6         EndFor
  7         Let        BOutTlg[0, 0].0 = 0                       1. byte musi mit bit 0=0
  8         Let        BOutTlg[0, 0].1 = 0                       1. byte musi mit bit 1=0
  9         Let        BOutTlg[0, 0].2 = (BAddr-1).0             Nastaveni 1. bytu zpravy
...
 10         Let        BOutTlg[0, 0].3 = (BAddr-1).1             ... ve kterem se nachazi
...
 11         Let        BOutTlg[0, 0].7 = (BAddr-1).2             ... adresa
 12         Let        BReqNdx=1
 13         Let        BMessLength = 1                           Delka odesilane zpravy
 14         Let        BAnswrLength = 11                         Pozadovana delka odpove-
di
 15         Let        BOutTlg[0, 0].4 = BMessLength.0           Nastaveni 1. bytu zpravy
...
 16         Let        BOutTlg[0, 0].5 = BMessLength.1           ... ve kterem se nachazi
...
 17         Let        BOutTlg[0, 0].6 = BMessLength.2           ... take delka zpravy.
 18         Let        BOutTlg[0, 1] = 0x4B                      Kod prikazu
 19         Let        BOutLen=2                                 Aktualni delka zpravy
 20         Call       599                                         Podprogram pro vypocet
CRC
 21         Let        B_L2_Event = 0x0002                       Generuj udalost Send
 22         If         @ComFree, :NONE
 23           Call     590
```

```
 24         EndIf

ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Lib590                                73   _Belimo-Stavovy automat
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ

  1         REM        "Rozskok dle stavu automatu"
  2         Switch     B_L2_Status, :NONE                          Rozskok dle stavu
  3           Case     1, :NONE                                    Idle: nic
  4             REM    "Zapis vystupniho telegramu"
  5              If     B_L2_Event.1, :NONE                            Udalost Send - vysli
ramec
  6               For  Bixxd, 0.0, 10.0, 1.0, :NONE
  7                 Let  BOutTlgBuf[0,Bixxd]=BOutTlg[0,Bixxd]
  8               EndFor
  9               Let  BOutLenBuf=BOutLen
 10               Let  @ComFree = 0
 11               Let  @PlatnaData = 0
 12               ComWrite  :50001, BOutTlgBuf, 0, BOutLenBuf,
                       NONE, NONE                                 Vyslani ramce
 13               TmoStart  :50002, 100                           TMO na odpoved
 14               Let  BReqStatus = 0x0002                        Stav pozadavku -> Busy
 15               Let  B_L2_Status = 2                            Stav -> Receive
 16               Let  BInLen=0
 17             EndIf
 18           EndCase
 19           Case     2, :NONE                                     Receive: ceka na prijem
znaku
 20             REM    "Cteni vstupniho telegramu"
 21             If     B_L2_Event.2, :NONE
 22               ComRead  :50001, BInTlg, BInLen, BInLen, NONE
 23               TmoStart  :50002, 100                           TMO meziznakovy
 24               REM  "-==Testovani zda se nevyskytla nejaka ch
                       yba==--"
 25               REM  "Je skutecna delka odpovedi rovna pozadova
                       ne?"
 26               Let  @Test1=BAnswrLength==BInLen
 27               If   @Test1, :NONE
 28                 Let  @Test1=0
 29                 Let  BInLen=0
 30                 REM  "Zjisteni shodnosti dotazu a 1.casti odp
                       ovedi"
 31                 REM  "Zacatek odpovedi je vzdy shodny s tvare
                       m dotazu"
 32                 TmoStop  :50002                               Zruseni TMO
 33                 Let  Bixxd=0
 34                 Repeat  :NONE
 35                   Let  @Chyba1=BInTlg[0,Bixxd]!=BOutTlgBuf[0,
                       Bixxd]
 36                   If  @Chyba1, :NONE
 37                     REM  "Chyba"
 38                     Let  BReqResult = 0x0008
 39                     Let  BErrType = 0x0001
 40                     Break
```

```
 41                  EndIf
 42                  Let  Bixxd=Bixxd+1
 43                  Let  @Test2 = Bixxd >= BOutLenBuf
 44                Until  @Test2                                    |
 45                REM  "Cteni 1. bytu odpovedi"
 46                REM  "Pokud je spodni pulbyte 1. bytu odpoved
                      i = 0xD a soucasne jeho nejvyssi bit (tj.7
                      ) je 0, je prenos v poradku"
 47                Let  Bixxa=BInTlg[0,BOutLenBuf]
 48                Let  @Chyba2=NOT(Bixxa.7) AND (Bixxa.0) AND N
                      OT(Bixxa.1) AND (Bixxa.2) AND (Bixxa.3)     Hodnota neni 0x_D
 49                If  @Chyba2, :NONE
 50                  REM  "Chyba"
 51                  Let  BReqResult=0x0008
 52                  Let  BErrType = 0x0002
 53                EndIf
 54                REM  "Konec algoritmu"
 55                Let  B_L2_Status = 0x0001                       Stav -> Idle
 56                Let  BReqStatus = 0x0004
 57                Let  @ComFree = 1
 58                Call  594
 59              EndIf
 60            EndIf
 61             If     B_L2_Event.3, :NONE                           Udalost Tmo - vyprsel
timeout
 62              REM  "Chyba - Vyprsel TimeOut"
 63              TmoStop  :50002                                  Zruseni TMO
 64              Let  BReqResult = 0x0008                         Vysledek -> Err
 65              Let  BErrType = 0x0004
 66              Let  BReqStatus = 0x0004                         Stav pozadavku -> Finish
 67              Let  B_L2_Status = 0x0001                        Stav -> Idle
 68              Let  BInLen = 0
 69              Let  @ComFree = 1
 70            EndIf
 71          EndCase
 72        EndSwitch                                              ... L2_Status
 73        Let        B_L2_Event = 0x0001                         Udalost zpracovana

ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Lib592                                2   _Belimo-Preruseni od prijmu
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ

  1         Let        B_L2_Event = 0x0004                         Udalost Char - prijem
znaku
  2       Call       590

ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Lib593                                2   _Belimo-Preruseni od timeoutu
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ

  1         Let        B_L2_Event = 0x0008                         Udalost Tmo - vyprsel
timeout
  2       Call       590

ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
```

```
Lib594                                44   _Belimo-Konec prijmu,dekodovani odpoved
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ

  1         REM        "Detekce a zaznam chyb"
  2         Call       597
  3         REM        "Dekodovani odpovedi na zaklade typu opera
                       ce"
  4         Switch     BT_Operace, :NONE
  5           Case     1, :NONE
  6             REM    "Na prikaz pro zapsani polohy..."
  7             REM    "...neni ocekavana zadna odpoved..."
  8           EndCase
  9           Case     2, :NONE
 10             REM    "Prijata poloha zarizeni"
 11             StrParse  BInTlg, 5, NONE, "", BinIntBE, 2, "",
                       Bixxa, NONE.0, 0.0
 12             Let    BResult[BAddr,0] = Bixxa
 13             StrParse  BInTlg, 7, NONE, "", BinIntBE, 2, "",
                       Bixxa, NONE.0, 0.0
 14             Let    BResult[BAddr,1] = Bixxa
 15          EndCase
 16          Case     3, :NONE
 17             REM    "Prijata teplota zarizeni"
 18             StrParse  BInTlg, 6, NONE, "", BinIntBE, 2, "",
                       Bfxx, NONE.0, 0.0
 19             Interpol  Bfxx, Bfxx, B_R2T
 20             Let    BFResult[BAddr,0] = bfxx
 21          EndCase
 22          Case     4, :NONE
 23             REM    "Prijato Minimum a Maximum"
 24             StrParse  BInTlg, 5, NONE, "", BinIntBE, 2, "",
                       Bixxa, NONE.0, 0.0
 25             Let    BResult[BAddr,2] = Bixxa
 26             StrParse  BInTlg, 9, NONE, "", BinIntBE, 2, "",
                       Bixxa, NONE.0, 0.0
 27             Let    BResult[BAddr,3] = Bixxa
 28          EndCase
 29          Case     5, :NONE
 30             REM    "Prijaty byte popisujici stav zarizeni"
 31             Let    BResult[BAddr,4] = BInTlg[0,5]
 32          EndCase
 33          Case     6, :NONE
 34             REM    "nastaveni funkce pro vycitani stavu exter
                       niho kontaktu "
 35          EndCase
 36          Case     7, :NONE
 37             REM    "P©ijaty byte se stavem externˇho kontaktu
                       "
 38             Let    BResult[BAddr,5] = BInTlg[0,6]
 39             REM    "Jestlize je BResult[BAddr,5] = 1 nebo 2 -
                       > Otevreny kontakt"
 40          EndCase
 41        EndSwitch
 42        Let        BReqStatus = 1                          Stav pozadavku -> Idle
```

```
 43         Let        BReqNdx=0
 44         Let        @PlatnaData = 1

ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Lib597                                13   _Belimo-Detekce a zaznam chyb
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ

  1         If         BReqResult.3, :NONE                        Chyba
  2           If       BErrType.0, :NONE
  3             Let    BCntErrorNEQ = BCntErrorNEQ + 1            Pocitadlo chyb
  4           EndIf
  5           If       BErrType.1, :NONE
  6             Let    BCntErrorD = BCntErrorD + 1                Pocitadlo chyb
  7           EndIf
  8           If       BErrType.2, :NONE
  9             Let    BCntErrorTMO = BCntErrorTMO + 1            Pocitadlo chyb
 10           EndIf
 11         EndIf
 12         Let        BReqResult = 0x0000                        Vymazani priznaku chyby
 13         Let        BErrType = 0

ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Lib599                                15   _Belimo-Vypocet CRC
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ

  1         Let        BParCross=0
  2         REM        "Cross parity"
  3         For        Bixxb, 0.0, 7.0, 1.0, :NONE
  4           Let      BParCross.Bixxb=0
  5           For      Bixxc, 0.0, 7.0, 1.0, :NONE
  6             Let    BParCross.Bixxb=BParCross.Bixxb XOR BOutTl
                       g[0,Bixxb].Bixxc
  7           EndFor
  8         EndFor
  9         Let        BParCross=BParCross << (8-BOutLen)
 10         Let        BOutTlg[0,BOutLen]=BParCross
 11         Let        BOutLen=BOutLen+1
 12         REM        "Lenght Parity"
 13         Let        BParLength=BOutLen
 14         ChkCreate  BOutTlg, 0, BParLength, BOutLen, XOR-8,
                       BOutLen
 15         Nop
```