MERG DCC8 Command Station. Description and Operation

(Refer to schematics DCCSTN8 Rev D for base station and DCC8HS Rev C for the handsets)

This description is for the upgraded system which incorporates consisting.

Specification:

Comprises a base station and upto 8 handsets.

The handsets plug into the base station with a 5 wire connection. (6 way RJ12 connectors may be used but the system is NOT compatible with commercial systems using the same connectors e.g. Digitrax)

The base station produces a DCC signal at 5v logic level. A separate booster is needed. Also the base station needs a 5v DC sypply at about 20mA.

Each handset controls one loco.

Speed is set by a single turn potentiometer with a knob.

Loco selection is with two thumbwheel switches - 0 to 9 decimal so address range is 00 to 99. Extended addressing is not supported.

Forward / reverse is with a conventional (on/off) switch.

Functions F0 (FL), F1, F2 and F3 are supported. Activation is with pushbuttons, one for each function.

A separate pushbutton is used for consist setup.

There is an on/off switch to enable/disable the handset.

There is a choice of speed modes. 14 step, 28 step or 128 step. Each handset can be in a different mode.

The handsets have been built in small diecast boxes (100mm x 50mm x 25mm)

The system allows for up to four consists, each of up to four locos. The consist information is stored in the base station and is retained after switching off.

Consisting is achieved by sending speed and direction information to all locos in a consist using their base address. Advanced consisting is not used so it will work with basic decoders.

Function information is sent to any loco in a consist using its base address.

The base station uses a PIC microcontroller (16F84-10/P). The handsets are intended to be low cost and use standard logic.

Operation

When first switched on, all handsets are disabled unless the speed control is at zero. This prevents any locos starting unexpectedly.

To select a loco, dial up the address on the thumbwheels and turn the speed control to zero. The loco will now be controlled by that handset. (handset on/off switch to be in ON position)

To select another loco, dial up the address and turn the speed to zero. The first loco will stop and the second will be selected. The original loco remains under control until the speed is zero even though the thumbwheel address is different so you can ‘pre-select’ the next loco.

Reversing is accomplished with the forward / reverse switch. This is not a pushbutton so the switch position indicates direction.

There are four function pushbuttons. These have a toggle action (in software) so push to activate - push again to deactivate.

Any handset can be ‘switched off’ with its on / off switch. This will stop any loco selected and completely deactivate that handset. Unplugging the handset has the same effect.

If you attempt to select a loco that is in use on another handset, an alarm will sound and that loco will not be selected.

Consisting.

Addresses 90 to 93 are reserved for consists.

To put a loco into a consist it first must be ‘active’ on a handset. Hold in the ‘consist’ pushbutton and then press one of the function buttons. F0 will add the loco to consist 90, F1 to 91, F2 to 92 and F3 to 93. The alarm will sound as an acknowledgement if the loco is added and the consist button is still in.

If the consist is full (4 locos) the alarm will not sound and the loco will not be added.

You cannot include a DC (address 00) loco in a consist but 00 may be used to clear locos from a consist.

The loco direction when in the consist depends on the position of the forward / reverse switch when the loco is added. If the switch is forward, the consist direction will be that of the consist handset. If the switch was in reverse, the loco direction in the consist will be opposite to that of the consist handset. (back to back running)

To drive the consist, select the consist address (90 to 93) on any handset as if it were a single loco. (the function buttons are inactive while driving a consist)

Function commands can be sent to any loco in a consist by selecting the loco address on another handset, activating by setting the speed to zero and then pressing a function button. This can be done whilst the consist is running.

As long as the consist is ‘active’, speed information to individual locos is blocked. As soon as the consist is deactivated by selecting another address and zero speed or by switching off that handset, the locos revert to individual control. This makes assembling and breaking up consists very easy.

Even though a consist may not be ‘active’, the locos in that consist are ‘remembered’ so the same consist may be reassembled without having to set the locos into the consist again.

To completely remove a loco from the consist in memory, repeat the action for putting it in. (loco active, consist pushbutton in, press correct function button). If you are unsure which locos are in a consist, you can clear by ‘adding’ address 00. Every time 00 is entered, one loco is removed on a first in, first out basis. To completely clear a consist, add 00 four times.

NOTE: With the present software it is possible to have the same loco in more than one consist. This will cause a conflict if both consists are activated at the same time.

There is no facility for ‘accessory control’ in this system.

Technical description.

The base station (drawing DCCSTN8)

The heart of the base station is a 16F84-10/P PIC microcontroller running at 8 MHz. The assembly source code and HEX code for the PIC are downloadable from the MERG resources page. The oscillator element for the PIC is a low cost ceramic resonator with built-in capacitors.

Each handset connects to tha base station via a 5 wire interface which is unique to this system. The objective was simplicity and low cost rather than interchangeability. The 5 wires comprise a +5v supply, a serial clock from the base station, a serial data line from the handset, an analog speed voltage and a zero volt line.

The analog speed value from each handset is converted to a digital value by the ADC chip (U3) which is a MAX 1112CPP. This has 8 input channels and a serial interface to the PIC. The resolution is 8 bit although only 7 bits are used on the system. (128 speed steps)

The remaining address, function and mode data are clocked out of the handsets by a synchronous serial interface. The clock is produced by the PIC and boosted by U1 to drive the handsets which may have significant cable capacitance. U4 is a digital multiplexer which selects the data stream from one handset at a time.

The resulting NMRA standard DCC bit stream is generated by the PIC and output to the booster via the buffer U1. Pin 11 of the PIC is the alarm output which activates

the piezo alarm and / or LED using transistor Q1 to provide the necessary current. The AWD must operate off DC and suitable devices with operating range of 3 to 15 volts are readily available.

There are two spare in/out lines on the PIC for future expansion.

The PCB layout for the base station has handset connections terminated in 0.1 in. spaced pads suitable for a PCB connector (e.g. Molex KK) or for direct wiring and also pads for vertical entry RJ12 (6 way) connectors. The handset sockets may be mounted separately on a suitable panel. 5 way audio DIN connectors were used on the prototype.

The RJ12 connectors may be mounted directly on the board. This saves on wiring. The connectors are vertical entry RJ12 data connectors widely used on computer networks, telephones etc. and on some commercial DCC systems.

The handsets. (drawing DCC8HS)

As many handsets are required, the objective was to keep the cost low and make the design as ‘user friendly’ as possible. (Handset operation on commercial DCC systems is often complex and not very intuitive)

The speed control is a simple linear potentiometer (10K) giving an output voltage from 0 to 4 volts. (the range of the ADC is 0 to 4.097v)

Address selection uses two BCD thumbwheel switches. Pullup resistors are attached to each switch output and form the inputs for a 8 bit parallel in - serial out shift register (74HC165). The four function pushbuttons, the reversing switch, the handset on/off switch and the mode select jumpers provide the inputs to a second 74HC165, again with pullup resistors.

The serial output from the first shift register is connected to the input of the second to create a system with 16 inputs. An extra bit is achieved by using the serial input of the first register for the consist set pushbutton.

The monostable IC (74HC123) is used to switch the shift register from parallel load mode to serial transfer mode when a clock signal is detected. The first transition of the clock triggers the monostable and the second and subsequent edges transfer the data from the shift register. The monostable is held active while the clock is present.

A PCB layout is available for the handset. This is a double sided board but suitable for DIY as the vias can easily be wired through. (PTH would be better but not necessary). A mixture of surface mount and conventional components are used to keep the size small. The prototypes were built in metal diecast boxes 100mm x 50mm x 25 mm as these were a comfortable size and matched existing non-DCC handsets used on the layout. The function pushbuttons mount on a separate (single sided) small PCB but the thumbwheels are fastened to the main PCB although the connectors need to be ‘wired’ to the PCB individually. (If PCB mounting thumbwheels are to be used, a different board desing will be necessary. I used cheap BCD switches at GBP 2.20 each.)

A cutting mask and label artwork are downloadable for my handsets but as this is a DIY system you can use whatever layout suits.

NOTE: The PCBs and artwork are .PDF files. These can be printed at high quality and exact size on a laser or ink jet printer using the Acrobat reader.

The PIC program (source DCC8D)

The program reads each handset socket in turn and converts the data into a DCC packet. This applies even if there is no handset on a given socket or that handset is inactive. Consequently a loco speed is only updated every 8 packets. If there is no active handset on a socket, an idle packet is sent. Function packets are sent following every 4th speed / idle packet.

The ADC speed value and all 17 bits of data are read from the handset during the first 5 cycles of the packet preamble. (clock is about 200KHz so cable length may be important). The remaining 5 cycles are used to 'parse' the packet information. During the rest of the packet, the serial clock is inactive.

An analog (DC) channel is available on address 00. This uses stretched zeros with a relatively low (approx 40 pps) PWM rate so may not be suitable for low inertia motors but gives good low speed running with ordinary motors. A nonlinear algorithm is used to convert the speed value to the degree of stretch giving a virtually linear relationship between speed value and average DC. The speed resolution in DC mode is reduced to 64 steps.

With the exception of the stretched zeros, all timing is controlled by polling the real-time clock (RTCC) in the PIC. Program loops are not used. This simplified writing the program and ensures that all intervals are within 1.5 microsecs of the correct value. 58 microsecs is used for the 'ones' half bit and 100 microsecs for the 'zeros' half bit. All preambles are 10 cycles.

The program detects the setting of the speed mode jumpers in the handset and sends the appropriate type of packet. For 128 step mode, this is a 4 byte packet. For the 14 step mode, bit 4 of the speed byte is set by function F0 (FL) to maintain compatibility with some 'basic' decoders. For the 28 step mode, bit 4 is the LSB of the speed. Both 14 and 28 step mode packets are 3 bytes long. Function packets are also 3 bytes.

For consisting, the loco addresses are held in a 16 byte stack in RAM during normal operation. The lower 7 bits are the loco address and the top bit indicates the direction when in a consist. Maintaining a stack in RAM allowed fast checking of which locos were in a consist. A 'shadow' of this RAM stack is also held in EEPROM. On switch-on, the EEPROM contents are transferred to the RAM. Any changes to the RAM stack also change the EEPROM. Consequently, consist information is never 'lost'.

For those familiar with PIC assembly language, the program is well commented and should be relatively easy to follow.

I will try to answer any queries and provide more info. if requested.

Mike Bolton 26/11/99. (bolton@fs1.with.man.ac.uk)