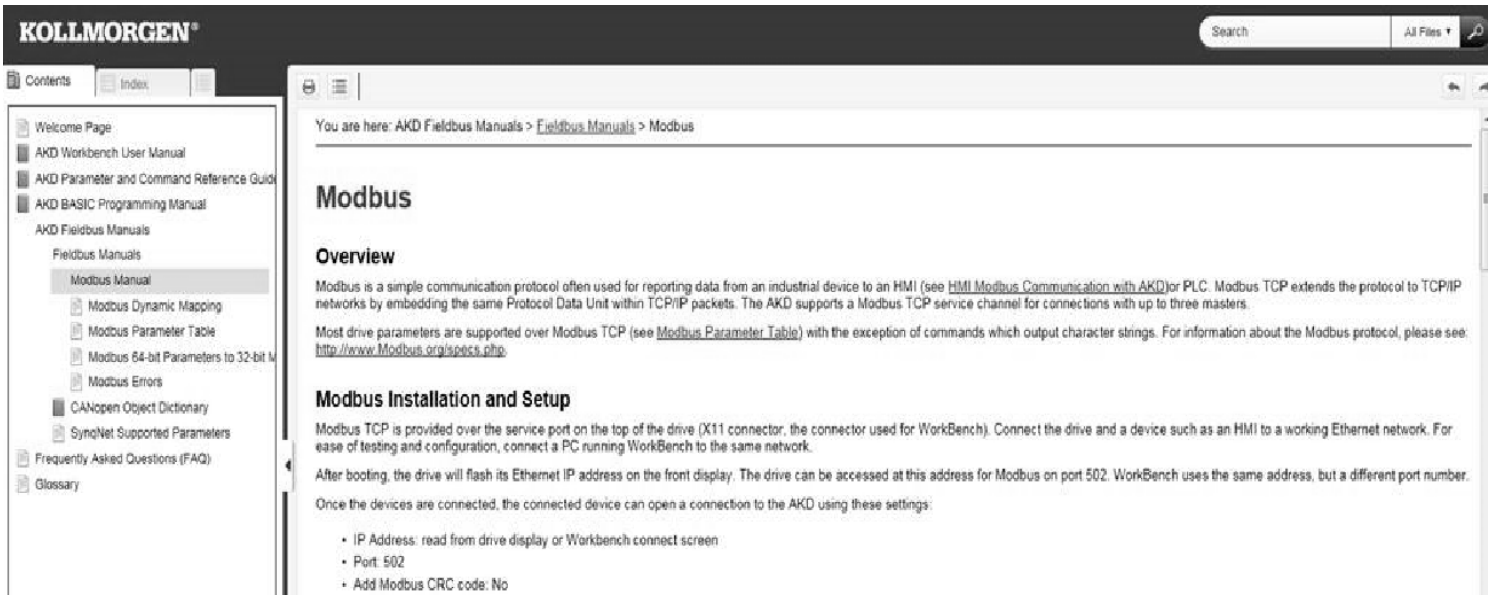
Here are some related articles at the following link that may be helpful.

http://kdn.kollmorgen.com/query/modbus%20motion%20tasking

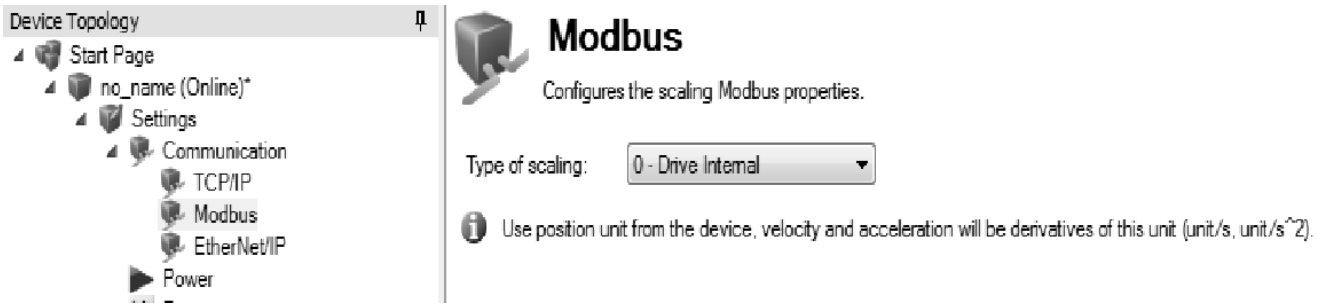
In general the Modbus TCP documentation can be found with the online web help or alternatively as shown below in the Workbench help menu under AKD Fieldbus Manuals->Fieldbus Manuals->Modbus Manuals (as shown below).

Dynamic mapping, special Modbus parameters, the parameter table, and a shorten list of parameters where some of the 64 bit parameters have been remapped to 32 bit Modbus addresses to make the interface from HMIs, PLCs, etc. that often work with 16 and 32 bit words easier.



An important note when being to work with the AKD using Modbus TCP:
The Modbus TCP standard as dictated by the Modbus.org calls for the ability to do scaling over Modbus. In a majority of the applications most users want the values in the Modbus controller to match the values seen when monitoring in Workbench.

To make them 1:1, under the Communication->Modbus->Type of scaling in Workbench, set the type of scaling to “0-Drive Internal”.



The gist of loading a motion task involves several steps:

Step 1: MT.NUM must point to the row in the drive's motion task table that you want to either load (read) or set (write).

Step 2: To create a motion task from scratch you set the MT parameters to values as needed. In the simplest form is the following (in this example we are setting motion task#1).

I demonstrate values that are arbitrary. Set them as needed for your application.

MT.NUM 1 (point to motion task 1, this is variable)

MT.ACC 10000

MT. DEC 10000

MT.V 100

MT.P 10000

MT.CNTL 0 (0=absolute move, 1=relative to command position, etc. see Workbench help for the parameter description).

Step 3: Send/Write the value to the motion task

MT. SET 1 (send/write the values to the motion task)

A key point is that the MT.V, MT.P, etc. are all essentially intermediate containers for holding the values to be sent or the values read. They will either be sent down to the motion task with the MT.SET or change to the values uploaded from the motion task (MT.LOAD). Either case uses MT.NUM as the motion task pointer.

The values are not written to the motion task until the MT.SET is executed.

Another point is even after the MT.SET is successfully executed, the motion task table is volatile the motion task table is volatile meaning unless a DRV.NVSAVE is executed, on power cycle the values written to the motion task via MT.SET will be lost.

DRV.NVSAVE	938		Command

Sometimes the motion task is already created and you need to only change one setting.

For example suppose you have created a motion task where the velocity, accel, decel, etc will always be the same but you want to vary only the position.

Step 1: MT.NUM 1 (again in this example we are pointing to motion task#1, this is variable)

Step 2: Upload (read) the values from the existing motion task using the MT.LOAD command

MT.LOAD 1

At this point all MT.x parameters will be set to the value stored in the motion task pointed to by MT.NUM.

Step3: Change the MT.x parameters as needed. In this example, I am only changing the position so

MT.P 20000

Step4: MT.SET 1 (send/write the values back down to the motion task)

As stated before, the motion task table is volatile meaning unless a DRV.NVSAVE is executed, on power cycle the values written to the motion task via MT.SET will be lost.

DRV.NVSAVE	938		Command

Abbreviated Modbus Parameter Tables showing all MT.x parameters:

Modbus Parameter Table

Parameter	Modbus Register Address	Is 64-bit?	Attributes
MT.ACC	526	Yes	64-bit
MT.CLEAR	530		16-bit, signed
MT.CNTL	532		32-bit
MT.CONTINUE	534		Command
MT.DEC	536	Yes	64-bit
MT.EMERGMT	540		16-bit, signed
MT.LOAD	542		Command
MT.MOVE	544		Command 16-bit
MT.MTNEXT	546		8-bit
MT.NUM	548		8-bit
MT.P	550	Yes	64-bit, signed
MT.SET	554		Command 8-bit
MT.TNEXT	556		16-bit
MT.TNUM	558		8-bit
MT.TPOSWND	560	Yes	64-bit, signed
MT.TVELWND	564		32-bit
MT.V	566	Yes	low 32-bit word
MT.VCMD	568	Yes	low 32-bit word, signed

The 32 bit versions of MT.ACC, MT.DEC, and MT.P. These will also change the MT.ACC, MT.DEC, and MT.P if written to.

MT.ACC_32	2056	Yes	low 32-bit word
MT.DEC_32	2058	Yes	low 32-bit word
MT.P_32	2060	Yes	low 32-bit word, signed

To execute/start a motion task:

MT.MOVE	544		Command 16-bit

Starts a motion task; active in opmode 2 (position) only.

Description

MT.MOVE starts a motion task. This command needs one argument in order to start a motion task. The drive must be homed, otherwise the motion task will not start (see also HOME commands).

Example

MT.MOVE 3 -> Start motion task number 3.

In some applications it is desired to jog the drive via Modbus. This can be done by using the Service Motion parameters.

SM.I1	746		32-bit, signed
SM.I2	748		32-bit, signed
SM.MODE	750		16-bit
SM.MOVE	752		Command
SM.T1	754		16-bit
SM.T2	756		16-bit
SM.V1	758	Yes	low 32-bit word, signed
SM.V2	760	Yes	low 32-bit word, signed

It is also possible to execute the homing as set by the homing type and other settings in the Home screen in Workbench.

The HOME.MOVE when set over Modbus performs the same function as clicking on the “Start” button in Workbench on the Home screen.

HOME.MOVE	408		Command