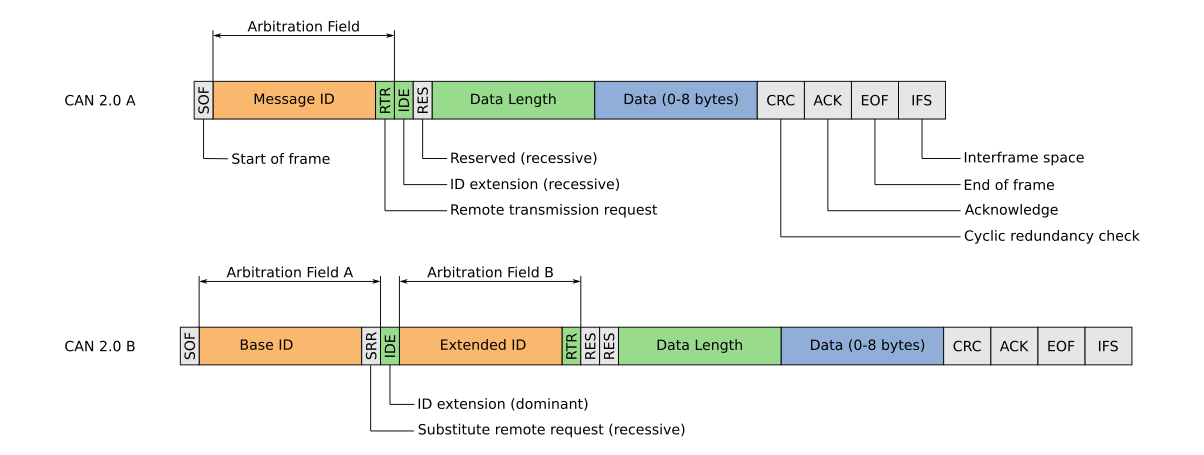
**Organizace dat přijímaných/vysílaných z/do CANu**

Úvodem považujeme za vhodné uživateli knihovny CANUSB32.DLL přiblížit, jak jsou v rámci CANu a standardu aplikační vrstvy CANOpen organizována data, ke kterým přes naše API prostřednictvím virtuálních proměnných či kanálů přistupuje.

Na úrovni síťové vrstvy můžeme CAN považovat za síť s multicastovou komunikací. Data jsou organizována do zpráv, které z uživatelského hlediska dle standardu CAN 2.0 A obsahují dvě základní části: 11bitové ID zprávy a 8bytové datové pole. Všechna zařízení mohou (ale nemusí) přijímat všechny zprávy. Čím nižší ID, tím má zpráva vyšší prioritu. Komunikace je bezeztrátová.

To, jak se s 11bitovým ID a 8 bajty dat nakládá, upravují protokoly aplikační vrstvy. My máme v našich zařízeních implementovány základní funkce protokolu CANOpen.



*Obr. 1: Struktura rámce dle standardu CAN 2.0 A*

Každé zařízení v síti (používá se i termín sběrnice) CAN má dle standardu aplikační vrstvy CANOpen svoje jedinečné ID v intervalu 1 – 127. Každé takové zařízení může po CANu komunikovat dvěma základními způsoby, a to pomocí objektů PDO (process data object) nebo SDO (service data object). PDO se používají zpravidla pro periodické odesílání dat z/do zařízení. SDO se používají pro nastavování/vyčítání různých provozních parametrů (jedná se o peer-to-peer komunikaci s potvrzováním). Každé zařízení může přijímat i vysílat až 4 různé PDO a jeden SDO.

Rozlišení toho, zda se jedná o SDO, PDO či jinou formu komunikace popsanou standardem CANOpen se provádí pomocí ID zprávy. ID zprávy se tvoří následovně:

***ID zprávy = CANOpenID zařízení (1..127) + kód funkce (viz tab. 1)***

*Tab. 1: Kódy vybraných funkcí ve standardu CANOpen*

|  |  |  |
| --- | --- | --- |
|  | Směr do zařízení | Směr ze zařízení |
| PDO1 | 512 | 384 |
| PDO2 | 768 | 640 |
| PDO3 | 1024 | 896 |
| PDO4 | 1280 | 1152 |
| SDO | 1536 | 1408 |

V případě SDO se první 4 B datového pole používají pro indexaci uvnitř tzv. adresáře objektů a rozlišení typu komunikace, další 4 B pak obsahují hodnotu zapisovaného/čteného parametru. Detailní popis SDO není cílem tohoto textu.

V případě PDO je všech 8 B dat ve zprávě ponecháno v gesci programátora. Knihovna CANUSB32.DLL v rámci těchto 8 B zavádí model očíslovaných proměnných, které sw aplikace může číst nebo do nich zapisovat. Tomuto číslu (indexu) říkáme číslo kanálu/sub kanálu. Použijeme-li při volání některé z dále popisovaných funkcí jako tento index ID zprávy dle standardu CANOpen, hovoříme pak o **čísle kanálu**, pracuje tato funkce s celým 8B datovým polem zprávy. Můžeme ho číst z příchozí fronty, zapsat do odchozího bufferu nebo odeslat.

***číslo kanálu = ID zprávy***

Použijeme-li index popsaný níže, přistupují funkce knihovny přímo k virtuálním proměnným a programátor se nemusí starat o pořadí bytů ve zprávě (little/big endian), maskování a celkovou strukturu 8B dat, která je popsána konfiguračním souborem dodaným výrobcem zařízení. Tímto způsobem lze proměnné číst z příchozí fronty a zapisovat do odchozího bufferu. Postup tvorby **čísla sub kanálu:**

***číslo sub kanálu = CANOpenID\*100 + 10000 + pořadové číslo proměnné v rámci zařízení***

Pořadové číslo se generuje podle pořadí v jakém jsou proměnné uvedeny v konfiguračním \*.COB souboru. Pro přijímané i vysílané proměnné je jen jedno společné číslování.

***Příklad:***

Mějme detekční kartu interferometru „DETINF“ s CANOpenID = 42. Chceme vyčíst stav počítadla a hodnoty signálů X a Y. Víme, že karta tyto proměnné posílá uvnitř svého PDO1 a jsou to první tři proměnné popsané v jejím konfiguračním souboru \*.COB. Hledáme číslo kanálu pro příjem celého PDO1 a čísla sub kanálů pro příjem jednotlivých proměnných.

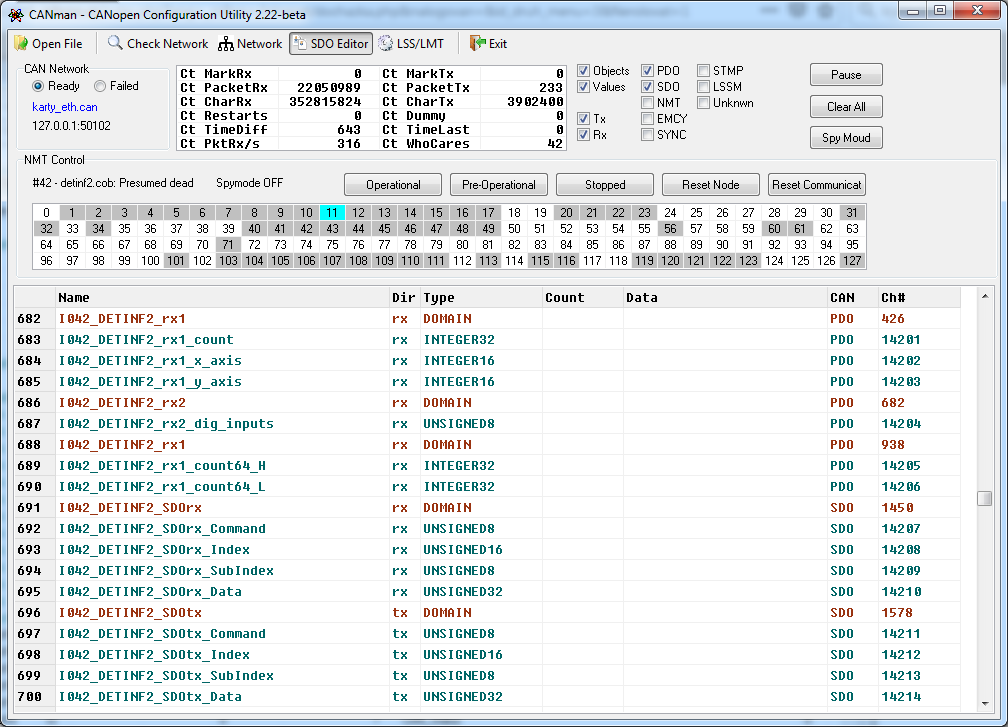
*číslo kanálu pro PDO1 = ID zprávy pro PDO1 = 42 + 384 = 426*

*číslo sub kanálu pro 1. proměnnou = 42\*100 + 10000 + 1 = 14201*

*číslo sub kanálu pro 2. proměnnou = 42\*100 + 10000 + 2 = 14202*

*číslo sub kanálu pro 3. proměnnou = 42\*100 + 10000 + 3 = 14203*

**Ukázka číslovaní kanálů a sub kanálů pro DETINF2 kartu s CANOpenID = 42**



*Toto je screenshot s aplikace CANMAN, kterou najdete v několika exemplářích v dodaném balíku obslužného SW. Je to dobrý pomocník mimo jiné i při zjišťování čísel kanálů (červeně) a subkanálů (modře).*

**Popis základních funkcí CANUSB32.DLL**

**bool CANInit(char \* IniName)**

Provede inicializaci knihovny na základě souboru popisujícího CAN sběrnici. Jméno souboru se předává parametrem IniName, jedná se o standardní nulou zakončený řetězec.

Konfigurační \*.CAN soubor obsahuje:

* Nastavení typu rozhraní, kterým se aplikace připojí k CAN sběrnici. Typicky se volí mezi přímým připojením přes USB/CAN interface, TCP/IP připojení přes server NETCANS a simulátorem sítě. Parametr *COMTYPE.*
* Pokud se připojujeme přes TCP/IP, tak IP adresu serveru. Parametr *server.*
* CANOpen ID zařízení (číslo 1-127), ze kterých chceme přijímat data nebo jim něco posílat. Sekce *[CanDevice001]*, *[CanDevice002]* atd. Parametr *CanOpenID*
* Formát proměnných přijímaných/odesílaných z/do zařízení. Parametr Device v dané sekci odkazuje na \*.COB soubor obsahující popis proměnných. Tento soubor dodává výrobce zařízení.
* Pozor! Pokud zařízení není uvedeno v \*.CAN souboru, nemůže s ním aplikace komunikovat.

Vrací:

true – inicializace se zdařila

false – chyba při inicializaci

Příklad:

if (!CANInit("moje\_sit.can")) {

printf("Chyba pri inicializaci CANu!\n");

return -1;

}

**bool CANClose(void)**

Zavře knihovnu.

*Vrací:*

true – deinicializace se zdařila.

false – chyba při deinicializaci, např. že třeba nebylo co deinicializovat.

*Příklad:*

if (!CANClose()) {

printf("Chyba pri deinicializaci CANu!\n");

return -1;

}

**FUNKCE PRO PŘÍJEM DAT ZE SÍTĚ CAN**

**long CANReadChanNum(void)**

Vrací číslo přijatého kanálu, který je aktuálně k dispozici ke čtení z přijímacího bufferu. Vrací 0, pokud je přijímací buffer prázdný. Typicky se tato funkce volá v nekonečné smyčce v separátním threadu. Pokud vrátí nenulovou hodnotu, provedeme pomocí funkce CANReadChan načtení 8 bajtů přijatých dat.

**long CANReadSubChanNum(void)**

Vrací číslo přijatého sub kanálu (proměnné), která je aktuálně k dispozici ke čtení z přijímacího bufferu. Vrací 0, pokud je buffer prázdný. Typicky se tato funkce volá v nekonečné smyčce v separátním threadu. Pokud vrátí nenulovou hodnotu, provedeme pomocí funkce CANReadChan načtení proměnné.

**bool CANReadChan( long ChanNum, void \*pData)**

Do bufferu pData načte z přijímacího bufferu buď celý CAN objekt (8 bajtů), nebo proměnnou. Podle toho, zda je v parametru ChanNum číslo kanálu nebo číslo sub kanálu.

*Příklad:*

*// Ukazka nacitani promennych z detekcni karty interferometru “DETINF2” // s CANOpenID = 42*

while(RxThreadControl){

long SubChanNum = CANReadSubchanNum();

switch(SubChanNum){

case 14001: //POCITADLO

CANReadChan(14001,&CNT);

break;

case 14002: //OSA X

CANReadChan(14002,&X);

//cteni chybovych priznaku

ERR\_VELOCITY = ((X&1)!=0);

ERR\_MAGNITUDE = ((X&2)!=0);

//ocistime data ADC od chybovych priznaku

X = X&0xFFFC;

break;

case 14003: //OSA Y

CANReadChan(14001,&Y);

break;

default:

break;

}

}

**FUNKCE PRO ODESÍLÁNÍ DAT NA SÍŤ CAN**

**bool CANWriteChan( long ChanNum, void \*pData)**

Zapíše data odkazovaná pointerem pData do odesílacího bufferu pro kanál nebo subkanál udávaný parametrem ChanNum. V případě použití čísla kanálu si funkce z dané adresy přečte celých 8 bajtů. V případě použití čísla subkanálu si funkce načte tolik dat, kolik je dle definice proměnné v souboru \*.COB potřeba, tj. např. pro INTEGER32 to budou 4 bajty, pro UNSIGNED16 2 bajty atd.

**bool CANWriteChanNum(long ChanNum)**

Na síť CAN fyzicky odešle data zapsaná v předchozím kroku do odesílacího bufferu funkcí CANWriteChan. Použijeme-li jako parametr číslo kanálu, odešlou se naráz všechny jeho subkanály. Použijeme-li číslo subkanálu, odešlou se kromě něj i všechny ostatní subkanály patřící pod stejný kanál. To je dáno tím, že 8B zpráva na CAN musí vždy odejít celá.

**Přidání CANUSB32.DLL do projektu C/C++**

Knihovna byla přeložena v prostředí MS Visual Studio 6.0. Pomocí utilit vámi používaného IDE je možné vytvořit \*.LIB soubor a ten společně s hlavičkovým souborem CANUSB32.H přidat do projektu. To máme vyzkoušeno jak na starších verzích MSVC tak Borland C++ Builderu. V novějších verzích začíná být trochu zmatek v tom, jaký který linker požaduje formát \*.LIB souboru. Jako nejjistější a nejfunkčnější cesta se v aktuálních prostředích jeví zavedení knihovny pomocí funkce LoadLibrary z Windows API, tak je to provedeno v ukázkovém souboru, který přikládám.

**FORMÁT DAT ODESÍLANÝCH DETEKČNÍ KARTOU KARTOU DETINF2**

**Kanál PDO1**

Číslo kanálu = CANOpenID + 384

***Subkanál COUNT***

Číslo subkanálu = CANOpenID \*100 + 10000 + 1

Datový typ: int32

Význam: počítadlo interferenční fáze, 1024 LSB odpovídá jednomu interferenčnímu proužku

***Subkanál X\_AXIS***

Číslo subkanálu = CANOpenID \*100 + 10000 + 2

Datový typ: int16

Význam: signál osy X z detekční jednotky interferometru + chybové stavy

|  |  |  |  |
| --- | --- | --- | --- |
| **Bit** | 15-2 | 1 | 0 |
| **Význam** | Hodnota z A/D převodníku | Podkročení minimální amplitudy | Překročení max. rychlosti |

***Subkanál Y\_AXIS***

Číslo subkanálu = CANOpenID \*100 + 10000 + 3

Datový typ: int16

Význam: signál osy Y z detekční jednotky interferometru

|  |  |  |  |
| --- | --- | --- | --- |
| **Bit** | 15-2 | 1 | 0 |
| **Význam** | Hodnota z A/D převodníku | - | - |

**KOMUNIKACE S KARTOU DETINF2 POMOCÍ PROTOKOLU SDO**

Karta pro nastavování různých provozních parametrů používá zjednodušenou verzi protokolu CANOpen SDO.

Karta má v paměti RAM a FLASH strukturu nazývanou adresář objektů. Každý objekt je jednoznačně identifikován 16bitovým indexem a 8bitovým sub indexem. Objekty mohou být typu char, u/int16, u/int32, float32.

**Jedná se o komunikaci, která má vždy 2 kroky – příkaz a odpověď.**

**Postup zápisu:**

1. Aplikace odešle příkaz k zápisu hodnoty, index, sub index a hodnotu.
2. Zařízení odpoví potvrzením zápisu, indexem, sub indexem a zapsanou hodnotou. V případě chyby odpoví chybovým kódem.

**Postup čtení:**

1. Aplikace odešle příkaz ke čtení hodnoty, index, sub index. Pole s hodnotou je ve zprávě ignorováno.
2. Zařízení odpoví potvrzením čtení, indexem, sub indexem a čtenou hodnotou. V případě chyby odpoví chybovým kódem.

**První krok komunikace, tedy příkaz, se provádí prostřednictvím kanálu s názvem SDOtx a jeho subkanálů.**

**Kanál SDOtx**

Číslo kanálu = CANOpenID + 1536

***Subkanál SDOtx\_Command***

Číslo subkanálu = CANOpenID\*100 + 10000 + 11

Datový typ: uint8

Význam: Identifikuje, zda se bude jednat o zápis do adresáře objektů, nebo žádost o vyčtení dat.

|  |  |
| --- | --- |
| **Příkaz** | **Hodnota SDOtx\_Command** |
| Zápis do adresáře objektů | 43 |
| Žádost o vyčtení dat | 64 |

***Subkanál SDOtx\_Index***

Číslo subkanálu = CANOpenID\*100 + 10000 + 12

Datový typ: uint16

Význam: Index zapisované nebo čtené hodnoty v adresáři objektů.

***Subkanál SDOtx\_SubIndex***

Číslo subkanálu = CANOpenID\*100 + 10000 + 13

Datový typ: uint8

Význam: Sub index zapisované nebo čtené hodnoty v adresáři objektů.

***Subkanál SDOtx\_Data***

Číslo subkanálu = CANOpenID\*100 + 10000 + 14

Datový typ: uint32

Význam: 4 bajty obsahující zapisovanou hodnotu. Při žádosti o vyčtení dat je hodnota ignorována.

**Druhým krokem komunikace je odpověď od zařízení na kanálu SDOrx.**

**Kanál SDOrx**

Číslo kanálu = CANOpenID + 1408

***Subkanál SDOrx\_Command***

Číslo subkanálu = CANOpenID\*100 + 10000 + 7

Datový typ: uint8

Význam: Sděluje, zda se jedná o potvrzení zápisu, vyčtená data, nebo chybový stav.

|  |  |
| --- | --- |
| **Příkaz** | **Hodnota SDOtx\_Command** |
| Potvrzení zápisu | 96 |
| Vyčtená data | 66 |
| Chybový stav | 128 |

***Subkanál SDOrx\_Index***

Číslo subkanálu = CANOpenID\*100 + 10000 + 8

Datový typ: uint16

Význam: Index zapisované nebo čtené hodnoty v adresáři objektů.

***Subkanál SDOrx\_SubIndex***

Číslo subkanálu = CANOpenID\*100 + 10000 + 9

Datový typ: uint8

Význam: Sub index zapisované nebo čtené hodnoty v adresáři objektů.

***Subkanál SDOtx\_Data***

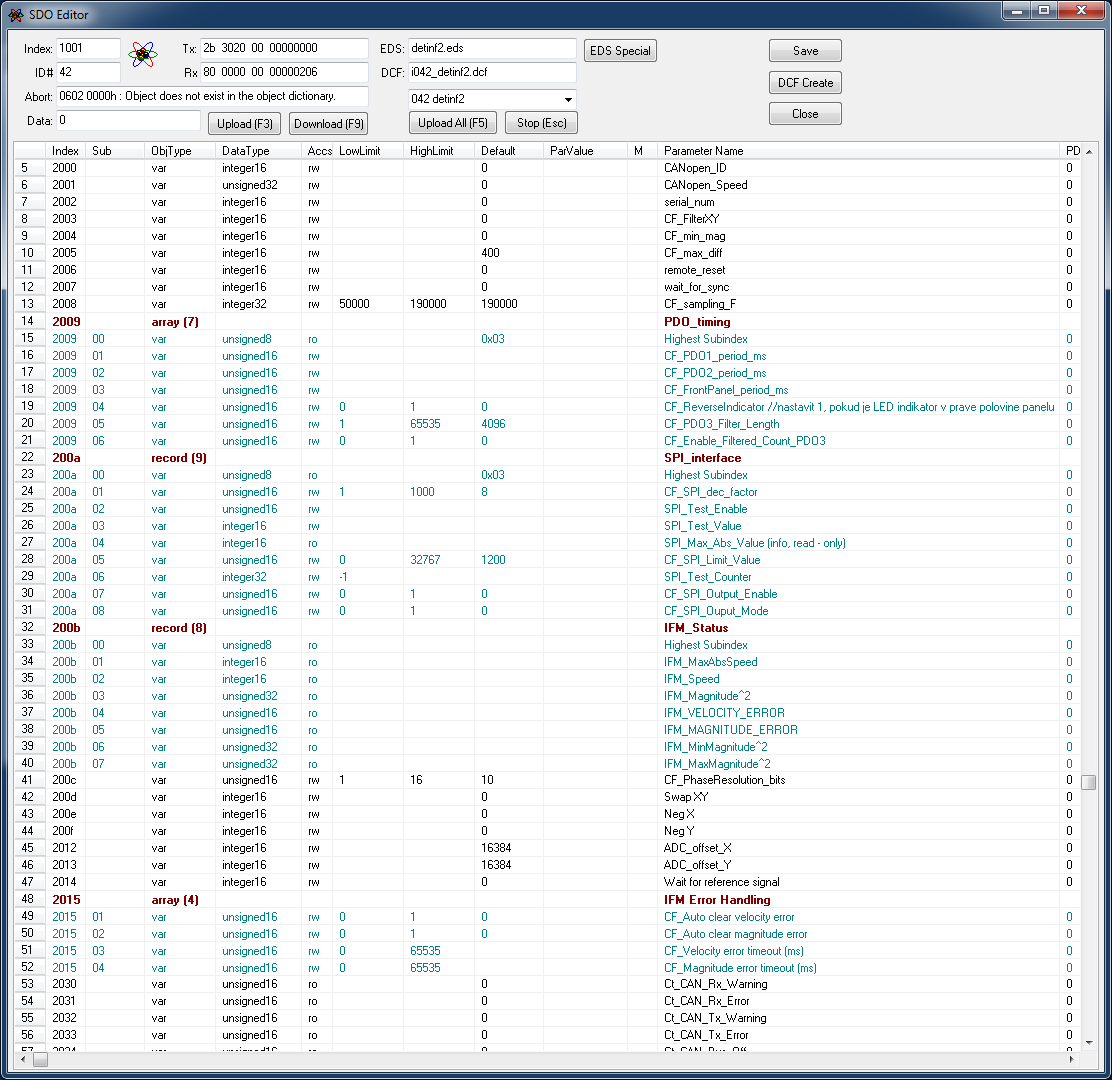
Číslo subkanálu = CANOpenID\*100 + 10000 + 10

Datový typ: uint32

Význam: 4 bajty obsahující zapisovanou nebo čtenou hodnotu. V případě chybového stavu obsahuje kód chyby.

|  |  |
| --- | --- |
| **Typ chyby** | **Kód chyby** |
| Neexistující objekt (nesprávný index nebo subindex) | 0x05040001 |
| Zapisovaná hodnota je vyšší než maximální možná | 0x06090031 |
| Zapisovaná hodnota je nižší než minimální možná | 0x06090032 |

**Přehled parametrů v adresáři objektů karty DETINF2:**



Pozn. indexy a subindexy jsou v hexadecimálním formátu.

**Nejpoužívanější SDO karty DETINF2**

*Vysvětlivky: rw – lze číst i zapisovat; wo – pouze k zápisu; ro – pouze ke čtení; flash – proměnná uložena ve flash, nedoporučuje se ji často přepisovat*

**CF\_min\_mag (rw, flash)**

Index: 0x2004

Sub Index: 0x00

Význam: Nastavení prahu pro chybu příliš nízké intenzity. Podkročení nastaví chybový příznak ERR\_MAGNITUDE na 1. Smaže se jedině resetem. U novější verze karty funguje automatické mazání chyb po určitém čase (tu zatím nemáte).

**CF\_max\_diff (rw, flash)**

Index: 0x2005

Sub Index: 0x00

Význam: Nastavení prahu pro chybu příliš rychlého pojezdu. Rychlost je vyjádřena jako změna hodnoty počítadla proužků mezi 2 po sobě jdoucími vzorky A/D převodníku. Doporučujeme neměnit přednastavenou hodnotu. Překročení nastaví chybový příznak ERR\_SPEED na 1. Smaže se jedině resetem. U novější verze karty funguje automatické mazání chyb po určitém čase (tu zatím nemáte).

**Remote\_reset (wo)**

Index: 0x2006

Sub Index: 0x00

Význam: Zápis hodnoty 1 provede reset počítadla a chybových příznaků.

**CF\_PDO1\_period\_ms (rw, flash)**

Index: 0x2009

Sub Index: 0x01

Význam: Perioda odesílání vzorků signálu X, Y a počítadla na CAN (PDO1). Doporučujeme nenastavovat kratší než 4 ms.

**IFM\_MaxAbsSpeed (ro)**

Index: 0x200B

Sub Index: 0x01

Význam: Lze vyčíst maximální pozorovanou změnu údaje počítadla mezi 2 po sobě jdoucími vzorky A/D převodníku. Resetem se statistika nuluje.

**IFM\_Speed (ro)**

Index: 0x200B

Sub Index: 0x02

Význam: Lze vyčíst aktuální změnu údaje počítadla mezi 2 po sobě jdoucími vzorky A/D převodníku.

**IFM\_Magnitude^2 (ro)**

Index: 0x200B

Sub Index: 0x03

Význam: Lze vyčíst druhou mocninu aktuální amplitudy interferenčního proužku (*I2 = X2 + Y2*).

**IFM\_VELOCITY\_ERROR (ro)**

Index: 0x200B

Sub Index: 0x04

Význam: Lze vyčíst stav chybového příznaku VELOCITY\_ERROR.

**IFM\_MAGNITUDE\_ERROR (ro)**

Index: 0x200B

Sub Index: 0x05

Význam: Lze vyčíst stav chybového příznaku MAGNITUDE\_ERROR.

**IFM\_MinMagnitude^2 (ro)**

Index: 0x200B

Sub Index: 0x06

Význam: Druhá mocnina minimální pozorované amplitudy interferenčního proužku (od posledního resetu).

**IFM\_MaxMagnitude^2 (ro)**

Index: 0x200B

Sub Index: 0x07

Význam: Druhá mocnina maximální pozorované amplitudy interferenčního proužku (od posledního resetu).

**ADC\_offset\_X (rw, flash)**

Index: 0x2012

Sub Index: 0x00

Význam: Offset A/D převodníku pro osu X. Lze doladit pozici kružnice/elipsy opisované signály X a Y tak, aby měla střed o souřadnicích (0, 0). Základní hodnota pro dokonalý signál bez offsetu je 16 384.

**ADC\_offset\_Y (rw, flash)**

Index: 0x2013

Sub Index: 0x00

Význam: Offset A/D převodníku pro osu Y. Lze doladit pozici kružnice/elipsy opisované signály X a Y tak, aby měla střed o souřadnicích (0, 0). Základní hodnota pro dokonalý signál bez offsetu je 16 384.

*Pozn.: U obou offsetů se jedná o číslo, které se od hodnoty vzorku z A/D převodníku automaticky hardwarově odčítá. Tzn. menší hodnota parametru způsobí posun signálu do více kladných hodnot. Doporučujeme nejprve vyladit interferometr na max. kontrast a intenzitu. Modifikace offsetů je až poslední možnost.*