Modbus RTU Proview 4.8 tutorial.

PLEASE EXCUSE MY ENGLISH !!!

What do I use??



1. Converter USB to RS232 settings:

In **Ubuntu**

Serial COM1 \rightarrow /dev/ttyUSB0 Serial COM2 \rightarrow /dev/ttyUSB1



VirtualBox \rightarrow Debian

	General	Serial Ports			
	Display	Port <u>1</u> Port <u>2</u>	N		
9 #	Storage Audio	☑ Enable Serial Po	brt 6		
₽	Network	Port <u>N</u> umber:	COM1		IRQ: 4 I/O Port: 0x3F8
	Serial Ports USB	Port <u>M</u> ode:	Host Device	•	
	Shared Folders		Create Pipe		
		Port/File <u>P</u> ath:	/dev/ttyUSB0		





VirtualBox → Windows XP



Windows $XP \rightarrow COM1$



Windows XP → Emulare COM1

Virtual Serial Ports Ed File View Language	mulator (Emulation started) Emulation Device Help 🗮 🦮 🦎 🍂 🎆 🕕	
Title	Specify device type COM1 Image: Splitter Image: Splitter	
Ready	New serial port Existing serial port Client application	com //

Initial modem registers Speed Initial modem registers state DTR/RTS Image: Initial modem registers state Parity Bits Stop bits	
Bits Stop bits	19200 NO
ReadIntervalTimeout	8 1 -1
	K Cancel

2. Proiect Proview

Create and then open the project for testing Modbus RTU:

×	PwR Project List							
<u>F</u> ile	<u>E</u> dit F <u>u</u> nctio	ns <u>∨</u> iew <u>O</u> ptions	<u>T</u> ools <u>H</u> elp					
📥 🖉 🐉 🗗 📾 🏷 🌲 💷 🔺 🕫 🔍 🔍								
	Bases	\$Hier		\frown				
	DANIEL	\$Hier						
Ø	Test	\$Hier	Open Object					
	test1Modbus	RTU ProjectReg	Crossreferences					
1	ppTl	ProjectReg	crossreterences					
⊛	pwrdemo48	ProjectReg	Help					
			Help Class					
			Open Project	=				
			2					

Requirements project (the project are real data)

I want to read seven words, starting with the address 3202 (0x0C82, {43203 - modbus}) of the three converters ATV71-type (Schneider-Electric), using Modbus RTU serial protocol.



Now, let's see the project. First, please study the tutorial from the following link: <u>http://www.proview.se/index.php?</u> <u>option=com_joomlaboard&Itemid=24&func=view&id=268&catid=3#268</u> and chapter 12.2.6 from designer's guide manual.

Proview

How to configure Modbus RTU:

Place a RemoteConfig in the node-hierarchy;

🗁 Nodes	\$	NodeHier
🗁 Test	lmodbusrtu	\$Node
🗁 Re	mote	RemoteConfig
þ r	mMb	RemnodeModbus
Ð	mbT×1	RemTrans
	mbTxBuff	Buff256
þ	mbRx1	RemTrans
	mbRxBuff	Buff256
đ	mbT×2	RemTrans
 ⊕	mbTxBuff	Buff256
Ð	mbR×2	RemTrans
	mbRxBuff	Buff256
Ð	mbT×3	RemTrans
	mbTxBuff	Buff256
Ď	mbR×3	RemTrans
	mbRxBuff	Buff256
കം	e unitsz	¢Security

Below the RemoteConfig-object you place a RemnodeModbus object for each communication link you
will have in your system. Configure serial port as shown below;

2	5	Nodes-Test1mo	odbusrtu-Remote-rmMb		3)					
1 E	ile	<u>F</u> unctions <u>H</u> elp								
F	•	Description		[~					
. –	•	Prio	15			<u>^</u>	Ø	Nodes	; \$	NodeHier
	•	DevName	/dev/ttyS0 💥				Ø	Test	lmodbusrtu	\$Node
_	•	Speed	19200 💥					Re	mote	RemoteConfig
Ē	•	Parity	0 💥				× (י ל	mMb	RemnodeModbus
Ē	•	StopBits	1 💥					Þ	mbTx1	RemTrans
٩ Ē	7	DataBits	8 💥						mbTxBuff	Buff256
	•	ReadTimeout	0.050000					Ð	mbRx1	RemTrans
_		LinkTimeout	10.000000					۲	mbRxBuff	Buff256
_	•	Disable	0					Ð	mbTx2	RemTrans
		RestartLimit	100					۲	mbTxBuff	Buff256
		ScanTime	0.100000					Ø	mbRx2	RemTrans
In	3	RemTransObjects				Ξ		۲	mbRxBuff	Buff256
1	3							đ	mbTx3	RemTrans
								۲	mbTxBuff	Buff256
				3				đ	mbRx3	RemTrans
								۲	mbRxBuff	Buff256
								Se	curity	\$Security

Below the RemnodeModbus-object you place a RemTrans-object;
 SENDING → mbTx {Address[0] = 1 (slave-address), Address[1] = 3 (modbus function code)}

🚰 N	lodes-Test1mod	lbusrtu-Remote-rn 🗕	.ox)	
<u>F</u> ile	<u>F</u> unctions <u>H</u> elp)		🗁 Nodes \$NodeHier
	Description TransName		<u>^</u>	☐ TestImodbusrtu \$Node ☐ TestImodbusrtu \$Node ☐ Remote Remote
	Address Address[0]	1 👾		(2) rmMb RemnodeModbus ★ (2) mbTx1 RemTrans
-	Address[0]	3 🕱		→ mbTxBuff Buff256 (☐ mbRx1 RemTrans
	Address[2] Address[3]	0		③ mbRxBuff Buff256 2 mbTx2 BemTrans
	Direction Receive	Send 💥 🗌		⊕ mbTxBuff Buff256
	Send DataLength	4 💥		mbRx2 Remirans mbRxBuff Buff256
	LoggLevel MaxBuffers	0		l mbTx3 RemTrans → mbTxBuff Buff256
-	Buffers	0		C→ mbRx3 RemTrans ③ mbRxBuff Buff256
	MaxLength	0	=	 ⊕ Security \$Security

RECEIVING \rightarrow **mbRx** {Address[0] = 1 (slave-address), Address[1] = 3 (modbus function code)}

(🔠 N	lodes-Test1m	odbusrtu-Remote-rmM	b-mbR - 🗆 🗙
File	Functions He	elp	International I
-	Description		🛆 🗁 Remote RemoteConfig
-	TransName		🗁 rmMb RemnodeModbus
	Address		🗁 mbTx1 RemTrans
-	Address[0]	1 💥	🛞 mbTxBuff Buff256
-	Address[1]	з 💥	🛛 💥 🗁 mbRx1 🛛 RemTrans
-	Address[2]	0	International In
-	Address[3]	0	🗁 mbTx2 RemTrans
Ø	Direction	Receive 💥	Improvement Improv
-	Receive		🗁 mbRx2 RemTrans
-	Send		Improvement Improv
-	DataLength	7 💥	🗁 mbTx3 RemTrans
-	LoggLevel	0	Improvement Improv
-	MaxBuffers	0	🗁 mbRx3 RemTrans
-	Buffers	0	🛞 mbRxBuff Buff256
	MaxLength	0	Security \$Security

Please note that $mbTx \rightarrow Address[0]$ and $mbRx \rightarrow Address[0]$ must coincide. Same $mbTx \rightarrow Address[1]$ and $mbRx \rightarrow Address[1]$!

 The Buffers-objects are the data area for the message. Place a Buff256-object for each RemTransobject.

Settings for RemTrans items:

```
mbTx1 \rightarrow Address[0] = 1, Address[1] = 3, Direction = Send, DataLenght = 4;

mbRx1 \rightarrow Address[0] = 1, Address[1] = 3, Direction = Receive, DataLenght = 10;

mbTx2 \rightarrow Address[0] = 2, Address[1] = 3, Direction = Send, DataLenght = 4;

mbRx2 \rightarrow Address[0] = 2, Address[1] = 3, Direction = Receive, DataLenght = 10;

mbTx3 \rightarrow Address[0] = 3, Address[1] = 3, Direction = Send, DataLenght = 4;

mbRx3 \rightarrow Address[0] = 3, Address[1] = 3, Direction = Receive, DataLenght = 4;

mbRx3 \rightarrow Address[0] = 3, Address[1] = 3, Direction = Receive, DataLenght = 10.
```

I notice that the DataLenght item for "Direction = Receive", can be set to any value !

Now that we have established communication parameters, let's have a look on the project variables.

Project hierarchy:



Send \rightarrow stAddr (the Data Address of the first register requested) \rightarrow InitialValue = 3202; \rightarrow noOfAddrs (the total number of registers requested) \rightarrow InitialValue = 7.

🗁 Se	nd	\$Planti	Hier
⊛! :	stAddr	lv	
	Description		
	InitialValue		3202
	PresMaxLimit		0.000000
	PresMinLimit		0.000000
-+	DefGraph		
-+	DefTrend		
	HelpTopic		
	DataSheet		
	CircuitDiagra	m	
	Photo		
📟 Note			
⊛! ı	noOfAddrs	lv	
	Description		
	InitialValue		7
	PresMaxLimit		0.000000
	PresMinLimit		0.000000
-+	DefGraph		
-+	DefTrend		
	HelpTopic		
	DataSheet		
	CircuitDiagra	m	
	Photo		
	Note		
🗂 Re	ceive	\$Plant	Hier

Receive $\rightarrow rx(1..7)(1..3)$

	Send	\$PlantHier
Þ	Receive	\$PlantHier
	rxl_l	lv
	rx2_1	Iv
	rx3_1	lv
	rx4_1	lv
	rx5_1	lv
	rx6_1	lv
	rx7_1	lv
	rxl_2	lv
	rx2_2	lv
	rx3_2	lv
	rx4_2	lv
	rx5_2	lv
	rx6_2	lv
	rx7_2	lv
	rxl_3	lv
	rx2_3	lv
	rx3_3	lv
	rx4_3	lv
	rx5_3	lv
	rx6_3	lv
	rx7_3	lv
	Controls	\$PlantHier

T.	
Controls	\$PlantHier
sendDvl	Dv
occDvl	Dv
errDvl	Dv
sendDv2	Dv
occDv2	Dv
errDv2	Dv
sendDv3	Dv
occDv3	Dv
errDv3	Dv
	T. Controls sendDv1 occDv1 errDv1 sendDv2 occDv2 errDv2 sendDv3 occDv3 errDv3

PLCRxTx \rightarrow plc program.

First, let's see how the protocol works:

1. Send (mbTx):

01	03	0C	82	00	07	A7	70
Address[0] (slave-address)	Address[1] (modbus function code)	stAd (the Data Addre register re	ldr ess of the first quested)	noOf (the total registers	Addrs number of requested)	С	RC

We need to create a data structure for the send messages. That will be defined in the file ra_plc_user.h in pwrp_inc-directory. This file is automatically included when you compile the plc-code.



The plc code



RemTransSend must have a subwindow. The send-buffer for the message to send is connected to a DataArithm.



In the end, I created a XttGraph to test the serial communication.



Now let's see what Proview send to the slaves.

01	03	82	0C	07	00	AF	81
Address[0] (slave-address)	Address[1] (modbus function code)	stAc (the Data Addre register re	<mark>ldr</mark> ess of the first quested)	noOt (the total registers	fAddrs number of requested)	C	RC

Not good at all!

Note though that Modbus works with Big Endian, so you need to **byte-swap**.

We need to correct the plc code.

The code from DataArithm-object will be:



Now everything's fine.

2. <u>Receive(mbRx)</u>:

01	03	0E	00	0A	00	64	03	E8	27	10	00	32	01	F4	13	88	64	86
a	b	c	(ł	6	e	1	f	£	g	1	h		i		j	CF	RC

	An	swer: 25.05.2	011 01:54:5	1.02464 (+8	3.1996 seco	nds)			
	e 0	1 03 0C 8	2 00 07	A7 70					
	Re	quest: 25.05.2	2011 01:54:5	52.52564 (+0	0.0000 secor	nds)			
	0	1 03 OE 0	00 A0 0	64 03 E8	27 10 00) 32 01 F	4 13		
		8 64 86							
	-	-	-	-	-	-	-	-	-
43181-43190	0	0	0	0	0	0	0	0	0
43191-43200	0	0	U 10	U 100	U 1000	U 10000	U 50	U 500	U 5000
43211-43220	0 0	0	0	0	0	n 10000	0	0	0
43221-43230	Ő	õ	õ	õ	ŏ	õ	ŏ	õ	õ
1 40004 40040	<u> </u>	~	~	~	~	~	<u> </u>	~	~

a- the slave address;
b- the function code;
c- the number of data bytes to follow (7 registers x 2 bytes each = 14 bytes);
d- the contents of register 0x0C82 (3202 in decimal, {43203 modbus});
e- the contents of register 0x0C83 (3203 in decimal, {43204 modbus});
f- the contents of register 0x0C84 (3204 in decimal, {43205 modbus});
g- the contents of register 0x0C85 (3205 in decimal, {43206 modbus});
h- the contents of register 0x0C86 (3206 in decimal, {43207 modbus});
i- the contents of register 0x0C87 (3207 in decimal, {43208 modbus});

We need to create a data structure for the receive messages. That will be defined in the file ra_plc_user.h in \$pwrp_inc-directory. This file is automatically included when you compile the plc-code.



The plc code:



RemTransRcv must have a subwindow. The send-buffer for the message to send is connected to a DataArithm.



After I compile the application and test the communication, I received the following answer:

01	03	00	0 E	00	0A	03	64	27	E8	00	10	01	32	13	F4	00	88
1	3	1	4	1	0	868		10216		16		306		5108		136	
a	b	c	d	e	f	g	h	i	j	k	1	m	n	0	p	q	r

	RemTrans		DataVa	<u>F</u> ile	<u>F</u> unctions <u>H</u> elp		SEND
	mbT×1		0		Data	<u>^</u>	
©	mbR×1		0		Data[0]	14	
\circledast	mbRxBuff	Buff256			Data[1]	10	14
					Data[2]	868	
					Data[3]	10216 📃	10
					Data[4]	16	
					Data[5]	306	262
					Data[6]	5108	000
					Data[7]	136	
					Data[8]	0	10216
					Data[9]	0	
					Data[10]	0	16
					Data[11]	0	
					Data[12]	0	306
					Data[13]	0	
					Data[14]	0	5108
					Data[15]	0	
					Data[16]	0	
					- · ·	-	

Now we have two problems. The byte-swap and the "d" (the number of data bytes) which is part of modbus protocol.

We need to correct the plc code.

The code from DataArithm-object will be:

Dat	aA1	Dat	aA2
classdef Da1 mbRx unsigned int rx1; unsigned int rx2; unsigned int rx3; unsigned int rx3; unsigned int rx4; unsigned int rx5; unsigned int msb; unsigned int lsb; rx1=Da1->rx1; rx2=Da1->rx2; rx3=Da1->rx3; rx4=Da1->rx4; rx5=Da1->rx5; msb=rx1 & 0xff00; lsb=rx2 & 0x00ff; OI1=msb+lsb;	msb=rx2 & 0xff00; lsb=rx3 & 0x00ff; OI2=msb+lsb; msb=rx3 & 0xff00; lsb=rx4 & 0x00ff; OI3=msb+lsb; msb=rx4 & 0xff00; lsb=rx5 & 0x00ff; OI4=msb+lsb;	classdef Da1 mbRx unsigned int rx5; unsigned int rx6; unsigned int rx7; unsigned int rx8; unsigned int msb; unsigned int lsb; rx5=Da1->rx5; rx6=Da1->rx6; rx7=Da1->rx7; rx8=Da1->rx8; msb=rx5 & 0xff00; lsb=rx6 & 0x00ff; OI1=msb+lsb;	msb=rx6 & 0xff00; lsb=rx7 & 0x00ff; OI2=msb+lsb; msb=rx7 & 0xff00; lsb=rx8 & 0x00ff; OI3=msb+lsb;



Finally, we will write code for the three inverters and we will see the result.

Modbus 1	Modbus 2	Modbus 3
SEND 1	SEND 2	SEND 3
10	20	30
100	200	0
1000	0	3000
10000	0	0
50	0	0
500	0	0
5000	6000	7000

				Port Monit	or:						7.20 Serv										
		10000 12 12		17			MODBU	S RTU R	5-232 PLC	- Simulato	or (port: COM	12 9600,8	,N,1,R-en)								
	22	=			T11 D C-	.232 DI C - Sim	Connected (1) : (recei	ved/sent) (3	/1) Serv. rea	ad data.		- • • 🛔 🛔 🗗 📼 ∔ 🖤 🚟 📃 🛄 🖗				0 7				
È –	Google	Transmissio)			232 FEC - 3111		-		La					-						
	👑 MODBUS R	TU R5-232	2 PLC - Simu	Connected (1) :	(receive	ed/sent) (3/1) Se	r Address :	⊖ Hex) Dec	1/0 [Hol	ding Hegisters		Fmt: de	cimal	Prot:	IMODBOS F	{S-(▼)	Clone			
	Connected (1):	(received/s	ent) (13/11) 9	Address : 🔿	Hex	• Dec 1/0	Address	+	0 +	1 +2	+3	+4	+5	+6	+7	+8	+9				
	Address · C	Hey 💽	Dec 1/0	Address	+0	±1	43031-431	10 0	0	0	0	0	0	0	0	0	0				
e	Address. C	1100 0	000 1/0	43151-43160	0	0	43111-431	20 0	ň	ň	ñ	ň	ñ	ñ	ň	ň	ñ				
	Address	+0	41	43161-43170	ñ	0	43121-431	30 0	ň	ň	ñ	ň	ñ	ñ	ñ	ň	ň				
	A001655	0		43171-43180	ň	ñ	43131-431	40 0	ŏ	ŏ	ŏ	ŏ	ŏ	ŏ	ŏ	ŏ	ŏ				
	43131-43140	0	ů l	43181-43190	ŏ	ŏ	43141-431	50 0	Ō	Ō	ō	ō	ō	ō	ō	ō	ō				
	43141-43150	ñ	ň	43191-43200	ō	ō	43151-431	60 0	0	0	0	0	0	0	0	0	0				
her	43151-43160	ñ	ň	43201-43210	0	0	43161-431	70 0	0	0	0	0	0	0	0	0	0				
	43161-43170	õ	õ	43211-43220	0	0	43171-431	80 0	0	0	0	0	0	0	0	0	0				
	43171-43180	Ō	0	43221-43230	0	0	43181-431	90 0	0	0	0	0	0	0	0	0	0				
	43181-43190	0	0	43231-43240	0	0	43191-432	00 0	0	0	0	0	0	0	0	0	0				
	43191-43200	0	0	43241-43250	0	0	43201-432	10 0	0	30	0	3000	0	0	0	7000	0				
e	43201-43210	0	0	43251-43260	0	0	43211-432	20 0	0	0	0	0	0	0	0	0	0				
3	43211-43220	0	0	43261-43270	0	0	43221-432	30 0	0	0	0	0	0	0	0	0	0				
	43221-43230	0	0	43271-43280	0	0	43231-432	40 0	0	0	0	0	0	0	0	0	0				
	43231-43240	0	0	43281-43290	0	0	43241-432	50 U	U	U	U	U	U	U	U	U	U				
	43241-43250	0	0	43291-43300	0	0	43251-432	50 U	U	U	U	U	U	U	U	U	U				
	43251-43260	0	0	43301-43310	U	U	43261-432	70 U	U	U	U	U	U	0	U	U	U				
.ini	43261-43270	0	0	43311-43320	U	U	43271-432	80 U	U 0	0	U	U	0	U	U	0	U	_			
	43271-43280	0	0	43321-43330	U	U	1.4.3/81-4.3/3	911 11													
	43281-43290	U	U	43331-43340	0	0	00 01 02	03 04	05 06 07	08 09 10	11 12 13 1	4 15 16	17 18 19	20 21 23	2 23 24	25 🔺	T	Comms			
	43291-43300	U	U	4.5.541-4.5.5501			26 27 28	29 30	31 32 33	34 35 36	37 38 39 4	0 41 42	43 44 45	46 47 48	3 49 50						
	43301-43310	U	U	00 01 02 03	04 05	06 07 08 09												7///			
	43311-433/11			26 27 28 29	30 31	32 33 34 35	36 37 38 3	39 40 41	1 42 43 4	4 45 46 4	47 48 49 50	•									
	00 01 02 03	04 05 06	6 07 08 09												₫						
	26 27 28 29	30 31 32	2 33 34 35	36 37 38 39	40 41	42 43 44 45	46 47 48 49	50	-												

Everything works great. Hope you enjoy this tutorial.

Daniel - 2011