MPC400
MODBUS RTU

Návod k provozu komunikace v režimech Modbus RTU Master a Slave
Popis prostředků v SIMPLE4

edice 11.2014
verze 1.1

MPC400 - Modbus RTU

Ing. Tomáš Rázga

http:/www.micropel.cz

OBSAH

OBSAH **2**
1. Úvod k Modbus RTU **3**
2. Modbus RTU Master **4**
2.1. Prostředky pro uživatelský program 4
2.2. Návod k tvorbě programu 4
2.3. Ilustrační příklad 5
2.4. Podporované příkazy Modbus 6
2.5. Datové struktura _modbusMA 11
+ _modbusMA 11
2.6. Funkce knihovny Modbus.lib 13
+ ModbusMA_Config 13
+ ModbusMA_STM 14
+ ModbusMA_Send 14
3. Modbus RTU Slave **15**
3.1. Parametry ovladače 15
3.2. Registry a bity přenášené po Modbus 15
3.3. Podporované příkazy Modbus 16
3.4. Význam Modbus registrů MEX400 16
3.5. Význam Modbus registrů MPC400 16
Stránky 16
Nulové registry 17
Obsah jednotlivých stránek 17
3.6. Mapování IO modulů na registry Modbus 18
3.7. Podpora v programu MPC400 19
Definice pole Modbus registrů 19
Funkce zpřístupnění pole Modbus registrů 19
+ ModbusSL_AssignRegs 19
Funkce zabránění přístupu ke stránkám registrů Modbus 20
+ ModbusSL_DisablePages 20

1. Úvod k Modbus RTU

Modbus RTU je standardní komunikační protokol na lince RS485 používaný mnoha výrobci. Protokol je postaven na principu komunikace master - slave a umožňuje jednomu zařízení Master na lince vyčítat a nastavovat různé registry a bity vetšího počtu (až 246) zařízení slave. Registry jsou definovány jako 16-bitové, význam jednotlivých registrů a bitů udává vždy výrobce slave zařízení. Hodnoty se přenáší ve formátu Big-endian.

2. MODBUS RTU MASTER

Na lince RS485 MPC400 lze snadno provozovat komunikaci protokolem Modbus RTU v režimu Master. Pro tuto funkci MPC400 zvolíme ovladačem linky ovladač USART a v programu použijeme prostředky knihovny Modbus.lib. V režimu Master může být k PLC připojen libovolný počet různých zařízení slave. Z programu PLC pak je možno přistupovat k jednotlivým bitům nebo 16-bitovým registrům těchto zařízení, jejichž význam je definován výrobcem. PLC má funkci mastera, takže komunikace probíhá tak, že PLC vždy iniciuje transakci vysláním adresného příkazu na linku a následně očekává odpověď. V dalším textu následuje popis prostředků a práce s knihovnou a popis podporovaných příkazů protokolu Modbus v režimu Master.

2.1. Prostředky pro uživatelský program

Knihovna Modbus.lib definuje datovou strukturu _modbusMA. Proměnná této struktury reprezentuje v programu linku RS485 s komunikací Modbus RTU Master a slouží k nastavování parametrů příkazů protokolu Modbus a ukládání stavových informací komunikace na lince. Je předávána odkazem do všech funkcí knihovny určených pro režim Master. V následujícím textu je odkazováno na jednotlivé položky struktury _modbusMA, proto je zde uvedena definice této struktury. Detailní popis jednotlivých položek lze pak lze nalézt v samostatné sekci věnované popisu struktury _modbusMA v tomto dokumentu.

```
type struct
      byte addrSL,      ; adresa cílového slave zařízení na lince
      byte cmd,         ; kód Modbus příkazu
      byte diagfun,     ; sub-kód pro Modbus příkaz 0x08 (Diagnostics)
      word nitems,      ; počet vyčítaných / zapisovaných registrů / bitů
      word daddr,       ; číslo (adresa) prvního registru / bitu příkazu
      word[16] reg,     ; pole se zapisovanými nebo vyčtenými daty
      word timesilent,  ; minimální doba klidu na lince
      byte state,       ; stav vykonávání zadaného příkazu
      byte comst,       ; pro uživatele nevýznamná položka
      longword rxendtime,     ; pro uživatele nevýznamná položka
      word ncopy,       ; pro uživatele nevýznamná položka
      word ncopyix,     ; pro uživatele nevýznamná položka
      word rxbix,       ; pro uživatele nevýznamná položka
      _uart ua          ; pro uživatele nevýznamná položka
end _modbusMA
```

Knihovna také definuje funkce ModbusMA_Config, ModbusMA_STM a ModbusMA_Send. Pomocí nich lze nastavit parametry komunikace a zadávat a obsluhovat jednotlivé transakce (příkazy) na lince. Definice funkcí a jejich popis lze nalézt v samostatné sekci.

Dále jsou pro hodnoty kódů jednotlivých podporovaných příkazů protokolu Modbus definovány symbolické konstanty, jejich název je vždy uvozen znaky "_MBCM_".

2.2. Návod k tvorbě programu

Pro reprezentaci příslušné linky se v programu založí proměnná s datovou strukturou _modbusMA. Před zadáním prvního Modbus příkazu v programu je třeba nejdříve nastavit komunikační parametry. To lze provést, nejlépe ihned po restartu automatu, zavoláním funkce ModbusMA_Config. Funkci v programu opětovně zavoláme pouze v případě, že chceme provést změnu nastavení komunikačních parametrů. Úspěšný průběh potvrzuje funkce ModbusMA_Config návratovou hodnotou 0.

Nový příkaz lze zadat odkudkoli z programu nastavením položek příslušné strukturované proměnné, reprezentující linku, na správné hodnoty a následným zavoláním funkce ModbusMA_Send. Pro správnou funkci programu je třeba mít ošetřeno, aby naposled zadaný příkaz již byl v tom okamžiku dokončen a předávané parametry nového příkazu byly platné. Při každém průchodu hlavní programovou smyčkou se volá procedura ModbusMA_STM, která zajišťuje obsluhu zadaných transakcí (příkazů) na lince. Procedura pracuje s proměnnou pro reprezentaci linky, v průběhu vždy nastaví její položku *.state* na hodnotu odpovídající stavu naposledy zadaného příkazu. Pomocí této hodnoty je tak možno v programu detekovat dokončení příkazu a i vyhodnotit výsledek, s jakým skončil. Jednotlivé hodnoty *.state* mají následující význam:

0 … probíhá naposled zadaný příkaz
1 … naposled zadaný příkaz dokončen (již dříve)
3 … naposled zadaný příkaz právě ukončen, bez chyby
5 … naposled zadaný příkaz právě ukončen, timeout
255 až 253 … naposled zadaný příkaz právě ukončen, chyba při dekódování odpovědi
240 … naposled zadaný příkaz právě ukončen, neočekávaný datový obsah v odpovědi
231 až 234 … naposled zadaný příkaz právě ukončen, nahlášena vyjímka (Exception Response) s kódem 1 až 4: (1) neplatná funkce, (2) neplatná adresa, (3) neplatná hodnota, (4) chyba zařízení
100 až 108 … naposled zadaný příkaz právě ukončen, neočekávaný stav ovladače USART linky, hodnota znamená výsledek výrazu (100+*ua.status)*

Posloupnost nastavovaných hodnot *.state* strojkem v proceduře ModbusMA_STM bude tedy obecně následující. V průběhu zadaného a neukončeného příkazu bude procedurou vždy nastavována hodnota 0. Při dokončení příkazu procedura jednorázově nastaví tuto položku na hodnotu větší než 1. Při následujících průchodech procedurou pak již bude nastavována hodnota 1, dokud nedojde k zadání nového Modbus příkazu v programu.

Pozn.: V případě, že cílové zařízení slave je přítomno na lince a nepodporuje masterem vyslaný příkaz, případně, nedisponuje-li všemi registry nebo bity, na něž je zadaný příkaz směřován, může zařízení slave odpověď vynechat, nebo může vrátit vyjímku (Exception Response).

2.3. Ilustrační příklad

V programu po resetu vybereme komunikační linku a nastavíme parametry Modbus komunikace, u automatu je třeba vybrat ovladač USART pro příslušnou linku. Poté lze vlastním kódem v programu zadávat Modbus příkazy na linku. Takže, když má proměnná *cmd_snd* hodnotu 0, lze do proměnné *modM* zapsat parametry příkazu a nastavit *cmd_snd* na hodnotu 1. Poté se začne příkaz vyřizovat. Jeho ukončení bude indikováno nastavením *cmd_snd* na hodnotu 0. Výsledek, s jakým příkaz dopadl, bude uložen v proměnné *cmd_result* (hodnota -1 značí neplatné parametry příkazu, hodnota 0 značí probíhající příkaz, hodnota 3 značí úspěšné provedení, další možné hodnoty odpovídají hodnotám položky *modM.state* s významem popsaným výše.

```
; -- HLAVNÍ SMYČKA PROGRAMU --
......
var _modbusMA modM
```

```
var byte cmd_snd, res
var int cmd_result
if (RESET) then
      begin
      cmd_snd=0
      cmd_result=32767
      ModbusMA_Config(modM,1,57600,0,20,5)     ; linka L2, 57600 Bd, bez
                                               ; parity, timeout 20 ms, doba
                                               ; doba klidu 5 ms
      end
ModbusMA_STM(modM)      ; obsluha Modbus transkací
res=modM.state
if (res > 1) then       ; zadaný Modbus příkaz právě dokončen
      begin
      cmd_result=int(res)     ; ukládáme výsledek příkazu,
                              ; příkaz bez chyby, když cmd_result=3
      cmd_snd=0
      end
else if (res = 1) then        ; poslední Modbus příkaz byl již dříve ukončen
      begin
      if (cmd_snd=1) then ; má být zadán nový příkaz
            begin
            if (ModbusMA_Send(modM) <> 0) then
                  cmd_snd=2
            else ; chyba v zadani Modbus prikazu
                  begin
                  cmd_snd=0
                  cmd_result=-1
                  end
            end
      end
else  ; probíhá zadaný Modbus příkaz
      cmd_result=0
......
; -- KONEC HLAVNÍ SMYČKY --
RESET=0
end
```

2.4. Podporované příkazy Modbus

Knihovna podporuje příkazy čtení a zápisu bitů i registrů. Registry Modbus zařízení jsou vždy 16-bitové (typu WORD). U příkazů pro zápis a čtení více hodnot je počet registrů omezen na 1 až 16 a počet bitů na 1 až 256. Také jsou podporovány diagnostické funkce protokolu Modbus. Symbolické konstanty knihovny Modbus.lib s kódy podporovaných Modbus příkazů mají definici:

```
const _MBCM_READ_COILS = 0x01,
      _MBCM_READ_DESCRETE_INPUTS = 0x02,
      _MBCM_READ_HOLDING_REGISTERS = 0x03,
      _MBCM_READ_INPUT_REGISTERS = 0x04,
      _MBCM_WRITE_SINGLE_COIL = 0x05,
      _MBCM_WRITE_SINGLE_REGISTER = 0x06,
      _MBCM_READ_EXCEPTION_STATUS = 0x07,
      _MBCM_DIAGNOSTICS = 0x08,
      _MBCM_GET_COM_EVENT_COUNTER = 0x0B,
```

```
_MBCM_WRITE_MULTIPLE_COILS = 0x0F,
_MBCM_WRITE_MULTIPLE_REGISTERS = 0x10,
_MBCM_MASK_WRITE_REGISTER = 0x16
```

Nyní následuje popis jednotlivých podporovaných příkazů. U každého příkazu je uvedeno, jaké hodnoty položek pracovní proměnné se strukturou _modbusMA je třeba pro daný příkaz nastavit a jak bude vypadat případný výsledek příkazu. Uváděné návratové hodnoty odpovídají případu, kdy byl zadaný příkaz dokončen bezchybně, tj. kdy při ukončení příkazu došlo k jednorázovému nastavení položky *.state* stavovým strojkem v ModbusMA_STM na hodnotu 3.

READ_COILS (0x01)

Příkaz čtení zadaného počtu bitů.

Parametry příkazu:
.addrSL ... adresa cílového zařízení slave
.cmd … 0x01
.nitems ... počet bitů (1-256)
.daddr ... číslo (adresa) nejnižšího bitu
Návratové hodnoty:
.reg[] ... hodnoty bitů (prvním bitem je bit *.reg[0]?0*)

READ_DESCRETE_INPUTS (0x02)

Příkaz čtení zadaného počtu digitálních vstupů (Read-Only bitů).

Parametry příkazu:
.addrSL ... adresa cílového zařízení slave
.cmd … 0x02
.nitems ... počet bitů (1-256)
.daddr ... číslo (adresa) nejnižšího bitu
Návratové hodnoty:
.reg[] ... hodnoty bitů (prvním bitem je bit *.reg[0]?0*)

READ_HOLDING_REGISTERS (0x03)

Příkaz čtení zadaného počtu registrů.

Parametry příkazu:
.addrSL ... adresa cílového zařízení slave
.cmd ... 0x03
.nitems ... počet registrů (1-16)
.daddr … číslo (adresa) nejnižšího registru
Návratové hodnoty:
.reg[] ... hodnoty registrů

READ_INPUT_REGISTERS (0x04)

Příkaz čtení zadaného počtu registrů vstupů (Read-Only registrů).

Parametry příkazu:
.addrSL ... adresa cílového zařízení slave
.cmd ... 0x04
.nitems ... počet registrů (1-16)
.daddr … číslo (adresa) nejnižšího registru
Návratové hodnoty:
.reg[] ... hodnoty registrů

WRITE_SINGLE_COIL (0x05)

Příkaz zápisu jednoho bitu.

Parametry příkazu:
.addrSL ... adresa cílového zařízení slave
.cmd ... 0x05
.daddr ... číslo (adresa) bitu
.reg[0]?0 ... hodnota bitu
Návratové hodnoty:
-

WRITE_SINGLE_REGISTER (0x06)

Příkaz zápisu jednoho registru.

Parametry příkazu:
.addrSL ... adresa cílového zařízení slave
.cmd ... 0x06
.daddr ... číslo (adresa) registru
.reg[0] ... hodnota registru
Návratové hodnoty:
-

WRITE_MULTIPLE_COILS (0x0F)

Příkaz zápisu zadaného počtu bitů.

Parametry příkazu:
.addrSL ... adresa cílového zařízení slave
.cmd ... 0x0F
.nitems ... počet bitů (1-256)
.daddr ... číslo (adresa) nejnižšího bitu
.reg[] ... hodnoty bitů (prvním bitem je bit *.reg[0]?0*)
Návratové hodnoty:
-

WRITE_MULTIPLE_REGISTERS (0x10)

Příkaz zápisu zadaného počtu registrů.

Parametry příkazu:
.addrSL ... adresa cílového zařízení slave
.cmd ... 0x10
.nitems ... počet registrů (1-16)
.daddr ... číslo (adresa) nejnižšího registru
.reg[] ... hodnoty registrů
Návratové hodnoty:
-

MASK_WRITE_REGISTER (0x16)

Příkaz nastavení hodnoty registru pomocí zadaných bitových masek *mskAND* a *mskOR*. Cílový registr *r* bude nastaven na hodnotu (*r* AND *mskAND*) OR (*mskOR* AND neg(*mskAND)*). Operace *neg(.)* má význam negace všech bitů. Příkazem tedy lze nastavit pouze vybrané bity registru, ostatní bity zůstanou bez změny.

Parametry příkazu:
.addrSL ... adresa cílového zařízení slave
.cmd ... 0x16
.daddr ... číslo (adresa) registru
.reg[0] ... hodnota *mskAND*
.reg[1] ... hodnota *mskOR*
Návratové hodnoty:
-

READ_EXCEPTION_STATUS (0x07)

Příkaz čtení hodnoty Exception Status.

Parametry příkazu:
.addrSL ... adresa cílového zařízení slave
.cmd ... 0x07
Návratové hodnoty:
.reg[0] ... hodnota Exception Status; tvoří ji pouze spodní bajt (8 bitů); význam specifikuje výrobce zařízení slave

GET_COM_EVENT_COUNTER (0x0B)

Příkaz čtení Com Event hodnot.

Parametry příkazu:
.addrSL ... adresa cílového zařízení slave
.cmd ... 0x0B
Návratové hodnoty:
.reg[0] ... hodnota Status
.reg[1] ... hodnota Event Count

DIAGNOSTICS (0x08)

Různé diagnostické funkce.

Parametry příkazu:
.addrSL ... adresa cílového zařízení slave
.cmd ... 0x08
.diagfun ... spodní bajt kódu diagnosticke funkce
Další parametry a návratové hodnoty se odvíjejí od typu diag. funkce

Return Query Data (0x00)
Funkcí se odešlou data, která by měl slave v odpovědi zopakovat.
Specifické parametry:
.nitems ... počet dat (wordů)
.reg[] ... hodnoty dat k odeslání
Návratové hodnoty:
.reg[] ... echo zadaných hodnot *.reg[]*

Restart Communications Option (0x01)
Funkce restartuje komunikaci slave a také zruší případný režim poslechu.
Specifické parametry:
.reg[0] … nastavit 0x0000 nebo 0xFF00, jinak nastane chyba - 0xFF00 značí požadavek i na smazání logu událostí komunikace
Návratové hodnoty:
.reg[0] ... echo zadané hodnoty *.reg[0]*
Return Diagnostic Register (0x02)
Funkce vyčte diagnostický registr.
Specifické parametry:
-
Návratové hodnoty:
.reg[0] ... hodnota diagnostického registru
Force Listen Only Mode (0x04)
Funkce přepne slave do režimu poslechu.
Specifické parametry:
-
Návratové hodnoty:
- funkce by měla skončit bez odpovědi od slave (timeoutem)
Clear Counters and Diagnostic Register (0x0A)
Funkce maže všechna počítadla a diagnostický registr.
Specifické parametry:
-
Návratové hodnoty:
.reg[0] ... hodnota 0x0000
Return Bus Message Count (0xB)
Return Bus Communication Error Count (0xC)
Return Bus Exception Error Count (0xD)
Return Slave Message Count (0xE)
Return Slave No Response Count (0xF)
Return Slave NAK Count (0x10)
Return Slave Busy Count (0x11)
Return Bus Character Overrun Count (0x12)
Funkce vyčtou požadovaný údaj slave.
Specifické parametry:
-
Návratové hodnoty:
.reg[0] ... hodnota požadovaného údaje
Clear Overrun Counter and Flag (0x14)
Funkce maže počítadlo i příznak Overrurn.
Specifické parametry:
-
Návratové hodnoty:
.reg[0] ... hodnota 0x0000

2.5. Datové struktura _modbusMA

+ _modbusMA

Použití :	**definice struktury proměnné v knihovně Modbus.lib** **pro komunikaci Modbus RTU Master na lince RS485 s ovladačem USART** **ve funkcích ModbusMA_Config / ModbusMA_STM / ModbusMA_Send**
Definice :	type struct byte addrSL, ; adresa cílového slave zařízení na lince byte cmd, ; kód Modbus příkazu byte diagfun, ; sub-kód pro Modbus příkaz 0x08 (Diagnostics) word nitems, ; počet vyčítaných / zapisovaných registrů / bitů word daddr, ; číslo (adresa) prvního registru / bitu příkazu word[16] reg, ; pole se zapisovanými nebo vyčtenými daty word timesilent, ; minimální doba klidu na lince byte state, ; stav vykonávání zadaného příkazu byte comst, ; pro uživatele nevýznamná položka longword rxendtime, ; pro uživatele nevýznamná položka word ncopy, ; pro uživatele nevýznamná položka word ncopyix, ; pro uživatele nevýznamná položka word rxbix, ; pro uživatele nevýznamná položka _uart ua ; pro uživatele nevýznamná položka end **_modbusMA**

Definice datové struktury je součástí knihovny Modbus.lib. Ve spolupráci s funkcemi ModbusMA_Config, ModbusMA_STM a ModbusMA_Send umožňuje zadávání a vyhodnocování Modbus RTU Master transakcí (příkazů). Některé položky jsou relevantní pouze pro některé příkazy. Seznam podporovaných příkazů Modbus, a informace, jaké položky tyto příkazy využívají, jsou uvedeny v souhrnném popisu režimu Master v tomto dokumentu. Nyní následuje podrobný popis významu položek struktury:

addrSL

Adresa slave zařízení na lince, kterému je příkaz adresován - každý slave na lince má mít nastavenu unikátní adresu z rozsahu 1 až 247 (adresa 0 je vyhrazena pro broadcast). Položku je třeba v programu nastavit před zadáním příkazu funkcí ModbusMA_Send.

cmd

Kód (číslo) Modbus příkazu. Použíjí se stejné kódy, jaké uvádí specifikace protokolu Modbus. Pro podporované příkazy nicméně knihovna Modbus.lib definuje i odpovídající symbolické konstanty - hodnoty a názvy konstant jsou uvedeny v souhrnném popisu režimu Master v tomto dokumentu. Položku je třeba v programu nastavit před zadáním příkazu funkcí ModbusMA_Send.

diagfun

Sub-kód (číslo) diagnostické funkce. Položku je třeba nastavit pouze u diagnostického příkazu. Použíjí se stejné sub-kódy, jaké uvádí specifikace protokolu Modbus. Všechny podporované diagnostické funkce jsou uvedeny v souhrnném popisu ovladače Modbus RTU Master v tomto dokumentu. Položku je třeba v programu nastavit před zadáním příkazu s kódem 0x08 (_MBCM_DIAGNOSTICS) funkcí ModbusMA_Send.

nitems

Počet vyčítaných / zapisovaných registrů / bitů v příkazu. Položku je třeba v programu nastavit před zadáním příkazu funkcí ModbusMA_Send, ale pouze u příkazů, u nichž je tento parametr volitelný.

daddr

Číslo (adresa) prvního registru / bitu příkazu, číslování začíná od 0. Položku je třeba v programu nastavit před zadáním příkazu funkcí ModbusMA_Send, ale pouze u příkazů, u nichž je tento parametr volitelný.

reg

Pole 16 WORDů pro zadání odesílaných dat v příkazu a pro uložení přijatých dat v odpovědi na příkaz. V případě, že daty jsou hodnoty Modbus registrů, jednotlivé prvky pole *.reg* přímo odpovídají hodnotám registrů. V případě Modbus bitů může každý prvek pole *.reg* obsahovat až 16 platných hodnot bitů - první hodnotou je vždy hodnota *.reg[0]?0* a např. osmnáctou je hodnota *.reg[1]?2*. Odesílané hodnoty je třeba v programu nastavit před zadáním příkazu funkcí ModbusMA_Send, případné přijaté hodnoty bude možno v poli *.reg* nalézt po úspěšném dokončení příkazu na základě vyhodnocení procedurou ModbusMA_STM.

timesilent

Doba v milisekundách počítaná od příjmu odpovědi na poslední vyslaný příkaz, po kterou komunikační strojek čeká, než může odvyslílat další příkaz na linku. Takže, když např. po vyhodnocení konce jednoho příkazu zadáme ihned příkaz další, dojde ke skutečnému odvysílání nového příkazu na linku až za stanovenou dobu. Paramter umožňuje respektovat požadavky na časování komunikace libovolného zařízení slave. Položka je nastavena funkcí ModbusMA_Config. Lze ji také měnit kdykoli během programu, ale ve většině případů to nebude nutné. Při konfiguraci se pak zvolí nejdelší doba, krterá vyhoví všem zařízením slave na lince.

state

Hodnota udává stav naposled zadné transakce (příkazu). Položku automaticky nastavují obslužné funkce Modbus komunikace, ve vlastním programu je určena jen pro čtení. Význam jednotlivých nastavovaných hodnot je vysvětlen v souhrnném popisu režimu Master v tomto dokumentu.

2.6. Funkce knihovny Modbus.lib

+ ModbusMA_Config

Deklarace :	**function byte ModbusMA_Config(var _modbusMA m, word un, longword br, byte p, longword tout, word ts)**	
Parametr 1 :	**m**	proměnná se strukturou _modbusMA pro reprezentaci jedné z linek RS485 s režimem komunikace Modbus RTU Master v programu
2 :	**un**	číslo linky RS485
3 :	**br**	komunikační rychlost na lince v Bd
4 :	**p**	nastavení pro použití využití paritního bitu v komunikaci
5 :	**tout**	maximální doba čekání na příjem odpovědi po vysílání v ms
6 :	**ts**	minimální doba klidu na lince po dokončení transakce v ms
Výstup :	**err**	kód chyby nastavení

Funkce provede nastavení parametrů linky RS485 pro režim komunikace Modbus RTU Master. Bude v programu zavolána nejspíše jen jednou, po restartu PLC (každopádně před zadáním prvního Modbus příkazu). V případě úspěšného nastavení vrátí hodnotu 0, hodnotu jinou vrátí při zadání chybných parametrů nebo pokud jako ovladač linky není zvolen ovladač USART.

Před zavoláním funkce není třeba nastavovat žádné položky proměnné **m**. Jako číslo linky **un** lze pro MPC400 zadat pouze jednu z hodnot {0;1}, hodnoty pak odpovídají volbě linky {L1;L2}, resp. volbě očekávaného ovladače {USART0;USART1}. Hodnota parametru **br** může být zadána libovolná z rozsahu 1000 až 230400. Parametr **p** zadáme 0 pro komunikaci bez parity, při hodnotě 1 bude každý znak doplněn paritním bitem pro dosažení sudé parity a při jiné hodnotě bude platit lichá parita bitů znaku. U komunikace bez parity budou automaticky nastaveny dva stop-bity ve znaku (namísto jednoho), znak je tak vždy 11-bitový. Parametr **tout** udává dobu v milisekundách, do níž je třeba přijmout alespoň první znak odpovědi, aby mohl být příkaz prohlášen za úspěšný - po uplynutí doby jinak dojde k ukončení příjmu a ohlášení timeoutu. Tuto dobu je v každém případě potřeba zadat delší, než je doba trvání jednoho znaku na lince s danou baudovou rychlostí. Parametr **ts** stanovuje minimální prodlevu mezi dokončením jedné transakce a zahájením další, v milisekundách.

```
var _modbusMA modM      ; založení proměnné strutury _modbusMA
......
ModbusMA_Config(modM,0,9600,1,50,10)     ; Modbus na lince L1, 9600 Bd,
                                         ; sudá parita, timeout 50 ms,
                                         ; min. doba klidu 10 ms
```

ModbusMA_STM

Deklarace :	**subroutine ModbusMA_STM(var _modbusMA m)**
Parametr 1 :	**m** proměnná se strukturou _modbusMA reprezentující linku RS485 s režimem komunikace Modbus RTU Master v programu
Výstup :	- - -

Procedura provádí obsluhu posledního Modbus příkazu (transakce) zadaného pomocí téže strukturované proměnné **m** a funkce ModbusMA_Send. Nastavováním hodnoty položky *.state* proměnné **m** informuje o průběhu příkazu. V případě přijetí odpovědi na zadaný příkaz tuto odpověď také vyhodnotí. Procedura by měla být volána jednou při každém průchodu programovou smyčkou. Před prvním voláním v programu je třeba zajistit, aby proběhlo nastavení parametrů komunikace na lince funkcí ModbusMA_Config. Význam nastavovaných hodnot *.state* je uveden u návodu k tvorbě programu.

ModbusMA_Send

Deklarace :	**function bit ModbusMA_Send(var _modbusMA m)**
Parametr 1 :	**m** proměnná se strukturou _modbusMA reprezentující linku RS485 s režimem komunikace Modbus RTU Master v programu předaná s požadovaným nastavením parametrů příkazu
Výstup :	**tst** hodnota indikující úspěch / neúspěch zadání transakce (příkazu)

Funkce pro zadání nového Modbus příkazu (transakce), lze ji volat kdekoli v programu. V případě, že předchozí transakce na lince již byla dokončena a nastavené parametry nového příkazu jsou správné, bude zahájena nová transakce a funkce vrátí hodnotu 1. V opačném případě vrací hodnotu 0. Před prvním voláním funkce v programu je třeba zajistit, aby proběhlo nastavení parametrů komunikace na lince funkcí ModbusMA_Config.

```
var _modbusMA modM      ; založení proměnné strutury _modbusMA
......
; zadáme příkaz pro čtení registrů 354 a 355 ze zařízení slave s adresou 2
modM.addrSL=2
modM.cmd=_MBCM_READ_HOLDING_REGISTERS
modM.nitems=2
modM.daddr=354
if (ModbusMA_Send(modM) = 0) then
      ......      ; nedokončena předchozí transakce nebo chyba v zadání
else
      ......      ; nový příkaz úspěšně zadán
```

3. MODBUS RTU SLAVE

Na lince RS485 automatu MPC400 i jednotky MEX400 lze snadno provozovat komunikaci protokolem Modbus RTU v režimu slave. Pro tuto funkci je třeba zvolit ovladačem linky ovladač MODBUS, poté již bude zařízení schopno na lince komunikovat autonomně. Možnosti ovladače u MPC400 mohou být ještě rozšířeny voláním podpůrných vestavěných funkcí v programu automatu. V dalším textu následuje popis parametrů ovladače komunikace Modbus RTU slave a popis podporovaných příkazů protokolu Modbus v tomto režimu. Také je uveden významu registrů a bitů, které automat na lince zpřístupňuje.

3.1. Parametry ovladače

Při volbě ovladače MODBUS uživatel také nastavuje jeho parametry. Těmi jsou parametry komunikace a číslo výchozí datové stránky (volba stránky se netýká jednotek MEX400). Význam datových stránek je popsán dále v dokumentu.

Komunikační rychlost

Protokol Modbus vyžaduje nastavení stejných komunikačních rychlostí u všech zařízení na lince. Ovladači MODBUS lze nastavit jednu z komunikačních rychlostí {9600Bd, 19200Bd, 38400Bd, 57600Bd, 115200Bd, 230400Bd}. Dále se nastavuje použití paritního bitu v komunikaci s možnostmi {bez parity, sudá parita, lichá parita}. Nastavení parity automaticky upravuje počet stop-bitů, při konfiguraci s paritou bude vysílán jeden stop bit, při konfigurace bez parity 2 stop-bity (vysílání znaku trvá vždy 11 bitových intervalů).

Adresa

Každý slave na sběrnici Modbus musí mít svou unikátní adresu, tu lze nastavit v rozsahu 1 až 247. Adresa 0 je broadcast adresa, adresy 248 až 255 se nepoužívají.

Prodleva před vysíláním odpovědi

U slave nastavujeme prodlevu mezi přijetím příkazu od mastera a vysíláním odpovědi na tento příkaz. Proto, aby mohlo být nastaveno požadované časování v případě komunikace v síti s různými zařízeními od různých výrobců. Lze nastavit jednu z hodnot {0ms; 5ms; 10ms; 20ms; 50ms; 100ms; 200ms; 500ms}.

3.2. Registry a bity přenášené po Modbus

Registry protokolu Modbus jsou 16-bitové (WORD). Kromě hodnot registrů lze přenášet také hodnoty bitů. V jednom rámci Modbus je možno přenést maximálně 125 hodnot registrů, nebo 2000 hodnot bitů. Resp. maximální počty přenášených hodnot u jednotlivých příkazů odpovídají hodnotám ve specifikaci protokolu Modbus. Příkazy Modbus lze adresovat až 65536 registrů a stejný počet bitů.

Pozn. V dalším textu je uvažováno číslování registrů a bitů od 0. Tedy např. registr 25 bude chápán jako šestadvacátý registr v pořadí.

Bitový přístup

Bitový prostor se vždy překrývá s prostorem registrů. Pole bitů má tak význam jednotlivých bitů registrů (položek WORD). Např. bit číslo 25 má význam bitu 9 (zbytek po dělení 25/16) registru 1 (celá část 25/16). Pokud ovladač poskytuje více než 4096 registrů, bude bitově přístupných pouze prvních 4096 registrů (celkem 65536 bitů).

3.3. Podporované příkazy Modbus

Podporovány jsou následující Modbus příkazy čtení a zápisu registrů:

0x03 - Read Holding Registers
0x04 - Read Input Register
0x06 - Write Single Register
0x10 - Write Multiple Registers
0x16 - Mask Write Register
0x17 - Read/Write Multiple Registers

a příkazy čtení a zápisu bitů:

0x01 - Read Coils
0x02 - Read Discrete Inputs
0x05 - Write Single Coil
0x0F - Write Multiple Coils

3.4. Význam Modbus registrů MEX400

Jednotka MEX zpřístupňuje přes Modbus parametry instalovaných IO modulů, disponuje tedy pouze registry číslo 0 až 95. Interpretace významu jednotlivých registrů je popsána v části „Mapování parametrů IO modulů na registry Modbus“.

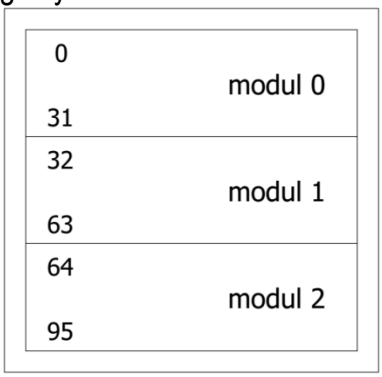

Fig_1 Modbus registry MEX400

3.5. Význam Modbus registrů MPC400

Automat zpřístupňuje přes Modbus parametry instalovaných IO modulů, dále zásobník (stack) a uživatelskou paměť. Kromě dostupnosti celé uživatelské paměti lze navíc v programu vyhradit pole wordů, které bude přes Modbus jako celek velmi snadno dostupné a bez rizika přístupu mimo něj.

Stránky

Automat vždy poskytuje Modbus registry 0 až 65535 (0x0000 až 0xFFFF). Volba významu Modbus registrů se provede nastavením spodního bajtu registru *MB_CTRL*. Registr je přístupný pro čtení a zápis jako registr číslo 65535, zapisovat ale lze jen spodní bajt. Současně s registrem již příkazem nelze číst ani zapisovat další jiný registr - např. čtením dvou registrů počínaje registrem 65534 hodnotu *MB_CTRL* nepřečteme, čtou se pouze dvě hodnoty z dané stránky. Číslo stránky (spodní bajt *MB_CTRL*) tedy udává, jaký obsah mají v dané chvíli Modbus registry automatu. Navíc 16. bit registru *MB_CTRL* indikuje, zda je zakázáno zpřístupnění obsahu stránky po Modbus. Všechny stránky jsou běžně povoleny k vyčítání, zákaz lze provést pouze voláním příslušné funkce v programu automatu. U zakázané stránky jsou hodnoty všech registrů a bitů vraceny nulové.

Nulové registry

Pokud Modbus registr (bit) nemá význam fyzické paměti, bude čten jako nulový a jeho zápis bude ignorován. Jako příklad poslouží stránka 2 s registry stacku (zásobníku). Stack má délku 11776 wordů, takže Modbus registry 0 až 11775 budou registry *StackW*, registry 11776 až 65535 pak budou nulové.

Obsah jednotlivých stránek

Stránka 0

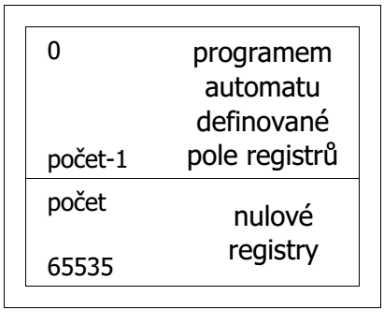

Fig_2 Modbus registry MPC400, stránka 0

Stránka zpřístupňuje přes Modbus registry, které uživatel v programu definoval a následně předal odkazem vestavěné funkci *ModbusSL_AssignRegs*. Pokud k volání přiřazovací funkce v programu nedojde, bude všech 65536 Modbus registrů nulových. Další upřesňující informace lze nalézt v sekci „Podpora v programu MPC400“.

Stránka 1

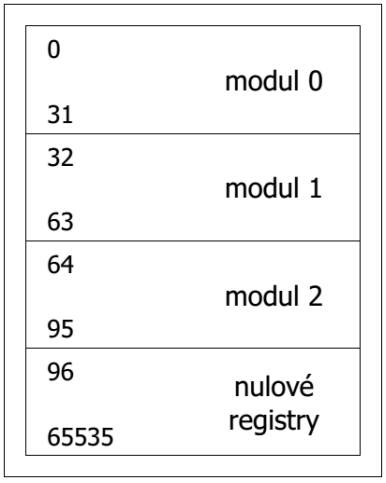

Fig_3 Modbus registry MPC400, stránka 1

Stránka zpřístupňuje přes Modbus parametry instalovaných IO modulů, ty lze nalézt v registrech číslo 0 až 95. Zbylé registry 96 až 65535 se nevyužívají. Interpretace významu registrů 0 až 95 na této stránce je popsána v části „Mapování parametrů IO modulů na registry Modbus“.

Stránka 2

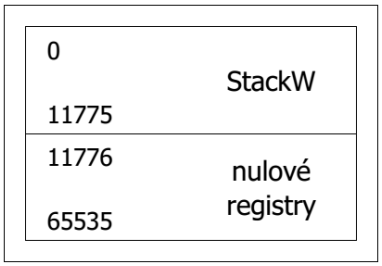

Fig_4 Modbus registry MPC400, stránka 2

Stránka zpřístupňuje přes Modbus celý zásobník automatu po wordech. Zásobník disponuje 11776 wordy, Modbus registry 0 až 11775 tedy odpovídají registrům *StackW*, registry 11776 až 65535 jsou pak nulové registry.

Stránky 3 až 255

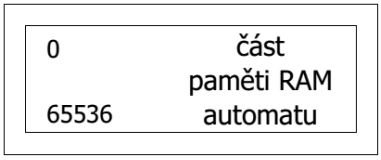

Fig_5 Modbus registry MPC400, stránky 3-255

Stránky zpřístupňují přes Modbus celou uživatelskou paměť (RAM). Slouží k možnosti přistupovat k polím wordů, případně jednotlivým wordům, fixovaným na libovolných bajtových adresách v paměti. Např. standardní požadavek Modbus mastera na četní tří registrů počínaje registrem 25 (číslování registrů je uvažováno od 0) bude ve skutečnosti ovladačem interpretován jako požadavek na čtení tří wordů od adresy 25 na stránce – parametr „číslo registru“ příkazu mastera tak má ve skutečnosti význam bajtové adresy uvnitř stránky.

Jednotlivé stránky mají velikost 65536 bajtů (tedy pouze 32768 wordů) a následující stránky se částečně překrývají, tak, že každý bajt (kromě prvních 32768) je přístupný vždy ze dvou následujících stránek. Tedy na stránce 3 lze přistupovat k wordům s počátečními fyzickými adresami 0 až 65534 ve skutečné paměti. A obecně na stránce N lze přistupovat k wordům s počátečními fyzickými adresami ((N-3)*32768) až ((N-3)*32768+65534) ve skutečné paměti.

Pozn.: Word s fyzickou adresou 65535 již ve stránce ve skutečnosti neleží, protože jeho horní bajt má adresu 65536, tedy mimo adresní rozsah stránky. Wordu tak bude interpretován jako nulový registr. Také každý word ležící mimo skutečnou paměť RAM automatu bude nulový. Např. u automatu s 84k paměti je reálně k dispozici 86608 bajtů. U tohoto automatu je tedy celá paměť dostupná přes stránky číslo 3 a 4. Na stránce 5 je ještě dostupný konec paměti (ten je i na stránce 4), zbylé registry stránky 5 a všechny registry stránek 6 až 255 již význam skutečné paměti nemají.

3.6. Mapování IO modulů na registry Modbus

Každý MEX400 nebo MPC400 může disponovat až 3 moduly, přičemž každý modul má pro přístup vyhrazeno 32 Modbus registrů. Registry 0-31 umožňují přístup k modulu 0, registry 32-63 přístup k modulu 1 a registry 64-95 přístup k modulu 2.

Modul lze popsat příslušnou datovou strukturou v jazyce SIMPLE4, která umožňuje práci se zapojeným modulem v programu automatu. Mapování na registry Modbus se pak odvodí ze znalosti této struktury. Modbus registry umožňují čtení hodnot, odpovídajících vstupním i výstupním parametrům modulu. Zapisovat lze pouze výstupní parametry. Parametry formátu BYTE, WORD a INT mají každý vyhrazen jeden Modbus registr. Parametry formátu LONGWORD, LONGINT a

FLOAT mají vyhrazeny 2 Modbus registry (první představuje horních 16 bitů hodnoty, druhý spodních 16 bitů). Princip přiřazení by měl ozřejmit následující příklad pro modul na pozici 1:

```
type struct
      BYTE p1,
      WORD p2,
      LONGWORD p3,
      BYTE p4,
      FLOAT p5,
      BYTE[28] nu ; nepoužívané bajty
end _modul_test
```

Položka *p1* pak bude přístupná přes Modbus registr 32, položka *p2* přes registr 33, položka *p3* přes registry 34 (horních 16 bitů) a 35 (spodních 16 bitů) , položka *p4* přes registr 36 a položka *p5* přes registry 37 (horních 16 bitů) a 38 (spodních 16 bitů). Registry 39-63 budou nulové, tj. budou čteny jako nulové a jejich zápis bude ignorován.

3.7. Podpora v programu MPC400

MPC400 podporuje dvě vestavěné funkce ovladače MODBUS. Jedna funkce zpřístupní po Modbus uživatelem vyhrazené pole registrů. Registry pak budou dostupné pro externí zařízení typu master přes stánku 0 ovladače. Použitím druhé vestavěné funkce je možno zamezit v přístupu k vybraným stránekám přes Modbus. Všechny registry zakázané stránky vyjma registru *MB_CTRL* (registru číslo 65535 pro výběr stránky) pak budou bez významu, tj. nulové registry.

Pole Modbus registrů

Polem registrů Modbus může být libovolná struktura v uživatelské paměti PLC, pouze počet bajtů této struktury musí být sudý. Ve většině případů si v programu nejspíše vystačíme s polem WORDů. Např. při deklaraci pole

```
var word[100] reg
```

a následném předání tohoto pole zmíněné vestavěné funkci bude Modbus registru číslo 25 na stránce 0 fyzicky odpovídat registr *reg[25].* Modbus bitu číslo 25 na stránce 0 bude odpovídat bit *reg[1]?9,* neboli *bit[(zbytek po dělení 25/16)]* registru *reg[(celá část 25/16)]*. Číslování Modbus registrů i bitů je uvažováno od 0.

Funkce zpřístupnění pole Modbus registrů

+ ModbusSL_AssignRegs

Deklarace :	**function byte ModbusSL_AssignRegs(byte n, var dataptr preg)** **function byte ModbusSL_AssignRegs(byte n, dataptr preg)**
Parametr 1 : 2 :	**n** číslo linky RS485 s ovladačem MODBUS **preg** ukazatel na registry poskytované automatem přes linku na stránce 0 ovladače
Výstup :	**err** kód chyby

Vestavěná funkce zpřístupňující uživatelem definované pole (strukturu) přes Modbus při použití ovladače MODBUS linky RS485 u MPC400. V programu ji stačí volat jednou, po resetu automatu. Parametr **n** udává číslo ovladače, je třeba zadat 0 při použití ovladače na lince L1, nebo 1 při použití ovladače na lince L2. Funkce vrací hodnotu 0, pokud předání proběhlo v pořádku. Jinak vrací 255, tj. požadovaný ovladač MODBUS není nastaven, příp. ukazatel *preg* neukazuje na strukturu se sudým počtem bajtů.

```
var word[100] mreg            ; založení pole registrů
; 1. způsob:
ModbusSL_AssignRegs(0,@mreg)  ; předání registrů ovladači MODBUS linky L1
; 2. způsob:
var dataptr ptr               ; založení proměnné typu dataptr
ptr = @mreg                   ; naplnění proměnné dataptr
ModbusSL_AssignRegs(1,ptr)    ; předání registrů ovladači MODBUS linky L2
```

Pozor! Funkci je možno použít až v MPC400 s verzí firmware 5.034 a vyšší, se starším firmware nepoužívat.

Funkce zabránění přístupu ke stránkám registrů Modbus

+ ModbusSL_DisablePages

Deklarace :	**function byte ModbusSL_AssignRegs(byte n, byte spg, byte epg)**	
Parametr 1 :	**n**	číslo linky RS485 s ovladačem MODBUS
2 :	**spg**	číslo první stránky z rozsahu stránek
3 :	**epg**	číslo poslední stránky z rozsahu stránek
Výstup :	**err**	kód chyby

Tato vestavěná funkce MPC400 ruší přístup k registrům na definovaném rozsahu stránek ovladače, tzn. všechny registry na stránce pak budou interpretovány jako nulové registry. Parametr **n** udává číslo ovladače, je třeba zadat 0 při použití ovladače MODBUS na lince L1, nebo 1 při použití ovladače na lince L2. Parametry **spg** a **epg** udávají počáteční a koncové číslo stránky tak, že definují rozsah stránek, který má být přes Modbus nepřístupný. Funkce vrací 0, pokud nastavení proběhlo v pořádku, jinak funkce vrací 255 (požadovaný ovladač MODBUS není nastaven), případně 254 (**spg** je větší **epg**). Funkci lze zavolat vícekrát (nejlépe po resetu automatu) s ruznými parametry pro zadání různých rozsahů stránek. Pokud je u ovladače právě aktivní stránka zakázána k vyčítání (číslo stránky udává hodnota spodního bajtu registru *MB_CTRL*), bude také 16. bit registru *MB_CTRL* v hodnotě 1.

```
if (RESET) then
      ModbusSL_DisablePages(1,3,255)      ; zákaz přístupu na stránky 3 až 255
                                          ; (tedy k celé RAM MPC400) externím
                                          ; Modbus masterem na lince L2
```