Trio Motion Technology Ltd.
Shannon Way, Tewkesbury,
Gloucestershire. GL20 8ND
United Kingdom

Tel: +44 (0)1684 292333
Fax: +44 (0)1684 297929
Email: apps@triomotion.com
Web: www.triomotion.com

Doc No.: **TN20-36**
Version: **1.0**
Date: **20th July 2001**
Subject: **Guide to using the Modbus Toolkit**

Application Note

Modbus Toolkit.

The toolkit consists of a Motion Perfect Project that contains a set of useful programs to assist in de-bugging most types of Modbus links with the Motion Coordinator. Here is a list of programs described in this document.

Modb_sys.bas	Example program to enable Modbus driver in system firmware.
Test232.bas	Displays incoming Modbus data blocks.
Mods_v07.bas	A fully functioning Modbus driver written in *Trio*BASIC.
Qcrc_tab.bas	Generates CRC lookup table for Mods_v07.
Modm_sym.bas	A Modbus Master simulator written in *Trio*BASIC.

What is Modbus?

Modbus RTU is a protocol layer that works over a RS232 or RS485 serial communications link. Trio's Motion Coordinator range responds to a sub-set of Modbus commands as a slave node. All communication between the HMI and the Motion Coordinator starts with the master (HMI) sending an appropriate request message. If the request is accepted, the Motion Coordinator then sends a reply packet back to the master. Unrecognised commands are simply ignored.

Generally, once powered-up, the Modbus master will send out a request packet at half second intervals until it gets a reply from the slave. Once the first reply is received, the master then sends request packets more frequently and displays the data depending on its internal program. If the link is broken, the master goes back to sending requests "blind" every 500 msec.

As the Motion Coordinator is the slave device, this can lead to problems when setting up the link for the first time. Until a good link is established, the Motion Coordinator will appear to be doing nothing so it is difficult to find out why the Modbus link is not working. The toolkit is designed to get over this problem.

Enabling the Modbus Driver firmware.

The supplied program **Modb_sys.bas** is an example of how to initialise either Port 1 or Port 2 as the Modbus port. Change the value of **mport** to enable the required serial port and set up the port to be the same as the HMI settings, i.e. parity, stop bits and baud rate must be the same at both ends of the link. **ADDRESS** should be set to 1 for point-to-point or to the required Modbus address in a RS485 multi-drop network.

Checking that HMI is sending Modbus request block.

Open terminal #5 in Motion Perfect and run Test232.bas. If the HMI is sending Modbus request packets, they will be printed in the terminal as a list of codes. The program attempts to sync to the Modbus packets and print them in a logical formatted manner. If the link is working correctly the terminal will display something like this:

1 3 0 1 0 8 21 204

The packet is decoded as follows:

1		Address of slave node
3		Code for "Read Holding Register" (or read VR() in Motion Coordinator)
0	1	16 bit value selects first Holding Register to be read
0	8	16 bit value gives number of Holding Registers in block
21	204	16 bit cyclic redundancy check

Running the *Trio*BASIC Modbus driver.

This program implements a fully functioning Modbus slave node but has the advantage that debug PRINT statements can be added to report the status of each Modbus message packet. To use the program, first set up the required port number (**mport**) and the node address, (**mod_addr**) baud rate etc. in the com_init routine. Note that the port is used in standard mode – NOT Modbus mode – as the BASIC program takes care of the protocol. The ADDRESS parameter in this case must be set to 255.

Run Qcrc_tab first then start the Mods_v07 program. The following information will be reported to terminal #5 when Modbus packet transfers take place:

fr rx - This message shows that a good modbus packet (or frame) was received.

The following additonal messages indicate the type of transfer requested and indicate that the Motion Coordinator has responded.

I/O Read or Write:

R io [number of I/O points requested] **A** [First address]

W io [number of I/O points to be set] **A** [First address]

Holding Register (VR) Read or Write:

R hr [number of VRs requested] **A** [First address]

W hr [number of VRs to be written] **A** [First address]

For example the request packet shown at the top of this page will result in these messages if it is successful:-

fr rx
R hr 8 A 1