TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: 2612 M – Elektrotechnika a informatika

Studijní obor: 2612 T – Automatické řízení a inženýrská informatika

Realizace software pro sběr dat z měřicích modulů na bázi jednočipových počítačů s využitím protokolu CANopen

(Realization of software for collecting data from measuring devices based on onechips computers using CANopen protocol)

Vedoucí práce: Ing. Jan Cvejn, Ph.D.

Rozsah práce:

Počet stran: 50

Počet obrázků: 23

Počet tabulek: 5

Počet příloh: 4

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Katedra softwarového inženýrství Akademický rok: 2002/2003

ZADÁNÍ DIPLOMOVÉ PRÁCE

pro: Jana Pospíšila

studijní program: 2612 M – Elektrotechnika a informatika

obor: 2612 T - Automatické řízení a inženýrská informatika

Vedoucí katedry Vám ve smyslu zákona o vysokých školách č.111/1998 Sb. určuje tuto diplomovou práci:

Název tématu:

Realizace software pro sběr dat z měřicích modulů na bázi jednočipových počítačů s využitím protokolu CANopen.

Zásady pro vypracování:

1. Vytvoření aplikace pro sledování stavu vybraných veličin měřicích modulů SMD firmy KMB, s.r.o. pomocí karty IXXAT pro práci se sběrnicí CAN.
2. Rozšíření současných možností komunikace modulů SMD po sběrnici CAN o aplikační vrstvu definovanou protokolem CANopen.
3. Rozšíření aplikace pro monitorování veličin modulů SMD o podporu rozhraní CANopen implementovaného v bodě 2.

+1CD ROM KSI/ÁŘI 64 s.

Rozsah grafických prací: dle potřeby dokumentace

Rozsah průvodní zprávy: cca 40 až 50 stran

Seznam odborné literatury:

[1] Specifikace CAN 2.0 (CAN in Automation (CiA) - www.can-cia.de)

[2] CAN Application Layer (CiA Draft Standard 201…207)

[3] CANopen Application Layer and Communication Profile (CiA Draft Standard 301)

[4] Interní dokumentace a zdrojové kódy firmy KMB, s.r.o.

Vedoucí diplomové práce: Ing. Jan Cvejn, Ph.D.

Konzultant:

Zadání diplomové práce: 20. 10. 2002

Termín odevzdání diplomové práce: 23. 5. 2003

L.S.

.....

Vedoucí katedry

.....

Děkan

V Liberci dne 20. 10. 2002

Anotace

Cílem této diplomové práce je implementace protokolu CANopen do měřících modulů SMD a realizace aplikace pro sběr aktuálních dat z modulů a jejich vizualizaci.

Práce zahrnuje teorii standardních protokolů CAN, CAL a CANopen. Jsou zde popsány jednotlivé služby protokolů pro konfiguraci a správu sítě, vzájemný vztah protokolů a definice komunikačního profilu zařízení.

V závěru je popsáno vytvoření aplikace pro vizualizaci měřených dat, způsob provedení protokolu CANopen v měřících modulech SMD a následné možné rozšíření služeb pro správu sítě.

Abstract

The goal of this diploma work is implementation the CANopen protokol into measuring modules SMD and realization of aplication for collecting topical data from modules and displaying these data.

The work includes the theory of the standard protokols CAN, CAL and CANopen. It describes individual services of protokols for net configuration and management, the reciprocal relation of protokols and the definition of the communications profile for devices.

The creation of aplication for displaying measured data, the way of implementation the CANopen protokol in measuring modules SMD and the consequential possible expansion of services for net management are described in conclusion.

Místopřísežné prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č.121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o využití mé DP a prohlašuji, že **s o u h l a s í m** s případným užitím diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum 23. 5. 2003 ..... Podpis Jan Rozsíval .....

Poděkování

Rád bych touto cestou poděkoval všem, kteří mi s vypracováním mé diplomové práce pomohli.

Zvláště děkuji vedoucímu diplomové práce Ing. Janu Cvejnovi, Ph.D. z katedry Softwarového inženýrství za odborný dohled a velmi cenné rady pro vypracování diplomové práce a panu Rojíkovi z firmy KMB s.r.o. za poskytnutí informací k přístroji SMD a jeho zdrojovému kódů.

Obsah :

I . Úvod **10**

II. Použité standardní protokoly **11**

1. CAN *11*
1. 1. Základní vlastnosti protokolu CAN 11
1. 2. Fyzické přenosové médium 12
1. 3. Spojová vrstva protokolu CAN 14
1. 3. 1. Řízení přístupu k médiu a řešení kolizí 14
1. 3. 2. Zabezpečení přenášených dat a detekce chyb 15
1. 3. 3. Signalizace chyb 16
1. 4. Základní typy zpráv 16
1. 4. 1. Datová zpráva (Data Frame) 17
1. 4. 2. Žádost o data (Remote Frame) 19
1. 4. 3. Chybová zpráva (Error Frame) 19
1. 4. 4. Zpráva o přetížení (Overload Frame) 20

2. CAL *21*
2.1. CMS 22
2.1.1. Proměnná 23
2. 1. 2. Doména 24
2. 1. 3. Událost 24
2. 2. NMT 25
2. 3. DBT 27
2. 4. LMT 28

3. CANopen *29*
3. 1. Objekt slovník 30
3. 2. Komunikační model 31
3. 2. 1. SDO 32
3. 2. 2. PDO 33
3. 2. 3. Synchronizace 34
3. 2. 4. Pohotovost 36

III. Realizace aplikace **37**

1. Použité hardwarové prostředky *37*
1. 1. Zařízení SMD 37
1. 2. Karta IXXAT pro přístup na CAN 38
1. 3. Mikroprocesor C515C 39

2. Protokol firmy KMB s.r.o. *40*
2. 1. Realizace aplikace dle protokolu firmy KMB s.r.o. 40

3. Implementace protokolu CANopen *43*
3. 1. Stavový diagram uzlu 44
3. 2. Služba Node Guarding 45
3. 3. Mapování objektů PDO 46
3. 4. Obsah objektu slovník 47
3. 5. Softwarové řešení implementace CANopen 49
3. 5. 1. Objekt zprávy 49

3. 5. 2. Hlídací mechanismus 50
3. 5. 3. Zdrojový kód přístroje 51
3. 5. 4. Provedení stavového diagramu 51
3. 5. 5. Detekce chybových stavů 52
3. 5. 6. Popis algoritmu 52

4. Realizace programu pro monitorování zařízení 54
4. 1. Popis aplikace 55
4. 2. Průběh algoritmu protokolu 55
4. 3. Použité datové struktury a metody 56
4. 4. Návrh úpravy aplikace 58

IV. Závěr 59

Seznam použitých zkratek a symbolů:

CAL	(CAN Application Layer) komunikační protokol aplikační vrstvy.
CAN	(Controller Area Network) sítový komunikační protokol navržený pro automobilový průmysl.
CMS	(CAN based Message Specification) definice výměny aplikačních dat.
COB	(Communication Object) komunikační jednotka protokolu CAN, každý objekt (zpráva) je rozlišen jedinečným identifikátorem a může obsahovat maximálně 8 datových bytů.
COB-ID	identifikátor zprávy CAN. Určuje prioritu přístupu na sběrnici.
DBT	(Distributor) protokol distribuce COB-ID objektům CMS.
LLC	(Logical Link Control) část spojové vrstvy protokolu CAN
LMT	(Layer Management) soubor služeb pro konfiguraci základních parametrů protokolu CAL.
MAC	(Medium Acces Control) část spojové vrstvy protokolu CAN
NMT	(Network management) soubor služeb pro správu sítě.
PDO	(Process Data Object) objekt sloužící pro výměnu dat v reálném čase.
SDO	(Service Data Object) objekt pro zápis a čtení položek profilu zařízení.
ss.	stejnosměrné napětí
stř.	střídavé napětí

I. ÚVOD

Téměř každý automatický systém obsahuje zařízení, která spolu komunikují. Existují standardní síťové komunikační protokoly, vytvořené pro řízení technicky funkčních elementů. Avšak někteří výrobci automatických systému vytváří své vlastní standardy. Následné řízení a informační tok se v pojetí výrobců liší a ve většině případů není použité řízení modulů kompatibilní.

Protokol CANopen byl vytvořen organizací Cia (CAN in Automation) pro průmyslové řídící systémy, které používají sběrnici CAN. Komunikační protokol CANopen vytváří rozhraní mezi CAN aplikační vrstvou a profilem zařízení. Zahrnuje model komunikace v reálném čase a protokoly, které jsou společné pro všechna zařízení v síti. Účelem definice standardních profilů zařízení je poskytnutí jednotné koncepce nezávislým výrobcům zařízení a tím vytvoření jednotného modulárního systému s možností integrace a vzájemné náhrady modulů různých výrobců bez nutnosti změn a zásahů do vytvořeného systému. V současné době má největší uplatnění v dopravních prostředcích, stavební mechanizaci, lékařství, námořní elektronice a strojních zařízeních.

Diplomová práce je rozdělena do dvou hlavních částí. První část obsahuje popis protokolu. Jsou zde vysvětleny principy síťové správy, mechanismus zabezpečení, synchronizace zpráv, komunikační modely a používané objekty. Dále je zde popsán princip standardního profilu zařízení a jeho využití pro vzájemnou komunikaci zařízení na síti.

Druhá část se zabývá hardwarovými prostředky a softwarovým řešením implementace protokolu CANopen do měřících modulů. Jsou zde objasněny použité metody a objekty protokolu a jejich realizace v konkrétním zařízení. Následně je popsáno vytvoření aplikace, která má řídící úlohu ve vytvořeném síťovém prostředí. Hlavním úkolem aplikace je vizualizace měřených dat modulů a monitorování jejich aktuálního stavu.

II. POUŽITÉ STANDARDNÍ PROTOKOLY

1. CAN

CAN (*Controller Area Network*) je sériový komunikační protokol, který byl původně vyvinut firmou Bosch pro nasazení v automobilech. Vzhledem k tomu, že přední výrobci integrovaných obvodů implementovali podporu protokolu CAN do svých produktů, dochází k stále častějšímu využívání tohoto protokolu i v různých průmyslových aplikacích. Důvodem je především nízká cena, snadné nasazení, spolehlivost, vysoká přenosová rychlost, snadná rozšiřitelnost a dostupnost potřebné součástkové základny.

CAN je definován normou ISO 11898. Norma popisuje fyzickou vrstvu protokolu a specifikaci CAN 2.0A. Později byla ještě vytvořena specifikace CAN 2.0B, která zavádí dva pojmy - standardní a rozšířený formát zprávy (rozdílná délka identifikátoru zprávy). Tyto dokumenty definují pouze fyzickou a spojovou vrstvu protokolu podle referenčního modelu ISO/OSI.

1. 1. Základní vlastnosti protokolu CAN

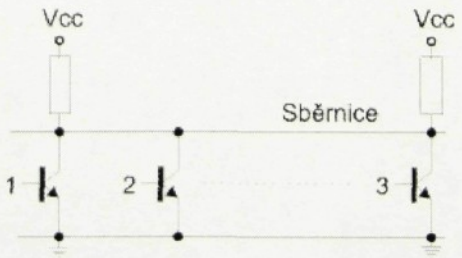
CAN byl navržen tak, aby umožnil provádět distribuované řízení systémů v reálném čase s přenosovou rychlostí do 1Mbit/s a vysokým stupněm zabezpečení přenosu proti chybám. Jedná se o protokol typu pán-otrok (*Master-Slave*), kde každý uzel sběrnice může být *Master*, který řídí chování jiných uzlů. Není tedy nutné řídit celou síť z jednoho nadřazeného uzlu, což přináší zjednodušení řízení a zvyšuje spolehlivost (při poruše jednoho uzlu může zbytek sítě pracovat dál). Pro řízení přístupu k médiu je použita sběrnice s náhodným přístupem, která řeší kolize na základě prioritního rozhodování. Po sběrnici probíhá komunikace mezi dvěma uzly pomocí zpráv. Zprávy se dělí na tři typy: základní zprávy (datová zpráva a žádost o data), zprávy pro správu sítě (signalizace chyb, pozastavení komunikace) a speciální zprávy (chybové zprávy a zprávy o přetížení).

Zprávy vysílané po sběrnici protokolem CAN neobsahují žádnou informaci o cílovém uzlu, kterému jsou určeny, a jsou přijímány všemi ostatními uzly připojenými ke sběrnici. Každá zpráva je uvozena identifikátorem, který udává význam přenášené zprávy

a její prioritu. Nejvyšší prioritu má zpráva s identifikátorem 0. Protokol CAN zajišťuje, aby zpráva s vyšší prioritou byla v případě kolize dvou zpráv doručena přednostně a dále je možné na základě identifikátoru zajistit, aby uzel přijímal pouze ty zprávy, které se ho týkají.

1. 2. Fyzické přenosové médium

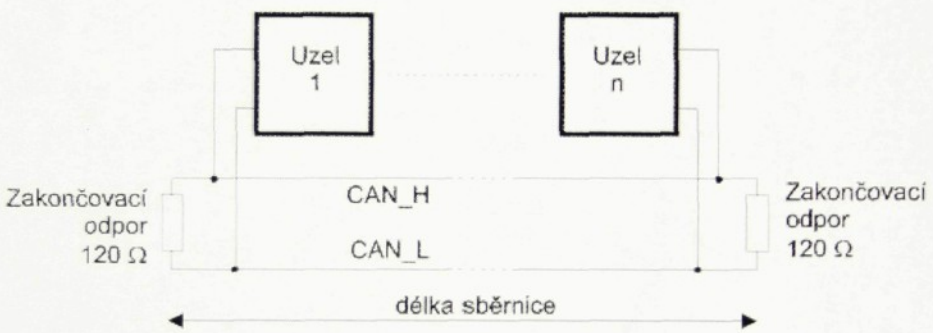
Základním požadavkem na fyzické přenosové médium protokolu CAN je, aby realizovalo funkci logického součinu. Standard protokolu CAN definuje dvě vzájemně komplementární hodnoty bitů na sběrnici – dominantní a recesivní (*Dominant, Recessive*). V podstatě jde o jakýsi zobecnělý ekvivalent logických úrovní, jejichž hodnoty nejsou určeny a skutečná reprezentace záleží na konkrétní realizaci fyzické vrstvy. Pravidla pro stav na sběrnici jsou jednoduchá a jednoznačná. Vysílají-li všechny uzly sběrnice recesivní bit, pak na sběrnici je úroveň *Recessive*. Vysílá-li alespoň jeden uzel dominantní bit, je na sběrnici úroveň *Dominant*. Příkladem může být optické vlákno, kde stavu *Dominant* bude odpovídat stav "svítí" a stavu *Recessive* "nesvítí". Dalším příkladem může být sběrnice buzená hradly s otevřeným kolektorem (obr. 1), kde stavu *Dominant* bude odpovídat logická nula na sběrnici a stavu *Recessive* logická jednička. Pak, je-li jeden tranzistor sepnut, je na sběrnici úroveň logické nuly (*Dominant*) a nezáleží již na tom, zda je či není sepnutý i nějaký jiný tranzistor. Pokud není sepnut žádný tranzistor, je na sběrnici úroveň logické jedničky (*Recessive*).



Obr. 1: Příklad realizace fyzické vrstvy protokolu CAN

Pro realizaci fyzického přenosového média se nejčastěji používá diferenciální sběrnice definovaná podle normy ISO 11898. Tato norma definuje jednak elektrické

vlastnosti vysílacího budiče a přijímače, ale i zároveň princip časování, synchronizace a kódování jednotlivých bitů. Sběrnici tvoří dva vodiče (označované CAN_H a CAN_L), kde úroveň *Dominant* či *Recessive* na sběrnici je definována rozdílovým napětím těchto dvou vodičů. Dle nominálních úrovní uvedených v normě je pro úroveň *Recessive* velikost rozdílového napětí $V_{diff} = 0V$ a pro úroveň *Dominant* $V_{diff} = 2V$. Pro eliminaci odrazů na vedení je sběrnice na obou koncích přizpůsobena ukončovacími odpory o velikosti 120 Ohm. Jednotlivá zařízení jsou na sběrnici připojena pomocí konektorů (např. konektor D-SUB).



Obr. 2: Principiální struktura sítě CAN podle ISO 11898

Na sběrnici může být čistě teoreticky připojeno neomezené množství uzlů, avšak s ohledem na zatížení sběrnice a zachování správných statických i dynamických parametrů je počet připojených uzlů omezený. Maximální délka sběrnice je pro přenosovou rychlost 1Mbit/s udána normou 40m. Pro jiné přenosové rychlosti délku sběrnice norma neudává, ale pro nižší přenosové frekvence bude maximální délka sběrnice větší. Maximální délky sběrnice pro jinou přenosovou rychlost než 1Mbit/s jsou uvedené v tabulce č. 1. Tyto rychlosti jsou pouze informativní a závisí na mnoha dalších parametrech (např. typu použitého kabelu).

Tab. č. 1: Závislost přenosové rychlosti na délce sběrnice

Přenosová rychlost [kbit/s]	Maximální délka sběrnice [m]
1000	40
500	112
300	200
100	640
50	1340
20	2600
10	5200

1. 3. Spojová vrstva protokolu CAN

Spojová vrstva protokolu CAN je tvořena dvěma částmi - MAC a LLC. První z nich MAC (*Medium Access Control*) má na starosti přístup k médiu a jejím úkolem je provádět kódování dat, vkládat doplňkové bity do komunikace (*Stuffing/Destuffing*), řídit přístup všech uzlů k médiu s rozlišením priorit zpráv, detekci chyb a jejich hlášení a potvrzování správně přijatých zpráv.

Druhou částí spojové vrstvy je LLC (*Logical Link Contol*), má za úkol provádět filtrování přijatých zpráv (*Acceptance Filtering*) a hlášení o přetížení (*Overload Notification*).

1. 3. 1. Řízení přístupu k médiu a řešení kolizí

Je-li sběrnice volná (stav *Bus Free*), může libovolný uzel zahájit vysílání. Zahájí-li některý uzel vysílání dříve než ostatní, získává sběrnici pro sebe. Ostatní uzly mohou zahájit vysílání až po odvysílání kompletní zprávy. Jedinou výjimku zde tvoří chybové zprávy, které může vyslat libovolný uzel, detekuje-li chybu v právě přenášené zprávě.

Zahájí-li vysílání současně několik uzlů, pak přístup na sběrnici získá ten, který přenáší zprávu s vyšší prioritou (nižším identifikátorem). Identifikátor je uveden na začátku zprávy. Každý vysílač porovnává hodnotu právě vysílaného bitu s hodnotou na sběrnici a zjistí-li, že na sběrnici je jiná hodnota než vysílá (jedinou možností je, že vysílač vysílá recesivní bit a na sběrnici je úroveň *Dominant*), okamžitě přeruší další vysílání. Tím je zajištěno, že zpráva s vyšší prioritou bude odeslána přednostně a že nedojde k jejímu

poškození, což by mělo za následek opakování zprávy a zbytečné prodloužení doby potřebné k přenosu zprávy. Uzel, který nezískal při kolizi přístup na sběrnici musí vyčkat až bude sběrnice opět ve stavu *Bus Free*, a pak zprávu vyslat znovu.

1. 3. 2. Zabezpečení přenášených dat a detekce chyb

Zprávy přenášené pomocí protokolu CAN jsou zabezpečeny několika mechanizmy, které jsou v činnosti současně.

Sledování (*Monitoring*) - již zmiňovaná metoda, kdy vysílač porovnává hodnotu právě vysílaného bitu s úrovní detekovanou na sběrnici. Jsou-li obě hodnoty stejné, vysílač pokračuje ve vysílání. Pokud je na sběrnici detekována jiná úroveň než odpovídá vysílanému bitu, a probíhá-li právě řízení přístupu na sběrnici (vysílá se *Arbitration Field*), přeruší se vysílání a přístup k sběrnici získá uzel vysílající zprávu s vyšší prioritou. Pokud je rozdílnost vysílané a detekované úrovně zjištěna jinde než v *Arbitration Field* a v potvrzení přijetí zprávy (*ACK Slot*), je vygenerována chybová zpráva (chyba bitu).

CRC kód (*Cyclic Redundancy Check*) - na konci každé zprávy je uveden 15-ti bitový CRC kód, který je generován ze všech předcházejících bitů příslušné zprávy podle polynomu: $x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$. Je-li detekována chyba CRC libovolným uzlem na sběrnici, je vygenerována chybová zpráva (chyba CRC).

Vkládání bitů (*Bit Stuffing*) - vysílá-li se na sběrnici pět po sobě jdoucích bitů jedné úrovně, je do zprávy navíc vložen bit opačné úrovně. Toto opatření slouží jednak k detekci chyb ale také ke správnému časovému synchronizování přijímačů jednotlivých uzlů. Je-li detekována chyba vládání bitů, je vygenerována chybová zpráva.

Např. data k vyslání: 01010000011011111100

vyslaná sekvence po vložení bitu: 01010000**1**11011111**0**1100

Kontrola zprávy (*Message Frame Check*) - zpráva se kontroluje podle formátu udaného ve specifikaci a pokud je na nějaké pozici bitu zprávy detekována nepovolená hodnota, je vygenerována chybová zpráva (chyba formátu zprávy).

Potvrzení přijetí zprávy (*Acknowlege*) - je-li zpráva v pořádku přijata libovolným uzlem, je toto potvrzeno změnou hodnoty jednoho bitu zprávy (ACK). Vysílač vždy na tomto bitu vysílá úroveň *Recessive* a detekuje-li úroveň *Dominant*, pak je vše v pořádku.

Potvrzování přijetí zprávy je prováděno všemi uzly připojenými ke sběrnici bez ohledu na zapnuté či vypnuté filtrování zpráv (*Acceptance Filtering*).

1. 3. 3. Signalizace chyb

Každý uzel má zabudována dvě interní počítadla chyb, udávající počet chyb při příjmu a při vysílání. Podle obsahů počítadel může uzel přecházet (dle hlášení chyb a jeho aktivity na sběrnici) mezi třemi stavy (aktivní, pasivní, odpojený). Pokud uzel generuje příliš velké množství chyb, je automaticky odpojen (přepnut do stavu *Bus-Off*). Z hlediska hlášení chyb tedy rozdělujeme uzly do následujících tří skupin.

Aktivní (*Error Active*) - tyto uzly se mohou aktivně podílet na komunikaci po sběrnici. V případě, že detekují libovolnou chybu v právě přenášené zprávě (chyba bitu, chyba CRC, chyba vkládání bitů, chyba rámce), vysílají na sběrnici aktivní příznak chyby (*Active Error Flag*). Aktivní příznak chyby je tvořen šesti po sobě jdoucími bity dominant, čímž dojde k poškození přenášené zprávy (poruší se pravidlo vkládání bitů).

Pasivní (*Error Passive*) - tyto uzly se také podílejí na komunikaci po sběrnici, ale z hlediska hlášení chyb, vysílají pouze pasivní příznak chyby (*Passive Error Flag*). Ten je tvořen šesti po sobě jdoucími recesivními bity, čímž nedojde k destrukci právě vysílané zprávy.

Odpojené (*Bus-Off*) - tyto uzly nemají žádný vliv na sběrnici, jejich výstupní budiče jsou vypnuty.

1. 4. Základní typy zpráv

Specifikace protokolu CAN definuje čtyři typy zpráv. První dvě se týkají datové komunikace po sběrnici. Je to jednak datová zpráva, která představuje základní prvek komunikace uzlů po sběrnici. Dále pak zpráva na vyžádání dat, kdy uzel žádá ostatní účastníky na sběrnici o zaslání požadovaných dat. Datová zpráva umožňuje vyslat na sběrnici 0 až 8 datových bajtů. Pro jednoduché příkazy uzlům (např. příkazy typu vypni/zapni) není nutné přenášet žádné datové bajty (význam příkazu je dán identifikátorem zprávy), což zkracuje dobu potřebnou k přenosu zprávy a zároveň zvětšuje propustnost sběrnice, zvláště pak při silném zatížení. Zpráva na vyžádání dat je vyslána

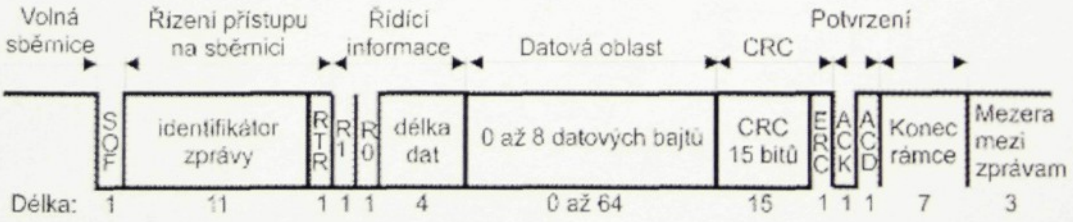
uzlem, který požaduje zaslání určitých dat. Odpovědí na tento požadavek je odeslání požadovaných dat uzlem, který tato data má k dispozici.

Poslední dvě zprávy (chybová zpráva a zpráva o přetížení) slouží k řízení komunikace po sběrnici, konkrétně k signalizaci detekovaných chyb, eliminaci chybných zpráv a vyžádání prodlevy v komunikaci.

1. 4. 1. Datová zpráva (Data Frame)

Protokol CAN používá dva typy datových zpráv. První typ je definován specifikací 2.0A a je označován jako standardní formát zprávy (*Standard Frame*), zatímco specifikace 2.0B definuje navíc tzv. rozšířený formát zprávy (*Extended Frame*). Jediný podstatný rozdíl mezi oběma formáty je v délce identifikátoru zprávy, která je 11 bitů pro standardní formát a 29 bitů pro rozšířený formát. Oba dva typy zpráv mohou být používány na jedné sběrnici, pokud je použitým řadičem podporován protokol 2.0B.

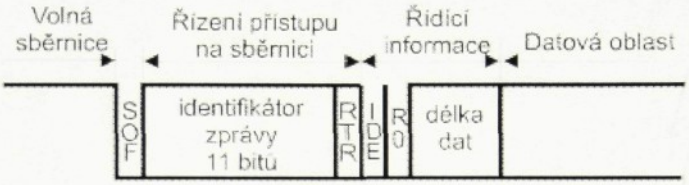
Vyslání datové zprávy je možné pouze tehdy, je-li sběrnice volná (stav *Bus Free*). Jakmile uzel, který má připravenou zprávu k vyslání, detekuje volnou sběrnici, začíná vysílat. Zda získá přístup na sběrnici či nikoliv, záleží na již popsaném mechanizmu řízení přístupu k médiu. Strukturu datové zprávy podle specifikace 2.0A ilustruje obr. 3.



Obr. 3: Datová zpráva podle specifikace CAN 2.0A

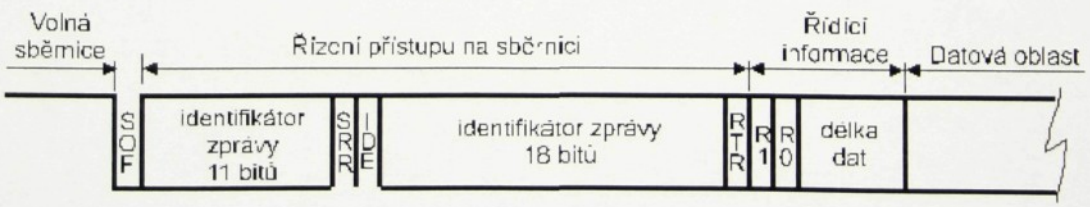
Význam jednotlivých částí datové zprávy podle specifikace 2.0A je uveden v příloze č.1 . Specifikace CAN 2.0B definuje dva formáty datového zprávy - standardní a rozšířený.

Standardní zpráva (*Standard Frame*) dle specifikace 2.0A, má délku identifikátoru zprávy 11 bitů. Jediným rozdílem je zde využití bitu R1 na indikaci, zda se jedná o rámec standardní nebo rozšířený. Zde se podle CAN 2.0B tento bit označuje IDE (*Identifier Extended*) a je dominantní pro standardní formát a recesivní pro rozšířený formát zprávy. Z obr. 4, který zobrazuje začátek rámce je vidět, že řízení přístupu na sběrnici (priorita zprávy) je dána opět 11-ti bity identifikátoru a hodnotou bitu RTR (*Remote Request*).



Obr. 4: Začátek datové zprávy (standardní formát) podle specifikace 2.0B

Rozšířený rámec (*Extended Frame*) používá celkem 29 bitový identifikátor zprávy. Ten je rozdělen do dvou částí o délkách 11 bitů (stejný identifikátor je použit ve standardním formátu) a 18 bitů (viz obr. 5). Bit RTR (*Remote Request*) je zde nahrazen bitem SRR (*Substitute Remote Request*), který je v rozšířeném formátu vždy recesivní. To zajišťuje, aby při vzájemné kolizi standardního a rozšířeného formátu zprávy na jedné sběrnici se stejným 11-ti bitovým identifikátorem, získal přednost standardní rámec. Bit IDE (*Identifier Extended*) je vždy recesivní. Bit (RTR) udávající, zda se jedná o datovou zprávu nebo žádost o data je přesunut za konec druhé části identifikátoru. Pro řízení přístupu k médiu jsou použity ID (11 bit), SRR, IDE, ID (18 bit), RTR. V tomto pořadí je určena priorita datové zprávy.



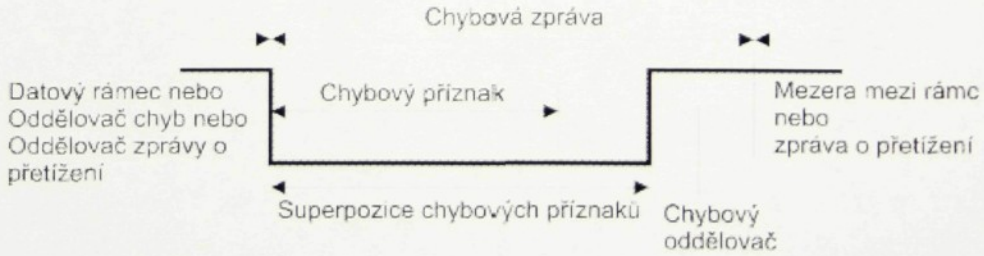
Obr. 5: Začátek datové zprávy (rozšířený formát) podle specifikace 2.0B

1. 4. 2. Žádost o data (Remote Frame)

Formát žádosti o data je podobný jako formát datové zprávy. Pouze je zde RTR bit (pole řízení přístupu na sběrnici) nastaven do úrovně *Recessive* a chybí datová oblast. Pokud nějaký uzel žádá o zaslání dat, nastaví takový identifikátor zprávy, jako má datová zpráva, jejíž zaslání požaduje. Tím je zajištěno, že pokud ve stejném okamžiku jeden uzel žádá o zaslání dat a jiný data se stejným identifikátorem vysílá, přednost v přístupu na sběrnici získá uzel vysílající datovou zprávu, neboť úroveň RTR bitu datové zprávy je dominantní a tudíž má tato zpráva vyšší prioritu.

1. 4. 3. Chybová zpráva (Error Frame)

Chybová zpráva slouží k signalizaci chyb na sběrnici CAN. Jakmile libovolný uzel na sběrnici detekuje v přenášené zprávě chybu (chyba bitu, chyba CRC, chyba vkládání bitů, chyba rámce), vygeneruje ihned na sběrnici chybový rámec (obr. 6). Podle toho, v jakém stavu pro hlášení chyb se uzel, který zjistil chybu, právě nachází, generuje na sběrnici buď aktivní (šest dominantních bitů) nebo pasivní (šest recesivních bitů) příznak chyby. Při generování aktivního příznaku chyby je přenášená zpráva poškozena (vzhledem k porušení pravidla na vkládání bitů), a tedy i ostatní uzly začnou vysílat chybové zprávy. Hlášení chyb je pak indikováno superpozicí všech chybových příznaků, které vysílají jednotlivé uzly. Délka tohoto úseku může být minimálně 6 a maximálně 12 bitů.



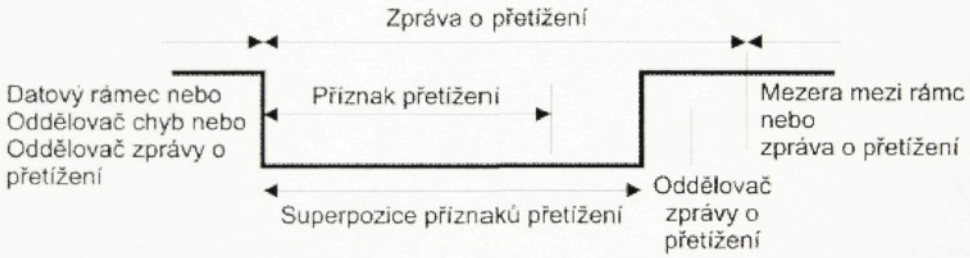
Obr. 6: Chybová zpráva protokolu CAN

Po vyslání chybového příznaku vysílá každá stanice na sběrnici recesivní bity. Zároveň detekuje stav sběrnice a jakmile najde první bit na sběrnici ve stavu *Recessive*,

vysílá se dalších sedm recesivních bitů, které plní funkci oddělovače chyb (ukončení chybové zprávy).

1. 4. 4. Zpráva o přetížení (Overload Frame)

Zpráva o přetížení slouží k oddálení vyslání další datové zprávy nebo žádosti o data. Zpravidla tento způsob využívají zařízení, která nejsou schopna kvůli svému vytížení přijímat a zpracovávat další zprávy. Struktura zprávy (obr. 7) je podobná zprávě o chybě, ale její vysílání může být zahájeno po konci zprávy (*End of Frame*), oddělovače chyb nebo předcházejícího oddělovače zpráv přetížení.

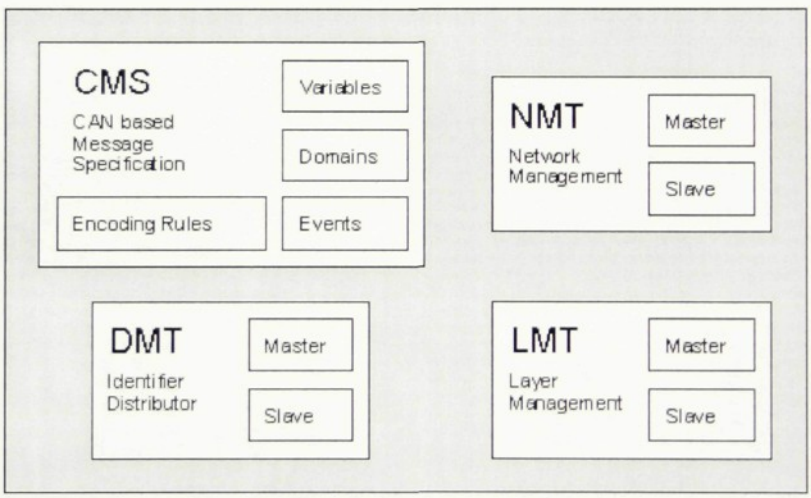


Obr. 7: Zpráva o přetížení

Zpráva o přetížení je složena z příznaku přetížení (šest dominantních bitů) a případné superpozice všech příznaků přetížení, pokud jsou generovány více uzly současně. Za příznaky přetížení následuje dalších sedm recesivních bitů, které tvoří oddělovač zprávy o přetížení.

2. CAL

Specifikace protokolu CAN odpovídající fyzické a spojové vrstvě modelu OSI a zaručuje pouze vyslání a příjem zprávy jednotlivými uzly zapojených do sítě. Pro systémy distribuovaného řízení v reálném čase využívající protokol CAN je další prakticky realizovatelná vrstva až aplikační, která zprostředkovává služby nutné pro chod celého systému. K těmto službám patří nejenom distribuce aplikačních a řídících dat ale i možnostmi konfigurace, správy a testování funkčnosti jednotlivých uzlů i celého systému. Nejdůležitější funkcí aplikační vrstvy je rozhodnout, co smí aplikace v komunikačním prostředím. Komunikační protokol CAL (*CAN application layer*) je tvořen čtveřicí skupin služeb - CMS, NMT, LMT a DBT (obr. 8).



Obr. 8: CAN aplikační vrstva

CMS (*CAN based Message Specification*) definuje protokol výměny aplikačních dat, založený na třech typech komunikačních objektů - proměnná, událost a doména. Dále určuje jak zakódovat a dešifrovat aplikační data do transportního tvaru,včetně implementace přenosu datových bloků delších než 8 bytů.

NMT (*Network Management*) slouží k řízení činnosti jednotlivých modulů i celé sítě z hlediska systémového správce (NMT Master). Řeší inicializaci a možné chyby ostatních modulů (NMT Slave).

DBT (*Idnetifier Distributor*) umožňuje během procesu konfigurace modulů (služba NMT) dynamicky přidělovat COB-ID objektům definovaných protokolem CMS. COB-ID určuje prioritu pro protokol MAC. Základním problémem v definování otevřeného prostředí CAN, je distribuce identifikátorů jednotlivým objektům zpráv. Jeden modul (DBT Master) přiděluje dynamicky COB-ID ostatním modulům (DBT Slave).

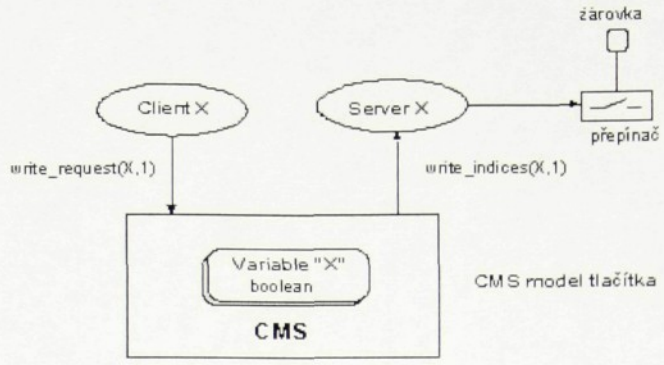
LMT (*Layer Management*) služba dovoluje změnit jedním modulem (LMT Master) základní parametry ostatních modulů (LMT Slave) - komunikační rychlost a identifikátory modulů.

2.1. CMS

CMS specifikuje způsob využití zpráv na úrovni protokolu CAN danou aplikací. Specifikace zahrnuje definici základních objektových typů (objekty CMS) s jejich atributy a dále servisní služby definované na těchto objektech. Protokol rozlišuje služby na lokální úrovni probíhající v rámci modulu a síťové, kdy dochází ke komunikace po sběrnici CAN. Atributy společné pro všechny typy objektů CMS:

- datový typ objektu - BOOLEAN,INTEGER,ARRAY,STRUCTURE, atd.
- prioritní skupina objektu (definováno 8 skupin)
- *inhibt time* - minimální doba mezi dvěmi následujícími vyslání daného objektu po síti. Definice tohoto času poskytuje možnost přístupu na sběrnici zprávám s nižší prioritou COB-ID.
- uživatel objektu z hlediska přístupu, server nebo klient (viz obr. 9)
- třída objektu podle počtu příjemců a poskytovatelů daného objektu (např. 1 vysílající - 1 příjemce, 0 nebo 1 vysílající - více jak 1 příjemce atd.)

Server CMS je zařízení, které obsahuje objekty CMS, s nimiž mohou pracovat klienti CMS . Objekty v serveru CMS často reprezentují reálná zařízení a zrcadlí jejich fyzický stav. Příklad vztahu mezi zařízením typu klient a zařízením typu server je na obr. 9.



Obr. 9: Klient / Server

2.1.1. Proměnná

Objekt typu proměnná může být buď základní nebo mnohonásobná. V druhém případě se do mnohonásobné proměnné přistupuje pomocí přirozeného čísla, které jednoznačně identifikuje proměnou uvnitř řady proměnných. Každá proměnná má definované vlastnosti :

- jméno
- uživatelský typ: „Client " nebo „ Server"
- priorita: hodnota v rozsahu [0, 7]
- omezení času: n*100 μsec, n >> 0
- datový typ
- chybový typ
- třída: „Basic" nebo „Multiplexed"
- typ přístupu: „Read_Only", „Write_Only" nebo „Read_Write"

K lokálním službám pro práci s proměnnými patří definice proměnných a skupin a obnova hodnot. K síťovým pak čtení a zápis proměnných.

Proměnná s typem přístupu „Read_Only" může být použita pouze pro sběr dat. Server předává hodnotu proměnné po její poslední aktualizaci. Hodnoty z minulých aktualizací jsou ztraceny. Proměnná s typem přístupu „Write_Only" je použita klientem k požadavku zápisu na jeden nebo více serverů. Klient nezná výsledek dané operace. Proměnná s typem přístupu „Read_Write" slouží klientovi k čtení hodnoty proměnné nebo

zápisu na daném serveru. Klient je informován o úspěchu či nezdaru daného příkazu. Objekt typu proměnná je určen pro přenos dat o maximální délce 8 bytů.

2. 1. 2. Doména

Objekty typu doména umožňují přenos datových bloků libovolné délky pomocí segmentace dat. Mohou být základní nebo mnohonásobné. Klient je vždy iniciátorem přenosu dat. Doména má tyto vlastnosti :

- jméno
- uživatelský typ: „Client" nebo „Server"
- třída: „Basic" nebo „Multiplexed"
- priorita: hodnota v rozsahu [0, 7]
- omezení času: n*100 μsec, n>>0

Lokální služby umožňují definice domén, síťovou inicializaci přenosu domény, vlastní přenos dat a možnost přerušení přenosu. Přenos dat je rozdělen do dvou fází. Prvotně si klient se serverem vymění informace o typu přenosu (normální či urychlený). V druhé fázi si již vyměňují informace o délce dat, zda bude následovat další výměna a vlastní data. Jak klient tak i server mohou z vlastních důvodů přenos přerušit.

2. 1. 3. Událost

Objekty typu událost jsou vhodné pro modelování asynchronního chování zařízení. Události se rozdělují na neřízené - událost je uzlem ihned oznámena, řízené - klient má možnost povolit nebo zakázat oznamování událostí a ukládané - událost je lokálně uložena a klient má možnost si událost přečíst. Atributy události jsou :

- jméno
- uživatelský typ: „Client" nebo „Server"
- třída: „Controlled", „Uncontrolled" nebo „Stored"
- datový typ
- priorita: hodnota v rozsahu [0, 7]
- omezení času: n*100 μsec, n>>0

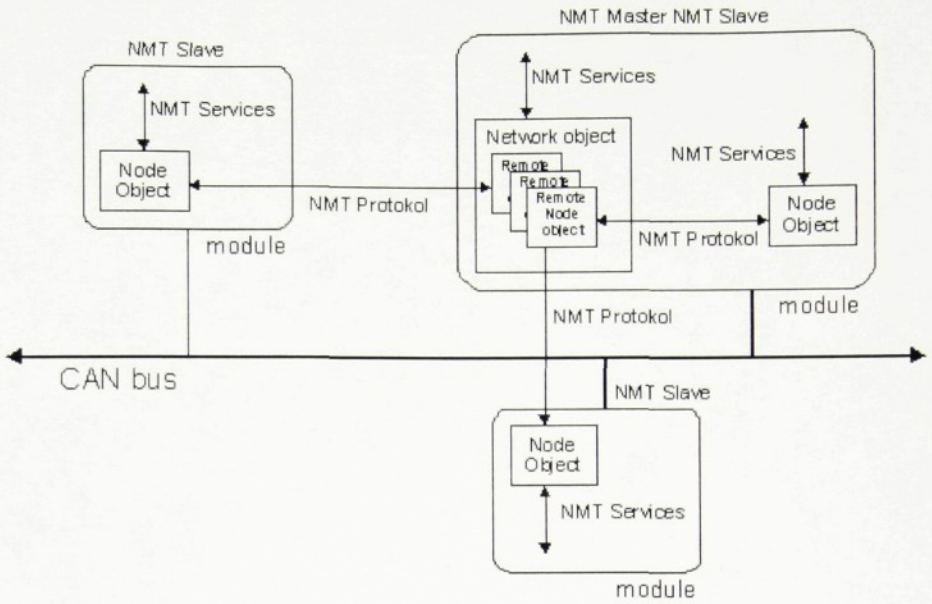
Lokální služby umožňují definici události a obnovení hodnot, síťové pak čtení události, oznámení události, povolení nebo zákaz oznamování a žádost o uložení události. Pomocí objektu událost je možné přenést data o maximální délce 8 bytů.

2. 2. NMT

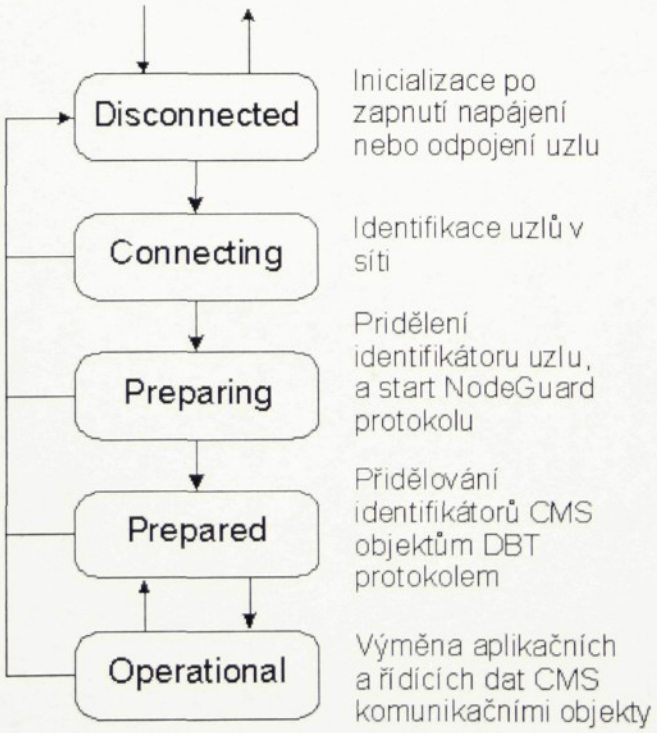
Servisní služby NMT se používají ke koordinaci a řízení činnosti jednotlivých modulů. Jsou rozděleny do třech základních skupin, řízení činnosti sítě (připojení, odstavení modulu, zahájení výměny dat CMS protokolem, atd.), konfigurace parametrů modulů (protokol umožňuje přenos datových bloků libovolné délky mezi NMT Master a modulem NMT Slave) a detekci chybových stavů (protokol *Node Guarding*).

Protokol definuje různé třídy zařízení pro moduly typu Master i Slave z hlediska počtu implementovaných služeb (řízení činnosti, řízení činnosti + detekce chyb a pod.). V síti existuje jediný modul NMT Master, který řídí činnost ostatních modulů NMT Slave. Master si udržuje databázi připojených modulů Slave s jejich aktuálními stavy, které testuje prostřednictvím služeb detekce chybových stavů. NMT model definuje tři druhy objektů (viz obr.10) - *network object* (jediný v síti, reprezentuje databázi všech modulů Slave v síti), *remote node object* (zastupuje stav připojeného modulu Slave) a *node object* (vlastní modul Slave z hlediska NMT).

NMT Master řídí přechod uzlu NMT Slave mezi jednotlivými stavy (obr. 11). Protokol využívá COB-ID = 0 (zpráva s nejvyšší prioritou) pro změnu stavu uzlu a COB-ID = 2026,2025,2022 pro konfiguraci parametrů. Vlastní uzel Slave je adresován ve stavu *disconnected* prostřednictvím jedinečné adresy (NMT adresa - možno měnit LMT servisní službou) nebo přiděleným identifikátorem *Node-ID* (v ostatních stavech) v rozsahu 1..255, nebo-li v síti může rozlišeno maximálně 255 modulů. Pokud modul implementuje protokol detekce chybových stavů je každému modulu Slave přidělen vlastní COB-ID (pro protokol *Node Guarding*), kterým má Master možnost testovat stavy modulů Slave a naopak modul Slave si může ověřovat živost nadřízeného sytému a v případě jeho výpadku může přejít do předdefinovaného stavu nebo zahájit autonomní činnost.



Obr. 10: Model síťové správy



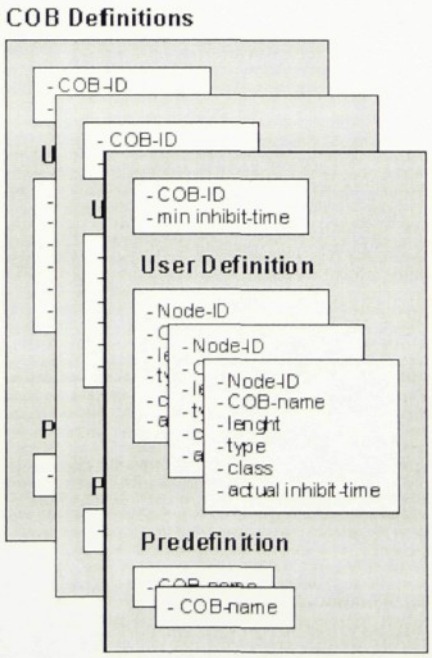
Obr. 11: Stavový diagram uzlu NMT Slave

2. 3. DBT

Služba DBT má na starosti přidělování identifikátorů komunikačním objektům vytvořených CMS a jejich správu. Při přidělování je nutné zamezit přidělení různých identifikátorů stejnému komunikačnímu objektu nebo stejného identifikátoru různým komunikačním objektům. Specifikace DBT rozlišuje 3 druhy přidělování COB-ID (v závislosti na DBT třídě zařízení):

- Standardní přidělení – pevně přidělené identifikátory výrobcem zařízení. Nelze změnit.
- Statické přidělení – identifikátory přidělené výrobcem zařízení. Dají se měnit pomocí výrobcem určených metod .
- Dynamické přidělení – COB-ID jsou přiřazeny jednotlivým zařízením přes síť služeb DBT.

Databáze objektů DBT (viz. obr. 12) se skládá z jednotlivých záznamů, kde ke každému COB-ID jsou přiřazeny jména a atributy objektů CMS využívající dané COB-ID.



Obr. 12: Databáze DBT objektů

V síti se nachází jediný uzel DBT Master jehož úkolem je udržovat databázi objektů DBT, kontrolovat souvztažnosti mezi jednotlivými objekty a poskytovat služby uzlům DBT Slave . Podpora služeb DBT je volitelná. Výsledné rozdělení identifikátorů dle specifikace CAL je zřejmé z tabulky č.2.

Tab. č. 2: Rozdělení identifikátorů protokolem CAL

CAL protokol	CAN identifikátor
NMT start/stop služby	0
CMS objekt priority 0	1 - 220
.....	
CMS objekt priority 7	1541 - 1760
NMT Noard Guard protokol	1761 - 2015
rezervováno	2016 - 2019
LMT služby - odpověď	2020
LMT služby - žádost	2021
NMT identifikace uzlu	2022
DBT služby - odpověď	2023
DBT služby - žádost	2024
NMT služby - odpověď	2025
NMT služby - žádost	2026
rezervováno	2027 - 2031

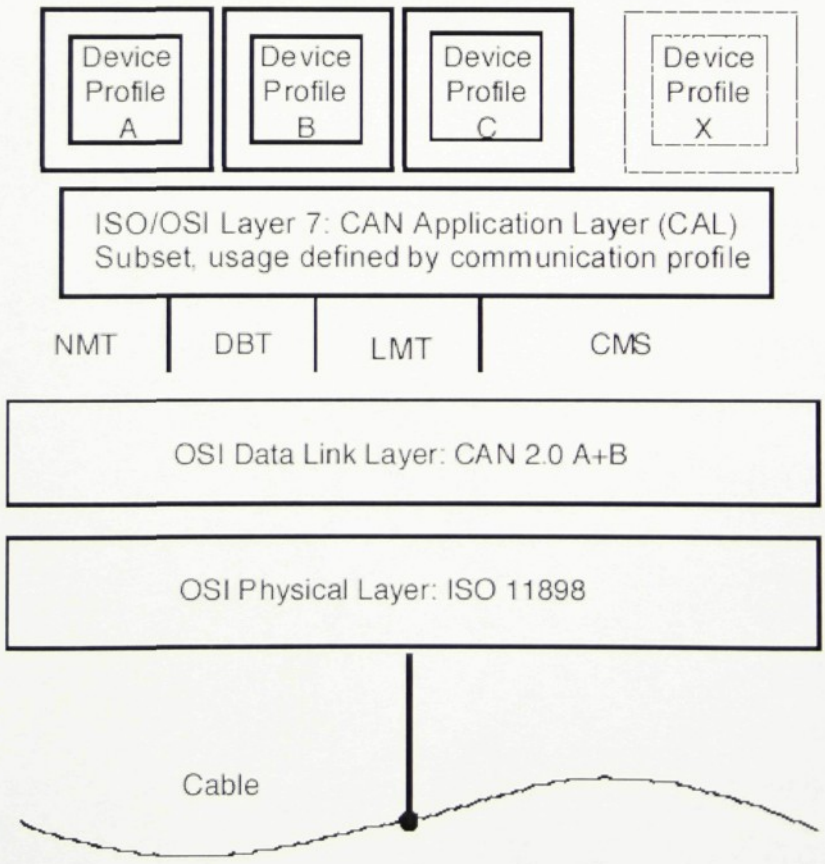
2. 4. LMT

Servisní služba LMT využívá Master-Slave komunikační model a umožňuje změnu základních parametrů modulu, ke kterým patří komunikační rychlost, adresa NMT a adresa LMT, kterou je uzel LMT Slave adresován uzlem Master. Adresa LMT se skládá ze jména výrobce, jména výrobku a sériového čísla. Změna komunikační rychlosti probíhá ve dvou krocích, nejprve je modulu zadán index do tabulky předdefinovaných hodnot komunikačních rychlostí a poté je modul aktivován zprávou obsahující čas zpoždění, kdy ke změna skutečně dojde.

Adresa NMT musí být pro každý modul v síti jedinečná a pokud modul neimplementuje služby LMT musí existovat jiný mechanizmus pro její změnu či bezkonfliktní nastavení před prvním začleněním modulu do sítě.

3. CANopen

Komunikační protokol CANopen vytváří rozhraní mezi CAN aplikační vrstvou a CANopen profilem zařízení. Zahrnuje model komunikace v reálném čase a protokoly, které jsou společné pro všechna zařízení v síti. Profily zařízení jsou společné pro zařízení se specifickými funkcemi. Účelem definice standardních profilů zařízení je poskytnutí jednotné koncepce nezávislým výrobcům zařízení a tím vytvoření jednotného modulárního systému s možností integrace a vzájemné náhrady modulů různých výrobců bez nutnosti změn a zásahů do vytvořeného systému. Každé takové vyráběné zařízení proto musí splňovat minimální požadavky dané definovaným standardem a případně může být rozšířeno o další specifické vlastnosti. Komunikační koncept je popsán pomocí referenčního modelu ISO-OSI (obr. 13). V podstatě CANopen popisuje způsob, jakým zařízení používá podmnožiny protokolu CAL a jeho služby.



Obr. 13: CANopen komunikační referenční model

3. 1. Objekt slovník

Nejdůležitější částí profilu zařízení je objekt slovník (The Object Dictionary). Objekt slovník (tab. č. 3) je základní skupinou objektů, které jsou přístupné přes sít definovaným způsobem. Každý objekt uvnitř slovníku je adresován pomocí 16 bitového indexu. Standardní objekt slovník obsahuje tedy maximálně 65 536 záznamů.

Tab. č. 3: Struktura objektu slovník

Index (hex)	Object
0000	not used
0001-001F	Static Data Types
0020-003F	Complex Data Types
0040-005F	Manufacturer Specific Data Types
0060-007F	Device Profile Specific Static Data Types
0080-009F	Device Profile Specific Complex Data Types
00A0-0FFF	Reserved for further use
1000-1FFF	Communication Profile Area
2000-5FFF	Manufacturer Specific Profile Area
6000-9FFF	Standardised Device Profile Area
A000-FFFF	Reserved for further use

Typy statických dat od indexu 1h až 1Fh obsahují definice pro standardní typy dat jako *boolean*, *integer*, *floating point*, *string* atd. Tyto záznamy jsou určeny pouze pro reference a nemohou být ani čteny či zapisovány. Komplexní typy dat od indexu 20h až 3Fh jsou předdefinované struktury skládající se ze standardních dat a jsou společné pro všechna zařízení. Typy dat specifikované výrobcem od indexu 40h až 5Fh jsou také struktury skládající se ze standardních dat, ale jsou určené pro daný speciální typ zařízení. V profilu zařízení smíme definovat dodatečné specifické typy dat vztahující se ke konkrétnímu typu zařízení. Typy statických dat definovaných profilem zařízení jsou od indexu 60h až 7Fh, typy komplexních jsou od 80h až 9Fh. Profil komunikační oblasti je od indexu 1000h až 1FFFh, obsahuje specifické komunikační parametry pro síť CAN. Tyto

položky jsou společné všem zařízením. Oblast standardního profilu zařízení od indexu 6000h až 9FFFh obsahuje všechny objekty dat společné pro třídu zařízení, které mohou být čteny nebo zapisovány přes sít. Objekt slovník má pro každý typ zařízení rozsah povinných položek. Tyto položky zaručí, že všechna zařízení konkrétního typu fungují definovaným způsobem.

Položky objektu slovníku jsou adresovány pomocí 16 bitového indexu. V případě jednoduché proměnné je odkaz přímý. Pokud se jedná o strukturu nebo pole odkazujeme se na celou datovou strukturu. Jednotlivé položky struktury jsou přístupné přes 8 bitový sub-index. Pro jednoduché proměnné je tento sub-index roven vždy nule. U strukturované proměnné udává sub-index 0 počet položek dané struktury. Například máme zařízení, které využívá rozhraní RS232. Na indexu 6092h je struktura (tab. č. 4), která definuje komunikační parametry modulu. Tato struktura obsahuje rychlost linky, počet start/stop bitů a typ parity. Položky přístupné pomocí sub-indexu mohou být různého datového typu.

Tab. č. 4: Použití indexu a sub-indexu

Main Index	Sub Index	Variable Accessed	Data Type
6092	0	Number of Entries	Unsigned8
	1	Baud Rate	Unsigned16
	2	Number Of Data Bits	Unsigned8
	3	Number Of Stop Bits	Unsigned8
	4	Parity	Unsigned8

3. 2. Komunikační model

Komunikační model specifikuje služby, rozdíly v komunikaci objektů a dostupné způsoby pro aktivaci přenosu zpráv. Komunikační model podporuje přenos synchronních i asynchronních zpráv. Synchronní přenos zpráv je podporován definovanými objekty (*Sync message, Time stamp message*). Synchronní zprávy jsou přenášeny s ohledem na definované synchronizační zprávy, asynchronní zprávy mohou být odvysílány kdykoliv.

Rozlišujeme čtyři typy zpráv, s ohledem na jejich funkčnost:

- *Administrative messages* (LMT, NMT, DBT)
- *Service Data Messages*
- *Process Data Messages*
- *Pre-defined Messages* (synchronizace, časová známka, nouzové zprávy)

První skupinu tvoří administrativní zprávy pro inicializaci, konfiguraci a kontrolu zařízení. Jednotlivé služby a protokoly jsou dané dle specifikace CAL. Zprávy služby dat (*Service Data Messages)* se používají pro čtení a zápis položek objektu slovník. Pomocí zpráv procesu dat (*Process Data Messages*) je podporován přenos dat v reálném čase. Poslední skupinou jsou předdefinované zprávy (*Pre-defined Messages*), které slouží k synchronizaci uzlů, časové korekci a ohlášení nouzového stavu. Implementace poslední skupiny objektů je volitelná.

3. 2. 1. SDO

Objekt SDO (*Service Data Object*) poskytuje přístup do objektu slovník pomocí indexu a sub-indexu. Zprávy mají nízkou prioritu a jsou přenášeny asynchronně. SDO je representován v CMS jako „Multiplexed Domain". SDO vytváří komunikační kanál mezi dvěma zařízeními. Majitel zpřístupněného objektu slovník je server SDO. Jeden objekt SDO je přednastavený případ, zařízení však smí podporovat i více jak jeden objekt SDO. Následující atributy specifikují objekt SDO:

- Jméno: „SDO_xxx" (xxx je číslo objektu SDO, xxx = 001 – 128 server SDO, xxx = 129 –256 klient SDO)
- Uživatelský typ: klient nebo server (majitel zpřístupněného objektu slovník je server)
- Třída: „Multiplexed Domain"
- Priorita: specifikována aplikací, doporučeno mezi [6,7]
- Datový typ multiplikátoru: struktura obsahující index (16 bitu) a sub-index (8 bitu)

Pro přenos dat s délkou menší než 5 bytu se používá urychlený typ, kdy se data vymění již v inicializační fázi. Pokud je třeba přenést data o velikosti vetší než 4 byty,

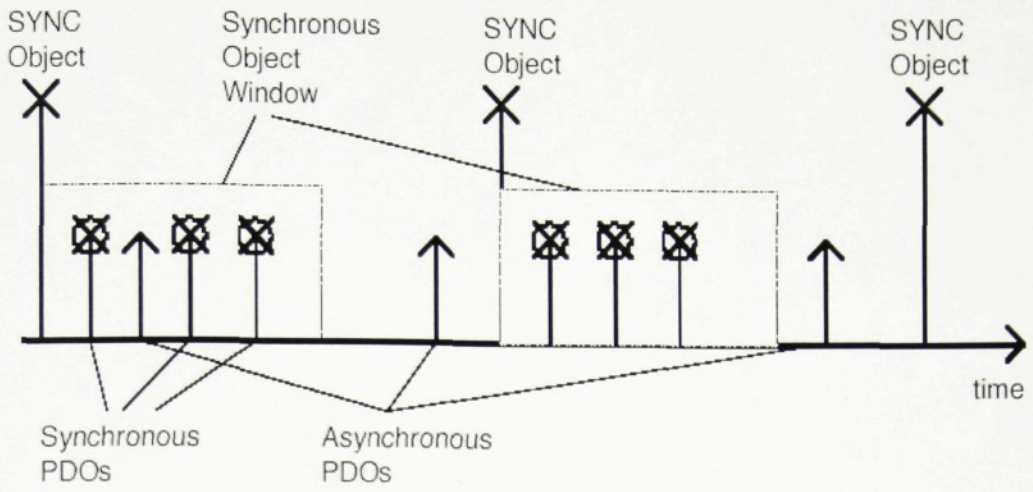
používá se normální typ přenosu. V případě přerušení přenosu je odvysílána zpráva, obsahující 4 byty specifikující důvod přerušení. První byte obsahuje třídu chyby, druhý kód chyby a poslední dva obsahují přídavný kód. Kombinací třídy a kódu chyby je vysvětlena chyba, která nastala (např. sub-index neexistuje, nesouhlasí datový typ, pokus o zápis do parametru pro čtení atd.). Přídavný kód je potřebný u typu chyb, kde požadujeme detailní informaci (např. zapisovaná hodnota je příliš velká atd.). Podrobný přehled kombinací chyb je uveden v příloze č. 2.

3. 2. 2. PDO

Objekt PDO (*Process Data Object*) se používá pro výměnu dat v reálném čase. Má typicky vysokou prioritu zpráv, které se přenáší synchronně i asynchronně. PDO odpovídá položkám v objektu slovník. Položky pro přenos jsou mapovány uvnitř zvláštní struktury objektu slovník (*PDO Mapping Parameter*). Je zde uveden datový typ položky a její index. PDO je reprezentován v CMS jako objekt typu „Stored Event". Následující atributy události v CMS jsou specifické pro PDO:

- Jméno: „xPDOyyy" (yyy = číslo PDO, x = {T,R} vysílá nebo přijímá PDO)
- Uživatelský typ: server nebo klient (server vysílá data, klient přijímá nebo žádá o data)
- Třída: „Stored Event"
- Priorita: specifikována aplikací, doporučeno mezi [2,5]
- Datový typ: odpovídá mapovaným položkám objektu slovník
- Omezení času: specifikováno aplikací

V případě synchronních zařízeních je synchronní objekt (*SYNC object*) vysílán periodicky aplikací. Princip synchronního a asynchronního přenosu je znázorněn na obr. 14. Synchronní PDO je vysláno uvnitř definovaného časového okénka ihned po odvysílaní objektu SYNC. Perioda pro přenos PDO je specifikovaná pod položkou objektu slovník (*PDO Communication Parameter*). Synchronní přenos PDO může být cyklický nebo acyklický. Pokud je acyklický, je perioda nulová. Asynchronní PDO smí být vysláno kdykoliv, bez ohledu na objekt SYNC.



Obr. 14: Synchronní a asynchronní přenos PDO

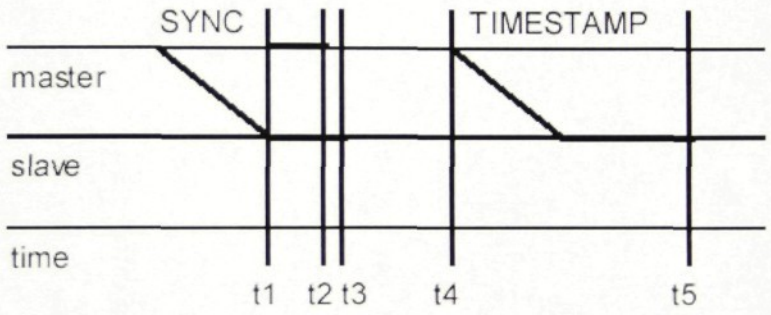
3. 2. 3. Synchronizace

Některé časově kritické aplikace, zejména v rozsáhlých sítích, vyžadují přesnou synchronizaci. Může být nutné synchronizovat lokální hodiny jednotlivých zařízení s přesností v řádu mikrosekund nebo je požadována vyšší přesnost společné časové osy. Pro tyto účely je definovaný protokol, který pomocí časové známky seřídí nevyhnutelný posun lokálních hodin.

Obr. 15 ukazuje časový průběh protokolu. V čase t1 začíná Master generovat objekt SYNC (to trvá do doby t2). Poté v čase t4 pošle Master časovou známku obsahující správný čas t1. Ostatní zařízení (Slave) mají lokální časovou známku (t3) pro přijetí zprávy SYNC a mohou si jí porovnat s přijatým časem t1. Rozdíl mezi hodnotami určuje množství času k seřízení lokálních hodin. U tohoto postupu je pouze lokální zpoždění (t2-t1 Master a t3-t1 Slave) časově kritické. Toto zpoždění závisí na lokálních parametrech (hardware, časy přerušení procesoru atd.).

Objekt SYNC poskytuje základní síťové hodiny. Časová perioda mezi dvěma objekty SYNC je určena standardním parametrem – komunikační perioda (index 1006h), který může být zapsán aplikací do objektu slovník během inicializačního procesu. Abychom garantovali objektu SYNC přístup na sběrnici CAN, je definována vysoká priorita objektu. SYNC objekt je reprezentován v CMS jako objekt typu „Basic Variable" s následujícími parametry:

- Jméno: „SYNC000“
- Uživatelský typ: server nebo klient (přijímací zařízení je server, synchronizační zařízení je klient)
- Třída: „Basic Variable“
- Priorita: 0 (doporučeno)
- Datový typ: NIL
- Typ přístupu: „Write_Only“
- Omezení času: komunikační perioda (index 1006h)



Obr. 15: Synchronizace s vysokým rozlišením

Objekt časové známky (Time Stamp Object) je definovaný s následujícími parametry:

- Jméno: „TIME000“
- Uživatelský typ: server nebo klient (poskytovatel známky je server, konzument známky je klient)
- Třída: „Stored Event“
- Priorita: 1 (doporučeno)
- Datový typ: „TIME-of-DAY“
- Omezení času: specifikováno aplikací

3. 2. 4. Pohotovost

Nouzové zprávy jsou odvysílány s vysokou prioritou při výskytu vnitřní chyby zařízení. Nouzový telegram však smí být poslán jen jednou při výskytu chyby, nesmí se opakovat. Jestliže se žádné nové chyby nevyskytují, nesmí se nouzová zpráva odeslat. Komunikační profil CANopen definuje chybový kód, chybový registr a doplňují informaci, specifikovanou v profilu zařízení. Tyto tři složky se dohromady přenáší v nouzové zprávě. Chybový kód popisuje zda se jedná o chybu např. napětí, teploty, hardware, software atd. Kompletní přehled chybových kódů je v příloze č. 3. Pokud zařízení opraví danou chybu nebo je po inicializaci, odvysílá nouzovou zprávu s nulovým chybovým kódem.

Objekt nouzového stavu (*Emergency object*) je volitelný. Pokud ho zařízení podporuje, musí obsahovat nejméně dva chybové kódy (00h – bez chyby, 01h – nastala chyba), ostatní jsou volitelné. Pro objekt nouzového stavu jsou definované vlastnosti:

- Jméno: „EMCY000"
- Uživatelský typ: server nebo klient (server oznamuje chybu, ostatní zařízení jsou klienti)
- Třída: „Stored Event"
- Priorita: 0 nebo 1 (doporučeno)
- Datový typ: struktura obsahující chybový kód (16 bit), chybový registr (8 bit), výrobcem definovaný chybový kód (5 x 8 bit)
- Omezení času: specifikováno aplikací

III. REALIZACE APLIKACE

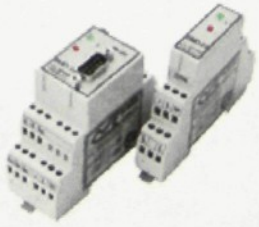
Prvním úkolem bylo vytvoření aplikace pro sledování vybraných veličin měřicích modulů SMD firmy KMB s.r.o. pomocí karty IXXAT (pro práci se sběrnicí CAN). Aplikační vrstva CAN je v tomto případě vytvořený firemní protokol. Následně pak rozšířit komunikaci modulů o aplikační vrstvu definovanou protokolem CANopen a upravit vytvořenou aplikaci pro daný protokol.

1. Použité hardwarové prostředky

Řídící jednotkou měřicích zařízení SMD je mikroprocesor Siemens C515C. Jednotlivá zařízení SMD tvoří na sběrnici jednotky typu Slave. Jednotka typu Master je aplikace na osobním počítači, která má přístup na sběrnici CAN pomocí karty IXXAT.

1. 1. Zařízení SMD

Měřící přístroje SMD 01/34 (obr. 16) firmy KMB s.r.o. slouží pro sledování a záznam napětí a proudů. Kromě měření skutečných efektivních hodnot napětí a proudů zaznamenávají také skutečnou hodnotu účiníku a frekvenci. V maximální konfiguraci umožňuje připojení až tří napěťových vstupů (230 V stř. se společným bodem) a čtyř proudových vstupů (max. 1 A). Přístroje lze napájet odděleným napětím 24 V ss. nebo 230 V stř. / 50 Hz (podle provedení). Přístroje obsahují 8 bitový mikroprocesor Siemens C515, dvě paměti typu FLASH pro uchování programového kódu a paměť typu RAM pro proměnné.



Obr. 16: Měřící přístroje SMD34-01

Dále přístroje SMD obsahují obvod reálného času, který je podobně jako paměť dat zálohován vestavěným akumulátorem. Akumulátor zajišťuje uchování dat během výpadku napájecího napětí.

Přístroje mohou komunikovat po třech komunikačních rozhraních a to RS232, RS485 a CAN. Rozhraní RS232 se používá především pro servisní účely (např. modifikace programového kódu). Maximální délka linky je do 15m a zároveň lze komunikovat pouze s jedním přístrojem. Rozhraní "RS485" je obdobou "RS232", ale lze nastavit rychlost komunikace až do 57,6 kbit/s. Na jednu linku lze připojit až 32 účastníků současně a délka linky může dosáhnou délky až 1,2 km. Maximální komunikační rychlost rozhraní CAN je odvozena především od délky komunikační linky a kvality kabelového vedení.

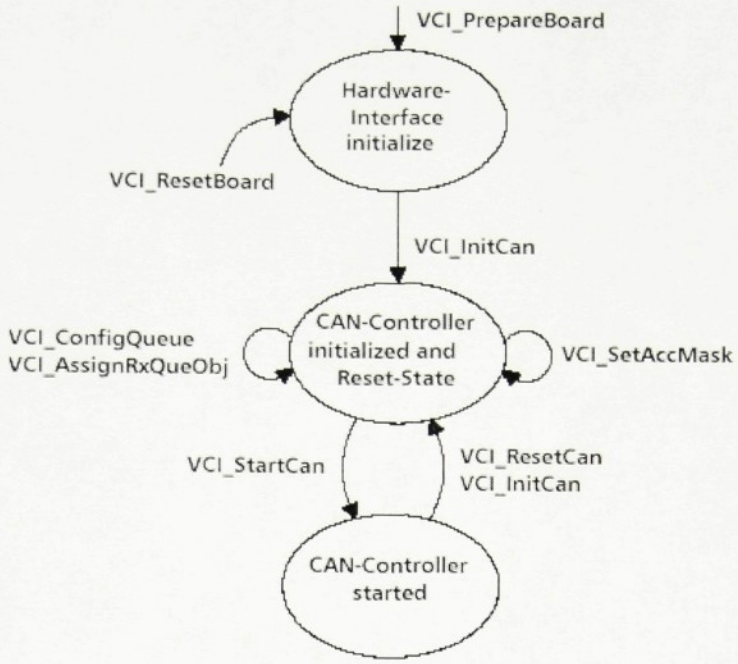
Zdrojový kód pro přístroje SMD se upravuje v prostředí µVision V2.10 a pomocí programu *Loader* firmy KMB s.r.o. se přes linku RS232 přehraje do přístroje.

1. 2. Karta IXXAT pro přístup na CAN

Karta IXXAT (iPC-I 320/PCI) slouží k přístupu na sběrnici CAN pomocí osobního počítače. Karta se ovládá přes softwarové rozhraní VCI (*Virtual CAN Interface*), které umožňuje současnou funkci několika rozdílných karet pro sběrnici CAN. Pro práci s kartou IXXAT je nutné, aby na daném počítači byly nainstalovány dynamické knihovny dodávané spolu s kartou.

Pomocí funkcí VCI se nastavuje typ karty, hodnota přerušení IRQ a její umístění (ISA, PCI, ...). Pro práci se zprávami lze použít vstupní/výstupní pole (*buffer*) nebo frontu. Fronta je z hlediska uživatele výhodnější, protože v případě pole se s příchodem nové zprávy automaticky přepisuje stará zpráva, přestože nebyla ještě přečtena. Stavový diagram pro inicializaci karty je na obr. 17.

Uživatel nastavuje v inicializaci parametry sběrnice CAN (přenosovou rychlost, délku identifikátoru) a vstupní/výstupní fronty (maska zpráv, velikost fronty, časové rozlišení). Pokud je zpráva zapsána do výstupní fronty, je odeslána ihned poté, kdy je sběrnice ve stavu *Bus Free*. Pro čtení příchozích zpráv používáme funkci, která vrací počet nových zpráv vstupní fronty. Následně již čteme daný počet zpráv. V aplikaci tuto funkci voláme opakovaně v dostatečně malém intervalu, což vytváří dojem spojitého příjmu příchozích zpráv.



Obr. 17: Stavový diagram IXXAT

1. 3. Mikroprocesor C515C

8 bitový mikroprocesor C515C s oscilační frekvencí 10 MHz obsahuje kromě řadiče pro sběrnici CAN také 10 bitový analogově digitální převodník, paralelní vstupy a výstupy, časové spínače, čítače a sériové rozhraní. Objekty zpráv a řídící registry pro sběrnici CAN zabírají celkem 256 bytů paměti mikroprocesoru. Řídících registrů je celkem šestnáct. Od adresy F706h až F70Fh jsou registry pro nastavení vstupní masky jak standardního i rozšířeného identifikátoru. Hodnota vstupní masky je 0xFF, nebo-li porovnáváme každý bit identifikátoru příchozí zprávy. Dále nastavujeme hodnotu dvou registrů BTR0 a BTR1, které určují přenosovou rychlost sběrnice. Pro použitou rychlost 125 kbit/s obsahuje registr BTR0 hodnotu 0xC3 a BTR1 0x6B.

Přerušovací registr IR určuje zdroj přerušení, kterým může být jakýkoliv objekt zprávy (příjem nebo vysílání) nebo fyzická chyba sběrnice. Stavový registr SR slouží k indikaci o úplném odeslání či přijmutí zprávy. Řídící registr CR umožňuje aktivaci přerušení a inicializaci řadiče. Celkem všechny řídící registry zabírají 16 bytů paměti.

2. Protokol firmy KMB s.r.o.

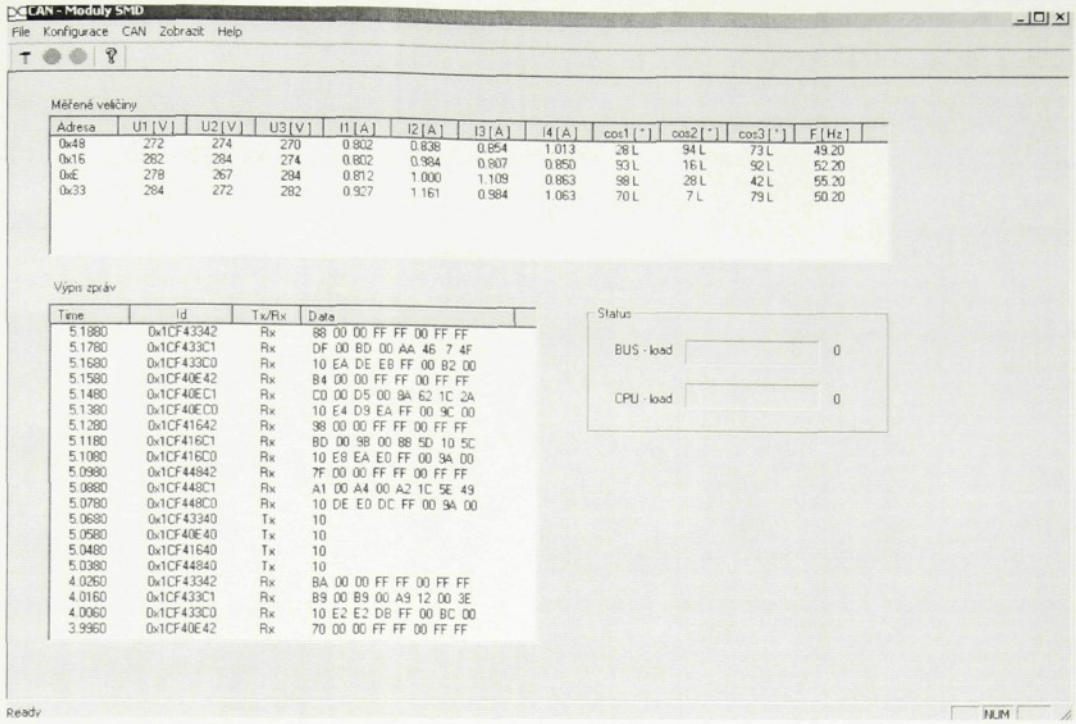
Firemní protokol vytvářející vlastně aplikační vrstvu používá 28 bitový identifikátor. Jsou definovány jisté funkce (např. konfiguruj zařízení, předej aktuální data, nastav omezení atd.), kterým je napevno přiděleno číslo. Tyto zprávy se nazývají RAC a jsou orientovány na adresu uzlu. U každé této zprávy je definován formát žádosti a odpovědi. Pro předání základních informací se využívá 28 bitový identifikátor, v kterém jsou následující údaje (význam jednotlivých bitů):

- 28 – 27 ... globální či specifická adresa
- 26 – 19 ... číslo RAC
- 18 ... příkaz od Master či odpověď od Slave
- 17 – 8 ... adresa konkrétního přístroje
- 7 ... poslední rámec
- 6 ... rezerva
- 5 ... potvrzení úspěšného provedení v odpovědi
- 4 – 0 ... číslo rámce či bližší specifikaci chyby

Použili jsme zprávu *GetIdentification* pro identifikaci zařízení (typ modulu a jeho adresa na sběrnici CAN). Zpráva *ActData* předává aktuální hodnoty napětí, proudů, účiníků a frekvenci. Hodnoty převodních konstant pro proudy jsou předány ve zprávě *GetConfig*. Tyto tři zprávy jsou postačující pro vytvoření aplikace, která provádí monitorování měřených veličin modulů.

2. 1. Realizace aplikace dle protokolu firmy KMB s.r.o.

Aplikace (obr. 18) je vytvořená v prostředí MS Visual C++. Uživatel má po spuštění aplikace možnost si vybrat mezi simulací hodnot (následně i počtu simulovaných modulů) nebo reálného přenosu dat. Při simulaci se hodnoty měření náhodně generují, aby rámcově odpovídali skutečným. V horní části aplikace se spolu s aktuálními hodnotami veličin zobrazuje i adresa modulu, která jednoznačně identifikuje zařízení (v případě simulace je náhodná). Pod aktuálními daty se zobrazuje stav sběrnice CAN, respektive posledních 20 zpráv. Poslední vizuální položka je situace zatížení mikroprocesoru na kartě IXXAT a sběrnice.



Obr. 18: Visuální podoba aplikace

Při výpisu aktuálních zpráv na sběrnici se zobrazuje čas odeslání zprávy, její identifikátor, typ zprávy (příkaz od PC – Tx nebo odpověď od SMD - Rx) a vlastní data. Čas příchozích zpráv je obsažen ve vstupní frontě a jeho rozlišení je dáno při inicializaci karty IXXAT. Abychom zjistili čas odesílaných zpráv, jsou spuštěny spolu s aktivací komunikace po sběrnici také vnitřní hodiny aplikace se stejným časovým rozlišením. Po ukončení monitorování veličin je možnost zobrazit výpis všech zpráv, které byly vyslány na sběrnici.

Aplikace je v podstatě modul typu Master, který požaduje po ostatních modulech typu Slave (SMD01/34) informace. Na počátku algoritmu se nejprve na příkaz aplikace identifikují všechny moduly a poté jsou jednotlivě dotazovány na hodnotu aktuálních dat. Časový interval mezi dvěma po sobě jdoucími dotazy na aktuální data je stanoven na jednu sekundu. Hodnota intervalu je dostačující pro počet modulů typu Slave, které jsem měl k dispozici (z výrobních důvodů firmy pouze dva). Pokud by byl jejich počet mnohonásobně větší, bylo by nutné interval zvětšit a popřípadě zvýšit i přenosovou rychlost sítě.

Algoritmus aplikace využívá funkci časového spínání jednotlivých kroků. Master vyšle zprávu a spustí daný časový spínač. Po uplynutí nastaveného intervalu se přečtou a vyhodnotí zprávy vstupní fronty (např. hodnoty aktuálních dat), vyšle se nová zpráva a opět se spustí časový spínač. Tento princip se také využívá při inicializaci zařízení, kdy je nutná časová prodleva, pro přihlášení všech zařízení na sběrnici.

Zprávy o vypršení intervalu daného časového spínače jsou umístěny ve frontě zpráv operačního systému spolu s ostatními uživatelskými zprávami (např. stisk tlačítka myši, klávesnice). Uživatel aplikace by mohl těmito zprávami frontu úplně zahltit a následně pozastavit chod aplikace. Z tohoto důvodu se používá pro časové spínače oddělená fronta, která zachytává pouze zprávy časových spínačů a je nezávislá na frontě uživatelských zpráv. Obsluha této fronty je realizována touto funkcí (zdrojový kód C++):

```
void CALLBACK EXPORT TimerProc(HWND hWnd,UINT nMsg,UINT nIDEvent,DWORD
dwTime){
CMainFrame* m_pWindow;
m_pWindow = (CMainFrame*) AfxGetApp()->GetMainWnd();

switch (nIDEvent){
      case 1 : KillTimer(hWnd,1);              // Inicializace zarizeni
               m_pWindow ->Identifikace();
               break;

      case 2 : KillTimer(hWnd,2);              // Zadost o aktualni data
               m_pWindow ->Aktual_hodnoty();
               break;

      case 3 : KillTimer(hWnd,3);              // Zobrazeni merenych hodnot
               m_pWindow ->Zobraz_hodnoty();
               break;

      case 4 : m_pWindow ->Status();           // Zobrazeni karty IXXAT
               break;

      case 11: m_pWindow ->Simulace_hodnot();// Simulace hodnot a zpráv
               break;
      }
}
```

Vytvořená aplikace i její zdrojový kód jsou na přiloženém disku CD.

3. Implementace protokolu CANopen

Protokol CANopen specifikuje minimální schopnosti zařízení (*Minimum Capability Device*), které musí každé zařízení podporovat, aby mohlo spolupracovat s ostatními. Jsou požadovány následující vlastnosti:

- Modul – ID
- Objekt slovník (závisí na funkci zařízení)
- Jeden SDO objekt, podporující povinné položky
- Upravený stavový diagram pro NMT Slave
- Předdefinované alokační schéma identifikátorů

Kromě těchto povinných minimálních vlastností jsou implementovány v modulech SMD také synchronizační objekty, chybový objekt a protokol *Node Guarding*. Z důvodů ulehčení počátečního konfigurování sítě je navrženo rozdělení identifikátorů (tab. č. 5).

Tab. č. 5: Podporované objekty a jejich identifikátory

object	function code (binary)	resulting COB-ID	Communication Parameters at Index	CMS priority group
NMT	0000	0	-	0
SYNC	0001	128	(1005h)	0
TIME STAMP	0010	256	-	1

object	function code (binary)	resulting COB-IDs	Communication Parameters at Index	CMS priority group
EMERGENCY	0001	129 - 255	-	0 , 1
PDO1 (tx)	0011	385 - 511	1800h	1 , 2
PDO1 (rx)	0100	513 - 639	1400h	2
PDO 2 (tx)	0101	641 - 767	1801h	2 , 3
PDO2 (rx)	0110	769 - 895	1401h	3 , 4
SDO (tx)	1011	1409 - 1535		6
SDO (rx)	1100	1537 - 1663		6 , 7
Nodeguard	1110	1793-1919	(100Eh)	-

Tyto identifikátory jsou platné ihned po inicializaci zařízení a mohou býti následně dynamicky změněny (služba DBT). Dle tohoto rozdělení se může být na sítí až 127 uzlů.

Používáme 11 bitový identifikátor, kde první 4 významově důležitější bity určují funkční kód a zbylých 7 bitů určuje adresu modulu.

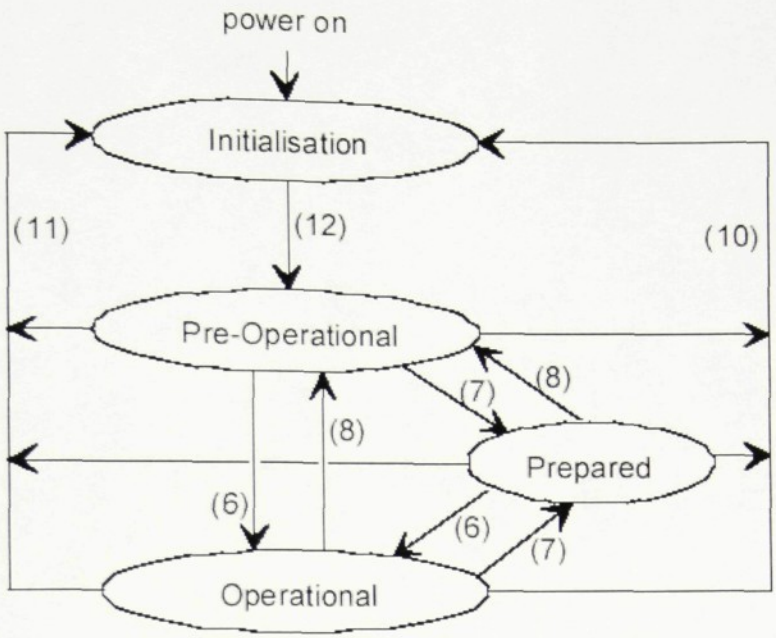
3. 1. Stavový diagram uzlu

Stavový diagram uzlu typu NMT Slave je na obr. 19. Zařízení se může nacházet v jednom ze čtyř definovaných stavů. Po uvedení zařízení do provozu se nachází v inicializačním stavu (*Initialisation*), kde se načítají předdefinované hodnoty identifikátorů a položek v objektu slovník. Poté zařízení automaticky přejde (12 viz. obr. 19) do před-operačního stavu (*Pre-Operational*), kdy se mohou číst nebo zapisovat položky objektu slovník pomocí objektů SDO. V tomto stavu zařízení ohlašuje svou přítomnost na síti a spouští se protokol *Node Guarding*.

Dále se v před-operačním stavu aktivuje synchronizační objekt. Po přepnutí zařízení do operačního stavu (*Operational*) se aktivují objekty PDO. Pokud je třeba pozměnit položky v objektu slovník daného zařízení, musíme jej nejprve přepnout do stavu připravenosti (*Prepared*), kde nedochází k přenosu objektů PDO. Zařízení lze přepnout do stavu inicializace dvěma příkazy. V prvním případě se jedná o restart komunikačních parametrů (např. zařízení neodpovídá na dotaz v protokolu *Node Guarding*), v druhém jde o kompletní restartování celého zařízení (např. uvedení jednotného stavu na všech zařízeních).

Přepínání mezi jednotlivými stavy řídí modul typu NMT Master (v tomto případě PC) zprávami s identifikátorem 0h. Datová oblast těchto zpráv se skládá ze dvou bytů. První určuje přepnutí stavu a druhý obsahuje adresu uzlu, pro který je tato zpráva určena. Pokud je adresa uzlu rovna nule, je tato zpráva společná pro všechna zařízení typu NMT Slave na síti. Hodnoty prvního bytu pro změnu stavu zařízení jsou:

- 01h – *Start_Remote_Node* (6)
- 02h – *Stop_Remote_Node* (7)
- 80h – *Enter_Pre-Operational_State* (8)
- 81h – *Reset_Node* (10)
- 82h – *Reset_Communication* (11)

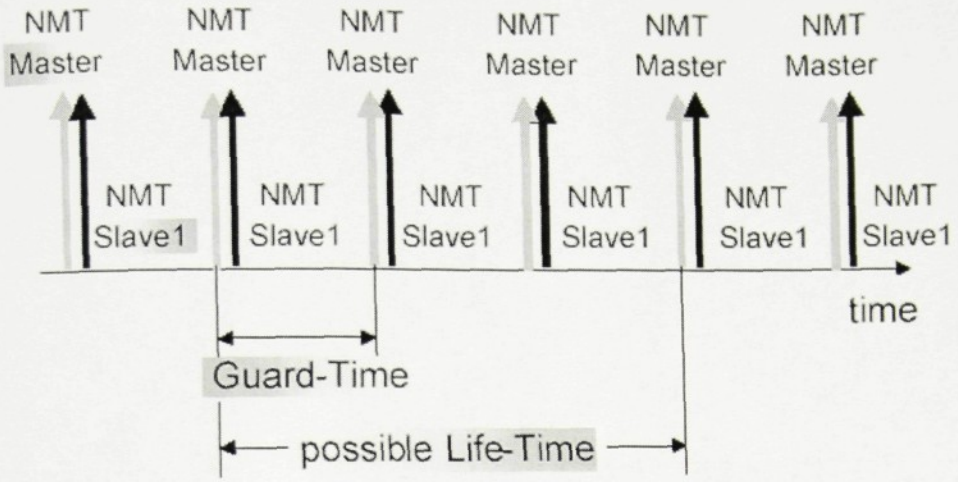


Obr. 19: Stavový diagram uzlu NMT Slave

Jednotlivé stavy zařízení odpovídají protokolu CAL, jsou pouze pozměněny názvy a dány pevné číselné hodnoty pro jednotlivé příkazy. Přijmutí příkazu od modulu NMT Master pro změnu stavu zařízení není potvrzováno moduly NMT Slave.

3. 2. Služba Node Guarding

Tento protokol se využívá k detekování komunikačních chyb na síti. Poté co se zařízení přepne do před-operačním stavu, ohlásí zprávou NMT svou existenci na síti. Identifikátor této zprávy je 700h + Modul-ID. Datová oblast obsahuje jeden byte, jehož hodnota je 0h. Následně NMT Master zahájí vlastní cyklus protokolu. Periodicky vysílá zprávu pro každý modul s identifikátorem 700h + Modul-ID a nulovou datovou oblastí. Modul NMT Slave odpovídá zprávou se stejným identifikátorem, ale datová oblast obsahuje jeden byte. První bit je tzv. *Toggle-Bit*, jehož hodnota se mezi dvěma následujícími cykly protokolu musí měnit. Zbylých sedm bitů obsahuje informaci o aktuálním stavu zařízení (3-*Pre-Operational*, 4-*Prepared*, 5-*Operational*). Průběh protokolu je znázorněn na obr. 20.



Obr. 20: Protokol Node Guarding

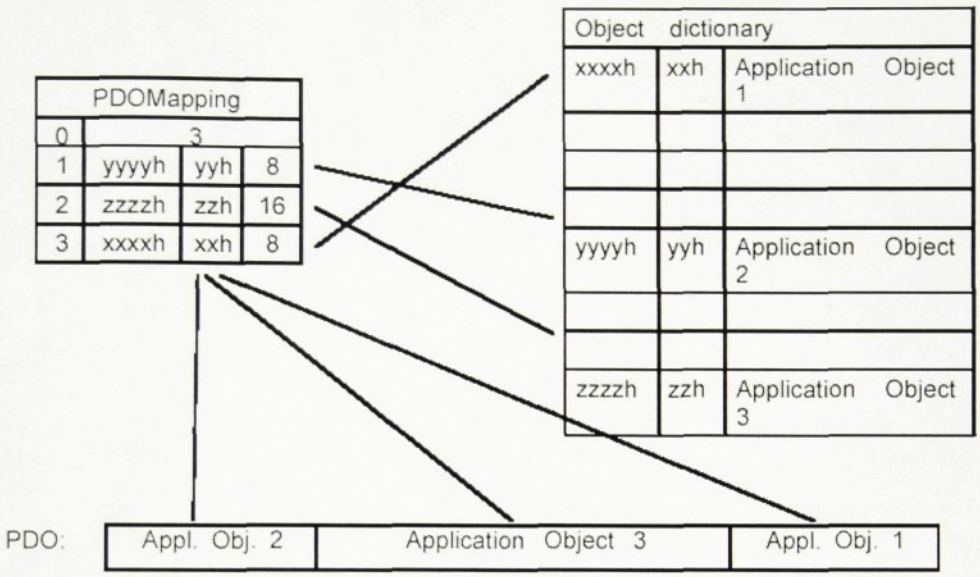
Interval (*Guard Time*) mezi dvěma zprávami od modulu NMT Master může být pro jednotlivé moduly NMT Slave rozdílný, ale z důvodu přehlednosti je pro všechna zařízení jednotný. Pokud modul NMT Master neobdrží odpověď v daném časovém intervalu, signalizuje to chybu aplikace modulu NMT Slave. Naopak pokud modul NMT Slave nedostane zprávu NMT do stanovené doby (*Life Time*), přejde automaticky do předoperačního stavu. Pří konfiguraci protokolu *Node Guarding* musíme nastavit tři položky objektu slovník. A to jsou 100Ch (*Guard Time*), 100Dh (*Life Time Factor*) a 100Eh (*Node Guarding Identifier*). Interval Life Time obdržíme součinem položek *Guard Time* a *Life Time Factor*.

Služby protokolu *Node Guarding* nejsou povinné, ale volitelné. Jejich implementací do modulů SMD má aplikace modulu NMT Master okamžitý přehled o počtu a stavu jednotlivých modulů.

3. 3. Mapování objektů PDO

Objekty PDO slouží k výměně dat v reálném čase. Objekt PDO prakticky přenáší položky objektu slovník. Před vlastním přenosem je nutno nejprve konfigurovat položky objektu slovník pro vysílání (*Transmit PDO Mapping Parameter*) a příjem dat (*Recieve PDO Mapping Parameter*) daného objektu PDO. Každá tato položka je datový typ struktura, který obsahuje seřazené odkazy do objektu slovník. Tyto odkazy jsou vlastně 32 bitová čísla obsahující 16 bitový index, 8 bitový sub-index a 8 bitů s délku typu objektu.

Dle pořadí odkazů budou pak následně objekty posílány po síti. Příklad mapování objektů je na obr. 21. Pokud je třeba změnit obsah objektu PDO, musíme dané zařízení přepnout do stavu *Prepared* a nastavit nové indexy do objektu slovník.



Obr. 21: Princip mapování objektu PDO

U zařízení SMD jsou definovány dva objekty PDO. První slouží k přenosu hodnot napětí (3x1 byte), účiníků (3x1 byte) a frekvence (1 byte). Druhý přenáší hodnoty proudů (4x2 byte). Použité objekty PDO jsou statické, nebo-li jsou mapovány v průběhu inicializace a nemohou být následně změněny.

3. 4. Obsah objektu slovník

Objekt slovník zařízení SMD obsahuje definice standardních datových typů, tedy index 1h až 1F. Dále pak využívá předdefinovaných struktur, které se skládají ze standardních datových typů. Tyto struktury se nacházejí od indexu 20h až 3Fh, obsahují např. komunikační parametr objektů PDO a SDO, mapovací parametr PDO atd. Komunikační parametr objektu PDO obsahuje identifikátor zprávy, která se používá k přenosu objektu PDO a typ přenosu. Typ přenosu objektů PDO u zařízení SMD je synchronní, cyklický a k přenosu dat dochází s každým příchodem synchronizačního

objektu (možnost nastavení periody přenosu 1 až 240 příchodů objektu SYNC). Komunikační parametr objektu SDO obsahuje dvě položky, identifikátory pro zprávy k přenosu objektu SDO. Jeden identifikátor náleží klientovi a druhý majiteli objektu slovník (čtení nebo zápis).

Datové typy specifikované výrobcem a datové typy profilu zařízení nejsou použity. Oblast v objektu slovník od 40h až FFFh tedy neobsahuje žádné informace. Komunikační profil zařízení obsahuje položky společné pro všechny typy zařízení na síti, tudíž i pro moduly SMD. Tyto položky jsou u zařízení SMD využívány:

- 1001h : *Error Register*
- 1005h : *COB-ID SYNC-Message*
- 1006h : *Communication Cycle Period*
- 1007h : *Synchronous Window Length*
- 100Bh : *Node-ID*
- 100Ch : *Guard Time*
- 100Dh : *Life Time Factor*
- 100Eh : *COB-ID Guarding Protokol*
- 1014h : *COB-ID Emergency*

Následně jsou vysvětleny a uvedeny konkrétní hodnoty jednotlivých použitých položek komunikační oblasti. Do objektu *Error Register* je zapisován vždy poslední chybový kód objektu *Emergency*. COB-ID synchronizačního objektu je definován na hodnotu 80h. Perioda komunikačního cyklu je zvolena na tři sekundy s ohledem na nastavenou periodu zařízení, které snímá nové hodnoty z analogových vstupů taktéž s periodou tří sekund. Délka synchronního okénka určuje interval, v kterém zařízení může synchronní objekty PDO vysílat nebo přijímat. Moduly SMD objekty PDO pouze vysílají a délka okénka je stanovena na jednu sekundu. Adresa modulu (*Node-ID*) musí být pro každý modul v síti jedinečná. V případě zařízení SMD se nastavuje pomocí linky RS232 a firemního programu CETIS32. Při zapojení nového modulu do sítě si musíme býti jisti jednoznačností nového identifikátoru.

Položky týkající se protokolu *Node Guarding* jsou tři. *Guard Time* je nastaven na pět sekund, *Life Time Factor* na dvě sekundy. Pokud tedy modul NMT Slave nedostane zprávu od modulu NMT Master do deseti sekund, přejde do stavu *Pre-Operational*. COB-

ID pro *Node Guarding* protokol je definováno 700h + *Node-ID*. Poslední položka je hodnota identifikátoru pro objekt *Emergency*, která je definována 80h + *Node-ID*.

Od indexu 1200h je prostor pro definice parametrů jednotlivých objektů SDO. Celkem je možnost definovat až 128 různých objektů SDO. Na indexu 1400h začíná oblast definicí pro mapování a přenos objektů PDO, kterých je možné definovat až 512. U modulů SMD jsou definovány dva objekty PDO a jeden objekt SDO.

Proměnné pro vizualizaci měřených dat jsou v oblasti profilu specifikovaným výrobcem. Tato oblast má rozsah od indexu 2000h až 5FFFh. Na indexu 2000h se nachází proměnná typu pole se třemi položkami typu *unsigned char*, které reprezentují efektivní hodnoty napětí. Index 2001h je také proměnná typu pole se čtyřmi položkami typu *integer*, které obsahují hodnoty proudů. Pro hodnoty účiníků je definována proměnná typu pole na indexu 2002h se třemi položkami typu *integer*. Proměnná s aktuální hodnotou frekvence je typu *unsigned char* a nachází se na indexu 2003h. Poslední dvě položky jsou převodní konstanty měřených proudů. Jsou typu *integer* na indexu 2004h a 2005h.

Oblast od indexu 6000h až 9FFFh se týká standardního profilu zařízení je nevyužita. Pro umístění proměnných do této oblasti je nutno postupovat dle protokolu pro vstupní a výstupní zařízení (CiA DSP-401), který je zpoplatněn a nebyl k dispozici při implementaci protokolu CANopen.

3. 5. Softwarové řešení implementace CANopen

Přístroje SMD obsahují zdrojový kód pro měření a záznam elektrických veličin. Ten umožňuje komunikaci po sběrnici CAN pomocí firemního protokolu. Úkolem bylo tuto komunikaci změnit a rozšířit dle protokolu CANopen. Dále pak vytvořit aplikaci pro osobní počítač, která by zastupovala úlohu jednotky NMT Master a jako klient pro objekty PDO požadovala měřená data od modulů SMD.

3. 5. 1. Objekt zprávy

Mikroprocesor C515C vytváří celkem patnáct objektů zpráv. Poslední patnáctý slouží pouze k příjmu zpráv, které se nevyskytují často na sběrnici. Objekt zprávy je zastoupen v paměti mikroprocesoru pomocí 16 bytů. První dva byty zastupují řídící registry pro daný objekt. Slouží k poslání zprávy, aktualizaci dat nebo k informaci o přerušení. Další čtyři byty obsahují registry přístupu na sběrnici. Jejich obsah je vlastně identifikátor daného objektu zprávy. Sedmý byte určuje délku přenášených dat, směr

komunikace (příjem nebo vysílání) a typ identifikátoru (standardní nebo rozšířený). Následuje osm bytů, které obsahují vlastní datovou část zprávy. Poslední šestnáctý byte je rezervován pro budoucí účely.

Pro implementaci protokolu CANopen je použito celkem devět objektů zpráv. Čtyři objekty pro příjem zpráv (NMT, protokol *Node Guarding*, SYNC, SDO) a pět objektů pro výsílání zpráv (protokol *Node Guarding*, objekt *Emergency*, dva objekty PDO, SDO). Příklad nastavení objektu zprávy (programovací jazyk C):

```
MCR1_8 = 0xFB;                                                    // CPUUPD=1
UAR0_8 = Slave.OD.Node_Guarding_Identifier >> 3;                  // 11 bit ID
UAR1_8 = (Slave.OD.Node_Guarding_Identifier & 0x7) << 5;          // 11 bit ID
MCFG_8 = 0x18;              // 1 data byte, transmit, standard indentifier
MCR1_8 = 0x55;              // RMTPND=0, TXRQ=0, CPUUPD=0, NEWDAT=0
MCR0_8 = 0xE5;              // MSGAVL=0, TXIE=1, RXIE=0, INTPND=0
```

Jedná se o vysílací objekt pro protokol *Node Guarding*. Tato inicializace objektů zpráv probíhá vždy po restartu zařízení. Následně je nutno objekty ještě aktivovat před zahájením komunikace. Hodnota přerušení od sběrnice CAN (příjem či vyslání zprávy) je druhá nejvyšší. To zajišťuje pružnou odezvu zařízení na přicházející požadavky. Nejvyšší hodnotu přerušení má funkce zajištující měření veličin.

3. 5. 2. Hlídací mechanismus

Součástí mikroprocesoru C515C je bezpečnostní mechanismus (*Watchdog*), který umožňuje zotavení po softwarové nebo hardwarové chybě. Princip spočívá v inkrementování hodnoty 16 bitového registru WDT. Pokud registr dosáhne maximální hodnoty, restartuje se mikroprocesor. K inkrementování registru dochází s periodou 1 μs dle oscilační frekvence mikroprocesoru, která je 10 MHz. Musíme tedy přibližně každých 35 ms vynulovat tento registr, aby zařízení zůstalo v provozu.

Moduly SMD mají ještě jeden externí *Watchdog*, z důvodu ochrany periferií zařízení. Pro vynulování tohoto registru je perioda daleko delší, zhruba 1 sekunda. Příkazy pro vynulování vnitřního a vnějšího mechanismu se pravidelně vyskytují v celém zdrojovém kódu přístrojů SMD, jinak by neustále docházelo jeho restartování.

3. 5. 3. Zdrojový kód přístroje

V prostředí µVision V2.10 se provádějí úpravy zdrojového kódu psaného programovacím jazykem C. Bohužel jediná možnost krokování programu v případě chyby, která není zřejmá, je rozsvícení kontrolních signalizací LED v daném bodě kódu a znovu přehrání celého programu do přístroje přes komunikační linku RS232. Tento postup není efektivní, ale jediný možný v případě chyby, jejíž důvod neznáme.

Tělo programu se skládá ze dvou částí. První část je inicializační, kde dochází k nastavení komunikace, nulování proměnných, čištění registrů atd. Druhá část je tvořena nekonečnou smyčkou, kde probíhá vlastní měření veličin.

V zásadě jsou dva možné postupy při zavádění objektu slovník do zdrojového kódu. První možností je vytvoření celé velké struktury se všemi položkami, což umožňuje rychlý přístup pomocí indexu. Na druhou stranu musíme počítat s nároky na paměť zařízení, neboť téměř každá položka objektu slovník je typu pole s různým počtem prvků. V případě přístrojů SMD, kde by bylo tímto způsobem plýtváno s pamětí, je lépe využít jiné řešení. Druhá možnost je vytvoření proměnných pro použité položky a pro přístup do objektu slovník pomocí indexu použít tuto programovací strukturu:

```
switch (index) { case 0x2004: ..... break;
                 case 0x2005: ..... break;
                 default : ..... break; }
```

V případě žádosti položku, která není zahrnuta v objektu slovník, vyšleme zpětně chybový kód.

3. 5. 4. Provedení stavového diagramu

Stavový diagram je reprezentován proměnnou typu *unsigned char*. Ta nabývá hodnot mezi 0 a 3, což jsou vlastně jednotlivé stavy diagramu. V těle nekonečné smyčky programu je umístěna procedura, která kontroluje příchod zprávy od modulu NMT Master. Pokud zařízení přijme zprávu typu NMT, vyhodnotí adresu, zda je určena pro něj a případně provede změnu hodnoty proměnné. Dle hodnoty této proměnné se poté vysílají nebo nevysílají objekty PDO, je nebo není možný přístup do objektu slovník atd.

Pokud je nutné provést příkaz na restartování komunikace, jsou načteny počáteční hodnoty komunikačních parametrů. V případě restartování celého zařízení je využit princip

bezpečnostního mechanismu. V tomto místě programu je zakázáno přerušení a poté je umístěna nekonečná smyčka, uvnitř které nenulujeme registr pro *Watchdog*.

3. 5. 5. Detekce chybových stavů

Protokol *Node Guarding* detekuje chybové stavy modulů NMT Slave i NMT Master. Po ukončení inicializační fáze přejde zařízení do před-operačního stavu, kde pošle již zmíněnou zprávu o své existenci na síti (před vstupem do nekonečné programové smyčky). Poté s příchodem první zprávy od modulu NMT Master se spustí vlastní časový spínač a odešle se odpověď. Inkrementace proměnné se provádí uvnitř funkce sloužící pro kontrolní blikání prvků LED a nulování registrů pro hlídací mechanismus. Úroveň přerušení je nízká a funkce je volána s periodou 25 ms.

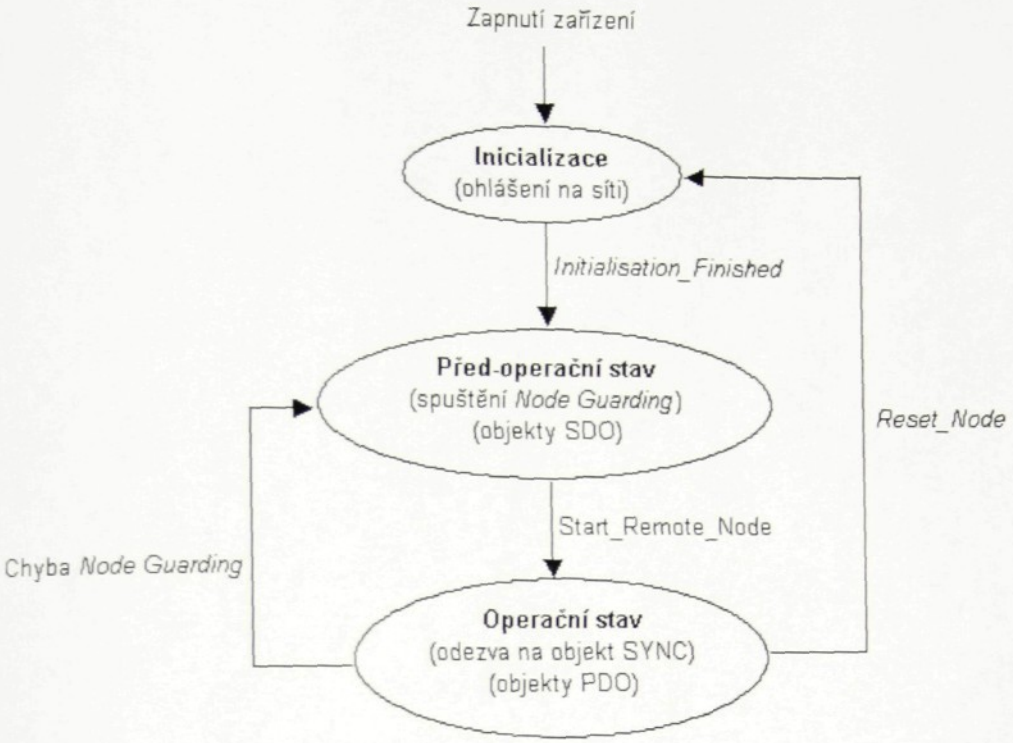
S příchodem nové zprávy od modulu NMT Master se proměnná vynuluje. Pokud však hodnota proměnné přesáhne limit *Life Time* (chyba aplikace modulu NMT Master), přejde zařízení do před-operačního stavu.

3. 5. 6. Popis algoritmu

Po zapnutí zařízení proběhne inicializační část, kdy se načítají definované hodnoty proměnných a nastavují objekty zpráv pro komunikaci. Poté se zařízení ohlásí na síti a čeká na příkazy modulu NMT Master. S první zprávou služby NMT se spouští protokol *Node Guarding*, který je zastaven pouze v případě restartu komunikace nebo celého zařízení. Ve stavu *Pre-Operational* a *Prepared* je možnost načtení konstant pro proudy pomocí objektů SDO, pokud je správný index do objektu slovník . Po změně stavu na *Operational* začne zařízení reagovat na příchozí objekty SYNC a vysílá objekty PDO.

V případě chyby ze strany modulu NMT Master vyšle objekt *Emergency*, s uvedenou chybou komunikace a přejde do stavu *Pre-Operational*. Stavový diagram zařízení SMD je na obr. 22.

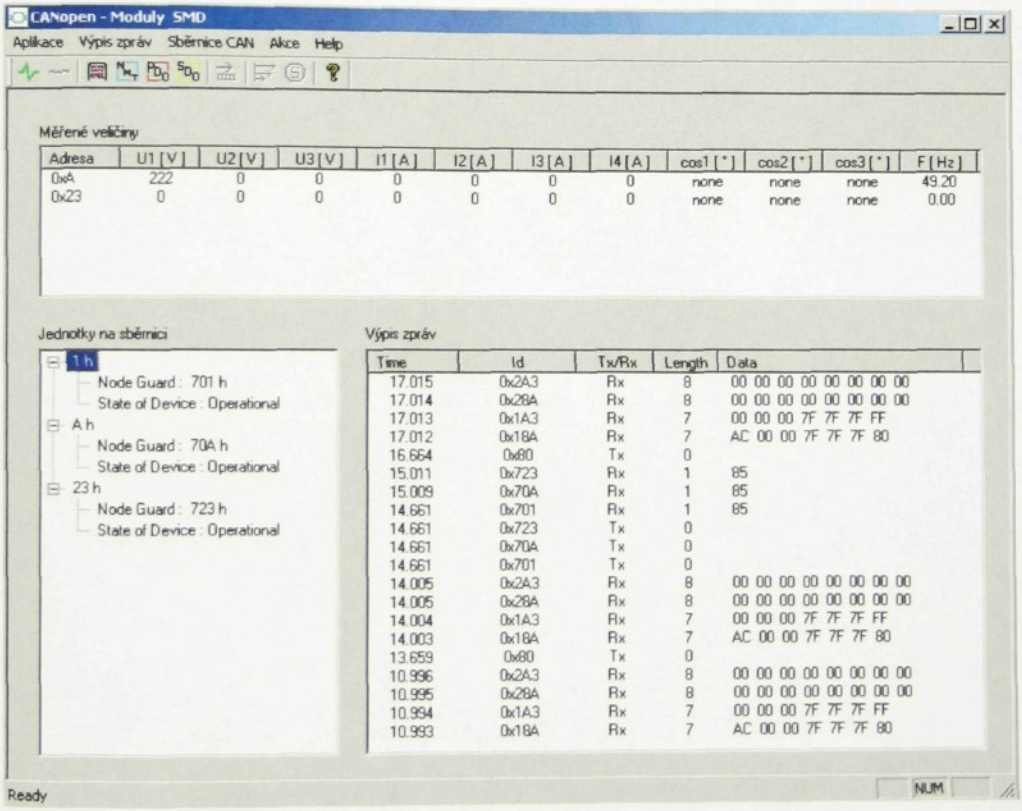
Na přiloženém disku CD je zdrojový kód k přístroji SMD01/34 (pouze provedené úpravy.)



Obr. 22: Stavový diagram modulů SMD

4. Realizace programu pro monitorování zařízení

Aplikace pro monitorování veličin modulů SMD (obr. 23) je také vytvořená v prostředí MS Visual C++. Současně jsou zobrazovány aktuální hodnoty měřených veličin, průběhy zpráv na sběrnici a seznam připojených modulů. Výpis zpráv zobrazuje posledních 20 zpráv poslaných na sběrnici. U každé zprávy se zobrazuje její čas odeslání, identifikátor, směr vysílání (zpráva od PC – Tx nebo od SMD - Rx), délka dat a vlastní data. Vizuální stromová struktura obsahuje seznam jednotek na síti, které jsou reprezentovány svojí jedinečnou adresou *Node-ID*. Dále je u každé jednotky uveden identifikátor pro protokol *Node Guarding* a stav, ve kterém se právě nachází (dle stavového diagramu uzlu NMT Slave).



Obr. 23: Visuální podoba aplikace

První položkou měřených veličin je adresa zařízení, které tyto aktuální hodnoty poslalo. Následují vlastní hodnoty napětí, proudů, účiníků a frekvence. Výpis seznamu jednotek i aktuálních hodnot měření není seřazen dle adresy zařízení, nýbrž v pořadí, v kterém se jednotlivé jednotky přihlásili aplikaci.

4. 1. Popis aplikace

Po spuštění aplikace má uživatel možnost zrušit nastavení pro záznam průběhu zpráv na sběrnici nebo si předešlý průběh zobrazit. Zobrazit lze všechny zprávy najednou nebo jen zprávy týkající se určitých objektů. Odděleně lze zobrazit zprávy pro službu NMT, zprávy objektů PDO (dohromady s objektem SYNC) a zprávy objektů SDO.

Pro další práci s aplikací je nutné provést žádost o přístup na kartu IXXAT, tedy na sběrnici CAN. Ten může být zakázán v případě, kdy s kartou pracuje již jiná aplikace, která ji také používá. V kladném případě se spustí algoritmus pro záznam zpráv a vnitřní hodiny aplikace, které jsou nutné pro určení času odesílaných zpráv. Od této chvíle má uživatel možnost kdykoliv pomocí nemodálního dialogu poslat na sběrnici jakoukoliv zprávu libovolné délky a obsahu (např. možnost testování kritických stavů jednotlivých zařízení).

Pokud máme přístup na sběrnici, můžeme už spustit vlastní protokol CANopen. Po jeho spuštění dojde k identifikaci jednotek a jejich zařazení do seznamu. Následně po výměně dat z objektu slovník a přechodu zařízení do stavu *Operational* dojde k zobrazení měřených dat. Hodnoty se aktualizují s každým příchodem objektů PDO. Probíhá vlastní monitorování hodnot měřených veličin a stavů jednotlivých modulů SMD. V tomto okamžiku můžeme navíc testovat zařízení vysíláním vlastních zpráv pomocí již zmíněného dialogu. Stav zařízení lze měnit za běhu protokolu kliknutím pravého tlačítka myši na položku *State of Device* v seznamu jednotek. Pokud se rozhodneme pro ukončení protokolu CANopen, přestane aplikace vysílat zprávy a zařízení zůstávají v daném stavu. Nyní můžeme protokol znovu spustit nebo přepnout přístup ke kartě IXXAT (ukončit sledování zpráv na sběrnici).

Jestliže jsme ukončili práci se sběrnicí, můžeme si prohlédnout záznam odeslaných zpráv a popřípadě je uložit do zvoleného textového souboru. S každým novým připojením na sběrnici je starý záznam zpráv smazán, proto je umožněna archivace v podobě textového souboru.

4. 2. Průběh algoritmu protokolu

Po spuštění protokolu CANopen dojde nejprve k identifikaci jednotek. Aplikace vyšle zprávu *Reset_Node* platnou pro všechna zařízení na síti, aby všechny moduly byly ve stejném stavu. Po přihlášení jednotek dojde k výměně informací o převodních konstantách pro hodnoty proudů pomocí objektů SDO (čtení z objektu slovník modulů SMD). Poté jsou

všechny jednotky převedeny do stavu *Operational* zprávou *Start_Remote_Node* a začíná vysílání objektu SYNC.

Služba NMT je aktivní po identifikace všech jednotek, které jsou v daném okamžiku již ve stavu *Pre-Operational*. Probíhá cyklické dotazování každé jednotky s periodou *Guard Time*. Algoritmus hlídá přepínání bitu, stav jednotky a dobu odezvy. Pokud jednotka poruší některou uvedenou podmínku, je jí ihned odeslána zpráva *Reset_Communication*. Poté je zařízení znovu přepnuto do operačního stavu pouze na žádost uživatele.

Pří identifikaci je přidávána fiktivní jednotka s adresou 1h. Tato jednotka simuluje vlastní uzel aplikace na sběrnici. Vytváří dle protokolu CAL jednotku typu NMT Slave, která je zároveň i jednotkou typu NMT Master a řídí správu sítě. Tato jednotka nedisponuje objekty SDO ani PDO.

4. 3. Použité datové struktury a metody

Pro sledování a kontrolu jednotek na síti je vytvořena speciální struktura, která vytváří obraz skutečného uzlu. Aplikace pracuje s proměnou *m_pNodes* typu pole, která má právě tuto strukturu jako základní proměnnou a jeho délka je omezena na 127 položek (maximální počet jednotek na síti, dle CANopen). Tato struktura obsahuje následující položky:

- *Node_State* : reprezentující stav jednotky
- *Toggle_Bit* : kontrola služby NMT
- *Node_ID* : identifikace jednotky
- *Guarding_Identifier* : služba NMT
- *Byty* : aktuální data z objektu PDO
- *MTP* : převodní konstanty pro proudy

Tyto položky se průběžně využívají v průběhu algoritmu. Jednotka simulující vlastní uzel aplikace s identifikátorem 1h je také obsažena v tomto poli a to vždy jako první položka. Pro jednotku s *Node_ID* 1h je ještě definována struktura, odpovídají objektu slovník. Jsou zde hlavně položky, které patří do komunikačního profilu. Ty se používají pro cyklus služby NMT a přenos objektu SYNC a objektu PDO a SDO.

Stejně jako u předešlé vytvořené aplikace je zde použita oddělená fronta pouze pro časové spínače, která není zatížena uživatelskými zprávami (stisk tlačítka myši atd.). Pro sledování přicházejících zpráv je definována funkce, která s periodou 100 ms čte zprávy z příchozí fronty. Podle hodnoty identifikátoru zprávy ji předá následující funkci, která provádí její vlastní zpracování. Jsou vytvořeny funkce pro zpracování zpráv pro služby NMT, SDO a PDO. Uvnitř těchto funkcí dochází ke kontrole formátu datové části zprávy a následně jsou změněny hodnoty proměnné *m_pNodes* u dané jednotky. Perioda 100 ms je dostatečně malá při nastavené rychlosti, aby se docílilo dojmu spojitého příjmu zpráv. Jako příklad je zde uvedena funkce pro čtení objektů PDO:

```
void CMainFrame::Cti_PDO(VCI_CAN_OBJ Aktual){
UINT8 i,j;
UINT32 pom;

      if(Aktual.id>0x180 && Aktual.id<0x200){                    // PDO 1
            pom=Aktual.id-0x180;
            i=0;
            while(m_pNodes[i].Node_ID!=pom) i++;
            for(j=0; j<7; j++) m_pNodes[i].Byty[j]=Aktual.a_data[j];
      }
      if(Aktual.id>0x280 && Aktual.id<0x300){                    // PDO 2
            pom=Aktual.id-0x280;
            i=0;
            while(m_pNodes[i].Node_ID!=pom) i++;
            for(j=0; j<8; j++) m_pNodes[i].Byty[j+7]=Aktual.a_data[j];
      }
}
```

Pro sledování stavu jednotek je definována funkce, která s periodou 500 ms kontroluje položky v proměnné *m_pNodes* a přepisuje hodnoty stavu v seznamu jednotek. Pokud se uživatel rozhodne pro změnu stavu zařízení a učiní tak velmi krátkou dobu před příchodem nové zprávy od modulu NMT Master, daná jednotka nemusí stihnout zpracování požadavku a odpoví zprávou, která obsahuje hodnotu předešlého stavu. Ovšem aplikace již má změněnou hodnotu stavu. Poté je vyhodnocena chyba zařízení a to musí přejít do stavu *Pre-Operational*.

Archivace průběhu zpráv na sběrnici probíhá ihned po přečtení zprávy nebo po jejím odvysílání. Celkem jsou vytvořeny čtyři binární soubory dat. První soubor obsahuje

všechny zprávy, obsah ostatních je dle jednotlivých služeb NMT, SDO a PDO. Binární soubory se s každým novým přístupem na sběrnici přemazávají, proto je výhodné využít možnosti a aktuální data uložit do textového souboru.

4. 4. Návrh úpravy aplikace

Vytvořená aplikace byla testována spolu se dvěma moduly SMD. Tomu odpovídá nastavení hodnot komunikačních položek v objektu slovník. Pokud bychom pracovali řádově s více jednotkami (např. 100), bylo nutné tyto hodnoty upravit. Zvětšit hodnotu intervalu pro synchronní okénko, aby všechna zařízení stihla odpovědět, nebo změnit periodu pro vysílání objektů PDO, aby odpovídaly v daném okamžiku jen některé moduly. Podobné změny by byly také nutné u služby NMT. V tomto případě by bylo výhodné nastavit komunikační položky v aplikaci (objekt slovník jednotky s adresou 1h) a poté jej pomocí objektů SDO zapsat do jednotlivých modulů, ještě před přepnutím do operačního stavu.

IV. ZÁVĚR

Cílem diplomové práce bylo vytvoření aplikace pro sběr a vizualizaci dat z měřících modulů SMD s využitím protokolu CANopen. Implementací protokolu v zařízeních SMD byla vytvořena nová aplikační vrstva, která nahradila původní protokol firmy KMB s.r.o.. Vzniklá aplikační vrstva umožňuje síťovou správu, výměnu dat a ošetření chybových stavů. Vytvořená aplikace má řídící úlohu v síťovém prostředí.

Realizace aplikace a implementace protokolu CANopen byly provedeny dle specifikace *Minimum Capability Device*, která umožňuje komunikaci s plně funkčními zařízeními. Aplikace zobrazuje hodnoty měřených veličin, průběhy zpráv na sběrnici a stavy jednotlivých jednotek. Moduly SMD nebyly testovány v jiné síti používající protokol CANopen. Z hlediska správy sítě a synchronizačních metod jsou moduly dostatečně vybaveny. Jistým omezením jsou statické objekty PDO, které přenášejí pouze námi definované položky pro vizualizaci a nemohou býti v jiné síti modifikovány.

Standardní profil zařízení a komunikace definovaný protokolem CANopen vytváří prostor pro systémy, které se skládají z jednotek různých výrobců. Vynaložené prostředky na konfiguraci sítě a její uvedení do provozního stavu se díky standardním protokolům minimalizují.

Seznam použité literatury:

[1] Kruglinski D. J., Mistrovství ve VISUAL C++, Computer Press, Brno, 1999

[2] Specifikace CAN 2.0 A , CAN 2.0 B

[3] CAN Application Layer for Industrial Applications DS-201…207, CiA, 1996

[4] CANopen Comunication Profile for Industrial Systems DS-301, CiA, 1996

[5] Interní dokumentace a zdrojové kódy firmy KMB s.r.o., Liberec

[6] C515C 8-Bit CMOS Microcontroller, Uživatelský manuál Siemens AG, München, 1997

[7] VCI-V2 Programmers Manual, IXXAT Automation GmbH, Weingarten

Internetové stránky:

[8] CAN in Automation http://www.can-cia.de

[9] IXXAT http://www.ixxat.de

[10] Amit s.r.o. http://www.amit.cz

[11] Developer of CANopen networks http://www.canopen.us

[12] Embedded systems academy http://www.canopenia.com

[13] Unicontrols a.s. http://www.unicontrol.cz

Příloha č.1: Popis struktury datové zprávy

Začátek zprávy (*SOF = Start Of Frame*) - začátek zprávy, 1 bit *dominant*.

Řízení přístupu na sběrnici (*Arbitration Field*) - určení priority zprávy

- Identifikátor zprávy - 11 bitů, udává význam přenášené zprávy

- RTR bit (*Remote Request*) - 1 bit, příznak udává, zda se jedná o datovou zprávu nebo o žádost o vyslání dat. V datové zprávě musí být tento bit *dominant*, v žádosti o data *recessive*.

Řídící informace (*Control Field*)

- R0, R1 - rezervované bity

- Délka dat - 4 bity, počet přenášených datových bajtů ve zprávě. Povolené hodnoty jsou 0 až 8.

Datová oblast (*Data Field*) - datové bajty zprávy. Maximálně je vysláno 8 bajtů.

CRC (*CRC Field*) - 16 bitů, zabezpečovací CRC kód

- CRC kód - 15 bitů

- ERC (*CRC oddělovač*) - 1 bit *recessive*

Potvrzení (*ACK Field*) - 2 bity

- ACK (bit potvrzení) - 1 bit

- ACD (oddělovač potvrzení) - 1 bit *recessive*

Konec zprávy (*End Of Frame*) - 7 bitů *recessive*

Mezera mezi zprávami (*Interframe Space*) - 3 bity *recessive*, odděluje dvě zprávy

Příloha č.2: Chyby při přenosu objektu SDO

Třída a kód chyby:

Error Class Value	Error Code Value Meaning	Example
5 Service error	3 Parameter Inconsistent	Toggle bit not alternated.
	4 Illegal Parameter	Time out value reached.
6 Access error	1 Object Access Unsupported	Attempt to write a read-only or to read a write-only parameter.
	2 Object non-existent	The object does not exist in the dictionary.
	6 Hardware fault	Access failed because of an hardware error.
	7 Type Conflict	Data type does not match.
	9 Object attribute inconsistent	The sub-index does not exist.
8 Other error	0	Transfer was aborted by user.

Doplňkový kód chyby:

Additional Code	Meaning
0	No precise details for the reason for the error
10H	Service parameter with an invalid value
11H	Sub-Index does not exist
12H	Length of service parameter too high
13H	Length of service parameter too low
20H	Service cannot currently be executed
21H	because of local control
22H	because of the present device state
30H	Value range of parameter exceeded
31H	Value of parameter written too high
32H	Value of parameter written too low
36H	Maximum value is less than minimum value
40H	Incompatibility with other values
41H	data cannot be mapped to the PDO
42H	PDO length exceeded
43H	General parameter incompatibility reason
47H	General internal incompatibility in the device

Příloha č.3: Chybový kód objektu *Emergency*

Error Code (hex)	Meaning
00xx	Error Reset or No Error
10xx	Generic Error
20xx	Current
21xx	Current, device input side
22xx	Current inside the device
23xx	Current, device output side
30xx	Voltage
31xx	Mains Voltage
32xx	Voltage inside the device
33xx	Output Voltage
40xx	Temperature
41xx	Ambient Temperature
42xx	Device Temperature
50xx	Device Hardware
60xx	Device Software
61xx	Internal Software
62xx	User Software
63xx	Data Set
70xx	Additional Modules
80xx	Monitoring
81xx	Communication
90xx	External Error
F0xx	Additional Functions
FFxx	Device specific

Příloha č.4: Obsah přiloženého disku CD

Zdrojové kódy:

- Úpravy kódu přístrojů SMD
- Kódy aplikace pracující s protokolem firmy KMB s.r.o.
- Kódy aplikace pracující s protokolem CANopen

Programy:

- Aplikace pro vizualizaci dle protokolu firmy KMB s.r.o.
- Aplikace pro vizualizaci dle protokolu CANopen