Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Information systems - µC

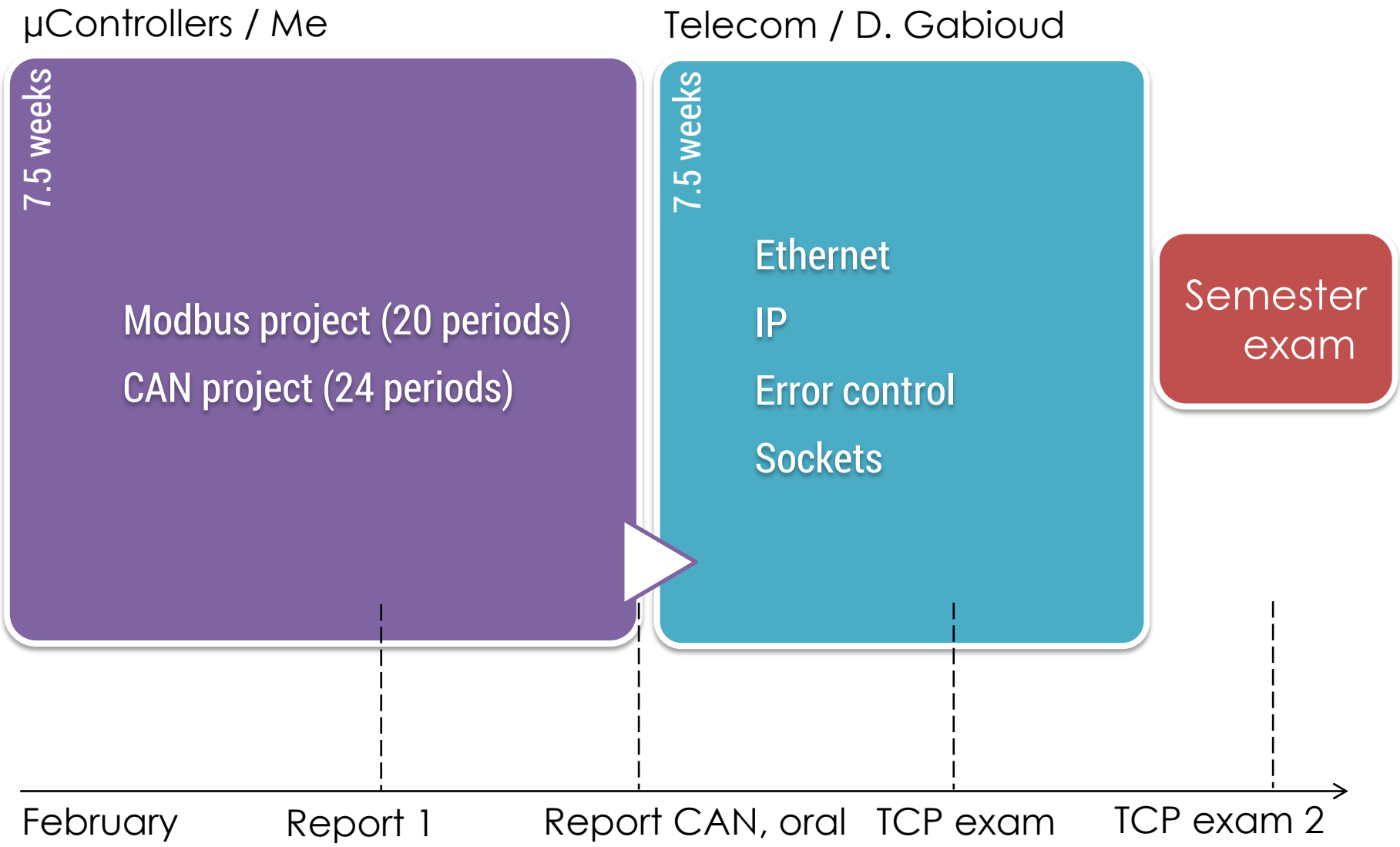
10. Modbus protocol

Dr Pierre-André Mudry
pierre-andre.mudry@hevs.ch

Rev 1.12

Disclaimer: those slides are based on the work of J. Corre / P. Sartoretti / D. Gabioud

Course content, 2nd semester



Lesson's objectives

Understand the Modbus protocol

1. General description
2. Transporting values
3. Modbus serial

One must start somewhere

1. GENERAL DESCRIPTION

Introduction to Modbus

Communication protocol for field devices and controllers

- Field devices: PLCs, meters, PV inverters…
- Controllers: data acquisition systems, PC, HMIs…

Modbus controllers can read and write data into field devices

Low complexity and low performance

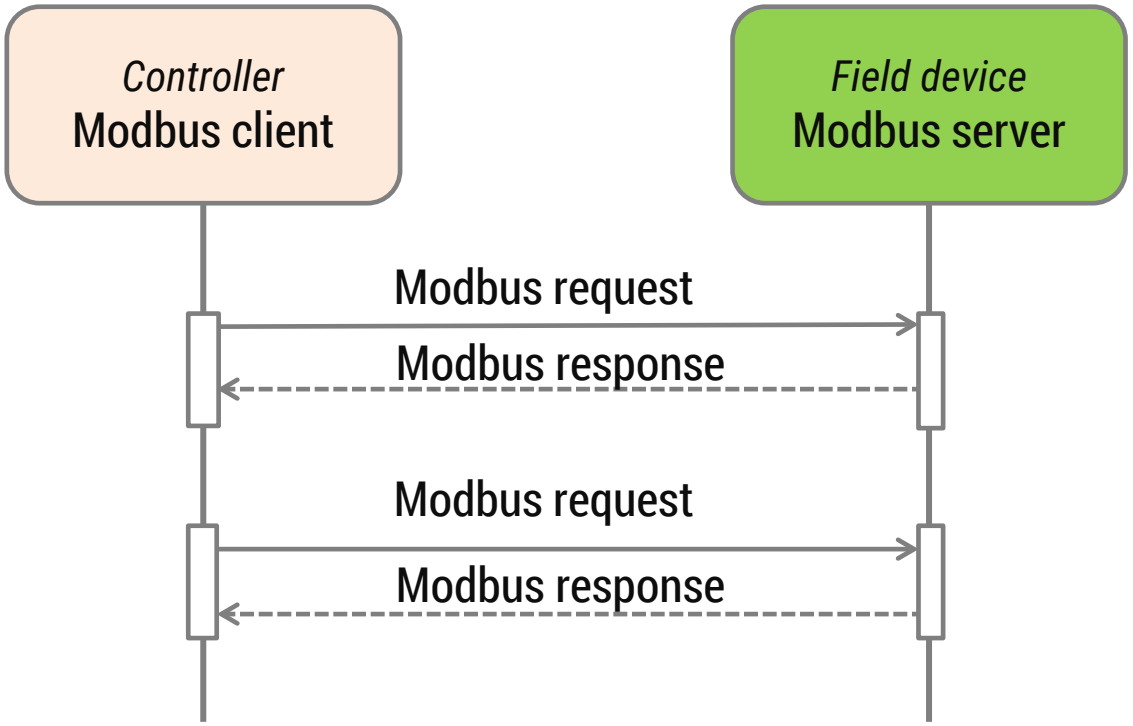
Modbus communication model

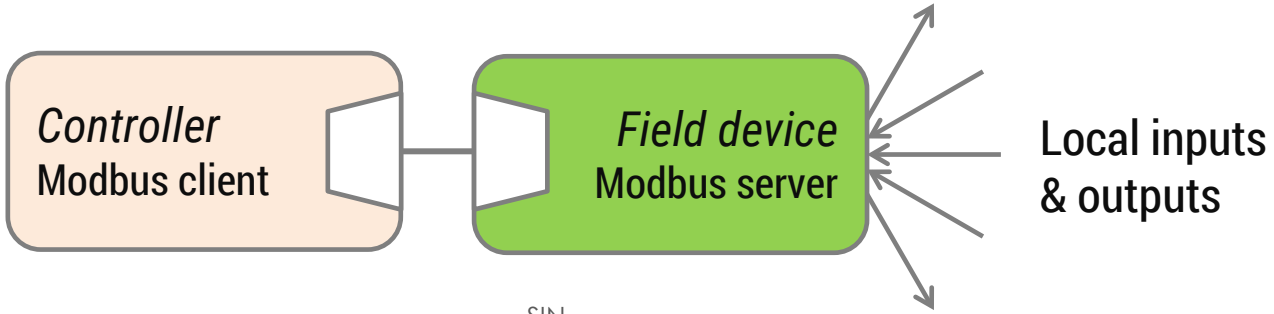
Client-server
communication model

Client: Controller

Server: Field device with local input / outputs

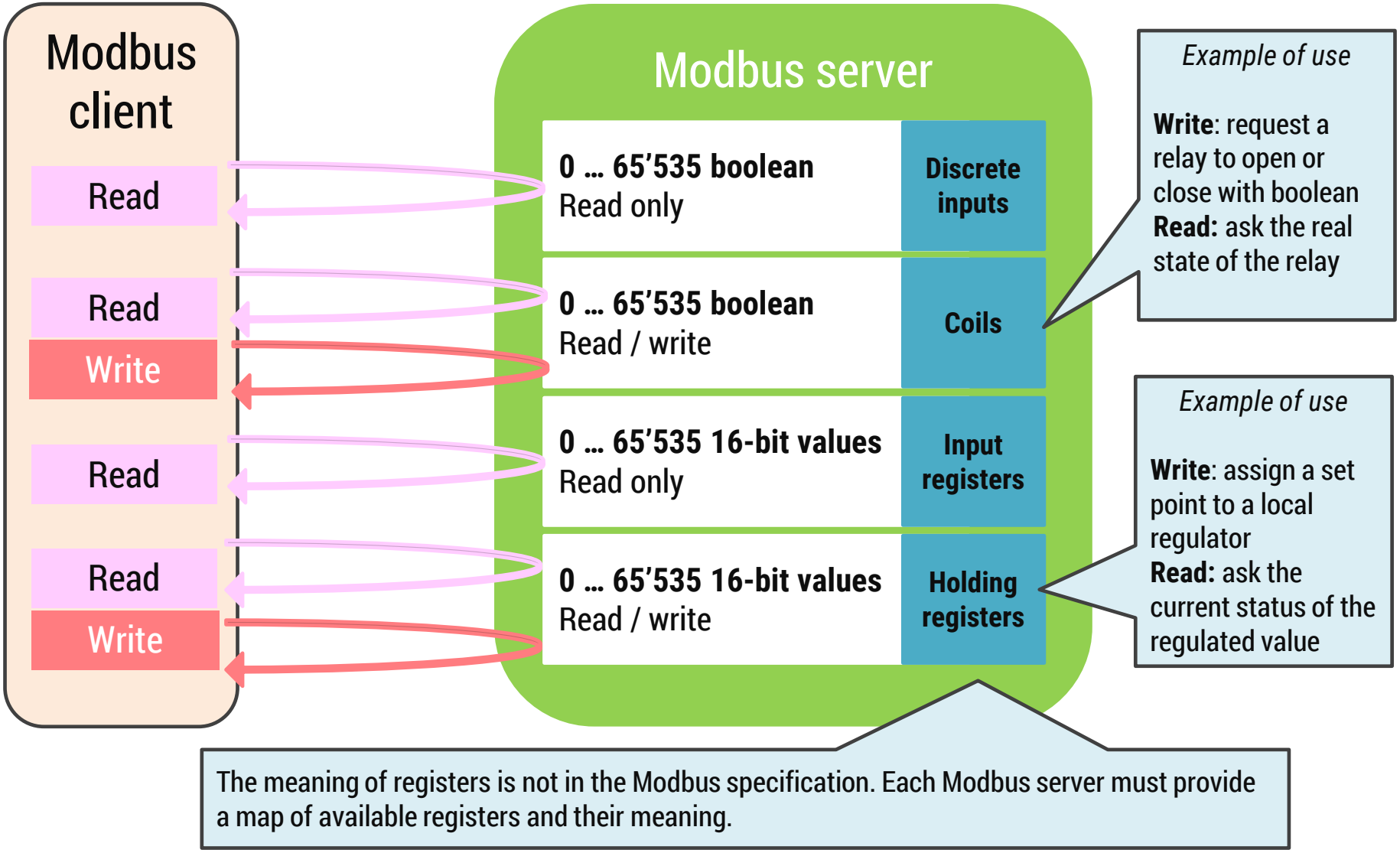
Via Modbus, a client can remotely read inputs and write outputs in the server





Modbus data model

Data and addresses



A real example : my heat pump

Register	ID	Name	TypeName	FunctionGroup	Unit	Min. value	Max. value	Writable
1	14	TKO1 collector temperature	S16	Collector	°C	-300	3000	No
2	14	TKO2 collector temperature	S16	Collector	°C	-300	3000	No
3	2034	Total yield collector_high	S32	Collector	kWh	0	0	No
4	2034	Total yield collector_low	S32	Collector	kWh	0	0	No
5	2030	Current collector output	S16	Collector	kWh	0	32767	No
6	2032	Operating hours pump Solar	U16	Collector	H	0	65535	No
7	2032	Operating hours pump Solar	U16	Collector	H	0	65535	No
8	2034	Total yield collector_high	S32	Collector	kWh	0	0	No
9	2034	Total yield collector_low	S32	Collector	kWh	0	0	No
10	2030	Current collector output	S16	Collector	kWh	0	32767	No
…								
9756	50	Current energy cold_high	U32	MBUS	MWh	0	0	No
9757	50	Current energy cold_low	U32	MBUS	MWh	0	0	No
9758	51	Current output cold_high	U32	MBUS	kW	0	0	No
9759	51	Current output cold_low	U32	MBUS	kW	0	0	No
9760	2	Current flow rate_high	U32	MBUS	l/h	0	0	No
9761	2	Current flow rate_low	U32	MBUS	l/h	0	0	No
9762	3	Current volume_high	U32	MBUS	M3	0	0	No
9763	3	Current volume_low	U32	MBUS	M3	0	0	No
9764	4	Current flow temperature	S16	MBUS	°C	0	0	No
9765	5	Current return temperature	S16	MBUS	°C	0	0	No

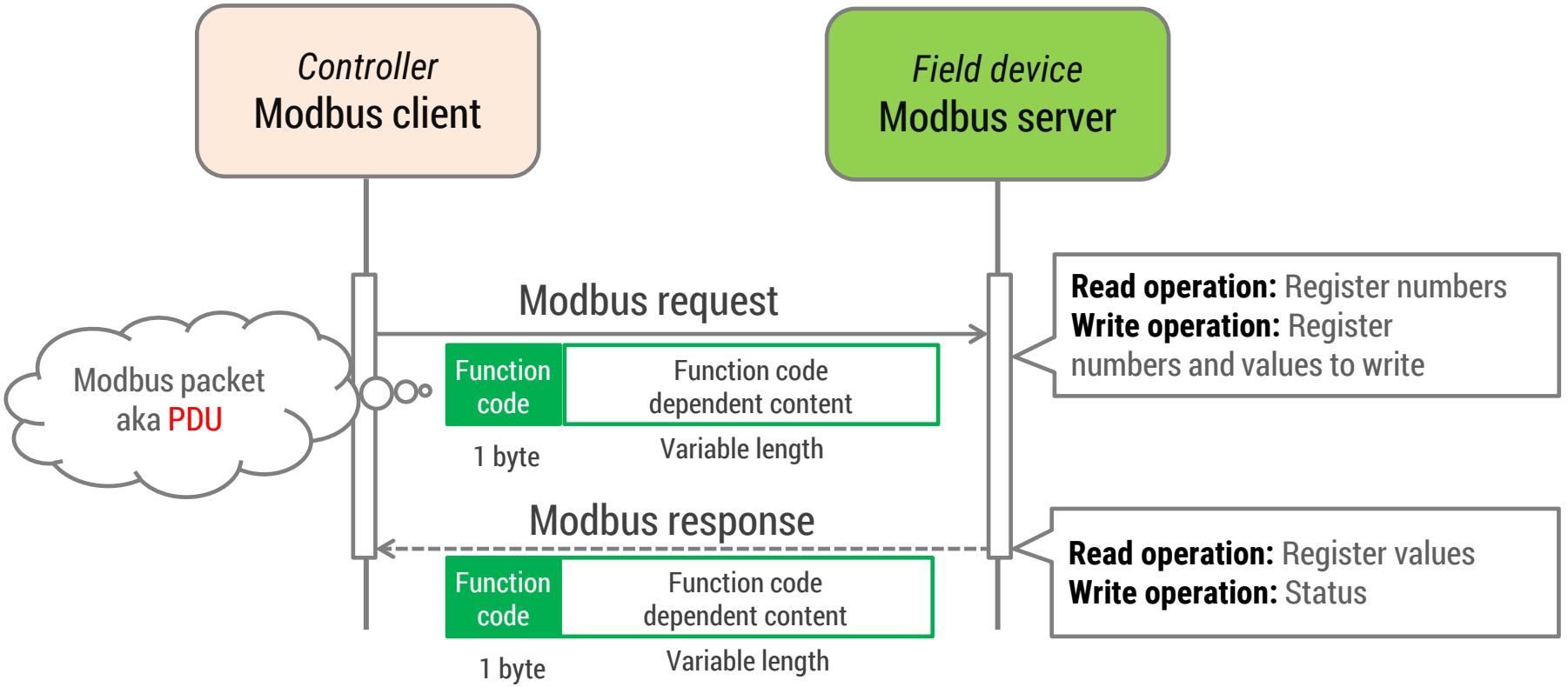
Modbus functions

Function		Object Type	Function Code
Boolean / bit access	Read	Discrete Inputs	0x02
	Read	Coils	0x01
	Write	Single Coil	0x05
	Write	Multiple Coils	0x0F
16-bit access	Read	Input Registers	0x04
	Read	Holding Registers	0x03
	Write	Single Holding Register	0x06
	Write	Multiple Holding Registers	0x10
	Read / Write*	Multiple Holding Registers	0x17

* The write operation is performed before the read

Modbus protocol

Registers are officially numbered between 1 and 65'536, but in PDUs are between 0 and 65'535.



Binary protocol (like IP or TCP), not text-based protocol !
For PDU format, see document *MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b*

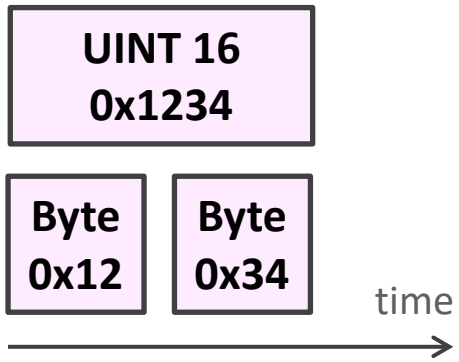
On the importance of types

2. VALUES ENCODING

Integer number transport

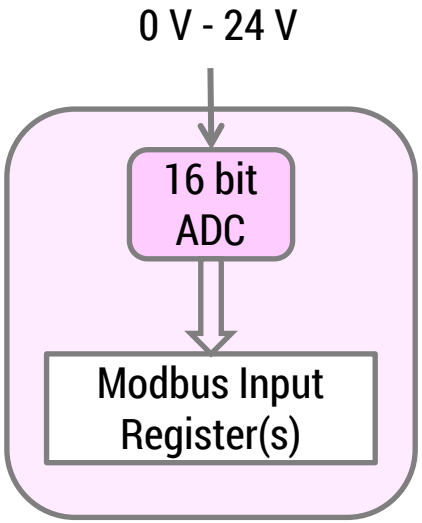
- For 16-bit Input and Holding registers, Modbus specifies that the Most Significant Byte is transmitted first

- Called big endian order, sometimes also called network order



Analog values on Modbus (I)

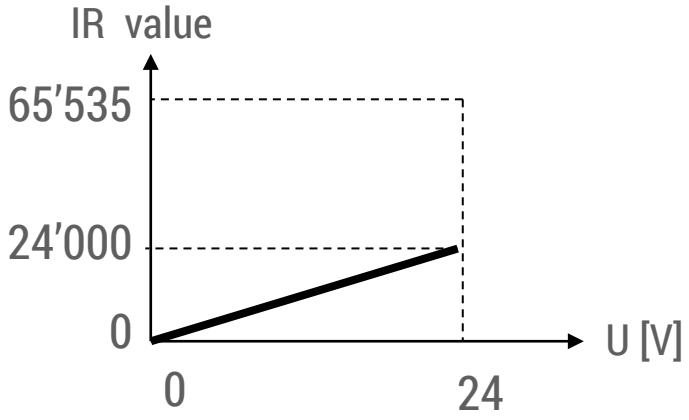
- Assume a Modbus server device is measuring a voltage in the range [0..24] Volt using an analog to digital converter (ADC)



- How to code the digitized voltage in one (or several) Modbus Input Register(s)?
  - Method 1: Encode the mV into a single Input Register IR

Voltage	Input Register
0 V	0
24 V	24'000

IR = (voltage [mV])



Limits: Loss of precision
Only positive values
Measured values < 65'535

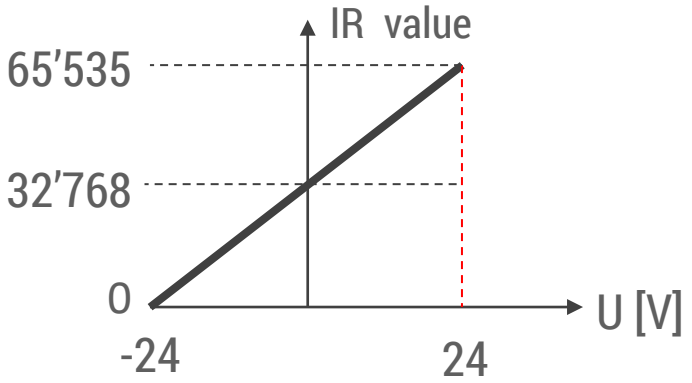
Analog values on Modbus (II)

- Assume now that we measure a voltage in the range [-24 V, 24 V]
  - Method 2: Use a range and offset parameter

IR = {(voltage [V] + *offset*) / *range*} * 65'535

With: *offset* = 24 V, *range* = 48 V

Voltage	Input Register
-24 V	0
24 V	65'535



The full Input Register value range is now used

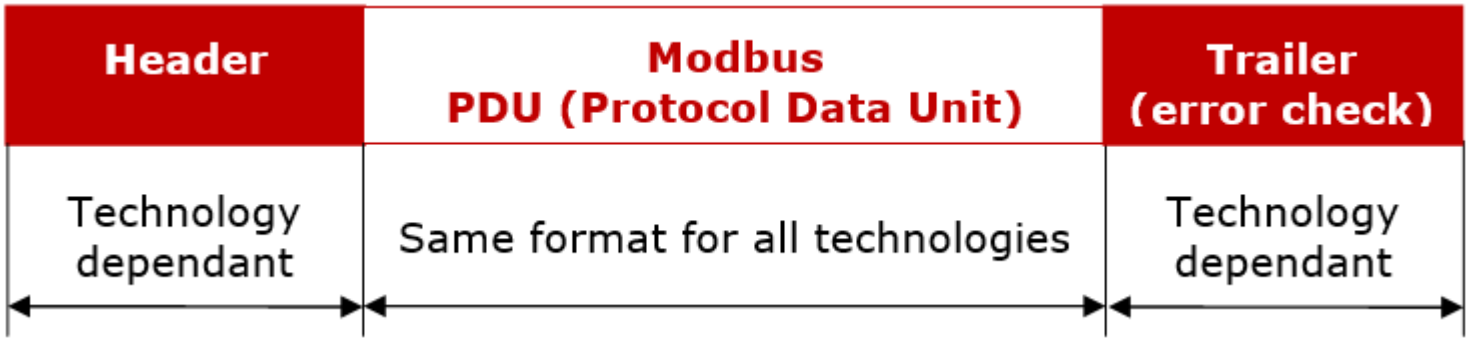
Limits: You need to define (in a user manual) an offset and a range for each measured value

RS-232 implementation of Modbus

3. SERIAL LINE MODBUS

Serial line Modbus frames

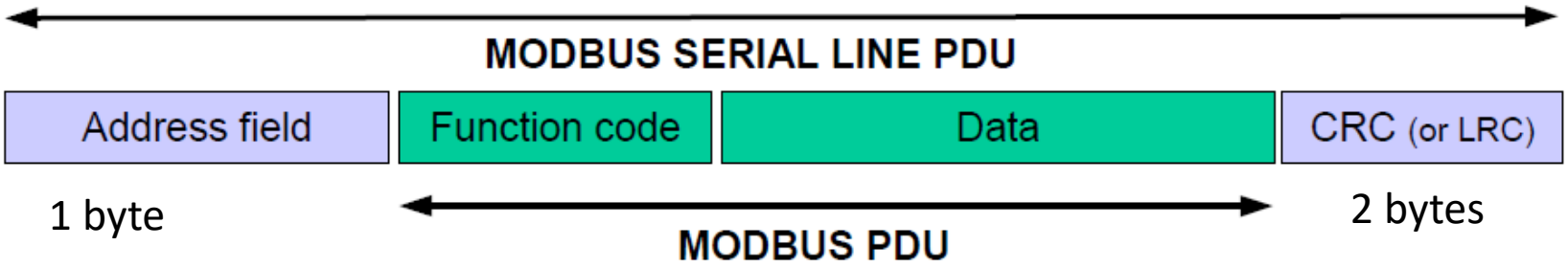
- Modbus frames are made of three parts
  - Header: technology dependent (RS232 not the same as TCP/IP)
  - PDU: Same for all technology, request (client to server) and answer (server to client)
  - Trailer: technology dependent (RS232 not the same as TCP/IP)



Serial line Modbus header / trailer

- Modbus header in serial line
  - Slave address: One byte defining the slave address (1 to 247, others are reserved values, 0 means multicast)

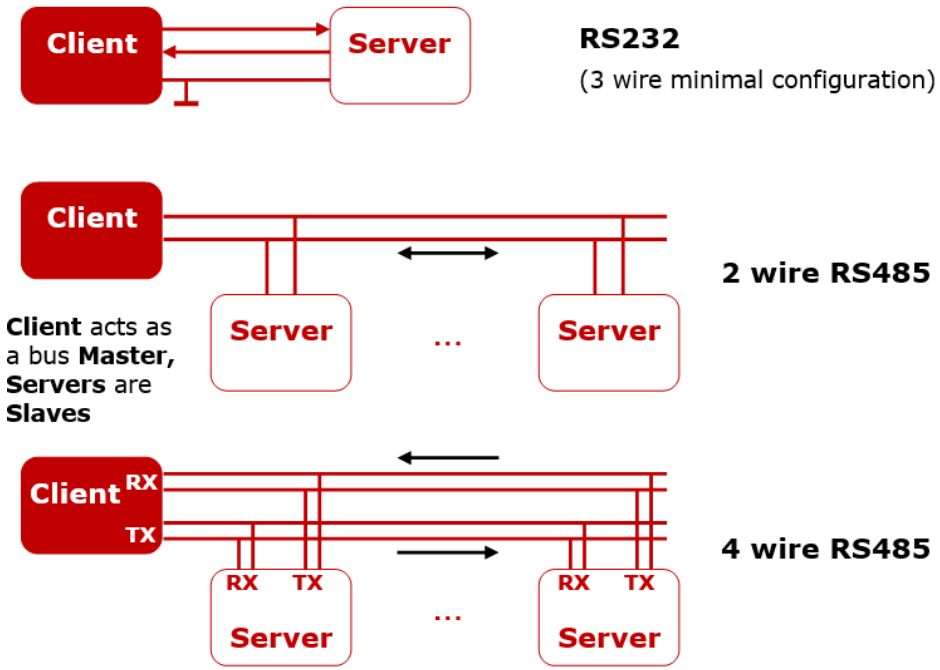
- Modbus trailer in serial line
  - CRC: Two bytes CRC computed on the header and PDU
  - Caution, CRC is LSB first



Serial line Modbus frames (physical)

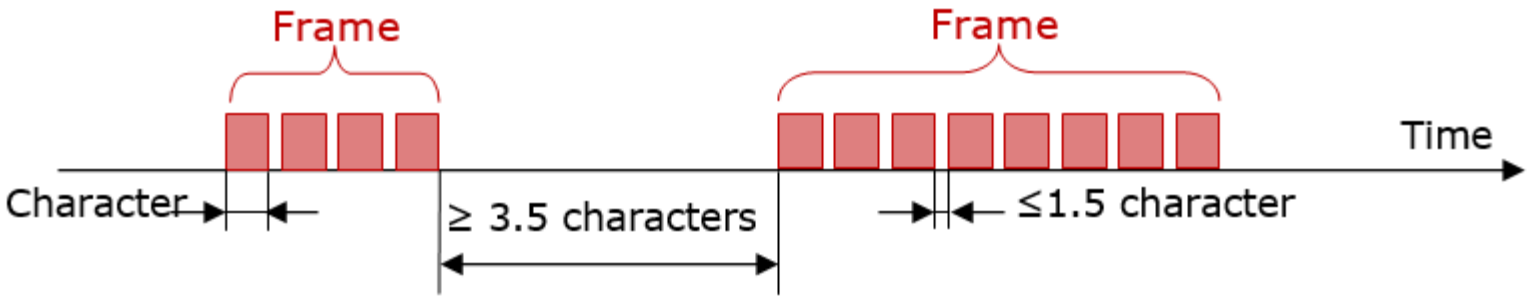
Three kinds of physical link

- RS232: three wire (RX, TX, GND), point to point
- RS485 2 wires: differential pair (half duplex), multi server
- RS485 4 wires: differential pair (full duplex), multi server



Serial line Modbus frames protocol

- A Modbus frame is defined by the time between bytes
  - One byte: 1 start bit, 8 data bits, 2 stop bits (or one stop and one parity)
  - Same frame byte: Duration between bytes is less than 1.5 byte length
  - End of frame: Duration between bytes is more than 3.5 byte length



Read holding registers (full example)

- Request PDU
  - Function code: 0x03 for read holding registers
  - Starting address: Two bytes (MSB,LSB) representing the hardware address of the first register to read (this is register number - 1)
  - Quantity of registers: Two bytes (MSB,LSB) representing the number of registers to read
  - Byte count: Number of returned bytes
  - Register value: Two bytes (MSB,LSB) for each returned register

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

Error

Error code	1 Byte	**0x83**
Exception code	1 Byte	01 or 02 or 03 or 04

What we have seen

- The various levels of a standard protocol (more will come with D. Gabioud)
- Half of the theory for this semester for the uC part