Komunikace v síti M-Bus (Gen2)

Abstrakt

Přenos technologických dat mezi řídicími systémy Generace 2 firmy AMiT a měřiči energií prostřednictvím protokolu M-Bus.

Autor: Michal Kupčík
Dokument: ap0062_ap_cz_002.pdf

Příloha

Obsah souboru: ap0062_ap_cz_002.zip

mbus_p1_cz_101.dsox	Načtení dat z měřičů do řídicího systému prostřednictvím objektů UserComBinary včetně dekódování
mbus_p2_cz_101.dsox	Načtení dat z měřičů do řídicího systému prostřednictvím protokolů MODBUS RTU a MODBUS TCP včetně dekódování

Obsah

Historie revizí

Verze	Datum	Autor změny	Změny
001	13. 04. 2021	Kupčík M.	Nový dokument.
002	06. 04. 2022	Kupčík M.	Lokální opravy textu, upravené hodnoty obsazení paměti v kap. 2.2.2.

Související dokumentace

1. Nápověda k části EsiDet vývojového prostředí **DetStudio**
   soubor: EsiDet_cs.chm
2. **DM-MBUS64** – komunikační převodník s protokolem M-Bus – Návod na obsluhu
   soubor: dm-mbus64_g_cz_xxx.pdf
3. Aplikační poznámka AP0056 – Komunikace v síti MODBUS RTU
   soubor: ap0056_ap_cz_xxx.pdf
4. Aplikační poznámka AP0057 – Komunikace v síti MODBUS TCP
   soubor: ap0057_ap_cz_xxx.pdf
5. EN 13757 – Sada norem specifikující protokol M-Bus

Určení aplikační poznámky

Aplikační poznámka je určena řídicím systémům Generace 2 – řídicím systémům s operačním systémem FreeRTOS programovaných v DetStudiu v editoru EsiDet. Více informací o generacích systémů lze nalézt na webu **amitomation.cz**.

1 M-Bus

Standard M-Bus (z anglického Meter-Bus) je určen pro sběr dat z měřičů odběru nejrůznějších médií (například pitné a užitkové vody, plynu, tepla, elektrické energie). Byl vyvinut ve spolupráci University Paderborn a firmy Texas Instruments.

1.1 Základní charakteristika

Vzhledem k relativně úzké a poměrně specializované aplikační oblasti jsou na M-Bus kladeny specifické požadavky. Musí zajistit propojení velkého množství zařízení na vzdálenost až několika kilometrů. Přenos dat musí být kvalitně zabezpečen proti chybám. Na druhé straně je typickou vlastností aplikace nepříliš časté odečítání naměřených hodnot s nízkými nároky na odezvy v reálném čase.

Charakteristické rysy standardní konfigurace lze shrnout do následujících bodů:

- speciální implementace fyzické vrstvy,
- galvanicky oddělené rozhraní,
- možnost napájení připojených zařízení po sběrnici,
- dvoudrátové vedení s délkou až několik kilometrů,
- řízení komunikace na principu Master – Slave,
- bez implementace síťové vrstvy maximálně 250 účastníků,
- asynchronní přenos znaků, 8 bitů dat, sudá parita,
- přenosová rychlost 300 bps až 38400 bps,
- zabezpečení datového bloku pomocí kontrolního součtu.

Standardní konfigurace M-Busu zahrnuje jedinou řídicí stanici (Master) a maximálně 250 účastnických stanic. Typický způsob zapojení M-Bus měřičů je do linie. Délka kabelového segmentu v této konfiguraci nesmí překročit 1000 m (350 m pro 9600 bps). Pro rozsáhlejší systémy je nezbytné přejít k složitějším konfiguracím, kdy je celý systém rozdělen na tzv. zóny. Jednotlivé zóny se skládají ze segmentů a jsou řízeny tzv. řadiči zóny.

Kompletní detailní popis M-Bus protokolu poskytuje EN 13757.

1.2 Fyzická vrstva

Standard M-Bus používá speciální implementace fyzické vrstvy protokolu, které jsou popsány v EN 13757-2 a EN 13757-4. Typicky se jedná o dvoudrátovou sběrnici na bázi běžného telefonního kabelu s poloduplexním přenosem dat a řízením přístupu Master – Slave. Aby bylo možné napájení účastnických stanic po tomtéž vedení, používá M-Bus pro přenos od řídicí stanice k účastnickým stanicím změny napěťových úrovní, ve směru opačném změny v odběru proudu. Ve směru od řídicí stanice k účastnickým stanicím odpovídá logické jedničce napětí +36 V na výstupu budiče řídicí stanice, logické nule pak napětí o 12 V nižší, tj. +24 V. Ve směru od účastnické stanice k řídicí stanici je logická jednička reprezentována proudovým odběrem 1,5 mA, logická nula odběrem o 11 mA až 20 mA vyšším. Proud odebíraný ve stavu log. 1 může být využit k napájení galvanicky odděleného rozhraní a případně i vlastního měřiče.

V klidovém stavu je na sběrnici úroveň log. 1 (+36 V) a odběr z budiče řídicí stanice odpovídá počtu účastnických stanic násobeným odběrem účastnické stanice ve stavu log. 1, tj. 1,5 mA. Je zřejmé, že zejména při vyšším počtu účastnických stanic klade standard na budič řídicí stanice poměrně vysoké nároky. Navíc vzhledem k proměnnému počtu účastnických stanic a k požadavku možnosti jejich přidávání a odebírání za provozu nemůže řídicí stanice detekovat absolutní odebíraný proud, ale pouze výše uvedenou změnu. Stejně tak napětí v různých bodech sběrnice, které je monitorováno účastnickými stanicemi, je proměnné a závisí na odporu vedení a na počtu účastnických stanic k němu připojených. To znamená, že také účastnické stanice musí reagovat na změny napětí sběrnice, nikoliv na jejich absolutní hodnoty. Vzhledem k poměrně velkým změnám jak napětí (12 V) tak proudu (11 mA až 20 mA) vykazuje fyzická vrstva standardu vysokou odolnost vůči vnějšímu rušení.

Druhá varianta je použití bezdrátového přenosu dat. Vzhledem k nemožnosti komunikace řídicích systémů AMiT v daných frekvenčních pásmech není tato varianta v této aplikační poznámce více řešena.

1.3 Formát rámce

M-Bus využívá čtyř formátů rámce:
- jednotlivý znak (Single Character),
- krátký rámec (Short Frame),
- řídicí rámec (Control Frame),
- dlouhý rámec (Long Frame).

Jednotlivý znak
0xE5

Krátký rámec
Start 0x10
C pole
A pole
Kontrolní součet
Stop 0x16

Řídicí rámec
Start 0x68
L pole = 3
L pole = 3
Start 0x68
C pole
A pole
CI pole
Kontrolní součet
Stop 0x16

Dlouhý rámec
Start 0x68
L pole
L pole
Start 0x68
C pole
A pole
CI pole
Uživatelská data (0 až 252) bajtů
Kontrolní součet
Stop 0x16

Jednotlivý znak
Tento rámec se skládá z jediného znaku, jmenovitě 0xE5. Používá se k potvrzení o přijetí jiného vyslaného rámce.

Krátký rámec
Tento rámec má pevnou délku. Začíná úvodním znakem 0x10, následuje pole C, pole A, kontrolní součet a koncový znak 0x16. Kontrolní součet se počítá pouze z polí C a A.

Řídicí rámec
Obsah řídicího rámce odpovídá dlouhému rámci, neobsahuje však položku „uživatelská data“. Kontrolní součet je počítán z polí C, A a CI.

Dlouhý rámec
Tento rámec začíná znakem 0x68, pokračuje dvakrát opakovaným polem L, obsahujícím délku dat a opět úvodním znakem 0x68. Poté následuje pole C, pole A, pole CI, uživatelská data o délce (0 až 252) bajtů, kontrolní součet a ukončovací znak 0x16. Pole L obsahuje počet bajtů uživatelských dat zvýšený o tři (tj. o délku polí C, A a CI). Kontrolní součet je počítán z polí C, A, CI a z uživatelských dat.

1.3.1 Význam polí rámce

Všechna pole mají délku jednoho znaku, což odpovídá osmi bitům.

Pole C (Control Field, Function Field)
Toto pole obsahuje řídicí parametry rámce. Mimo jiné určuje směr toku dat. V popise níže je označován jako hlavní bajt rámce.

Pole A (Address Field)
Pole A slouží k adresaci či určení účastnické stanice při vysílání, resp. příjmu rámce.

Jednotlivým účastnickým stanicím (slaveům) mohou být přiřazeny adresy 0 až 250. V případě, že bude do adresního pole zadána hodnota 255, bude vyslán tzv. nepotvrzovaný broadcast (nepřijde žádná odpověď). Pokud bude do adresního pole zadána hodnota 254 dojde k vyslání tzv. potvrzovaného broadcastu (očekává se odpověď). Je zřejmé, že při potvrzení broadcastu dojde na sběrnici ke kolizím vždy, je-li připojena více než jedna účastnická stanice. Jeho použití je pouze pro testovací účely. Adresa 253 indikuje tzv. sekundární adresaci. Adresy 251 a 252 jsou rezervovány pro budoucí využití.

V této aplikační poznámce se sekundární adresace neřeší.

Pole CI (Control Information Field)
Pole CI nese informaci, která je již součástí aplikačního protokolu. Jeho úkolem je kromě jiného rozlišení formátu dlouhého a řídicího rámce.

Uživatelská data
V případě, že se jedná o odpověď měřiče s běžnými daty (pole CI má hodnotu 0x72 nebo 0x76 – RSP_UD), mají tyto bajty následující význam:

- Hlavičkové bajty 8 až 19:
  - 4 bajty identifikačního čísla měřiče, často odpovídající části sériového čísla nebo sekundární adresy,
  - 2 bajty kódující tři identifikační písmena výrobce měřiče,
  - 1 bajt verze měřiče,
  - 1 bajt kódující měřené médium,
  - 1 bajt počítadla rámců,
  - 1 bajt kódující stav měřiče,
  - 2 bajt „signature“ – příprava pro budoucí specifikaci, hodnoty 0x00 0x00.
- následující datové bajty:
  - 1 bajt DIF – definice typu čísla hodnoty; pokud je nastaven nejvyšší bit, bude následovat DIFE,
  - 0 až 10 bajtů DIFE – rozšiřující identifikátor hodnoty; pokud je nastaven nejvyšší bit, bude následovat další DIFE,
  - 1 bajt VIF – jednotka a násobitel hodnoty; pokud je nastaven nejvyšší bit, bude následovat VIFE,
  - 0 až 10 bajtů VIFE – rozšiřující identifikátor jednotky a násobitele hodnoty; pokud je nastaven nejvyšší bit, bude následovat další VIFE,
  - 0 až xx bajtů hodnota; v případě ASCII jednotky je tato prvně poslána a následně jsou poslány bajty hodnoty,
  - následuje tolik opakování kombinací DIF (+ n*DIFE) + VIF (+ m*VIFE) + hodnota, kolik veličin měřič v této zprávě zasílá.

1.3.2 Využití rámců

V této aplikační poznámce se využívají rámce:

- jednotlivý znak – odpověď měřiče na inicializaci,
- krátký rámec – inicializace měřiče (SND_NKE) – použit hlavní bajt rámce 0x40,
- krátký rámec – požadavek na běžná data (SND_UD2) – použit hlavní bajt rámce 0x5B nebo 0x7B,
- dlouhý rámec – odpověď měřiče s běžnými daty (RSP_UD) – přijat hlavní bajt 0x08/0x18/0x28/0x38.

2 Komunikace řídicích systémů pomocí M-Bus

Řídicí systémy firmy **AMiT** zastávají **v síti M-Bus** vždy roli **Master** a nelze je tedy v síti M-Bus provozovat v režimu Slave.

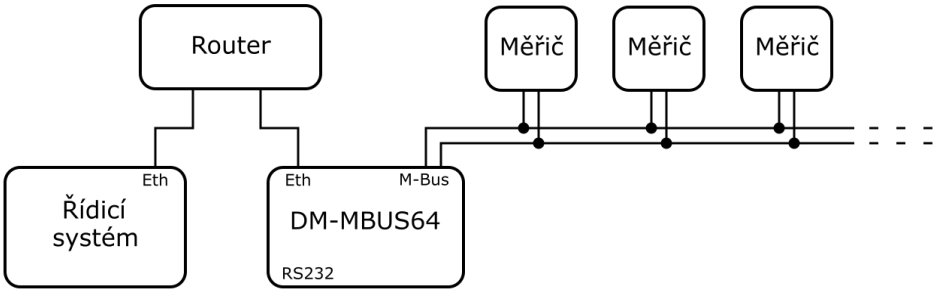
2.1 HW parametrizace

Pro komunikaci řídicích systémů v síti M-Bus je nutné využít převodník. Firma AMiT nabízí pro komunikaci v síti M-Bus převodník **DM-MBUS64**, který lze k řídicímu systému připojit jak prostřednictvím sériového rozhraní, tak prostřednictvím rozhraní Ethernet. Převodník **neprovádí** žádné přímé dekódování hodnot z měřičů.

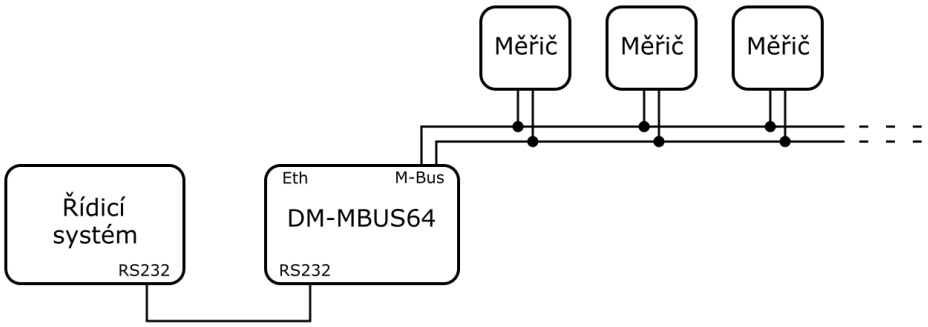
Parametrizace převodníku se provádí typicky prostřednictvím webových stránek, když je převodník v tzv. servisním režimu. Více informací o parametrizaci převodníku jsou uvedeny v návodu na obsluhu převodníku.

Převodník umožňuje komunikaci prostřednictvím transparentního přenosu dat (režim Direct), zapouzdření M-Bus dat do protokolu APE a prostřednictvím protokolů MODBUS. V následující kapitole se očekává nastavení převodníku na transparentní přenos dat (režim Direct).

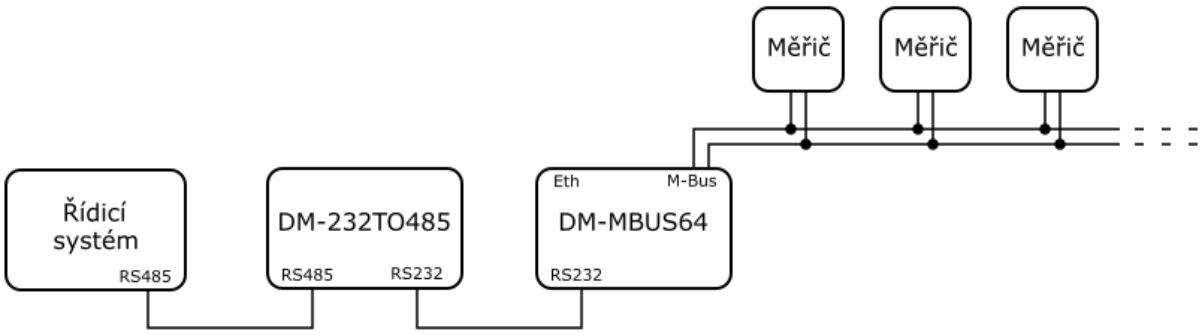
Bližší popis protokolu APE je uveden v návodu na obsluhu převodníku.



Obr. 1 – Připojení řídicího systému k převodníku **DM-MBUS64** přes rozhraní Ethernet



Obr. 2 – Připojení řídicího systému k převodníku **DM-MBUS64** přes rozhraní RS232



Obr. 3 – Připojení řídicího systému k převodníku **DM-MBUS64** přes RS485 a převodník **DM-232TO485**

Algoritmy uvedené níže lze použít i v případě použití převodníku jiného výrobce. Převodník jiného výrobce však musí umožnovat transparentní přenos dat (co přijme prostřednictvím sériového rozhraní nebo rozhraní Ethernet, to musí poslat na rozhraní M-Bus bez jakékoliv změny [po stránce SW] a obráceně).

Pozor
Na jednom sériovém rozhraní řídicího systému nebo terminálu, lze provozovat pouze jeden komunikační protokol. V případě využití komunikace s převodníkem po sériovém rozhraní musí být zvolen stejný komunikační protokol jako s případnými jinými zařízeními na tomto sériovém rozhraní.

2.2 SW parametrizace

Pro komunikaci s měřičem se typicky využívá následujícího postupu:

1. Vyslání inicializačního krátkého rámce,
2. čekání na jednoznakovou odpověď 0xE5,
3. a) v případě, že odpověď nepřijde, opakovat vyslání inicializačního krátkého rámce,
   b) v případě, že odpověď přijde, vyslat krátký rámec s požadavkem na běžná data,
4. a) v případě, že odpověď s běžnými daty nepřijde, opakovat vyslání krátkého rámce s požadavkem na běžná data,
   b) v případě, že odpověď s běžnými daty přijde, provést jejich dekódování,
5. ve výjimečných případech, kdy se z měřiče mají získat další běžné data, opakovat bod 3. b) s druhou hodnotou hlavního bajtu rámce 0x5B / 0x7B.

Funkční blok **`fb_MBusDevice`** řeší vyslání inicializačního rámce a vyslání požadavku na běžná data. Dále funkční blok řeší příjem dat a případné opakování posledního zaslaného rámce.

Funkční blok **`fb_MBusDecode`** provádí dekódování přijatých běžných dat na numerické a textové hodnoty, případně signalizuje problémy při dekódování.

2.2.1 Funkční blok fb_MBusDecode

Základní charakteristiky funkčního bloku:

- dekódování až 40 numerických hodnot z měřiče – parametr **`Values`**,
- dekódování až 10 textových hodnot z měřiče – parametr **`ValuesStrings`**,
- ke každé hodnotě je uvedena zjištěná jednotka – parametr **`UnitsStrings`**,
- prezentace číselného kódu média měřiče – parametr **`DeviceInfo`**,
- dekódování třípísmenného identifikátoru výrobce – parametr **`DeviceInfo`**,
- podpora fixních standardních formátů hodnot: Int (1 až 8 bajtů), Real (4 bajty) a BCD (2 až 12 cifer),

- podpora „LVAR“ variabilních formátů hodnot: ASCII řetězec (maximálně 20 znaků), pozitivní a negativní BCD hodnotu (maximálně 16 cifer) a binární číslo (maximálně 8 bajtů),
- hodnoty DIFE se nijak nezpracovávají,
- podpora standardních VIF jednotek: Wh, J, m^3, kg, sec., min., hours, days, W, J/h, m^3/h, m^3/min., m^3/s, kg/h, °C, K, bar, DateTime,
- podpora rozšiřujících VIFE jednotek pro VIF 0x7B a 0x7D: MWh, GJ, m^3, tons, MW, GJ/h, °C, W, Volts, Ampers,
- podpora textového vyjádření jednotky pro VIF 0x7C,
- uvedení znaku otazníku v případě, že kombinace VIF a VIFE nespadá mezi podporované,
- kontrola rámce: kontrola hlavičky, kontrola správnosti přijaté adresy, ověření CRC, kontrola kompletnosti dat rámce, signalizace nedekódovatelných typů hodnot či dat – parametr **`Errors`**.

Funkční blok **`fb_MBusDevice`** pracuje ve svém těle s instancí funkčního bloku **`fb_MBusDecode`**. Pro korektní funkčnost proto musí být definice funkčního bloku **`fb_MBusDecode`** přítomná v projektu DetStudia v okně „Projekt“ v uzlu „Funkční bloky“.

Kompletní popis funkčního bloku **`fb_MBusDecode`** je uveden v záložce „Dokumentace“ definice funkčního bloku v DetStudiu.

2.2.2 Funkční blok fb_MBusDevice

Základní charakteristiky funkčního bloku:
- využívá se v procesu s periodou 100 ms,
- spuštění komunikace s měřičem při náběžné hraně vstupní vlastnosti **`SyncMarkIn`**,
- střídání definovaných odchozích komunikačních rámců,
- dvojí opakování odchozích rámců v případě nepřijetí korektní odpovědi,
- předání přijatých běžných dat funkčnímu bloku **`fb_MBusDecode`**,
- uložení přijatého rámce běžných dat pro případné doplňující uživatelské zpracování,
- prezentace dekódovaných hodnot, jednotek a chyb – parametry **`Values`**, **`ValuesStrings`**, **`UnitsStrings`**, **`DeviceInfo`** a **`Errors`**,
- existence prostředků pro sdílení jednoho komunikačního rozhraní pro více měřičů definovaných instancemi funkčních bloků **`fb_MBusDevice`**.

Adresace měřiče
V běžném použití platí, že jedna instance funkčního bloku odpovídá právě jednomu měřiči. Tento měřič je dán svou primární adresou. V případě, že se zadá záporná hodnota adresy, je daný blok vyřazen z komunikačního celku.

Vícerámcové měřiče
V některých případech může měřič poskytovat více běžných dat, než se vejde do jednoho rámce běžných dat. Např. v případě, že by měřič poskytoval 600 bajtů běžných dat, je potřeba komunikaci s měřičem řešit zasláním rámců s požadavky v následujícím pořadí:
- inicializační krátký rámec,
- požadavek na běžná data s hlavním bajtem rámce 0x5B,
- požadavek na běžná data s hlavním bajtem rámce 0x7B,
- požadavek na běžná data s hlavním bajtem rámce 0x5B.

Vlastnost **`CommType`** funkčního bloku definuje, jestli daná instance má posílat inicializační rámec či nikoliv a jaký bajt má být použit jako hlavní bajt rámce. Možné hodnoty:
- 0 – posílá se inicializační rámec a dotaz na hodnoty s hlavním bajtem 0x5B,
- 1 – posílá se inicializační rámec a dotaz na hodnoty s hlavním bajtem 0x7B,
- 2 – posílá se pouze dotaz na hodnoty s hlavním bajtem 0x5B,
- 3 – posílá se pouze dotaz na hodnoty s hlavním bajtem 0x7B.

Uvedený vícerámcový měřič tedy budou reprezentovat tři instance funkčního bloku, přičemž hodnota adresy bude vždy stejná a hodnoty vlastností **`CommType`** budou nadefinovány pro jednotlivé instance bloků na 0, 3 a 2.

Paměťové nároky
Každé jedno použití funkčního bloku zabere v paměti systému 4,4 kB v paměti RAM a 1,9 kB v paměti FLASH. U řídicích systémů s 1 MB pamětí RAM z toho plyne omezení na cca 230 bloků typu **fb_MBusDevice**.

V případě, že je požadováno použití více bloků, lze toto vyřešit snížením počtu řádků výstupních maticových parametrů s dekódovanými hodnotami:

- Funkční blok **fb_MBusDevice**:
  - parametr **Values** s výchozí dimenzí 40×1,
  - parametr **UnitsStrings** s výchozí dimenzí 40×20,
  - parametr **ValuesStrings** s výchozí dimenzí 10×20,

Např. pokud by se u prvních dvou výše zmíněných matic s výchozí dimenzí 40 řádků zmenšil počet řádků na 20 a u třetí matice s výchozí dimenzí 10 řádků se zmenšil na 2 řádky, klesne náročnost na RAM na 3,1 kB. Tímto se u řídicího systému s 1 MB paměti RAM navýší maximální počet bloků typu **fb_MBusDevice** na cca 320.

Kompletní popis funkčního bloku **fb_MBusDevice** je uveden v záložce „Dokumentace“ definice funkčního bloku v DetStudiu.

2.2.3 Použití funkčního bloku fb_MBusDevice

Funkční blok poskytuje rámce k odeslání, definuje okamžiky pro jejich odeslání a okamžiky pro vyčtení přijatých dat, při kterých se má bloku předat počet přijatých bajtů v rámci a rámec samotný.

Toto je reprezentováno parametry bloku:

- **FrameReceived** – přijatý rámec,
- **BytesReceived** – počet dat přijatého rámce,
- **FrameSend** – rámec k odeslání,
- **Command** – definice komunikační akce.

Na základě hodnoty parametru **Command** se očekává provedení některé z následujících akcí:

- 0, 1 – bez akce,
- 2 – posílání 5 znaků rámce připraveného ve vlastnosti **FrameSend** a nastavení hodnoty **Command** na 1,
- 3 – uložení přijatého rámce do vlastnosti **FrameReceived**, uložení počtu přijatých bajtů do vlastnosti **BytesReceived** a nastavení hodnoty **Command** na 1.

Funkční blok neprovádí žádnou komunikaci přes některé rozhraní. Pro poslání a příjem rámců se využívá komunikační objekt **UserComBinary**, resp. jeho metoda **Send()** a vlastností **BufferWrite**, **BufferRead** a **BytesReceived**. Je doporučeno, aby se při odesílání rámce promazal příchozí buffer a při příjmu rámce promazal odchozí buffer. Jestli data budou komunikována sériovým rozhraním nebo rozhraním Ethernet záleží na nastavení vlastností komunikačního objektu **UserComBinary**.

Více informací o komunikačním objektu **UserComBinary** lze nalézt v nápovědě vývojového prostředí **DetStudio**.

Použití sdílených proměnných
K jednomu převodníku je běžně připojeno více měřičů, které jsou reprezentovány instancemi funkčního bloku **fb_MBusDevice**. Pouze jediná instance funkčního bloku může v jeden okamžik přistupovat k jednomu komunikačnímu objektu **UserComBinary** a vyčítat data z jednoho měřiče. Další instance funkčního bloku **fb_MBusDevice** může přistupovat ke stejnému komunikačnímu objektu **UserComBinary** až v době, kdy předchozí instance funkčního bloku **fb_MBusDevice** dokončí svou činnost. Z toho vyplývá nutnost sdílení proměnných zadaných za parametry funkčního bloku **fb_MBusDevice**. Jedná se o čtyři proměnné typů a rozměrů:

- ARRAY OF INT [1×262] – používaná v parametrech **FrameReceived**,
- INT – používaná v parametrech **BytesReceived**,
- ARRAY OF INT [1×5] – používaná v parametrech **FrameSend**,
- INT – používaná v parametrech **Command**.

Ukázka použití pro tři instance funkčního bloku **fb_MBusDevice** definující tři měřiče:

```
fb_MBusDevice1(
     FrameReceived = MBusFrameRec,
     BytesReceived = MBusBytesRec,
     FrameSend = MBusFrameSend,
     Command = MBusCommand
);

fb_MBusDevice2(
     FrameReceived = MBusFrameRec,
     BytesReceived = MBusBytesRec,
     FrameSend = MBusFrameSend,
     Command = MBusCommand
);

fb_MBusDevice3(
     FrameReceived = MBusFrameRec,
     BytesReceived = MBusBytesRec,
     FrameSend = MBusFrameSend,
     Command = MBusCommand
);
```

V této ukázce se jedná o proměnné pojmenované **MBusFrameRec**, **MBusBytesRec**, **MBusFrameSend** a **MBusCommand**.

Algoritmus práce s objektem UserComBinary

Jakmile jsou výše uvedené parametry funkčních bloků **fb_MBusDevice** navázány na sdílené proměnné, je možné vytvořit jednoduchý algoritmus práce s komunikačním objektem **UserComBinary** na základě hodnoty sdílené proměnné nadefinované za vlastnosti **Command**.

Výsledný kód:

```
CASE MBusCommand OF
     2:
          FOR i = 0 TO 4 DO
               UserComBinary1.BufferWrite[0,i] = Int_To_Byte(MBusFrameSend[0,i]);
          ENDFOR;
          UserComBinary1.Send(5);
          UserComBinary1.ClearReadBuffer();
          MBusCommand = 1;
     3:
          FOR i = 0 TO UserComBinary1.BytesReceived DO
               MBusFrameRec[0,i] = UserComBinary1.BufferRead[0,i];
          ENDFOR;
          MBusBytesRec = UserComBinary1.BytesReceived;
          UserComBinary1.ClearWriteBuffer();
          MBusCommand = 1;
ENDCASE;
```

Vlastnost **Command** je funkčními bloky využívána nejenom k zápisu požadované akce, ale zároveň se i načítá. Nemůže tak dojít ke kolizi, že by vícero instancí v ten stejný okamžik začalo vysílat či akceptovat příchozí data.

3 Ukázková aplikace

Součástí aplikační poznámky je ukázková aplikace pro komunikaci s měřiči. V této aplikaci jsou nadefinovány dva komunikační objekty **`UserComBinary`**. U každého je nadefinována jiná IP adresa reprezentující jiný komunikační převodník. Každý komunikační objekt **`UserComBinary`** využívá tří instancí funkčního bloku **`fb_MBusDevice`**.

Jednotlivé komunikace jsou pro přehlednost rozděleny do podprogramů volaných z periodického procesu s periodou 100 ms. Komunikace přes komunikační objekt se povoluje nastavením lokální proměnné **`AutomatRun`** na kladnou nenulovou hodnotu. Samotné vyčítání je startováno synchronizačním bitem od lokálního bloku **`SyncMark1`**. Ve výchozím nastavení je nastavena komunikace každou hodinu. Četnost vyčítání lze měnit změnou hodnot vlastností bloku **`SyncMark1`**.

Více informací o bloku **`SyncMark`** lze nalézt v nápovědě vývojového prostředí **DetStudio**.

Ukázková aplikace je v příloze uložena pod názvem „mbus_p1_cz_xxx.dsox“. Tento projekt je vytvořen pro řídicí systém **AMR-OP87 RevA**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, který disponuje rozhraním Ethernet, pomocí menu DetStudia „Nástroje/Změnit typ Stanice…“.

Pozor
Ukázková aplikace je určená pro použití v DetStudiu 2.2.43 a novějším!

4 Dodatek A

4.1 Komunikace prostřednictvím protokolů MODBUS

V případě, kdy z libovolného důvodu nelze použít komunikaci s převodníkem prostřednictvím komunikačního objektu **`UserComBinary`**, lze pro komunikaci s převodníkem **DM-MBUS64** využít komunikace protokoly MODBUS. A to buď MODBUS RTU prostřednictvím některého ze sériových rozhraní nebo MODBUS TCP prostřednictvím rozhraní Ethernet.

4.1.1 HW parametrizace

Parametrizace převodníku **DM-MBUS64** se provádí typicky prostřednictvím webových stránek, když je převodník v tzv. servisním režimu. Převodník musí mít nastaven mód komunikace MODBUS na požadovaném rozhraní.

4.1.2 SW parametrizace

Pro komunikaci s převodníkem se používá holding registrů na adresách:

- 100 – nastavení akce převodníku,
- 101 až 361 – odchozí rámec do sítě M-Bus,
- 400 – zjišťování stavu příchozího rámce,
- 402 – počet příchozích znaků ze sítě M-Bus,
- 403 až 663 – příchozí rámec ze sítě M-Bus.

Přesný popis akcí převodníku a komunikačních registrů je uveden v návodu na obsluhu převodníku **DM-MBUS64**.

Pro inicializaci měřiče nebo požadavku na běžná data se používá krátkého rámce s délkou 5 bajtů. Jedna z funkcí převodníku **DM-MBUS64** je automatické počítání hodnoty kontrolního součtu a připojování koncového znaku 0x16. Do převodníku tedy stačí zapisovat pouze 3 bajty odchozího rámce a komunikovat registry 101 až 103. Vzhledem k možným budoucím použitím je v ukázkové aplikaci zvolena definice komunikace s 20 registry v rozsahu 101 až 121.

Pro inicializaci vyčítání, hlídání chyb a dekódování hodnot lze použít dříve popsaný funkční blok **`fb_MBusDevice`**. Pro jeho použití se veškerá komunikace s převodníkem nastaví jako tzv. událostní, tedy zápis do registrů bude probíhat po zápisu do příslušné proměnné a čtení registrů bude proveden voláním metody **`Refresh()`** dané proměnné. Z tohoto důvodu se nastaví všechny priority komunikace registrů na **`None`**.

Protokol MODBUS podporuje vyčtení maximálně 125 holding registrů. Příchozí rámec ze sítě M-Bus však může mít až 261 znaků. Z tohoto důvodu se musí registry 403 až 663 rozdělit do tří komunikovaných skupin. V ukázkové aplikaci je zvoleno dělení na 90, 90 a 81 registrů. Následný obrázek zobrazuje seznam používaných definic MODBUS registrů. Toto platí pro komunikaci protokolem MODBUS RTU i protokolem MODBUS TCP.

ModbusMaster1.ModbusDevice1

Jméno	Typ	Řádků	Sloupců	Registr	Druh	Priorita	MB Funkce čtení	MB Funkce zápis	Komentář
MBus_Flags	WORD			100	Holding register	None	Function_03	Function_16	
MBus_OutTlg	WORD[,]	1	20	101	Holding register	None	Function_03	Function_16	
MBus_ComStat	WORD			400	Holding register	None	Function_03	Function_16	
MBus_RecChars	WORD			402	Holding register	None	Function_03	Function_16	
MBus_InTlg0	WORD[,]	1	90	403	Holding register	None	Function_03	Function_16	
MBus_InTlg1	WORD[,]	1	90	493	Holding register	None	Function_03	Function_16	
MBus_InTlg2	WORD[,]	1	81	583	Holding register	None	Function_03	Function_16	

Obr. 4 – Seznam používaných definic MODBUS registrů

Více informací o komunikačních objektech **`ModbusMaster`**, **`ModbusDevice`**, **`ModbusMasterTCP`** a **`ModbusDeviceTCP`** lze nalézt v nápovědě vývojového prostředí **DetStudio**.

Jakmile je nadefinován příslušný komunikační objekt **`ModbusDevice`** nebo **`ModbusDeviceTCP`**, lze přistoupit k úpravě komunikačního algoritmu. Pro komunikaci prostřednictvím komunikačního protokolu MODBUS s převodníkem **DM-MBUS64** je doporučován následující postup:

1. uložení odchozího rámce do převodníku a vynulování počtu přijatých znaků,
2. pauza pro provedení komunikace,
3. zapsání do registru akce převodníku hodnotu 1 pro vyslání rámce do sítě M-Bus,
4. pauza pro provedení komunikace,
5. vyčtení hodnoty registru stavu příchozího rámce,
6. pauza pro provedení komunikace,
7. zjištění stavu příchozího rámce; možné varianty:
   a) příchozí rámec zatím nepřišel, proto opakovat od bodu 4.,
   b) příchozí rámec přišel, proto provést vyčtení hodnot rámce z převodníku, počtu příchozích bajtů a vynulovat hodnotu stavu příchozího rámce a pokračovat v bodu 8.,
   c) převodník hlásí chybu, proto se algoritmus komunikace předčasně ukončí a vynuluje se hodnota stavu příchozího rámce,
8. pauza pro provedení komunikace,
9. zpracování hodnot a promazání bufferu příchozího rámce ze sítě M-Bus převodníku zapsáním do registru akce převodníku dekadickou hodnotu 203.

Uvedený postup reprezentuje kód:

```
CASE Modbus_step OF
     1:
          tmpOutTlg[0,0] = Int_To_Byte(MBusFrameSend[0,0]);
          tmpOutTlg[0,1] = Int_To_Byte(MBusFrameSend[0,1]);
          tmpOutTlg[0,2] = Int_To_Byte(MBusFrameSend[0,2]);
          ModbusMaster1.ModbusDevice1.MBus_OutTlg = tmpOutTlg;
          MBusBytesRec = 0;
          Modbus_step = 2;
     2:
          Modbus_step = 3;
     3:
          ModbusMaster1.ModbusDevice1.MBus_Flags = 1;
          Modbus_time = 0;
          Modbus_step = 4;
     4:
          Modbus_time = Modbus_time + 1;
          Modbus_step = Modbus_time >= 2 ? 5 : 4;
```

```
      5:
            ModbusMaster1.ModbusDevice1.MBus_ComStat.Refresh();
            Modbus_time = 0;
            Modbus_step = 6;
      6:
            Modbus_time = Modbus_time + 1;
            Modbus_step = Modbus_time >= 2 ? 7 : 6;
      7:
            IF ModbusMaster1.ModbusDevice1.MBus_ComStat.0 THEN
                  Modbus_time = 0;
                  Modbus_step = 4;
            ELSE
                  IF ModbusMaster1.ModbusDevice1.MBus_ComStat.1 THEN
                        ModbusMaster1.ModbusDevice1.MBus_RecChars.Refresh();
                        ModbusMaster1.ModbusDevice1.MBus_InTlg0.Refresh();
                        ModbusMaster1.ModbusDevice1.MBus_InTlg1.Refresh();
                        ModbusMaster1.ModbusDevice1.MBus_InTlg2.Refresh();
                        ModbusMaster1.ModbusDevice1.MBus_ComStat = 0;
                        Modbus_time = 0;
                        Modbus_step = 8;
                  ELSE
                        ModbusMaster1.ModbusDevice1.MBus_ComStat = 0;
                        Modbus_step = 0;
                  ENDIF;
            ENDIF;
      8:
            Modbus_time = Modbus_time + 1;
            Modbus_step = Modbus_time >= 6 ? 9 : 8;
      9:
            MBusBytesRec = ModbusMaster1.ModbusDevice1.MBus_RecChars;
            j = 0;
            FOR i = 0 TO 89 DO
                  MBusFrameRec[0,j] = ModbusMaster1.ModbusDevice1.MBus_InTlg0[0,i];
                  j = j + 1;
            ENDFOR;
            FOR i = 0 TO 89 DO
                  MBusFrameRec[0,j] = ModbusMaster1.ModbusDevice1.MBus_InTlg1[0,i];
                  j = j + 1;
            ENDFOR;
            FOR i = 0 TO 80 DO
                  MBusFrameRec[0,j] = ModbusMaster1.ModbusDevice1.MBus_InTlg2[0,i];
                  j = j + 1;
            ENDFOR;
            ModbusMaster1.ModbusDevice1.MBus_Flags = 203;
            Modbus_step = 0;
ENDCASE;
```

Zopakujme, že funkční blok **fb_MBusDevice** pomocí vlastnosti **Command** určuje okamžiky pro odeslání zpráv a pro převzetí vyčtených hodnot. Algoritmus pro hodnotu 2 (odeslání dat) se zjednoduší na odstartování výše uvedeného algoritmu nastavením proměnné **Modbus_step** na 1. Pro hodnotu 3 (předání dat) se algoritmus zcela odstraňuje, neboť data se předávají v MODBUS algoritmu vyčítání v kroku 9. Výsledný kód algoritmu zpracování řídicích hodnot:

```
CASE MBusCommand OF
      2:
            Modbus_step = 1;
            MBusCommand = 1;
      3:
            MBusCommand = 1;
endcase;
```

4.1.3 Ukázková aplikace

Součástí aplikační poznámky je ukázková aplikace pro komunikaci s měřiči prostřednictvím protokolů MODBUS RTU i MODBUS TCP. Pro každou komunikaci se využívají tři instance funkčního bloku **`fb_MBusDevice`**.

Jednotlivé komunikace jsou pro přehlednost rozděleny do podprogramů volaných z periodického procesu s periodou 100 ms. Komunikace přes komunikační objekt se povoluje nastavením lokální proměnné **`AutomatRun`** na kladnou nenulovou hodnotu. Samotné vyčítání je startováno synchronizačním bitem od lokálního bloku **`SyncMark1`**. Ve výchozím nastavení je nastavena komunikace každou hodinu. Četnost vyčítání lze měnit změnou hodnot vlastností bloku **`SyncMark1`**.

Více informací o bloku **`SyncMark`** lze nalézt v nápovědě vývojového prostředí **DetStudio**.

Ukázková aplikace je v příloze uložena pod názvem „mbus_p2_cz_xxx.dsox“. Tento projekt je vytvořen pro řídicí systém **AMR-OP87 RevA**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, který disponuje rozhraním Ethernet, pomocí menu DetStudia „Nástroje/Změnit typ Stanice…“.

Pozor
Ukázková aplikace je určená pro použití v DetStudiu 2.2.43 a novějším!

5 Technická podpora

Veškeré informace ohledně komunikace v síti M-Bus v řídicích systémech firmy AMiT, Vám poskytne oddělení technické podpory firmy AMiT. Technickou podporu můžete kontaktovat nejlépe prostřednictvím e-mailu na adrese **support@amit.cz**.

6 Upozornění

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

Tento dokument může být kopírován a rozšiřován za následujících podmínek:

1. Celý text musí být kopírován bez úprav a se zahrnutím všech stránek.
2. Všechny kopie musí obsahovat označení autorského práva společnosti AMiT, spol. s r. o. a veškerá další upozornění v dokumentu uvedená.
3. Tento dokument nesmí být distribuován za účelem dosažení zisku.

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