



SnIP Quick Start Guide

M5/500 Ethernet Option Interface

1.0 Optional SnIP Ethernet Interface Overview

This Quick Start Guide, or QSG is intended for technical personnel who are familiar with networks and routing plus the Linux Operating System. It is intended to familiarize someone already up to speed on these subjects with the unique features and operating mechanisms of the SnIP.

The optional **S**atellite **n**etwork **I**nterface **P**rocessor or "**SnIP**" Ethernet Interface is an all new processor based Ethernet and TCP/IP interface for the PSM-4900 and PSM-500 series of satellite modems. The interface is contained on a small, low power (approx 1 Watt) printed circuit card that installs within the modem chassis. The main rear panel data interface is an Ethernet 10/100 Base-T, RJ45 type. The SnIP to modem interface is also an internal synchronous serial data port and modem control port that connect directly to the modem.

The main purpose of the SnIP is to convert Ethernet TCP/IP traffic to a form suitable for satellite transmission, specifically over Datum Systems' satellite modems. When two companion SnIP units are on each end of a satellite link, it becomes a smart cable.

The processor runs a customized version of the open source Linux Operating System with sophisticated bridging, routing and statistical functions. This operating system is also capable of loading and running numerous other IP related open source and proprietary third party applications. The implications of running Linux are wide ranging including both good and bad features. You could think of the SnIP as a normal computer running Linux, minus the video display, but with an added high speed synchronous serial port.

Linux uses names for the interface ports used to carry traffic. The RJ45 Ethernet connection is named "**eth0**", while the internal high speed synchronous serial port connected to the modem is named "**hdlc0**". It is named hdlc0 because the basic packet type used is HDLC. Some other routers may name this port "serial1" or "wan0" (Cisco style "serial 1" or "Serial 0/0").

This guide does not intend to explain the operation and features of Linux itself, nor extensive details on how to set up and operate routers as those are subjects already occupying many volumes of material, which is constantly changing and evolving. Instead it covers basic operation of the standard software built in, and how it relates to the operation in conjunction with a satellite modem. Basic network topologies and SnIP modes are discussed in Section 3 of this guide.

As of this writing in February 2009 the SnIP is still evolving, and not all of the final intended features are fully implemented.

Since this is a Quick Start Guide, we will begin immediately with abbreviated information on how to set up the SnIP for each of 3 standard functions, Bridge, Router and Monitor & Control only interface plus a few other common procedures. Further explanations of the interfaces, Operating System and procedures follow these brief descriptions.

Important! Any satellite link presents unique problems for TCP/IP traffic because of the 500 mS round trip delay caused by the physical distance of the satellite above the earth. Without some form of PEP or "packet acceleration" the TCP/IP data rate is limited to between 128 kbps and 1 Mbps! UDP packets are not rate limited.

We are currently testing full functionality with XipLink acceleration appliances.

1.0.1 Common Setup To Configure the SnIP

Before you can configure a SnIP or any network device there are several pieces of information you need to know:

1. The IP Address range of the LAN or network(s) you will be connecting the SnIP into. For a link you need this for both ends of the link. From this you should then select the SnIP's desired IP Address and Netmask.
2. The connected LAN may already have an Internet "Gateway" address, which is the address that it goes to for external "Internet or Web" access. The SnIP may be the Gateway at some sites. You need to know the web Gateway address if used.
3. Your Internet Service Provider or ISP should have also given you the names and IP Addresses of two or more DNS Servers (Dynamic Name Servers). You need these numbers. Otherwise you may optionally use the addresses of some public DNS servers. More information and a web link is provided in Section 8.2.5 of this document and in the SnIP itself along with some common addresses. Having different DNS server addresses in the SnIP and the local LAN may slow down some operations.
4. You have to know what configuration you want to set the SnIP link to, that is, Bridge, Router or controller only. Network Topology is discussed in Section 3.1.
5. You have to know the WAN and LAN addresses on the other end of the satellite link.

It also helps to have a diagram, even a crude one, of the network and its connections, showing relevant items like the local LAN connections, major devices, gateways, and addresses, preferably at both ends of the link.

In all three standard configuration modes described below you must start by connecting the SnIP "Console" DB9 connector to a PC running a terminal emulation program (like Hyperterminal on a Windows machine) set to a format of 38400, N, 8, 1 and VT100 emulation. This connection is via a standard DB9 male to female cable, which is supplied with all SnIP equipped modems. This control computer will be called the PC in the following descriptions.

1. Connect the SnIP Ethernet RJ45 connector to your local network (LAN) or into your operating system via a standard Ethernet RJ45 cable. The SnIP can automatically negotiate its rate and other parameters via AutoMDX.
2. Determine your desired mode of operation – that is, will it be a Bridge, Router or just for Monitor and Control. In all cases you will need to have an IP Address selected for the SnIP capable of operating within your existing LAN or operating network.
3. Start the PC and the terminal emulation program and insure the serial parameters are set to 38400, N, 8, 1 and VT100 emulation. No hardware flow control is used.
4. Turn on the modem equipped with the SnIP - the terminal output from the SnIP will begin immediately displaying lines of information as first the bootloader and then Linux itself loads. When completed a login prompt will be displayed. If you connect after the modem/SnIP is already running then hit "Enter" to bring up the login prompt.
5. Enter the login as "**root**". When asked for the password enter "**datum**". If the default password has been changed then you need to know the new one.
6. In most cases a basic configuration will have been set at the factory during test, but if the "configwiz" program starts automatically you should be ready to enter information initially, at least a name for the SnIP and the IP address. The remainder will be set below for each configuration. SnIP names could be anything descriptive like "snip3", or "Sat-Router", or "Honolulu". A location name might help identify it later. In the factory we normally preset the name to "snip-xxxx", where xxxx is the last four digits of the serial number.
7. You should also determine and have ready the Gateway address. A "Gateway" is the IP Address a computer, or the SnIP uses to gain access outside the existing LAN address range. For example in a simple home network the Gateway is the address for access to the internet if connected. In most cases you will need to know the Gateway address to program into the SnIP. You can usually get this address from any PC on the LAN.
8. The default "yes" for the Modem Polling option should be left this way unless you are operating as a Monitor Control only with no internal connection. The polling send regular messages to the modem internally as a keep-alive. It is not polling external modems.

The SnIP has default factory values for most of the necessary parameters and probably will not function on your LAN initially. Its default IP Address is 192.168.1.xxx and the default Gateway address is set to 192.168.1.1. Note also that Linux, and therefore the SnIP, are **case sensitive** for all command entries, so care should be taken in entry. Linux can also be very frustrating initially for Windows GUI users who are not familiar with command line operation.

You can view the current configuration settings by typing “**config**” at either the console or a telnet virtual console session. You also have all of the normal Linux command line utilities to provide configuration and operating information plus control.

About IP Addresses shown in this document - There is nothing special or magical about any of the IP Addresses used here. By convention and keeping with the recommendations of the IANA the addresses used are in the ranges 10.0.0.0 ~ 10.255.255.255 and 172.16.0.0 ~ 172.31.255.255 for most of the Network and LANs shown, and 192.168.55.0 ~ 192.168.55.255 for the WAN addresses on the satellite side since this would be considered a private network. You can change these in any way that suites your requirements. For example use 192.168.10.x for the WAN. Local LANs may also be shown in the 192.168.a.b range as this is common behind Gateways such as home cable or DSL modems.

Now, go to the appropriate SnIP function configuration type below for the remaining base setup. Remember that unlike any bridge or router applications you may see on the web or in documentation, this setup contains a satellite link and has two bridges or two routers as part of the overall connection. Both ends of the link must be set up for the whole link to operate.

1.0.2 How To Configure the SnIP as a Bridge Device

A “**bridge**” is a standard network device designed to connect two LAN segments with the **same** IP address range. It is an OSI Model Layer 2 device that only operates on Ethernet MAC Addresses. The SnIP’s IP Address is part of Layer 3 and is used for functions other than bridging, for example ftp, telnet/ssh control and the built in web server.

1. Start the configuration wizard by executing the command at the command line “**configwiz**” **Note:** Option selection below is not case sensitive
2. Select the “L” option for the LAN IP Address, and set your IP address.
3. Select the “N” option for Network Mask and set the Network Mask. The default of 255.255.255.0 can be selected by pressing return.
4. Select the “G” option if you wish the bridge to have a default Gateway IP address, and set it to the proper value. Normally in a bridge this is the same on both ends of the link.
5. Select the “M” LinkMode option, then enter “b” or “B” to select the bridge mode option. This sets the variables Bridge=yes and Router=no.
6. Select “S” for Save and enter to quit the configwiz program.

You will need to reboot the SnIP. Type “**reboot**” and hit enter. If you are only switching from Router to Bridge mode you will not have to reboot.

In network terms the bridge device br0 “enslaves” the two normally included eth0 and hdlc0 interfaces. The “iptables” and “ebtables” network programs are available to bridge operation.

Important! Bridge devices work by looking at responses to packets sent out. Therefore they WILL NOT WORK on one way links!

Note 1: See Section 11 for a current bug related to the bridging mode when the Ethernet connection is removed and replaced.

Note 2: When in bridge mode you should not set the modem into IF Loopback. Since the bridge learning may create a feedback mechanism that causes a data storm. Normal filtering typically prevents this however.

Note 3: The Linux Bridge application can only handle the Ethernet packet protocol. It is possible to use the tools provided to switch to a Cisco or other protocol, but the Bridge is incapable of using these.

1.0.3 How To Configure the SnIP as a Router Device

A “router” is a standard network device designed to connect two Internet segments with different address ranges. It is an OSI Model Layer 3 device that operates on TCP/IP Addresses. The SnIP Ethernet port’s IP Address is part of layer 3 and it must be within the locally connected LAN’s address mask. It is also used for functions other than routing, for example tftp, telnet/ssh control and the built in web server. This simple procedure below can set up a point to point router link. For point to multipoint systems see the more detailed descriptions later in this document.

1. Start the configuration wizard by executing the command at the command line
“**configwiz**” option selection below is not case sensitive
2. Select the “L” option for the LAN IP Address, and set your IP address.
3. Select the “N” option for Network Mask and set the Network Mask. The default of 255.255.255.0 can be selected by pressing return.
4. Select the “W” option for the WAN IP Address, and set the desired unique address.
5. Select the “t” WAN P-t-P Address option and enter the Point to Point IP address of the WAN hdlc port at the other end of the link. The other end is set to this unit’s WAN hdlc port address, thus cross connecting them.
6. Select the “G” option if you wish the router to have a default Gateway IP address, and set it to the proper value. Normally the link end with a web Gateway would use that device’s address and the other end would use the address of the hdlc0 port on this end. See Section 3 for more information.
7. Select the “M” LinkMode option, then enter “r” or “R” to select the Router mode option. This sets the variables Router=yes and Bridge=no.
8. Select “S” for Save and enter to quit the configwiz program.

You will need to reboot the SnIP. Type “**reboot**” and hit enter. If you are only switching from Bridge to Router mode you will not have to reboot.

Static routing applications get their routing rules from the “/etc/config/static-routes” file. Unless you are familiar with the Linux route command you should study its application on the web. An example of point to point router links is shown later in this document, Section 3.

Dynamic routing is provided via one of two methods: either the standard Linux built in router daemon, “**routed**” or by using the open source Zebra/Quagga router suite, which requires setup and configuration in multiple files. No information on the operation of Quagga is given here.

The SnIP also supports the powerful Linux “**iptables**” filtering, network address translation (NAT) and packet mangling functions. Like routing, it is best described as “knowledge intensive” and requires study of its operation. The web is a good place to find information on iptables.

Note 1: When set to Router mode the hdlc link protocol is automatically changed to Cisco HDLC protocol (CHDLC). This behaviour can be changed, but probably offers the most compatibility with other equipment.

Note 2: Any form of routing normally requires that a particular process variable named “ip_forward” be set to 1. See section 3.1.1.2 to insure proper setting.

1.0.4 How To Configure the SnIP as a Monitor/Control Device Only

This configuration is used only if the bridge and/or router functions are not desired. An example may be if a modem is using a different main data interface such as V.35 or RS-449, but the monitor and control functions of the SnIP are needed. A Snip has two main monitor control

applications built in. One is the command line “m500ctl” and “m49ctl” programs and the other is the web server at the SnIP’s IP Address. The SnIP’s IP Address is part of OSI layer 3 and must be within the connected LAN’s address mask. It is used for functions other than bridging or routing, for example ftp, telnet/ssh control and the built in web server.

1. Start the configuration wizard by executing the command at the command line
“configwiz” option selection below is not case sensitive
2. Select the “L” option for the LAN IP Address, and set your IP address.
3. Select the “N” option for Network Mask and set the Network Mask. The default of 255.255.255.0 can be selected by pressing return.
4. Select the “G” option if you wish the SnIP’s Ethernet port to have a default Gateway IP address, and set it to the proper value.
5. Select the “M” LinkMode option, then enter “n” or “N” to select the “None” mode option. This sets the variables Router=no and Bridge=no.
6. Select the “W” option, then press the enter key without entering any HDLC Address. This will prevent the SnIP from loading the driver module for this port since it is not used.
7. Select “S” for Save and enter to quit the configwiz program.

You will need to reboot the SnIP. Type **“reboot”** and hit enter.

Caution: there are two possible internal connections depending on the modem’s use. For the special applications with a SnIP, but redundancy on the main data connector, then the internal connection to the modem is missing the actual data and management interfaces. In that application the modem polling should always be set to “no”, the Link Mode set to “none” and the WAN hdlc0 address should also be set to a blank (no entry). These three items cause the modem to not poll and not load the internal interface driver – both necessary because there is nothing connected to these two ports. See Section 6.1.2 “A Unique m49ctl and m500ctl Case – Redundancy”.

1.0.5 How To Set Persistent Configuration Options

Configuration options are stored in multiple places in a Linux system, and can vary from one system to another. Within the SnIP standard and non-standard default configuration values are stored as follows:

- ❑ **/etc/config/system** Contains the options set by the configwiz program. To modify run configwiz and save the changes.
- ❑ **/etc/config/static-routes** Contains static routes and alternately the default gateway if not done in configwiz. To modify, use nano or vi on this file to edit.
- ❑ **/etc/sysctl.conf** Contains system control settings, and must set the “ip_forward”.flag to 1 for routing to work. See Section 3.1.1.2.
- ❑ **/etc/resolv.conf** Contains one or more DNS server addresses required for Internet operations. A sample can be created with “./mk-resolv”. To modify, use nano or vi on this file to edit.
- ❑ **/etc/config/iptables and iptables-config** Are the default iptables (filtering/NAT) data and configuration files respectively. These files must be set up by a user to work. To create or modify, use nano or vi on these files to create or edit.
- ❑ **/etc/config/Quagga, zebra, rip etc.conf** Contains the default dynamic route configurations for the Quagga, zebra, etc routers. To modify, use nano or vi on the appropriate file to edit the configuration.
- ❑ **/etc/profile.d/*.sh** Each shell script with an “.sh” extension placed in this directory is executed on power-up. To create, use nano or vi, then chmod 755 to set permissions. For example a common one that we provide by default is the “baseconfig.sh” script which might contain the following two lines:


```
export TERM=vt100
rdate -s 129.6.15.28
```

Before any changes to these configuration options will take effect, you will need to reboot the SnIP. Type "reboot" and hit enter. Switching between Router and Bridge mode is a special case and may not require rebooting if all other variables are correctly set.

There are two ways to switch between Router and Bridge mode. The normal method described above using the configwiz program results in the variables in the /etc/config/system file being modified which insures that the current configuration is restored on power up. You can also change modes using the "reconfig" command which will maintain the setting only as long as the SnIP is not rebooted.

If you create a persistent configuration that does not allow you to access the SnIP via the Ethernet port for Telnet or SSH, you have several options. One is to use the modem front panel to set a new ip address, second, the console management port may be accessed with a physically connected PC and last, see Disaster Recovery in Section 4.7 below.

1.0.6 Basic Linux and SnIP Operating Commands

Linux is a collection of many related individual programs that give it a lot of power. Unless you are familiar with each of these it is very difficult for someone new to Linux to easily and properly set up communications configurations. We have tried to make this a little easier by providing several basic functions that aid users in setup. Several of those are discussed briefly here:

Three main scripts are used to perform basic configuration. The 'configwiz' program has already been discussed and used above. Two more are the "config" command and the "reconfig" command. The "config" command shows the current settings contained in the "/etc/config/system" file maintained by configwiz, plus a few other system release numbers.

Changes made to configwiz are read on power on and are thus "persistent". The "reconfig" command allows changing the basic Link Mode type between Bridge, Router or None on-the-fly without storing changes in the system file. Thus changes made with reconfig are temporary until power off or a reboot. The configwiz program uses the reconfig script to change the LinkMode when it is modified.

Other important commands within Linux are the **ifconfig**, **route**, **iptables**, **ebtables** and **brctl** commands. You should read some basic information on each of these on the web or in other documentation.

To allow modem monitor and control from the SnIP, we include the "m49ctl" and "m500ctl" command line programs described later in this document. There is also a simple script named mdm-find in the root home directory to find modems attached to the SnIP's control ports.

The SnIP is not limited to specific functions related to the satellite modem. The full power of a modern implementation of Linux version 2.6 is available. Programming in bash scripting and Perl plus the many precompiled Linux applications from ping, traceroute, grep, rsync and wget to a fully functional web server are fully available to the user. Some minor limitations are introduced by compiling against uClibc and using "BusyBox", but most users will not see these.

We'll start with an often used and basic operation, which is to create a remote connection to a SnIP via TCP/IP called a "session".

1.0.7 How To Create a Telnet Session with the SnIP

A telnet session is a remote terminal connection to the SnIP via TCP/IP, either on the local Ethernet interface or over a link on the hdlc interface. It looks and acts just like a hard serial connection via the "Console" connection on the modem rear panel. It is also called a virtual terminal connection. Kernel messages are not shown on virtual terminals.

When a usable IP address is assigned, the telnet connection can be made from a Windows machine using a DOS Command prompt window or from a shell terminal window in Linux. The command is the same for either – **“telnet [ip address]”**. For example if the IP address is set to 192.168.150.210 then the command is:

telnet 192.168.150.210

The SnIP should respond with the login prompt and then the password entry:

The default Login is “root”, and the default password is “datum”.

Almost everything possible from a direct Console connection is also possible from a virtual session under Telnet. Please see more information concerning security and Telnet and SSH in later sections. Telnet sessions are very useful on the local LAN and as a remote connection as they do not require physically connecting a serial port to the SnIP’s console port, but provides the same capabilities. Telnet and ssh sessions can also be started using PuTTY from Windows.

Telnet sessions use the Linux command line interface and by default are established in the Bash shell. Once logged in you can type commands and get responses. For example you might try typing “whoami” which should give the response “root”. The Linux equivalent of the Windows “dir” command is “ls”, so to see the current directory contents type “ls” or to see the “/bin” directory type “ls /bin”.

1.0.8 How To Log into a Quagga Session

The Quagga/Zebra router has multiple ways to log into it. All except the first **“vtysh”** method below require that at least the **“zebra”** and one or more routing protocol daemons be running already. All protocols are turned off by default. To turn on individual protocol daemons you must go to the **“/etc/Quagga/daemons”** file and set the value to **“yes”** for zebra and at least one protocol. Then reboot the SnIP.

Most of these interfaces/sessions appear very similar to a Cisco IOS session, but it is not as **“unified”**. Help and command completion is almost identical to an IOS session also, so using the **“?”** and tab key can be used to show what is possible and automatically complete commands.

One interface is called the **vtysh**, short for Virtual Terminal Shell. You can log into that either from the console () or a telnet root session by typing **“vtysh”** at the prompt. You will normally not be asked for user authentication because you have already logged in as the root superuser.

The other login sessions are only reachable via telnet, and each router protocol must be configured for access via specific interfaces. The default interface specified is only the internal **“lo”** or loopback ip address of 127.0.0.1, meaning that you cannot telnet in from either the Ethernet or hdlc ports. Reading the Quagga documentation is very important in this respect. See Section 8.2.4 for more information on configuration of the Quagga/zebra suite.

The router protocols include zebra plus individual sessions for each running protocol. Each protocol has a unique access number associated with it and that number is appended to the telnet session invocation to log into that protocol. The list is shown below, and for example to log into the RIPD protocol (if it is running) on a SnIP with the IP Address of 10.9.3.22 you would type **“telnet 10.9.3.22 2602”**, assuming you had set the ripd.conf file to allow access from a physical port. From an existing SnIP telnet session you could also type **“telnet localhost 2602”**.

Protocol	Access	Notes
zebrasrv	2600	# zebra service (Not working?)
zebra	2601	# zebra vty
ripd	2602	# RIPd vty
ripngd	2603	# RIPngd vty (Only with IPv6)
ospfd	2604	# OSPFd vty

```

bgpd      2605      # BGPd vty
ospf6d    2606      # OSPF6d vty (Only with IPv6)
ospfapi   2607      # ospfapi (Unknown Not working?)
isisd     2608      # ISISd vty (Not in all versions)

```

The password, prompt name and other configuration for each of these protocols are determined by the individual configurations files in either “/etc/config” or “/etc/Quagga” as determined by your installation.

Quagga/Zebra like all routers are complex. It is highly recommended that you read the documentation available on the web at “<http://www.quagga.net/>”. We are currently using the latest stable version which is 0.98.6.

1.0.9 How To Create New User Accounts

As shipped the SnIP has only two users – the Linux “root” superuser that has access and permissions to modify or do almost anything that Linux is capable of, and the “quagga” user and group intended for control of the Quagga/Zebra routing suite. While that may be suitable for some applications, it is often desirable to create new users with limited access and permissions within the Linux structure. Linux uses the concept of “users” and “groups” of users, each with names and IDs - UID and GID respectively. The concepts and procedures for managing users and groups should be reviewed in Linux documentation or on the web. The basic commands used in this distribution of Linux are “**adduser**” and “**addgroup**”, with their complementary “**deluser**” and “**delgroup**” functions.

The standard locations for an individual user’s files are in the “/home/<UserName>” directory, which is currently blank in the SnIP. By default that is where the Linux “adduser” command will create a new user’s directory. So in its simplest form, if the root creates a new user with the command “adduser mike”, then after inputting mike’s password information, the program will automatically create a new directory named “/home/mike”. When mike logs in he is taken directly to the /home/mike directory. Note that the root user’s home directory is “/root”, and Quagga has no home directory.

Account passwords (including the root) are set or changed using the “passwd” command. For example, to change the Quagga user password for logging in as the quagga user you would type “passwd quagga”. The routine will also ask to repeat the entered password to insure that it is correct. Note that this is not the same as the router daemon passwords discussed later.

1.1 More Overview Information on the SnIP

The above are very brief introductions to setting up the SnIP. It is however a full computer, with only the video adaptor and a large hard disk missing. In addition to the bash interpreted command language, the SnIP also has the miniPerl version of the Perl programming language plus the full capabilities of Linux at the user’s disposal. Following is more information on the hardware and software present in the SnIP. The full name of this device is SnIP875, indicating its use of the MPC875 Freescale processor at its core.

All of the standard software currently on the SnIP in the Linux implementation is open source, under the GPL license. The user is permitted to modify or use this software as they see fit without limitation. The setup to compile this software however is quite complex using multiple cross compilers. See the technical information at the end of this document for more information on the software and applications plus the locations and compilers/libraries used.

A brief hardware description follows:

1.1.1 Interfaces

The SnIP has 3 basic interfaces which are discussed, the Control Port, the Ethernet TCP/IP Port and the internal Synchronous Serial Port. Other interfaces are listed here but not discussed in this guide:

Rear Panel Interfaces -

- **“RS-232 Control Port - DB9”**– In normal operation the rear panel DB9 has a standard RS-232 Console port used as the main tty interface to Linux (and U-Boot). The port is set up to operate at 38.4 kbps, N, 8, 1. Connect this to a Windows PC running Hyperterminal or a Linux PC running a terminal emulation program such as gtkTerm. It shows the boot status and allows logging in as root locally. Log in as “root” and use the password “datum”. Once set up the SnIP may be logged into remotely over the network using Telnet or SSH with the same password. Named “ttyCPM0” in Linux.
- **“RS-485 External Control Port – DB9”**– The SnIP can talk to other modems locally by running a daisy chain RS-485 connection from this port, as controller, to the other modem’s J6 Remote Control connection. By default this operates at 9600, N, 8, 1. Named “ttyU1” in Linux. Since this shares the same physical connector as the RS-232 Control Port, then a break out cable is required to use both simultaneously.
- **“Ethernet / TCP/IP Port – RJ-45”** – Standard 10/100 Ethernet connection with auto-mdx functions. Named “eth0” in Linux.
- **“USB Control Port – USB type B”**. Only very partially implemented in U-Boot and not at all in Linux.

Internal Interfaces -

- **“Modem Synchronous Serial Data Port - HDLC”** – or High-level Data Link Control is a bit oriented synchronous data link layer protocol. HDLC is commonly used for Wide Area Network (WAN) communications protocols at the OSI model layer 2. It may encompass both the physical and electrical layers. It is a packet based structure which includes opening and closing flags, addresses, a checksum and a “bit stuffing” method to insure that the flags do not occur within the packet itself. There are several commonly used forms of HDLC in use, mainly the standard and a version modified by Cisco with a corresponding Cisco header. The HDLC protocol in various forms is used on the satellite link connection between SnIP cards. Depending on the driver module loaded this interface is named either “hdlc0 ” or rarely “eth1”.

The standard HDLC port driver is called snip_gen_hdlc, named because it can attach itself to a common set of Linux generic HDLC control routines. A user program named **“sethdlc”** permits control of the protocol used by this driver.

- **“Modem RS-232 Internal Control Port”** . The modem is monitored and controlled via this interface built into the internal 40-pin ribbon cable between the SnIP and Modem. The port is the same one used on the normal rear panel 37 pin serial data interface, and therefore precludes using the standard Y cable for redundancy. This port operates at 9600, N, 8, 1. When enabled the SnIP runs a small “daemon” that sends “keep-alive” packets on this port to let the modem know that all is well with the SnIP. This port is referred to as “ttyU0” in Linux. By default this port operates at 9600, N, 8, 1.
- **“Compact Flash Socket”** . Not fully developed yet. Its intent is to add additional disk drive like storage and allow specialized software/hardware addition.

In Monitor/Control applications, the ports used are the RS-232 Local Control Port, The RS-232 Internal Modem Control Port and the RS-485 External Control Port. The rear of the modem and the external ports are shown below.



LAN 10/100
Base T Auto
MDX Port

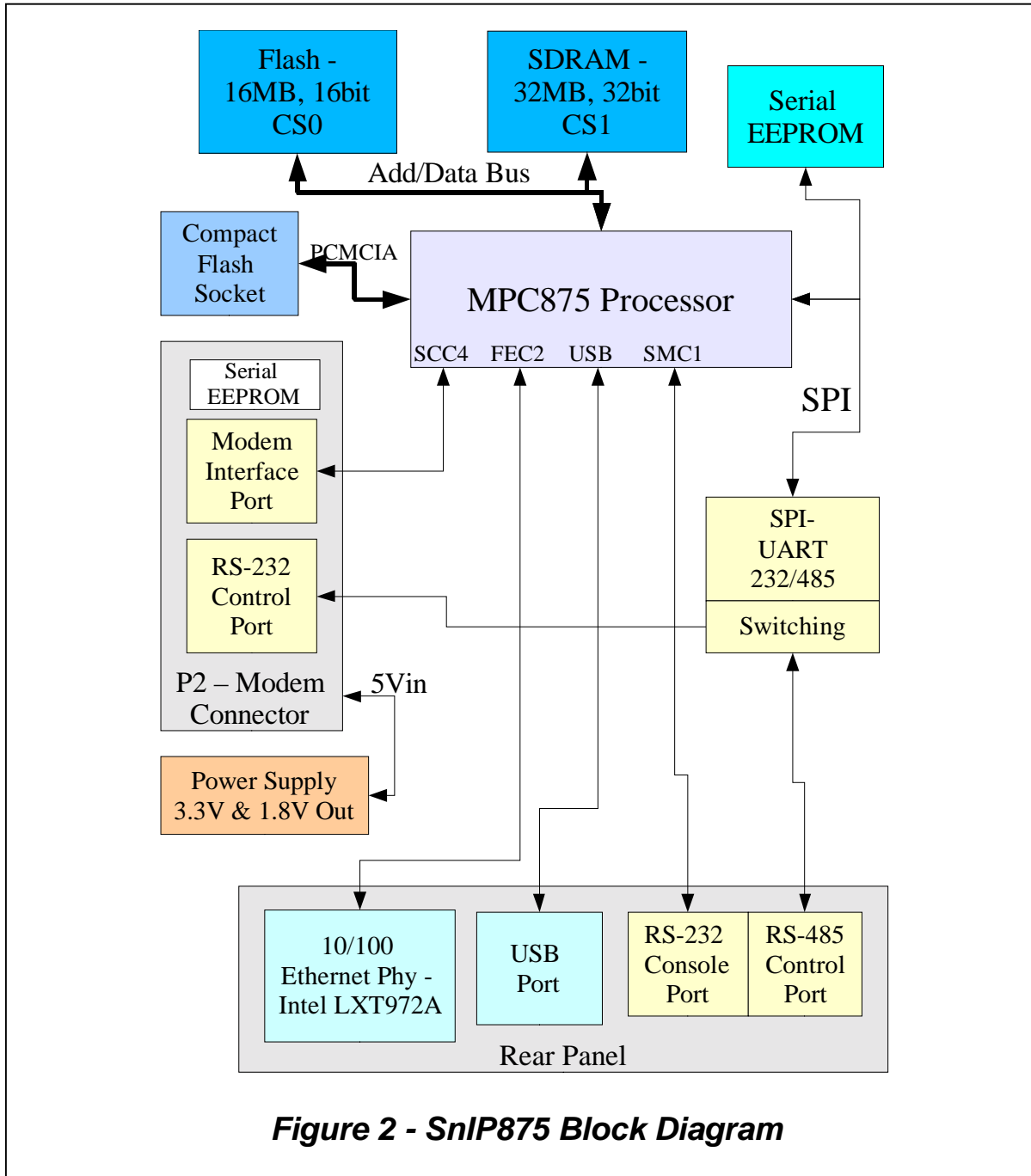
Figure 1 – SnIP Rear Panel Connections

RS-232 and
RS-485
Control Ports

On the lower edge of the Ethernet LAN connection are two common LED indicators showing “link” status and connection speed. When both are lit then a link has been established with the other end of the cable and the speed is 100 Mbps.

The 3 vertical LEDs show (from the top) Ethernet traffic activity, USB enable and Control Port currently in RS-485 mode when lit.

Below is a block diagram of the SnIP showing its main components and interfaces:



1.1.2 Compatibility

The SnIP described here is only for use within either M5 or M500 series Modems. It can be added to any of the M5, PSM-4900 modems including the 70 MHz, Hybrid and L-Band versions. It can also be added to any of the M500, PSM-500 series modems including the 70 MHz, Hybrid and L-Band versions. It will work in modulator only and demodulator only versions of these modems, although obviously with limitations.

It is potentially possible to use the SnIP outside of a modem. For use as an external bridge or router it would need a power supply and appropriate driver/receiver ICs for the desired Synchronous Serial Protocol, i.e. RS-449 or V.35. As a Datum Systems modem controller it only needs power and RS-485 connection to modems to be controlled. Some of the control lines on the modem interface may need to be set via hard connections in this case.

1.1.2.1 Protocol Compatibility

When a SnIP is set to the standard bridging application it loads the generic HDLC driver and sets the protocol to "Ethernet encapsulated". This is the same protocol used by the previous generation "SDMS" card with bridging software loaded. Therefore two link ends, one with an SDMS and the other with a SnIP will interoperate without anything else required.

A SnIP set to routing configuration will by default load the generic HDLC driver and set the protocol to "Cisco" or CHDLC, This should allow the SnIP to link properly with either another SnIP in the same Link Mode, the older SDMS with router code installed or a router on the far end connected to the modems's standard synchronous serial data port. The latter 2 are not guaranteed as it has not been fully tested yet, but currently these seem to work.

1.2 Overview – Initial Setup

The SnIP875 is a small low power computer running a fairly standard embedded Linux operating system (currently Version 2.6.15). It has two specialized interfaces that most common computers do not have. Each of these interfaces has custom drivers created and compiled into Linux to allow their use directly by Linux.

First and most important is a high-speed synchronous serial port that connects internally to the same port that appears on the rear of the modem as the "Data Interface". This port has the Linux name "hdlc0" by default, but may be termed "eth1" if an alternate Ethernet encapsulated only driver is installed. The interface logic is low voltage (3.3V) TTL, capable of tolerating 5V TTL. The physical interface to the modem is buffered, and tri-stated when not selected by the modem since this interface is shared with the modem's normal rear panel data interface transceivers.

The second unique port is a switchable RS-232/RS-485 communications port. When set to internal it talks to the modem as a remote control device. When switched to external it talks on 4 pins of the rear panel DB-9 connector as a 4-wire RS-485 control port, which can be used to control other modems within a station. The internal modem control connection is a standard Linux serial port named "ttyU0", while the external 485 port is named "ttyU1".

The Linux operating system allows the logical connection between the SnIP's two data Interfaces to be operated in many modes depending on the application, but basically they come down to either "Bridge" or "Router" applications. Bridge mode is the default for the SnIP unless the configuration is changed as shown earlier, and described in more detail later. It is also possible to use the SnIP as a modem controller for 1 or more modems in a single location. Linux has a wealth of built-in and external programs available for operating bridges and routers. The standard low level bridge control is via Linux "brctl" and the low level router control is via "route", both built into the kernel itself.

The bridge is a very straightforward and simple device to use. It requires minimal setup for normal operation. The SnIP Ethernet Bridge converts the Ethernet packets to use a basic HDLC packet structure over the satellite link. The modems at both ends of the link must be either SnIP equipped or one end can be SnIP and the other the older "SDMS" with bridge code installed. These instructions naturally only apply to the SnIP and not the SDMS. The modems on the link must be set up in a complementary fashion. That is, whatever parameters are set on the transmit at one end of the link must also be set the same for the receive on the other end.

- Choose a mode of operation suitable for the service desired. In most cases the standard settings of QPSK with the Viterbi or Turbo Product Codes FEC at rate $\frac{1}{2}$ or $\frac{3}{4}$ is suitable.
- Set the data rate based on the expected level of traffic. In the M5 modems, at rate $\frac{3}{4}$ they are capable of up to 4.92 Mbps operation, while the M500 series is capable of 29.5 Mbps operation if equipped with the proper “Feature Set” and mode settings. Suitable performance may be achieved at a modem data rate of 256 kbps or less for some applications. The data rate setting should be determined by the systems engineer. Since TCP/IP and Ethernet are basically asynchronous packet protocols, the use of a lower bandwidth channel limits data flow when multiple users are making constant demands on the channel. See the information in Sections 2.1 of this Appendix for more information on refining the settings to help optimize them for a particular application.
- The SnIP interface is unaffected by and compatible with all of the FEC and Multiplexer Options available in either series of modems. The SnIP is however capable of operating as an alternate AUPC channel for systems requiring that function.
- Connect the SnIP 10/100 BaseT interface on each link end to the appropriate network via switch or router interfaces.
- Two basic settings are required to use the SnIP.
 1. Both ends must enable the SnIP interface. This is done using the modems front panel interface and setting the **<Interface: I/O - Mode>** parameter to “SnIP”. This is currently option number 8.
 2. The IP address of the SnIP card should be set to reside in the system where it is used. The IP address is set at the **<Interface: SnIP – IP Addr>** parameter. The default for this parameter is 192.168.0.0.
Since the bridge function is by definition a layer 2 device, it does not work on IP Addresses. This parameter is only used to control the bridge from the Ethernet connection itself. Most systems however will require an IP address within its local range.

1.2.1 Overview – First Time Startup and the Configwiz Script

Your SnIP may have already been initialized, especially if it has a strictly bridge or M&C function.

The first time that an un-initialized SnIP is turned on it has no basic configuration files. To set up the SnIP properly you should connect a terminal such as a PC running Hyperterminal or a Linux computer running a terminal emulation program such as gtkTerm. Connection is via a standard 1:1 DB9 male to female cable between the computer and the SnIP's rear panel DB9 connector. Set the program to operate on a COM port at 38400 bps, 8 data bits, No parity, and 1 stop bit. This is commonly referred to as “38400,N,8,1”. An uninitialized SnIP automatically runs a small configuration program named ‘configwiz’. To run the program if the configuration has already been set up, or to re-run it, use the command line entry “configwiz”. If the SnIP is already running you will have to log into the console. At the “user” prompt type “root”, then at the password prompt type the default “datum”. This can be changed later, but it is the default password.

Normally the configwiz script has already run and set the IP Address, mask and other basic elements to defaults. Therefore to set the IP address for any use including Telnet or SSH control the script will have to be re-run by logging into the console and running the configwiz script.

Answer the questions in the configwiz program, specifically setting the IP Address to a value suitable for your network. Then use the “s” option to save the configuration. The program will automatically create a file “/etc/config/system” that contains the basic information entered, plus a few more entries.

One of these added entries is the “LinkMode” line which defines whether the SnIP is set to Bridge mode (default) or Router mode or None (used when only a M&C function is desired). The BRIDGE setting determines if the bridge initialization script is run on power up, setting up a basic bridge by putting both the eth0 and hdlc0 interfaces into a new bridge named br0. br0 takes on the MAC and IP addresses of the eth0 interface allowing it and the web server to be accessed.

If you do not wish the bridge to be set up on power on, you must edit the file “/etc/config/system” with either the configwiz program or an editor like nano to modify the LINK_Mode, BRIDGE and Router lines. The preferred method is to run the configwiz program script and select the desired LinkMode option.

Either the configwiz script or editing the system file directly with the “nano” or “vi” editor can also be accomplished via a telnet session. When running the nano editor from a PC Hyperteminal direct console connection you may first have to set the “TERM” type environment before nano will run properly. The command is “export TERM=vt100”. Then you can use “nano /etc/config/system” to modify the file and save it. On the next reboot the changes will be in effect.

Note : If you have modified the operating configuration using the “reconfig” command, the config and configwiz programs may not properly display the current operating configuration. They will show the power up configuration.

1.2.2 Overview - Modes of Operation

There are a variety of functional modes afforded by having a fairly complete Linux based processor managing the connections between the rear panel user accessible connections and the satellite side internal connection. In this document we focus on only the two most common functional processes; Bridging and Routing. In bridge mode there is only a single function available with the SnIP. That is to perform standard bridge functions of filtering and forwarding packets based on the MAC Address.

Bridging is an OSI model layer 2 (link layer) function, working with MAC addresses. The bridge functions include dynamic production of a “hash” table used to determine which Ethernet addresses should stay local and which should be passed over the satellite link.

Packets transferred over the satellite link are first converted to HDLC format packets. At the far end the receiving SnIP converts the packets back to the original Ethernet packet. For those packets transferred the satellite link is transparent except for the introduction of a 250 mS path delay. This delay can be significant in some applications. There are many different types of HDLC packets in use today and those used by the SnIP are intended to be compatible with specific formats. The basic format is that generated by Motorola QUICC processors.

The HDLC protocol port to the modem has no driver installed into the kernel. It is loaded as a module, so that it can be changed for possible different service types. Initially two drivers are written for this port. Note that the standard and default driver loaded is the second one listed here which can function in either Bridge or Router modes.

- ❑ The first is strictly Ethernet encapsulated with an HDLC structure, and is named “snip_raw_hdlc”. This is the basic structure used by an SDMS card with bridging software loaded. You would load this module using the command “**modprobe snip_raw_hdlc**”.
- ❑ The second is named “snip_gen_hdlc” and is a more sophisticated driver intended to attach to Krzysztof Halasa’s generic HDLC support routines for Linux. These routines are commonly used in X86 PCs running Linux with one of several PCI type serial synchronous cards (for example Sangoma). You would load this module using the command “**modprobe snip_gen_hdlc**”. It supports several protocols including “raw”, “eth-raw”, “Cisco” and “frame-relay”. To set any of these protocols you use the separate “sethdlc” command. It has minimal usage instructions included in the program.

Either of these hdlc driver modules can be loaded and unloaded, but unless you are very knowledgeable about Linux and drivers it is suggested that you use the standard snip_gen_hdlc driver.

The configwiz, reconfig and bridge start/stop scripts currently uses the generic hdlc module, but could use the raw module as well.

There are several not immediately obvious restrictions on use of HDLC drivers and protocols. There are two main protocols used over the satellite's synchronous serial port, either "Ethernet" or "Cisco". The bridge function **must** use the Ethernet protocol. The Linux bridge is not capable of handling packets which contain the Cisco protocol headers. The Router mode however could use either, but is standardized on the Cisco protocol since that is intended to allow connection to other equipment types that use this protocol. There are three specific cases as minimum. First is if an SDMS card with routing software loaded is on the other end of the link. Second is if no Ethernet interface was on the other link end, instead with a WIC card attachment to a Cisco router. And the third are some systems already built using Linux PC based routers with Sangoma type synchronous serial PCI interface cards.

2.0 SnIP875 Ethernet Interface Setup and Operation

The addition of the SnIP Ethernet interface card to the PSM-4900 or PSM-500 modem provides an additional selectable interface to the existing complement of synchronous serial interfaces already available at the modem front panel I/O controls. This Ethernet interface combines two new standard types: a Packet based Ethernet physical interface with TCP/IP and bridge functions or a TCP/IP interface with routing functions.

2.0.1 Multi-Purpose 9 Pin "D-Sub" Female Connector RS-232 Console and/or RS-485 Control

The SnIP uses a single connector for both an RS-232 Console connection and a RS-485 Party Line control output. It is a DB-9 female connector labeled 485/Console Cntrl. Both connections can be made simultaneously when connections are made via a "break out" cable. Although not always necessary this cable isolates the 485 connections from a local RS-232 to computer connection, and isolates the 232 lines from local RS-485 control connections.

The SnIP's internal Linux operating system is set up with 3 "tty" devices normally available with the following fixed connections:

- ❑ **ttyCPM0** – Serial RS-232 Console port to the rear panel Console port.
- ❑ **ttyU0** – Serial RS-232 control port to the internal modem control lines.
- ❑ **ttyU1** – Serial RS-485 control port to the rear panel 485/Console port.

The purpose of RS-232 ttyCPM0 connection is to provide a local console port used by either a terminal, or a PC acting as a terminal by using software such as Hyperterminal, Procomm or UUCP and gtkTerm on a Linux machine. Some literature may also call this a "management" port and is considered secure because its use does not involve the internet. The default connection is 38,400 bps, 8 data bits, no parity and 1 stop bit. ("38400,N,8,1"). It is the default console device for both the U-Boot bootloader and Linux. All standard messages and errors use this port.

RS-232 Terminal 9 Pin "D-Sub" Female Connector



Rear panel connector labeled **485/Console Cntrl**

RS-232 DB9 Female Terminal Connector Pin Assignment	
Signal Function	Pin #
Ground	5
Transmit Data (Output)	2
Receive Data (Input)	3

The female connector and wiring are such that a standard DB-9 male to female cable can be used to connect the SnIP to a PC's standard COM1,2 (Windows) or ttyS0,1 (Linux) serial communications ports. If the external RS-485 port is not being used no provision is required to block any of those signal which might interfere with the control lines in common use.



Rear Panel connector labeled **485/Console Cntrl**

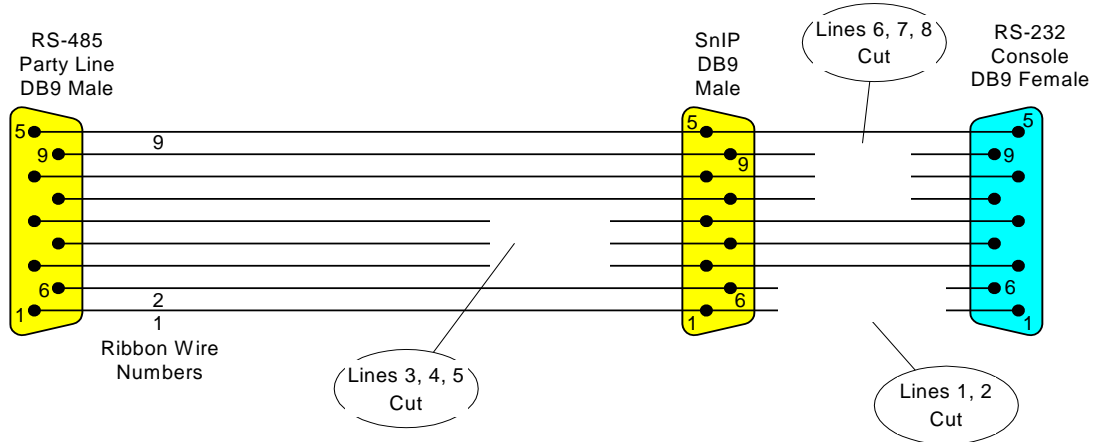
RS-485 DB-9 Female Connector Pin Assignment	
Signal Function	Pin #
Transmit + (Output)	8
Transmit - (Output)	9
Receive + (Input)	1
Receive - (Input)	6

The RS-485 connection is available for special programming running in the SnIP to control external RS-485 devices. It uses the same UART device as the internal ttyU0 connection to the modem, which is not a problem because both ports only initiate traffic under control of the SnIP's processor. It is not intended for or is usable as a control input. It can however be linked to multiple other modems set into RS-485 mode, all controllable by the SnIP.

2.0.2 Console and RS-485 Connections

In many applications the only connections to the SnIP's rear panel DB9 connector is for initial or occasional console connection to a computer running a terminal emulation program. In this case the cabling required is a simple DB9 Male to Female 1:1 cable.

If the SnIP is being used for Control applications via the rear panel DB9's RS-485 connections then additional wiring may be required. Specifically a "breakout" cable is used to separate the RS-232 and 485 lines so that interference is not possible. The diagram below shows the fabrication of such a cable and the wiring diagram of a typical connection.



SnIP RS-232/485 Control Breakout Cable Fabrication
Using Ribbon Cable and IDC Connectors

Figure 3

In this figure, note that several wires are cut. This isolates the RS-232 from the RS-485. The left hand male connector could be 10 or 20 connectors all in parallel on the ribbon cable for connection to multiple M5 or M500 Modems.

The wiring diagram below shows the connections made by such a cable between various devices. Using this scheme a single SnIP can control all the normal modems in a station via the Ethernet/TCP/IP connection.

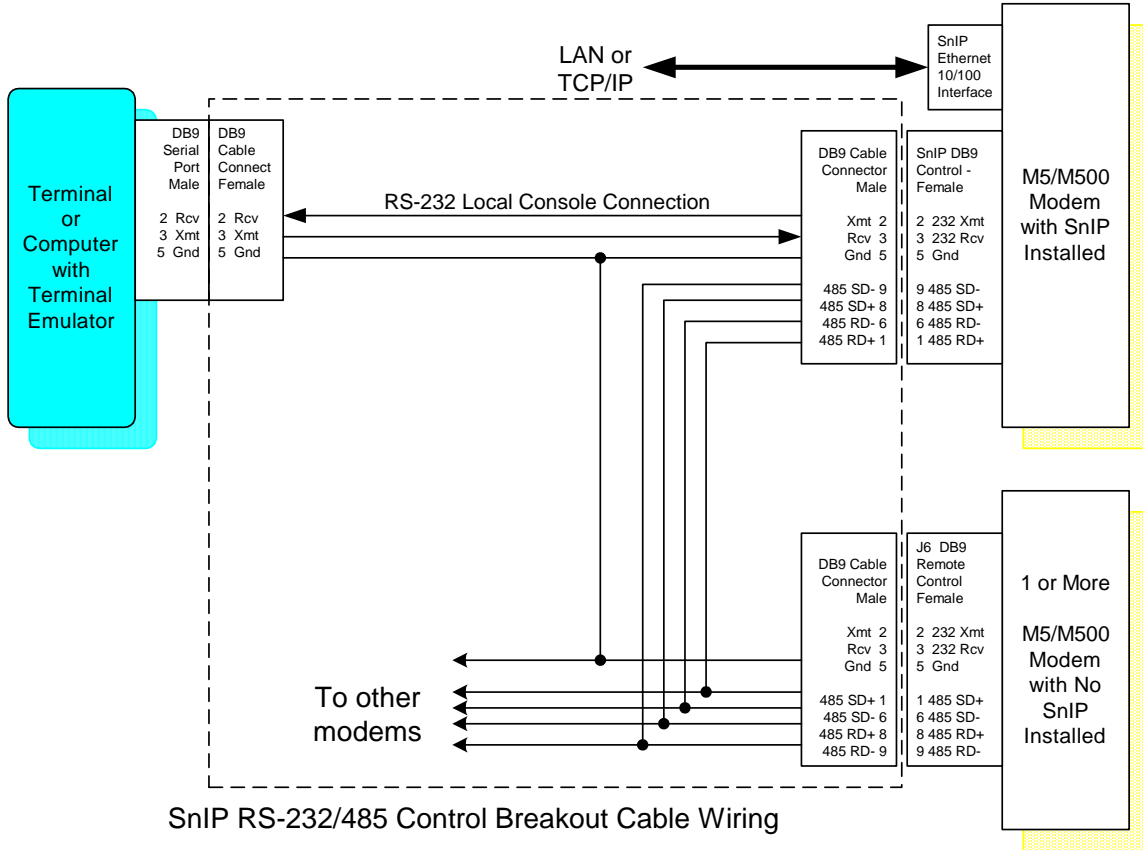


Figure 4

2.1 Console/Terminal Mode

The SnIP in an M5 or M500 modem can be interactively configured in the terminal mode from the rear panel Console DB9 connector. This is the Linux terminal console, and the root user must sign in and present the proper password before being granted access. Once logged on, the root user has the full power of a command line Linux system 2.6 available.

The SnIP's command set and the whole root filesystem were created and compiled against uClibc, not the standard Glibc used on computers with more memory available. Using uClibc significantly reduces the root filesystem size.

Most of the common Linux commands are implemented in "BusyBox" which simply eliminates most of the rarely used functions and switches that have little meaning in an embedded system. You can see a list of the commands by doing an "ls" in the /bin, /sbin, /usr/bin and /usr/sbin directories. An example Linux command output is shown below for the "ifconfig" command, which shows and controls the interface configuration.

Typical terminal output display in Bridge Mode - in this case an "ifconfig" command:

```
[snip2:0 ~] ifconfig
br0      Link encap:Ethernet  HWaddr 00:19:78:00:00:05
         inet addr:192.168.15.82  Bcast:192.168.15.255  Mask:255.255.255.0
         inet6 addr: fe80::219:78ff:fe00:5/64 Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:133094 errors:0 dropped:0 overruns:0 frame:0
         TX packets:97809 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:7567549 (7.2 MiB)  TX bytes:13748125 (13.1 MiB)

eth0     Link encap:Ethernet  HWaddr 00:19:78:00:00:05
         inet6 addr: fe80::219:78ff:fe00:5/64 Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:17069 errors:0 dropped:0 overruns:0 frame:0
         TX packets:35002 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:2383301 (2.2 MiB)  TX bytes:5720376 (5.4 MiB)

hdlc0    Link encap:Ethernet  HWaddr 00:01:B6:C4:7F:DE
         inet6 addr: fe80::201:b6ff:fec4:7fde/64 Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:144663 errors:0 dropped:1 overruns:0 frame:1
         TX packets:114852 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:13488107 (12.8 MiB)  TX bytes:16078180 (15.3 MiB)
         Base address:0x3f00

lo       Link encap:Local Loopback
         inet addr:127.0.0.1  Mask:255.0.0.0
         inet6 addr: ::1/128 Scope:Host
         UP LOOPBACK RUNNING  MTU:16436  Metric:1
         RX packets:6940 errors:0 dropped:0 overruns:0 frame:0
         TX packets:6940 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:201260 (196.5 KiB)  TX bytes:201260 (196.5 KiB)

[snip2:0 ~]
```

Notice in this output that there are four interfaces shown, br0, eth0, hdlc0 and lo. This was displayed immediately after boot up with static routes resulting in bringing the hdlc0 physical interface up and the LinkMode set to bridging. If this were in router mode the hdlc0 line would be significantly different with "unspec" Link encap and a string of 00-00... for the HWaddr. The

interface named “**br0**” is the bridge device. If the driver for `hdlc0` had not been loaded, then the `hdlc0` entry would not appear. If bridge mode had not been established then the `br0` entry would not appear. That is one way to determine what driver or LinkMode, if any, has been loaded for the operating mode and the modem’s data port. If the `hdlc0` driver was loaded then the protocol used can be changed with the “`sethdlc`” command line function.

The `ifconfig` command shows the current operating condition, not the persistent configuration that will be loaded on power up, although they are normally the same.

2.2 Telnet Terminal Mode

Telnet is a basic operating protocol via the internet or local LAN that permits command line control of equipment. The Telnet mode will look virtually identical to operation of the SnIP from the local RS-232 console port. This mode is very useful for most local LAN applications.

WARNING: Basic Telnet operation is minimally secure. It asks for a default or user defined password. Its security is suitable within a controlled LAN environment. SSH is supported by the SnIP and provides a higher level of security. It is recommended for any application involving connection via the Internet itself.

Telnet mode requires that a previously setup IP address is already configured.

The Telnet Mode allows all of the control of the local console connection except from any computer that can address this SnIP’s IP address. When the bridge mode is enabled the bridge takes over the Ethernet port’s IP address, allowing telnet sessions to be set up. The Telnet session can also be established from either the LAN or Satellite port sides of a link.

When a usable IP address is assigned the telnet connection can be made from a Windows machine using a DOS Command prompt window or from a shell terminal window in Linux. The command is the same for either – “`telnet [ip address]`”. For example if the IP address is set to 192.168.150.210 then the command is:

```
telnet 192.168.150.210
```

The SnIP should respond with:

```
[Hostname] Login:
```

```
Password:
```

```
The default Login is “root”, and the default password is “datum” .
```

If the password is changed it is the responsibility of the administrator to keep track of passwords. The password can be reset from the Terminal mode with connection to the DB9 “Console” connector.

With some Windows browser versions the invocation of the URL address of:

```
telnet://192.168.1.120/
```

will automatically launch the “Hyperterminal” communications program that is part of many Windows installations. The Hyperterminal display will mirror that of the DOS command prompt window.

Multiple user “sessions” are possible simultaneously. Part of the telnet command prompt tells which virtual terminal is connected.

When the telnet session is no longer needed the user can sign off using the command “`exit`”. The SnIP will then close the connection.

Telnet sessions can also easily be set up from a Windows machine using PuTTY.

2.3 Open SSH Terminal Mode

Although Telnet is simple and does not ask for a password, it is not very secure. Consider that on the Internet the password that you use to log into a Telnet session is sent in the clear. The SnIP supports the common “Open SSH” secure protocol that encrypts everything, including the login. When the SnIP is started the first time it creates the keys necessary to run ssh. The entire Open SSH suite includes multiple applications including also **scp** for secure copy and others.

SSH remote login mode requires that a previously setup IP address is already configured.

Once logged in via ssh the console connection looks similar to a telnet or local console connection. It allows control of the basic SnIP card itself and its configuration. Control of the modem is provided by the “m500ctl” or “m49ctl” application in the basic software set of the SnIP.

When a usable IP address is assigned the ssh session can be made from a Windows machine using a DOS Command prompt window or from a shell terminal window in Linux. The command is the same for either – “ssh -l root [ip address]”. For example if the IP address is set to 192.168.150.210 then the command is:

```
ssh -l root 192.168.150.210
```

The switch after ssh is the lower case “l”. The SnIP should respond with:

```
root@192.168.150.210's password:
```

The default password is “datum”.

If the password is changed it is the responsibility of the administrator to keep track of passwords. The password can be reset from the Terminal mode with connection to the DB9 local terminal connector.

Multiple user ssh and ssh/telnet “sessions” are possible simultaneously.

When the ssh session is no longer needed the user can sign off using the command “exit”. The SnIP will then close the connection.

SSH sessions can also easily be set up from a Windows machine using PuTTY.

Initial Configuration

The local Terminal or remote Telnet or SSH modes can be used to completely control the PSM-4900i setup and operating parameters. The Front panel keys initially configure the units network IP address, subnet mask, and transmit and receive transmission parameters.

2.4 Setting The Unit IP Address

To set the IP Address temporarily you use the normal Linux methods. “ifconfig” will change an address if it is not in use. If you are running in bridging mode then the eth0 IP Address has been set to 0 and br0 has taken on what was originally eth0’s address. To shut down the bridge, the command “bridge stop” should work.

To set the IP address so that it will automatically start with that address use the “**configwiz**” script. It will set the value in the etc/config/system file. Changes of this type are permanent but require that the SnIP be rebooted.

2.4.1 Setting IP Address and Mask from Front Panel

Limited monitor and control of the SnIP is available using the Modem front panel. The front panel can tell you if a SnIP is installed and enabled plus its version, serial number and operational status. The only controls aside from selecting the SnIP as the current data interface are the ability to set and read the LAN IP Address and LAN Net Mask. Setting either of these values from the front panel will result in a reboot cycle in the SnIP.

3.0 Basic Operation Setup

There are two main operation modes, Bridging and Routing. Both of these defined functions require at least two interfaces to be present. In the SnIP these are the eth0, RJ45 type 10/100 Ethernet port and a second high speed synchronous serial port, hdlc0 internally connected between the SnIP card and the modem. The classic basic protocol used for this serial synchronous port is "HDLC".

We currently have two Linux drivers for this port implemented as modules, meaning they are not built into the Linux kernel itself, but are loaded or unloaded on demand. Unless the LinkMode is set to "None" and the hdlc0 address is defined as null (blank) then the standard generic hdlc port driver is loaded by default. **Any alternate driver must be loaded by the Linux "modprobe" command.** This is normally done by software based on the type of operation selected. The two drivers available currently are "snip_gen_hdlc" which has the ability to bind with some standard Linux "generic" protocols for Ethernet or Cisco type encapsulation, and a simpler "snip_raw_hdlc" driver that only uses the standard Ethernet encapsulation. This second type is fully compatible with the former SDMS Ethernet interface in bridging mode. These drivers may change in the future.

3.0.1 Linux Startup Note

Linux manually starts processes and services by typing commands in a terminal window. These same processes can be automated to start services desired at power on by scripts. The startup process, scripts and locations vary from one Linux distribution to the next. Most of the SnIP procedures are common to Debian type systems.

3.1 Basic Network Topologies

Having read some about the SnIP and its capabilities plus Linux and the incredible range of functions and procedures that it can accommodate, you are possibly confused about how to set up your modem and the SnIP. All of this capability and complexity is aimed at meeting some basic connection types via satellite that are generally termed "Network Topology". There are currently only a few topologies that are needed to accomplish common tasks. These topologies come down to two main types:

- ❑ Point-to-Point Links with two main variations – Bridge or Router.
- ❑ Point-to-Multipoint Links with two main variations – Managed and Un-Managed.

3.1.1 Point-to-Point Links

These are the simplest type of links. Whether a bridge or a router SnIP configuration (LinkMode) is required depends on what is on each end of the link as follows.

3.1.1.1 Bridging

If the two ends of the link are intended to connect two segments of the same Network LAN with the same IP Address range then a Bridge is the proper configuration. For example if a main office has one or more remote offices and you want them to all be on the same LAN then you use a Bridge. An example is shown below.

In this bridging example note that the IP addresses of all the devices (except the WAN side of the SnIP) are within the same IP Mask range (we assume a Mask of 255.255.255.0). The WAN and the LAN side port IP addresses of the SnIP are actually set to 0.0.0.0 by the bridge, and it is the bridge device, br0 that has the IP Address shown on the SnIP below. Both SnIPs use the Gateway address of 192.168.1.1. DHCP, assumed to be provided by the DSL modem, works on both sides of the satellite link. This is a very easy set-up.

There is also no other information required to tell the bridge to take Ethernet packets not destined for the local end to be sent to the WAN port. The bridge does that automatically.

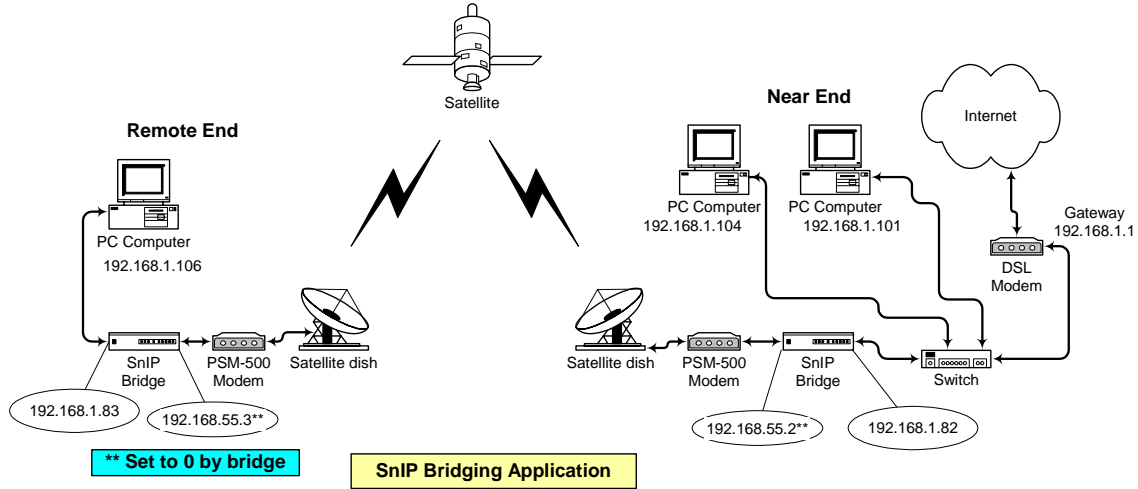


Figure 5 – Point-to-Point Example Bridging Diagram

You could literally take an existing LAN all connected by Cat 5 cable and insert a pair of SnIP equipped modems in Bridge LinkMode in place of one of the cables and everything should continue to operate as before. The individual computer configurations would not have to change and functions like DHCP will continue to operate as mentioned above.

3.1.1.2 Routing

If the two ends are part of different networks and are intended to be isolated other than the transmission facility provided by the satellite link then a Router is the proper configuration. For example, two major facilities that want to connect over a satellite link, but maintain their own addresses would use a Router type link. An implementation of a common example of a Point-to-Point “Closed” router link is shown below.

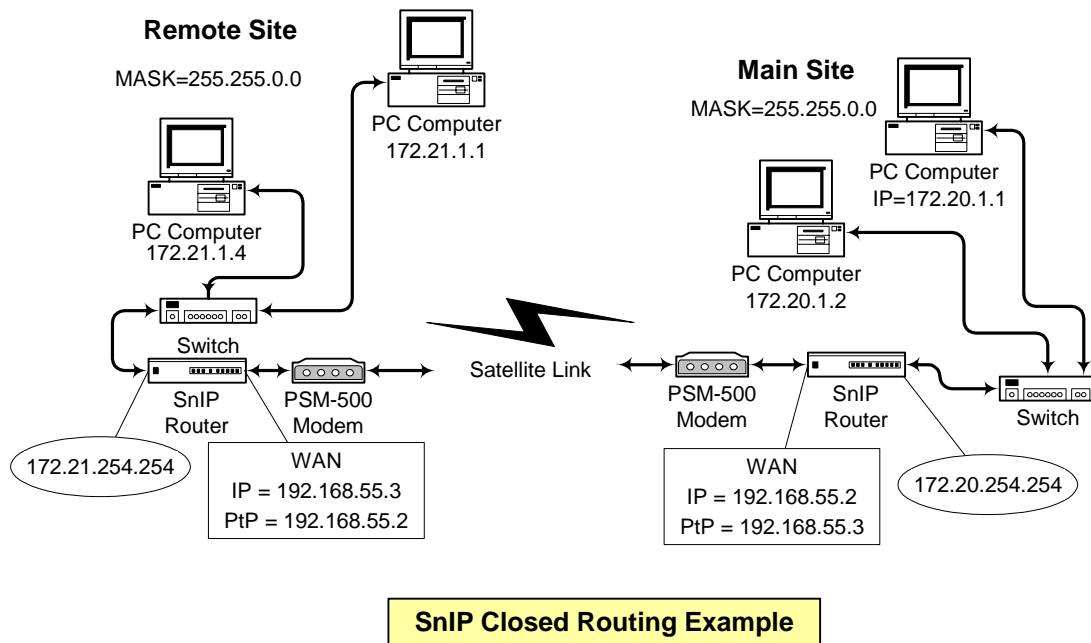


Figure 6 – Simple Point-to-Point Closed Example Routing Diagram

In this example note that the two LAN networks on the two link ends are not within each others masks. Specifically note that the WAN Point-to-Point addresses are in use and each end points to the WAN IP Address of the other end. We have also removed a lot of the redundant pieces such as the satellite and antennas that cluttered the previous diagram. The LANs at each end are Class B networks, and the WAN link is a Class C network.

This is called a “Closed” network because it has no other connections to other networks or the Internet, which would complicate the setup. Routing requires configuration and setup of the computers and the SnIPs on each end of the link specific to the network topology.

Routing on a SnIP’s Linux computer will only work if the process variable “**ip_forward**” is set to 1. It can be checked by issuing the “cat” command at the command prompt:

```
"cat /proc/sys/net/ipv4/ip_forward"
```

if the response is “0” then it can be set until the next reboot by using the command:

```
"echo 1 > /proc/sys/net/ipv4/ip_forward"
```

This variable should normally be set by the initialization process when the sysctl program reads from its configuration file “/etc/sysctl.conf”. If this file is missing or has been modified to comment out the applicable line then the contents are as follows.

```
#
# Sample /etc/sysctl.conf file
#
# Activate IP forwarding
#
net.ipv4.ip_forward = 1

#
# Activate IP forwarding too!
#
net/ipv4/ip_forward = 1
```

The bridge link required no static routes or entries to the routing table or other information, because the bridge automatically sent packets to the WAN port. The router however requires that routes be set up in Linux’s routing tables. Basically we want to tell Linux to send specific IP Addresses encountered to the WAN port. There are normally a minimum of 3 route entries required at each SnIP:

1. **A route to the Local LAN with a Gateway of my Ethernet port, at device eth0.**
2. **A route to the WAN with a Gateway of my HDLC port, at device hdlc0.**
3. **A route to the Remote LAN with a Gateway of the Remote HDLC port, at device hdlc0.**

For the example closed network these would be represented by the following route commands at the Main site end: (persistently added in the “/etc/config/static-routes” file)

- | |
|---|
| <ol style="list-style-type: none"> 1. route add –net 172.20.0.0 netmask 255.255.0.0 gw 172.20.254.254 dev eth0. 2. route add –net 192.168.55.0 netmask 255.255.255.0 gw 192.168.55.2 dev hdlc0. 3. route add –net 172.21.0.0 netmask 255.255.0.0 gw 192.168.55.3 dev hdlc0. |
|---|

And these routes at the Remote End:

- | |
|---|
| <ol style="list-style-type: none"> 1. route add –net 172.21.0.0 netmask 255.255.0.0 gw 172.21.254.254 dev eth0. 2. route add –net 192.168.55.0 netmask 255.255.255.0 gw 192.168.55.3 dev hdlc0. 3. route add –net 172.20.0.0 netmask 255.255.0.0 gw 192.168.55.2 dev hdlc0. |
|---|

Add these commands to the “static-routes” file (minus the “route add”) to make them persistent. These routes are complimentary at each end of the link as you might expect. These are not the only routing rules that will cause the packets to be routed between the two link ends. Like programming there are sometimes more than one way to achieve the same results.

One item of note mentioned in routing documentation is to avoid default routes set at the the two link ends pointing to each other, creating the possibility of a routing loop.

The computers at each end in this example would also have their gateway addresses set to the IP Address of the local SnIP’s eth0 address.

An example route tables for the SnIP on the remote end of this link might be as follows: **Note:** Use the “route” command to view the table.

Remote Site Route Table

```
[snip3:1 ~] route
Kernel IP routing table
Destination      Gateway          Genmask         Flags Metric Ref    Use Iface
192.168.55.0    192.168.55.3   255.255.255.0   UG    0      0      0 hd1c0
172.20.0.0      192.168.55.2   255.255.0.0     UG    0      0      0 hd1c0
172.21.0.0      172.20.254.254 255.255.0.0     UG    0      0      0 eth0
[snip3:1 ~]
```

The exact routing table display may vary depending on prior routes and the order added.

3.1.1.2.1 Adding an Internet Connection

Now assume that we add a connection to the internet at the Main Site and want both LAN networks to have access to it. The modified diagram adding a DSL connection is shown in the figure below.

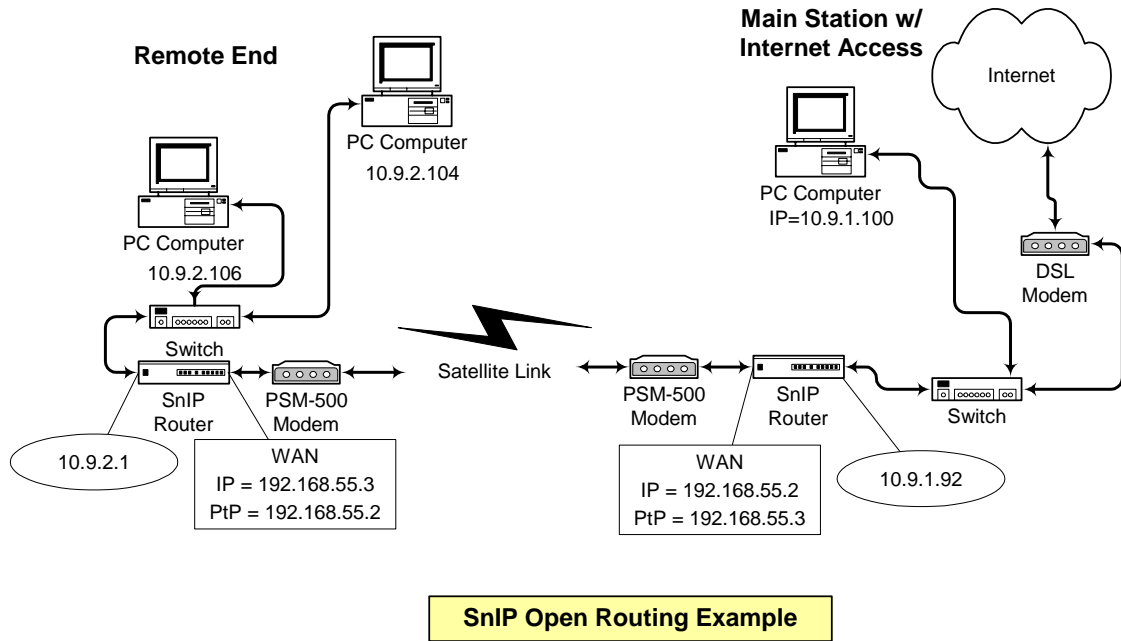


Figure 7 – Simple Point-to-Point Open Example Routing Diagram

The network addresses on each end have changed in this example, but the WAN addresses remain the same. The same basic routing rules are required for this setup as the previous with two additions. The Main site SnIP will have a default route set to the address of the DSL modem's address. The remote site will have a default gateway route added pointing to the WAN address of the Main SnIP end. To set these routes you would enter the following commands at each end. Note, this is only a simple example and setting routing tables can be a complex subject.

Remote End

```
"route add default gw 192.168.55.2" (Main WAN address)
```

Main Station End

```
"route add default gw 10.9.1.1" (The DSL gateway address).
```

The computers on each end of the link will also require additional configuration to allow working with the web access. At the remote end the computers must have the DNS servers set in their configuration. Now a command like "ping datumsystems.com" from a computer on the remote end is passed on to the main end which has a default gateway of the DSL gateway device. This must be set in the computer even if the SnIP knows these addresses from the "resolv.conf" file.

The computers at the Main Site now having Internet connections may have received their setup for Gateway and DNS from a DHCP server either in the Internet Gateway (like a DSL modem/router) or from another computer. They will require at least one added route instruction to telling them that the remote site's IP and Netmask and using the local SnIP's Ethernet port address as the gateway for this network. Without this last setting these computers would direct their packets addressed to the remote site out the Internet Gateway and on to the web. For a Linux computer this is done using the "route add -net..." command as root. On a Windows computer this is entered as an added gateway in the Network and Dialup Connections→Local Area Connection→TCP/IP Properties→Advanced TCP/IP Settings Dialog.

Router configuration is one of those disciplines that requires significant study and cannot be fully explained here. Many good tutorials and books are available for both Cisco and Linux type routers both on the web and in print. Many of the network variables and their setting from sysctl are documented at sites like the following:

<http://lxr.linux.no/source/Documentation/networking/ip-sysctl.txt?v=2.6.18>

3.1.1.2.2 Static Routing Between a SnIP and Cisco Type Router

Cisco routers are very common in networks and it is possible to perform static routing between a SnIP on one end of a link and typical Cisco router on the other end. The Cisco router would typically have a V.35 or RS-449 connection to its modem.

The SnIP would have the same type configuration as in the above example. The Cisco router would require the following configurations assuming it was at the remote site and consisted of a FastEthernet port (0/0) and a Serial WIC port (0/0).

```
Router> enable
Router# configure terminal
Router(config)#
Router(config)# interface fastethernet 0/0
Router(config)# ip address 172.21.0.0 255.255.0.0
Router(config)# no shut
Router(config)# interface serial 0/0
Router(config)# ip address 192.168.55.3 255.255.255.0
Router(config)# no shut
Router(config)# ip routing
Router(config)# ip route 172.20.0.0 255.255.0.0 192.168.55.3
Router(config)# end
```

```
Router# copy running-config startup-config
Router# OK
```

Notice that the routing rule is almost the same as that for the SnIP it replaces. The “ip routing” command preceding the route is necessary to enable static ip routing.

3.1.2 Point-to-Multi-Point Links

These are more complex links usually consisting of a hub stations that connects to multiple remotes where a single outbound carrier contains information for many remote sites. The return link (inbound to the hub) may take several forms, but they are usually one direction point to point links. This topology is commonly referred to as a “Star”.

Star configurations are normally forced to use routers instead of bridges. That is because the bridge logic does not allow splitting the transmit and receive. It sets up its hash tables of MAC addresses and what port they are located on based on seeing responses to outgoing packets.

The Figure below shows a typical setup of one type of Point-to-Multi-point network. In this one a single modem at the Hub transmits information to all of the remotes, and there is also a receive only Demodulator to receive the inbound traffic from each remote. The remotes each have a full transmit and receive modem.

The Remote sites could be set up to “filter” the incoming traffic to pass only common information plus their unique information. The return inbound link to the hub from each remote gets combined at the switch device.

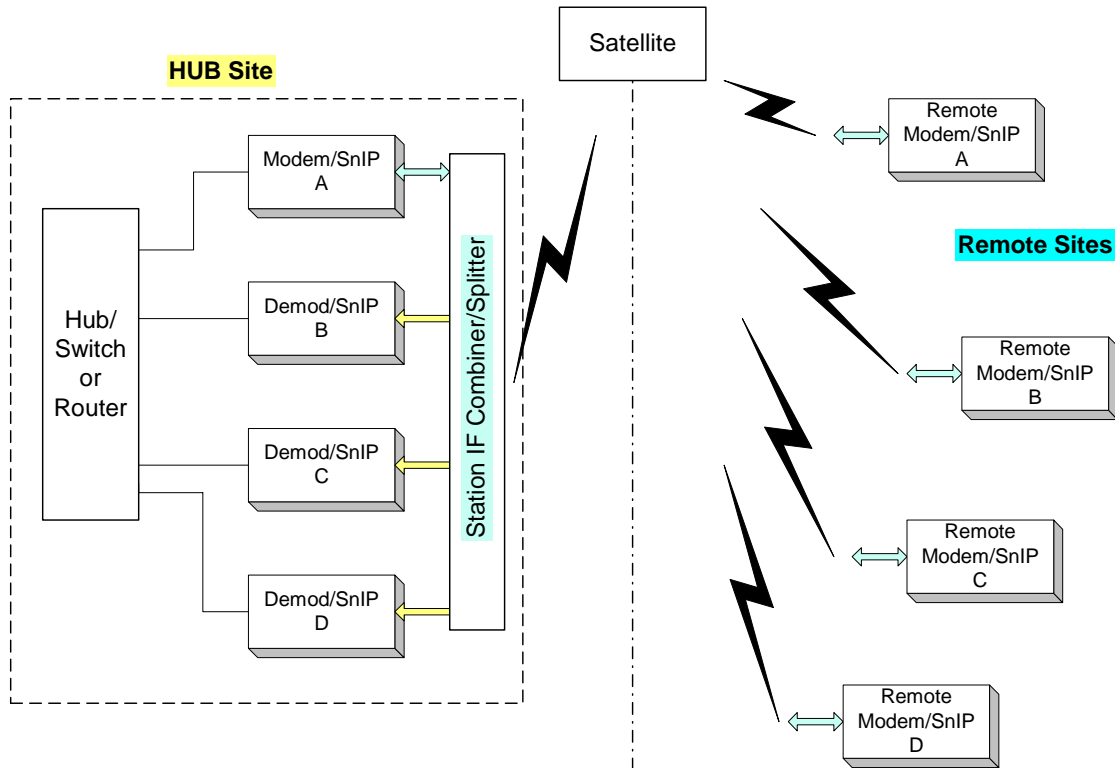


Figure 7 Point-to-Multi-Point Network Topology

The WAN IP and Point-to-Point Addresses might be set up as follows:

Point to Multi-Point WAN Addresses			
Station	Modem	WAN IP Address	WAN P-t-P Address
Hub	A	192.168.55.2	192.168.55.3
Hub	B	192.168.55.2	192.168.55.3
Hub	C	192.168.55.2	192.168.55.3
Hub	D	192.168.55.2	192.168.55.3
Remote	A	192.168.55.3	192.168.55.2
Remote	B	192.168.55.3	192.168.55.2
Remote	C	192.168.55.3	192.168.55.2
Remote	D	192.168.55.3	192.168.55.2

There is nothing specific about the particular WAN Addresses chosen. They could have just as easily been in the 10.x.y.z or 14.1.1.z ranges. The 192.168's are standard private addresses but could be confused with other LAN side IP Addresses of 192.168.1.z commonly used.

Notice that all of the modems or demodulators at the Hub have the same IP and Point-to-Point addresses, and that all of the remotes have the same complementary WAN addresses. This works because the WAN addresses are private to each link. Modem A at the Hub transmits to all of the Remotes at a common address. There are other ways to do this, but are not discussed here.

Setting the routing rules at each location is more difficult, and is not yet covered in this document. We plan on adding an example in the future.

3.2 Bridge Setup

To set up the bridging mode it is normally sufficient to simply call the "configwiz script and select LinkMode and set it to bridge. The configwiz script will in turn call the reconfig script at exit, reconfiguring the SnIP for bridge operation, without requiring a reboot. On reboot the bridge will be started automatically. Without the "BRIDGE=yes" line in the "/etc/config/system" file the bridge start-stop-script will not start the bridge operation.

This configuration was created with a default set of commands contained in a script similar to the one below, shown only to illustrate the method used:

- ❑ `modprobe snip_gen_hdlc (Note - only if not already installed).`
- ❑ `brctl addbr br0`
- ❑ `brctl addif br0 hdlc0`
- ❑ `brctl addif br0 eth0`
- ❑ `ifconfig eth0 0.0.0.0`
- ❑ `ifconfig hdlc0 0.0.0.0`
- ❑ `ifconfig br0 [eth0's original ip address]`
- ❑ `ifconfig br0 up`

In the above lines we used the bridge control program, brctl, to create a bridge called br0 and add the two interfaces eth0 (Ethernet) and hdlc0 (satellite hdlc port) to this bridge. Then we made the original port IP Addresses 0 to eliminate them and set the bridge's IP Address to what the Ethernet port's address was originally. That last is necessary so that the web server, telnet and any other access method to the SnIP are still viable. Note that we set up the normal Ethernet connection eth0 after the hdlc port. We do that because the bridge takes on the ip address and MAC address of the last interface added to the bridge.

The actual script used to start and stop the bridge is a little more complex, allowing the bridge to be built and torn down by software. The script is named "bridge" and is located in the "/etc/init.d" directory with the other initialization scripts.

More information on using brctl and bridge applications can be found on the web.

3.3 Router Setup

To set up the routing mode it is normally sufficient to simply call the "configwiz" script and select LinkMode and set it to Router. The configwiz script will in turn call the reconfig script at exit, reconfiguring the SnIP for router operation, without requiring a reboot. On reboot the router mode will be started automatically. Without the "BRIDGE=no" and "ROUTER=yes" lines in the "/etc/config/system" file the router mode will not start properly.

The WAN P-t-P Address presented in the configwiz script is not used in bridging mode. It only appears in ifconfig when the CHDLC protocol is invoked. The P-t-P Address is the address that the hdlc port will send packets to. In a point to point link this should be the WAN address of the other end of the link. In a point to multipoint system the transmit P-t-P Address of the hub is used as the WAN address of all of the remote receives

3.3.1 Setting Static Routes

When Linux starts it looks at multiple configuration files to determine settings for its operation. One of these is in the "network" script called by the etc/init.d/sysconfig script. It first sets the IP Addresses of the interfaces and then looks for static routes in a file named "/etc/config/static-routes".

If it finds the file it executes the contents by feeding them as arguments to the Linux "route add" command.

For example this file might have a line as follows for setting the default gateway-

```
"default gw 192.168.1.1 eth0"
```

Which is executed by the network script as if the following was typed at the console:

```
"route add default gw 192.168.1.1 eth0"
```

This is telling the static route to use the address 192.168.1.1 as the default gateway. The other standard type of static route command would

```
"route add -net 10.1.1.1 netmask 255.255.255.0 dev eth0"
```

Which is setting up a Class C route table entry for any addresses 10.1.1.xxx to be routed out device eth0. The "dev" can be eliminated as it is the last entry.

This last type of command can be used to set up static routes over the WAN, HDLC modem port link, and might be entered as:

```
"route add -net 192.56.76.0 netmask 255.255.255.0 hdlc0"
```

There is also the Linux "ip route" command which itself will feed routes to the route command.

3.3.2 Setting Dynamic Routes

Next the network script looks for a router suite like Zebra or Quagga (a relative of Zebra) in the "usr/sbin" directory. If it finds the proper files then it proceeds to set up the defined dynamic routes according to the instructions in the configuration files located in "etc/config" directory. You should read the documentation at www.quagga.net. The version installed in the SnIP may vary by time of manufacture. Check our web site for updates.

3.3.3 Setting Up Multi-Cast Routes

Multi-Cast routing is a special routing protocol that allows setting up special point to multi-point links. It uses the kernels “mroured” daemon and also a special version modified for satellite use named “mroured-nc”. Multi-Cast routing is normally used for video or teleconferencing functions and possibly requires additional Network Management System software to permit ease in setting up and managing the connections.

3.3.4 Setting Up Filtering or NAT using ‘iptables’

The iptables initialization is currently set up in the inittab configuration file, and points to the “iptables” script in the “/etc/init.d” directory, but the “iptables” configuration and data files do not exist, so the script immediately exits. To set it up to work you must create appropriate data and configuration files in the “etc/config” directory. If it finds the proper files then it proceeds to set up the defined filter policy chains. You should read the documentation at <http://www.netfilter.org/>. The version currently installed in the SnIP is 1.3.5 but may vary by time of manufacture. Check our web site for updates.

Iptables is the advanced successor to the “ipchains” filter functions used in Linux 2.2 and some 2.4 implementations. The configuration rules are similar and involves “Adding” or Deleting filtering instructions. A typical entry into the “iptables” file might be:

```
-A FORWARD -i hdlc0 -d 10.9.12.32/24 -j DROP
```

3.3.5 Setting Up Bridge Filtering or NAT using ‘ebtables’

The ebtables initialization is currently set up in the inittab configuration file, and points to the “ebtables” script in the “/etc/init.d” directory, but the “ebtables” configuration and data files do not contain any usable information, so the script immediately exits. To set it up to work you must create appropriate data and configuration files in the “etc/config” directory. If it finds the proper files then it proceeds to set up the defined filter policy chains. You should read the documentation at <http://ebtables.sourceforge.net/>. The version currently installed in the SnIP is 2.0.8-2 but may vary by time of manufacture. Check our web site for updates.

Ebtables configuration rules are similar to iptables and involves “Adding” or Deleting filtering instructions. A typical entry into the “ebtables” file for basic filtering from the examples page might be:

```
ebtables -P FORWARD DROP
ebtables -A FORWARD -p IPv4 -j ACCEPT
ebtables -A FORWARD -p ARP -j ACCEPT
ebtables -A FORWARD -p LENGTH -j ACCEPT
ebtables -A FORWARD --log-level info --log-ip --log-prefix EBFW
ebtables -P INPUT DROP
ebtables -A INPUT -p IPv4 -j ACCEPT
ebtables -A INPUT -p ARP -j ACCEPT
ebtables -A INPUT -p LENGTH -j ACCEPT
ebtables -A INPUT --log-level info --log-ip --log-prefix EBFW
ebtables -P OUTPUT DROP
ebtables -A OUTPUT -p IPv4 -j ACCEPT
ebtables -A OUTPUT -p ARP -j ACCEPT
ebtables -A OUTPUT -p LENGTH -j ACCEPT
ebtables -A OUTPUT --log-level info --log-ip --log-arp --log-prefix EBFW
-j DROP
```

4.0 SnIP Software: U-Boot and Linux Operations

The SnIP card contains a flash memory chip that serves as non-volatile storage. It is not just the equivalent of a disk drive. Instead it is logically divided into 4 segments each representing a different function. The four partitions are U-Boot, Environment, Kernel and JFFS2.

- ❑ The U-Boot partition contains the boot-loader.
- ❑ The Environment partition contains the boot loaders stored environment variables plus a serial number and the SnIP's MAC address.
- ❑ The Kernel partition contains the Linux kernel, compressed.
- ❑ The JFFS contains a compressed Journaling Flash File System, which is the equivalent of the hard drive in a standard computer.

The SnIP875 ships with Linux version 2.6.15 or later depending on the date of manufacture. There are many standard Linux defaults, directories (folders) configuration methods and operating procedures that, although familiar to Linux administrators may not be familiar to the more casual user. Below are some of those procedures and methods. Note that unlike Windows the directory separator is a forward slash "/" not the back slash "\".

The basic SnIP configuration contains only a single user – the "root" user who has all of the privileges available on the Operating System or OS. Unlike some Linux implementations that have a /home/username

4.1 Basic Linux Directory Structure

/

/bin /boot /dev /etc /home /lib /mnt /proc /root /sbin /tmp /usr /var

- ❑ /bin and /sbin directories – contains basic root accessible commands and utilities.
- ❑ /usr/bin and /usr/sbin directories – contains basic user accessible commands and utilities. There may also be application libraries in these directories.
- ❑ /dev directory contains the system device special files.
- ❑ /etc and its subdirectories contains system and application configuration files.
 - /etc/init.d directory contains initialization scripts.
 - /etc/profile.d directory contains automatic user scripts.
- ❑ /home and its subdirectories contain the home directories for non-root users added to the system.
- ❑ /lib and its subdirectories contains system and application library files plus kernel modules.
- ❑ /mnt and its subdirectories contains other devices mounted in the filesystems.
- ❑ /proc contains information on currently running processes.
- ❑ /root is the root user home directory.
- ❑ /tmp is a container for non-persistent files not written to flash.
- ❑ /var and its subdirectories contains main application programs like appWeb, with their associated libraries.

Each of these directories may have other sub-directories. Most of the common Linux commands are located in for directories: /bin, /sbin, /usr/bin and /usr/sbin. The SnIP's web site is located in the /var/appWeb/web directory.

Virtually all of the startup and configuration files are located in the /etc directory, including the startup scripts for networking, bridging and routing. The /etc/config directory holds most of the operating configuration files.

The /tmp file is unique because it is created and mounted strictly in the SnIP's operating RAM memory. When the SnIP is rebooted, or powered down, all of the information and files in this directory are lost. This is a valuable resource used for log files, and as a temporary holding location when updating either the kernel image or the bootloader.

Linux is implemented as two main elements: Linux itself which is a compressed file expanded and started by the bootloader, and a "Root Filesystem" which in this implementation is a Journaling Flash File System Version 2 (JFFS2). These two elements are stored in two different partitions of the flash memory.

The Linux OS allows commands to operate on the files in these directories and applications that perform specific functions.

4.2 Secure Copy Procedure (scp) –

Requires:

- ❑ Operating SnIP with a known IP address
- ❑ Host Computer with the files to be copied to the SnIP (scp is part of the Linux SSH suite). For Windows computers there is an scp host application named WinSCP3 which implements parts of the SSH suite including the scp protocol. There is also a command line application as part of the PuTTY Windows application named "pscp.exe".

Procedure:

The basic command is "**scp sourceFile destinationFile**". The more explicit directions below show its use from either a host computer or the SnIP. Unlike the tftp protocol, the access on the host computer is not limited to a fixed tftp directory.

Scp always asks for the password for the user at the destination location. If multiple files are transferred in one transaction then the password is only required once. If a script is used to transfer multiple files separately then the password is requested on each transfer.

A. From a Host Computer to copy to a SnIP:

On the host computer navigate to the directory containing the file(s) to be copied. The files could be new or updated versions of existing files. You also must know the proper directory where these files belong on the SnIP. For example a user created test program might reside in /usr/bin while a script to be executed by the root would be directly in the /root directory. The ip address used is that of the SnIP.

scp directory/file root@ipaddress:directory

For example: "**scp bridge-test root@192.168.100.82:/root**" which might be copying a user bash script into the /root directory.

Another example – "**scp myprogram root@192.168.1.82:/usr/bin**"

And to copy from the SnIP to the host computer an example command might be:

scp root@192.168.1.82:/etc/hosts.allow hosts.allow

In each case the SnIP will ask for the SnIP's root password before allowing the copy to proceed. On the first transfer by this method to a SnIP from a particular computer you will also be asked for confirmation to add the SnIP and its RSA certificate to your list of known secure addresses.

If you later reload the Linux software into the SnIP and it generates new RSA certificates then you may not be allowed to secure copy from a computer. You may have to remove the SnIP with that

IP address from the “home/name/.ssh/known_hosts” (Linux name) file and re-initiate the certification with a new secure copy.

A1. SCP from a Windows Computer using PuTTY – pscp.exe

The Windows based Putty Program contains many useful features. One of those is a program named pscp.exe is normally installed in the “C:\Program Files\PuTTY” folder when installing PuTTY. To use it in the above first example there are two added concerns. First the directory is not part of the normal Windows Path, and second Windows uses the backslash to separate directories. So the full command to copy the file named “myprogram” from the present working directory would be:

C:\program files\putty\pscp myprogram root@192.168.1.82:/usr/bin

The pscp.exe program apparently needs no installation and could therefore be dropped into any directory already in the Path and the commands would then be almost identical to those on the Linux computer – don’t forget the backslash on the Windows end though.

B. From a SnIP Terminal to copy from a Host Computer:

This procedure is essentially the reverse of the above. From the SnIP console or Telnet/SSH session you would use the following command to copy a file from any directory on the host to any directory on the SnIP. The ip address used is that of the host.

scp user@ipaddress:directory directory/file

For example to copy the file test1 from mike’s home directory named “mike” the command might be:

scp mike@192.168.1.101:/home/mike/test1 test1

The protocol will ask for the user’s password to access the computer and files. The first time this is accomplished the secure protocol will also create a file of known hosts in the SnIP so that this host and its key will be remembered. The password is always required though.

4.3 File Transfer Protocol (FTP) (Using wget instead)

The “ftp” server or client is not normally implemented on the SnIP. This is partially for security reasons, and the preferred procedure is “scp” for secure copy. The powerful Linux “wget” command however is available and is capable of retrieving files from web “HTTP” or “FTP” resources. Although not essential, normally DNS server connections are needed to properly use wget. See Section 8.25 on specifying a DNS server later. Suppose that you wanted to retrieve a file from the http web site named www.datum-direct.com and the file was a new “mdm-find” located in the download directory. The command to get this file and put it into the local /tmp directory to test it would be”

wget -P /tmp “http://www.datum-direct.com/download/mdm-find”

The “-P” option tells wget where to put the file that it is downloading from the shown URL. Note the quotation marks around the URL which although not necessary here help protect URL names that may have spaces included.

Now assume that you wanted to get the same file from an FTP server at the same base web location, but with the address “ftp://snip875.datum-direct.com” and in the directory “/updates/root”. Further consider that this is a password protected site and your user name is “vtcorp” and your password is “dogdays”. So now the command is:

wget -P /tmp “ftp://vtcorp:dogdays@snip875.datum-direct.com/updates/root/mdm-find”

There is a lot of info here, but it breaks down to get from the web, putting into my absolute /tmp directory from user “vtcorp” with password “dogdays” at the ftp site “snip875.datum-direct.com” the file “/updates/root/mdm-find”. You could test the file and if satisfied copy it to the directory of your choice.

4.4 Trivial File Transfer Protocol – (tftp)

TFTP is not a secure protocol!

The TFTP client is implemented in BusyBox on the SnIP. To use TFTP to transfer a file between a host computer and the SnIP the procedure varies slightly by which end you are operating from and what direction the transfer is to occur. TFTP requires that a host computer be set up with the TFTP server installed and running. TFTP also requires a standard directory on the host (named "/tftpboot" on Linux computers). All file transfers occur to and from this directory

From the SnIP console or Telnet session the user command to get a file from the tftp server is:

tftp -g -l local-filename hostaddress -r remote-filename

For example a user would type the following to copy the file named "test1" from the /tftpboot directory on the host to the present working directory on the SnIP.

tftp -g 192.168.1.101 -r test1

In that example we skipped the local filename because it was the same. But if, for example, the remote was in a subdirectory of the tftpboot directory then specifying the local filename is required. For example if a new kernel image was in the server directory /tftpboot/snip6 and you had changed to the SnIP's /tmp directory and wanted to download the new image the command would be:

tftp -g -l ulmage 192.168.1.101 -r snip6/ulmage

tftp is also fairly lenient about the order of the options and they are somewhat arbitrarily placed here.

The procedure to put a file from the SnIP to a host is the same as the above example except that the "-g" is replaced by "-p" for put. This direction would be rarely used since it is not required normally and tftp servers are often configured to be read only.

4.5 Telnet

Telnet mode requires that a previously setup IP address is already configured.

The Telnet Mode allows all of the control of the local console connection except from any computer that can address this SnIP's IP address. When the bridge mode is enabled the bridge takes over the Ethernet port's IP address, allowing telnet sessions to be set up. The Telnet session can also be established from either the LAN or Satellite port sides of a link.

When a usable IP address is assigned the telnet connection can be made from a Windows machine using a DOS Command prompt window or from a shell terminal window in Linux. The command is the same for either – "telnet [ip address]". For example if the IP address is set to 192.168.150.210 then the command is:

telnet 192.168.150.210

The SnIP should respond with:

[Hostname] Login:

Password:

The default Login is "root", and the default password is "datum" .

If the password is changed it is the responsibility of the administrator to keep track of passwords. The password can be reset from the Terminal mode with connection to the DB9 "Console" connector.

With some Windows browser versions the invocation of the URL address of:

telnet://192.168.1.120/

will automatically launch the “Hyperterminal” communications program that is part of many Windows installations. The Hyperterminal display will mirror that of the DOS command prompt window.

Multiple user “sessions” are possible simultaneously. Part of the telnet command prompt tells which virtual terminal is connected.

When the telnet session is no longer needed the user can sign off using the command “exit”. The SnIP will then close the connection.

4.6 Local Console Control

When shipped from the factory the SnIP is a bridging application set up to operate with all ports and traffic directions enabled. The SnIP is highly configurable and is mainly controlled via the local Console connection on the rear panel of the modem. Use a 1:1 male to female cable to connect this port to a standard PC Com port. You can then use Hyperterminal, gtkTerm (Linux) or some other VT100 or ANSI terminal emulation program to manage the SnIP. The default communications settings are 38400,N,8,1.

This port will ask for a standard Linux login before you can gain access to any control functions. The default login user is “root” and the default password is ‘datum’. Once logged in you have full access to the Linux operating system and all of its programs.

Some useful programs are the standard Linux “route” command, which can show or modify the static routing currently in use, “configwiz” which allows setting up the basic SnIP configuration and all of the other Linux commands. Those can be seen by using “ls” on the /bin, /sbin, /usr/bin and /usr/sbin directories. Configuration information is normally kept in the /etc directory.

The configuration files and scripts, which control the Snip, can be modified using either the common vi, or the more advanced “nano” editors. Either will require entering the following environment variable for Hyperterminal. It is also required on a Linux computer when using the virtual terminals from Telnet or SSH.

export TERM=vt100

This will probably be modified in the future to make the vt100 terminal the default.

The nano editor is a full screen simulated editor on a terminal, and is fairly competent. The basic commands are “nano filename” to load a file for editing, “^o” (Control key and the letter “o”) to save the modified file, and “^x” to quit or exit nano.

4.7 Disaster Recovery

Linux is extremely powerful, and it is not uncommon to operate as the root user within the SnIP which will also allow you to make some creative mistakes. Some may be fairly easily recoverable though.

If you have set up routing rules or iptables rules that no longer permit you to access the modem via telnet, it might be possible to recover using the modem front panel. A special non-valid network mask consisting of “000.a.b.c” where a.b.c can be the previous setting, is used as a trigger to tell the SnIP on start-up to not load the “static-routes” or “iptables” rules, thus permitting access again via telnet or ssh. When you enter a new IP address or Network Mask on the modem’s front panel the SnIP reads the new values and enters a reboot cycle, allowing this trigger to work on the resulting startup. The SnIP reads this non-valid mask and restores it to 255.255.255.0 for setup, but retains the value in the system config file.

Some types of setups, especially those using custom applications, may result in a lock-up not allowing either telnet or console access. A special mechanism can be used to help prevent this if

the application is designed to recognize it. By stopping the boot-up procedure while still in the boot-loader (use the “qt” special keys) and then changing the bootargs environment variable to add the variable “rescue=yes”, special actions can be accomplished in the Linux at startup. The sequence is:

1. Stop the bootloader with “qt” when prompted.
2. Type “**setenv bootargs root=1F03 rw rescue=yes rootfstype=jffs2**”
3. Type “**bootm fe060000**”

Linux should now boot and disable the offending application if previously set up to do that. You have to put your rescue procedure in the script file named “/etc/init.d/rescue” before this will work.

You may also be able to recover some types of errors by reloading them using the procedures outlined below in Section 5. As a last resort, you may need to reload a new kernel and Root Filesystem using the bootloader as outlined in Section 10.1. That will result in returning the SnIP to its factory condition.

5.0 New Software Load or Upgrade

The SnIP is very similar to a standard PC type computer running Linux, but without the video output and graphical interface. The hard disk is replaced by 16 MegaBytes of solid state NOR flash memory, which appears to Linux as a “Memory Technology Device” or MTD. Normally the MTD does not have to be “mounted” as standard Linux hard disk drives do. The flash is separated into 4 partitions similar to partitions on a hard disk containing the following information:

- ❑ mtd0 – 256 kBytes – “U-Boot” (the bootloader)
- ❑ mtd1 – 128 kBytes – “Environment” (U-Boot’s environment variables)
- ❑ mtd2 – 1.2 MBytes – “Kernel” (The Linux Kernel)
- ❑ mtd3 – 14 MBytes – “Linux Root Filesystem” (JFFS2 format)

Note that each has a name “mtd n ” where n is the partition number starting at 0. The basic flash segments are 128 kBytes which is the minimum element that can be erased and written in the flash memory. The actual devices are located in the /dev directory and fully named “/dev/mtd n ”.

U-Boot and the Linux Kernel are images that can be rewritten or updated from either U-Boot or Linux. Updating files and programs in the filesystem are accomplished by overwriting with new files or programs similar to the procedures in Linux or Windows.

Background Aside:

In the factory when nothing is loaded into the flash at all a special JTAG/BDM hardware and loading procedure must be used to load the boot loader (U-Boot) itself and then U-Boot is used to tftp and burn into flash the Linux Kernel and the raw Root Filesystem.

Normally the bootloader does not have to be updated. The following procedures show methods to update the Linux kernel or individual programs in the Root Filesystem. For an alternate method to reload the entire Linux Kernel and the Root Filesystem see Section 10.1.1.

5.1 Updating the Linux Kernel

The basic procedure is to download a new kernel image into the /tmp directory and copy that image into the proper mtd partition. Either ‘scp’ or ‘tftp’ can be used to download the new image, and a special mtd program named “flashcp”, is used to erase the current kernel and write the new one into the partition.

We download the new kernel image, named “ulmage”, to the “/tmp” directory. Since this directory is not written to flash itself we will not clutter the file system.

To use tftp you must have previously set up a tftp server with the new image located in the server’s default tftp directory. The command to execute on the SnIP is:

```
cd /tmp
tftp -g -l ulmage 192.168.1.101 -r ulmage
flashcp -v ulmage /dev/mtd2
```

In the above tftp command you must change the IP address to that of your tftp server. If the flash partition had been locked previously then you might get the response

While erasing blocks xxxxxx on /dev/mtd2: Read Only Filesystem

You should then issue the command:

```
flash_unlock /dev/mtd2
```

And retry the **flashcp** command.

The tftp command requires that a tftp server had been previously set up on the LAN. Without this it is possible to use the “**scp**” (secure copy) procedure to transfer the new image to the SnIP. To use scp you must have previously set up a working scp link between either a Linux PC or a Windows PC running a program like “WinSCP”. with the new image located in a known location in that PC.

The command to execute on the PC is:

```
scp ulmage root@192.168.1.82:/tmp
```

The IP address in the above command must be changed to that of the SnIP card.

The command to execute on the SnIP is:

```
cd /tmp  
flashcp -v ulmage /dev/mtd2
```

The “-v” option is used to show progress. After either of the above procedures the SnIP must be rebooted for the new kernel to become effective. If you reboot from a Telnet session you will have to re-establish the session after the reboot is complete. Rebooting also clears the /tmp directory, eliminating the temporary image file used to burn the new image into its flash partition.

5.2 Updating the Bootloader

The same basic procedure outlined above for updating the kernel can be used by substituting the bootloader image name, “u-boot.bin” and the proper mtd partition, mtd0. Thus after downloading the u-boot.bin file to the /tmp directory the command would be:

```
flashcp -v u-boot.bin /dev/mtd0
```

The new bootloader will not be used until the next time the SnIP is powered on or re-booted. Rebooting also clears the /tmp directory, eliminating the temporary image file used to burn the new image into its flash partition. If you are not rebooting immediately you can recover the space in the /tmp directory by deleting the image file from it.

5.3 Updating Individual Files and Applications

5.3.1 Using SCP between a Linux Computer and the SnIP

The procedure is to simply use scp or tftp to download the new file to the proper location in the SnIP’s directory structure, where it will overwrite the existing file. For example if a newer version of the m49ctl program were available then by using the scp command from the SnIP command line, either at the console or a telnet session, the new file will overwrite the existing one.

```
scp mike@192.168.1.101:/home/mike/ m49ctl /usr/bin/m49ctl
```

Where the first parameter is the host computer that contains the new file and the second is the location and file name of the new file. This example copies the file from a Linux computer with an IP address of 192.168.1.101 and the user named “mike”.

Or from the Linux computer end at a terminal session to download the new program into the SnIP the command would be:

```
scp m49ctl root@192.168.1.82:/usr/bin
```

It is important that after transferring a program by either method the properties of an executable file should be checked to insure that it can be executed. Use “chmod” if necessary to set the properties to something like “755”.

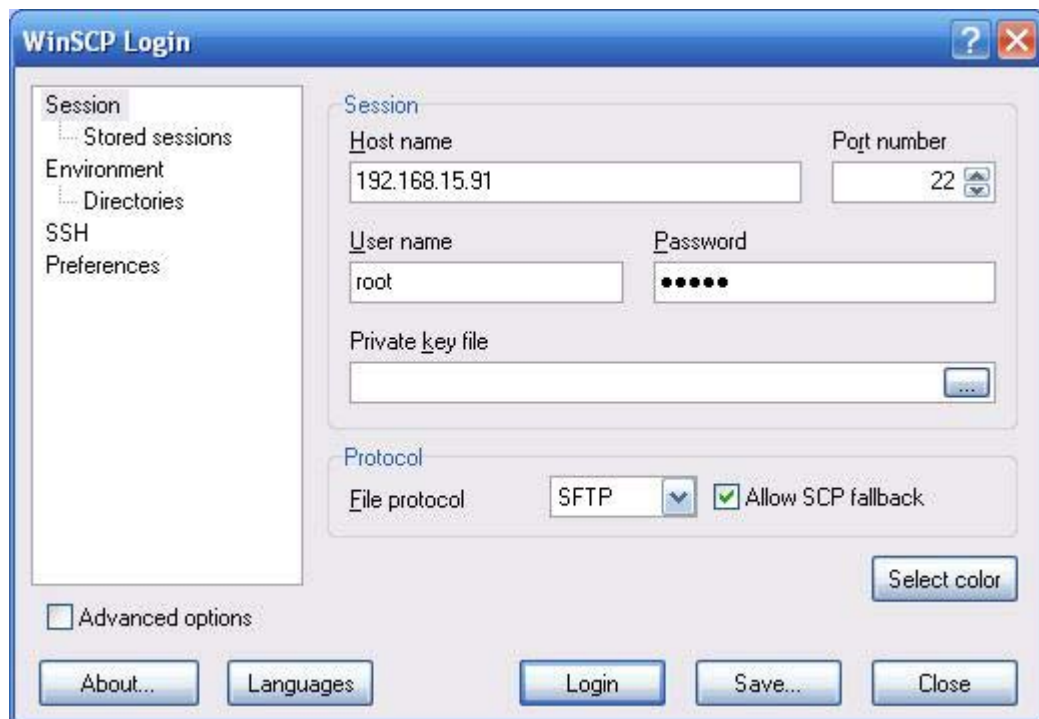
5.3.2 Using Win SCP between a Windows Computer and the SnIP

WinSCP is a very capable program that runs on various versions of Windows and provides a graphical method of transferring files between a Windows based computer and the SnIP. The procedure is generally performed as follows:

Preparation - First you must download the open source (free) WinSCP program and install it onto a Windows computer. This program is also included on the CD that came with the modem. The program must be installed on a computer that can share the same LAN as the SnIP. The WinSCP web site where the program can be downloaded: <http://winscp.net/eng/index.php> also has flags at the top representing different language versions.

First Use – When you first start WinSCP or when connecting to a new SnIP there are several initial steps required. Note that these may change with newer versions of WinSCP.

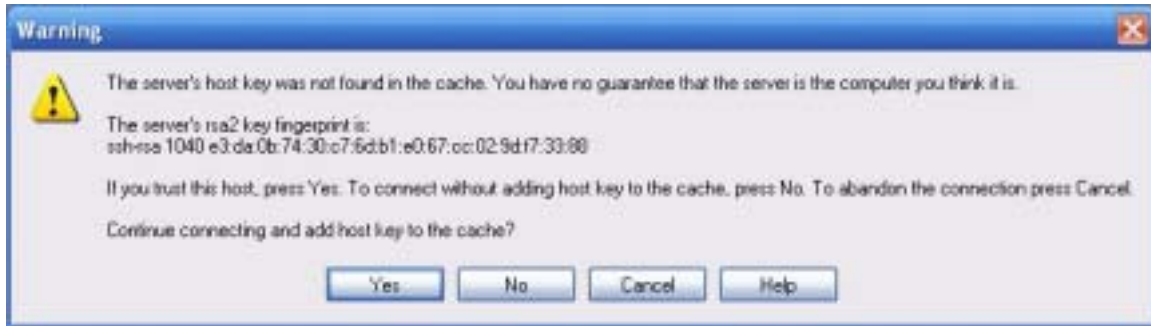
- At the main screen press the “New” button to create a new session.



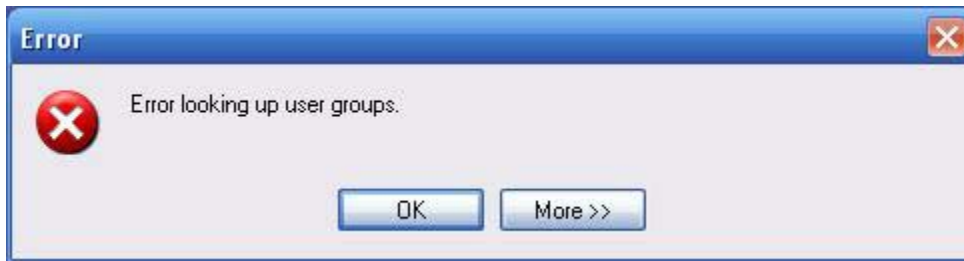
- In the Host name box enter the IP address of the SnIP, for example. 192.168.15.91.
- In the User name box enter “root”.
- In the Password box enter “datum” or your own password if you changed it.
- In the Protocol area either use SFTP with the Allow SCP fallback checked, or SCP.
- The “Private key file” is for a special purpose and can be ignored.
- You can save the session parameters now or later, giving it a unique name.

Press the “Login” button to start the login procedure. The first time the secure keys are determined you will be asked if you wish to continue and save the secure keys into the cache. Answer Yes.

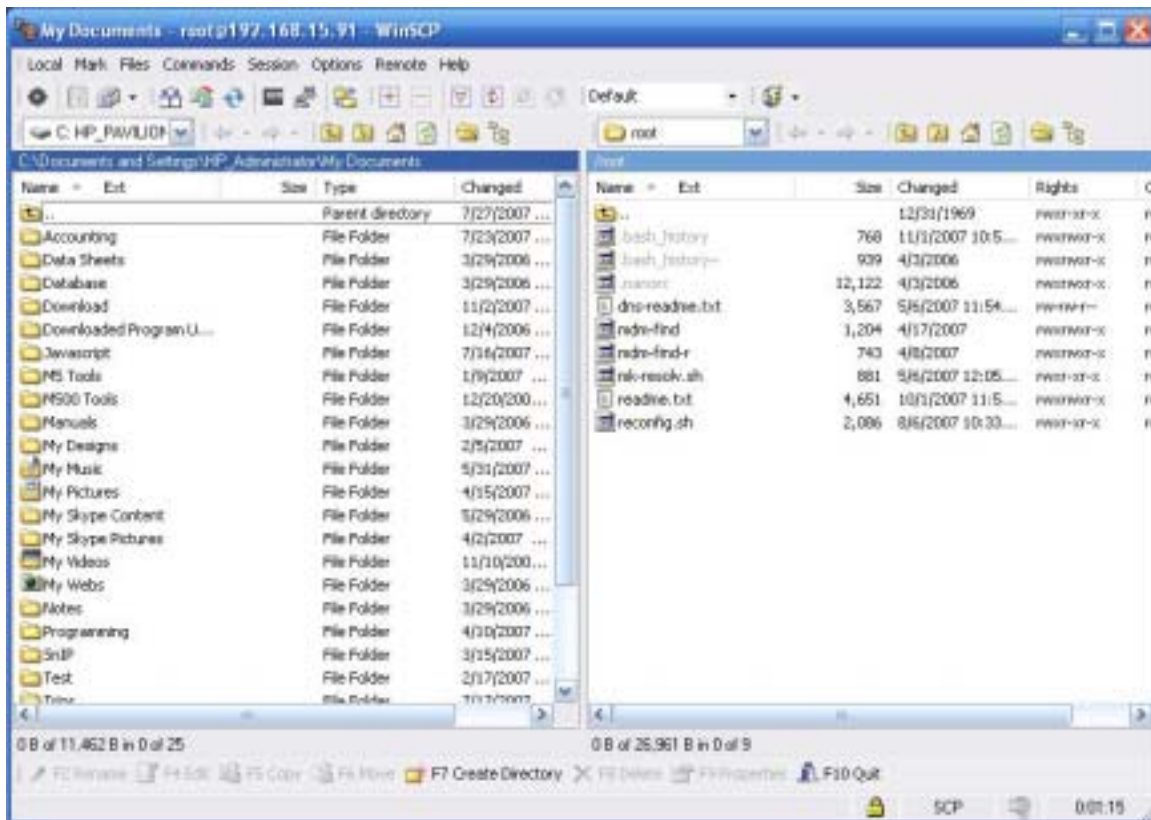
M5/500 Satellite Modem SnIP Quick Start Guide



Logging In – Each time you log in you will probably get an error message about looking up user groups. Click OK to bypass this message which does not apply to the SnIP implementation.

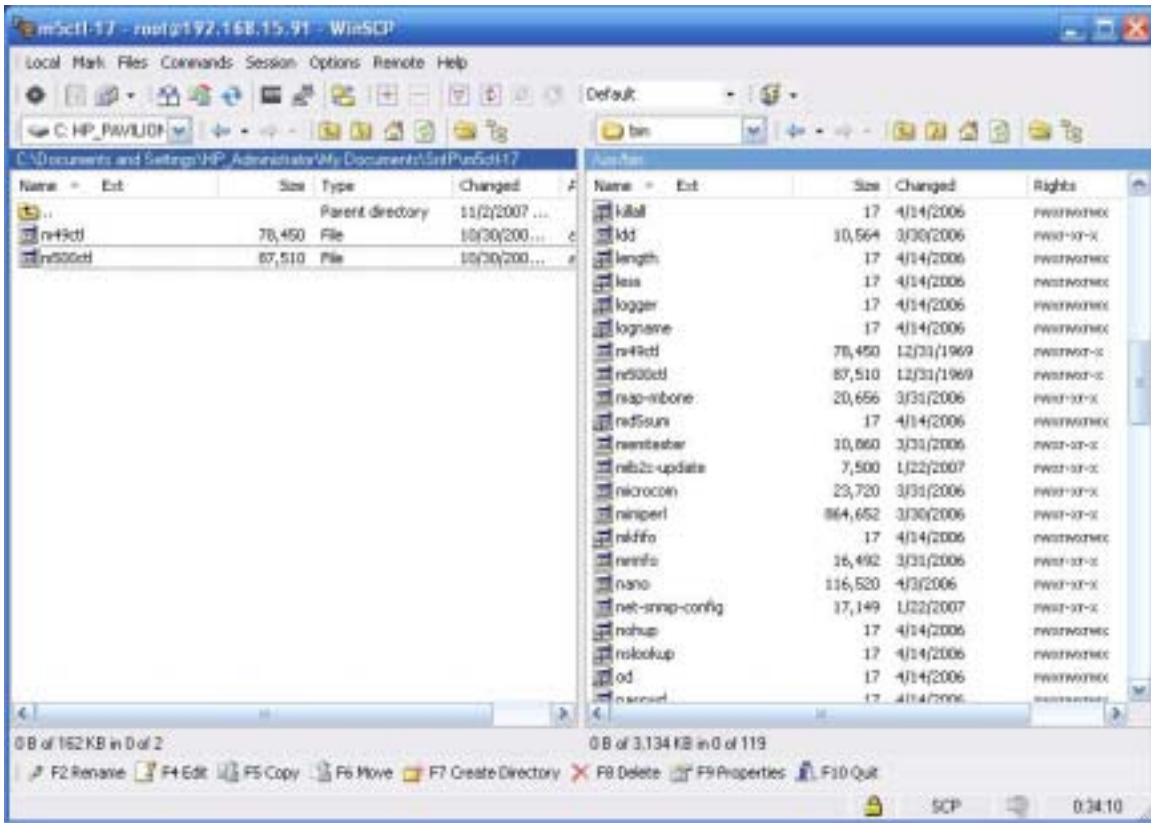


WinSCP will take a few moments to read the files locally and in the SnIP and display a screen similar to the following



In this window you will see the local computer's "My Documents" folder on the left and the SnIP's "/root" folder on the right pane (this is the root user which is where you logged in). Navigate in the

local computer pane to get to the location of the new file you wish to update or install in the SnIP. Navigate in the SnIP pane to the location where the file will be inserted. In the picture below we are ready to copy the file “m5ctl” from the local computer to the “/usr/bin” directory in the SnIP.



The local computer’s file pane is on the left and the SnIP pane is on the right. To transfer a file from the local computer to the SnIP you would select the location and file from the local computer pane and select the proper directory in the SnIP for the file to be placed.

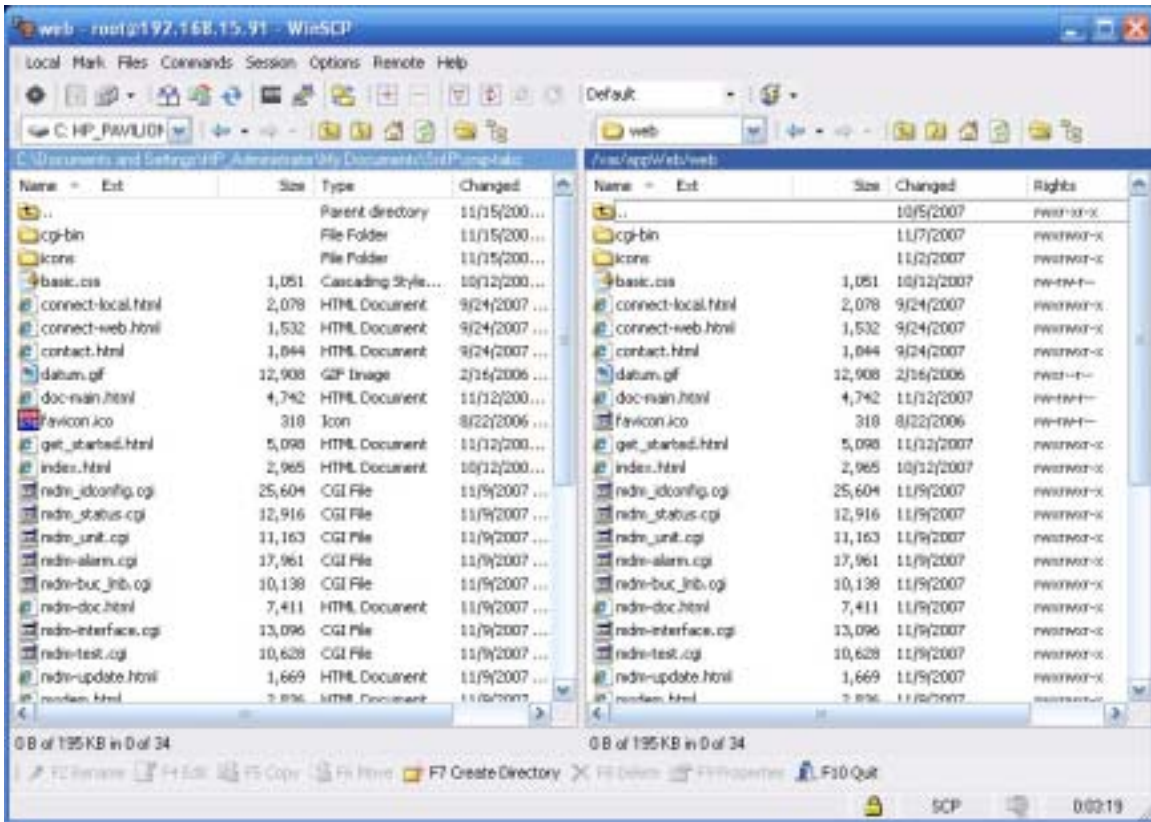
You can copy a single or multiple marked files using the “Files – Copy” command or simply drag and drop individual files from one pane to the other. There are multiple other commands available in WinSCP that can be investigated using the documentation.

WinSCP will even allow you to change the user permissions or rights for a file in the SnIP by right clicking on the file and selecting “Properties”.

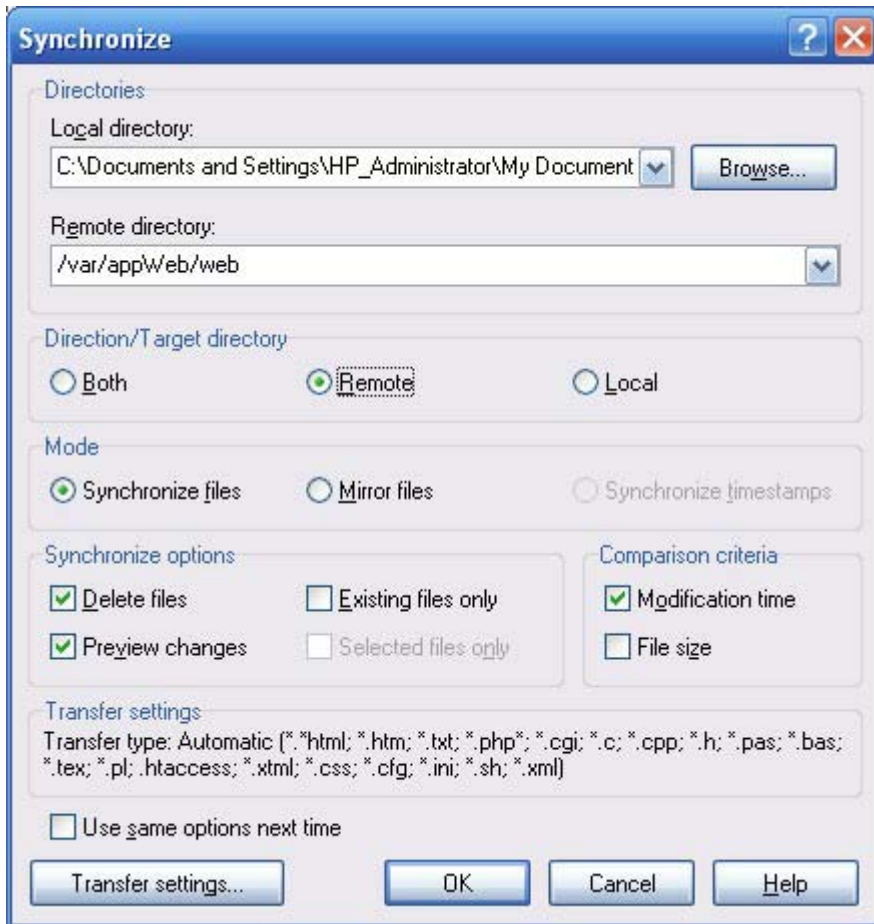
5.3.2.1 Updating and Synchronizing an Entire Directory

The SnIP includes the Linux program “rsync” and a Linux PC can be used to synchronize directories between the two computers. The same functionality is possible with WinSCP in a graphical environment and it is fairly easy to accomplish. Caution must be taken however as any of these types of facilities can radically alter the SnIP’s file structure. As an example of accomplishing this we illustrate how to update the entire SnIP Web Interface from a directory structure on a Windows PC.

After starting WinSCP as shown above navigate the Window’s pane to the downloaded and unzipped (uncompressed) web directory. Extracting using a compressed image maintains the date of the files. If they are copied directly from another location the file times will become the date of copying which will not work correctly in updating using WinSCP. In the picture below notice that we have also navigated on the SnIP pane (right) to the /var/appWeb/web directory where the web pages are located.



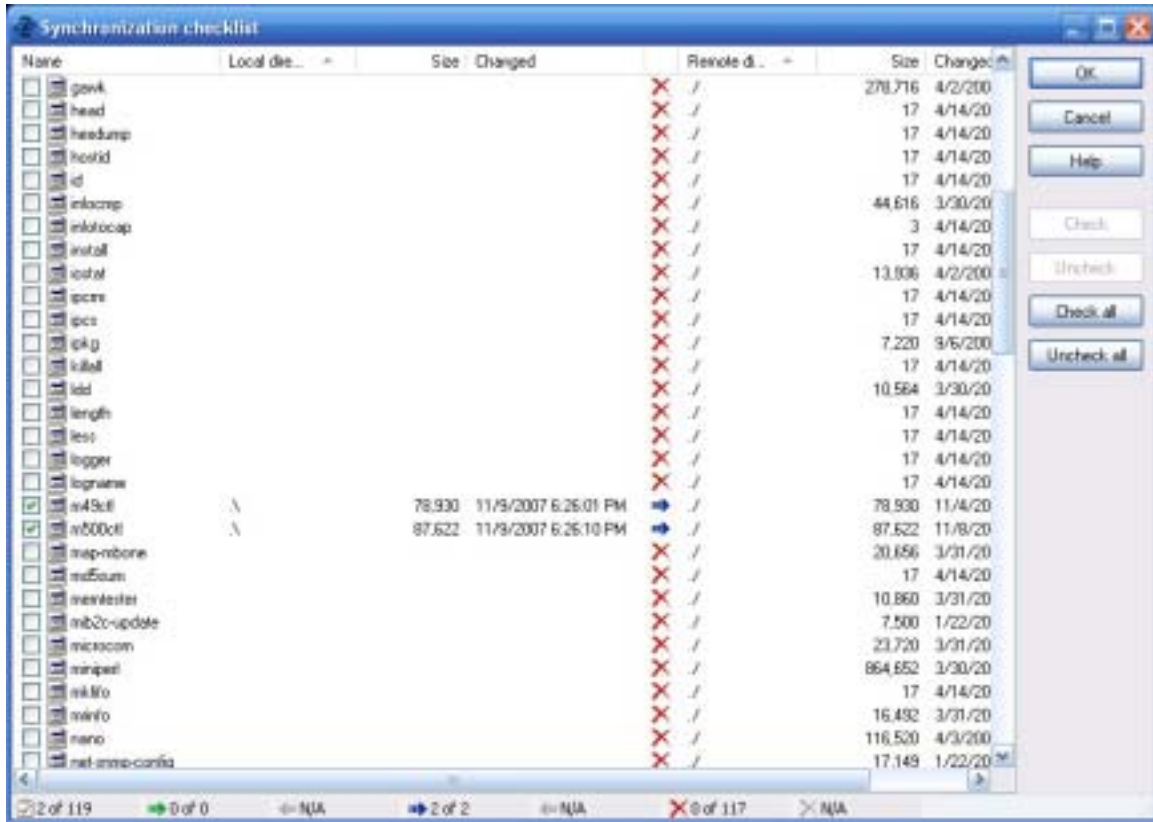
Next we will use the WinSCP Synchronize facility. The tool tip for this button with two blue arrows in a circle says “synchronize local directory with remote directory (ctrl S)”. This is shown as the icon 6 from the left in the picture above. We actually want to synchronize the remote with the local directory and that is one of the options provided. Push this button on the toolbar and a new window will pop up as shown below.



You should first select the radio button to select the target directory as “Remote”, then select the options “Delete Files”, “Preview Changes” and “Modification Time” comparison. If the date and time are not correct in the Windows PC then it is also possible to match based on “File Size” but it may not correctly catch all changes. The “Synchronize files” mode is selected by default.

Caution! In this example we are using the “Delete files” option. This will delete files on the target SnIP directory that do not exist in your host computer directory. This option should **only** be used when comparing and synchronizing complete file structures.

When you click the “OK” button the following type of preview window will be displayed showing a list of the files with selection checkboxes. This illustration happens to be from a different directory but shows the files, checkboxes, what has changed, and what direction the files are going. This preview allows you to modify the selections if needed. When ready click the “OK” button and the file transfers will occur.



That SnIP directory and its subdirectories should now be up to date.

When finished use the “F10” Quit option to close the session.

5.3.3 Using RSYNC between a Linux Computers and SnIPs

Rsync is a sophisticated file and directory synchronization tool used within or between Linux computers. For example a complete web site can be developed on a Linux PC computer in a directory, for example /home/mike/webs/snip, and then used to update the SnIP web pages as needed. The command on the Linux PC would be:

```
rsync -avz --delete /home/mike/webs/snip/ root@192.168.15.83:/var/appWeb/web
```

This command will take the local web directory and add and delete files as necessary on the SnIP to make the two directories match. It will even delete sub-directories as needed.

Rsync can also be used to synchronize files and directories between two SnIPs. The commands are slightly modified because the SnIP’s equivalent of the ssh protocol used for transfer is named “dropbear” and it does not have the same option parameters. To correct that a different dropbear interface is called named “dbclient”. Here is an example command to synchronize the web from one SnIP to another – the target is at IP address 192.168.15.91.

```
rsync -avz --delete -e dbclient /var/appWeb/web / root@192.168.15.91:/var/appWeb/web
```

Notice in both of these examples that the source directory has a trailing “/” while the destination does not. Read the rsync documentation at <http://samba.anu.edu.au/rsync/>

5.3.4 Using “ipkg” to install and update packages in the SnIP from the web

“ipkg” is a small platform version of a Linux package manager like rpm or apt. ipkg has a number of commands used to maintain a database of currently loaded optional applications and those in a repository. Datum Systems maintains a web repository at a special subdomain of datum-direct.com.

A good web location to see an ipkg tutorial is <http://www.dd-wrt.com/wiki/index.php/lpkg> Some of the basic commands are:

```
ipkg update # pulls latest list of package listing from the default sites.  
ipkg list # gives the list of ipkg's available  
ipkg -d <location> install <packagename> # installs the package of your choice
```

Currently there are very few optional packages for the SnIP since all normal programs are already within the standard root filesystem. More may be added in the future.

The basic configuration for ipkg is in the “/etc/ipkg.conf” file. This file contains the web address of the Datum Systems site which maintains its package repository. It is also possible to set up a local repository of optional packages. The address of that repository would need to be added to the configuration file.

6.0 SnIP Monitor and Control Applications

The SnIP has two monitor and control channels to the modem's main processor that are independent of its use as a data interface. These are an internal RS-232 control port and the external RS-485 port, which can be connected to the host modem or any other co-located modems. The SnIP's management ports can be used for M&C functions with or without being selected as the modems data interface.

There are two main applications for M&C built into the SnIP. The first is named "m49ctl" or "m500ctl" and is a command line application intended for either local console control or Ethernet control via Telnet or SSH. The second is the SnIP's built in web server with its browser based GUI for modem control.

6.1 *m49ctl and m500ctl Applications*

These are Linux command line applications that accept easy to remember mnemonics for monitor and control. The programs convert the input commands into binary packet messages between itself and the modem, taking the modem response and formatting it into a man or machine readable form. The human readable or "verbose" result is directly usable at the console or via telnet or SSH from anywhere accessible to the Ethernet interface.

The "m49ctl" is intended for the M5 series of PSM-4900 modems and uses its binary packet protocol while the "m500ctl" program is for use with M500 series of PSM-500 modems. Since the M500 series has M5 compatible commands, the m49ctl program could potentially be used with either modem type within the limitations of the M5 capabilities.

Use of these applications remotely requires that a SnIP IP address has been set, and that a remote console is first logged into. The applications are called from the console or telnet terminal command line.

Either of these programs uses a 2 to 5 letter mnemonic representing the command and a value as either a "?" for query or a numeric entry to send that value with the command. The command – value are always represented by a pair of entries separated by a space. The mnemonic entries mirror the convention used on the front panel in most cases. The first letter is the first letter of the 4 areas available, either "u", "m", "d", "i" standing for Unit, Mod, Demod or Interface. The second letter represents the packet (which coincides with the column on the front panel display, e.g. "d" or Data or "i" for IF or "t" for Test) and the third is the parameter. For example the modulator IF frequency is represented by the letters "mif", while the entire modulator IF packet is "mi". Staying with this example, to show the mod IF frequency you would type:

m500ctl 1 mif ?

And the modem might respond with

75125000

representing 75.125 MHz.

The "1" after the command mnemonic is the address of the unit to be read or controlled. The address "0" is used for the internal connection to the modem hosting the SnIP card. Any other address is directed out the rear panel DB9 connector on the RS-485 bus. Multiple modems can be connected to this RS-485 bus as long as they all have unique addresses between 1 and 253.

Note also that all input to the programs are lower case.

If you wished to have a more easily readable response you could use one of the available switches, "-v", to tell the program to provide a verbose output.

m500ctl -v 1 mif ?

And the modem might respond with

Modulator IF Freq = 75.125000 MHz

To set the demodulator IF frequency you would replace the “?” with the desired frequency in 1 Hz increments. Therefore to set it to 67.052 MHz, the command would be:

```
m500ctl 1 dif 67052000
```

Most commands either have values as above or selections. The selections and numbers used to set them are the same as those used on the front panel. So, to enable any “on/off” parameter, you would use “0” for disable or set off, and “1” to enable or set on. Values for selections can be found in the modem’s Appendix B – Remote Control Protocol. They are also the same values as those used on the front panel.

One difference to note is that the current implementation of the m49 and m500ctl programs use the full value input down to the lowest increment possible. Thus in the above example you could not enter “67”, “67.025” or any value other than “67052000” which carries the value down to the Hertz. The single exception is in L-Band modems where the BUC and LNB LO frequencies are entered in MHz values with no decimal points. No known BUCs or LNBs use sub MHz increments today.

You can also have multiple commands on a single command line. To see all of the Modulator and Demodulator IF and Data Settings and to set the Modulator IF output on, you would use the command line:

```
m500ctl 1 mi ? di ? md ? dd ? mie 1
```

Available switches are “-v” for “verbose”, “-r” for raw mode output and “-sp” for “show packet”, which is used for debugging to see the actual commands sent and received between the SnIP and the modem. There is also a Time output for the status commands only in verbose mode using the “-t” switch. To see the currently available mnemonics use the command “m500ctl -c”.

The order of switches are important. They should be used in the order: -sp, -v or r or x, -t.

6.1.1 Special Outputs – Raw Status and Status Elements

There are a few special cases to note in the commands and responses and how they can be used:

6.1.1.1 Raw and Verbose Outputs

The default output format for two letter commands, or any command that results in multiple response items is verbose. To force these outputs to a raw data format that would be machine readable, use the “-r” switch. In this mode each output parameter is separated by a carriage return. Then a language like Perl, which the SnIP includes, can directly import the information into a standard array.

6.1.1.2 Status Outputs

The 4 status commands, us, ms, ds and is contain information useful for multiple applications, for example when polling. The raw and verbose outputs of these commands have 3 elements at the beginning of the responses:

- ❑ **Events**
- ❑ **Alarms**
- ❑ **Tests**

In addition the verbose outputs of these four commands can optionally display the current time before the Events output by adding the “-t” switch. The time is in seconds since January 1, 1970 (GMT) which is the default date on power up without an rdate change (this date has something to do with Linux). In 2007 that number is approx 1,169 Million. The “-t” switch is used before the

address and after the “-v” switch if used. Raw outputs assume a computer with its own time is used to read the data, and adding a variable may modify array settings.

The Event flags are one time read flags and indicate if a change has occurred in this parameter since the last read. The read is therefore destructive, resetting the flag to 0. The Alarm and current Test flags are live, indicating the current status.

The verbose outputs group these with labels, and indicate only those currently active. The raw outputs have 3 numbers as the first 3 returned scalar values which contain bit variables for the individual status elements as shown below.

Unit Status Raw Output Events - Integer Scalar Value 0

- Bit 0 – Modem
- Bit 1 – Reference
- Bit 2 – Redundancy Switch
- Bit 3 – Name
- Bit 4 – Test

Modulator Status Raw Output Events - Integer Scalar Value 0

- Bit 0 – Carrier
- Bit 1 – Data
- Bit 2 – Bit Clock
- Bit 3 – Test

Demodulator Status Raw Output Events - Integer Scalar Value 0

- Bit 0 – Carrier Lock
- Bit 1 – Eb/No
- Bit 2 – Carrier Level
- Bit 3 – Carrier Offset

Interface Status Raw Output Events – Integer Scalar Value 0

- Bit 0 – IO Mode
- Bit 1 – RTS
- Bit 2 – CTS
- Bit 3 – DCD
- Bit 4 – DTR
- Bit 5 – DSR
- Bit 6 – Test
- Bit 7 – BER Test

To determine the status of any one bit you simply “AND” the value with the appropriate bit value. An example use of the Event flags would be to determine if any reads were necessary. First read the unit status and if the first value is 0 then nothing has changed since the last read. But, if the value were 1 then that would mean that the modem had a value change. You would then read the mod and demod status to see what changed. If the mod status event value was 0 and the demod event value was 1 then the demod carrier lock had changed. Reading further within the demod status would show if the carrier was now locked or unlocked.

Unit Status Raw Output Alarms - Integer Scalar Value 1

- Bit 0 – Mod
- Bit 1 – Demod
- Bit 2 – Reference
- Bit 3 – Clock
- Bit 4 – Redundancy
- Bit 5 – No Backup (valid only in Redundancy)
- Bit 6 – Backup in Alarm (valid only in Redundancy)
- Bit 7 – Over Temperature

- Bit 8 – Mod Hardware Failure
- Bit 9 – Demod Hardware Failure
- Bit 10 – FEC A
- Bit 11 – FEC B
- Bit 12 – Optional Interface Failure
- Bit 13 – No Send Data Activity
- Bit 14 – No Receive Data Activity

Modulator Status Raw Output Alarms - Integer Scalar Value 1

- Bit 0 – Carrier
- Bit 1 – Data
- Bit 2 – AUPC at Limit
- Bit 3 – AUPC Fail
- Bit 4 – Data Timing
- Bit 5 – No Data Activity
- Bit 6 – Clock
- Bit 7 – Output Level
- Bit 8 – LO Alarm
- Bit 9 – Synth Step Alarm
- Bit 10 – System Alarm
- Bit 11 – Reference Alarm
- Bit 12 – OCXO Alarm (L-Band only)
- Bit 13 – FEC Alarm

Demodulator Status Raw Output Alarms - Integer Scalar Value 1

- Bit 0 – Eb/No
- Bit 1 – AGC
- Bit 2 – Data
- Bit 3 – Level
- Bit 4 – LO Alarm
- Bit 5 – Synth Step Alarm
- Bit 6 – System Alarm
- Bit 7 – Reference Alarm
- Bit 8 – Backward Alarm
- Bit 9 – Buffer Slip

Interface Status Raw Output Alarms – Integer Scalar Value 1

- Bit 0 – Interface
- Bit 1 – Optional Interface Failure
- Bit 2 – Optional Interface in Reset

The third standard variable returned to the raw status commands shows the current test modes in process.

Unit Status Raw Output Tests - Integer Scalar Value 2

- Bit 0 – Unit Test Active
- Bit 1 – Mod Test Active
- Bit 2 – Demod Test Active
- Bit 3 – Interface Test Active

Modulator Status Raw Output Tests - Integer Scalar Value 2

- Bit 0 – Pure Carrier Test Mode
- Bit 1 – Alternate 1/0 Carrier Test Mode
- Bit 2 – SideBand Carrier Test Mode

Demodulator Status Raw Output Tests - Integer Scalar Value 2

- Bit 0 – IF Loop-Back Enabled

Bit 1 – Fixed Carrier Mode

Interface Status Raw Output Tests – Integer Scalar Value 2

Bit 0 – Terrestrial Loop-Back Enabled

Bit 1 – Satellite Loop-Back Enabled

Bit 2 – Modulator Pattern Generator Enabled (BER)

Bit 3 – Demodulator Pattern Generator Enabled (BER)

Bit 4 – Demodulator Pattern Generator Locked (BER)

6.1.2 A Unique m49ctl and m500ctl Case - Redundancy

There are no special connections required if the SnIP is used for control in a system where it is not also used as the Data Interface except for one case. Occasionally however one may want to use the SnIP for control where the synchronous serial interface is used in a redundant mode. A modem with a SnIP installed will normally not allow enabling redundancy because the same serial port is used both for redundancy communications between the modems and for SnIP internal communications. If this is the desired connection then a special internal cable between the SnIP and modem must be installed that removes both the serial port and the SnIP identification lines. The modem then does not know that a SnIP is installed and it therefore is not an available interface option. The SnIP to host modem communication must now be connected via an external RS-485 cable.

6.1.2.1 Redundancy Control via “m49/m500ctl”

The m49ctl and m500ctl applications include redundancy control parameters, but are only usable in this special case stated above. There are several redundancy control limitations, which are stated in the main manual, and repeated here for reference.

- ❑ One unit of a redundant pair can send its full configuration to the other unit in the pair. But, the “Send Configuration” command, “urc 1” can only be executed from the currently On-Line unit. The configuration will be sent even if the Off-Line unit is in alarm.
- ❑ The command to switch to the redundant unit, “uss 1”, is essentially switch away from the currently On-Line unit to the currently Off-Line unit. It can only be issued to the unit that is currently On-Line. There is no effect to telling the Off-Line unit to switch. The switch will also fail if the currently Off-Line unit is in alarm.

Other items that are not obvious is that the Unit Redundancy packet/command, “ur ?”, does not contain the information of which unit is currently On-Line. That information comes from the Unit Status request, “us ?”. Likewise the “urs” command does not control which unit is on-line, instead it is a control for what results in a switch request, whether any alarm, Alarm A, Alarm B, etc. The On/Off-Line status of a unit and the request to switch from a currently On-Line unit to its backup is done from the Unit Status. Thus to switch, the command is “m500ctl 1 uss 1”. The “-v 1 uss ?” verbose query returns not only the On/Off-Line status, but also the backup status if the unit is currently On-Line and its own status as ready to go On-Line if currently Off-Line.

Unit address setting discipline is a great help in redundancy control. In order to know the address of both units in a redundant pair it is strongly suggested that adjacent odd-even address pairs be used. For example one pair would use the addresses 1 and 2, and the next pair would use 3 and 4. If Telnet control is being used the same type of numbering scheme might be used with the IP address used to access the SnIP cards.

The M5 and M500 modem’s built in redundancy operates on a non-priority basis. That means that there is no preference for which unit is On-Line if both are out of alarm and operating normally.

A proper sequence for bringing up a redundant pair might be as follows:

1. Set the Unit Address of the first unit of a pair to an odd number, e.g. 1. Then set the other unit’s address to the next even number, e.g. 2.
2. Set each unit into 1:1 redundancy mode. “m500ctl 1 ure 1” and “m500ctl 2 ure 1”.
3. Set the unit parameters on the first unit for normal operation and insure that all alarms are off. If the alternate is in alarm, this should place this unit On-Line.

4. Insure that the first unit with the correct configuration settings is On-Line, if necessary, go to the other unit and force it offline with the “uss 1” command.
5. Send the first unit’s configuration to the backup with the command “m500ctl 1 urc 1”.
6. The backup unit should now go out of alarm, and be ready for redundancy service.

6.1.3 m49ctl and m500ctl Command Set

The currently implemented m500ctl commands can be viewed by using the m500ctl program itself and typing

m500ctl -c

Which will result in a display similar to the one below.

```
[snip2:0 ~] m500ctl -c
M500 Modem Control Version 0.04
  Parameter to Command Letter Codes Reference
  2 letter codes print verbose packet full info,
  3-4 letter codes print/set individual parameters

  Unit                Mod                Demod                Int
  us - Status          ms - Status          ds - Status          is - Status
  uss - Red Switch     mi - IF              di - IF              isbr - Reset BERT
                        mif - Freq           dif - Freq
                        mio - Offset         dia - Acq Range
                        mil - Cxr Level
                        mie - Cxr Enable          io - Int IO
  uc - Config          mim - Modulation     dim - Modulation     iom - IO Mode
  ucs - Store Cnfg     md - Data            dd - Data            ie - Ethernet
  ucr - Recall Cnfg    mdr - Data Rate      ddr - Data Rate      iep - IP Address
  ucyl..8 - Delay      mdf - FEC Type       ddf - FEC Type       iem - MAC Address
                        mdo - FEC Option     ddo - FEC Option
  uo - Ref Osc         mdc - Code Rate      ddc - Code Rate
  ux - Remote          mdfec - FEC T,O,C    ddfec - FEC T,O,C
  uxp - Protocol       mdrs - R-S Mode      ddrs - R-S Mode
  uxt - Port Type      mdrp - R-S n,k,d     ddrp - R-S n,k,d
                        mds - Scrambler      dds - DeScrambler
  ur - Redundancy      mdk - Clock Source   ddk - Clock Source
  ure - Red Enable     ddd - Buffer Delay
  urc - Send Config    ddb - Buffer Bits
  urs - Switch On      dl - LNB
  urh - Hold Time      mbf - BUC LO         dlf - LNB LO
                        mbe - BUC Pwr En     dle - LNB Pwr En
                        mbr - BUC 10MHz Ref  dlr - LNB 10MHz Ref
  ut - Test            mt - Mod Test        dt - Demod Test      it - Test
  uts - Self Test      mtm - Test Cxr Mod   dtl - IF Loopback    itber-BER Tx,Rx,Dir

[snip2:0 ~]
```

The four columns here represent commands for the Unit, Mod, Demod and Interface, and as can be seen each command begins with the first letter of one of these names.

Additional commands may be implemented in the future. Notice that the commands follow the general format of the front panel controls in that they consist of 3 elements to define a particular parameter. In the Excel worksheet idiom they are page, column and row, where page is either Unit, Mod, Demod or Interface (the 4 buttons on the front panel labeled Unit, Mod, Dem, Intf). The rows and columns are symbolic of the up/down and left/right front panel arrow controls. As explained in the modem manual the scheme is synonymous with moving to a particular Excel spreadsheet page, then cell row and column using a computer’s keyboard arrow keys.

Three of the commands, Mod Data FEC <mdfec>, Demod Data FEC <ddfec> and Interface Test BER <itber>, are unique in that they require 3 parameters separated by commas when changing.

Make certain that you do not have any spaces in the parameters. The FEC commands also have commands to set the individual elements, but the 3 parameter version is recommended because it avoids the possibility that one value is set but not allowed because of the others currently set. The data FEC values can be read directly from the Modem FEC Modes document available on the web or in the manual.

For the M500 there is additionally the **mdmfr** and **ddmfr** commands which set 5 related parameters at once for complete data setup. They are the numeric selections for **M**odulation, **F**EC **T**ype, **F**EC **O**ption, **F**EC **C**ode Rate and **R**eed-Solomon (acronym MTOCR for short). For example "m500ctl 0 mdmfr 11000" sets the modulator to QPSK, Viterbi, Standard Code rate 1/2 without R-S. This one command plus the IF frequency (mif/dif) and data rate (mdr/ddr) are all that is normally required to set up a modem. The MTOCR can be entered as for example 32001 or more properly 3,2,0,0,1.

The M5 modem controls available in the m49ctl program are very close to those shown above for the m500ctl program. The main difference is that the M5 modem's FEC controls are simpler than those in the M500.

6.1.4 Other Control Manipulation

Since the SnIP is running Linux, this implementation has multiple other possibilities to manipulate anything controllable from the command line, including the m49 and m500ctl programs. These are the bash script and Perl. Both contain powerful processes for executing commands and using the output. Indeed, the web server uses Perl to issue m500ctl commands. It then takes the responses in raw mode and places them into variable arrays. Those arrays are handed off to JavaScript arrays when a web page is created, allowing display of current values.

No attempt is made here to show how this is done.

6.2 Web Server Monitor and Control

The SnIP also runs a web server by default. The web server can be used to control the SnIP itself, the SnIP Interface functions including bridging and routing and the host modem plus others connected to the external RS-485 port.

The web interface is accessed at the IP Address that has been set for the modem.

7.0 The Web Server

The SnIP contains a full web server designed by appWeb. It allows creation of almost all normal types of web pages including html, cgi, esp (unique to appWeb). It also includes support for Javascript, Perl, PHP (not implemented), CGI, etc. The basic SnIP contains a web site that allows monitor and control of the SnIP, Linux and modem functions.

You can also write your own web site to be hosted by and run on the SnIP.

7.1 Basic Web Operations

The web site hosted by the SnIP can be accessed by using a standard browser and pointing the URL to the SnIP's IP address. For example if the SnIP is configured with an IP address of 192.168.15.83 then to access this SnIP web you would type into the web browser address "192.168.15.83" and press return.

By default the web server has two levels of security implemented, basic and ssl. The basic web site is accessed as described above and a window will pop up requesting your user name and password. The default for these is "root" and "datum" respectively unless you have changed them. The URL address in the browser will show "http://192.168.15.83".

There is also an ssl implementation of the web site at "<https://192.168.15.83>". This not only uses the ssl protocols, but also uses an Authorization protocol named "Digest" by default. Digest authorization will only be properly handled by Firefox and Internet Explorer version 7 browsers, and not Internet Explorer version 5 or 6.

The authorization and other configuration options for the web server are controlled via the file "/etc/appWeb/appWeb.conf" in the SnIP. Information on the options are available using Google with "appWeb.conf".

8.0 FAQ and HowTo Collection

In the style of Linux documentation we have a collection of frequently asked questions or FAQ and also a "HowTo" for common tasks that many users desire. Remember also that the SnIP uses the Linux operating system, and there is a wealth of information available on the web to help with programming, the command line and bash shell, Perl, conventions etc.

8.1 FAQ - Frequently Asked Questions

Following are brief answers to questions we have received:

Do I have to do anything special when starting or stopping the SnIP?

The SnIP was designed to be, and is very rugged, both in hardware and software. We have found nothing to be cautious about or any special processes required when powering up or down the SnIP. However, an unpowered PC connected to the console port may act like a keypress and hold the SnIP in the bootloader on power-up. Also if some PC's are left connected to the console port when turned off, they can cause the bash console application to occupy 99% of the processor's time serviceing the dead port.

If the SnIP is not the selected interface, but it is in bridge or router mode, then a connected Ethernet traffic will attempt to be sent to the modem data channel, which is turned off. This will cause Linux to occupy about 99% of its time serviceing a soft interrupt. Both of these are not apparent to the user, unless you view the current processes using the "top" command.

What is the difference between bridging and routing?

Bridging is an OSI model Layer 2 process that is intended to connect two segments of the same LAN together having common IP addresses within the same LAN mask. The bridge function only looks at MAC addresses. An example might be a branch office or a single computer separated from the main LAN by a satellite link.

Routing is intended for use in a transmission system between two or more LANs with different Addresses. It operates on the OSI Model Layer 3, working with IP Addresses.

What is the difference between static and dynamic routing?

Routing is intended for use in a transmission system between two or more LANs with different Addresses. It operates on the OSI Model Layer 3, working with IP Addresses.

The SnIP has the multiple forms of router like protocols as follows:

- ❑ Brouter – special combination bridge and router using ebttables.
- ❑ Static router – The standard Linux Kernel implemented static routing tables that are fixed and manually entered into the route table by hand or from the static-routes file.
- ❑ Mroute – the mrouted daemon is a multi-cast router.
- ❑ Dynamic "routed" router. This is the basic Linux kernel RIPv1 and 2 protocols.
- ❑ Dynamic Zebra and Quagga routers. These routers implement RIP, RIPNG and OSPF protocols. It also emulates the operating format of a Cisco router.

Dynamic routers communicate with other routers and determine their routing tables semi-automatically based on information sent back and forth between the routers.

How long does the SnIP take to boot up?

The SnIP typically takes less than 30 second to boot, including starting the web server.

Why do I get a message on the console that says Netdev Timeout...?

Currently if the SnIP is enabled and the modem's carrier is turned off then this message will appear everytime the network protocols try to send a packet to the modem port. We will have the ability to suppress these type messages in a future release or Tech Note. Linux reports this message because the CTS signal goes down when the carrier is not enabled. The message can be avoided by using the front panel or the web interface to force the CTS Active. If you do this from the front panel you will have to first select a different interface such as RS-449, to make this option visible.

Why do I get an error message on the console during boot up when Linux sets up the HDLC channel that says "SIOCSIFFLAGS: No Such Device"...?

If the SnIP is not the currently selected interface by the modem, then Linux does not see the interface as viable, and reports no such device. Hdlc0 does not appear in ifconfig unless the "-a" option is used and then does not show it as "UP". Selecting the SnIP interface should resolve the issue.

Why do I get a message on the console that "Unable to connect to remote host (address): No route to host?"

By default the SnIP is set up to connect to a time server on the internet to get the current time. Until both the IP address is properly set and a gateway to the internet is available, it cannot connect with the time server. If you will not have any connection to the internet, the "rdate -s ..." line in the /etc/profile.d/baseconfig.sh file can be commented out to remove this from the boot procedure. In software versions after 0.5.21 the rdate function is separate and controlled by the "/etc/rdate.conf" file. Edit this file to control the time server used.

Ifconfig shows no IP Address for either eth0 or hdlc0

You are probably running in the bridge LinkMode. The bridge sets it constituent interfaces' IP Addresses to zero and takes on the address of the last interface added to the bridge.

After saving new 'configwiz' settings the console displays "iep (iem) change failed or already set".

The configwiz script attempts to change the modem's setting for the IP Address (iep) and the IP Mask (iem) if these are changed in configwiz. They may have been already set to the values that were changed to.

When I ping a URL I get no or a bad response

This could have a number of causes. Some possibilities are:

- If you cannot ping even the local LAN at a known address then you have no connection or your IP is not within the local mask.
- If you can ping the local LAN, but not the other end of the link then the bridging or routing setup is not correct.
- If you can ping locally but not a static IP on the web, then you have no or a bad gateway to the web.
- If you can ping a static address on the web, but not a named address, then you probably have bad or no DNS server definitions.

Does the SnIP implement RIP or RIP2 routing protocol?

Yes – You must set up either the "routed" Linux router or the Zebra/Quagga routing configuration.

Does the SnIP have the PHP language installed?

Not at the present. PHP would occupy over 1 Mbyte of space which was not currently deemed necessary. It does however contain a fairly complete implementation of Perl.

Can I compile my own “C” programs to run on the SnIP?

Yes, but be forewarned that it is not a simple task for the novice. Even if you are familiar with programming or using gcc on a PC, SnIP programs are both cross-compiled (for the Freescale powerpc MPC875 processor) and are compiled against uClibc to conserve memory in an embedded system. See the Resources at the end of this document to see how the cross-compiler can be created.

Does the SnIP’s rear panel USB connector do anything?

Not yet.

Can I modify or create my own web site to be hosted by the SnIP?

Yes you can. The appWeb server is similar in functionality to the standard Apache web server. The same rules apply to creating and maintaining a web site. The web is contained in the “/var/appWeb/web” directory.

8.2 *HowTo – Common Tasks*

Following are short descriptions of how to accomplish common tasks on the SnIP. More detailed information is available on the web for these and other tasks.

8.2.1 *How To Use the “nano” Editor*

The “nano” text editor is an easy to use full terminal screen editor that is probably more comfortable for Windows users. If you already use and prefer the “vi” editor then the SnIP has that capability too, and you should need no introduction.

The configuration files and scripts, which control the SnIP, can be modified using the “nano” editor. Nano requires that the environment variable “TERM” be changed in advance from the default vt102 to vt100. It is also required on a Linux computer when using the virtual terminals from Telnet or SSH. Use the command:

```
export TERM=vt100
```

Currently the standard profile script /etc/profile.d/baseconfig.sh already issues this command. If using nano via telnet from a Windows Hyperterminal session, insure that Hyperterminal properties are set to VT100 also.

The nano editor is a full screen simulated editor on a terminal, and is fairly competent. The basic commands are “nano filename” to load (or create) a file for editing, “^o” (Control key and the letter “o”) to save the modified file, and “^x” to quit or exit nano.

8.2.2 *How To Set the SnIP for your Time Zone*

Your systems’ default Time Zone is set in the file “etc/TZ”, which is persistent and used on each boot up. It should be modified with your particular Time Zone info. You can also change the current Time Zone on the fly using the command”

```
export TZ=<time zone designation> for example export TZ=CAT+1 for Azores
```

The time zone designations have several formats but a common one is

```
TZ=std, offset, dst
```

where “std” is the standard, like PST, MST, EST, BST, then “offset” is the offset in hours from GMT and lastly “dst” is optionally used if the zone uses a form of daylight savings time. Here are some common values to use either with the export command or persistently in the /etc/TZ file are:

- GMT Greenwich Mean Time
- PST8PDT Pacific Standard Time, offset of 8 hours and use Daylight Saving Time.
- PST+8PDT Same as line above.
- EST5EDT Like above but for Eastern Standard Time
- EET-2 European Eastern Time, offset of 2 hours (note entered as “-2”
- BST-6 Central Asia, note the “-“ again.

- AET-10EADT Eastern Australia with Daylight Savings Time.

More are available by Googling on the web, and one site with more information is:

http://souptonuts.sourceforge.net/how_to_linux_and_open_source.htm

You should note two items specifically. The offset is the negative of that shown on sites like this, for example PST is 8 hours behind GMT or -8, but is entered into TZ as 8 or +8. The other is that the TZ file must have a line feed at the end or it may not work correctly. It also appears that if you just enter any 3 letter code only, the result is always GMT, that is you must at least also enter the offset value.

8.2.3 How To Automatically Set the Date & Time with “rdate”

The SnIP does not have a battery backed real time clock and each time it is powered on the date is reset to some value usually in 1970. You can manually set the date using the date command in the format:

date MMddhhmm[yyyy][.ss]

Once set the SnIP keeps very accurate time for months. You read the date and time by simply typing “date” at any console connection.

If the SnIP has been setup with internet access then the date and time can be automatically set each time the SnIP is powered up. The rdate program requires the ip address of a time server only to work. The command is

rdate -s <ipaddress>

rdate will then contact the specified time server, get the current date and time and set the SnIP’s clock. A common US time server is 128.6.15.28. Another is 69.25.96.13, but you should find one local to your area of operation. Root Filesystem versions prior to 0.5.22 used a very simple call to the rdate function in the “/etc/profile.d/baseconfig.sh” script which would run for each login. From filesystem 0.5.22 on a more complete sequence is used which calls the rdate function from the inittab sequence. It looks for the time server IP address or URL in the “/etc/rdate.conf” file.

There is one other file that you will probably have to modify to get the correct time setting. That is the “**etc/TZ**” file which contains a designation for the physical time zone you are located in. The default stored in this file is for the Pacific Standard Time with Daylight Savings Time activated. You should find the code for your area and edit this file with either that or the GMT designation if desired.

8.2.4 How To Set Up Quagga to Run on the SnIP

As of this writing the version of Quagga installed on the SnIP is 98.6. This may change to a later version when compiling is possible. It is very important to read the Quagga documentation at [www.http://quagga.net](http://www.quagga.net). Quagga consists of multiple modules, one for each protocol and one for the zebra manager, plus an additional virtual tty shell named “vtysh”. These are also known as daemons in Linux. Currently the /etc/init.d/network script starts up the components specified in the file named “/etc/quagga/daemons” and connects it only to the internal loopback or “lo” port. This default configuration allows access to the root user logged into either the console or a telnet/ssh session via Ethernet.

To specify which router protocol daemons run on startup you edit the file named “/etc/quagga/daemons”. This file contains one line for each protocol and a value setting of “yes” or “no”. As an example setting if you wished to run the RIP protocol daemon you would edit this file to set zebra=yes and ripd=yes and all other protocols =no. You must always set the zebra daemon to yes if any other protocol is set to yes. Alternately set the zebra protocol and all others to “no” to turn off all Quagga/zebra functions.

After selecting the desired routing protocols, each selected protocol should be edited in their respective configuration file in the “/etc/quagga” directory. For example the RIP daemon protocol configuration is contained in the “/etc/quagga/ripd.conf” file. Refere to the Quagga documentation for details of how to set these files up. The configuration in these files can also be set from within the vtysh or the individual protocol telnet sessions.

Once configured there are multiple ways to login to the Quagga sessions. Most if not all will work either from the console or a telnet/ssh session.

- ❑ At a login prompt enter Quagga and the default password of zebra to enter the vtysh shell.
- ❑ From within an existing session type “vtysh” to enter the vtysh shell.
- ❑ From within an existing session type “telnet 127.0.0.1 260x” to enter one of the running daemons.

8.2.5 How To Create New User Accounts on the SnIP

Refer to Section 1.0.9 at the beginning of this document.

8.2.6 How To Specify a Domain Name Server (DNS)

Many computers get their IP Address and pointer to a DNS from a gateway or DHCP server. With a fixed IP Address you would need to supply a pointer to a DNS in order to resolve names to IP Addresses. This is done by creating a file in the “/etc” directory named “resolv.conf”. The contents of this file would be similar to the following which you could create with nano as follows:

```
[snip3:1 ~] nano /etc/resolv.conf    (command to create file)
nameserver 205.166.226.38
nameserver 69.67.108.10
search localdomain
```

Of course the nameserver IP Addresses above need to be real ones for your system, which could possibly be determined from a local computer on the LAN. If your local LAN computers are Linux devices then you can see their settings by typing:

```
cat /etc/resolv.conf
```

If they are Windows computers you can check the Settings→Network and Dialiup→Local Area Connection→Internet Protoco(TCP/IP) properties, but that probably says to get it automatically. So then go to the Gateway or Router device that provides DHCP and use their procedure to find the DNS addresses. Again these addresses would have been provided by the ISP.

The SnIP has a script in the /root directory that will create a default “/etc/resolv.conf” file. You will however still have to edit the file created to place your own nameserver IP Address(es) inside. To create the sample file from the root directory type:

```
./mk-resolv.sh
```

The “./” (dot-slash) before the script name is required because, unlike Windows, Linux does not include the current home directory in the “Path”, so this notation tells Linux to get the named file from the current directory.

The default DNS configuration file created by the script lists several public DNS Server address that you may choose from if you do not have your own DNS servers from your ISP. To edit the file for your particular use you can use the nano editor to comment out all but two addresses on lines beginning with the keyword “nameserver” that are in your geographic area. To comment out a line causing it to be ignored, place a hash symbol “#” as the first character on the line.

Using different server addresses from those on the other computers on the local LAN will still work, but some operations seem to take longer when they are different, for example the “scp” operations.

More information is available in the "dns-readme.txt" file in the root directory. Information on Public DNS Servers can also be obtained by Googling with that term. One site that shows what appears to be relevant Public servers is:

www.tech-faq.com/public-dns-servers/shtml

Several of the public addresses provided in the sample resolv.conf file that can be created came from this site. It is still recommended to use the addresses provided by your ISP.

8.2.7 How To Do Almost Anything in Linux

This is not a Linux manual, but an incredible amount of information can be found on the web using a search engine such as Google.

8.2.8 How To Recover the Default Configuration of a SnIP (Disaster?)

As the root user in Linux it is possible to seriously damage the files, configuration, scripts, etc. Luckily it should not be easily possible to destroy the boot-loader that can be used to reload and re-burn to flash the entire Linux kernel or the entire Root Filesystem.

If Linux still runs then the procedures in Section 5 of this manual may be usable. If not then Contact Datum Systems to get a copy of the latest kernel and or Root Filesystem plus a special script file which the bootloader can use to automatically load and burn to flash the whole Linux system. You will need a TFTP server within your network, and have placed the script (named "burna") and the kernel and root filesysem images into the TFTP directory. Then follow the directions given in Section 10.1 below.

9.0 SnIP – Basic Commands for Status and Administration

Linux contains several built-in test programs to check out a systems health and current status. Below of few of these are shown which will allow you to determine if particular hardware and software are working the way you intended.

9.1 Checking the Current SnIP Configuration

Use the command “**config**” to see the SnIP’s current persistent configuration.

9.2 Checking the Ethernet Connection – “mii-diag”

The “mii-diag” program checks the proper functioning of the Media Independent Interface or “MII” that is part of the rear panel Ethernet connection. The program is called by simply typing “mii-diag” at either a console or telnet/SSH session terminal. Typical results might be:

```
[snip3:0 ~] mii-diag
Using the default interface 'eth0'.
Basic registers of MII PHY #0: 3100 782d 0013 78e2 05e1 c5e1 000f 2001.
The autonegotiated capability is 01e0.
The autonegotiated media type is 100baseTx-FD.
Basic mode control register 0x3100: Auto-negotiation enabled.
You have link beat, and everything is working OK.
Your link partner advertised c5e1: Flow-control 100baseTx-FD 100baseTx
10baseT-FD 10baseT, w/ 802.3X flow control.
End of basic transceiver information.

[snip3:0 ~]
```

Notice above on line 4 that it shows that the negotiated setup is 100baseT and is Full Duplex. This particular switch that the SnIP was connected to is actually capable of 1000baseT, but the SnIP’s 10/100 interface MAC cannot tell that. Note also line 6 which tells you that everything is working OK.

9.3 Checking the Interface Configuration – “ifconfig”

The “ifconfig” program displays and controls the different interfaces on the SnIP (or any Linux computer). The program is called by simply typing “ifconfig” at either a console or telnet/SSH session terminal. Control and selective display are possible by adding parameters after the ifconfig on the invocation line. It is important to note that changes made using the ifconfig command are local and not persistent and may conflict with setting in the persistent startup configuration files and scripts.

There is also a quick program written by Datum Systems that will simply show the current IP Addresses available to the kernel using the command “getip”.

Typical base results for a SnIP without a bridge running might be as follows:

```
[snip3:0 ~] ifconfig
eth0      Link encap:Ethernet  HWaddr 00:19:78:00:00:06
          inet addr:192.168.15.83  Bcast:192.168.15.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:203 errors:0 dropped:0 overruns:0 frame:0
          TX packets:92 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:17867 (17.4 KiB)  TX bytes:6698 (6.5 KiB)

hdlc0     Link encap:Ethernet  HWaddr 00:01:2F:B1:82:7A
          inet addr:192.168.55.3  Bcast:14.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Base address:0x3f00

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:22 errors:0 dropped:0 overruns:0 frame:0
          TX packets:22 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:638 (638.0 B)  TX bytes:638 (638.0 B)

[snip3:0 ~]
```

Notice that 3 interfaces are shown, the rear panel Ethernet interface named “eth0”, the internal serial synchronous connection to the modem named “hdlc0” and the standard Linux local loopback connection named “lo”. The third line of each shows that the interface is “UP” meaning that it is currently enabled.

The “Hwaddr” shown for the eth0 and hdlc0 are the MAC addresses. The eth0 is a unique Datum Systems owned MAC address, while the hdlc0 MAC address is automatically and randomly generated by Linux.

When a bridge is currently created and operational, a 4th interface will appear as br0. The bridge interface takes over eth0 and hdlc0 whose IP addresses are set to 0, while the bridge takes over the normal IP address, MAC address and possibly the default gateway of eth0.

For control of the interfaces via ifconfig please refer to Linux documentation.

9.4 Checking the Current Route Status – “route” and “iproute”

The Linux “route” program displays and controls the different static routes on the SnIP (or any Linux computer). The program is called by simply typing “route” at either a console or telnet/SSH session terminal. Control and selective display are possible by adding parameters after the route on the invocation line. The iproute2 version is “ip route” or “iproute” either with an optional “show”. Typical base results of the route command for a SnIP without a bridge running might be as follows:

```
[snip3:0 ~] route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
192.168.15.0     *                255.255.255.0   U        0      0      0 eth0
14.0.0.0         *                255.0.0.0       U        0      0      0 hdlc0
default          192.168.15.1    0.0.0.0         UG       0      0      0 eth0
[snip3:0 ~]
```

Notice the default gateway shown on the third line of the table. In this case it says that anything with an unknown IP address will be sent to the gateway address. In this case the gateway is the address of a DSL modem which is connected to the internet.

9.5 Checking the Current Network Status – “netstat”

The Linux “netstat” program displays the current network status of the interfaces on the SnIP (or any Linux computer). The program is called by simply typing “netstat” at either a console or telnet/SSH session terminal. Selective display is possible by adding parameters after the netstat on the invocation line. Typical base results for a SnIP without a bridge running for the more complete “netstat -l” command might be as follows:

```
[snip3:0 ~] netstat -l
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp      0      0 192.168.55.3:http      :::*                    LISTEN
tcp      0      0 192.168.15.83:http     :::*                    LISTEN
tcp      0      0 localhost:http         :::*                    LISTEN
tcp      0      0 *:ftp                  :::*                    LISTEN
tcp      0      0 *:ssh                  :::*                    LISTEN
tcp      0      0 *:telnet               :::*                    LISTEN
tcp      0      0 192.168.55.3:https     :::*                    LISTEN
tcp      0      0 192.168.15.83:https    :::*                    LISTEN
tcp      0      0 localhost:https        :::*                    LISTEN
netstat: no support for `AF_INET6 (tcp)' on this system.
udp      0      0 localhost:9473          :::*                    LISTEN
netstat: no support for `AF_INET6 (udp)' on this system.
netstat: no support for `AF_INET6 (raw)' on this system.
Active UNIX domain sockets (only servers)
Proto RefCnt Flags      Type           State         I-Node Path
[snip3:0 ~]
```

In this case you can see that both the eth0 and hdlc IP addresses have an http and https component. Https represents the secure protocol.

9.5 Checking the Current Network “iptables” Status – “iptables”

The Linux “iptables” program is used to both control and display the current network filter status of the interfaces on the SnIP (or any Linux computer). The program is called by simply typing “iptables” followed by appropriate commands at either a console or telnet/SSH session terminal. Selective display is possible by adding parameters after the iptables on the invocation line. Typical base results for a SnIP using the List command, “iptables -L” or “iptables -list” might be as follows:

```
[snip8:1 ~] iptables --list
Chain INPUT (policy ACCEPT)
target      prot opt source                destination

Chain FORWARD (policy ACCEPT)
target      prot opt source                destination
DROP        all  -- anywhere              10.9.13.0/24
DROP        all  -- anywhere              10.9.14.0/24
DROP        all  -- anywhere              10.9.15.0/24
DROP        all  -- anywhere              10.9.50.0/24
DROP        all  -- anywhere              10.9.51.0/24
DROP        all  -- anywhere              10.9.52.0/24
DROP        all  -- anywhere              10.9.53.0/24
DROP        all  -- anywhere              10.9.54.0/24
DROP        all  -- anywhere              10.9.61.0/24
DROP        all  -- anywhere              10.9.62.0/24
DROP        all  -- anywhere              10.9.63.0/24
DROP        all  -- anywhere              10.9.64.0/24
DROP        all  -- anywhere              10.9.65.0/24

Chain OUTPUT (policy ACCEPT)
target      prot opt source                destination
[snip8:1 ~]
```

In this case only the Forward chain of the three default chains is populated.

9.7 Checking the Modem Status

You can use the `m49ctl` or `m500ctl` commands to check the modem. For example typing:

```
m500ctl 0 us ?
```

See Section 6 on the use of the modem control programs to find out much more about the modem.

9.8 Linux Basic Commands for Administration

Here is a list of commands to make you feel like a Linux “Guru”. They are common commands used by many Linux users and administrators: No real detail is given here, and the commands previously covered are only briefly listed if at all. A common technique to find out how to use a command is to first type the command with an option of “—help” which will usually give some usage information. Next go to the web and Google for “linux + command name” to get more information than you may want.

Remember too that this is all running from less than ½ of a 16 MB flash emulating a hard disk and a similar portion of 32 MB of RAM. It still does most of a full Linux system on a normally much larger platform. A few items did have to be left out.

File Related Commands:

- ❑ **ls [-l]** – list files in a directory.
example: **ls -l /etc/config**
- ❑ **cp source destination** – copy a file from source to destination location.
example: **cp -i mdm-find /tmp** (note interactive option to avoid mistakes)
- ❑ **mv** – Move or rename a file.
example: **mv -i /tmp/mdm-find /tmp/find-modem** (rename)
- ❑ **rm** – Removes or deletes a file

- ❑ **which** – finds a file in the current PATH. Some systems name this find or locate.
example: **which sethdhc**
- ❑ **touch** – Set a file's date and time or create an empty file.
- ❑ **cat** – Concatenate or send a files contents to a file or device. Usually displays a file's contents. It's handy to pipe this to "less" for longer files.
example: **cat readme.txt** or **cat readme.txt | less**
- ❑ **grep [options] pattern [file...]** – looks for the pattern in the file(s).
example: **grep root /proc/cmdline**
- ❑ **sed [options] pattern [file...]** – stream editor for patterns in the file(s).
example: **sed root /proc/cmdline**
- ❑ **awk [options] pattern [file...]** – complex file manipulation sometimes described as a combination of grep and sed.
example: **awk root /proc/cmdline**
- ❑ **df -h** – Show disk (flash in this case) usage.
- ❑ **free** – Show RAM usage.
- ❑ **wget** – Get a file(s) from a web http or ftp location.
- ❑ **chmod** – change a files permissions.
example: **chmod 755 mdm-find**
- ❑ **chown** – change a files owner.
- ❑ **mkdir** – Make a new directory.
- ❑ **cd** – Change to a new directory.
example: **cd /usr/sbin**
- ❑ **pwd** – Display the present working directory.
- ❑ **scp** – Secure copy described in other sections of this guide.
- ❑ **rsync** – Remote sync of files – a sophisticated file copy.

Currently Running

- ❑ **who** – Displays the list of people currently logged onto this SnIP.
- ❑ **ps** – shows the current processes running with info about there resource usage.
- ❑ **cat /proc/modules** – shows the current kernel modules running.
- ❑ **top** – A formatted repeating "ps". Quit using "q" or "ctrl c".
- ❑ **uptime** – Time since last reboot with some load info.
- ❑ **date** – Display or change the current date and time.
- ❑ **rdate** – Get the date and time from a remote server.
- ❑ **watch <command>** - repeating run of command.
example: **watch ifconfig hdhc0**
- ❑ **time <command>** Displays the time taken to run any command.
example*: **time nslookup datumsystems.com**
- ❑ **reboot** – Shut down and then restart the SnIP.

Network Commands

- ❑ **netstat [-r] [-l]** – Alone or with one of these options gives different results.
- ❑ **ifconfig** – Interface status and configuration.
- ❑ **route** – Show or modify the kernel routing table.
- ❑ **iproute** – Show or modify the routing table.
- ❑ **ip [address | link | route | tunnel]** – Show or modify the routing tables.
- ❑ **iptables** – Show or modify the network filter chains.
example: **iptables --list**
- ❑ **ebtables** – Show or modify the network bridge filter chains.
example: **ebtables --list**
- ❑ **brctl** – Bridge control.
example: **brctl showmacs br0**
- ❑ **sethdhc** – Allows changing the hdhc0 ports protocol.
- ❑ **tc** – Traffic Control. Linux kernel software for traffic shaping and prioritizing.
example: **tc show dev eth0**

- ❑ **nslookup <URL>** - Look up an address.
example*: ***nslookup yahoo.com***
- ❑ **ping** – Ping a URL.
example: ***ping 192.168.22.104***
example*: ***ping datumsystems.com***
- ❑ **traceroute** – Trace the connection to a specific URL.
example: ***traceroute 192.168.54.82***

***Note: The commands with asterisks use named URLs and require Internet access and that the "/etc/resolv.conf" file exist and contains valid IP Address(es) of Name Servers.**

And there are a significant number of other commands available, like grep, gawk, awk, etc, etc. To see a list of the main executable files use the "ls" command on the 4 directories in the default path:

```
ls /bin
ls /sbin
ls /usr/bin
ls /usr/sbin
```

Other applications may also exist in the /usr/local/ or the /opt directories.

There are also the commands meant to be used with "pipes" represented by the "|" character. For example typing "**ls -l /etc**" will result in more than can fit on the screen. You can use "**ls -l /etc | more**" which waits for a keypress at each screenfull, or even better "**ls -l /etc | less**", which allows you to use the up and down arrow keys or page up and page down keys. Type "**q**" when you are finished viewing either of these examples to return to the prompt. Yes this works over telnet!

The web is an excellent training resource, and sites like "www.linux.org/lessons" have tutorials for beginners and intermediates. The Linux Documentation Project at "http://tldp.org" is another of the literally hundreds of resources available.

10.0 SnIP Technical Information

The SnIP uses the Linux operating system, and has several built-in and module type drivers added to it for the various interfaces and processes. There are also applications that add capabilities. Below is a current listing of those items that are unique or compiled specifically for the SnIP and where the software to access them resides. Note that most of the standard Linux programs are also available with the exception of “X-windows” and GUI related or desktop items, since the SnIP only has a command line interface.

Process/Application	Compiled as	Compiler/Library	Location	Program/Module Name	Device Name
Boot Loader	Stand Alone	gcc for powerpc	mtd partition		
Linux Kernel	Kernel	gcc for powerpc	mtd partition	vmlinuz	
Root Filesystem	Stand Alone	BuildRoot gcc using uClibc	mtd partition	snip875jffs2.img	
Console port	Kernel	gcc	Kernel		ttyCPM0
Internal Modem Control Port	Kernel	gcc	Kernel		ttyU0
External Modem Control Port	Kernel	gcc	Kernel		ttyU1
Ethernet Port	Kernel	gcc	Kernel		eth0
Modem Data Port	Module	gcc	/lib/modules	snip_raw_hdlc	eth1
Modem Data Port	Module	gcc	/lib/modules	snip_gen_hdlc	hdlc0
HDLC port control	Application	gcc / uClibc	/usr/bin	sethdlc	hdlc0
Modem Control Keep-Alive Poll	Daemon	gcc / uClibc	/usr/bin	mdm-poll	--
Web Server	Application	gcc / uClibc		appWeb	
Web files			/var/appWeb /web	*.html, .cgi, .pl, .esp	
Modem CLI Control	Application	gcc / uClibc	/usr/bin	m49ctl or m500ctl	
Web Server Link to Modem Control	Application	gcc / uClibc			
Quagga/Zebra Router Suite	Application daemons	gcc / uClibc	/usr/sbin, vtysh in /usr/bin	zebra, vtysh, ripd, ripngd, ospfd, ospf6d, isisd	
Iptables manage and control	Application	gcc / uClibc	/usr/sbin	iptables, iptables-save, iptables-restore	
Multi-Cast routing control	Application	gcc / uClibc	/usr/sbin	mrouted, mrouted-nc	
SNMP	Application	gcc / uClibc	/usr/sbin	multiple	

Notes:

The data port can have different names depending on the application that registers it with Linux when the module is loaded. The dumb but efficient module that only is capable of Ethernet encapsulation of data on the modem channel names the port “eth1”. The more sophisticated module that uses Krzysztof Halasa’s generic HDLC routines names the port “hdlc0” and is capable of encapsulating the data packets on the modem channel as either protocols Ethernet, HDLC-Ethernet, Cisco, PPP, Raw. Most of these are not usable, but the real reason for creating a special module with this access is for the Cisco embedded WAN protocol.

The capability to run a bridge is built into the Linux Kernel. The bridging application is set up to operate with all ports and traffic directions enabled. This configuration was created with a default set of commands contained in a script similar to the one below:

- ❑ `brctl addbr br0`
- ❑ `brctl addif br0 hdlc0`
- ❑ `brctl addif br0 eth0`
- ❑ `ifconfig eth0 0.0.0.0`
- ❑ `ifconfig hdlc0 0.0.0.0`
- ❑ `ifconfig br0 [eth0's original ip address]`
- ❑ `ifconfig br0 up`

In the above lines we used the bridge control program, `brctl`, to create a bridge called `br0` and add the two interfaces `eth0` (Ethernet) and `eth1` (satellite hdlc port) to this bridge. Then we made the original port ip addresses 0 to eliminate them and set the bridge's IP address to what the Ethernet port's address was originally. That last is necessary so that the web server, telnet and any other access method to the SnIP are still viable. Note that we set up the normal Ethernet connection `eth0` after the hdlc port. We do that because the bridge takes on the ip address and MAC address of the last interface added to the bridge.

The normal bridge script that runs if "BRIDGE=yes" is set in the configwiz program also takes the given "Gateway" IP address from `eth0` and assigns it to the new `br0` interface, allowing programs like `rdate` to properly set the date and time if set up.

10.1 The SnIP Bootloader

The bootloader used in the SnIP is part of the Denx Embedded Linux Development Kit or "ELDK" and is named U-Boot for universal bootloader. U-Boot is a fairly sophisticated small operating system. A significant amount of work went into customizing U-Boot to work with the SnIP875, especially its unique HDLC and SPI-UART functions which are not supported by the native U-Boot setup.

One main difference between U-Boot and Linux is that all devices in Linux are interrupt driven, while they are mainly polled in U-Boot.

To work in the bootloader you connect a terminal (like Hyperterminal or `gtkTerm`) to the console port at 38.4kbps,N,8,1 and power on the modem. Before the timeout for booting the kernel you press a keys on the keyboard to halt the Linux boot process. In the current bootloader the keys are "qt" because that does not interfere with some potentially connected applications

Once you are stopped in the bootlaoder there is information on line, and also just by typing "help" at the U-Boot command line. The console command line in U-Boot is also somewhat similar to that in Linux, except that it is strictly limited to a local console directly connected to the SnIP's serial port.

Note that U-Boot itself was originally loaded into the SnIP using a special "BDM" interface (Freescale's advanced JTAG type interface) tool used in development and manufacturing. The location of U-Boot is protected so that it would be extremely difficult to damage it or make it unusable.

10.1.1 Using the SnIP Bootloader to reload the Linux Images

U-Boot can be used to download and program both the Linux kernel and the Root Filesystem into the SnIP. Both of those images must be available on a TFTP server on the same LAN as the SnIP. The TFTP server will offer a specific directory for TFTP transfers, which is typically "tftpboot" on a Linux server or perhaps TFTP-Root on Windows Whatever that directory is the file

“burna” must be in that directory and then another directory named “snip6” must exist containing the SnIP’s kernel named “ulmage”, and Root Filesystem named “jffs2.img”.

The SnIP requests files from the IP Server. Before you can download any files from the server, U-Boot must be told what its own IP Address is and the server IP Address is with the commands:

```
setenv ipaddress 192.168.1.82 /* Replace with your desired address */
setenv serverip 192.168.1.104 /* Replace with your server's address */
```

You can always see what the current environment variable are using the U-Boot command “printenv”. You can also save these settings and changes with the command “saveenv”. It is a good idea to make certain that you have Ethernet connectivity by using the U-Boot ping command as in “ping 192.168.1.104”, or whatever the server address is.

Then the following two commands can be used to download a simple script named “burna” from the server. This auto script will then download the kernel and root filesystem images. The commands are: (just type the first part, not that within the comment tags /* */)

```
ftp 100000 burna /* download the script - note 5 zeros after the 1 */
autoscr /* This executes the script just loaded */
```

The auto script is verbose and will display its process as it runs. The kernel download and burn to flash takes only a few minutes, but the Root Filesystem is approximately 6 Mbytes and will take about 5 minutes. Be patient, and the script will display each of 7 steps that it goes through. After the 7th step is completed a final display will be:

=> done

when the process has completed. Then you type the U-Boot command “**reset**” to restart the entire boot process from scratch, this time allowing the new Linux image to boot.

The first time Linux boots up several processes can take a variable amount of time. An initial “Configure Shared Libraries” can take about 30 seconds. Specifically the “Creating DSS (and RSA) Key for SSH...” can take from 10 seconds to 3 minutes to complete.

One main reason for covering this much on the bootloader is that it is currently the main method for downloading the Linux image and the root filesystem.

10.2 The Linux Boot Process

The Linux boot process is somewhat unique among different distributions of Linux. In this one the basic process is as follows.

On initial power up or a reboot command within Linux, the first item to be loaded is the bootloader (currently U-Boot). The bootloader waits approximately 5 seconds for a keypress which would halt further loading. If no keypress it found it starts to decompress and load the Linux kernel image from flash memory into the dynamic RAM.

The Linux kernel then loads and installs all of its drivers that are built into the kernel itself. If a terminal is connected to the console port during this process the bootloader and Linux startup messages can be viewed.

Once the Linux kernel is running it must decompress and install the root filesystem, which is the same as the disk drive in a full PC type computer.

Linux then goes to the “inittab” file located in the “/etc” directory of the root filesystem. This file contains a list of startup and shutdown routines or daemons to call. The shutdown is generally in the reverse order of the startup.

After mounting the filesystem image, the inittab then goes to the “/etc/init.d/initconfig” file which checks to see if the file “/etc/config/system” exists. If it does it exits with a success return. If not it calls the “configwiz” shell script to set up the /etc/config/system” file. This file contains some basic information like the Host name of the SnIP, its IP address and netmask, and whether it is to be used as a bridge or a router. This information is used by later scripts and daemons in the inittab sequence.

Next, the /etc/init.d/sysconfig” shell script runs. It first loads the “etc/config/system” file which contains the defaults for several parameters. It then accomplishes the following:

- A. Builds Library dependencies.
- B. Builds Library cache
- C. Sets kernel variables – calls /etc/config/sysctl.conf
- D. Sets Hostname

The etc/init.d/network” shell script then starts up the network and does several things including

- A. Sets the eth0 IP address.
- B. If the HDLC port has a non-null IP address then it loads the module driver to implement the “hdlc0” interface and sets its IP address.
- C. Checks for the file “etc/config/static-routes” and if found executes the “route add” command on the applicable lines (any line not beginning with “!” or “#”. This may include setting the default gateway and fixed routes. Note that when implemented the Quagga/Zebra router sets up dynamic routes.
- D. Checks for the Zebra/Quagga configuration files and sets up the dynamic routing parameters according to the configurations contained in those files.

One of the last items run from the inittab list is the modem polling daemon. It is only run if the configwiz setup resulted in the line “MDMPOLL=yes”. This is a background daemon that checks to see if the SnIP card is currently selected as the main data interface. If it is then periodic polling messages are sent to the modem from the SnIP via the internal control channel as a keep alive. Without this keep-alive the modem will report that the SnIP has failed.

11.0 SnIP Known Bugs and Issues

11.1 Bugs

The SnIP uses mainly standard Linux drivers and functions in its operating system. Sometimes those drivers and functions may have unresolved problems themselves. We try to identify and notify users of those problems in advance as well as attempting to fix them. Of course the SnIP has several unique programs that may have bugs or issues too.

11.1.1 Eth0 does not properly notify bridge of status on re-connect

The bridge (not router) mode exhibits an odd behaviour if the Ethernet cable is removed and re-inserted. Both the ifconfig and bridge seems to be aware that the connection has been restored, but the bridge is no longer functioning correctly.

Filesystem versions prior to 0.5.34 (June 2008) have no automatic correction of this condition. To restore proper operation Linux must be given the command “ifconfig eth0 up” even though it claims that it is already “up”. The command can be issued remotely, so even if this occurred on a remote unmanned site, proper operation could be restored.

By connecting a switch or hub to the SnIP's Ethernet connection, other devices on the other side of the switch can be removed and installed without this happening.

11.1.2 IP Address Sync between Modem and SnIP not reliable

The proper operation would seem to be that the Modem displays the IP address and mask that are set in the persistent configuration of the SnIP. And if the IP Address is changed on the Modem's front panel then it would change in the SnIP's current and persistent configuration. It does not yet work that way, and something is wrong with the procedures. They are under review now.

11.2 Not Finished Yet

This may be a permanent category because there is so much possible that could be added to the SnIP, both in software and hardware.

11.2.1 Web Server

The web server itself is a production item, but the web pages that it serves are in very preliminary stage. Look for big improvements in this area soon.

11.2.2 Dynamic Router Suite - Quagga

Quagga appears to be compiled and running, but no time has yet been devoted to setting it up and testing. This is one of those "knowledge intensive" areas.

11.2.3 Software Update Programs

The SnIP could really use a Windows or Browser based software update facility. Right now updates are done by copying new programs directly into the SnIP, which is a normal method for Linux gurus, but the rest of us want an easier and less error prone method.

11.2.4 External USB Connection

Almost no work has been done on this yet. It should work but needs drivers.

11.2.5 Internal Compact Flash Socket

Very little has been done with this socket yet. Its intent is to allow third parties to develop proprietary software that can be run from the SnIP's hardware and software base. For example Network Management Systems are a natural candidate.

This socket could also be used to hold a larger development system allowing more sophisticated programs that do not fit in the large but limited space of the base SnIP. For example, the gcc compiler, PHP, SNMP, databases, etc.

12.0 Resources

The SnIP uses the Linux operating system. If you are working on a Linux PC communicating with the SnIP then most if not all of the software required is already available. This first section describes how to set up a new easy to use Linux development environment for the SnIP. If you are working on a Windows PC then many programs are not commonly provided as part of the setup. Go to section 12.2 which shows some open source software that is available to run on a Windows computer to aid in working with the SnIP.

12.1 Linux Programming Resources

Since the SnIP is running Linux it makes sense to do development of the kernel, drivers, daemons and applications on a PC already running Linux. Most of the initial development work on the SnIP was done on a PC running Linux from the Fedora Core 4 and 5 distributions, which are free to download.

Virtually all of the kernel and drivers were developed using a semi-packaged set of cross-compilers available from www.denx.de and called the ELDK, short for Embedded Linux Development Kit. The majority of that was done on ELDK versions 3 and 4.0 which had no provisions for developing a compact Root Filesystem. The Root Filesystem was built using the smaller “uClibc” libraries instead of the common “glibc”, and developed on an old MAC with a PowerPC processor providing a native compiler environment. Later work was done on an implementation of a cross compiler on a PC created using the BuildRoot development system which is tightly coupled with BusyBox. This last is extremely difficult to reproduce.

Luckily the latest version of the ELDK, version 4.1 is now available as of 2007 with a version containing a built-in pre-created PowerPC C and C++ cross-compiler using the desired uClibc libraries. **Important – this is version 0.9.27 of uClibc.** That is what the SnIP has and all uClibc programs must use the same version. Programs cross-compiled against another version of uClibc will not run properly.

If you want to do fairly straightforward programming on the SnIP, remember that it already contains the powerful Bash scripting language and the miniPerl version of Perl, so it is already highly programmable using just the SnIP itself. For some applications however you would normally use either C or C++. If you understood any of the previous and want to develop C or C++ applications for the SnIP, the following procedure can be used to create the proper environment on a standard PC running one of the supported distributions of Linux. You will need about 1 Gbyte of free space for the ELDK environment. **We cannot tell you how to develop or program!**

First, go to the denx site at www.denx.de and download the iso image of the ELDK for Freescale processors and uClibc. From the denx or mirror ftp site you would first select the directory for “ppc-linux-x86-uclibc” and from that page select the “iso” directory. Then from the listed iso images select the one named “ppc-2007-01-21-uclibc_freescale.iso”. Currently this page is at [ftp://ftp.denx.de/pub/eldk/4.1/ppc-linux-x86-uclibc/iso/](http://ftp.denx.de/pub/eldk/4.1/ppc-linux-x86-uclibc/iso/) although that may change or be different at one of the mirror sites. When you have downloaded this file it must be burned to a CD which is then used to install the software. We only want to install the minimum required for SnIP development which is only the ppc_8xx processor version. After creating a new directory folder at “/opt/eldk41u”, I used the install command as follows:

```
$ /media/cdrecorder1/install -d /opt/eldk41u ppc_8xx
```

This command says to run the install program on the CD putting only the files for the ppc_8xx processor into the directory /opt/eldk41u created above. There is nothing magic about the name or location but it is where denx suggests and I add the version info of 41u so I know which one it is. This is about 800k+, but if you install additional processor families it will go up.

The html file on the CD has the necessary information needed to set the \$PATH and \$CROSS_COMPILE environment to identify the proper cross compiler. You should read all of the information provided with the ELDK.

Programs to be compiled could be kept either in your home directory or somewhere accessible in the /opt/eldk41u directory or anywhere else you choose with the proper permissions. You should be able to install and run the cross compiler without root or superuser privileges.

Programs you create can easily be loaded into the SnIP for testing using the “scp” methods shown earlier.

12.1.1 SnIP Kernel Development

We do not recommend this, however if you have an idea for modifications to the kernel and its resources then contact us. There are also about 20 files required to modify the kernel compiler to recognize and use the SnIP. We use the ELDK from www.denx.de for development.

12.2 PC Program Sources

The SnIP uses the Linux operating system. If you are working on a Linux PC communicating with the SnIP then most if not all of the software required is already available. But if you are working on a Windows PC then many programs are not commonly provided as part of the setup. This section describes some open source software that is available to run on a standard computer to aid in working with the SnIP. These are all currently free.

One Linux program that would probably have to be installed is “gtkTerm” which is a simple serial terminal emulation program. In Linux Ubuntu you may have to kill the brlty process and in Fedora you may need to add yourself to the uucp group to get gtkTerm to attach to serial ports.

12.2.1 Windows Programs for the SnIP’s Linux world

Communicating with the SnIP from a Linux computer is fairly easy for users familiar with the Linux environment. Oddly though there are some programs available for Windows that are easier to use than equivalent tools in Linux. Some of those tools are listed below – and they are all free.

=> Windows SSH/Telnet Client – **PuTTY**.

What: Small and very capable implementation of Telnet and SSH for Windows.

Where: <http://www.chiark.greenend.org.uk/~sgtatham/putty/>

=> Windows Graphical Secure Copy and SSH Suite – **WinSCP** I wish this was available for Linux. The latest versions seem to also support FTP.

What: Looks almost like a graphical FTP application for scp. Very Nice!

Where: <http://winscp.net/eng/index.php>

=> Windows Terminal Emulator – **Hyperterminal** – Built in but sometimes exhibits strange behaviour on some computers!

=> Windows TFTP Server – **PumpKIN** by Klever Group

What: very small and capable graphical TFTP server for Windows.

Where: <http://kin.klever.net/pumpkin/>

=> Windows TFTP Server – **tftpd32** by Phillipe Jourdin

What: very small and capable graphical TFTP server for Windows.

Where: <http://tftpd32.jounin.net/>

=> Windows RSYNC Client/Daemon – “rsync” is a mainstay of Linux. In Windows almost everything seems to involve running Cygwin to achieve rsync capability. You might be better served using Winscp until something respectable is available. Suggestions welcome.

What:.

Where:

=> Windows Programming and General Purpose Editor – “**Notepad++**” is a very simple, small, fast but powerful editor for Windows that is a replacement for the standard Notepad. It is small but has some wonderful features unmatched by most big editors.

What:. Programming and general purpose editor. Very Nice!

Where: <http://notepad-plus.sourceforge.net/uk/about.php>