Modbus Addressing and Troubleshooting Rev. A 6.14.2016

It is important when working with Modbus to establish the addressing conventions in the beginning before programming your Modbus Master (i.e. PLC, HMI, etc.) and understand how the master's addressing convention correlates to the AKD drive's Modbus addressing.

In general, regardless of data size the AKD Modbus parameters are reserved with either 32 bits or 64 bits of space.

For data <=32 (one bit, 8 bits, 16 bits, 32 bits etc.) 32 bits (two consecutive 16bit words) are reserved whether the data requires all of it or not.

For data=64 bit then there are four consecutive 16 bit words reserved.

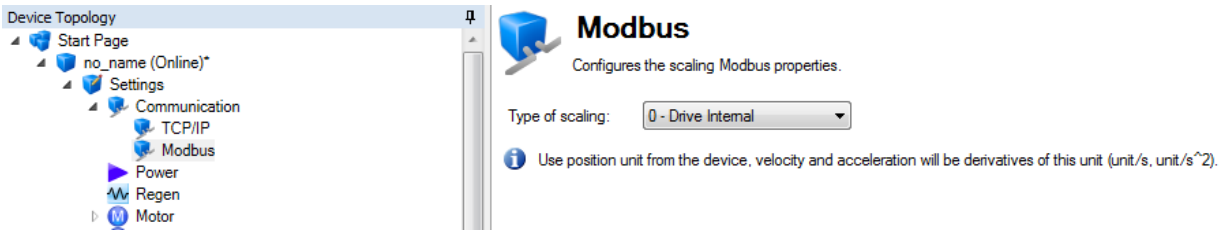
Parameters that have the attribute "command" have 32 bits or two 16 bit words reserved (more explanation below) and the general method for sending the command is to write a value of 1 to the Modbus register. Once the value has been written and the event/result confirmed, generally you want to set the value back to 0.

Modbus Parameter Table

Parameter	Modbus Register Address	Is 64-bit?	Attributes
AIN.CUTOFF	0		32-bit
AIN.DEADBAND	2		16-bit
AIN.ISCALE	4		32-bit
AIN.OFFSET	6		16-bit, signed
AIN.PSCALE	8	Yes	64-bit, signed
AIN.VALUE	12		16-bit
AIN.VSCALE	14		32-bit
	16		Command
AOUT.ISCALE	18		32-bit
AOUT.MODE	20		16-bit
AOUT.OFFSET	22		16-bit, signed
AOUT.PSCALE	24	Yes	64-bit
AOUT.VALUE	28	Yes	64-bit, signed

For example, from the Modbus Parameter Table above, the first register is address 0 and the numbering is based on Modbus registers being at a minimum 16 bit words. AIN.CUTOFF is a 32bit parameter so the starting address is 0 but the data is contained in Modbus address 0 and address 1.

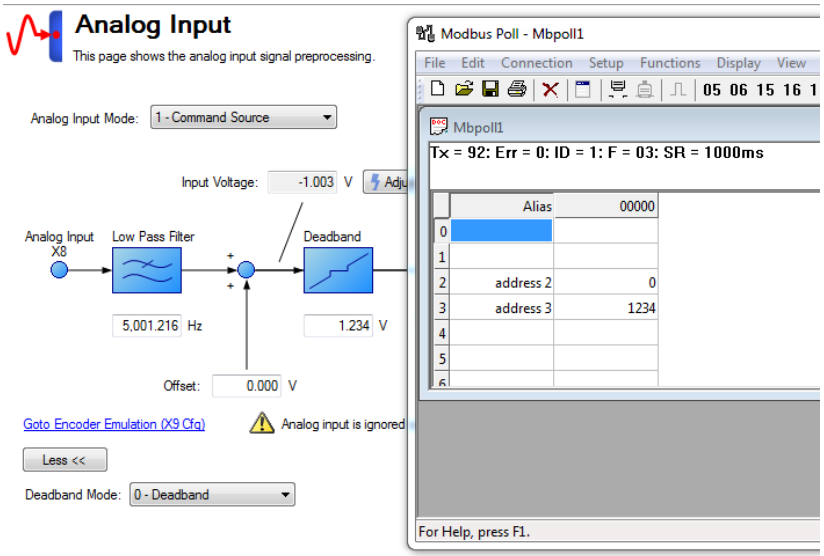
Whether addresses 0 and 1 is the high word or low word will depend on your master and whether word swapping is required or not in order to get the data in the master to match the data displayed in Workbench. Also note that Modbus provides a method for Modbus Specific scaling but most of the time users forgo using that method and set the type of scaling to 0-Drive Internal which uses the units/scaling in Workbench. This usually allows the data in the master and data in the drive to correlate (i.e. 5000 from the master=5.000 inches in Workbench, etc.).



Next note in the Modbus Parameter Table the next parameter AIN.DEADBAND has a starting address of 2 (since the previous parameter consumed addresses 0 and 1 this is intuitive).

AIN.DEADBAND is a 16 bit parameter but note the next parameter AIN.ISCALE starts at register 4. Per the above, AIN.DEADBAND has addresses 2 and 3 reserved even though the data is 16 bit. The master that I use to test Modbus communications actually displays the data in address 3 and the data in address 2 is 0 (unused). Sometimes users get confused because the Modbus Parameter Table in Workbench shows the starting address only.

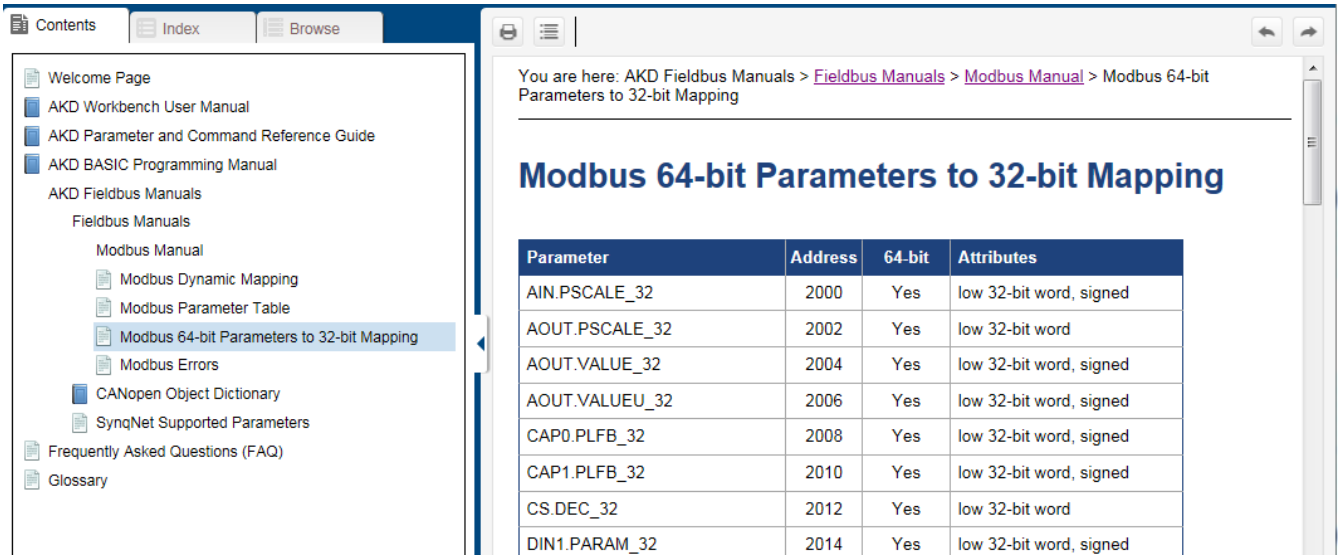
To demonstrate how to establish addressing conventions, I manually set the deadband value to 1.234 and set my Modbus master to read address 2 and address 3 and the data appeared in address 3 (see below). You will need to determine which address the data appears in because what the master calls the high and low word may differ than the AKD drive. Some Modbus masters give you a configuration option to automatically do word swapping. My master starts at Modbus address 0 so I didn't need to offset my addressing (more on offset below).



For a 64 bit parameter (i.e. AIN.PSCALE) the starting address is 8 but 8,9,10, and 11 are consumed.

One option which I have found to simply things is to declare your variables, data, mappings, etc. as all 32 bit (DINT).

Limit your data so that it conforms to the attribute (i.e. 16 bit, 32 bit). For 64 bit parameters always look in the Modbus 64 bit to 32 bit mapping chart in Workbench Help first (starting Modbus address 2000) to see if a 32 bit version of the parameter exists.



Obviously if you want to map only a 16 bit word when the parameter requires only that, it will work, but again be aware of two common issues in mapping Modbus addresses: 1) word swapping 2) offset.

Word swapping: If the master doesn't agree with the AKD in regards to what the high word and low word addresses are you can potentially write to the high word with unexpected results in the data passed.

Offset occurs often because a lot of Modbus masters start their holding register counts at 1 or 40001 or 400001 or some similar convention. The AKD starts at 0 which may equate to 1 in the master. In the example above, the AIN.DEADBAND had a starting address of 2 in the table but addresses 2 and 3 reserved. In my Modbus master the data appeared in address 3. In a different master that starts at, 40001 for example, the address could be 3 and 4 (add +1 to each address for the offset). In my case, the Modbus master starts at 0 so I don't have to worry about offset.

For 2000 (64 to 32 bit mapped parameters) or 8192 (dynamic mapping) you may find the starting address to be 42001 or 48193 for example.

For AKD BASIC variable Modbus addressing (range 5000-5999) you may find the addressing to be 45001 for example.

It is up to the programmer to determine the correct addressing conventions in the master and how they correlate to the AKD's Modbus addressing. Using software based tools like Modbus Scan, Modbus Poll, and similar Modbus master software are often useful for troubleshooting Modbus communication issues experienced with another Modbus master (i.e. PLC, HMI, etc.).