ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA024

MODBUS APPLICATION AT JEFFERSON LAB

J. Yan, C. Seaton, S. Philip
Thomas Jefferson National Accelerator Facility, Newport News, VA 23606, U.S.A.

Abstract

Modbus-TCP is the Modbus Remote Terminal Unit (RTU) protocol with the TCP interface running on Ethernet. In our applications, an XPort device utilizing Modbus-TCP is used to control remote devices and communicates with the accelerator control system (EPICS). Modbus software provides a layer between the standard EPICS asyn support and EPICS asyn for TCP/IP or serial port driver. The EPICS application for each specific Modbus device is developed and it can be deployed on a soft IOC. The configuration of XPort and Modbus-TCP is easy to setup and suitable for applications that do not require high-speed communications. Additionally, the use of Ethernet makes it quicker to develop instrumentation for remote deployment. An eight-channel 24-bit Data Acquisition (DAQ) system is used to test the hardware and software capabilities.

MODBUS TCP/IP

Modbus is a serial communication protocol widely used by many manufacturers throughout industry since it was published by Modicon in 1979 for use with its programmable logic controllers (PLCs). It has become a standard communication protocol and is now a common available means of connecting electronic devices. Many applications in the field of accelerator control have implemented Modbus to communicate and control devices [1,2,3]. Modbus TCP is a Modbus variant used for communications over TCP/IP network. The Modbus TCP messaging service provides a Client/Server communication between devices connected on an Ethernet TCP/IP network[4]. Figure 1 shows a general Modbus TCP/IP communication architecture which includes different types of device, such as, Modbus TCP/IP Client and Server devices directly connected to the TCP/IP network, the interconnection devices like a bridge, router or gateway for interconnection between the TCP/IP network and a serial line sub-network, which permit connections of Modbus serial line client and server end devices. The messaging service of Modbus TCP/IP has four basic types of messages: Request, Confirmation, Indication, and Response. When a client initiates a Modbus Request message for a transaction, the server side will have a Modbus Indication and send back a Response message. Finally, the Client has the Confirmation message for the received Response message. By using these messages, the Modbus messaging services perform a real time information exchange between devices on the network.

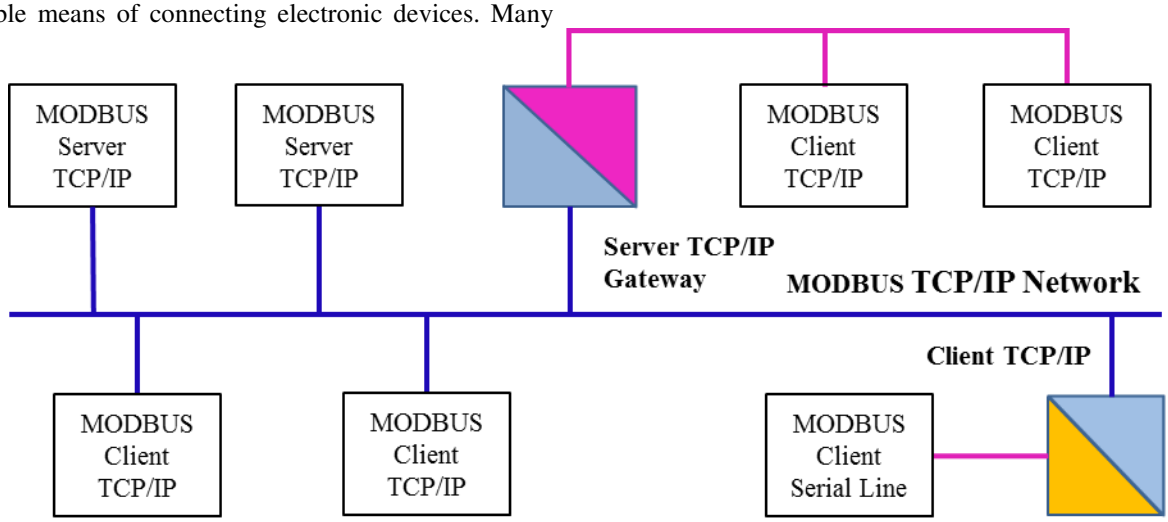

Figure 1: Modbus TCP/P Communication Architecture.

The Modbus protocol defines a basic Protocol Data Unit (PDU) independent of the underlying communication layers. The mapping of the Modbus protocol on a specific network will introduce some additional fields on the Application Data Unit (ADU). Figure 2 shows the Modbus TCP/IP ADU with a dedicated Modbus Application Protocol (MBAP) header which contains the fields of Transaction Identifier, Protocol Identifier, Length, and Unit Identifier. Modbus Function Code defines the various reading, writing and other operation of the Modbus data.

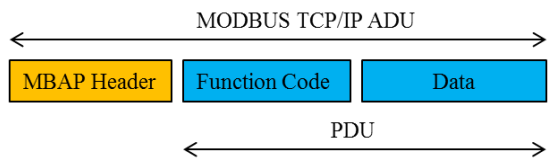

Figure 2: Modbus TCP/IP Request/Response Message.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA024

MODBUS TCP/IP APPLICATIONS

Modbus is a client/server communication model. In our applications, the embedded Ethernet device XPort is designed as the server and a SoftIOC running EPICS Modbus is the client. The SoftIOC builds a Modbus request from parameter contained in a demand that is sent by the EPICS application to the Modbus Client interface. On reception of the Modbus request, the Modbus server activates a local action to read, write, or achieve some other action. Thus, the main Modbus server functions are to wait for a Modbus request on 502 TCP port, treat this request, and then build a Modbus response.

Modbus Server

Xport is an embedded Ethernet device server that includes all essential networking features including a 10Base-T/100Base-Tx Ethernet connection, reliable operating system, embedded web server, and a full TCP/IP protocol stack. It also provides a serial interface having data transfer rate of 300 bps to 921,600 bps. By incorporating XPort, we can easily design products that can be accessed and controlled over the internet. The software tool 'DeviceInstaller' was used to set the IP address and configure the serial port. Based on the XPort, we designed a prototyped Modbus data acquisition (DAQ) board, which includes eight serial 24-bit analog-to-digital converter (ADC) channels, two digital-to-analog converters (DACs), a serial interface, USB connector, digital I/O pins, SDRAM memory buffer, and a field-programmable gate array (FPGA). Figure 3 shows the diagram of the DAQ board. The FPGA assigns a block of registers for each connected device and these registers can be read or written by Modbus protocol. For instance, ADCs convert an analog signal to digital data and store digital data in the FPGA. Then the FPGA sends this data to the XPort via serial communications, and the XPort transfers all data to the remote clients with the Modbus protocol.

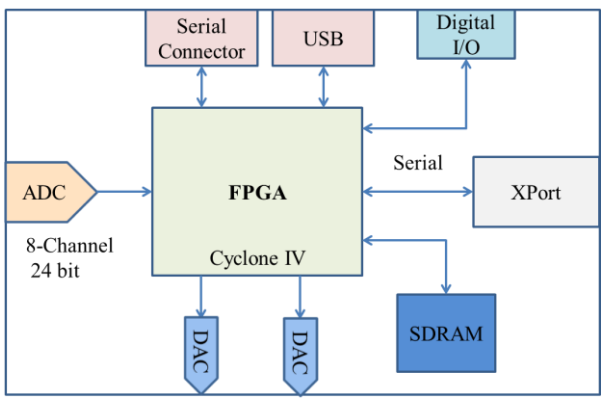

Figure 3: The Diagram of the Modbus DAQ Board.

Four types of Modbus protocol messages are defined for this prototype. They are Write Single Registers Request with function code 0x06, Read Holding Registers Request with function code 0x03, Response message to Read Request with function code 0x03, and the Exception Response with function code 0x88. For instance, a protocol encoding for a Read Request is 0100 0000 0006 01 03 0000 0001. Here, the first four bytes 0100 are the Transaction ID, the second four bytes 0000 are the Protocol ID, the third four bytes 0006 are the Length, 03 is the Function code, the next 0000 is the starting address, and the last 0001 is the quantity of register. For this Request, the Response will be 0100 0000 0005 01 03 02 3478 resulting in two bytes of data 0x3478.

EPICS Modbus

The EPICS Modbus package which is a Modbus Driver Support for Modbus Protocol under EPICS system was installed for the Modbus applications[5]. The architecture of the Modbus module consists of the following four layers: EPICS Asyn Device Support, EPICS Asyn Port driver, Asyn Interpose Interface, and Asyn Port Driver (Figure 3). The EPICS Asyn Device Support is the general purpose device support provided with asyn. It is separated with the Modbus package and there is no special device support needed with Modbus. The EPICS Asyn Port Driver functions as a Modbus client. It communicates with EPICS Asyn Device Support using the standard asyn interfaces, such as asyInt32, asynUInt32Digital, etc. This driver sends and receives device-independent Modbus frames via the standard asynOctet interface to the asyn Interpose Interface layer. The asyn Interpose Interface layer communicates via the standard asynOctet interface to both the overlying Modbus driver (layer 2) and to the underlying asyn hardware port driver. The asyn Port Driver handles the low-level TCP/IP communication. It is a standard port driver provided by asyn, such as drvAsynIPPort. So, we can see only layer 2 and layer 3 on the Modbus package.

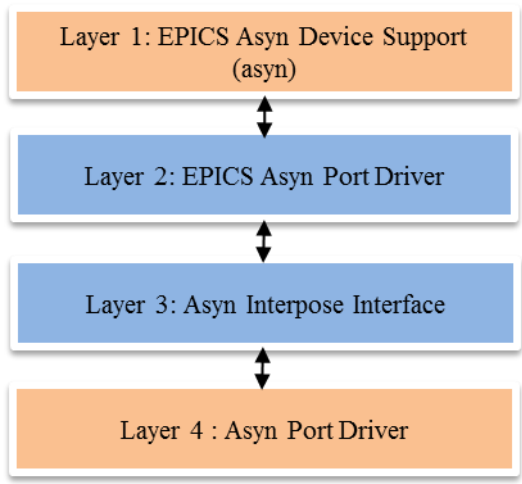

Figure 4: The Architecture of EPICS Modbus Module.

Each Modbus port driver is assigned a single Modbus function code and a single contiguous range of Modbus memory up to 125 words. Before a Modbus port driver can be created, the TCP/IP port driver that directly communicates with the hardware should be first to be generated. The following is step-by-step procedure to create a specific Modbus port driver:

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA024

- Run the function drvAsynIPPortConfigure ("ssa1", "129.57.200.226:502", 0, 0, 1) to create an asyn IP port driver named "ssa1" on port 502 at the Modbus server's with IP address of "129.57.200.226".
- Run the function modbusInterposeConfig("ssa1", 0, 2000, 0) to configure the interpose driver with the name of "ssa1", TCP/IP Modbus link, 2000 ms of timeout, and no write-delay.
- Once the asyn IP port diver is created and the interpose driver is configured, the command drvModbusAsynConfigure("RF_In", "ssa1, 0, 3, 1, 125, 0, 100, "RK") was called to create a Modbus port named "RF_In" on the asyn IP port "ssa1", the slaveAddress or "Unit ID" of "0", Modbus function code 3, the start address "1" for the Modbus data segment, the length of 125 data segment, the default data type 0, polling delay 100 msec, and a parameter "RK".
- Load the database for the Modbus port by calling dbLoadTemplate("ssa1.substitutions").

Modbus for RF Amplifier Control

The Crymodule Test Facility (CMTF) at Jefferson Lab is a facility that conducts performance testing of newly designed cryomodules. An example cryomodule contains eight cavities of nine cells each with coaxial Radio Frequency (RF) power couplers operation at 1.3 GHz and eight Solid State RF Amplifiers (SSA) providing the RF power. The RF amplifier is a self-contained system that produces RF power, encompasses water-cooling and circuit protection, and can also be controlled with a Lantronix XPort. We developed the Modbus application for control of all RF amplifiers. Each amplifier has its own IP Port driver name and multiple Modbus ports, such as, "RF1_In_word" for read and "RF1_Out_Word" for write. Figure 5 shows the control screen of SSA1. Function code 0x06 is applied to write six registers on the SSA, and has operations of Power Supply Enable/Disable, RF Enable/Disable, Power Supply Output Voltage setting, Fault Reset, and System Reboot. Over 500 registers can be read by using function code 0x03. These registers include AC, DC Power Supplier status, RF power, temperature of all devices, cooling water, unit current, and so on. All the read back data will be updated once a second. All eight SSAs are controlled by a single soft IOC that runs in a Linux machine.

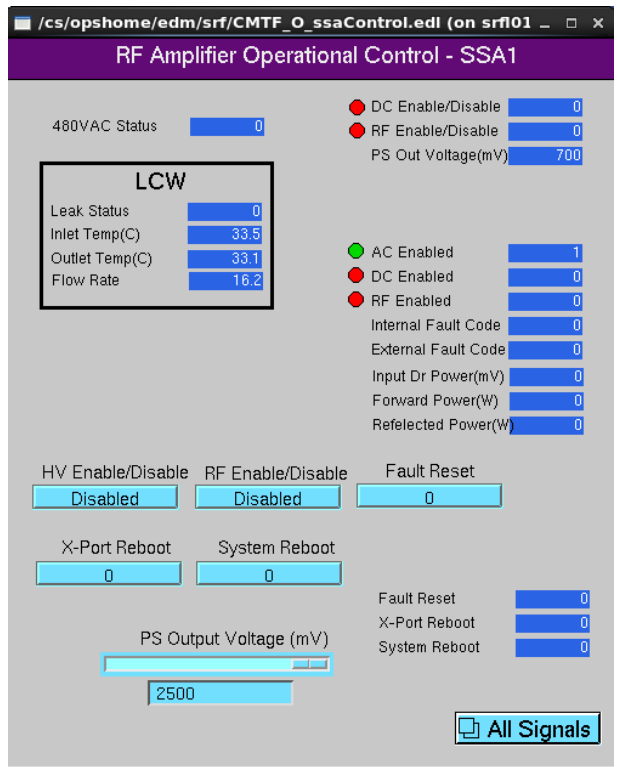

Figure 5: RF Amplifier Operational Control Screen.

CONCLUSIONS

The Modbus data acquisition board, based on the XPort and Modbus TCP/IP protocol, has been prototyped. The board can be used in various applications that require remote communications. The Modbus driver support for Modbus protocol under EPICS is installed and applied for the cryomodule RF amplifier control system. The configuration of XPort, Modbus TCP/IP, and the EPICS Modbus package is easy to setup and suitable for applications that do not require high-speed communications.

REFERENCES

[1] J. Odagiri, etc. "Interfacing Modbus Plus to EPICS for KEK Accelerator Control System", *Proceeding of ICALEPCS'99*, Trieste, Italy, Page 367-369.

[2] S. Cohen, etc. "Modbus/TCP Controller for the Power Supplies in ALS BTS Beam Line", *Proceeding of PAC'07*, Albuquerque, New Mexico, USA Page 425-427.

[3] Y.K. Chen, etc. "Application of Modbus-TCP in TPS Control System", *Proceeding of IPAC'10*, Kyoto, Japan, Page 2719-2721.

[4] Modbus Messaging on TCP/IP Implementation Guide V1.0b, http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf

[5] Driver Support for Modbus Protocol under EPICS R2-10-1, http://cars9.uchicago.edu/software/epics/ModbusDoc.html