TechNote

Model 5300 Peer-to-Peer Communications

15

Control Technology Corporation, Hopkinton, MA • 800.282.5008 • www.ctc-control.com

What is it?

CTC Peer-to-Peer communications allows a 5300 controller to monitor (read) and change (write) a remote CTC controller's registers (variables), over an Ethernet network. It is based upon a high speed CTC communications mechanism called CTC Binary Protocol. Register update times are dependent upon network traffic, communication settings, and how fast the remote device can fetch the requested data. That said, typically the 5300 will be able to read and update 50 remote registers from another 5300 within 2.5 ms of it's request; and can write one remote register in less than 20 ms. A 5300 controller can act as a master or a remote on this network, whereas older CTC controllers (52xx, 51xx, 27xx controllers with a 2717) can only act as remotes. CTC controllers older than the 2700 are not supported.

What can I do with it?

The CTC Binary Protocol provides a very simple way for a master 5300 to monitor and control functionality on the remote CTC controller(s). Designated registers on the remote controller are made public and a copy of their contents is periodically transmitted across the network, to the requesting 5300 controller(s), thereby making those registers appear as though they are local on the requesting 5300 system. The update scan time is configurable and the registers may be read from or written to in a manner similar to normal registers or variables on the master controller. Multiple 5300 masters are supported on the network.

Typical uses:

- Synchronizing operations between multiple controllers
- Allowing an HMI on the master to monitor and change functionality on a remote controller.
- Provide a central point for data logging.
- Allow a new 5300 controller to be added to an existing system with older CTC controllers.

How does it work?

The 5300 can map a group of registers from a remote controller to registers (variables) in the 5300. Up to ten register groups can be mapped and groups can come from the same or different remote controllers. Each group can access up to 256 registers in the remote controller(s) however there is a maximum of 2000 total registers that can be mapped by a given master. The registers in the remote controller can be consecutive or randomly located. Once the mapping is set up, the registers in the master 5300 automatically update with the remote data at the update rate (typically set to 100 ms). Writes from the master are sent out on-demand one at a time when their value is changed on the master.

The 5300 master communicates to the remote controller(s) over a standard Ethernet network connecting directly to the Ethernet ports on Blue Fusion controllers, or the 2717 Ethernet card on 27xx series controllers.

Peer-to-Peer Protocol Registers

Registers 21000-21299 are read/write registers that are reserved for peer-to-peer networks. Each block of 10 sequential registers is assigned to a designated peer node and defines the peer environment for that connection. Also note that
this register block can be used for many other functions, besides Peer-to-Peer, such as Modbus, interfacing in a similar manner. For more information refer to the CTC website.

Initiating a Peer-to-Peer Session

In general, the initializing of the peer-to-peer mechanism works as follows:

- Write the desired number of registers (maximum of 256) to 21XX5 register.
- Write the remote's IP address to 21XX0 - 21XX3 register
- Write the register to begin reading from the remote device to 21XX4
- Write a 1007 to register 21XX8 to select the re-mapping area.
- Write where in the 23000-24999 register range you want the remote register to appear to the register 21XX9.
- Write a 0 to the index register 21XX8 to default it back to viewing the first data item.
- Write the scan time, typically 100ms to register 21XX6, to initiate the connection and begin Peer-to-Peer.

Monitor status register 21XX7 for a 1 prior to reading/writing to either the 21XX9 data area or the re-mapped area in the 23000-24999 block.

Programming Examples

Either QuickStep 2 or QuickBuilder may be used to set up the Peer-to-Peer communications link. As a reference, a QuickStep2 and a QuickBuilder example are given below to show how to set up a Peer-to-Peer session. Be sure to check the 5300 Quick Reference Register Guide on our web site for the latest register assignment information and any important user notes:
http://www.ctc-control.com/customer/techinfo/docs/5300_951/951-530006.pdf.

QuickStep2 Peer-to-Peer Example

The following Quickstep2 example is based upon the example given in the Model 2217 Ether Networking Module Installation Guide. You can find the original at our website here: http://www.ctc-control.com/customer/techinfo/docs/2217IG.pdf. Note: The caveats listed at the end of this example apply to all controllers set up in a peer-to-peer networking scheme.

General Notes:

When you design peer-to-peer programs, the following conditions apply:

1. All nodes, including PCs, must have unique node numbers in order to place these values in the peer-to-peer registers (21000-21999).
2. You must check the transaction status registers (24000-24999) before using the data in the peer access registers (23000-23999) or after sending data to another controller.

The following sample Quickstep2 program demonstrates peer-to-peer networking:

```
[1] INITIALIZE_MASTER
   ;;; This program is used to initialize the TCP port
   ;;; for the Master TCP operation. A single
   ;;; device is polled using device ID 1 and 256 registers
   ;;; are read and mapped into the 23000 block. Therefore
   ;;; registers 23000-23255 are used, with 23000 referencing
   ;;; remote register #501. Make sure your remote
   ;;; device has at least 255 consecutive registers starting
   ;;; at '501' otherwise Exception errors will occur.
   ;;; Begin by doing the following:
   ;;; 21005= 255= Set number of registers to read 255 (maximum 256)
   ;;; 21000-21003= Set this to the IP address to connect to.
   ;;; 21004= 501= remote start register '501'
   ;;; 21008= 1003= Set index to point to protocol register
   ;;; 21009= 8= CTC Binary Master UDP, 9 is TCP (think of 5300 being CTCMON or VB)
   ;;; 21008= 1007= Set index to point to where to view data.
   ;;; 21009= 23000= Start re-mapped area at 23000 for 255 regs.
   ;;; 21008= 0= Always set the index back to 0 before begin.
   ;;; 21006= 100= Set scan poll time to 100ms/block read,
   ;;;        min=100ms. This also initiates polling.
   ------------------------------------------------------------------------
   <NO CHANGE IN DIGITAL OUTPUTS>
   ------------------------------------------------------------------------
```

```
   store 255 to R21005_MAX_REGS
   store 192 to R21000_IP_OCT1
   store 168 to R21001_IP_OCT2
   store 1 to R21002_IP_OCT3
   store 153 to R21003_IP_OCT4
   store 501 to R21004_MOD_START_REG
   store 1003 to R21008_INDEX
   store 8 to R21009_MOD_CONFIG
   store 1007 to R21008_INDEX
   store 23000 to R21009_MOD_CONFIG
   store 0 to R21008_INDEX
   store 100 to R21006_POLL_TIME
   goto Next

[2] WAIT_FOR_ONLINE
   ;;; Once the Master starts to poll, we must wait until
   ;;; it is online before proceding.
   ------------------------------------------------------------------------
   <NO CHANGE IN DIGITAL OUTPUTS>
   ------------------------------------------------------------------------
   if R21007_STATUS=1 goto MASTER_ONLINE
   delay 500 ms goto WAIT_FOR_ONLINE

[3] MASTER_ONLINE
   ;;; It is ok to read and process data now since the Master
   ;;; is online to the remote device. If you wish to monitor another
   ;;; device other than unit ID 1, then change the index register
   ;;; 21008 to 1005 and write the desired ID to register
   ;;; 21009, then set 21008 back to 0 and monitor 21007 for
   ;;; a '1' for online state once again. Results will appear
   ;;; in the 23000 block.
   ------------------------------------------------------------------------
   <NO CHANGE IN DIGITAL OUTPUTS>
   ------------------------------------------------------------------------
   delay 1000 ms goto MASTER_ONLINE
```

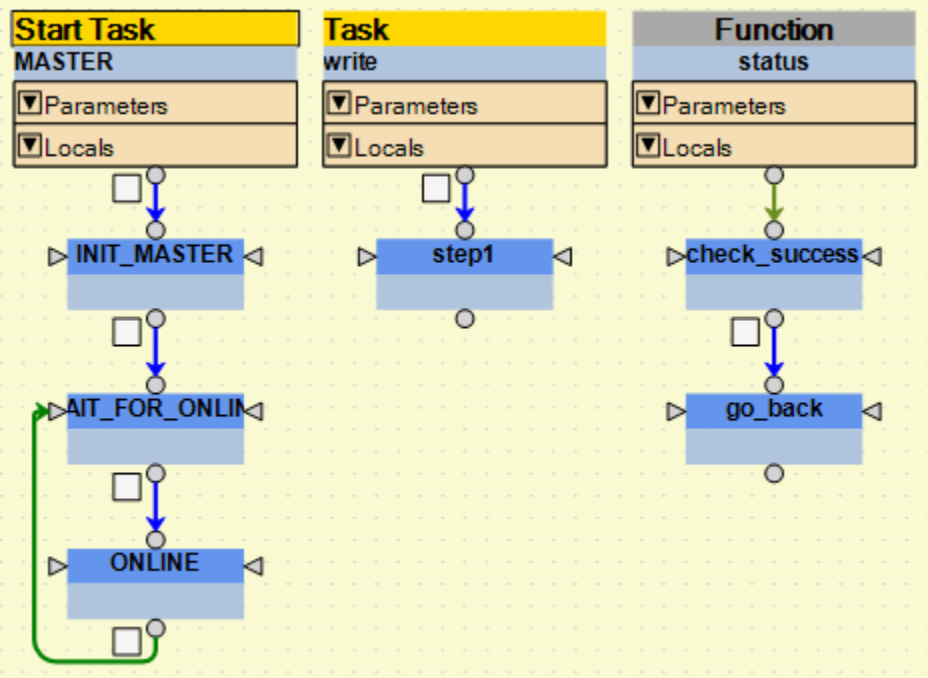

Caveats on using Peer-to-Peer Communications

Be aware of the following caveats of peer-to-peer networks:

1. A node may not always be operational. Retry a transaction with the appropriate action.
2. Limit the number of hosts on your network or nobody will be listening.
3. Do not bog down the network with unnecessary traffic.

QuickBuilder:

Now let's take a look at how we would set up Peer-to-Peer on the 5300 controller using the QuickBuilder development environment. Here's a simple QuickBuilder project that uses the Peer-to-Peer network to automatically read and write registers in a remote 2700 controller. The graphical SFC view of the project is shown below. The code for each of the steps is also included. To replicate this example simply setup the project steps and then cut and paste the code into the QuickStep4 editor. Modify the registers, IP addresses, etc as needed.



There are two tasks in this project (MASTER and write). This project also uses a function called "status" that checks for successful writes. The code below was copied directly out of each of the steps from the QuickBuilder Step Editor. The resource listing for the project is given at the end of this example.

Start Task: **MASTER**

step: **INIT_MASTER** consists of the following comments (green print) and instructions:

```
/*INITIALIZE_MASTER
A single device is polled using device ID 1. 255 registers
are read and mapped into the 23000 block. Therefore
registers 23000-23255 are used, with 23000 referencing
register #501 in the slave controller. Make sure your slave
controller has at least 255 consecutive registers starting
at '501' otherwise Exception errors will occur.
Begin by doing the following:
21005= Maximum number of registers to read (255)
21000-21003= Set this to the IP address to connect to.
21004= Slave Controller start register '501'
21008= 1003= Set index to point to protocol register
21009= 2= Set protocol (8=TCP, 9=UDP,
21008= 1007= Set index to point to where to view data.
21009= 23001= Start re-mapped area at 23000 for 255 regs.
21008= 0= Always set the index back to 0 before begin.
21006= 100= Set scan poll time to 100ms/block read,
       min=50ms. This also initiates polling.*/

begin write;
R21005_MAX_REGS =255;
R21000_IP_OCT1 =192;
R21001_IP_OCT2 =168;
R21002_IP_OCT3 =1;
R21003_IP_OCT4 =153;
R21004_MOD_START_REG =501;
R21008_INDEX =1003;
R21009_MOD_CONFIG =8;
R21008_INDEX =1007;
R21009_MOD_CONFIG =23000;
R21008_INDEX =0;
R21006_POLL_TIME =100;
```

step: **WAIT_FOR_ONLINE** consists of the following comments (green print) and instructions:

```
/* WAIT_FOR_ONLINE
    Once Modbus Master sterts to poll, we must wait until
    it is online before proceding.*/
if R21007_STATUS==1 goto ONLINE;
delay 500 ms;
goto WAIT_FOR_ONLINE;
```

step: **ONLINE** consists of the following comments (green print) and instructions:

```
/*CONTROLLER_ONLINE
    It is ok to read and process data now since controller
    is online to the device. If you wish to monitor another
    device other than unit ID 1, then change the index register
    21008 to 1005 and write the desired ID to register
    21009, then set 21008 back to 0 and monitor 21007 for
    a '1' for online state once again. Results will appear
    in the 23001 block.*/
delay 1000 ms;
```

Start Task: **write**

step: **step1** consists of the following comments (green print) instructions:

```
/*This step writes to registers 23000-23004 on the master thus sending
the same information to registers 501-505 on the slave controller.*/

R23000=1;
call status;
R23001=2;
call status;
R23002=3;
call status;
R23003=4;
call status;
R23004=5;
call status;
```

Function: **status**

step: **check_success** consists of the following comments (green print) instructions:

```
/*This step checks status register 21007 to ensure the register
information was written to the slave.*/

when R21007_STATUS ==1 goto go_back;
```

step: **go_back** consists of the following comments (green print) instructions:

```
/*After checking the status of register 21007 we return to the program
to perform the next instruction.*/

return;
```

The previous example initiates the master controller then writes to the first five registers of the block defined. Notice that much of this program was ‘copied and pasted’ from the

QuickStep2 program. If there are questions regarding the functionality of this example or peer-to-peer networking, in general you can email customersupport@ctc-control.com or call 1- 800.282.5008