Modeling and Automation of Industrial Processes

Modelação e Automação de Processos Industriais / MAPI

PLC Programming languages

Structured Text - Networking

http://www.isr.tecnico.ulisboa.pt/~jag/courses/mapi2122

Prof. Paulo Jorge Oliveira, original slides
Prof. José Gaspar, rev. 2021/2022

Structured Text

Networking (in Unity Pro)

Keywords: MODBUS, READ_VAR, WRITE_VAR

Modbus is a serial communications protocol originally published by Modicon (now Schneider Electric) in 1979 for use with its programmable logic controllers (PLCs). Simple and robust, it has since become a de facto standard communication protocol, and it is now a commonly available means of connecting industrial electronic devices.

Examples of Field Bus (IEC 61158) standards: ***MODBUS*** *(Schneider),* ***PROFIBUS*** *(Field Bus type, Siemens),* ***CAN*** *bus (Controller Area Network, 1983 Robert Bosch GmbH), ...*

Structured Text

Modbus RTU — Binary representation of the data for protocol communication. Includes CRC. Modbus messages are framed (separated) by idle (silent) periods.

Modbus ASCII — Makes use of ASCII characters for protocol communication.

Modbus TCP/IP or Modbus TCP — Modbus variant for communications over TCP/IP networks, connecting over port 502.

RTU = Remote Terminal Unit
MTU = Main Terminal Unit
CRC = Cyclic Redundancy Check
TCP = Transmission Control Protocol
ASCII = American Standard Code for Information Interchange

Modbus	Function type	Function name / Function code	
Bit access	Physical Discrete Inputs	**Read Discrete Inputs**	**2**
	Internal Bits or Physical Coils	**Read Coils**	**1**
		Write Single Coil	**5**

Function Code	Function	Request	Normal Response
1	Read Coil Status	Address of first coil to read (2 bytes). Number of coils to read (2 bytes).	Number of bytes of coil values to follow (1 byte). Coil values (8 coils per byte).
2	Read Input Status	Address of first discrete input to read (2 bytes). Number of discrete inputs to read (2 bytes).	Number of bytes of discrete input values to follow (1 byte). Discrete input values (8 discrete inputs per byte).
3	Read Holding Registers	Address of first register to read (2 bytes). Number of registers to read (2 bytes).	Number of bytes of register values to follow (1 byte). Register values (2 bytes per register).
4	Read Input Registers	Address of first register to read (2 bytes). Number of registers to read (2 bytes).	Number of bytes of register values to follow (1 byte). Register values (2 bytes per register).
5	Force/Write Single Coil	Address of coil (2 bytes). Value to force/write: 0 for OFF and 65,280 (FF 00 in hexadecimal) for ON.	Same as request.
6	Write Single Holding Register	Address of holding register to write (2 bytes). New value of the holding register (2 bytes).	Same as request.
15	Force/Write Multiple Coils	Address of first coil to force/write (2 bytes). Number of coils to force/write (2 bytes). Number of bytes of coil values to follow (1 byte). Coil values (8 coil values per byte).	Address of first coil (2 bytes). Number of coils (2 bytes).
16	Write Multiple Holding Register	Address of first holding register to write (2 bytes). Number of holding registers to write (2 bytes). Number of bytes of register values to follow (1 byte). New values of holding registers (2 bytes per register).	Address of first written holding register (2 bytes). Number of written holding registers (2 bytes).

Modbus, send a message (Matlab) :

```
t = tcpip('85.16.111.143', 502, 'NetworkRole', 'client');
fopen(t);
packet = [0 1 0 0 0 4 0 90 0 2];
fwrite(t, packet, 'uint8');
```

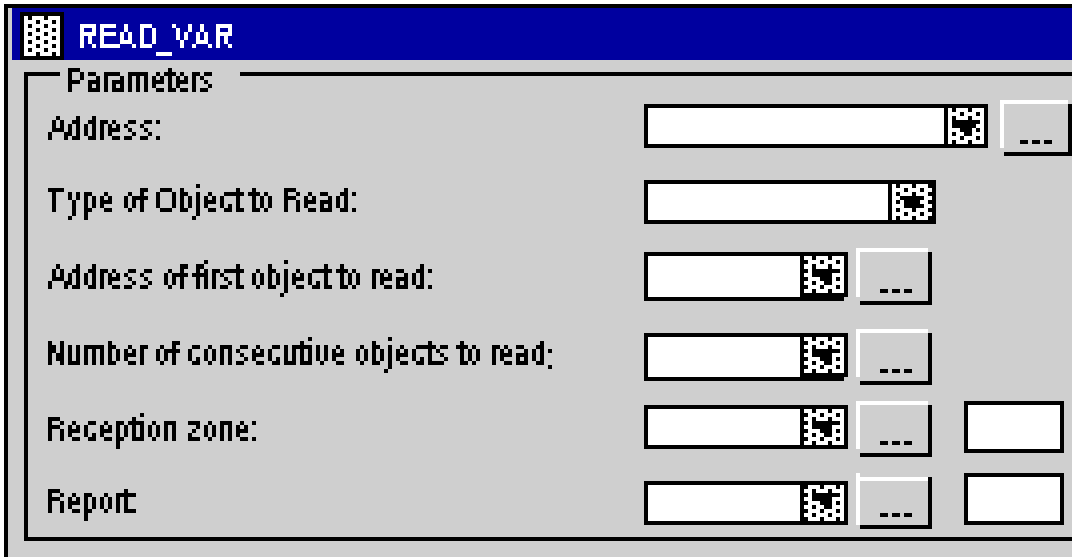
Modbus, see the message (Wireshark) :

Source	Destination	Protocol	Length	Info
85.16.111.1	85.16.111.143	TCP	66	52131 → 502 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
85.16.111.143	85.16.111.1	TCP	70	502 → 52131 [SYN, ACK] Seq=0 Ack=1 Win=4096 Len=0 MSS=1452 WS=1
85.16.111.1	85.16.111.143	TCP	54	52131 → 502 [ACK] Seq=1 Ack=1 Win=66560 Len=0
85.16.111.1	85.16.111.143	TCP	54	[TCP Window Update] 52131 → 502 [ACK] Seq=1 Ack=1 Win=2097152 Len=0
85.16.111.1	85.16.111.143	Modbus…	64	Query: Trans: 1; Unit: 0, Func: 90: Unity (Schneider)
85.16.111.143	85.16.111.1	TCP	64	502 → 52131 [ACK] Seq=1 Ack=11 Win=4093 Len=0
85.16.111.143	85.16.111.1	Modbus…	117	Response: Trans: 1; Unit: 0, Func: 90: Unity (Schneider)
85.16.111.1	85.16.111.143	TCP	54	52131 → 502 [ACK] Seq=11 Ack=56 Win=2096896 Len=0

```
> Modbus/TCP
v Modbus
    .101 1010 = Function Code: Unity (Schneider) (90)
    [Request Frame: 10]
    Data: 00fe0530021100000000200200002100050211000000000000...
```

```
0000  c8 d3 ff d5 e6 1d 00 80  f4 02 6f 8f 00 67 aa aa   ........ ..o..g..
0010  03 00 00 00 08 00 45 00  00 5f 00 1a 00 00 40 06   ......E. ._....@.
0020  f1 ce 55 10 6f 8f 55 10  6f 01 01 f6 cb a3 93 b9   ..U.o.U. o.......
0030  75 be 2f e4 dd 19 50 18  10 00 10 1c 00 00 00 01   u./...P. ........
0040  00 00 00 31 00 5a 00 fe  05 30 02 11 00 00 00 00   ...1.Z.. .0......
0050  20 02 00 00 21 00 05 02  11 00 00 00 00 00 0d 54    ...!... .......T
0060  53 58 20 50 35 37 20 32  36 33 34 4d 01 01 01 00   SX P57 2 634M....
0070  00 00 80 02 00                                     .....
```

READ_VAR



Address:
ADDR(STRING)
ARRAY [0..5] OF INT

Type of object to read:
'%M' for reading internal bits
'%MW' for reading internal words
'%S' for reading system bits
'%SW' for reading system words
'%I' for reading input bits
'%IW' for reading input words

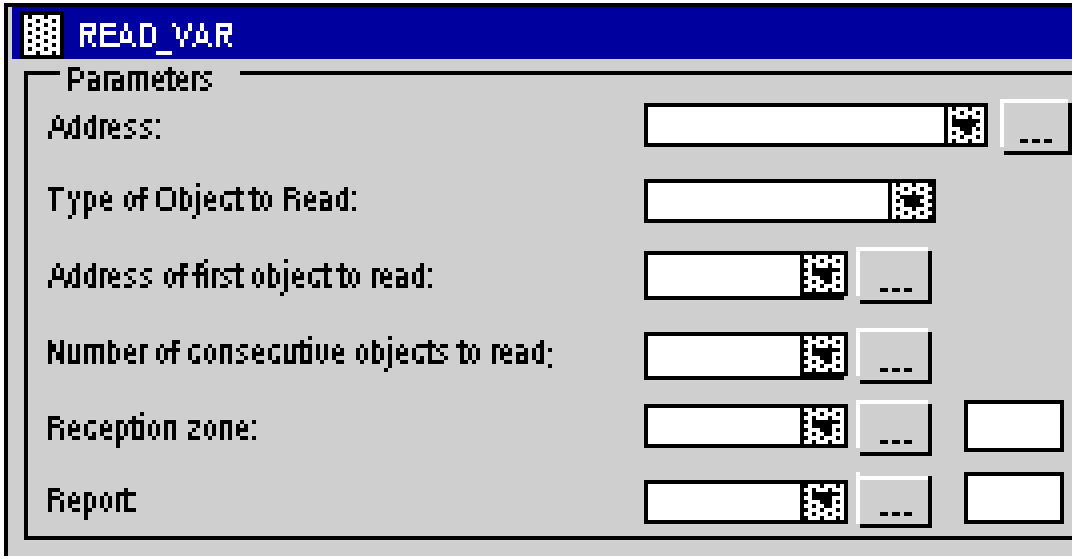
Address of first object to read:
The possible objects are of the DINT type (variables, constants, immediate value)

Number of consecutive objects to read:
The possible objects are of the INT type (variables, constants, immediate value)

Reception zone:
The reception zone is an integer array. The size of this array depends on the number of objects to read. This integer array can be located or not.

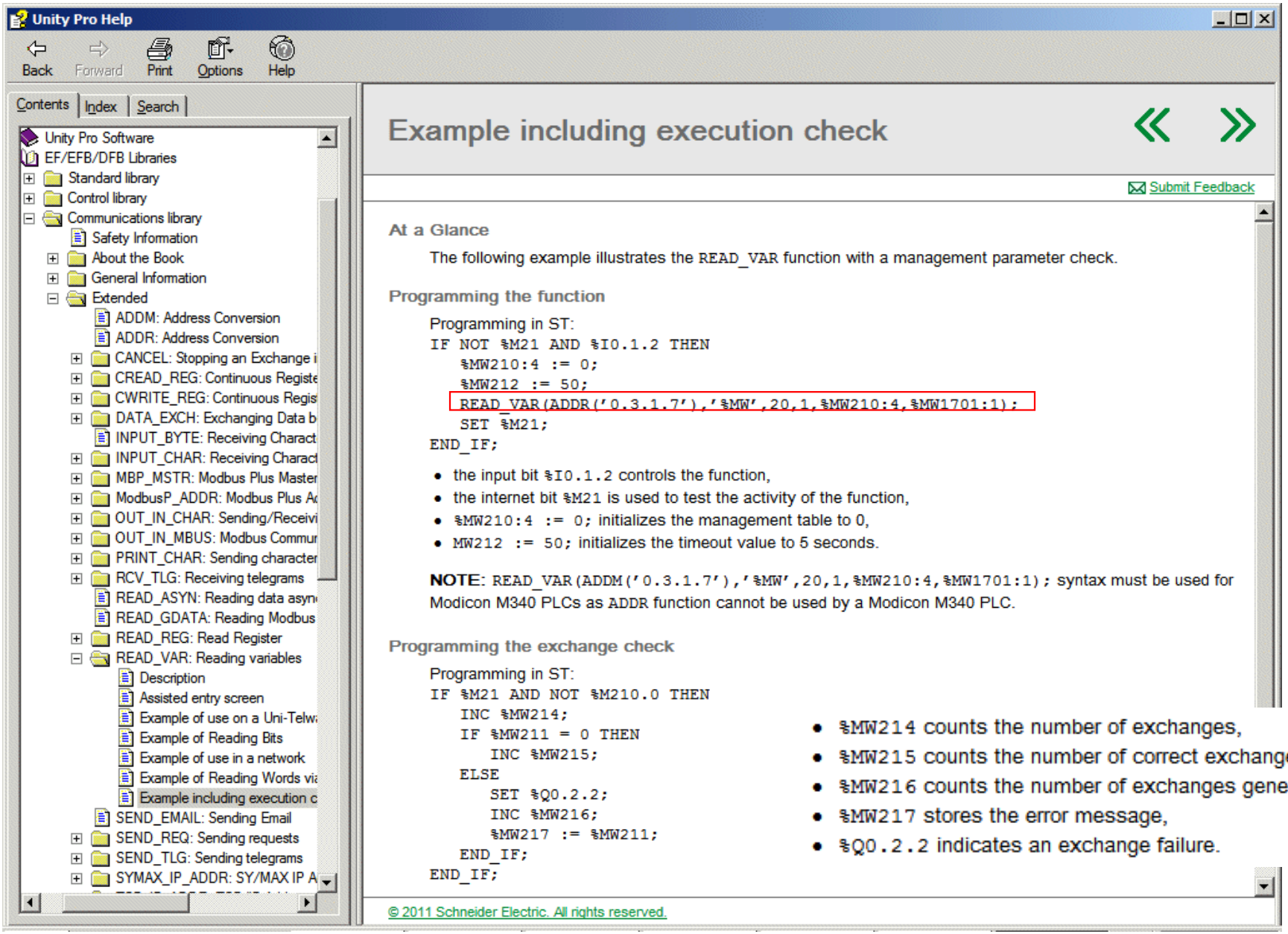
Report: The report is an array of 4 integers

READ_VAR



Challenge: how to make READ_VAR non-blocking in an operating system without using processes nor threads?

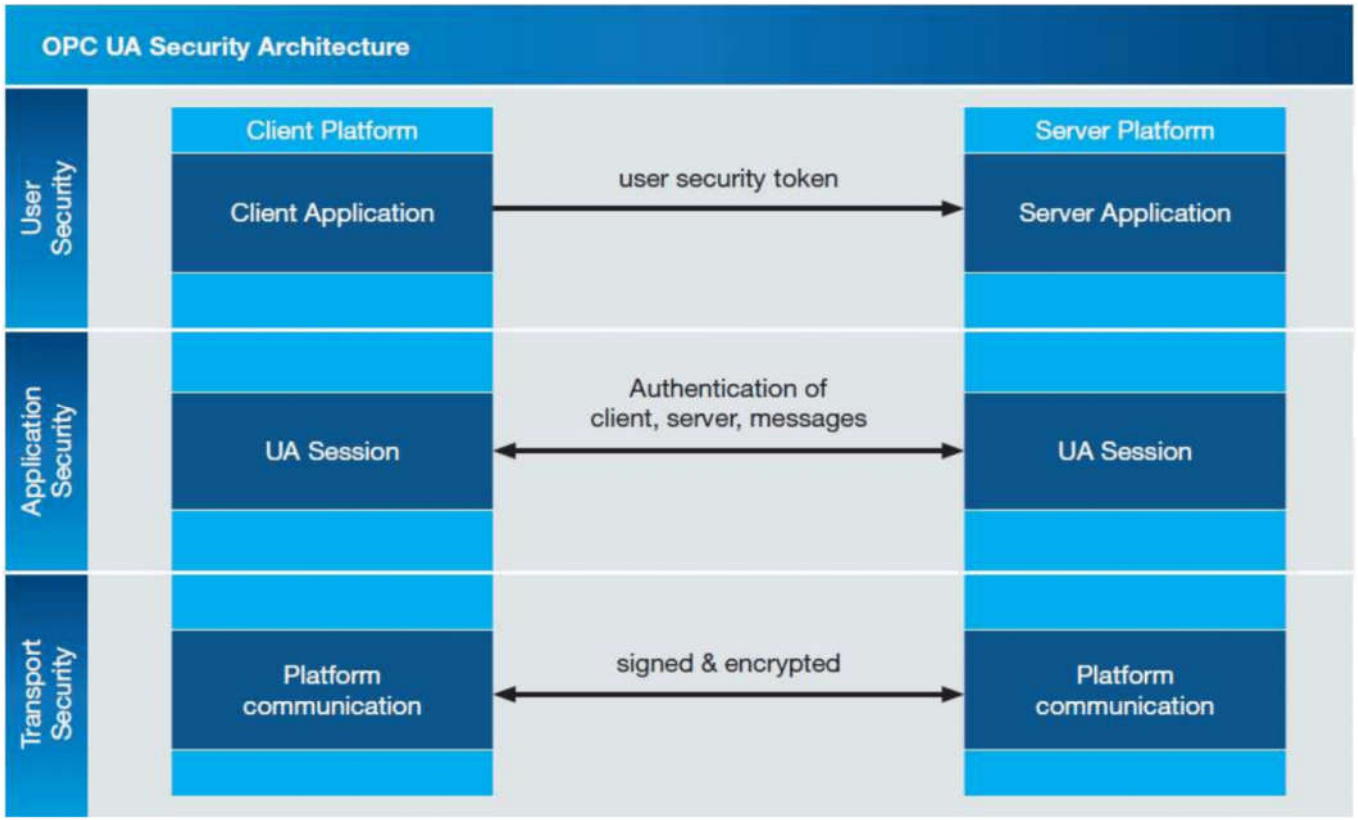
READ_VAR


Unity Pro Help
Back
Forward
Print
Options
Help
Contents
Index
Search
Unity Pro Software
EF/EFB/DFB Libraries
Standard library
Control library
Communications library
Safety Information
About the Book
General Information
Extended
ADDM: Address Conversion
ADDR: Address Conversion
CANCEL: Stopping an Exchange i
CREAD_REG: Continuous Registe
CWRITE_REG: Continuous Regis
DATA_EXCH: Exchanging Data b
INPUT_BYTE: Receiving Charact
INPUT_CHAR: Receiving Charact
MBP_MSTR: Modbus Plus Master
ModbusP_ADDR: Modbus Plus Ad
OUT_IN_CHAR: Sending/Receivi
OUT_IN_MBUS: Modbus Commur
PRINT_CHAR: Sending character
RCV_TLG: Receiving telegrams
READ_ASYN: Reading data asyn
READ_GDATA: Reading Modbus
READ_REG: Read Register
READ_VAR: Reading variables
Description
Assisted entry screen
Example of use on a Uni-Telw
Example of Reading Bits
Example of use in a network
Example of Reading Words via
Example including execution c
SEND_EMAIL: Sending Email
SEND_REQ: Sending requests
SEND_TLG: Sending telegrams
SYMAX_IP_ADDR: SY/MAX IP A
Example including execution check
Submit Feedback
At a Glance
The following example illustrates the READ_VAR function with a management parameter check.
Programming the function
Programming in ST:
IF NOT %M21 AND %I0.1.2 THEN
%MW210:4 := 0;
%MW212 := 50;
READ_VAR(ADDR('0.3.1.7'),'%MW',20,1,%MW210:4,%MW1701:1);
SET %M21;
END_IF;
the input bit %I0.1.2 controls the function,
the internet bit %M21 is used to test the activity of the function,
%MW210:4 := 0; initializes the management table to 0,
MW212 := 50; initializes the timeout value to 5 seconds.
NOTE: READ_VAR(ADDM('0.3.1.7'),'%MW',20,1,%MW210:4,%MW1701:1); syntax must be used for Modicon M340 PLCs as ADDR function cannot be used by a Modicon M340 PLC.
Programming the exchange check
Programming in ST:
IF %M21 AND NOT %M210.0 THEN
INC %MW214;
IF %MW211 = 0 THEN
INC %MW215;
ELSE
SET %Q0.2.2;
INC %MW216;
%MW217 := %MW211;
END_IF;
END_IF;
%MW214 counts the number of exchanges,
%MW215 counts the number of correct exchanges,
%MW216 counts the number of exchanges generating errors,
%MW217 stores the error message,
%Q0.2.2 indicates an exchange failure.
© 2011 Schneider Electric. All rights reserved.

Open Platform Communication - Unified Architecture (OPC-UA)

Device and machine builders must ensure data integrity, confidentiality, ownership and **security** [Sikora'18].

In [ABB'18] is indicated **OPC UA is currently the best solution** that realizes the use case in a secure way to interface to the PLC and the World Wide Web (WWW).



[Sikora'18] "Practical Security Recommendations for building OPC UA Applications", A. Sikora, Industrial Ethernet Book, pages 2–6, 2018
[ABB'18] "What is OPC UA?", ABB Group, https://www.automation.com/automation-news/industry/abb-announces-support-for-opc-ua-over-tsn-communication-standard. Accessed: 2018-12-27