# SPROG DCC Decoder Programmer

Specification, User Guide and Assembly Instructions
Version 3d, December 2003

© 2003 Andrew Crosland
web: http://www.sheerstock.fsnet.co.uk/dcc
e-mail: dcc@sheerstock.fsnet.co.uk

# Table of Contents

# 1 Introduction And History

SPROG is a DCC decoder programmer for connection to a USB or serial port of a personal computer or similar device. It was inspired by the MERG standalone programmer.

The original requirement for SPROG was that it be easy to use and not require any specific hardware or software, at least in basic operation. To this end, the human interface to SPROG uses simple ASCII text messages.

The advent of the DecoderPro package (http://jmri.sourceforge.net/) circumvents this requirement to a large extent as, by use of the java programming language, it allows platform independent support of a wide range of DCC hardware. Also, it is shareware available as a free download.

All design data is freely available on the web at http://www.sheerstock.fsnet.co.uk/dcc/sprog. PCBs, programmed PICs, the USB interface chip and complete kits are also available, see the web site for details. The PCB is single sided and uses surface mount components.

If you do not feel able to construct a SPROG yourself then contact me and I may be able to help.

## 1.1 Features

SPROG takes input form a standard USB or RS232 serial connection (from the 'host') and provides a formatted DCC bit stream with pre-amble, start bits and error byte added. It is intended for computer based programming of DCC decoders and includes a low power DCC booster for that purpose, limited to 1A by the on-board power supply. More generally, SPROG can generate any arbitrary DCC packet from an ASCII input, inserting the correct preamble, start, stop and error check bits and can be used as a computer controlled booster for small layouts.

The choice of USB or serial interface is made when SPROG is built. It is not possible, at present, to support both interfaces in one unit.

SPROG requires an external AC power supply capable of delivering the output current required from SPROG at 12-16V AC.

The recommended way to use SPROG for decoder programming is to use DecoderPro, see above. Otherwise, communication with SPROG is via a simple "command line" interface. The USB version uses virtual COM port (VCP) drivers so that it appears to be attached to the computer via a COM port.

SPROG may also be used as a command station/booster for rolling road decoder testing. After specifying a decoder address, forward and reverse speed step packets are easily generated. A more general packet output command is also provided (These features are not yet available through DecoderPro).

Future firmware releases will include:

- The ability to vary the DCC bit timing for decoder testing and
- Sensor inputs for the DecoderPro speedometer function to allow speed measurement of a programmed loco giving information useful in adjusting the locos speed table

## 1.2 Connector Pinouts

SPROG connector and switch positions are shown in Figure 1 as viewed with the connectors towards the observer (surface mount components away from the observer).

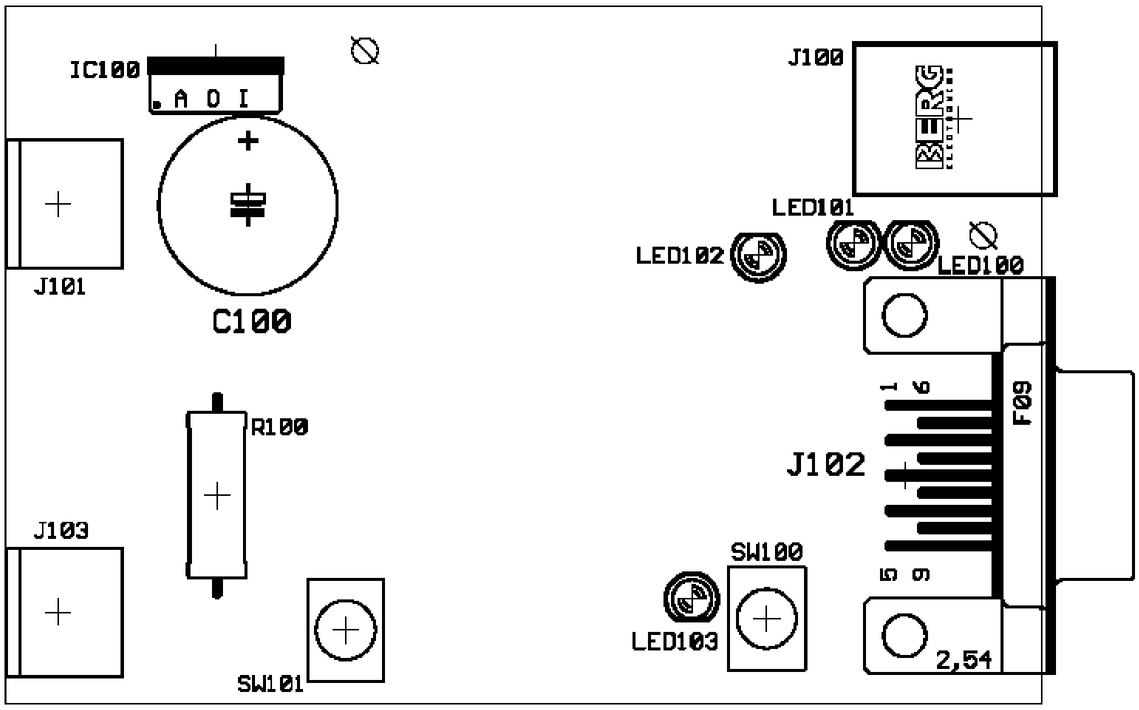

**Figure 1 Connector and switch Locations**

### 1.2.1 J101 Power Input

SPROG requires an AC input in the range of 12-16V. This rectified on board. The smoothing capacitor is rated at 25V.

An on-board adjustable regulator allows the DCC output voltage to be adjusted. A further regulator generates the 5V required by SPROG components.

A DC power source may also be used, in which case the allowable range is extended to 25V DC. Input polarity is not important. The voltage available to the programming track will be reduced by the voltage drop across the bride rectifier.

The minimum input voltage is dependent upon the required DCC voltage, the drop-out voltage of the adjustable regulator and the voltage drop in the rectifier. As a rule-of-thumb, a DC power supply should be (DCC voltage required + 5V). An AC power supply should be (DCC voltage required + 5V)/1.4.

During programming the rack current is limited to 250mA after an initial delay of 100ms. A 500mA supply should be more than adequate. SPROG has successfully been used with a “wall wart” rated at 12V/300mA.

### 1.2.2 J103 DCC Output

There is no requirement to observe any particular polarity when connecting SPROG to a programming track.

### 1.2.3 J102 RS-232

SPROG is configured as a DCE (Data Communication Equipment) and has a 9-pin female connector (socket) with the pinout shown in Table 1.

| J3 Pin | Function | Direction |
|---|---|---|
| 1 | - | |
| 2 | RXD – Receive Data | Output |
| 3 | TXD – Transmit Data | Input |
| 4 | | See Note |
| 5 | 0V | - |
| 6 | | See Note |
| 7 | RTS – Request To send | Input |
| 8 | CTS – Clear To Send | Output |
| 9 | - | |

**Table 1 J102 RS-232 Connector Pinout**

Note: Pins 4 and 6 are connected together by SPROG.

For normal operation only Transmit data, Receive Data and ground need be connected in the serial cable. To use the bootloader to download new firmware, RTS and CTS must also be connected in the serial cable and must be enabled in the terminal emulator (hardware flow control). Depending upon the configuration of the computers serial port, a null modem cable may be required which swaps Receive data and transmit data (and RTS and CTS for the bootloader).

### 1.2.4 J100 USB Connector

A B type USB connector is fitted to SPROG for use with an A-to-B cable between SPROG (supplied with SPROG kits) and the host/hub. SPROG is a self-powered USB device and is suitable for use with low power hubs.

### 1.2.5 Switches

SW101 is used to reset SPROG. Reset will occur automatically during power on and should not normally be required during operation.

SW100 is used in conjunction with the reset switch to start the bootloader to accept new firmware via RS-232 or USB. Normally the bootloader would be started using the B command, see 4.4. It should only be necessary to use SW100 to start the bootloader after a failed or corrupted firmware download. Hold SW100 closed whilst resetting SPROG to start the bootloader.

## *1.3 Specification*

Table 2 gives the operating conditions for SPROG.

| Parameter | Minimum | Nominal | Maximum | Units | Note |
|---|---|---|---|---|---|
| AC Input supply voltage | 9V | | 15V | V | 1 |
| DC Input supply voltage | 11V | | 18V | V | 1 |
| Vin supply current – not programming | | 50 | | | |
| Vin supply current – programming | | 300 | | mA | 2 |
| Vin supply current – Rolling road mode | | 1 | | A | 3,4 |
| Operating Temperature Range | | 25 | | °C | |
| Output Load - programming | | | 250 | mA | 2 |
| Output Load – rolling road mode | | 1 | | A | 3,4 |

Notes:
1. Minimum depends upon decoder being programmed, see 1.2.1. Figure given is sufficient to guarantee correct regulation of on-board 5V supply.
2. SPROG will remove track power if output current exceeds 250mA as measured 100mS after applying power. Surge current during decoder power-up may be considerably greater than this.
3. Depends upon required DCC current.
4. Booster mode DCC current limit may be set by user (*** *not implemented yet)*

**Table 2 Operating Conditions**

# 2 Theory of Operation

SPROG uses a PIC microcontroller to translate commands sent over the USB or RS-232 interface into DCC packets to be driven by a power MOSFET output stage configured as a H bridge. The PIC forms an integral part of the booster, generating the correct switching waveforms for the MOSFETs and sensing the output current through it's internal A/D converter.

A regular interrupt sets the DCC timing at 58us for a '1' half-bit and 116us for a '0' half-bit. During each period of interrupt processing the next DCC bit is generated, the A/D converter is read and set to do a new conversion and the UART is polled to check for new data received from the USB or RS-232.

The MOSFETS in the output stage are chosen to have a very low on-resistance and thus dissipate very little power even with high output currents.

The lower half of the output H-bridge uses N-channel MOSFETS with logic level gate threshold, the upper half uses P-channel devices. To avoid problems with the P-channel MOSFET in one half of the bridge not having turned off when the associated N-channel MOSFET turns on (shoot-through), the PIC inserts a small dead period between each transition when neither MOSFET is turned on. The transistors in the output stage convert the logic level outputs of the PIC to the correct levels to drive the P-channel MOSFETs. This results in relatively fast turn-on when the transistor is on but a slow turn off controlled by the 560R collector resistors. This slow turn off increases the likelihood of shoot-through even with the dead period. To get around this the PIC outputs are turned into open-collector outputs by always setting the output low and using the data direction register to switch the output between output (low) and input with external pull-up. This has the effect of slowing down the turn-on of the N-channel MOSFETs.

Output current is sensed by a small value resistor in the ground of the H-bridge. The voltage developed across this resistor is filtered and sampled by the PICs A/D converter to detect programming acknowledge pulses and short circuit conditions.

# 3 Construction

If you have obtained your SPROG ready built then you may skip this section.

It is recommended that you follow the instructions in the order given. The instructions assume the use of the SPROG PCB, either purchased or home made from the layout available from the SPROG web site.

## *3.1 Parts List*

In case of corrections, the latest parts list should be downloaded from the SPROG website.

### 3.1.1 PIC Programming

The PIC U5 must be programmed with the SPROG firmware. PICs supplied with a SPROG kit are pre-programmed. Alternatively, individual pre-programmed PICs may be purchased. A blank PIC may be programmed after it has been soldered to the board by connecting a programmer to the 6-pin header (J1) as shown in Table 3. You should check with you programmer specification as to which connections should be made. If the programmer does not supply +5V then it will be necessary to power up the board during programming.

A completed SPROG may be updated to the latest firmware version using the bootloader feature, see 4.4.

| Pin | Function |
|---|---|
| 1 | PGD (RB7) |
| 2 | PGC (RB6) |
| 3 | PGM (RB3) |
| 4 | 5V |
| 5 | Gnd |
| 6 | MCLRn/Vpp |

**Table 3 PIC programming Header J1**

## *3.2 Assembling SPROG (Initial Stage)*

The term Ux/y means Ux pin y, e.g. U1/3 refers to pin 3 of U1.

Most of the components are fitted on the side of the board with the silkscreen and copper pattern. Components numbered greater than 100 (e.g. SW100) are fitted on the plain side of the board. Please download the top and bottom component placements from the SPROG website.

Those more confident or experienced may decide to assemble SPROG differently according to experience. For the less experienced, a suggested assembly order and intermediate tests are given.

Testing generally involves connecting the chosen power supply to the power input J101.

- **Always double check component placement and orientation before applying power.**
- **Always check for solder bridges or short circuits before applying power.**
- **Always disconnect the power supply before proceeding to the next stage of assembly.**

The surface mount resistors and capacitors supplied in the SPROG kit are small and easily lost – look after them! If you need a few spares then send me two first class stamps with a list of values required. I can also provide other replacement components (or even for your own projects if they're used on SPROG) - prices on request.

**NOTE:** Switches SW100 and SW101 are a tight fit in the PCB. They are easier to fit if the legs are first straightened out.
Terminal blocks J101 and J103 are a tight fit in the PCB. They can be **gently** pushed home with a screwdriver in the screwhead after first opening them to their fullest extent. Alternatively, carefully open out the PCB holes with a 1.2mm drill.

### 3.2.1 Identifying Components

Where there may be confusion between components they will be packaged separately in the kit and identified. E.g. the low value decoupling capacitors which are unmarked. The transistors and diode look very similar, both being SOT-23 packages, but can be identified by the fact that the kit contains two transistors and only one diode.

Surface mount resistors are marked with a numeric version of the familiar resistor colour code, e.g. a resistor marked 152 is 1.5k ohms. If you cannot find the resistor you need try reading them the other way around! E.g. 251 is not a "preferred value", turn it around and it becomes 152, which is.

Tantalum capacitors (C7, C12 and C21) are polarized and **must** be fitted the correct way round as shown on the placement diagrams. They will be identified in the kit. He electrolytic capacitor C100 is also polarized and must be fitted correctly.

### 3.2.2 Soldering Surface Mount Components

The USB interface is the most difficult device to solder having 0.35mm pins on a 0.8mm pitch. It is possible to solder this device with 22 AWG solder (not included in the SPROG kit) and a fine tipped soldering iron, but many people prefer to use solder paste (not included in the SPROG kit).

To solder the 0805 resistors and capacitors apply a small amount of solder to one pad. Hold the device in a pair of tweezers and position whilst reflowing the solder with the iron. Solder the second pad.

For the ICs, apply a small amount of solder to a corner pad of the IC location on the PCB (enough to cover the pad without leaving a large bump). Position the IC correctly and reflow the just applied solder using the soldering iron allowing the IC to seat itself on the PCB. When the joint is set, the IC may be gently twisted to ensure the other legs are aligned with their pads (assuming the first leg just soldered is aligned). Solder the opposite corner leg to secure the IC and then solder the other legs.

### 3.2.3 Power Supplies

1. Fit the power connector (J101), bridge Rectifier (BR1) and smoothing capacitor (C100), referring to the placement diagram for polarity. Connect a suitable power supply and check for appropriate rectified voltage across (C100).
2. Fit adjustable regulator (IC100), feedback resistors and preset (R7, R8, VR1) and decoupling capacitors (C6, C7), paying attention to the polarity of C7. With the preset (VR1) turned fully anti-clockwise the voltage across C7 should be approx 8V. The maximum voltage with VR1 fully clock-wise depends upon the input supply voltage, see 1.2.1.
3. Fit the 5V regulator (U3), series resistors (R29, R21) and decoupling capacitors (C19, C12), paying attention to polarity of C12. Check for 5V across C12.
4. Fit the remaining decoupling capacitors (C4, C21-23), LED102 and R11, paying attention to polarity of C21 and LED102. Check that LED102 light when power is applied.

### 3.2.4 PIC and Related Components

1. Fit R17, R23, R26, R34, R30 - R33, C11, C16, C17, C18, C20, D1, LED103, SW100, SW101 and X2. Check the input sense divider at the junction of R32 and R33. This should be Vin (input

voltage) * 0.175 and always less than 5 Volts. Check that U5/1 and U5/21 are pulled to ground when SW101 and SW100, respectively, are closed.
2. Fit the PIC U5.

### 3.2.5 RS-233 Interface

Skip this section if you are building a USB SPROG.

Fit R12, R22, R24, R25, C9, C10, C13 – C15, J102 and U4. The RS-232 interface generates it's own positive and negative supply voltages. Check that U4/2 and U4/6 are approximately +9V and -9V respectively.

### 3.2.6 USB Interface

Skip this section if you are building a serial SPROG.

Fit R1 – R6, R9, R10, R13 – R16, R18 – R20, C1 - C5, C8, J100, LED100, LED101, X1, and U1. Check the 3.3V output at U1/6.

## *3.3 Initial Testing*

### 3.3.1 Serial SPROG Initial Testing

Skip this section if you are building a USB SPROG.

For initial testing of a serial SPROG you need a terminal emulator package such as Windows HyperTerm. Connect SPROG to a spare COM port on your host computer and start the terminal emulator which should be configured to use that COM port as follows:

- No Parity
- 1 Stop bit
- Hardware handshake

Power up SPROG and you should see the reset message and prompt similar to:

```
SPROG Ver 3.0
P>
```

Type ?<return>. The characters you type will not be echoed by SPROG (you may configure your terminal emulator for local echo). SPROG should display the reset message once again.
Type '+<return>' (without the quotes) to turn on the DCC output. LED103 should light.
Type '-<return>' (without the quotes) to turn off the DCC output. LED103 should extinguish.

### 3.3.2 USB SPROG Initial Testing

Skip this section if you are building a serial SPROG.

The USB SPROG requires FTDI's Virtual COM Port (VCP) drivers to be installed. These drivers appear to the computer as extra COM ports. To keep installation simple, SPROG kits are not supplied with the USB EEPROM (U2), which would require programming. This means that a kit built SPROG must use the unmodified VCP drivers, which will result in SPROG being enumerated as a generic FTDI USB-serial device.

#### 3.3.2.1 Installing USB Drivers

The latest version of the VCP drivers, appropriate to your operating system, should be downloaded from the drivers page of the FTDI website http://www.ftdichip.com/FTWinDriver.htm. FTDI also have an app note available for download (http://www.ftdichip.com/Documents/AN232-03.PDF) detailing the installation process . This is based upon earlier generation devices but may still be found useful.

For MAC OS, drivers are also available from the FTDI website (http://www.ftdichip.com/FTMacDriver.htm).

Power up SPROG and connect to your computer using the provided USB cable.

When using windows, the computer should detect the new device and prompt you to supply the drivers for it, if they are not already installed in your system. Refer to FTDI AN232-03.

#### 3.3.2.2 USB COM Port Assignment

To check the COM port assigned to the SPROG drivers, right click on "My Computer", select "properties" then the "Device Manager" tab. Open up "Ports(COM & LPT)". SPROG will be listed as "USB Serial Port (COMx)".

The current version of DecoderPro, when run under Windows98, only supports COM1-4. If SPROG has been assigned to a higher numbered COM port then this will need to be changed as follows: Open "Ports (COM & LPT) in "My Computer" (previous paragraph). Double click on the COM port assigned to SPROG. Select the "Port Settings" tab and click "Advanced" from where you may select a different COM port. Be sure to only select a COM port that is not already used on your computer. Click "OK" three times.

## *3.4 Final Assembly*

### 3.4.1 DCC Output Stage

Fit the remaining components: R27 – R28, R35 - R38, R100, C17, Q1, Q2, U6, U7 and J103.

## *3.5 Final Testing*

Your SPROG is ready for final testing, which requires a short length of track and a known to be good decoder equipped locomotive (or a bare decoder with motor or load resistor attached to the motor outputs) with a known value in CV1.

***Warning***: At this stage there is still a risk that an incorrectly assembled unit will cause damage to either the power supply, SPROG or the DCC decoder and loco used to test SPROG. Carefully check that all components are fitted correctly and that there are no short circuits on the PCB. You are referred to the disclaimer at the front of this document.

Connect SPROG to the host computer as for initial testing and to the power supply. Connect SPROG's DCC output to the track or decoder. Place a decoder equipped locomotive on the track.

Power up SPROG and press the reset button. You should see the reset message and prompt similar to:

```
SPROG Ver 3.xx
P>
```

If the decoder supports direct mode programming then type 'C 1<return>' to read the value of CV1, otherwise use 'V 1<return>' for paged mode. After a short delay (longer with paged mode) you should see

```
    > =hnn
```

in the terminal window where nn is the value read from CV1 (the decoder's short address). You may hear clicks from the locomotive motor and/or see the locomotive creep as the decoder sends acknowledge pulse(s). If nn is indeed the value previously programmed into CV1 then your SPROG is now functional.

## *3.6 Troubleshooting*

*To Be Written.*

# 4 Decoder Programming With SPROG

If you have purchased a built and tested USB SPROG please refer to the USB driver installation in section 3.3.2.1

## 4.1 Reset

*To Be Written.*

## 4.2 Basic Operation With Terminal Emulator

As noted in the previous sections on testing, SPROG may be operated with a terminal emulator program such as windows HyperTerm, see 3.3.1.

The preferred method of operation is to use DecoderPro, see 4.3.

The following sections detail the commands available when using SPROG with a terminal emulator.

### 4.2.1 Command Set Summary

All commands must be entered on a single line terminated by carriage return. Maximum input line length is 32 characters, including carriage return**.** Format of parameters is dependant upon the command. The maximum number of parameters on any line is 8.

*Commands in italics are either unimplemented or implemented but not finalized. The format of input parameters or output may change.*

**General Commands**

- **M [n]** – Display [Set] operating mode
- **R** – Read mode from EEPROM
- ***S*** *– Display Status*
- **W** – Write mode to EEPROM
- **?** - Display Help
- **ESC** - immediately shutdown power to track

**Programmer Commands**

- **C CV [Val]** - Read [program] CV using direct bit mode
- **V CV [Val]** - Read [program] CV using paged mode

**Rolling Road Tester Commands**

- **A [n]** – Display [Set] Address
- ***F n*** *- Toggle Function output (not yet implemented)*
- **O byte [byte] [byte] [byte]** - Output bytes as DCC packet.
- **+** - Track power on
- **-** - Track power off
- **<[step | <]** - Reverse speed step[s]
- **>[step | >]** - Forward speed step[s]

**Bootloader Command**

- **B a b c** – Start Bootloader

**Input Format**

Input values are always parsed as decimal, unless overridden with 'b' or 'h' prefix for binary or hexadecimal, respectively. E.g. h15 is equivalent to 21 decimal.

**Acknowledgement Messages**

CV addresses will always be given in decimal. Other numeric values are given in hexadecimal, decimal and binary.

| Message | Meaning |
|---|---|
| !O | Overload |
| !E | Error |
| No-ack | No acknowledge pulse received during programming |
| OK | Programming operation completed |

**Table 4 Acknowledgement Messages**

### 4.2.1.1 General Commands

**M - Display operating mode**
**M n - Set operating mode to n**
The mode word, n, is interpreted as a 16 bit binary value with each bit corresponding to a particular feature, as shown in Table 5.

| Bit | Name | Feature |
|---|---|---|
| 0 | spare | Do not use, always set to 0 for future compatibility |
| 1 | ECHO_ON | SPROG echoes all received characters if this bit is set |
| 2 | spare | Do not use, always set to 0 for future compatibility |
| 3 | CALC_ERROR | Set to calculate error byte for O command. If clear then error byte must be supplied on the command line |
| 4 | RR_MODE | Set for rolling road/test mode |
| 5 | spare | Do not use, always set to 0 for future compatibility |
| 6,7 | spare | Do not use, always set to 0 for future compatibility |
| 8 | DIR | Direction for rolling road/test mode and booster mode. Set for reverse |
| 9 | SP14 | Select 14 speed step mode for rolling road/test mode and booster mode. |
| 10 | SP28 | Select 28 speed step mode for rolling road/test mode and booster mode. |
| 11 | SP128 | Select 128 speed step mode for rolling road/test mode and booster mode. |
| 12 | LONG | Use long addresses in rolling road/test mode and booster mode |
| 13-15 | spare | Do not use, always set to 0 for future compatibility |

**Table 5 Mode Word Bits**

*Examples required*
See ‘Reset’ for a description of the reset state of the mode word.

**R – Read Mode from EEPROM**
Read a previously saved operating mode from EEPROM, see M command.

**S – Display Status**
Output SPROG status, including A/D (Analogue-to-digital converter) readings and throttle selector inputs. A/D readings are output as 4-digit hex values in the order:

AN0: Vin sense input
AN1
***

**W – Write Mode to EEPROM**
Write current operating mode to EEPROM, see M command.

**? - Display Help**
Displays the SPROG firmware version.

```
SPROG Ver 3.0
>
```

**ESC - Shutdown**
DCC output is turned off immediately.

#### 4.2.1.2 Programmer Commands

**C CV** - Read a CV using direct bit mode
**C CV Val** - Program a CV using direct bit mode
**V CV** - Read a CV using paged mode
**V CV Val** - Program a CV using paged mode

If no VAL value is given, then this command reads the specified CV and displays it in the form `CV <hexadecimal>`, otherwise writes the value Val to the specified CV.

#### 4.2.1.3 Rolling Road Tester Commands

**A – Display Address**
**A n – Set Address**
Display or set the decoder address (decimal) to be used in speed/direction packets. If a new address is set then current speed step will be reset to zero. This command does not perform any programming of decoder CVs.

***I – Display output current limit***
***I n – Set output current limit***
*Display or set the booster stage output current limit for rolling road/test mode and booster mode.*
*Not implemented yet.*

***F n - Toggle Function output***
*Send DCC packet to toggle the state of function output Fn.*
*Not implemented yet.*

**O byte [byte] [byte] [byte] - Output bytes as DCC packet.**
Any arbitrary DCC packet may be generated using this command. SPROG will add the correct pre-amble bits, start bits, and error byte. Note that all address and data bytes and, optionally, the error byte must be given on the command line, this command does not use the address set by the 'A' command. If the mode word CALC_ERROR bit is set then SPROG will calculate the correct error byte which must not be given on the command line. If CALC_ERROR is not set then the error byte must be given on the command line, allowing erroneous packets to be generated for decoder testing.

**+ - Track power on**
Turn on track power and check for overload condition after 100ms. Twenty-four reset packets will be transmitted. At all other times when there is no DCC data being transmitted, DCC pre-amble will be transmitted.

**- - Track power off**
Turn off track power.

**<<[<] - Reduce/Reverse speed step[s]**
**>>[>] - Increase/Forward speed step[s]**
Adjust speed step relative to current speed. If decoder is running in reverse then Reduce/Reverse will increase the reverse speed and Increase/Forward will decrease the reverse speed. If decoder is running forward then Reduce/Reverse will decrease the forward speed and Increase/Forward will increase the forward speed. Increment or decrement is determined by the number of '<' or '>' characters in the command. Speed will not increment/decrement past maximum forward or reverse speed step nor through zero. The current speed step will be reported after performing this command:

**< – display Reverse speed step**
**< step – Set Reverse speed step**
**> display Forward speed step**
**> step – Set Forward speed step**
Set forward or reverse speed step directly.

### 4.2.1.4 Bootloader Command

**B a b c – Start Bootloader**
Exactly three arguments, a, b, c, must be given with the B command but their values are not checked. This helps prevent inadvertent issuing of the b command. The B commands enters the bootloader ready to receive updated SPROG firmware. See 4.4 for full details of how and when to use the bootloader.

## *4.3 Operation With DecoderPro Software*

DecoderPro is a program written in the Java programming language, allowing it to be used on most popular operating systems. It supports a wide range of DCC hardware and Version 1.1.6 and above includes support for SPROG.

To use SPROG with DecoderPro you must first ensure that character echoing is disabled by clearing bit 1 of the mode word (see description of M command) using a terminal emulator to connect to SPROG. Then issue a W command to save the mode word in EEPROM. The default state of the programmed PICs supplied in SPROG kits is correct for operation with DecoderPro.

You must have the Java runtime environment installed on your computer to use DecoderPro and it must be installed before DecoderPro is installed.

- Connect SPROG and power up
- Start DecoderPro
- The first time you run DecoderPro the preferences window should open. If not it can be found under the DecoderPro Edit menu
- Select SPRG as the layout connection and specify the COM port to which it is connected
- Save your preferences, exit and restart DecoderPro
- Click on “Use programming Track”
- Select the decoder type to be programmed and click “open programmer”
- Select the programming mode either paged or bit direct depending on the decoder being programmed

You are now ready to edit CVs, speed tables, function mapping, etc., for more information see the DecoderPro web pages.

## 4.4 Upgrading SPROG Firmware

Occasionally it will be necessary to release new versions of the SPROG firmware, the program that is programmed into the PIC micro-controller, in order to fix bugs or add new features. SPROG includes a feature known as a bootloader which makes this easy to do without requiring any special programming hardware or removing the PIC. New firmware will be supplied as a text file (a .hex file), available from the SPROG homepage.

SPROG needs to be connected to a terminal emulator program that can transmit the text file, e.g. Windows HyperTerm. Enter 'B 1 1 1<return>' to start the bootloader, you should see the bootloader prompt:

```
L>
```

Use the terminal emulator to transmit the .hex file. As each line of the file is received, SPROG will decode the data and re-program it's internal program memory. It is not possible to re-program the bootloader portion of the memory in this way. When the download of the new firmware is complete, reset the SPROG to resume normal operation. The previous setting of the mode word will be retained after a firmware upgrade.

If the bootloader operation is interrupted or corrupted in some way such that SPROG is left with a corrupt program, it is still possible to start the bootloader by pressing and holding the boot switch, SW100, whilst resetting SPROG.

# 5 Useful Links

SPROG homepage http://www.sheerstock.fsnet.co.uk/dcc/sprog.htm
FTDI http://www.ftdichip.com
Microchip http://www.microchip.com
Model Electronics Railway Group (MERG), http://www.merg.org.uk.
Java Model railroad Interface (for DecoderPro) http://jmri.sourceforge.net/
Sun Microsystems (for Java) http://java.sun.com