VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

Řízení pohonů pomocí PLC s využitím sběrnice CAN

PLC DRIVE CONTROL BY MEANS OF CAN BUS

DIPLOMOVÁ PRÁCE
DIPLOMA THESIS

AUTOR PRÁCE **Tomáš Polach**
AUTHOR

VEDOUCÍ PRÁCE **Ing. Radomil Matoušek Ph.D.**
SUPERVISOR

BRNO 2008

z

y

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Tomáš Polach

Bytem: Sivice 266

Narozen/a (datum a místo): 29.1.1982 v Brně

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta strojního inženýrství

se sídlem Technická 2896/2, 616 69 Brno

jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....

(dále jen „nabyvatel“)

Čl. 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):

- □ disertační práce
- ■ diplomová práce
- □ bakalářská práce
- □ jiná práce, jejíž druh je specifikován jako

.....

(dále jen VŠKP nebo dílo)

Název VŠKP:	Řízení pohonů pomocí PLC s využitím sběrnice CAN
Vedoucí/ školitel VŠKP:	Ing. Radomil Matoušek, Ph.D.
Ústav:	Ústav automatizace a informatiky
Datum obhajoby VŠKP:	1 .6.2008

VŠKP odevzdal autor nabyvateli v:

- ■ tištěné formě – počet exemplářů 1
- ■ elektronické formě – počet exemplářů 2

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - □ ihned po uzavření této smlouvy
   - □ 1 rok po uzavření této smlouvy
   - ■ 3 roky po uzavření této smlouvy
   - □ 5 let po uzavření této smlouvy
   - □ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: V Brně dne: 23.5. 2008

…….. ……

Nabyvatel Autor

Abstrakt:

Diplomová práce se zabývá návrhem a aplikací řídícího algoritmu všesměrového autonomního robotu. Robot je řízen pomocí PLC řady X20 společnosti Bernecker&Rainer. Pohon je realizován třemi integrovanými regulačními pohony IclA D065 společnosti SIG Positec, komunikujícími po sběrnici CAN. Robot dále využívá čidlo snímání polohy podvozku z důvodu možného prokluzu kol na povrchu se sníženou adhezí. Toto čidlo využívá pro komunikaci s automatem sběrnici RS 232. Díky všem výše uvedeným vlastnostem a napájení z baterie, se jedna o plně autonomní zařízení.

Abstract:

The theme and goal of this diploma is a draft and a method of application of a drive algorithm of an omni directional robot. The robot is driven with the aid of a PCL from the X20 series, produced by Bernecker&Rainer. The propulsion is provided using 3 integrated regulating driving mechanisms, IclA D065 produced by SIG Positec, which communicate via the CAN bus. Furthermore, the robot uses sensors to read the positioning of its chassis and undercarriage in order to identify the traction quality of the surface of the terrain below and subsequently reduce skidding. This sensor uses the RS 232 bus to communicate with the PLC. Thanks to all the above-mentioned and characteristics of power supply, the result achieved is a totally autonomous machine.

Klíčová slova:

PLC (programovatelný automat), CANopen, pohon, program, sběrnice, čidlo

Key words:

PLC (programmable logical controller), CANopen, actuator, program, bus, sensor

Diplomová práce vznikla při řešení výzkumného záměru Inteligentní systémy v automatizaci podporovaného MŠMT ČR pod registračním číslem MSM 0021630529.

Acknowledgement: This research has been supported by the Czech Ministry of Education in the frame of MSM 0021630529 Research Intention *Inteligent Systems in Automation*.

Obsah

1. Úvod

Požadavky na zvyšování kvality i kvantity při stejných nebo mnohdy i nižších nákladech v průmyslové výrobě jsou v současnosti celosvětovým trendem a v nejbližší době se tento směr vývoje pravděpodobně měnit nebude. Nejmarkantněji je to patrné v automobilovém a elektrotechnickém průmyslu.

Aby bylo možné všem těmto požadavkům vyhovět, je nutné nahrazovat lidské zdroje průmyslovou automatizací. Jedná se pak především o roboty a manipulátory, jejichž řízení obstarávají počítače a programovatelné automaty. Roboti jsou schopni prakticky nepřetržité práce, bez rizika snížení kvality, nebo navýšení finální ceny výrobku. Dalším důvodem nasazení automatizovaných strojů jsou fyzické přírodní nebo výrobní podmínky v místě vykonávání dané práce.

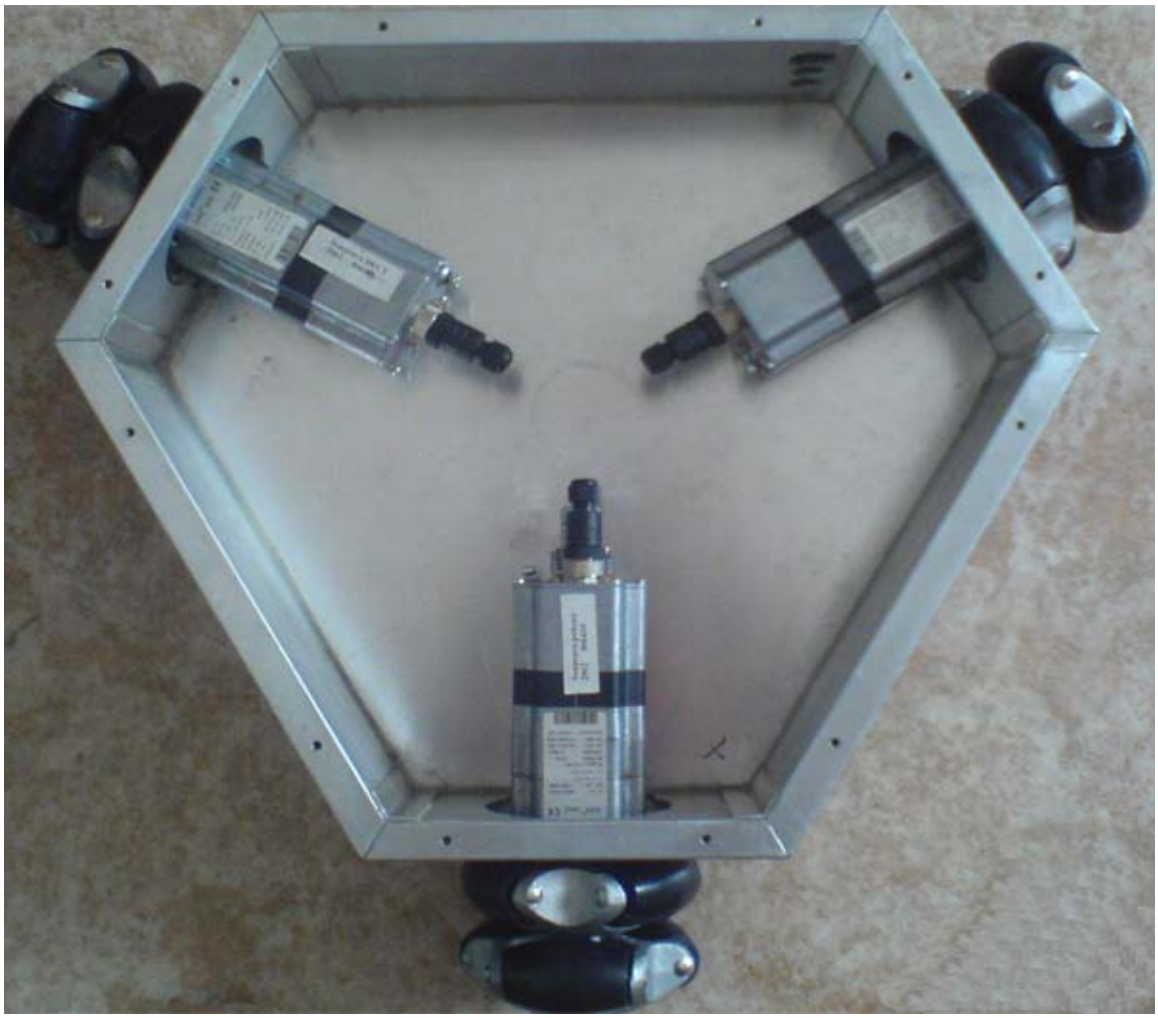
Aby byla výroba efektivní, musí být často schopná plynule přejít na jiný typ nebo druh výrobku na stejné výrobní lince. To je důvod proč se zavádí stupně řízení (centralizované i decentralizované – dle potřeby) s možnou změnou řídícího i výrobního programu. Aby robot mohl pracovat je však třeba více než jen řídícího průmyslového počítače nebo PLC, je také třeba komunikační sběrnice, různých pohonů, čidel, spínačů atd. Komunikace po průmyslových sběrnicích umožňuje řídící jednotce mít v každém okamžiku informace o stavu všech periferiích, které ovládá a také s nimi komunikovat za účelem jejich řízení v reálném čase, což je pro průmyslové využití podstatná vlastnost. Pohony se starají o vlastní pohyb zařízeni. Aby bylo možné pomocí pohonů vykonávat přesnou práci je třeba znát, v jakém stavu se jednotlivé části zařízení nachází a mít možnost nastavit požadované polohy, či dynamiku pohybu. K tomu se využívá zpětné vazby z rozličných čidel a koncových spínačů.

Diplomová práce se zabývá návrhem a aplikací programového řídícího algoritmu všesměrového autonomního robotu řízeného právě pomocí PLC, konkrétně pomocí moderní řady X20 compact společnosti Bernecker&Rainer Automation. Jako akční členy jsou využity tři integrované regulační pohony IclA D065 společnosti SIG Positech. Pohony disponují vlastním výkonovým kontrolerem, integrovaným absolutním enkoderem a komunikujícími pomocí standardní průmyslové sběrnice CAN. Konstrukčně jsou motory na podvozkové části robotu rozloženy do tvaru hvězdy. Pro pohyb robotu jsou rovněž důležitá Omni Stanfordská kola.

Robot dále využívá optické čidlo snímání relativní polohy, které bude umístěno ve spodní podvozkové části. Toto čidlo bylo integrováno pro případnou kompenzaci prokluzu kol na povrchu se sníženou adhezí. Komunikační protokol čidla pracuje pro PLC ve zcela nekompatibilním režimu a jeho připojení k PLC je tak řešeno pomocí speciálního převodníku, který s PLC jednostranně v asynchronním režimu komunikuje pomocí RS 232. Implementace daného čidla i vlastní programové řešení v rámci PLC je nestandardní a v dané oblasti poměrně inovativní.

2. Popis robotu

Jedná se o autonomní systém, schopný jakéhokoliv pohybu v rovině dle zadaných parametrů souřadnic a rychlosti přesunu. Povely mohou být zadávány ze stacionární (externí) stanice, komunikace mezi počítačem a PLC se realizuje pomocí rozhraní Ethernet.



Obr. 1 Podvozek robotu osazený pohony a Omni Stanfordskými koly

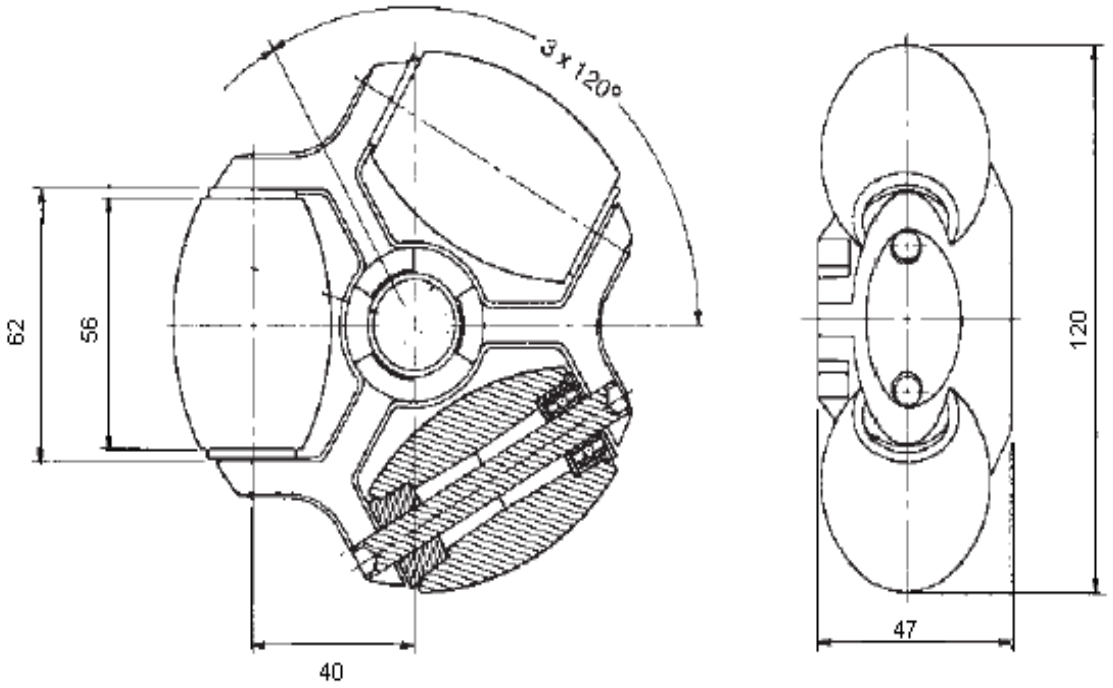
2.1. Mechanické vlastnosti a konstrukce

Tělo podvozku robotu je vyrobeno z kovu a má tvar polygonu, jehož základem byl rovnostranný trojúhelník. Na podvozku jsou umístěny tři integrované polohy IclA Positec, které rozmístěny po 120°. Dále pak řídící PLC a napájecí zdroj, který je tvořen celkem čtyřmi akumulátory.

Pro pohyb robotu byly použity speciální pojezdová kola - Omni Stanfordská kola, vyráběná firmou Interroll, která jsou usazena na hřídelích integrovaných pohonných jednotek na vnější straně podvozku. Kola jsou složena z celkem šesti valivých součástí vždy po třech (po 120° na obvodu kola) na jednom ze dvou kruhových elementů. Oba tyto elementy jsou vzájemně posunuty o 45° a složeny do páru tak, aby při jakémkoli natočení měl valivý „soudeček“ styk s podložkou. Dle zatížení jaké bude na Omni Stanfordská kola působit se volí ložiskové uložení valivých součástí. Pro malé zatížení jsou určena kluzná ložiska a pro velké zatěžování se používána valivá ložiska.

Výhodou je manipulovatelnost zařízení. Vzhledem ke konstrukci kol a tříosému systému podvozku se může robot otáčet kolem své osy, nebo v každém okamžiku změnit směr jízdy bez nutnosti zastavení a natočení na nové souřadnice.

Kladnou vlastností zvolených kol byla jejich nosnost, která pro jeden funkční pár činí 110 kg. Naopak nevýhodou celého systému Omni Stanfordských kol je riziko znehybnění celého zařízení z důvodu zaseknutí valivých částí kteréhokoli z kol. Další značnou nevýhodou řešení této konstrukce je překonávání povrchových nerovností a překážek. Důvodem je malá světlá výška podvozku robotu, která činí jen 35mm. Ta byla záměrně zvolena nižší z důvodu vyšší stability (podvozek s pohony jsou základem pro budoucí vývoj, celkovou výšku a hmotnost zařízení dnes ještě bohužel neznám).



Obr. 2 Všesměrové Omni Stanfordské kolo [6]

2.2. Základní struktury řízení

Celé řízení je rozděleno do dvou úrovní. První a vyšší z nich je řízení robota ze stacionární stanice, kdy jsou zadávány povely pro robota jako celek (jedná se o komunikaci mezi řídícím počítačem a programovatelným automatem). Druhou pak samotné řízení vlastní částí robotu, kdy se zadávají příkazy pro jednotlivé integrované pohony a probíhají zde všechny výpočtové algoritmy rychlostí jednotlivých kol atd. (jedná se o komunikaci mezi programovatelným automatem, a polohovým čidlem umístěném na podvozku).

Řízení na první úrovni se realizuje pomocí Ethernetu , na druhé úrovni probíhá komunikace po sběrnici CAN pod komunikačním profilem CANopen. Úkolem je této diplomové práce je především řešení na druhé úrovni řízení a proto jí bude dále věnována větší pozornost. Celý systém se dá rozdělit no několika základních bloků, kterými jsou:

a) Stacionární stanice:

Jako stacionární stanice je použit osobní počítač (nebo notebook), na kterém je nainstalován software Automation studio společnosti B&R, které slouží jako vývojové prostředí a ve spojení s Etrernetovým rozhraním jako komunikační prostředek, kterým se přenáší data do paměti automatu.
Komunikuje s robotem (s programovatelným automatem), požadavky zadávaných v podobě parametrů pro přesun, nebo naopak získává informace o stavu systému.

b) Programovatelný automat B&R řady X20:

Programovatelný automat (PLC - Programable Logical Controller) zajišťuje Jak komunikaci se stacionární stanicí (ethernet), tak i s integrovanými pohony (po sběrnici CAN i využitím komunikačního profilu CANopen) a v neposlední řadě také s čidlem polohy (komunikující po sběrnici RS 232).

Programovatelný automat od společnosti B&R byl vybrán záměrně z několika důvodu, zejména pak z důvodu snadného připojení všech použitých komunikačních rozhraní, protože všechny výše zmíněné komunikační rozhraní jsou v podporovány. Dále pak z důvodu nízké spotřeby energie, která tvoří pouze 2,7 W.

PLC tvoří rozhraní mezi oběma úrovněmi řízení a zpravují všechny pohybové algoritmy (výpočet pohybových parametrů pohonů komunikací mezi nimi), proto jsou na tyto zařízení kladeny velké požadavky na rychlost zpracování dat a způsob psaní jednotlivých částí řídících algoritmů.

c) Pohonné jednotky – IclA Positec:

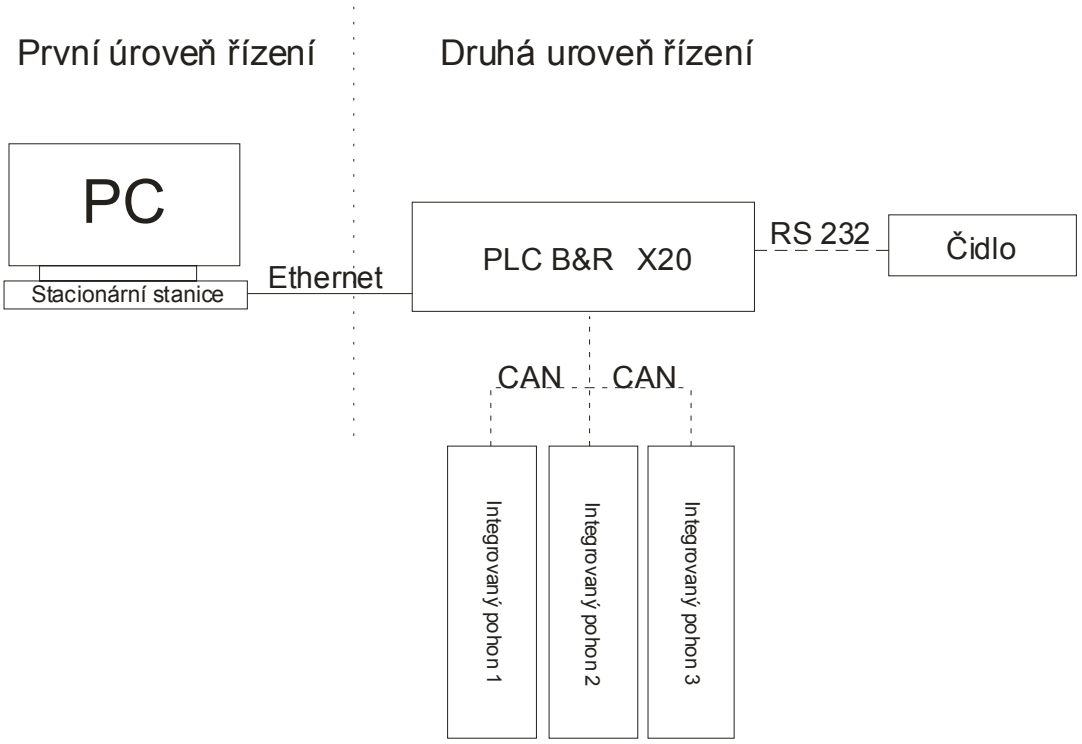
Regulační integrované pohony IclA Posilec představují z hlediska automatizačního pojetí akční členy mobilního robotu. S řízením (automatem) komunikují pomocí sběrnice CAN s profilem CANopen. Integrované pohony jsou řízeny výlučně z programovatelného automatu a data které se k nim dostávají jsou výsledkem výpočtů polohových algoritmů. Na přesnosti a rychlosti akčního zásahu závisí chování celého systému robotu.

d) Čidlo:

Je využíváno ke snímání relativní polohy robotu a je namontováno na spodní části podvozku. Podobně jako integrované pohony tvořily akční členy regulační soustavy tak čidlo je využito jako zpětná vazba. Informace z něj získané jsou posílány do řídícího automatu po sériové lince (RS 232), kde jsou následně vyhodnoceny a případně použity v komparačním algoritmu (pokud robot nedosáhnul přesně cíle, souřadnice takto získané jsou dány jako nové výchozí a pohonné jednotky dostanou od automatu příkaz k posunu dle nových parametrů).

Jako čidla polohy byla v diplomové práci použita vyhodnocovací část počítačové myši (komunikující přes sběrnici RS 232).

Všem výše uvedeným částem bude později věnována zvláštní pozornost v podobě samostatných kapitol.



Obr. 3 Schéma struktury úrovní řízení

3. Pohonné jednotky – IclA Positec

Nejpodstatnější vlastností mobilního robotu je schopnost přesně vykonávat pohyb dle zadaných podmínek (trajektorie, rychlost) a to v reálném čase. Jednotlivé pohony musí umět rychle a bezchybně komunikovat s řídící částí (automat X20), aby bylo možné ovládat jednotlivé kola s požadovanou přesností. Proto je výhodné použít komunikační sběrnice, v tomto případě se nabízí sběrnice CAN (na protokolu CANopen), kterou podporují jak integrované regulační pohony IclA Positec, tak i řídící programovatelný automat společnosti B&R.

Obr. 4 Regulační pohon IclA [16]

3.1. Integrovaný regulační pohon

IclA - Integrated closed loop Actuator

Jde o kompaktní zpětnovazební regulační pohon se zabudovaným mikroprocesorovým regulátorem, komunikujícím na rozhraní CAN. Je určen pro strukturované pohonné systémy, kde se uplatní jako autonomní akční člen s vlastní inteligencí.

V pouzdře pohonu se mino již uvedeného mikroprocesorového regulátoru a komunikačního rozraní také výkonový stupeň, polohový enkodér, stejnosměrný motor a čelní převodovka.Pohony nepotřebují žádnou přídavnou brzdu díky velkému přídržnému momentu v bezproudovém stavu.

Další velkou výhodou použitých motorů je možnost řízení většího počtu pohonů díky jejich propojení po sběrnici (CAN). Každé zařízení připojené na sběrnici (pohony i

řídící automat) má svoji adresu, čímž získává celý systém větší přehlednost a zařízení je snadněji dostupné. Klesají i finanční nároky na fyzické vedení (při rozsáhlejších sítích). Řídící systém (PLC) posílá pohonům po sběrnici příkazy a data pro jejich nastavení (rozběhová a brzdící rampa, rychlost, pracovní proud – potažmo kroutící moment atd.). A naopak, pohony jsou schopny zpětně řídícímu systému poskytovat získané informace jako jsou například:aktuální poloha a rychlost, jejich aktuální nastavení. Díky této zpětní vazbě lze takto získané informace využít ke zpětné kontrole, nebo pro případnou vizualizaci.

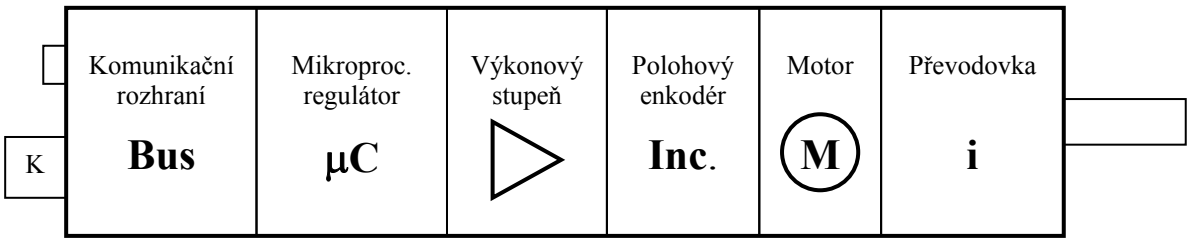
3.1.1. Základní parametry pohonu

Výrobce	Berger Lahr
Typové označení	D065 DC024 S018
Převodový poměr	160:9
Jmenovitý moment	2,8 Nm
Přídržný moment	0,7 Nm
Jmenovité otáčky	210 min^{-1}
Polohové rozlišení	1,667°
Jmenovitý výkon	80W
Napájecí napětí	24 V_{DC} ±20 %
Jmenovitý proud	3,9 A
Proud při chodu naprázdno	< 0,5 A
Proudový odběr v pohotovostním režimu	0,05 A
Hmotnost	1,3 kg
Maximální přípustná radiální síla	200 N
Maximální přípustná axiální síla	80 N
Maximální teplota okolí	60 °C
Relativní vlhkost	15% až 85%
Stupeň krytí	IP54
Izolační třída	F
Průměrem hřídele	10 mm
Příruba	66 mm

pozn.: Společnost SIG Positec se roku 2001 přejmenovala na původní jméno Berger Lahr, proto je v tabulce jako výrobce uváděn již aktuální název společnosti

3.1.2. Části integrovaného pohonu

Základem integrovaného pohonu IclA je DC motor ve spojení s čelní převodovkou a snímačem polohy (Inc. – polohový enkodér). Dále se pohon skládá z mikroprocesorového regulátoru, výkonového zesilovače a sběrnicového rozhraní CAN s komunikačním protokolem CANopen .



Obr. 5 Uspořádání regulačního pohonu IclA SIG Positec

Komunikační rozhraní:

Je reprezentováno sběrnicí typu CAN (s komunikačním profilem CANopen). Tento způsob síťového připojení dovoluje propojení většího počtu těchto pohonných jednotek a je tak umožněno jejich současné řízení pomocí jednoho řídícího zařízení (např. pomocí programovatelného automatu, nebo počítače).

Mikroprocesorový regulátor:

Mikropočítač zajišťuje řídící i ochranné funkce. Nepřetržité je monitorování vzniku podpětí nebo přepětí chránění motor proti přetížení. V klidovém stavu mikropočítač zajistí automatické vypnutí napájení výkonových zesilovačů pohonu, aby nedošlo k přehřátí a tím odstavení pohonu. Zde jsou zpracovávány informace přijaté přes komunikační rozhraní, které jsou dále upraveny pro potřeby dalších částí integrovaného pohonu.

Výkonový stupeň:

Elektronika v této části integrovaného regulačního pohonu IclA zpracovává a modifikuje instrukce z mikroprocesorového regulátoru a s patřičně upravenými signály řídí samotný motor.

Polohový enkodér:

Poloha rotoru je snímána třemi halovými snímači. Řídicí elektronika tohoto stupně z naměřených údajů vypočte tzv. „přibližnou“ absolutní hodnotu polohy rotoru s rozlišením 12±1 INC (inccremental – přírůstek) na jednu otáčku. Polohový enkodér dále přepočítává aktuální polohu rotoru do 32 bitové absolutní hodnoty.

Před vypnutím pohonu a po ukončení regulačního procesu ukládán do vnitřní datové paměti pohonu komutační stav společně s absolutní hodnotou polohy rotoru. Když je regulační pohon znovu zapnut, senzor absolutní hodnoty rozpozná, zda byla hřídel mechanicky pootočena.

Motor:

Motorová část je reprezentována stejnosměrným bezkartáčovým motorem s permanentními magnety, který pracuje s vnitřním rozlišením 12 [Inc] (na jednu otáčku). Díky použití zpomalovací převodovky je dosaženo rozlišení přibližně 213 [Inc] na jednu otáčku výstupní hřídele.

Zadávání hodnot rychlosti, zrychlení a cílové pozice (jakožto počtu otáček), se realizuje přepočtem na motorové přírůstky [Inc], tj. počet otáček [Inc], rychlost [$Inc.s^{-1}$], rychlení [$Inc.s^{-2}$].

Motor generuje dostatečný přídržný moment oproti magnetickému poli permanentních magnetů rotoru. V klidovém stavu dosahuje tato hodnota přídržného momentu motoru 0,7 Nm (tato hodnota odpovídá as 20% nominálního momentu motoru). V případě použití pohonů IclA jako akčních členů mobilního robotu není třeba řešit případné brždění systému proti nechtěnému pohybu (a to ani při předpokládaném zatížení systém tak jak je zatím navržen počítá s celkovou maximální hmotností až 180 kg).

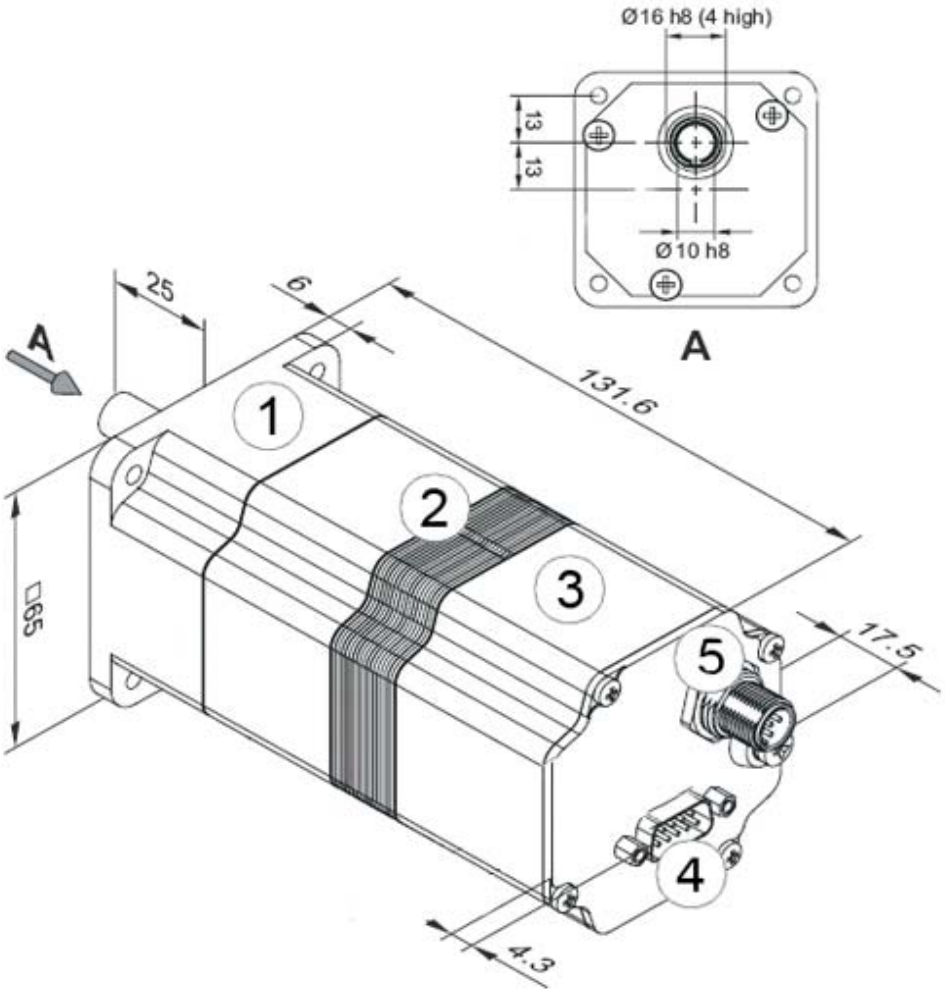
Převodovka:

V kovovém těle opláštění integrovaného pohonu je zabudována bezúdržbová čelní převodovka. Do vložek ošetřených trvanlivou bronzovou vrstvou jsou umístěny kovová převodová kola. Účinnost převodovky je přibližně 77.8%. Pro aplikaci mobilního robotu byl zvolen převodový poměr 160:9. Vzhledem ke zvolenému uložení převodových kol odpadá nutnost mazání.

3.1.3. Mechanické uspořádání pohonu

Celkové uspořádání samotných částí je patrné z obrázku 6. Napájecí napětí 24 V_{DC} se přivádí do pohonu přes 9-pinový konektor (typu Canon) - na zadním čele motoru(4), na kterém jsou mimo jiné vstupy pro ruční ovládání (obou směrů). Popis jednotlivých pinů tohoto konektoru je patrný z obr.7. Zapojení konektoru pro signály sběrnice CAN (6) je popsáno na obr.8.
Na čelní straně pohonu je hřídel vystupující z převodovky (na ni budou osazeny všesměrová kola).

1) Čelní převodovka
2) Stejnosměrný motor
3) Snímač polohy, výkonová a řídící elektronika
4) Konektor pro napájecí napětí a signálové rozhraní (manuální operace a nouzové vypnutí)
5) Konektor rozhraní CAN

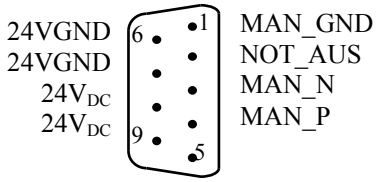


Obr. 6 Mechanické uspořádání regulačního pohonu IclA Positec [16]

3.1.4. Elektrické zapojení konektorů

Zapojení napájecího konektoru:

Je realizováno devíti pinovým konektorem typu CANON - zástrčka. Rozmístění pinů je znázorněno na obr. 7 a jejich význam je popsán v tabulce.



Obr. 7 Zapojení pinů napájecího konektoru [17]

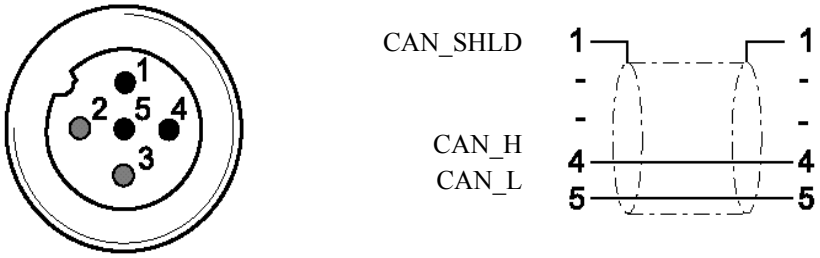
Tabulka signálů napájecího konektoru pohonu IclA

Pin	Signál	Význam jednotlivých pinů
1	MAN_GND	Zem, referenční potenciál pro řídicí signály MAN_P a MAN_N
2	NOT_AUS	Řídicí signál pro bezpečnostní odpojení přes bezpečnostní vypínač
3	MAN_N	Řídicí signál pro záporný pohyb při manuálním režimu
4	MAN_P	Řídicí signál pro kladný pohyb při manuálním režimu
6	24VGND	GND pro napájecí napětí 24 V_{DC}
7	24VGND	GND pro napájecí napětí 24 V_{DC}
8	24V_{DC}	Napájecí napětí 24 V_{DC}
9	24V_{DC}	Napájecí napětí 24 V_{DC}

Zapojení konektoru rozhraní CAN

Pro rozhraní CAN je použit konektor typu eurofast s typovým označením RSC. Rozmístění jednotlivých pinů konektoru je na Obr. 8 a jejich význam potom v následující tabulce.

Zapojení sběrnicového konektoru CAN:



Obr. 8 Zapojení sběrnicového konektoru [17]

Tabulka signálů CAN

Pin	Signál	Význam jednotlivých pinů
1	CAN_SHLD	Stínění, PE konektor
4	CAN_H	Datový vodič, dominantní 1
5	CAN_L	Datový vodič, dominantní 0

4. Systém řízení mobilního robotu-PLC

Pro řízení robotu byl vybrán kompaktní programovatelný automat společnosti Bernecker & Rainner řady X20 (byl vybrán z důvodu snadné implementace obou sběrnic a také z důvodu nízkého energetického zatížení celého systému mobilního robotu). PLC musí být schopno plně koordinovat všechny úkony na druhé úrovni řízení (přepočet souřadnic, komunikace s integrovanými pohony atd..) a navíc komunikovat s první úrovní řízení (stacionární stanicí – po RS 232, nebo pomocí ethernetu.

4.1. Programovatelný automat B&R řady X20 compact

Jede o programovatelný automat rakouského výrobce Bernecker & Rainner s typovým označením X20CP0291. Skládá se z výkonné jednotky, ke které jsou tzv. systémem X2X připojovány další periferní jednotky (vstupní/výstupní jednotky, přídavné moduly např. RS 232 atd.). Filosofií konstrukce PLC řady X20 je modularita, základem je tzv. „back plate“, tedy modul, který je základem a je montován na DIN listu. Tento modul zajišťuje napájení a komunikaci s ostatními moduly pomocí proprietární X2X sběrnice (uživatel se o tuto sběrnici „nemusí starat“ ani nemá oprávnění do ní nějakým způsobem zasahovat). Na „back plate“ moduly se připojují funkční moduly a vlastní uP PLC. Funkční moduly (I/O, čítače, …) disponují odpojitelnými konektory, které značně usnadňují další konektivitu modulů s okolím.

Obr. 9. Programovatelný automat B&R X20 třídy compact [23]

Díky modulárnosti celého takto nastaveného „stavebnicového“ systému je možnost zajištění optimální konfigurace pro každou úlohu. Pro potřeby řízení robotu byl vybrán

automat B&R z několika důvodů: nízká spotřeba – 2,7 W, samostatná CPU jednotka (viz obr. 9) disponuje rozhranim RS232 i CAN.

Na těle automatu je dále umístěn ethernetový konektor RJ45 (automatu je možno nastavit adresy dle protokolu TCP/IP), dva přepínače (nastavení pracovního módu automatu a nastavení adresy), LED indikující napájení a aktuální pracovní stav a pochopitelně konektor pro napájení.

Tabulka hlavních parametrů X20CP0291:

Typ	X20CP0291
Procesor	16 Mhz, architektura RISC
Komunikační rozhraní	RS232, CAN, ethernet RJ45 100MBit/s
Paměť RAM	100 KB SRAM
Paměť uživatelská	1 MB flash PROM
Zpracování jedné instrukce	0,8 μs
Minimální doba cyklu	2 ms
Firmware	2.40
Napájení	24 ± 6V
Spotřeba	2,7 W

4.1.1. Popis zapojení

Napájení

Napájecí napětí, které jak bylo uvedeno výše, činní 24 V_{DC} je přivedeno na svorkovnici. Mobilní robot bude napájen z bateriového zdroje. Při vývoji řídícího algoritmu byl automat napájen ze sítě pomocí stabilizovaného zdroje (viz. Obr. 10). Napájecí vodiče (stejně jako všechny ostatní, včetně signálových) jsou přivedeny na svorkovnici, konektory pracují na stejném principu jako WAGO svorky.

Obr. 10 Zdroj napětí pro PLC B&R

Rozhraní RS 232

Tato část technologie je primárně určena pro programovaní PLC, vizualizaci atd. V této diplomové práci je jí také využito jako rozhranní pro čidlo polohy, dále při získávání základních zkušeností se systémem (pro zadávání základních povelů a funkcí ze stacionární stanice).

Pro rozhraní RS232 bývá standardně použit konektor typu CANON 9. Význam jednotlivých pinů je popsán v následující tabulce:

Tabulka signálů pro RS 232

Pin	Signál	Význam
2	RXD	Příjem dat
3	TXD	Vysílání dat
4	+ $5V_{DC,}$ (max 500 mA)	Zdroj napětí vizualizačních panelů
5	GND	Uzemnění
7	RTS	Žádost o data
8	CTS	Ukončení spojení

Pozn.: pro komunikaci PLC a stacionární stanice je použit křížený kabel s překřížením pinů 2. $\rightarrow$ *3. a 7.* $\rightarrow$ *8.*

Rozhraní CAN

Podobně jako RS 232 je i CAN rozhraní standardizované, ale na rozdíl od předchozího je elektricky izolováno. I zde je realizace pomocí konektoru CANON 9. Následující tabulka ukazuje význam zapojení jednotlivých pinů.

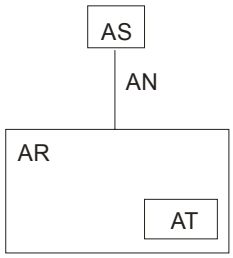
Tabulka zapojení pinů konektoru CAN:

Pin	Signál	Význam jednotlivých pinů
1	CAN_GND	Stínění, PE konektor
4	CAN_H	Datový vodič, dominantní 1
5	CAN_L	Datový vodič, dominantní 0

4.1.2. Popis systému X20

Systém v prostředí B&R je možno rozložit do několika základních bloků:

1) Automation studio (AS) Vývojové prostředí (později jako samostatná kapitola).
2) Automation Net (AN) Reprezentuje veškerou komunikaci PCL a ostatních periferních zařízení s vývojovým prostředím, které jsou připojeny přes jakékoli podporované rozhraní.
3) Automation Runtime (AR) Operační systém, tvoří rozhraní mezi použitým SW a HW.
4) Automation Target (AT) Jde o konkrétní HW na kterém je spuštěn operační systém Automation Runtime.



Obr. 11 Schéma hierarchie v systémech B&R [8]

4.2. Pracovní režimy automatu B&R

Pracovní režimy jsou na automatu voleny pomocí otočného přepínače MODE. Poloha přepínače se detekuje pouze při zapnutí a operační systém si tuto hodnotu zapamatuje. Pokud je během uživatelského programu tato poloha změněna systém vygeneruje příslušné varování.

Na obrázku je čelní panel automatu X20 s přepínači MODE v tabulce jsou uvedeny významy jednotlivých poloh přepínačů.

Pozice přepínače	Popis
00	Přístup k systémové paměti. (např. přehrání verze operačního systému)
01 -FE	Pracovní režim, v tomto režimu poběží řídící automat robotu
FF	Diagnostický mód

Obr. 12 Čelní panel PLC třídy X20 s přepínači [23]

4.3. Vývojové prostředí Automation studio

Tvorba řídícího algoritmu probíhala v prostředí Automation studia společnosti B&R. Studio umožňuje využití jazyků, které jsou součástí normy IEC 1131 (LAD –*Ladder diagram*, IL – *Instruction List*, ST – *Structured text*, SFC – *Sequentail Function chart),* ale také dalších (většinou vyšších) jazyků které tato norma nezahrnuje, jako jsou *C Language* a také *Automation Basic* (který je čistě firemní záležitostí B&R).

Pro tvorbu software řídícího algoritmu mobilního robotu bylo, použito jazyku ST. Jedná se o vyšší jazyk vyvinutý pro účely programování PLC, s jednoduchou a srozumitelnou strukturou. *Structured text* svojí syntaxí velmi připomíná programovací jazyk Pascal, v některých ohledech je ovšem možné najít synktickou podobnost k ANSI C.

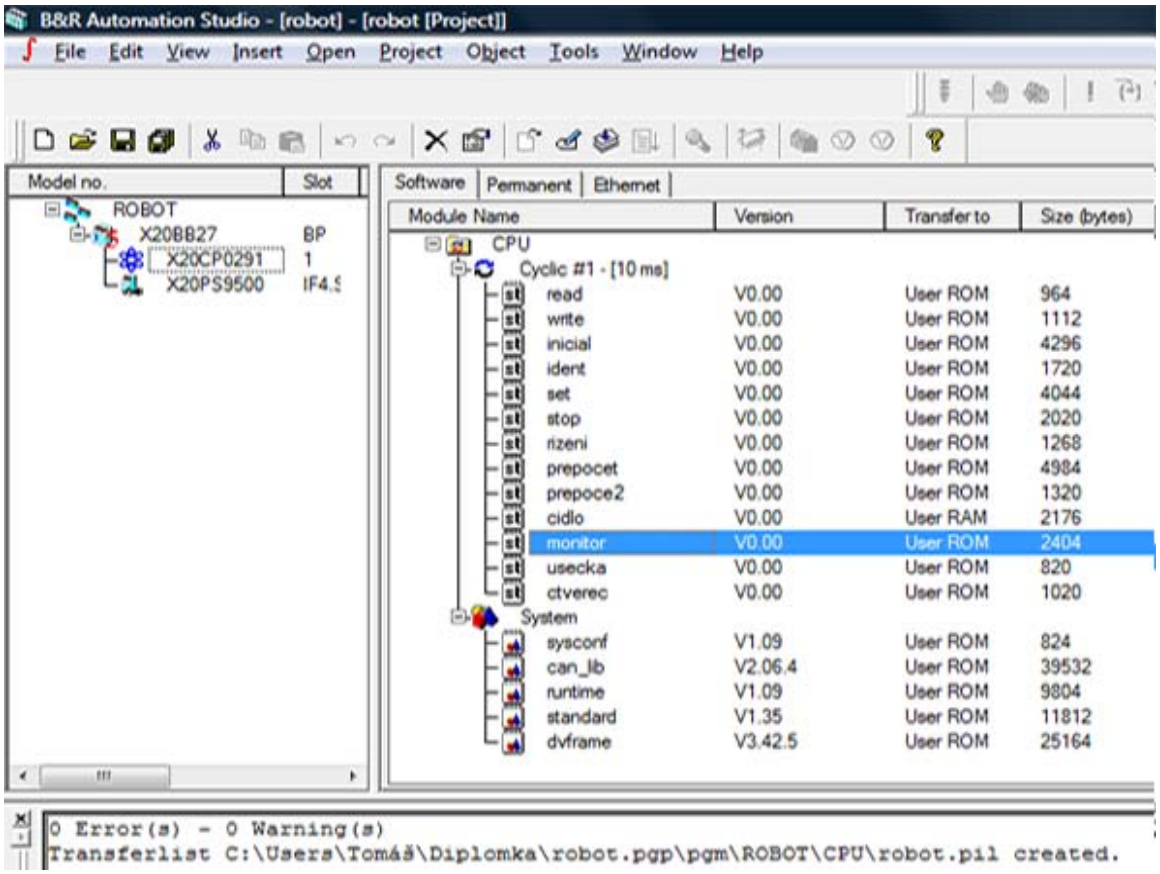
V Automation studiu se nastavují také například komunikační rozhraní (ať už RS232 nebo CAN). Je zde možnost nastavení jak komunikačních rychlostí, portů přes které bude PLC komunikovat se stacionární stanicí, tak také například všechny parametry

potřebné pro komunikaci v Ethernetových sítích s protokolem TCP/IP (zde se nastavení rychlosti děje automaticky, zadávají se pouze IP adresa, maska podsítě a výchozí brána). V Automation studiu je také možno nastavit routing, ale vzhledem ke skutečnosti že pro účely mobilního robotu popisovaného v této diplomové práci nebude nutný, nebude tahle problematika rozebírána.

Mezi často užívané funkce, kterými Automation studio disponuje jsou diagnostické, simulační a vizualizační prostředky. Zde bych vyzvednul zejména *TRACER,* kterým je možno graficky „online“ sledovat vybrané veličiny (např. sledování aktuální polohy motorů). Dále je zde tzv. *WATCH*, který nabízí možnost sledovat hodnoty vybraných veličin při průchodu cyklu programu (výhodné zejména pokud se program krokuje).

Další možností, které nabízí Automation studio je možnost práce bez připojení PLC (simulační mód), které jsem využil při psaní a zkoušení řídícího programu. Tato možnost nabízí pracovat stejně jako s automatem, s tím rozdílem že zde chybí možnost výběru jak automatu tak i vstupních/výstupních obvodů. Pro výběr této možnosti je nutné se rozhodnout prakticky ihned po zvolení nového projektu, kde se místo možnosti volby příslušného systému (X20, 2003, Power Panel ..) vybere možnost „*PC Based AR“,* která v sobě skýtá možnost volby z dalších třech možností.

Pro účely práce v simulačním režimu na odzkoušení základních funkcí řídícího programu postačí první nejjednodušší možnost z daných tří, která je AR000 (ta zahrnuje simulaci AR, AT, AN dle struktury na Obr. 11). Zbylé dvě nabízí rozšířené možnosti v oblasti realtime operací se sítěmi a dále zpřístupní některé knihovny, které nejsou pro první zvolený simulační mód běžně dostupné.



Obr.13 Okno Automation studia s bloky programu mobilního robotu

4.4. Nastavení komunikace z prostředí Automation studia

Sériová komunikace – RS 232

Historicky prvním způsobem komunikace všech programovatelných prostředků (ať už se jedná o jednočipy, mikrokontrolery, nebo později o PLC) je sériová komunikace (RS 232, RS 422, RS 485). Pro tuto možnost jsem se rozhodnul v první fázi z důvodu jednoduché a snadné komunikace (později bude RS 232 uvolněno pro připojení čidla).

Zde postačí z prostředí Automation studia nastavit příslušný komunikační port stacionární stanice (např. COM1, nebo COM2). Dalším krokem je nastavení přenosové rychlosti od 115200 bps do 1200 bps (automat se stacionární stanicí komunikují 57600 bps). Následuje nastavení parity (jako výchozí je nastaveno Even, což se ale týká komunikace PLC a stacionární stanice nikoliv však komunikace PLC s čidlem, tam je parita nastavena na hodnotu none, tj. „0“). Ostatní nastavení sériové komunikace je možno pro tyto účely ponechat. V případě čidla se mění ještě délka komunikačního rámce, která je zkrácena o bit parity (děje se automaticky).

Ethernet

Dnes kromě sériového rozhraní nabízí většina výrobců programovatelných automatů i možnost komunikace přes klasické Ethernetové rozhraní. Této možnosti je využito i v případě komunikace robotu s nadřazeným systémem (stacionární stanicí) a to z důvodu obsazení sériového portu čidlem podvozku.

Pro nastavení parametrů tohoto komunikačního rozhraní postačí zadat příslušné adresy dle standardů protokolu TCPIP:

- IP adresa,
- maska podsítě,
- výchozí brána.

Všechna ostatní nastavení probíhají automaticky a běžný uživatel k nim nemá přístup.

4.5. Vlastnosti softwaru pro PLC

Při programování v Automation studiu jsou bloky, tzv. „tasky“, zpracovávány cyklicky, z čehož plyne řada výhod ale i problémů. Velkou výhodou je determinismus vykonávaných příkazů (v každý okamžik je přesně známo která instrukce se zrovna vykonává) a tím také pro některé operace zvýšit prioritu tj. zařadit ji do jiné taskové třídy (s menší dobou na průběh cyklu). Lépe se také hledají a odstraňují chyby. Na druhou stranu při cyklickém zpracování bloků programu vzniká nebezpečí např. při nesplnění některé podmínky – systém čeká na odpověď a program může uváznout v uzavřené smyčce. Díky tomu již nestihne dokončit zpracování tohoto bloku programu v čase k tomu určeném. To má za následek vygenerování chybové hlášení, které je nutno dále ošetřit. Vzhledem k těmto skutečnostem je nutné vytvářený program koncipovat podle výše uvedených faktů.

5. Sběrnice CAN a CANopen profil

Sběrnice CAN byla zvolena z důvodu dosažení větší spolehlivosti a rychlosti komunikace (přenosová rychlost je 1000 kbit.s^{-1} na vzdálenost 40 m, prodloužením se pochopitelně snižuje) mezi řídícím automatem a jednotlivými integrovanými pohony, dále pak ke zjednodušení a větší přehlednosti celého zapojeni.

Tato kapitola se zaměřuje jen na ty části, které popisuji komunikaci mezi automatem a pohony (celá problematika týkající se CAN sběrnice je podstatně obsáhlejší), referenčně se tato část práce odvolává především na zdroj [3].

5.1. Sběrnice CAN

Sběrnice CAN (*CAN – Controller Area Network*) byla původně vyvinuta společností Bosch pro rychlé přenosy dat s vysokou úrovní spolehlivosti. Vyvíjená byla pro automobilový průmysl, kde našla prvotní využití (např. vůz Škoda Fabia první generace měla hned dvě sběrnice typu CAN). Díky své celkové efektivnosti a nízké ceně nutné součástkové základny a snadné implementaci se velmi rychle rozšířila i v jiný průmyslových aplikacích.

Protokol CAN definuje norma ISO 11898, která popisuje jak specifikaci CAN 2.0A tak později vyvinutou specifikaci CAN 2.0B. Obě výše uvedené varianty popisují pouze fyzickou a linkovou vrstvu podle referenčního modelu ISO/OSI. Schéma bude uvedeno později (viz obr. 17). Aplikační vrstva je definována několika vzájemně nekompatibilními standardy z nichž je pro tuto práci podstatný jen protokol CANopen (dále je to např. Device Net atd.).

CAN je standardizovaná otevřená sběrnice určená ke komunikaci mezi čidly a aktuátory v řízené technologii. Každé zařízeni na CAN sběrnici vysílá a přijímá data prostřednickým zpráv – tzv. *Message* (*Message oriented communication*).

Data jsou mezi jednotlivými zařízeními přenášena sériově, pro časově náročnější aplikace je zvolena vyšší priorita (nevyšší priorita je tedy přidělována operacím řízení v reálném čase a pro to jsou posílány jako první). Velkou výhodou je vysoká spolehlivost přenosu dat. Díky bezpečnostním procedurám které na síti probíhají je pravděpodobnost poruchy přenosu dat nižší jak 4,7.10^{-11}. To v praxi znamená téměř 100% bezpečnost přenosu.

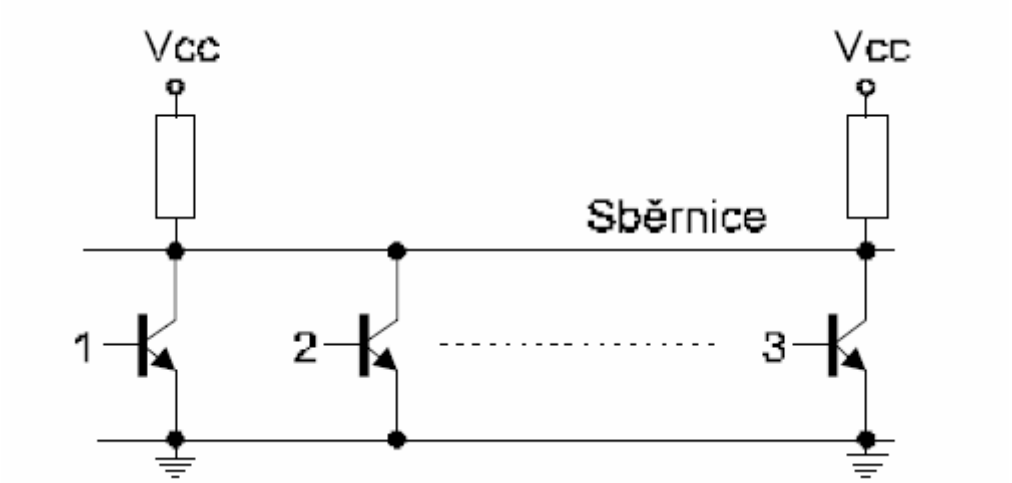
Přenosová rychlost je závislá na délce vedení a pohybuje se od 10 kbit.s^{-1} pro vzdálenost 6,7 km až do 1 Mbit.s^{-1} pro vzdálenost do 40 m (pro účely této práce byla zvolena rychlost 20 kbit.s^{-1}).

5.2. Fyzické uspořádání sběrnice

Základním požadavkem [3] na fyzické přenosové médium je schopnost realizace logické funkce AND (logický součin). Protokol CAN definuje dvě vzájemně komplementární hodnoty bitu na sběrnici. Jsou to *dominant* a *reccesive*. Jde o zobecněný ekvivalent logických úrovní, jejichž hodnoty nejsou přesně určeny a skutečná reprezentace závisí pouze na konkrétní realizaci fyzické vrstvy.

Vysílají-li všechny uzly bit recessive je celá sběrnice ve stavu recessive. Pokud vysílá alespoň jeden uzel bit dominant je na sběrnici stav dominant. Tento stav bude odpovídat logické nule a stav recessive logické jedničce na celé sběrnici.

Pak pokud bude sepnut jeden tranzistor, je na celé sběrnici nastaven stav dominant (logická nula) a nezáleží na tom jestli bude, nebo nebude sepnut nějaký další. Stavu recessive (odpovídající logické jedničce) odpovídá skutečnost, kdy není sepnut žádný tranzistor (platí na celé sběrnici).

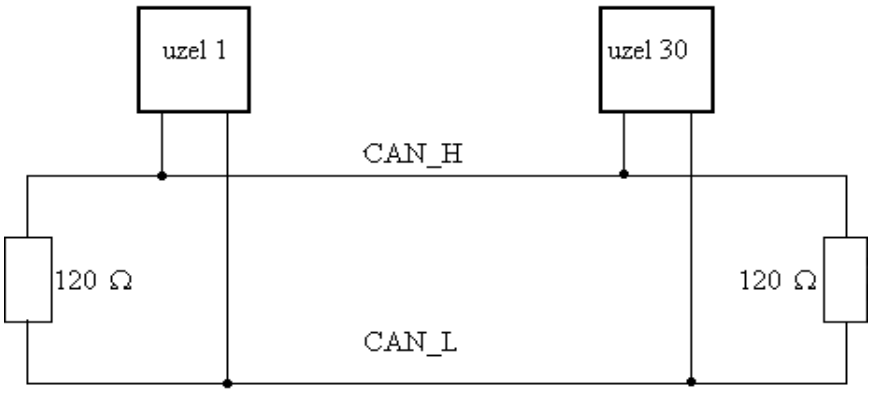


Obr. 14 Příklad realizace fyzické vrstvy protokolu CAN

Sběrnici tvoří dva vodiče CAN_H (CAN Higth) a CAN_L (CAN Low), kde se stavy dominant nebo recessive definují rozdílem napětí na těchto vodičích. Dle definovaných normovaných úrovní je pro stav na sběrnici recessive rozdílové napětí odpovídající $V_{diff} = 0$ V a pro úroveň dominant je rozdílové napětí určeno jako $V_{diff} = 2$ V.

Pro kompenzaci odrazů metalického vedeni je sběrnice na obou koncích přizpůsobena zakončovacími odpory o velikosti 120 Ω.

Vzhledem k zatížení a zajištění definovaných parametrů sběrnice udává norma pro CAN maximálně 30 uzlů připojených na sběrnici.



Obr. 15 Zapojení uzlů na sběrnici CAN [3]

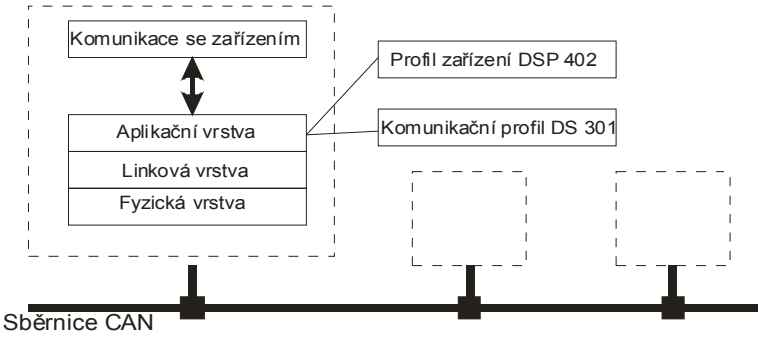
5.3. Komunikace

Komunikace na sběrnici CAN je realizována [5] pomocí posílání datových rámců (*frame*). Každý rámec obsahuje identifikátor zprávy (COB-ID), typ a formát zprávy, délku dat, data a časovou značku. Délka jednoho datového rámce může obsahovat 0 až 8 bytů dat.

COB-ID určuje význam přenášené zprávy a identifikuje i formát rámce (vysílaného nebo přijímaného). Rámce posílaných zpráv mohou mít dva základní typy formátů. Standardní formát („Standard“ – 11 bitové COB-ID, vyplývá ze specifikace 2.0A.), nebo rozšířený formát („Extended“ – 29 bitové COB-ID, odpovídá specifikaci 2.0B.). Integrované regulační pohony IclA SIG Positec jsou kompatibilní se specifikaci CAN 2.0A. Popis jednotlivých bitů rámců a jejich natavování bude uvedeno později.

5.4. CANopen protokol

CANopen je nezávislým popisným jazykem [3] určeným pro komunikaci po sběrnici CAN. CANopen zajišťuje základní datové služby podle třívrstvého datového modelu (vhodné porovnat se standardem ISO/OSI, dále viz obr. 16).



Obr. 16 Vrstvový datový model ISO-OSI pro sběrnici CAN [17]

Fyzická vrstva - *CAN Physical Layer*

Fyzická vrstva definuje mechanické specifikace (délka a typy propojovacích kabelů a typy konektorů) a elektrické specifikace (rozhraní mezi fyzickou vrstvou a fyzickými prostředky) sběrnice CAN. Zajišťuje také fyzický přenos bitů po metalickém vedení.

Linková vrstva - *CAN Data Link* Layer

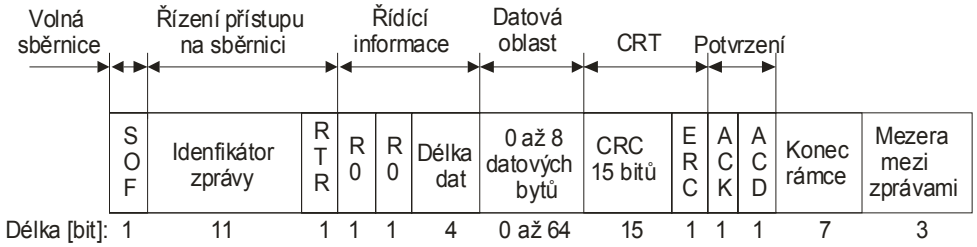
Linková vrstva má dvě podvrstvy, které jsou MAC(*Medium Access Contro l*) a LLC (*Logical Link Control*). První z nich řídí přístup všech uzlů k médiu s rozlišením priority jednotlivých zpráv. Jejím úkolem je mimo jiné kódování dat, vkládání doplňkových bitů do komunikace *Stuffing/Destuffing*, detekce chyb, jejich hlášení opravy a potvrzování správně přijatých zpráv. Druhá podvrstva filtruje přijaté zprávy.

Aplikační vrstva - *CAN Application Layer*

Přístup aplikačním procesům ke komunikačnímu systému poskytuje aplikační vrstva. Dále umožňuje provádění základních funkcí této úrovni odpovídajících (vytváření, udržování a rušení spojení mezi jednotlivými zařízeními, čtení a zápis přenášených dat)

5.5. Popis datové zprávy CAN

Jak bylo uvedeno, výše komunikace na sběrnici CAN probíhá pomocí přenosu zpráv. Ty přenášejí komunikační objekty a další administrativní a řídící informace, které se starají o bezchybný datový přenos. V této kapitole bude ukázána struktura a popis takové zprávy podle specifikace CAN 2.0A.



Obr. 17 Datová zpráva (*Data Frame*) [3]

Význam jednotlivých částí datové zprávy :

SOF – *Start Of Frame* - začátek zprávy, jeden bit (dominant)
Identifikátor zprávy - *COB-ID* – udává význam přenášené zprávy, 11 bitů
RTR bit – *Remote Request* – příznak udává jestli se jedná o datovou zprávu,nebo o žádost o vysílání dat. V datové zprávě musí mít úroveň dominant, v žádosti o data má úroveň recessive, jeden bit
R0, R1 – rezervované bity, vždy po jednom bitu
Délka dat – počet přenášených datových bytů ve zprávě. Povolené hodnoty jsou 0 až 8, 4 bity
Datová oblast – *Data field* – pole pro přenos dat, 0 až 64 bitů
CRC kód - zabezpečující kód, generující polynom $X^{15}+X^{10}+X^8+X^7+X^4+X^3+X+1$, 15 bitů
ERC – *ERC delimeter* - ERC oddělovač, jeden bit (recessive)
ACK – *Acknowledge* – bit potvrzení, jeden bit
ADK – *Acknowledge delimeter* - oddělovač potvrzení, jeden bit (recessive)
Konec rámce – *End Of Frame* – ukončení zprávy, 7 bitů (recessive)
Mezera mezi zprávami – *Interframe Space* – oddělení dvou po sobě jdoucích zpráv, 3 bity (recessive)

5.6. Objekty CANopen

Všechny procesy pod CANopen jsou realizovány pomocí objektů. Tyto pak uskutečňují veškeré potřebné úkony, sloužící jako komunikační objekty pro datový přenos po sběrnici, řídící procedury připojení a kontrolu zařízení, které jsou v síti připojené.

5.6.1. Objektový slovník protokolu CANopen

Nejpodstatnější částí spojení pro všechny objekty je objektový slovník každého síťového zařízení. Ostatní zařízení zde mohou najít seznam všech objektů, pomocí kterých mohou navázat kontakt s tímto zařízením. V seznamu jsou např. objekty pro provádění komunikačních úloh a objekty funkcí zařízení pracujících pod protokolem CANopen.

Každé zařízení tedy disponuje vlastním objektovým slovníkem, ve kterém jsou uvedeny všechny komunikační objekty pro CANopen komunikaci s ostatními zařízeními.

Index objektu:

Všechny objekty jsou adresovány šestnácti bitovým indexem, ve tvaru čtyřmístného hexadecimálního čísla. Objekty jsou pak sloučeny do objektového slovníku a seřazeny podle hodnot jejich indexů a dle zařazení do dané objektové skupiny.

Tabulka řazení objektů do skupin dle jejich indexu:

Index	Objektová skupina
1000_h - $1FFF_h$	Komunikační profil (standardizovány v DS 301 specifikaci)
2000_h - $5FFF_h$	Profil výrobní specifikace zařízení
6000_h - $9FFF_h$	Standardizovaný profil zařízení (specifikace DSP 401)

Sub-index

Každý objekt dále obsahuje jeden nebo více osmi bitových sub-indexů, specifikují jedinečnost datového pole v objektu. Sub-indexy stejně jako indexy jsou reprezentovány hexadecimální soustavou.

Následující příklad demonstruje hodnoty indexů a sub-indexů pro nastavení pracovního rozsahu pohonu (pracovní oblast pohonu). Po identifikaci limitů dojde k softwarovému vypnutí polohovací funkce (viz. kapitola 7.4.3).

Tabulka indexů a sub-indexů pro polohování pohonu:

Index	Sub-index	Jméno	Význam
$607D_h$	00_h	-	-
$607D_h$	01_h	Minimální limit polohy	Spodní hodnota pracovní oblasti
$607D_h$	02_h	Maximální limit polohy	Horní hodnota pracovní oblasti

5.7. Profily CANopen

Profily zařazují objekty do skupin dle jejich přidělených úkolů [3, 5].

Standardizované profily

Standardizované profily popisují objekty, které mhou být využívány ostatními zařízeními bez dalších přizpůsobení. Automatizační asociací – CiA (Automation Association) jsou standardizovány následující základní profily:

- komunikační profil DS 301 (*the communikations profile)*
- profil zařízení DSP 401 (*the device profile)*

Oba tyto profily jsou součástí aplikační vrstvy v třívrstvém modelu ISO/OSI, jak je patrno z obr. 17. Další služby jsou dostupné pouze pomocí specifických výrobních profilů.

Komunikační profil DS 301 - *Communication profile DS-301*

Tento profil tvoří rozhraní mezi profily zařízení (device profile DSP-401) a sběrnicí CAN. Byl specifikován v roce 1995 pod názvem DS-301 a definuje jednotný standard pro výměnu dat a parametrů mezi jednotlivými zařízeními pod CANopen. Komunikační profil dále zajišťuje např. zařízení a monitorování zařízení v síti.

Základní objekty komunikačních zařízení: PDO *Process Data Object,*
SDO *Service Data Object,*
SYNC *Synchronization Object,*
NMT *Network Management Object,*
LMT *Layer Management Object.*

Přesný význam těchto objektů bude popsán v dalších kapitolách.

Profil zařízení DSP 402 - D*evice profile DSP-402*

Tento profil popisuje standardizované objekty pro polohovací úlohy, monitorování a nastavování zařízení.

Úkolem těchto objektů je:
- monitorování a kontrola stavu,
- standardizované nastavování parametrů,
- změny mezi monitorovacími a provozními režimy.

Specifické výrobní profily MSP - *Manufacturer-specific profile*

Základní funkce zařízení mohou být využívány prostřednictvím objektů standardizovaných profilů zařízení. Celý funkční rozsah jednotlivých zařízení je však možné využít pouze pomocí specifických výrobních profilů zařízení. Tyto definují objekty se specielními funkcemi zařízení, které mohou být používány pod CANopen.

5.8. Seznam použitých zkratek

CAN – *Controller Area Network***-** Standardizovaná sběrnicová síť
COB – *Communikation Objecst* – Komunikační objekt. Přenosový element v síti CAN
COB-ID - *Communikation Objects Identifier* – Identifikační rámec linkové vrstvy. Každý COB je jedinečně identifikován v síti CAN číslem COB-ID,m to určí prioritu COB pro přenos
RTR – *Remoute Transmition Requiest* – Žádost o data
SYNC – *Synchronization Objecst* – Specielní objekt pro synchronizaci
PDO – *Process Data Objects* – Objekty pro rychlý přenos informací v síti CAN.Dělí se na T_PDO transmmit PDO vysílání a R__PDO receive PDO příjem dat.
SDO – *Service Data Objects* – Slouží k přenosu systémových dat,. Dělí se na T_SDO a R_SDO
NMT – *Network Management* – Objekty síťového managementu.Slouží k inicializaci zařízení, Monitorování chyb a stavů
LMT – *Layer Management* – Objekty vrstvového managementu.Zajišťují konfiguraci komunikačních parametrů.

6. Objekty komunikačních profilů CANopen

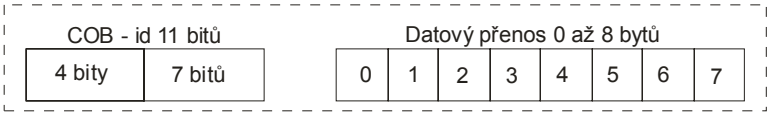
Tyto objekty jsou standardem v CANopen komunikačním profilu DS 301. Komunikační objekty lze rozdělit do čtyř základních skupin podle způsobu jejich využití [3, 4, 5]:

- SDO objekty (T_SDO, R_SDO) – Objekty určené pro zápis nebo čtení do objektových slovníků zařízení jednotlivých zařízení.
- PDO objekty (T_PDO1, R_PDO1, T_PDO2, R_PDO2) – Objekty sloužící pro rychlý přenos dat v síti podle protokolu CANopen. Proto je s výhodou můžeme použít pro aplikace pracující v reálném čase (např. mobilní robot).
- Objekty kontrolních zpráv:
    - SYNC: Objekty pro synchronizaci síťových zařízení (synchronní spouštění pohonů).
    - EMCY: Bezpečnostní objekty pro zobrazování chyb zařízení nebo jejich periferií.
- Služby managementu sítě a správy vrstev:
    - MNT: Objekty síťového managementu, slouží pro inicializaci zařízení, kontrolu
- chyb a stavů.
    - LMT: Objekty správy vrstev, slouží pro nastavení bázových adres a přenosových rychlostí.

Zjednodušený tvar zprávy CAN

Jak je patrno z obr. 15, velké množství bitů zprávy je využito pro zabezpečení bezchybného přenosu. Tyto bity jsou pak před odesláním zprávy vkládány linkovou vrstvou a opět automaticky odstraněny od odeslané zprávy vrstvou ochrany dat.

Pro práci s komunikačními objekty CANopen a datovou komunikací je CAN zpráva nahrazena zjednodušenou formou, viz následující obrázek 18.



Obr. 18 Zpráva CANopen ve zjednodušeném tvaru [17]

COB-ID - *Communication object identifier*

COB-ID zajišťuje dvě základní funkce:

- sběrnicová arbitráž – definice přenosových priorit
- identifikace komunikačních objektů

Výše uvedený COB identifikátor je definován pro polohovací zařízení v souladu s CANopen specifikací 2.0A a je složen z následujících dvou částí:

- Operační znak – *Function code* (4 bity)
- Uzlové ID – *Node-ID* (7 bitů)

Operační znak

Operační znak klasifikuje komunikační objekty. Bity operačních znaků jsou v COB-ID nejvýznamnější a řídí prioritu přenosu (objekty s nízkým operačním znakem mají vyšší prioritu a tak například objekt s operačním znakem 3 bude odeslán později než objekt s operačním znakem 1)

Adresa uzlu

Každému zařízení musí být přidělena specifická sedmi bitová adresa (Node-ID) v rozsahu 1 až 127 ($7F_{h)}$. Adresa zařízení „0“ je rezervována pro tzv. „*Broadcast transmissoin“* – vysílání zpráv všem zařízením v síti současně.

COB-ID komunikačních objektů

Tabulka komunikačních COB-ID všech objektů. V posledním sloupci jsou vypsány indexy specielních objektů odpovídající změnám nastavení komunikačních objektů a čtení.

Komunikační objekt	**Operační znak**	**Node-ID [1…127]**	**COB-ID**	**Indexy parametrů objektů**
NMT služby	0000	0000000	0	-
SYNC objekt	0001	0000000	128 (80_h)	$1005_{h...}1007_h$
EMCY objekt	0010	- - - - - - -	128 (80_h) + Node-ID	1014_h
T_PDO1	0011	- - - - - - -	384 (180_h) + Node-ID	1800_h
R_PDO2	0100	- - - - - - -	512 (200_h) + Node-ID	1400_h
T_PDO2	0101	- - - - - - -	540 (280_h) + Node-ID	1801_h
R_PDO2	0110	- - - - - - -	768 (300_h) + Node-ID	1401_h
T_SDO	1011	- - - - - - -	1048 (580_h) + Node-ID	1200_h
R_SDO	1100	- - - - - - -	1536 (600_h) + Node-ID	1201_h
NMT kontrola chyb	1110	- - - - - - -	1792 (700_h) + Node-ID	$100C_{h...}100E_h$
LMT služby	1111	110010-	2020 ($7E4_h$), 2021 (7E5h)	

Příklad výpočtu COB-ID [17]

Pro zařazení s bázovou adresou 1_h bude COB-ID komunikačního paketu R_SDO:600 + Node-ID = $600_h + 1_h = 601_h$ (1537_d)

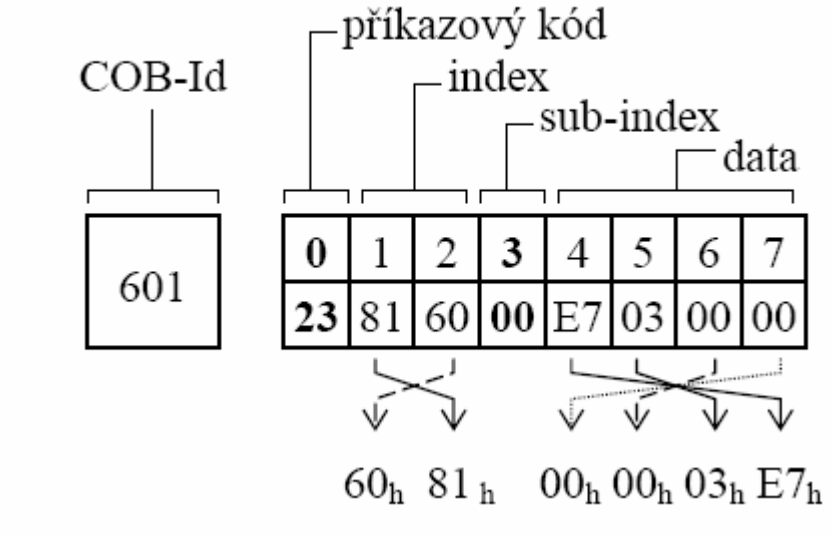
6.1. Service data communication – SDO

Objekty datových služeb jsou využívány pro zpřístupnění záznamů v objektovém slovníku prostřednictvím jejich indexů a sub-indexů. Hodnoty objektů mohou být čteny, pokud je to dovoleno pro danou aplikaci tak i měněny. Jde především o zadávání hodnot rychlosti, zrychlení, čtení aktuální polohy, aktuální rychlosti atd. Data jsou vysílána prostřednictvím T_SDO a přijímaná pomocí R_SDO.

SDO komunikace využívá vztahu Klient/Server. Server je zařízení, kterému se SDO zpráva poukazuje. Komunikace je zavedena klientem za účelem přenosu hodnot parametrů k serveru nebo jejich získání. V obou případech klient zavede komunikaci se žádostí a přijímá odpověď ze serveru. Objekty SDO mají vždy vyšší COB-ID než PDO a proto jsou přenášena s nižší prioritou.

6.1.1. SDO zpráva

SDO zpráva je ve zjednodušené formě složena z COB-ID a SDO datového přenosu [3, 5], kterým mohou být přenášeny až 4 byte dat.



Obr. 19 Příklad SDO zprávy s konverzí hodnot [17]

Datový přenos:

Příkazový ccd kód (*command code*) udává typ SDO zprávy a délku přenášených dat. Index a sub-index určují konkrétní položku v objektu jehož data budou přenášena v SDO zprávě.

Vyhodnocení numerických hodnot (konverze)

Indexy objektů i data jsou přenášena ve formátu Intel systémem „*Flush left*“. Jestliže SDO obsahuje číselné hodnoty o délce větší než jeden Byte, pak tyto hodnoty musí být zkonvertovány byt po bytu před i po přenosu. Příklad takové konverze hodnot je znázorněn výše na obr. 19.

6.1.2. SDO čtení a zápis

Zápis dat

Klient (v tomto případě stacionární stanice) zavádí požadavek zápisu zadáním indexu, sub-indexu, délkou dat a hodnotou dat (požadovaná rychlost, akcelerace, nebo deakcelerace). Příklad takového zápisu je na obr. 19 zápis hodnoty finálové rychlosti pomocí objektu rychlosti 6081_h).

Pohon (server) pošle odpověď, jestli data byla správně zpracována. Odpověď obsahuje stejný index i sub-index, ale neobsahuje již data.

V následující tabulce jsou uvedeny ccd kódy pro zápis hodnot parametrů. ccd se mění v závislosti na typu zprávy a délce přenášených dat.

Tabulka ccd kódů pro zápis hodnot parametrů:

Zprávy	ccd pro délky dat [byte] 4	3	2	1	Význam
Požadavek zápisu	23_h	27_h	$2B_h$	$2F_h$	Parametry přenosu
Potvrzení zápisu	60_h	60_h	60_h	60_h	Odpověď
Chybová odpověď	80_h	80_h	80_h	80_h	Chyba

Jestliže je zpráva vyhodnocena bez chyb odpovědí serveru je ccd = 60_h pokud je detekována chyba v datové oblasti jako odpověď od pohonu dostaneme ccd = 80_h.

Čtení dat

Klient požaduje čtení dat zadává index objektu a sub-index položky která má být čtena. Server potvrdí požadavek odesláním požadovaných hodnot. SDO odpovědí která obsahuje stejný index i sub-index a délku vrácených hodnot udává příkazový kód dle následující tabulky.

Tabulka ccd kódů pro zápis hodnot parametrů:

Zprávy	ccd pro délky dat [byte] 4	3	2	1	Význam
Požadavek čtení	40_h	40_h	40_h	40_h	Žádost pro čtení hodnoty
Potvrzení čtení	43_h	47_h	$4B_h$	$4F_h$	Vrácení čtené hodnoty
Chybová odpověď	80_h	80_h	80_h	80_h	Chyba

[17] Příklad čtení aktuální pozice :
Node-ID $_{Pohonu}$ = 2_h (objekt aktuální pozice 6064_h , sub-index 00_h)
T_SDO: 602_h 40 64 60 00
R_SDO: 602_h 40 64 60 00 E8 03 00 00 =>Aktuální pozice pohonu=$3E8_h$=1000_D [INC]

6.2. Proces Data Communication – PDO

Jsou komunikační objekty určené pro výměnu procesních dat v reálném čase [3, 4, 5], jako je například nastavení nové aktuální polohy, nebo provozního stavu pohonu. Přenos je rychlý, protože nejsou posílána žádná další řídící data a také není žádná odpověď od příjemce.

Proměnná délka dat PDO také zvyšuje datovou prostupnost. PDO zpráva může přenášet až osm bytů dat. Když jsou využity např. jen tři byty jsou přenášeny jen tyto tři byty.

PDO datová výměna

Datová výměna je prováděna komunikačním vztahem producent – konzument a může být spouštěna třemi způsoby:

- Synchronně.
- Asynchronně, řízenou událostí.
- Asynchronně, požadavkem konzumenta (IclA pohony neumí vyžadovat zprávy).

Synchronizace dat je prováděna synchronizačními objekty SYNC, které jsou popsány v další kapitole. Synchronní PDO zpráva je přenášena okamžitě, stejně jako všechny ostatní PDO zprávy, ale je vyhodnocena až po příchodu synchronizační zprávy SYNC. Synchronizační výměna tedy umožňuje spuštění několika pohonů současně.

PDO zprávy, které jsou volané na žádost nebo následkem nějaké události, jsou zařízením zpracována okamžitě. Například zprávu nouzového zastavení je nutné poslat asynchronně. Po příchodu zprávy dojde k okamžitému vypnutí pohonu (ale pouze toho pohonu kterého se to přímo týká).

PDO zpráva

Pro PDO komunikaci využívají integrované regulační pohony IclA následující čtyři objekty :

- asynchronní příjem PDO - R_PDO1
- asynchronní příjem PDO - R_PDO2
- asynchronní vysílací PDO - T_PDO1
- synchronní vysílací PDO - T_PDO2

PDO mapování

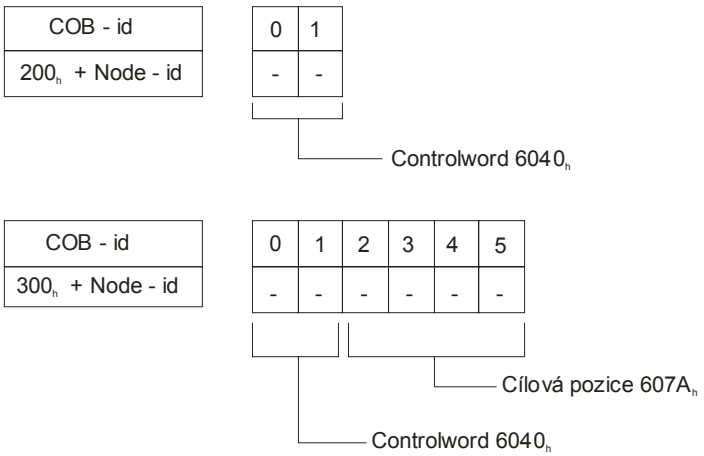
Princip je promítnutí dat vybraných objektů profilu zařízení DSP 402 (objekt cílové pozice, aktuální pozice, řídící slovo, stavové slovo), přímo do datové oblasti komunikačních PDO objektů. Výhody a princip PDO mapování budou ještě dále vysvětleny na příkladě.

Přijímací PDO

Pomocí prvních dvou bytů R_PDO lze pohonu posílat řídící slovo – *controlworld* (6040_h) kterým jsou nastavovány různé stavy pohonů.

První dva byty R_PDO2 jsou také využívány pro příjem řídícího slova. Navíc lze

další čtyři byty využít pro zadání nové cílové pozice – objekt *target positron* ($607A_h$). R_PDO2 je synchronní PDO, tj. je vyhodnocen až po příjmu SYNC zprávy a lze jej tedy použít pro synchronizované spouštění několika pohonů současně.



Obr. 20 Příjem PDO pomocí R_PDO1 a R_PDO2 [17]

Pozn.: PDO zprávy využívají stejnou konverzi hodnot indexů a dat jako SDO

Příklad:
Ukázka zadání cílové pozice třem motorům (10 otáček) s adresami 2_h, 3_h, 4_h a jejich spuštění pomocí SDO a PDO.

a) Pomocí SDO [17]

COB-ID	Data	Význam
602_h	23 7A 60 00 55 08 00 00	600 - T_SDO, 23 ccd , 607A - Index objektu cílové pozice , 00 - Sub-index, 855 - Cílová pozice (8 otáček)
603_h	23 7A 60 00 55 08 00 00	Nastavení pohonu 2
604_h	23 7A 60 00 55 08 00 00	Nastavení pohonu 3
602_h	2B 40 60 00 1F 00	2B - ccd, 6040 - controlword, 00 - Sub-index, 001F - start
603_h	2B 40 60 00 1F 00	Spuštění pohonu 2
604_h	2B 40 60 00 1F 00	Spuštění pohonu 3

b) Pomocí PDO [17]

COB-ID	Data	Význam
302_h	1F 00 55 08 00 00_h	300 - R_PDO2, 001F - controlword (start), 855 - Cílová pozice
303_h	1F 00 55 08 00 00_h	Nastavení pohonu 2
304_h	1F 00 55 08 00 00_h	Nastavení pohonu 3
80_h	0_h	SYNC - synchronizační zpráva (současné spuštění všech pohonů)

Z možnosti využití „b“ jsou patrny následující výhody: Pro nastavení a spouštění je třeba poslat pouze čtyři zprávy, na proti tomu při využití možnosti „a“ je to zpráv šest. Rovněž počet přenesených datových bytů po sběrnici je 18 / 42. Zatížení sběrnice je tedy přibližně poloviční.

Tento způsob je výhodný i pro jeden osamocený pohon. V tom případě bude množství přenesených zpráv nutných k nastavení a spuštění shodný (po jedné pro nastavení i spuštění), ale počet přenesených bytů je pouze 14 v případě užití SDO a 6 v případě přenosu pomocí PDO.

Z toho vyplývá že pro řízení mobilních robotů bude lepší možnost „b“ (klade asi poloviční nároky na vytížení sběrnice, což je pro řízení v reálném čase velká výhoda).

Vysílací PDO

Slouží k přenosu informací o aktuálním stavu pohonu .T_PDO jsou pohonem generovány jako reakce na změny stavů. Uspořádání je obdobné jako je na Obr. 20 (strana 40).
Pro výpočet vysílacích PDO platí:

$$\text{COB-ID T\_PDO1} = \text{Node-ID} + 180_h$$
$$\text{COB-ID T\_PDO2} = \text{Node-ID} + 280_h$$

První dva byty T_PDO1 obsahují stavové slovo (6041_h)ve kterém je obsažen aktuální stav daného pohonu. T_PDO1 je posílaný na základě řízené události, jako například po příchodu stavového slova, nebo při dosažení cílové pozice a proto je asynchronní.

T_PDO2 obsahuje také stavové slovo a aktuální pozici motoru (*position actual value* – 6064_h), ale T_PDO2 ho odvysílá až po příchodu objektu SYNC.

6.3. Synchronization - SYNC

[3, 4, 5]
Důvod používání synchronizace byl již v minulých kapitolách nastíněn. Pomocí synchronizačních objektů SYNC se řídí synchronní výměna zpráv v síti. Data přijatá přes R_PDO jsou zpracovávána až po příchodu SYNC zprávy. Objekty SYNC jsou posílána všem zařízením v síti (i když samy nepřenáší žádná data).

Z hlediska řízení je dobré se zamyslet jestli se vyplatí, aby řídící jednotka generovala SYNC zprávy v pravidelných intervalech, nebo aby byla synchronizační zpráva vyslána až po dokončení určité činnosti (jako např. po dokončení nastavení cílových poloh všech pohonů).

COB-ID pro SYNC objekty

Pro bezpečný a rychlý přenos synchronizačních SYNC objektů, jsou tyto objekty posílány bez potvrzení o jejich přijetí daným zařízením a je jim tudíž přidělena vysoká priorita.

COB-ID SYNC objektů je standardně nastaveno na hodnotu 80_h (128_D). Tato hodnota může být změněna po inicializaci sítě pomocí objektu (1005_h).

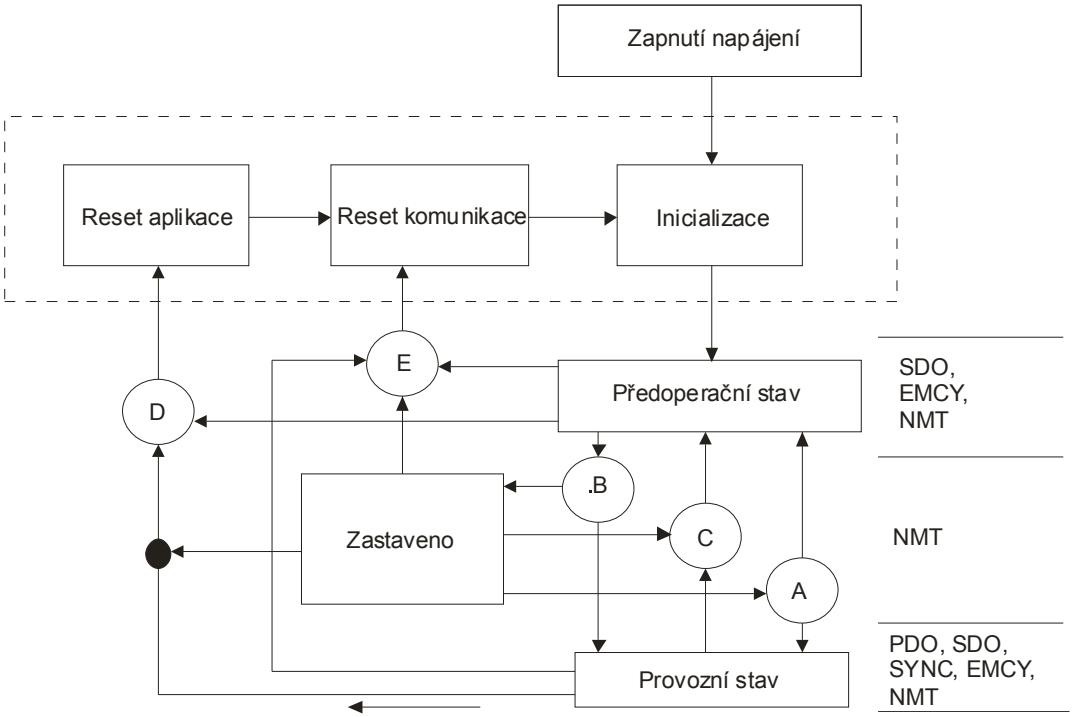
6.4. Network management service – NMT

[3, 4, 5]
Součástí CANopen komunikačního profilu jsou mino jiné NMT (služby síťového managementu). Ty jsou využívány pro zpuštění, zastavování, monitorování zařízení, ale hlavně pro inicializaci sítě a jejich zařízení.

Inicializace pohonu

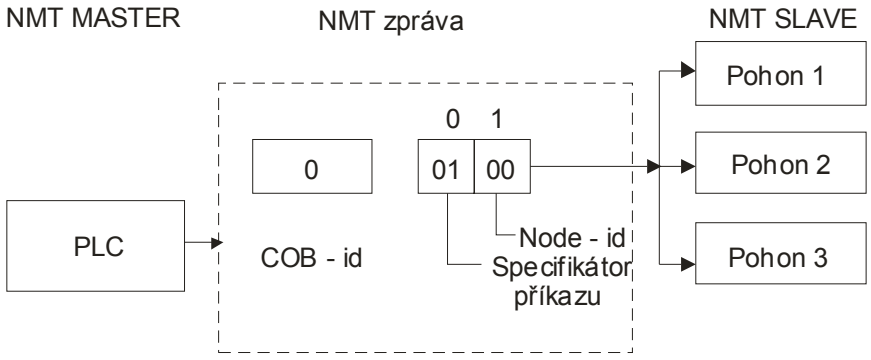
Po zapnutí napájení pohony procházení automatickou inicializační částí, která je připraví pro výkon operací v CAN síti. Po dokončení této fáze se pohon přepne do „předoperačního stavu“ a pošle boot-up zprávu. Do dalších stavů se pohony mohou dostat pouze po příkazu nadřazené řídící jednotky.

Na pravé straně následujícího obrázku jsou jmenovány objekty, které lze použít pro komunikaci v jednotlivých stavech. Sledování těchto stavů je nutné jednak z důvodu samotného řízení a dále potom z důvodu následného ošetření chybových stavů. Způsob přepínání mezi jednotlivými stavy je dán tabulkou na konci této podkapitoly.



Obr. 21 Diagram stavů zařízení a jeho dostupné komunikační objekty [5]

MNT zpráva:



Obr. 22 NMT zpráva [17]

Zprávy jsou přenášeny bez potvrzení příjmu s COB-ID = 0 (tudíž mají při přenosu nejvyšší prioritu). Datová část NMT se skládá ze dvou bytů.

První byte specifikuje *command specifier* (přechod stavů). Druhý byte potom označuje adresu příjemce NMT zprávy s uzlovou adresou $7F_h$ (1 až 127_D). Zpráva s uzlovou adresou rovnou nule je adresována všem zařízením (integrovaným pohonům). Toho se dá využít pro přepínání operačních režimů všech pohonů současně.

Tabulka hodnot příkazových identifikátorů pro přechody stavů:

Příkazový identifikátor	**MNT služba**	**Přechod stavů dle obr. 22**
2_h (2_D)	Start vzdáleného uzlu	A
3_h (3_D)	Zastavení uzlu	B
80_h (128_D)	Vstup do předoperačního stavu	C
82_h (130_D)	Reset uzlu	D
82_h (131_D)	Reset komunikace	E

6.5. Layer management service – LMT

[4, 5]

Služby managementu vrstev (LMT) se využívají pro nastavení adres zařízené v síti CAN (každé zařízení musí mít svoji jedinečnou adresu podobně jako IP adresa v počítačových sítích) a nastavení komunikačních rychlostí (všechny zařízení spolu musí komunikovat stejnou rychlostí).

Adresa uzlu

U integrovaných regulačních pohonů jsou adresy uzlů (Node-ID) implicitně nastaveny na adresu $7F_h$ (127_D). Protože pohony jsou použity tři byly adresy změněny na nové 02_h 03_h 03_h a to právě prostřednictvím LMT služeb.

Následující tabulka ukazuje příklad nastavení adresy uzlu pro první pohon s Node-ID = 02_h, nastavení adres ostatních dvou pohonů bude obdobné.

Tabulka nastavení adresy uzlu:

COB-ID	**Data**	**Význam**
$7E5_h$	04 01 -- -- -- -- -- -- $_h$	Přepnutí do konfiguračního režimu
$7E5_h$	11 02 -- -- -- -- -- -- $_h$	Přiřazení nové uzlové adresy na 02_h
81_h		Boot-up zpráva
$7E5_h$	17 -- -- -- -- -- -- -- $_h$	Uložení nové konfigurace
$7E5_h$	17 00 00 -- -- -- -- -- $_h$	Odpověď - nová konfigurace uložena
$7E5_h$	04 00 -- -- -- -- -- -- $_h$	Přepnutí zpět do operačního režimu

Nastavení přenosové rychlosti

Pohony mají nastaveny komunikační rychlost na 20 $kbits^{-1}$ a tato rychlost jim byla nastavena pomocí PC (rychlost se mění v podle nejpomalejšího zařízení v síti).

Tabulka indexu pro změnu rychlosti:

Index	0	1	2	3	4	5	6	7	8
Přenosová rychlost [kbits^{-1}]	1000	800	500	250	125	100	50	20	10

Příklad postupu pro změnu rychlosti:

COB-ID	**Data**	**Význam**
$7E5_h$	04 01 -- -- -- -- -- -- $_h$	Přepnutí do konfiguračního režimu
$7E5_h$	13 00 07 -- -- -- -- -- -- $_h$	Nastaví přenosovou rychlost na "7" (odpovídá 20 KBd)
$7E4_h$	13 00 00 -- -- -- -- -- -- $_h$	Odpověď - nastavení přenosové rychlosti povoleno
$7E5_h$	17 -- -- -- -- -- -- -- $_h$	Uložení nové konfigurace
$7E4_h$	17 00 00 -- -- -- -- -- -- $_h$	Odpověď - nová konfigurace uložena
$7E5_h$	15 E8 03 -- -- -- -- -- -- $_h$	Aktivace změny přenosové rychlosti
$7E5_h$	04 00 -- -- -- -- -- -- $_h$	Přepnutí zpět do operačního režimu

6.6. Emergancy servise – EMCY

[3, 4, 5]

Služby zabezpečení udávají zprávy o chybových zprávách zařízení. Zpráva o chybě je poslána objektem EMCY s hodnotou 80_h (128_D) a to současně všem zařízením. Popis významu jednotlivých bytů této zprávy je možné nalézt v seznamu chybových hlášení v datashitu integrovaných regulačních pohonů IclA SIG Positec.

Boot-up zpráva

Komunikační profil DS 301 definuje další úkol pro EMCY a sice posílání boot-up zprávy. Tato zpráva oznamuje všem zařízením v síti CAN že zařízení které boot-up zprávu vyslalo je připraveno pro činnost v síti CAN. Zpráva je složena ze dvou částí COB-ID a EMCY objektu (nepřenáší žádná data). Standardní nastavení COB-ID je 80_h (128_D) + Node-ID.

Této skutečnosti je využito v programu řízení robota. Boot-up zpráva slouží jako inicializace ke startu řídící části programu (po zapnutí napájení čeká program na příchod této zprávy). U integrovaných pohonů IclA dojde k vysílání této zprávy asi po čtyřech sekundách.

7. Operační režimy pohonů

Integrované regulační pohony IclA mohou provozovány v několika pracovních režimech. Přesný význam těchto pracovních režimů bude později ještě popsán. Pro účely této práce využijeme jen tři z nich. Zde bude rozveden pouze základní – Manuální režim

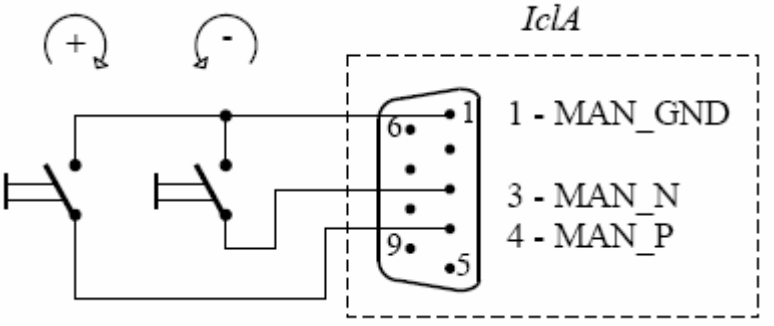
Tabulka pracovních režimů integrovaných pohonů IclA:

Pracovní režim	**Řízení po sběrnici CAN**	**Řízení bez sběrnice**
Manuální režim	Ano	Ano
Polohovací režim	Ano	Ne
Referencování	Ano	Ne

Manuální režim

Pro nastavení tohoto pracovního režimu není nutné připojení sběrnice (pro všechny ostatní již ano, jak je patrné z tabulky výše), takže přechod do jiného režimu je možný jen s využitím sběrnice. Pro účely řízení mobilního robotu je tento režim nepoužitelný. Jeho využití spočívá v testování základních funkcí pohonu bez připojené sběrnice CAN.

Po přivedení napájecího napětí na svorky konektoru, dojde k inicializaci pohonu a následnému automatickému přechodu do manuálního režimu. Zapojení konektoru je na Obr. 23 (níže). Spojením pinů MAN_GND a MAN_P na konektoru pohonu je zvolen kladný směr otáčení hřídele, spojením pinů MAN_GND a MAN_N je zvolen záporný směr otáčení výstupního hřídele pohonu.

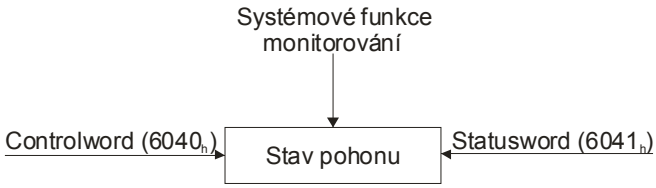


Obr. 23. Schéma zapojení signálového konektoru pohonu IclA pro manuální režim [17]

7.1. Přechody mezi pracovními režimy

Pro potřeby přechodu jednotlivých režimů, případně přechodu mezi pracovními režimy, je nutné pohon postupně přepínat do definovaných stavů (nelze přepnout např. z manuálního režimu na referencování). Posloupnost jednotlivých přechodů daných stavů záleží na aktuálním stavu pohonu a je nutné s ohledem na tuhle skutečnost vytvářet daný program řízení (při výskytu chyby pohon automaticky přejde do stavu „9.chyba“ (dle obr. 25) a pro návrat je nutné vytvořit rutinu která postupně obstará přechody všemi nutnými stavy zpět do původního nastavení před vyvolání chyby.

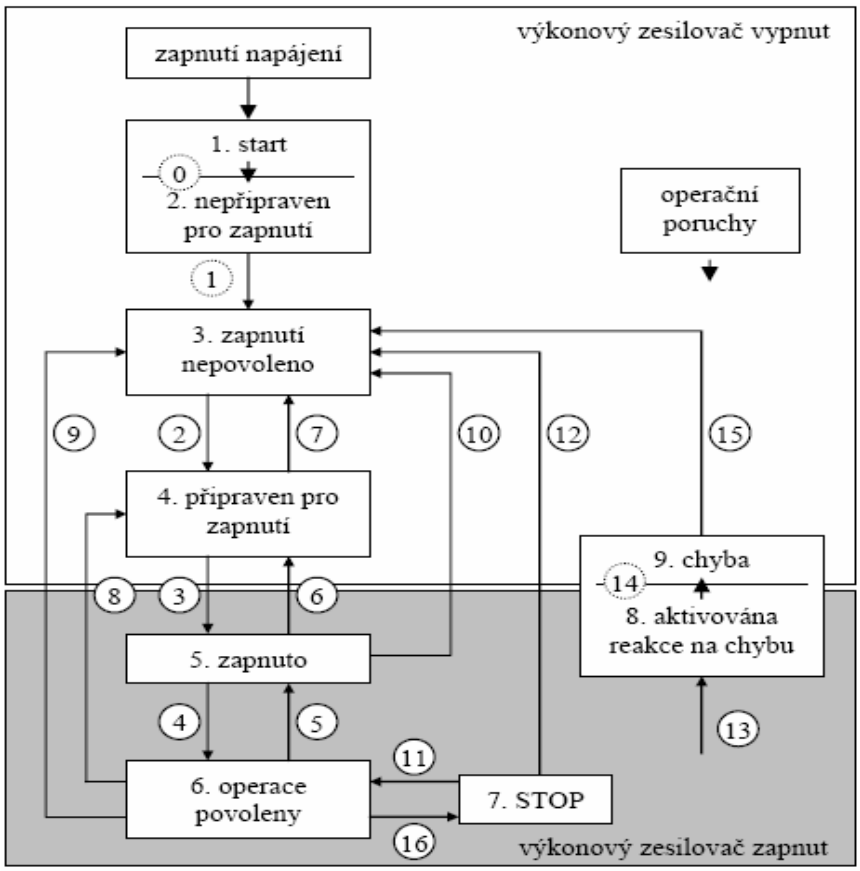
Uživatel nastavuje pomocí CANopen objektu - řídící slovo (6040_h) pracovní režimy pohonu. Kontrolovány jsou dalším CANopen objektem - stavovým slovem (6041_h). Je zde i možnost monitorovacích systémových funkcí některých veličin (teplota nebo proud pohonu). Změna stavu je schématicky znázorněna na následujícím obrázku.



Obr. 24 Změna operačních stavů [17]

7.2. Stavy pohonů

Výčet možných stavů a přechodů mezi nimi je znázorněn vývojovým diagramem na Obr. 25. Přechody mezi stavy (označeny kroužkovanými čísly), jsou dále uvedeny v tabulce objektu controlword (6040_h) na následující straně.



Obr.25 Diagram stavů a přechodů pohonu IclA [5]

Výkonový stupeň se skládá z výkonových tyristorů, které se průchodem proudu poměrně hodně zahřívají. Proto je tento stupeň vypnut okamžitě jak se motory zastaví (toto vypnutí je automatické), zapnut je opět až při příchodu požadavku na nové nastavení. To bylo důvodem zavedení koncepce, kde se dělí na oblasti kdy je zapnut výkonový stupeň pohonu a kdy ne.

Přepínání stavů

Defaultní nastavení pohonu po zapnutí je na stavu 3 (zapnutí nepovoleno). Přechody 0 a1 jsou pak spouštěny automaticky a stejně jako přechod 14 (při inicializaci chyby) ihned vypínají výkonový stupeň. Všechny ostatní změny je nutné inicializovat příkazem pomocí objektu *controlword* (6040_h).

Pro nejrychlejší možný přístup ke změně stavu jsou data *controwordu* přenášena jako součást datové oblasti objektů R_PDO1 a R_PDO2. Tento princip je vysvětlen v kapitole 6.2. Vynechané bity (-) nejsou podstatné pro jednotlivé změny stavu. Význam bitů 4 až 6 bude ještě dále rozebrán.

Tabulka objektu kontrolword (6040_h):

Příkaz	**Přechod do stavu**	**Bit 7** - Reset chyby	**Bit 7** - Povolení operace	**Bit 2** - Stop	**Bit 1** - Vypnutí napájení	**Bit 0** - Zapnutí
2,6,8 - Vypni	4 - Připraven pro zapnutí	-	-	1	1	0
3 - Zapni	5 - Zapnuto	-	-	1	1	1
7,9,10,12 – Vypni napájení	3 - Zapnutí nepovoleno	-	-	-	0	-
7, 10 - Stop 11 – Stop	3 -Zapnutí nepovoleno 7 - stop	-	-	0	1	-
5 - Nepovolení operace	5 - Zapnuto	-	0	1	1	1
4,16 - Povolení operace	6 - operace povoleny	-	1	1	1	1
15 - Reset chyby	3 - Zapnuté nepovoleno	0 -> 1	-	-	-	-

Monitorování stavů

Operační stavy jsou obsaženy sedmi byty objektu CANopen *statusword* (6041_h). Aktuální stav je posílán pohonem jako odpověď na příkaz který obdržel od řídícího slova (6040_h), jako obsah prvních dvou bytů T_PDO1 a T_PDO2, nebo prostřednictvím objektu *statusword* (6041_h).

Tabulka objektu statuswordu (6041_h):

Stav	**Bit 6** - Povolení zapnutí	**Bit** - Stop .	**Bit 3** - Chyba .	**Bit 2** - Povolení operace	**Bit 1** – Zapnutí .	**Bit 0** - Připraven na zapnutí
2 - Nepřipraven na zapnutí	0	-	0	0	0	0
3 - Zapnutí nepovoleno	1	-	0	0	0	0
4 - Připraven na zapnutí	0	1	0	0	0	1
5 - Zapnuto	0	1	0	0	1	1
6 - Operace povoleny	0	1	0	1	1	1
7 - Stop	0	0	0	1	1	1
9 - Chyba	0	-	1	1	1	1

Pozn.: Řízení robotu bude probíhat v polohovacím režimu, který je popsán dále.

7.3. Volba operačního režimu

Jak již bylo uvedeno v kapitole 7., pohony IclA mohou pracovat ve několika různých pracovních režimech. Automaticky je po zapnutí zvolen manuální režim, přepnutí na jiný se děje pomocí CANopen objektu *modes of operation* (6060_h) a je možný jen pomocí připojeného rozhraní CAN. Současně však v závislosti na zvoleném pracovním režimu musí být pohon přepnut do odpovídajícího operačního stavu (obr.25).

7.4. Polohovací režim

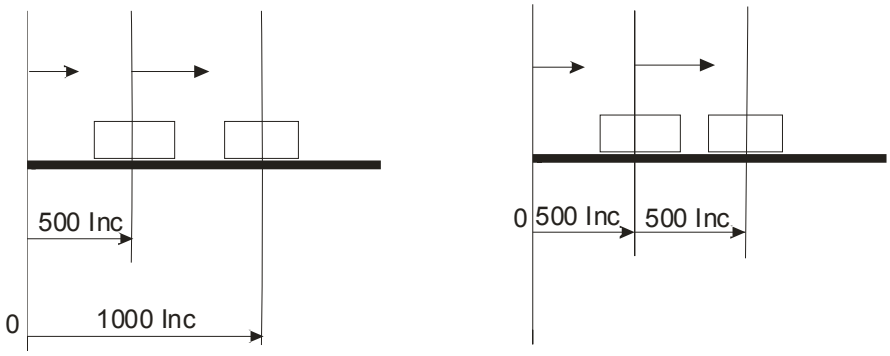
Pro použití tohoto pracovního režimu pohonu je již nutné připojení sběrnice. V tomto režimu může být pohon přestaven zadáním cílové pozice robotu společně s hodnotami rychlosti, akcelerace a deakcelerace vykonávaného pohybu a děje se tak o předem určený počet otáček hřídele (respektive rotoru pohonu). Stejně tak je možné pomocí sběrnice CAN získávat zpět od pohonu aktuální informace (poloha, akcelerace, atd..)

7.4.1. Polohování absolutní a relativní

Hodnoty pro cílovou pozici mohou být zadávány jako absolutní nebo relativní přírůstky (Inc). Jedinou podmínkou je že musí ležet uvnitř pracovní referenční oblasti (touto problematikou se zabývá kapitola 7.5).

Pro režim absolutního polohování platí že je zvolen *machina zero point* (nulový strojový bod), od kterého se všechny přírůstky odměřují. V případě relativního polohování je další cílová pozice odměřována od té minulé tj. od aktuální polohy.

Aktuální pozice je pohonem posílána jako čtyř bytová hodnota *statuswordu* obsažená v objektu T_PDO2. Případně je možno si tuto hodnotu přímo vyžádat objektem 6064_h (kapitola 6.1.2, příklad na straně 38).



Obr. 26 Příklad absolutního a relativního polohování [17]

Pozn.: Programové bloky vytvořené jako demonstrační pohyby robotu využívají absolutního polohování.

7.4.2. Start polohování

Aby po zadání pohony vykonali daný příkaz musí být nastaven operační stav 6 (operace povoleny). Také je nutné zadat novou cílovou pozici pomocí objektu SDO $607A_h$, nebo přes T_PDO2 (byty 2,3,4,5). Proces přenastavování dle zadaných kritérií se spouští se čtvrtým bitem v objektu *controlword ,*v okamžiku kdy se hodnota bitu změní na „1“, oznámí pohon povolení pohybový příkazu. Pohon potvrdí přijetí bitem dvanáct, jehož hodnota se musí rovnat „1“ (potvrzení požadované hodnoty). Poté může být bit čtyři restartován a zadává se nová pozice, pokud nebyl ještě pohybový příkaz vykonán, pohon vygeneruje chybové hlášení. Dosažení cílové pozice je reprezentováno „1“ v hodnotě bitu deset ve *statuswoldu* (cíl dosažen). Pohon o tom informuje pomocí objektu T_PDO1.

Tabulka hodnot pro start polohování

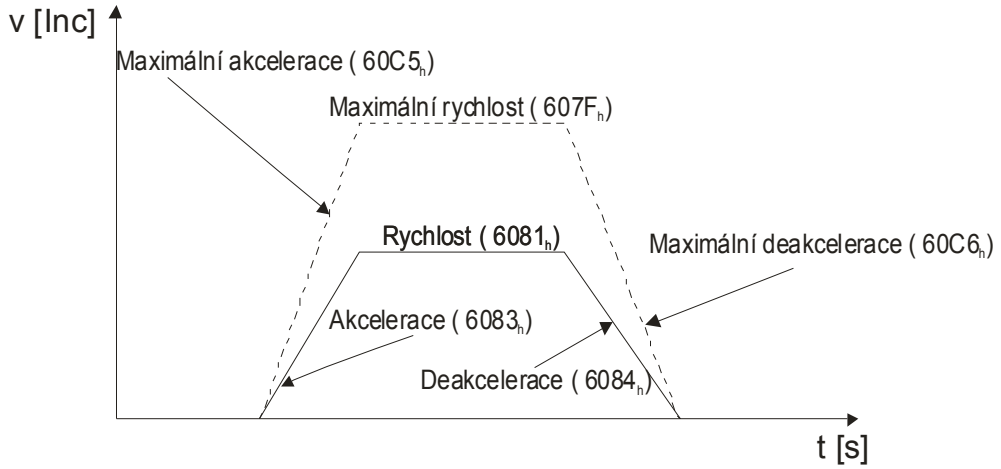
Objekt / příkaz	**Význam**
Controlword (6064_h)	Řídící slovo
Bit 4 - Nová požadovaná hodnota	0->1 - start poohování
Bit 6 - Absolutní/relativní polohování	0 - absolutní polohování 1 - relativní polohování
Bit 8 - Zastavení	0 - vykonávání procesu 0->1 - přerušení procesu
Statusworld (6041_h)	Stavové slovo
Bit 10 - Cílová pozice	1->0 - nová pozice dosažena 0->1 - žádaná pozice dosažena
Bit 12 - Potvrzení žádané hodnoty	0 - nová pozice může být přijata 1 - Nová pozice je přijata

Pozn.: Přerušení aktuálně vykonávaného příkazu je použit v hlavním řídícím programu (blok stop).

7.4.3. Nastavení

Pohybový profil

Maximální hodnoty pro přestavování pohonů, které mohou být nastaveny, jsou trvale uloženy v paměti pohonu a nelze je měnit. Pro parametry vykonávaného pohybu z výchozí do cílové pozice lze definovat (krom těchto dvou bodů) také tzv. pohybové rampy. Jsou to rampa finální rychlosti, rampa akcelerační a rampa neakcelerační. Pokud nastavení rampy překračuje maximální meze uloženy v paměti pohonu (je možno zjistit pomocí objektů maximálních hodnot), pohon automaticky generuje chybové hlášení, které má za následek automatické zastavení pohonu a tím znehybnění celého systému. Chybu je nutné dále ošetřit nastavením patřičných mezí v referenčním režimu (viz. Následující kapitola).



Obr. 27 Rampy finální rychlosti, akcelerace a deakcelerace [17]

7.5. Referenční režim pohonu

Třetím možným pracovním režimem IclA pohonů je referencování, tj. nastavování bezpečnostních a pracovních parametrů, ve kterých může pohon pracovat v ostatních režimech. V případě zadání parametrů pro přesun mimo danou pracovní oblast, pohyb ani nezačne a pohon vygeneruje chybové hlášení. Údaje o poloze zůstávají v paměti pohonu (společně s informací o komutačním stavu a absolutní hodnotě polohy rotoru, viz. kapitola 3.1.2) i po vypnutí napájení.

Tyto meze jsou přednastaveny na maximální hodnotu pro provozování integrovaných regulačních pohonů IclA za běžných podmínek a pro úlohu řízení mobilního robotu není nutné je měnit. Naopak je žádoucí sledovat aby při provozu nebyly tyto meze překročeny, což by mělo za následek zastavení robotu. Řídící algoritmus proto obsahuje rutinu (blok iniciál), která před startem práce s integrovanými IclA pohony zajistí nulování polohových čítačů.

Příklad nulování polohového čítače:

COB-ID	**Data**	**Význam**
602_h	2F 60 60 00 06	přepnutí do naváděcího režimu
582_h	60 60 60 60 xx	odpověď - přepnuto
602_h	2F 98 60 00 FF	výběr typu referencování
582_h	60 98 60 00 xx	odpověď - potvrzen výběr
602_h	23 0B 20 00 00 00 00 00	nastavení aktuální pozice na "0"
582_h	60 0B 20 00 xx xx xx xx	odpověď - pozice nastavena
602_h	2B 40 60 1F 00	aktivace nastavení
582_h	60 40 60 00 xx xx	odpověď - pozice aktivována

8. Čidlo - snímání polohy

Každé pohybující se zařízení potřebuje ke své správné funkci znát svoji polohu v prostoru kterým se pohybuje. K tomu slouží čidla a snímače polohy. Jejich podstata je vždy stejná, převést měřenou neelektrickou veličinu na elektrický signál (v případě průmyslových zařízení většinou unifikovaný), který může být dále zpracován. Různé jsou metody snímání jednotlivých čidel, mohou být s mechanický snímáním (dotyk čidla nebo části čidla s měřeným prostředím), nebo je jejich funkce založena na jiných principech a potom se nemusí měřeného prostředí dotýkat (magnetické, indukční, nebo termoelektrické snímání atd..)

V této části se budeme zabývat určením polohy mobilního robotu. Jakýkoli styk čidla, nebo jeho části s dráhou po které se robot pohybuje by mohl znamenat problém (v důsledku nerovností by mohlo dojít k zachycení měřící části a následného poškození) a tím i problémům s orientací. Proto je dobré volit bezkontaktní způsob měření. Pro tyto účely bylo vybráno vyhodnocování pomocí snímací části optické počítačové myši.

8.1. Popis principu čidla

Pomocí sériového rozhraní RS 232 je k PLC připojeno optické čidlo podvozku sledující relativní polohu robotu. Využit je převodník PS/2-RS232 [Matoušek, R. a kol., Technická zpráva 2008/A VZ MSM 0021630529 – Inteligentní systémy v automatizaci, Interní dokument VZ, VUT v Brně, 2008]. V případě že dojde k prokluzu kol, vlivem prudkého zrychlení, nebo vlivem nedostatečně adhezního povrchu, zajistí čidlo zjištění relativní hodnoty dráhy, kterou robot urazil.

Naměřené hodnoty jsou po sériové lince předávány do PLC, kde dojde v komparačním cyklu ke srovnání hodnoty, která byla dána informací od pohonů a hodnoty, která je k dispozici od čidla.

Pokud se obě hodnoty v dané toleranci rovnají, robot dorazil do bodu určení a komparační cyklus je automaticky ukončen. Pokud se hodnoty liší, dojde k přepočtu nových souřadnic, které jsou předány bloku „prepoce2“ a dále je postup stejný jako u nastavování souřadnic, který je dále popsaný (postup je zřejmý z programového bloku „set“ viz kapitola 10.1.5).

Čidlo je ze zásady inkrementální (obdobně jako je to u enkoderu v IclA regulačních pohonech) a proto je nutné informace z něj také tak zpracovávat (podrobnější popis je v programovém bloku „cidlo“ – kapitola 10.2.1).

Protože se jedná o vyhodnocovací část klasické počítačové myši, je dobré uvést zde jak vlastně tahle část myši funguje.

8.1.1. Myš

Jako prototyp počítačově orientovaného psacího zařízení, který byl předveden o pět let později vznikla myš roce 1963. Myš nám umožňuje přenášet pohyb ruky po podložce

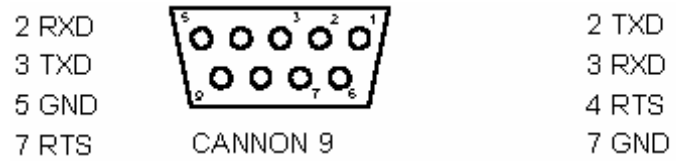
na obrazovku počítače. Dříve používané typy ve své spodní části obsahovali kuličku, ta se po podložce otáčí a svůj pohyb přenáší na válečky, které jsou mechanicky zpraženy děrovaným diskem. Jak se pohybuje kulička, otáčí válečky a je přerušováno světlo které vysílá infradioda přes tento disk na fotodiodu, vyhodnocovací elektronika takto získané informace dále zpracuje a odesílá tyto informace jako patřičné signály o pohybu myši (viz níže 8.1.2 komunikace myši). V dnes rozlišujeme dva základní typy mechanických myší: Microsoft Mouse (má dvě tlačítka) a PC Mouse (tří tlačítková myš). Každá z nich komunikuje pomocí jiného protokolu, takže jsou vzájemně nekompatibilní.

Kromě klasických mechanických existuje druhá skupina, kterou tvoří optická myš. Je to typ počítačové myši, která nevyužívá mechanického snímání. Tyto myši mají na své spodní části malou kameru CCD a osvětlovací diodu. Tato problematika je blíže rozvedena v kapitole 8.3 (strana 56). Detektor snímá odraz světla při pohybu myší po podložce a interpretuje jej jako pohyb kurzoru (v našem případě se nejde o pohyb kurzoru ale pouze jsou čteny souřadnice).

8.1.2. Komunikace myši

Součástí zadání byla možná využití RS 232 k připojení čidel snímajících relativní polohu robotu. Jako zajímavá a perspektivní se jevila možnost snímání pomocí optické počítačové myši, která ovšem byla vyráběna pouze pro jiná komunikační rozhraní (PS\2, USB). Ta ovšem neumožňuje žádný známý programovatelný automat, signály senzoru jsou na PLC přivedeny přes rozhraní RS 232 což bylo důvodem zavedení převodníku PS/2 a RS 232 postaveného na bázi dvou jednodnočipů ATMEGA a MAX 232 (příloha 1).

Dále je zde popsána jen komunikace pomocí RS232. Je realizováno pomocí standardního sériového asynchronního rozhraní RS232. Používají konektory typu CANNON 9. Rozmístění jednotlivých signálů je na následujícím obrázku.



Obr. 28 Rozmístění signálů podle jednotlivých pinů [20]

Z obrázku je zřejmé, že myš je připojena pomocí tří signálových vodičů a uzemnění. Signály RXD (*Receive Data*) a TXD (*Transmit Data*) slouží k přenosu dat. Signál RTS slouží k inicializaci myši. Je-li signál RTS (*Request to Send*) v logická „1“ (pro RS232 -12V) je myš inicializována a pracovat začne až po změně této hodnoty na logickou „0“ (tento signál nebude pro tuto práci nutný). Pro komunikaci jsou nutné jen tři signály RXD, TXD a GND -společná zem. Data jsou vysílána standardně ve formátu 1 start bit, 8 bitů dat 1 stop bit.

Jsou dva základní komunikační protokoly pro sériovou komunikaci a jeden pro PS/2:

1) Protokol dvou tlačítkové myši -Microsoft Mouse

Signál, který vysálá myš s protokolem *Microsoft Mouse* při každé změně polohy, nebo stisku tlačítek je na obr. 29. První byte obsahuje informaci o stavu tlačítek a směru pohybu. Následující dva obsahují velikost odchylky v ose x a v ose y. Napětí při tomto komunikačním protokolu je 12 V.

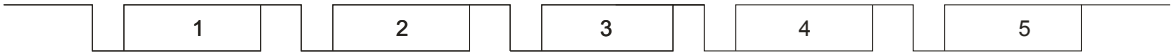
Obr. 29 Blok dat dvou tlačítkové myši [19]

Tabulka významu jednotlivých bytů dvou tlačítkové myši

Byte	Bit	Význam
1	0.-1.	směr X (00 ... doprava, 11 ... doleva)
	2.-3	směr Y (00 ... doprava, 11 ... doleva)
	4.	Pravé tlačítko (0 ... OFF, 1 ... ON)
	5.	levé tlačítko (0 ... OFF, 1 ... ON)
	6.-7.	vždy 11
2	0.-5.	směr X velikost změny (-32..31)
	6.	vždy 0
	7.	vždy 1
3	0.-5.	směr Y velikost změny (-32..31)
	6.	vždy 0
	7.	vždy 1

2) Protokol tří tlačítkové myši - Mouse System Mouse

Signál vysílaný myší s protokolem *Mouse System Mouse* odpovídá pěti datovým blokům a je znázorněn na Obr. 30. První byte obsahuje informaci o stavu tlačítek. Další dva obsahují velikost odchylky v ose x a v ose y. Byty 4 a 5 obsahují rychlost změny souřadnic v ose x a v ose y. Stejně jako v předchozím případě činí napětí 12V.



Obr. 30 Blok dat tří tlačítkové myši [19]

Tabulka významu jednotlivých bytů dvou tlačítkové myši

Byte	Bit	Význam
1	0.	pravé tlačítko (1 ... OFF, 0 ... ON)
	1.	střední tlačítko (0 ... OFF, 1 ... ON)
	2.	levé tlačítko (0 ... OFF, 1 ... ON)
	3.-6.	vždy 0
	7.	vždy 1
2	0.-7.	změna ve směru X (-128..127)
3	0.-7.	změna ve směru Y (-128..127)
4	0.-7.	rychlost změny ve směru X (-128..127)
5	0.-7.	rychlost změny ve směru Y (-128..127)

3) PS/2 protokol

Personal System/2

Sběrnice PS/2 (*IBM)* se prvně objevila koncem osmdesátých let a i dnes je stále používána pro komunikaci s klávesnicí a myší. Pro myš i klávesnici se původně používají konektory mini-DIN. Používají se dva typy konektorů., pěti pinový a šesti pinový. Opět jsou ke komunikaci nutné pouze tři signály obsahují GND, clk (hodiny), data. V případě protokolu *Personal System/2* je napájení + 5V.

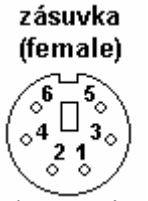
Zapojení konektorů, význam signálů:

5-pinová DIN zásuvka:

1 – Clk
2 – Data
3 – Nc
4 – GND
5 - +5V

Obr.31 Din PS/2 5 pinů [19]

6-pinová mini-DIN zásuvka pro PS/2 myš:



1 – Data
2 – Nc
3 - GND
4 - +5V
5 - Clk
6 - Nc

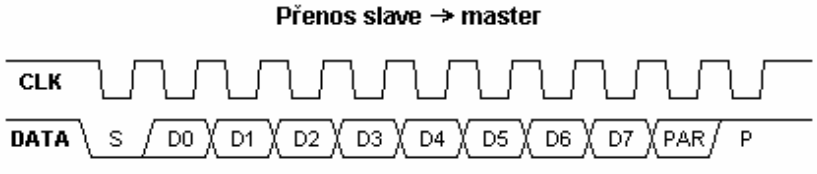
Obr.32 Din PS/2 6 pinů [19]

Přenos pomocí PS/2 protokolu

Jde o synchronní obousměrný protokol master-slave, který umožňuje připojit jen jediné slave zařízení (v našem případě myš). Důležitou vlastností je, že hodiny generuje vždy slave. Master (PC nebo v našem případě převodník RS232 – PS/2) je ovšem nadřazen a pomocí Clk signálu povoluje, zakazuje nebo přerušuje přenos dat. Oba signály (Data i hodiny) jsou řešeny s otevřeným kolektorem.

Přenos "slave - master"

Slave může samovolně zahájit přenos jednoho bytu pokud jsou obě linky (CLK i Data v úrovni high po dobu nejméně 50us. Datový rámec obsahuje celkem 11 bitů. Přenos je zahájen start bitem s úrovní low, následuje 8 bitů dat řazených vzestupně co do priority. Dále je zde narozdíl od sériové komunikace nutný i paritní bit a stop bit s úrovní high. Master čte stav datové linky nikoliv se sestupnou hranou hodin, ale přesně uprostřed low úrovně a hodin. To odpovídá přibližně 15-25µs po sestupné hraně hodin.



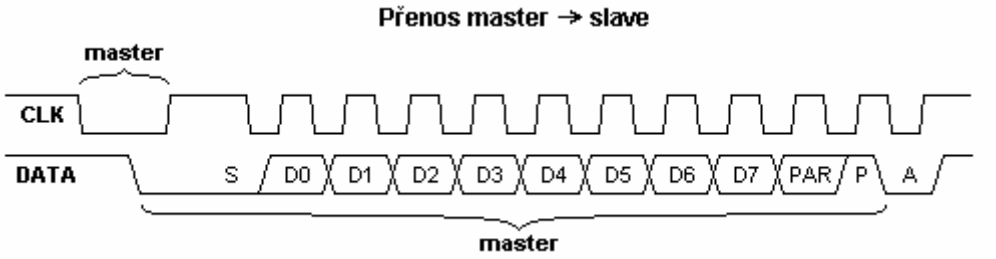
Obr. 33 Signály Clk a Data pro přenos slave – master [19]

Přenos "master - slave"

Přenos bytu do zařízení se od opačného směru liší. Master musí vyslat požadavek na přenos dat (*Request To Send*). Master nejdřív překlopí hodiny do low úrovně na minimální

dobu 100μs pro blokaci případného přenosu opačným směrem. Následně master překlopí do úrovně low i data (což odpovídá start bitu) a čeká alespoň 5μs, pak uvolní hodiny. Slave reaguje maximálně do 10ms tak, že začne generovat hodiny. Polarita hodin je při přenosu do zařízení přesně opačná. Slave vzorkuje data přesně uprostřed high úrovně hodin, master mění hodnotu dat během low úrovně hodin (nejlépe přesně uprostřed).

Po přenosu všech datových bitů, liché parity a stop bitu následuje ještě potvrzovací bit A generovaný slave zařízením a to ještě během high úrovně hodin, kde má slave vzorkovat stop bit. Je vhodné po detekování poslední vzestupné hrany hodin počkat min. 100μs než dojde k dalšímu přenosu. Toto zpoždění je zde z důvodu stability přenosu.



Obr. 34 Signály CLK a Data pro přenos master – slave [19]

Pro účely mobilního robotu je využita tří tlačítková optická myš, původně s PS/2 výstupem, dále upraveno na signály pro RS 232 (ATmega převodník).

8.2. Sériová komunikace

Recommended Standard číslo 232

Tento způsob komunikace, který vznikl v roce 1969. U většině počítačů ho nalezneme v podobě konektoru CANON 9 (to bohužel pro většinu notebooků neplatí, musíme použít USB-RS 232 redukcí). RS 232 je snad nejjednodušší způsob komunikace s většinou externích zařízení.

Pro případ užití v systému mobilního robotu postačí nejpoužívanější varianta, kdy komunikační kabel má pouze tři žíly: jednu pro společnou zem, jednu pro příjem a jednu pro vysílání Informace je kódována pomocí rozdílných napětí mezi společnou zemí a příslušným pinem pro příjem (signál RXD), nebo pro vysílání (signál TXD).

RS 232 komunikuje pomocí rámců (*frames*). Pokud neprobíhá žádná komunikace, je linka v klidovém stavu, pro který se používá kladné napětí. Každý rámec začíná start bitem, což je změna na záporné napětí na dobu danou rychlostí komunikace (např. pro 9600 bps, je to 1/9600s, tj. cca 104μs). Následují datové bity, kdy logická „1“ odpovídá zápornému napětí a logická „0“ napětí kladnému. Vysílání začíná od nejméně důležitého bitu. Celý rámec je zakončen stop bitem, kdy je linka opět v klidovém (má znovu kladném napětí). Po stop bitu může následovat pauza, nebo hned start bit.

Rámce - *Frame*

Rámce mohou po sobě hned následovat, takže pokud používáme přenosovou rychlost 9600 bps, tak za 1s můžeme poslat maximálně 9600/10=960 bajtů (číslo 10 odpovídá jednomu start bitu, 8 datovým bitům a jednomu stop bitu).

Existuje mnoho konfiguraci RS232, z nichž asi nejpoužívanější je:
- Start bit
- 8 datových bitů
- žádný paritní bit
- jeden stop bit

8.3. Optické snímání

Principem optické myši je malá kamera (CCD či CMOS prvek s maticí o velikosti několik desítek pixelů), která snímá obraz v podobném ve velmi malém rozlišení. Rychlost jejího snímání je ovšem velmi vysoká, od jednoho tisíce snímků za sekundu se můžete dostat (u nejkvalitnějších myší) až k šesti tisícům vyhodnocených obrazů.

Pokud myší hýbnete, obraz se posune, čímž je možno zjistit, jakým směrem se pohybujete, a jak rychle. K vyhodnocení pohybu se musí v myši nalézat relativně výkonný procesor, který u mechanických myší nebyl nutný. Aby tato kamera něco zachytila, musí být plocha viditelná. K osvětlení plochy slouží jedna LED dioda, jejíž světlo je namířeno pomocí hranolu nebo zrcátka na podložku. Důvod proč je osvětlení červené je krom množství možných technologických příčin, především v ceně. Princip optické myši, je velmi jednoduchý, ale také složitější na realizaci, z důvodu použitých součástí (optiky, kamery a dražšího mikroprocesoru).

U prvních prototypů optických myší byla nutná speciální podložku pod myš, která byla vysoce kontrastní a měla na sobě vykreslenou černou mřížku pro snadnější rozeznávání pohybu. Dnes si optická myš vystačí takřka s jakýmkoliv povrchem. Jediné, kde nelze myš provozovat, je sklo, zrcadlo, případně jiný povrch s velmi velkou odrazivostí nebo průhledností.

8.4. Kalibrace čidla

Hodnoty které jsou z čidla čteny nekorespondují s hodnotami, které jsou k dispozici od enkoderů integrovaných pohonů IclA. Měřením bylo zjištěno, že průměrná hodnota posuvu čidla o 10 cm odpovídá 850 přírůstkům [Inc].

Posuv robotu o jednu otáčku kola (v přímém směru) odpovídá 213 [Inc] = 37,7 cm.
Posuvu robotu o 1 cm = 6 [Inc]
Posuvu čidla o 1 cm = 85 přírůstkům.
Srovnáním obou hodnot dostaneme kalibrační konstantu „ K“, kterou určíme jako:

$$K = \frac{\text{posuvu robotu}}{\text{posuvu čidla}} = \frac{85}{6} = 14{,}1666$$

Pokud touto konstantou vynásobíme počet přírůstků od čidla dostaneme číslo, které odpovídá počtu přírůstků pohonu v [Inc].

9. Tříosý systém řízení

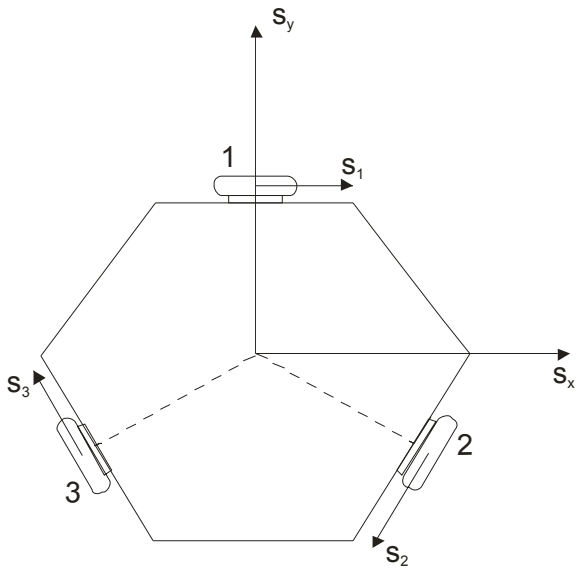
Předně je nutné předeslat, že i když robot využívá tříosého systému, zadávání souřadnic pro jeho polohový algoritmus se děje v pravoúhlých souřadnicích. Aby bylo možné takto řízení provozovat je nutné zadané souřadnice cílových pozic během provozu transformovat z kartézského systému zobrazení do tříosého systému robotu a naopak. To se děje v řídícím programu robotu pomocí přepočtů s využitím goniometrických funkcí (bloky prepocet, prepoce2).

9.1. Přepočtové vztahy z kartézského systému do tříosého systému

Po zadání všech parametrů pohybu (cílová pozice, rychlost atd.), které je realizováno v systému pravoúhlých souřadnic (s_x, s_y) musí dojít k přepočtu jednotlivých veličin do systému mobilního robotu, tj. určení těchto pohybových parametrů pro jednotlivá kola.

Aby bylo možné využít přepočtových vztahů, je nutné si zvolit souřadný systém pro znázornění vektorů všech zadávaných parametrů realizovaného pohybu jednotlivých kol mobilního robotu.

Příklad takového znázornění je níže, stejné zákonitosti platí pro všechny pohybové parametry.



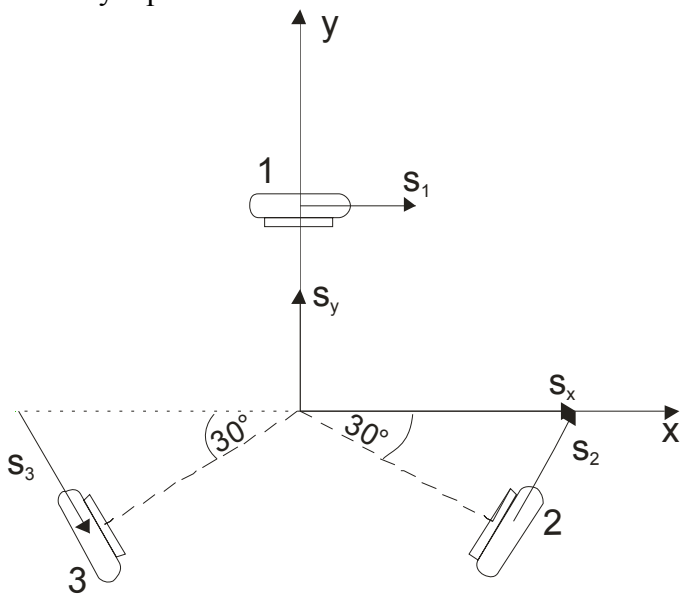
1, 2, 3 - kolo1, kolo2, kolo3
s_1, s_2, s_3 – směrové vektory dráhy jednotlivých kol v tříosém systému
s_x, s_y - směrové vektory dráhy v kartézském systému

Obr.35 Systém souřadnic mobilního robotu

Pokud pro jednoduchost pochopení uvažujeme rychlost pouze v jedné ose (v tomto případě v ose x) dostáváme několik závěrů :

- při takovém pohybu postačí pokud pohony otáčí pouze dvěma koly a výslednicí těchto kol je výsledný vektor pohybu
- stejné zákonitosti budou platit i pro osu y za podmínky, že se systém robotu otočí tak aby některé z kol (2, nebo 3) mělo svůj vektor pohybu rovnoběžný s osou y.

Pro pohyb ve směru osy x platí:



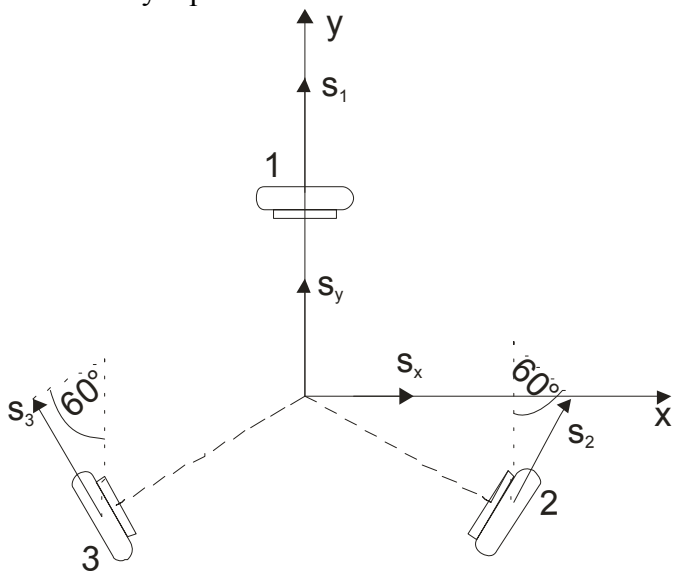
Obr. 36 Směrové vektory pro pohyb v ose x

kolo1: směrový vektor kola je rovnoběžný se směrem pohybu, tedy výsledný pohyb kola 1 je již výsledná hodnota (platí i pro rychlost, akceleraci a deakceleeraci) $\underline{s_{1x} = s_x}$

kolo 2: $s_{2x} = -s_x \sin 30° = -s_x \frac{1}{2}$

kolo 3: $s_{3x} = -s_x \sin 30° = -s_x \frac{1}{2}$

Pro pohyb ve směru osy x platí:



Obr. 37 Směrové vektory pro pohyb v ose y

kolo1:směrový vektor kola je rovnoběžný se směrem pohybu, ale obvodová rychlost tohoto kola se rovný nule (platí i pro rychlost, akceleraci a deakceleeraci) $\underline{s_{1y} = 0}$

kolo 2: $s_{2y} = -s_y \sin 60° = -s_y \sqrt{\frac{3}{2}}$

kolo 3: $s_{3y} = s_y \sin 60° = s_y \sqrt{\frac{3}{2}}$

Pro pohyb v rovině (tedy směru x i y) platí :

$s_1 = s_{1x} + s_{1y} \quad \rightarrow \quad s_x + 0 = s_x \quad \rightarrow \quad s_1 = s_x$

$s_2 = s_{2x} + s_{2y} \quad \rightarrow (-s_x \frac{1}{2}) + (-s_y \sqrt{\frac{3}{2}}) \quad \rightarrow \quad s_2 = -s_x \frac{1}{2} - s_y \sqrt{\frac{3}{2}}$

$s_3 = s_{3x} + s_{3y} \quad \rightarrow \quad (-s_x \frac{1}{2}) + s_y \sqrt{\frac{3}{2}} \quad \rightarrow \quad s_3 = -s_x \frac{1}{2} + s_y \sqrt{\frac{3}{2}}$

Jak již bylo několikrát zmíněno stejné principy platí pro všechny pohybové parametry jednotlivých kol, tudíž platí :

$v_1 = v_x \qquad a_1 = a_x$

$v_2 = -v_x * \frac{1}{2} - v_y \sqrt{\frac{3}{2}} \qquad a_2 = -a_x \frac{1}{2} - a_y \sqrt{\frac{3}{2}}$

$v_3 = -v_x \frac{1}{2} + v_y \sqrt{\frac{3}{2}} \qquad a_3 = -a_x \frac{1}{2} + a_y \sqrt{\frac{3}{2}}$

pozn.:Deakcelerace – brždění, je záporná akcelerace

9.2. Přepočtové vztahy z tříosého systému do kartézského systému

Pro transformaci zpět do kartézského systému musíme vyjít z předpokladu že :

$s_x = f(s_1, s_2, s_3)$
$s_y = f(s_1, s_2, s_3)$

Pokud vyjdeme z výše uvedeného předpokladu a aplikujeme ho na obr. 37 dostáváme:

$s_1 = s_x \rightarrow \quad s_x = s_1$

Této vlastnosti využijeme a dosadíme do rovnice $s_2 = -s_x \frac{1}{2} - s_y \sqrt{\frac{3}{2}}$ a dostaneme :

$$s_2 = -s_1 \frac{1}{2} - s_y \sqrt{\frac{3}{2}}$$

Následně řešením této rovnice se dostáváme :

$$s_2 = -s_1 \frac{1}{2} - s_y \sqrt{\frac{3}{2}} \quad \rightarrow \quad s_y = -s_1 \sqrt{\frac{3}{3}} - s_2 \frac{2\sqrt{3}}{3}$$

Potom platí :

$$s_1 = s_x$$
$$s_y = -s_1 \sqrt{\frac{3}{3}} - s_2 \frac{2\sqrt{3}}{3}$$
$$s_y = -s_1 \sqrt{\frac{3}{3}} + s_3 \frac{2\sqrt{3}}{3}$$

A také:

$$v_x = v_1 \qquad a_x = a_1$$
$$v_y = -v_1 \sqrt{\frac{3}{3}} - v_2 \frac{2\sqrt{3}}{3} \qquad a_y = -a_1 \sqrt{\frac{3}{3}} - a_2 \frac{2\sqrt{3}}{3}$$
$$v_y = -v_1 \sqrt{\frac{3}{3}} + v_3 \frac{2\sqrt{3}}{3} \qquad a_y = -a_1 \sqrt{\frac{3}{3}} + a_3 \frac{2\sqrt{3}}{3}$$

Pozn.: Pro zpětné promítnutí do osy y jsou oba vztahy nominálně totožné, pouze se pouze jednou se vychází z kola 2 a podruhé z kola 3.

10. Programové algoritmy mobilního robotu

Součástí zadaní bylo vytvořit řídící algoritmus mobilního robotu a sestavit demonstrační pohybový program. Kapitola deset se bude věnovat právě této problematice. Celý program mobilního robotu se skládá z jednotlivých tasků (bloků programů – viz obr. 13). Tyto bloky se potom dají rozdělit do tří základních skupin, tedy na tasky, které se mají na starost samotnou realizaci pohybu robotu (read, write, iniciál, set, atd..), tasky monitorovací (monitor, cidlo) a bloky s předdefinovanými pohyby (čtverec, úsečka atd.).

Jak již bylo řečeno celý program je psán v prostředí Automation studia společnosti B&R programovacím jazykem ST – *Structured Text,* který je součástí mezinárodní normy popisných jazyku pro automatizaci - IEC 1131. Každý task ma dvě nezávislé části, nazvané *init program* (tj. inicializační část, nastavují, nebo povolují se zde komunikační a funkční bloky následně užité v programu, nebo také jednotlivé vstupy a výstupy automatu, atd.). Druhou částí je *cyclic program*, kde se píše už samotný program. Tyto tasky jsou pak zařazeny do tzv. taskové třídy, podle času na průchod jedním cyklem celého programu. Celý program mobilního robotu je zařazen do stejné třídy s časem průchodu 10 ms. Každý task se dělí na inicializační a cyklickou část.

Popisu jednotlivých programových bloků se budeme věnovat podrobněji, v posloupnosti dle výše uvedeného rozdělení.

Jako základ pro tvorbu programových bloků byla použita práce Řízení pohybu tříkolového mobilního robota autora Bartoše Stanislava z roku 2005.

10.1. Programové bloky - pohybové

Této části odpovídá celkem devět tasků nutných k inicializaci, zadávání parametrů, komunikaci, přepočtů z obou užitých systémů zobracení v rovině a samozřejmě řízení.

10.1.1. Read

Blok programu „read“ realizuje povolení komunikace a nastavení přenosových parametrů sběrnici CAN. Jak již vypovídá samotný název, tento blok krom výše uvedených skutečností také čte data posílaná integrovanými pohony po sběrnici.

Čtení sběrnice probíhá neustále, bez ohledu na vykonávanou část programu. Jednotlivé byte objektu CANopen COB – id jsou načítána do proměnné „cobidread“. Načtená data jsou uloženy v proměnné „Data“ (8 bytové pole typu USINT), odkud je možné je dále použít a zpracovávat.

Struktura bloku „read“
Init část

obsahuje konfiguraci a povolení komunikace pomocí funkčního bloku CANopen (z knihovny can_lib) nastavení přenosové rychlosti, chybových hlášení a maxima přenosových objektů.

Ukázka části kódu:

```
CANopen_01(enable:=1,baud_rate:=2,cob_anz:=5,error_adr:=ADR(error)
,device:=ADR('IF3'), info:=0);
(*povoleni funkcniho bloku pro CANopen,nastaveni prenosové
rychlosti na 20Kbit ,nastaveni maxima komunikacnich objektu,adresa
chybovych hlaseni,definice zarizeni, info*)
```

Cyklic část

Realizace čtení probíhá pomocí funkčního bloku CANopen (opět z knihovny can_lib), který nastavuje je požadované COB-ID, a načtená data ukládá do proměnné 'data'.

Ukázka části kódu:

```
CANread_01(enable:=1,us_ident:=CANopen_01.us_ident,
can_id:=cobidread, data_adr:=ADR(data));
(*povoleni funkcniho bloku CANread, nastaveni pozadovaneho COB-ID,
ulozeni dat do promenne data*)
```

10.1.2. Write

Obdobně jako v předchozím případě, i v blok „write" obsahuje nastavení komunikačních parametrů CAN rozhraní. Blok „write" realizuje zápis instrukcí pro jednotlivé akční členy po sběrnici CAN, pomocí objektů profilu CANopen.

Samotný zápis potom probíhá pomocí proměnné „Dataw" (8 bytové pole typu USINT), ale na rozdíl od předchozího případu nemůže probíhat neustále (riziko kolizního stavu na sběrnici). Proto je zápis na sběrnici řízen prostřednictvím proměnné „writeen" (typu BOOL) a sní spojené proměnné „writeok" (také typu BOOL), která obsahuje potvrzení. Po úspěšně provedeném zápisu je „writeen" opět nastavena na svou původní hodnotu („0").

Init část

Obsahuje konfiguraci komunikace pomocí funkčního bloku CANopen (opět z knihovny can_lib), stanovení komunikační rychlosti, chybových hlášení, maxima přenosových objektů a inicializaci proměnných („writeen", „writeok").

Ukázka části kódu:

```
CANopen_01(enable := 1,baud_rate:=2, cob_anz := 4,
 error_adr := ADR(error), device :=ADR('IF3'), info := 0);
(*povoleni funkcniho bloku CANopen, nastaveni přenosové rychlosti
na 20KBbit, nastavenimaxima komunikacnich objektu, adresa
chybovych hlsseni, definice zarizeni, info*)
```

Cyklic část:

Povolení zápisu pomocí funkčního bloku CANwrite, realizace zápisu dat z proměnné „Dataw" .

Ukázka části kódu:

```
CANwrite_01(enable:=1,us_ident:=CANopen_01.us_ident,
can_id:=cobidwrite, data_adr:=ADR(dataw),data_lng:=dataleng);
(*cobidwrite specifikuje COB-ID, dataw – posilana data,specifikace
delky posilanych dat*)
```

10.1.3. Inicial

Tímto blokem je realizována identifikační a spouštěcí procedura pohonů. To znamená nulovaní polohového čítače (pomocí programové rutiny) a přepnutí pohonu do režimu č.2 – polohovacího (viz. kapitola 7.4 strana 48) a stavu ‚operace povoleny' (opět je zde vytvořena programová rutina, která tak jako v předchozím případě proběhne postupně pro všechny tři pohony).

Init část

Jsou zde obsaženy definice COB-ID komunikačních objektu (PDO, SDO, SYNC, atd..), nastaveni proměnných a pomocných konstant a nulování pole pro zápis i čtení dat („Dataw" a „Data").

Ukázka části kódu:

```
(*Definice COB-ID komunikacnich objektu*)
rpdo1:= 512;                        (*PDO - Process Data Objcts*)
tpdo1:= 384;
rpdo2:= 768;
tsdo:= 1536;                        (*SDO - Service Data Objects*)
rsdo:= 1408;
sync:= 128;                         (*SYNC*)
```

Cyklic část

Nastavení proměnné „cobieread" (složené z Boot-up a Node-ID na hodnotu 81_h), čekání na příchod Boot-up zprávy, po jejím příchodu se přejde na rutinu pro nulování polohového čítače, která je uzavřena pomocí CASE do dílčích podprogramů (přepnutí do naváděcího režimu + odpovědi na tuto operaci od pohonů, výběr typu referebcování + odpovědi na tuto operaci od pohonů, samotného nulování polohového čítače + odpovědi na tuto operaci od pohonů, aktivace nového nastavení po nulování polohového čítače v předchozím kroku dále odpovědi na tuto operaci od pohonů a restartu uzlu + odpovědi na tuto operaci od pohonů).

Dalším obsahuje rutinu pro přechod do stavu operace povoleny, která je stejně jako v předchozím případě opět realizovány pomocí CASE (přepnutí pohonu do operačního stavu + odpovědi na tuto operaci od pohonů, požadavek na přepnuti do stavu "připraven pro zapnuti" + odpovědi na tuto operaci od pohonů, požadavek na přepnuti do stavu "zapnuto" + odpovědi na tuto operaci od pohonů, přechod do stavu "operace povoleny" + odpovědi na tuto operaci od pohonů, přepnuti do polohovacího režimu + odpovědi na tuto operaci od pohonů, ukončení inicializační fáze pohonu+nastavení předávacích podmínek).

Pro většinu z těchto úloh je nutné prováděl výpočet COB-ID (cobidwrite := tsdo+nodeid a cobidread := rsdo+nodeid).

Ukázka části kódu:

```
      (*cekani na boot-up zpravu od pohonu s nastavenym nodeid*)
IF motorzap = 0 THEN
      (*pokud pohon neni zapnut*)
      cobidread := bootup + nodeid;
      (*vypocet COB-ID boot-up zpravy *)
      IF precetlok = 1 THEN
      (*pokud prisla zprava s COB-ID 82h*)
            initok := 2;     (*nastav initok=2*)
            motorzap := 1;   (*signalizace zapnuti pohonu*)
            kroky := 1;      (*nastaveni ridici promenne*)
      END_IF;
END_IF;
      (*rutina pro nulovani polohoveho citace*)
IF motorzap = 1 AND initok = 2 THEN
      (*provede se az po prichodu boot-up zpravy*)
      CASE kroky OF

1:    (*prepnuti do navadeciho rezimu*)
      cobidwrite := tsdo + nodeid;
      (*vypocet COB-ID pro zapis*)
      cobidread := rsdo + nodeid;
      (*vypocet COB-ID pro cteni odpovedi *)
      (*posilana data, 2f=poslan bude 1byte objektu 6060h, sub-
      index 00, data 06h*)
      dataw[0] := 16#2f; dataw[1] := 16#60; dataw[2] := 16#60;
      dataw[3] := 16#00;
      dataw[4] := 16#06; dataw[5] := 16#00; dataw[6] := 16#00;
      dataw[7] := 16#00;
      dataleng := 5;
      (*specifikace delky posilanych dat - 5byte*)
      writeen := 1;
(*povoleni zapisu, write enable = 1*)
      kroky := 11;
      (*prechod na daldi krok*)
11:   (*cekani na potvrzeni o prepnuti do navadeciho rezimu*)
      IF data[0] = 16#60  THEN
      (*staci kontrola, zda nedoslo k chybe (80h)*)
            data[0]:= 0;    data[1]:= 0;    data[2]:=0;
      (*nulovani datoveho pole, prvnich*)
            data[3]:= 0;    data[4]:= 0;    data[5]:=0;
      (*5 pouzitych byte*)
            kroky := 2;
      (*pokud bylo prepnuti potvrzeno, prechod na krok 2*)
      END_IF;
```

10.1.4. Vstupy

Blok „vstupy“ vznikl z důvodu oříapdné prezentace předdefinovaných pohybů, které jsou podrobně popsány v kapitole 10.3 programové bloky předdefinovaných pohybů. Úkol tohoto bloku spočívá v kontrole vstupů programovatelného automatu, na něž jsou pomocí spínačů přiváděny logické jedničky a kombinací jednotlivých vstupů (vždy je to kombinace některého vstupu a vstupu 8) jsou určeny jednotlivé předdefinované pohyby.

Init část:

Deklarace proměnných (tj. jednotlivých vstupů, souřadnic „x“ a „y“ pro určení cílové pozice robotu atd.)

Ukázka části kódu:

```
x := 0;
(*x-ova slozka zadane cilove pozice robota, zadana stac. stanici*)
y := 0;
(*y-ova slozka zadane cilove pozice robota, zadana stac. stanici*)
a := 0;
(*promenna pro ulozeni akcelerace zadane stacionarni stanici*)
v := 0;
(*promenna pro ulozeni rychlosti zadane stacionarni stanici*)
d := 0;
(*promenna pro ulozeni deakcelerace zadane stacionarni stanici*)
```

Cyklic část:

Popis jednotlivých kombinací vstupů dle jednotlivých vzorů zvolených pohybů, nastavení přechodu robotu zpět do původní polohy (vstupy 7 a 8).

Ukázka části kódu:

```
IF ident = 1 THEN
       (*kontrola povoleni bloku "vstupy"*)
       IF x <> xsave OR y <> ysave THEN
       (*kontrola, zda nedoslo k zadani souradnic od stacionární
      stanice*)
            rizeni := 1;     (*pokud ano, povoli se blok "rizeni"*)
            riz := 1;        (*nastaveni ridici promenne*)
       END_IF;
       IF ident = 1 AND rizeni = 0 THEN
       (*poku je povolen blok "vstupy" a neni povolen blok
      "rizeni", je povoleno cteni vstupu*)
            IF vstup1 = 1 AND vstup8 = 0 THEN
                 pohyb := 0;      (*nulovani ridici promenne*)
                 vzor := 1;
       (*povoleni vstupu do bloku "usecka"*)
```

10.1.5. Set

Blok „set“ slouží k nastavení pohybových parametrů (zvolení odpovídající rychlostní rampy pohonů – Obr. 27 strana 49, tzn. rychlosti, akcelerace a deakcelerace vykonávaného pohybu), prostřednictvím počtu otáček jednotlivých pohonů, které jsou spouštěny synchronně pomocí CANopen objektu SYNC (viz. kapitola 6.3 strana 41).

Init část

Deklarace proměnných (tj. jednotlivých pohybových parametrů - rychlosti, akcelerace a deakcelerace vykonávaného pohybu, cílové pozice robotu atd.).
Ukázka části kódu:

```
v1 := 0;
v2 := 0;          (*promenne pro zadani finalove rychlosti*)
a1 := 0;
a2 := 0;          (*promenne pro zadani akcelerace*)
d1 := 0;
d2 := 0;          (*promenne pro zadani deakcelerace*)
```

Cyklic část

Nejprve proběhne kontrola zda došlo k inicializaci pohonů a jestli je zvolený režim [č.2] polohovací režim. Další část je opět seskupena prostřednictvím CASE do dílčích podprogramů (uložení hodnot rychlostní rampy a cílové pozice jednotlivých pohonu, zápis cílové pozice prvního pohonu a nastaveni objektu controlword na start, prostřednictvím objektu CANopen synchronní PDO - RPDO2, zápis nove finálové rychlosti, prostřednictvím objektu CANopen SDO + potvrzení zápisu od pohonů, zápis hodnot akcelerace + potvrzení zápisu od pohonů, zápis hodnot deakcelerace + potvrzení zápisu od pohonů, přepnutí nastavení dalšího pohonu pomocí „Nodeid“ – (nodeid = nodeid + 1, pokud se nodeid = 4 dojde k přechodu na další krok v CASE), kontrola trajektorie robotu a případná komparace dráhy dle nutnosti – pokud se údaje od pohonů a čidla neshodují, současný start všech pohonů – prostřednictvím CANopen objektu SYNC + potvrzení od pohonů.

Ukázka části kódu

```
(*ulozeni hodnot rychlostni rampy a cilove pozice pohonu 3*)
ELSIF nodeid = 4 THEN
    poz1 := poz31;
    poz2 := poz32;
    poz3 := poz33;
    poz4 := poz34;
    v1 := rych31;
    v2 := rych32;
    a1 := akc31;
    a2 := akc32;
    d1 := deak31;
    d2 := deak32;
    faze := 2;
(*prechod na další krok*)
END_IF;
```

```
2:    (*zapis cilove pozice pohonu 1 a nastaveni objektu
      controlword na start*)
       cobidwrite := rpdo2+nodeid;
       (*RPDO2 - synchronni PDO*)
       (*dataw[0], [1] = controlword - fh = stav "operace
       povoleny", 5h = bit 4 - start a bit 6 - relat. polohovani*)
       dataw[0] := 16#5f; dataw[1] := 16#00;
       (*dataw[2], [3], [4], [5] - zapis cilove pozice*)
       dataw[2]:=poz1;dataw[3]:=poz2;dataw[4]:=poz3;dataw[5]:=poz4;
       dataw[6] := 0; dataw[7] := 0;
       dataleng := 6;
       writeen := 1;
       faze := 3;
```

10.1.6. Stop

Struktura bloku stop umožňuje zastavení robotu v jakémkoli okamžiku bez ohledu na vykonávaný pohyb (nebo příkaz), je-li zadán požadavek na zatavení. Tohoto bloku využívá také task „rizeni", v případě požadavku na zadání polohy robota ze stacionární stanice. Slouží také jako 'reset' při „zatuhnutí" – na kroku 12 v bloku „set" (čekání na potvrzení o dosažení cílové pozice). Operace které v této části programu robotu nastavují pohony zpět do stavu „operace povoleny“ jsou bez potvrzováni, z důvodu rizika zacyklení při čekání na odpověď a následného vyvolání chyby.

Init část:

Deklarace proměnných a adres jednotlivých pohonů (Node-ID).

Cyklic část:

Sledování pohybu robotu a tím povolení vstupu do bloku „stop“ se děje pomocí proměnné ‘startmer‘ (typu bool), která je nastavena do hodnoty log ‚1‘ pokud je robot v pohybu tj. pokud některý z pohonu posílá informace o své poloze (která je proměnná v čase). Další část bloku je pomocí CASE rozdělena do několika podprogramů. Jednotlivé pohony se postupně nastaví do stavu č. 7 (dle obr. 25. na straně 46), což se děje prostřednictvím CANopen objektu RPDO2 (synchronní PDO, viz. strana 39), zastavení se realizuje pro všechny společně prostřednictvím objektu SYNC (synchronizační objekty CAN open – viz strana 39). Následuje rutina pro uvedení pohonů zpět do stavu č. 6 „ perace povoleny“, což se děje pomocí RPDO1.

Ukázka části kódu:

```
1:    (* pohon 1 *)
      cobidwrite := rpdo2+nodeid1;
      (*RPDO2 - synchronni PDO*)
      (*bit 1 objektu controlword = "1" a bit 2 = "0", pohon prejde
      do stavu "7. STOP", dojde k ukonceni vykonu prikazu*)
```

```
      dataw[0] := 16#02; dataw[1] := 16#00;
      dataw[2] := 0;dataw[3] :=0 ;dataw[4] := 0;dataw[5] := 0;
      dataw[6] := 0; dataw[7] := 0;
      dataleng := 2;
      writeen := 1;
      krok := 2;
2:    (* pohon 2 *)
      cobidwrite := rpdo2+nodeid2;
      (*RPDO2 - synchronní PDO*)
      (*bit 1 objektu controlword = "1" a bit 2 = "0", pohon prejde
      do stavu "7. STOP", dojde k ukonceni vykonu prikazu*)
      dataw[0] := 16#02; dataw[1] := 16#00;
      dataw[2] := 0;dataw[3] := 0;dataw[4] := 0;dataw[5]:= 0;
      dataw[6] := 0; dataw[7] := 0;
      dataleng := 2;
      writeen := 1;
      krok := 3;

3:    (* pohon 3 *)
      cobidwrite := rpdo2 + nodeid3;
      (*RPDO2 - synchronní PDO*)
      (*bit 1 objektu controlword = "1" a bit 2 = "0", pohon prejde
      do stavu "7. STOP", dojde k ukonceni vykonu prikazu*)
      dataw[0] := 16#02; dataw[1] := 16#00;
      dataw[2] :=0 ;dataw[3] := 0;dataw[4] := 0;dataw[5] := 0;
      dataw[6] := 0; dataw[7] := 0;
      dataleng := 2;
      writeen := 1;
      krok := 4;

4:    cobidwrite := sync;
      (*COB-ID SYNC objektu = 80h*)
      data[0] := 0; data[1] := 0; data[2] := 0; data[3] := 0;
      data[4] := 0; data[5] := 0; data[6] := 0; data[7] := 0;
      dataleng := 0;
      writeen := 1;
```

10.1.7. Rizeni

Pokud přijde požadavek na zadání polohy robota ze stacionární stanice, změní se proměnné 'x' a 'y', zavolá se blok "stop", zjisti si polohu robota pomoci bloku „monitor", „cidlo“, robot se vrátí buď do počáteční nulové polohy, nebo se přesune na zadanou souřadnici 'x' a 'y'. Další možností je přesun na tuto souřadnici plynule. Způsob přechodu určuje proměnná 'prechod'.

Init část:

Nulování proměnné ' riz '

Cyklic část:

Nejdříve se povolí vstup do bloku, zadávají se souřadnice a nastaví se rampa pohybu. Cyklická část bloku „rizeni“ je realizována znovu pomocí CASE. V prvním kroku se zavolá se blok „stop“, následuje zjištění aktuální pozice robotu. Dále se volí způsob přesunu robotu (plynulý nebo zpět do počáteční pozice), přepočítají se souřadnice z pravoúhlého systému do tříosého a tyto se potom předají do bloku „set“, odkud jsou zadány jednotlivým pohonům.

Ukázka části kódu:

```
IF rizeni = 1 THEN
      (*kdyz je povolen blok "rizeni"*)
      xsave := x;
      (*hodnoty zadanych souradnic x a y se ulozi do promennych
      'xsave' a ysave'*)
      ysave := y;
      (*pomoci nich se v bloku "ident" kontroluje, zda prisly nove
      souradnice od st. stanice*)

      rychlost := 248;      (*rychlost 248Inc/s = 70 ot/min*)
      akcelerace := 1000;   (*zrychleni 1000Inc/s^2*)
      deakcelerace := 400;  (*deakcelerace 400Inc/s^2*)
      prepocet := 1;

      CASE riz OF

1:    stop := 1;
      (*pokud je  prommenna 'riz'=1, vola se podprogram "stop"*)

2:    IF aktualnipoz1 = 0 AND aktualnipoz2 = 0 THEN
            riz := 5;       (*prechod na krok 5*)
      ELSE
            prepoc2 := 1;   (*jinak je volan blok "prepoce2"*)
            riz := 3;       (*prechod na dalsi krok*)
      END_IF;
```

10.1.8. Prepocet

Pomocí bloku přepočet se realizuje transformace souřadnic z kartézského systému do tříosého systému mobilního robotu a děje se tak pro všechny přenosové parametry (souřadnice zadané jako x a y, finálové rychlosti zrychlení a deakcelerace dle zvolené pohybové rampy pohonů viz Obr. 27. strana 50).

Init část:

Nulování proměnných použitých v bloku „přepočet“

Cyklic část:

Přepočet pravoúhlých souřadnic (x, y) do tříosého systému, přepočet finálové rychlosti, přepočet finálové akcelerace a přepočet finálové deakcelerace mobilního robotu. Vždy se tak děje pro jednotlivé pohony a následuje také pokaždé rozklad na bity, pro přímé zadání pohonům.
Pozn.: Časté přetypování proměnných je z důvodu použití v PLC
Ukázka části kódu:

```
(*Vypocet konstant pro prepocty rychlosti, zrychleni a
deakcelerace do pravoúhlých souradnic *)
konstx := REAL_TO_DINT(10000*COS(ATAN(osay/osax)));
konsty := REAL_TO_DINT(10000*SIN(ATAN(osay/osax)));
(*Pohonu 1*)
pozice1 := DINT_TO_UDINT(sourx);
(*Vypocet poctu otacek pohonu 1*)

(*rozklad na byty*)
poz11 := UDINT_TO_USINT(pozice1 / 1 MOD 256);
poz12 := UDINT_TO_USINT(pozice1 / 256 MOD 256);
poz13 := UDINT_TO_USINT(pozice1 / (256*256) MOD 256);
poz14 := UDINT_TO_USINT(pozice1 / (256*256*256) MOD 256);
```

10.1.9. Prepoce2

V tomto bloku se realizuje přepočet z tříosých souřadnice mobilního robotu (z důvodu určení aktuální pozice robotu) zpět do kartézského souřadného systému (viz. kap. 9.2 strana 59) a také přepočet souřadnic aktuální polohy robotu a to jak od pohonů tak i od čidla. Proto zde také dochází k případné korekci dráhy (dojde ke komparaci hodnot uražené dráhy – údajů o poloze zařízení, které jsou k dispozici od pohonů a od čidla relativní polohy umístěného na podvozku robotu). Hodnotě naměřené čidlem polohy je přiřazena vyšší priorita, tudíž v případě nesrovnalostí obou hodnot jsou nové pohybových parametrů vypočítány jako pro směr x platí: aktualx := (aktualCidloX/K) – aktualx, pro směr y platí: aktualy := (aktualCidloY/K) – aktualy (přičemž „K“ je kalibrační konstanta, viz kapitola 8.4 strama 56) . Výhodou je že tento algoritmus proběhne v každém průchodu cyklu podle zařazení do příslušné taskové třídy (10 ms).

Init část:

Nulování proměnných použitých v bloku „prepoce2“

Cyklic část:

Přepočet se děje pro kombinace pohonů (i pro případ pokud je některý již a jiný ještě setrvává v pohybu). K přepočtu dochází vždy pro x-ovou i y-ovou složku.

Ukázka části kódu:

```
      aktualy:=(-aktualnipoz1*konstanta1/10000)-
      (aktualnipoz2*2*konstanta1/10000);
      prepoc2 := 0;
      (*zakaz vstupu do tohoto bloku*)
ELSE
      (*vypocet y-ove souradnice robotu zprumerovanim hodnot pozic
      od  pohonu  2 a 3*)
      aktualy:=((-aktualnipoz1*konstanta1/10000)-
      (aktualnipoz2*2*konstanta1/10000)+
      (aktualnipoz1*konstanta1/10000)
      +(aktualnipoz3*2*konstanta1/10000))/2;
      prepoc2 := 0;
      (*zakaz vstupu do tohoto bloku*)
END_IF;
      (*komparacni algorytmus, dochazi zde ke srovnani hodnot
      polohy od motoru a od cidla *)
IF aktualx <> aktualCidloX THEN
      aktualx :=  (aktualCidloX/K ) - aktualx;
      (*K = kalibracni konstanta  pro srovnani hodnot od obou
      cidel*)
ELSE
      aktualx := aktualx;
END_IF;

IF aktualy <> aktualCidloY THEN
      aktualy := (aktualCidloY/K) - aktualy;
ELSE
      aktualy := aktualy;
END_IF;
```

10.2. Programové bloky - monitorovací

Do této části byly zahrnuty dva tasky, které se starají o zpětnou vazbu pro kontrolu polohy mobilního robotu. Jsou jimi „cidlo“, které aktivně měří relativní poloho robotu a „monitor“, který bere polohu od pohonů pomocí integrovaného polohového enkodéru (viz. Obr. 5 strana 19).

10.2.1. Cidlo

Blok realizuje jednu ze dvou zpětných vazeb zařízení a to pomocí jednostranné komunikace po sériové lince, mezi PLC a čidlem relativní polohy, umístěného ve spodní části podvozku mobilního robotu. Hodnoty které čidlo pošle jsou automaticky načítány do buffru, odkud jsou dále čteny a zpracovávány.

Komunikace s čidlem probíhá bez aktivního potvrzování příjmu dat, vychází se pouze ze 'start Bytu' obsaženého na začátku přijatého komunikačního paketu.

Init část:

Obsahuje deklaraci proměnných config struktury předdefinovaného programového bloku automation studia, nastavení časovače „TON“, který je součástí programového vybavení automation studia, jako předdefinovaný programový blok knihovny ‘standard‘ a jeho nastavení.

Ukázka části kódu:

```
    (*initialize config struktury preddefinovaneho programoveho
    bloku automation studia studia*)
config_struct.idle := 4;
config_struct.delimc := 0;
strcpy(ADR(config_struct.delim), ADR('00'));
config_struct.tx_cnt := 2;
config_struct.rx_cnt := 2;
config_struct.tx_len := 255;
config_struct.rx_len := 255;
config_struct.argc := 0;
config_struct.argv := 0;
```

Cyklic část:

Nejprve je nutné nadefinovat a otevřít pro komunikaci a nastavit typ (komunikace s čidlem, jak již bylo několikrát uvedeno probíhá po RS232), dále nastavit parametry tohoto přenosu (počet bitů datového rámce a přenosová rychlost). Otevření se děje pomocí předdefinovaných programových bloků knihovny „DV frame“, konkrétně potom funkcí ‘FRM_xopen_01‘. Následuje část pro čtení dat z čidla, která se realizuje pomocí programových bloků z téže knihovny (FRM_read_01) do bufferu (znovu funkcí z knihovny DV frame. Dále jsou data z buffreu načtena do pole (proměnná ‘RX_pole‘). Čtení se děje pomocí příkazu ‘memcpy‘ , který vezme obsah bufferu a jeho délku a nakopíruje ho na adresu, která je určena (do ‘RX_pole‘). Nyní je možné buffer uvolnit pomocí ‘FRM_rbuf_01 ‘ (znovu součást knihovny „ DV frame “, jako předdefinovaný blok) . Poslední částí je potom filtrování dat načtených do pole a oddělení velikosti změny v obou osách směru pohybu (proměnné aktualCidloX a aktualCidloX), vychází se zde ze znalosti komunikačního paketu (obdobného jako kap. 8.1.2 strana 54).
Nejprve se hledá znak „a“, který značí začátek datového paketu, který posílá převodník ve formátu 7 bytů. V tomto bloku dochází také k filtrování přijatého paketu, kde třetí byte znamená změnu ve směru X a pátý byte změnu ve směru Y (popis datového paketu z převodníku bude v Příloze 1) hodnot naměřených čidlem. Tyto jsou pak načítány do cyklického bufferu (který běží ve ‘FOR cyklu‘) jako integrační přírůstky v obou směrech (prirustekX a prirustekY). Ty se teprve sčítají do již zmíněných proměnných aktualCidloX a aktualCidloY pro komparační cyklus.

Zpoždění 5 s (pomocí předdefinovaného programového bloku „TON“ z knihovny „Standard“)při inicializaci čidla je z důvodu časové prodlevy při otevírání portu. Aby přenos byl korektní a nedocházelo ke kolizím při otevírání portu bylo nutné zavést tuto zpožďovací smyčku.

Ukázka části kódu:

```
      (*Otevirani portu*)
IF OpozdenyStart.Q THEN
      IF NOT(portJeOtevreny) THEN
            FRM_xopen_01(enable:=1, device := ADR('IF1'),
            mode := ADR('RS232,9600,N,8,1'),
            config := ADR(config_struct));
      (*povoleni a otevreni jednostranne komunikace -puze prijem;
      nastaveni komunikacniko paketu dat*)
                  status_open := FRM_xopen_01.status;
      (*povoleni preddefinovaneho bloku autom. studia*)
                  frm_ident := FRM_xopen_01.ident;
      (*povoleni preddefinovaneho bloku autom. studia*)

                  IF status_open = 0 THEN
      (*kontrola otevreni portu*)
                        portJeOtevreny := TRUE;
                END_IF;
      END_IF;
END_IF;
      (*Prijimani dat z cidla*)
      IF portJeOtevreny THEN
            FRM_read_01(enable := 1, ident := frm_ident);
      (*precte data;povoleni preddefinovaneho bloku autom. studia*)
            read_buffer := FRM_read_01.buffer;
      (*povoleni nactenni dat do buffru pomoci předdefinovaného
      bloku autom. studia*)
            read_buffer_length := FRM_read_01.buflng;
            status_read := FRM_read_01.status;
            IF status_read = 0 THEN
      (*nastavení delky prijmu dat, vychazi se ze znalosti délky
      komunikačního paketu*)
                  IF (pocetPrijatychBytu + read_buffer_length) <
                  SIZEOF(RXpole) THEN
      (*zkopiruje obsah buffru o definovane delce na adresu v
      pameti *)
                        memcpy(ADR(RXpole)+pocetPrijatychBytu,
                        read_buffer, read_buffer_length);
                        pocetPrijatychBytu := pocetPrijatychBytu +
                        read_buffer_length;
                        FRM_rbuf_01(enable := 1, ident := frm_ident,
                        buffer := read_buffer,buflng :=
                        read_buffer_length);
                  END_IF;
      END_IF;
END_IF;

      (*uvolni buffer*)
IF pocetPrijatychBytu >= 7 THEN
      FOR i := 0 TO ( pocetPrijatychBytu -1 )DO
      (*cyklický buffer*)
            IF ( pocetPrijatychBytu - i) >= 7 THEN
```

```
      (*alespon 7 znaku do konce pole*)
                IF RXpole[i] = 97 THEN
      (*hleda zacatek paketu - znak "a"  = 97 ascii*)
                    memmove(ADR(RXpole[0]),       ADR(RXpole[i]),
                    SIZEOF(RXpole) - i *SIZEOF(RXpole[0]));
      (*jakmile najde "a",umaze znaky ktere byly pred nim a
      presune obsah v RXpole a na zacatek*)
                    pocetPrijatychBytu := pocetPrijatychBytu - i;
      (*cidlo ze sve podstaty je integracni je tedy nutne pocitat
      prirustky do nejake globalni promenne - cyklickeho
      buffru,odkud teprve bereme hodnoty do komparacniho algorytmu,
      ktery je v bloku prepoce2*)
                    prirustekX ACCESS ADR(RXpole[i+2]);
                    prirustekY ACCESS ADR(RXpole[i+4]);
                    aktualCidloX := aktualCidloX + prirustekX;
                    aktualCidloY := aktualCidloY + prirustekY;
                END_IF;
            END_IF;
      END_FOR;
END_IF;
OpozdenyStart();
      (*casovac z duvodu spozdeni aby se stihnul otevirany port
      inicializovat*)
```

10.2.2. Monitor

Druhou zpětnovazební smyčkou robotu jako celku je informace o poloze, kterou vyhodnocují a posílají samy integrované regulační pohony. To je realizováno právě blokem „monitor“. Zde jsou načítány informace z polohového enkodéru, který je součástí pohonu (viz. 3.1.2 strana 19).Informace je přijímána pomocí sběrnice CAN prostřednictvím CANopen objektu SDO (viz. příklad na straně 38) a to postupně od všech pohonu. Následně je tato informace převedena z hexadecimálního na dekadické číslo z důvodu dalšího zpracování v ostatních částech programu mobilního robotu.

Init část:

Inicializace a nulování proměnných, povolení vstupu do tohoto bloku.

Cyklic část:

Cyklická část zpracovává postupně pro všechny pohony čtení aktuální pozice pohonu podle výše uvedeného příkladu (str. 30), tj. pokud stavové slovo (6041_h) signalizuje změnu nastavení pohonu, je pomocí CANopen objektu RPDO (6064_h) čtena aktuální pozice, která je ukládána datového pole (‘data‘),prostřednictvím CANopen objektu TPDO a následně převedena na dekadické číslo (‘aktualnipoz‘). Poté jsou hodnoty pro čtení a zápis nastaveny zpět na původní hodnoty a je možno stejný algoritmus opakovat pro další pohony.

Pozn.:Tento blok je spouštěn jen když alespoň jeden z motorů integrovaných pohonů běží, aby zbytečně nezatěžoval svými procesy program.

Ukázka části kódu:

```
(*Pohon 2*)
          (*pokud statusword signalizuje prestavovani pohonu a
          soucasne ubehlo 8 pruchodu*)
     IF   data[0] = 16#27 AND data[1] = 16#12 AND n = 4 THEN
          cobbackub := cobidread;
     (*zalohovani aktualniho COB-ID pro cteni*)
          cobbackupw := cobidwrite;
     (*zalohovani aktualniho COB-ID pro zapis*)
          cobidread := rsdo + nodeid2;
     (*vtpocet COB-ID RSDO - pro ctení akt.poz.*)
          cobidwrite := tsdo + nodeid2;
                     (*COB-ID pro zapis pozadavku cteni*)
     (*40h =pozadavek cteni dat z objektu aktualni poz. 6064h*)
          dataw[0] := 16#40;dataw[1] :=16#64;dataw[2] :=16#60;
          dataw[3] := 16#00;dataw[4] :=16#00;dataw[5] :=16#00;
          dataw[6] := 16#00;dataw[7] :=16#00;
     (*nulovani datoveho pole pro ulozeni vracenych hodnot*)
          data[0] := 0; data[1] := 0; data[2] := 0; data[3] := 0;
          data[4] := 0; data[5] := 0; data[6] := 0; data[7] := 0;
          dataleng := 4;
          writeen := 1;
     END_IF;
```

10.3. Programové bloky předdefinovaných pohybů

Dalším bodem zadání diplomové práce bylo: „Sestavení demonstračního pohybového programu“. Jako ukázku pohybových schopností robotu jsem si vybral následující tři pohyby - úsečku, čtverec a polygon (kdy robot vykresluje svůj vlastní půdorys). Pro jednoduchost ukázky vznikl blok „vstupy“ který je blíže popsán výše (10.1.4 strana 64), kdy kombinací vstupů automatu (přiváděním logických jedniček na odpovídající vstupy) jsou voleny příslušné předdefinované pohyby robotu.

Tabulka kombinací vstupů pro demonstrační pohyby:

Kombinace vstupů:	Vykonávaný pohyb:
Vstup 1	Úsečka, rychlostní rampa 1 (rychlost 500 Inc/s)
Vstupy 1 a 8	Úsečka, rychlostní rampa 2 (rychlost 200 Inc/s)
Vstup 2	Čtverec, rychlostní rampa 1 (rychlost 500 Inc/s)
Vstupy 2 a 8	Čtverec, rychlostní rampa 2 (rychlost 200 Inc/s)
Vstup 4	Polygon, rychlostní rampa 1 (rychlost 500 Inc/s)
Vstupy 2 a 8	Polygon, rychlostní rampa 2 (rychlost 200 Inc/s)
Vstup 7	Návrat robotu zpět do nnulové polohy, podle rychlostní rampy 1 (rychlost 500 Inc/s)

10.3.1. Usecka

Prvním z možných předdefinovaných pohybů mobilního robotu je úsečka. Jde o pohyb, kdy se zřízení přesune po přímce na předem definovanou vzdálenost (definovanou rychlostí, s definovanou akcelerací a deakcelerací) a opět se vrátí po přímce zpět.

Init část:

Pro definovaný pohyb není nutné do init části cokoliv deklarovat.

Cyklic část:

Nejdříve se realizuje podmínka výběru vzoru pohybu (rychlostní rampa 1, nebo rychlostní rampa 2), dále se nadefinují příslušné parametry pohybu (rychlost, akcelerace a deakcelerace) pro obě rychlostní rampy a následuje zadání cílové pozice (v kartézských souřadnicích).

Ukázka části kódu:

```
IF pohyb = 0 AND faze = 0 AND initok = 3 THEN
     sourx := 1065;
(*zadani souradnic cilove pozice 1065 Inc, 1065/213 = 5
otacek = 188,4 cm*)
     soury := 0;
     initok := 1;          (*volani bloku set*)
     faze := 1;            (*nastaveni ridici promenne*)
ELSIF pohyb = 1 AND faze = 0 AND initok = 3 THEN
     sourx := -1065;
     soury := 0;
     initok := 1;
     faze := 1;
END_IF;
```

10.3.2. Ctverec

Dalším z pohybů mobilního robotu je čtverec. Jde o pohyb, kdy se zřízení přesouvá po přímce na předem definovanou vzdálenost a vždy se „otočí o 90° a vykoná stejný pohyb. Toto se opakuje dokud robot nedosáhne opět nulové polohy. Opět je pohyb nadefinován pro dvě možné rychlosti přesunu (jsou ve všech případech stejné).

Init část:
Pro definovaný pohyb není nutné do init části cokoliv deklarovat.

Cyklic část:
Tak jako v předchozím případě se nejprve zadá podmínka výběru pohybového vzoru (rychlostní rampa 1, nebo rychlostní rampa 2), dále se nadefinují příslušné parametry vykonávaného pohybu (rychlost, akcelerace a deakcelerace) pro obě rychlostní rampy a následuje zadání cílových pozice (v kartézských souřadnicích) a to s absolutním polohováním (viz . Obr. 26 na straně 48).

Ukázka části kódu:

```
IF pohyb = 0 AND faze = 0 AND initok = 3 THEN
           sourx := 1065;        (*zadani souradnic*)
           soury := 0;
           initok := 1;               (*volani bloku set*)
           faze := 1;
     ELSIF pohyb = 1 AND faze = 0 AND initok = 3 THEN
           sourx := 0;                (*zadani souradnic*)
           soury := -1065;
           initok := 1;               (*volani bloku set*)
           faze := 1;
     ELSIF pohyb = 2 AND faze = 0 AND initok = 3 THEN
           sourx := -1065;            (*zadani souradnic*)
           soury := 0;
           initok :=1;                (*volani bloku set*)
           faze := 1;
     ELSIF pohyb = 3 AND faze = 0 AND initok = 3 THEN
           sourx := 0;                (*zadani souradnic*)
           soury := 1065;
           initok := 1;               (*volani bloku set*)
           faze := 1;
     END_IF;
```

10.3.3. Polygon

Posledním z předdefinovaných pohybu je polygon. Opět je zde využito absolutního polohování pro zadávání souřadnic cílových pozic (zadávání je v kartézském souřadném sytému). Robot se postupně přesouvá do předem určených vrcholů polygonu. Tak jako v předešlých případech předdefinovaných pohybů i zde je možnost výběru ze dvou rychlostí (rampa 1 a rampa 2).

Init část:

Pro definovaný pohyb není nutné do init části cokoliv deklarovat.

Cyklic část:

Cyklická část je opět podobná předešlým. Výběr vzoru (rychlosti pohybu), nadefinování pohybových parametrů (rychlost, akcelerace a deakcelerace) pro obě rychlostní rampy a zadávání cílových pozic dílčích pohybů robotu. Vykonání tohoto pohybu opět končí v počátku [0;0].

Ukázka části kódu:

```
IF pohyb = 0 AND faze = 0 AND initok = 3 THEN
           sourx := 536;                    (*zadani souradnic*)
           soury := -230;
           initok := 1;                     (*volani bloku"set"*)
           faze := 1;
     ELSIF pohyb = 1 AND faze = 0 AND initok = 3 THEN
           sourx := 536;                    (*zadani souradnic*)
           soury := 230;
           initok := 1;                     (*volani bloku"set"*)
           faze := 1;
     ELSIF pohyb = 2 AND faze = 0 AND initok = 3 THEN
           sourx := 230;                    (*zadani souradnic*)
           soury := 536;
           initok := 1;                     (*volani bloku"set"*)
           faze := 1;
     ELSIF pohyb = 3 AND faze = 0 AND initok = 3 THEN
           sourx := -230;                   (*zadani souradnic*)
           soury := 536;
           initok := 1;                     (*volani bloku"set"*)
           faze:=1;
     ELSIF pohyb = 4 AND faze = 0 AND initok = 3 THEN
           sourx := -536;                   (*zadani souradnic*)
           soury := 230;
           initok := 1;                     (*volani bloku"set"*)
           faze := 1;
     ELSIF pohyb = 5 AND faze = 0 AND initok = 3 THEN
           sourx := -536;                   (*zadani souradnic*)
           soury := -230;
           initok := 1;                     (*volani bloku"set"*)
           faze := 1;
     ELSIF pohyb = 6 AND faze = 0 AND initok = 3 THEN
           sourx := 230;                    (*zadani souradnic*)
           soury := -536;
           initok := 1;                     (*volani bloku"set"*)
           faze := 1;
     ELSIF pohyb = 7 AND faze = 0 AND initok = 3 THEN
           sourx := -230;                   (*zadani souradnic*)
           soury := -536;
           initok := 1;                     (*volani bloku"set"*)
           faze := 1;
     END_IF;
```

11. Další možná čidla

Mobilní robotické zařízení diskutované v této práci má možnost využít, co se týče snímačů, pouze enkodéry od motorů a přídavné optické čidlo polohy s relativním snímáním souřadnic. Vzhledem k možnostem B&R automatů se nabízí močnost připojení dalších typů snímačů. Jako zajímavé a z hlediska vyhodnocení dynamiky zařízení účelné by se mi jevilo připojení akcelerometru.

Akcelerometry

Jsou zařízení, která využívají setrvačnosti hmoty k měření rozdílu mezi kinematickýma gravitačním zrychlením. Jsou to zařízení určená k měření vzdálenosti, naklonění tělesa,vibrací s velmi vysokou přesností. Ale také pro určování pozice tělesa v prostoru.

Trendem v oblasti vývoje jsou v poslední době MEMS senzory (*micro electronic magnetic sensitivity*). Na dnešním trhu je k dostání několik druhů akcelerometrů fungujících na různých principech a vyráběnými různými technologiemi výroby. Existuje také několik způsobů snímání, mezi základní se potom řadí:

Piezoelektrické akcelerometry:

Využívají piezoelektrický krystal (přírodní nebo keramický), který generuje náboj úměrný působící síle, která se projeví při zrychlení.

Piezoresistivní akcelerometry:

Vyžívá mikrokřemíkovou mechanickou strukturu, kde zrychlení je úměrné změně odporu.

Akcelerometry s proměnnou kapacitou:

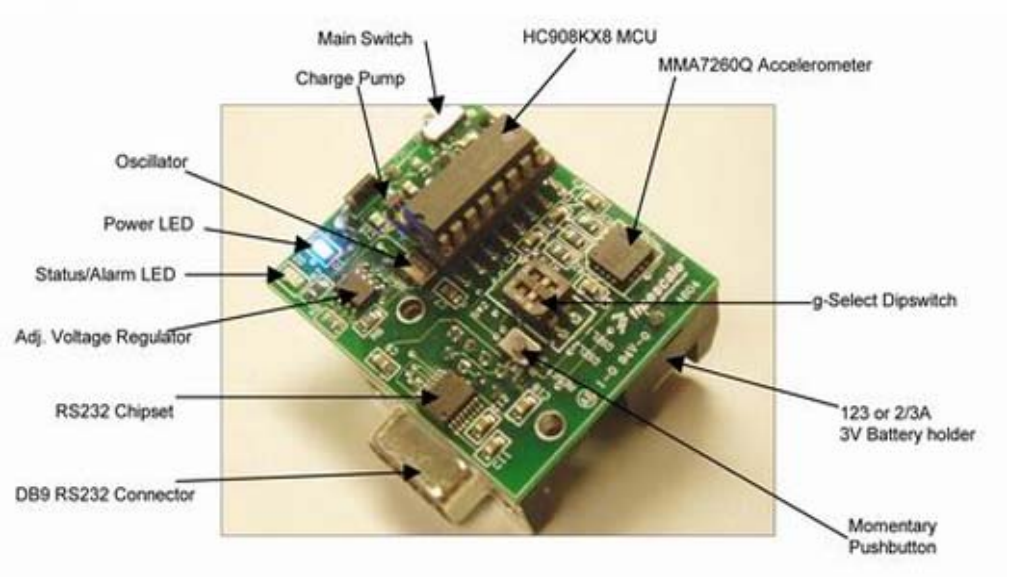
Kapacitní snímání pohybu narozdíl od piezorezistvního, nebo piezoelektrického má několik výhod k nimž patří teplotní stabilita, opakovatelnost, CMOS obvodová kompatibilita a schopnost měření zrychlení o nízké frekvenci. Z těchto důvodů také většina akcelerometrů vyráběných technologií bulk MM (slepení senzoru a elektronického obvodu) využívají kapacitního snímání pohybu. U těchto akcelerometrů došlo ke zlepšení mnoha parametrů, přesto mezi limitující faktory stále patří omezení jen na jednu osu snímání, nízká rezonanční frekvence.

Zde bych krátce zmínil některé z nich:

Analog Devices ADXL202E - patří z hlediska měření rozdílu zrychlení ke kapacitním. Jedná se o dvouosý digitální akcelerometr vyráběný i MEMS technologií. Mezi jeho základní parametry patří rozsah ±2g (±20m/s$^{2)}$, výhodou je nízká spotřeba a přijatelná cena. ADXL202E je vhodný jak pro měření dynamických zrychlení (např. vibrací) tak pro měření statických zrychlení (např. gravitačního zrychlení).

Motorola 1260D – Jedná se o klasický jednoosý kapacitní akcelerometr (technologie MEMS) s digitálními výstupy. Měřící rozsah je ±1,5 g a úrovní šumu 500 mg/$\sqrt{Hz}$

Jako poslední a co do využití zřejmě nejvhodnější se jeví kompaktní zařízení od společnosti Freescale, obsahující mimo akcelerometru, který je tak jako všechny předchozí vyráběný technologií MEMS, také komunikační rozhraní. Jedná se o kit s akcelerometrem MMA6260Q stejného výrobce s parametry ± 1.5g, citlivost 800 mV/g, komunikujícím pomocí RS 232 (po případě možno upravit i na ethernet pomocí přídavného modulu). Takto koncipované zařízení je schopné určit polohu s velkou přesností a díky jeho komunikačním schopnostem by bylo vhodné jako alternativa za pro čidlo snímání polohy využité v této diplomové práci.



Obr. 38 Kompaktní zařízení pro určování polohy společnosti Freescale [21]

12. Poznatky získané při řešení úkolu

Při základním poznávání problému jsem vycházel z řady referencí (všechny jsou uvedeny v seznamu použité literatury), které popisovaly jak vyžitý systém řízení PLC X20 CP0291, tak užité komunikačního rozhraní CAN, protokol CANopen, regulační pohony IclA, standardy sériové komunikace apod.

Pro tvorbu řídícího algoritmu a demonstračních pohybových programů v prostředí Automation Studio 2.6 bylo využito jazyka *ST – Structured Text*. Řídící program se skládá z několika dílčích podprogramů (subrutin). Ve většině z nich je využito předdefinovaných funkcí (nebo bloků funkcí), jejichž struktura je pevně daná. V těchto podprogramech je možné nastavit některé vstupní podmínky (určit přes jaký interface se bude přistupovat k zařízení, případně jakou rychlostí) a naopak tyto funkce z pravidla vrací odpověď o svém stavu, tzv. status, který je vyhodnocen jako korektní pokud odpovídá nule. Jakákoliv jiná hodnota je potom chápána jako identifikátor chyby, se incidencí k danému problému (tyto kódy chyb lze identifikovat pomocí Helpu a dále řešit). Předdefinovaných funkcí bylo poměrně hojně využito i při komunikaci s periferiemi (jak s pohony tak i s polohovým čidlem).

Program byl vytvářen pro zcela nový systém řízení pomocí PLC X20 řady 29x a oproti předchozím podobným verzím (na systému PLC 2003 stejného výrobce) byl doplněn i o možnost monitorování pomocí snímače relativní polohy. Snímač polohy je tvořen pomocí optické myši (PS/2) připojené k PLC (na port RS 232 pomocí převodníku).

Ačkoliv po kompilaci daného programu nebyly nalezeny žádné chyby, po připojení externích zařízení (pohony a převodník s čidlem) se jich musela více či méně úspěšně řešit celá řada. Záměr použití snímací a vyhodnocovací elektroniky optické počítačové myši jako čidla relativní polohy robotu, se z počátku ukázal jako lichý. Automat data nenačetl korektně (naplnil se buffer, ale načtená data neodpovídá hodnotám v mezích komunikačního paketu) a tím bylo znemožněno další zpracování. Toto chování se však neprojevilo ani u klasického PC, ani později u automatů vyšších řad, se kterými byl převodník také zkoušen.

Možnosti vlivu jakéhokoliv rušení se vyloučily, z důvodu ověřování na několika různých místech i jiných PLC obdobné struktury (X20CP201 a X20CP292). Napěťové úrovně byly také prověřeny osciloskopickým měřením a bylo zjištěno, že jsou v pořádku.

Jako pravděpodobné příčiny výše popsaného problému se jevily tři základní důvody. Jako první, byla možná nekompatibilita chipsetových sad v převodníku (ATmega32) a automatu (Motorola). Další příčinou, která se zdála být pravděpodobná byla nedostatečná přesnost přenosové rychlosti, na kterou už by automat nereagoval. Posledním z důvodů byla komunikační rychlost převodníku, která byla stanovena na 115200 bps (i když by PLC mělo podporovat komunikaci s ostatními zařízeními i touto přenosovou rychlostí). Jako řešení daného problému se ukázalo stanovit tuto rychlost na nižší hodnotu 9600 bps, která by vzhledem k rychlosti pohybu robotu byla dostatečná. Přičemž je tato komunikační rychlost lehce pod limitem určeným pro sériovou komunikaci PS/2. Ten je stanoven na 10000 bps. Vyvstává tak teoretické riziko možných chyb při snímání. Další nejbližší přenosová rychlost, kterou automat disponuje (dle normované řady) je 19200 bps. Při testech na určení kalibrační konstanty „K“ se však žádné chyby neprojevili, tudíž byla rychlost ponechána na nižší hodnotě.

Další komplikací byl problém s pohony. Systém je nebyl schopný identifikovat, takže jim byly znovu zadány adresy uzlů (Node-ID, byly zvoleny 2_h, 3_h, 4_h) a také rychlost pro komunikaci s automatem (ta byla určena na hodnotu 20 $kbit.s^{-1}$). Při těchto úpravách bylo zjištěno, že jeden z motorů nereaguje. Byl proto poslán do servisu. I po těchto úpravách se ovšem nepovedlo komunikaci navázat Byly zkontrolovány kabely a odpor sloužící jako ukončení sběrnice nahrazen z původních 133Ω za nominální 120Ω (realizováno terminátorem fundovaného výrobce), ale ani tato úprava nebyla řešením popsaného problému.

Po hardwarových úpravách jsem (viz. výše) začal hledat chybu v softwaru. Postupným ověřováním a vylučováním možností jsem se dostal k předefinovanému bloku „CANopen_01“ z knihovny CAN_Lib. Status bloku byl 0, což znamená že proběhl v pořádku. Následoval další předefinovaný funkční blok „CANwrite_01“, který ovšem vrátil jako status hlášení číslo „8877“, které odpovídá chybovému hlášení: „pro tento identifikátor (Node-ID) nejsou momentálně dostupná žádná data“. Byly tedy dvě možnosti. První že pohony neposílají „Boot-up“ zprávu (kterou posílají po zapnutí napájení a to automaticky dokud není potvrzeno její přijetí), ale ta byla následně vyloučena připojením jednotlivých pohonu k PC, kde bylo možno tuto zprávu přečíst. Jako druhá se jevila varianta špatně inicializované sběrnice (pomocí předdefinované bloku CANopen_01). Po ověření všech zadaných údajů (které se znovu jevily jako v pořádku) jsem se rozhodnul sestavit znovu již zmíněný systém s PLC řady 2003 a pokusit se pohony zprovoznit na starší verzi automatu. Pro tento systém je typický problém s určováním Node-ID adresy (swich, který je k tomu určený a z nepochopitelných důvodů přičte k adrese uzlu vždy 1 – děje se tak hardwarově a uživatel nemá možnost tuto vlastnost ovlivnit, protože po nastavení tohoto swiche na hodnotu 0 dojde k automatickému vypnutí automatu). Nicméně i na tomto systému došlo k chybě ‚8877‘, tedy stejné se systémem X20. Tato chyba se potom opakovala i po zkušebním nahrání staršího programu z minulé práce [6], kde se tato část navázání komunikace se zařízením také řešila.

Zmíněnou chybu ovšem nebyli schopni řádně odůvodnit ani pracovníci na Brněnském supporu společnosti B&R automation, na které jsem se obrátil se žádostí o vysvětlení. Pouze potvrdili že funkce „CANopen_01“ a „CANwrite_01“ jsou volány správně. Do odevzdání práce se nepovedlo tuto chybu odstranit, tedy větší část práce, kam se řadí i demonstrační pohybové programy, bylo možné ověřit spíše teoreticky. O správnosti programu lze usuzovat jen dle výsledků kompilátoru a provedených simulací.

Řešením by mohlo být použít jako řídící člen mikropočítač na místo PLC, kde by se dala sběrnice CAN inicializovat jinými způsoby. U automatu se musí využít předdefinovaného programového bloku, který je k tomu určený, jiný způsob zde není.

13. Závěr

Prvním cílem této práce bylo „Abstrahovat použité technologie PLC B&R automation a protokolu CAN Open“ čemuž je věnována převážná část diplomové práce, z důvodu složitosti a velkého rozsahu teoretických znalostí použitých technologií nutných pro další práci, zejména pak při tvorbě řídícího programu a programových bloků.

Tvorba programových bloků řídícího algoritmu všesměrového autonomního mobilního robotu probíhala v softwarovém prostředí Automation studia společnosti B&R Automatinon. Všechny programové bloky byly psány programovacím jazykem *ST – Structured Text*, který je součástí mezinárodní normy IEC 61131-6.

Vytvořené programové bloky řídícího algoritmu byly implementovány do programovatelného automatu řady X20, stejného výrobce. Pro download řídícího programu byly postupně využity obě základní možnosti komunikace, jak po sériové lince, tak později i pomocí Ethernetového rozhraní, z důvodu pozdějšího obsazení RS 232 čidlem pro snímání relativní polohy robotu.

Byly také vytvořeny čtyři programové bloky pro demonstraci pohybu robotu. Tři z nich jsou přímo bloky jednotlivých pohybů, čtvrtý potom pro obsluhu jednotlivých vstupů programovatelného automatu, z důvodu snadné implementace pohybových programovatelných bloků pomocí automatu.

Při implementaci softwaru se projevily komplikace při komunikaci s pohony. Program řídícího algoritmu nejevil při kompilaci žádné známky chyb a rovněž byl ověřen simulací. Tento problém se dosud nepodařilo odstranit ani v součinnosti s pracovníky support oddělení společnosti B&R automation. Další využití kombinace PLC a pohonů je tedy dle mého názoru na zvážení.

Jako jednu z příčin lze označit možnou nekompatibilitu CANopen v knihovně CAN_Lib obsažené v Automation studiu. Nevylučuji ani možný problém na úrovni fyzické vrstvy (příklad konfrontace reálného a manuálového popisu zapojená modulu PS9500). Vhodnou alternativou by se mohly stát programovatelné automaty výrobce Berger Lahr – positec (tedy výrobce integrovaných pohonů IclA).

Sériového rozhraní bylo využito pro připojení čidla snímajícího relativní polohu zařízení. Naměřené hodnoty jsou srovnány s hodnotami naměřenými polohovými enkodéry regulačních pohonů a následně využity v komparačním algoritmu pro případnou korekci dráhy robotu. Pro tento účel bylo čidlo kalibrováno.

Zvolená konstrukce optického čidla relativní polohy vytváří jistá omezení a neshoduje se plně s průmyslovými standardy odpovídající takovému zařízení i proto byly v závěru práce diskutovány některé z alternativních senzorických řešení.

Koncepčně se jedná o inovativní řešení jak z hlediska systému robotu, tak z hlediska problematiky PLC a připojených pohonů a senzoru. Tato skutečnost naopak hovoří pro využití automatů stávajícího výrobce a dalším výzkumu v tomto směru.

Použitá literatura:

[1] ŠVARC, Ivan. Automatizace, Automatické řízení. Vysoké učení technické v Brně, Fakulta strojního inženýrství. CERM Brno, 2005. 262 s. ISBN 80-214-2943-7

[2] NOVÁK, Petr. Mobilní roboty, pohony, senzory,řízení. BEN – technická literatura, Praha 2005. 248 s. ISBN 80-7300-141-1

[3] KOCOUREK, Petr; NOVÁK Jiří. Přenos informace. České vysoké učení technické v Praze.vydavatesltví ČVUT, 2004. 164 s. ISBN 80-01-02892-5

[4] KÁLLAY, Fedor; PENIAK, Peter. Počítačové sítě a jejich aplikace. Grada Publishing a.s. Praha;2003. 356 s. ISBN 80-247-0545-1

[5] TŘÍLETÝ,Luboš. Řízení pohybu tříosého mobilního robota. Brno 2003. 65 s. Diplomová práce na fakultě strojního inženýrství VUT na ústavu automatizace a informatiky. Vedoucí .diplomové práce Ing. Zdeněk Němec, Csc

[6] BARTOŠ, Stanislav. Řízení pohybu tříkolového mobilního robota. Brno 2005. 51 s. Diplomová práce na fakultě strojního inženýrství VUT na ústavu automatizace a informatiky. Vedoucí diplomové práce Ing. Zdeněk Němec, Csc

[7] BERNECKER & RAINER, user manual TM210. B&R 2005. TM210TRE.25-ENG

[8] BERNECKER & RAINER, user manual TM211. B&R 2005. TM211TRE.25-ENG

[9] BERNECKER & RAINER, user manual TM213. B&R 2006. TM213TRE.25-ENG

[10] BERNECKER & RAINER, user manual TM223. B&R 2006. TM223TRE.25-ENG

[11] BERNECKER & RAINER, user manual TM230. B&R 2005. TM230TRE.25-ENG

[12] BERNECKER & RAINER, user manual TM240. B&R 2006. TM240TRE.25-ENG

[13] BERNECKER & RAINER, user manual TM246. B&R 2006. TM246TRE.00-ENG

[14] BERNECKER & RAINER, user manual TM250. B&R 2005. TM250TRE.00-ENG

[15] BERNECKER & RAINER, user manual TM260. B&R 2006. TM260TRE.25-ENG

[16] SIG POSITEC AUTOMATION., Lahr.: IclA Field bus manual.

[17] SIG POSITEC AUTOMATION., Lahr.: IclA Device manual.

[18] BOSCH., Stuttgart 1.: CAN Specification version 2.0.

[19] Řízení PS/2 myši.[online]. 8.9.2007.[citace 5.4.2008].dostupné z: http://elektronika.kvalitne.cz/ATMEL/necoteorie/tutorial/PS2/PS2_mouse.html

[20] Základní parametry RS-232.[online].2003.[citace 18.3.2008].dostupné z: http://rs232.hw.cz/#parametry

[21] Vojáček, Antonín. Freescale akcelerometry pro malá zrychlení (nízká g).[online]. 3.7.2005.[citace 5.5.2008].dostupné z: http://automatizace.hw.cz/mereni-a-regulace/ART167-freescale-akcelerometry-pro-mala-zrychleni-nizka-g.html

[22] Integrovaný regulační pohon ICLA.[citace 19.2.2008].dostupné z: http://www.regulacni-pohony.cz/data/komp_icla.html

[23] Oficielní stránky společnosti B&R automation: http://www.br-automation.com

Příloha

Pro převod z synchronního signálu, který používá počítačová myš komunikující protokolem PS/2, na asynchronní signál odpovídající standardům klasické RS232 byl použit převodník založený na bázi ATmega32 (16 MHz) a MAX 232. Schéma zapojení je na obr. níže.

Protokol PS/2 komunikuje pomocí jednoho vodiče, který je společný pro vysílání i příjem dat. O vysílání nečinnosti nebo přijímání datového paketu rozhoduje CLK (hodiny) signál, který myš sama generuje. Pokud takový signál přijme přes svá rozhraní na vstupu tento převodník, upraví jej na výstupní signál odpovídající protokolu „Mouse Protocol Mouse“, který již automat přes svoje rozhraní (RS232) dokáže zpracovat a ten je dále použit v programovém bloku „cidlo“, kde je dále zpracován na změnu polohy ve směru ‘x‘ nebo ‘y‘.

Další rozdílností jsou napěťové úrovně obou technologií. PS/2 odpovídá napětí 5 V, zatím co pro použití RS232 je nutné 12 V napájení. I tento problém řeší výše uvedené zapojení.

Datový paket na výstupu z převodníku těchto sedm bytů:

1. znak „a“ – značí začátek paketu
2. znak „i“ – značí myš s komunikačním protokolem Microsoft Mouse
   znak „m“ – značí myš s komunikačním protokolem Mouse System Mouse
   znak „n“ – značí že nebylo žádné zařízení připojeno
   znak „d“ – značí neznámé zařízení
   znak „k“ – značí připojení klávesnice
3. X1 – přírůstek v ose x (nabývá hodnot – 127 až 128)
4. X2 – směr pohybu (0 kladný směr -1 pro záporný směr)
5. Y1– přírůstek v ose x (nabývá hodnot – 127 až 128)
6. Y2– směr pohybu (0 kladný směr -1 pro záporný směr)
7. znak „x – značí konec paketu

Nastavení převodníku:

Rychlost: 9600 $bit.s^{-1}$
Počet bitů: 8
Parita: žádná
Počet start bitů: 1
Počet stop bitů: 1

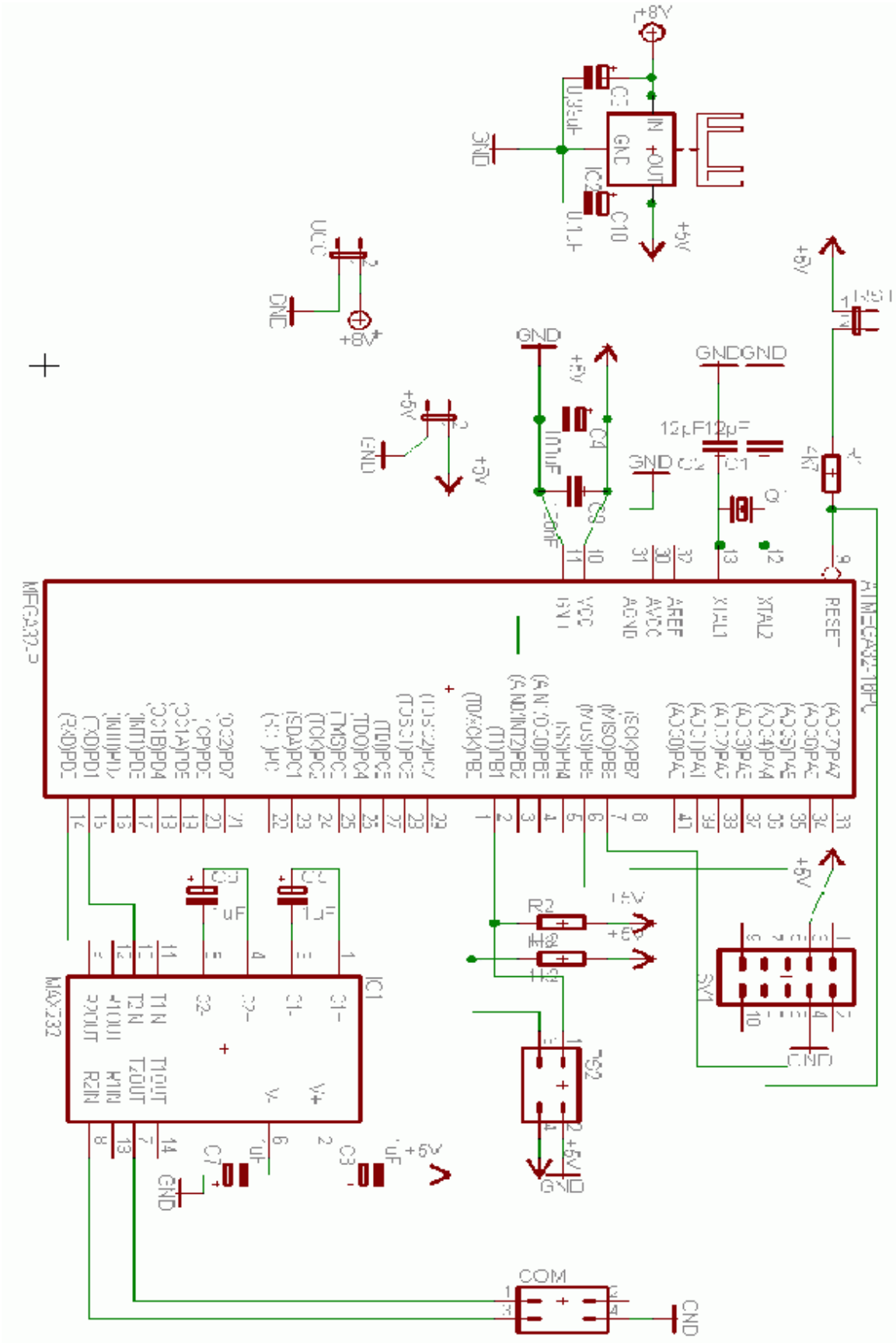
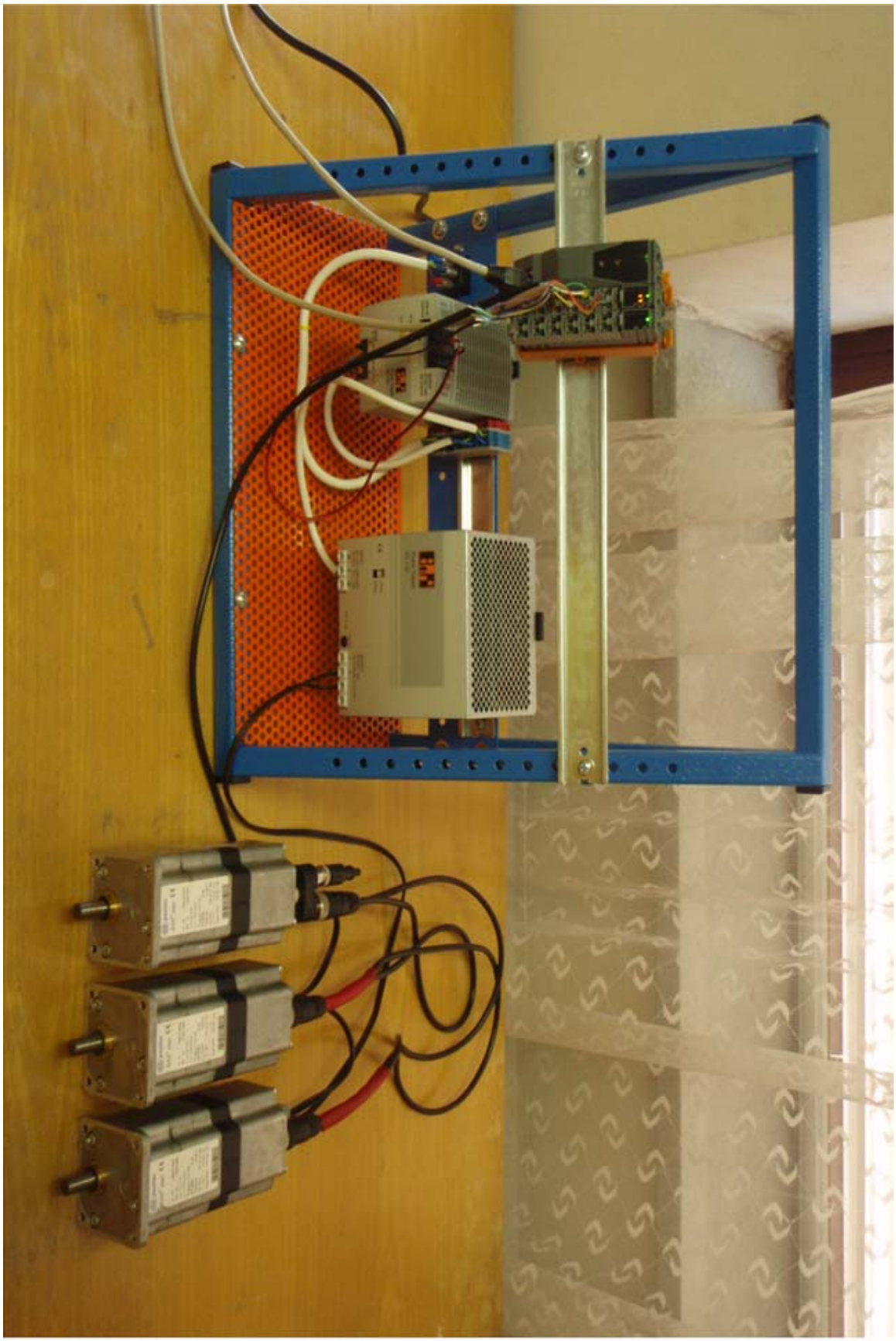


Schéma zapojení převodníku PS/2 – RS232



Vývojový kit s automatem, zdroji a IclA pohony