# Modbus TCP for the Dynaview III

**Porcine Associates**

**244 O'Connor Street**

**Menlo Park, CA 94025**

**650-326-2669**

**www.dynaviewinc.com**

# Table of Contents

# Modbus TCP for the Dynaview III

## 1 Overview

Modbus TCP is a messaging service protocol optimized for machine-to-machine communication on the factory floor. Typically a Modbus master is connected to one or more Modbus slaves. A master is usually a PC or a PLC. Dynaview III is a Modbus slave.

Another way of thinking about the master-slave relationship is to call them client and server. Modbus slaves (Dynaview) are Modbus servers as they serve data on command from a master. Modbus masters are Modbus clients as they ask to send and receive data from a Modbus server. This paper will refer to the end points as client and server.

Dynaview can communicate with one or more Modbus clients.

Dynaview implements Modbus TCP, which means that it communicates with its Modbus clients via Ethernet using the TCP/IP standards. It does *not* support serial Modbus that uses RS-422.

Modbus is an open standard; read more about it at http://www.Modbus-IDA.org.

Dynaview exposes three distinct objects to the Modbus interface:

1) The current Dynaview state. The state includes such items as measured length, web speed, alarms, and other such transient items. This object is read-only and mapped to the Modbus input registers.
2) A recipe buffer which can contain either the current recipe or can optionally be loaded with a recipe from a Modbus client. This object is read-write and mapped to the Modbus output registers.
3) A bit-mapped communication area for signals between the client and server. This object is read-write and mapped the the Modbus output binaries.

Dynaview can optionally create end-of-roll reports. These reports are not available via Modbus, but are available via FTP (File Transfer Protocol). FTP is a standard method of moving files from one location to another via a network.

## 2 Installation

### *2.1 Physical connection*

All that is necessary is connection to your network via 10baseT or 100baseT Cat-5 cable. There is a network port on the bottom of the Dynaview.

### *2.2 Addressing*

Modbus TCP doesn't use Modbus addresses as does serial Modbus. Instead a Modbus server responds to a port number at a particular network address.

### 2.2.1 Modbus port

Dynaview responds to the standard Modbus TCP port 502.

### 2.2.2 Network address

Every node on a network has a unique network address. If your network has a DHCP server Dynaview will automatically pick up its address from the server. If you don't have a DHCP server or wish to give Dynaview a static address you will need to follow the procedure in appendix A. Note that any particular Dynaview only has one network address; this address is used for the web server, the ftp server, the Modbus server, plus any remote logins via ssh or scp.

# 3 Operation

The Dynaview–Modbus interface operates in two distinct modes, depending on the state of the Modbus R/W bit. This bit can be set via the manage Dynaview screen by an operator and via the Modbus communication object by a Modbus client. A Modbus client can always read the Dynaview state object and can read and write the Dynaview recipe buffer object. A client can acknowledge alarms and signal a roll change via the Modbus communication object. The difference in the two modes lies in how a recipe is made active in the Dynaview.

## *3.1 Modbus read–only mode, Modbus R/W bit cleared*

In this mode the client cannot load the recipe from the recipe buffer object as the current active recipe. The local operator has access to the entire local recipe storage and management screens on the Dynaview. Any time an operator changes a recipe the Modbus recipe buffer is updated with the change.

## *3.2 Modbus read–write mode, Modbus RW bit set*

In this mode the client can load a recipe and make it active. The local operator does not have access to local recipe storage and cannot load or modify the active recipe.

The procedure to load and activate a recipe remotely is as follows:

1) Load the desired recipe into the Modbus recipe buffer object.

2) Set the "Load recipe from recipe buffer" coil in the Modbus communication object.

3) Monitor the "Recipe Error" coil in the Modbus communication object. If it goes true, the Modbus R/W bit is false and remote recipe management is disabled.

4) Watch the "Load recipe Rec'd" coil in the Modbus communication object. When it goes true the Dynaview has received the load command. Reset the "Load recipe from recipe buffer" coil. Alternatively, wait at least 30 seconds and the "Load recipe from recipe buffer" coil will automatically reset.

# 4 Supported Modbus commands

| Modbus Function Code | Current Terminology | Classic Terminology |
|---|---|---|
| **16-bit Access** | | |
| 3 | Read Multiple Registers | Read Holding Registers |
| 4 | Read Input Registers | Read Input Registers |
| 6 | Write Single Register | Preset Single Register |
| 16 (10 Hex) | Write Multiple Registers | Preset Multiple Registers |
| 22 (16 Hex) | Mask Write Register | Mask Write Register |
| 23 (17 Hex) | Read/Write Registers | Read/Write Registers |
| **Bit access** | | |
| 1 | Read Coils | Read Coil Status |
| 2 | Read Inputs Discretes | Read Input Status |
| 5 | Write Coil | Force Single Coil |
| 15 (0F Hex) | Force Multiple Coils | Force Multiple Coils |
| **Diagnostics** | | |
| 7 | Read Exception Status | Read Exception Status |
| 8 sub code 00 | Diagnostics – Return Query Data | Diagnostics – Return Query Data |

Data types:

| | |
|---|---|
| string | Most strings are null terminated, but if a string takes up the entire allocated space it will not be null terminated. Strings are packed two 8-bit characters per register. |
| int<br>unsigned int | Integers have 32 bits; the least significant 16 bits are in the first register. As an example a 32-bit "1" is represented as 1 in the first register and 0 in the next register.<br>By convention the most significant bit of an integer is a sign bit. This allows a signed 32-bit integer to take on values from -2,147,483,648 to 2,147,483,647. An unsigned integer has no sign bit, so a 32-bit unsigned integer can take on values from 0 to 4,294,967,296, |
| float | 32-bit IEEE floating point. The first register has the least significant 16 bits, the next register the most significant. |
| bit-mapped | Individual bits are set and reset independently. |

# 5 Coil and register mapping

## *5.1 Input registers (read–only)*

Commands 0, 4, 23

Mapped to the Dynaview state object.

Each address refers to one 16-bit register. Most Dynaview variables are 32 bits in length, so it takes two Modbus registers to represent them.

| **Address** | **Data Type** | **Contents** |
|---|---|---|
| 0,1 | bit-mapped | Status – bit encoded, see below |
| 2,3 | bit-mapped | Alarm – bit encoded, see below |
| 4,5 | float | Length Average ; |
| 6,7 | int | Length Average count ; |
| 8,9 | float | Target Value ; // from recipe |
| 10,11 | unsigned int | Image Count ; |
| 12,13 | float | Roll Length ; |
| 14,15 | float | Web Speed ; |
| 16–20 | string | Folder Name; |
| 21–40 | string | Folder Description; |
| 41–60 | string | Job Name ; |
| 61–70 | string | Roll Name ; |
| 71–75 | string | Recipe Name ; |
| 76–95 | string | Recipe Description ; |
| 96–97 | string | Image Length Units ; |
| 98–101 | string | Speed Units ; |
| 102–103 | string | Roll Length Units ; |
| 104–116 | string | Roll Start Time Stamp |

The rest is private data

Status word bitmap:

Note that command 7 only returns the first 8 bits of this register.

| **Bit** | **meaning** |
|---|---|
| 0 | SaveDataF – true if saving data for a report |
| 1 | PauseDataF – true if pausing in a report |
| 2 | Heartbeat – blinks with a 4 second period |
| 3 | Echo – replicates the echo bit in the communication area |
| 4 | a major alarm is present |
| 5 | a minor alarm is present |
| 6 | Process steady – (currently unused, replicates bit in the communications area) |
| 7 | not used |
| 8 | Modbus Alarm Ack Rcvd – This bit replicates the Alarm Ack bit in the communications area |
| 9 | Modbus New Roll Rcvd – This bit replicates the New Roll bit in the communications area |
| 10 | Modbus Recipe load command Rcvd – This bit replicates the Recipe Load bit in the communications area |
| 11 | Modbus Recipe load error – This bit indicates that a Recipe Load command was received but the Dynaview is currently set up to use local recipes only. |
| 12 | Modbus R/W - This bit replicates the Modbus R/W bit in the Output binaries. |
| 13–31 | reserved |

Note that bits 8–11 also appear in the Modbus communication object.

Alarm word bitmap:

| **Bit** | **meaning** |
|---|---|
| 0–3 | Private |
| 4 | Major Alarm Active |
| 5 | Minor Alarm Active |
| 6–31 | Reserved |

## *5.2 Output registers (read–write)*

Commands 3, 6, 16, 22

This object is mapped to a Dynaview recipe buffer.

| **Address** | **Type** | **Contents** |
|---|---|---|
| 0–15 | reserved | reserved |
| 16 | bit encoded | recipe status (see below) |
| 17–21 | string | Recipe Name |
| 22,23 | int | image length units enumeration (see below) |
| 24,25 | int | web speed units enumeration (see below) |
| 26,27 | float | Wheel Factor |
| 28,29 | float | PWM Span |
| 30,31 | float | Target Value |
| 32,33 | float | Chart Width |
| 34,35 | float | ignore footage after roll change |
| 36,37 | float | High Limit alarm level |
| 38,39 | float | High Warning alarm level |
| 40,41 | float | Low Limit alarm level |
| 42,43 | float | Low Warning alarm level |
| 44,45 | int | Minimum Prescale Value |
| 46,47 | unsigned int | Average Num |
| 48,49 | unsigned int | Report sampling Interval |
| 50,51 | unsigned int | How Many samples per sampling interval |
| 52,53 | float | Chart cal (unsupported) |
| 54,55 | float | Eye Gate Offset |
| 56,57 | float | Eye Gate Width |
| 58,59 | unsigned int | Gates Per Sync |
| 60–79 | string | recipe description |

Recipe Status bit encoding:

| **Bit** | **Meaning** |
|---|---|
| 0 | Fixed Averaging enabled |
| 1 | Warning ALARMS enabled |
| 2 | Limit ALARMS enabled |
| 3 | Eye Light On Dark enabled |
| 4 | Text Reports enabled |
| 5 | Graphic Reports enabled |
| 6 | Keep Data for reports and storage |
| 7 | Continuous Reports enabled (store every data point) |
| 8 | Eye Gating enabled |
| 9 | Eye Automatic mode enabled (currently unused) |
| 10–15 | reserved |

Image length units enumeration:

| **value** | **Meaning** |
|---|---|
| 0 | inches |
| 1 | feet |
| 2 | meters |
| 3 | centimeters |
| 4 | millimeters |
| 5 | yards |

Web speed units enumeration:

| **value** | **Meaning** |
|---|---|
| 0 | inches/second |
| 1 | inches/minute |
| 2 | feet/minute |
| 3 | centimeters/second |
| 4 | meters/second |
| 5 | meters/minute |
| 6 | yards/minute |

## 5.3 Input binaries (read–only)

Command 2

Mapped to the same memory area as Input Registers above. Registers bit encoded and are returned as 16 coils, so that that the address for the coil that represents bit 0 of any register is the register address multiplied by 16.

## 5.4 Output binaries (read–write)

Commands 1, 5, 15

This object is mapped to the Dynaview Modbus communication area

32 coils as show below

| **Coil #** | **Direction** | **Meaning** |
|---|---|---|
| 0 | Client→Server | Acknowledge alarm (rising edge-triggered, see below) |
| 1 | Client→Server | New roll (rising edge-triggered, see below) |
| 2 | Client→Server | Load recipe from recipe buffer (rising edge-triggered, see below) |
| 3 | Client→Server | Web stable (currently unused) |
| 4 | Client→Server | Echo input (simply copied to echo bit in the state status word above) |
| 5–15 | | reserved |
| 16 | Server→Client | Ack Rec'd (mirror of bit 0 above) |
| 17 | Server→Client | New roll Rec'd (mirror of bit 1 above) |
| 18 | Server→Client | Load recipe Rec'd (mirror of bit 2 above) |
| 19 | Server→Client | Recipe Error – Load recipe rec'd but Dynaview is currently set up for local recipes only. |
| 20 | Client→Server<br>Server←Client | 0 – Dynaview uses local recipes only.<br>1 – Dynaview suppresses local recipes and allows Modbus to load recipes remotely.<br>(edge triggered – rising or falling edge)<br>*Note that this bit can be set/reset in the Manage Dynaview recipe screen locally on the Dynaview as well as here.*<br>This bit also appears (read-only) in the status word of the state object. |
| 21–31 | | reserved |

Note that bits 16–19 are replicated in the Dynaview state status word, bits 8–11

Dynaview only acts on edge-triggered bit-values when the value changes. The Modbus master can handle these bit-values in two ways:

- (Open-loop) Leave the bit in the ‘1’ state long enough to ensure that Dynaview has seen and reacted to it.
- (Closed-loop) These edge-triggered bits are echoed here in bits 16–18 and in the state status word bits 8–11. To ensure that Dynaview sees the bit, first set it, then watch the associated echoed bit. When it goes positive, reset the bit.

## *5.5 Diagnostics (read–only)*

Command 7

This command is mapped to the first 8 bits of offset 0 in the Input Registers (status).

# Appendix

## A. Procedure to give Dynaview a static network address

### *a. Connect to the Dynaview and log in as root*

You will need to connect with a serial cable to the Dynaview. Use a laptop and a terminal emulator. Hyperterminal is a terminal emulator that ships with Windows. You can use Minicom on a Linux or Unix system.

Set the serial port to 115.2K baud, 8 bits, 1 stop bit and no parity.

When connected push the <ENTER> key and you should see a login prompt on the screen. The bold face below is the prompt from the Dynaview. The normal text represents commands that you should type. <enter> is the enter key on the keyboard. Linux commands and filenames are case-sensitive.

```
DynaviewIII login: root<enter>
Password: <enter root password here>
DynaviewIII root#
```

### *b. Modify the interfaces.static file*

Navigate to the /etc/network folder:

```
DynaviewIII root# cd /etc/network<enter>
```

Now edit the file “interfaces.static”. The text editor *vi* is available on the Dynaview. VI is a moded editor – that is, it has a command mode and a text entry mode. When you start VI it is in command mode. Here are a few of the most important commands:

- You can move the cursor around with the arrow keys and with home and end keys when in command mode.
- x – delete the character under the curser
- dd – delete the line under the cursor
- r – replace the character under the curser
- i – start text entry mode, insert text before the cursor
- a – start text entry mode, insert text after the curser
- <esc> return to command mode from text entry mode
- :wq – write the file and exit

- :q! – exit without writing the file (good if you get completely confused!).

Start the editor:

```
DynaviewIII network# vi interfaces.static<enter>
```

Edit the four lines address, netmask, network and gateway with the addresses appropriate for your network. See your local network administrator for details.

Write the new file and exit vi with the command :wq.

### *c. Overlay the interfaces file*

Overwrite the current interfaces file with the edited interfaces.static file. “cp” is the copy command.

```
DynaviewIII network# cp interfaces.static interfaces<enter>
```

### *d. Restart the network*

Restart the network interface:

```
DynaviewIII network# ifdown eth0<enter>
DynaviewIII network# ifup eth0<enter>
```

Check to see if the changes got done correctly:

```
DynaviewIII network# ifconfig<enter>
```

This command will write information about each active network interface to the terminal. You should see an entry for eth0 and an entry for lo (the loopback interface). The eth0 entry should reflect the changes you made to the file above.

### *e. Done – log out*

```
DynaviewIII network# exit<enter>
```

Now disconnect from the Dynaview.