

Sierra Radio Systems Station Controller

Advanced Technical Manual



sierraradio.net

INTRODUCTION

This advanced technical manual is intended for people interested in building compatible station control modules, control software or just want a greater understanding about how the system works.

CONTENTS

- The Device Control Network (DCN)
- Connecting to the Raspberry Pi with ssh (Secure Shell)
- Station Controller Menu Utility
- Monitoring the Device Control Network (DCN) from your PC



DCN - Device Control Network

INTRODUCTION

The Sierra Radio Systems Device Control Network (DCN) provides a way to allow multiple devices to communicate on a wired or wireless network. The primary application for the DCN is to allow one or more master nodes, typically a conventional computer, embedded linux SBC like a Raspberry Pi or a dedicated hardware control device, to control and monitor a network of real-time control devices. The DCN specification defines three components or layers, of the network definition.

Layer 1 – The physical and electrical connections. This layer defines the physical standard mechanical connectors, pin assignments, signaling voltages, and RF frequencies

Layer 2 – The packet format, speed and timing. This layer defines the format of data packets sent on the network including packet addressing, error detection, and retry mechanism.

Layer 3 – The payload or application data format. This layer defines the format of the payload being sent from point A to point B for various predefined device types such as remote relays, volt meters, temperature probes, etc.

The payload format is defined but the format allows for users to define their own devices and payload structure.

LAYER 1 - PHYSICAL CONNECTIONS AND ELECTRICAL SIGNALING

The DCN is a “dual-band” system meaning that data may be transmitted over wired or RF communications paths or both.

Physical Connections

The control protocol can be transmitted over any type of communications medium. The most common connections are wired through an RS485 connection, a wireless connection, typically using an RF mesh data network or over ethernet.

Wired RS-485

The RS485 wired implementation can be a simple pair of wires or commonly available Ethernet CAT5 cable and RJ-45 connectors. While the DCN protocol has nothing at all to do with Ethernet except that we take advantage of the wide availability of premade cables with 8 wires. The CAT5 cable provides 8 wires which carry the network traffic and power. Devices may support the simple twisted pair or the

RJ45 connector or both. This approach makes it easy to add more devices to the network without the need for any kind of hub or switch.

RJ-45 Pin Assignments

1	Network data signal A	5	+12 VDC
2	Network data signal B	6	Reserved
3	Reserved	7	GROUND
4	+12 VDC	8	GROUND

For the RS-485 wired connections, this signaling technique is a half-duplex, differential pair that allows multiple devices to be connected to a single pair of wires forming a “multi-drop network”. RS-485 also has the advantage of allowing devices to be spread over 1000’s of feet of cable without the need for signal conditioning or repeaters.

Power can be supplied by any device and delivered to all devices on the network. Only 1 device is allowed to supply power to the local network at a time. Network voltage should be between 12-15 VDC. This provides enough headroom to power any DCN compatible device. All DCN devices have their own built in switching voltage regulators to bring the operating voltages down to 3.3, 5 or whatever voltage that devices needs to operate.

Wireless RF Data Network

The DCN can also use commonly available RF modules that typically operate on 2.4 GHz and 868/900 MHz ISM bands. Several data radios are compatible with the DCN standard. These include various LoRa, Xbee, packet and other data radios. Any data radio that can provide a 9600 baud serial port and conversion to RS-485 can be used. Serial to ethernet adapters can be used to extend the DCN all over the world without the need to add computer systems.

If mesh network radio modules are used, if you add new nodes to the RF network, they automatically become part of the network. This is particularly convenient when extending the range between devices. Each node can be thought of as a serial port that taps into an invisible network of other devices. When data is sent into the serial port of the data radio, the packet will be delivered to every data radio in the network and the packet will be transmitted out of the RF module’s serial port into the local device’s CPU.

A network can consist of a mixture of wired RS-485 and RF data network enabled nodes.

LAYER 2 - DATA RATES AND PACKET FORMAT

The DCN data protocol sends ASCII data at 9600 baud, non-inverted, 8 bits, no parity.

The protocol defines the format of packets of data transmitted on the network. The simple way to think of a packet is a string of ASCII characters that contains the payload to be transmitted from point A to point B and the additional characters necessary to provide synchronization, packet type identification, addressing, and error checking.

A typical packet looks like this...

```
/0001:RY3,1:XX <13>
```

Start of Packet	From Address	To Address	:	Payload	:	CRC or LREC Value	End of Packet
-----------------	--------------	------------	---	---------	---	-------------------	---------------

Start of Packet (/)

A forward slash character / is reserved for the start of packet indication. When a slave device sees the slash, it knows there is a new packet.

From Address / To Address (00 01)

The next two characters (00) are the “from address”. This is typically 00 as the master address but it can be any legal address.

The next two characters (01) are the “to address”. This is the destination of the packet and can be any legal address.

All devices on the network must have an assigned address. Two devices should not have the same address to avoid collisions on the network.

Special Broadcast Packet Addressing

You can direct packets to all devices on the network with a broadcast Direct Packet type //

(Example: //reset)

This addressing mode is mostly used for testing devices where you want to send commands to a device without knowing it’s address. If you send a command to multiple devices with the // start of packet, all devices on the network will respond and this will cause a big conflict.

The most useful application is to send a reset to all devices at the same time or to configure a device over the USB connection when this is the only device connected to the configuring PC.

Device Addresses

You can assign any node an address using any printable character. However, for maximum functionality, we recommend using numbers as the addresses.

Pre-assigned default device addresses

00 System master

01-15 Devices 1-15

Addresses 01-15 are the most common because many devices have dip switches used to set their address. Addresses can also be set over their serial ports in software which will override the hardware address settings. With software addressing, a device can have an address from 16-99. While a device

address can include alphabetic characters as well, we don't recommend using them. The main reason is because there is a master command that will cause all devices to announce their presence and the time of response is based on their address. The higher the address the longer it takes to respond. This mode is only used when the master wants to poll for new, unregistered nodes in the network.

Delimiter

The colon character : is used as a delimiter to mark the beginning and end of the payload. As such the packet can not contain any additional colons.

Payload

The payload is application dependent. See standard command summary.

Error Check Value

The error check value is either an 8 bit LRC or 16 bit CRC. The checking code is applied to all characters in the packet except the initial start of packet character /. Of course the LRC is not applied to the LRC characters either.

Most devices, as of this writing, do not implement the checking code but use a dummy value of "XX" as a place holder. Future revisions of firmware will support the CRC / LRC modes.

End of Packet

The end of packet character is a carriage return, ASCII byte value 013 (decimal). When an end of packet character is encountered, the input buffer is evaluated.

The evaluation process identifies a packet by finding the start of packet synchronizing character "/", addresses, delimiters, payload and LRC/CRC.

If the packet format is correct and the "to address" matches the devices address the payload is extracted and put into the command parser.

If at any point the packet format is not correct or the address does not match, the buffer is flushed.

If the packet is good, the payload will be further decoded and executed.

PAYLOAD FORMAT

The payload may contain any printable characters (0-9, A-Z , a-z, and punctuation except / and :)

The payload may contain from zero to 20 fields delimited by commas. The comma delimiter must be placed between fields and not at the beginning of the payload.

The payload field assignment is a command field followed by zero to 20 argument fields.

For example:

RY1 , 0

Where the command is "RY1" and argument 1 is "0". In this example the command tells the target device to set relay 1 to a value of 0 (off).

Upon successful execution of a command, the device may send a response packet back to the "from address", typically the master "00". The form is usually an UPDATE packet that indicates the state of the device after the command was executed. The UPDATE command may be suppressed in a device if it is configured to be quiet but devices typically will provide the update to the master.

STANDARD COMMANDS

Every device may support a different set of commands. Refer to the device's reference manual for specific commands supported. These examples are presented to provide examples of typical commands.

Common System Commands

PING	Pings device
*ROLLCALL	Every device will announce it's presence with a delay calculated by its address
REBOOT	Restart the device
*REGISTER	Tell the device the master knows it is there
*RELEASE	Tell the device it is no longer registered to the master
SETADDR,5	Set device address to 5.
*ECHO,BLA	Send the string BLA back to the master
HELP	Display help information

* These commands implemented in the commercial version of the modules.

STANDARD COMMANDS – ALL DEVICES

Administrative Commands

Command SETADDR,<string>
Example //SETADDR, 5
Definition Sets the network address for the device to <string>, where <string> is a single, printable ASCII character.

Command REBOOT
Example //REBOOT
Definition This is a soft reboot command will re-start the device.

Command PING
Example //PING
Definition This will return the name, address and type of connected device.

Command ROLLCALL
Example //ROLLCALL * Only implemented in the commercial firmware
Definition This will return the name, address and type of connected device from all devices on the network. Each device will wait for a short period of time before sending its response. The delay time is calculated based on the device address. For example an address of 5 will wait $5 * 100 \text{ ms} = 500 \text{ ms}$. This will minimize or avoid collisions.

Command STATUS
Example //STATUS
Definition This will return useful status information about the connected device

GPIO Module Specific Commands

Relay commands

RY3 , 1 Set relay 3 to be on (options: 1=on, 0=off)

RY , 01000000 Turns off all relays except relay 2 which is on

RY Turns all relays off

General commands

STATE Causes the GPIO module to measure all inputs and return an UPDATE packet with all current measurements and output states.

UPDATE return packet format

/0100:UPDATE,GPIO1,01000000,0110,12.000,13.700,13.850,4.950,72.00,74.00:XX

Where...

GPIO Device type which tells the master the format of the following arguments

UPDATE This packet is an update packet showing the status of all sensors and relays

GPIO1 Module type – This tells the main control software how to parse the rest of the payload

01000000 State of relays 1..8

0110 State of digital inputs 1..4

12.000 Value of volt meter 1 (referenced to common ground)

13.700 Value of volt meter 2 (referenced to common ground)

13.850 Value of volt meter 3 (referenced to common ground)

4.950 Value of volt meter 4 (referenced to common ground)

72.00 Temperature probe 1 measurement in degrees F

74.00 Temperature probe 2 measurement in degrees F

- Field not currently used

Connecting to the Raspberry Pi with ssh (Secure Shell)

If you need to log into your Raspberry Pi to install software, get an IP address or do any sort of maintenance, you can easily log into the system using ssh.

You will need an ssh client like Putty. Putty is a free ssh program for Windows.

You can download Putty here: putty.org

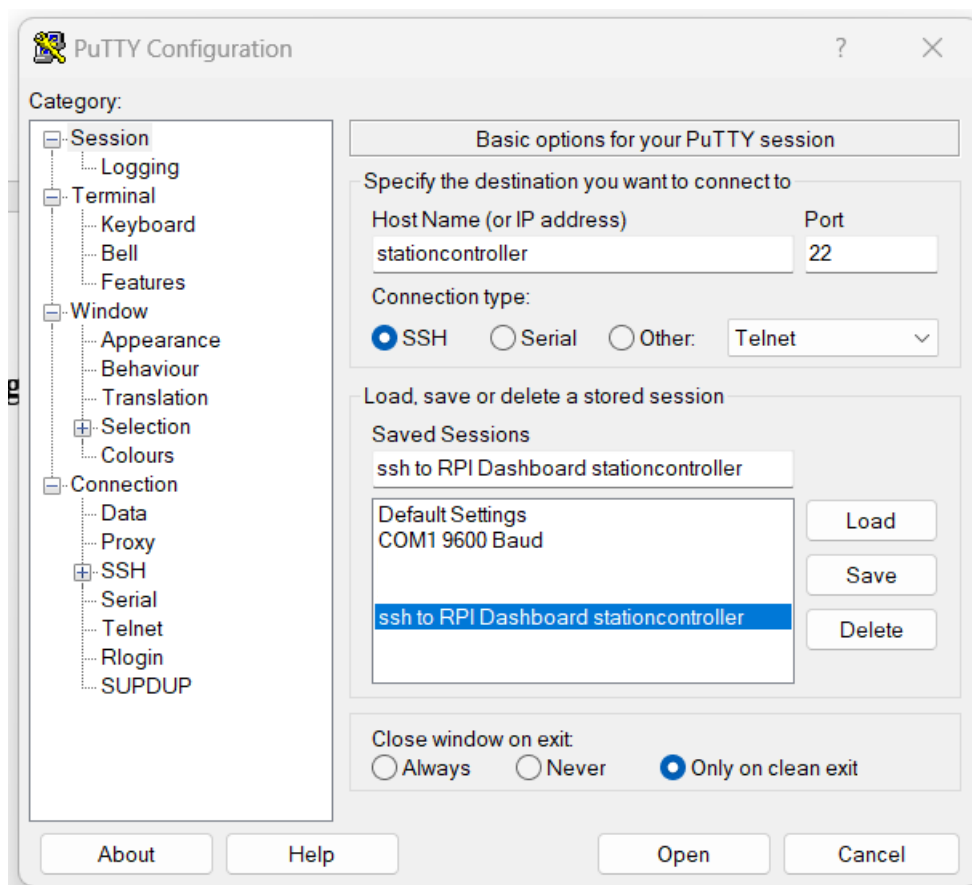
When you launch putty you will create a new connection specifying the following:

Host Name: **stationcontroller**

Port: **22**

Conn type: **SSH**

Then click “Open”



When the connection is made for the first time a dialog box asks you to agree and accept the connection.

Login to the Raspberry Pi

The default login for a SRS RPI image is:

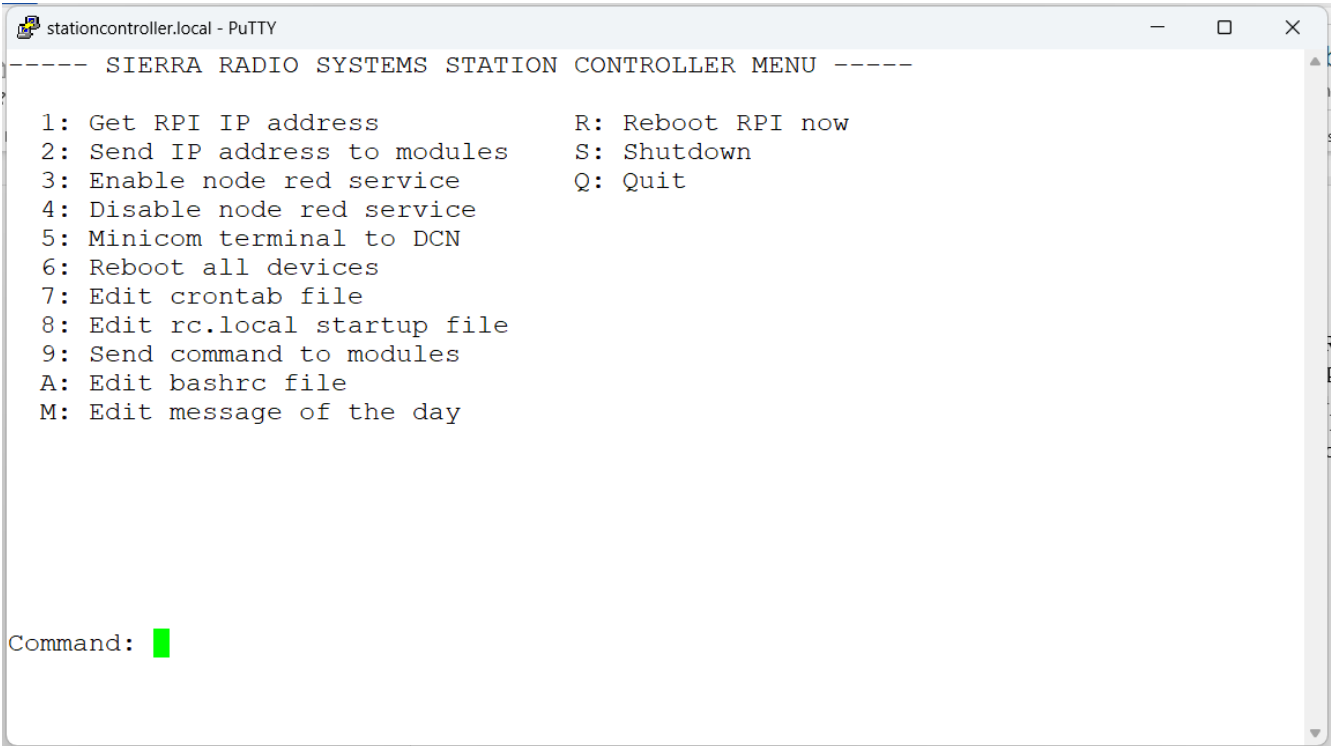
Login **sierra**

Password **61466146** (You old timers will recognize those numbers)

Station Controller Menu Utility

Once you are logged into the Raspberry Pi through ssh or in a terminal window on the RPI desktop you can perform any standard linux / Raspberry Pi OS command. To make things easier, especially for new or casual linux users, the standard SRS SD card image includes a simple menu program written in Python3. To launch the menu program, simply type the word “menu” on the command line from anywhere in the file system. There is also an alias that lets you just type in “m” to launch menu.

The menu program looks like this:



```
stationcontroller.local - PuTTY
----- SIERRA RADIO SYSTEMS STATION CONTROLLER MENU -----
1: Get RPI IP address           R: Reboot RPI now
2: Send IP address to modules   S: Shutdown
3: Enable node red service      Q: Quit
4: Disable node red service
5: Minicom terminal to DCN
6: Reboot all devices
7: Edit crontab file
8: Edit rc.local startup file
9: Send command to modules
A: Edit bashrc file
M: Edit message of the day

Command: █
```

Just type the number or letter of the item you want to execute and hit Enter ! That’s it.

Get RIP IP address

Returns the IP address in text on the screen.

Send IP address to modules

This command will send the IP address to be displayed on the control module LCD displays.

Enable node red service / Disable node red service

By the default, node red starts up automatically when you boot the Raspberry Pi. These commands let you turn that service on or off. The state you set will remain in place until you change it. Even after rebooting.

Minicom terminal to DCN

The minicom program is a great dumb terminal program. Minicom is configured on the SRS SD card to point to the serial port that talks over the Device Control Network to the control modules. This is a simple way to send commands to the modules manually.

Reboot all devices

This command sends the //reboot command to all control modules on the network.

Edit crontab file

In linux, the crontab file defines scheduled events. Cron is generally not used directly in the station controller installation but you may want to set events to meet your needs.

Edit rc.local file

The rc.local file contains the commands executed at boot time. There are some additions in the standard SRS SD card image to launch background tasks and configure the user interface.

Send command to modules

Let's you send commands manually to the device control network. (Simpler than minicom)

Edit bashrc file

The bashrc file contains user interface customizations. In the SRS SD card image we shorten the linux command line prompt to be just the current directory and a \$. The default is normally the computer name and other information. This takes up a lot of screen space so we shortened it.

Edit message of the day

When you log into the Raspberry Pi, the message of the day file is displayed. You can customize it.

Reboot RPI now

Reboots the RPI... now... of course.

Shutdown

Well... shuts down the RPI... now.

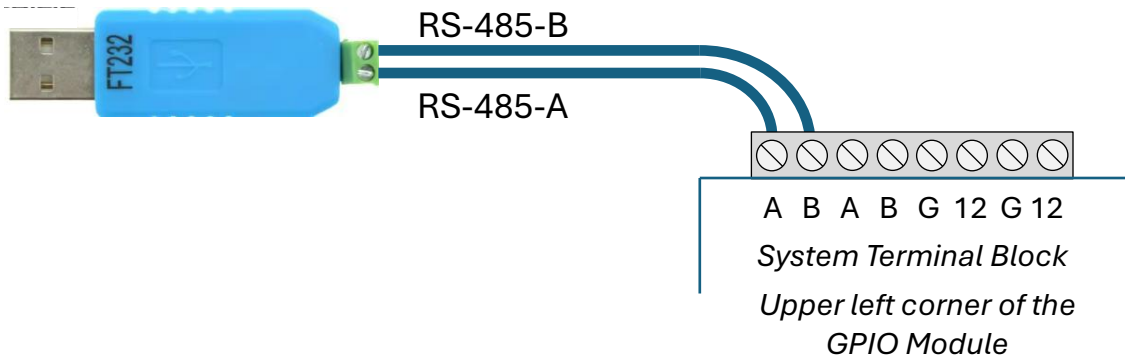
Quit

Exit to the linux command line.

You can customize the menu program by changing the Python3 code. You can find the menu program at /home/sierra/bin/menu

Monitoring the Device Control Network (DCN) from your PC

One of the advantages of the SRS Device Control Network (DCN) is the ability to connect several devices on the network at the same time. A simple way to monitor and even control the network is to use a USB to RS-485 dongle with the pair of wires connected to one of the spare A/B terminal pins on one of the control modules.



Configure a dumb terminal program like Putty to use a serial connection set to 9600 baud. Here is an example of monitoring the DCN for traffic. You can see the Raspberry Pi dashboard polling each of the three control devices connected.

```
COM5 - PuTTY
/0200:UPDATE,CX1,0001:xx
/0003:state:XX
/0300:UPDATE,WM1,00,0.00,0.00,0.00,0.00:xx
/0001:state:XX
/0100:UPDATE,GPIO1,10010000,0000,0.00,12.50,0.00,0.00,63.72,77.67,-:xx
/0002:state:XX
/0200:UPDATE,CX1,0001:xx
/0003:state:XX
/0300:UPDATE,WM1,00,0.00,0.00,0.00,0.00:xx
/0001:state:XX
/0100:UPDATE,GPIO1,10010000,0000,0.00,12.50,0.00,0.00,63.72,77.67,-:xx
/0002:state:XX
/0200:UPDATE,CX1,0001:xx
/0003:state:XX
/0300:UPDATE,WM1,00,0.00,0.00,0.00,0.00:xx
/0001:state:XX
/0100:UPDATE,GPIO1,10010000,0000,0.00,12.50,0.00,0.00,63.72,77.67,-:xx
/0002:state:XX
/0200:UPDATE,CX1,0001:xx
/0003:state:XX
/0300:UPDATE,WM1,00,0.00,0.00,0.00,0.00:xx
/0001:state:XX
/0100:UPDATE,GPIO1,10010000,0000,0.00,12.50,0.00,0.00,63.72,77.67,-:xx
```

Breaking this down:

Line 4: Raspberry Pi, device 00, sending a “state” command to device 01, the GPIO module.

Line 5: GPIO module device 01, responding with an UPDATE packet showing the state of all relays, digital inputs, volt meters and temperature sensors.

Line 6: Raspberry Pi, device 00, sending a “state” command to device 02, the coax switch module.
Line 7: Coax switch device 02, sending an UPDATE packet back to the Raspberry Pi with the state of the 4 coax relays 0001 indicating that port 4 is currently selected.
Etc.

For more details on the format and content of the command packets, check out the control module detailed reference manuals.

You can insert commands with your dumb terminal program. Be aware that the Raspberry Pi dashboard will continue to poll the devices. To avoid collisions you should disable the automatic update by clicking on the “Automatic Update” button on the dashboard. This is a toggle that turns on/off the automated polling.

When you are using a dumb terminal program like Putty, be aware that every keystroke goes out to the network. The backspace key generally does not do anything. If you make an error, just hit Enter and enter the command again. If you want to test this on the GPIO module, lets turn on/off relay 1.
Enter this:

```
/0001:RY1,1:XX    Turns on relay 1  
/0001:RY1,0:XX    Turns off relay 1
```

The syntax is:

```
/      Start of packet  
00    From device 00 (Usually the Raspberry PI)  
01    To device 01 (the GPIO module)  
RY1,1 Turn on relay 1  
XX    Placeholder for CRC checking
```

Another good command to try is the “state” command. This tells the device to return the state of the device.

```
/0001:STATE:XX
```

This will return the state of all sensors and relays. See the GPIO Module Detailed Reference Manual for all the command and data field information.

