Komunikace v síti DIOCAN

Abstrakt

Realizace přenosu dat prostřednictvím sítě DIOCAN® (implementace protokolu CANopen v produktech firmy AMiT).

Autor: Jan Kučera, Václav Kaczmarczyk
Dokument: ap0007_cz_01.pdf

Příloha

Obsah souboru: ap0007_cz_01.zip

can_p1_cz_01.dso	Inicializace a komunikace po sběrnici DIOCAN
can_p2_cz_01.dso	Příklad zasílání dat s iniciativy SLAVEa
can_p3_cz_01.dso	MULTIMASTER komunikace, Varianta 1 – SW modul **CAN_S_DI**, Řídicí systém 1
can_p4_cz_01.dso	MULTIMASTER komunikace, Varianta 1 – SW modul **CAN_S_DI**, Řídicí systém 2
can_p5_cz_01.dso	MULTIMASTER komunikace, Varianta 2 – SW modul **CAN_S_DO**, Řídicí systém 1
can_p6_cz_01.dso	MULTIMASTER komunikace, Varianta 2 – SW modul **CAN_S_DO**, Řídicí systém 2
can_p7_cz_01.dso	MULTIMASTER komunikace, Varianta 3 – SW modul **CAN_S_DI** a **CAN_S_DO** zároveň, Řídicí systém 1
can_p8_cz_01.dso	MULTIMASTER komunikace, Varianta 3 – SW modul **CAN_S_DI** a **CAN_S_DO** zároveň, Řídicí systém 2
can_p9_cz_01.dso	Komunikace po sběrnici DIOCAN s použitím virtuálního uzlu
can_p10_cz_01.dso	Obsluha uzlu jiného výrobce (WAGO 750-3x7)
can_p11_cz_01.dso	Obsluha uzlu jiného výrobce (WAGO 750-3x7) pomocí SDO rámců

Obsah

Historie revizí

Verze	Datum	Změny
001	4. 3. 2009	Nový dokument

Související dokumentace

1) Nápověda k vývojovému prostředí DetStudio
2) Katalogový list k modulu **ADC-CAN**
   soubor: adc-can_d_cz_xxx.pdf
3) AP0029 – Zásady používání sítě CAN
   soubor: ap0029_cz_xx.pdf
4) AP0030 – Komunikace v síti CANOPEN
   soubor: ap0030_cz_xx.pdf
5) Dokumentace k výrobkům Wago
   stránka: www.wago.com

1. Definice použitých pojmů

DetStudio
Návrhové prostředí firmy AMiT, které slouží pro parametrizaci řídicích systémů. Toto prostředí je volně ke stažení na www.amit.cz.

Komunikační objekt
Zpráva přenášená po sběrnici CAN. Komunikačními objekty obsahují veškeré informace o vlastnostech a funkčních schopnostech jednotlivých zařízení. Všechny komunikační objekty příslušné určitému zařízení (údaje specifické pro aplikaci a konfigurační parametry) jsou popsány předepsaným způsobem v pevně stanoveném formátu ve slovníku objektů. Každý komunikační objekt je dostupný prostřednictvím šestnáctibitového indexu, v případě objektů typu polí a záznamů (objektů složených z několika dalších objektů) doplněného osmibitovým subindexem.

Slovník objektů
Každý decentralizovaný systém řízení vyžaduje odlišné komunikační služby a protokoly. Protokol CANopen je všechny definuje, a to spolu s nezbytnými komunikačními objekty. Komunikační objekty jsou zařazeny v tzv. slovníku objektů (Object Dictionary) uloženém v zařízení, které je součástí sítě, a slouží jako rozhraní mezi samotným zařízením a aplikačním programem.

PDO
Proces Data Object jsou komunikační objekty přenášející časově kritická technologická data. Jsou přenášeny v jedné zprávě CAN s využitím všech jejich osmi bajtů v poli dat. Každý PDO má unikátní identifikátor a je vyslán pouze jediným uzlem sítě, přičemž může být přijat libovolným počtem zařízení. Vyslání zprávy s PDO může být vyvoláno vnitřní/vnější událostí nebo požadavkem.

SDO
Service Data Object jsou komunikační objekty nesoucí servisní data. Tyto komunikační objekty umožňují číst a zapisovat jednotlivé položky do slovníku objektů. Protokol pro přenos SDO dovoluje přenášet objekty libovolné délky. Každý objekt je charakterizován šestnáctibitovým indexem, případně osmibitovým subindexem.

NMT
Network Management object jsou komunikační objekty pro správu sítě. Slouží ke konfigurování a řízení provozu sítě s protokolem CANopen (hlášení stavu zařízení, povel k novému spuštění zařízení apod.) a mají v síti největší prioritu.

NMT-služby
NMT-služby jsou komunikační rámce zajišťující inicializaci sítě a dohled nad sítí pomocí dohlížecích rámců (NODE GUARDING). NMT-služby podporují NMT-MASTER a NMT-uzel. Na úrovni NMT-služeb se jednotlivé uzly rozlišují na jediný NMT-MASTER a ostatní NMT-SLAVE.

Datové služby
Datové služby jsou datové rámce zajišťující přenos dat. Datové služby podporují všechny zařízení na síti. Na úrovni datových služeb se uzly nerozlišují a všechny jsou si rovnocenné.

NODE GUARDING
NODE GUARDING je NMT-služba sítě, sloužící jednak k tomu, aby NMT-MASTER detekoval poruchy NMT-uzlů a za druhé k tomu, aby NMT-uzel detekoval ztrátu spojení s MASTERem (popřípadě výpadek NMT-MASTERa) a přešel do autonomního režimu.
Provádí se tím způsobem, že NMT-MASTER se pravidelně dotazuje na stav NMT-uzlu a porovnává ho se stavem, ve kterém by NMT-uzel měl být. Zjistí-li rozdíl, provede kompletní inicializaci NMT-uzlu.
Je-li zvolen aktivní NODE GUARDING, NMT-uzel sleduje, jestli od NMT-MASTERa pravidelně přicházejí dotazy na stav a pokud takovýto dotaz nepřijde během doby dané násobkem Guard

Time a Lifetime Factor, přejde do autonomního režimu a zároveň do stavu "connecting" (v tomto stavu čeká na inicializaci).

NMT-MASTER
Na síti DIOCAN vždy existuje pouze jedno zařízení typu NMT-MASTER. Toto je MASTERem sítě a spravuje celou komunikaci (řídí NMT-služby). Datové služby podporuje stejně jako všechny ostatní zařízení na síti. Pokud se stane, že je vyřazeno z činnosti, síť se rozpadne, a to i tehdy, pokud je přítomno více zařízení NMT-SLAVE schopných aktivně vysílat. NMT-MASTER může být jakýkoliv řídicí systém firmy AMiT, který má rozhraní CAN.

NMT-SLAVE
Zařízení na síti DIOCAN, které při MULTIMASTER komunikaci sice plní roli MASTERa sítě, ale je závislé na zařízení, které je nakonfigurováno jako NMT-MASTER.

NMT-uzel (uzel)
Zařízení na síti DIOCAN, které poskytuje NMT-služby, tj. odpovídá na NMT rámce NMT-MASTERA a je inicializované pomocí NMT-služeb. Datové služby může poskytovat v plném rozsahu, tj. může i samo vysílat data ostatním uzlům, nebo omezeně tj. pouze odpovídá na dotaz buď NMT-uzlu nebo NMT-MASTERa. NMT-uzlem může být jakýkoliv řídicí systém firmy AMiT s rozhraním CAN, modul **ADC-CAN** s digitálními moduly systému **ADiS** (viz DODATEK B) nebo zařízení jiných výrobců (viz DODATEK C).

Virtuální uzel
Virtuální uzel je část NMT-uzlu se samostatnou adresou na síti. Neposkytuje NMT-služby, ale přebírá je z nadřazeného NMT-uzlu, kam je logicky zapojena. Datové služby poskytuje ve stejném rozsahu jako nadřazený NMT-uzel. Virtuální uzel se definuje pro rozšíření I/O prostoru. Datové rámce pro komunikaci s jedním uzlem mají maximální délku 64 bitů pro DO a 64 bitů pro DI, tj. maximálně obslouží 64 digitálních vstupů a 64 digitálních výstupů. Každý virtuální uzel rozšiřuje velikost dat o dalších 64 bitů pro DO a 64 bitů pro DI.
Aby bylo možno rozšířit velikost bloku dat vysílaných a přijímaných z a do uzlu, je třeba vytvořit tzv. virtuální uzel, který umožní přenášení dalších osmi bajtů dat.
Neexistuje omezení, kolik maximálně virtuálních uzlů je možno na jednom klasickém uzlu vytvořit, ale při jejich vytváření je nutné zabezpečit, aby jejich adresy nekolidovaly s adresami jiných fyzických uzlů.

COB-Id
Identifikátor zprávy (komunikačního objektu). Definuje implicitně její prioritu na sběrnici.

SW modul
Jedná se o modul z knihoven návrhového prostředí DetStudio.

2. Úvod

Komunikační rozhraní CAN (Controller Area Network) je díky svým vlastnostem vhodné pro použití v průmyslových aplikacích. CAN byl vyvinut firmou Bosch na technologii Intel původně pro potřeby automobilového průmyslu. Pro svou otevřenost, relativně vysokou rychlost a podporu mnohými výrobci mikroelektronických součástek, kteří implementují protokol do svých mikrokontrolérů, je velmi rozšířen. Základní verze CAN 2.0 získala podobu v roce 1991. Vývojem se modifikovala do dvou navzájem kompatibilních systémů 2.0A a 2.0B. V roce 1993 byl CAN přenesen do mezinárodního standardu ISO 11898.

Dvě nejnižší vrstvy sedmivrstvého ISO/OSI modelu jsou zajišťovány obvodově speciálními řadiči. Třetí až šestá vrstva jsou vynechány (jak je u jednoduchých lokálních sběrnic obvyklé). Pro sedmou komunikační vrstvu sběrnice CAN existuje několik úrovní standardizace, z nichž jednou je standard CANopen.

CANopen je vyšší komunikační protokol vytvořený na základě sběrnice CAN. Byl vyvinut jako široce konfigurovatelný standardní protokol pro vestavné řídicí sítě, po nichž komunikují stroje a zařízení s řízenými pohyblivými částmi, jako např. manipulátory. V současné době je využíván v mnoha rozličných odvětvích v průmyslu, v lékařské technice, automobilech, námořních systémech, ve veřejné dopravě, při automatizaci ve stavebnictví atd.

Firma AMiT implementovala protokol CANopen ve svých produktech a interně tento způsob komunikace pojmenovala jako DIOCAN.

Komunikační knihovna CAN (součást standardní instalace DetStudia) je postavena právě na standardu CANopen, což zajišťuje možnost propojení se širokou škálou zařízení jak z produkce firmy AMiT, tak i jiných výrobců. Podmínkou připojitelnosti jakéhokoliv zařízení k řídicím systémům firmy AMiT, obsahujícím řadič sběrnice CAN, je podpora standardu CANopen dotyčným zařízením, a to alespoň na úrovni Minimum Capability Device.

Maximální počet všech zařízení připojených do jedné komunikační sítě je 32 (doporoučení ohledně počtu zařízení připojených do komunikační sítě DIOCAN naleznete v aplikační poznámce AP0029 – Zásady používání sítě CAN), čímž lze dosáhnout (s použitím zařízení vyráběných firmou AMiT) rozšíření řídicího systému o dalších 7936 číslicových vstupů a 7936 číslicových výstupů. V **dodatku B** naleznete přehled dostupných zařízení firmy AMiT umožňujících komunikaci na síti DIOCAN. V **dodatku C** jsou uvedeny výrobky ostatních firem umožňující komunikaci na síti DIOCAN.

Síť DIOCAN používá k přenosu sériový protokol a z toho vyplývají jistá omezení. Při vzrůstajícím počtu připojených zařízení narůstají časové nároky na přenos dat z a do modulů a tomu odpovídá i prodlužující se perioda možné komunikace s připojenými moduly.

3. Komunikační síť DIOCAN

3.1. Služby sítě DIOCAN

V komunikační síti DIOCAN jsou definovány následující komunikační služby:

- NMT-služby,
- Datové služby,
- NODE GUARDING.

Vysvětlení jednotlivých pojmů naleznete v kapitole 1.

3.2. Autonomní režim

Do autonomního režimu přechází NMT-uzel v těchto případech:

- Po zapnutí zařízení až do dokončení inicializace.
- Při kritické chybě sběrnice CAN, detekované řadičem.
- Při selhání NODE GUARDING, byl-li tento aktivován.

Účelem autonomního režimu je zabránit tomu, aby při ztrátě komunikace NMT-uzel držel na svých výstupech naposledy přijatou hodnotu, která může být při trvalé aktivaci nebezpečná (např. signál pro chod serva, otevírání ventilu apod.) Proto NMT-uzel při přechodu do autonomního režimu uvádí všechny své výstupy do definovaného "bezpečného stavu".

Bezpečný stav je na modulech **ADC-CAN** definován jako nulová úroveň (popř. rozepnuté kontakty relé) na všech výstupech. Projekt měření a regulace musí být navržen tak, aby při trvající nulové hodnotě na výstupech nedošlo ke škodám v řízené technologii (servo stojí, ventil se podle požadavků dané technologie buďto zavírá nebo zůstává na naposledy nastaveném stupni otevření).

3.3. Komunikační zařízení sítě DIOCAN

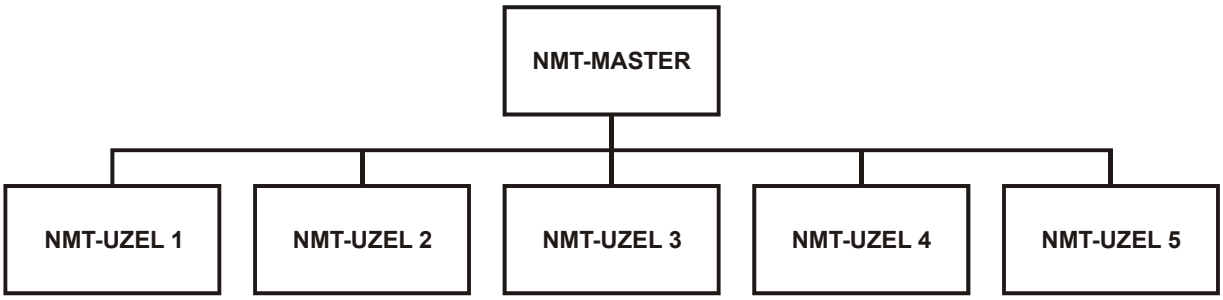
Protokol sítě DIOCAN definuje následující zařízení v komunikační síti:

- NMT-MASTER,
- NMT-uzel,
- Virtuální uzel.

Vysvětlení jednotlivých pojmů naleznete v kapitole 1.

3.4. Komunikace na úrovni NMT-služeb

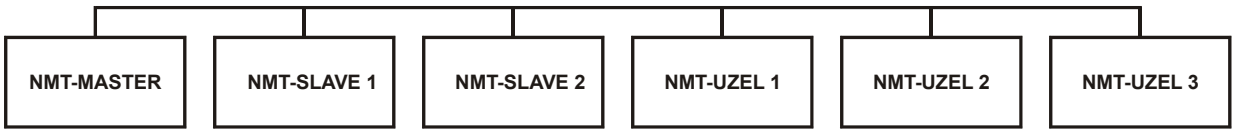
Jedno zařízení na síti je NMT-MASTER (řídicí systém naprogramovaný v prostředí DetStudio), ostatní jsou NMT-uzly (uzly s připojenými V/V moduly). NMT-MASTER jako jediné zařízení na síti řídí NMT-služby, tj. inicializuje síť NMT-uzlů a posílá rámce NODE GUARDING.



Obr. 1 - Hierarchie jednotek na úrovni NMT-služeb

3.5. Komunikace na úrovni datových služeb

Na úrovni datových služeb se již nerozlišuje NMT-MASTER a NMT-uzel a všechna zařízení jsou si rovna, tj. komunikaci může zahájit jakékoliv zařízení bez omezení.



Obr. 2 - Hierarchie jednotek na úrovni datových služeb

3.6. HW prostředky pro realizaci sítě DIOCAN

ADC-CAN
Pokud jako NMT-uzel použijeme modul **ADC-CAN**, lze k takovémuto NMT-uzlu přiřadit až 3 virtuální uzly, tj. maximálně lze obsloužit až 256 digitálních signálů na jednom **ADC-CAN**.

Na jeden **ADC-CAN** lze fyzicky zapojit maximálně 16 jednotek DI/DO. Pokud použijeme osmibitové moduly, lze na jednom **ADC-CAN** obsloužit maximálně 128 I/O v libovolné kombinaci DI/DO, pokud použijeme šestnáctibitové moduly, lze na jednom **ADC-CAN** obsloužit až 256 I/O v libovolné kombinaci DI/DO.

Uzel s 256 DI
Maximální sestavu 256 DI vytvoříme připojením 16 modulů **AD-DI16A** k jednomu modulu **ADC-CAN**. V tomto případě bude tato sestava představovat v síti DIOCAN jeden NMT-uzel se třemi virtuálními uzly.

Uzel s 256 DO
Maximální sestavu 256 DO vytvoříme připojením 16 modulů **AD-DO16** k jednomu modulu **ADC-CAN**. V tomto případě bude tato sestava představovat v síti DIOCAN jeden NMT-uzel se třemi virtuálními uzly.

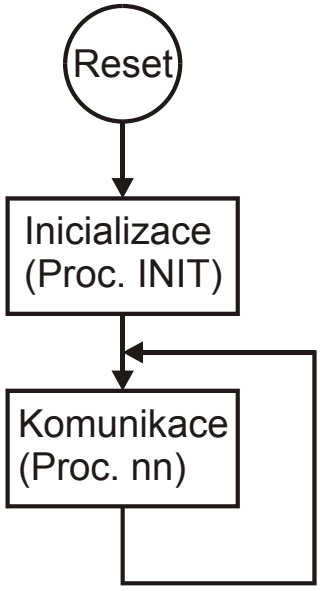
Při použití osmibitových modulů **AD-DI8A** a **AD-PDO8** je maximálně dosažitelný počet I/O na jednom modulu **ADC-CAN** 128 I/O.

3.7. Doporučený postup realizace sítě DIOCAN

Pro správnou funkci celé sítě je nutno správně nastavit všechny požadované parametry a správně zapojit celou síť DIOCAN. Bližší informace o zapojení sítě naleznete v aplikační poznámce AP0029 – Zásady používání sítě CAN.

4. Časové poměry

Při startu programu, který má komunikovat s rozšiřujícími moduly prostřednictvím sítě DIOCAN, je nutné celou síť inicializovat a nakonfigurovat. Při inicializaci řídicí systém posílá do jednotlivých uzlů na síti rámce dat, které obsahují informace o parametrech komunikace, a čeká na jejich odpověď. Pokud odpověď nedostane do uplynutí stanovené doby, prohlásí, že uzel není připojen a pokračuje dalším modulem. Celou inicializaci je nutné provést programově v procesu typu Init (popsáno dále). Po ní se již může odehrávat vlastní komunikace s moduly, která probíhá obvykle v periodickém procesu tak, jak ukazuje stavový diagram.



Obr. 3 - Stavový diagram programu

4.1. Doba inicializace sítě

4.1.1 Doba inicializace úplné sítě

Doba inicializace sítě je čas, za který řídicí systém rozešle všem uzlům komunikační parametry a obdrží ode všech potvrzení. Úplnou sítí se rozumí taková síť, kde všechny uzly, které se v programu inicializují, jsou fyzicky na síti přítomny a správně fungují (je připojena komunikační linka, napájení, uzly jsou správně adresovány a mají nastavenou správnou komunikační rychlost). Doba inicializace sítě je přímo úměrná počtu inicializovaných uzlů a nepřímo úměrná komunikační rychlosti. Následující tabulka udává dobu nutnou k inicializaci sítě pro různý počet připojených uzlů při nejčastěji používaných komunikačních rychlostech:

Doba inicializace

Přenosová rychlost	**Přibližná doba inicializace sítě** [ms]
20 kbit/s	T = 70 × n + 1020
125 kbit/s	T = 30 × n + 1030
500 kbit/s	T = 30 × n + 1030

4.1.2 Doba inicializace neúplné sítě

Pokud se stane, že některé uzly, které jsou v programu inicializovány, nejsou na síti fyzicky přítomny nebo není připojen síťový kabel, napájení nebo je špatně zvolena adresa nebo komunika-

ční rychlost, doba inicializace sítě se prodlouží o určitý čas za každý nedostupný uzel tak, jak je uvedeno v následující tabulce:

Doba inicializace

Přenosová rychlost	**Doba, o kterou se prodlouží inicializace při jednom chybějícím uzlu** [ms]
20 kbit/s	1000 ms
125 kbit/s	1050 ms
500 kbit/s	1100 ms

Příklad:
Při komunikační rychlosti 125 kbit/s a sedmi připojených uzlech bude doba inicializace trvat cca 1240 ms. Pokud u tří uzlů nebude zapojeno napájení, inicializace se prodlouží o 3150 ms na téměř 4,4 s.

4.2. Výpočet minimální periody komunikace s moduly

Doba komunikace

Doba, za kterou proběhne právě jedna komunikace se všemi uzly, je závislá na objemu komunikovaných dat a komunikační rychlosti. Je nutno počítat s rezervou pro komunikace, které mohou selhat např. vlivem nadměrného elektromagnetického rušení nebo jiných vlivů. V tabulce uvádíme maximální hodnoty pro výpočet maximální doby odezvy v nezarušené síti. Pro přesný výpočet doby přenosu kontaktujte technickou podporu firmy AMiT.

Doba komunikace

Přenosová rychlost	**Maximální doba komunikace** [ms]
20 kbit/s	18 × U
125 kbit/s	6 × U
500 kbit/s	3 × U

U – Součet všech uzlů na síti (fyzických i virtuálních)

Minimální perioda
Jako minimální periodu komunikace s moduly doporučujeme hodnotu

T = t × 2

T minimální perioda pro komunikaci s moduly
t maximální doba komunikace s moduly

Takto vypočtená perioda komunikace je téměř nejhorší možný stav. Počítá s periodickou komunikací se všemi moduly včetně těch vstupních. Optimalizací dotazů a využitím aktivního zasílání vstupů pouze při změnách lze dosáhnout výrazně kratších odezev celé komunikační sítě. Pokud bude potřeba síť optimalizovat na maximální propustnost, kontaktujte technickou podporu firmy AMiT.

Příklad:
Na síti jsou obsluhovány 3 NMT-uzly a 2 virtuální uzly, konfigurace je následující:

1.	NMT-uzel	8 × **AD-DI8A**	64 DI
		8 × **AD-PDO8**	64 DO
2.	NMT-uzel	4 × **AD-DI16A**	64 DI
		4 × **AD-DO16**	64 DO
	1. virtuální uzel	4 × **AD-DI16A**	64 DI
		4 × **AD-DO16**	64 DO

3.	NMT-uzel	4 × **AD-DI16A**	64 DI
		4 × **AD-DO16**	64 DO
	2. virtuální uzel	4 × **AD-DI16A**	64 DI
		4 × **AD-DO16**	64 DO

Maximální doba komunikace se všemi I/O při rychlosti 20 kbit/s

t = 18 × 5 = 90 ms

Minimální doporučená perioda komunikace s moduly

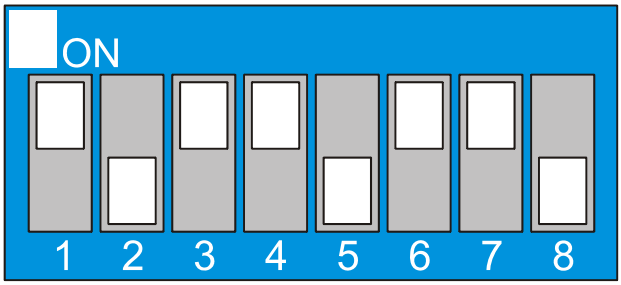
T = 90 × 2 = 180 ms

4.3. Doba první odezvy sítě

Je to časová prodleva od startu programu (a tedy začátku inicializace sítě) do proběhnutí první komunikace. Je to minimální doba, za kterou je možno získat ze sítě platná data. Doba první odezvy je dána součtem doby inicializace sítě a periody komunikace s moduly.

5. Konfigurace HW uzlů

Každý uzel musí mít svou adresu, která je na síti jedinečná, a musí komunikovat stejnou rychlostí jako všechny ostatní uzly. Nastavení těchto parametrů se provádí na modulu **ADC-CAN** přepínači označenými SW.



Obr. 4 - Ukázka nastavení komunikačních parametrů

5.1. Nastavení adresy uzlu

Přepínače 1 .. 5 slouží k nastavení adresy uzlu na sběrnici. Teoreticky je tedy možno adresovat až 32 zařízení (doporoučení ohledně počtu připojených zařízení do komunikační sítě DIOCAN naleznete v aplikační poznámce AP0029 – Zásady používání sítě CAN). Kvůli možnosti vytváření virtuálních uzlů jsou skutečné adresy uzlů přiřazeny jednotlivým stavům DIP přepínačů takto:

Nastavení adresy uzlu

		DIP4	OFF	ON	OFF	ON
		DIP5	OFF	OFF	ON	ON
DIP1	**DIP2**	**DIP3**	**Skutečná adresa modulu**			
OFF	OFF	OFF	1	33	65	97
ON	OFF	OFF	5	37	69	101
OFF	ON	OFF	9	41	73	105
ON	ON	OFF	13	45	77	109
OFF	OFF	ON	17	49	81	113
ON	OFF	ON	21	53	85	117
OFF	ON	ON	25	57	89	121
ON	ON	ON	29	61	93	125

U starších provedení modulu **ADC-CAN**, vybavených firmware 2.20[(1)] a nižší je tabulka přiřazení adres modulů tato:

Nastavení adresy uzlu u starších provedení

		DIP4	OFF	ON	OFF	ON
		DIP5	OFF	OFF	ON	ON
DIP1	**DIP2**	**DIP3**	**Skutečná adresa modulu**			
OFF	OFF	OFF	1	17	33	49
ON	OFF	OFF	3	19	35	51
OFF	ON	OFF	5	21	37	53
ON	ON	OFF	7	23	39	55
OFF	OFF	ON	9	25	41	57
ON	OFF	ON	11	27	43	59
OFF	ON	ON	13	29	45	61
ON	ON	ON	15	31	47	63

Poznámka

(1) Verzi firmware lze zjistit ze samolepícího štítku na procesoru modulu. Pokud štítek zcela chybí, jedná se o modul s verzí firmware 2.20 nebo nižší.

Příklad
Za předpokladu, že jsou přepínače nastaveny v polohách jako na obrázku 4, je skutečná adresa pro modul s novějším firmwarem 53 a pro modul se starším firmwarem 27.

5.2. Nastavení komunikační rychlosti uzlu

Pro nastavení komunikační rychlosti slouží jednotlivým uzlům přepínače 6 .. 8, kde jednotlivé stavy přepínačů odpovídají takto podporovaným komunikačním rychlostem:

Nastavení komunikačních rychlostí

Nastavení DIP přepínačů	**DIP6**	OFF	ON	OFF	ON	OFF	ON	OFF	ON
	DIP7	OFF	OFF	ON	ON	OFF	OFF	ON	ON
	DIP8	OFF	OFF	OFF	OFF	ON	ON	ON	ON
Kom. rychlost [kb/s]		20	50	100(2)	125	250	500	800(2)	1000

Poznámka
(2) Tuto rychlost nemusí podporovat některá novější zařízení. Nedoporučuje se ji používat.

Příklad
Za předpokladu, že jsou přepínače nastaveny v polohách jako na obrázku 4, je zvolena komunikační rychlost 125 kb/s.

5.3. Význam LED

Všechny moduly firmy AMiT umožňující komunikaci po sběrnici CAN jsou vybaveny indikačními LED, které umožňují vizuální kontrolu činnosti. V následujících tabulkách jsou uvedeny popisy jejich funkce:

Komunikační LED

Modul	LED	Funkce
Všechny moduly	**Rx**	Svítí, pokud modul přijímá data.
	Tx	Svítí, pokud modul vysílá data.

Stavy komunikace na sběrnici CAN

Stav LED ERR	MASTER	SLAVE
Trvale zhasnutá	Komunikace na sběrnici CAN v pořádku, všechny uzly (typu SLAVE) uvedené v konfiguraci jsou úspěšně inicializovány.	
Bliká 50 % - 50 %	Ztráta spojení s některým SLAVEm uvedeným v konfiguraci.	Ztráta spojení s MASTERem.
Bliká 20 % - 80 %	Nevyskytuje se.	Spojení s MASTERem v pořádku, uzel zinicializován, ale spojení s některým z ostatních SLAVEů uvedených v konfiguraci se ztratilo.
Trvale svítí	Řadič sběrnice detekoval kritickou chybu na sběrnici CAN (např. zkrat), vůbec se nedaří vysílat rámce na sběrnici. Pozn.: Ne každý zkrat na sběrnici je takto řadičem detekován.	

Systémové LED modulu AD-CAN

<table>
<tr><th>Modul</th><th>LED</th><th>Funkce</th></tr>
<tr><td>Všechny moduly</td><td>ST</td><td>Svítí, pokud procesor komunikuje s modulem/moduly.</td></tr>
<tr><td></td><td>PWR</td><td>Svítí, pokud je připojeno napájecí napětí.</td></tr>
<tr><td>MASTER</td><td>RDY</td><td>
<table>
<tr><th>Stav</th><th>Akce</th></tr>
<tr><td>Trvale svítí</td><td>Komunikace probíhá v pořádku, všechny uzly uvedené v konfiguraci jsou úspěšně zinicializovány.</td></tr>
<tr><td>Bliká
50 % - 50 %</td><td>Ztráta spojení s některým uzlem, uvedeným v konfiguraci.</td></tr>
<tr><td>Trvale zhasnutá</td><td>Řadič sběrnice detekoval kritickou chybu na sběrnici CAN (např. zkrat) a vůbec se nedaří vysílat rámce na sběrnici CAN.</td></tr>
</table>
</td></tr>
<tr><td>SLAVE</td><td>RDY</td><td>
<table>
<tr><th>Stav</th><th>Akce</th></tr>
<tr><td>Trvale svítí</td><td>Komunikace probíhá v pořádku, všechny uzly uvedené v konfiguraci jsou úspěšně zinicializovány.</td></tr>
<tr><td>Bliká
50 % - 50 %</td><td>Ztráta spojení s MASTERem.</td></tr>
<tr><td>Bliká
80 % - 20 %</td><td>Spojení s MASTERem v pořádku, uzel zinicializován, ale nastala ztráta spojení s některým z ostatních SLAVEů.</td></tr>
<tr><td>Trvale zhasnutá</td><td>Řadič sběrnice detekoval kritickou chybu na sběrnici CAN (např. zkrat) a vůbec se nedaří vysílat rámce na sběrnici CAN.</td></tr>
</table>
</td></tr>
</table>

Systémové LED modulu ADC-CAN

<table>
<tr><th>Modul</th><th>LED</th><th>Funkce</th></tr>
<tr><td>Všechny moduly</td><td>PWR</td><td>Svítí, pokud je připojeno napájecí napětí.</td></tr>
<tr><td>ADC-CAN</td><td>ST</td><td>
<table>
<tr><th>Stav</th><th>Akce</th></tr>
<tr><td>Trvale svítí</td><td>Komunikace probíhá v pořádku, všechny uzly uvedené v konfiguraci jsou úspěšně zinicializovány.</td></tr>
<tr><td>Bliká
50 % - 50%</td><td>Ztráta spojení s MASTERem.</td></tr>
</table>
</td></tr>
<tr><td>Ostatní moduly</td><td>ST</td><td>Svítí, pokud procesor modulu ADC-CAN komunikuje s modulem/moduly.</td></tr>
</table>

6. Programová obsluha uzlů s ADC-CAN

Pro komunikaci v síti DIOCAN se řídicí systém konfiguruje v prostředí DetStudio prostřednictvím SW modulů knihovny CAN.

NODE GUARDING
Správné nastavení NODE GUARDING může podstatným způsobem ovlivnit správnou funkci celé sítě. Nastavení NODE GUARDING by mělo vycházet především z požadavků technologie. Hodnota se nastavuje podle toho, za jak dlouho se požaduje, aby se výstupy v případě problémů s komunikací nastavily na bezpečné hodnoty.

6.1. Komunikace MASTER-SLAVE

V následujících odstavcích je popsána komunikace MASTER-SLAVE (jedno zařízení je MASTER a ostatní jsou SLAVE).

6.1.1 Inicializace sítě

Konfigurace (a samotné sestavení) komunikační sítě probíhá vždy na začátku v procesu typu Init. K tomu slouží následující moduly:

Moduly sloužící k konfiguraci komunikační sítě

Modul	Řídicí systém
CNC_ADCAN	**ADiS**
CNC_ADOS	**ADOS**
CNC_AMP99	**AMAP99**
CNC_AMR99	**AMiRiS99**
CNC_APT2k	**APT2x00**
CNC_ART4k	**ART4000**
CNC_APT3K2	**APT3200T**
CNC_C167	Všechny ŘS s interním řadičem na procesoru C167
CNC_ST10	Všechny ŘS s interním řadičem na procesoru ST10

Těmito SW moduly se definuje globální nastavení parametrů sítě. Modulem **CAN_NMT_M** je určeno, že zařízení, na němž poběží program, je NMT-MASTERem sítě DIOCAN. Jednotlivé uzly jsou následně definovány SW moduly **CAN_Node**.

Pozor!
Maximální počet všech zařízení připojených na sběrnici CAN je 32!

Příklad
Vytvoříme síť DIOCAN složenou z pěti uzlů a jednoho zařízení MASTER. MASTERem je řídicí systém **ADiS**, v němž je modul **AD-CAN**(1) připojen jako třetí v řadě (za moduly **AD-FDI8** a **AD-PDO8**). Použijeme tedy SW modul **CNC_ADCAN**. Proces typu Init s názvem ProcINIT pak bude vypadat takto:

```
:10000 CNC_ADCAN 2, 4
:10100 CAN_NMT_M :10000
       CAN_Node :10100, 1, 333, 3, Stav.1, 0, 0
       CAN_Node :10100, 5, 333, 3, Stav.2, 0, 0
       CAN_Node :10100, 9, 333, 3, Stav.3, 0, 0
       CAN_Node :10100, 13, 333, 3, Stav.4, 0, 0
       CAN_Node :10100, 17, 333, 3, Stav.5, 0, 0
```

Síť DIOCAN je nyní naparametrizována pro obsluhu pěti uzlů, jejichž fyzické adresy jsou 1, 5, 9, 13 a 17 s komunikační rychlostí 125 kb/s. Je zaručeno, že všechny uzly detekují ztrátu spojení a nastaví se do bezpečného stavu do 999 ms od rozpojení sítě. Aktuální stav spojení s každým uzlem je uložen v příslušném bitu proměnné `Stav`.

V příkladu jsou barevně označena návěští, která spolu souvisejí.

Pozor!
Vlastní komunikaci není možno zahájit dříve, než proběhne celá inicializace sítě!

Poznámka

(1) V DetStudiu se změnila definice I/O prostoru modulárního řídicího systému ADiS (IO konfigurace). Oproti PSP3 probíhá kontrola správnosti zapojení modulů a není již možné v prostředí volně definovat pozici modulu v sestavě. Jelikož probíhá kontrola zapojení, je vyžadována stoprocentní shoda definice v DetStudiu se skutečnou sestavou. V aplikacích, které modul AD-CAN obsahují, se musí v sestavě tento modul doplnit dle skutečného umístění (v PSP 3 modul nebyl vůbec definován a jen se nechávalo volné místo v řazení sestavy). Je-li modul AD-CAN posledním modulem sestavy, není potřeba tento modul definovat.

6.1.2 Periodická komunikace

Vlastní komunikaci můžeme zahájit, až proběhne inicializace sítě. Podle způsobu, jakým je tato podmínka vyhodnocena, rozlišujeme komunikaci na kontrolovanou a nekontrolovanou.

Kontrolovaná komunikace
Tato komunikace je povolena až v okamžiku, kdy se stavový bit `Fail` SW modulu **CAN_Node** nastaví na hodnotu 0 a tím je potvrzeno, že spojení s modulem bylo skutečně úspěšně navázáno a modul je připraven ke komunikaci. Při větším počtu modulů je nutno testovat před začátkem komunikace stavové bity všech nevirtuálních uzlů definovaných SW moduly **CAN_Node**. Stavový bit `Fail` SW modulu **CAN_Node** nemá u virtuálních uzlů význam. Komunikaci s virtuálními uzly lze kontrolovat pomocí stavového bitu `Fail` SW modulu **CAN_Node** nevirtuálního uzlu ke kterému virtuální uzly náleží.

Testováním stavových bitů `Fail` těchto modulů i v průběhu programu lze detekovat ztrátu spojení s modulem a tedy poruchu.

Nekontrolovaná komunikace
Stavové bity `Fail` jednotlivých SW modulů **CAN_Node** nejsou testovány a komunikace je zahájena od uplynutí určitého časového intervalu od začátku inicializace (je potřeba řešit správným nastavením ofsetu periodického procesu viz nápověda DetStudia – Správce procesů).

U tohoto způsobu komunikace je nutno ovšem počítat při volbě velikosti offsetu procesu také s možností, že některé uzly nebudou připojeny. Pokud nejsou kontrolovány stavové bity SW modulů **CAN_Node**, nelze detekovat ztrátu spojení – poruchu.

Doporučení
Pro celkovou přehlednost a lepší funkčnost programu je vhodné zapisovat a číst z/do všech uzlů v jednom periodickém procesu (s výjimkou uzlů, jejichž perioda komunikace se výrazně liší).

Příklad
Máme síť DIOCAN složenou z pěti zařízení SLAVE (viz tabulka níže) a jednoho zařízení MASTER. Síť byla zapojena a komunikace inicializována v předchozí kapitole.

Zapojení jednotlivých zařízení (uzlů)

	Uzel 1	Uzel 2	Uzel 3	Uzel 4	Uzel 5
Seznam modulů v každém uzlu	PDO8a	DI8Aa	PDO8	DI16A	DO16
	PDO8b	DI8Ab	DI8A	DI16A	
	PDO8c	DI8Ac	DO16	PDO8	
	PDO8d	DI8Ad	DI16A		

Komunikace bude umístěná v řádném periodickém procesu s názvem Proc00 a bude vypadat takto:

```
//První uzel
CAN_DO :10000, 1, 1, Prenos.1, 4, 0, N1DO[0,0]
//Druhý uzel
CAN_DI :10000, 5, 1, Prenos.2, 4, 0, N2DI[0,0]
//Třetí uzel
CAN_DI :10000, 9, 1, Prenos.3, 3, 0, N3DI[0,0]
CAN_DO :10000, 9, 1, Prenos.13, 3, 0, N3DO[0,0]
//Čtvrtý uzel
CAN_DI :10000, 13, 1, Prenos.4, 4, 0, N4DI[0,0]
CAN_DO :10000, 13, 1, Prenos.14, 1, 0, N4DO[0,0]
//Pátý uzel
CAN_DO :10000, 17, 1, Prenos.5, 2, 0, N5DO[0,0]
```

Před voláním tohoto bloku kódu musí být do matic `NxDO` vloženy požadované hodnoty, na které mají být nastaveny číslicové výstupy modulů. Po volání jsou v maticích `NxDI` uloženy hodnoty, které odpovídají stavům číslicových vstupů modulů, viz následující tabulka.

Prvek matice	**1.**		**2.**	
Jméno matice	**Vyšší Byte**	**Nižší byte**	**Vyšší byte**	**Nižší byte**
N1DO	PDO8b	PDO8a	PDO8d	PDO8c
N2DI	DI8Ab	DI8Aa	DI8Ad	DI8Ac
N3DI	DI16AL	DI8A		DI16AH
N3DO	DO16L	PDO8		DO16H
N4DI	DI16Aa		DI16Ab	
N4DO		PDO8		
N5DO	DO16			

Symboly H a L znamenají dolní, resp. horní byte u modulu se šestnácti vstupy nebo výstupy. Z tabulky je zřejmé, že nejdříve jsou obsazovány nižší bajty proměnných.

Pozor!
Při parametrizaci komunikace není možno použít následující konstrukci:

```
//Digitální výstupy DIOCAN
If @Priznak1, :NONE
  CAN_DO :10000, 1, 1, Stav.0, 1, 0, Vystup1A
Else :NONE
  CAN_DO :10000, 1, 1, Stav.0, 1, 0, Vystup1B
EndIf
If @Priznak2, :NONE
  CAN_DO :10000, 1, 1, Stav.1, 2, 1, Vystup2A
Else :NONE
  CAN_DO :10000, 1, 1, Stav.1, 2, 1, Vystup2B
EndIf
```

Její použití způsobí, že nezávisle na hodnotě podmínky jsou vždy vyvolávány oba SW moduly **CAN_DO** v dané podmínce. Není možné použít 2 nebo více SW modulů **CAN_DO** zapisujících

do stejného uzlu na stejnou pozici, každý s jinou proměnnou. Stejné omezení platí i u SW modulů **CAN_DI**.

Předchozí příklad je tedy nutno modifikovat takto:

```
//Digitální výstupy DIOCAN
If @Priznak1, :NONE
  Let Vystup1 = Vystup1A
Else :NONE
  Let Vystup1 = Vystup1B
EndIf
CAN_DO :10000, 1, 1, Stav.0, 1, 0, Vystup1
If @Priznak2, :NONE
  Let Vystup2 = Vystup2A
Else :NONE
  Let Vystup2 = Vystup2B
EndIf
CAN_DO :10000, 1, 1, Stav.1, 2, 1, Vystup2
```

6.1.3 Zasílání dat z iniciativy SLAVEa

Čtení dat z číslicových vstupů SLAVE uzlů není nutno provádět v periodickém procesu a zbytečně zatěžovat linku redundantními daty. V okamžiku, kdy SLAVE vyhodnotí změnu na svých vstupech, odešle nová data do řídicího systému.

Komunikační SW moduly **CAN_DI** se v tomto případě neumisťují do periodického procesu, ale do procesu typu Init a do parametru `Transfer` se zadává hodnota 0.

```
:10000 CNC_ADCAN 0, 4
:11000 CAN_NMT_M :10000
       CAN_Node :11000, 1, 333, 3, NONE.0, 0, 0
       CAN_DI :10000, 1, 0, NONE.0, 8, 0, DI[0,0]
```

6.2. MULTIMASTER komunikace

Postup při inicializaci MULTIMASTER se liší od inicializace sítě typu MASTER-SLAVE. Na síti existuje více zařízení MASTER, z nichž jedno je nadřazeno všem ostatním (NMT-MASTER). V NMT-MASTERu se v procesu typu Init nadefinují pomocí SW modulů **CAN_Node** všechny uzly SLAVE, které jsou na síti přítomny a stejným způsobem (opět přes modul **CAN_Node**!) i všechny ostatní uzly MASTERů (tzv. NMT-SLAVE). Na ostatních zařízeních MASTER (NMT-SLAVE) je možno pomocí modulů **CAN_Node** sledovat pouze stav připojení daného uzlu pomocí parametru `Fail` (modul **CAN_Node** nedefinuje daný uzel). Na každém ze zařízení, které poskytuje data (ostatním zařízením), je nutno nadefinovat umístění těchto dat v PDO. K tomuto slouží SW moduly **CAN_S_DI** a **CAN_S_AI**. Maximální přípustná délka bloku dat (bez použití virtuálních uzlů) je 8 bajtů v každém směru.

Příklad
Mějme síť složenou ze dvou zařízení MASTER a dvou zařízení (uzlů) SLAVE. MASTERy jsou řídicí systémy **ADiS** s připojenými moduly **AD-CAN**(1), popis uzlů SLAVE je uveden v následující tabulce:

Soupis modulů jednotlivých uzlů

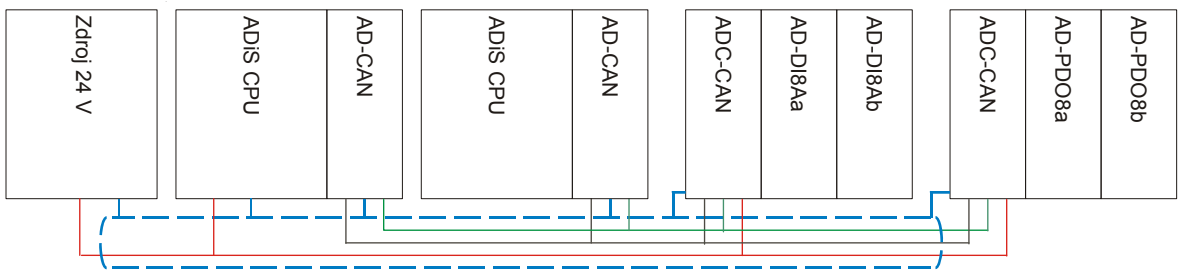
	Uzel 1	**Uzel 2**
Seznam modulů	DI8A	PDO8
	DI8A	PDO8

Poznámka

(1) V DetStudiu se změnila definice I/O prostoru modulárního řídicího systému ADiS (IO konfigurace). Oproti PSP3 probíhá kontrola správnosti zapojení modulů a není již možné v prostředí volně definovat pozici modulu v sestavě. Jelikož probíhá kontrola zapojení, je vyžadována stoprocentní shoda definice v DetStudiu se skutečnou sestavou. V aplikacích, které modul AD-CAN obsahují, se musí v sestavě tento modul doplnit dle skutečného umístění (v PSP 3 modul nebyl vůbec definován a jen se nechávalo volné místo v řazení sestavy). Je-li modul AD-CAN posledním modulem sestavy, není potřeba tento modul definovat.

Oba řídicí systémy MASTER musí umět komunikovat s oběma uzly SLAVE a zároveň si mezi sebou vyměňovat data.

V příkladech jsou barevně rozlišena související návěští a také jsou barevně rozlišeny proměnné, které při správné komunikaci musí mít stejné hodnoty na obou řídicích systémech.



Obr. 5 - Zapojení sítě v příkladu MULTIMASTER komunikace

Ztráta spojení s některým z uzlů SLAVE je indikována nastavením příslušného bitu proměnné `Stav(.0,.2)` v SW modulu **CAN_Node** do 1. V našem příkladu může NMT-SLAVE detekovat pouze ztrátu spojení s NMT-MASTERem, nikoli však už s uzly SLAVE.

U SW modulů **CAN_S_DO** a **CAN_DO** nastavení stavového bitu do 1 znamená, že data byla úspěšně vyslána na sběrnici.

U SW modulů **CAN_S_DI** a **CAN_DI** nastavení stavového bitu do 1 znamená, že byla úspěšně přijata data ze sběrnice.

Dvojice stavových bitů `Stav.3` + `Stav.4`, `Stav.5` + `Stav.6` jsou vzájemně ekvivalentní a je možno testovat vždy pouze jeden z nich.

Varianta 1 – SW modul CAN_S_DI

Program pro 1. řídicí systém

Proces ProcINIT typu Init

```
:10000 CNC_ADCAN 0, 4
:10100 CAN_NMT_M :10000
       //Adresu 5 má přidělenu druhý MASTER řídicí systém
       CAN_Node :10100, 5, 333, 3, Stav.0, 0, 0
       //Adresy 9 a 13 mají přiděleny uzly
       CAN_Node :10100, 9, 333, 3, Stav.1, 0, 0
       CAN_Node :10100, 13, 333, 3, Stav.2, 0, 0
```

Proces Proc00 typu Normal

```
       //Řídicí systém poskytuje tato data
       //Parametr Transfer je nastaven do 0, protože
       //o zaslání dat žádá druhý řídicí systém v modulech CAN_DI
       CAN_S_DI :10000, 1, 0, Stav.3, 2, 0, M1Out1
       CAN_S_DI :10000, 1, 0, Stav.4, 2, 2, M1Out2
```

```
        //Čtení dat z druhého řídicího systému
        CAN_DI :10000, 5, 1, Stav.5, 2, 0, M2In1
        CAN_DI :10000, 5, 0, Stav.6, 2, 2, M2In2
        //Čtení a zápis dat z/do uzlů SLAVE
        CAN_DI :10000, 9, 1, Stav.7, 2, 0, S1DI
        CAN_DO :10000, 13, 1, Stav.8, 2, 0, S2DO
```

Program pro 2. řídicí systém

Proces ProcINIT typu Init

```
:10000 CNC_ADCAN 0, 4
:10100 CAN_NMT_S :10000, 5, 333, 3, Stav.0, 0
```

Proces Proc00 typu Normal

```
        //Řídicí systém poskytuje tato data
        //Parametr Transfer je nastaven do 0, protože
        //o zaslání dat žádá první řídicí systém v modulech CAN_DI
        CAN_S_DI :10000, 5, 0, Stav.3, 2, 0, M2Out1
        CAN_S_DI :10000, 5, 0, Stav.4, 2, 2, M2Out2
        //Čtení dat z prvního řídicího systému
        CAN_DI :10000, 1, 1, Stav.5, 2, 0, M1In1
        CAN_DI :10000, 1, 0, Stav.6, 2, 2, M1In2
        //Čtení a zápis dat z/do uzlů SLAVE
        CAN_DI :10000, 9, 1, Prenos.7, 2, 0, S1DI
        CAN_DO :10000, 13, 1, Prenos.8, 2, 0, S2DO
```

V tomto příkladu jsou obě zařízení rovnocenná – obě mohou iniciovat přenos dat po sběrnici. Pokud bude první zařízení (NMT-MASTER) odpojeno od sběrnice, rozpadne se celá komunikace a ani druhé zařízení (NMT-SLAVE) již nebude schopné komunikovat s ostatními připojenými zařízeními (SLAVE). Připojené a funkční zařízení NMT-MASTER je pro fungování celé sítě nezbytné!

Varianta 2 – SW modul CAN_S_DO

Program pro 1. řídicí systém

Proces ProcINIT typu Init

```
:10000 CNC_ADCAN 0, 4
:10100 CAN_NMT_M :10000
        // Adresu 5 má přidělenu druhý MASTER řídicí systém
        CAN_Node :10100, 5, 333, 3, Stav.0, 0, 0
        //Adresy 9 a 13 mají přiděleny oba uzly
        CAN_Node :10100, 9, 333, 3, Stav.1, 0, 0
        CAN_Node :10100, 13, 333, 3, Stav.2, 0, 0
```

Proces Proc00 typu Normal

```
        // Řídicí systém poskytuje tato data
        CAN_DO :10000, 5, 0, Stav.3, 2, 0, M1Out1
        CAN_DO :10000, 5, 1, Stav.4, 2, 2, M1Out2
        //Čtení dat z druhého řídicího systému
        //Parametr Transfer je nastaven do 0, protože
        //o zápis dat je postaráno v druhém řídicím systému v modulech CAN_DO
        CAN_S_DO :10000, 1, 0, Stav.5, 2, 0, M2In1
        CAN_S_DO :10000, 1, 0, Stav.6, 2, 2, M2In2
        //Čtení a zápis dat z/do uzlů SLAVE
```

```
        CAN_DI :10000, 9, 1, Stav.7, 2, 0, S1DI
        CAN_DO :10000, 13, 1, Stav.8, 2, 0, S2DO
```

Program pro 2. řídicí systém

Proces ProcINIT typu Init

```
:10000 CNC_ADCAN 0, 4
:10100 CAN_NMT_S :10000, 5, 333, 3, Stav.0, 0
```

Proces Proc00 typu Normal

```
        // Řídicí systém poskytuje tato data
        CAN_DO :10000, 1, 0, Stav.3, 2, 0, M2Out1
        CAN_DO :10000, 1, 1, Stav.4, 2, 2, M2Out2
        //Čtení dat z prvního řídicího systému
        //Parametr Transfer je nastaven do 0, protože
        //o zápis dat je postaráno v prvním řídicím systému v modulech CAN_DO
        CAN_S_DO :10000, 5, 0, Stav.5, 2, 0, M1In1
        CAN_S_DO :10000, 5, 0, Stav.6, 2, 2, M1In2
        //Čtení a zápis dat z/do uzlů SLAVE
        CAN_DI :10000, 9, 1, Stav.7, 2, 0, S1DI
        CAN_DO :10000, 13, 1, Stav.8, 2, 0, S2DO
```

V tomto příkladu, stejně jako v předchozím, jsou obě zařízení rovnocenná – obě mohou iniciovat přenos dat po sběrnici. Pokud bude první zařízení (NMT-MASTER) odpojeno od sběrnice, rozpadne se celá komunikace a ani druhé zařízení (NMT-SLAVE) již nebude schopné komunikovat s ostatními připojenými zařízeními (SLAVE). Připojené a funkční zařízení NMT-MASTER je pro fungování celé sítě nezbytné!

Varianta 3 – SW modul CAN_S_DI a CAN_S_DO zároveň

Program pro 1. řídicí systém (NMT-MASTER)

Proces ProcINIT typu Init

```
:10000 CNC_ADCAN 0, 4
:10100 CAN_NMT_M :10000
        // Adresu 5 má přidělenu druhý MASTER řídicí systém
        CAN_Node :10100, 5, 333, 3, Stav.0, 0, 0
        //Adresy 9 a 13 mají přiděleny oba uzly
        CAN_Node :10100, 9, 333, 3, Stav.1, 0, 0
        CAN_Node :10100, 13, 333, 3, Stav.2, 0, 0
```

Proces Proc00 typu Normal

```
        // Řídicí systém poskytuje tato data
        CAN_DO :10000, 1, 0, Stav.3, 2, 0, M1Out1
        CAN_DO :10000, 1, 1, Stav.4, 2, 2, M1Out2
        //Čtení dat z druhého řídicího systému
        CAN_DI :10000, 5, 1, Stav.5, 2, 0, M2In1
        CAN_DI :10000, 5, 0, Stav.6, 2, 2, M2In2
        //Čtení a zápis dat z/do uzlů SLAVE
        CAN_DI :10000, 9, 1, Stav.7, 2, 0, S1DI
        CAN_DO :10000, 13, 1, Stav.8, 2, 0, S2DO
```

Program pro 2. řídicí systém (NMT-SLAVE)

Proces ProcINIT typu Init

```
:10000 CNC_ADCAN 0, 4
:10100 CAN_NMT_S :10000, 5, 333, 3, Stav.0, 0
```

Proces Proc00 typu Normal

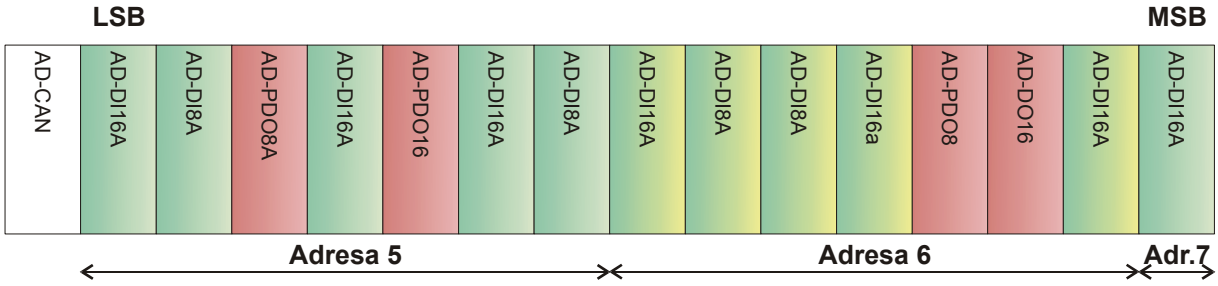
```
// Řídicí systém poskytuje tato data
CAN_S_DI :10000, 5, 1, Stav.3, 2, 0, M2Out1
CAN_S_DI :10000, 5, 0, Stav.4, 2, 2, M2Out2
//Čtení dat z prvního řídicího systému
CAN_S_DO :10000, 1, 0, Stav.5, 2, 0, M1In1
CAN_S_DO :10000, 1, 1, Stav.6, 2, 2, M1In2
//Čtení dat z/do uzlů SLAVE
CAN_DI :10000, 9, 1, Stav.7, 2, 0, S1DI
CAN_DO :10000, 13, 1, Stav.8, 2, 0, S2DO
```

Použití této konstrukce je nezbytné, pokud existuje omezení, že jedno nebo druhé zařízení může pracovat pouze jako NMT-MASTER nebo pouze jako NMT-SLAVE.

6.3. Komunikace s virtuálním uzlem

Virtuální uzel je nutno konfigurovat v případě, že je pro přenos dat z/do uzlu potřebných více než 8 bytů dat. Uzel muže obsahovat jeden až tři virtuální uzly s po sobě jdoucími adresami. Je tedy možno přenášet až 32 bytů dat a to v obou směrech. Komunikace poté probíhá analogicky jako by se jednalo o uzly fyzické.

Následující obrázek ukazuje případ, kdy je nutno konfigurovat dva virtuální uzly:

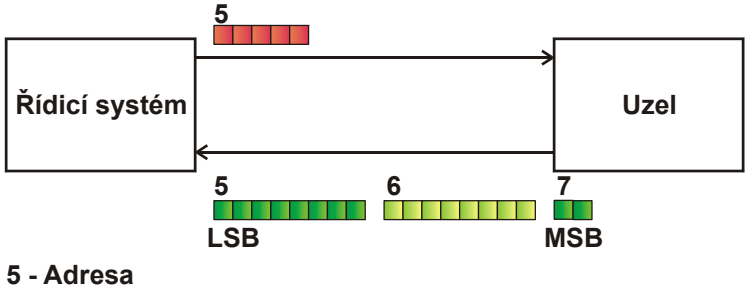


Obr. 6 - Příklad rozdělení modulů

Konfigurace těchto uzlů v prostředí DetStudio vypadá takto:

```
:10000 CNC_ADCAN 0, 4
:10100 CAN_NMT_M :10000
       CAN_Node :10100, 5, 333, 3, Stav.0, 0, 0
       CAN_Node :10100, 6, 333, 3, NONE.0, 5, 0
       CAN_Node :10100, 7, 333, 3, NONE.0, 5, 0
```

Data ze zde zapojených uzlů jsou přenášena takto:



Obr. 7 - Přenos dat

Data do zde zapojených modulů číslicových výstupů jsou všechna přenášena v jednom rámci. Při sestavování komunikačního rámce jednoho typu (DI, DO, AI nebo AO) se berou v potaz pouze moduly příslušného typu. Ostatní moduly, které leží mezi těmito moduly, jsou vždy ignorovány a neposouvají pozici za nimi ležících modulů daného typu v datovém rámci.

6.4. Podrobný popis komunikačních SW modulů DetStudia

Pro vlastní komunikaci má DetStudio pro každý typ datového rámce (DI, DO, AI, AO) implementován samostatný SW modul. Ten je volán vždy, když je potřeba periodicky číst stav vstupů (číslicových nebo analogových) nebo zapisovat na výstupy (číslicové nebo analogové). Podrobnější informace o názvech SW modulů dává následující tabulka:

Popis SW modulů

Název SW modulu	Operace
CAN_DI	Čtení analogových vstupů
CAN_AI	Čtení číslicových vstupů
CAN_DO	Zápis analogových výstupů
CAN_AO	Zápis číslicových výstupů

Zatímco SW modul **CAN_Node** je nutno volat v procesu typu Init, SW moduly pro čtení nebo zápis jsou obvykle volány v periodickém procesu (není to ale pravidlem, viz kapitola 6.1.3). Uskutečnění fyzického přenosu po síti dosáhneme dosazením jedničky za parametr `Transfer`. Parametr `Offset` udává číslo pořadí (0 .. 7) bajtu v datové části rámce, od kterého jsou uloženy příslušné hodnoty. Do proměnné `CNC` je nutno zadat návěští příslušného SW modulu **CNC_...**, který zajišťuje fyzické připojení k síti DIOCAN. Do proměnné `Node-ID` zadáváme adresu uzlu, s nímž se komunikuje. Proměnná `State`, která se při vložení požadavku do fronty nastaví do 0, se nastaví do jedničky, jestliže přenos proběhl v pořádku.

SW modul **CAN_DI** má jeden z výstupních parametrů jménem `Variable`. Po uskutečnění přenosu po síti je v této proměnné hodnota odpovídající stavům vstupů uzlu.

```
CAN_DI :10000, 1, 1, @PrenDI1, 4, 0, Cteni[0,0]
        |      |  |  |         |  |  └ Matice vstupů[1x2]
        |      |  |  |         |  └ Začátek dat v rámci
        |      |  |  |         └ Počet bajtů dat
        |      |  |  └ Stavová proměnná (State)
        |      |  └ Povolení fyzického přenosu po síti
        |      └ Adresa (Node-ID) uzlu, s nímž se komunikuje
        └ Návěští příslušného modulu CNC_… (např. CNC_ADCAN)
```

SW modul **CAN_DO** má jeden z vstupních parametrů jménem `Variable`. Po uskutečnění přenosu po síti se hodnota této proměnné binárně zapíše na příslušné výstupy modulu.

```
CAN_DO :10000, 5, 1, @PrenDO1, 6, 2, Zapis[0,0]
        |      |  |  |         |  |  └ Matice výstupů [1x3]
        |      |  |  |         |  └ Začátek dat v rámci
        |      |  |  |         └ Počet bajtů dat
        |      |  |  └ Stavová proměnná (State)
        |      |  └ Povolení fyzického přenosu po síti
        |      └ Adresa (Node-ID) uzlu, s nímž se komunikuje
        └ Návěští příslušného modulu CNC_… (např. CNC_ADCAN)
```

Jestliže máme v úmyslu číst z modulu analogových vstupů pomocí SW modulu **CAN_AI** více hodnot než jednu, použijeme konstrukci z následujícího příkladu. Za parametr `Transfer` zadáme hodnotu 1 jen při čtení prvního signálu – tj. zde signálu ze vstupu AI0. Jeho velikost se uloží do proměnné `CteniAI1`. Při čtení dalších signálů (AI1, ...) již za parametr `Transfer` zadáme hodnotu 0 a postupujeme stejně. Při čtení pouze jednoho vstupu postupujeme obdobně. Všechny

ostatní parametry jsou shodné jako u předchozích funkcí. Významy parametrů `Range .. PhysMax` jsou vysvětleny v závěru této kapitoly.

```
CAN_AI :10000, 9, 1, @PrenAI, 0x0010, 0, CteniAI1,…
        │      │  │  │       │       │  └ 1. Čtený vstup
        │      │  │  │       │       └ Začátek dat v rámci
        │      │  │  │       └ Parametry převodníku (16b)
        │      │  │  └ Stavová proměnná (State)
        │      │  └ Povolení fyzického přenosu po síti
        │      └ Adresa (Node-ID) uzlu, s nímž se komunikuje
        └ Návěští příslušného modulu CNC_… (např. CNC_ADCAN)

CAN_AI :10000, 9, 0, NONE.0, 0x0010, 2, CteniAI2,…
                  │  │       │       │  └ 2. Čtený vstup
                  │  │       │       └ Začátek dat v rámci
                  │  │       └ Parametry převodníku (16b)
                  │  └ Stavová proměnná se nedefinuje
                  └ Fyzický přenos po síti neprobíhá

CAN_AI :10000, 9, 0, NONE.0,  0x0010, 4, CteniAI3,…
                                       └ 3. Čtený vstup
```

Podobně postupujeme i při zápisu více analogových výstupů pomocí SW modulu **CAN_AO**. Za parametr `Transfer` zadáme hodnotu 1 jen při zápisu posledního signálu. Ve všech předchozích voláních funkce bude tedy parametr `Transfer` roven nule.

```
CAN_AO :10000, 13, 0, NONE.0, 0x0010, 0, ZapisAO1,…
        │      │   │  │       │       │  └ 1. Zapisovaný výstup
        │      │   │  │       │       └ Začátek dat v rámci
        │      │   │  │       └ Parametry převodníku (16b)
        │      │   │  └ Stavová proměnná se nedefinuje
        │      │   └ Fyzický přenos po síti neprobíhá
        │      └ Adresa (Node-ID) uzlu, s nímž se komunikuje
        └ Návěští příslušného modulu CNC_… (např. CNC_ADCAN)

CAN_AO :10000, 13, 0, NONE.0, 0x0010, 2, ZapisAO2,…
                                       └ 2. Zapisovaný výstup

CAN_AO :10000, 13, 1, @PrenAO, 0x0010, 4, ZapisAO3,…
                  │   │       │       │  └ 3. Zapisovaný výstup
                  │   │       │       └ Začátek dat v rámci
                  │   │       └ Parametry převodníku (16b)
                  │   └ Stavová proměnná (State)
                  └ Povolení fyzického přenosu po síti
```

6.4.1 Přehled parametrů SW modulů zpracovávajících analogový signál

Range
Horní hranice výstupního rozsahu HW modulu v elektrických jednotkách. Je možno volit z předdefinovaných hodnot 5 V, 10 V, 20 mA, 40 mA, přičemž je možno zvolit jakoukoli jinou hodnotu.

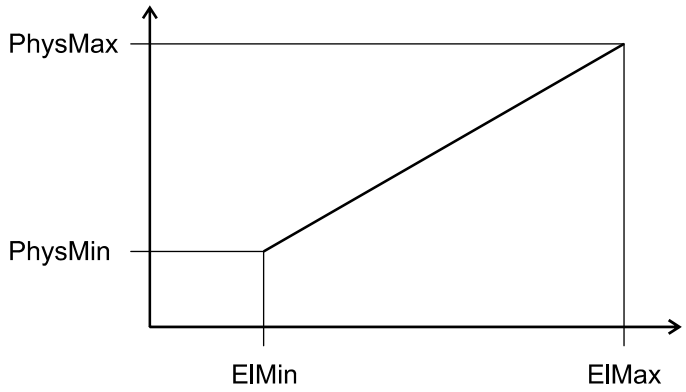
ElMin
Dolní mez signálu v elektrických jednotkách. Je možno volit z předdefinovaných hodnot 0 V/0 mA, -5 V, -10 V, 4 mA, -20 mA, -40 mA, nebo si zvolit hodnotu vlastní.

ElMax
Horní mez signálu v elektrických jednotkách. Je možno volit z předdefinovaných hodnot 5 V, 10 V, 20 mA, 40 mA, nebo si zvolit hodnotu vlastní.

PhysMin
Dolní mez signálu v elektrických jednotkách.

PhysMax
Horní mez signálu v elektrických jednotkách.

Přepočet
Přepočet z elektrických jednotek na fyzikální (nebo naopak) si je možno představit na základě následujícího obrázku:



Obr. 8 - Přepočet elektrických jednotek na fyzikální

7. Připojení zařízení jiných výrobců

7.1. Obsluha uzlu s WAGO 750-3x7

7.1.1 Konfigurace hardware uzlu

Před započetím komunikace je nejprve nutno nakonfigurovat adresu (Node-Id) uzlu a komunikační rychlost pomocí DIP přepínače, ovšem zcela jiným způsobem, než je popisováno v kapitole 3.7.

Postup při nastavování adresy a komunikační rychlosti:

- Přepnutí modulu do konfiguračního režimu (všechny DIP přepínače vypnuty) a zapnutí napájení. Aktuální nastavená rychlost je indikována blikáním příslušných LED (viz tabulka).

Signalizace nastavené rychlosti komunikace pomocí LED

LED	1 Mb	800 kb	500 kb	250 kb	125 kb	100 kb	50 kb	20 kb	10 kb
CAN-ERR	0	1	0	1	0	1	0	1	0
RUN	0	0	1	1	0	0	1	1	0
Tx-Owf	0	0	0	0	1	1	1	1	0
Rx	0	0	0	0	0	0	0	0	1

- Nastavení komunikační rychlosti dle tabulky:

Nastavení rychlosti komunikace

DIP	1 Mb	800 kb	500 kb	250 kb	125 kb	100 kb	50 kb	20 kb	10 kb
1	0	1	0	1	0	1	0	1	0
2	0	0	1	1	0	0	1	1	0
3	0	0	0	0	1	1	1	1	0
4	0	0	0	0	0	0	0	0	1

- Uložení nastavení přepnutím DIP přepínače 8 do polohy ON.
- Vypnutí napájení, přepnutí DIP přepínače 8 do polohy OFF.
- Nastavení adresy (Node-Id) uzlu. Adresa může nabývat hodnot 1 .. 127 a nastavuje se pomocí prvních sedmi DIP přepínačů.
- Zapnutí napájení.

7.1.2 Programová obsluha uzlu WAGO 750-3x7

S uzlem, jehož základ tvoří zařízení WAGO 750-3x7, je možno komunikovat předdefinovanými SW moduly DetStudia. Pro čtení digitálních vstupů slouží SW modul **CAN_DI**, pro zápis digitálních výstupů SW modul **CAN_DO**, pro čtení analogových vstupů SW modul **CAN_AI** a pro zápis analogových výstupů SW modul **CAN_AO**. Při zápisu do modulu analogových výstupů je nutno vždy vyslat tolik hodnot, kolik je kanálů tohoto modulu, jinak nedojde k oživení hodnot na výstupech.

V případě, že velikost dat komunikovaných z/do uzlu překročí v některém směru a pro některý typ jeden PDO rámec, je nutno použít speciální konstrukci – komunikaci prostřednictvím rámců SDO(1).

Poznámka

(1) Prostřednictvím SDO se dají vyčítat z NMT-uzlů s zařízeními WAGO 750-3x7 nejen data, ale také celá řada velmi podrobných informací. Další upřesnění viz manuál firmy WAGO, WAGO www.wago.com. SDO rámce se v DetStudiu realizují SW modulem CAN_PDO.

7.1.3 Příklady

Příklad

Obsluha uzlu, ve kterém je připojeno šest modulů číslicových vstupů, osm modulů číslicových výstupů, a po jednom modulu čtyř analogových vstupů a výstupů. U číslicových vstupů je nakonfigurováno zasílání dat z iniciativy SLAVEa (tedy při každé změně na vstupech).

Proces ProcINIT typu Init

```
:10000 CNC_ADCAN 0, 4
:10100 CAN_NMT_M :10000
     //Uzel, se kterým budeme komunikovat
     CAN_Node :10100, 1, 333, 3, Stav.0, 0, 1
     //Zasílání dat ze všech modulů čísl. vstupů.
     //Výsledek se uloží do matice DI.
     CAN_DI :10000, 1, 0, Stav.1, 6, 0, DI[0,0]
```

Proces Proc00 typu Normal

```
     //Z matice DO se vyberou 4 dvoubajtové hodnoty
     //a vyšlou se do uzlu.
     CAN_DO :10000, 1, 1, Stav.3, 8, 0, DO[0,0]
     //Modul AO je vždy nutno vykomunikovat celý!!!
     CAN_AO :10000, 1, 0, Stav.4, 0x0050, 0, AO[0,0], 10.000, -10.000, 10.000,
     -10.000, 10.000
     CAN_AO :10000, 1, 0, Stav.5, 0x0050, 2, AO[0,1], 10.000, -10.000, 10.000,
     -10.000, 10.000
     CAN_AO :10000, 1, 0, Stav.6, 0x0050, 4, AO[0,2], 10.000, -10.000, 10.000,
     -10.000, 10.000
     //Za parametr Transfer zadáváme jedničku pouze
     //v posledním SW modulu CAN_AO.
     CAN_AO :10000, 1, 1, Stav.7, 0x0050, 6, AO[0,3], 10.000, -10.000, 10.000,
     -10.000, 10.000
     //Modul AI může, ale nemusí být vykomunikován
     //celý. Za parametr Transfer zadáváme jedničku
     // pouze v prvním příkazu CAN_AI.
     CAN_AI :10000, 1, 1, Stav.8, 0x0010, 0, AI[0,0], 10.000, 0.000, 10.000,
     0.000, 10.000
     CAN_AI :10000, 1, 0, Stav.9, 0x0010, 2, AI[0,1], 10.000, 0.000, 10.000,
     0.000, 10.000
     CAN_AI :10000, 1, 0, Stav.10, 0x0010, 4, AI[0,2], 10.000, 0.000, 10.000,
     0.000, 10.000
     CAN_AI :10000, 1, 0, Stav.11, 0x0010, 6, AI[0,3], 10.000, 0.000, 10.000,
     0.000, 10.000
```

7.1.4 SDO

Pokud objem dat potřebných pro komunikaci přesáhne pro každý typ modulů osm bajtů, je nutno komunikovat prostřednictvím rámců SDO. Výhodou tohoto typu komunikace je, že je možno

obsluhovat v každém směru až 256 číslicových a až 256 analogových signálů. Nevýhodou je, že je možno v jednom cyklu čtení/zápisu přenést hodnoty pouze jednoho analogového nebo osmi číslicových signálů.

Následující příklad objasňuje komunikaci prostřednictvím rámců SDO. Je napsán obecně tak, aby byl kód použitelný pro všechny čtyři typy komunikací (AI, AO, DI, DO). Odlišnosti jednotlivých alternativ jsou uvedeny a označeny.

Databáze proměnných

Proměnná	Typ	Popis
CAN_In	MI[1,4]	Přijímaný telegram
CAN_Out	MI[1,4]	Vysílaný telegram
CAN_Out_Sab	MI[1,8]	Šablona vysílaného telegramu
CAN_Result	M**x**[1,8]	Matice výsledků, typ MI nebo MF
Pocet_Bajtu	I	Počet odesílaných bajtů dat
Pocet_Kanalu	I	Počet obsluhovaných kanálů
Poradi	I	Poř. číslo aktuálně R/W I/O
Stav	I	Stavová proměnná

Alias – Databáze

Název aliasu	Bit proměnné	Popis
@Stav_NMT	Stav.0	Stav příchozí komunikace
@Stav_I	Stav.1	Stav spojení s uzlem
@Stav_O	Stav.2	Stav odchozí komunikace

Proces ProcINIT typu Init

```
     //Inicializace CANu
:10000 CNC_ADCAN 0, 4
:10100 CAN_NMT_M :10000
     CAN_Node :10100, 1, 333, 3, @Stav_NMT, 0, 1
```

V procesu ProcINIT typu Init je zcela běžným způsobem naparametrizována DIOCAN síť.

Proces Proc00 typu Normal

```
     //Inkrementace pořadového čísla aktuálně
     //čteného nebo zapisovaného kanálu
     Let Poradi = if(Poradi>Pocet_Kanalu-1, 1, Poradi+1)
     Let CAN_Out_Sab[0, 3] = Poradi
     //Kódovaní řetězce pro vstupy
     Let CAN_Out[0, 0] = CAN_Out_Sab[0, 0]&0xFF +(CAN_Out_Sab[0, 1]&0xff)<<8
     Let CAN_Out[0, 1] = CAN_Out_Sab[0, 2]&0xFF +(CAN_Out_Sab[0, 3]&0xff)<<8
     Let CAN_Out[0, 2] = CAN_Out_Sab[0, 4]&0xFF +(CAN_Out_Sab[0, 5]&0xff)<<8
     Let CAN_Out[0, 3] = CAN_Out_Sab[0, 6]&0xFF +(CAN_Out_Sab[0, 7]&0xff)<<8
     //Odeslání odchozího telegramu
     //např. vyslání požadavku čtení vstupu
     CAN_PDO :10000, 1, 1537, 1, 1, @Stav_O, 4, 0, CAN_Out[0,0]
```

Vždy při jednom běhu periodického procesu Proc00 (typ normal) je přečtena nebo uložena hodnota odpovídající jednomu analogovému nebo osmici číslicových vstupů nebo výstupů (dále kanálů). O kterou eventualitu se jedná, rozhoduje tvar vysílaného telegramu (proměnná CAN_Out), potažmo jeho šablony, ze které je telegram vždy plněn, a která je vytvořena již při vývoji programu (proměnná CAN_Out_Sab). Tvary šablon vysílaného telegramu jsou uvedeny v následující tabulce:

Tvary šablon vysílaného telegramu

Typ komunikace	Počet bajtů	Tvar šablony(1)							
		1.	2.	3.	4.	5.	6.	7.	8.

DI	4	0x40	0x00	0x20	0xnn				
DO	6	0x2F	0x00	0x21	0xnn	0xDD	0x00		
AI	4	0x40	0x00	0x24	0xnn				
AO	6	0x22	0x00	0x25	0xnn	0xDL	0xDH		

Poznámka

(1) nn – pořadové číslo aktuálně komunikovaného kanálu
DD – jeden bajt dat – popisuje osmici číslicových vstupů/výstupů
DL – nižší bajt dat u analogových vstupů/výstupů
DH – vyšší bajt dat u analogových vstupů/výstupů
E1 .. E4 – bajty identifikující charakter vzniklé chyby
ii – libovolná hodnota.

Na řádcích 3 a 4 je do šablony vloženo číslo uzlu, na který se budeme dotazovat. Toto číslo se periodicky mění v rozsahu od jedné až po celkový počet obsluhovaných kanálů.

Na řádcích 5 .. 9 jsou data ze šablony sestavena do vysílaného telegramu a to tak, že vždy dvojice bajtů jsou mezi sebou vyměněny. (tedy z tabulky bajty 1. a 2., 3. a 4., 5. a 6.)

Vlastní odesílání telegramu zabezpečuje SW modul **CAN_PDO** na řádku 11. V tomto modulu je nutno správně vyplnit parametr `Cob-Id`, který se vypočte jako 0x600 + Node-Id komunikovaného uzlu a počet bajtů, který se bude odesílat (viz tabulka).

Proces Proc01 typu Normal

```
//Příjem příchozího telegramu
//Např. Načtený vstup
CAN_PDO :10000, 1, 1409, 0, 1, @Stav_I, 8, 0, CAN_In[0,0]
//Dekódovaní příchozího telegramu
Let Can_In_Vysl[0, 0] = Can_In[0, 0]&0xFF
Let Can_In_Vysl[0, 1] = (Can_In[0, 0]>>8)&0xFF
Let Can_In_Vysl[0, 2] = Can_In[0, 1]&0xFF
Let Can_In_Vysl[0, 3] = (Can_In[0, 1]>>8)&0xFF
Let Can_In_Vysl[0, 4] = Can_In[0, 2]&0xFF
Let Can_In_Vysl[0, 5] = (Can_In[0, 2]>>8)&0xFF
Let Can_In_Vysl[0, 6] = Can_In[0, 3]&0xFF
Let Can_In_Vysl[0, 7] = (Can_In[0, 3]>>8)&0xFF
```

Pro zpracování analogových hodnot je možno použít tento algoritmus:

```
Let CAN_Result[0, Poradi-1] = CAN_In[0, 2]/3277
```

Do matice `CAN_Result`, zde typu MF se ukládají přímo hodnoty napětí 0 .. 10 V.

Pro zpracování číslicových hodnot je možno použít tento algoritmus:

```
Let CAN_Result[0, Poradi-1] = CAN_In[0, 2]
```

Do matice `CAN_Result`, zde typu MI je do spodního bajtu každé hodnoty uložena jedna osmice bitů odpovídající stavům číslicových vstupů.

V periodickém procesu Proc01, který má periodu stejnou jako Proc00, ale tato perioda je offsetem posunuta tak, aby se stihla data po sběrnici CAN vykomunikovat. Řádek 1 zabezpečuje příjem odpovědi, která je uložena do matice `CAN_In`. `Cob-Id` příkazu se vypočte jako 0x580 + Node-Id. Počet bajtů je vhodné vždy nastavit na 8 (aby byl zabezpečen korektní přenos případné chybové zprávy).

Stejně tak, jako je nutno odesílaný telegram sestavit ze šablony, je pro lepší orientaci vhodné s přijímaným telegramem provést operaci opačnou.

Na řádcích 2 .. 14 je uveden algoritmus, který příchozí telegram uvede do srozumitelnější podoby. V následující tabulce jsou uvedeny možné tvary těchto telegramů:

Tabulka tvarů zpráv

Typ komunikace	Počet bajtů	Tvar šablony[1]							
		1.	2.	3.	4.	5.	6.	7.	8.
DI	8	0x4F	0x00	0x20	0xnn	0xDD	0x00	0x00	0x00
DO	8	0x60	0x00	0x21	0xnn	0x00	0x00	0x00	0x00
AI	8	0x4B	0x00	0x24	0xnn	0xDL	0xDH	0x00	0x00
AO	8	0x60	0x00	0x25	0xnn	0x00	0x00	0x00	0x00
Chyba[2]	8	0x80	0xii	0xii	0xii	0xE1	0xE2	0xE3	0xE4

Poznámka

[1] *nn – pořadové číslo aktuálně komunikovaného kanálu*
DD – jeden bajt dat – popisuje osmici číslicových vstupů/výstupů
DL – nižší bajt dat u analogových vstupů/výstupů
DH – vyšší bajt dat u analogových vstupů/výstupů
E1 .. E4 – bajty identifikující charakter vzniklé chyby
ii – libovolná hodnota.

[2] *Kódy všech chybových zpráv jsou uvedeny v manuálu k příslušné jednotce. Podrobnější informace lze nalézt na internetových stránkách společnosti WAGO www.wago.com.*

8. DODATEK A – Seznam SW modulů pro komunikaci v síti DIOCAN

Seznam SW modulů pro komunikaci v síti DIOCAN je uveden v tabulce níže. Podrobný popis modulů naleznete v nápovědě k návrhovému prostředí DetStudio.

Seznam SW modulů

Modul	Popis
CNC_C167	SW modul obsluhuje komunikaci na sběrnici CAN prostřednictvím interního řadiče na procesoru C167 u řídicích systémů, které jsou vybaveny tímto procesorem a zároveň potřebnými obvody pro fyzické připojení ke sběrnici. Poskytuje jednotné rozhraní modulům vyšších komunikačních vrstev NMT a PDO.
CNC_ART4k	Zajišťuje připojení ke sběrnici CAN na řídicích systémech typu **ART4000**. SW modul je funkčně plně ekvivalentní a stoprocentně zaměnitelný s SW modulem **CNC_C167**, jakož i s ostatními SW moduly, které jsou s **CNC_C167** plně ekvivalentní.
CNC_APT2k	Zajišťuje připojení ke sběrnici CAN na řídicích systémech typu **APT2100G**. SW modul je funkčně plně ekvivalentní a stoprocentně zaměnitelný s SW modulem **CNC_C167**, jakož i s ostatními SW moduly, které jsou s **CNC_C167** plně ekvivalentní.
CNC_AMR99	Zajišťuje připojení ke sběrnici CAN na řídicích systémech typu **AMiRiS99**. SW modul je funkčně plně ekvivalentní a stoprocentně zaměnitelný s SW modulem **CNC_C167**, jakož i s ostatními SW moduly, které jsou s **CNC_C167** plně ekvivalentní.
CNC_AMP99	Zajišťuje připojení ke sběrnici CAN na řídicích systémech typu **AMAP99**. SW modul je funkčně plně ekvivalentní a stoprocentně zaměnitelný s SW modulem **CNC_C167**, jakož i s ostatními SW moduly, které jsou s **CNC_C167** plně ekvivalentní.
CNC_ADOS	Zajišťuje připojení ke sběrnici CAN na řídicích systémech typu **ADOS1xx/2xx**. SW modul je funkčně plně ekvivalentní a stoprocentně zaměnitelný s SW modulem **CNC_C167**, jakož i s ostatními SW moduly, které jsou s **CNC_C167** plně ekvivalentní.
CNC_ST10	SW modul obsluhuje komunikaci na sběrnici CAN prostřednictvím interního řadiče na procesoru ST10 u řídicích systémů, které jsou vybaveny tímto procesorem a zároveň potřebnými obvody pro fyzické připojení ke sběrnici. Poskytuje jednotné rozhraní modulům vyšších komunikačních vrstev NMT a PDO.
CNC_APT3k2	Zajišťuje připojení ke sběrnici CAN na řídicích systémech typu **APT3200T**. SW modul obsluhuje komunikaci na sběrnici CAN prostřednictvím interního řadiče na procesoru ST10 u řídicího systému, který je vybaven potřebnými obvody pro fyzické připojení ke sběrnici. Poskytuje jednotné rozhraní modulům vyšších komunikačních vrstev NMT a PDO.
CNC_ADCAN	SW modul obsluhuje komunikaci na sběrnici CAN prostřednictvím řadiče v rozšiřujícím modulu **AD-CAN** modulárního řídicího systému řady **ADiS**. Poskytuje jednotné rozhraní modulům vyšších komunikačních vrstev NMT a PDO.
CAN_NMT_M	SW modul zajišťuje funkci NMT-MASTERa sběrnice CAN dle standardu CANopen.
CAN_NMT_S	Modul zajišťuje funkci NMT-SLAVEa sběrnice CAN dle standardu CANopen.
CAN_NMT_N	SW modul je určen k použití na místě SW modulů **CAN_NMT_M** nebo **CAN_NMT_S** v případě systémů, které nemají vzhledem k vrstvě NMT vykonávat ani funkci MASTERa ani funkci SLAVEa sběrnice CAN dle standardu CANopen.

Modul	Popis
CAN_Node	Modul slouží k přidání uzlu do seznamu uzlů, které má obsluhovat modul vrstvy NMT, na který se SW modul **CAN_Node** odkazuje návěštím. V případě NMT-MASTERa seznam uzlů slouží k řízení jejich inicializace a detekce stavu, v případě NMT-SLAVEa pouze k řízení detekce stavu.
CAN_AI	SW modul definuje příjem dat jednoho analogového vstupního signálu ze vstupního zařízení připojeného na sběrnici CAN.
CAN_AO	SW modul definuje vysílání dat jednoho analogového výstupního signálu na výstupní zařízení připojené na sběrnici CAN.
CAN_DI	Modul definuje příjem dat zvoleného počtu digitálních vstupních signálů ze vstupního zařízení připojeného na sběrnici CAN.
CAN_DO	SW modul definuje vysílání dat zvoleného počtu digitálních výstupních signálů na výstupní zařízení připojené na sběrnici CAN.
CAN_S_AI	SW modul definuje vysílání dat jednoho analogového vstupního signálu ze vstupního zařízení připojeného na sběrnici CAN.
CAN_S _AO	SW modul definuje příjem dat jednoho analogového výstupního signálu výstupním zařízením připojeným na sběrnici CAN.
CAN_S _DI	SW modul definuje vysílání dat zvoleného počtu digitálních vstupních signálů ze vstupního zařízení připojeného na sběrnici CAN.
CAN_S _DO	SW modul definuje příjem dat zvoleného počtu digitálních výstupních signálů výstupním zařízením připojeným na sběrnici CAN.
CAN_PDO	Tento SW modul definuje obecný procesní datový objekt sběrnice CAN. Používá se ve specifických případech viz kapitola 7. Jeho použití je mnohem komplikovanější než použití předdefinovaných modulů uvedených výše (to vyplývá z jeho všestrannosti).

9. DODATEK B - Seznam zařízení firmy AMiT komunikující v síti DIOCAN

V následující tabulce jsou uvedeny všechny řídicí systémy z produkce firmy AMiT, se kterými je možno komunikovat po sběrnici CAN a také způsob, jakým toho lze dosáhnout.

Řídicí systémy umožňující komunikaci pomocí sběrnice CAN

Řídicí systém	Způsob
ADOS + AM-CAN	Do kanálu COM1 je třeba osadit modul **AM-CAN**. Pokud je modul osazen, nelze řídicí systém již rozšířit o rozhraní RS485, M-Bus nebo další RS232. CAN rozhraní je galvanicky oddělené od ostatní elektroniky řídicího systému.
AMAP99 + AM-CAN	Do kanálu COM1 je třeba osadit modul **AM-CAN**. Pokud je modul osazen, nelze řídicí systém již rozšířit o rozhraní RS485, M-Bus nebo další RS232. CAN rozhraní je galvanicky oddělené od ostatní elektroniky řídicího systému.
AMiRiS99 + AM-CAN	Do kanálu COM1 je třeba osadit modul **AM-CAN**. Pokud je modul osazen, nelze řídicí systém již rozšířit o rozhraní RS485, M-Bus nebo další RS232. CAN rozhraní je galvanicky oddělené od ostatní elektroniky řídicího systému.
ART4000AC	Galvanicky oddělené rozhraní CAN je dostupné pouze ve verzi **ART4000AC**.
ADiS + AD-CAN	Připojením max. jednoho modulu **AD-CAN** na libovolnou pozici v sestavě lze získat galvanicky oddělené rozhraní CAN.
APT3100S + AM-CAN	Pro rozhraní CAN je třeba osadit modul **AM-CAN**. Pokud je modul osazen, nelze řídicí systém již rozšířit o rozhraní RS485, M-Bus nebo další RS232. CAN rozhraní je galvanicky oddělené od ostatní elektroniky řídicího systému.
APT3200T + CM-CAN	Pro rozhraní CAN je třeba osadit modul **CM-CAN**. **APT3200T** je možno osadit až dvěma volitelnými moduly (**CM-CAN**, **CM-RS232** a **CM-RS485**).

Systémy, které nejsou uvedeny v této tabulce, neumožňují žádným způsobem komunikaci po sběrnici CAN.

Seznam dostupných rozšiřujících modulů systému ADiS, které lze připojit k modulu ADC-CAN

Modul	Poznámka
AD-DI8A	Modul lze s centrálním modulem **ADC-CAN** používat pouze ve stejnosměrném režimu.
AD-FDI8	Modul lze připojit k modulu **ADC-CAN** bez omezení.
AD-DI16A	Modul lze s centrálním modulem **ADC-CAN** používat pouze ve stejnosměrném režimu. Modul lze připojit pouze k modulům **ADC-CAN**, vybaveným firmware verze 2.30 a vyšší. Verzi firmware lze zjistit ze samolepícího štítku na procesoru modulu. Pokud u nejstarších výrobků štítek zcela chybí, jedná se zaručeně o modul s verzí firmware 2.20 a nižší. Starší moduly s firmware V1.xx nelze ani dodatečně vybavit firmware V2.xx.
AD-PDO8	Modul lze připojit k modulu **ADC-CAN** bez omezení.
AD-RDO5S	Modul lze připojit k modulu **ADC-CAN** bez omezení.
AD-DO16	Modul lze připojit pouze k modulům **ADC-CAN**, vybaveným firmware verze 2.30 a vyšší. Verzi firmware lze zjistit ze samolepícího štítku na procesoru modulu. Pokud u nejstarších výrobků štítek zcela chybí, jedná se zaručeně o modul s verzí firmware 2.20 a nižší. Starší moduly s firmware V1.xx nelze ani dodatečně vybavit firmware V2.xx.

10. DODATEK C - Seznam zařízení jiných výrobců komunikujících v síti DIOCAN

Seznam zařízení jiných výrobců umožňující komunikaci na síti DIOCAN

Zařízení	Výrobce
WAGO 750-3x7 Modulární I/O systém navržený pro decentralizované řízení. Jsou nabízeny moduly obsahující různý počet číslicových i analogových vstupů/výstupů i speciální moduly (modul pro pulzně-šířkovou modulaci či inkrementální senzor). Přizpůsobení komunikace sběrnicovému systému CANopen je možno dosáhnout připojením příslušného komunikačního modulu. Detailní informace lze získat na www.wago.com. **Parametrizace v prostředí DetStudio probíhá prostřednictvím rámců PDO či SDO, viz kapitola 7.1.2.**	**WAGO**

11. Technická podpora

Veškeré informace ohledně komunikace v síti DIOCAN, Vám poskytne oddělení technické podpory firmy AMiT. Technickou podporu můžete kontaktovat nejlépe prostřednictvím emailu na adrese **support@amit.cz**.

12. Upozornění

AMiT spol. s r. o. poskytuje informace v tomto dokumentu, tak jak jsou, nepřejímá žádné záruky, pokud se týče obsahu tohoto dokumentu a vyhrazuje si právo měnit obsah dokumentu bez závazku tyto změny oznámit jakékoli osobě či organizaci.

Tento dokument může být kopírován a rozšiřován za následujících podmínek:

1. Celý text musí být kopírován bez úprav a se zahrnutím všech stránek.

2. Všechny kopie musí obsahovat označení autorského práva společnosti AMiT spol. s r. o. a veškerá další upozornění v dokumentu uvedená.

3. Tento dokument nesmí být distribuován za účelem dosažení zisku.

V publikaci použité názvy produktů, firem apod. mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.