

uNetSerial – Free Version

Internet Processor for Dialup and Wireless Phone Systems

www.nchip.com

©2001-2003 nChip

Introduction

uNetSerial is a device to Internet processor that enables simple devices to connect to the Internet via dialup or wireless phone infrastructure. It is based on the advanced IR command processor which offers simple but powerful extensions to the AT modem command set. 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.

uNetSerial's processor can negotiate a PPP (Point to Point Protocol, the standard way to connect by modem or wireless phone) connection using the standard PAP or script authentication methods. It will negotiate nameserver addresses and supports full name resolution. TCP and UDP sockets can be created and communicated over. Full non-volatile configuration support for serial port speeds, flow control methods, Internet stack parameters and many other features is supported.

UNetSerial 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.

* Please note that the DEMO version only supports UDP sockets.

Kit Contents

This document (unetserial.pdf), unetser.exe and the AVR MEGA128 hex file (unetser_v05d.hex) are all that is included with this kit. The hex file is meant to be programmed in a AVR Mega128 processor and this document describes its operation.

'unetser.exe' is a windows emulation platform that emulates the uNetSerial command processor on a windows based PC.

More information on the Windows emulation platform can be found below in the Emulator section or at www.nchip.com

Getting Started

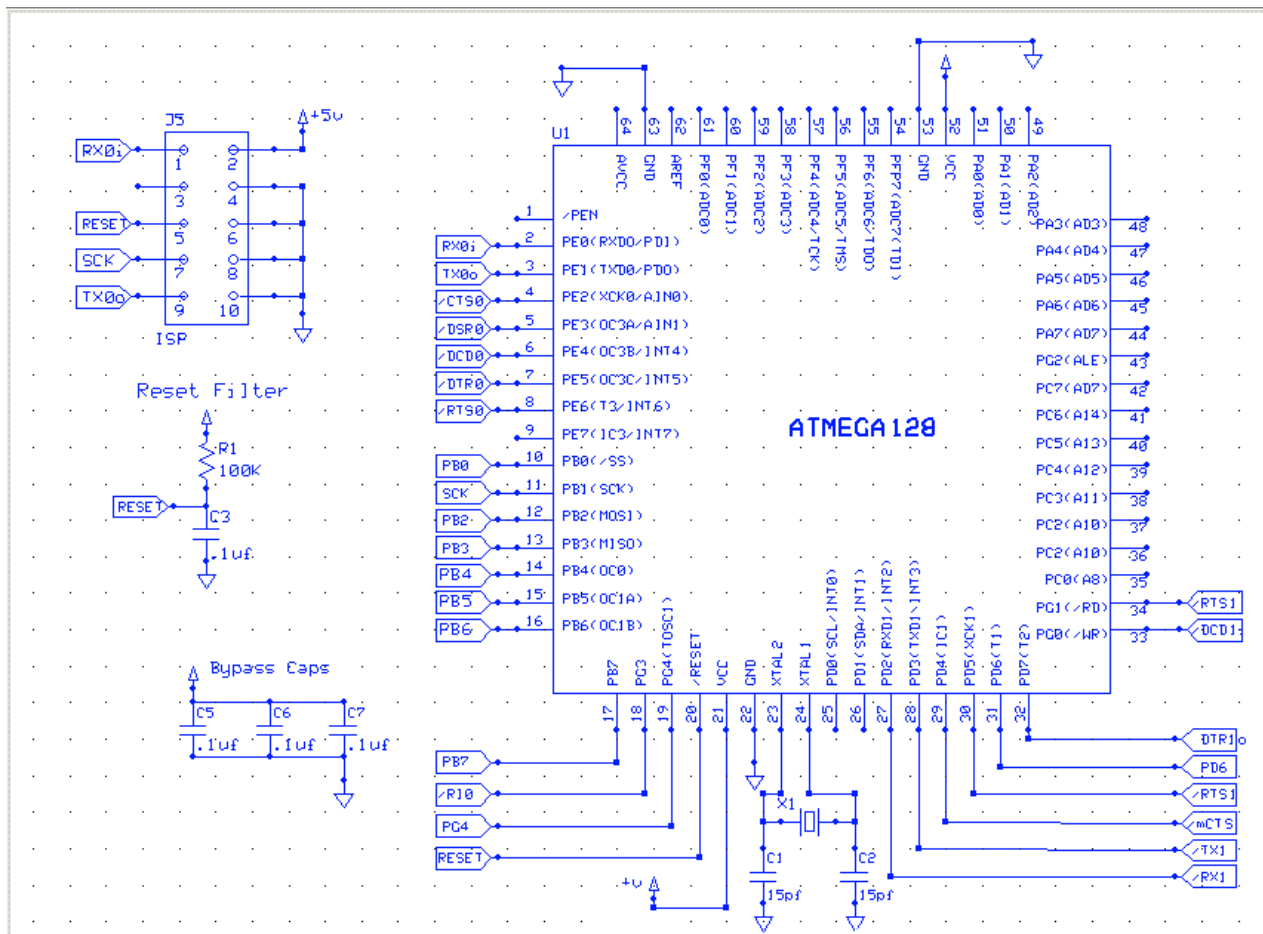
Hook up the chip as specified in the schematic, you may need RS232 level converters if the serial ports that you are connecting the chip to are not at TTL levels.

The demo version does not support autobaud of the console port (RX0/TX0 on the schematic) and by default is set the 19200bps. The modem port defaults to 9600bps. Both these values can be configured by the S-Register commands and saved into the EEPROM of the AVR.

Once programmed and connected various commands can be issued to the processor. If the modem port is connected to a modem commands that the uNetSerial command processor doesn't understand get passed through to the modem until a PPP connection has been established.

Schematic

This is the schematic of the chip, at minimum the RX0, TX0, RX1 and TX1 connections have to be utilized from the serial ports. All power and ground, resistors and caps must be used. The X1 crystal frequency is 7.3728Mhz. You must tie Pin 9 to Pin 2 to get Auto BAUD detection to work.



Programming

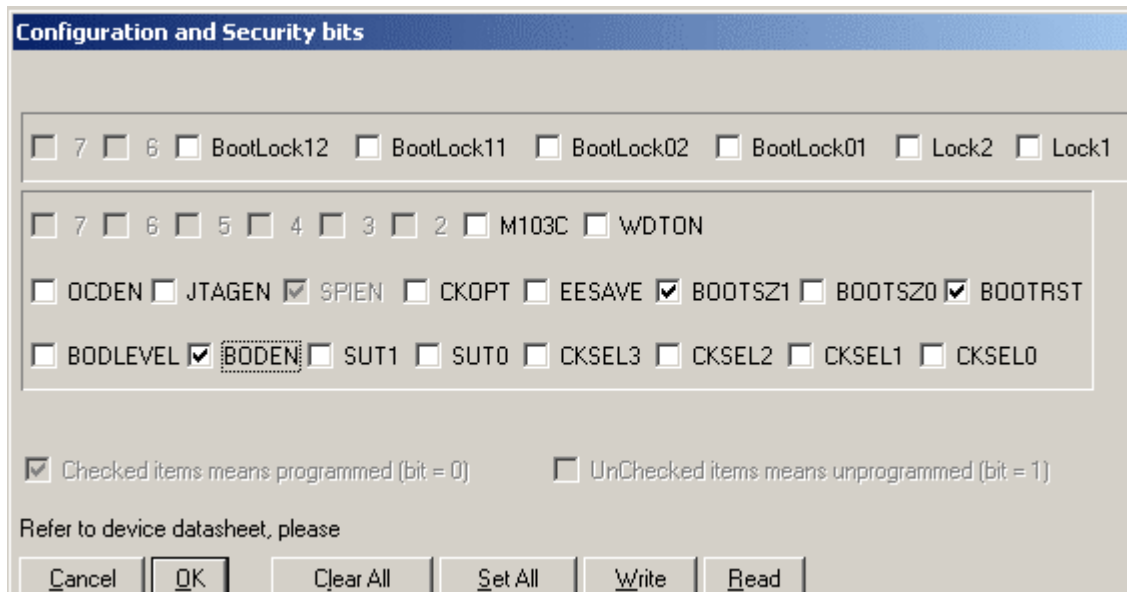
'unetser.hex' can be programmed using any programmer that supports Intel HEX format. We at nChip use ponyprog. Ponyprog can be found at <http://www.lancos.com/>

'unetser.hex' is to be programmed into an AVR MEGA128 processor only, other processor support is coming soon.

FUSE Bits

When programming the demo, using a 7.3728Mhz crystal the only fuse that needs to be programmed is the BODEN, if you are running the AVR at 3.3v then the BODLEVEL also needs to be set.

If you do not enable the BODEN (Brown Out Detect) then there is a slight possibility that the EEPROM configuration may get corrupt.



Emulator

unetser.exe' is a windows emulation platform that emulates the uNetSerial command processor on a windows based PC. It contains extra debugging features that can help a user debug some connection problems. 'unetser.exe' takes two parameters, a number between 1 and 4 to specify which windows serial port to use for the modem connection, and a number representing the serial port speed to use. (Example on the command line type 'unetser 4 57600' to select COM port 4 at 57600bps. If no parameters are typed on the command line 'unetser.exe' defaults to COM 1 at 19200bps.) 'unetser.exe' requires Windows 2k or Windows XP.

The emulator will also write a log file named unetser.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.

Command Processor Commands

The following are the commands supported in uNet Serial Command Processor Version 0.5.

Commands:

<required parameter> [optional parameter]

PPP COMMANDS

IRC [name],[password] - IRC will try to raise a PPP connection. Name and or Password are optional. If the PPP peer (peer server) requests PAP the device will use the passed name and password for authentication.

Returns:

OK[CR/LF] if PPP has been raised successfully.

FAIL PPP[CR/LF] PPP timed out or otherwise failed to negotiate a connection

AUTH FAIL PPP[CR/LF] if PAP authentication failed.

IRD - disable the PPP connection and return to command mode. This will also toggle the modem DTR pin to drop the carrier.

DNS COMMANDS

IRN <name> - This command will try to resolve a name into an IP address using the Domain Name Service protocol.

Returns:

IP address[CR/LF] if name was correctly resolved

FAIL DNS[CR/LF] if DNS resolution has failed.

ICMP COMMANDS

IRP <hostname/ip> - Sends a ping (ICMP Echo Request) and waits for a reply.

Returns:

OK - [xxx]ms if an ping response was returned

FAIL - if there was no response

UDP COMMANDS

IRU<cmd> - UDP commands

IRUB <port> - binds a UDP port to UDP for receive.

IRUP [ip/name]:[destport]:[sourceport] - 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 IP address 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.

IRUG - Gets a UDP packet that was received on the port set by the last IRU 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 the 3 lines are:
IP address of peer that sent packet[CR/LF]
Port of peer that sent the packet[CR/LF]
Length of packet[CR/LF]
packet data

Note there is no [CR/LF] after packet data, and the number of packet data bytes is as specified in the Length of packet field. The IP address, Port Line and Length Line can be turned on or off via the UDP config S-Register.

IRUV - stops the device from listing on the previously specified port (IRU)
Returns:
OK[CR/LF] Always

TCP COMMANDS (not on demo version)

IRT<socket><name/ip>:<destport> - Try's to connect a socket to a remote host at destport.

Returns:
OK if socket connected
DNS FAIL - if name could not be resolved to IP
TCP FAIL - Could not connect to socket.
ERROR - Error in passed parameter format.

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.

IRR <socket>-Reconnect to a specified TCP socket from command mode.

Returns:
OK [CR/LF] - if the socket is reconnected
FAIL SOCKET DOWN[CR/LF] - Socket is down.

IRX <socket>-Closes a specified TCP socket. Always returns OK

MISC COMMANDS

IRE<I,S,L or location>[=value] - view or set EEPROM location, (I)nvalidate EEPROM config, (S)ave EEPROM, config, or (L)oad EEPROM config.

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

IRO - turn off IR command processor. After issued device will just act as a serial pass through device. The only way to get back into IR command processor mode is to reset the chip or toggle the DTR pin.

IRM - toggle the modems DTR pin <this can force modem into command mode and drop modem carrier.

S-REGISTERS (subject to change, some serial configs not usable on windows emulation yet)

<Lower Registers>

0	- TCP socket 0 status
1	- TCP socket 1 status
2	- UDP status
3	- SW version
4	- bootloader version
5	- ip address
6	- primary DNS server address
7	- secondary DNS server address
8	- Modem Baud Rate - Rate in DEC (see table), Takes effect instantly
9	- Console Baud Rate - Rate in DEC (see table), Takes effect instantly
a	- GPIO Direction Register, 0=input, 1=output, 8 bits for pins 61-54
b	- GPIO pins, write to change outputs or pull up inputs, read for levels

<Upper Registers>

20	- IMMConfig DEBUG_SEND DEBUG_MODE x DATA_MODE_ESC INBAND_ESC RESULT ECHO_STREAM ECHO_CMD
21	- escapeChar - this is the char used for IMM escape to cmd mode
22	- escape_timeout - this is the guard time between data and escap chars
23	- TCPStreamTickTime - TCP send timer (0-255 in 10's of MS)
24	- UDPStreamTickTime - UDP send timer (0-255 in 10's of MS)
25	- DNS Timeout
26	- SerialConfig Byte CAB CD CD x CCTS CRTS MCTS MRTS
27	- ModemBaud
28	- ConsoleBaud
29	- ppp_connect_timeout (in seconds 0-255)
2a	- ppp_ACCM -ACCM for ppp link Highbyte
2b	- ppp_ACCM; -ACCM for ppp link
2c	- ppp_ACCM; -ACCM for ppp link
2d	- ppp_ACCM; -ACCM for ppp link Lowbyte
2e	- udp_flags - UDPDVCHAR DAVDSR DAVDV DIRECT LEN IP PORT
2f	- tcp_connect_timeout (0-255 seconds)
30	- tcp_retransmit_timeout (0-255) in 10's of MS
31	- ip_ttl
32	- ip_tos
33	- our_ipaddr;
34	- our_ipaddr;
35	- our_ipaddr;
36	- our_ipaddr;
37	- peer_ip_addr;
38	- peer_ip_addr;
39	- peer_ip_addr;
3a	- peer_ip_addr;
3b	- pri_dns_addr;
3c	- pri_dns_addr;
3d	- pri_dns_addr;
3e	- pri_dns_addr;
3f	- sec_dns_addr;
40	- sec_dns_addr;
41	- sec_dns_addr;
42	- sec_dns_addr;

//

// Read Only, writing can crash system

```
//
43          - ircmd_state
44          - ppp_flags
45          - lcp_state
46          - pap_state
47          - ipcp_state
48          - ppp_tx_mru (lowbyte)
49          - ppp_tx_mru (highbyte)
```

EEPROM Memory Map (subject to change, not used for windows)

```
-----
0x0          - if 0xa5 then configuration is valid
0x01-0x0a    - reserved
0x0a-0x50    - non-volatile configuration storage that mirrors S-Register 0-0x29
0x51-0x5f    - reserved
0x60->emax   - reserved for commands processor code
```

BAUD rate table for AVR for sreg 8 and sreg 9

```
-----
191          = 2400bps
95           = 4800bps
47           = 9600bps
31           = 14400bps
23           = 19200bps
15           = 28800bps
11           = 38400bps
7            = 57600bps
5            = 76800bps
3            = 115200bps
1            = 230400bps
```