Monitoring Current over Ethernet IP using the AKD-P-NBEI drive

If you want to read the drive's current, my opinion is using an AOI for this purpose is not a very good method.
As mentioned before, Ethernet/IP is not a super-fast communication bus. RPI scan (for each Generic Ethernet Module) is absolute fastest at 20msec. Keep this in mind in regards to monitoring or sampling current values.
Only one AOI per axis can be executing at a time.
Whether it is a AKD_Get_Parameter or AKD_Get_Attribute…they both tie up your EIP communications because all AOIs use the same static bytes 0-35 of the command and response assemblies.
I see a note in the manual that a response type can be viewed I presume by requesting the response type 5 in the command assembly but I've never used it.
I would expect this to be a conflict because other AOIs call Response Type 14 Command/Error Response to detect an error.
I've also not used the Controller Attribute that is shown in the Controller Attribute Listing.
A "torque" parameter does not exist in the AKD drive so only milliamps or amps can be queried regardless of EIP or not.

6.2.3.4 Response Type 0x05 - Actual Torque

This I/O response assembly is used to return the actual torque (current) of the motor in milliarms. Data will be received in the Data field, bits 4-7. Set Response Type = 0x05 in the command assembly to read this value.

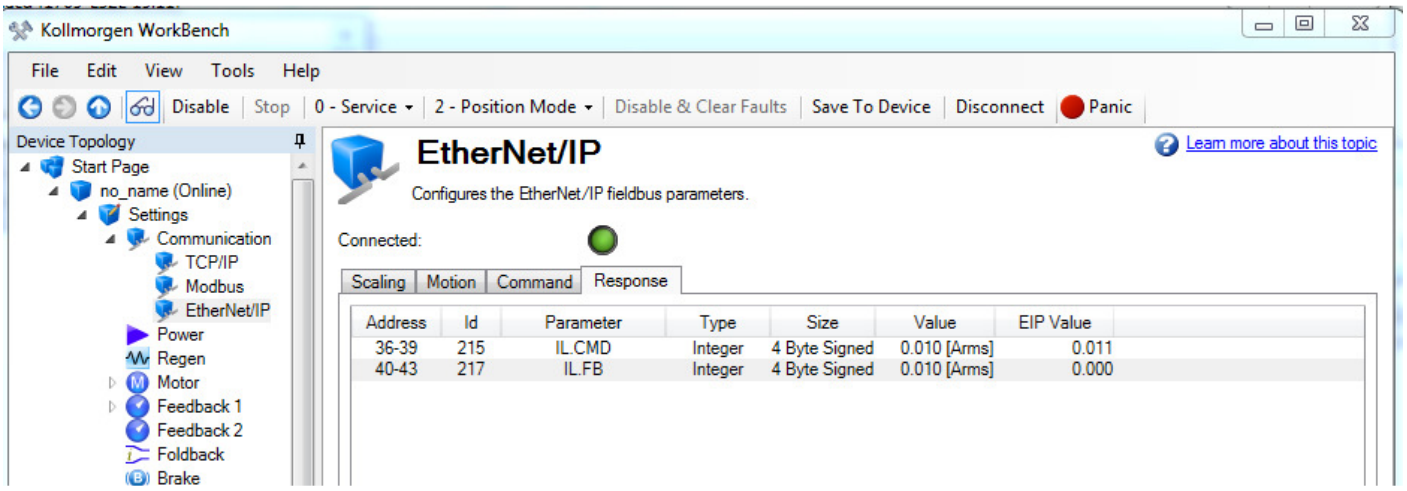
25	Torque	Get/Set	DINT	Output torque.

If you want to monitor current in the drive you can look at IL.FB (feedback) or IL.CMD (command).
We do not have a filtered parameter for the current loop like we do on the velocity loop so you're going to find the value is going to bounce around quite a bit (IL.FB may be more noisy than IL.CMD). You can monitor both as well if you'd like.

I would presume you want to read the value all the time.
The best method for this is dynamic mapping (cyclic data) as opposed to triggering MSG (explicit) message blocks all the time
with a timer or to a similar effect the AKD_Get_Parameter AOI (remember only 1 AOI per axis can be executing at a time).
Dynamic mapping puts the parameters on the cyclic data (command (write); response (read)) which gets updated every RPI scan.
Bytes 36-63 of both the command assembly and the response assembly can have AKD parameters mapped.

Instance	Parameter	Data Size	Data Type
215	IL.CMD	4 Byte Signed	Float
217	IL.FB	4 Byte Signed	Float

Map the desired parameters in the Response assembly since you want to read. Use the latest Workbench version (1-17). An earlier version had a bug with using the GUI
to dynamically map. Under the Settings->Communications->Ethernet/IP navigation and the "Response" tab you can map your desired instances.



Mapping does not put the parameters on the poll; it merely maps them.

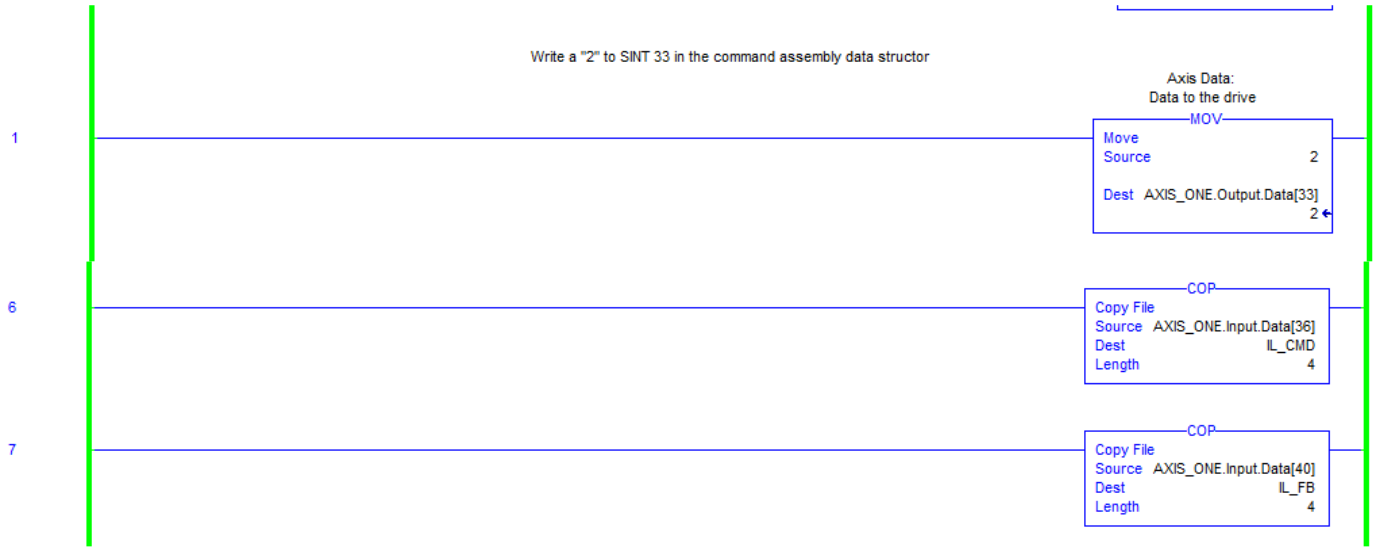
To enable polling it is required to move a value of “2” into byte 33 of the command assembly.

Ethernet/IP Communications | 6 Communication Profile

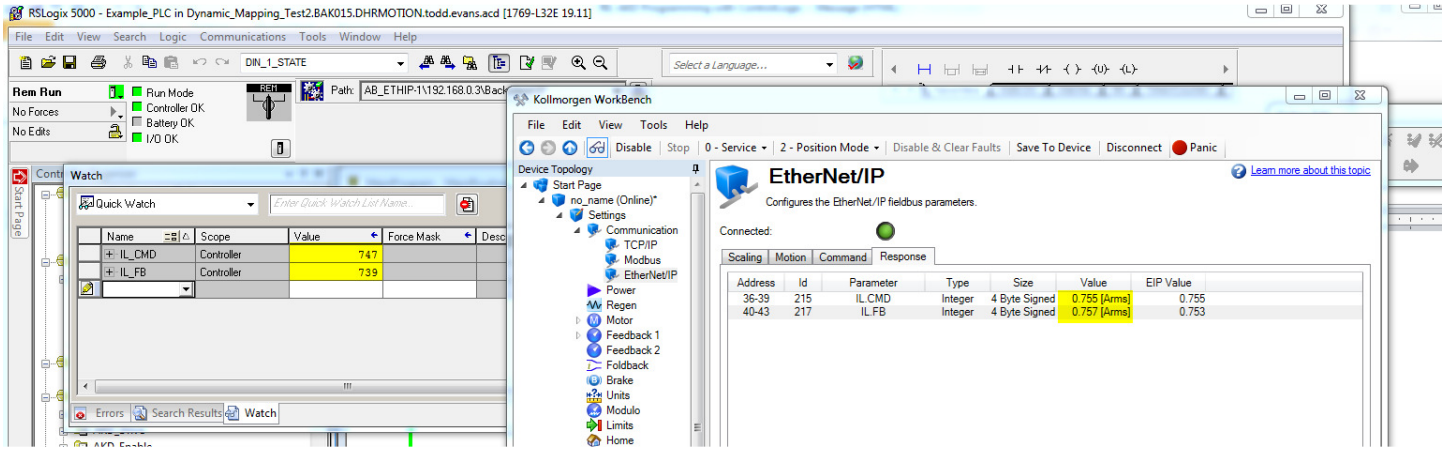
Byte	Data	Comment
33	Map Type	0: Static Map (only bytes 0 to 35 are sent) 1: Custom Map 1 2: Dynamic Map (bytes 36-63 are dynamically configurable)
34-35	Reserved	
36-63	Command Dynamic Map	See EIP.CMDMAP (➜ p. 32).

*Least significant byte first for all data fields

Below you can see I did just that and also copied the data into tags in the ladder using the COP instruction.
The size of the data is 4 byte signed or a DINT data type which is what I declared the IL_CMD and IL_FB tags as in the PLC's controller tags.



Note the values in the PLC are x1000 since these are DINT (double integers). I monitored both the PLC data using the Quick Watch and Online in Workbench on the EIP screen.



I wrote 2 articles related to this.

https://kdn.kollmorgen.com/content/akd-ethernet-ip-diagnostics-and-dynamic-mapping

https://kdn.kollmorgen.com/content/reading-drive-parameter-akd-ethernet-ip

Using the latest 1-17-0-0 firmware, I checked writing to IL.LIMITP (positive current limit), IL.LIMITN (negative current limit) and then PL.ERRFTHRESH (maximum following error threshold).
They worked fine. Because the AB PLC works in DINT or INTs, 4 bytes is the maximum I could write even if it is an 8 byte parameter in the drive.
There is a known issue in the firmware in regards to writing negative values to some parameters. One note on this is there is a unreleased beta that fixes this problem.
I anticipate it will be rolled out as an official production release sometime later this year. It will enable the usage of 32 bit (4 byte) versions (instances) of 8 byte parameters.
I was able to monitor, IL.CMD (current loop command), IL.FB (current loop feedback), PL.ERR (position loop error; again an 8 byte parameter but I could only read a value up to 4 bytes in the PLC) over Ethernet IP.
The PL.FB (position loop feedback) is built in to the response assembly of Ethernet IP.