uNetEthernet Chip

Technical Specification

Version Number:	0.2
Date:	**February 14, 2006**
Contact:	www.nchip.com
Document Number:	001-0004

Preliminary – Subject to change

nChip
www.nchip.com

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 GENERAL INFORMATION

1.1 SCOPE

This document contains an overview of the uNetEthernet Module. It defines the architecture of the Module, block diagrams, theory of operation, I/O's, timing information, features, command set and examples.

1.2 DEFINITIONS

TBD	To Be Defined

1.3 APPLICABLE DOCUMENTS

"**Document**" – Description.

1.4 REVISION HISTORY

Rev.	Date	Comments
0.1	11/03/03	Initial Specification

2 uNetEthernet Overview

2.1 Overview

The uNetEthernet module is an inexpensive, updateable TCP/IP coprocessor that connects between a host processor or microcontroller and the Internet. In later code releases the uNetEthernet module will have to ability to operate without a microcontroller allowing user programmability. The uNetEthernet module can connects to a network or the Internet via an Ethernet connection. The uNetEthernet module has been tested with many brands of Ethernet switches and hubs for reliable and compatible connectivity. The uNetEthernet module supports DHCP or static configuration with full name resolution support. Its fully configurable TCP/IP stack and other configuration parameters are stored in the uNetEthernet's FLASH and EEPROM memory and all parameters and code are upgradeable.

The first firmware release for the uNetEthernet module is based on nChip's IR command processor. Though other firmware build are planned to give the uNetEthernet module multiple personalities.

nChip's IR command processor connects to a device's processor via a serial connection and uNetSerial's powerful yet simple IR command Interface which enables Internet Protocols like DNS, ICMP, UDP, TCP, SNMP, POP and HTTP with 'Simple Text Strings'. These IR commands offer powerful Internet functionality including the innovative Streaming Socket Technology that allows multiple TCP connections to be effectively managed over a single serial port. The IR command Interface is flexible and efficient, does not limit usage, almost any Internet protocol can be implemented. It is as simple as writing a few simple commands and letting uNetEthernet and its IR command processor take care of all the complex Internet protocols.

uNetEthernet and its related products are one of the simplest ways to get a device talking on the Internet. Please go to the website www.nchip.com for up to date info on innovative Internet products.

2.2 Supported Feature List

The following features are supported by uNetEthernet

1. ICMP echo request and echo reply.
2. Automatic DNS name resolution.
3. Fully configurable (TCP/IP parameters and timeouts, Port Speed and much more) parameters that can be saved in non-volatile memory.
4. UDP Send and Receive.
5. Two TCP streaming socket connections.
6. Generic I/O pin control.
7. Two full flow-controlled serial ports with auto baud rate detection.
8. FLASH upgrade-able firmware.
9. Minimal external components required. (3.3 volt regulator, Ethernet magnetics and a few passive parts.)
10. Interfaces with any standard Ethernet Switch or Hub.

2.3 Non-Supported Feature List

The following features are ***not*** currently supported by uNetEthernet but are under development for a later version.

1. Webserver Support
2. ADC support.
3. Programmable Scripting Language.
4. Multicast-DNS - Bonjour

2.4 uNetEthernet System Block Diagrams

2.3.1 uNetEthernet Block Diagram

The following block diagram depicts uNetEthernet's main features.

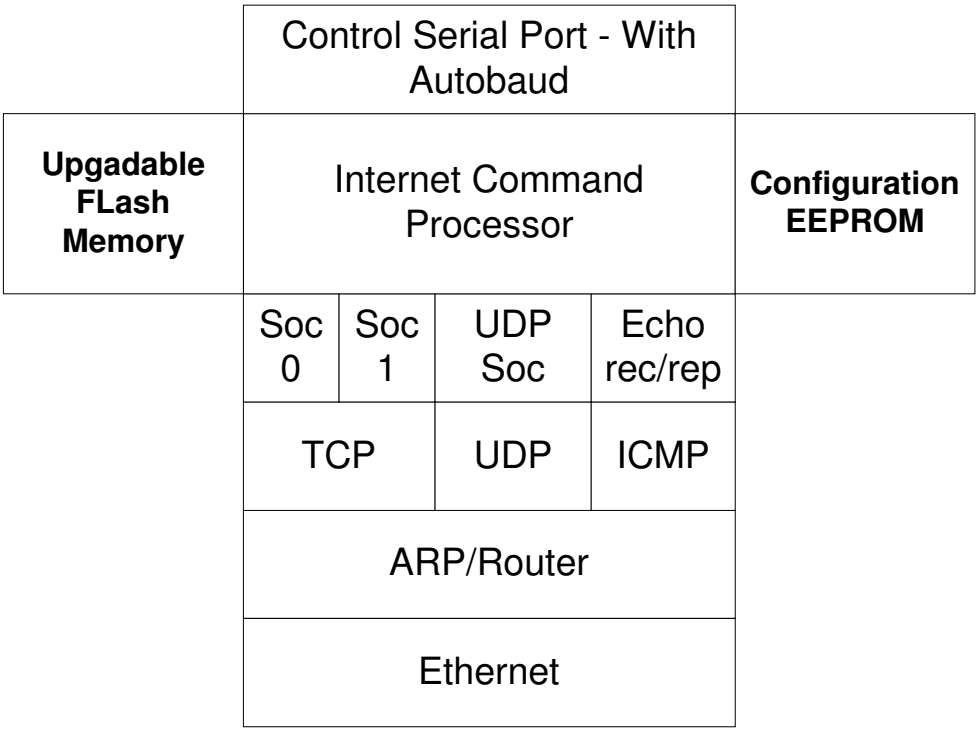


Figure 2-1: uNetEthernet Block Diagram

2.3.2 Typical Usage Diagram

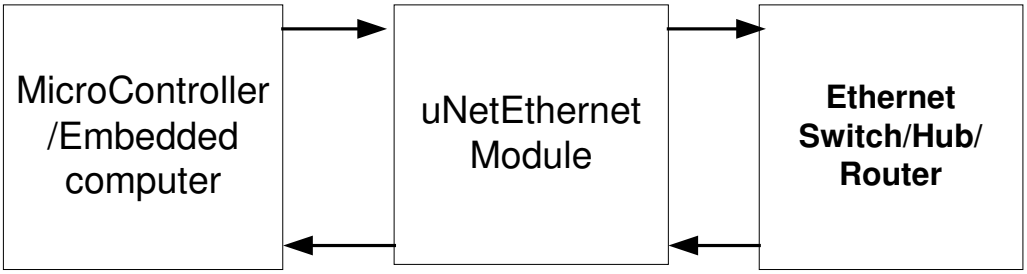


Figure 2-2: uNetEthernet Usage Diagram

3 uNetEthernet Pin Configuration

The uNetEthernet comes in the form of a solder or plug in module.

3.1 uNetEthernet Module Pin Configuration

Figure 4 shows the uNetEthernet module pin configuration. The uNetEthernet module is a PCB module that contains a uNetEthernet Chip, Ethernet controller, reset and power filter circuits and oscillators. The module pins out into standard 2mm headers. The uNetEthernet Module package Diagram is shown in Figure 5.

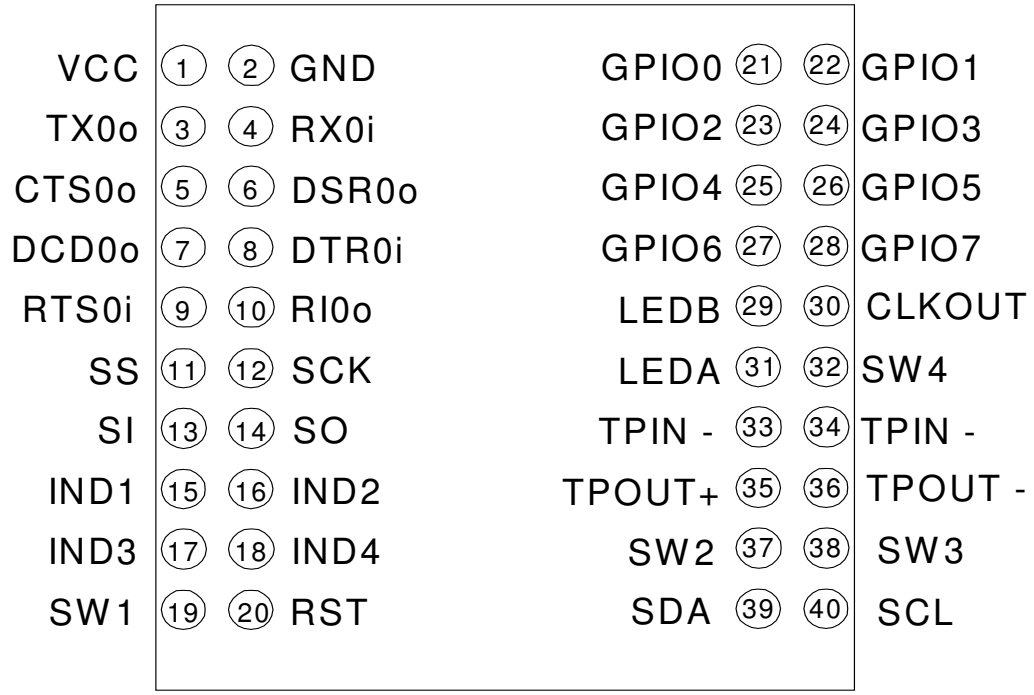


Figure 3-1: uNetEthernet Pinout Diagram

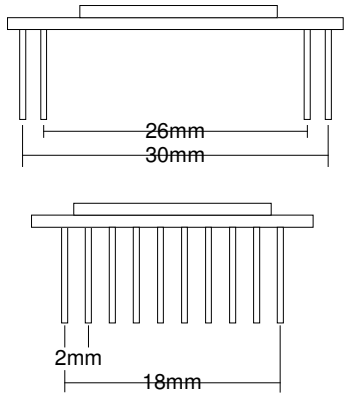


Figure 3-2: uNetEthernet Module Package Diagram

3.2 uNetEthernet Module I/O List

Module Pin	Signal	B/I/O	Description
1	VCC	I	Supply Voltage (3.3 volts)
2	GND	I	System Ground
3	TX0o	O	Data Out – Serial Port 0
4	RX0i	I	Data In – Serial Port 0
5	CTS0o	O	Clear To Send – Serial Port 0
6	DSR0o	O	Data Set Ready – Serial Port 0
7	DCDo	O	Carrier Detect – Serial Port 0
8	DTR0I	I	Data Terminal Ready – Serial Port 0
9	RTS0I	I	Request To Send – Serial Port 0
10	RI0o	O	RI incoming connection –or- Data Available (DAV)
11	SS	I	Reserved – Do Not Connect – for future SPI interface
12	SCK	I	Reserved – Do Not Connect – for future SPI interface
13	SI	I	Reserved – Do Not Connect – for future SPI interface
14	SO	O	Reserved – Do Not Connect – for future SPI interface
15	IND1	O	Indicator 1
16	IND2	O	Indicator 2
17	IND3	O	Indicator 3
18	IND4	O	Indicator 4
19	SW1	I	Switch/interrupt 1
20	RST	I	Chip Reset, a low level pulse will cause a chip reset.
21-28	GPIO	B	General Purpose I/O
29	LEDB	O	LED B output
30	CLKOUT	O	Programmable Clock Output
31	LEDA	I	LED A output
32	SW4	I	Switch4/Force Bootloader
33	TPIN-	I	Differential Ethernet Signal Input
34	TPIN+	I	Differential Ethernet Signal Input
35	TPOUT+	O	Differential Ethernet Signal Output
36	TPOUT-	O	Differential Ethernet Signal Output
37	SW2	I	Switch/interrupt 2
38	SW3	I	Switch/interrupt 3
39	SDA	B	TWI data line
40	SCL	O	TWI clock line (output)

Table 3-1: uNetEthernet I/O List

4 EMULATION PLATFORM

An Emulation Platform is available via a Microsoft Windows program (Windows 2K or Windows XP) that emulates much of the uNetEthernet firmware. This platform is a great evaluation or debugging platform. It provides extra debugging information and is very valuable for debugging network connection issues and developing communications scenarios for use later on an embedded platform This platform uses most of the same code as the module firmware releases and should behave much the same as the corresponding release firmware.

Typically the emulation program will be named 'uneteth.exe' though in certain packages it could be named something else. This program is should be run from the command line. 'uneteth.exe' can also use a configuration file if specified (Example 'uneteth unetconfig.txt') If a configuation file is not specified DHCP is used to configure the emulator. To exit the emulator at any time, press the <ctrl>-c key pair.

The emulator will also write a log file named unetelog.txt. To get valid output into the log file, make sure you exit the emulator using the <ctrl-c> command sequence while in the emulator, this will make sure the full log file contents are flushed to the log file. This log file can be used to easily submit bugs or develop and record command sequences.

4.1 EMULATOR REQUIREMENTS

The emulation platform requires Windows 2K or Windows Xp. The emulator also requires that WinPcap is installed on the target machine before running the uNetEthernet emulator. WinPcap is used to emulate the Ethernet chip on the uNetEthernet module, and with it the emulator can send and receive packets exactly as the module does without interfering with the host machines network stack. It also allows the host machines network stack to communicate with the uNetEthernet Emulator. Currently only WinPcap version 3.1 has been tested, although other versions may work.

WinPcap is free to download and use. It is available at www.winpcap.org.

4.2 EMULATOR CONFIGURATION

If no configuration file is specified DHCP will be used to configure the emulator. There is no emulation of the non-volital memory of the uNetEthernet Module, so any non-default S-Register settings need to be inputted by hand if necessary.

The configuration file understands 5 commands; each command must be entered on a separate line. A pound sign (#) at the beginning of the line tells the emulator to ignore the line. A sample configuration file follows:

```
# unete config file
#
# dhcp, iface, ip, mask, gateway, pridns, secdns, dhcp
#
dhcp 0
iface 0
ip 10.10.11.9
mask 255.255.255.0
gateway 10.10.11.1
pridns 10.10.11.2
secdns 64.40.160.10
```

Table 4-1: Example Configuration File

In this example the first four lines are ignored as they start with the pound sign. The next line is the dhcp config, it must be first in the configuration file, 0 for no DHCP or 1 for use DHCP. If DHCP is set to 1 the ip, mask, gateway and dns servers are ignored. The iface command if set to non-zero will automatically set that PCAP interface, set to zero first to find the correct PCAP interface. The next line configures the emulators IP address as 10.10.11.9, followed by the next line which configures the emulators netmask to 255.255.255.0. The default gateway is specified next followed by the primary and secondary name servers to use.

5 INTERNET COMMAND PROCESSOR

5.1 OVERVIEW

This section describes the usage of the IR Internet Command Processor. The Internet Command Processor provides Internet functionality for serial connected devices. . It can be uses in conjunction with dialup and wireless phones and modems, to enable Internet and modem functionality over a simple serial port. This section describes an IR command processor command set to provide flexible Internet functionality. The IR command processor also allows complete configuration of an nChip device of which all Internet parameters, serial port configurations can be configured for any application.

5.1 IR COMMAND PROCESSOR USAGE

5.1.1 Command-Line

The IR command interface provides specific Internet functionality. The IR commands follow similar guidelines as the standard modem AT commands.

- When using IR commands, start every command line (except the +++ escape command) with the IR code characters.
- The commands following the IR prefix can be uppercase or lowercase or a combination of both.
- Commands must be terminated with a carriage-return character. This is ASCII 13 (0x0D).

Each command line to the IR command processor follows the format below:

<IR><Command>{Argument}

Note: Information above in "angle" brackets <> must be included as part of the command line, while information in "curly" braces { } may or may not be necessary as part of the command line.

<IR> – This is the attention code.
<Command> – A command consists of one letter.
{Argument} – Optional information specific to the Command.
{=n} – Used in some instances to qualify an Argument specific to the Command.

5.1.2 Operating Modes

The operation modes of the uNetEthernet is much simpler than the uNetSerial module. It consists of only two modes, a command mode and a streaming socket mode. In command mode IR commands can be issued, in streaming socket mode the serial port is connected directly to a TCP socket and IR commands are not recognized and are treated as data.

Transition from streaming socket mode to command mode by "escaping" the connection. This is done by sending an inband escape character sequence "+++" or by the outband escape sequence by toggling the DTR pin. The inband escape sequence needs to be prefaced and followed by a guard time, or the escape character sequence will be interpreted as data. Both the guard time and the escape sequence character may be configured by S-Registers.

The IR command processor for uNetEthernet mode state diagram is shown below.

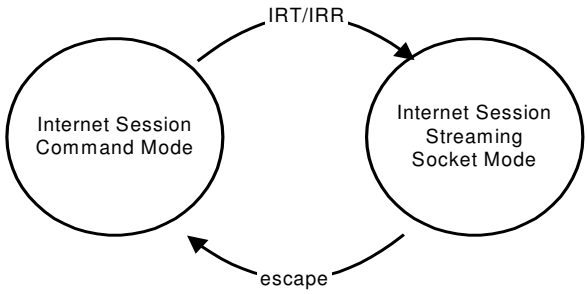


Figure 5-1: uNetEthernet State Diagram

When in the Internet Session Mode (command mode) only the IR commands are interpreted by the uNetEthernet. In the Internet Session modes the IM receives characters from the serial port, converts the data into Internet protocol packets, then converts them to Ethernet packets and transmits these signals over the Ethernet wire.

The uNetEthernet transitions from the Internet Session Command mode to the Internet Session Steaming Socket mode after a successful IRT command. To transition from the Internet Session Streaming Socket mode back to the Internet Session Command mode and escape sequence of characters must be sent to the uNetEthernet of from the socket closing from the peer.

5.1.3 Streaming Sockets

Streaming Sockets are a simple yet advanced way to manage multiple TCP connections over a serial connection.

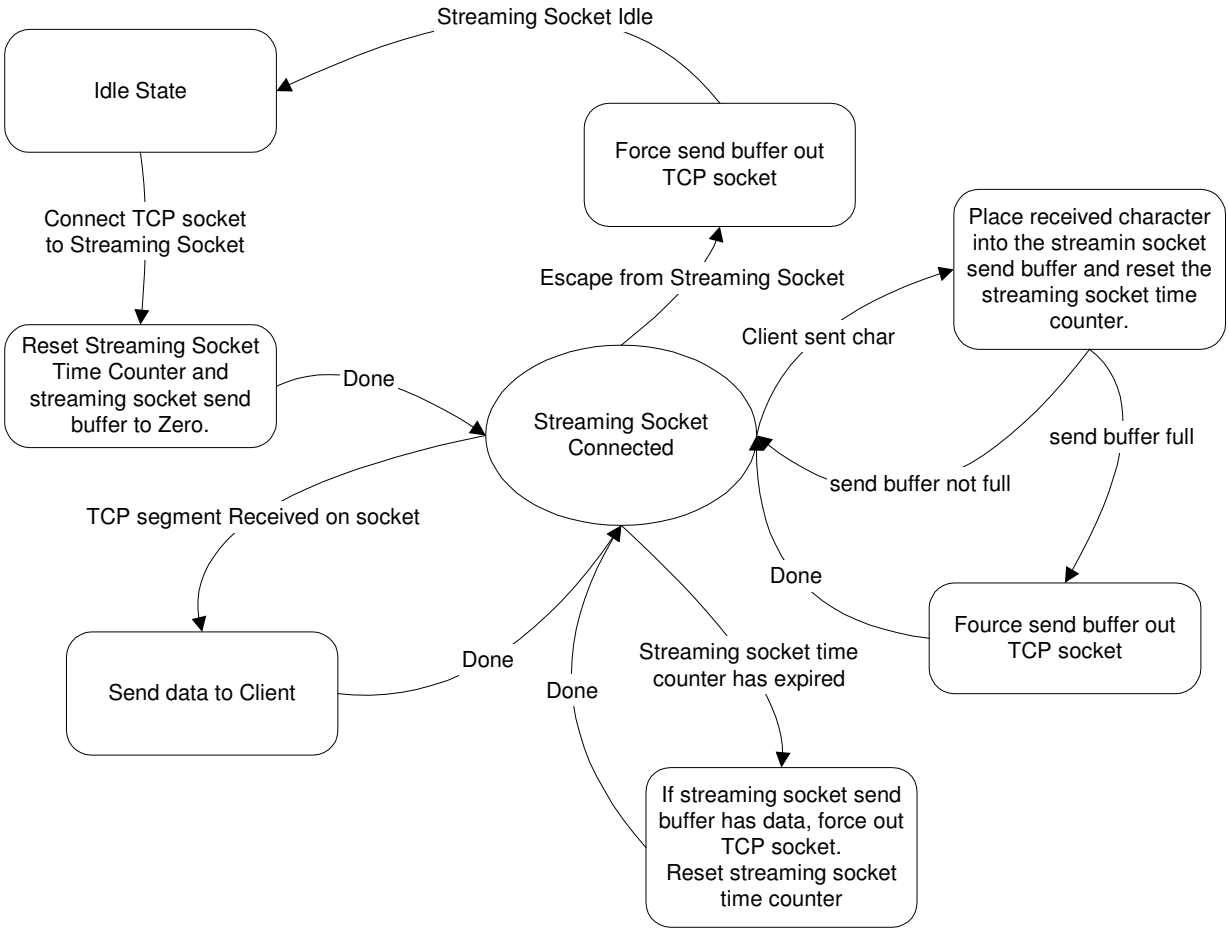


Figure 5-2: Streaming Socket State Machine

5.2 IR COMMANDS

This communications protocol is intended to be similar in execution to a standard modem AT commands. All commands are prefixed with the characters IR. Sending IR alone will always return OK or 0 if the IR command processor is ready to receive commands. All commands return a result code indicating the result of the command.

5.2.1 Ethernet Commands (uNetEthernet Only)

5.2.1.1 IRD

Usage:
IRD

This command will try to refresh the network configuration via DHCP.

Returns:

OK[CR/LF]	- if network information was refreshed via DHCP
FAIL[CR/LF]	- if DHCP failed.

5.2.2 DNS COMMANDS

This section describes a set of commands that resolves hostnames into IP addresses via the Domain Name System protocol.

5.2.2.1 IRN

Usage:
IRN <name>

This command will try to resolve a name into an IP address using the Domain Name System protocol.

Returns:

IP address[CR/LF]	- if name was correctly resolved
FAIL DNS[CR/LF]	- if DNS resolution has failed.

5.2.3 ICMP COMMANDS

This section describes a set of commands that generate ICMP packets.

5.2.3.1 IRP

Usage:
IRP <hostname/IP>

Sends a ping (ICMP Echo Request) and waits for a reply.

Returns:

OK [xxx]ms[CR/LF]	- if an ping response was returned.
FAIL[CR/LF]	- if there was no response.

5.2.4 UDP COMMANDS

This section describes a set of commands that generate and receive UDP packets. Currently only one UDP port can be enabled for read.

5.2.4.1 IRUB

Binds a UDP port to UPD for receive. (enables UDP receive on a port)

Usage:
IRUB <port>

Returns:
ERROR[CR/LF] - passed command is invalid format.
OK[CR/LF] - source port has been successfully set for receive.

5.2.4.2 IRUP

Sends a UDP packet to IP:destport from ourip:sourceport. All parameters are optional. If no parameters are set the packet will be sent to the Ipaddress and Port of the last received UDP packet (or zeros if there were none). Source port, if not specified will be zero or the port set by the last IRU command.

On successful input of the IRUP command the IR command processor will return OK. Input will then be queued to send in the packet until there is no input for the UDP_STREAM_TICK timeout (see S-Register 0xe) or the maximum number of allowable characters have been inputted. When the packet has been sent the IR command processor returns OK.

Usage:
IRUP [ip/name]:[destport]:[sourceport]

Returns:
DNS FAIL[CR/LF] - if name could not be resolved to IP
ERROR[CR/LF] - passed command is invalid format
OK[CR/LF] - ready to accept packet to send, then again when packet has been sent.

5.2.4.3 IRUG

Get a UDP packet that was received on the port set by the last IRUB command. By default the IRG command will return 'NO DATA' if there are no UDP packets waiting, or 3 lines followed by the packet data.

Usage:
IRUG

Returns:

NO DATA	- if there is no data waiting on the UDP socket
-or-	
IP address of peer that sent packet[CR/LF]	- followed by the following if there is a UDP packet waiting.
Port of peer that send the packet[CR/LF]	
Length of packet[CR/LF]	
[Packet data]	

5.2.4.4 IRUV

Stops the device from listing on the previously specified port (IRU)

Usage:
IRUV

Returns:

OK[CR/LF]	- Always

5.2.5 TCP COMMANDS

This section describes a set of commands that generate and receive TCP packets. Currently two TCP sockets (connections) can be in use at the same time.

5.2.5.1 IRT

Try's to connect a TCP socket to a remote host at a specified port. If command returns OK the socket is connected in streaming socket mode and any data sent or received on the socket will be sent to/from console serial port. You may go back to command mode by sending +++ or toggling the DTR pin. Going back to command mode will not close the socket and socket can be reconnected or dropped by the commands IRR and IRX respectively.

Usage:
IRT<socket><name/ip>:<destport>

Returns:

FAIL DNS[CR/LF]	- if name could not be resolved to IP
FAIL TCP[CR/LF]	- Could not connect to socket.
ERROR[CR/LF]	- Error in passed parameter format.

5.2.5.2 IRR

Reconnect to a specified TCP socket from command mode.

Usage:
IRR <socket>

Returns:
OK [CR/LF] - if the socket is reconnected
FAIL SOCKET DOWN[CR/LF] - Socket is down, cannot reconnect

5.2.5.3 IRX

Closes a specified TCP socket. Always returns OK

Usage:
IRX <socket>

Returns:
OK [CR/LF] - Always returns OK

5.2.6 MISC uNetSerial COMMANDS

This section describes commands that control aspects of the uNetSerial Chip not related to other command groups. These commands are not available on the uNetEthernet products.

5.2.6.1 IRO

Turn off the IR command processor. After issued uNetSerial will act as a serial pass through device. To turn the IR command processor back on, uNetSerial needs to be reset or the DTR pin needs to be toggled.

Usage:
IRO

Returns:
OK [CR/LF] - Always returns OK

5.2.6.2 IRM

This command toggles the modem port's DTR pin <this can force modem into command mode and drop modem connection if configured on modem>,

Usage:
IRM

Returns:
OK[CR/LF] - Always Returns OK

5.2.7 EEPROM COMMANDS

This section describes the IR commands that control the loading and saving of the configuration status into the EEPROM and other EEPROM access commands.

The baseline usage is:
IRE<I,S,L or location>[=value]

Specific usage is show below.

5.2.7.1 IRES

This command saves the current running configuration state into non-volatile EEPROM memory and marks it as a valid configuration. If there is a valid configuration state currently in EEPROM memory, it is over written.

Usage:
IRES

Returns:
OK [CR/LF] - Always returns OK

5.2.7.2 IREL

This command loads a previously saved valid configuration into the current running configuration. If the EEPROM does not contain a valid configuration the default configuration is loaded.

Usage:
IREL

Returns:
OK [CR/LF] - Always returns OK

5.2.7.3 IREI

This command invalidates the EEPROM configuration valid byte. This can be issued before an 'IREL' to force the device to its default configuration. This is the same as the IRE0=0 command.

Usage:
IREI

Returns:
OK [CR/LF] - Always returns OK

5.2.7.4 IRE<address>

This command can read or set any location in the nChip Devices EEPROM memory. If only an address is specified this command returns, in hex, the data bytes stored in the EEPROM at the specified address. If an address and an equal value are specified, the specified value is stored in the EEPROM at the specified address.

Usage:
IRE<location>[=value]

Returns:
<value>[CR/LF] -

5.2.7.5 EEPROM Memory Map

The following is the EEPROM memory map used by uNetEthernet

Address	Contents
0x0	This byte is 0xa5 if EEPROM configuration is valid.
0x1-0x9	Reserved
0xA-0x50	Non-volatile configuration storage that mirrors S-Registers 0-0x29
0x60- EEMAX	Reserved for command processor scripts

Table 5-1: EEPROM Memory Map

5.3 S-REGISTER COMMANDS

This section describes the IR S-Register command that access the current running configuration and state.

Usage:
IRS<register>[=value] - view or set S-Register

Returns:
<value> - value of S-Register

The next section will describe the S-Register map and it usage.

5.3.1 S-REGISTER Map

(subject to change, Serial Control not usable on Windows emulation yet)

The following is the S-REGISTER map used by uNetEthernet. It is broken up into 2 sections, a lower S-Register section and an Upper S-Register section. The lower S-Register section is special function S-Registers, the operation of each Lower S-Register is unique to itself, and whereas the upper S-Register section registers all operate in a consistent fashion (accessing data in hexadecimal bytes.)

Below are the Lower and Upper S-Register map tables:

Address	Register Name	Register Definition
0x0	TCP0_STATUS	TCP socket 0 status
0x1	TCP1_STATUS	TCP socket 1 status
0x2	UDP_STATUS	UDP status
0x3	FIRMWARE_VER	Firmware Version
0x4	BOOT_VER	Bootloader Version
0x5	OUR_MAC_ADDR	MAC address of device
0x6	OUR_IPADDR	IP Address
0x7	OUR_NETMASK	IP Address Mask
0x8	OUR_GATEWAY	Gateway IP address
0x9	PRI_DNS	Primary DNS server address
0xa	SEC_DNS	Secondary DNS server address
0xb	CONSOLE_BAUD	Console Baud Rate
0xc	GPIO_DIR	GPIO Direction Byte
0xd	GPIO_STATE	GPIO State Byte
0xe	DHCP_LEASE_TIME	DHCP Lease Time left in seconds

Table 5-2: S-Register Lower Registers

Address	Register Name	Register Definition
0x20	UNET_CONFIG	uNetSerial Config Byte
0x21	ESC_CHAR	Escape Char
0x22	ESC_TIMEOUT	Escape Timeout (Guard Time)
0x23	TCP_STREAM_TICK	TCPStreamTickTime
0x24	UDP_STREAM_TICK	UDPStreamTickTime
0x25	DNS_TIMEOUT	DNS Timeout
0x26	SERIAL_CONFIG	Serial_Config_Byte
0x27	CONSOLE_BAUD_HEX	ConsoleBaud similar to S-Register 0x9 but hex number.
0x28	UDP_FLAGS	UDP_FLAGS
0x29	TCP_CONNECT_TIME	tcp_connect_timeout
0x2a	TCP_RETRANS_TIME	tcp_retransmit_timeout
0x2b	IP_TTL	IP_TTL
0x2c	IP_TOS	IP_TOS
0x2d	NETWORK_CONFIG	DHCP
0x2e-0x33	OUR_MAC_ADDR	OUR_MAC_ADDR
0x34-0x37	OUR_IPADDR_HEX	OUR_IP_ADDR
0x38-0x3b	OUR_NETMASK	Our IP addresses Netmask
0x3c-0x3f	GATEWAY_IPADDR	Gateway IP address
0x3c-0x3f	PRI_DNS_HEX	Primary DNS Server
0x40-0x43	SEC_DNS_HEX	Secondary DNS Server
0x44	IRCMD_STATE	The current Mode of the IRCMD processor.

Table 5-3: S-Register Upper Registers

5.3.2 Lower S-Registers

The first eleven S-Registers are special S-Registers, their format is different than the Upper Range S-Registers. This section describes the lower S-Registers and there operation.

5.3.2.1 TCP0_STATUS

This S-Register returns the status of the TCP 0 socket. This S-Register is read only. See table 5-4 for return codes and there corresponding TCP state.

Usage:
IRS0

Returns:
AA BBBB - AA = Socket State BBBB= Source Port of socket

State	Meaning
0x0	Socket is closed
0x1	Socket is in Listen State

0x2	Socket is in SYN Sent State
0x3	Socket is in SYN Received State
0x4	Socket is in Established State
0x5	Socket is in Closed Wait State
0x6	Socket is in Last Ack State
0x7	Socket is in FIN wait 1 state
0x8	Socket is in FIN wait 2 state
0x9	Socket is in CLOSING State
0xA	Socket is in TIME_WAIT State

Table 5-4: TCP Socket State (see RFC793)

5.3.2.2 TCP1_STATUS

This S-Register returns the status of the TCP 1 socket. This S-Register is read only.

Usage:
IRS1

Returns:
AA BBBB - AA = Socket State BBBB= Source Port of socket

5.3.2.3 UDP_STATUS

This S-Register returns the status of the UDP socket. This S-Register is read only. Returns length of waiting packet (0-0xff), source IP of waiting packet and source port of waiting packet.

Usage:
IRS2

Returns:
AA B.B.B.B CC - AA = length of waiting packet (0-255) B= source IP of waiting packet CC= source port of waiting packet.

5.3.2.4 FIRMWARE_VER

This S-Register returns the uNetSerial's firmware version. This S-Register is read only.

Usage:
IRS3

Returns:
<firmware version>[CR/LF]

5.3.2.5 BOOT_VER

This S-Register returns the uNetSerial's Bootloader version. This S-Register is read only. Boot Version is returned as a 4 digit hex number, for example 0104 would be boot version 1.4.

Usage:
IRS4

Returns:
<Bootloader version>[CR/LF]

5.3.2.6 OUR_MAC_ADDR

This S-Register sets or returns the uNetEthernet's MAC Address. Setting this address to 0 sets the MAC address to the system default. The MAC address should not be changed unless operator understands the impact of this action. Setting the MAC address to all zeros or resetting the EEPROM will have the uNetSerial device attempt to restore the devices factory set MAC address.

Usage:

IRS5	- To return the MAC address.
-or-	
IRS5=xx:xx:xx:xx:xx:xx	- To set the current MAC address.

Returns:

<MAC address>[CR/LF]	- current MAC address Returned.

5.3.2.7 OUR_IPADDR

This S-Register sets or returns the uNetSerial's IP Address.

Usage:

IRS6	- To return the current IP address.
-or-	
IRS6=xxx.xxx.xxx.xxx	- To set the current IP address.

Returns:

<IP address>[CR/LF]	- current IP address Returned.

5.3.2.8 OUR_NETMASK

This S-Register sets or returns the IP Netmask for the device.

Usage:

IRS7	- To return the current IP Netmask.
-or-	
IRS7=xxx.xxx.xxx.xxx	- To set the IP Netmask address.

Returns:

<IP Netmask>[CR/LF]	- current IP Netmask address returned.

5.3.2.9 OUR_GATEWAY

This S-Register sets or returns the Default Gateway Address.

Usage:

IRS8	- To return the Default Gateway IP address.
-or-	
IRS8=xxx.xxx.xxx.xxx	- To set the Default Gateway IP address

Returns:

<IP address>[CR/LF]	- current Default Gateway IP address Returned.

5.3.2.10 PRI_DNS

This S-Register sets or returns the primary DNS server IP Address.

Usage:

IRS9 - To return the current primary DNS server IP address.

-or-

IRS9=xxx.xxx.xxx.xxx - To set the current primary DNS server IP address.

Returns:

<IP address>[CR/LF] - current primary DNS server IP address Returned.

5.3.2.11 SEC_DNS

This S-Register sets or returns the Secondary DNS server IP Address.

Usage:

IRSA - To return the current secondary DNS server IP address.

-or-

IRSA=xxx.xxx.xxx.xxx - To set the current secondary DNS server IP address.

Returns:

<IP address>[CR/LF] - current secondary DNS server IP address Returned.

5.3.2.12 CONSOLE_BAUD

This S-Register sets or returns the Console Port Baud Rate Divider. If the command is a 'set' the Console Ports Baud Rate is changed immediately. . See Table 5-5 for baud rate divider values and corresponding serial port speeds.

Usage:

IRSB - To return the current Console Port Baud Rate Divider.

-or-

IRSB=xxx - To set the current Console Port Baud Rate Divider.

Returns:

<Divider Value>[CR/LF] - current Console Port Baud Rate Divider.

Serial Data Rate	S-Register 0x8, 0x9 Dec Value	S-Register 0x11, 0x12 Hex Value
2400bps	191	0xBF
4800bps	95	0x5F
9600bps	47	0x2F
14400bps	31	0x1F
19200bps	23	0x17
28800bps	15	0x0F
38400bps	11	0x0B
57600bps	7	0x07
76800bps	5	0x05
115200bps	3	0x03
230500bps	1	0x01

Table 5-5: BAUD rate table for sreg 0x8,0x9 and sreg 0x11,0x12

5.3.2.13 GPIO_DIR

This S-Register sets or returns the direction setting of the GPIO Pins (Pins 54-61). GPIO_DIR is an 8-bit hex value where each bit in the returned or set value corresponds to an individual GPIO pin. The lowest significant bit corresponds to the GPIO0 pin (Pin 61) and the most significant bit corresponds to the GPIO7 pin (Pin 54). Writing a '1' to a pins location will turn the pin into an output, driving the pin high or low depending on the setting of the GPIO_STATE S-Register register. Writing a '0' to a pins location sets the pin up as a High-Z input who's state (high or low) can be read by the GPIO_STATE S-Register. This register cannot be saved into the EEPROM and must be initialized if intended use differs from its default values.

Usage:

IRSC	- To return the current GPIO_DIR S-Register Value
-or-	
IRSC=xxx	- To set the GPIO_DIR S-Register Value

Returns:

<GPIO_DIR Value>[CR/LF]	- current GPIO_DIR S-Register Value

Default Value:

0x00	- Register always defaults to 0x00 upon power cycle.

5.3.2.14 GPIO_STATE

This S-Register sets or returns the current value of the GPIO Pins (Pins 54-61). GPIO_STATE is an 8-bit hex value where each bit in the returned or set value corresponds to an individual GPIO pin. The lowest significant bit corresponds to the GPIO0 pin (Pin 61) and the most significant bit corresponds to the GPIO7 pin (Pin 54). Writing a '1' to a pins location will either 1) turn on a weak pull-up of the corresponding GPIO_DIR pin is set to input or 2) drive the pin high. Writing a '0' to a pins location drives the pin low. This register cannot be saved into the EEPROM and must be initialized if intended use differs from its default values.

Usage:

IRSD	- To return the current GPIO Pin state (Pins 54-61)
-or-	
IRSD=xxx	- To set the output characteristics of the GPIO Pins

Returns:

<GPIO_STATE Value>[CR/LF]	- current GPIO pin state

Default Value:

0x00	- Register always defaults to 0x00 upon power cycle.

5.3.3 Upper S-Registers

The remaining S-Registers are all read/write values that take and return hex byte values. Be careful to not try setting read only state variables without understanding the consequences.

The usage for these S-Register values are as follows:
IRS<register>[=value]

Register is required, equals value is optional and will set the register to the passed value. If no equals value is passed the current registers contents are returned. The value parameter is a 2-byte hex value from 00 to ff corresponding to a decimal value between 0 to 255.

5.3.3.1 UNET_CONFIG

This S-Register sets or returns the uNetSerials configuration byte. This configuration byte controls the following features and each feature is controlled by one bit in the value set. Multiple configuration options to be set must be set by combining their corresponding bit values. The bit controls shown in Table 6.

Bit	Configuration	Action If Set
0x1	ECHO_CMD	Turns on echoing of command port input in command mode.
0x2	ECHO_STREAM	Turns on echoing of command port input in streaming socket mode.
0x4	RESULT	Turns on numeric response codes.
0x8	INBAND_ESCAPE	Turns on inband escape (+++)
0x10	Reserved	reserved
0x20	BOOT_BANNER	Setting this bit turns on the boot banner after reset.
0x40	DEBUG_MODE	Turns on debug messages.
0x80	RESULT_OFF	No Messages will be sent by the device. Flying Blind Mode.

Table 5-6: Configuration Bits for UNET_CONFIG

Usage:
IRS20
-or-
IRS20=value

Returns:
<Divider Value>[CR/LF] - current UNET_CONFIG value

Default Value:
27 - Register defaults to 27 (**BOOT_BANNER | ECHO_CMD | ECHO_STREAM | RESULT**) when default configuration is loaded

5.3.3.2 ESC_CHAR

This S-Register sets or returns the ESC_CHAR configuration byte. This byte defaults to '+'. Changing this value can change the inband escape character.

Usage:
IRS21
-or-
IRS21=value

Returns:
<Value>[CR/LF] - current ESC_CHAR value

Default Value:
2B - Register defaults to 2B ('+' ASCII) when default configuration is

loaded.

5.3.3.3 ESC_TIMEOUT

This S-Register sets or returns the ESC_TIMEOUT configuration byte. This byte describes the inband escape timeout (or guard time) in 10's of MS. This byte defaults to 200, which is equivalent to 2 seconds. Changing this value can change the ESC_TIMOUT from zero to 255 (or zero to 2.55 seconds)

Usage:
IRS22
-or-
IRS22=value

Returns:
<Value>[CR/LF] - current ESC_TIMEOUT value

Default Value:
C8 - Register defaults to C8 (200 or 2 seconds) when default configuration is loaded.

5.3.3.4 TCP_STREAM_TICK

This S-Register sets or returns the TCP_STREAM_TICK byte. This byte describes the TCP stream tick time. The TCP stream tick time is settable from 0 to 255 in 10's of MS. This byte defaults to 200, which is equivalent to 2 seconds. This value determines how long to wait for more input when in streaming socket mode. If this timeout occurs the current pending input data is sent in a TCP segment.

Usage:
IRS23
-or-
IRS23=value

Returns:
<Value>[CR/LF] - current TCP_STREAM_TICK value

Default Value:
C8 - Register defaults to C8 (200 or 2 seconds) when default configuration is loaded.

5.3.3.5 UDP_STREAM_TICK

This S-Register sets or returns the UDP_STREAM_TICK byte. This byte describes the UDP stream tick time. The UDP stream tick time is settable from 0 to 255 in 10's of MS. This byte defaults to C8 (200 decimal), which is equivalent to 2 seconds. This value determines how long to wait for more input when in UDP send mode. If this timeout occurs the current pending input data is sent in a UDP packet.

Usage:
IRS24
-or-
IRS24=value

Returns:
<Value>[CR/LF] - current UDP_STREAM_TICK value

Default Value:
C8 - Register defaults to C8 (200 or 2 seconds) when default configuration is loaded.

5.3.3.6 DNS_TIMEOUT

This S-Register sets or returns the DNS_TIMEOUT. This byte describes the DNS_TIMOUT in seconds. The DNS_TIMOUT byte is settable from 0 to 255 seconds. This byte defaults to 07 (decimal 7), which is 7 seconds. This value determines how long to wait for a DNS query response from a DNS server.

Usage:
IRS25
-or-
IRS25=value

Returns:
<Value>[CR/LF] - current DNS_TIMEOUT value

Default Value:
07 - defaults to 7 seconds when default configuration is loaded.

5.3.3.7 SERIAL_CONFIG

This S-Register sets or returns the SERIAL_CONFIG configuration byte. This configuration byte controls the various features of the modem and console serial ports. Table 7 lists the serial configuration features and bit value that controls each feature option.. Multiple configuration options to be set must be set by combining their corresponding bit values.

Bit	Configuration	Action If Set
0x1	Reserved	
0x4	CONSOLE_RTS_CTS	Turns on the use of RTS_CTS hardware flow-control on the console serial port.
0x10	Reserved	
0x20	Reserved	
0x80	AUTOBAUD	Sets auto BAUD rate detection on the console port.

Table 5-7: Configuration Bits for SERIAL_CONFIG

Usage:
IRS26
-or-
IRS26=value

Returns:
<Value>[CR/LF] - current SERIAL_CONFIG value

Default Value:
0 - Register defaults to 0 when default configuration is loaded.

5.3.3.8 CONSOLE_BAUD_HEX

This S-Register sets or returns the CONSOLE_BAUD_HEX configuration byte. This register is similar to S-Register 0x09 but value written and returned is in hex. Also setting this register does not take effect at write time, it only sets the configuration byte. This configuration may be saved into the EEPROM and the changes will take effect after the next system reset or power-cycle.

Usage:

IRS27
-or-
IRS27=value

Returns:
<Value>[CR/LF] - current CONSOLE_BAUD rate setting in hex

5.3.3.9 UDP_FLAGS

This S-Register sets or returns the UDP_FLAGS configuration byte. This configuration byte controls the various features of the UDP stack. Table 5-8 lists the serial configuration features and bit value that controls each feature option. Multiple configuration options to be set must be set by combining their corresponding bit values.

Bit	Configuration	Action If Set
TBD		

Table 5-8: Configuration Bits for UDP_FLAGS

Usage:
IRS28
-or-
IRS28=value

Returns:
<Value>[CR/LF] - current UDP_FLAGS value

Default Value:
00 – Register defaults to 0 when default configuration is loaded.

5.3.3.10 TCP_CONNECT_TIME

This S-Register sets or returns the TCP_CONNECT_TIME register. This byte describes the TCP connect timeout in seconds. The TCP_CONNECT_TIME byte is settable from 0 to 255 seconds. This byte defaults to 0a (decimal 10), which is 10 seconds. This value determines how long to wait for a TCP connection to establish before giving up.

Usage:
IRS29
-or-
IRS29=value

Returns:
<Value>[CR/LF] - current TCP_CONNECT_TIME value

Default Value:
0a - Register defaults to 10 seconds when default configuration is loaded.

5.3.3.11 TCP_RETRANS_TIME

This S-Register sets or returns the TCP_RETRAMS_TIME register. This byte describes the TCP retransmission timeout in. The TCP_CONNECT_TIME byte is settable from 0 to 255 in 10's of MS. This byte defaults to FA (decimal 250), which is 2.5 seconds. This value determines how long to wait for an

ACK before re-transmitting as segment.

Usage:
IRS2a
-or-
IRS2a=value

Returns:
<Value>[CR/LF] - current TCP_RETRANS_TIME value

Default Value:
FA - Register defaults to 2.5 seconds when default configuration is loaded.

5.3.3.12 TTL

This S-Register sets or returns the IP_TTL register. This byte describes the IP time to live value transmitted in all IP packets. IP TTL defaults to 64 (decimal 100). This value determines how may hops an IP packet generated from this device can take before being discarded

Usage:
IRS2b
-or-
IRS2b=value

Returns:
<Value>[CR/LF] - current IP_TTL value

Default Value:
64 - Register defaults to 100 hops

5.3.3.13 IP_TOS

This S-Register sets or returns the IP_TOS register. This byte describes the IP type of service value transmitted in all IP packets. IP TOS defaults to 0.

Usage:
IRS2c
-or-
IRS2c=value

Returns:
<Value>[CR/LF] - current IP_TOS value

Default Value:
0 - Regular type of service

5.3.3.14 Network Configuration registers

This S-Register sets or returns the NETWORK_CONFIG register.

Bit	Configuration	Action If Set
0x1	DHCP_ENABLE	Set if DHCP is enabled if set

Table 5-9: Configuration Bits for NETWORK_CONFIG

Usage:
IRS2d
-or-
IRS2d=value

Returns:
<Value>[CR/LF] - current Network Configuation Register Setting

Default Value:
0 - No DHCP

5.3.3.15 OUR_MAC_ADDR_HEX

This is a set of S-Registers that contains the device MAC address. There are six consecutive registers from 0x2e to 0x33 that represent the OUR_MAC_ADDR_HEX. The first register 0x2e represents the lower significant byte of OUR_MAC_ADDR_HEX (i.e. In bold 00:00:14:00:00:**05**). The last register in the series (0x33) represents the most significant byte of OUR_MAC_ADDR_HEX (i.e. In bold **00**:00:14:00:00:00). The IP address bytes represented in this series of registers are HEX values. To access the devices MAC address as a single command instead of individual bytes see S-Register 0x05.

Usage:
IRS2e
-or-
IRS2e=value

Returns:
<Value>[CR/LF] - current Lowest Significant byte of IP address.

Default Value:
00
- AC address bytes default to 00:00:14:00:xx:xx
- Where XX is device dependant

5.3.3.16 OUR_IP_ADDR_HEX

This is a set of S-Registers that contains the device IP address, this can be a user set or DHCP negotiated value. There are four consecutive registers from 0x34 to 0x37 that represent the OUR_IP_ADDR. The first register 0x34 represents the lower significant byte of OUR_IP_ADDR (i.e. In bold 192.168.11.**150**). The last register in the series (0x37) represents the most significant byte of OUR_IP_ADDR (i.e. In bold **192**.168.11.150). The IP address bytes represented in this series of registers are HEX values. To access the devices IP address in decimal values please see S-Register 0x06.

Usage:
IRS34
-or-
IRS34=value

Returns:
<Value>[CR/LF] - current Lowest Significant byte of IP address.

Default Value:
0 - All IP address Bytes default to 0

5.3.3.17 OUR_NETMASK_HEX

This is a set of S-Registers that contains the device NETMASK address. This value is set via DHCP or by the USER. There are four consecutive registers from 0x38 to 0x3b that represent the OUR_NETMASK. The first register 0x38 represents the lower significant byte of OUR_NETMASK (i.e. In bold FF.FF.FF.**0**). The last register in the series (0x3b represents the most significant byte of OUR_NETMASK (i.e. In bold **FF**.FF.FF.0). The IP address bytes represented in this series of registers are HEX values. To access the devices Netmask address in decimal values please see S-Register 0x07.

Usage:
IRS38
-or-
IRS38=value

Returns:
<Value>[CR/LF] - current Lowest Significant byte of Netmask.

Default Value:
00 - All netmask bytes default to 0.

5.3.3.18 GATEWAY_HEX

This is a set of S-Registers that contains the device IP Gateway address. This value is set via DHCP or by the USER. There are four consecutive registers from 0x3c to 0x3f that represent the IP Gateway Address. The first register 0x3c represents the lower significant byte of the IP Gateway address (i.e. In bold 10.10.10.**1**). The last register in the series (0x3f represents the most significant byte of the IP Gateway Address (i.e. In bold **10**.10.10.1). The IP address bytes represented in this series of registers are HEX values. To access the devices IP Gateway address in decimal values please see S-Register 0x08.

Usage:
IRS3C
-or-
IRS3C=value

Returns:
<Value>[CR/LF] - current Lowest Significant byte of Gateway IP address

Default Value:
00 - All Gateway IP bytes default to 0.

5.3.3.19 PRI_DNS_ADDR_HEX

This is a set of S-Registers that contains the primary DNS server IP address. There are four consecutive registers from 0x40 to 0x43 that represent the PRI_DNS_ADDR_HEX. The first register 0x40 represents the lower significant byte of primary DNS server address (i.e. in bold 192.168.11.**150**). The last register in the series (0x44) represents the most significant byte of primary DNS server address (i.e. in bold **192**.168.11.150). The IP address bytes represented in this series of registers are HEX values. To access the primary DNS server address in dotted decimal format, see S-Register 0x9.

Usage:
IRS40
-or-
IRS40=value

Returns:
<Value>[CR/LF] - Lowest Significant byte of the primary DNS servers IP address.

Default Value:
0 - All secondary DNS server IP address Bytes default to 0

5.3.3.20 SEC_DNS_ADDR_HEX

This is a set of S-Registers that contains the secondary DNS server IP address, this can be a user set or a DHCP negotiated value. There are four consecutive registers from 0x44 to 0x47 that represent the SEC_DNS_ADDR_HEX. The first register 0x44 represents the lower significant byte of primary DNS server address (i.e. in bold 192.168.11.**150**). The last register in the series (0x47) represents the most significant byte of secondary DNS server address (i.e. in bold **192**.168.11.150). The IP address bytes represented in this series of registers are HEX values. To access the secondary DNS server address in dotted decimal format, see S-Register 0xa.

Usage:
IRS44
-or-
IRS44=value

Returns:
<Value>[CR/LF] - Lowest Significant byte of the secondary DNS servers IP address.

Default Value:
0 - All secondary DNS server IP address Bytes default to 0

5.3.3.21 IRCMD_STATE

This is a read only S-Registers, writing to it can cause unpredictable results. This register returns the state of the IR command processor. The states are shown below in table 5-10. These states are also described in Earlier in this section and shown in Figure 5-1.

Value	State
0x1	Modem Command State (not available on uNetEthernet)
0x3	Modem Data State (not available on uNetEthernet)
0x4	Internet Command State
0x6	Streaming Socket State
0x9	Modem Pass Through State (not available on uNetEthernet)

Table 5-10: IR command processor state table

Usage:
IRS48
-or-
IRS48=value

Returns:
<Value>[CR/LF] - Current IR command processor state.

Default Value:
04 - On Reset or power up uNetEthernet defaults to Internet Command State.

6 BOOTLOADER OPERATION

All uNetSerial and uNetEthernet modules contain bootloaders. A bootloader allows the module to be upgraded by sending the chip new code via the serial port. This allows enhancements, bug fixes and special versions of the module firmware to be loaded at any time. There are 2 methods of entering into bootloader mode, one by issuing the 'IRB' command, and two by pulling SW4 low and holding it there during a power cycle or reset. The bootloader accepts x-modem upload of new firmware. If the bootloader is entered via reset and holding SW5 low the console serial port will be running at 19200bps. If the bootloader is entered from the uNetEthernet firmware via the 'IRB' command the serial port will be running at the same rate as the uNetEthernet firmware was running at.

When entering the boot loader the console port will output:

nChip Bootloader V1.4

This signifies that the bootloader is active. The following screenshot shows entering the bootloader and getting ready to send a new image via x-modem using the windows program HyperTerminal.

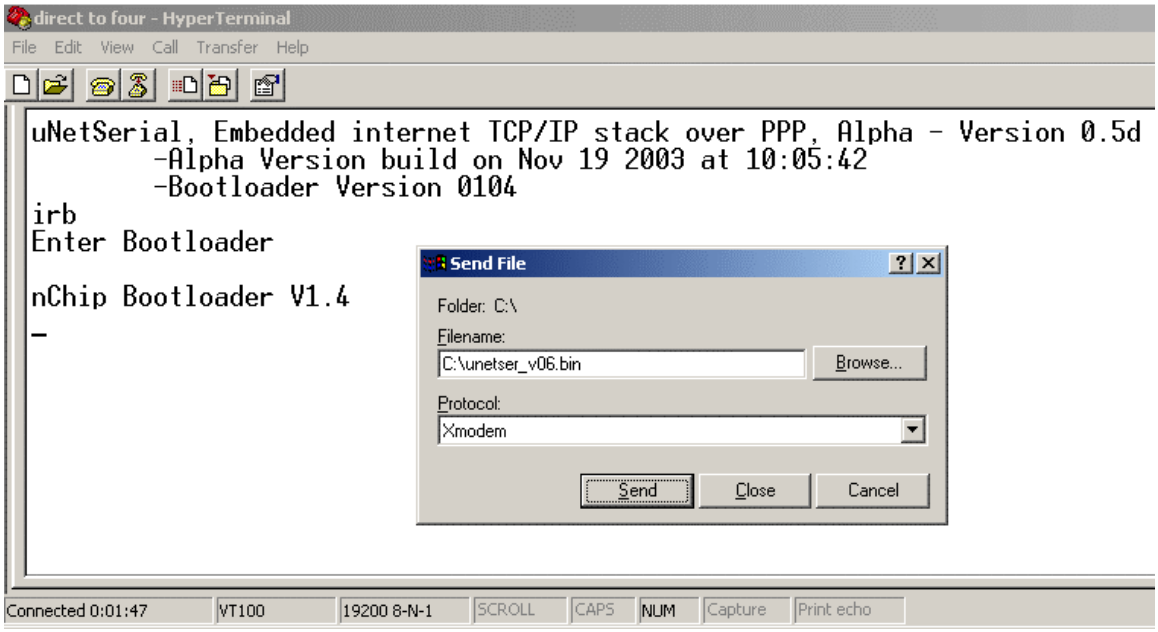


Figure 6-1: Bootloader Operation

Once the x-modem transfer of the new boot image has completed the new version of the firmware should startup and its version number should be printed to the screen.

Note Figure6-1: Shows the uNetSerial product in operation, uNetEthernet Bootloader operation is exactly the same except you would use a uNetEthernet .bin file.

7 CORRUPT EEPROM RECOVERY

It is possible that a user could write a wrong value into a serial baud rate S-Register or other such miss-configuration that renders the uNetEthernet Chip inaccessible. In these rare cases it is possible to reset the EEPROM to factory defaults by pressing and holding SW4 for 15 seconds.

Once the reset of the EEPROM has take place, Indicator 2 will light up.

After a successful reset of the EEPROM, simply reset or power cycle the uNetEthernet device and it will operate in its default configuration.

8 SAMPLE IR COMMAND SEQUENCES

The follower is a set of sample IR command sequences for common Internet services. If your are evaluating with a terminal emulator note that internet commands need to be terminated with both a return and a linefeed, where as IR commands only need a return.

8.1 SAMPLE HTTP CLIENT

This communications protocol provides a straightforward way to transmit and receive data with an HTTP server. This is not limited to HTML content because any ASCII data can be transmitted via HTTP. This is an effective way of transmitting any ASCII data. The use of CGI can make this a powerful tool.

A simple CGI program can accept an HTTP GET with data parameters, and process the data, possibly checking for checksums included in the data, possibly parsing the data for specific information. A response can be generated on the fly and returned in the same HTTP GET transaction, possibly containing status, or configuration information. The data can then be stored in a database or simple file. This also provides a capability for dynamic formatting an presentation of the data in HTML format for a web browser. This can be done through various means, depending on the data storage and the web tools available.

For example:

1. The client device connects to the HTTP server through the uNetEthernet device.
2. The client device performs a GET to a CGI program running on the server.
3. The client device sends parameter data to the CGI program in the GET URL.
4. The CGI program parses the parameter data and saves it to the server.
5. The CGI returns status to the client device in the GET return body.
6. The client device disconnects.
7. The data can be viewed through any PC web browser through another CGI program running on the server.

In the example above, the communications would look something like this:

➔ IR

Check to see if there is an uNetEthernet device present.

🡐 OK

➔ IRT0 www.mycal.net:80

Connect to an TCP server at www.mycal.net and port 80 (HTTP).

🡐 CONNECT

The TCP connection was made for socket 0. uNetEthernet is now in connected and online modes. (Note GET must be all caps).

➔ GET /vending-demo/?id=09&slota=1&slotb=2

This is data being sent to a HTTP Server. It is a HTTP 0.9 GET transaction to a CGI script that can capture the argument data (id # and slota and slotb values). The script logs the data and return status in the HTTP 0.9 GET Body. Make sure both return and line feed are sent on this command. Alternatively you can request via HTTP 1.0 with 'get http://www.mycal.net/vending-demo/?id=1 HTTP/1.0' but you will get a more complex return.

🡐 01 10 10;

This is data being returned from the HTTP Server. It is a HTTP 0.9 GET Body. This is the data output by the CGI script (01=ok 10=new slota value 10=new slotb value ;=end data.)

➔ +++

Escape uNetEthernet from online mode to command mode. The uNetEthernet is now in connected and command modes.

➔ OK

➔ IRX0

Release the socket 0 connection.

➔ OK

The TCP socket was closed and the socket 0 was released.

➔ IRD

Terminate the PPP connection.

➔ OK

Done.

8.2 SAMPLE SMTP CLIENT

This communications protocol provides a straightforward way to transmit data with an SMTP server. This is generally used to send data to a specific email address. This is an effective way of transmitting any ASCII data.

For example:

1. The client device connects to the SMTP server through the uNetEthernet device.
2. The client device enters the return address of the device.
3. The client device enters one or more target addresses.
4. The client device sends data to the server.
5. The client device disconnects.
6. The data can be retrieved from the mailbox of the target address.

In the example above, the communications would look something like this:

➔ IR

Check to see if there is an uNetEthernet device present.

← OK

➔ IRT0 mycal.net:25

Connect to an TCP server at mycal.net and port 25 (SMTP).

← CONNECT

The TCP connection was made for socket 0. uNetSerial is now in connected and online modes.

← 220 mycal.com ESMTP Sendmail 8.9.3/8.9.3; Tue, 10 Dec 2000 15:27:02 -0800

This is data being sent from the SMTP server upon receiving a TCP connection.

➔ HELO remotedevice.nchip.com

This is the client checking for an active SMTP Server session and "logging in". Make sure both return and line feed are sent on this command and all other commands sent to the server.

← 250 OK

This is SMTP Server's reply.

➔ MAIL FROM:<remotedevice@nchip.com>

This is the client setting the return address of the message.

← 250 OK - mail from < remotedevice@nchip.com>

This is SMTP Server's reply.

➔ RCPT TO:<devicelogger@nchip.com>

This is the client setting a recipient of the message. This command can be called multiple times for a single message to set multiple recipients of the message.

← 250 OK - Recipient <deviceserver@nchip.com>

This is SMTP Server's reply.

➔ DATA

This is the client telling the server that it is going to begin streaming the message body.

← 354 Send data. End with CRLF.CRLF

This is SMTP Server's reply. The message body is terminated by a "CRLF.CRLF" sequence.

➔ Subject: this is a test, this is only a test

➔ if this were real data, it would have meaning

➔ .

This is the message body. Notice it was terminated with a "CRLF.CRLF". This sequence is not included in the message data that the server receives.

← 250 OK

This is SMTP Server's reply. It has sent the message at this point.

➔ QUIT

This is the client terminating the SMTP Server session.

← 221 closing connection

This is SMTP Server's reply. The Server then terminates the socket connection.

← NO CARRIER

There is no need to escape uNetEthernet from online mode to command mode because the server closed the socket. The uNetEthernet is now in connected and command modes.

➔ IRX0

Release the active socket connection.

← OK

The TCP socket was closed and the socket 0 was released.

8.3 SAMPLE UDP TRANSACTION

In next release.

8.4 SAMPLE FTP CLIENT

This communications protocol provides a straightforward way to transmit data with an FTP server. This is generally used to send or receive data to a specific file on a server. This requires both sockets to be used and uses that PASV protocol version of FTP. See **RFC** 959 for more information on the FTP protocol

A FTP session would look something like this:

➔ IR

Check to see if there is an uNetSerial device present.

🡐 OK

➔ IRT 0 ftp.hosts.co.uk:21

Connect to an TCP server at ftp.hosts.co.uk and port 21 (FTP command port).

🡐 CONNECT

The TCP connection was made for socket 0. uNetEthernet is now in connected and in streaming socket mode..

🡐 220 ProFTPD 1.2.10 Server (NamesCo Limited) [192.168.0.7]

This is data being sent from the FTP server upon receiving a TCP connection.

➔ user bigdog

Login with user bigdog

🡐 331 Password required for bigdog

User bigdog needs a password

➔ pass nchip

Login with user bigdog

🡐 230 User bigdog logged in.

User bigdog is logged in, ready for ftp protocol.

➔ TYPE I

Login

🡐 200 Type set to I

User \

➔ PASV

Set PASV mode

🡐 227 Entering Passive Mode (212,84,175,70,152,37).

Server set for PASV mode on 212.84.175.70 port 38949 for data port

➔ [guard time]+++[guard time]

Escape to command mode, can also be done by toggling DTR. Guard time is ~1 second or what is set in S-Register

🡐 OK

uNetEthernet is now in command mode.

➔ IRT 1 212.84.175.70:38949

Connect seconds socket to data FTP port.

🡐 OK

Socket connected

➔ [guard time]+++[guard time]

Escape to command mode, can also be done by toggling DTR

🡐 OK

uNetEthernet is now in command mode.

➔ IRR0

Return to socket 0 (command FTP socket)

🡐 OK

Socket connected

➔ STOR /testfile.txt

Send a data file called testfile.txt

🡐 150 Opening BINARY mode data connection for /eaglepics.txt

FTP server now ready to receive file.

➔ [guard time]+++[guard time]

Escape to command mode, can also be done by toggling DTR

🡐 OK

uNetEthernet is now in command mode.

➔ IRR1

Return to socket 1 (data FTP socket)

🡐 OK

Socket connected

➔ EaglePICS provides a range of products and services supporting Internet connectivity over GPRS networks.

Send data to socket.

➔ [guard time]+++[guard time]

Escape to command mode, can also be done by toggling DTR

🡐 OK

uNetEthernet is now in command mode.

➔ IRX1

Close socket 1, which completes data transfer

🡐 OK

Socket now closed

➔ IRR0

Connect back to socket 0, command socket.

🡐 OK

Connected back to socket 0.

🡐 226 Transfer complete.

Transfer complete message returned from server.

➔ [guard time]+++[guard time]

Escape to command mode, can also be done by toggling DTR

🡐 OK

uNetEthernet is now in command mode.

➔ IRX0

Close socket 0, can also use ‘quit’ command

🡐 OK

Socket now closed

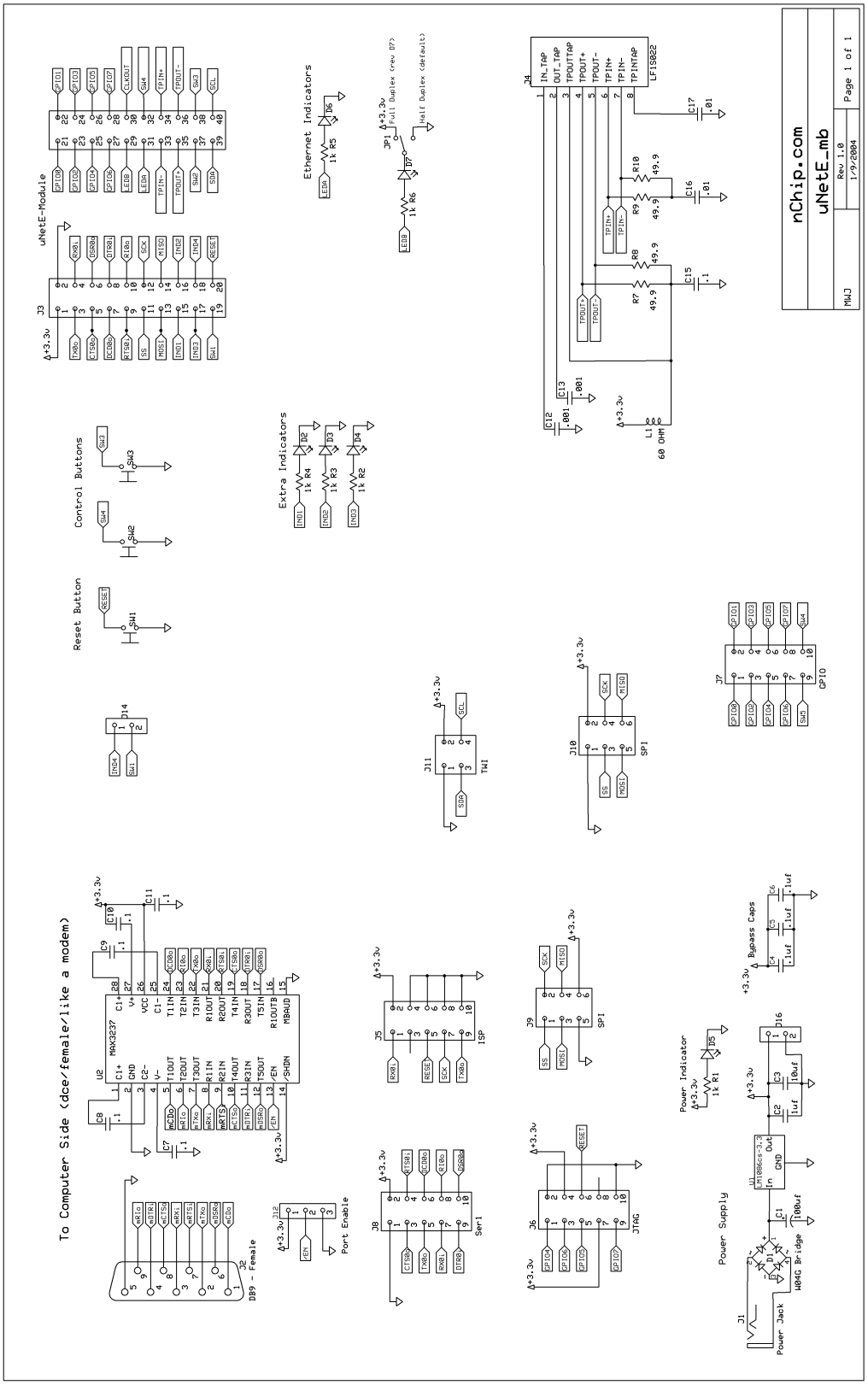
9 SCHEMATICS

The following page contains and example system that incorporates an uNetEthernet Internet command processor.

This first schematic is that of the uNetEthernet Module. The second schematic is the uNetEthernet development board that accepts an uNetEthernet Module.

9.1 uNetEthernet Module Schematic

9.2 uNetEthernet Development Board Schematic



page left blank