DATUM SYSTEMS

M500/M7 Modem Technical Note

SnIP Interface Update for M7 SnIP Compatibility

Revision History

Rev 0.1	3-29-2015	Preliminary Release.
Rev 0.2	4-6-2015	Initial Release.

Overview

The Datum Systems M500 has had the SnIP interface for several years. With the recent release of an M7 series version of the SnIP the software and filesystem required several changes for compatibility between the two units.

The major changes were the addition of a new modem polling routine used specifically for the M7 modem plus modifications to most of the initialization and configuration scripts to determine and accommodate the particular modem in which installed. These two changes are transparent to the user.

During the process other capabilities have been added in a mainly standardized fashion and once the update is installed become available in both versions of the SnIP. Those additions include:

- The ability to set the M500 SnIP and M7 SnIP to a router protocol using Ethernet encapsulation where previously it was only Cisco protocol. This is significant because it is not only useful by itself between any two SnIPs but also allows an M7 with I7 IP interface to talk to an M500 with SnIP interface.
- A standardized method of creating virtual alias interfaces and VLAN interfaces both temporarily and persistently surviving reboot.
- A standardized method of adding at least one static route rule via either the kernel route method or the more flexible iproute method.
- The ability to set the hdlc (modem WAN) port protocol to Point to Point protocol (PPP) in addition to the existing HDLC-eth, Cisco and Frame Relay protocols.
- In the M7 modem the SnIP is now capable of determining what slot it is installed in. That information is needed in M7 dual demodulator configurations with dual SnIPs.

This document further describes all of the changes implemented by the update for M7 compatibility between these interfaces.

The update package is named “update-m7snip_0.6.34-x” where x is currently 4. The update is small but in the future it will be incorporated into a new file-system version 0.6.35 and above. Instructions on how to install this package into SnIPs in the field are given later in this document.

1.0 WAN Protocols--

The SnIP devices use a combination of a built in basic hardware HDLC interface to the modem plus the standard Linux "Generic" HDLC software protocols to generate the different protocols. The link protocol is manually changed by a Linux routine named "sethdlc" which can be used to set and read protocols on the WAN interface, hdlc0. When setting protocols the hdlc0 interface must be put in the down state first, changed and then brought up.

You can also see the current WAN protocol with the command "sethdlc hdlc0".

1.1 Available WAN Link Protocols --

Link protocols are intended to be compatible with devices on the other end of a link that share the same protocol. All are only compatible with another IP interface card except the Cisco HDLC protocol which is compatible with either another SnIP in that mode or a sync serial connection to an external router via a synchronous serial connection. For Cisco routers these are commonly called a WAN interface card or WIC, or a High Speed Serial Interface or HSSI. This type connection is becoming less common and is not currently available in other interfaces.

The lower case protocol name in parenthesis is the key used by sethdlc to set that protocol, for example "sethdlc hdlc0 hdlc-eth".

- **HDLC** – (hdlc) This mode is not used in any current scenarios.
- **HDLC-Eth** – (hdlc-eth) Possibly the most useful mode which encapsulates Ethernet packets in HDLC and looks to the rest of the network stack like a normal interface. Bridging will only work in this mode normally.
- **Cisco HDLC** – (cisco) This mode was used exclusively for router applications as it offers the Cisco WIC/HSSI compatibility mentioned above.
- **Frame Relay** – (fr) Used in specialized systems where the ability of the address matching hardware built into the processor makes it possible to build point-to-multipoint networks that filter out unneeded packets at a higher rate than the software can handle. Setup can be complex although we have configuration options built into the SnIP.
- **PPP** – (ppp) or point-to-point protocol Not currently used since the generic hdlc routines do not provide the useful compression control available in a full PPP protocol.

Currently the SnIP uses hdlc-eth for bridge mode and cisco for routine mode. After the update it uses hdlc-eth for bridge or routing mode and alternately cisco for routing mode depending on settings.

1.2 *Summary of IP Interfaces and Link Protocols --*

The currently available Datum IP interfaces and the possible WAN protocols for each are shown in the following table.

Modem-Interface	M7 Flex HDLC (1)	HDLC-Eth (2)	Cisco HDLC (3)	Frame Relay	HDLC PPP (4)	PPPoE
M500-SnIP	No	Yes	Yes	Yes	Yes	No
M7-SnIP	No	Yes	Yes	Yes	Yes	No
M500-I5 (5)	No	No	No	No	No	No
M7-I7	Yes	Yes	No	No	No	Yes
M7-E7	Yes	Yes	No	No	No	No

Notes:
1. The M7 Flex-HDLC is a custom mode that has a 32 bit CRC and allows overhead channel packet insertions plus other features.
2. HDLC-Eth is HDLC encapsulated Ethernet packets. In the M7 this is selected as “M500 Mode”.
3. Cisco HDLC is a serial protocol adapted to the modem interface, which is serial, by the Linux Generic HDLC process. It includes the Cisco keep-alive or SLARP packets.
4. HDLC PPP is the type of PPP produces by the Linux Generic HDLC process and does not include the link control mechanisms or compression possible with regular PPP.
5. Currently the I5 interface is only compatible with itself.

2 Operation

The updated software procedures are similar to previous but allow more capabilities as described here. In addition Section 2.2 discusses use with remote M500/SnIP modems.

2.1 *Selecting The Link Mode or WAN Link Protocol --*

This can be done from the modem front panel, from a SnIP console connection and via the web interface:

What you should notice about all of the possible settings is that now the terms “erouter” and “crouter” are used to denote an open or router type interface configuration with hdlc-eth or Cisco type WAN interface protocols respectively. However, until the M500 modem software is changed the front panel only has one “Router” selection and the choice of hdlc-eth or Cisco protocol must be made in the SnIP.

2.1.1 M7 or M500 Modem Front Panel

In the M500 modem you have the same choices as previously, this was done to retain compatibility with the modem code going back several years:

1. None – Used when the SnIP is only used for monitor and control.
2. Bridge – Used to place the SnIP in a bridge mode connecting the LAN and WAN ports which essentially extends to the other end of the link if it is also placed in bridge mode. The entire link from local LAN to remote LAN
3. Router – This is an open mode which in previous versions set the link protocol to Cisco HDLC, and therefore only compatible with another SnIP ins the same mode or a Cisco WIC/HSSI interface connected to the remote end sync serial interface.

   In the new update SnIP code a new variable set in the SnIP itself determines whether Cisco or hdlc-eth protocol is used in router mode. This Cisco_Proto variable can currently only be set by a console or telnet/ssh session using “configwiz”. The setting is “sticky” and changes to bridge and back will retain this setting.
4. Pass – A now unused mode originally intended to work in point to multi-point links. In future updates this may be used for the erouter option even though the modem software is not updated to reflect the name erouter.

In the M7 modem you have the similar choices but it directly supports the crouter and erouter nomenclature:

1. None – Used when the SnIP is only used for monitor and control.
2. Bridge – Used to place the SnIP in a bridge mode connecting the LAN and WAN ports which essentially extends to the other end of the link if it is also placed in bridge mode. The entire link from local LAN to remote LAN
3. Crouter – This is an open mode which sets the link protocol to Cisco HDLC, and therefore only compatible with another SnIP in the same mode or a Cisco WIC/HSSI interface connected to the remote end sync serial interface.
4. Erouter – This is an open mode which sets the link protocol to HDLC-Eth, and compatible with another SnIP ins the same mode or an M7 modem with I7 interface set in any mode.

Notice that the M7 version can fully specify the router mode from the front panel while the M500 version requires an additional configuration setting in the SnIP itself to determine the router protocol. This may be changed in future versions of M500 code.

2.1.2 M7 or M500 Using Configwiz

The configwiz program is used from a console session and is necessary in the M500 SnIP to set the Cisco_Proto variable as it cannot be set from the modem front panel:

```
# configwiz
At the prompt use the “m” key to select Link Mode.
Select either b, c, e or n for Bridge, Cisco router, Ethernet router, or none.
Select s and enter to save the file.
```

When you select “c” the Router mode is selected and the Cisco protocol is set. When you select “e” the Router mode is selected and the hdlc-eth protocol is selected.

The configuration process is the same in the M7 version of configwiz and it also keeps track of the Cisco_Proto variable but the resulting /etc/config/system file uses the notation crouter and erouter while the M500 SnIP only has the Link Mode router option. This however should be transparent to the user.

2.1.3 The “reconfig” script

The reconfig script has been rewritten to work in both the M500 and M7 versions of SnIP plus provide more options on configuration in line with the new link protocol needs. Although applicable for general users it is normally used by other scripts to change the current configuration, such as when the modem front panel is changed from bridge to router mode. The reconfig script also keeps track of the LAN interface especially and tries to maintain the same IP Address and MAC when changing. Additionally it can be used to specify custom settings in user scripts.

Reconfig now has the following options and meanings:

1. 'bridge' – Builds a bridge named br0 from eth0 and hdlc0 -\n hdlc-eth protocol on hdlc0. br0 takes on the eth0 IP addr, netmask and MAC.
2. 'crouter' - tears down any bridge and leaves eth0 and hdlc0 ready for routing. hdlc0 set to Cisco hdlc protocol compatible with previous default on M500 with SnIP.
3. 'erouter' - tears down any bridge and leaves eth0 and hdlc0 ready for routing. hdlc0 set to hdlc-eth protocol compatible with M7 modem with I7 interface
4. 'eth' - will change an existing hdlc0 Cisco protocol to hdlc-eth router.
5. 'cisco' - will change an existing hdlc0 hdlc-eth protocol to Cisco router.
6. 'moa' and 'hrouter' are the same - standard hdlc for non-Cisco router. Note this is hdlc and not hdlc-eth which is hdlc Ethernet encapsulated. Only for special use.
7. 'flags' - hdlc protocol with idle flags (experimental - for some routers).
8. 'none' - is used for monitor/control, setting removes hdlc driver.
9. Alt Usage: reconfig <save|save-current|save-last|restore|load-last|reset>
10. 'save' or 'save-current' or 'save-last' saves a copy of the configuration.
11. 'restore' or 'load-last' restores the configuration from the saved one.
12. 'reset' loads the factory default configuration and reboots.

2.1.4 The SnIP Web Interface

The M7 SnIP web interface still requires significant changes to work properly in the M7 modem application as the entire modem related areas are now unused. Only the SnIP portions are still applicable because of the router nomenclature changes. The M7 modem has it own modem web interface accessible via the M7 Ethernet control port.

The M500 SnIP web interface also requires changes to work with the new erouter and crouter nomenclature.

2.1.5 The SnIP SNMP Interface

The M7 SnIP SNMP interface still exists in the M7 version but it only loads the standard Net-SNMP portion and not the modem agent. The M7 modem has its own modem monitor and control SNMP agent accessible via the M7 Ethernet control port.

2.2 Working with Remote M500/SnIP Modems --

A potential problem exists in working with existing remote M500 modems with SnIPs using the current, non-updated software because without the update they are always in Cisco router protocol unless operating in Bridge mode. Remote modems are sometimes not easily physically accessible and if communications are lost while making changes then recovery is difficult. Some scenarios are:

1. An M7 modem with an I7 interface cannot link protocols with an M500 SnIP in router mode because the I7 interface only operates with the hdlc-eth protocol.
2. An M500 modem with an I5 interface cannot link protocols with an M500 SnIP in router mode because the I5 interface only operates with the hdlc-eth protocol.
3. An M7 or M500 modem with a SnIP interface in erouter mode cannot link protocols with an M500 SnIP in router mode.

The only solution to incompatibility 1 or 2 is to apply the update-m7snip package to the remote modem and set it into erouter mode. More information on accomplishing this is given in Section 3.

The solution to incompatibility 3 is either to use crouter mode in the local SnIP or to apply the update-m7snip package to the remote modem and set both into erouter mode.

2.3 Persistent Alias and VLAN Interfaces --

Previous SnIP filesystems did not include a standardized method for creating persistent Alias or VLAN interfaces for the LAN, eth0, interface. Such structures had to be built by a user script that was called at boot time. Although possible this method was not intuitive and confusing.

The update-m7snip package includes the directories and methods that are common in modern Linux systems for configuring these type interfaces.

Note that the following methods are not intended for the main eth0 and hdlc0 physical interfaces because those are handled by configwiz and reside in the /etc/config/system file allowing modification by the modem front panel and via the modem polling routines.

2.3.1 Creating Manual Temporary Alias and VLAN Interfaces

On a temporary basis an Alias, sometimes called virtual interface is created by the Linux command:

```
# ifconfig eth0:x ipaddr/24
```

And a VLAN interface by the commands:

```
# modprobe 8021q
# vconfig add eth0 vid          [vid is the vlan id, e.g. 27
# ifconfig eth0.27 10.10.10.1/24]
```

Some notes on manually creating these interfaces:

1. A virtual alias interface is denoted by a colon between eth0 and the interface number, e.g. eth0:3 while a VLAN interface is denoted by a period between eth0 and the interface vid number, e.g. eth0.110.
2. Do not give a VLAN id of 0, and also not normally 1 as these have special meanings.
3. The ifconfig on the last line of each not only assigns the ipaddress and mask it also implicitly includes the command ifconfig eth0yy up, bringing the interface to the up state.
4. The 8021q package for VLAN is installed in the SnIP, but not loaded. Once the update-m7snip package is installed it is automatically loaded. It only needs to be called once before the first VLAN interface created.
5. We are using the shorthand CIDR method for defining the netmask with a /xx after the ipaddress. You could also include “netmask 255.255.255.0” for example to set the same value as /24.

Virtual or Alias ports can be used for various purposes. For example the SnIP uses them when in redundant mode to create an interface that can be turned on and off easily and therefore is useful for the data interface connecting into a LAN that is switched. You can also build bridges from Alias ports just like real physical ports and they do not limit further port configuration. When a physical port is enslaved to a bridge it can no longer be used for other purposes.

VALNs are also very useful in building separate private networks. The one disadvantage of a VLAN port is that it will only accept or send VLAN tagged packets, and is not useful as a general purpose port.

2.3.2 Creating Persistent Alias and VLAN Interfaces

As mentioned above the update-m7snip package include the resources needed to build persistent Alias and VLAN interfaces without resorting to custom scripts. What this package adds for this function is.

- The standard /etc/network files and directories including the /etc/network/interfaces file containing default auto port setup.
- Additions to the /etc/init.d/network script to automatically load the 8021q module needed for VLANs.
- Simple processing for the instructions in the /etc/network/interfaces file when structured as a standard Linux Debian interfaces file.

This means that most of the rules and settings in the /etc/network/interfaces file are handled just like in Debian and most instructions found on the web will work in the SnIP.

The following examples are from the default /etc/network/interfaces file installed by the update-m7snip package, although they are commented out in the default file as supplied with a single "#".

```
## add vlan 700 on eth0 - static IP address
## VLAN requires loading the 8021q module
auto eth0.700
iface eth0.700 inet static
   address 10.100.10.77
   netmask 255.255.255.0

## add virtual (alias) 2 on eth0 - static IP address
## Note that this also creates a static route using iproute
## instead of the normal kernel 'route add -net ...'
auto eth0:2
iface eth0:2 inet static
   address 10.11.1.21
   netmask 255.255.255.0
   up ip route add 10.11.3.0/24 via 10.11.1.1 dev eth0 src 10.11.1.21
```

The network script on boot with create an interface preceded by the keyword "auto". The keywords "inet static" tell it that the following address and netmask are static addresses. Also note the space before the parameters set for the interface. I do not know if these are necessary, but always seem to be used on examples.

The first example creates a VLAN for interface eth0 with an id of 700 and brings it up with an address of 10.100.10.77 and netmask of 255.255.255.0. It could also be given a static route as shown in the second example.

The second example creates an Alias for interface eth0 with an id of 2 and brings it up with ad address of 10.100.10.77 and netmask of 255.255.255.0.

The second example also shows creating a static route for the alias interface using the iprout2 method of "ip route add ipaddr ...". It could also have been specified using the kernel "route" method with the pattern "route add -net <netaddr>...". Note the wording here begins with "up".

3 Update Package Installation

This software update is provided in an ipkg package named "update-m7snip_0.6.34-x_powerpc.ipk". The revision x is replaced by the current revision number. For refernce "ipkg" is the name of the SnIP's update/package manager that can handle changes to the programs and files in the Linux file-system.

3.1 Working with Local M500/SnIP Modems --

If your modem has an Internet connection and proper gateway and DNS settings then the update package can be installed from the web repository with the commands below from a console, telnet or ssh session.

```
# ipkg update
# ipkg install update-m7snip
# successfully terminated
```

and you should see the successfully terminated response. The preceding # indicates you are in root mode. Note here the package name is just update-m7snip and the remainder is included in the file name for full identification.

The SnIP web Update page also has links to updating packages.

Many systems do not have Internet access to the SnIP and would then use manual installation. That is commonly accomplished in a local modem/SnIP by using scp (from a Linux PC) or WinSCP (from a Windows PC) to first transfer the file to the /tmp directory in the SnIP. Then from a SnIP console/telnet/ssh session use the following command:

```
# ipkg install /tmp/update-m7snip_0.6.34-x_powerpc.ipk
# successfully terminated
```

and you should see the successfully terminated response. It is not necessary to type the full package name as the command line will fill it out after a few letters of the name by hitting the [Tab] key, for example after the upd hit tab. The preceding # indicates you are in root mode. Note that if ipkg is given a path and full file name then it gets the package locally rather than from the web repository. Note here that we put the package into the /tmp folder where the package is lost on reboot, but the changes remain. Although this package is very small we do this especially for larger packages so that it does not occupy valuable space in the Flash memory.

3.2 A method for Installation in Remote M500/SnIP Modems --

All of our IP interface modems running Linux have the scp (Secure Copy) and telnet programs available, so it is possible to easily use a local modem linked to the remote modem to update the remote. Here is one method that works. You will need to know the local SnIP's LAN IP address *<local-snip-ip-addr>* and the IP address of the hdlc0 port of the remotely linked SnIP *<remote-snip-hdlc0-ip-addr>*.

Note in the following method we are using telnet to log into the local SnIP. You could also use the console connection via RS-232 or SSH.
From a local PC download the update-m7snip program from our web site at …

1. Use scp from the local PC if its Linux or WinSCP if its Windows to transfer the update

package to the local SnIP in the /tmp directory (or any other if you desire). A typical scp command would be
"scp update-m7snip_0.6.34-x_powerpc.ipk root@*<local-snip-ip-addr>*:/tmp/"

2. Telnet or SSH into the local SnIP as root providing the user name and password root and "datum" unless changed.
   "telnet *<local-snip-ip-addr>*"
3. From the SnIP command line copy the package from the local SnIP to the remote one using scp. A typical command might be
   "scp /tmp/update-m7snip_0.6.34-x_powerpc.ipk root@*<remote-snip-hdlc0-ip-addr>*:/tmp/"
   Remember you don't have to type the full package name instead using the tab key after perhaps "/tmp/upda" and the rest should be auto-filled in.
4. From the local SnIP command line you can telnet into the remote SnIP as root and perform operations there.
   "telnet *<remote-snip-hdlc0-ip-addr>*"
   When running over the link there will be noticeable delay in responses due to the satellite round-trip delay.
5. Now logged into the remote SnIP you can install the package that was previously copied into the /tmp directory.
   "ipkg install /tmp/update-m7snip_0.6.34-x_powerpc.ipk"
   Use [Tab] to shorten your typing. There should be a "successfully terminated" message. The installation should not change the current operating parameters, and leave the link operating.
6. To update the remote SnIP's configuration file at /etc/config/system you can call configwiz and issue the save command. Even with no other changes the configuration file will be modified to add the new variable, Cisco_Proto, so it is ready for using different protocols.
7. Be prepared that if you now change the link protocol to hdlc-eth while logged into the remote you will lose communications and be blocked from any further changes. You would then have to create another telnet/ssh session with the local SnIP and modify its link parameters to match the remote, re-establishing communications. The order of any protocol (or modem parameters for that matter) must always be remote first and then the local changed to match.

Everything shown above is fairly straightforward until you get to the part of modifying the remote link protocol, losing communications and having to match it on the local end to re-establish communications. This is not for the faint-at-heart if the remote modem is truly remote and unattended.

MAB
END