ChnAdam

JEDNOTKA DEFINUJÍCÍ KOMUNIKAČNÍ PROTOKOL PRO KOMUNIKACI S MODULY ADAM 4000

Příručka uživatele a programátora

***SofCon*®spol. s r.o.**
Střešovická 49
162 00 Praha 6
tel/fax: +420 220 180 454
E-mail: sofcon@sofcon.cz
www: http://www.sofcon.cz

Verze dokumentu 1.20

Informace v tomto dokumentu byly pečlivě zkontrolovány a SofCon věří, že jsou spolehlivé, přesto SofCon nenese odpovědnost za případné nepřesnosti nebo nesprávnosti zde uvedených informací.
SofCon negarantuje bezchybnost tohoto dokumentu ani programového vybavení, které je v tomto dokumentu popsané. Uživatel přebírá informace z tohoto dokumentu a odpovídající programové vybavení ve stavu, jak byly vytvořeny a sám je povinen provést validaci bezchybnosti produktu, který s použitím zde popsaného programového vybavení vytvořil.
SofCon si vyhrazuje právo změny obsahu tohoto dokumentu bez předchozího oznámení a nenese žádnou odpovědnost za důsledky, které z toho mohou vyplynout pro uživatele.

Datum vydání: 10.11.2003
Datum posledního uložení dokumentu: 10.11.2003
(Datum vydání a posledního uložení dokumentu musí být stejné)

Obsah :

1. O dokumentu

1.1. Revize dokumentu

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis změn
1.00	1.XX	Wi		První vydání.
1.10	4.XX	Tu	22.05.2003	Úprava dokumentu dle ISO9000.
1.20	4.XX	Wi	10.11.2003	Úprava popisu ChSend

1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky definující komunikační protokol pro komunikaci s moduly ADAM 4000.

1.3. Rozsah platnosti

Určen pro programátory a uživatele programového vybavení SofCon.

1.4. Související dokumenty

Pro čtení tohoto dokumentu je potřeba seznámit se s manuálem ChnTypes a ChnVirt.

Popis formátu verze knihovny a souvisejících funkcí je popsán v manuálu LibVer.

2. Termíny a definice

Používané termíny a definice jsou popsány v samostatném dokumentu Termíny a definice.

3. Úvod

Knihovna definuje formát komunikačního protokolu používaného při komunikaci s moduly ADAM řady 4000. Obstarává zabezpečení dat, vkládání a vyjímání nadbytečností, tak jak to tento protokol předepisuje. Fyzický přenos dat je zajištěn prostřednictvím nižší komunikační vrstvy.

Knihovna ChnAdam definuje komunikační objekt **tChnAdam**, který je dědicem od rodičovského komunikačního objektu tChnVirt. Instance objektu tChnAdam representuje vyšší komunikační vrstvu v komunikačním kanálu. Transformuje předávaná data mezi komunikačními objekty nižších vrstev, které provádějí fyzický přenos, a aplikací nebo případně další vyšší komunikační vrstvou. Určení, přes jakou fyzickou komunikační vrstvu bude komunikace probíhat, je voleno až parametry nastavovací metody ChSetParam.

Uživatel (aplikace) může využívat protokolu Adam ve dvou režimech. Prvním z nich je **textový režim**, kdy vysílací i přijímací buffery jsou již aplikací nadefinované textové řetězce včetně čísla cílové stanice ale bez závěrečného CR a bez případného kontrolního součtu. Tímto režimem lze provádět komunikaci velice efektivně, ale za předpokladu, že uživatel velmi dobře zná strukturu jednotlivých zpráv. Pomocí tohoto režimu lze vysílat a přijímat jakékoliv v budoucnu nové zprávy. Druhým režimem je **datový režim**, kdy uživatel nastavuje pouze hodnoty patřičných položek vysílacího bufferu a sestavení výsledné zprávy zajistí metody objektu tChnAdam. Stejně tak provedou i rozdekódování přijaté zprávy a naplnění patřičných položek přijímacího bufferu.

Knihovna rovněž definuje objekt **tAddChnAdam**, který je dědicem od rodičovského objektu tAddChnVirt. Objekt tAddChnAdam zajistí, aby daný komunikační objekt (objekt tChnAdam) byl k aplikaci připojen a popřípadě zajistí vytvoření instance tohoto objektu. Po přilinkování této jednotky do aplikace (příkazem "uses ChnAdam"), se jméno objektu tChnAdam automaticky vloží do seznamu správců komunikačních objektů pro případné použití.

Protože je objekt **tChnAdam** dědicem rodičovského komunikačního objektu **tChnVirt**, jsou v této příručce popsány jen odlišnosti a speciality pro tento druh sériové komunikace. Ostatní naleznete v příručce **ChnVirt**. Některé použité konstanty a typy jsou předdefinované v jednotce **ChnTypes**.

4. Popis konstant a typů

```
cVerNo = např. $0251; { BCD formát }
cVer   = např. '02.51,07.08.2003';
```

Číslo verze jednotky v BCD tvaru a v textové podobě včetně datumu změny.

```
cName = 'ADAM';
```

Konstanta **cName** definuje jméno komunikačního objektu **tChnAdam**.

4.1. Řídící znaky protokolu

Řídící znaky začátku zprávy od stanice Master

```
DChM_Dolar = '$';
```

Znak pro analogové vstupy a výstupy.

```
DChM_Pound = '#';
```
Znak pro všechny stanice či jiné rozlišení.
```
DChM_Perc  = '%';
```
Znak pro nastavení konfigurace.
```
DChM_At    = '@';
```
Znak pro digitální vstupy a výstupy.

Řídící znaky začátku zprávy od stanice Slave
```
DChS_RepDat= '>';
```
Znak pro odpověd s daty.
```
DChS_RepAck= '!';
```
Znak pro pozitivní odpověd.
```
DChS_RepNak= '?';
```
Znak pro negativní odpověd.

Řídící znaky konce zpráv
```
EndChar    = #$0D;
```
Ukončovací znak všech zpráv (CR).

4.2. Kódy příkazů protokolu

Tyto kódy se nastavují do položky Cmd ve vysílacím bufferu (Master i Slave stanice) při použití datového režimu protokolu. Stejně tak je i položka Cmd v přijímacím bufferu (Master i Slave stanice) naplněna na některou z těchto konstant při přijetí patřičné zprávy.

Všeobecné
```
cCmdInvalidCmd = $FF;
```
Slave stanice hlásí neznámý příkaz.
```
cCmdConfigure  = $00;
```
Nastavení konfigurace.
```
cCmdCfgStatus  = $01;
```
Čtení konfigurace.
```
cCmdRdVer      = $02;
```
Čtení verze firmware.
```
cCmdRdName     = $03;
```
Čtení jména modulu.

Kódy modulů analogových vstupů - analogové vstupy
```
cCmdADataIn    = $10;
```
Čtení analogových dat.
```
cCmdADataInN   = $11;
```
Čtení analogových dat z kanálu N.
```
cCmdRdChnStatus= $12;
```
Čtení stavu všech 8mi kanálů.
```
cCmdEnDiChnMux = $13;
```
Povolení/zakázání multiplexování vstupních kanálů.
```
cCmdSpanCal    = $14;
```
Kalibrace zesílení analogového vstupu.
```
cCmdOffsCal    = $15;
```
Kalibrace ofsetu analogového vstupu.

`cCmdSynchSampl = $16;`
Příkaz pro všechny vstupy všech modulů pro začátek vzorkování do speciálních registrů.
`cCmdRdSyncAData= $17;`
Čtení hodnoty uchované v registru po příkazu CmdSynchSampl.
`cCmdDetTermCoup= $18;`
dotaz na otevření/zavření výměníku.
`cCmdCJCStatus = $19;`
Čtení hodnoty CJC čidla.
`cCmdCJCOffsCal = $1A;`
Kalibrace ofsetu CJC čidla.

Kódy modulů analogových vstupů - konverze dat a displeje

`cCmdRdHLLinMap = $20;`
Čtení horní/dolní meze pro lineární mapování.
`cCmdRdInLinMap = $21;`
Čtení horní/dolní meze mapovaného vstupu pro lineární mapování.
`cCmdWrHLLinMap = $22;`
Zápis horní/dolní meze pro lineární mapování.
`cCmdWrInLinMap = $23;`
Zápis horní/dolní meze mapovaného vstupu pro lineární mapování.
`cCmdEnDiLinMap = $24;`
Povoluje/zakazuje lineární mapování vstupů.
`cCmdLEDDataOrig= $25;`
Nastavení má-li LED displej zobrazovat data z modulu či z nadřízeného PC.
`cCmdSndLEDData = $26;`
Vyslání dat pro LED displej.

Kódy modulů analogových vstupů - rozšíření pro 4018M

`cCmdSetMemCfg = $30;`
Nastavení konfigurace ukládání vzorků do paměti.
`cCmdGetMemCfg = $31;`
Čtení konfigurace ukládání vzorků do paměti.
`cCmdSetMemOper = $32;`
Spustí/zastaví nahrávání stavu paměti.
`cCmdGetMemOper = $33;`
Čtení spuštění/zastavení nahrávání stavu paměti.
`cCmdRdNumEvent = $34;`
Čtení počtu záznamů událostí.
`cCmdRdNumStand = $35;`
Čtení počtu standardních záznamů.
`cCmdRdRecord = $36;`
Čtení specifického záznamu.
`cCmdSetAlarmLim= $37;`
Zápis horní/dolní meze pro alarm.
`cCmdGetAlarmLim= $38;`
Čtení horní/dolní meze pro alarm.

Kódy modulů analogových vstupů - analogové výstupy

`cCmdRdLastOutV = $40;`
Čtení naposledy zapsaných dat pro přímý výstup napěťových dat.
`cCmdOutVoltage = $41;`
Zápis dat pro přímý výstup napěťových dat.
`cCmdStoreDefV = $42;`
Uložení default hodnoty napěťových dat.
`cCmdTrimCalib = $43;`
Kalibrace počtu jednotek pro vyvážení.

```
cCmdZeroCalib  = $44;
```
Kalibrace nuly.
```
cCmdSpanCalib  = $45;
```
Kalibrace rozsahu.

Kódy modulů analogových vstupů - digitální IO, alarmy a události

```
cCmdDDataInAl  = $50;
```
Čtení digitálních vstupů a stavu alarmu.
```
cCmdDDataOut   = $51;
```
Zápis digitálních výstupů.
```
cCmdEnbAlarm   = $52;
```
Povolení alarmu v každém přechodném či záchytném režimu.
```
cCmdSetHiAlarm = $53;
```
Nastavení horní meze pro alarm.
```
cCmdSetLoAlarm = $54;
```
Nastavení dolní meze pro alarm.
```
cCmdDisAlarm   = $55;
```
Zakázání všech alarmů.
```
cCmdClrLatchAl = $56;
```
Vymazání záchytného alarmu.
```
cCmdGetHiAlarm = $57;
```
Čtení horní meze pro alarm.
```
cCmdGetLoAlarm = $58;
```
Čtení dolní meze pro alarm.
```
cCmdGetEventCnt= $59;
```
Čtení počítadla událostí.
```
cCmdClrEventCnt= $5A;
```
Vymazání počítadla událostí.

Kódy modulů analogových výstupů

```
cCmdADataOut   = $60;
```
Zápis analogových výstupů.
```
cCmdDefAOut    = $61;
```
Zápis defult hodnoty analogových výstupů.
```
cCmdTrimCalAOut= $62;
```
Kalibrace počtu jednotek pro vyvážení.
```
cCmd4mACalib   = $63;
```
Zápis parametrů pro kalibraci 4mA.
```
cCmd20mACalib  = $64;
```
Zápis parametrů pro kalibraci 20mA.
```
cCmdGetADataOut= $65;
```
Čtení naposled zapsané hodnoty analogových výstupů.
```
cCmdCurReadback= $66;

cCmdResetSts   = $67;
```
Čtení příznaku resetu modulu od posledního tohoto příkazu.

Kódy modulů digitálních IO a reléových výstupů

```
cCmdDigDataIn  = $70;
```
Čtení digitálních vstupů.
```
cCmdDigDataOut = $71;
```
Zápis digitálních výstupů.
```
cCmdRdSyncDData= $72;
```
Čtení hodnoty uchované v registru po příkazu CmdSynchSampl.

Kódy modulů čítačů a frekvenčních generátorů - vstupy čítačů a displeje

```
cCmdSetInMode  = $80;
```
Nastavení módu vstupního signálu.
```
cCmdGetInMode  = $81;
```
Čtení módu vstupního signálu.
```
cCmdRdCntFreq  = $82;
```
Čtení hodnoty čítače či frekvence.
```
cCmdRdLEDDataOr= $83;
```
Čtení příznaku, má-li LED displej zobrazovat data z modulu či z nadřízeného PC.

Kódy modulů čítačů a frekvenčních generátorů - nastavení čítačů

```
cCmdSetGateMode= $86;
```
Nastavení módu brány modulu čítačů/frekvenčních generátorů.
```
cCmdGetGateMode= $87;
```
Čtení módu brány modulu čítačů/frekvenčních generátorů.
```
cCmdSetMaxCnt  = $88;
```
Nastavení maximální hodnoty pro daný čítač.
```
cCmdGetMaxCnt  = $89;
```
Čtení maximální hodnoty pro daný čítač.
```
cCmdStartStopCt= $8A;
```
Spuštění/zastavení čítače.
```
cCmdGetStsCt   = $8B;
```
Čtení stavu (start/stop) čítače.
```
cCmdClrCounter = $8C;
```
Vynulování čítače.
```
cCmdRdOverFlag = $8D;
```
Čtení příznaku přetečení čítače.

Kódy modulů čítačů a frekvenčních generátorů - digitální filtry

```
cCmdEnDiFilter = $90;
```
Povolení/zakázání digitálního filtru čítače.
```
cCmdGetFilterSt= $91;
```
Čtení stavu (povolení/zakázání) digitálního filtru čítače.
```
cCmdSetMinHiLev= $92;
```
Nastavení minimální šířky signálu pro horní úroveň.
```
cCmdGetMinHiLev= $93;
```
Čtení minimální šířky signálu pro horní úroveň.
```
cCmdSetMinLoLev= $94;
```
Nastavení minimální šířky signálu pro dolní úroveň.
```
cCmdGetMinLoLev= $95;
```
Čtení minimální šířky signálu pro dolní úroveň.
```
cCmdSetNoIHiLev= $96;
```
Nastavení horní spouštěcí úrovně pro neodizolovaný vstup.
```
cCmdGetNoIHiLev= $97;
```
Čtení horní spouštěcí úrovně pro neodizolovaný vstup.
```
cCmdSetNoILoLev= $98;
```
Nastavení dolní spouštěcí úrovně pro neodizolovaný vstup.
```
cCmdGetNoILoLev= $99;
```
Čtení dolní spouštěcí úrovně pro neodizolovaný vstup.

Kódy modulů čítačů a frekvenčních generátorů - digitální výstupy a alarmy

```
cCmdSetIniValCt= $A0;
```
Nastavení počáteční hodnoty čítače.
```
cCmdGetIniValCt= $A1;
```
Čtení počáteční hodnoty čítače.

```
cCmdEnbAlarmCt = $A2;
```
Povolení alarmu čítače.
```
cCmdDisAlarmCt = $A3;
```
Zakázání alarmu čítače.
```
cCmdSetAlLimit0= $A4;
```
Nastavení hodnoty pro alarm čítače 0.
```
cCmdSetAlLimit1= $A5;
```
Nastavení hodnoty pro alarm čítače 1.
```
cCmdGetAlLimit0= $A6;
```
Čtení hodnoty pro alarm čítače 0.
```
cCmdGetAlLimit1= $A7;
```
Čtení hodnoty pro alarm čítače 1.
```
cCmdSetDigOuts = $A8;
```
Nastavení digitálních výstupů.
```
cCmdGetDigOuts = $A9;
```
Čtení digitálních výstupů a stavu alarmu.
```
cCmdSetAlLo0   = $AA;
```
Nastavení dolní hodnoty pro alarm čítače 0.
```
cCmdSetAlHi0   = $AB;
```
Nastavení horní hodnoty pro alarm čítače 0.
```
cCmdGetAlLo0   = $AC;
```
Čtení dolní hodnoty pro alarm čítače 0.
```
cCmdGetAlHi0   = $AD;
```
Čtení horní hodnoty pro alarm čítače 0.

4.3. Kódy rozsahů

4.3.1. Pro moduly 4011, 4011D, 4018, 4018M a některé i pro 4016

```
cRng1_15mV         = $00;
```
Rozsah +/- 15mV. I pro moduly 4016.
```
cRng1_50mV         = $01;
```
Rozsah +/- 50mV. I pro moduly 4016.
```
cRng1_100mV        = $02;
```
Rozsah +/- 100mV. I pro moduly 4016.
```
cRng1_500mV        = $03;
```
Rozsah +/- 500mV. I pro moduly 4016.
```
cRng1_1V           = $04;
```
Rozsah +/- 1V.
```
cRng1_2_5V         = $05;
```
Rozsah +/- 2.5V.
```
cRng1_20mA         = $06;
```
Rozsah +/- 20mA. I pro moduly 4016.
```
cRng1_TCoupleTypJ = $0E;
```
Výměník typu J: 0°C až 760°C.
```
cRng1_TCoupleTypK = $0F;
```
Výměník typu K: 0°C až 1000°C.
```
cRng1_TCoupleTypT = $10;
```
Výměník typu T:-100°C až 400°C.
```
cRng1_TCoupleTypE = $11;
```
Výměník typu E: 0°C až 1000°C.
```
cRng1_TCoupleTypR = $12;
```
Výměník typu R: 500°C až 1750°C.
```
cRng1_TCoupleTypS = $13;
```
Výměník typu S: 500°C až 1750°C.

```
cRng1_TCoupleTypB = $14;
```
Výměník typu B: 500°C až 1800°C.

4.3.2. Pro moduly 4012,4014D,4017

```
cRng2_10V         = $08;
```
Rozsah +/- 10V.
```
cRng2_5V          = $09;
```
Rozsah +/- 5V.
```
cRng2_1V          = $0A;
```
Rozsah +/- 1V.
```
cRng2_500mV       = $0B;
```
Rozsah +/- 500mV.
```
cRng2_150mV       = $0C;
```
Rozsah +/- 150mV.
```
cRng2_20mA        = $0D;
```
Rozsah +/- 20mA.

4.3.3. Pro moduly 4013

```
cRng3_Pt_100100_1 = $20;
```
Platina -100°C až 100°C, $\alpha = 0.00385$.
```
cRng3_Pt100_1     = $21;
```
Platina 0°C až 100°C, $\alpha = 0.00385$.
```
cRng3_Pt200_1     = $22;
```
Platina 0°C až 200°C, $\alpha = 0.00385$.
```
cRng3_Pt600_1     = $23;
```
Platina 0°C až 600°C, $\alpha = 0.00385$.
```
cRng3_Pt_100100_2 = $24;
```
Platina -100°C až 100°C, $\alpha = 0.003916$.
```
cRng3_Pt100_2     = $25;
```
Platina 0°C až 100°C, $\alpha = 0.003916$.
```
cRng3_Pt200_2     = $26;
```
Platina 0°C až 200°C, $\alpha = 0.003916$.
```
cRng3_Pt600_2     = $27;
```
Platina 0°C až 600°C, $\alpha = 0.003916$.
```
cRng3_Ni_80100    = $28;
```
Nikl -80°C až 100°C.
```
cRng3_Ni100       = $29;
```
Nikl - 0°C až 100°C.

4.4. Kódy komunikačních rychlostí Baud Rate

```
cBd_1200          = $03;
```
Rychlost 1200 bps.
```
cBd_2400          = $04;
```
Rychlost 2400 bps.
```
cBd_4800          = $05;
```
Rychlost 4800 bps.
```
cBd_9600          = $06;
```
Rychlost 9600 bps.
```
cBd_19200         = $07;
```
Rychlost 19.2 kbps.
```
cBd_38400         = $08;
```
Rychlost 38.4 kbps.

4.5. Typy formátů reálných čísel

```
tRealFormat =
 (tRF_EngUnits,
```
Běžný inženýrský formát v klasické formě reálného desítkového čísla.
Př. +020.00
```
  tRF_PercFSR,
```
Procenta z větší z hodnot daného rozsahu.
Př1. bude-li rozsah –15 až +15 bude:
```
-15 = -100.00
0   = ±000.00
+15 = +100.00
```
Př2. bude-li rozsah 0 až 1000 bude:
```
0    = ±000.00
1000 = +100.00
```
Př3. bude-li rozsah –100 až +400 bude:
```
-100 = -025.00
0    = ±000.00
+400 = +100.00
```
```
  tRF_TwosCompl,
```
Dvojkový doplněk procent z větší z hodnot z daného rozsahu, kde:
Př1. bude-li rozsah –15 až +15 bude:
```
-15 = -100.00% = $8000
0   = ±000.00% = $0000
+15 = +100.00% = $7FFF
```
Př2. bude-li rozsah 0 až 1000 bude:
```
0    = ±000.00% = $0000
1000 = +100.00% = $7FFF
```
Př3. bude-li rozsah –100 až +400 bude:
```
-100 = -025.00% = $E000
0    = ±000.00% = $0000
+400 = +100.00% = $7FFF
```
```
  tRF_Ohms);
```
Hodnota v Ohmech (Ω). Jen pro moduly 4013 a jen pro některý typ dat.

4.6. Konstanty výsledků přijímacího automatu

V této kapitole jsou popsány definice chybových kódů, které může vracet metoda **ChReceiveResult** v průběhu přijímacího automatu.

```
res_ErrSume  = $20;
```
Výsledek **res_ErrSum** indikuje chybu kontrolního součtu přijaté zprávy.
```
res_ErrFrame = $21;
```
Výsledek **res_ErrFrame** indikuje chybu rámce přijaté zprávy.

4.7. Výčet modulů ADAM řady 4000

Komunikační objekt byl naprogramován pro vysílání a příjem zpráv pro následující moduly Adam řady 4000.

```
tAdamName =
 (tAd4011,  {ADAM-4011  Termocouple Input Module}
  tAd4011D, {ADAM-4011D Termocouple Input Module with LED Display}
```

```
  tAd4012,  {ADAM-4012  Analog Input Module}
  tAd4013,  {ADAM-4013  RTD Input Module}
  tAd4014D, {ADAM-4014D Analog Input Module with LED Display}
  tAd4016,  {ADAM-4016  Strain Gauge Input Module}
  tAd4017,  {ADAM-4017  8-channel Analog Input Module}
  tAd4018,  {ADAM-4018  8-channel Analog Input Module}
  tAd4018M, {ADAM-4018M 8-channel Analog Input Data Logger}
  tAd4021,  {ADAM-4021  Analog Output Module}
  tAd4050,  {ADAM-4050  Digital I/O Module}
  tAd4052,  {ADAM-4052  Isolated Digital Input Module}
  tAd4053,  {ADAM-4053  16-channel Digital Input Module}
  tAd4060,  {ADAM-4060  Relay Output Module}
  tAd4080,  {ADAM-4080  Counter/Frequency Input Module}
  tAd4080D  {ADAM-4080D Counter/Frequency Input Module with LED Display}
    );
```

4.8. Pole jmen modulů ADAM řady 4000

```
cAdamName : array[tAdamName]of string[6] =
   ('4011',
    '4011D',
    '4012',
    '4013',
    '4014D',
    '4016',
    '4017',
    '4018',
    '4018M',
    '4021',
    '4050',
    '4052',
    '4053',
    '4060',
    '4080',
    '4080D'
    );
```

4.9. Množina podporovaných příkazů pro jednotlivé moduly

```
  cAdamCmdSet : array[tAdamName]of set of byte =
   ({ADAM-4011 }
    [cCmdConfigure      , cCmdCfgStatus    ,
     cCmdRdVer          , cCmdRdName       ,
     cCmdADataIn        ,
     cCmdSpanCal        , cCmdOffsCal      ,
     cCmdSynchSampl     , cCmdRdSyncAData  ,
     cCmdCJCStatus      , cCmdCJCOffsCal   ,
     cCmdDDataInAl      , cCmdDDataOut     , cCmdEnbAlarm    ,
     cCmdSetHiAlarm     , cCmdSetLoAlarm   , cCmdDisAlarm    ,
     cCmdClrLatchAl     , cCmdGetHiAlarm   , cCmdGetLoAlarm  ,
     cCmdGetEventCnt    , cCmdClrEventCnt
     ],
    {ADAM-4011D}
    [cCmdConfigure      , cCmdCfgStatus    ,
     cCmdRdVer          , cCmdRdName       ,
     cCmdADataIn        ,
     cCmdSpanCal        , cCmdOffsCal      ,
     cCmdSynchSampl     , cCmdRdSyncAData  ,
     cCmdDetTermCoup    ,
     cCmdCJCStatus      , cCmdCJCOffsCal   ,
     cCmdDDataInAl      , cCmdDDataOut     , cCmdEnbAlarm    ,
     cCmdSetHiAlarm     , cCmdSetLoAlarm   , cCmdDisAlarm    ,
     cCmdClrLatchAl     , cCmdGetHiAlarm   , cCmdGetLoAlarm  ,
     cCmdGetEventCnt    , cCmdClrEventCnt
```

```
 ],
{ADAM-4012 }
[cCmdConfigure      , cCmdCfgStatus    ,
 cCmdRdVer          , cCmdRdName       ,
 cCmdADataIn        ,
 cCmdSpanCal        , cCmdOffsCal      ,
 cCmdSynchSampl     , cCmdRdSyncAData  ,
 cCmdDDataInAl      , cCmdDDataOut     , cCmdEnbAlarm    ,
 cCmdSetHiAlarm     , cCmdSetLoAlarm   , cCmdDisAlarm    ,
 cCmdClrLatchAl     , cCmdGetHiAlarm   , cCmdGetLoAlarm  ,
 cCmdGetEventCnt    , cCmdClrEventCnt
 ],
{ADAM-4013 }
[cCmdConfigure      , cCmdCfgStatus    ,
 cCmdRdVer          , cCmdRdName       ,
 cCmdADataIn        ,
 cCmdSpanCal        , cCmdOffsCal      ,
 cCmdSynchSampl     , cCmdRdSyncAData
 ],
{ADAM-4014D}
[cCmdConfigure      , cCmdCfgStatus    ,
 cCmdRdVer          , cCmdRdName       ,
 cCmdSpanCal        , cCmdOffsCal      ,
 cCmdSynchSampl     , cCmdRdSyncAData  ,
 cCmdRdHLLinMap     , cCmdRdInLinMap   , cCmdWrHLLinMap  ,
 cCmdWrInLinMap     , cCmdEnDiLinMap   , cCmdLEDDataOrig ,
 cCmdSndLEDData     ,
 cCmdDDataInAl      , cCmdDDataOut     , cCmdEnbAlarm    ,
 cCmdSetHiAlarm     , cCmdSetLoAlarm   , cCmdDisAlarm    ,
 cCmdClrLatchAl     , cCmdGetHiAlarm   , cCmdGetLoAlarm  ,
 cCmdGetEventCnt    , cCmdClrEventCnt
 ],
{ADAM-4016 }
[cCmdConfigure      , cCmdCfgStatus    ,
 cCmdRdVer          , cCmdRdName       ,
 cCmdADataIn        ,
 cCmdSpanCal        , cCmdOffsCal      ,
 cCmdSynchSampl     , cCmdRdSyncAData  ,
 cCmdDDataInAl      , cCmdDDataOut     , cCmdEnbAlarm    ,
 cCmdSetHiAlarm     , cCmdSetLoAlarm   , cCmdDisAlarm    ,
 cCmdClrLatchAl     , cCmdGetHiAlarm   , cCmdGetLoAlarm  ,
 cCmdRdLastOutV     , cCmdOutVoltage   , cCmdStoreDefV   ,
 cCmdTrimCalib      , cCmdZeroCalib    , cCmdSpanCalib
 ],
{ADAM-4017 }
[cCmdConfigure      , cCmdCfgStatus    ,
 cCmdRdVer          , cCmdRdName       ,
 cCmdADataIn        , cCmdADataInN     ,
 cCmdEnDiChnMux     , cCmdRdChnStatus  ,
 cCmdSpanCal        , cCmdOffsCal      ,
 cCmdSynchSampl     , cCmdRdSyncAData
 ],
{ADAM-4018 }
[cCmdConfigure      , cCmdCfgStatus    ,
 cCmdRdVer          , cCmdRdName       ,
 cCmdADataIn        , cCmdADataInN     ,
 cCmdEnDiChnMux     , cCmdRdChnStatus  ,
 cCmdSpanCal        , cCmdOffsCal      ,
 cCmdSynchSampl     , cCmdRdSyncAData  ,
 cCmdCJCStatus      , cCmdCJCOffsCal
 ],
{ADAM-4018M}
[cCmdConfigure      , cCmdCfgStatus    ,
 cCmdRdVer          , cCmdRdName       ,
 cCmdADataInN       , cCmdEnDiChnMux   ,
 cCmdRdChnStatus    ,
 cCmdSpanCal        , cCmdOffsCal      ,
```

```
 cCmdSynchSampl     , cCmdRdSyncAData  ,
 cCmdCJCStatus      , cCmdCJCOffsCal   ,
 cCmdSetMemCfg      , cCmdGetMemCfg    , cCmdSetMemOper  ,
 cCmdGetMemOper     , cCmdRdNumEvent   , cCmdRdNumStand  ,
 cCmdRdRecord       , cCmdSetAlarmLim  , cCmdGetAlarmLim
 ],
{ADAM-4021 }
[cCmdConfigure      , cCmdCfgStatus    ,
 cCmdRdVer          , cCmdRdName       ,
 cCmdADataOut       , cCmdDefAOut      , cCmdTrimCalAOut ,
 cCmd4mACalib       , cCmd20mACalib    , cCmdGetADataOut ,
 cCmdCurReadback    , cCmdResetSts
 ],
{ADAM-4050 }
[cCmdConfigure      , cCmdCfgStatus    ,
 cCmdRdVer          , cCmdRdName       ,
 cCmdDigDataIn      , cCmdDigDataOut   ,
 cCmdSynchSampl     , cCmdRdSyncDData  ,
 cCmdResetSts
 ],
{ADAM-4052 }
[cCmdConfigure      , cCmdCfgStatus    ,
 cCmdRdVer          , cCmdRdName       ,
 cCmdDigDataIn      ,
 cCmdSynchSampl     , cCmdRdSyncDData  ,
 cCmdResetSts
 ],
{ADAM-4053 }
[cCmdConfigure      , cCmdCfgStatus    ,
 cCmdRdVer          , cCmdRdName       ,
 cCmdDigDataIn      ,
 cCmdSynchSampl     , cCmdRdSyncDData  ,
 cCmdResetSts
 ],
{ADAM-4060 }
[cCmdConfigure      , cCmdCfgStatus    ,
 cCmdRdVer          , cCmdRdName       ,
 cCmdDigDataIn      , cCmdDigDataOut   ,
 cCmdSynchSampl     , cCmdRdSyncDData  ,
 cCmdResetSts
 ],
{ADAM-4080 }
[cCmdConfigure      , cCmdCfgStatus    ,
 cCmdRdVer          , cCmdRdName       ,
 cCmdSetInMode      , cCmdGetInMode    , cCmdRdCntFreq ,
 cCmdSetGateMode    , cCmdGetGateMode  , cCmdSetMaxCnt ,
 cCmdGetMaxCnt      , cCmdStartStopCt  , cCmdGetStsCt  ,
 cCmdClrCounter     , cCmdRdOverFlag   ,
 cCmdEnDiFilter     , cCmdGetFilterSt  , cCmdSetMinHiLev ,
 cCmdGetMinHiLev    , cCmdSetMinLoLev  , cCmdGetMinLoLev ,
 cCmdSetNoIHiLev    , cCmdGetNoIHiLev  , cCmdSetNoILoLev ,
 cCmdGetNoILoLev    ,
 cCmdSetIniValCt    , cCmdGetIniValCt  , cCmdEnbAlarmCt  ,
 cCmdDisAlarmCt     , cCmdSetAlLimit0  , cCmdSetAlLimit1 ,
 cCmdGetAlLimit0    , cCmdGetAlLimit1  , cCmdSetDigOuts  ,
 cCmdGetDigOuts
 ],
{ADAM-4080D}
[cCmdConfigure      , cCmdCfgStatus    ,
 cCmdRdVer          , cCmdRdName       ,
 cCmdSetInMode      , cCmdGetInMode    , cCmdRdCntFreq  ,
 cCmdLEDDataOrig    , cCmdSndLEDData   , cCmdRdLEDDataOr,
 cCmdSetGateMode    , cCmdGetGateMode  , cCmdSetMaxCnt ,
 cCmdGetMaxCnt      , cCmdStartStopCt  , cCmdGetStsCt  ,
 cCmdClrCounter     , cCmdRdOverFlag   ,
 cCmdEnDiFilter     , cCmdGetFilterSt  , cCmdSetMinHiLev ,
 cCmdGetMinHiLev    , cCmdSetMinLoLev  , cCmdGetMinLoLev ,
```

```
    cCmdSetNoIHiLev    , cCmdGetNoIHiLev  , cCmdSetNoILoLev ,
    cCmdGetNoILoLev    ,
    cCmdSetDigOuts     , cCmdGetDigOuts   ,
    cCmdEnbAlarm       , cCmdDisAlarm     , cCmdClrLatchAl  ,
    cCmdSetAlLo0       , cCmdSetAlHi0     ,
    cCmdGetAlLo0       , cCmdGetAlHi0
    ]
   );
```

4.10. Struktury přijímacích a vysílacích bufferů

```
pMaSendRecord = ^tMaSendRecord;
tMaSendRecord = record
  case Cmd : byte of {kod zpravy}
    cCmdConfigure:   { %aannttccff }
      (NewNode : byte; {nova adresa pro Slave stanici}
       RangeCd : byte; {kod rozsahu (viz kody cRng...)}
       BdRate  : byte; {nova komunikacni rychlost pro Slave stanici
                        (viz kody cBd...)}
       Cfg     : byte; {konfiguracni bity s nasledujici strukturou:
      );
    cCmdCfgStatus  , { $aa2 }
    cCmdRdVer      , { $aaF }
    cCmdRdName     , { $aaM }
    cCmdADataIn    , { #aa }
    cCmdRdChnStatus, { $aa6 }
    cCmdSpanCal    , { $aa0 }
    cCmdOffsCal    , { $aa1 }
    cCmdSynchSampl , { #** }{na tento prikaz neni odpoved}
    cCmdRdSyncAData, { $aa4 }
    cCmdDetTermCoup, { $aaB }
    cCmdCJCStatus  , { $aa3 }
    cCmdRdHLLinMap , { $aa3 }
    cCmdRdInLinMap , { $aa5 }
    cCmdDDataInAl  , { @aaDI}
    cCmdDisAlarm   , { @aaDA}
    cCmdClrLatchAl , { @aaCA}
    cCmdGetHiAlarm , { @aaRH}
    cCmdGetLoAlarm , { @aaRL}
    cCmdGetEventCnt, { @aaRE}
    cCmdClrEventCnt, { @aaCE}
    cCmdRdLastOutV , { $aa6}
    cCmdStoreDefV  , { $aaS}
    cCmdZeroCalib  , { $aaA}
    cCmdSpanCalib  , { $aaB}
    cCmdDefAOut    , { $aa4}
    cCmd4mACalib   , { $aa0}
    cCmd20mACalib  , { $aa1}
    cCmdGetADataOut, { $aa6}
    cCmdCurReadback, { $aa8}
    cCmdResetSts   , { $aa5}
    cCmdDigDataIn  , { $aa6}
    cCmdRdSyncDData, { $aa4}
    cCmdGetFilterSt, { $aa4}
    cCmdGetMinHiLev, { $aa0H}
    cCmdGetMinLoLev, { $aa0L}
    cCmdGetNoIHiLev, { $aa1H}
    cCmdGetNoILoLev, { $aa1L}
    cCmdRdLEDDataOr, { $aa8}
    cCmdGetGateMode, { $aaA}
    cCmdGetInMode  , { $aaB}
    cCmdGetAlLimit0, { @aaRP}
    cCmdGetAlLimit1, { @aaRA}
    cCmdGetAlLo0   , { @aaRP}
    cCmdGetAlHi0   , { @aaRA}
    cCmdGetDigOuts:  { @aaDI}
```

```
  (
  );
cCmdADataInN:    { $aan }
  (Channel : 0..7;
  );
cCmdEnDiChnMux:  { $aa5vv }
  (ChnEnDi : byte; {pole priznaku povoleni jednotlivych vstupu}
  );
cCmdCJCOffsCal:  { $aa9+ $aa9- }
  (IncOffs   : boolean;
   NumOfPoint: word;
  );
cCmdWrHLLinMap:  { $aa6(dataA)(dataB) }
  (DataA,
   DataB : real;
  );
cCmdWrInLinMap:  { $aa7(dataC)(dataD) }
  (DataC,
   DataD : real;
  );
cCmdEnDiLinMap:  { $aaA0 $aaA1 }
  (EnbLinMap : boolean;
  );
cCmdLEDDataOrig: { $aa81 $aa82 }
  (LedDataFromPC : 0..2;{0=z modulu, 1=z modulu, 2=z PC}
  );
cCmdSndLEDData : { $aa9(data) }
  (LedData : real;
  );
cCmdEnbAlarm   : { @aaEAt}
  (AlType    : char; {typ alarmu 'M'=memory, 'L'=latch}
  );
cCmdSetHiAlarm , { @aaHI(data)}
cCmdSetLoAlarm : { @aaLO(data)}
  (AlLimData : real
  );
cCmdOutVoltage : { $aa7(data)}
  (AOutData  : real; {vystupni napeti}
  );
cCmdTrimCalib  , { $aaE(data)}
cCmdTrimCalAOut: { $aa3(data)}
  (NumOfCnts : byte; {pocet zvyseni/snizeni vystupniho napeti o
                      1mV}
  );
cCmdADataOut :   { #aa(data)}
  (ADataOut  : real;
  );
cCmdDDataOut  ,  { @aaDO(data)}
cCmdSetDigOuts:  { @aaDO(data)}
  (DDataOut  : byte;
  );
cCmdDigDataOut:  { #aabb(data)}
  (ChannelSet: byte;
   DigDataOut: byte;
  );
cCmdSetInMode :  { $aaBs}
  (InSigMode : 0..1; {mod vstupniho signalu 0=TTL, 1=optron}
  );
cCmdRdCntFreq ,  { #aan}
cCmdGetMaxCnt ,  { $aa3n}
cCmdGetStsCt   , { $aa5n}
cCmdClrCounter , { $aa6n}
cCmdRdOverFlag , { $aa7n}
cCmdGetIniValCt, { @aaGn}
cCmdEnbAlarmCt,  { @aaEAn}
cCmdDisAlarmCt:  { @aaDAn}
  (Counter   : 0..1; {cislo citace (tato polozka musi byt v
```

```
                          teto strukture na stejnem offsetu jako
                          ostatni CounterX)}
      );
    cCmdSetGateMode: { $aaAg}
      (GateMode: 0..2; {mod brany 0=Lo, 1=Hi, 2=zakazan}
      );
    cCmdSetMaxCnt  : { $aa3n(data)}
      (Counter1 : 0..1;   {cislo citace (tato polozka musi byt v
                           teto strukture na stejnem offsetu jako
                           ostatni CounterX)}
       MaxCtVal : longint;{horni mez citace}
      );
    cCmdStartStopCt: { $aa5ns}
      (Counter2 : 0..1;   {cislo citace (tato polozka musi byt v
                           teto strukture na stejnem offsetu jako
                           ostatni CounterX)}
       FlStart  : boolean;{priznak spusteni citace, jinak
                           zastaveni}
      );
    cCmdEnDiFilter : { $aa40 $aa41}
      (EnbFilter : boolean; {povoleni digitalniho filtru}
      );
    cCmdSetMinHiLev, { $aa0H(data)}
    cCmdSetMinLoLev: { $aa0L(data)}
      (MinInSig : word; {minimalni sirka vstupniho signalu
                         horni/dolni uroven}
      );
    cCmdSetNoIHiLev, { $aa1H(data)}
    cCmdSetNoILoLev: { $aa1L(data)}
      (NonIsTrig : 1..50; {horni/dolni spousteci uroven pro
                           neoddelene vstupy}
      );
    cCmdSetIniValCt: { @aaPn(data)}
      (Counter3 : 0..1;   {cislo citace (tato polozka musi byt v
                           teto strukture na stejnem offsetu jako
                           ostatni CounterX)}
       CtInitVal: longint;{inicializacni hodnota citace}
      );
    cCmdSetAlLimit0, { @aaPA(data)}
    cCmdSetAlLimit1, { @aaSA(data)}
    cCmdSetAlLo0   , { @aaPA(data)}
    cCmdSetAlHi0   : { @aaSA(data)}
      (AlCtLimVal : longint; {mez pro alarm citace}
      );
  end;
```

TMaSendRecord je typ variantního záznamu, který svou strukturou odpovídá datům protokolu Adam posílaným Master stanicí ven z jednotky, přičemž je zbaven nadbytečností, které jsou při vysílání doplněny. Tuto strukturu má vysílací buffer Master stanice v datovém režimu (v textovém režimu je vysílací buffer textový řetězec - string). Položka **Cmd** definuje základní typ vysílané zprávy (viz. 4.2 Kódy příkazů protokolu). Podle hodnoty v Cmd se používají další položky záznamu.

```
pMaRecRecord  = ^tMaRecRecord;
tMaRecRecord  = record
  case Cmd : byte of {kod zpravy}
    cCmdInvalidCmd: { ?aa }
      (
      );
    cCmdConfigure:  { !aa }
      (NewNode : byte; {adresa Slave stanice}
      );
    cCmdCfgStatus:  { !aattccff }
      (RangeCd : byte; {kod rozsahu (viz kody cRng...)}
```

```
    BdRate  : byte; {komunikacni rychlost Slave stanice (viz
                    kody cBd...)}
    Cfg     : byte; {konfiguracni bity}
   );
cCmdRdVer:      { !aa(Ver) }
  (Version : string[10];
  );
cCmdRdName:     { !aa(ModulName) }
  (Name    : string[10];
  );
cCmdADataIn,    { >(data1..dataN) }
cCmdADataInN:   { >(data) }{u teto zpravy ma smysl pouze prvek
                AData[1]}
  (AData   : array[1..16]of real;   {pole hodnot vstupu}
  );
cCmdRdChnStatus: { !aavv }
  (ChnEnDi : byte; {pole priznaku povoleni jednotlivych vstupu}
  );
cCmdRdSyncAData: { !aa(status)(data) }
  (Sts       : boolean;
   ASyncData : real;
  );
cCmdDetTermCoup: { !aa0 !aa1 }
  (TermoCoupleOpen : boolean;
  );
cCmdCJCStatus  : { >(data) }
  (CJCData   : real;
  );
cCmdEnDiChnMux,  { !aa }
cCmdSpanCal,     { !aa }
cCmdOffsCal,     { !aa }
cCmdCJCOffsCal,  { !aa }
cCmdWrHLLinMap,  { !aa }
cCmdWrInLinMap,  { !aa }
cCmdEnDiLinMap,  { !aa }
cCmdLEDDataOrig, { !aa }
cCmdSndLEDData,  { !aa }
cCmdDDataOut  ,  { !aa }
cCmdEnbAlarm  ,  { !aa }
cCmdSetHiAlarm,  { !aa }
cCmdSetLoAlarm,  { !aa }
cCmdDisAlarm  ,  { !aa }
cCmdClrLatchAl,  { !aa }
cCmdClrEventCnt, { !aa }
cCmdOutVoltage,  { !aa }
cCmdStoreDefV ,  { !aa }
cCmdTrimCalib ,  { !aa }
cCmdZeroCalib ,  { !aa }
cCmdSpanCalib ,  { !aa }
cCmdADataOut  ,  { > }
cCmdDefAOut   ,  { !aa }
cCmdTrimCalAOut, { !aa }
cCmd4mACalib  ,  { !aa }
cCmd20mACalib ,  { !aa }
cCmdDigDataOut,  { > }
cCmdEnDiFilter,  { !aa }
cCmdSetMinHiLev, { !aa }
cCmdSetMinLoLev, { !aa }
cCmdSetNoIHiLev, { !aa }
cCmdSetNoILoLev, { !aa }
cCmdSetInMode  , { !aa }
cCmdSetGateMode, { !aa }
cCmdStartStopCt, { !aa }
cCmdClrCounter , { !aa }
cCmdSetIniValCt, { !aa }
cCmdEnbAlarmCt , { !aa }
cCmdDisAlarmCt , { !aa }
```

```
cCmdSetAlLimit0, { !aa }
cCmdSetAlLimit1, { !aa }
cCmdSetAlLo0   , { !aa }
cCmdSetAlHi0   , { !aa }
cCmdSetDigOuts : { !aa }
  (
  );
cCmdRdHLLinMap : { !aa3(dataA)(dataB) }
  (DataA,
   DataB : real;
  );
cCmdRdInLinMap : { !aa3(dataC)(dataD) }
  (DataC,
   DataD : real;
  );
cCmdDDataInAl  : { !aashhll }
  (AlState : 0..2; {stav alarmu 0=disabled, 1=MOMENTARY mode
                   enb, 2=LATCH mode enb}
   DigInsH,        {digitalni vstupy}
   DigInsL : byte;
  );
cCmdGetHiAlarm , { !aa(data) }
cCmdGetLoAlarm : { !aa(data) }
  (AlLimData : real;
  );
cCmdGetEventCnt: { !aa(data) }
  (EvCounter : word;
  );
cCmdRdLastOutV , { !aa(data) }
cCmdGetADataOut: { !aa(data) }
  (AOutData  : real; {vystupni data}
  );
cCmdCurReadback: { !aa(data)}
  (CurRdBack : real;
  );
cCmdResetSts   : { !aa0 !aa1}
  (FlReset   : boolean;
  );
cCmdDigDataIn:   { !aa(dataOut)(dataIn)}
  (DDataO: byte;
   DDataI: byte;
  );
cCmdRdSyncDData :{ !aa(status)(dataOut)(DataIn)00}
  (DSts      : boolean;
   DSyncDataO: byte;
   DSyncDataI: byte;
  );
cCmdGetInMode :  { !aas}
  (InSigMode : 0..1; {mod vstupniho signalu 0=TTL, 1=optron}
  );
cCmdRdCntFreq :  { >(data)}
  (CountVal  : longint;
  );
cCmdRdLEDDataOr: { !aav}
  (LedDataFromPC : 0..2;{0=data z modulu citace0, 1=data z
                         modulu citace0, 2=data z PC}
  );
cCmdGetGateMode: { !aag}
  (GateMode : 0..2; {mod brany 0=Lo, 1=Hi, 2=zakazan}
  );
cCmdGetMaxCnt  : { !aa(data)}
  (MaxCtVal : longint;{horni mez citace}
  );
cCmdGetStsCt   : { !aas}
  (FlStart  : boolean;{priznak spusteni citace}
  );
```

```
      cCmdRdOverFlag : { !aao}
        (OverCount: boolean;{priznak preteceni citace}
        );
      cCmdGetFilterSt: { !aa1 !aa0}
        (EnbFilter : boolean;
        );
      cCmdGetMinHiLev, { !aa(data)}
      cCmdGetMinLoLev: { !aa(data)}
        (MinInSig : word; {minimalni sirka vstupniho signalu
                           horni/dolni uroven}
        );
      cCmdGetNoIHiLev, { !aa(data)}
      cCmdGetNoILoLev: { !aa(data)}
        (NonIsTrig : 1..50; {horni/dolni spousteci uroven pro
                             neoddelene vstupy}
        );
      cCmdGetIniValCt: { !aa(data)}
        (CtInitVal: longint;{inicializacni hodnota citace}
        );
      cCmdGetAlLimit0, { !aa(data)}
      cCmdGetAlLimit1, { !aa(data)}
      cCmdGetAlLo0   , { !aa(data)}
      cCmdGetAlHi0   : { !aa(data)}
        (AlCtLimVal : longint; {mez pro alarm citace}
        );
      cCmdGetDigOuts:  { !aas(data)00}
        (CtSts    : byte;
         CtDataOut: byte;
        );
    end;
```

TMaRecRecord je typ variantního záznamu, který svou strukturou odpovídá datům protokolu Adam přijímaným Master stanicí, přičemž je zbaven nadbytečností, které jsou při příjmu odstraněny. Tuto strukturu má přijímací buffer Master stanice v datovém režimu (v textovém režimu je přijímací buffer textový řetězec - string). Položka **Cmd** definuje základní typ přijaté zprávy (viz. 4.2 Kódy příkazů protokolu). Podle hodnoty v Cmd se používají další položky záznamu.

```
tSlSendRecord = tMaRecRecord;
pSlSendRecord = ^tSlSendRecord;
```

TSlSendRecord je typ variantního záznamu, který svou strukturou odpovídá datům protokolu Adam vysílaným Slave stanicí, přičemž je zbaven nadbytečností, které jsou při vysílání doplněny. Svou vnitřní strukturou je shodný s typem **TMaRecRecord**.

```
tSlRecRecord  = tMaSendRecord;
pSlRecRecord  = ^tSlRecRecord;
```

TSlRecRecord je typ variantního záznamu, který svou strukturou odpovídá datům protokolu Adam přijímaným slave stanicí, přičemž je zbaven nadbytečností, které jsou při příjmu odstraněny. Svou vnitřní strukturou je shodný s typem **TMaSendRecord**.

5. Objekty

5.1. tChnAdam

5.1.1. Položky

`CH_RTick : Boolean;`

Položka **CH_RTick** označuje, že je vykonávaná činnost přijímacího automatu. Tato položka se používá pro ladění.

`CH_SBuff : Pointer;`

Položka **CH_SBuff** definuje ukazatel na vysílací buffer.

`CH_MSBuff : Word;`

Položka **CH_MSBuff** definuje délku vysílacího bufferu.

`CH_LRMess : Word;`

Položka **CH_LRMess** definuje délku přijímané zprávy.

`CH_Master : Boolean;`

Položka **CH_Master** definuje, je-li stanice zapojena v síti jako nadřazená jednotka (Master) nebo jako podřízená jednotka (Slave).

`CH_RSum : tSum8;`

Položka **CH_RSum** se používá pro počítání kontrolního součtu přijímače.

`CH_SSum : tSum8;`

Položka **CH_SSum** se používá pro počítání kontrolního součtu vysílače.

`CH_AdamName: tAdamName;`

Položka **CH_AdamName** určuje, typ modulu stanice Adam v případě stanice Slave i Master. Master stanice používá tuto položku pro správné dekódování příchozí odpovědi od Slave stanice (jelikož různé moduly odpovídají formálně naprosto stejnou zprávou, která má ale pokaždé jiný význam).

`CH_FlUseSum: Boolean;`

Položka **CH_FlUseSum** určuje příznak používání 8-mi bitového kontrolního součtu.

`CH_FlUseStr: Boolean;`

Položka **CH_FlUseStr** určuje příznak, že vysílací buffer je již sestavený textový řetězec (zpráva) pro odvysílání bez případného kontrolního součtu a koncového znaku CR – **textový režim**. Totéž platí i pro přijímací buffer. V opačném případě je použit **datový režim** (vysílací i přijímací buffer je struktura skládající se z jednotlivých položek).

`CH_RangeCode : byte;`

Položka **CH_RangeCode** určuje kód rozsahu reálných dat (modulů analogových vstupů). Tato položka nemá prakticky žádný význam při textovém režimu (CH_FlUseStr = true) pokud uživatel nechce volat metodu RangeRealToStr.

`CH_RealFormat: tRealFormat;`

Položka **CH_RealFormat** určuje formát zápisu reálných dat (modulů analogových vstupů). Tato položka nemá prakticky žádný význam při textovém režimu (CH_FlUseStr = true) pokud uživatel nechce volat metodu RangeRealToStr.

5.1.2. Metody

5.1.2.1. Init konstruktor

```
constructor Init;
```

Konstruktor **Init** slouží k vytvoření a inicializaci instance komunikačního objektu. Ve svém těle zavolá zděděný konstruktor **Init** (inherited Init) od rodičovského objektu tChnVirt a inicializuje položky objektu. Tělo konstruktoru vypadá následovně:

```
  inherited Init;
  CH_Type    := cName;
  CH_Name    := CH_Type;
  CH_NumName := ChNumName(CH_Type);
  CH_RTick   := false;
  CH_SBuff   := nil;
  CH_MSBuff  := 0;
  CH_LRMess  := 0;
  CH_FlUseSum:= false;
  CH_FlUseStr:= false;
  CH_Master  := true;
  CH_AdamName:= tAd4050;
  CH_RangeCode := cRng1_1V;
  CH_RealFormat:= tRF_EngUnits;
  LastCmd    := cCmdInvalidCmd;
  RecStr     := '';
```

5.1.2.2. ChInitParam konstruktor

```
constructor ChInitParam(const S: tParamStr);
```

Konstruktor **ChInitParam** je sloučením konstruktoru **Init** a metody **ChSetParam**. Slouží ke zkrácenému vytvoření instance komunikačního objektu s nastavením parametrů komunikace.

5.1.2.3. Done destruktor

```
destructor Done;
```

Destruktor **Done** slouží ke zrušení instance komunikačního objektu. Pokud je alokován vysílací buffer, je odstraněn z paměti. Na konci destruktoru je volána zděděná metoda **Done** od přímého rodičovského objektu pro uzavření podřízené komunikační vrstvy.

5.1.2.4. ChSetOneParam funkce

```
function ChSetOneParam(const S: tWordString; var CmdL: tCmd)
         : tChResult;
```

Metoda **ChSetOneParam** slouží k dekódování a nastavení jednoho konkrétního parametru, který je zadán v parametru S. Tato metoda se volá v aplikaci prostřednictvím metody **ChSetParam**. Metoda **ChSetOneParam** komunikačního objektu tChnFesto dekóduje tyto parametry:

MAS=MASTER / SLAVE

Parametrem **MAS** ("Master or Slave") se určuje, zda je jednotka v komunikační síti jako Master (nadřízená) nebo jako Slave (podřízená).

LSB=Size

Parametrem **LSB** ("Length of Send Buffer") je alokován nový vysílací buffer **CH_MSBuff** dané velikosti Size.

NOD=Node

Parametrem **NOD** ("Node") se určuje číslo (adresa) stanice **CH_Node** v komunikační síti.

DNO=DNode

Parametrem **DNO** ("Destination Node") se určuje číslo (adresa) stanice **CH_DNode** v komunikační síti, které budou zprávy určeny. Tuto položku je možno také definovat prostřednictvím metody **ChDestNode**.

SUM=ON/OFF

Parametrem **SUM** ("Using CheckSum 8bit") se určí příznak **CH_FlUseSum**, který udává používání kontrolního součtu na konci každé zprávy.

ADN=Name

Parametrem **ADN** ("Adam Name") se určuje jméno modulu Adam. Name je textový řetězec.

STR=ON/OFF

Parametrem **STR** ("Using String format") se určí příznak **CH_FlUseStr**, který udává textový/datový formát vysílacího a přijímacího bufferu.

RNG=bbb

Parametrem **RNG** ("Range Code") se určí kód rozsahu **CH_RangeCode**. bbb nabývá některé z hodnot konstant 4.3 Kódy rozsahů.

RF=ENG/%FSR/2COMP/OHMS

Parametrem **RF** ("Real Format") se určí typ formátu **CH_RealFormat** reálných dat. Hodnota parametru může být:

- 0 nebo ENG - Běžný inženýrský formát v klasické formě reálného desítkového čísla.
- 1 nebo %FSR - Procenta z rozsahu, kde spodní hodnota rozsahu = -100.00, nula = ±000.00, horní hodnota rozsahu = +100.00.
- 2 nebo 2COMP - Dvojkový doplněk procent z rozsahu.
- 3 nebo OHMS - Hodnota v Ohmech. Jen pro moduly 4013.

5.1.2.5. ChGetParam funkce

```
function  ChGetParam(const S: TParamStr): TParamStr;
```

Metoda **ChGetParam** navrací nastavené hodnoty parametrů komunikačního objektu. Nejprve vrátí nastavení parametrů rodičovského komunikačního objektu tChnVirt a poté k nim připojí seznam svých parametrů. Seznam parametrů je uveden výše u popisu metody **ChSetOneParam**.

5.1.2.6. ChConnect procedura

```
procedure ChConnect;
```

Metoda **ChConnect** zavolá zděděnou metodu **ChConnect** od přímého rodičovského objektu a pokud nenastala žádná chyba, nastaví automat přijímače **CH_RCtrl** do počátečního stavu pro příjem zprávy v protokolu Adam.

5.1.2.7. ChDisConnect procedura

```
procedure ChDisConnect;
```

Metoda **ChDisConnect** zavolá zděděnou metodu **ChDisConnect** od přímého rodičovského objektu a pokud nenastala žádná chyba, nastaví automat přijímače **CH_RCtrl** do neaktivního stavu, aby se nepřijímaly žádné zprávy.

5.1.2.8. ChSend procedura

```
procedure ChSend(Buff : Pointer; Len : Word);
```

Metoda **ChSend** způsobí započetí vysílání zprávy podle protokolu Adam. V datovém režimu ukazuje ukazatel Buff na datovou strukturu typu **tMaSendRecord** pro Master stanici nebo **tSlSendRecord** pro Slave stanici. V textovém režimu ukazuje ukazatel Buff na první znak textového řetězce. Parametr Len udává délku vysílacího bufferu pro vysílání (v textovém režimu vlastně délku vysílaného textového řetězce).

Příklad vyslání zprávy pro čtení konfigurace (CmdCfgStatus) ze stanice s adresou 01 v textovém režimu:

```
S : string[20];
...
S:='$012';
ChSend(@S[1],Length(S));
...
```

5.1.2.9. ChReceiveReady funkce

```
function  ChReceiveReady: tChState;
```

Metoda **ChReceiveReady** způsobí provedení kroku přijímacího automatu na základě volání metody **ChReceiveTick**. Jako svoji funkční hodnotu vrací aktuální stav automatu přijímače komunikačního kanálu, který je uložen v položce **CH_RCtrl**. Zpravidla se provádí test pouze na stabilní stav **CHS_ReceiveReady** (který znamená, že byla přijata nějaká zpráva), protože ostatní stavy jsou stavy probíhajícího příjmu.

5.1.2.10. ChReceive procedura

```
procedure ChReceive(var Len: Word);
```

Metoda **ChReceive** provede přijetí celé zprávy a její uložení do přijímacího bufferu, který svou vnitřní strukturou odpovídá datům záznamu typu tRecRecord a byl definován metodou **ChReceiveBuffer**. V datovém režimu metoda naplní přijímací buffer typu **tMaRecRecord** pro Master stanici nebo **tSlRecRecord** pro Slave stanici pouze patřičnými položkami, úvodní a zakončovací řídicí znaky i případný kontrolní součet metoda ze zprávy vyhodnotí a pro uživatele odstraní. V textovém režimu naplní přijímací buffer typu **string** celou přijatou textovou zprávou bez závěrečného ukončovacího znaku CR a případného kontrolního součtu. V parametru Len vrátí velikost přijaté zprávy. Odpověď na zprávu, ať došla v pořádku nebo porušená, generuje uživatel sám pomocí metody **ChSend**.

5.1.2.11. ChReceiveFlush procedura

```
procedure ChReceiveFlush;
```

Metoda **ChReceiveFlush** způsobí vyprázdnění přijímacích bufferů a nastavení stavu automatu přijímače na počátek příjmu zpráv v protokolu Adam.

5.1.2.12. ChReceiveTick procedura

```
procedure ChReceiveTick;
```

Metoda **ChReceiveTick** způsobí provedení jednoho či více kroků automatu přijímače. Je nutné ji periodicky volat při přijímání. Metoda **ChReceiveTick** je rovněž automaticky volána v metodě **ChReceiveReady**.

5.1.2.13. RangeRealToStr funkce

```
function RangeRealToStr(R : real) : tString32;
```

Metoda **RangeRealToStr** převede podle položek **CH_RealFormat** a **CH_RangeCode** reálné číslo R na jeho textovou reprezentaci v daném formátu.

Pokud CH_RealFormat = tRF_EngUnits zadává se v R přímo daná reálná hodnota v příslušných jednotkách.

Pokud CH_RealFormat = tRF_PercFSR nebo tRF_TwosCompl zadávají se v R příslušná procenta z FSR.

Pokud CH_RealFormat = tRF_Ohms zadává se v R hodnota v ohmech.

5.1.2.14. RangeStrToReal funkce

```
function RangeStrToReal(S : tString32) : real;
```

Metoda **RangeStrToReal** převede podle položek **CH_RealFormat** a **CH_RangeCode** textovou reprezentaci čísla v daném formátu na reálné číslo. Je to inverzní funkce k funkci RangeRealToStr.

5.2. tAddChnAdam

Typ **tAddChnAdam** je typem objektu, který slouží k definování prvku v seznamu správců komunikačních objektů (tzv. správce komunikačního objektu tChnAdam v seznamu správců). Objekt tAddChnAdam je dědicem od rodičovského objektu **tAddChnVirt**.

5.2.1. Metody

5.2.1.1. ChInit funkce

```
function ChInit: pChnVirt;
```

Metoda **ChInit** slouží k vytvoření instance komunikačního objektu tChnAdam a ukazatel na instanci tohoto objektu vrací jako svoji funkční hodnotu.

6. Struktura zpráv protokolu ADAM

Jelikož protokol ADAM disponuje přibližně 150 jedinečnými zprávami, byl by popis velice rozsáhlý, měla-li by se zde popisovat každá zpráva. Proto jsou zde podrobně popsány nejběžnější všeobecné zprávy a pro detailní popis ostatních odkazuji čtenáře na příslušnou dokumentaci firmy **Advantech Co., Ltd**, ze které bylo také čerpáno při tvorbě této komunikační knihovny.

Pokud bude uživatel používat **textový režim** protokolu, kdy musí nadefinovat celou zprávu sám, musí znát strukturu těchto zpráv.

Pokud bude uživatel používat **datový režim** protokolu, musí **vždy** správně **naplnit** položku **Cmd** ve struktuře vysílacího bufferu. Ze struktury (viz. Struktura přijímacích a vysílacích bufferů) dále plyne, jaké další položky se používají pro konkrétní hodnoty Cmd.

6.1. Struktura zprávy pro čtení jména připojeného modulu

Master stanice vysílá zprávu:

'$'	SlaveNode	'M'	(Sum)	CR

SlaveNode – 2 Ascii znaky v hexadecimální representaci – adresa cílové Slave stanice. V datovém režimu se tato adresa doplní sama z položky CH_DNode. V textovém režimu ji uživatel musí nadefinovat sám (položka CH_DNode nemá význam).

(Sum) – variabilní kontrolní součet – není ho třeba definovat, ke zprávě se automaticky připojí (pokud je položkou CH_FlUseSum určeno, že se má používat).

CR – hodnota $0D – není ho třeba definovat, ke zprávě se automaticky připojí.

Při použití **datového režimu** odpovídají této zprávě tyto nastavené položky ve struktuře vysílacího bufferu Master stanice:

```
Cmd = cCmdRdName;
```

V případě **textového režimu** musí uživatel naplnit vysílací buffer (který je representován textovým řetězcem) až do položky „(Sum)“ kromě ní.

Slave stanice odpovídá zprávou:

'!'	SlaveNode	ModulName	(Sum)	CR

ModulName – jméno modulu Slave stanice.

Při použití **datového režimu** odpovídají této zprávě tyto nastavené položky ve struktuře vysílacího bufferu Slave stanice:

```
Cmd = cCmdRdName;
Name = '...'; {jméno dané Slave stanice, např.'4011'}
```

6.2. Struktura zprávy pro čtení konfigurace připojeného modulu

Master stanice vysílá zprávu:

'$'	SlaveNode	'2'	(Sum)	CR

Při použití **datového režimu** odpovídají této zprávě tyto nastavené položky ve

struktuře vysílacího bufferu Master stanice:

```
Cmd = cCmdCfgStatus;
```

V případě **textového režimu** musí uživatel naplnit vysílací buffer (který je representován textovým řetězcem) až do položky „(Sum)“ kromě ní.

Slave stanice odpovídá zprávou:

‘!’	SlaveNode	TT CC FF	(Sum)	CR

TT – 2 Ascii znaky v hexadecimální representaci – kód rozsahu.
CC – 2 Ascii znaky v hexadecimální representaci – kód komunikační rychlosti.
FF – 2 Ascii znaky v hexadecimální representaci – pole konfiguračních bitů.

Při použití **datového režimu** odpovídají této zprávě tyto nastavené položky ve struktuře vysílacího bufferu Slave stanice:

```
Cmd = cCmdCfgStatus;
RangeCd = cRng1_50mV; {nebo jiný kód rozsahu}
BdRate  = cBd_9600; {nebo jiný kód komunikační rychlosti}
Cfg     = $80; {závisí podle daného modulu a nastavení}
```

6.3. Struktura zprávy pro nastavení konfigurace připojeného modulu

Master stanice vysílá zprávu:

‘%’	SlaveNode	NewSlaveNode	TT CC FF	(Sum)	CR

NewSlaveNode – 2 Ascii znaky v hexadecimální representaci – nová adresa připojeného modulu.
TT – 2 Ascii znaky v hexadecimální representaci – kód rozsahu.
CC – 2 Ascii znaky v hexadecimální representaci – kód komunikační rychlosti.
FF – 2 Ascii znaky v hexadecimální representaci – pole konfiguračních bitů.

Při použití **datového režimu** odpovídají této zprávě tyto nastavené položky ve struktuře vysílacího bufferu Master stanice:

```
Cmd = cCmdConfigure;
NewNode = 10; {nebo jiná nová adresa modulu}
RangeCd = cRng1_50mV; {nebo jiný kód rozsahu}
BdRate  = cBd_9600; {nebo jiný kód komunikační rychlosti}
Cfg     = $80; {závisí na daném modulu a nastavení}
```

V případě **textového režimu** musí uživatel naplnit vysílací buffer (který je representován textovým řetězcem) až do položky „(Sum)“ kromě ní.

Slave stanice odpovídá zprávou:

‘!’	SlaveNode	(Sum)	CR

SlaveNode je již nová adresa modulu.

Při použití **datového režimu** odpovídají této zprávě tyto nastavené položky ve struktuře vysílacího bufferu Slave stanice:

```
Cmd = cCmdConfigure;
```

6.4. Struktura zprávy „Invalid Cmd“ od Slave stanice

Pokud Slave stanice (modul ADAM) nebyla schopna zprávu od Master stanice správně dekódovat, odpovídá právě touto zprávou.

Master stanice posílá nějakou nesmyslnou zprávu:

nesmysly	CR

Slave stanice odpovídá zprávou:

‘?‘	SlaveNode	(Sum)	CR

Tomu odpovídají tyto nastavené položky ve struktuře vysílacího bufferu Slave stanice:

```
Cmd = cCmdInvalidCmd;
```

7. Příklad

Následující příklad ukazuje způsob vyslání zprávy pro čtení jména modulu Adam nejprve v datovém režimu a poté v textovém režimu. Fyzický přenos se realizuje prostřednictvím knihovny ChnCom.

```
uses
  NumToStr,
  uString,
  ChnVirt,
  ChnCom,
  ChnAdam;

const
  ParamStr : tParamStr =
          'NAM=ADAM NOD=0 DNO=1 LSB=200 SUM=OFF STR=OFF ADN=4011'+
          ' RNG=$01 RF=ENG' +
          'NAM=COM COM=1 IRQ=4 BD=9600 BIT=8 STOP=1 PAR=N LRB=1000';

var
  Chn   : pChnVirt;      {ukazatel na komunikační objekt}
  SMess : pMaSendRecord; {ukazatel na vysílací buffer v datovém rež}
  RMess : pMaRecRecord;  {ukazatel na přijímací buffer v datovém rež}
  SStr  : string;        {vysílací buffer v textovém režimu}
  RStr  : string;        {přijímací buffer v textovém režimu}
  LSMess: word;          {délka vysílané zprávy}
  LRMess: word;          {délka přijaté zprávy}

begin
  ...
  New(SMess);
  New(RMess);
  ...
 { vytvoření instance Chn }
  Chn:=ChnCollection^.ChNewInit(ChnAdam.cName);
  with Chn^ do
  begin
   { nastavení parametrů komunikace }
    ChSetParam(ParamStr);
    if ChResult<>res_Ok then WriteLn('Chyba');
```

```
  ChOpen;
  repeat
    if ChResult<>res_Ok then WriteLn('Chyba');
  until ChReady=CHS_Open;
  if ChResult<>res_Ok then WriteLn('Chyba');
  { definování místa, kam se má přijatá zpráva uložit }
  ChReceiveBuffer(RMess,SizeOf(RMess^));
  if ChReceiveResult<>res_Ok then WriteLn('Chyba');
  ChConnect;
  repeat
    if ChResult<>res_Ok then WriteLn('Chyba');
  until ChReady=CHS_Connect;
  if ChResult<>res_Ok then WriteLn('Chyba');
  ...
 { příklad naplnění zprávy daty (např. pro čtení jména modulu)}
  with SMess^ do
  begin
    Cmd   := cCmdRdName;
  end;
 { vyslání zprávy }
  if ChSendReady=CHS_SendReady then
  begin
    ChSend(SMess, 1);
    { čekání na odvysílání zprávy }
    repeat
      if ChSendResult<>res_Ok then WriteLn('Chyba');
    until ChSendReady=CHS_SendReady;
    if ChSendResult<>res_Ok then WriteLn('Chyba');
    ...
  end;
  ...
 { čekání na příjem zprávy }
  while not ChReceiveReady=CHS_ReceiveReady do
  begin
    if ChReceiveResult<>res_Ok then WriteLn('Chyba');
  end;
  { příjem zprávy }
  ChReceive(LRMess);
  if ChReceiveResult<>res_Ok then WriteLn('Chyba')
  else
    { dekódování správné odpovědi (např. odpověď na ReadModulName)}
    if RMess^.Cmd  = cCmdRdName then
      writeln('Jmeno stanice je ', RMess^.Name)
    else
      writeln('Chyba');
  ...
 { změna z datového režimu na textový }
  ChSetParam('NAM=ADAM STR=ON');
  if ChResult<>res_Ok then writeln('Chyba');
  ChReceiveBuffer(@RStr[0],SizeOf(RStr));
  if ChReceiveResult<>res_Ok then WriteLn('Chyba');
  { příklad naplnění zprávy daty (např. pro čtení jména modulu)}
  SStr:= '$01M';
  { vyslání zprávy }
  if ChSendReady=CHS_SendReady then
  begin
    ChSend(@SStr[1],Length(SStr));
    { čekání na odvysílání zprávy }
    repeat
      if ChSendResult<>res_Ok then WriteLn('Chyba');
    until ChSendReady=CHS_SendReady;
    if ChSendResult<>res_Ok then WriteLn('Chyba');
    ...
  end;
  ...
```

```
   { čekání na příjem zprávy }
    while not ChReceiveReady=CHS_ReceiveReady do
    begin
      if ChReceiveResult<>res_Ok then WriteLn('Chyba');
    end;
    { příjem zprávy }
    ChReceive(LRMess);
    if ChReceiveResult<>res_Ok then WriteLn('Chyba')
    else
    begin
      RStr[0]:=Chr(LRMess);
      Writeln('Prijata zprava: ',RStr);
    end;
    ...
   { ukončení }
    ChDisConnect;
    repeat
      if ChResult<>res_Ok then WriteLn('Chyba');
    until ChReady=CHS_DisConnect;
    if ChResult<>res_Ok then WriteLn('Chyba');
    ChClose;
    repeat
      if ChResult<>res_Ok then WriteLn('Chyba');
    until ChReady=CHS_Close;
    if ChResult<>res_Ok then WriteLn('Chyba');
  end; {with}

  { zrušení instance Chn }
  Dispose(Chn,Done);
  Dispose(SMess);
  Dispose(RMess);
  ...
end.
```