**Connecting Motion Control in the Factory**Carroll Wontrop  
  
December 7, 2017

The need for Ethernet-based communication within the factory has grown over the last 10 years. Detailed information about the tasks and performance of machines at every stage is critical to productivity. As capabilities have expanded from performance reporting to predictive maintenance warnings and beyond, the need to connect multiple controllers within the same machine or even elsewhere within the factory has also grown.

*This paper reviews options that are available for Ethernet and non-Ethernet connections within the motion controller. We will begin with a summary of what information can be transferred in and out of the motion controller followed by network and non-network examples for communicating with controllers. Finally, we will present the factors that need to be considered when choosing the right solutions for your factory*

**Machine Control Formats**

There are two formats when incorporating high performance servo motion control in a machine. The first (figure 1) has separate motion and machine controllers. The motion controller focuses on motion and the machine control (resident PLC or PC) handles the rest of the machine control. The advantage of having a separate motion controller is it will often have more functionality and processing power dedicated to performing the machine’s motion. Examples: more motion types such as camming, more operational modes such as: torque speed, gearing, and more I/O, plus with motion control as the central focus, higher motion performance is often possible.

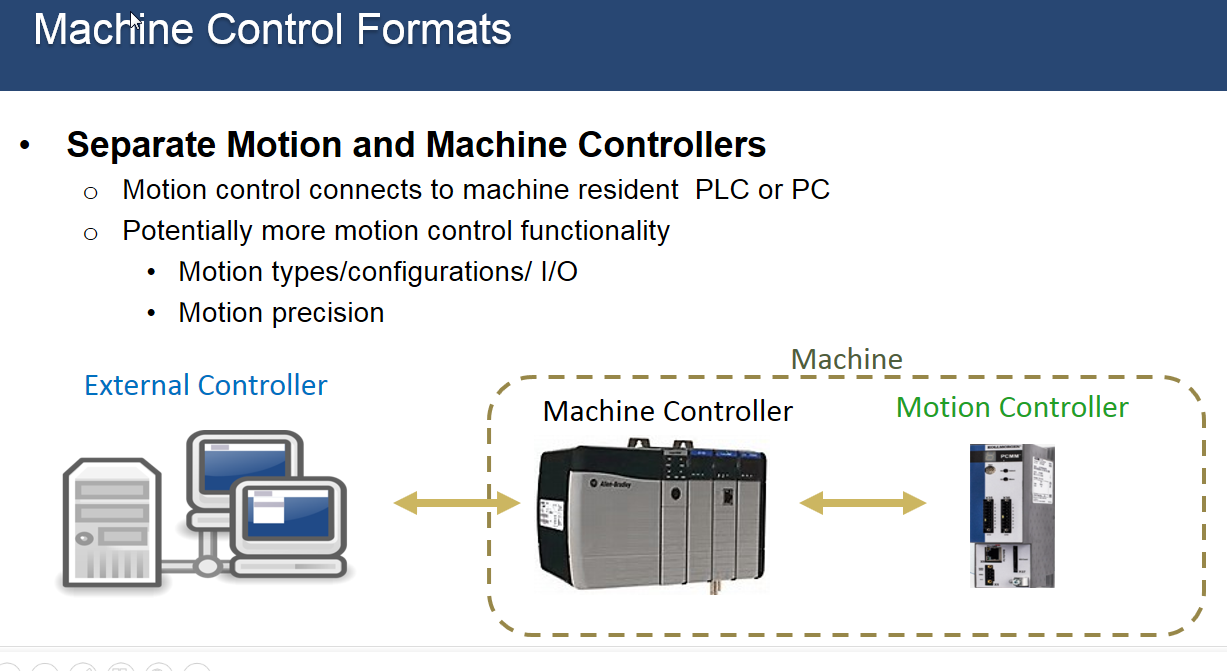


Figure 1

The second type of machine control (figure 2) combines both the machine and motion control into one controller. With increasingly powerful processors, there are PLC now available to incorporate the motion control. Also, some non-PLC type controllers will include motion control with the other machine control in the CPU. There are also traditional motion controllers with the processing power to provide complete control eliminating the PLC. The advantage of the integrated control is potentially lower total machine control cost.

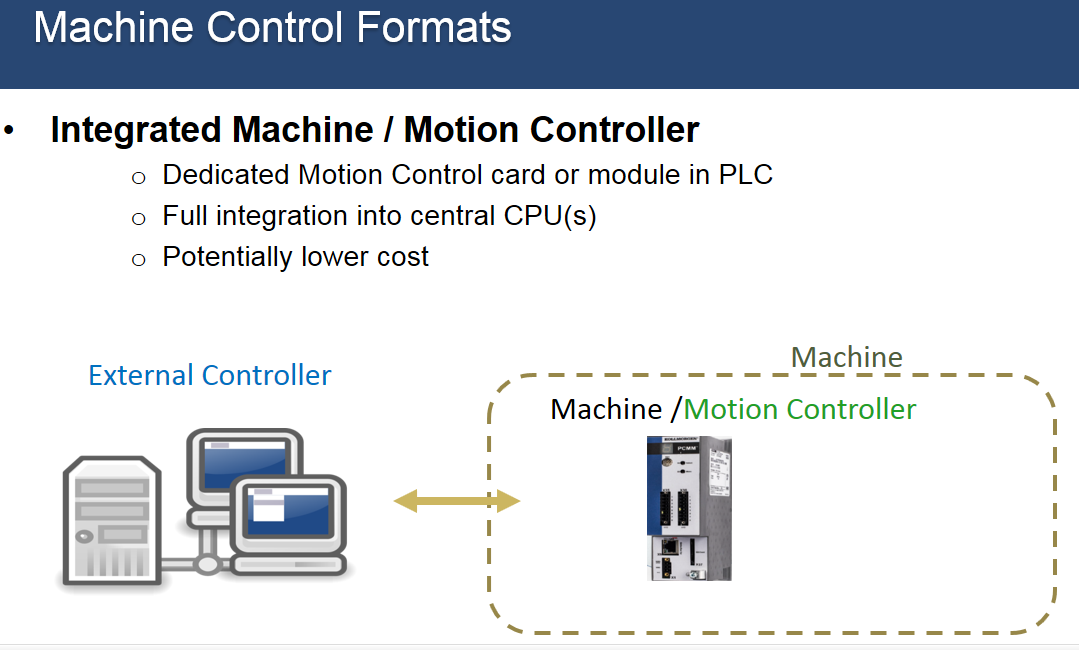


Figure 2

**Information Transferred**

Motion controllers are used in a variety of industries including medical, laboratory automation, robotics, printing, labeling, material forming, pharmaceutical, packaging and converting, food and beverage, tire and rubber, and postal sorting.

The primary data out of motion controller (Figure 3) is performance related. For example, how well a machine is making the “widgets”, how many widgets have been made, machine errors or limitations, and unexpected variations such as a heater on the machine that is not operating at the expected temperature.

While knowing that a machine is operating properly and at peak efficiency provides peace of mind, knowing when it is not running properly can save money, time and resources. Suppose you have a motor drawing more current than it should indicating a mechanical problem or a cut-to-length machine making improper cuts? Having a system programmed to provide machine performance data is critical to identifying and fixing these issues quickly.

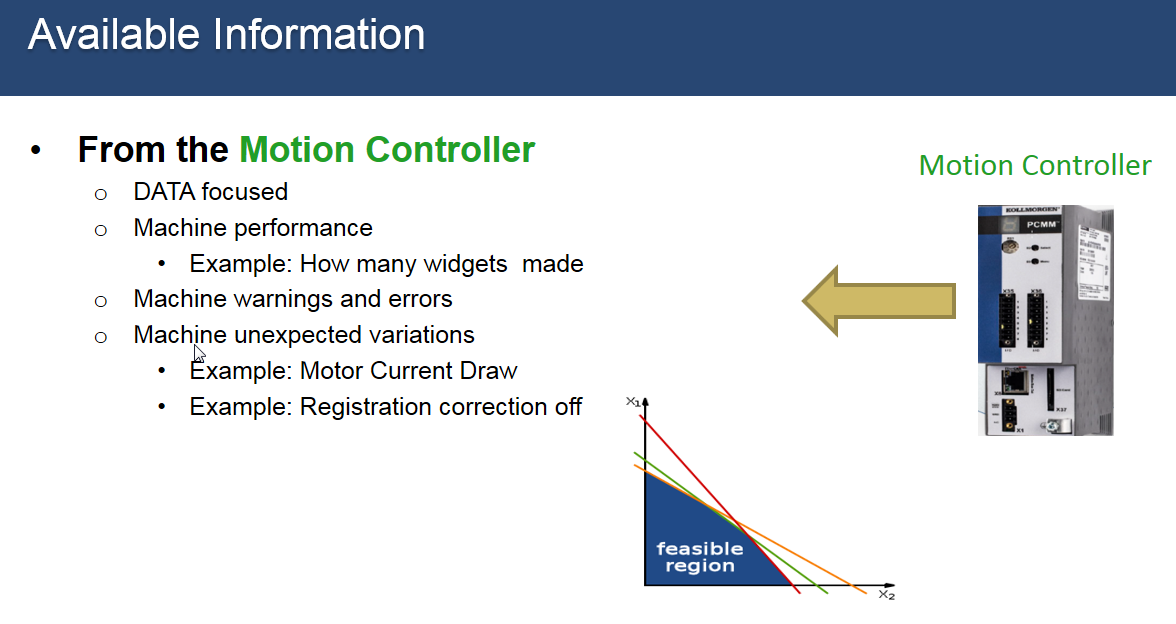


Figure 3

Information moving in the opposite direction, from the machine controller to the motion controller, is more command focused (Figure 4). This type of information might include instructions or recipes for machine set up such as part numbers or the parameters for making a specific widget type. It may also include motion specs such as distance, speed, camming motion points and other operation details. An example would be temperature changes in the heater required for the next part to be made.

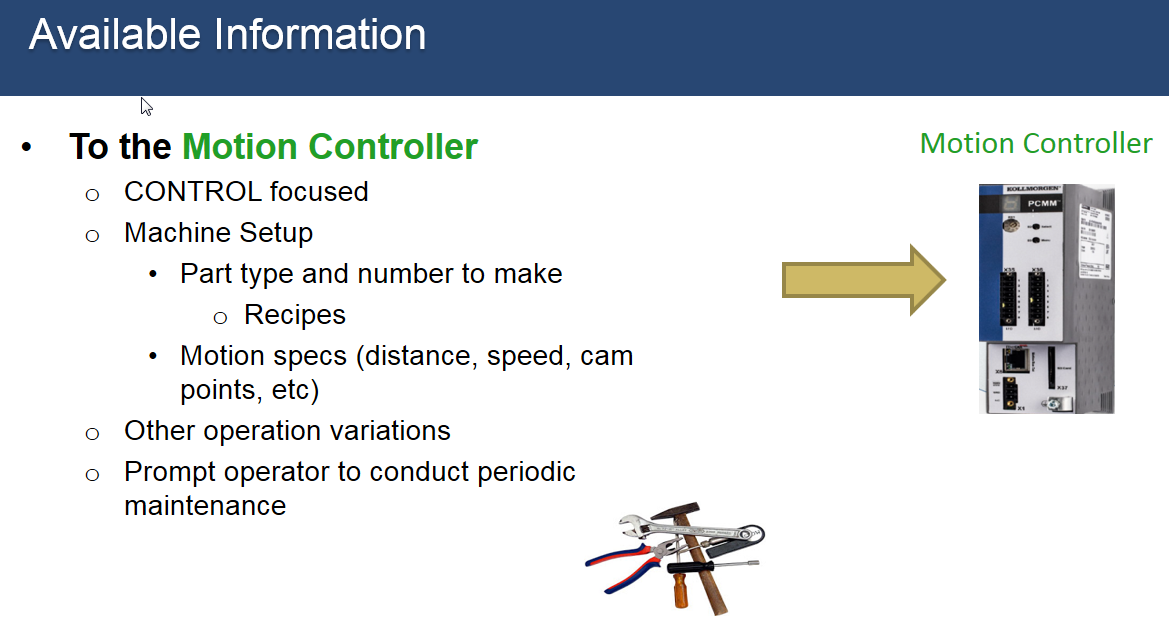


Figure 4

**Information Transfer Mediums**

Three types (Figure5):

* Ethernet - PLC oriented: These are the Industrial Ethernet Fieldbuses that a lot of us are aware of. Examples: Ethernet/IP, Profinet, EtherCAT, SERCOS III, vendor specific and others. Many are incorporated in traditional PLCs.
* Ethernet - PC oriented: These are traditional PC based networks. Examples: UDP and HTTP
* Non-Ethernet network methods. Examples: web server, SD card, remote memory, FTP and VPN/eon.

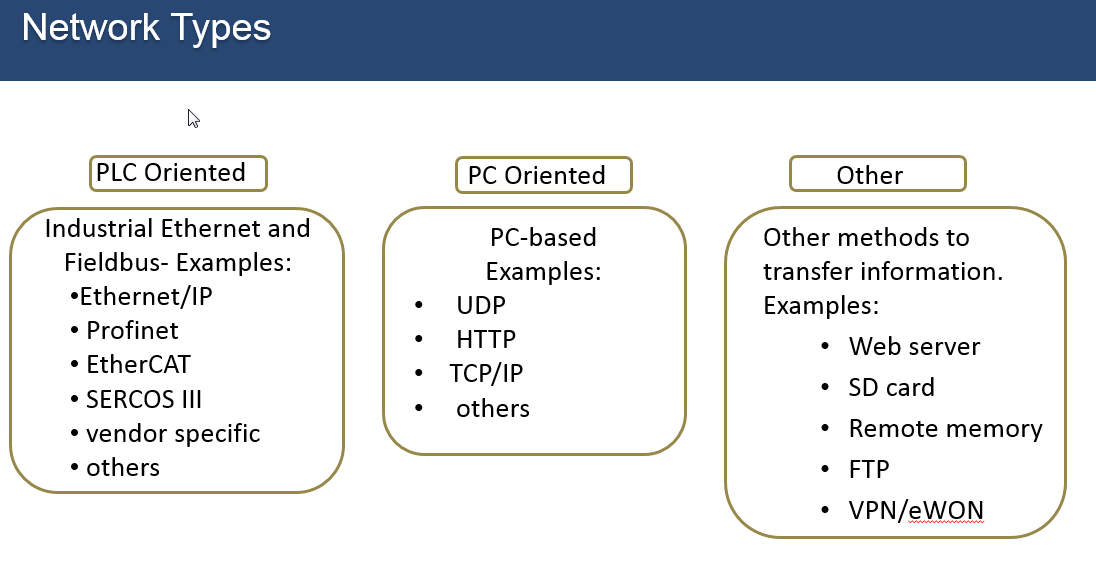


Figure 5

**Ethernet Communications Basics**

Before getting into specific networks let’s review Ethernet communication basics:

* The transportation medium is copper wire, CAT5 or CAT6 cable with RJ45 termination. Cables can be up to 100 meters in length between each node.
* The connection topology can be a line, star, or ring, although many networks only support a subset of the three.
* Built in electrical isolation which is needed in a lot of motion applications to help eliminate part variances and maintain precision motion control.
* Deterministic or non-deterministic transfer of information, depending on the network and how it is configured.
* Update times can be 500 milliseconds or higher down to 250 microseconds depending on the application and the network.
* Automatic network configuration and integrity checks are sometimes available to ensure the quality of the network transmission.
* Information passed between the motion controller and the machine controller or external controller is often called parameters, variables or tags, in the form of single objects or in the form of a data array or structure.

**Modbus TCP/IP**

The Modbus TCP/IP (Figure 6) is a long-time industrial standard built on the Modbus RTU standard. It’s used with HMIs and other devices, and is supported by many control products. Modbus TCP/IP has a standard address block scheme for transferring binary and non-binary data typically in 32-bit or 16-bit formats. It is non-deterministic, and the data update time can vary. In general, the performance range for update times between two devices is 20 to 500 milliseconds.

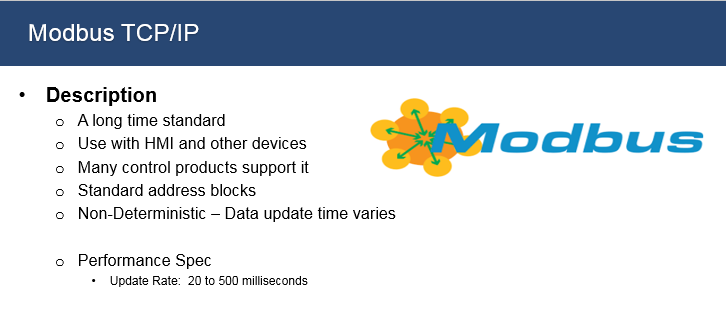


Figure 6

To integrate a Modbus TCP/IP interface into a motion controller, there will typically be a configuration setup in the motion controller’s programming software. In figure 7, on the right-hand side there is a selection for inserting a network into a Fieldbus editor.

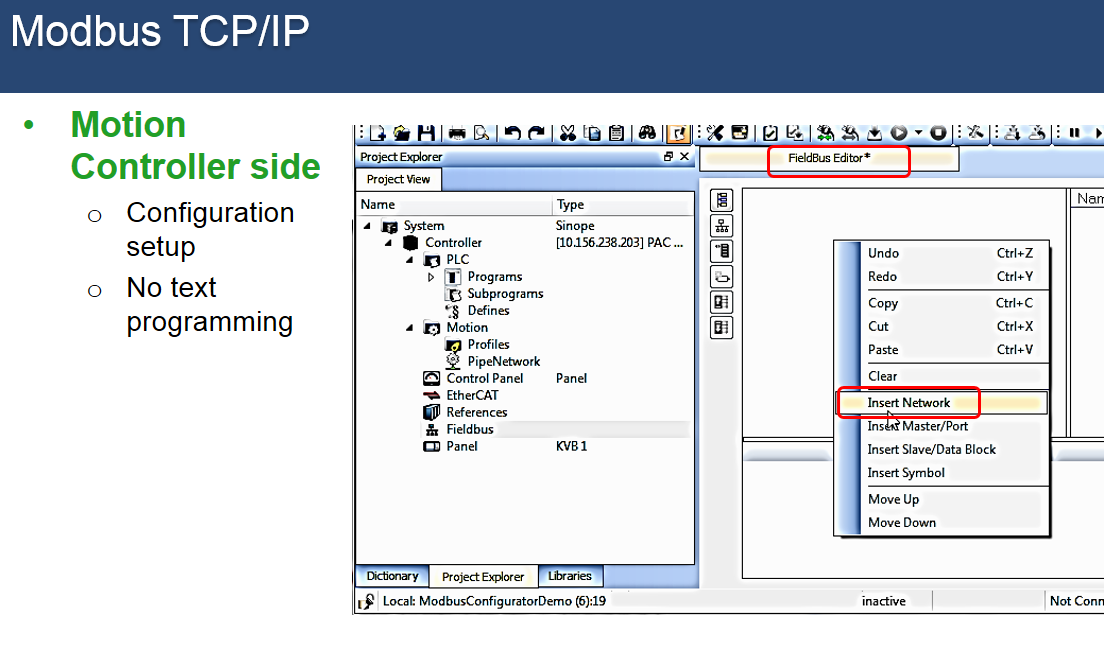


Figure 7

There is no text programming for this. Once you have set it up, it’s straight forward to link Modbus addresses to tags or variables (figure 8) already in your motion controller’s project, either by a drag and drop method or by adding them via a drop-down dialog box.

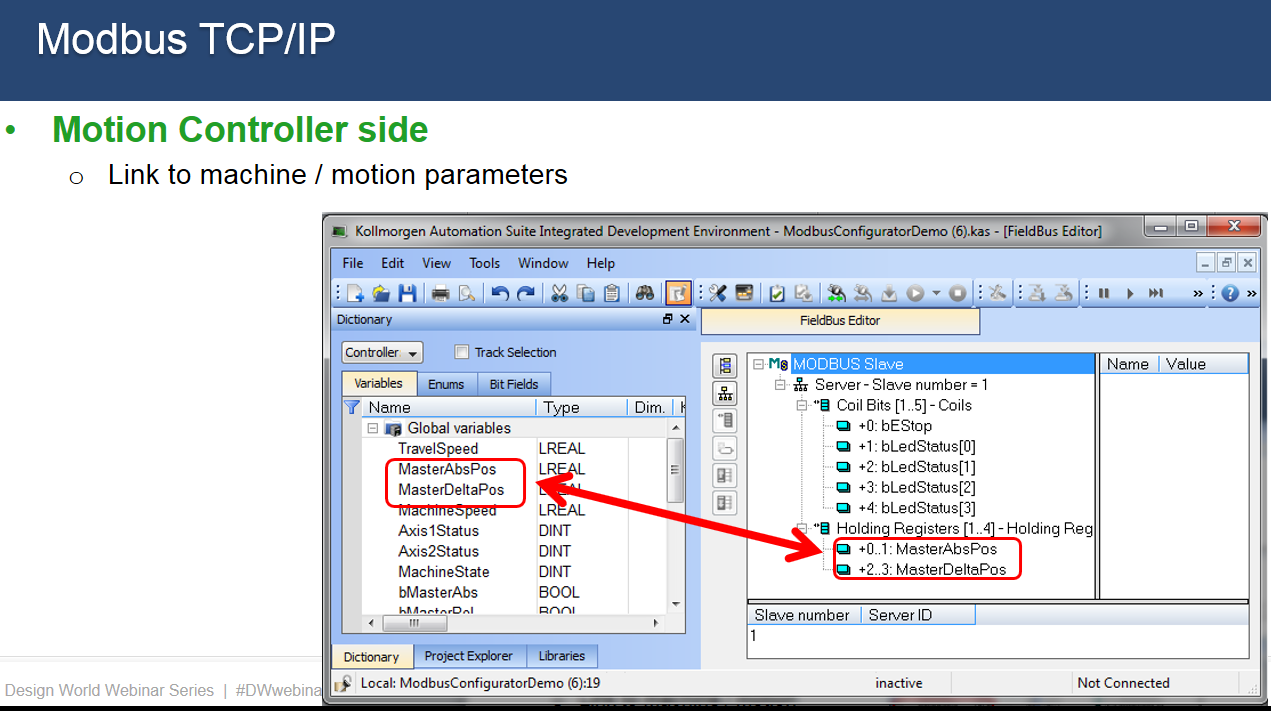


Figure 8

To setup the Modbus connection on the external controller side, there is often a set up screen (Figure 9) to configure the IP address.

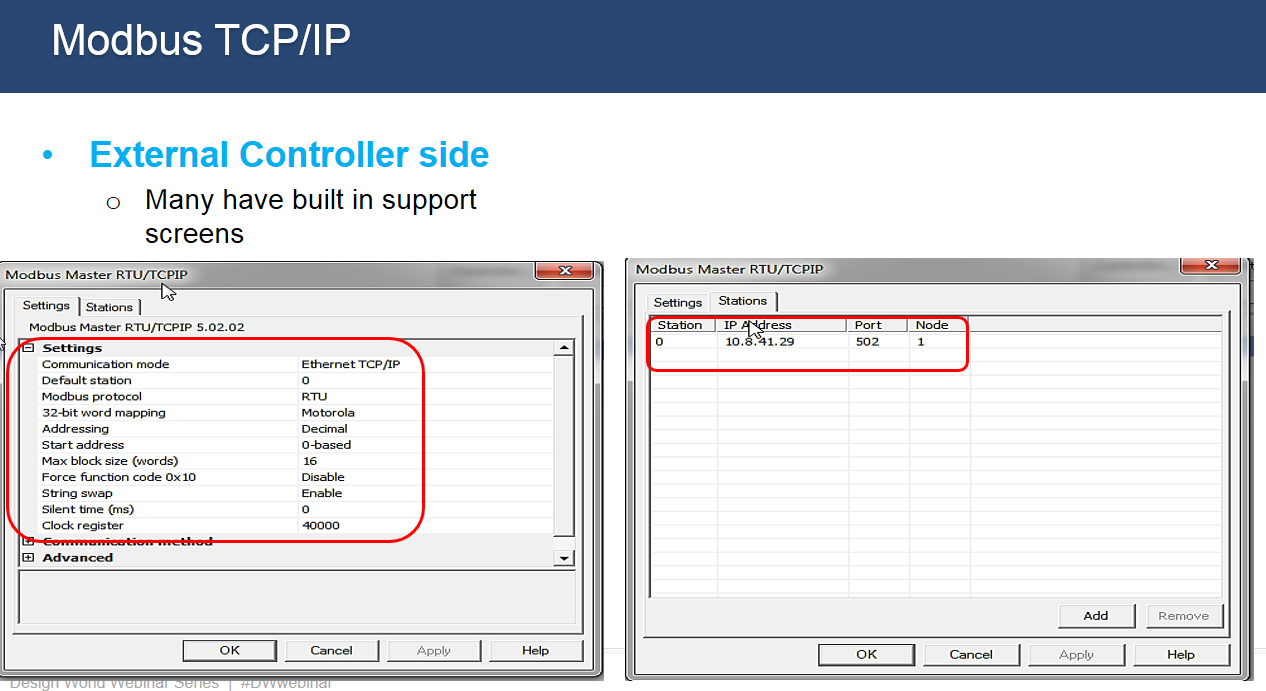


Figure 9

Then, by importing a tag file, motion parameters are available for use in the external controller’s tag or variable dictionary (Figure 10).

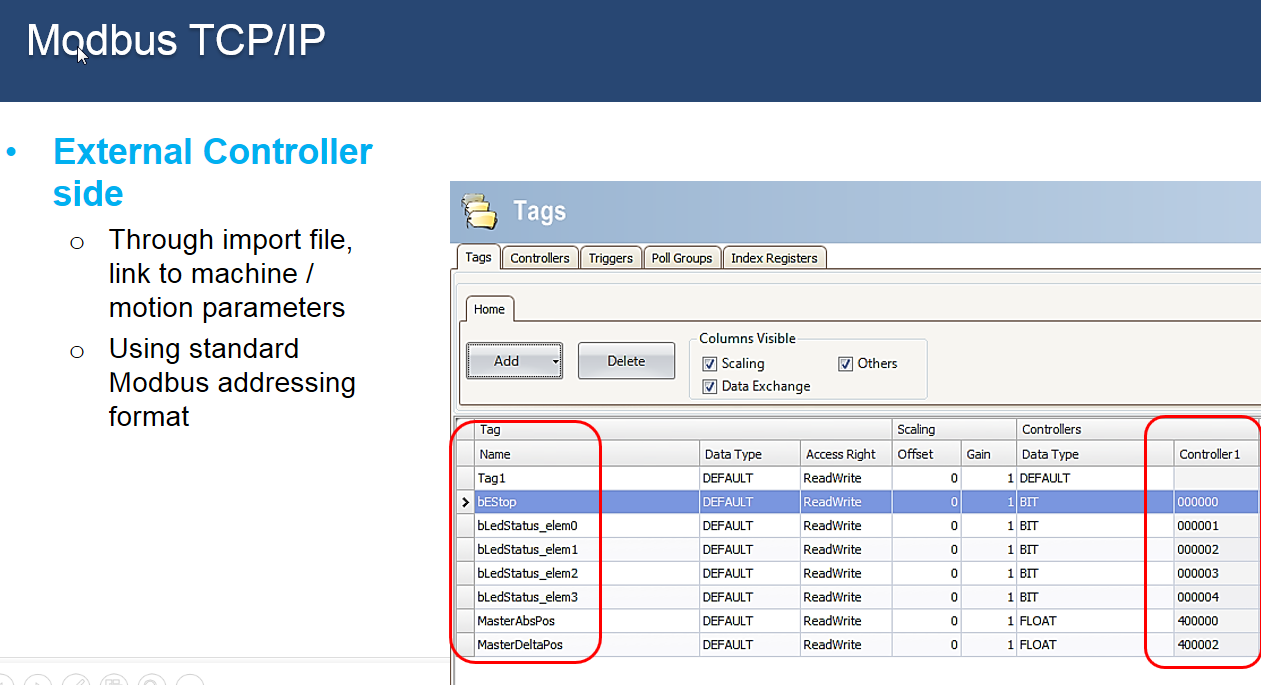


Figure 10

**Ethernet/IP**

Ethernet/IP can come in various configurations (Figure 11). It also is built into some motion controllers and can provide direct access to the motion controller’s machine and process parameters. Transmission update rates, can go down to 10 milliseconds with the Kollmorgen PxMM controller, although many applications work well with update times in the 30 to 100 milliseconds range.

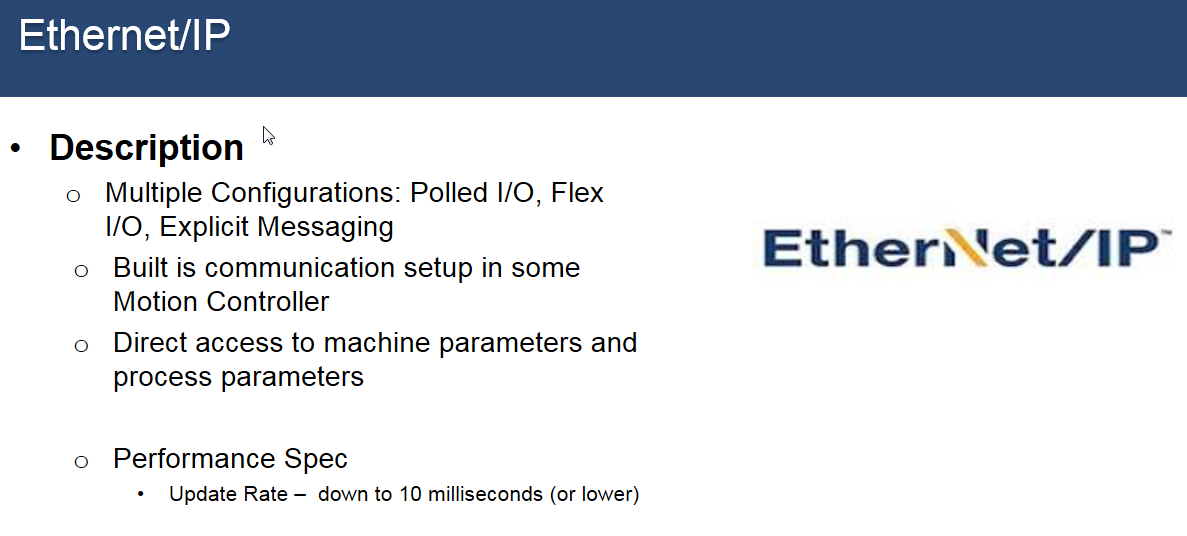


Figure 11

Implementation on the motion control side is accomplished via a set up screen that allows a user to set parameters pertinent to Ethernet/IP, and easily tie in to variable (tags) that are in the application program (Figure 12).

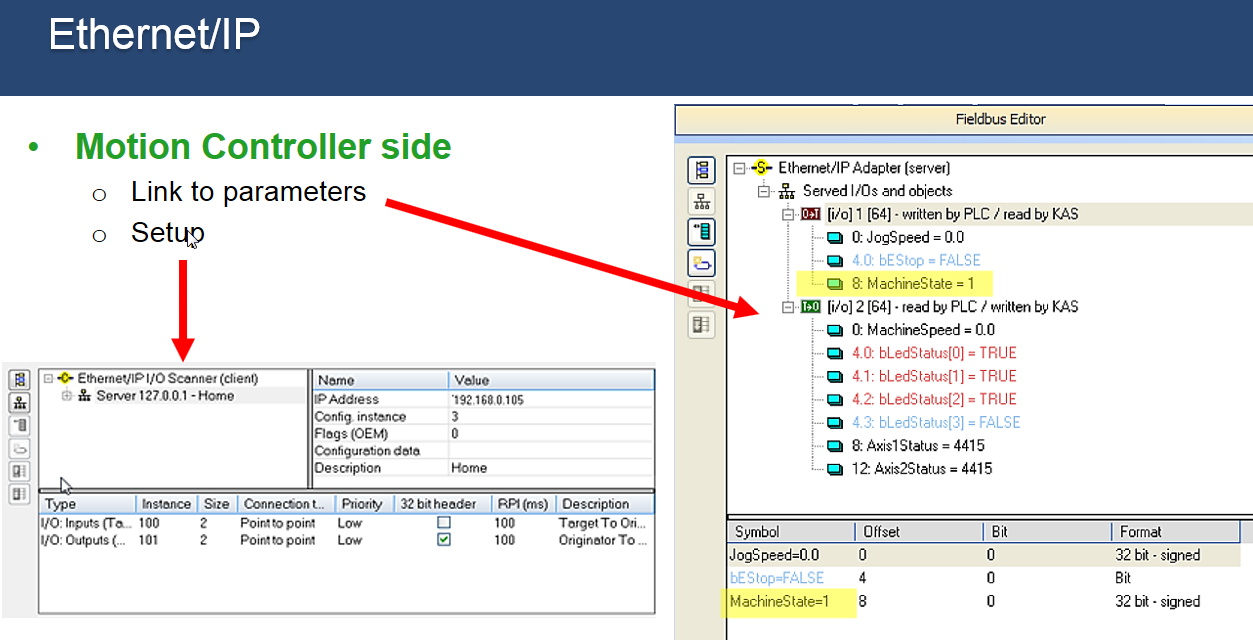
. 

Figure 12

From the external controller side (Figure 13), in this case it’s a PLC, you also have built-in set up screens. There is a screen for node set up as well as a screen to set up the data table, which are the parameters tags that are going to be passed back and forth.

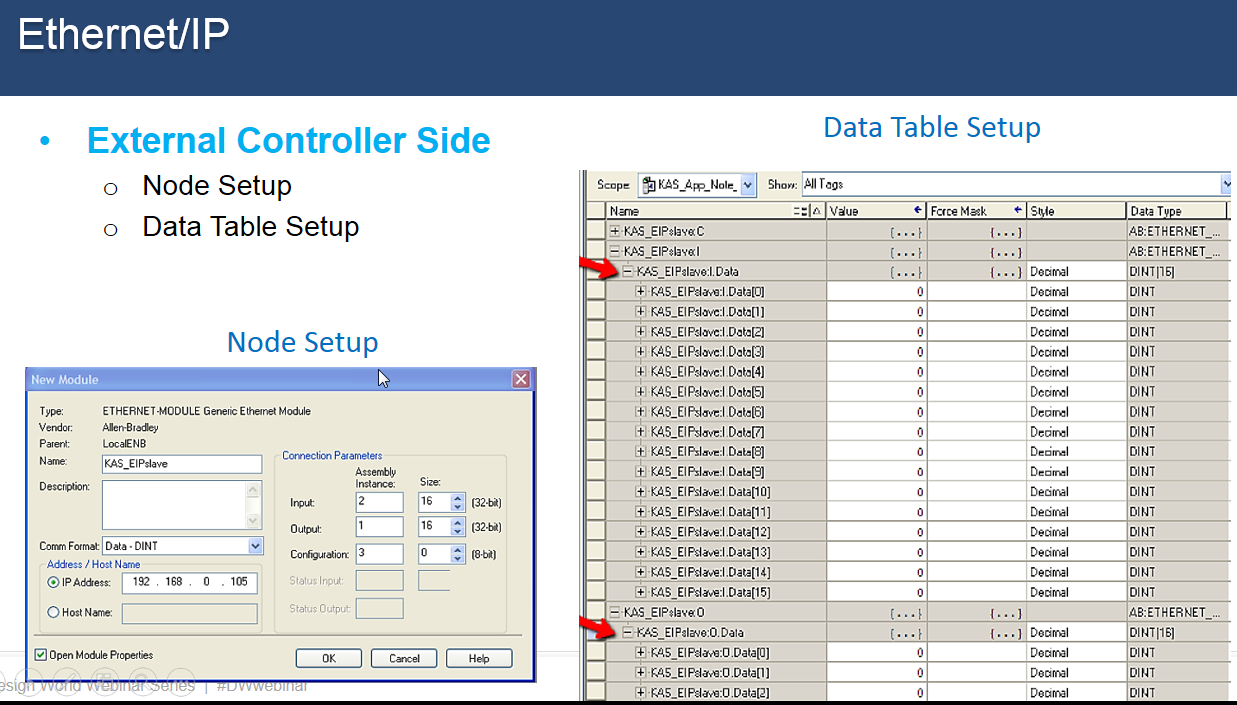


Figure 13

Figure 14 shows an example of integration in the external controller’s program code, specifically it shows a rung on a ladder that’s setting a motion parameter called machine state.

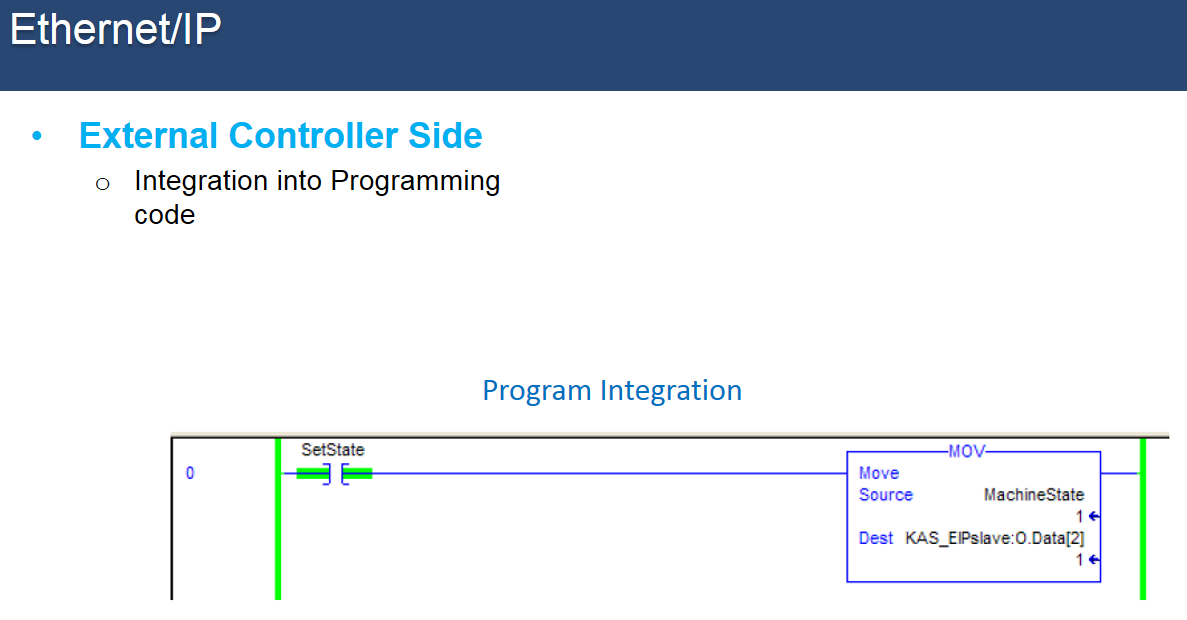


Figure 14

**UDP**

UDP (User Datagram Protocol) didn’t grow out of the industrial automation world but is now being used in different machine control applications (Figure 15). UDP is often used in applications developed with Visual Basic, Visual Studio and others. These languages have typically been used outside the PLC world, in the PC control world. UDP, with a low communication overhead, it can provide update rates down to 1-4 milliseconds.

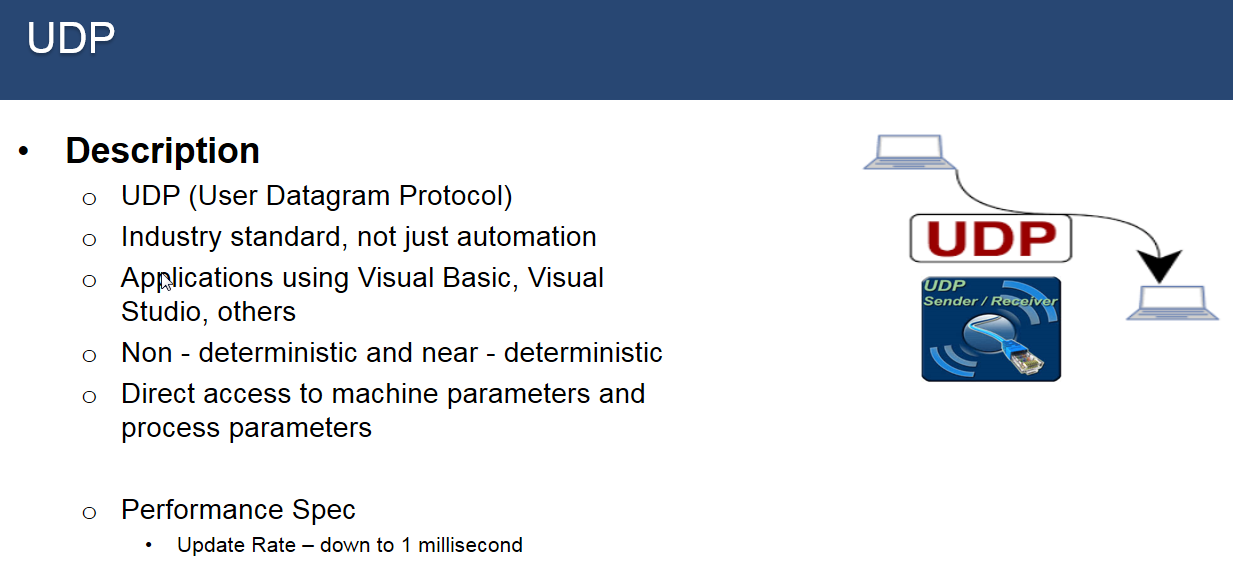
****

Figure 15

A UDP connection in a Kollmorgen PDMM motion controller is implemented differently, there is no fieldbus screen as with Ethernet/IP. Communications is set-up via PLC function blocks directly in the application program. Figure 16 shows programming code for implementing a UDP interface, specifically initializing communication by creating and validating the socket and ensuring it is ready to receive information.



Figure 16

Figure 17 then shows sample code to read incoming information and parse it into the motion controller parameters.

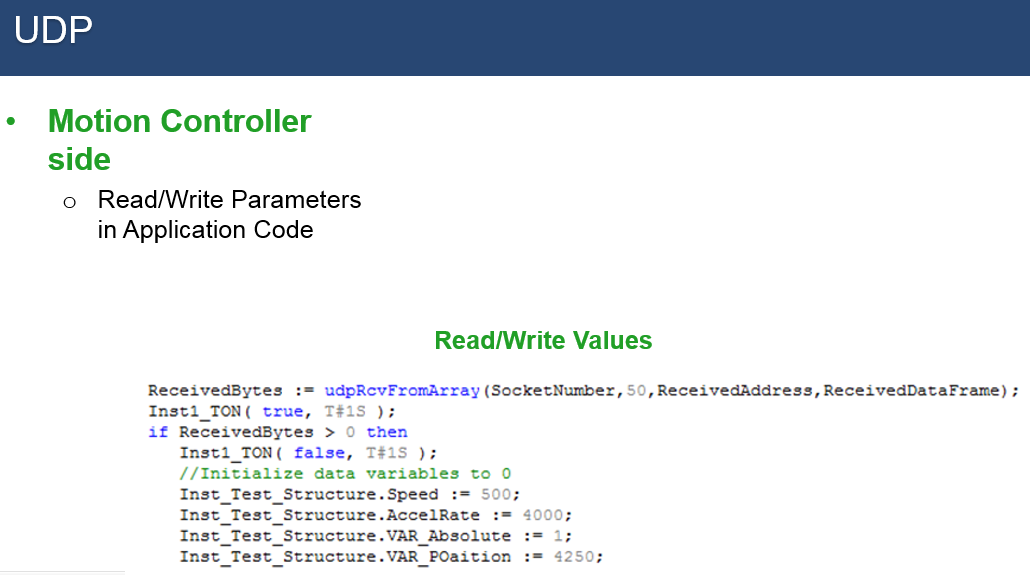


Figure 17

**HTTP**

HTTP (Hypertext Transfer Protocol) has been called the language that “*makes the web work.*” Although not specifically developed for machine automation, HTTP can also be used in the factory automation application world (Figure 18). Languages used in PC-based controllers such as Visual Basic (VB), Visual Studio (VS), Excel, C#, C++, and Java offer HTTP communications support. HTTP is non-deterministic with communication rates of 50 to 300 milliseconds, which works out well in applications that don’t need to pass time-critical information at the servo update rate.

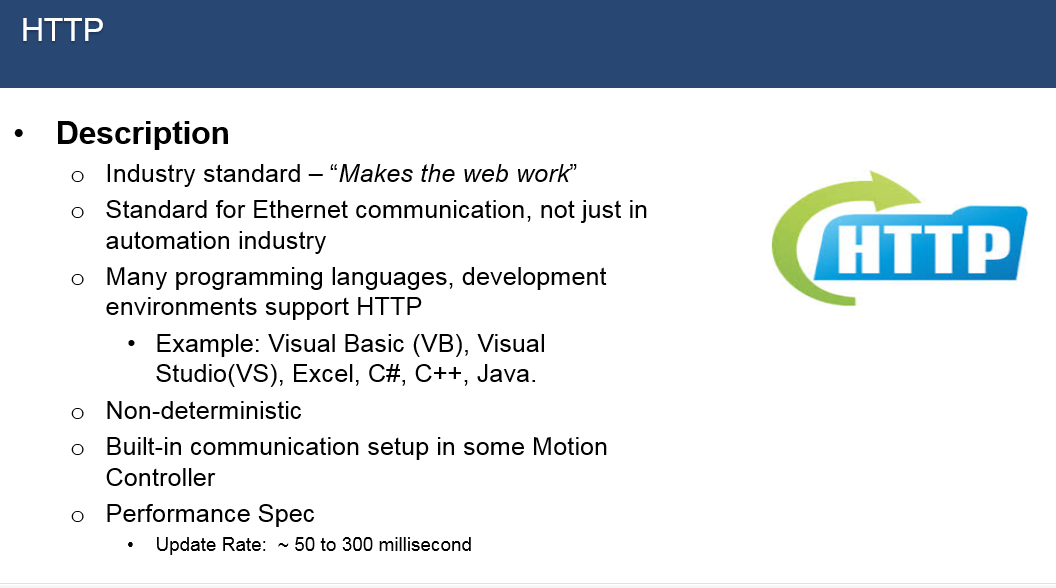
****

Figure 18

In a PxMM motion controller, implementing the HTTP interface is done by selecting an option in the controller setup screen. In figure 19, on the right-hand (highlighted by the red box) there is an option, “*enable PLC variable code access*”. By checking this option all the variables in the dictionary (or tag list) are made available through the HTTP network.

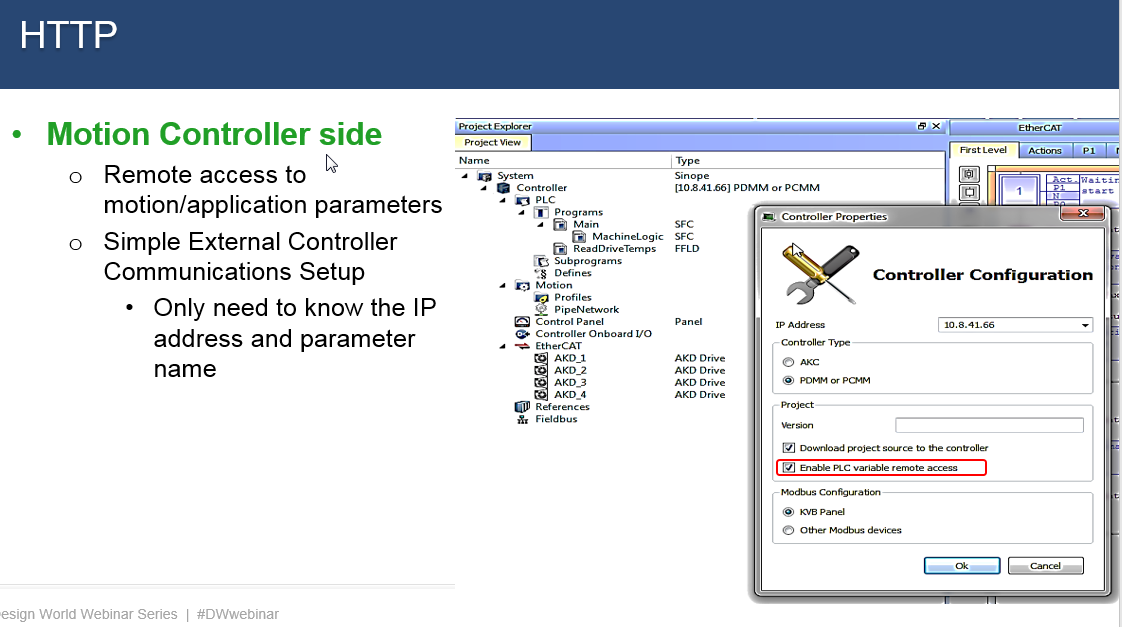


Figure 19

For external controller side, HTTP set up and communication is very simple. To execute a transmission the motion controller’s IP address and parameter name are only needed.

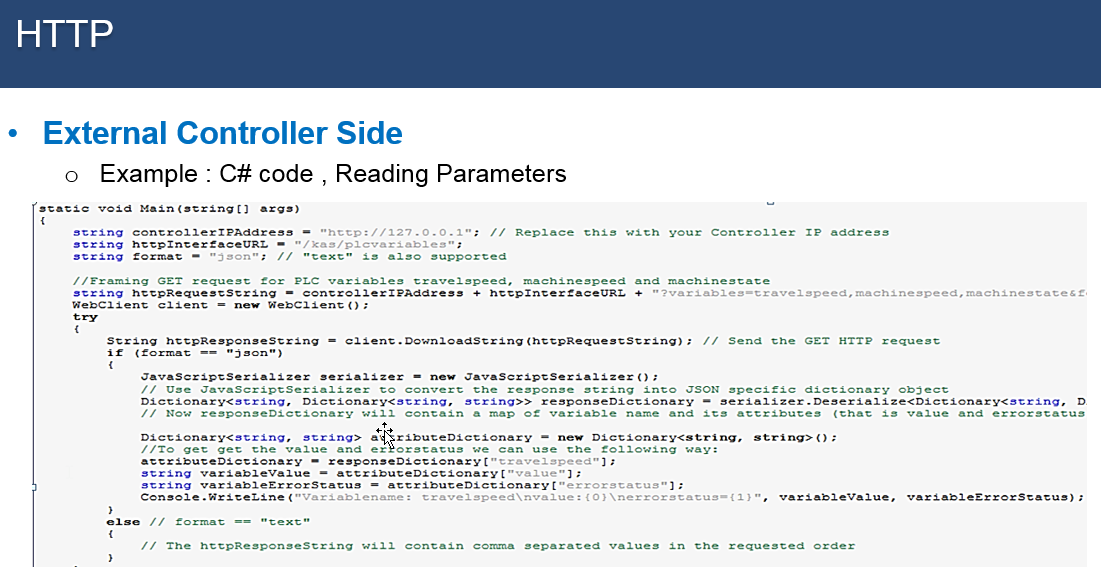
Figure 20 shows an example of C# code to read/write parameters from the external controller side. 

Figure 20

Figure 21 is an example of HTTP communication from a VB2008 platform to the motion controller. The right side of the slide shows a machine control screen/panel to both read and write motion parameters. The data from the motion controller is read using a cyclic read command. Data is transferred or written to the motion controller is event driven when a user clicks on a button or enters a motion parameter value.

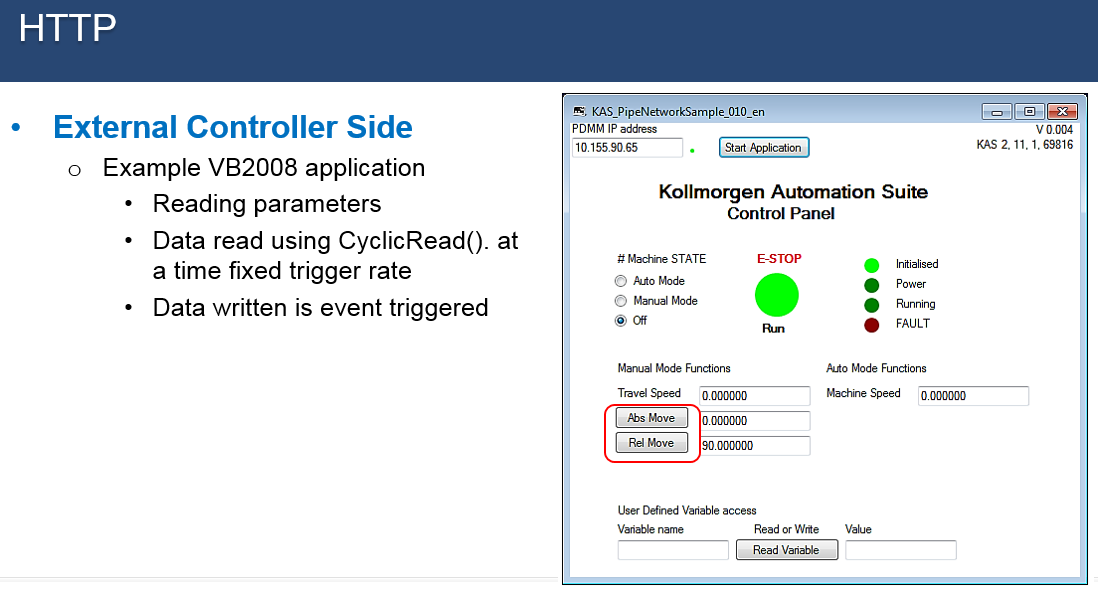


Figure 21

**EtherCAT**

EtherCAT is a motion bus primarily used for connecting devices to the motion controller such as remote IO or drives (Figure 22). A common format in industrial automation applications is COE (Can Over EtherCAT). It can send data deterministically in which the data update time is repeatable. The update time can be down to ¼ millisecond or 250 microseconds. With a performance spec rate down to 250 microseconds, the network can be updated 4,000 times a second.You also have, as part of that communication, a non-deterministic side, called SDO or Mailbox, typically transferring data that doesn’t need to be transferred every update cycle. For instance, typically something that would be sent every update cycle is a position command. An example of something that would not need to be sent every cycle would be a motion profile start command or type of widgets to make, where a 30 or greater milliseconds transfer update rate would not affect performance.

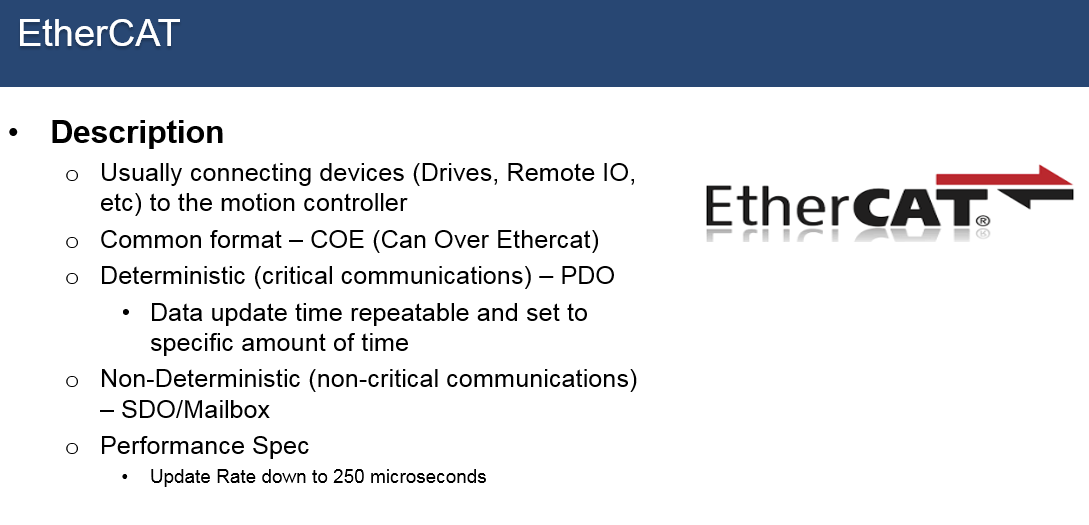
****

Figure 22

On the device side, a predefined ESI (EtherCAT Slave Information) file (Figure 23) from the vendor defines a set of parameters that can be transferred. Figure 15 shows an ESI file that contains parameters that are updated cyclically, called process data objects (PDO). Other parameters that are transferred at a slower rate in the background (SDO or mailbox channel) can also be defined. Additionally, the motion controller, containing the Ethercat master, can set device parameters when the network is initialized that can configure the device for how it will be used in the application

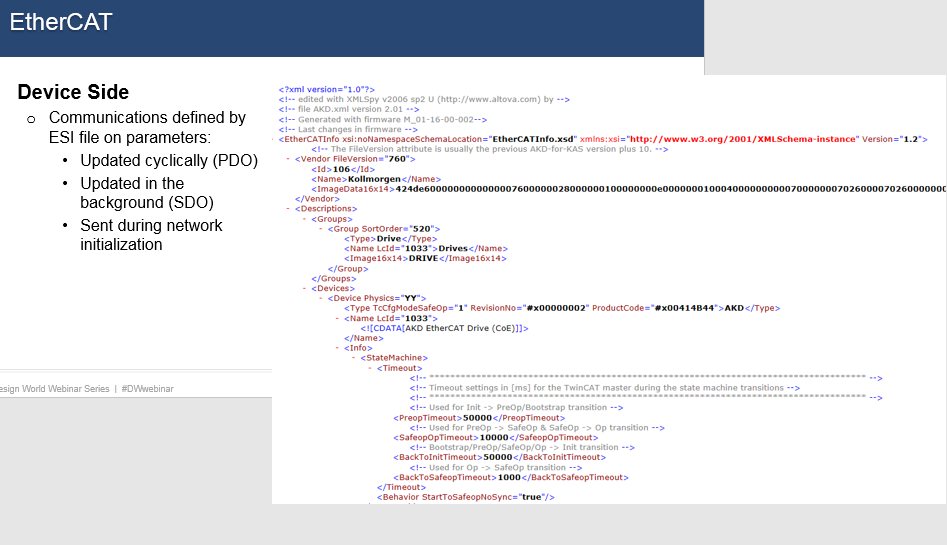
****

Figure 23

For network initialization, some motion controllers have the capability to scan for devices on the EtherCAT network. In figure 24 on the left-hand side under Ethercat, a scan discovered and mapped 5 devices.

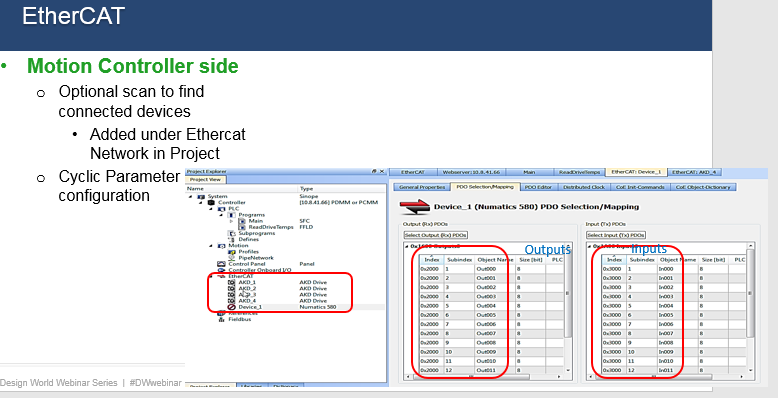
****

Figure 24

In the motion controller (Figure 25) there is a window that will show the PDO parameters and can be sent cyclically. In the setup screen, the user can map the PDO parameters to PLC variables in the application.

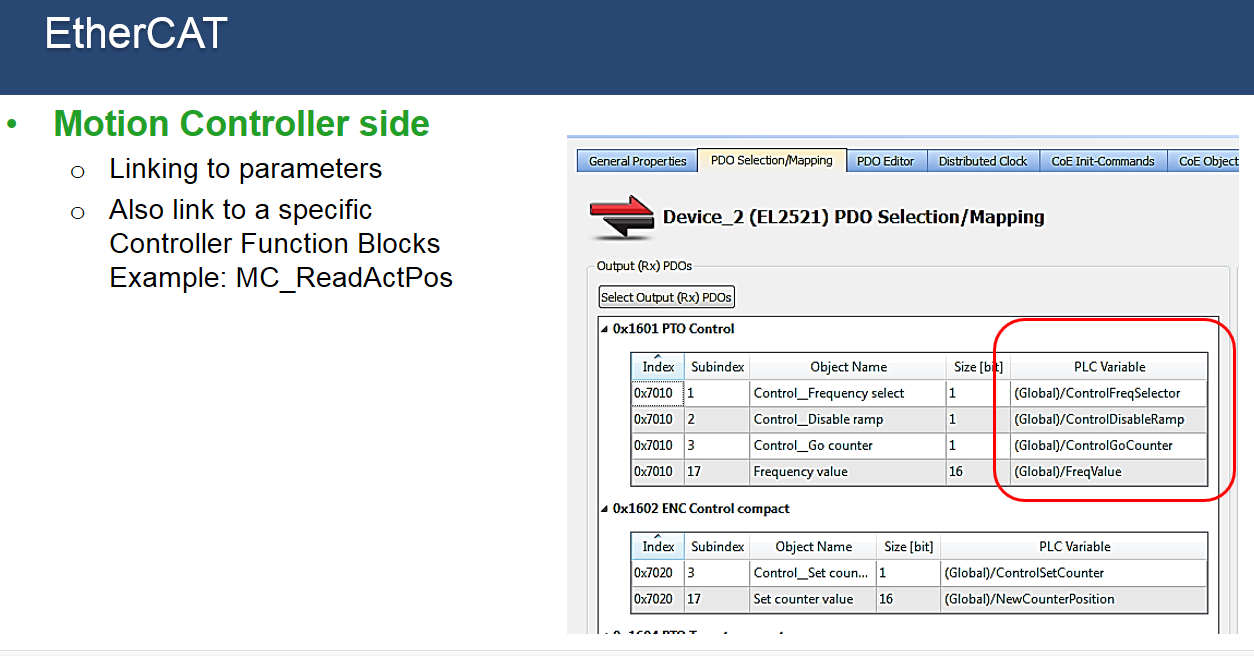
****

Figure 25

Parameters that are coming through the Ethercat network can also be linked to PLC programming variables through specific controller function blocks.

For example, if you want to read the position of a servo axis you can use a standard PLC open function block such as MC\_ReadActPosition. On the motion controller side, there is also a wide range of variables that can be passed non-cyclically. Figure 26 shows an example of changing a servo gain value Position Proportional Gain via a drive write parameter function block.

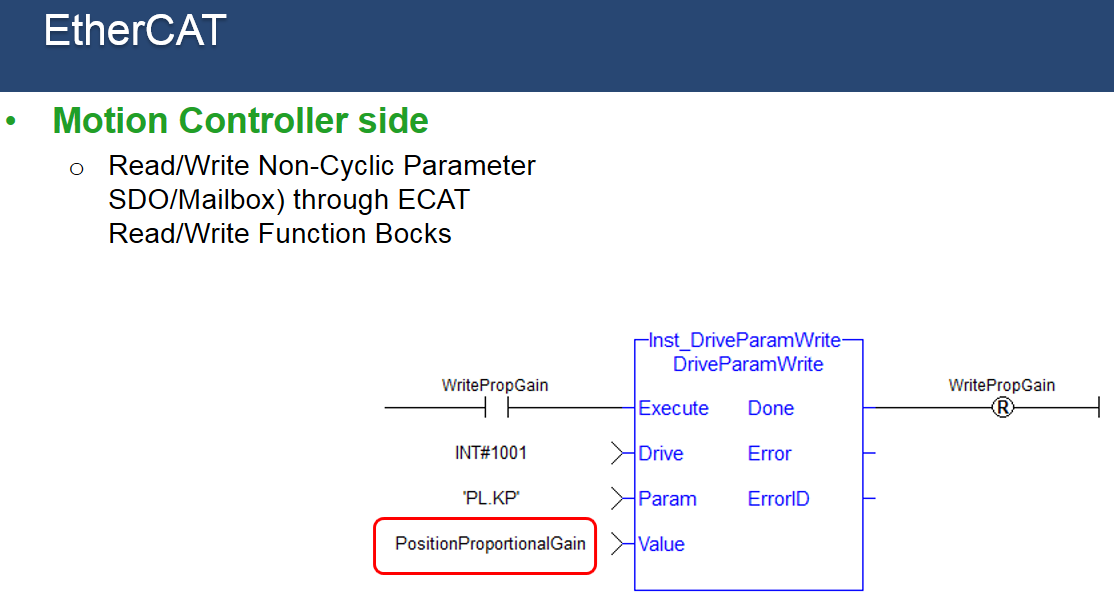
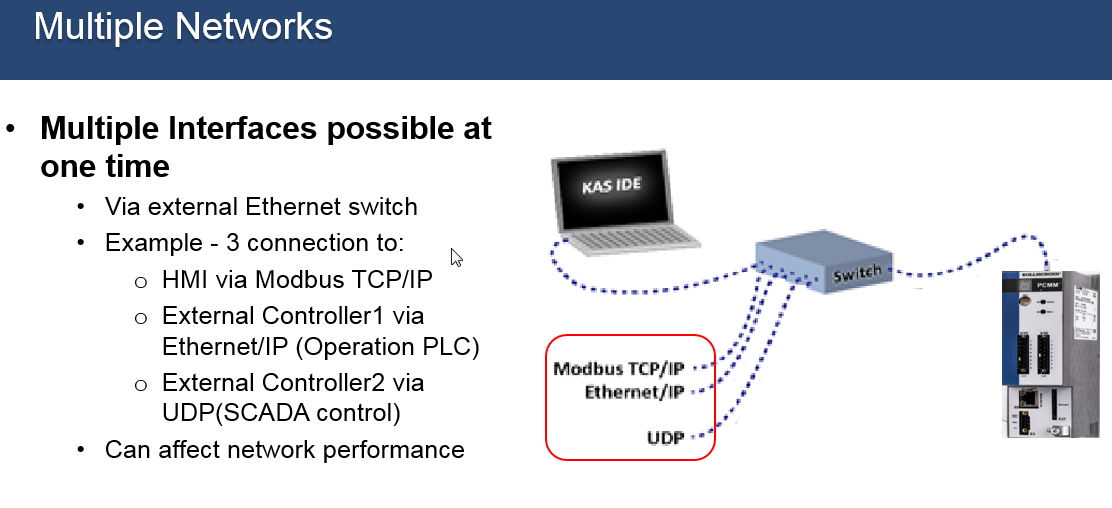
****

Figure 26

Figure 27

**Multiple Interfaces (at the same time)**

What if the application requires multiple Ethernet based network interfaces to connect to the motion controller? If the motion controller’s processor can adapt to it, one way to configure it is through an external switch connected to a single RJ45 port on the motion controller. In the PxMM motion controller example below, there are three connections, Modbus TCP/IP, Ethernet/IP and UDP. Modbus TCP/IP is going to the HMI. External controller #1 is connected via Ethernet IP, and then a third connection uses UDP to connect to external controller #2 for SCADA control. One practical question to ask is: How will using 3 networks affect performance? It can have some influence on performance. The user should carefully plan the networks that are going to be used, optimizing the update rate for each and the information transferred, as well as the update rates of the programs in the motion controller to minimize any negative effects on performance

  
Figure 28

**Web Server**

In addition to Ethernet based communications (Figure 29), there are other ways of getting information in and out of a motion controller. One way is through a web server. To access, a web server built into the motion controller simply requires an IP address be entered in the web browser. Once connected, machine operation information is available as well as a certain amount of remote control, even though that is often limited because the connection can be to a remote location and controller manufacturers are concerned about safety with remote operation.

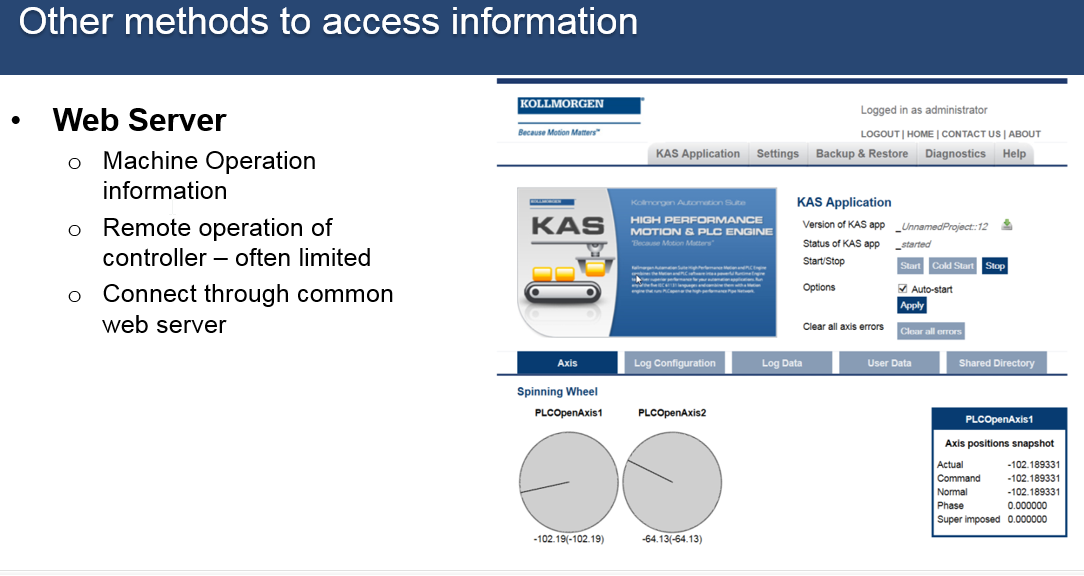
****

Figure 29

Figure 30, controller log information, displays another PxMM controller web server screen showing log files that can be opened and reviewed. For the controls engineer, the log files detail what is going on with the machine operation to monitor performance and help trouble shoot issues. The right side of the figure (Machine performance data) shows how machine performance data files created by the machine programmer can be exported through the web server.

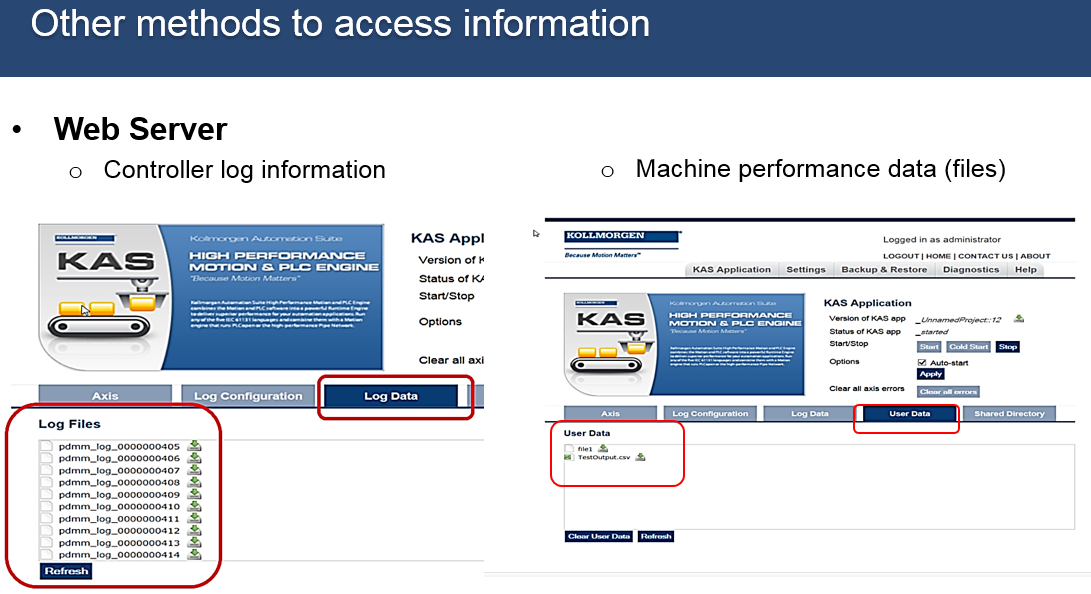


Figure 30

**Memory Card**

Another common way to access information, which is not network based is with a memory card (Figure 31). These allow users to import production recipes and other data into the motion controller, as well as get operational data written from the application program to a file. In addition, an SD card can be used to transfer controller, system, and drive configuration information from one motion controller to another. This provides a time efficient setup method for duplication of the controller’s firmware, application software, and parameters from one machine to the next.

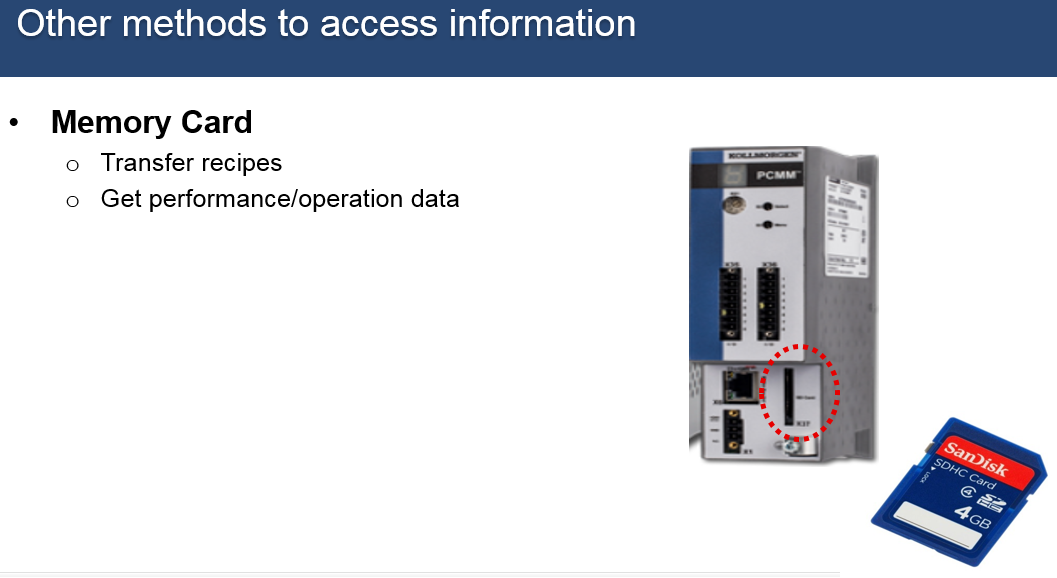


Figure 31

**Remote Memory**

Connecting to an external hard drive through an Ethernet connection is another option to pass information (Figure 32). To make the connection, the motion controller’s web server is used. An external hard drive can be accessed in the same building or elsewhere making it easier for the user to place controller data on a remote central memory location that stores factory-wide operation information. Also in the motion controller’s program, the remote memory can be read when running the machine.

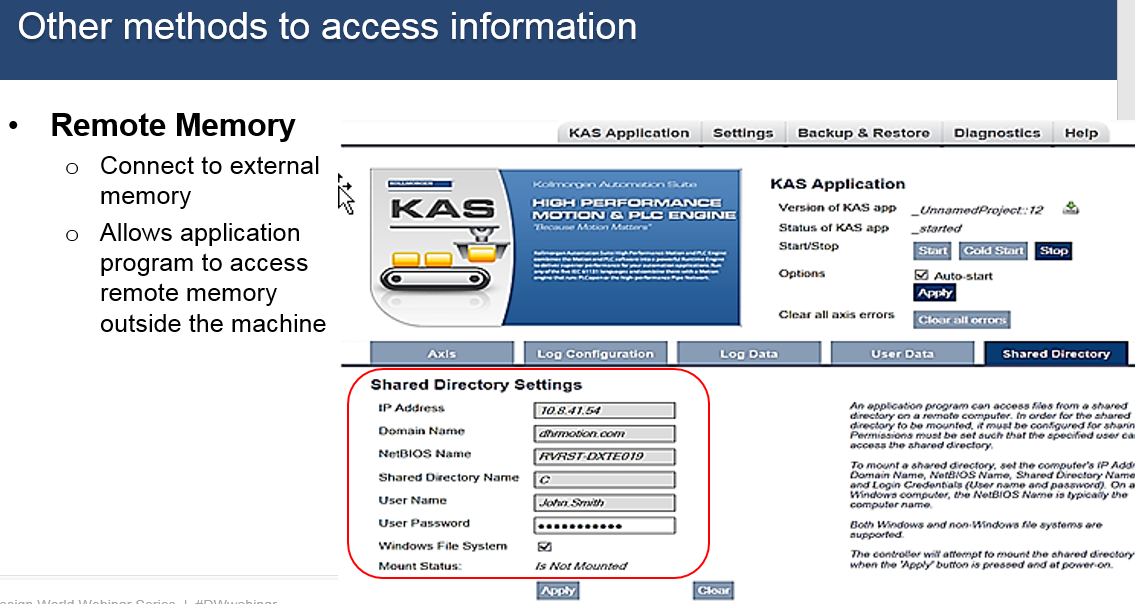


Figure 32

**Which network to choose?**

With all the options available, which one makes the most sense for your application (Figure 33)? Considerations include:

* What is available on the present controllers you are using?
* What is your personal experience with the network? Experience with a particular network, example: Ethernet/IP or HTTP, will help minimize the time to get the network up and working.
* What is your supplier’s experience with a specific network? Do they have application experience that is often critical for a timely integration?
* Can the network update at the rates the machine requires? Look at your application specifics to determine what update times you need to pass data. In many applications, there will be two levels. Example: information that needs to get there in 1 to 10 milliseconds and information that can get there in 50 to 200 milliseconds.
* What is the network’s effect on other areas of motion controller performance? Will the network loading compromise other areas of machine performance?
* Tools – For a particular network, what tools and documentation does the product/vendor have to help establish communications and monitor network performance?
* What technical support (human interaction) does the supplier have in case you need it?
* What Third party tools are available? Example: – With Modbus or HTTP or UDP, there are free tools available online for setting up an interface to communicate with the motion controller.
* Finally, security and safety are becoming more important/popular.
* LINK TO COMPARISON CHART - https://kdn.kollmorgen.com/content/kas-network-overview

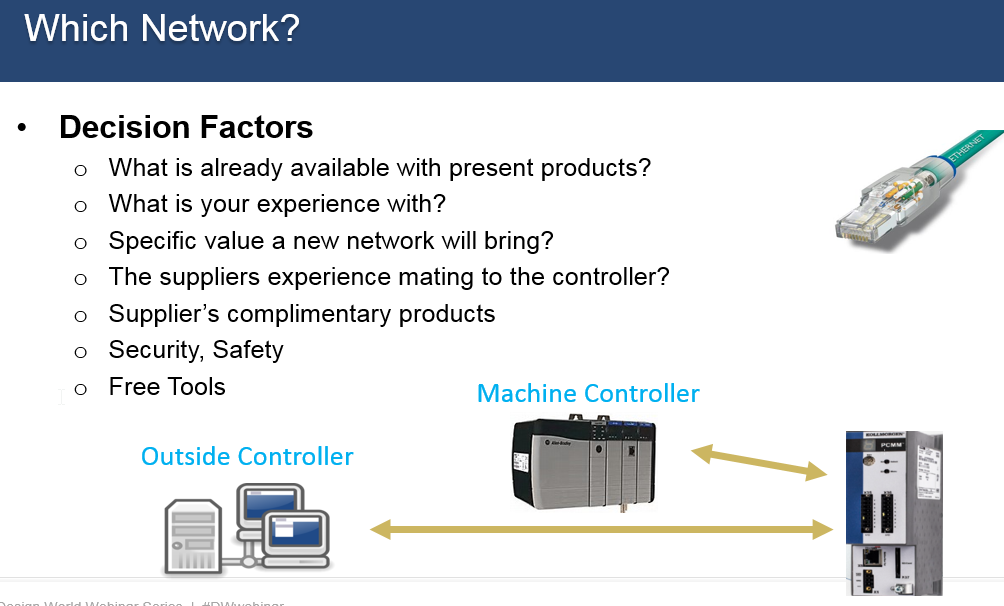


Figure 33

**Commissioning**

. Steps to getting the network to work and achieve the desired performance goals.

* Define the need. What information needs to be passed through the network: Motion, process, IO, status, etc?
* What is the required update rate of each parameter?
* Use an industrial grade Ethernet cable, the extra cost is well worth it to prevent any noise issue and machine down time.
* Setup communication parameters such as update rate, data size, IP address on the controller to establish the connection
* Start small. Get basic communication working first. Pick one or a few parameters to send and receive successfully. It is faster to make adjustments to a few parameters during the development process.
* Add the rest of the parameters
* Verify machine performance on both sides of the network. Is all the information getting there and at the needed update rate? Any effect on motion or other aspects of the control?

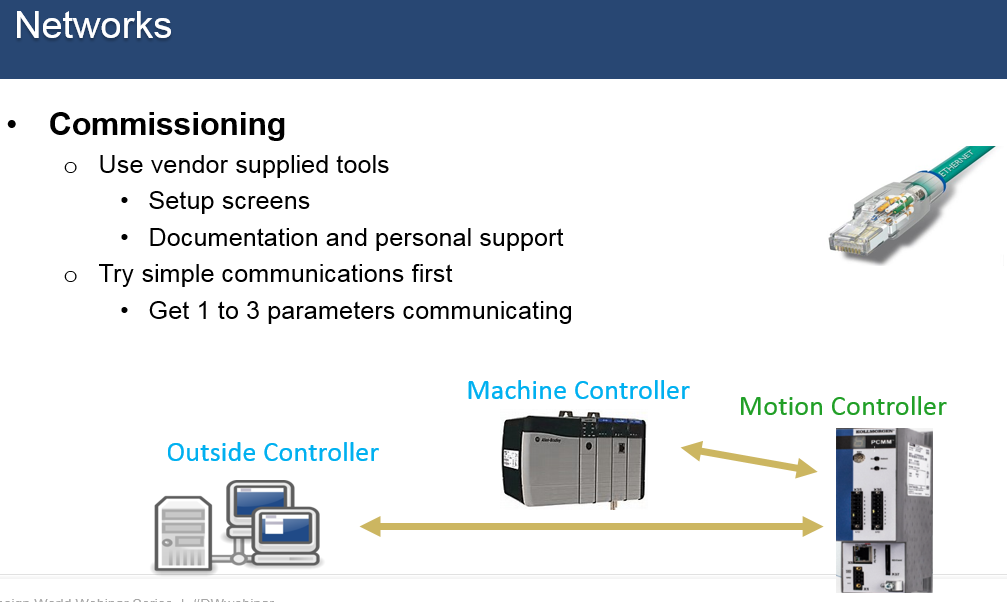


Figure 34

**Final thoughts**

Transferring information in and out of a motion controllers is a function that is critical to the daily operation and overall productivity of today’s factories. This paper has presented many of the set up options that are available, and many of the factors that need to be considered when determining which one works best for you.

Ethernet-based communication within factories has been on the rise over the last 10 years, and as more capabilities are developed to measure and track machine performance it is sure to keep growing. The diversity of applications and information available can provide a competitive edge for those factories making use of the technology. Building the expertise of your staff to include the most current technologies will put you in position to capitalize on new advancements.