CP/NET on the WIZ81xMJ or WIZ550io

- DRAFT -

11/09/18

Background

CP/NET uses an 8-bit “node ID” to identify all participants on a network. This is roughly equivalent to the modern IP Address. Since the WIZ81xMJ requires IP addresses (as do modern networks to which it will be attached), a scheme is needed to relate CP/NET node IDs to IP addresses.

The local (client) node is assigned a CP/NET Node ID using the PMAGIC register. This register would be used for it’s original purpose if the device were to be connected using PPPoE, as would be done when directly connecting to an Internet Service Provider. Since it seems unlikely that an H89 would be connected directly to a cable/DSL modem, it should be fine to use the PMAGIC register for the CP/NET Node ID. This establishes the correlation between the local CP/NET Node ID and the IP address assigned to the device, giving the H8/H89 an IP Address. This node ID may be any value between 01H and FEH, excluding values used for remote servers. Note that CP/NET assumes that node 00H will be a server, so this value is not used for a client ID.

Remote (server) nodes are configured into the socket registers (up to 4 in the WIZ81xMJ, 8 in the WIZ550io). The Source Port (PORT) field uses a convention from which the CP/NET Node ID can be inferred. The source port for each socket must have 31H as it’s high byte. This also allows easy determination of whether a given socket has been configured for a remote server. The low byte of the source port contains the remote server’s CP/NET Node ID. This may be any value from 00H to FEH, excluding the local CP/NET Node ID.

The Destination Port (DPORT), along with Destination IP (DIPR), specify the remote connection. These values are whatever is used (for listening) on the remote node, which must be running code that will recognize CP/NET requestor packets sent over the socket and provide meaningful CP/NET response packets in return.

This implementation for CP/NET always uses TCP/IP Sockets to communicate with remote servers. The socket connection is established on the first send, and remains active until manually deconfigured, system RESET (confirm this), or power off (confirm this). Presumably, some mechanism on servers (such as TCP keep-alive) will ensure that orphaned connections are eventually discovered and cleaned up.

Configuration

The utility WIZCFG.COM may be used to program essential values into the WIZ550io. Note that all memory in these devices is volatile, apparently even across RESET, and so the values must be reprogrammed each time the system is booted.

WIZCFG.COM uses the following command syntax:

```
WIZCFG cmd param...
```

Where ‘cmd’ is a single-character command, followed by one or more parameters (depending on command). The following commands are recognized:

```
N node-id
I ip-addr
G gateway-ip
S sub-net-mask
M mac-addr
{0-7} node-id ip-addr port
```

- The commands ‘0’, ‘1’, ‘2’, ... ‘7’ (max ‘3’ on the WIZ81xMJ) are used to setup remote servers using sockets 0, 1, 2, ... 7, respectively.
- ‘node-id’ is a CP/NET Node ID and must conform to the range required by its context (local/client vs. remote/server IDs). This value is in hex.
- IP Addresses (‘I’, ‘G’, ‘S’, param 2 of {0-7}) use dotted decimal format.
- MAC addresses use colon-hex format.
- Port numbers are in decimal.

Running WIZCFG with no command will show the current configuration of the device.

Note that the list of (up to 8) remote servers is not managed dynamically/automatically. This means that a maximum of 8 (4) remote servers may be used at the same time. It is possible to change the list of servers without restarting CP/NET, under certain conditions. Note that each server can host up to 16 drives and up to 16 LST devices, and so 8 (4) servers may represent a very large set of resources. Attempts to contact a server that is not currently configured will result in an error.

WIZCFG also configures some basic settings in the device every time it runs. For WIZ81xMJ it sets all of the buffers to 1K, and enables Bus Indirect mode with Auto-increment. The SNIOS also sets Bus Indirect mode with Auto-increment. Any outside access to the device must maintain these settings to assure proper operation of CP/NET.

Implementation Details

CP/NET in general did not/does not support spontaneous messages to the clients. The NDOS sends a message to a server, waits for a message to be received, and assumes the message received is the response to the message sent. Considerable complexity would be needed to allow for messages which are not the response to be handled.

The SNIOS for WIZnet devices watches all sockets when receiving, since the “RCVMSG” API provides no context for receive to know what response is expected, or from what node. It assumes that whatever socket first presents data is the message the caller wants.

Only CP/NET message format “0” is used/allowed. This format consists of a 5-byte header followed by the payload. The header contains:

FMT – 00 request, 01 response.
DID – destination node ID.
SID – source (sender) node ID.
FNC – BDOS function code *.

SIZ – payload size -1.
* Additional functions may be defined, provided servers recognize them.

Note that the minimum message payload size is 1 byte, and the maximum is 256 bytes. This means the maximum message packet size will be 261 bytes. The minimum send/receive buffer size is 1K, so it is never possible for a single message to overrun the buffer.

The SNIOS has no knowledge of the receive buffer size provided to RCVMSG. It is the caller's responsibility to ensure that adequate space exists for the message received. The NDOS always uses a 261 byte buffer, however the SNIOS receives based on the amount of data available (TODO: fix this?) and so overrun is conceivable.