Agenda

Schneider/Modbus-IDA /ODVA Relationship

- Value of ODVA to Schneider
- Value of Schneider to ODVA

Current CIP Modbus Status
Purpose and major requirements
Technical solution requirements & goals
Technical solution summary
Technical solution details

- Modbus Features
- CIP Functions supported with Modbus
- Explicit message
- Implicit I/O
- CIP object representing Modbus/TCP device
- New Modbus TCP CIP object
- EDS
- Impact on CIP

Brief Modbus/TCP tutorial
The Modbus protocol and the EtherNet/IP protocol make up the majority of the installed device level products to date. Both are widely accepted standards with outstanding membership and participation. There is a large overlap of membership between Modbus-IDA and the ODVA. Seamless integration between the protocols is now available. A larger installed base is available to both Modbus-IDA and ODVA members through CIP connectivity to other protocols. Suppliers benefit from this cooperation through the ability to reach new markets.
Schneider Electric/Modbus-IDA/ODVA Cooperation

Customers want a cost-effective, coherent networking solution from plant floor to boardroom.

Customers want an industrial Ethernet solution that provides interoperability of multi-vendor systems and protects their investment.

Schneider Electric has formed a new cooperation with ODVA to combine the two most successful Industrial Ethernet solutions into a defacto standard.

This alliance delivers on the promise of a broad, standard and unmodified Ethernet-based fieldbus for the plant floor and access to that information for the rest of the enterprise.

Multi-vendor, standard solution to compete with closed solutions, like Profinet.
Cooperation to Accelerate Transparent Ready

Schneider Electric wants to accelerate customer adoption of Transparent Ready.

- Reduce the risk of making a decision to use today’s Industrial Ethernet solutions.
- Protect investment for existing users

To achieve this, Schneider Electric has decided to pursue a partnership agreement

Key partner criteria:
- Shares philosophy around “standard” Ethernet
- Collaborative organization – not single vendor driven
- Appreciation for Schneider Electric’s commitment to installed base and forward migration
Why a Network Cooperation with ODVA

Cooperation combines Modbus and EtherNet/IP
  ▶ Brings together the market power of the two most successful protocols.
  ▶ Both protocols are based on standard Ethernet technology

ODVA will take advantage of Modbus
  ▶ Leverages Modbus installed base & market share

Schneider Electric can easily take advantage of EtherNet/IP services
  ▶ No custom hardware required
  ▶ Existing products can be migrated
  ▶ Unique values of EtherNet/IP complement simplicity of Modbus
Nature of Cooperation

ODVA announces that Schneider Electric will be invited to become an ODVA Principal Member, joining Rockwell, Eaton, Omron, and CISCO, and provide individuals to serve on the ODVA Board of Directors and the Technical Review Board along with active participants in other key ODVA activities.

Schneider Electric has expanded its Transparent Ready architecture by supporting both Modbus TCP and EtherNet/IP protocol and services.

ODVA’s EtherNet/IP technology is being enhanced to support the Modbus TCP protocol and the Modbus Serial protocol.

ODVA’s technical work will protect the investment of existing Modbus TCP, Modbus Serial and EtherNet/IP device vendors and customers.
Technical Solution Summary

CIP standard has been enhanced to provide seamless access to Modbus devices

Solution Requirements and Commitments:
- The solution fits within the ODVA framework
- No change for Modbus target devices
- No change for EtherNet/IP target devices

Schneider Electric will upgrade its PLC modules to incorporate EtherNet/IP protocol while ensuring compatibility with existing Modbus devices and systems
Agenda

Schneider / Modbus-IDA / ODVA Relationship

- Value of ODVA to Schneider
- Value of Schneider to ODVA

Current CIP Modbus Status

Purpose and major requirements

Technical solution requirements & goals

Technical solution summary

Technical solution details

- Modbus Features
- CIP Functions supported with Modbus
- Explicit message
- Implicit I/O
- CIP object representing Modbus/TCP device
- new Modbus/TCP CIP object
- EDS
- Impact on CIP

Brief Modbus TCP tutorial
Current Modbus Integration Status

Formed the Modbus Integration jSIG within the ODVA
► 20 different companies are members
► Have held over 40 meetings since inception
► Ongoing topics to enhance and improve the new Volume 7

Volume 7: Integration of Modbus Devices into CIP
► Version 1.0 has been published by the ODVA.
  • Covers Modbus TCP and hooks for Modbus Serial
► An updated version is expected in the next publication cycle from the ODVA.
  • The next version will include Modbus Serial requirements.

ODVA Conformance Testing
► Updating ODVA conformance tests with Modbus translator capabilities.
Agenda

Schneider/Modbus-IDA/ ODVA Relationship
- Value of ODVA to Schneider
- Value of Schneider to ODVA

Current CIP Modbus Status
Purpose and major requirements
Technical solution requirements & goals
Technical solution summary
Technical solution details
- Modbus Features
- CIP Functions supported with Modbus
- Explicit message
- Implicit I/O
- CIP object representing Modbus/TCP device
- New Modbus/TCP CIP object
- EDS
- Impact on CIP

Brief Modbus/TCP tutorial
Purpose & Requirement

Extend the CIP standard to provide seamless access to Modbus devices

Requirements

- The solution fits within the ODVA framework
  - The solution is consistent with the existing CIP model
  - The solution has significant technical merit, as viewed from outside ODVA
- No change for the Modbus target devices
- No change for the EtherNet/IP target devices

Constraints

- Minimize impact to existing EtherNet/IP originator
Agenda

Schneider/Modbus-IDA/ODVA Relationship
- Value of ODVA to Schneider
- Value of Schneider to ODVA

Current CIP Modbus Status
Purpose and major requirements

Technical solution requirements & goals

Technical solution summary

Technical solution details
- Modbus Features
- CIP Functions supported with Modbus
- Explicit message
- Implicit I/O
- CIP object representing Modbus/TCP device
- New Modbus/TCP CIP object
- EDS
- Impact on CIP

Brief Modbus TCP tutorial
Technical Solution Requirements 1/2

Technical requirement

- The solution allows Modbus devices to be integrated and accessed from CIP originators and software tools

- The solution allows CIP originators basic read/write access to data in Modbus devices

- The solution allows the CIP originator to bring Modbus data into the CIP application in a way that is consistent with the CIP communications model

- Support all existing devices that can function as a Modbus server device (including simple devices and complex such as a programmable controller)
Technical Solution Requirements 2/2

Technical requirement

Integration of Modbus devices is consistent with the existing CIP application and communication models

- Minimize impact to existing CIP originator devices
- No impact to existing CIP target devices
- No Impact to existing Modbus server devices

Allow existing vendor specific CIP – Modbus gateway products to work without change.

- Desirable that the new mechanism be feasible for implementation in existing products.
Agenda

Schneider / Modbus-IDA / ODVA Relationship
- Value of ODVA to Schneider
- Value of Schneider to ODVA

Current CIP Modbus Status
Purpose and major requirements
Technical solution requirements & goals

Technical solution summary

Technical solution details
- Modbus Features
- CIP Functions supported with Modbus
- Explicit message
- Implicit I/O
- CIP object representing Modbus/TCP device
- new Modbus/TCP CIP object
- EDS
- Impact on CIP

Brief Modbus/TCP tutorial
The Modbus Integration mechanism:

- The Modbus translator translates CIP objects and services to Modbus messages. The translator can be implemented in the CIP originator or a CIP router.
- Defined CIP objects and services that provide read and write access to Modbus data. Allows Modbus function code execution via the Modbus Object.
- Explicit and implicit messaging from the CIP originator to the Modbus target device. Conversion of CIP messaging to Modbus requests and Modbus responses to CIP messages.
- EDS files that describe the Modbus capabilities in CIP terms. A generic Modbus file or optionally a device specific file.
Summary of Technical Solution 2/5

The Target Audience:

**Developers of CIP originators** who wish to implement the Modbus Translator in their device, or wish to allow their devices to communicate via a CIP-to-Modbus router.

**Developers of CIP router devices** who wish to implement the Modbus Translator in a CIP-to-Modbus router (e.g., an EtherNet/IP to Modbus Serial router).

**Modbus device vendors** who wish to understand how their device can be integrated into the CIP to Modbus integration solution (e.g., to be able to create EDS files for Modbus devices).
Summary of Technical Solution 3/5

Device architecture view: CIP originator to Modbus TCP

App can send CIP messages to native EIP devices

App can send CIP messages to Modbus devices via the CIP-Modbus translation

- Application functions
- Object Library
- Explicit Messages, I/O Messages
  Connection Management, Routing
- EtherNet/IP
  CIP-Modbus translation
- Modbus/TCP
- TCP/IP Stack
  Ethernet data link & physical layers
Modbus TCP translation

- End device Identify through CIP path
- Translation layer translating Modbus requests in CIP object
- New CIP Modbus Objects

Summary of Technical Solution 4/5
New CIP communication mechanisms allow existing, unmodified Modbus devices to be seamlessly accessed from a CIP-originator devices.
Agenda

Schneider/Modbus-IDA/ODVA Relationship
- Value of ODVA to Schneider
- Value of Schneider to ODVA

Current CIP Modbus Status
Purpose and major requirements
Technical solution requirements & goals
Technical solution summary

Technical solution details
- **Modbus Features**
- CIP Functions supported with Modbus
- Explicit message
- Implicit I/O
- CIP object representing Modbus TCP device
- New Modbus TCP CIP object
- EDS
- Impact on CIP

Brief Modbus/TCP tutorial
There are no required Modbus function codes for use with CIP

- Modbus devices do not share common function codes.
- Required to work with any Modbus device

Recommended Modbus function codes are:

- FC 03 Read Holding Registers
- FC 16 Write Multiple Registers
- FC 23 Read/Write Multiple Registers
- FC 43/14 Read Device Identification

Use would allow a Modbus device to more easily integrate with CIP.

- Vendors designing new Modbus products or updating existing firmware are strongly encouraged to support at least this recommended function list
Modbus Features Used with CIP 2/4

CIP does not specify a configuration method. For Modbus TCP configuration BootP, DHCP, integrated HMI or an external port could be used.

- DHCP is recommended since this is the method recommended by Ethernet/IP

For device configuration use of contiguous registers is recommended as then the device can receive configuration upon the opening of a CIP I/O connection.

- Vendors are encouraged not to mix read-only and read-write registers or to create “holes” of unsupported registers within configuration areas.

Use of an EDS file is recommended to help CIP-based configuration tools.
Modbus Features Used with CIP 3/4

CIP is not aware of Modbus timeouts.
▶ Mismatched timeout lengths between CIP and Modbus need to be considered.

The Modbus Translator will handle byte swapping to the Modbus device.
▶ CIP is Little Endian and Modbus is Big Endian, the translator takes care of this for the user.

For I/O Connections the connection size indicates how many registers will be read or written on the device.
▶ This is indicated by the Instance ID.

For best use in CIP, detailed data map of the Modbus tables on the Modbus device from the vendor is important.
Care is needed to be sure CIP refresh rate is not faster than the rate at the target Modbus device can consume the corresponding writes.

- Mismatched timeout lengths between CIP and Modbus need to be considered.
- It is possible to have multiple writes from the CIP side through the Modbus translator to the target Modbus device.
- There is no correlation between the CIP sequence number and the Modbus transaction identifier.

Modbus devices require an independent read function in order to work with CIP I/O connections.

- Devices that otherwise would use Function Code 23 (FC23) only are recommended to also implement the read Function Code 03. Schneider does not have devices using only FC23.
Agenda

Schneider/Modbus-IDA/ODVA Relationship
- Value of ODVA to Schneider
- Value of Schneider to ODVA

Current CIP Modbus Status
Purpose and major requirements
Technical solution requirements & goals
Technical solution summary

Technical solution details
- Modbus Features
- CIP Functions supported with Modbus
- Explicit message
- Implicit I/O
- CIP object representing Modbus TCP device
- New Modbus TCP CIP object
- EDS
- Impact on CIP

Brief Modbus TCP tutorial
CIP Functions Supported with Modbus 1/2

Unconnected Explicit Messaging

Connected Class 3 Explicit Messaging

Connected Class 1 Implicit Messaging
  ▶ With Cyclic triggering

Connected Implicit
  ▶ works with transport class 3

32-bit Run/Idle headers for O->T
  ▶ Modeless in the T->O direction

Unicast in the T->O and O->T directions
CIP Functions Supported with Modbus 2/2

Multicast for T->O (with RPI matching)

Data Segment
- For Modbus devices with their configuration data in a contiguous item block

Implicit ExclusiveOwner and InputOnly
- There is no notion of redundant ownership

Parameter Objects

Assembly Objects

Identity Object

Modbus Object
Agenda

Schneider/Modbus-IDA/ODVA Relationship

► Value of ODVA to Schneider
► Value of Schneider to ODVA

Current CIP Modbus Status

Purpose and major requirements

Technical solution requirements & goals

Technical solution summary

Technical solution details

► Modbus Features
► CIP Functions supported with Modbus
► **Explicit message**
► Implicit I/O
► CIP object representing Modbus TCP device
► New Modbus TCP CIP object
► EDS
► Impact on CIP

Brief Modbus TCP tutorial
Explicit Message : What the CIP Originator Sees

Explicit messaging

- Explicit messages support connected and unconnected mode.

Modbus TCP is identified CIP connection via the path

- Both Modbus TCP and Modbus Serial have specific CIP Port Types. The port number signifies connection to a Modbus target.

Each Modbus device appears as a CIP device with CIP application objects and native CIP communications.

- List of CIP objects (parameter object or Modbus object) and services exposed for the Modbus TCP device are referred in CIP Object translation.
- The Modbus Translator provides the CIP capability.
Explicit Connected Message to Modbus TCP

CIP Originator

FwdOpen request

FwdOpen reply

Explicit msg request, class=Param or MB object

CIP reply

FwdClose Request

Translator

Open TCP connection

TCP connection opened

Modbus request

Modbus reply

Modbus reply

Close TCP connection

TCP connection closed

Modbus TCP Target
Explicit Connected Message to Modbus TCP with No Modbus Reply

CIP Originator

- FwdOpen request
- Explicit msg request, class=Param or MB object
- Start inactivity timer (e.g. 4*RPI)
- CIP retry at each RPI
- CIP retry at each RPI
- CIP retry at each RPI
- Inactivity timer expires
- No CIP reply

Translator

- FwdOpen reply
- Open TCP connection
- TCP retransmission algorithm started
- Reset and restart
- Modbus request
- Inactivity timer (e.g. 4*RPI) at each reception of a CIP message.
- Modbus request
- Modbus request
- Modbus request
- Inactivity timer expires
- Delete CIP connection
- TCP Re Transmission finished; TCP connection is reset

Modbus TCP Target

- TCP connection opened
- Modbus request
- Modbus request
- Modbus request
- Modbus request
- Inactivity timer expires

Hannover Messe 2008
April 21-25, 2008
page 33
Explicit Unconnected Message to Modbus TCP

CIP Originator

UnconnectedSend

Translator

Open TCP connection

TCP connection opened

Modbus request

Modbus reply

UnconnectedSend reply

Modbus TCP Target

Close TCP connection

TCP connection closed

TCP connection closed
Explicit Unconnected Message to Modbus TCP with No Modbus Reply

CIP Originator

UnconnectedSend

Translator

Open TCP connection

TCP connection opened

Modbus request

Start Modbus Response Timer = CIP Request Timeout (provided by CIP Request) and TCP retransmission

Modbus TCP Target

No Modbus reply

Request Time-out Timer expires

TCP Retransmission finished: TCP connection is reset

UnconnectedSend reply indicating a Routing Error
Explicit Message: CIP Object to Modbus TCP

The Parameter Object instances are used for explicit messaging

- Get/Set single Attribute service for reading/writing one data item

New Modbus TCP application object allows access to multiple data items and additional Modbus TCP function codes

- Services for read and write of multiple data items (Holding Registers, Coils, etc.)
- Service to allow explicit “pass through” of any Modbus TCP function code
- Appropriate EDS constructs if needed to support the new object (in the translation device and/or the Modbus TCP device EDS)
Explicit Connection Example

to read 4 contiguous input registers starting at index 11

- calling the service “Get_Member”,
- for Class ID = 0x04 (Assembly Class ID),
- Instance ID = 0x0001000B
- with the parameter Member ID = 1,
- Using the Member Services “Multiple Sequential Member” protocol, specify 4 members
Explicit Connection Example: Alternatives

to read 4 contiguous input registers starting at index 11, Using the Modbus Object.

- calling the service “Read Input Registers” (0x4D),
- for Class ID = 0x?? (Modbus Class ID),
- Instance ID = 1,
- with the Starting Address parameter = 11 and the Quantity of Input Registers parameter = 4,

Can also be accessed one at a time with the Parameter object. Using the GetAttributeSingle service.
Agenda

Schneider/Modbus-IDA/ODVA Relationship
- Value of ODVA to Schneider
- Value of Schneider to ODVA

Current CIP Modbus Status
Purpose and major requirements
Technical solution requirements & goals
Technical solution summary

Technical solution details
- Modbus Features
- CIP Functions supported with Modbus
- Explicit message
- Implicit I/O
- CIP object representing Modbus TCP device
- New Modbus TCP CIP object
- EDS
- Impact on CIP

Brief Modbus/TCP tutorial
Implicit I/O : Modbus TCP Translation 1/4

Assembly Objects point to 250 byte blocks within the four Modbus data items tables

Parameter Objects point to individual data item in the tables

Each Modbus data table is assigned a block of Assembly and Parameter Instances

- Holding Registers (table of 0x10000 data)
  => accessible through Assembly Instance #0x0001 to instance #0x10000
- Input Registers (table of 0x10000 data)
  => accessible through Assembly instance #0x10001 to instance #0x20000
- Coils (table of 0x10000 data)
  => accessible through Assembly instance #0x20001 to instance #0x30000
- Discrete Input (table of 0x10000 data)
  => accessible through Assembly instance #0x30001 to instance #0x40000

Each Modbus TCP data items offset in a table is represented by an instance number (1 based offset)
The number of data items to be read/written is determined by the produced/consumed connection size within the ForwardOpen.

Example: to write 4 Holding Registers, starting at register number 20 (table offset 19):
- O->T Connection path of Class Assembly, Instance 20
- O->T Connection size of 8

I/O connection to read 8 Input Registers, starting at register number 4 (table offset 3):
- T->O Connection path of Class Assembly, Instance 0x10004
- T->O Connection size of 16 (connection size is specified in bytes)
Implicit I/O: Modbus TCP Translation 3/4

CIP Originator

FwdOpen request : assembly , instance xx, connection size

FwdOpen reply

Output packet (originator-to-target)


This sequence is repeated indefinitely.

Input packet (target-to-originator)

Fwd Close request

Fwd Close reply

Translator

Open TCP connection

TCP connection opened

Modbus write request (FC 16)

Modbus reply

Modbus read request (FC 03)

Modbus reply

Close TCP connection

TCP connection closed

Modbus/TCP Target

Hannover Messe 2008
April 21-25, 2008
page 42
Implicit I/O: Modbus TCP Translation with No Modbus Reply  4/4

CIP Originator

- input packet (target-to-originator)
- Start inactivity timer (e.g. 4*RPI)
- Output packet (originator-to-target)
- Output packet (originator-to-target)
- Inactivity timer expires
- Delete CIP connection (T->O and O->T)

Translator

- Modbus read request (FC 03)
- Modbus write request (FC 16)
- TCP retransmission algorithm started
- Reset Inactivity timer (e.g. 4*RPI) at each reception of a CIP message.
- Inactivity timer expires
- Delete CIP connection (T->O and O->T)
- TCP Retransmission finished: TCP connection is reset

Modbus/TCP Target

- Modbus reply
- No Modbus reply

Hannover Messe 2008
April 21-25, 2008
page 43
Agenda

Schneider/Modbus-IDA/ODVA Relationship
- Value of ODVA to Schneider
- Value of Schneider to ODVA

Current CIP Modbus Status
Purpose and major requirements
Technical solution requirements & goals
Technical solution summary

Technical solution details
- Modbus Features
- CIP Functions supported with Modbus
- Explicit message
- Implicit I/O
- CIP object representing Modbus TCP device
- New Modbus TCP CIP object
- EDS
- Impact on CIP

Brief Modbus/TCP tutorial
### Minimum CIP Objects Required:

<table>
<thead>
<tr>
<th>Object Class</th>
<th>Optional / Required</th>
<th># of Instances</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity</td>
<td>Required</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Message Router</td>
<td>Required</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Connection Manager</td>
<td>Required</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Modbus</td>
<td>Required</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Required</td>
<td>1..n</td>
<td>Device Specific</td>
</tr>
<tr>
<td>Assembly</td>
<td>Required</td>
<td>1..n</td>
<td>Device Specific</td>
</tr>
</tbody>
</table>
CIP Object Representing a Modbus Device 2/4

Identity:
- A minimum set of attributes, populated with data from the Modbus device

Message Router: supported
- Behavior will be managed by the translator

Connection manager: supported
- Behavior will be managed by the translator

Modbus:
- Provides Modbus Table specific services

Parameter:
- Provides access to an individual data item within the Modbus data tables

Assembly:
- Provides a translation into the Modbus data item tables
Identity Object translation

- Attributes of the Identity Object will be populated using data obtained through Modbus Services.
  - An example is FC 43/14

- Device Identification Objects not supported by the Modbus device are represented by zero length strings.

- If a Modbus device does not support the Read Device Identification function then it returns “Unknown Modbus Device”.
CIP Object Representing a Modbus Device 4/4

Modbus Specific Instance Attributes

- All information in ASCII string with no standardization about the content

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Data Type</th>
<th>Required/Conditional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Vendor ID</td>
<td>ASCII String</td>
<td>Required</td>
</tr>
<tr>
<td>2</td>
<td>Device Type</td>
<td>ASCII String</td>
<td>Required</td>
</tr>
<tr>
<td>3</td>
<td>Product Code</td>
<td>ASCII String</td>
<td>Required</td>
</tr>
<tr>
<td>4</td>
<td>Revision</td>
<td>ASCII String</td>
<td>Required</td>
</tr>
<tr>
<td>5</td>
<td>Status</td>
<td>ASCII String</td>
<td>Required</td>
</tr>
<tr>
<td>6</td>
<td>Serial Number</td>
<td>ASCII String</td>
<td>Required</td>
</tr>
<tr>
<td>7</td>
<td>Product Name</td>
<td>ASCII String</td>
<td>Required: returns “unknown Modbus device” w/o FC 43/14</td>
</tr>
<tr>
<td>8-17</td>
<td>Not allowed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Modbus ID info</td>
<td>Device dependent</td>
<td>Conditional: required for FC 43/14</td>
</tr>
<tr>
<td>All others</td>
<td>Not allowed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Agenda

Schneider/Modbus-IDA/ODVA Relationship
- Value of ODVA to Schneider
- Value of Schneider to ODVA

Current CIP Modbus Status
Purpose and major requirements
Technical solution requirements & goals
Technical solution summary

Technical solution details
- Modbus Features
- CIP Functions supported with Modbus
- Explicit message
- Implicit I/O
- CIP object representing Modbus/TCP device
- new Modbus/TCP CIP object
- EDS
- Impact on CIP

Brief Modbus/TCP tutorial
EDS files

Modbus devices are **not required to have EDS files**
- GenericModbusDevice EDS files allow EDS-based originators to connect to Modbus target devices
  - Indicate use of the Assembly Object instances for I/O connections

**Modbus devices may, if desired, provide an EDS file to better describe the device’s specific data**
- E.g., indicate which Modbus data items are in the device; use Param entries to detail device-specific parameters

The **GenericModbusDevice EDS file provides a starting point for a configuration tool to create the I/O Connection setup for a Modbus device represented by a CIP model.**
- Defines the potential CIP capability in any Modbus device.
Agenda

Schneider/Modbus-IDA/ODVA Relationship
   ▶ Value of ODVA to Schneider
   ▶ Value of Schneider to ODVA

Current CIP Modbus Status
Purpose and major requirements
Technical solution requirements & goals
Technical solution summary

Technical solution details
   ▶ Modbus Features
   ▶ CIP Functions supported with Modbus
   ▶ Explicit message
   ▶ Implicit I/O
   ▶ CIP object representing Modbus TCP device
   ▶ New Modbus TCP CIP object
   ▶ EDS
   ▶ Impact on CIP

Brief Modbus/TCP tutorial
Impact on CIP

The goal was to minimize the impact of incorporating Modbus into CIP.

There is some impact on CIP because of Modbus.

CIP Common Specification Enhancement CIPSE-001-078 was approved by the ODVA to support these changes.

The possible and likely sharing of a physical Ethernet port by EtherNet/IP and Modbus TCP creates the need for the changes.
Impact on CIP: Port Object

There are **two new port types** to the Port Object (Port Object Instance Attributes):

- Modbus/TCP
- Modbus/SL

**ModbusTCP** indicates it is a **Modbus TCP port**.
- Indicated by the semantic 201

**ModbusSL** indicates it is a **Modbus Serial port**.
- Indicated by the semantic 202
Impact on CIP: Port Object

Updated the fourth attribute to the Port Object called the Port Name.

- Requires all CIP Ports that share the same physical network to have the same Port Name.
- SHORT _STRING which names the Physical Network Port. (maximum of 64 characters, example Port A).
- Must be unique for each port.
  - One cannot apply the same Port Name to two (or more) Physical Network Ports.
Impact on CIP: Object Library

The “Modbus Object” has been added to the Object Specifications in the CIP Object Library

- The Class Code is 44 Hex.

Addition of the “Modbus Identity Info” attribute to the Identity Object Instance Attributes.

- A conditional attribute for Modbus devices only.
Impact on CIP: EDS

“Write Only” parameter added to the Device Profiles

- Added to the Parameter Object Descriptor Attribute.
- Indicate the Parameter Value attribute can be set and NOT read.

Two new Device Classifications:

- Modbus/TCP
- Modbus/SL

The Port Name field is now conditional.

- Was optional
- Required if the CIP port shares a physical network with another CIP port
Impact on CIP: EDS

Scaling Offset field was changed

- A Field in the Parameter Section, Field Number 16
- Was a data type INT
- Changed to data type DINT
- Increases the range to accommodate Modbus, the mapping is hidden in the translator.
Impact on CIP: General

A new general error code added for Modbus Integration

- "Unknown Modbus Error" hex code 2B
- When an unknown Modbus Exception code is seen be the Modbus translator.
Agenda

Schneider/Modbus-IDA/ODVA Relationship
- Value of ODVA to Schneider
- Value of Schneider to ODVA

Current CIP Modbus Status
Purpose and major requirements
Technical solution requirements & goals
Technical solution summary
Technical solution details
- Modbus Features
- CIP Functions supported with Modbus
- Explicit message
- Implicit I/O
- CIP object representing Modbus TCP device
- New Modbus TCP CIP object
- EDS
- Impact on CIP

Brief Modbus/TCP tutorial
MODBUS® Protocol is a messaging structure introduced to the market by Modicon in 1979, used to establish master-slave/client-server communication between intelligent devices.

Since Modbus protocol is just a messaging structure, it is independent of the underlying physical layer.

It is usually implemented using
- RS232, RS422, RS485,
- TCP IP over Ethernet
- Over a variety of other media (e.g. wireless, fiber, radio, cellular, etc.)
- Modbus Plus, a high speed token passing network

MODBUS® is a de facto standard and open. Schneider Electric has transferred the Specifications for Modbus and MODBUS TCP protocols to Modbus-IDA.
Modbus is a simple Master (Client) / Slave (Server) Application protocol

- The Client prepares and sends a Request to the server
- The Server analyses the request, processes the request, and sends an answer to the Client.

Modbus is a scalable protocol

- A server can implement a subset of Modbus protocol

Modbus comprises Public and User Defined function codes

Modbus application protocol proposes a list of function codes to read or write data or to activate remote processing on a remote server

- Read / Write list of data words, Read/Write list of bits
- Diagnostic, Identification
Modbus TCP Tutorial: Application Data Unit 4/6

The MODBUS protocol defines a simple protocol data unit (PDU) independent of the underlying communication layers. The translation of MODBUS protocol on specific buses or network can introduce some additional fields on the application data unit (ADU).

For Ethernet TCP/IP

MODBUS TCP/IP ADU

MBAP (Modbus Application Protocol) TCP/IP encapsulation header
MODBUS uses a ‘big-Endian’ representation for some addresses and data items.

MODBUS bases its data on a series of tables. The four primary tables are:

<table>
<thead>
<tr>
<th>Primary tables</th>
<th>Object type</th>
<th>Type of</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discretes Input</td>
<td>Single bit</td>
<td>Read-Only</td>
<td>This type of data can be provided by an I/O system.</td>
</tr>
<tr>
<td>Coils</td>
<td>Single bit</td>
<td>Read-Write</td>
<td>This type of data can be alterable by an application program.</td>
</tr>
<tr>
<td>Input Registers</td>
<td>16-bit word</td>
<td>Read-Only</td>
<td>This type of data can be provided by an I/O system.</td>
</tr>
<tr>
<td>Holding Registers</td>
<td>16-bit word</td>
<td>Read-Write</td>
<td>This type of data can be alterable by an application program.</td>
</tr>
</tbody>
</table>
Modbus TCP Tutorial:
All Public Function Code Definitions 6/6

Public function codes:
- cover both Modbus TCP and Modbus SL
- cover Read/write data and other command/services

<table>
<thead>
<tr>
<th>Physical Discrete Inputs</th>
<th>Read Discrete Inputs</th>
<th>Function Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal Bits Or Physical coils</td>
<td>Read Coils</td>
<td>01</td>
</tr>
<tr>
<td></td>
<td>Write Single Coil</td>
<td>05</td>
</tr>
<tr>
<td></td>
<td>Write Multiple Coils</td>
<td>15</td>
</tr>
<tr>
<td>Physical Input Registers</td>
<td>Read Input Register</td>
<td>04</td>
</tr>
<tr>
<td></td>
<td>Read Holding Registers</td>
<td>03</td>
</tr>
<tr>
<td></td>
<td>Write Single Register</td>
<td>06</td>
</tr>
<tr>
<td></td>
<td>Write Multiple Registers</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Read/Write Multiple Registers</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>Mask Write Register</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>Read FIFO queue</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>Read File record</td>
<td>20 6</td>
</tr>
<tr>
<td>Physical Output Registers</td>
<td>Diagnostic</td>
<td>08 00-18,20</td>
</tr>
<tr>
<td></td>
<td>Get Com event counter</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Get Com Event Log</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Report Slave ID</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Read device Identification</td>
<td>43 14</td>
</tr>
<tr>
<td>Diagnostics</td>
<td>Encapsulated Interface</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>Transport</td>
<td></td>
</tr>
</tbody>
</table>

Hannover Messe 2008
April 21-25, 2008
page 65