



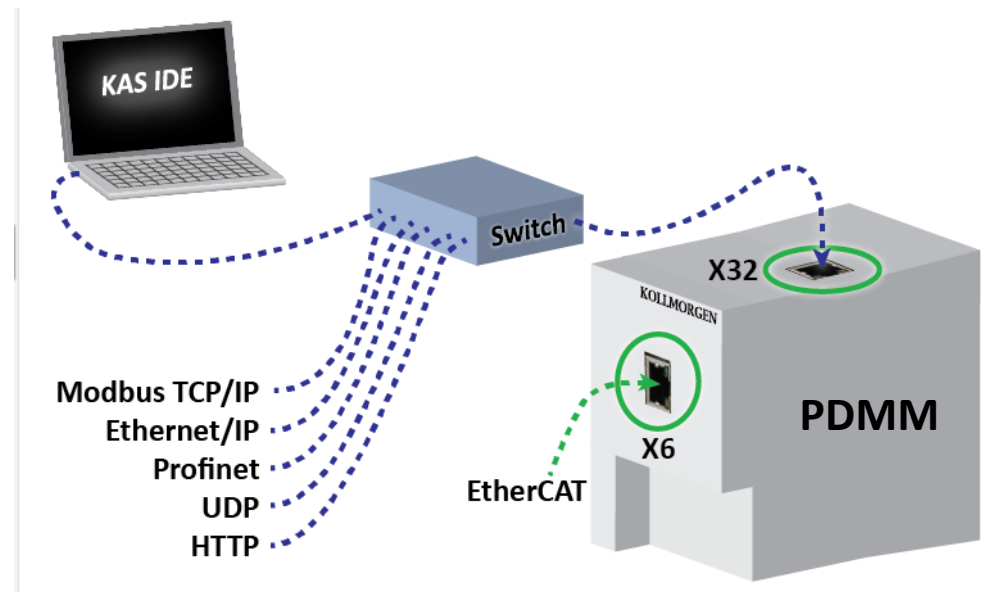
KOLLMORGEN®

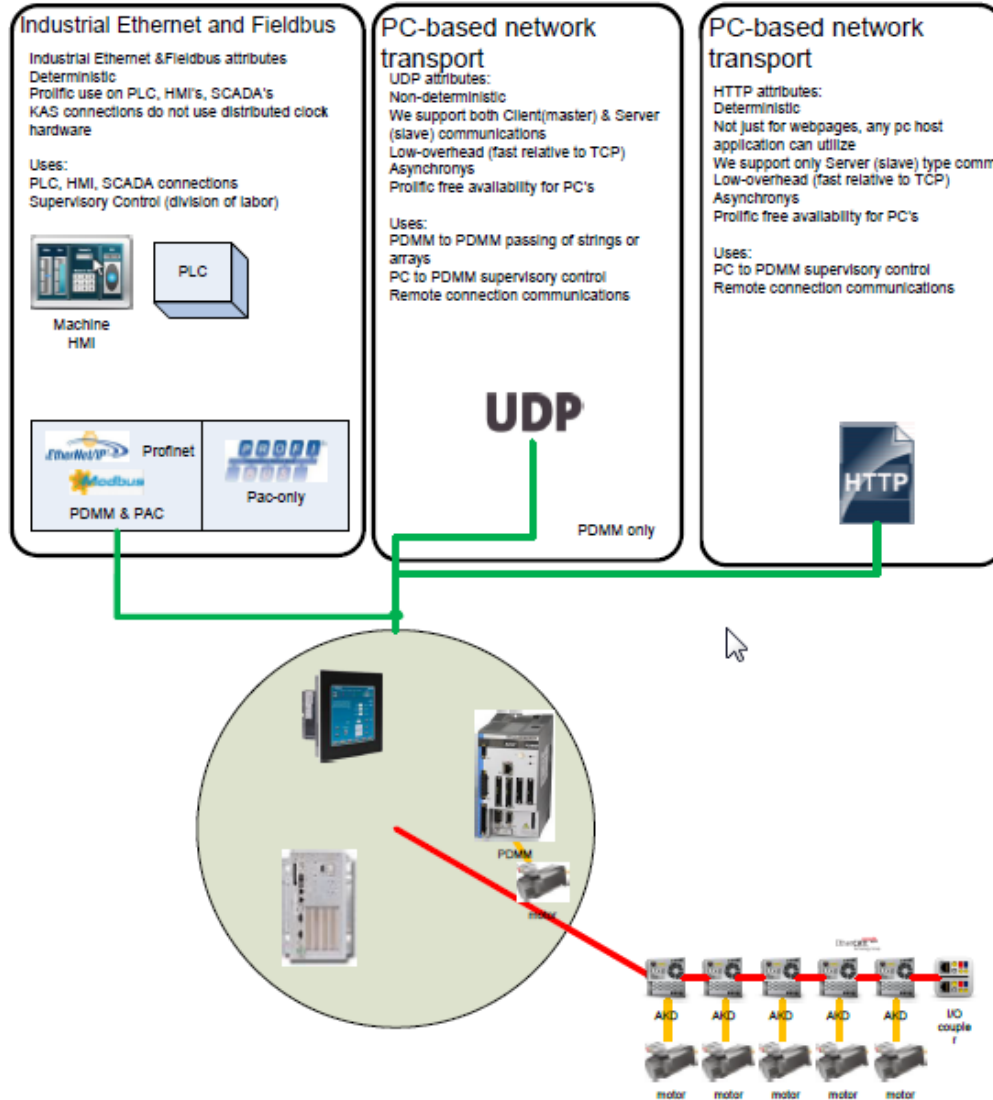
KAS Network Interface Overview

March 30, 2017

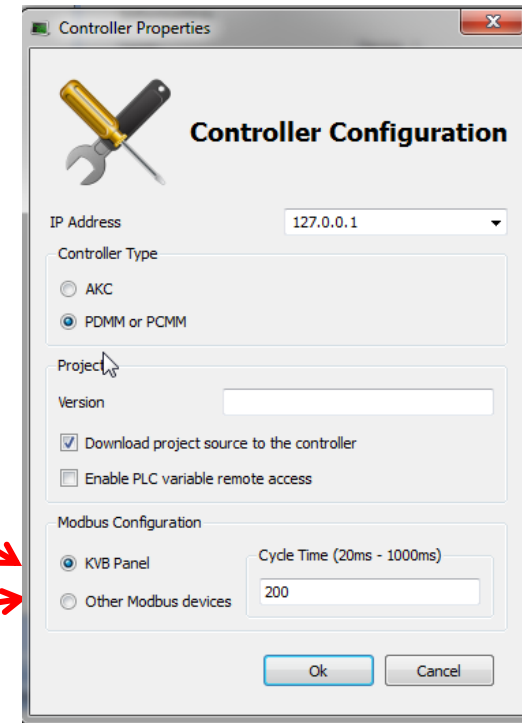


- Multiple Interfaces possible at one time
 - via external switch
- All use Ethercat HW Standard



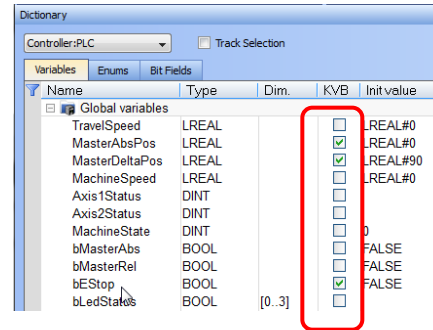


- KAS is a Modbus Slave
- Two options
- KVB Panel
 - with IDE auto assigned addressing
 - Setup to work with Kollmorgen AKI HMIs
 - Address file created when compiling, loaded in KVB when opening
- Other Modbus Devices
 - With user assigned addressing
 - Allows setting up addresses to preconfigured Master HMI addresses
 - Cycle times 20 to 1000 msec

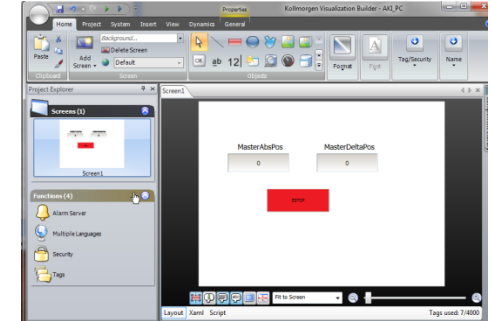


- KVB Panel Option
 - Check KVB box in Project Dictionary

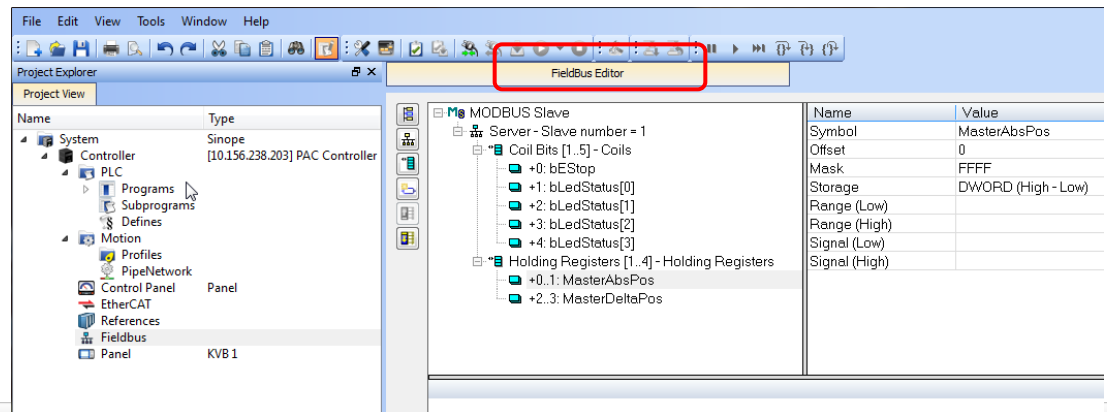
Dictionary



KVB











- Other Modbus Devices Option
 - Use Fieldbus Screen



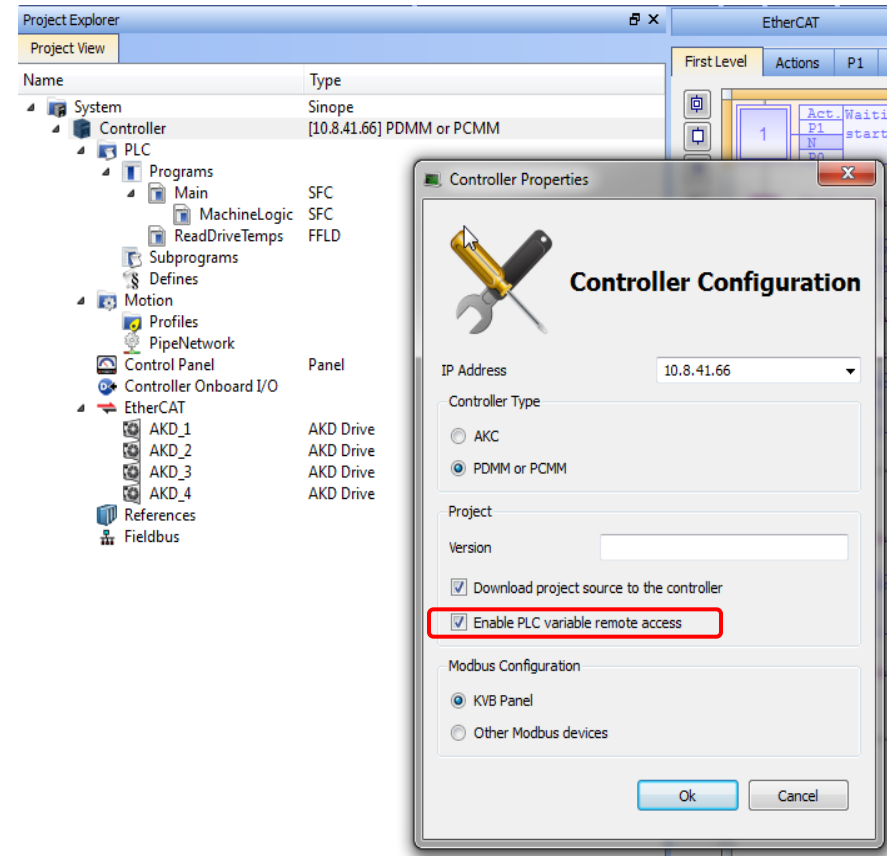
- PDMM can communicate with external devices or other PDMM controllers
- Send/Receive either strings or array of integers
- Standard for Ethernet communication not just in automation industry
- Faster than Modbus communication or HTTP communication, but no error status
- Demo program can receive UDP packets from utility on separate computer
- Support Function Blocks in the IDE Library

Supporting Function Blocks

UDP

 udpAddrMake	Prepare a UDP address
 udpClose	Close a socket
 udpCreate	Create a UDP socket
 udpIsValid	Check if a socket is valid
 udpRcvFrom	Receive a telegram
 udpRcvFromArray	Receive a telegram
 udpSendTo	Send a telegram
 udpSendToArray	Send a telegram

- Remote access of KAS Project variables
- Most modern programming languages, development environments support HTTP
 - C#, C++, Java. Others
 - Visual Basic (VB), Visual Studio(VS)
 - only need to know the IP address of the PxMM and variable name
 - PxMM String based variable names (whether Boolean, Integer, Real, String, etc) requires no addressing like with Modbus
 - All PxMM variable can be read or written to
- Standard for Ethernet communication, not just in automation industry
- Faster than Modbus communication
- Support JSON and Text format (JSON includes error status)
- Optional feature. Can disable for security purposes



- Demo Excel program with Visual Basic objects to demonstrate HTTP communication <http://kdn.kollmorgen.com/content/kas-http-communication>
- In IDE to enable check “Enable PLC Variable Remote Access” box
- Commands can be entered directly in a web browsers’ address bar
 - The URL and syntax for both reading and writing variables is (controller URL)/kas/plcvariables&format=<MIME type>.
 - Examples:

Request	Syntax	Example
GET	<var1>,<var2>	http://198.51.100.0/kas/plcvariables?MachineSpeed,projST.LocalVariable
PUT	<var1>=<val1>,<var2>=<var2>	http://198.51.100.0/kas/plcvariables?MachineSpeed=100.000000,projST.LocalVariable='SampleString'

- Example :To device programmed with C#
 - Sample Code – Reading PLC Variables

```
static void Main(string[] args)
{
    string controllerIPAddress = "http://127.0.0.1"; // Replace this with your Controller IP address
    string httpInterfaceURL = "/kas/plcvariables";
    string format = "json"; // "text" is also supported

    //Framing GET request for PLC variables travelspeed, machinespeed and machinestate
    string httpRequestString = controllerIPAddress + httpInterfaceURL + "?variables=travelspeed,machinespeed,machinestate&format=" + format;
    WebClient client = new WebClient();
    try
    {
        String httpResponseString = client.DownloadString(httpRequestString); // Send the GET HTTP request
        if (format == "json")
        {
            JavaScriptSerializer serializer = new JavaScriptSerializer();
            // Use JavaScriptSerializer to convert the response string into JSON specific dictionary object
            Dictionary<string, Dictionary<string, string>> responseDictionary = serializer.Deserialize<Dictionary<string, Dictionary<string, string>>>(httpResponseString);
            // Now responseDictionary will contain a map of variable name and its attributes (that is value and errorstatus)

            Dictionary<string, string> attributeDictionary = new Dictionary<string, string>();
            //To get get the value and errorstatus we can use the following way:
            attributeDictionary = responseDictionary["travelspeed"];
            string variableValue = attributeDictionary["value"];
            string variableErrorStatus = attributeDictionary["errorstatus"];
            Console.WriteLine("VariableName: travelspeed\nvalue:{0}\nerrorstatus={1}", variableValue, variableErrorStatus);
        }
        else // format == "text"
        {
            // The httpResponseString will contain comma separated values in the requested order
        }
    }
    catch (WebException someWebException) // If server returns some error code
    {
        if (null != ((HttpWebResponse)someWebException.Response))
        {
            Stream reader = ((HttpWebResponse)someWebException.Response).GetResponseStream();
            byte[] message = new byte[reader.Length];
            reader.Read(message, 0, (int)reader.Length);
            string httpErrorCode = someWebException.Message;
            string httpErrorDescription = Encoding.ASCII.GetString(message);
        }
    }
    catch (Exception someException) // If some other exception happens
    {
        string exceptionMessage = someException.Message;
    }
}
```

- Example: To device programmed in Visual Basic Environment
 - <http://kdn.kollmorgen.com/content/kas-interface-using-http-request-programmed-vb2008>

- The following formats are supported:

PROFINET IO controller
 PROFINET IO device

- Use the IDE Fieldbus screen to setup

Variable name	Area	Format	Slot	Subslot	Offset	△	Bit
_0_Slot14_Subslot1_Output0	Output	Signed 16 bit integer	14	1	0		0
_0_Slot15_Subslot1_Output0	Output	Signed 16 bit integer	15	1	0		0
_0_Slot16_Subslot1_Output0	Output	Signed 16 bit integer	16	1	0		0
_0_Slot17_Subslot1_Output0	Output	Signed 16 bit integer	17	1	0		0
_0_Slot18_Subslot1_Output0	Output	Signed 16 bit integer	18	1	0		0
_0_Slot19_Subslot1_Output0	Output	Signed 16 bit integer	19	1	0		0
_0_Slot20_Subslot1_Output0	Output	Signed 16 bit integer	20	1	0		0
_0_Slot2_Subslot1_Output0	Output	Unsigned 8 bit integer	2	1	0		0
_0_Slot3_Subslot1_Input0	Input	Unsigned 8 bit integer	3	1	0		0
_0_Slot4_Subslot1_Input0	Input	Unsigned 8 bit integer	4	1	0		0
_0_Slot5_Subslot1_Input0	Input	Unsigned 8 bit integer	5	1	0		0
_0_Slot6_Subslot1_Input0	Input	Unsigned 8 bit integer	6	1	0		0
_0_Slot7_Subslot1_Input0	Input	Unsigned 8 bit integer	7	1	0		0
_0_Slot8_Subslot1_Input0	Input	Unsigned 8 bit integer	8	1	0		0
_0_Slot8_Subslot1_Output0	Output	Unsigned 8 bit integer	8	1	0		0

_0_Slot1_Subslot1_Input0	Input	Unsigned 16 bit integer	1	1	0		0

_0_Slot14_Subslot1_Output1	Output	Signed 16 bit integer	14	1	2		0
_0_Slot15_Subslot1_Output1	Output	Signed 16 bit integer	15	1	2		0
_0_Slot16_Subslot1_Output1	Output	Signed 16 bit integer	16	1	2		0
_0_Slot17_Subslot1_Output1	Output	Signed 16 bit integer	17	1	2		0

- The following formats are supported:

- Ethernet/IP Adapter (server)
- Ethernet/IP I/O Scanner (client)
- Ethernet/IP Tag Client
- FlexIO / PointIO

- Use the IDE Fieldbus screen to setup

The screenshot shows the IDE Fieldbus configuration interface. On the left, a tree view shows the configuration hierarchy: Ethernet/IP I/O Scanner (client) > Server 127.0.0.1 - Home. On the right, a configuration table displays the following data:

Name	Value
IP Address	127.0.0.1
Config. instance	3
Flags (OEM)	0
Configuration data	
Description	Home

Below the configuration table is a summary table with the following columns: Type, Instance, Size, Connection t..., Priority, 32 bit header, RPI (ms), and Description.

Type	Instance	Size	Connection t...	Priority	32 bit header	RPI (ms)	Description
I/O: Inputs (Ta...	100	2	Point to point	Low	<input type="checkbox"/>	100	Target To Ori...
I/O: Outputs (...)	101	2	Point to point	Low	<input checked="" type="checkbox"/>	100	Originator To ...

- Can be used to communicate PDMM to PDMM

- The PxMM can support multiple fieldbus connections at one time
- Available through port X32
- Typical configuration would be:
 - External Controller to PxMM
 - HMI to PxMM
- Additionally a third device the PC connected to the PxMM can be operational at the same time

- Used to connect Kollmorgen Drives (AKD and S700) plus Kollmorgen ATI IO modules to the PxMM
- Update rates: 2, 1, 0.5, and 0.25 msec
- Deterministic
- COE (Can Over Ethercat) Supported
 - PDO communications
 - Mailbox Communications (including SDO communications)

AKD



S700



AKI Remote I/O



- IDE Quickly configure 3rd party EtherCAT devices
- Add Third Party devices to the KAS Ethercat Network.
Examples: specialty I/O, Safety PLCs, Bridge modules, etc
- High speed, deterministic communication
- Added support for Multiple Device Profile (MDP) products
- Import manufacturer EtherCAT Slave Information (ESI) file and user can scan devices just like Kollmorgen products
- Choose between available PDO mappings, map program variables directly to devices, and can add Init-commands to ensure device parameters are correct when KAS program starts
- Offline setup of devices supported
- Only available through port X6

- To setup use screens within the Ethercat section of the Project Tree
- PDO Selection Mapping – Setup cyclic information
- PDO Editor - Setup cyclic information
- COE Init-Commands – Setup parameters written on Ethercat startup

The screenshot shows the 'PDO Selection/Mapping' configuration screen for 'Device_1 (Numatics 580)'. The interface includes a Project Explorer on the left and a main configuration area with several tabs. The 'PDO Selection/Mapping' tab is active, showing two tables for configuring PDOs.

0x1600 Outputs0

Index	Subindex	Object Name	Size [bit]	PLC
0x2000	1	Out000	8	
0x2000	2	Out001	8	
0x2000	3	Out002	8	
0x2000	4	Out003	8	
0x2000	5	Out004	8	
0x2000	6	Out005	8	
0x2000	7	Out006	8	
0x2000	8	Out007	8	
0x2000	9	Out008	8	
0x2000	10	Out009	8	
0x2000	11	Out010	8	
0x2000	12	Out011	8	

0x1A00 Inputs0

Index	Subindex	Object Name	Size [bit]	PLC
0x3000	1	In000	8	
0x3000	2	In001	8	
0x3000	3	In002	8	
0x3000	4	In003	8	
0x3000	5	In004	8	
0x3000	6	In005	8	
0x3000	7	In006	8	
0x3000	8	In007	8	
0x3000	9	In008	8	
0x3000	10	In009	8	
0x3000	11	In010	8	
0x3000	12	In011	8	

PxMM Networks - Summary Table

Category	Non-Motion Networks					Motion Networks
	Modbus TCP/IP	UDP	HTTP	Profinet	Ethernet/IP	EtherCAT
Transmission Speed	100 Mbts/sec	100 Mbts/sec	100 Mbts/sec	100 Mbts/sec	100 Mbts/sec	100 Mbts/sec
Deterministic?	No	No	No	No	No	Yes
HW layer	Ethernet Technology	Ethernet Technology	Ethernet Technology	Ethernet Technology	Ethernet Technology	Ethernet Technology
Update Rate	down to 200msec	down to 1msec	down to xxxx msec	down to 8 msec	down to 10 msec	0.25 to 2.5 msec
Support in IDE	Dedicated setup screen for mapping dictionary variables to Modbus transmission addresses	Set of FBs to setup communications and send data in non-string or string format:	Setting on Controller Properties in Project view for making all parameters accessible to http link. Can use standard http commands to start, stop a project and reading/writing variable values	Dedicated setup screen for mapping dictionary variables to transmission locations	Dedicated setup screen for mapping dictionary variables to transmission locations	Function blocks for reading/writing parameter to Drives and third party I/O: DriveParamRead, DriveParamWrite, ECATReadSDO, ECATWriteSDO Automatic network setup through scan feature. Screens to setup init and PDO variables. Tools to integrate Third party products
Types supported	Slave Only	Send or Receive	Server Only	Profinet IO RT Controller, Profinet IO RT Device	Scanner, Adapter, Tag Client	COE (Can Over EtherCAT) Master
Documentation Available	Yes	Yes	Yes - includes examples using C# and C++ programming languages	Yes	Yes	Yes
Sample Project available?	Yes - in KDN	Yes	Yes on KDN	No	Yes - in KDN	N/A
Transfer Motion ?	Non- Motion Profile only	Non- Motion Profile only	Non- Motion Profile only	Non- Motion Profile only	Non- Motion Profile only	Yes
Use with PC based embedded Controller (Developed in VB, VS etc and in C#, C++, etc)	Typically Not	Yes	Yes	Typically Not	Typically Not	No - PDMM is master only
PDMM connection point	Connector X32	Connector X32	Connector X32	Connector X32	Connector X32	Connector X6
IDE requires Node Config File	No	No	No	No	No	Yes
Typical Mating products	Common for Third party HMI interfaces	PC based Controller	PC based Controller	Typically used with Siemens PLC	Typically used with Rockwell PLC	Kol Drives, Kol I/O, Third party products