Kepware Technologies

U-CON Driver: Evaluating Hardware and Communication Protocols

Febraury, 2013
Ref. 1.02

Table of Contents

1.Overview

This document intends to provide guidelines for evaluating hardware and communication protocols for use with the User-Configurable (U-CON) Driver. It will discuss the basic information that users should determine for a protocol, as well as what questions may be asked of the manufacturer.

1.1 Evaluating Hardware and Communication Protocols

Users should ask the following six questions when evaluating a hardware and communication protocol:

1. How complex is it?
2. How is data presented?
3. How many items will the U-CON Driver request?
4. How are the packets terminated?
5. How does the device communicate?
6. Is the device a slave (in which case U-CON will request data from it) or a master (in which case it will send data to U-CON)?

As users gain experience evaluating protocols and creating U-CON projects, it is likely that additional items and concerns will be added to their list of things to consider.

1.1.1 How Complex Is It?

Learning how to determine the complexity of a protocol can be difficult for users who have never done so before. Most devices that are encountered will have a simple ASCII or Binary protocol, however, and will generally request a piece of data consisting of 4-5 bytes. They will receive a response back from the device.

Users should find out whether the protocol will combine the Device ID and command into one byte, as well as whether it will require a lot of back and forth handshaking under stringent time requirements. Although protocols with these characteristics may still be used, they can add considerable time to development.

Note: Device protocols that require token passing in order to acquire and relinquish communications control are not suitable for U-CON projects.

1.1.2 How is Data Presented?

The U-CON Driver supports most common data types and formats. Users must determine whether the data is ASCII or Binary. If it is ASCII, is it fixed length or variable length? If it is variable length, is there a way to identify the end of the data with a delimiter or packet terminator? Binary data is easier to work with because there are 2 bytes for a Word and 4 bytes for a Float in typical use cases. Users should check whether the Binary response has a defined length.

Note: Protocols that do not have a way to determine where the data ends are not suitable for U-CON projects.

See Also: U-CON (User-Configurable) Driver Help

1.1.3 How Many Items Will the U-CON Driver Request?

This question is mostly a judgment call. Users must remember that the U-CON Driver processes the transaction commands synchronously. Items will be requested individually unless the project is configured to request tag blocks (in which multiple items are included in a single request transaction). Both the method and the frequency at which data is requested will determine how long it takes to acquire data in a single scan cycle.

1.1.4 How Are the Packets Terminated?

It is important to consider how the packets will be terminated. In order for the U-CON Driver to process responses from the device, it must be able to determine the end point of the data. The U-CON Driver has three methods for determining this information: Fixed Length, Termination Characters, and Data Length Field. Descriptions of the methods are as follows:

- **Fixed Length:** In this method, the packet coming from the device will always have the same length in bytes.
- **Termination Characters:** In this method, the packet coming from the device will have a specific character or set of characters that signify the end of the data.
- **Data Length Field:** In this method, one or more bytes near the start of the packet will specify the length of the data being returned.

1.1.5 How Does the Device Communicate?

Because the U-CON Driver is essentially a serial driver, it should be able to connect to any device from a serial port as long as the proper adapter is being used. This includes any PC board, Serial-to-Ethernet adapter, or USB-to-Serial adapter that presents itself as a system serial port. The driver also provides users the ability to encapsulate the packet in an Ethernet packet for use with Serial-to-Ethernet converters and devices that utilize TCP/IP or UDP connections.

Whereas some devices are true Ethernet protocol devices with specialized packets, some devices use Telnet and will broadcast back an unsolicited message once connected. The U-CON Driver may be able to access these types of devices and connections. For more information, refer to Technical Support.

1.1.6 Is the Device a Slave or a Master?

Slave devices (including many Barcode Scanners, Magnetic Card Readers, RFID Readers, and scales) send unsolicited transactions to the U-CON Driver. With these types of messages, U-CON must be able to both receive the packet and determine how to handle it. The slave devices will send one item or one block of items in a single packet; the next packet will be new data. A single unsolicited transaction is typically needed for these types of devices.

Users must be aware of devices that may return many different packets or that have originally been designed to output to an ASCII printer. For these devices, the U-CON Driver must be able to evaluate the packet to update the proper tags. The most efficient way to do this is with Transaction Keys. If that is not a feasible option, the next best method is the Test Character and Test String commands.

Note: Protocols that do not have a way to evaluate the packet are not suitable for U-CON projects.

See Also: U-CON (User-Configurable) Driver Help

2. Utilizing the Protocol Information

At this time, users should put the information presented above into practice by using the provided protocol and applying the discussed methods. Examples of the Command Format are as follows:

```
Read Command Format - <STX><Device ID><Command><ETX>
Read Response Format - <STX><Device ID><Command>,<Data1>, <Data2>,...<ETX>
Write Command Format - <STX><Device ID><Command><Data><ETX>
Write Response Format - <STX<Device ID><Command><Response><ETX>
```

This protocol is fairly simple. The device is a slave because it makes a read request. The data is presented as variable length ASCII that is delimited with commas, and the packets are terminated with "ETX". This protocol can be used in the U-CON Driver.

2.1 Examples of Protocol Types

In the following examples, what is shown on the wire as hexadecimal bytes will be displayed before the packet is broken down and explained. To ease reading, the hexadecimal values will be displayed without the lead "0x" identifier. A space will also be placed between each byte. For example, "0x01 0x03" will be shown as "01 03".

2.1.1 Binary Protocol

```
Holding Register Request
01 03 00 00 00 01   84 0A
```

2.1.1.1 Modbus Holding Register Request

```
Device ID = 01
Function Code = 03
Starting Register = 00 00
Number of Registers = 00 01
Checksum = 84 0A
```

2.1.2 Hexadecimal ASCII Protocol

```
Holding Register Request
3A 30 31 30 33 30 30 30 30 30 30 31 46 42 0D 0A
   01       03       00 00          00 01        FB
```

2.1.2.1 Modbus ASCII Holding Register Request

```
Start Character = 3A ( : )          Number of Registers= 30 30 30 31 (00 01)
Device ID = 30 30 (01)              Checksum = 46 42 (FB)
Function Code = 30 33 (03)          End Characters = 0D 0A (CR LF)
Start Address = 30 30 30 30 (00 00)
```

2.1.3 ASCII Protocol

Unsolicited Packet from a Bar Code Reader
30 31 2D 39 31 36 35 30 30 39 38 2D 30 30 31 0D 0A
01-91650098-001

2.1.3.1 Unsolicited Packet Sent from Bar Code Reader

Lens Inspection System
02 30 31 2C 30 30 39 31 35 2C 50 2C 50 2C 50 2C 50 2C 46 2C 50 2C 46 03
01,00915,P,P,P,P,F,P,F

3.Summary

At this point, users should understand how to start evaluating hardware and communication protocols for use with the U-CON Driver.