1 General

The intended purpose of this document is to provide information to people testing DCC decoders against NMRA Standards to verify that they conform to all applicable NMRA Standards.

1.1 Introduction and Intended Use (Informative)

The NMRA conformance test DCC decoder test system is used to test a locomotive or accessory DCC decoder. The tester verifies that the decoder meets the baseline NMRA DCC standards S-9.1 and S-9.2. This paper describes how to install the conformance test generator, set up the test jig, load and configure the test software, and run the tests.

The SEND.EXE program is supported by 2 hardware platforms, the Sender V2 System and the Sender V3 System. This document details the hardware and software setup of the Sender V2 System. The Sender V3 System hardware and software setup is detailed in TR-3-2015 Sender V3 Getting Started Guide.pdf

Section 5 Test Descriptions and section 6 SEND.EXE Reference detail the SEND.EXE program. These sections apply to both the Sender V2 System and the Sender V3 System.

1.2 References

This standard should be interpreted in the context of the following NMRA Standards, Technical Notes, and Technical Information.

1.2.1 Normative

- S-9 Electrical
- S-9.1 DCC Electrical Standard
- S-9.2 DCC Communication Standard
- S-9.2.1 DCC Extended Packet Formats
- S-9.2.4 DCC Fail-Safe

1.2.2 Informative

- Sender V2 Document Folder
- Sender V3 Document Folder
- TR-3-2015 Sender V3 Getting Started Guide
- Sender V3 Schematic Send104E.pdf

1.3 Terminology

Term	Definition
Glitch Injecting Booster	A booster (power station) specifically designed to create a crossover glitch at the zero crossing of the signal, to mimic situations found in installations with long bus runs.
Noise Injecting Booster	A booster (power station) specifically designed to induce a 100kHz noise on the DCC signal to test the decoder's ability to reject the noise.
SEND.EXE	Software used to send DCC instruction packets to the decoder under test.

Table of Contents

2 Sender V2 System Hardware and Software Requirements

2.1 Hardware and software required to conduct the tests:

An IBM PC compatible computer is needed. The hardware and software have been tested on a 386SX- 16 CPU, a 486-33 and 486DX-66 machine, and a Pentium 90 machine. The sender test board occupies one 8-bit slot.

The SEND.EXE runs under DOS 5, DOS 6, and the DOS box of Windows 95, 98, or ME.

The SEND.EXE program occupies about 150 Kbytes of space. The log and summary files for each test run will occupy about 150 Kbytes of space.

A DCC booster is required to power the decoder. This can be a normal commercial booster or a special variable Voltage, noise injecting booster used to test compliance with S 9.1.

A test jig must be constructed to route signals to and from the decoder. Directions for building this jig are given in section 4.1.3 **Connecting the Decoder** .

3 Sender V3 System Hardware and Software Requirements

3.1 The Sender V3 System setup is detailed in TR-3-2015 Sender V3 Getting Started Guide.

4 Getting Started with the Sender V2 System

4.1 Hardware Installation

The sections below give the details for configuring, installing and connecting the hardware.

4.1.1 Configuring the Sender Board

The NMRA DCC sender board must be configured before installing it in your PC. The sender board does not use any interrupts so you should make sure that no jumpers are installed on jumper pins JP301 through JP306.

The base IO port address for the sender board must be set next. The default port address 0x340 is generally unused. You should use the default unless it conflicts with other hardware in your PC.

SW301 is used to select the IO port addresses for the sender board. Table 1 shows the IO port addresses supported by the sender board:

Table 1: Sender Board IO Port Address

Address	*SW1*	*SW2*	*SW3*	*SW4*	*Default*
240-257	**Off**	**Off**	**On**	**Off**	
260-277	**Off**	**Off**	**On**	**On**	
280-297	**Off**	**On**	**Off**	**Off**	
2A0-2B7	**Off**	**On**	**Off**	**On**	
2C0-2D7	**Off**	**On**	**On**	**Off**	
2E0-2F7	**Off**	**On**	**On**	**On**	
300-317	**On**	**Off**	**Off**	**Off**	
320-337	**On**	**Off**	**Off**	**On**	
340-357	**On**	**Off**	**On**	**Off**	**X**
360-377	**On**	**Off**	**On**	**On**	
380-397	**On**	**On**	**Off**	**Off**	
3A0-3B7	**On**	**On**	**Off**	**On**	
3C0-3D7	**On**	**On**	**On**	**Off**	
3E0-3F7	**On**	**On**	**On**	**On**	

4.1.2 Installing the Sender Board

You should install the sender board in your PC once you have configured it. To do this, follow the instructions supplied with your PC for installing an optional board. The sender board requires one half size 8-bit ISA slot. You should do these things to install the sender board:

1. Turn off the power to the PC
2. Remove the cover. This usually involves removing several screws on the rear of the PC.
3. Locate an empty half size 8-bit ISA slot.
4. Carefully install the sender board in the slot and make sure it is firmly seated in the slot.
5. Reinstall the cover.
6. Turn on power to the PC and make sure it boots normally.

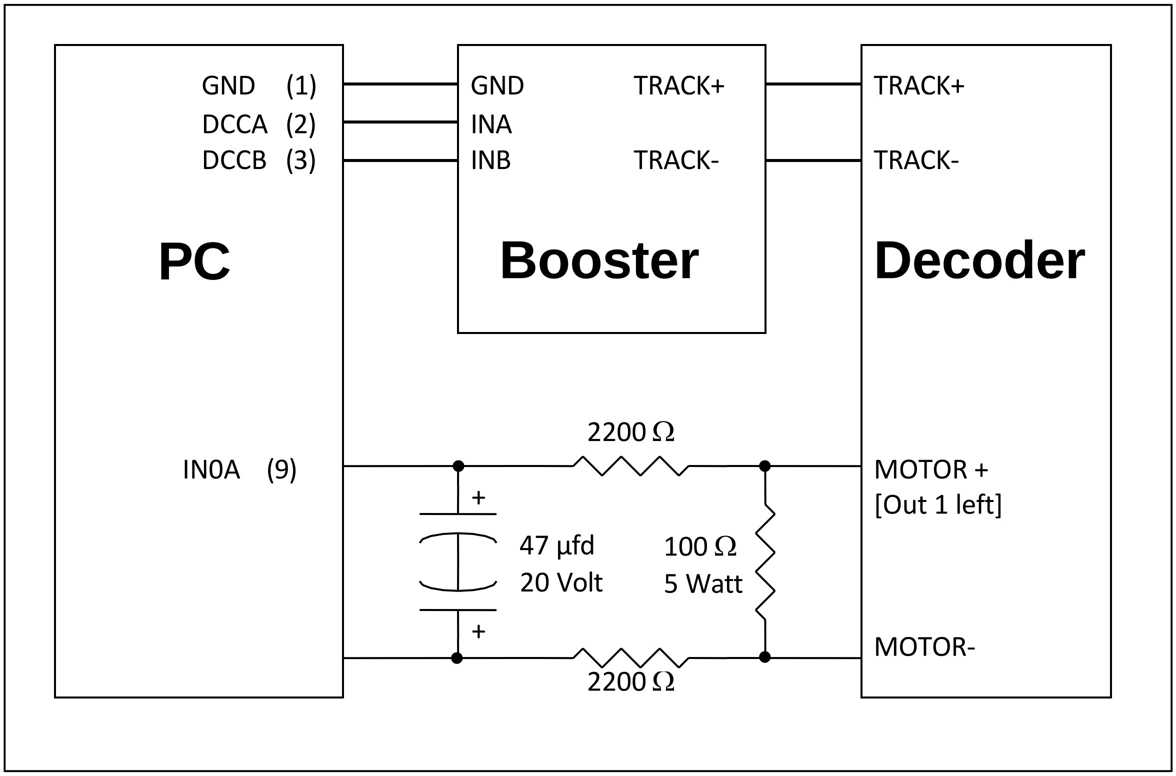
4.1.3 Connecting the Decoder

A test jig must be built to connect the sender board, DCC booster, and test decoder together. Figure 1 shows a schematic of the test jig. This test jig has been tested with several locomotive decoders and an accessory decoder and has worked properly with all of them. The wire lengths for the test jig are not critical so you can build the test jig on a simple perf board. The booster can be a commercial DCC booster, or it can be the special noise injecting booster used to test S-9.1.

The test jig provides a booster to drive the decoder being tested and conditions the output signal from the decoder prior to sending it to the sender board. All the tests are based on sending a preset command to the decoder for one second, sending a trigger command to the decoder, and then noting whether the decoder output has changed state. The sender board provides a balanced pair RS-422 compliant output signal on DCCA and DCCB to drive the booster. The sender board provides an RS-422 compliant input pair on IN0A and IN0B that can be used to test for a zero crossing of the decoder output.

The booster can be a normal DCC compliant booster or can be a special booster that can vary the DCC signal level and inject noise into the signal. Most boosters can be driven by connecting DCCA to one booster input and DCCB to the other booster input. The test jig conditions the output signal from the decoder before returning it to the sender board. For locomotive decoders, the motor output signal is used. For accessory decoders, the first switch control pair is used. The filter circuit shown below filters out any alternating Voltage and then connects it to the first sender board input.

Figure 1: Sender System Components



Locomotive decoders work by presetting the decoder to half speed reverse and then sending an emergency stop signal. The test must be able to tell the difference between half speed reverse and stopped. The test jig shown can reliably determine this change of state for all the locomotive decoders tested to date.

Accessory decoders work by presetting the first switch control output to throw the switch in one direction and then sending a command to reverse the direction of the switch. The filter shown in the test jig also works for accessory decoders if they can be programmed to leave the switch output active for at least one second.

Other types of decoders such as sound decoders can be tested as long as they produce an output signal that will change state in response to either a change in motor direction or to a change in headlamp state.

4.2 Software Installation

Once the hardware is installed, it is necessary to install the software. The DCC decoder test software consists of a single program called SEND.EXE and its associated configuration file SEND.INI. The program also generates log and summary files as a decoder is tested. The simplest installation consists of creating an empty directory on your PC to hold the test software, log, and summary files. Next, copy the SEND.EXE and SEND.INI files to this directory. The program will look for the SEND.INI file in the same directory as SEND.EXE. SEND.INI can also be placed other places if necessary. You should refer to section 6.1 Command Line and SEND.INI Parameters for a complete description of where the SEND.INI configuration file can be placed.

4.3 Running A Test

Once the hardware and software is installed, you can run the self-test on the sender board and then run a test on a decoder. This section will give a quick description of running the self and decoder tests. A complete description of each test is given in section 5 Test Descriptions .

4.3.1 Initial Configuration

The software is set up to use reasonable defaults for the critical parameters. You generally do not need to change these parameters to run tests. This section lists the most important initialization parameters together with their default values. It is easiest to set the parameters that seldom change in the SEND.INI file. It is also possible to change these parameters using command line switches. Table 2 shows the complete list of configuration parameters.

The following parameters are the most important for running tests:

ADDRESS This is the decoder address. The default value is address ‘3’ for locomotive decoders and address ‘2’ for accessory decoders.

TYPE This is type of decoder to test. The possible values are ‘l’ for locomotive decoders and ‘a’ for accessory decoders. The default is to test locomotive decoders.

PORT This is the IO port address used by the sender board. The default port is 0x340.

For locomotive decoders, you should program the decoder for the proper address and 14 speed step baseline operation. The motor control variables should be set to give the largest motor Voltage changes between speed steps possible with the minimum acceleration and braking momentum. The tests assume that the locomotive decoder can go from half speed reverse to stop within one second of receiving an emergency stop command.

For accessory decoders, you should program the decoder for the proper address. The accessory decoder should be programmed to keep the switch output active for at least one second for the tests to work reliably.

4.3.2 Starting the Test Run

You can run a trial test when you have the hardware and software installed and configured. You can start and run a test automatically by simply changing to the directory in which you loaded the software and typing ‘**SEND**,' assuming you have the **SEND.INI** file configured properly.

The program will first ask you a series of questions about the decoder. The following questions will be asked:

Enter base of the log and statistics file name >

This is the base name of the files to store the log and statistical information from the tests. For example, if you typed ‘ACME’ at this point, the program would create a file called ACME.LOG to store the full test log and a file called ACME.SUM to store the summary statistics for the tests.

Enter Manufacturer >

You should enter the decoder manufacturer’s name (e.g., ACME) in response to this question.

Enter Model number >

You should enter the decoder’s model number (e.g., AA22) in response to this question.

Enter Serial number >

You should enter the decoder’s serial number (e.g., 12345) in response to this question.

Enter comments. Begin line with ‘.’ to end input >

You should enter any comments in response to this question. Begin a line with just a ‘.’ to end the comments.

After you have completed these questions, the program will first run a self-test on the hardware and then begin the tests on the decoder. The self-test is done by using the special feedback registers and the software-controlled clock to step the hardware through initialization, signal generation, etc., one clock phase at a time. It next runs the board using the hardware clock to verify that the hardware clock is running properly. You should watch the screen for any self-test errors that might indicate a problem with the hardware or configuration.

The decoder tests are run automatically if all board self-tests pass. You can interrupt the decoder tests by typing the **<ESC>** key on the keyboard followed by the **‘q’** key. This will stop the decoder tests, and you will get the program command line prompt shown below:

```
Command mode (h for help, q to quit) >>
```

Typing ‘q’ at this point will end the program and return you to DOS. A complete explanation of the other commands is given in section 6.3 Manual Commands .

4.3.3 Interpreting the Results

Any self-tests’ errors indicate a problem with the hardware and invalidate the associated decoder test. You should correct any self-test errors before testing a decoder. Assuming the self-tests pass, the decoder tests will run, and the results displayed to the screen as well as being stored in the log and summary files. Section 5 Test Descriptions gives full details on each of the decoder tests.

A perfect decoder test produces a 100% pass on all test cycles. You should see 100% pass or nearly so for all tests. Consistent results of 0% probably indicate a problem with the test jig or decoder itself. You can use the manual commands described in section 6.3 Manual Commands to help debug the test jig.

5 Test Descriptions

This section describes each of the decoder tests in detail. You must program the locomotive or accessory decoder for the proper address, delay times, etc., prior to running the tests.

5.1 DCC one (1) Margin Test

This test determines the minimum and maximum 1 bit times that the decoder will accept. It sets the 0 bit time to the nominal value and then changes the 1 bit time up and down to determine the range.

It does this by using a version of the packet acceptance test described in section 5.3.2 DCC Packet Acceptance Test to verify that the decoder will pass 100% of the packets for a given value of 1 bit time. It uses a binary search algorithm to select the next 1 bit time to test and then runs the packet acceptance test for 100 cycles or until a failure occurs.

5.2 DCC one (1) Duty Cycle Test

This test determines the minimum and maximum 1 bit duty cycle that the decoder will accept. It sets the 0 bit time to the nominal value and then changes the 1 duty cycle up and down to determine the range. It does this by using a version of the packet acceptance test described in section 5.3.2 DCC Packet Acceptance Test to verify that the decoder will pass 100% of the packets for a given value of 1 duty cycle. It uses a binary search algorithm to select the next 1 bit time to test and then runs the packet acceptance test for 100 cycles or until a failure occurs.

5.3 Repetitive DCC Test

All of the tests in this section are repeated for a variety of 1 and 0 bit times, 0 stretch values, etc. The 1 and 0 bit times are set and then each test sequence is run. The exact 1 and 0 low and high times are printed out prior to each test series. All 1 and 0 bit times are within the specification so all tests should pass for all 1 and 0 bit times.

5.3.1 DCC Ramp Test

This test sends all valid baseline commands to the decoder and verifies that it responds. The exact tests vary for locomotive and accessory decoders. For locomotive decoders, you can visually verify that the motor and optional lamp states change as shown on the computer screen. For accessory decoders, you can verify that the decoder changes polarity if the decoder provides a status lamp.

The program reads back the Voltage polarity from the decoder output and verifies that it is correct. For locomotive decoders, this is only done for the higher speed steps to make sure the motor Voltage is clearly positive or negative. For accessory decoders, they cycle through each of the possible output pairs and also cycles the active bit on and off.

Note that the tester can only detect output polarity and does not verify the Voltage output for locomotive decoders. You should connect a Voltmeter to the locomotive decoder outputs to verify that the Voltage changes at each speed step. You can also test this by using the manual decoder tests described in section 6.3 Manual Commands.

5.3.2 DCC Packet Acceptance Test

The packet acceptance test verifies that at least 95% of good packets are properly received by the decoder. For locomotive decoders, the test begins by presetting the decoder to one half speed reverse. This is done by repeatedly sending the proper reverse speed command. The program then verifies that the motor Voltage is correct. Next, the program sends the appropriate number of idles for that test cycle, one emergency stop command with the appropriate number of preamble bits, and one second of idle commands. The idle commands are sent to give the decoder time to stop the motor. The program then verifies that the motor has stopped. This sequence is repeated 100 times, at least 95 of which must pass.

For accessory decoders, the test begins by presetting the decoder to activate the switch in one direction. This is done by repeatedly sending the proper accessory command. The program then verifies that the output Voltage is correct. Next, the program sends the appropriate number of idles for that test cycle, one command with the appropriate number of preamble bits to reverse the switch state, and one second of idle commands. The idle commands are sent to give the decoder time switch state. The program then verifies that the decoder has switched state. This sequence is repeated 100 times, at least 95 of which must pass.

The overall 100 test sequence is repeated for different numbers of inter-packet idles and different numbers of packet preamble bits. These idle and preset counts are displayed in the test log and summary files. For example, the line that begins ‘**pre 10 idle 1**’ would indicate that the test was run with 10 preamble bits and one inter-packet idle command. It should be noted that the ‘**pre 10 idle 0**’ test does not allow enough inter-packet time to meet S-9.2. It is included in the sequence to stress test the decoder and a decoder may fail this test and still conform to S-9.2.

5.3.3 Bad Address Test

This test verifies that the decoder only responds to its address and no others. It uses the same preset and trigger commands used for the locomotive and accessory packet acceptance test described in section 5.3.2 DCC Packet Acceptance Test. The program begins each test cycle by pre-setting the decoder to the initial state. It then switches to an incorrect address and attempts to change the state using one second of commands that are correct except for the address. In no case should the decoder switch state.

This test is repeated for each invalid address. A final test is then done with the valid address to verify that the decoder is responding to a good command.

5.3.4 Bad Bit Test

This test verifies that the decoder rejects all commands with a single wrong bit. It tests all bits in a packet from the first bit of the preamble to the last bit of the checksum.

Each test cycle begins by presetting the decoder as in the packet acceptance test. A command to change state with one wrong bit set is then repeatedly sent to the decoder for a period of one second. A properly working decoder will not respond to this command and change state.

This test is repeated for the preamble, address, command, and checksum portions of a command packet. A final test is then done with all bits valid to verify that the decoder is responding to a good command.

5.3.5 Single Stretched Zero (0) Test

This test verifies that the decoder will accept a packet with a single stretched 0 in the packet. It is similar to the Packet Acceptance Test except that the first 0 after the preamble is stretched by the appropriate amount for that test cycle.

Each test cycle begins by presetting the decoder as in the Packet Acceptance Test. A single command to change the state with a stretched 0 is then sent. This is followed by one second of idle packets. A properly working decoder should accept this trigger packet and work exactly as it did during the ‘pre 10 idle 1’ portion of the Packet Acceptance Test.

The total length and high time in microseconds of the single stretched 0 is shown in the log and summary files for each test cycle. For example, ‘0T 12000 0H 11890’ would be a stretched 0 with a total duration of 12000 microseconds and a high time of 11890 microseconds.

5.4 Truncated Packet Test

Note: This test is not run by default and is not used to determine pass or fail criteria. It is included to gather data on how decoders accept packets that are preceded by truncated packets.

This test determines if a decoder will accept a trigger packet that is immediately preceded by a truncated packet. It does this by using a version of the packet acceptance test described in section

5.3.2 DCC Packet Acceptance Test to verify that the decoder will accept a trigger packet that is preceded by a packet that is truncated by one bit each test cycle. The length of the truncated packet is displayed in the log file.

5.5 Prior Packet Test

This test determines if a decoder will accept a trigger packet that is immediately preceded by a packet with various combinations of 0 and 1 bits. It does this by using a version of the packet acceptance test described in section 5.3.2 DCC Packet Acceptance Test to verify that the decoder will accept a trigger packet that is preceded by a packet that has various bits changed each test cycle. A conforming decoder should accept all trigger packets regardless of the type of packet preceding the trigger packet. The exact bits sent prior to the trigger packet are displayed in the log file.

5.6 Six (6) byte Prior Packet Test

This test determines if a decoder will accept a trigger packet that is immediately preceded by a 6 Byte packet without a packet end bit. It does this by using a version of the packet acceptance test described in section 5.3.2 DCC Packet Acceptance Test to verify that the decoder will accept a trigger packet that is preceded by a variety of 6 Byte packets without packet end bits. A conforming decoder should recognize these 6 Byte packets as too long to be valid and accept the trigger packet. The exact Bytes sent prior to the trigger packet are displayed in the log file.

5.7 One (1) Ambiguous Bit Test

This test determines if a decoder will accept a trigger packet that is immediately preceded by a packet with 1 inter Byte 0 replaced by an ambiguous bit. It does this by using a version of the packet acceptance test described in section 5.3.2 DCC Packet Acceptance Test to verify that the decoder will accept a trigger packet that is preceded by a packet that has various bits changed each test cycle. A conforming decoder should accept the trigger packet even if the previous packet has an inter Byte 0 bit converted to an ambiguous bit. The bit position of the 0 replaced by the ambiguous bit is displayed in the log file.

5.8 Two (2) Ambiguous Bits Test

This test determines if a decoder will accept a trigger packet that is immediately preceded by 2 ambiguous bits. It does this by using a version of the packet acceptance test described in section 5.3.2 DCC Packet Acceptance Test to verify that the decoder will accept a trigger packet that is preceded by a packet that has various bits changed each test cycle. A conforming decoder should accept the trigger packet even if it immediately preceded by 2 ambiguous bits. The 2 ambiguous bits are inserted immediately after the packet end bit of the last preset packet and are immediately followed by the preamble of the trigger packet. The test results are displayed in the log file.

6 SEND.EXE Reference

This section of the manual gives details on running the **SEND.EXE** program. This manual was updated to match revision B.2.1 of the **SEND.EXE** program. Typing:

> SEND.EXE -u

will create a file named ‘**S_USER.TXT**’ in the current directory. This file contains a summary of the command line and initialization parameters described in the following sections. You should run this command and print out the resulting file in order to get an up-to-date command summary for **SEND.EXE**.

6.1 Command Line and SEND.INI Parameters

Several program parameters can be modified from the command line or in the **SEND.INI** file. The program first looks for the **SEND.INI** file in the following places, stopping at the first file it finds:

1. In the directory you were in when you started the program.
2. In the location specified by the **SEND_INI** environment variable, if set. You can use the DOS **SET** command to set this environment variable to the entire path and file name of the **SEND.INI** file.
3. In the same directory as the **SEND.EXE** program.

The program next reads any parameters passed in on the command line. These parameters will override any equivalent parameters set in the **SEND.INI** file.

Table 2 below shows the complete set of command line parameters and associated **SEND.INI** key words.

Table 2: SEND.EXE Configuration Parameters

Switch	*INI Parameter*	*Description*	*Default*
-?		**Print usage message and exit program.**	**FALSE**
-u		**Print S_USER.TXT file and exit program.**	**FALSE**
-m	**MANUAL**	**Don’t automatically run decoder tests.**	**FALSE**
-a <addr>	**ADDRESS**	**Decoder address.**	**3 - Locomotive** **2 - Accessory**
-d <l|a>	**TYPE**	**Decoder type.**	**Locomotive**
-p <port>	**PORT**	**Base sender board IO port number.**	**0x340**
-f	**FRAGMENT**	**Test all fragments**	**FALSE**
-x	**CRITICAL**	**Disable interrupts during critical regions.**	**FALSE**
-r	**REPEAT**	**Repeat decoder tests continuously.**	**FALSE**
-t <mask>	**TESTS**	**Bit mask of tests to run.**	**0xffffffff**
-c <mask>	**CLOCKS**	**Bit mask of clocks to run.**	**0xffffffff**
-E <pre>	**EXTRA_PRE**	**Extra margin test preamble bits.**	**0**
-T	**TRIG_REV**	**Use reverse as trigger command.**	**FALSE**
-F <fill>	**FILL_MSEC**	**Fill time in milliseconds.**	**1000**
-R <reps>	**TEST_REPS**	**Non packet acceptance test repeats.**	**4**
-P	**LOG_PKTS**	**Send packets to log, not hardware.**	**FALSE**
-A	**NO_ABORT**	**Do not stop program on an error**	**FALSE**
-s	**LATE_SCOPE**	**Put scope trigger after event**	**FALSE**

Each parameter is described below:

MANUAL This flag prevents the decoder tests from running automatically. The program will run the self-tests and then wait for a command from the keyboard without running any decoder tests.

ADDRESS This is the decoder address. The default value is address ‘3’ for locomotive decoders and address ‘2’ for accessory decoders.

TYPE This is the type of decoder to test. The possible values are ‘l’ for locomotive decoders and ‘a’ for accessory decoders. The default is to test locomotive decoders.

PORT This is the IO port address used by the sender board. The default port is 0x340.

FRAGMENT Setting this flag causes all fragment lengths of the truncated packet test to be sent. By default, it sends just the packet fragments that must be detected to meet the standard.

CRITICAL This flag disables interrupts during critical sections of the code. This flag should only be used if you get errors running tests because of terminate and stay resident (TSR) programs such as disk caching routines, etc. The default is not to disable interrupts.

REPEAT This flag causes the decoder tests to be run repeatedly. You must type the <ESC> key followed by the ‘q’ key to stop the decoder tests if this flag is set. The default is run the decoder tests only once and then stop.

TESTS This flag takes a bit mask as a parameter and can be used to disable one or more decoder tests. The first bit in the mask activates the first test, etc. By default, all decoder tests are run.

CLOCKS This flag takes a bit mask as a parameter and can be used to disable one or more of the clock sequences used for the repetitive decoder tests described in section 5.2 DCC 1Duty Cycle Test . The first bit in the mask activates the first clock sequence, etc. By default, all decoder clock sequences are used.

EXTRA_PRE This is the number of extra preamble bits to send before the trigger command in the DCC 1 Margin Test. The default value is ‘0’ which results in a total of 10 preamble bits being sent as part of the trigger command.

TRIG_REV This flag changes the type of trigger command used for the Packet Acceptance Test. By default, an emergency stop command is used as the trigger command. Setting this flag will cause a half speed forward command to be sent as the trigger instead of the emergency stop command.

FILL_MSEC This is the amount of idle time to wait between the trigger command and testing the output. This time is filled with idle commands. The default value is ‘1000’ milliseconds.

TEST_REPS Sets the number of times to repeat non packet acceptance tests such as the prior packet test. The default is ‘4’ repeats.

LOG_PKTS When set, the packet data is formatted and sent to the log file rather than to the sender hardware. This allows the user to see exactly what data is being sent to the sender board for each test. No actual hardware need be installed to run the program with this flag set. Note: Setting this flag sends tremendous amounts of data to your disk. You should limit the tests in order to prevent your disk from filling up.

NO_ABORT Normally, the program terminates if it detects any hardware errors such as underflow. Setting this flag will prevent the termination. This flag is for debug only and should never be set for a normal decoder test.

LATE_SCOPE Normally, the scope trigger is pulsed just before the beginning of the trigger event. Setting this flag causes scope trigger to occur one idle packet after the trigger event.

6.2 Log and Summary Files

The program creates and log and summary file to store test results. The file names are based on the base file name you entered when the program started. For example, if you entered '**ACME**' as the base name, the program would create a file called **ACME.LOG** to store the full test log and a file called **ACME.SUM** to store the summary statistics for the tests.

The log file contains complete test data including a full description of any errors found. The log file can grow quite large if many errors are found. The summary file only includes a summary of the number of tests passed, etc., and is much shorter than the log file.

6.3 Manual Commands

Under normal operation, the program will run one set of self-tests and one set of decoder tests and stop. It is also possible to send packets, etc., manually. The following manual commands are available from the command line prompt:

ESC - Return to command line
h - Print header

c - Send single clock phase
C - Send series of clock phases

u - Clear underflow
0 - Send zeros

1 - Send ones
a - Send scope A pattern

b - Send scope B pattern
o - Send scope timing packet

w - Send warble packets
S - Send stretched 0 pattern

r - Send DCC reset packets
d - Send DCC packets

D - Send stretched DCC packets
s - Change loco speed, acc. output

e - Set speed to E-STOP
f - Change loco direction, acc. on/off

E - Set speed to E_STOP(I)
t - Run self-tests repeatedly

z - Run decoder tests
i - Send DCC idle packets

R - Send hard resets
g - Test generic I/O

q - Quit program

All commands are entered as a single letter at the command line prompt shown below:

Command mode (h for help, q to quit) >>

The commands shown below control overall program operation and are used most often:

h - Print the command summary shown above.

z - Begin running decoder tests. The tests can be interrupted by typing the **<ESC>** key followed by the '**q**' key.

ESC - Interrupt a manual command and return to the command prompt.

q - Stop the program and return to the operating system.

The commands shown below are helpful for verifying that a decoder is connected and running properly:

0 - Send continuous 0s.

1 -Send continuous 1s.

o - Send scope time base test pattern. This will send a square wave with a period of 10 µsec., 100 µsec., 1000 µsec., and 10000 µsec. respectively as the ‘o’ key is pressed repeatedly.

r - Send continuous DCC reset packets.

R - Send continuous DCC hard reset packets.

V - Send continuous idle packets.

v - Send continuous warble packets consisting of a string of 0s followed by a string of 1s.

a - Send a special pattern with the scope trigger output active. It sends a 0 Byte with the scope trigger active followed by a 1 Byte with the scope trigger inactive. This sequence is repeated continuously.

b - Send the alternate scope trigger pattern. . It sends a 1 Byte with the scope trigger active followed by a 0 Byte with the scope trigger inactive. This sequence is repeated continuously.

S - Send the single stretched 0 test pattern. . It sends a 0xff Byte followed by a 0x02 Byte with a stretched most significant bit. This sequence is repeated continuously.

g - Test generic input and output bits. The first time you execute this command; the generic outputs are set to 0x15, and all inputs are read. Repeating the command will toggle the output bits and reread the input bits.

The ‘**d**’ or ‘**D**’ commands are used to send a continuous stream of baseline DCC or accessory packets depending on the type of decoder specified by the **TYPE** parameter. Several commands can be used to change the state of the DCC packet stream once it is active. The ‘**d**’ and ‘**D**’ commands are shown below followed by all the commands that change their behavior:

d - Send a continuous stream of DCC baseline packets for a locomotive decoder or accessory packets for an accessory decoder.

D - Send a continuous stream of DCC baseline packets for a locomotive decoder or accessory packets for an accessory decoder. The first 0 following the preamble in each packet is stretched.

f - Change the direction of the DCC baseline packet or change the switch state of the accessory packet.

l - Toggle the lamp state in the locomotive DCC packet.

s - Go to the next speed step in the locomotive DCC packet or switch to the next switch output of the accessory decoder.

e -Begin sending emergency stop commands in the locomotive DCC packet.

E - Begin sending direction independent emergency stop commands in the locomotive DCC packet.

The last commands are used to debug the sender board hardware and are not generally used:

t - Begin running self-tests continuously. Typing the ‘q’ key will interrupt the tests.

c -Send a single clock phase.

C - Use the series of software clocks.

u - Clear the underflow flag.

7 Document History

Date	**Description**
12-Aug-2015	First Release. K West
28-Feb-2025	Migrated to new template. Renamed TR-1-2015 to resolve a numbering conflict. Various format, spelling and grammar changes. JM Juett

Important Notices and Disclaimers Concerning NMRA Standards Documents

The Standards (S), Recommended Practices (RP), Technical Note (TN), and Technical Information (TI) documents of the National Model Railroad Association ("NMRA Standards documents") are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page, appear in all standards and may be found under the heading "Important Notices and Disclaimers Concerning NMRA Standards Documents."

Notice and Disclaimer of Liability Concerning the Use of NMRA Standards Documents

NMRA Standards documents are developed within the Standards and Conformance Department of the NMRA in association with certain Working Groups, members, and representatives of manufacturers and sellers. NMRA develops its standards through a consensus development process, which brings together volunteers representing varied viewpoints and interests to achieve the final product. NMRA Standards documents are developed by volunteers with modeling, railroading, engineering, and industry-based expertise. Volunteers are not necessarily members of NMRA, and participate without compensation from NMRA.

NMRA does not warrant or represent the accuracy or completeness of the material contained in NMRA Standards documents, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard or recommended practice, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, NMRA disclaims any and all conditions relating to results and workmanlike effort. In addition, NMRA does not warrant or represent that the use of the material contained in NMRA Standards documents is free from patent infringement. NMRA Standards documents are supplied "AS IS" and "WITH ALL FAULTS."

Use of NMRA Standards documents is wholly voluntary. The existence of an NMRA Standard or Recommended Practice does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the NMRA Standards documents. Furthermore, the viewpoint expressed at the time that NMRA approves or issues a Standard or Recommended Practice is subject to change brought about through developments in the state of the art and comments received from users of NMRA Standards documents.

In publishing and making its standards available, NMRA is not suggesting or rendering professional or other services for, or on behalf of, any person or entity, nor is NMRA undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any NMRA Standards document, should rely upon their own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given NMRA Standards document.

IN NO EVENT SHALL NMRA BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: THE NEED TO PROCURE SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD OR RECOMMENDED PRACTICE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

NMRA's development of NMRA Standards documents involves the review of documents in English only. In the event that an NMRA Standards document is translated, only the English version published by NMRA is the approved NMRA Standards document.

Official Statements

A statement, written or oral, that is not processed in accordance with NMRA policies for distribution of NMRA communications, or approved by the Board of Directors, an officer or committee chairperson, shall not be considered or inferred to be the official position of NMRA or any of its committees and shall not be considered to be, nor be relied upon as, a formal position of NMRA.

Comments on Standards

Comments for revision of NMRA Standards documents are welcome from any interested party, regardless of membership. However, **NMRA does not provide interpretations, consulting information, or advice pertaining to NMRA Standards documents.**

Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since NMRA standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, NMRA, its departments, Working Groups or committees cannot provide an instant response to comments, or questions except in those cases where the matter has previously been addressed. For the same reason, NMRA does not respond to interpretation requests. Any person who would like to participate in evaluating comments or in revisions to NMRA Standards documents may request participation in the relevant NMRA working group.

Laws & Regulations

Users of NMRA Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any NMRA Standards document does not constitute compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. NMRA does not, by the publication of NMRA Standards documents, intend to urge action that is not in compliance with applicable laws, and NMRA Standards documents may not be construed as doing so.

Copyrights

NMRA Standards documents are copyrighted by NMRA under US and international copyright laws. They are made available by NMRA and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of modeling, structural and engineering practices and methods. By making NMRA Standards documents available for use and adoption by public authorities and private users, NMRA does not waive any rights in copyright to the NMRA Standards documents.

IMPORTANT NOTICE

NMRA Standards documents do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other systems, devices or networks. NMRA Standards documents development activities consider research and information presented to the standards development group in developing any safety recommendations. Other information about safety practices, changes in technology or technology implementation, or impact by peripheral systems also may be pertinent to safety considerations during implementation of the standard. Implementers and users of NMRA Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.