## Security Notice

Note that CP/NET sends all data unencrypted. It is not practical to try to do modern encryption on a Z80, even with extra memory and higher clock speeds. Be aware, when exchanging data over public networks, that is is POSSIBLE for someone to spy on the exchange. One possible workaround is to generate/copy the file(s) to a local CP/NET server, then use modern techniques to transfer the file to over public networks.

## Hardware

The interface to the WIZ850io is SPI. The interface card provides an efficient interface from the H8/H89 (8-bit CPU) to SPI. It may also be used to communicate with other SPI devices such as a Raspberry Pi microcontroller (or NVRAM), however that is beyond the scope of this document. The parallel-to-SPI circuit runs at the CPU clock rate, and shifts all 8 bits after the input/output strobe occurs. For the 8080/Z80, the minimum number of cycles for an I/O instruction is 11 (basic 8080 IN/OUT) and so back-to-back I/O instructions will not overrun. Z80 block I/O instructions take more time and also cannot overrun. This eliminates the need to poll the hardware between I/O. There is one anomaly for the case of performing input, because the input data is shifted *after* the input instruction, and so the first input instruction of a sequence retrieves "dummy" data and it should be discarded. It also means that N+1 input instructions, total, are required to input N bytes.

The SPI interface uses 2 I/O ports. Base+0 is the data port. Base+1 is the control/status. The only control bit is for the SCS signal (also known as SS, "slave select"). The bit must be asserted prior to starting an SPI sequence, and must be promptly deasserted at the end of the sequence. The only status bit is the RDY signal, but the WIZ850io does not generate a RDY so it is tied high. On the older WIZ550io module, this signal indicates when the MAC address programming has completed after power on/RESET. It is generally not a concern for CP/M software as by the time CP/M can boot, the RDY signal would have long since been asserted. The WIZ850io datasheet specifies a 50mS delay after power on/RESET for "Internal Auto Configuration Time" (which sounds like MAC address programming but the schematic shows no PIC to perform that task – in contrast to the WIZ550io). This delay is not likely to be an issue except maybe for Network Auto-Boot in the Monitor ROM. Both RDY and SCS for WIZ850io are bit 0 of their respective input/output ports. "SPI-1" (used for NVRAM) are bit 1 of the input/output ports.

## CP/NET Background

CP/NET consists of two logical OS components: NDOS and SNIOS. The NDOS is similar in purpose to the BDOS in that it services "system calls". It differs in that it never accesses local resources. It consults the "network configuration table" to determine whether a given target is local or remote. Local operations are passed directly to the local BDOS. The SNIOS is similar in purpose to the BIOS. It

provides basic, hardware-specific, functions on the network. The NDOS is generic and should work on any CP/M system. The SNIOS must be customized for the network hardware.

CP/NET uses an 8-bit "node ID" to identify all participants on a network. This is roughly equivalent to the modern IP Address. The node ID must be unique inside a given CP/NET domain. Unless a node will always be in a private domain, node IDs should be registered to avoid conflict with other CP/NET users across the internet. See the section CP/NET Node IDs for more information.

Since the WIZ850io 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 and other clients. 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 8 in the WIZ850io). The Source Port (PORT) field uses a convention from which the remote 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 any other assigned node IDs on the network. While the WIZ850io and SNIOS only use the server ID to locate the IP/port of the remote server, remote servers typically validate that this node ID matches their configuration.

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 or power off. Upon RESET (soft or hard), let alone power off, sockets are not closed. Presumably, some mechanism on servers (such as TCP keep-alive) will ensure that orphaned connections are eventually discovered and cleaned up. Re-opening a connection to the server will also trigger cleanup of the old socket connection.

## Configuration

The utility WIZCFG.COM may be used to program essential configuration values for use in the WIZ850io. Note that all memory in these devices is volatile, even across RESET, and so the values must be reprogrammed each time the system is booted. To simplify this, an NVRAM chip (also attached to the SPI interface) is used to (semi-)permanently store the network configuration, and this configuration is programmed into the WIZ850io automatically when starting NDOS3. Once the

network configuration is set in the NVRAM, it does not need to be set again unless changes are made or the NVRAM chip is erased or replaced.

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 [keep]
```

- The commands '0', '1', '2', ... '7' 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. Note that programming the MAC address in the WIZ550io is not required, however the WIZ850io will require that.
- Port numbers are in decimal (unsigned 16-bit).
- 'keep' is the optional keep-alive time interval, in seconds. WIZ850io maximum is 1275.

Running WIZCFG with no command will show the current configuration stored in the NVRAM. Prefixing any command with a 'W' will make the command operate on the WIZ850io instead of NVRAM. This includes the "show" command, e.g. "wizcfg w" shows the current WIZ850io chip configuration. Note that socket DIPR and DPORT registers are actually split internally, such that reading these registers will only show meaningful data after the socket has been connected. The values that were written are actually stored, and may be reused to open connections, until RESET.

It is convenient to maintain your network configuration in a SUBMIT file containing the WIZCFG commands to setup the network. Here is a simple example (NETCFG.SUB):

```
wizcfg n F0
wizcfg i 192.168.1.200
wizcfg g 192.168.1.1
wizcfg s 255.255.255.0
wizcfg m 02:00:5D:0D:F1:2E
wizcfg 0 00 192.168.1.17 31100
```

This sets the CP/NET client node ID to "F0", the client IP address to "192.168.1.200", the gateway address to "192.168.1.1", the network mask to "255.255.255.0", and the MAC address to "02:00:5D:0D:F1:2E". It also configures one server, node ID "00", to be found at IP address "192.168.1.17" port "31100". Keeping this file up to date makes it easy to store your configuration in the NVRAM, should that be necessary after initial setup.

Note that the list of (up to 8) remote servers is not managed dynamically/automatically. This means that a maximum of 8 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 servers may represent a very large set of resources. Attempts to contact a server that is not currently configured will result in an error from the SNIOS, however the NDOS may silently ignore the error. For example, the command "DIR P:" on a server that is not reachable results in "No File".

Also note that the WIZ850io does not automatically program a MAC address into the W5500 (the network chip). This means it is necessary to assign and set a MAC address before using the device. In most cases, a locally administered MAC address will suffice, which need only be unique on your local, private, network. Like the static IP address, the MAC address should not change unless absolutely necessary, and the two should be paired. A thorough discussion of MAC addresses is beyond the scope of this document. If your local network is already using a scheme to deploy locally administered MAC address, you should use that same scheme to generate a new, non-conflicting, MAC address for the WIZ850io. Other schemes involve generating a random number or using the Unix time value (e.g. "date +%s") and ensuring that the locally administered bit is "1" and the unicast bit is "0" (0bxxxxxx10 for the first octet). A globally administered MAC address can be obtained by recycling the MAC address of a retired or discarded piece of network equipment (e.g. an old NIC, or a cheap one that will not be used). Again, ideally this MAC address would not change for the life of the WIZ850io.

When WIZCFG is used to set a socket configuration in the WIZ850io, any active connection on that socket is closed prior to changing the values. CP/NET will re-connect (using new values) on the next access through that socket.

Note that some of the WIZ850io socket registers cannot be read exactly as written (separate registers exist for read vs. write), and so showing the WIZ850io configuration directly from the chip might not look correct.

The command "wizcfg r" may be used to program the WIZ850io from the NVRAM configuration, and is the same operation that NDOS3 performs when starting up.

## Notes on TCP Keep-alive

There are some conditions under which idle connections will be automatically terminated. One observed situation involved connections through certain routers/ISPs over the internet. Connections that get dropped may benefit from setting a none-zero keep-alive. Typically, something on the order of 45-75 seconds will suffice.

Note that a zero keep-alive value in NVRAM means "do not change the WIZ850io". This can be useful in cases where the WIZ850io KPALVTR register is being manually set before starting CP/NET.

A zero value in the WIZ850io KPALVTR register means "disable keep-alive" - no keep-alive packets will be sent. This is the default value for all sockets after power-on or RESET.

## 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 response 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.

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 WIZ850io minimum send/receive buffer size is 1K, so it is never possible for a single message to overrun the buffer.

Other formats may be defined. CP/NET documentation states that formats 0-127 are reserved for use by DRI. While it seems unlikely DRI will be defining any new message formats, formats 128-255 will be used for supporting additional features. It is recommended that HDOS use formats 0x80 (request) and 0x81 (response), although the 256-byte record size of HDOS complicates things (may require the SIZ field to be more than 8 bits). HDOS support is yet To Be Defined. Network Boot will use formats 0xb0 (boot request) and 0xb1 (boot responses), with the function code defining specific actions related to booting. Boot formats use single-byte (5-byte) header fields. Network boot protocol is defined in that section.

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 to be 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 for other callers.

## CP/NET Node IDs

Some conventions will be proposed here to help organize the "public" CP/NET domain over the internet. It is recommended that these conventions be followed in all cases, if possible, since a node (client or server) may want to participate in the public network at some time, and if it's node ID is not unique then it will conflict.

1. Node ID 0xFF is designated by CP/NET as the broadcast destination. It is also used internally for an invalid/unconfigured value. It cannot be used as an active node ID.
2. Node ID 0x00 is reserved for a local CP/NET server. Since this is the default server for CP/NET commands, this provides the maximum convenience for accessing the "primary" local server. This server must not be accessible on the internet (e.g. it's port must not be exposed, and should not be in the range 0x3100-0x31FF).
3. Node IDs 0xF0-0xFE are reserved for private networks and should not be used for nodes that interact (are visible) on the internet. They may be used as servers or clients.
4. Publicly accessible nodes should use TCP/IP port 0x31nn, where 'nn' is the hexadecimal node ID. This equates to decimal port numbers 12544-12799. WIZCFG on clients will use this convention for the source port numbers of sockets to servers.
5. Node IDs 0x01 upward will be registered for servers that are accessible on the internet. These servers will use (listen on) TCP/IP port 0x31nn, where 'nn' is the node ID.
6. Node IDs 0xEF downward will be registered for clients that will access servers on the internet.
7. The registry for "public" CP/NET node IDs (0x01-0xEF) will be the file
   https://github.com/durgadas311/MmsCpm3/blob/master/CPNET-registry.md

## Example: Setting up a CpnetSocketServer for WIZ850io

CpnetSocketServer uses a configuration file to specify settings. This file is plain text and contains "property = value" lines. The most-relevant properties are:

| | |
|---|---|
| cpnetserver_host | The hostname or IP address on which to listen for connections. |
| cpnetserver_port | The port number on which to listen for connections. |
| cpnetserver_root_dir | The top-level directory to export to CP/NET clients. Create subdirectories "a" through "p", for drives A: through P:. |
| cpnetserver_sid | The node ID for this server. Must be unique on the CP/NET domain. A hexadecimal number 00-FE. |

This configuration file is specified on the JAVA command line with the "conf=*file*" option. For example:

```
java -jar CpnetSocketServer.jar conf=file
```

By default, the server will log various messages to stderr. The property "cpnetserver_log=*file*" (or commandline option "log=*file*") may be used to direct that output to a file.

Here is an example configuration file:

```
# These are for CpnetSocketServer.jar
#cpnetserver_log = cpnet00.log

cpnetserver_host = 192.168.1.17
cpnetserver_port = 31100
cpnetserver_temp = P
cpnetserver_sid = 00
cpnetserver_max = 16
cpnetserver_root_dir = /home/drmiller/CpnetServer00
```

```
# Up to 16 printers, 0-f
cpnetserver_lst5 = Diablo630Stream file=lst00.5.ps

diablo630_nogui = true
diablo630_jobend = save
```

Other properties are explained below.

## Example: Setting up a CP/NET client on CP/M 3

This example assumes that you have already programmed your network configuration into the NVRAM chip.

A complete bundle, for the WIZ850io, is available at:

[http://sebhc.durgadas.com/mms89/wiz850io/cpnet3.tgz](http://sebhc.durgadas.com/mms89/wiz850io/cpnet3.tgz)

The CP/NET add-on for CP/M 3 is an RSX contained in NDOS3.COM. This RSX may be added by invoking the command "NDOS3". After starting NDOS3, CP/NET is active and available. However, no drives (or LST: device) has been redirected to network servers. Use the NETWORK.COM program to redirect local drives to remote drives on servers. For example:

```
network p:=c:
```

Will map local drive P: onto server 00 drive C: (00 is the default server node ID, if none is specified). After this, using drive P: under CP/M 3 will perform operations on server 00 drive C:. For these examples, that would map to the server directory "/home/drmiller/CpnetServer00/c".

The program NETSTAT.COM (or CPNETSTS.COM) will display the current mappings. For example:

```
A>netstat

CP/NET Status
=============
Requester ID = 02H
Network Status Byte = 10H
Device status:
  Drive P: = Drive C: on Network Server ID = 00H
A>
```

Note that CpnetSocketServer forces all filenames to be lowercase in the actual subdirectory. It also requires files to conform to the 8+3 convention used by CP/M. If you copy files into these subdirectories, for use by CP/M, they must adhere to these conventions. Also note that text files may require line-ending conversion in order to function properly on CP/M. CP/M requires text file line-endings to be CR-LF. There are several utilities available to do this conversion, depending on your platform. Linux usually has a package "dos2unix" that provides two utilities, for converting to and from CR-LF line-endings. There is also a simple file-copy CP/M program TR.COM that may be used to convert line-endings.

A drive may be unmapped (returned to local use) with the program LOCAL.COM. This is most useful if a mapping actually obscured a local drive, which is now needed. Typically, an unused drive letter would be mapped to a network drive.

CP/NET may be "shutdown" with the program RSXRM.COM, which causes the RSX to remove itself. For example:

```
rsxrm ndos3
```

Note that the WIZ850io does not lose it's configuration when NDOS3 is shutdown, only when the system is RESET. The command "wizcfg" (with no options) will show the current configuration of the WIZ850io (as it pertains to CP/NET).

The CP/M 3 HELP facility has been expanded to cover these commands (except WIZCGF).

## Example: Setting up CP/NET 1.2 on CP/M 2.2

This example assumes that you have already programmed your network configuration into the NVRAM chip.

CP/NET 1.2 (for CP/M 2.2) may also be run. Note that this will consume more TPA than the CP/M 3 version, and requires a cold boot (system RESET) to remove.

The standard CP/NET 1.2 distribution files are used, in combination with "snios-w.spr" (which must be renamed to "SNIOS.SPR") and the WIZCFG.COM and associated utilities and files. CP/NET 1.2 distribution files are available here (as well as other sources on the internet):

> https://github.com/durgadas311/MmsCpm3/tree/master/net/dist/bin

A complete bundle, for the WIZ850io, is available at:

> http://sebhc.durgadas.com/mms89/wiz850io/cpnet12.tgz

Copy all these files onto a bootable CP/M 2.2 disk (with standard CP/M utilities). Boot this disk and ensure the NVRAM contains the WIZ850io config. Then start CP/NET with the program CPNETLDR.COM. You will get output like this:

```
A>cpnetldr

CP/NET 1.2 Loader
=================

BIOS         EB00H  1500H
BDOS         DD00H  0E00H
SNIOS   SPR  D900H  0400H
NDOS    SPR  CD00H  0C00H
TPA          0000H  CD00H
```

```
    CP/NET 1.2 loading complete.

    A>
```

At this point, CP/NET is active and you may use NETWORK.COM to map local drives to network drives. Note that CP/NET 1.2 uses a different CCP than standard CP/M, it is in the file CCP.SPR so that it can be relocated as needed. Consult the DRI CP/NET 1.2 documentation for more information on added features of the CP/NET CCP.

## NVRAM Option

The SPI adapter contains a second SPI port, which may be connected to a Microchip 25LC512 non-volatile memory (SEEPROM) device. This is used to store the WIZ850io configuration over RESET and power cycles. Only the first 512 bytes of the 64K NVRAM are used by WIZCFG, so the rest may be used as desired for other purposes.

The WIZ850io configuration in NVRAM must be created using WIZCFG and/or SUBMIT. This configuration contains a checksum, which is used to detect when a valid configuration exists. The WIZ850io will not be configured from NVRAM unless this checksum is good.

Once a configuration has been saved, each time the system is RESET or powered on the configuration may be manually restored to the WIZ850io using the command:

```
    wizcfg r
```

NDOS3 and CPNETLDR automatically perform a "wizcfg r", and so running this command is not necessary when using CP/NET.

The utility NVRAM.COM may be used to directly examine (and change) the contents of the NVRAM. Note that the WIZ850io configuration block is protected by a checksum, so altering that area is likely to result in loss of the saved configuration. This utility is discussed under Advanced Operation, below.

## Network Boot

CP/NET does not define how to boot a system over the network. The proposed method is to use the same header format as for CP/NET (256-byte max message payload), and use FMT codes 0xb0 and 0xb1. The following function codes are defined:

| Requests | | | | |
|---|---|---|---|---|
| FMT | FNC | SIZ | Data | Purpose |
| B0 | 01 | nn | string | Request boot using image(s) for SID, with optional string. ("no string" designated by a single NUL character) |
| B0 | 00 | 00 | 00 | ACK previous response, send next (handshake/flow control). |
| Responses | | | | |
| B1 | 00 | 00 | 00 | NAK - Boot failed/refused. End of transmission. |

| B1 | 01 | 7F | message | ACK - Boot accepted, image responses follow. *message* is '$'-terminated string to be displayed to user. |
|----|----|----|---------|----------------------------------------------------------------------------------------------------|
| B1 | 02 | 01 | address | Set DMA address for subsequent records. |
| B1 | 03 | 7F | record | Load *record* into DMA address, bump address by 128. |
| B1 | 04 | 01 | address | Boot complete, jump to *address*.  End of transmission. |

The server receives the boot request and examines the SID and optional string to see if a boot image exists. If booting is not allowed/possible, a NAK response is sent. If booting is possible, an ACK message is sent, containing a signon string for the user (e.g. the load message from CPM3.SYS).

If the client receives the ACK message, it will then prepare to accept and process additional messages until FNC 04 is received. Note that FNCs 02 and 03 are general-purpose data loading messages. The client should accept these and process them. There is no way to prematurely stop the boot, so if the client decides to abort it must continue the protocol but ignore all messages until FNC 04.

Once the client has completed processing of each FNC 01, 02, or 03 message, it must send a ACK request to tell the server to send the next message. FNC 04 messages are not ACKed, no more messages follow. If the client sends an ACK request when booting is not active on the server (before FNC 01 or after FNC 04), it will receive a NAK response.

The format of boot images on the server is not defined, however current plan is for CpnetSocketServer to use files in the format of CPM3.SYS. This will allow for up to two separate memory segments for the OS image (RES and BNK for CP/M 3, but segment may have any meaning). Generating image files, other than for CPM3.SYS, is yet to be developed. CP/NOS images are fairly simple (CP/NET 1.2 diskless clients), however CP/M 3 images that include CP/NET are more of a problem. CP/M 3 normal booting into CP/NET requires loading of CCP.COM and executing the NDOS3.COM (an RSX). There is currently no solution for getting CCP.COM and NDOS3.COM (with pre-configured network drive(s)) to the CP/M 3 OS in cold boot. Note, the WIZ850io will have been configured by the ROM in order to boot, so that step is not required for CP/M 3. However, in order to operate independent of local disks, (at least) drive A: must be mapped to a network server in the table built into NDOS3.COM. It is currently expected that the boot image will have a pre-configured network table built into the SNIOS. It may be possible to create a SETUP utility to do this, so that is is not required to rebuild NDOS3.COM whenever changes are made. It may also be desirable for the server to overlay the boot image with a custom network table at boot time.

## Advanced Operation

The program WIZDBG.COM may be used to directly examine, and alter, any register memory in the WIZ850io. It operates in arbitrary ranges of bytes, within a specified block. Consult the W5500 datasheet for specifics on register layout and operation. Basic usage is obtained by invoking the command with no options:

```
Usage: WIZDBG {G bsb off num}
       WIZDBG {S bsb off dat...}
       bsb = Block Select Bits, hex 00..1F
       off = Offset within BSB, hex
```

```
              num = Number of bytes to GET, dec
              dat = Byte(s) to SET, hex
```

Use the option 'G' to get, or examine, a range of bytes. 'S' will set, or alter, a range of bytes. 'bsb' is the Block Select Bits, taken as a 5-bit hexadecimal value. Note that not all possible 5-bit values are valid, and may cause malfunction if used. For example, to dump the first 32 bytes of the Common Register area:

```
    A>wizdbg g 0 0 32
    00:0000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00:0010 00 00 00 00 00 00 00 00 00 07 D0 08 28 00 00 00
    A>
```

In this example, the chip has not been configured yet so only the Retry Time, Retry Count, and PPP LCP Request Timer fields have non-zero values.

Note that the CP/M commandline input buffer is 128 bytes (127 characters, maximum) in length, so the SET option is limited to how many values can be entered into the command buffer. The GET option is limited to buffer space (TPA), but reading beyond the physical end of registers blocks is not recommended.

The following table shows the bsb values for each socket, in addition to the Common Register block at BSB 00.

| Socket | Socket Registers BSB | Tx Buffer BSB | Rx Buffer BSB |
|--------|----------------------|---------------|---------------|
| 0 | 01 | 02 | 03 |
| 1 | 05 | 06 | 07 |
| 2 | 09 | 0A | 0B |
| 3 | 0D | 0E | 0F |
| 4 | 11 | 12 | 13 |
| 5 | 15 | 16 | 17 |
| 6 | 19 | 1A | 1B |
| 7 | 1D | 1E | 1F |

Consult the W5500 Datasheet for the meanings of the offsets within a BSB.

The SET option is currently <u>allowed</u> if CP/NET is active. Use with extreme caution.

-------
The program NVRAM.COM may be used to directly access the NVRAM device (25LC512), if it is installed. Help is printed if the command is invoked with no options:

```
    Usage: NVRAM R adr len
           NVRAM W adr val...
           NVRAM CE
```

```
NVRAM SE adr
NVRAM PE adr
```

The "R" command reads 'len' bytes (decimal) from the NVRAM starting at 'adr' (hexadecimal). Any number of bytes may read, limited only by the amount of TPA available to buffer the data (currently, the program does not prevent you from crashing the OS).

The "W" command writes the 'val...' bytes (hexadecimal) into NVRAM starting at address 'adr' (hexadecimal). Note, this program does not hide the 128-byte page limitations of the device. In other words, 'adr' and the number of values must all represent a single 128-byte. Just as described in the chip's datasheet, writing beyond the 128-byte boundary will wrap back to the start of the same 128-byte page (does not continue to the next page).

The "CE", "SE", and "PE" commands erase all, or part, of the chip. See chip documentation. Note that sending an erase command before WRITE is not required, as the chip does that for you.

Just like "wizdbg s ...", the WRITE command is limited by the space available in the CP/M commandline buffer.

See also the Microchip 25LC512 datasheet.

## HDOS/NET

Some preliminary work has been done on a networking version of HDOS (citation). If that were to be implemented using a compatible message structure, then a single server could handle both types of clients. A recommendation is to use CP/NET user-defined message FMT 0x80 (requests) and 0x81 (responses), and set FNC to the HDOS syscall number. As is the case with CP/M, not all HDOS syscalls are applicable over the network. And some would be broken up into multiple messages for affected servers (and a possible local syscall). One TBD is how to handle HDOS' 256-byte record size when that maximum payload of the normal CP/NET message format is 256 bytes. Some possible solutions are to define the SIZ field as 16-bits for HDOS, with a size restriction, or split messages that contain records into request/response and data messages.