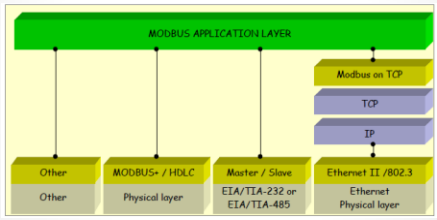
MODBUS

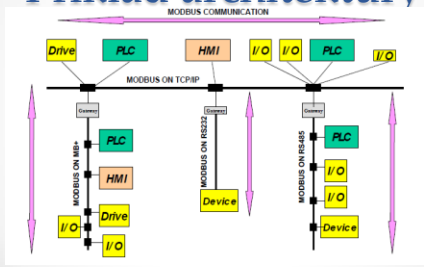
Základná charakteristika

- Mod Bus je vysokoúrovňový protokol definovaný na vrstve 7 (aplikačná vrstva)
- Princíp komunikácie požiadavka/odpoveď (klient=master/server=slave)
- Komunikáciu vždy inicializuje klient (master)
- Nezávisí od fyzického média
- Umožňuje jednoduchú komunikáciu medzi systémami s rôznou fyzickou vrstvou
- 3 typy implementácie:
  - Sériové rozhrania (RS485, 422, optika, ...)
  - TCP/IP (rezervovaný port 502)
  - Vysokorýchlostná aplikácia (Modbus plus)
- Vznik 1979, v súčasnosti veľmi rozšírený pre senzory, meracie moduly a podobné zariadenia
- www.modbus.org, http://www.csimn.com/CSI_pages/Modbus101.html

Implementácia

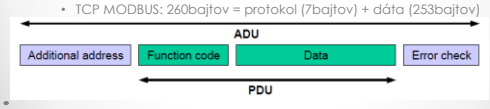


Príklad architektúry



Protokol

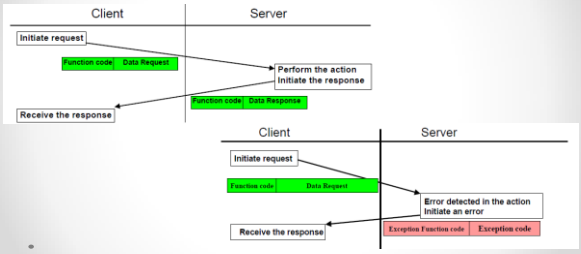
- Definuje sa:
  - Protocol data unit (PDU) nezávislá od nižších komunikačných vrstiev
  - Application data unit (ADU) predstavujúca aplikáciu PDU na konkrétne prenosové médium
  - Maximálna dĺžka ADU
    - Pre sériové prenosové médium: 256bajtov = adresa (1bajt) + CRC (2bajty) + dáta (253bajtov)
    - TCP MODBUS: 260bajtov = protokol (7bajtov) + dáta (253bajtov)



Protokol

- Komunikáciu aktivuje klient (master) posielajúci požiadavku serveru (klient)
- Function code identifikuje typ požiadavky (1 byte, 0 sa nepoužíva, používané 1 až 127, 128 až 255 rezervované) – určuje, čo má server urobiť
- Data posielané z klienta predstavujú pomocnú informáciu používanú serverom pri realizácii požiadavky klienta, napr. adresy registrov, počet položiek, ktoré majú byť spracovávané a počet bajtov v dátovom poli
  - Dátové pole môže byť aj prázdne (nulová dĺžka)
- Ak server príjme bezchybnú požiadavku vráti odpoveď, kde:
  - Kód funkcie je opakovaním kódu požadovaného klientom
  - dátové pole obsahuje dáta žiadané klientom.
- Ak je požiadavka chybná, server vráti kód funkcie nazývaný „exception code" = kód funkcie + 80H

Modbus komunikácia

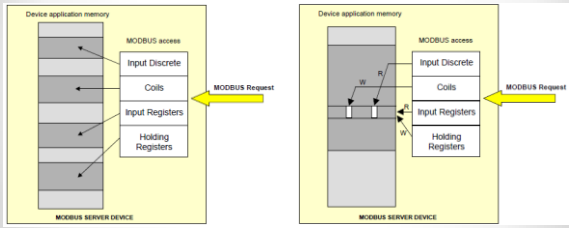


Dátový model

- Dáta sú u servra uložené ako série tabuliek (registrov)
- Každá tabuľka môže obsahovať maximálne 65536 položiek
- Tabuľky sa môžu vzájomne prekrývať
- Základné typy tabuliek:

Primary tables	Object type	Type of	Comments
Discretes Input	Single bit	Read-Only	This type of data can be provided by an I/O system.
Coils	Single bit	Read-Write	This type of data can be alterable by an application program.
Input Registers	16-bit word	Read-Only	This type of data can be provided by an I/O system
Holding Registers	16-bit word	Read-Write	This type of data can be alterable by an application program.

Dátový model - príklady



Kódy funkcií 1

- Read Coils (0x01) -1až 2000 stavových bitov (coils)
  - N=Quantity/8
  - Hodnota coil na Start. Address je LSB v 1. bajte

Request

Function code	1 Byte	**0x01**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of coils	2 Bytes	1 to 2000 (0x7D0)

Response

Function code	1 Byte	**0x01**
Byte count	1 Byte	**N***
Coil Status	n Byte	n = N or N+1

Request		Response	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	01	Function	01
Starting Address Hi	00	Byte Count	03
Starting Address Lo	13	Outputs status 27-20	CD
Quantity of Outputs Hi	00	Outputs status 35-28	6B
Quantity of Outputs Lo	13	Outputs status 38-36	05

Kódy funkcií 2

- Read Discrete Inputs (0x02) -1až 2000 bitových vstupov

Request

Function code	1 Byte	**0x02**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Inputs	2 Bytes	1 to 2000 (0x7D0)

Response

Function code	1 Byte	**0x02**
Byte count	1 Byte	**N***
Input Status	**N*** x 1 Byte	

***N** = Quantity of Inputs / 8 if the remainder is different of 0 ⇒ N = N+1

Request		Response	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	02	Function	02
Starting Address Hi	00	Byte Count	03
Starting Address Lo	C4	Inputs Status 204-197	AC
Quantity of Inputs Hi	00	Inputs Status 212-205	DB
Quantity of Inputs Lo	16	Inputs Status 218-213	35

Kódy funkcií 3

- Read Holding Registers(0x03) -1až 125 registrov

Request

Function code	1 Byte	**0x03**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Registers	2 Bytes	1 to 125 (0x7D)

Response

Function code	1 Byte	**0x03**
Byte count	1 Byte	2 x **N***
Register value	**N*** x 2 Bytes	

***N** = Quantity of Registers

Request		Response	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	03	Function	03
Starting Address Hi	00	Byte Count	06
Starting Address Lo	6B	Register value Hi (108)	02
No. of Registers Hi	00	Register value Lo (108)	2B
No. of Registers Lo	03	Register value Hi (109)	00
		Register value Lo (109)	00
		Register value Hi (110)	00
		Register value Lo (110)	64

Kódy funkcií 4

- Read Input Registers(0x04)-1až 125 registrov

Request

Function code	1 Byte	0x04
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Input Registers	2 Bytes	0x0001 to 0x007D

Response

Function code	1 Byte	0x04
Byte count	1 Byte	2 x N*
Input Registers	N* x 2 Bytes	

*N = Quantity of Input Registers

Request		Response	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	04	Function	04
Starting Address Hi	00	Byte Count	02
Starting Address Lo	08	Input Reg. 9 Hi	00
Quantity of Input Reg. Hi	00	Input Reg. 9 Lo	0A
Quantity of Input Reg. Lo	01		

Kódy funkcií 5

- Write Single Coil(0x05)
  - 0xFF00 v dátach znamená ON
  - 0x0000 znamená OFF

Request

Function code	1 Byte	0x05
Output Address	2 Bytes	0x0000 to 0xFFFF
Output Value	2 Bytes	0x0000 or 0xFF00

Response

Function code	1 Byte	0x05
Output Address	2 Bytes	0x0000 to 0xFFFF
Output Value	2 Bytes	0x0000 or 0xFF00

Request		Response	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	05	Function	05
Output Address Hi	00	Output Address Hi	00
Output Address Lo	AC	Output Address Lo	AC
Output Value Hi	FF	Output Value Hi	FF
Output Value Lo	00	Output Value Lo	00

Kódy funkcií 6

- Write Single Register(0x06)

Request

Function code	1 Byte	0x06
Register Address	2 Bytes	0x0000 to 0xFFFF
Register Value	2 Bytes	0x0000 to 0xFFFF

Response

Function code	1 Byte	0x06
Register Address	2 Bytes	0x0000 to 0xFFFF
Register Value	2 Bytes	0x0000 to 0xFFFF

Request		Response	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	06	Function	06
Register Address Hi	00	Register Address Hi	00
Register Address Lo	01	Register Address Lo	01
Register Value Hi	00	Register Value Hi	00
Register Value Lo	03	Register Value Lo	03

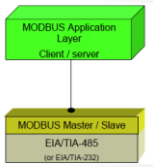
Kódy funkcií 7

- (0x07) Read Exception Status (Serial Line only)
- (0x08) Diagnostics (Serial Line only)
- (0A Hex) Clear Counters and Diagnostic Register
- (0B Hex) Return Bus Message Count
- ...

MODBUS na sériovej linke

- OSI model

Layer	ISO/OSI Model	
7	Application	MODBUS Application Protocol
6	Presentation	Empty
5	Session	Empty
4	Transport	Empty
3	Network	Empty
2	Data Link	MODBUS Serial Line Protocol
1	Physical	EIA/TIA-485 (or EIA/TIA-232)

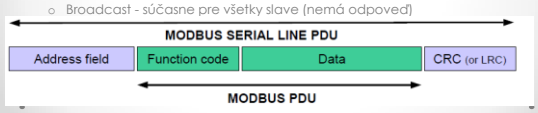


Charakteristiky

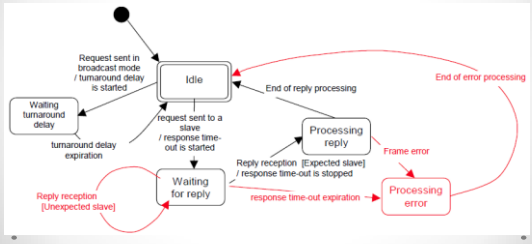
- Slave majú unikátnu adresu od 1 po 247

0	From 1 to 247	From 248 to 255
Broadcast address	Slave individual addresses	Reserved

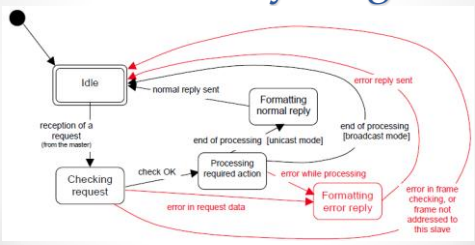
- Dva typy vysielania:
  - Unicast - pre jediny slave
  - Broadcast - súčasne pre všetky slave (nemá odpoveď)



Master - stavový diagram



Slave - stavový diagram



RTU prenosový mód

- RTU móde - hexa kód, prvý bit LSB, párna parita

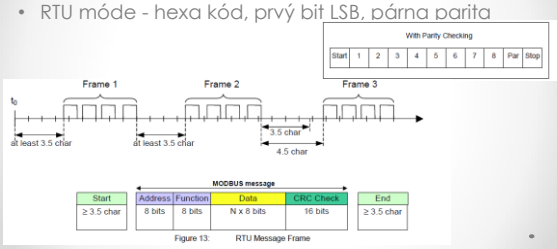
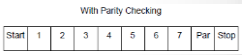


Figure 13: RTU Message Frame

ASCII mód

- Každých 8 bitov správy sa posiela ako dva ASCII znaky (0-9, A-F)
- Správa začína vždy znakom :
- Správa končí delimiterom, štandardne CRLF (0x0D0x0A)



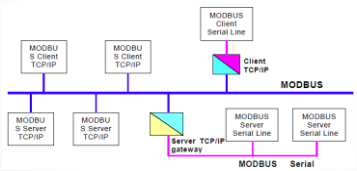
Start	Address	Function	Data	LRC	End
1 char :	2 chars	2 chars	0 up to 2x252 char(s)	2 chars	2 chars CR,LF

CRC a LRC

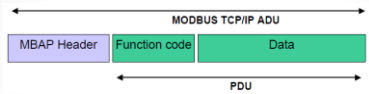
- CRC sa používa pre RTU
  - POLY = calculation polynomial of the CRC 16 = 1010 0000 0000 0001
  - (Generating polynomial = 1 + x2 + x 15 + x 16)
- LCR je v prípade ASCII
  - Add all bytes in the message, excluding the starting 'colon' and ending CRLF. Add them into an 8–bit field, so that carries will be discarded.
  - Subtract the final field value from FF hex (all 1's), to produce the ones–complement.
  - Add 1 to produce the twos–complement.

MODBUS na TCP/IP

- Klient a server sú prepojení cez komunikáciu postavenú na TCP/IP (Internet cez Ethernet, ...)



Dáta



- MBAP (MODBUS Application Protocol header)
  - Doplnkové info k infu pre sériové linky ako je dĺžka správy + fixné identifikátory, že ide o MODBUS protokol
- Neobsahuje CRC/LRC (ochrana je daná priamo TCP/IP protokolom

MBAP

Fields	Length	Description -	Client	Server
Transaction Identifier	2 Bytes	Identification of a MODBUS Request / Response transaction.	Initialized by the client	Recopied by the server from the received request
Protocol Identifier	2 Bytes	0 = MODBUS protocol	Initialized by the client	Recopied by the server from the received request
Length	2 Bytes	Number of following bytes	Initialized by the client (request)	Initialized by the server (Response)
Unit Identifier	1 Byte	Identification of a remote slave connected on a serial line or on other buses.	Initialized by the client	Recopied by the server from the received request

SOAP

Základná charakteristika

- SOAP je skratka z Simple Object Access Protocol
- Používa sa pre prenos dát napr. v senzorových sieťach
- Využíva Internet, presnejšie http/https protokol nad ktorým je postavený (aplikačná vrstva) ale je ho možné aplikovať aj nad iné protokoly ako ftp, ...
- Dáta sa prenášajú s využitím XML kódovania ale môžu byť kódované inak, napr. JSON
- Princíp klient/server, aktivita vždy od klienta
- SOAP môže byť využitý aj na diaľkové volanie rôznych procedúr
- Server je postavený na tzv. Web services

Formát správy

- Obálka (Envelope) definuje začiatok a koniec správy a je povinná
- Hlavička(Header)- obsahuje nepovinné ľubovoľné doplnkové info k správe, ktoré sa používa pri spracovaní správy
- Telo správy (Body) - samotné dáta v XML formáte a je povinné
- Chyba (Fault) - nepovinné pole s informáciou o chybe, ktorá vznikla pri spracovaní správy

Envelope

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope" SOAP-ENV:encodingStyle="
http://www.w3.org/2001/12/soap-encoding">
...
  Message information goes here
...
</SOAP-ENV:Envelope>
```

Príklad s POST

```
POST /OrderEntry HTTP/1.1
Host: www.tutorialspoint.com
Content-Type: application/soap; charset="utf-8"
Content-Length: nnnn
<?xml version="1.0"?>

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope" SOAP-ENV:encodingStyle="
http://www.w3.org/2001/12/soap-encoding">
...
  Message information goes here
...
</SOAP-ENV:Envelope>
```

Header

```
<?xml version="1.0"?>

<SOAP-ENV:Envelope xmlns:SOAP-ENV=" http://www.w3.org/2001/12/soap-envelope" SOAP-ENV:encodingStyle="
http://www.w3.org/2001/12/soap-encoding">

  <SOAP-ENV:Header>
    <t:Transaction xmlns:t="http://www.tutorialspoint.com/transaction/" SOAP-ENV:mustUnderstand="true">5</t:Transaction>
  </SOAP-ENV:Header>
  ...
  ...
</SOAP-ENV:Envelope>
```

Ak je použitý nepovinný parameter ***SOAP-ENV:mustUnderstand="true"***, prijímateľ správy musí porozumieť hlavičke a korektne ju spracovať

Body

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
  .....
  <SOAP-ENV:Body>
    <m:GetQuotation xmlns:m="http://www.tp.com/Quotation">
      <m:Item>Computers</m:Item>
    </m:GetQuotation>
  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Odpoveď

```
<?xml version="1.0"?>

<SOAP-ENV:Envelope
.....
  <SOAP-ENV:Body>
    <m:GetQuotationResponse xmlns:m="http://www.tp.com/Quotation">
      <m:Quotation>This is Qutation</m:Quotation>
    </m:GetQuotationResponse>
  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Fault

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode xsi:type="xsd:string">SOAP-ENV:Client</faultcode>

      <faultstring xsi:type="xsd:string">
        Failed to locate method (ValidateCreditCard) in class
(examplesCreditCard) at /usr/local/ActivePerl-
5.6/lib/site_perl/5.6.0/SOAP/Lite.pm line 1555.
      </faultstring>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Namiesto alebo spolu s FAULT sa môže použiť aj kódovanie v rámci http (stavový kód 200 - 299 znamená úspešné doručenie a spracovanie, 500-599 chybu)

Sub-element	Description
<faultCode>	It is a text code used to indicate a class of errors. See the next Table for a listing of predefined fault codes.
<faultString>	It is a text message explaining the error.
<faultActor>	It is a text string indicating who caused the fault. It is useful if the SOAP message travels through several nodes in the SOAP message path, and the client needs to know which node caused the error. A node that does not act as the ultimate destination must include a *faultActor* element.
<detail>	It is an element used to carry application-specific error messages. The detail element can contain child elements called detail entries.

Kódovanie a prenos

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:getPriceResponse xmlns:ns1="urn:examples:priceservice" SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
      <return xsi:type="xsd:double">54.99</return>
    </ns1:getPriceResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
</SOAP-ENV:Envelope>
```

Viac o kódovaní pozri:
http://schemas.xmlsoap.org/soap/encoding/
http://www.w3.org/2001/12/soap-encoding

Pre prenos sa používa hlavne GET a POST pri http:

- GET umožňuje získať dáta od servra
- POST odoslať dáta na server