## Simple Example To Test AKD BASIC Modbus TCP Communications ( Part 2: using KVB ) Rev. A 9/25/2019

This application note will demonstrate how to prove you can read and write to the AKD BASIC variables over Modbus TCP. In Part1 of this series we demonstrated communications using a software based Modbus TCP Master called Modbus Poll.

In part 2 we will accomplish the same objectives using Kollmorgen Visualization Builder software. If you've already created the program in part 1 it can be used to start part 2 otherwise the development is repeated here.

There are several ranges of Modbus TCP Parameters (and addresses) in the AKD-B, -P, and -T drives

(note the AKD-P-NBPN, Profinet drive does not support Modbus TCP).

The Modbus Manual is embedded in Workbench Help. See each individual parameter descriptions to determine if the parameter is supported by the AKD-T (AKD BASIC) or not.

🥑 Kollmorgen WorkBench Help					-	- 🗆	$\times$
KOLLMORGEN®				Search		All Files 🔻	<u>م</u>
Contents Index	E						
📄 Welcome Page		You are here: AKD Fieldbus Manuals > Fieldbus Manuals > Modbu	us Manual	> Modbus F	Parameter Table		~
Workbench User Manual							-
AKD Parameter and Command Reference Guide		Modbus Parameter Table					
AKD BASIC Programming Manual Using the Program View AKD BASIC Language		Parameter	Modbus Register Address	ls 64-bit?	Attributes		
Quick Reference: Parameters, Functions, Operators AKD BASIC Functions		AIN.CUTOFF	0		32-bit		
AKD BASIC Operators		AIN.DEADBAND	2		16-bit		
AKD BASIC Parameters and Statements Appendix A: BASIC Sample Programs	•	AIN.ISCALE	4		32-bit	-	
AKD Fieldbus Manuals		AIN.OFFSET	6		16-bit, signed		
Fieldbus Manuals		AIN.PSCALE	8	Yes	64-bit, signed	-	
Modbus Manual		AIN.VALUE	12		16-bit	-	
Modbus Parameter Table		AIN.VSCALE	14		32-bit		
Modbus 64-bit Parameters to 32-bit Mapping Modbus Errors		AIN.ZERO	16		Command	-	
CANopen Object Dictionary	1	AOUT.ISCALE	18		32-bit		
SynqNet Supported Parameters Frequently Asked Questions (FAQ)		AOUT.MODE	20		16-bit	-	
📄 Glossary		AOUT.OFFSET	22		16-bit, signed		
		AOUT.PSCALE	24	Yes	64-bit		~

In addition to the standard parameter table and the Modbus 64bit Parameters to 32bit Parameters there are also registers that can be mapped to variables in the AKD BASIC Program so a HMI or PLC can read/write values from/to the AKD BASIC program. In this case there is a specific section of Workbench under AKD BASIC Proramming Manual->AKD BASIC Language->AKD BASIC Modbus. Per below the available address range is from 5000 to 5999 ( each 16 bit integer registers ).

🥹 Kollmorgen WorkBench Help	- □ >
KOLLMORGEN®	Search Search
Contents 📃 Index	
Welcome Page Vorkbench User Manual KD Parameter and Command Reference Guide KD BASIC Programming Manual Using the Program View AKD BASIC Language AKD BASIC Language AKD BASIC Program Structure Program Memory Space Reference	You are here: <u>AKD BASIC Programming Manual &gt; AKD BASIC Language</u> > AKD BASIC Modbus <b>AKD BASIC ModBus TCP/IP</b> . Modbus TCP/IP, or Modbus TCP, is a Modbus variant used for communications over TCP/IP networks, connecting over port 502. A max of 3 masters can be connected to one drive at any time. Modbus standard limits one master to 256 slaves. Modbus and Workbench (telnet) can be connected to the same drive at the same time. Because Modbus and Telnet are processed in the AKD's background task, 5~10mS delay between messages will prevent over running the back ground task stack limits. ModBus Parameter Table
	For a list of parameters and their Modbus addresses, visit the <u>Modbus Parameter Table</u> . For 64 bit to 32 bit mapping, visit <u>Modbus 64-bit Parameters to 32-bit Mapping</u> <b>ModBus Register and Data Types</b> All predefined AKD parameters are Modbus 32 bit or 64 bit (some with and some without sign). Because most HMI products don't support 64 bit numbers, the AKD also has a series of matching 32 bit parameters. Ex: PL:FB is a signed 64 bit integer with address number 588. There is also PL:FB_32 as a signed 32 bit integer with address number 2072. PL:FB_32 is the lower two 16 bit registers of PL:FB. Typical HMI tag addressing will need the additional "40000" be added to the Modbus address number. Ex: To create a tag in an HMI to read PL:FB_32, the address number 42072 should be used. All AKD command parameter will require writing a "1" to trigger the command. Ex: DRV:EN is used to enable the drive. To trigger the command, send "1" as a 32 bit integer to address 254. <b>User Created Variables with Assigned Modbus Address Numbers</b>
ARU BASIC Parameters and Statements     Appendix A: BASIC Sample Programs     AKD Fieldbus Manuals     Frequently Asked Questions (FAQ)     Glossary	User variables can be assigned an Modbus address number. The range of available numbers is from 5000 to 5999. An example program: Dim int2 as integer Dim log1 as float Dim log1 as long MBInfo SMEMap22(5001, int2)

Note in order for the 5000 range (AKD BASIC variables Modbus) registers to be valid and not produce Modbus errors on attempt to access the AKD BASIC program must run and on execution declare variables (and possibly initialize them to default values) and map them to specific Modbus registers (in the 5000 range). For the first example we will read the value of a variable in the AKD BASIC program to confirm we have Modbus communications.

Create a New Program in the AKD BASIC drive once online with Workbench.

Next create a new variable called "var1".



You'll note there is a wand with a star on it in the toolbar this is the Wizards icon. One of the choices is "Modbus Wizard".



Selecting the Modbus Wizard calls up the following window that allows you to map a Modbus address to the variable we just created.

16-bit register
<ul> <li>32-bit register</li> </ul>
64-bit register
<ul> <li>Hoating point register</li> </ul>

Click on the + button in the bottom left side of the Modbus Wizard window. It will add a variable starting at 5000. At this point it is possible to make changes to the setup for this first item in the list so I changed the variable name to "var1" and the type of register to "32-bit". This means over Modbus the high word will be 5000 and the low word is 5001 ( 2 consecutive 16 bit registers ). It is possible to use dynamic mapping to do word swapping but often the Modbus TCP master can handle it if required. This is beyond the scope of this Quick Start. Click ok to accept the changes.

Modbus Wizard		- 🗆 ×
Nodbus Map Items	Address :	5000
Address Variable Name Type of Register	Variable Name :	var1
5000 myVariable1 RegisterBit32	Type of Register :	<ul> <li>16-bit register</li> <li>32-bit register</li> <li>64-bit register</li> <li>Floating point register</li> </ul>
<b>4</b>		OK Cancel

Note between Dim and Main the Modbus mapping is now declared MBINFO:End and inside the declaration is the mapping of the 32 bit variable, var 1, and the starting address 5000.

Simple_Modbus_BASIC.bas* ×	
1 ' Device Params	
2 ⊟Params	
3 End Params	
4	
5 ' Define (dim) Global Variables	
6 Dim var1 as integer	
7 ⊟MBInfo	
8 \$MBMap32(5000, var1)	
9 End	
10 ' Main Program	
11 ⊟Main	
12	
13 End Main	
14	
15 ' Subroutines and Functions	
16	
17 ' Interrupt Routines	
18	

Next so the Modbus TCP Master can read a value (other than zero) we will initialize the var1 variable to a value sufficiently large enough to be contained in the high and low words. I also added a main loop using the While:Wend statement. The program now looks like this:



The program is now ready for the first test. In the toolbar click on the Compile Icon.



The bottom toolbar should indicate "Compiled successfully".

📸 Error List 💻 Console 📷 Output 👼 Watch 🚂 Call Stack Compiled successfully

Next download the compiled program to the drive.

in nie, and load it into the drive.



The status bar at the bottom of the Program screen in Workbench should indicated "Downloaded successfully" and the name of the program should be at the far right.



Finally run the program by pressing the Run icon in the toolbar.



The status bar should show Running and a scanning bar to the left of the Program name should also indicate the program is running.

Debug session - Running	Ln 17	Ch 1	Simple_Modbus_BASIC
-------------------------	-------	------	---------------------

At this point the only thing the program does is map the var1 parameter to Modbus TCP addresses and initializes it to a value.

In my AKD BASIC drive the following IP Address was set:

Device Topology        4     todd_test       image: follower (Online)	Configures	<b>TIP</b> the TCP/IP properties used by different fieldbuses.
	Current settings —	
	IP Address:	192.168.0.10
	Subnet Mask:	255.255.0.0
	Default Gateway:	0.0.0.0
	DHCP Server:	0.0.0.0
	MAC Address:	00-23-1B-00-E6-36
	Configuration —	
	IP Mode:	1 - Fixed IP address $\qquad \checkmark$
	IP Address:	192.168.0.10
	Subnet Mask:	255.255.0.0
	Gateway:	0.0.0.0
Add New Device Add New Group	Apply	
<ul> <li>If ollower (Online)</li> <li>Settings</li> <li>Communication</li> <li>TCP/IP</li> <li>Modbus</li> </ul>		

I also did not want to use additional Modbus scaling in the AKD drive so under Communications->Modbus I set the Type of Scaling to "0-Drive Internal".

Device Topology 4	Modbus
<ul> <li>dd_test</li> <li>follower (Online)</li> </ul>	Configures the scaling Modbus properties.
	Type of scaling: 0 - Drive Internal V
	Use position unit from the device, velocity and acceleration will be derivatives of this unit (unit/s, unit/s^2).
Add New Device Add New Group	
follower (Online)     Settings     Gommunication     JCP/IP	
Medbus	

We are now ready to read the value using KVB. I am using v2.4 at the time of this application note. After opening the KVB application on your PC the following screen appears. For this quick start select "Create New Project".

Kollmorgen Visualiza by Kollmorgen Create New Project Create a New Project using the Wizard Open Project Open an Existing Project Upload from target Upload a project from a target	Action Builder™ 2.40
From Information Designer	Project15
Getting Started	Open Exit

Next choose the appropriate AKI model. For the quick start and demonstration I am not going to use the AKI hardware and simply simulate on the monitor of my PC. See the KVB manual for procedures for

Choose Target	Choose Controller	Select Location
below	or OPC server in the menu below	in the menu below
	AKI2G-CDA-MOD-07T-000	Next >
	Target rotate 0	~
	Touch operator panel	< Previous
	7" TFT, 16:9, 800 x 480 pixels Application memory: 200 MB Interfaces: 1 x 10/100 Mbps Ethernet 1 x USB 4 x serial ports (RS232/RS485/RS422)	Finish Cancel
		Product series
		All ~
DA-MOD-04T AKI2G-CDA-MOD-10T-0	00 AKI2G-CDA-MOD-07T-000 AKI2G-CDA	-MOD-05T-01 🔎 Search 🐼
<		>

setting the AKI's IP address, etc. I arbitrarily choose at 7 inch operator panel.

On clicking the Next button you will be given the option to select the Controller. In this case we want Kollmorgen and the Modbus TCP protocol.

Choose Target	Choose Controller	Select Location
Choose your target in the menu	Choose your preferred controller	Select the location of your project
below	or OPC server in the menu below	in the menu below
Controllers Select brand  DEMO  G & L Motion Control  Kollmorgen  OPC UA Server URL:  OPC Classic Server Localhost  Remote Server OPC Server:	Select protocol Modbus Master RTU/TCPIP Brows	Next > Previous Finish Cancel

Click Next and give the project a name and a path to where you want the file(s) to be placed. Then click Finish

Choose Choose below	your target in the menu Choose Controller Choose your preferred control or OPC server in the menu bel	ller ow Select the location of your proj
		Next >
Name:	SimpleExample_Part_2	
Location:	C:\Users\Todd.Evans\Documents\Kollmorgen Projects	Browse < Previous
		Finish
		Cancel

The project will now be created and you will have the development environment opened at this point:

	Properties	Kollmorgen Visualization Builder <sup>®</sup> 2.40 - Simple	Example_Part_2		- 8 ×
Home Project System	n Insert View Dynamics General Actio	ns			
Paste ↓ Cut La Copy ✓ Format Painter Add Screen •	Parent Screen     Parent Screen       Parent Screen     Parent Screen	A A A A A A A A A A A A A A A A A A A	Image: Shape Fill -         Select Tag           Image: Shape Outline -         Select Security Groups           Other         Image: Shape Effects -           Colors         Image: Shape Effects -	Screen1 Default	•
Clipboard	Screen Objects	Font	Format Fig / Security	Name Design La	nguage
Project Explorer 👻 🖣 🗙	Screen1 ×				
▼ Screens (1)					
All Screens					
Screen1					
Search (S)					
▼ Functions (4)					
Alarm Server				🕬 🔲 👬 Fit to Screen 🔹	• = •
Multiple Languages	Layout Script Xaml Aliases		eie D- D-		Tags used: 0
Output					<b>•</b> 1

The first thing that must be done is to configure the communications so KVB knows the Modbus TCP slave's IP address.

Use the scroll bar on the left in the Project Explorer to scroll down and then click on Tags. The Tags window will appear and there are tabs Tags, Controllers, Triggers, Poll Groups, and Index Registers. Click on the "Controllers" tab.

,													,	A			<u> </u>		
Project Explorer	ųΧ	Screen1	× Tags ×																
	^	1	Tags																
Screen1		Tags Co	ontrollers Trigge	ers Poll Gro	oups Ind	lex Registers													
		Home																	
					Column	s Visible								Filter					
		Add	• Del	lete 👻	Scali	ing	Others								Cross Re	ference	Show Se	election •	Import
					Data	Exchange													
		Tag					Controllers			Scaling				Others					
		Name		D	ata Type	Access Right	Data Type		Controller 1	Offset	Gain	Read Expr	Write Expr	Description		Poll G	roup	Always Active	Non-volatile
		> Tag1		DE	EFAULT	ReadWrite	DEFAULT				0	1				PoliGre	oup1		
Search	2																		
▼ Functions ( 4 )																			
Alarm Server																			
Multiple Languages																			
Security																			
Tags		Design S	Script																Tags used
Output					_			_			_					_	_		

Click on the "Settings" button:

Screent × Tags ×		
Tags Controllers Triggers Poll Groups Index Registers		
Home Add Delete		Controller Settings Show Selection •
Name	ID	Active
> Controller1		
Design Script		Tags used: 0

To help with communication traffic I changed "Silent Time(ms)" on the Settings tab to 20 msec. Note the default is Ethernet TCP/IP communication mode. Next click on the Stations tab.

Modbus Master RTU/TC	PIP			$\times$
Settings Stations				
Modbus Master RTU/TC	PIP 5.02.02			
Settings				<u> </u>
Communication mode		Ethernet TC	P/IP	
Default station		0		
Modbus protocol		RTU		
32-bit word mapping		Motorola		
Addressing		Decimal		
Start address		0-based		
Max block size (words)	)	16		
Force function code 0:	x10	Disable		
String swap		Enable		
Silent time (ms)		20		
Clock register		40000		
Communication me	ethod			
Advanced				<b>-</b>
Silent time (ms) Extra time (ms) that the o RTU the silent time is add	communication lir led to the norma	ne has to be silen I silent time of 3.	nt between two te 5 characters.	elegrams. For
	ОК	Cancel	Apply	Help

On the Stations tab I changed the IP Address to the same IP Address as the target AKD BASIC drive. Then I clicked "Apply" and then "OK".

Modbus I	Master RTU/TC	PIP			$\times$
Settings	Stations				
Station	IP Address	Port	Node		
0	192.168.0.10	502	1		
L					
				Add	Remove
		ок	Cancel	Apply	Help

Click on the Tags tab and add the following tags. Note the tag names are the same as the variable names declared in the AKD BASIC program and the Tag Data Type was configured for INT32 as well as the "Controllers" Data Type in the Tag chart below. Under Controller 1 it is necessary to declare the Modbus TCP address for each variable ( tag ). Also take note that although the starting addresses in the AKD BASIC were 5000 and 5002 respectively KVB requires that you use 40000 + AKD Modbus Address so in this example 40000 + 5000= 45000 and 40000 + 5002=45002.

	Screen1 × Tags ×							
and the second	👌 Tags							
T	ags Controllers Triggers Po	ll Groups In	ndex Registers					
ſ	Home							
	Add 🔸 Delete	▼ Colum ✓ Sca	ns Visible iling [ ta Exchange	☑ Others				
-	Tag			Controllers		Scaling		
	Name	Data Type	Access Right	Data Type	Controller 1	Offset	Gain	Read Expr
	var1	INT32	Read	INT32	45000	0	1	
I	var2	INT32	ReadWrite	INT32	45002	0	1	

Before concerning ourselves with writing the value we want to prove the project can communicate and simply read var1 from the AKD BASIC program.



Click on Screen1 and select the Analog Numeric button from the toolbar.

On selection click on the screen's workspace and use the crosshairs and your mouse to add the data field.



With the data field selected as above click on the General pull-down menu which will show the default format is Integer. There is a "Select Tag" list box you can click on to list the tags declared in the Tags Table.

	Prop	perties	Kolimorgen visualization bui	lder 2.40 - 5im
rt View Dynamics	General	Actions		
		• 🗖	Auto Stretch Font	e <mark>e</mark> i e
Tags	Aliases			
refix/Suffix Name				<u>ant</u>
en1 × Tags × var1				
var2				
				Г
			OK	Cancel

Home Project System	Insert View I	Dynamics General Actic	ons			
Integer    Zero Fill  Limit Characters to 10  Number of Decimals 1  Display Format	Prefix	Select Tag • Select Security Groups • Default • Tag / Security	Auto Stretch Font     Disable Operator Input     Transparent     Format	Text Alignment	On Input On Display Lower Limit O Upper Limit O Validation	
Project Explorer 🗸 🕂 🗙	Screen1 × Tags ×					
Screen1			ſ	366.	0	50.0
© © @ Search © ▼ Functions (4) ↓ Alarm Server ↓ Multiple Languages			L		23	50

We want to select var1.

				F	rope	rties		Kolimorge	en visua	lizauon	builder	2.40	) - Sim
rt	View	Dyn	amics	Gener	al	Actio	ns						
					•	· ···		Auto Stret	ch Font			=	
			Tags	Aliases									
		11	÷										
refix/	/Suffix	-	Name	2									ant
en1	× Tags	×	var1										
			var2										
									0	)K	Ca	ncel	
										/1%		ncer	

We're ready to Build and Run the Project. First selected the Project menu and clicked on Build ( there should be no errors after building in the Output window at the bottom of KVB. Next I clicked on Run.



Because I am running the project on my PC instead of downloading it to an actual KVB unit it states "Running in demo mode- time left X minutes". Note the value of var1 is 1234567 which is what it was initialized to in the AKD BASIC program.

		$\sim$
1234567		
1254567		
Running in demo mode- time left: 30 minutes		

Click the X ( close window ) in the simulated screen to end the simulation.

Next we will add the ability to enter a value from the screen and write to var2 ( and see it change in the AKD BASIC program ).

I simply added another Analog Numeric field on the screen and configured it for the tag var2.

I then selected the Project Menu and then Build and Run as before.



On simulation you'll note the value of var2 is still 33333333 since I left it in the same state as Part1.



To change the value I click on the field which pops up a Keypad. I changed the value to 11111111.





Checking back at the Console on the Program screen of Workbench the value indeed changed.

Con	sole		
	var2=	11111111	
	Input:		

It is possible to use buttons to trigger values as well. To demonstrate I made the following additions to the program. I declared a variable called pushbutton1, mapped it to the next starting Modbus address, commented out the first print statement so we can focus on the button for now and added a new print statement to monitor the variable pushbutton1.



I Saved, Compiled, and Downloaded the program to the AKD BASIC.

Moving back to the KVB software and the Tags table, I added a new tag named it pushbutton1, set the data type to INT32 ReadWrite and set the Controllers Data Type to INT32 and set the address to 45004.

Tags Controllers Triggers Poll G Home Add • Delete •	Groups Inde	ex Registers					
Tags Controllers Triggers Poll G Home Add - Delete -	Groups Inde Columns	ex Registers					
Home Add • Delete •	Columns	Visible					
Add - Delete -	Columns	Visible					
	Data I	ng Exchange	✓ Others				
Tag			Controllers		Scaling		
Name	Data Type	Access Right	Data Type	Controller 1	Offset	Gain	Read Expr
var1 I	INT32	Read	INT32	45000	0	1	
var2 I	INT32	ReadWrite	INT32	45002	0	1	
I pushbutton 1 J	INT32	ReadWrite	INT32	45004	0	1	

From the toolbar in KVB I selected the button object.



Draw the button on Screen1. Select the Actions menu and in this case I set the Mouse Down and Mouse Up function to "Set Analog", pointed the operation to the variable "pushbutton1" and set the value to 1 when the button (mouse) is pressed DOWN and a value of 0 when the button (mouse) is pressed UP (i.e. released). Note it is possible to on Click Set Analog to a constant value (i.e. 1). You will find this is the case in the sample AKI to AKD BASIC program. I encourage you to download the sample project from our website and open it up offline to explore the objects and their configurations.

Set Analog	Set Analog 🔹
nuchhuttan1 *	nuchhutten 1
pushbutton	pushbutton
1	0
୮୬ Mouse Down ୮୬	Mouse Up 🕞
259.0	
	"
DuchButton1	
62.D	503.0
	7
90.0	

Now we are ready to simulate. Build and Run as before. Note in the AKD BASIC console, the value is 0 without the Pushbutton on the screen being pressed.

[ 🕐 - 🗷 🚰 - 뇌 교 김 승규 - 🕹 는 🕒 🖉 이 연 🖄 소 🕨 🔟 🔲 🔲 Debu	ıg - View - 📑 📰 🌮 - 📩 🔓 ≪	🔑 All 🔹		
Simple_Modbus_BASIC.bas  X SimpleExample_Part	_2 -		– 🗆 X	-
1         '			1	
13 ⊟Main 14 var1=2234567 15 while 1=1 16 'Print "var2=",var2 17 Print "vusbutton1=",ousbbutton1		1234567		
18         Wend           19         End Nain           20            21         '	node- time left: 29 minutes			v
<	PushButton1			>
Console           Image: Pushbuton1 = 0           Pushbuton1 = 0           Pushbuton1 = 0           Pushbuton1 = 0		0		ф 
Pushbutton1= 0 Input:				~
Debug session - Running	Ln 16	Ch 23		Simple_Modbus_BASIC

While holding pushbutton on the screen down, the value in the program changes to 1.

	Program			Learn more about this topic
	This page is used to edit the BASIC program file and load it into the dr	ve.		
19+	2 🚰 - 日日は鳥 - よした ツマ 出土ト	🕅 🔲 🖬 Debug - View - 🖃 🗐 🜮 - 🟦 🔓	🧠 🔑 All 🔹	-
Simple	Modbus_BASIC.bas 🕯 🗙	SimpleExample_Part_2 -		 ×
	1 ' Device Params 2 [Farams.] 4 Define (dim) Global Variables 6 Dim vari,var2, pushbuttoni as integer 7 Define (dim) Global Variables 9 [SHBtep32(Seee, var1) 9 [SHBtep32(Seee, var2) 9 [SHBtep32(Seee, var2) 1 [End 1 vari-1224567 14 vari-1224567 15 while 1=1 16 [*rint "var2= ",var2	Running in demo mode- time lef	t: 27 minutes	
Consol P P P In R	17     Print "Pushbutton1= ",pushbutton1       18     Wend       19     Iend Main       20     21       21     '	PushButton1	0	2 4 2 2
Debug s	ession - Running	Ln 16	Ch 23	Simple_Modbus_BASIC