Přehled protokolu MODBUS

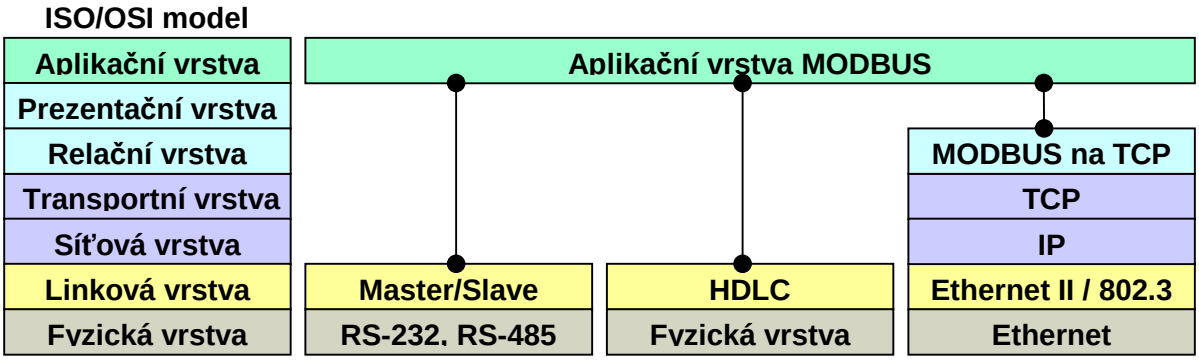
Ing. Andrea Ronešová

květen 2005

Přehled protokolu MODBUS

1 Úvod

MODBUS je komunikační protokol na úrovni aplikační vrstvy ISO/OSI modelu, umožňující komunikaci typu klient-server mezi zařízeními na různých typech sítí a sběrnic. Vytvořen v roce 1979 firmou MODICON. V současné době je podporována celá řada komunikačních médií např. sériové linky typu RS-232, RS-422 a RS-485, optické a rádiové sítě nebo síť Ethernet s využitím protokolu TCP/IP. Komunikace probíhá metodou požadavek-odpověď a požadovaná funkce je specifikována pomocí kódu funkce jež je součástí požadavku.

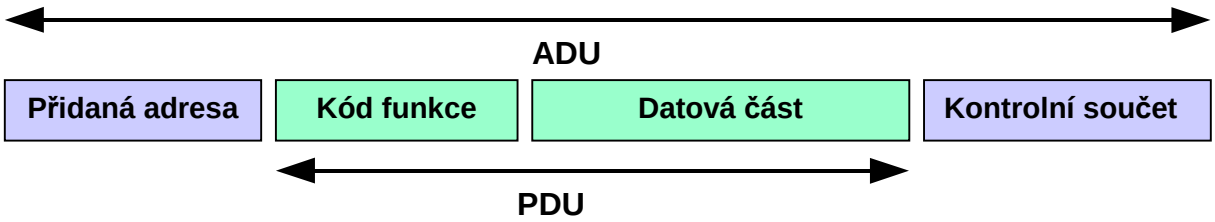


Obr. 1.1: Příklady implementace

2 Obecný popis

2.1 Popis protokolu

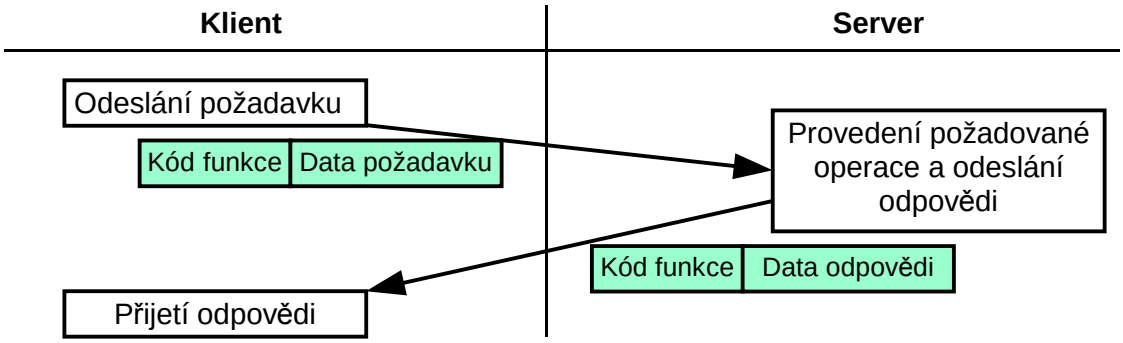
Protokol MODBUS definuje strukturu zprávy na úrovni protokolu (PDU – Protocol Data Unit) nezávisle na typu komunikační vrstvy. V závislosti na typu sítě, na které je protokol použit, je PDU rozšířena o další části a tvoří tak zprávu na aplikační úrovni (ADU – Application Data Unit).



Obr. 2.1: Základní tvar MODBUS zprávy

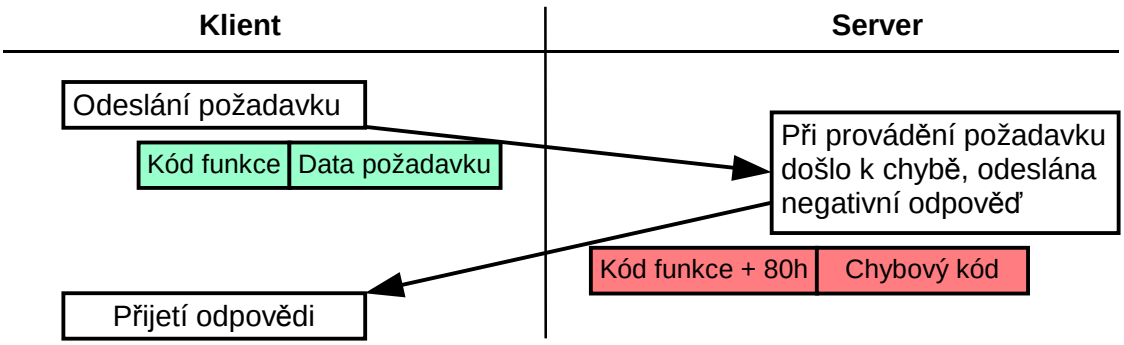
Kód funkce udává serveru jaký druh operace má provést. Rozsah kódů je 1 až 255, přičemž kódy 128 až 255 jsou vyhrazeny pro oznámení záporné odpovědi (chyby). Některé kódy funkcí obsahují i kód podfunkce upřesňující blíže požadovanou operaci. Obsah datové části zprávy poslané klientem slouží serveru k uskutečnění operace určené kódem funkce. Obsahem může být například adresa a počet vstupů, které má server přečíst nebo hodnota registrů, které má server zapsat. U některých funkcí nejsou pro provedení operace zapotřebí další data a v tom případě může datová část ve zprávě úplně chybět.

Pokud při provádění požadované operace nedojde k chybě (Obr. 2.2), odpoví server zprávou, která v poli Kód funkce obsahuje kód provedené (požadované) funkce jako indikaci úspěšného vykonání požadavku. V datové části odpovědi předá server klientovi požadovaná data (pokud jsou nějaká).



Obr. 2.2: MODBUS transakce s bezchybným provedením požadavku

Pokud při vykonávání požadované operace dojde k chybě (Obr. 2.3), je v poli Kód funkce vrácen kód požadované funkce s nastaveným nejvyšším bitem indikujícím neúspěch (exception response). V datové části je vrácen chybový kód (exception code) upřesňující důvod neúspěchu.



Obr. 2.3: MODBUS transakce s chybou při provádění požadavku

Pozn.: Z důvodu možné ztráty požadavku nebo odpovědi, je žádoucí implementovat na straně klienta časový limit pro přijetí odpovědi, aby klient nečekal donekonečna na odpověď, která nemusí přijít.

Maximální velikost PDU je zděděna z první implementace MODBUSu na sériové lince RS-485, kde byla maximální velikost ADU 256 bytů. Tomu odpovídá maximální velikost PDU 253 bytů.

Max. velikost PDU** na sériové lince = 256 – adresa serveru (1 byte) – kontrolní součet CRC (2 byty) = **253 bytů.

Odtud:

*Velikost **ADU na RS-485** = 253 bytů PDU + adresa(1 byte) + CRC (2 byty) = **256 bytů***
*Velikost **ADU na TCP/IP** = 253 bytů PDU + MBAP = **260 bytů***

Protokol MODBUS definuje 3 základní typy zpráv (PDU):

- Požadavek (Request PDU)
  - 1 byte Kód funkce
  - n bytů Datová část požadavku – adresa, proměnné, počet proměnných…
- Odpověď (Response PDU)
  - 1 byte Kód funkce (kopie z požadavku)
  - m bytů Datová část odpovědi – přečtené vstupy, stav zařízení …
- Záporná odpověď (Exception Response PDU)
  - 1 byte Kód funkce + 80h (indikace neúspěchu)
  - 1 byte Chybový kód (identifikace chyby)

2.2 Kódování dat

MODBUS používá tzv. „Big-endian“ reprezentaci dat. To znamená, že při posílání datových položek delších než 1 byte je jako první posílán nejvyšší byte a jako poslední nejnižší byte.

Např.: 16-bitová položka s hodnotou 1234h - nejprve je poslán byte 12h, poté byte 34h

2.3 Datový model

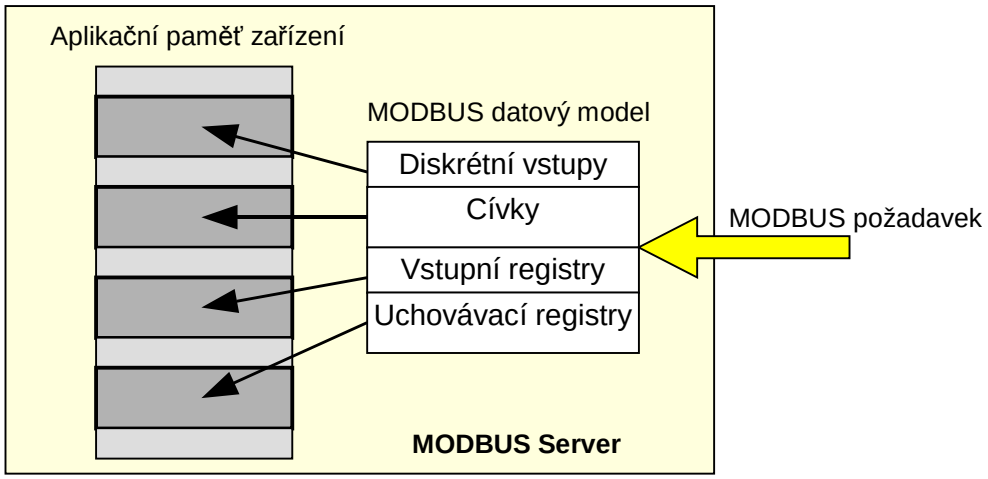
Datový model MODBUSu je založen na sadě tabulek, s charakteristickým významem. Definovány jsou čtyři základní tabulky:

Tabulka 2.1.: Datový model MODBUS

Tabulka	Typ položky	Přístup	Popis	Adresa (MODICON)
Diskrétní vstupy *(Discrete Inputs)*	1-bit	Pouze čtení	Data poskytovaná I/O systémem	10000÷19999
Cívky *(Coils)*	1-bit	Čtení/zápis	Data modifikovatelná aplikačním programem	0÷9999
Vstupní registry *(Input Registers)*	16-bitové slovo	Pouze čtení	Data poskytovaná I/O systémem	30000÷39999
Uchovávací registry *(Holding Registers)*	16-bitové slovo	Čtení/zápis	Data modifikovatelná aplikačním programem	40000÷49999

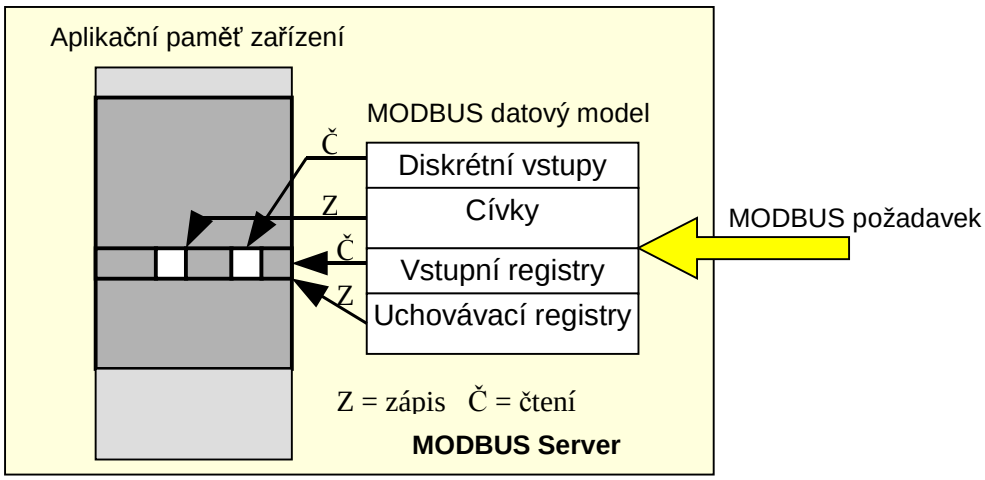
Mapování tabulek do adresního prostoru je závislé na konkrétním zařízení. Každá z tabulek může mít vlastní adresní prostor nebo se mohou částečně či úplně překrývat. Každá z tabulek může mít dle protokolu až 65536 položek. Z důvodu zpětné kompatibility bývá ale adresní prostor rozdělen na bloky o velikosti 10000 položek tak jak je uvedeno ve sloupci Adresa tabulky 2.1. Přístupná je každá položka jednotlivě nebo lze přistupovat ke skupině položek najednou. Velikost skupiny položek je omezena maximální velikostí datové části zprávy.

Na obrázcích 2.4 a 2.5 jsou znázorněny dva možné způsoby organizace dat v zařízení. Obrázek 2.4 znázorňuje zařízení, u nějž není žádný vztah mezi položkami jednotlivých tabulek a každá tabulky má tedy svůj oddělený prostor v aplikační paměti zařízení. Do jednotlivých tabulek lze přistupovat prostřednictvím příslušné funkce MODBUSu.



Obr. 2.4: Datový model MODBUS se čtyřmi oddělenými bloky

Obrázek 2.5 znázorňuje zařízení, které má pouze jeden datový blok. K položkám lze přistupovat prostřednictvím různých funkcí MODBUSu v závislosti na tom, co je pro aplikaci v daném okamžiku výhodné.



Obr. 2.5: Datový model MODBUS s jediným blokem

2.4 Adresovací model

Protokol MODBUS přesně definuje adresovací pravidla ve zprávách (PDU):

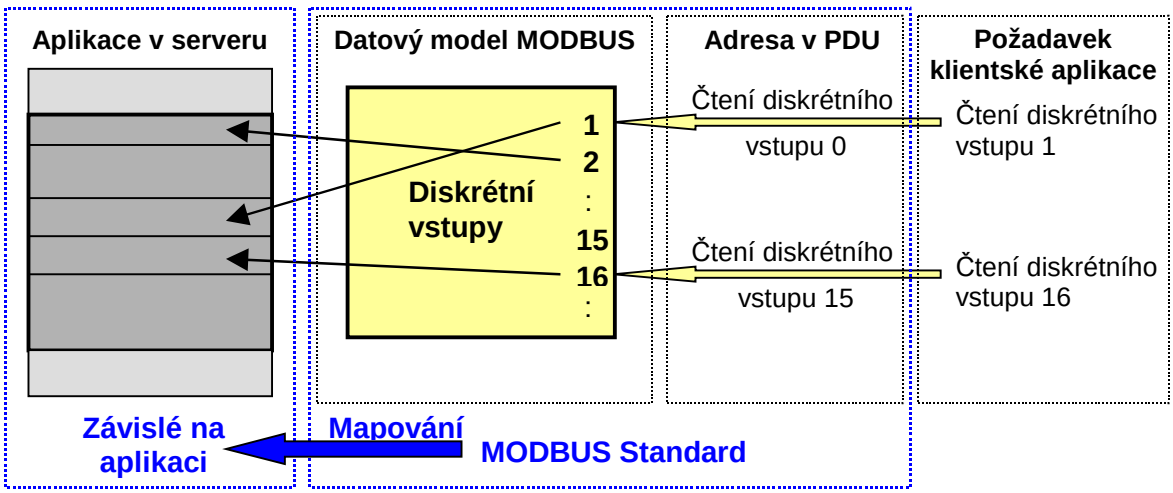
V MODBUS zprávách (PDU) jsou datové položky adresovány od 0 do 65535

Dále je definováno adresování v rámci datového modelu složeného ze 4 datových bloků (tabulek):

V MODBUS datovém modelu jsou položky v datových blocích číslovány od 1 do n

Mapování položek MODBUS datového modelu do aplikace v serveru je zcela v režii výrobce.

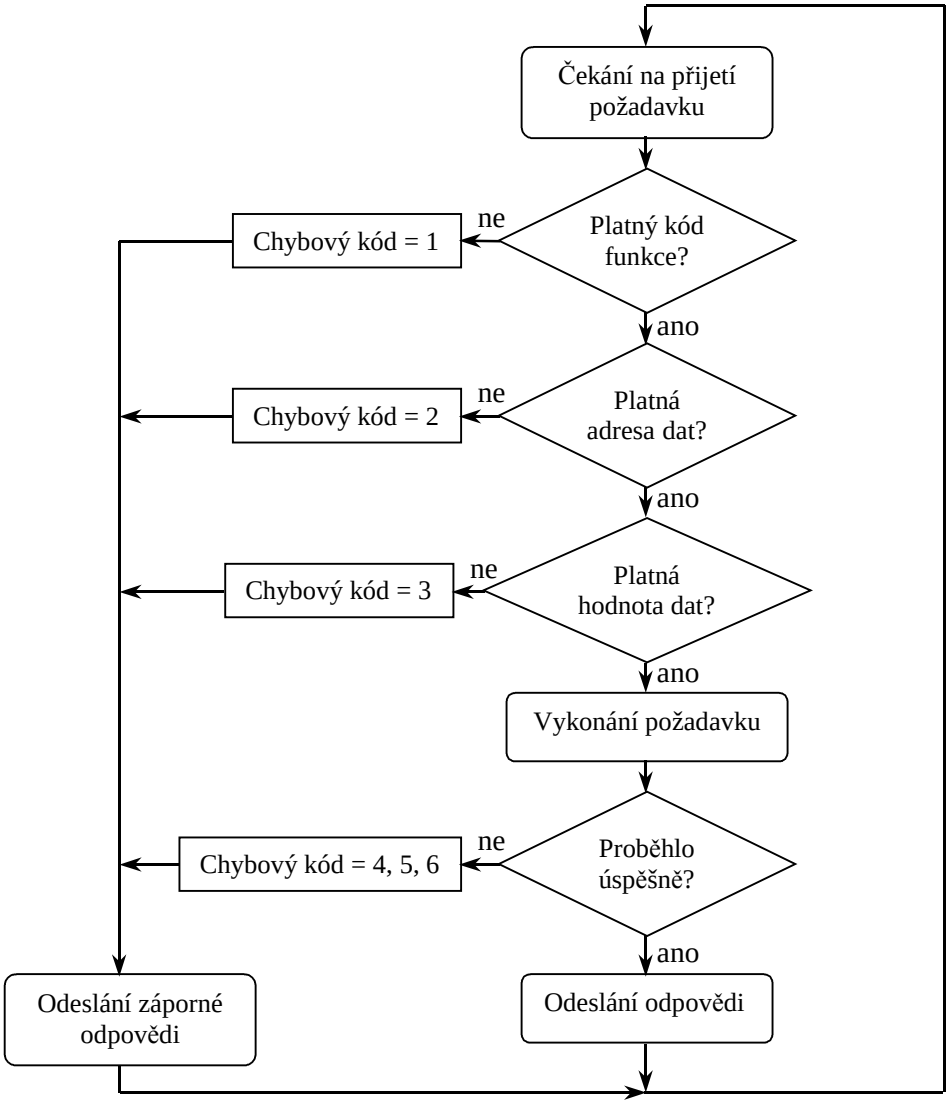
Na obrázku 2.6 je znázorněn příklad adresování od požadavku klienta až po aplikaci v serveru. Z obrázku je zřejmé, že datová položka X v datovém modelu je v PDU adresována jako položka X-1.



Obr. 2.6: Příklad adresování dle MODBUS

2.5 Definice MODBUS transakce

Stavový diagram na obrázku 2.7 popisuje obecný postup zpracování MODBUS požadavku na straně serveru.



Obr. 2.7: Obecný postup zpracování MODBUS požadavku na straně serveru

Jakmile server zpracuje požadavek (ať úspěšně či neúspěšně), sestaví odpověď a odešle ji klientovi. V závislosti na výsledku zpracování požadavku je vytvořena jedna ze dvou možných odpovědí:

- Pozitivní odpověď (Response):
  - kód funkce v odpovědi = kód funkce v požadavku
- Negativní odpověď (Exception Response) (blíže viz. odstavec 5):
  - kód funkce v odpovědi = kód funkce v požadavku + 80h
  - je vrácen kód chyby udávající důvod neúspěchu

3 Kategorie kódů funkcí

MODBUS protokol definuje tři skupiny kódů funkcí:

Veřejné kódy funkcí

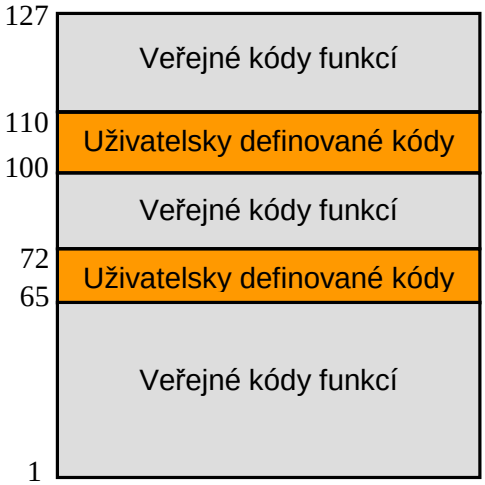
- jasně definované
- je garantována unikátnost
- schvalovány společností MODBUS-IDA.org
- veřejně zdokumentované
- je k nim dostupný test shody
- zahrnují veřejné přiřazené kódy funkcí i nepřiřazené kódy rezervované pro budoucí použití

Uživatelsky definované kódy funkcí

- dva rozsahy uživatelsky definovaných kódů funkcí: 65 ÷ 72 a 100 ÷ 110
- umožňují uživateli implementovat funkci, která není definována touto specifikací
- není garantována unikátnost kódů
- lze je po projednání přesunout do veřejných kódů

Rezervované kódy funkcí

- kódy funkcí, které jsou v současnosti používány některými firmami a které nejsou dostupné pro veřejné použití



Obr. 3.1: Kategorie funkčních kódů

3.1 Definice funkčních kódů

				Kódy funkcí			
				Kód	Podfunkce	hex	
Přístup k datům	**Bitový přístup**	Fyzické diskrétní vstupy	Čti diskrétní vstupy	**02**		**02**	
		Interní bity nebo fyzické cívky	Čti cívky	**01**		**01**	
			Zapiš jednu cívku	**05**		**05**	
			Zapiš více cívek	**15**		**0F**	
	16-bitový přístup	Fyzické vstupní registry	Čti vstupní registr	**04**		**04**	
		Interní registry nebo fyzické výstupní registry	Čti uchovávací registry	**03**		**03**	
			Zapiš jeden registr	**06**		**06**	
			Zapiš více registrů	**16**		**10**	
			Čti/zapiš více registrů	**23**		**17**	
			Zapiš registr s maskováním	**22**		**16**	
			Čti FIFO frontu	**24**		**18**	
	Přístup k záznamům v souborech		Čti záznam ze souboru	**20**	**6**	**14**	
			Zapiš záznam do souboru	**21**	**6**	**15**	
Diagnostika			Čti stav	**07**		**07**	
			Diagnostika	**08**	**00-18, 20**	**08**	
			Čti čítač kom. událostí	**11**		**0B**	
			Čti záznam kom. událostí	**12**		**0C**	
			Sděl identifikaci	**17**		**11**	
			Čti identifikaci zařízení	**43**	**14**	**2B**	
Ostatní			Zapouzdřený přenos	**43**	**13, 14**	**2B**	
			CANOpen základní odkaz	**43**	**13**	**2B**	

4 Popis kódů funkcí

4.1 01 (0x01) Čti cívky (Read Coils)

Tato funkce slouží ke čtení stavu 1 až 2000 cívek. V požadavku je specifikována adresa první cívky a počet cívek. V odpovědi je v jednom bytu přenášen stav celkem 8 cívek. Nejnižší bit prvního bytu je stav první (adresované) cívky.

Požadavek

Kód funkce	1 byte	**0x01**
Počáteční adresa	2 byty	0x0000 až 0xFFFF
Počet cívek	2 byte	1 až 2000 (0x7D0)

Odpověď

Kód funkce	1 byte	**0x01**
Počet bytů	1 byte	**N**
Stavy cívek	**N** bytů	

N = počet cívek / 8, je-li zbytek po dělení nenulový, **N** = **N** + 1

Chyba

Kód funkce	1 byte	**0x81**
Chybový kód	1 byty	01, 02, 03 nebo 04

4.2 02 (0x02) Čti diskrétní vstupy (Read Discrete Inputs)

Tato funkce slouží ke čtení stavu 1 až 2000 diskrétních vstupů. V požadavku je specifikována adresa prvního vstupu a počet vstupů. V odpovědi je v jednom bytu přenášen stav celkem 8 vstupů. Nejnižší bit prvního bytu je stav prvního (adresovaného) vstupu.

Požadavek

Kód funkce	1 byte	**0x02**
Počáteční adresa	2 byty	0x0000 až 0xFFFF
Počet vstupů	2 byte	1 až 2000 (0x7D0)

Odpověď

Kód funkce	1 byte	**0x02**
Počet bytů	1 byte	**N**
Stavy vstupů	**N** bytů	

N = počet vstupů / 8, je-li zbytek po dělení nenulový, **N** = **N** + 1

Chyba

Kód funkce	1 byte	**0x82**
Chybový kód	1 byty	01, 02, 03 nebo 04

4.3 03 (0x03) Čti uchovávací registry (Read Holding Registers)

Tato funkce slouží ke čtení obsahu souvislého bloku až 125 uchovávacích registrů. V požadavku je specifikována adresa prvního registru a počet registrů. V odpovědi odpovídá každému registru dvojice bytů.

Požadavek

Kód funkce	1 byte	**0x03**
Počáteční adresa	2 byty	0x0000 až 0xFFFF
Počet registrů	2 byte	1 až 125 (0x7D)

Odpověď

Kód funkce	1 byte	**0x03**
Počet bytů	1 byte	2*__N__
Hodnoty registrů	2*__N__ bytů	

N = počet registrů

Chyba

Kód funkce	1 byte	**0x83**
Chybový kód	1 byty	01, 02, 03 nebo 04

4.4 04 (0x04) Čti vstupní registry (Read Input Registers)

Tato funkce slouží ke čtení obsahu souvislého bloku až 125 vstupních registrů. V požadavku je specifikována adresa prvního registru a počet registrů. V odpovědi odpovídá každému registru dvojice bytů.

Požadavek

Kód funkce	1 byte	**0x04**
Počáteční adresa	2 byty	0x0000 až 0xFFFF
Počet registrů	2 byte	1 až 125 (0x7D)

Odpověď

Kód funkce	1 byte	**0x04**
Počet bytů	1 byte	2*__N__
Hodnoty registrů	2*__N__ bytů	

N = počet registrů

Chyba

Kód funkce	1 byte	**0x84**
Chybový kód	1 byty	01, 02, 03 nebo 04

4.5 05 (0x05) Zapiš jednu cívku (Write Single Coil)

Tato funkce slouží k nastavení jednoho výstupu do stavu ON nebo OFF. V požadavku je specifikována adresa výstupu, který se má nastavit a hodnota na, kterou se má nastavit. 0x0000 znamená OFF, 0xFF00 znamená ON. Normální odpověď je kopií požadavku.

Požadavek

Kód funkce	1 byte	**0x05**
Adresa výstupu	2 byty	0x0000 až 0xFFFF
Hodnota výstupu	2 byte	0x0000 nebo 0xFF00

Odpověď

Kód funkce	1 byte	**0x05**
Adresa výstupu	2 byty	0x0000 až 0xFFFF
Hodnota výstupu	2 byty	0x0000 nebo 0xFF00

Chyba

Kód funkce	1 byte	**0x85**
Chybový kód	1 byty	01, 02, 03 nebo 04

4.6 06 (0x06) Zapiš jeden registr (Write Single Register)

Tato funkce slouží k zápisu jednoho uchovávacího registru. V požadavku je specifikována adresa registru, který se má zapsat a hodnota, která se má zapsat. Normální odpověď je kopií požadavku a je vrácena poté, co je registr zapsán.

Požadavek

Kód funkce	1 byte	**0x06**
Adresa registru	2 byty	0x0000 až 0xFFFF
Hodnota registru	2 byte	0x0000 až 0xFFFF

Odpověď

Kód funkce	1 byte	**0x06**
Adresa registru	2 byty	0x0000 až 0xFFFF
Hodnota registru	2 byty	0x0000 až 0xFFFF

Chyba

Kód funkce	1 byte	**0x86**
Chybový kód	1 byty	01, 02, 03 nebo 04

4.7 07 (0x07) Čti stav (Read Exception Status) – pouze pro sériovou linku

Tato funkce slouží ke čtení stavu osmi stavových výstupů. Normální odpověď obsahuje stav těchto výstupů, přenášený v jednom bytu.

Požadavek

Kód funkce	1 byte	**0x07**

Odpověď

Kód funkce	1 byte	**0x07**
Stav výstupů	1 byte	0x00 až 0xFF

Chyba

Kód funkce	1 byte	**0x87**
Chybový kód	1 byty	01 nebo 04

4.8 08 (0x08) Diagnostika (Diagnostics) – pouze pro sériovou linku

Tato funkce slouží k provedení série testů pro zkontrolování komunikace mezi klientem (Master) a serverem (Slave) nebo ke kontrole různých interních chybových stavů serveru. Funkce používá dvoubajtový kód podfunkce, který specifikuje požadovaný typ testu. Normální odpověď obsahuje kopii požadavku případně další data, pokud jsou výsledkem testu.

Požadavek

Kód funkce	1 byte	**0x08**
Podfunkce	2 byty	viz. dále
Data	N*2 bytů	

Odpověď

Kód funkce	1 byte	**0x08**
Podfunkce	2 byty	viz. dále
Data	N*2 bytů	

Chyba

Kód funkce	1 byte	**0x88**
Chybový kód	1 byty	01, 03 nebo 04

4.8.1 Kódy podfunkcí podporovaných sériovými zařízeními

Kód podfunkce		Název
Hex	Dec	
00	00	Vrať data požadavku
01	01	Restartuj komunikaci
02	02	Vrať diagnostický registr
03	03	Změň ASCII oddělovací znak
04	04	Přejdi do pasivního režimu (pouze poslouchej)
	05...09	REZERVOVÁNO
0A	10	Vynuluj čítače a diagnostický registr
0B	11	Vrať počet zpráv
0C	12	Vrať počet komunikačních chyb
0D	13	Vrať počet negativních odpovědí
0E	14	Vrať počet zpracovaných zpráv
0F	15	Vrať počet nezodpovězených zpráv
10	16	Vrať počet zpráv s negativním potvrzením
11	17	Vrať počet zpráv s příznakem zaneprázdněn
12	18	Vrať počet ztracených znaků (zpráv)
13	19	REZERVOVÁNO
14	20	Vynuluj čítač ztracených znaků (zpráv)
	21 ... 65535	REZERVOVÁNO

4.9 11 (0x0B) Čti čítač komunikačních událostí (Get Comm Event Counter) – pouze pro sériovou linku

Tato funkce slouží k získání stavového slova a hodnoty čítače komunikačních událostí. Čítač událostí je inkrementován po každém úspěšném dokončení požadavku. Normální odpověď obsahuje dvoubajtové stavové slovo a dvoubajtový počet událostí.

Požadavek

Kód funkce	1 byte	**0x0B**

Odpověď

Kód funkce	1 byte	**0x0B**
Status	2 byty	0x0000 nebo 0xFFFF
Počet událostí	2 byty	0x0000 až 0xFFFF

Chyba

Kód funkce	1 byte	**0x8B**
Chybový kód	1 byty	01 nebo 04

4.10 12 (0x0C) Čti záznam komunikačních událostí (Get Comm Event Log) – pouze pro sériovou linku

Tato funkce slouží k získání stavového slova, hodnoty čítače komunikačních událostí, čítače zpráv a záznamu komunikačních událostí. Stavové slovo a čítač událostí má stejný význam jako u funkce 11 (0x0B). Normální odpověď obsahuje dvoubajtové stavové slovo, dvoubajtový počet událostí, dvoubajtový počet zpráv a pole obsahující 0 až 64 bytů záznamu událostí.

Požadavek

Kód funkce	1 byte	**0x0C**

Odpověď

Kód funkce	1 byte	**0x0C**
Počet bytů	1 byte	N
Status	2 byty	0x0000 nebo 0xFFFF
Počet událostí	2 byty	0x0000 až 0xFFFF
Počet zpráv	2 byty	0x0000 až 0xFFFF
Záznam událostí	(N-6) bytů	

N = počet bytů záznamu událostí + 3*2 byty (status, počet událostí, počet zpráv)

Chyba

Kód funkce	1 byte	**0x8C**
Chybový kód	1 byty	01 nebo 04

4.11 15 (0x0F) Zapiš více cívek (Write Multiple Coils)

Tato funkce slouží k nastavení až 1968 cívek do stavu ON nebo OFF. V požadavku je specifikována adresa prvního výstupu, který se má nastavit a hodnoty, na které se mají výstupy nastavit. Normální odpověď obsahuje počáteční adresu a počet nastavených cívek.

Požadavek

Kód funkce	1 byte	**0x0F**
Počáteční adresa	2 byty	0x0000 až 0xFFFF
Počet výstupů	2 byty	1 až 1968 (0x7B0)
Počet bytů	1 byte	**N**
Hodnota výstupů	**N** bytů	

N = počet výstupů / 8, je-li zbytek po dělení nenulový, **N** = **N** + 1

Odpověď

Kód funkce	1 byte	**0x0F**
Počáteční adresa	2 byty	0x0000 až 0xFFFF
Počet výstupů	2 byty	1 až 1968 (0x7B0)

Chyba

Kód funkce	1 byte	**0x8F**
Chybový kód	1 byty	01, 02, 03 nebo 04

4.12 16 (0x10) Zapiš více registrů (Write Multiple Registers)

Tato funkce slouží k zápisu bloku až 120 registrů. V požadavku je specifikována adresa prvního registru, který se má zapsat, počet registrů a hodnoty, které se mají zapsat. Normální odpověď obsahuje počáteční adresu a počet zapsaných registrů.

Požadavek

Kód funkce	1 byte	**0x10**
Počáteční adresa	2 byty	0x0000 až 0xFFFF
Počet registrů	2 byty	1 až 120 (0x78)
Počet bytů	1 byte	2***N**
Hodnoty registrů	2***N** bytů	

N = počet registrů

Odpověď

Kód funkce	1 byte	**0x10**
Počáteční adresa	2 byty	0x0000 až 0xFFFF
Počet registrů	2 byty	1 až 120 (0x78)

Chyba

Kód funkce	1 byte	**0x90**
Chybový kód	1 byty	01, 02, 03 nebo 04

4.13 17 (0x11) Sděl identifikaci (Report Slave ID) – pouze pro sériovou linku

Tato funkce slouží ke zjištění typu zařízení, současného stavu a dalších informací o zařízení. Konkrétní obsah odpovědi je závislý na typu zařízení.

Požadavek

Kód funkce	1 byte	**0x11**

Odpověď

Kód funkce	1 byte	**0x11**
Počet bytů	1 byte	
ID zařízení	závislé na zařízení	
Indikátor běhu	1 byte	0x00 = OFF, 0xFF = ON
Další data		

Chyba

Kód funkce	1 byte	**0x91**
Chybový kód	1 byty	01 nebo 04

4.14 20 / 6 (0x14 / 0x06) Čti záznam ze souboru (Read File Record)

Tato funkce slouží ke čtení záznamu ze souboru. Soubor je složen až z 10000 záznamů číslovaných od 0 do 9999. Délka záznamu je udávána v počtu 16-bitových registrů. Funkce může číst několik bloků současně. Každý blok je v požadavku definován v samostatném sub-požadavku o délce 7 bytů. Normální odpověď je sérií sub-odpovědí, jedné pro každý sub-požadavek.

Požadavek

Kód funkce	1 byte	**0x14**
Počet bytů	1 byte	0x07 až 0xF5
Sub-požad. x, typ reference	1 byte	06
Sub-požad. x, číslo souboru	2 byty	0x0000 až 0xFFFF
Sub-požad. x, číslo záznamu	2 byty	0x0000 až 0x270F
Sub-požad. x, délka záznamu	2 byty	**N**

Sub-požad. x+1, ...		

Odpověď

Kód funkce	1 byte	**0x14**
Počet bytů	1 byte	
Sub-odp. x, počet bytů	1 byte	
Sub-odp. x, typ reference	1 byte	06
Sub-odp. x, data záznamu	**N***2 bytů	
Sub-odp. x+1, ...		

Chyba

Kód funkce	1 byte	**0x94**
Chybový kód	1 byty	01, 02, 03, 04 nebo 08

4.15 21 / 6 (0x15 / 0x06) Zapiš záznam do souboru (Write File Record)

Tato funkce slouží ke zápisu záznamu do souboru. Soubor je složen až z 10000 záznamů číslovaných od 0 do 9999. Délka záznamu je udávána v počtu 16-bitových registrů. Funkce může zapisovat několik bloků současně. Každý blok je v požadavku definován v samostatném sub-požadavku o délce 7 bytů + data. Normální odpověď je kopií požadavku.

Požadavek

Kód funkce	1 byte	**0x15**
Počet bytů	1 byte	0x07 až 0xF5
Sub-požad. x, typ reference	1 byte	06
Sub-požad. x, číslo souboru	2 byty	0x0000 až 0xFFFF
Sub-požad. x, číslo záznamu	2 byty	0x0000 až 0x270F
Sub-požad. x, délka záznamu	2 byty	**N**
Sub-požad. x, data záznamu	**N***2 bytů	
Sub-požad. x+1, ...		

Odpověď

Kód funkce	1 byte	**0x15**
Počet bytů	1 byte	0x07 až 0xF5
Sub-požad. x, typ reference	1 byte	06
Sub-požad. x, číslo souboru	2 byty	0x0000 až 0xFFFF
Sub-požad. x, číslo záznamu	2 byty	0x0000 až 0x270F
Sub-požad. x, délka záznamu	2 byty	**N**
Sub-požad. x, data záznamu	**N***2 bytů	
Sub-požad. x+1, ...		

Chyba

Kód funkce	1 byte	**0x95**
Chybový kód	1 byty	01, 02, 03, 04 nebo 08

4.16 22 (0x16) Zapiš registr s maskováním (Mask Write Register)

Tato funkce slouží k modifikaci uchovávacího registru použitím AND a OR masky. Funkci lze použít k nastavení nebo vynulování jednotlivých bitů registru. V požadavku je specifikována adresa registru, AND maska a OR maska. Algoritmus funkce je následující:

Registr = (Registr AND And_Maska) OR (Or_Maska AND (NOT And_Maska))

Normální odpověď je kopií požadavku a je vrácena poté, co je registr modifikován.

Požadavek

Kód funkce	1 byte	**0x16**
Adresa registru	2 byty	0x0000 až 0xFFFF
And_Maska	2 byty	0x0000 až 0xFFFF
Or_Maska	2 byte	0x0000 až 0xFFFF

Odpověď

Kód funkce	1 byte	**0x16**
Adresa registru	2 byty	0x0000 až 0xFFFF
And_Maska	2 byty	0x0000 až 0xFFFF
Or_Maska	2 byte	0x0000 až 0xFFFF

Chyba

Kód funkce	1 byte	**0x96**
Chybový kód	1 byty	01, 02, 03 nebo 04

4.17 23 (0x17) Čti/Zapiš více registrů (Read/Write Multiple Registers)

Tato funkce provádí kombinaci čtení a zápisu registrů v jedné MODBUS transakci. Operace zápisu je provedena před operací čtení. V požadavku je specifikována adresa prvního registru a počet registrů, které se mají číst a adresa, počet registrů a hodnoty, které se mají zapsat. Normální odpověď obsahuje data přečtená z registrů.

Požadavek

Kód funkce	1 byte	**0x17**
Počáteční adresa pro čtení	2 byty	0x0000 až 0xFFFF
Počet registrů pro čtení	2 byty	1 až 118 (0x0076)
Počáteční adresa pro zápis	2 byty	0x0000 až 0xFFFF
Počet registrů pro zápis	2 byty	1 až 118 (0x0076)
Počet zapisovaných bytů	1 byte	**N***2
Hodnoty registrů	**N***2 bytů	

N = počet registrů pro zápis

Odpověď

Kód funkce	1 byte	**0x17**
Počet bytů	1 byte	**M***2
Hodnoty přečtených registrů	**M***2 bytů	

M = počet registrů pro čtení

Chyba

Kód funkce	1 byte	**0x97**
Chybový kód	1 byty	01, 02, 03 nebo 04

4.18 24 (0x18) Čti FIFO frontu (Read FIFO Queue)

Tato funkce umožňuje číst obsah FIFO fronty registru. Funkce vrací počet a obsah registrů ve frontě. Může být přečteno až 32 registrů; délka fronty + až 31 registrů ve frontě. Je-li fronta delší než 31 registrů, je vrácen chybový kód 03.

Požadavek

Kód funkce	1 byte	**0x18**
Adresa FIFO	2 byty	0x0000 až 0xFFFF

Odpověď

Kód funkce	1 byte	**0x18**
Počet bytů	2 byty	
Délka fronty	2 byty	<= 31
Obsah fronty	**N***2 bytů	

N = délka fronty

Chyba

Kód funkce	1 byte	**0x98**
Chybový kód	1 byty	01, 02, 03 nebo 04

4.19 43 (0x2B) Zapouzdřený přenos (Encapsulated Interface Transport)

MODBUS zapouzdřený přenos (MODBUS Encapsulated Interface – MEI) je mechanizmus pro tunelování požadavků a jejich odpovědí uvnitř PDU.

Požadavek

Kód funkce	1 byte	**0x2B**
Typ MEI	1 byte	0x0E
Data dle typu MEI	n bytů	

Odpověď

Kód funkce	1 byte	**0x2B**
Typ MEI	1 byte	0x0E
Data dle typu MEI	n bytů	

Chyba

Kód funkce	1 byte	**0xAB**
Typ MEI	1 byte	0x0E
Chybový kód	1 byty	01, 02, 03 nebo 04

4.20 43 / 13 (0x2B / 0x0D) CANOpen základní odkaz (CANOpen General Reference)

Tato funkce je zapouzdřením služeb, které slouží pro přístup ke CANOpen zařízením a systému.

4.21 43 / 14 (0x2B / 0x0E) Čti identifikaci zařízení (Read Device Identification)

Tato funkce umožňuje čtení identifikace a dalších údajů týkajících se popisu zařízení. Identifikace zařízení je složena z množiny objektů, z nichž každý má svou identifikaci.

Existují tři skupiny objektů:

- Základní identifikace zařízení
- Obvyklá identifikace zařízení
- Rozšířená identifikace zařízení

ID objektu	Název objektu / popis	Typ	Povinný / nepovinný	Kategorie
0x00	Název výrobce	ASCII řetězec	**Povinný**	**Základní**
0x01	Kód produktu	ASCII řetězec	**Povinný**	
0x02	Hlavní/vedlejší verze	ASCII řetězec	**Povinný**	
0x03	URL výrobce	ASCII řetězec	Nepovinný	**Obvyklá**
0x04	Název výrobku	ASCII řetězec	Nepovinný	
0x05	Název modelu	ASCII řetězec	Nepovinný	
0x06	Název uživatelské aplikace	ASCII řetězec	Nepovinný	
0x07 ... 0x7F	*Rezervováno*		Nepovinný	
0x80 ... 0xFF	*Závislé na produktu, lze definovat vlastní objekty*	Závislé na zařízení	Nepovinný	**Rozšířená**

Požadavek

Kód funkce	1 byte	**0x2B**
Typ MEI	1 byte	0x0E
ID kód	1 byte	01 / 02 / 03 / 04
ID objektu	1 byte	0x00 až 0xFF

Odpověď

Kód funkce	1 byte	**0x2B**
Typ MEI	1 byte	0x0E
ID kód	1 byte	01 / 02 / 03 / 04
Úroveň shody	1 byte	
Pokračování	1 byte	0x00 nebo 0xFF
ID dalšího objektu	1 byte	ID objektu
Počet objektů	1 byte	
Seznam		
ID objektu	1 byte	
Délka objektu	1 byte	**N**
Hodnota objektu	**N**	Závisí na ID objektu

Chyba

Kód funkce	1 byte	**0xAB**
Typ MEI	1 byte	0x0E
Chybový kód	1 byty	01, 02, 03 nebo 04

5 Záporné odpovědi

Když klient posílá serveru požadavek, očekává na něj odpověď. Mohou nastat čtyři situace:

- Jestliže server přijme bezchybně požadavek a je schopen jej normálně zpracovat, vrátí klientovy normální odpověď.
- Jestliže server požadavek nepřijme z důvodu komunikační chyby, není vrácena žádná odpověď. Na straně klienta dojde k vypršením časového limitu pro příjem odpovědi.
- Jestliže server přijme požadavek, ale detekuje komunikační chybu (parita, CRC...), nevrací žádnou odpověď. Na straně klienta dojde k vypršením časového limitu pro příjem odpovědi.

- Jestliže server přijme bezchybně požadavek, ale není schopen jej normálně zpracovat, vrátí klientovi zápornou odpověď s udáním důvodu neúspěchu.

Normální a záporná odpověď se liší nejvyšším bitem kódu funkce. Je-li bit nulový, jedná se o normální odpověď, je-li bit nastavený, jedná se o zápornou odpověď. V případě záporné odpovědi je v datové části předán kód chyby. V následující tabulce je seznam možných chybových kódů.

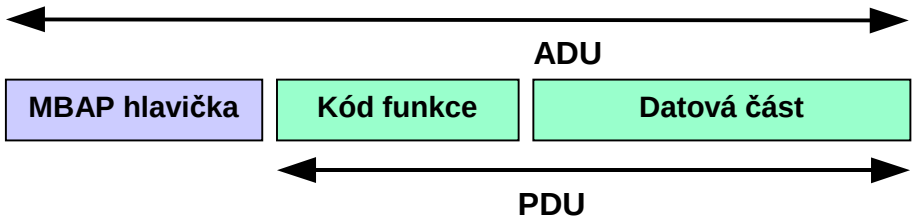
MODBUS chybové kódy		
Kód	**Jméno**	**Význam**
01	Ilegální funkce	Požadovaná funkce není serverem podporována
02	Ilegální adresa dat	Zadaná adresa je mimo serverem podporovaný rozsah
03	Ilegální hodnota dat	Předávaná data jsou neplatná
04	Selhání zařízení	Při provádění požadavku došlo k neodstranitelné chybě
05	Potvrzení	Kód určený k použití při programování. Server hlásí přijetí platného požadavku, ale jeho vykonání bude trvat delší dobu
06	Zařízení je zaneprázdněné	Kód určený k použití při programování. Server je zaneprázdněn vykonáváním dlouho trvajícího příkazu.
08	Chyba parity paměti	Kód určený k použití při práci se soubory. Server při pokusu přečíst soubor zjistil chybu parity
0A	Brána – přenosová cesta nedostupná	Kód určený k práci s bránou (gateway). Brána není schopná vyhradit interní přenosovou cestu od vstupního portu k výstupnímu. Pravděpodobně je přetížená nebo nesprávně nastavená.
0B	Brána – cílové zařízení neodpovídá	Kód určený k práci s bránou (gateway). Cílové zařízení neodpovídá, pravděpodobně není přítomno.

6 Implementace MODBUSu

MODBUS standard definuje kromě aplikační vrstvy ISO/OSI modelu i některé implementace protokolu na konkrétní typ sítě nebo sběrnice. Příkladem je MODBUS na TCP/IP a MODBUS na sériové lince.

6.1 MODBUS na TCP/IP

Na obrázku 6.1 je znázorněn formát MODBUS zprávy na TCP/IP. Pro identifikaci MODBUS ADU je použita MBAP hlavička (MODBUS Application Protocol Header).



Obr. 6.1: MODBUS zpráva na TCP/IP

Pro posílání MODBUS/TCP ADU je na TCP vyhrazen registrovaný port 502.

6.2 MODBUS na sériové lince

MODBUS Serial Line protokol je protokol typu Master-Slave a je definován na úrovni 2 ISO/OSI modelu. Na fyzické úrovni 0 ISO/OSI modelu můžou být použita různá sériová rozhraní, například RS-232 nebo RS-485 a jejich varianty.

Princip protokolu

Jedná se o Master/Slave protokol. V jeden okamžik může být na sběrnici pouze jeden master a 1 až 247 slave jednotek. Komunikaci vždy zahajuje master, slave nesmí nikdy vysílat data bez pověření mastera .
Master posílá požadavky slave jednotkám ve dvou režimech:
- unicast režim – master adresuje požadavek jedné konkrétní slave jednotce a ta pošle odpověď
- broadcast režim – master posílá požadavek všem jednotkám, žádná jednotka neodpoví.

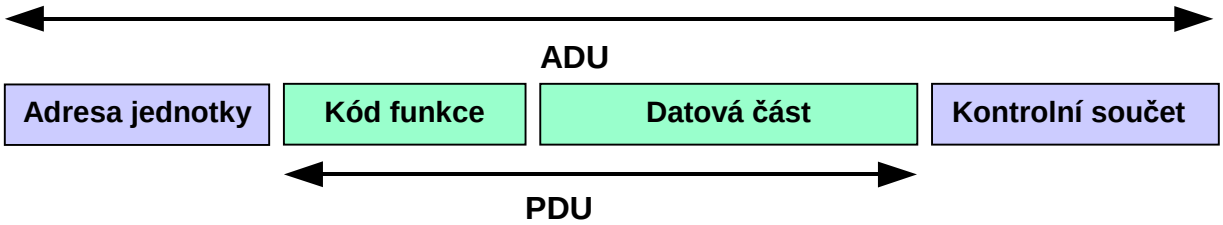
Adresovací pravidla

Adresní prostor zahrnuje 256 různých adres.

0	1 až 247	248 až 255
Broadcast adresa	Individuální adresa slave jednotky	Rezervováno

Master nemá žádnou specifickou adresu, pouze slave jednotky musejí mít adresu a ta musí být v celé MODBUS síti jedinečná.

Na obrázku 6.2 je znázorněn základní formát MODBUS aplikační zprávy na sériové lince. Zpráva kromě standardní MODBUS PDU obsahuje pole Adresa jednotky. Toto pole obsahuje adresu slave jednotky. Pole Kontrolní součet slouží k detekci chyb a obsahuje CRC nebo LRC kód v závislosti na vysílacím režimu.



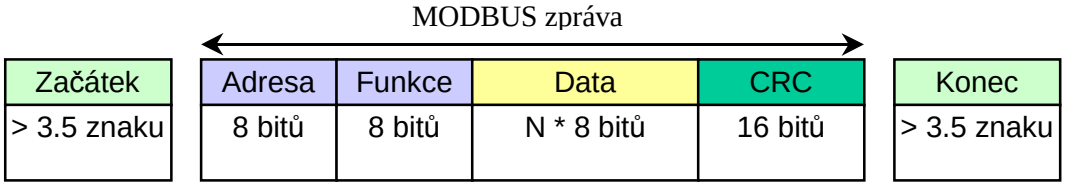
Obr. 6.2: Základní tvar MODBUS zprávy na sériové lince

Vysílací režimy

MODBUS protokol definuje dva sériové vysílací režimy, MODBUS RTU a MODBUS ASCII. Režim určuje v jakém formátu jsou data vysílána jak dekódována. Každá jednotka musí podporovat režim RTU, režim ASCII je nepovinný. Všechny jednotky na jedné sběrnici musejí pracovat ve stejném vysílacím režimu.

6.2.1 MODBUS RTU

V režimu RTU obsahuje každý 8-bitový byte zprávy dva 4-bitové hexadecimální znaky. Vysílání zprávy musí být souvislé, mezery mezi znaky nesmějí být delší než 1.5 znaku. Začátek a konec zprávy je identifikován podle pomlky na sběrnici delší než 3.5 znaku. Formát RTU rámce je znázorněn na obrázku 6.3.



Obr. 6.3: RTU rámec zprávy

K detekci chyb slouží 16-bitové CRC pole s generujícím polynomem $x^{16} + x^{15} + x^2 + 1$.

Formát bytu (11 bitů):

1 start bit
8 datových bitů
1 bit parita
1 stop bit

Každá jednotka musí podporovat sudou paritu. Pokud není použita parita, je nahrazena druhým stop bitem.

6.2.2 MODBUS ASCII

V režimu ASCII je každý 8-bitový byte posílán jako dvojice ASCII znaků. Oproti režimu RTU je tedy pomalejší, ale umožňuje vysílat znaky s mezerami až 1 s. Začátek a konec zprávy je totiž určen odlišně od RTU módu. Začátek zprávy je indikován znakem „:“ a konec zprávy dvojicí řídicích znaků CR, LF. Formát ASCII rámce je na obrázku 6.4.

Začátek	Adresa	Funkce	Data	LRC	Konec
znak „:“	2 znaky	2 znaky	0 až 2*252 znaků	2 znaky	2 znaky CR, LF

Obr. 6.4: ASCII rámec zprávy

K detekci chyb slouží 8 bitové LRC pole.

Formát bytu (10 bitů):

1 start bit
7 datových bitů
1 bit parita
1 stop bit

Každá jednotka musí podporovat sudou paritu. Pokud není použita parita, je nahrazena druhým stop bitem.