IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DXQBC-B-D

PRODUCT NAME: DECNET DEC/X11 USER'S GUIDE

DATE: JANUARY 1977

MAINTAINER: DEC/X11 DIAGNOSTIC GROUP

How to use DECNET DEC/X11 Modules

1.0 INTRODUCTION

Every installation has one particular configuration of communications equipment and it would be nice if this product, DECNET DEC/X11 (D.D.), would match your installation perfectly with no modifications. However, since we are required to test many installations, you, the user, must custom tailor your D.D. exerciser to match your equipment. The number of variations possible with communications equipment is almost endless, and if you match 99% of the D.D. exerciser characteristics to your installation characteristics it is still not good enough. You must specify a D.D. exerciser that EXACTLY matches your equipment to avoid wasted hours of chasing "non-bugs". If you successfully build a D.D. exerciser you will have a very useful product. The D.D. exerciser can now do complete DEC/X11 system exercising while simultaneously exercising all your communications lines, using the exact same data paths and data links as the operating system normally run on your system. Every attempt has been made such that a user of D.D. can build and use an exerciser without requiring any documentation on hand, if, and this is a big if, he has a complete knowledge of normal DEC/X11 building and running and he has read and understood this entire document. If you come across any instance where this is not true, please notify the DEC/X11 group in Maynard and the situation will be remedied. If you are completely unfamiliar with communication equipment, I suggest you get and read Introduction to Data Communication by Murphy and Kallis, and Introduction to Minicomputer Networks by Stelmach. These are both small, easy reading paperbacks available from DEC.

2.0 BUILDING A DECNET DEC/X11 EXERCISER

****
You must be familiar with building a normal DEC/X11 exerciser. If you are not, please read, and then build and run some exercisers before continuing on with this section.
****

2.1 Modules Required

There is a one-to-one correspondence between communications options which D.D. supports and D.D. modules used to test these options. The D.D. modules have a 4 letter name, NXO?.OBJ, where, NX is formed by replacing the D in a communications option name with an N, the O is an option selection, and ? is the revision letter. (Throughout the rest of this document, a ? as the last charcater in a module name indicates the latest revision available.) For example, the first D.D. module written to test the DL11-E option was called NLAA.OBJ. NL comes from changing DL to NL, the next A indicates which option type. (In the case of the DL, there are no others to confuse it with, but if a DLX11 module was ever manufactured, the module name for it would be NLBA.OBJ.) D.D. modules will, in general, support 16 lines per module. In the case of a single line device, such as the DU11, the NUA? module will handle up to 16 devices, in the case of multiplexor devices with 16 lines, only one device is supported per module. So if your system had 2 DH11's, you would have to configure 2 NHA? modules to drive all 32 lines. The exception to this rule will be the DZ module, NZA?, which only supports 1 device, eight lines.

In addition to these device dependent modules, 2 extra utility modules must be configured to support any D.D. module. The first, N1A?.OBJ, contains a message buffer used by all D.D. modules to transmit from. In addition, it contains many common subroutines which all D.D. modules use. This helps keep down the size of the D.D. modules. The second module, N2A?.OBJ, serves as a "synchronization" module, it gives all systems being exercised in the network a "pause" point, to enable all data links to be established. It also services any DN11's (auto-call devices) which a system might have. Lastly, a clock module must be configured, either KWA? or KWB?. These modules keep a system time for the DEC/X11 monitor. D.D. modules have many time out loops which are implemented by using this system clock.

****
In the future, a fake clock module, KWX?, will be written, this will be a background module which will up the system time, but in an extremely inaccurate manner. This will allow D.D. modules to be run on systems without a KW11-L or KW11-P clock. The time-out loops in the D.D. modules will likely become undesirably long, but this cannot be avoided.
****

For any given communication option, there will likely be two DEC/X11 modules, the normal one whose first letter of the name is a D (e.g., DLA?), and a DECNET DEC/X11 one, whose name starts with an N (e.g., NLA?). They are mutally exclusive. You should not configure two modules to test the same device. It is ok to, for example, configure DLAF to test one DL11-E and NLA? to test a second, but extreme care must be taken not to have 2 modules trying to test the same device. In the case of a multiplexor, such as the DJ11, it is impossible to test, say, 4 lines in DECNET DEC/X11 mode, and the 12 in normal DEC/X11 maintenance mode, at the same time.

2.2 The order of Modules

In normal DEC/X11, there are no ordering constraints with regards to modules. They can be configured in any order you desire, with the possible exception of the clock modules. They should be configured directly after the monitor to insure the run time is not made inaccurate by any delay in the clock module being initiated, but this is not a requirement. When building a D.D. exerciser, there are some very definate rules about ordering of modules. In particular, directly after the monitor, the N1A? module must be configured. Following it, in any order, ALL D.D. modules must come, followed by the N2A? module. Next the clock module should come, followed by all the rest of the normal DEC/X11 modules in any order. This results in a exerciser that looks like:

```
Monitor (normal DEC/X11 or 11/70 monitor)
N1A?
N?A?    !Any and all
  .     !D.D. type
  .     !modules
N?A?
N2A?
KW??     Clock module
XXXX    !Any normal
  .     !DEC/X11 modules
  .     !in any order.
XXXX
```

This results in a "sandwich" of all D.D. modules with N1A? on top and N2A? on bottom. If these ordering rules are violated many different errors will result. If no clock module is included in the configuration, the exerciser will halt itself after typing a message, and will therefore be useless. The library file which contains all these D.D. type modules has N1A? as the first module and N2A? as the last, so when using the MAKE command to build an exerciser, this library should be specified after the monitor library and this will guarantee the correct order of modules.

2.3 Module Parameters at Configuration Time

When building a D.D. exerciser, DVA, VCT, BR1, BR2, DVC, and SR1 can all be specified, exactly as is done in normal DEC/X11. DVA and VCT should be the lowest address of the devices which any one module is intended to exercise. BR1 and BR2 should be the highest of the devices receiver and transmitter bus request levels, respectively. DVC, at configuration time, is the number of lines to be tested for that module. It is converted into a bit map in the resulting exerciser. SP1 should be set to a bit map, with bits=1 for lines which should act as Master and 0 for lines to act as Slaves. Master and Slave are defined in section 3. For example, if your system had 3 DU11's, with the following hardware parameters:

#1 DU11, at 175610 and vector at 400, receiver priority of 5 and transmitter priority of 5, to be tested as a Master.

#2 DU11, at 175620 and vector at 410, receiver priority of 5 and transmitter priority of 5, to be tested as a Slave.

#3 DU11, at 175630 and vector at 420, receiver priority of 6, and transmitter priority of 5, to be tested as a Master.

You would configure the NUA? module, specifying DVA of 175610 (the lowest of the 3), vector of 400 (the lowest of the 3). BR1 as 6 (the highest of the receivers) and BR2 of 5 (the highest of the transmitters). DVC should be 3, meaning 3 lines to be tested, however, if you examimed location 14 in the header of NUA? in the resulting run time exerciser, you would find it was a 7, being 3 binary bits on, one for each line. SR1 should be a 5, being bit 0 on, bit 1 off, and bit 2 on, corresponding to the Master, Slave, Master testing modes which we wanted for line 0, 1 and 2. All of these parameters may be altered when running the exerciser, but remember the different method of representing DVC at configuration and run time. If the bit for a line is not set in DVC, this means the line will not be accessed at all, so the corresponding bit in SR1 will have no meaning.

None of these 6 parameters have any meaning for the N1A? module. For the N2A? module SR1 should be zero unless you have 1 or more DN-11's, in which bit 0 should be set to a one.

NOTE , this DN-11 option is unsupported so far because no hardware has been available to test it on.

Set DVA for the N2A? module to the lowest device address of your DN-11's, VCT to the lowest vector of your DN-11's, and BR1 to the interupt level.

3.0 TESTING THE NETWORK NODE

After successfully building a D.D. run time exerciser, you are now ready to test your Network Node.

3.1 Which Tests to Run

To assure correct identification of any problems, I would strongly recommend running stand-alone diagnostics on all devices first. If they all pass, you should run a normal DEC/X11 exerciser which exercises all devices, including the communication equipment. If this passes, then continue on and attempt to run DECNET DEC/X11. Of course there are many exceptional cases where this "bottom up" testing is impossible for time or other reasons. If you are positive a problem is in some specific communication device, then you might skip running diagnostic on all the other equipment and just run the stand-alone on that one communications device, if that passes you should then run regular DEC/X11 to all the devices, and only then, run DECNET DEC/X11. If you do not think there are any problems with the system and only want to run a confidence test on the system or the quality of the communication links, you might go directly with DECNET DEC/X11, but if problems develop you should fall back to the simplest test which will fail, either normal DEC/X11, or preferably, a stand-alone diagnostic.

3.2 Running DECNET DEC/X11, Overview

NOTE

Reference is made to "multi-drop" testing and DN11 auto-call units throughout this document and individual DECNET DEC/X11 module documents. Both these features are "designed in" but completely untested and will not be supported until some future unknown date.

First boot the system, load and start the D.D. exerciser. It will look exactly like normal DEC/X11. You can use the MAP command to find out which and where the modules are. You can use the MOD command to modify any parameters or options. You can use the SEL and DES commands to turn on and off modules.

****

WARNING

The N1AA, N2AA, clock, and at least 1 DECNET DEC/X11 module must be left selected if any D.D. module is selected. You should not select N1AA without N2AA and at least 1 DECNET module, etc. It is perfectly all right to deselect N1A?, N2A? and ALL D.D. type modules, and just run normal modules, or deselect all normal modules and run just N1A?, N2A?, clock, and one or more D.D. modules, but any D.D. type module which is selected positively requires N1A?, N2A?, and the clock, to be selected, and they in turn, if selected, expect at least 1 D.D. module to be running.

****

The first time an exerciser is run, it will type out a table of error codes and their meanings for D.D. modules. This is further explained in the section on errors. (Leaving bit 0 up in the switch register will inhibit this typing)

First deselect N1A?, N2A?, and ALL D.D. modules and select all other modules, Run this for 10 minutes, or long enough for every module to make 2 passes, whichever is longer. Next, deselect all modules and select N1A?, N2A?, and all D.D. type modules, and the clock module. Run this. It will not begin running all modules yet. It will ask you to select which message you want to use for Network testing. You have a choice of all zeros, all ones, alternating 1 and 0's, a digit count pattern, or a precanned ASCII message, (called the standard message). These are all 501 character long messages, plus 11 characters which get added as a header and CRC's. You can also type in any message you choose which will also have 11 characters added to it. Lastly, you can specify to use the message which was previously specified. (This is for re-starting the exercising when you have not re-booted.) This message which you specify is the only message which the D.D. modules will use. It is very important that any node which you are connected to, must use the EXACT same message, otherwise, you will have false errors reported. At this point, the N1A? module will do an end of pass call and, because it is a special NBKMOD, it will not run any more passes. More about this will be mentioned later in this document.

Next, each one of the DECNET DEC/X11 modules will be started, in turn. When started, they will check to see which lines you have selected to test (by the DVC command when configuring or by MOD'ing location 14 in the module's header). For each line selected, the module will ask if the line is to be tested in full duplex or not (not means half duplex). Remember that although most of Digital's communication equipment is capable of running full duplex, many modems are not. If the modem for that line is not, you must run half duplex. Next it will ask for each line selected, if that line is to be tested in Multi-drop mode or not (not means point-to-point). If the answer is yes, it will ask for the station address of that multi-drop line, or if that line has been selected as a Master in SR1, it will ask for station addresses to be tested repeatedly until either 15 addresses have been entered or a - (minus sign) is typed. By far and away the most common lines are half-duplex and point-to-point. When the module has this information, it then sets the Data Terminal Ready line to the modem. A modem can not make and maintain a connection to another modem unless both modems have their Data Terminal Ready lines selected. The module will then EXIT to the monitor and wait for a Global location called WAIT in the N2AA module to be cleared, indicating that all phone connections have been made. This sequence is repeated by each D.D. module selected until they have all been started. By modifying location 164 in module N1A? to a 000001, the questions about lines will not be asked each time the exerciser is started. If the line parameters need to be changed, this location must be cleared. While answering line questions about a particular module, the "@" character will skip the rest of the questions for that module and leave the remaining parameters unchanged.

Next the N2AA module is started. It, if SR1 has bit 0 set,

will ask for a phone number to supply to a DN11 auto call option. It will then attempt to make the phone call, if it fails, it will type out;
DN ERROR X
where X is;
0 when present next digit bit is not set
1 when power in the auto-call unit failed or abandoned call and retry bit was set
2 when the software timed out while waiting for a call to complete
3 when power is not on in auto-call unit
After it makes that call it returns for another number until only a carriage return is typed, indicating no more DN11's. Next, or first if SR1 was 0 indicating no DN11's, the module will type a message requesting the operator to dial all manual connections, and type a carriage return when complete.

It is important that all nodes in the Network being tested have this message (or the 'P#' message if DN11's are selected) typed out before any phone connections are attempted. This is the only way you can guarantee that all modems have had their Data Terminal Ready line set. When ALL nodes have had the message for manual phone connections or the DN11 prompt for the first phone number (P#) typed, the operators at each node should make the required automatic and manual phone connections, typing the carriage return when done. This last carriage return is really the start of DECNET DEC/X11 testing. Immediately after it is typed, the global location WAIT is cleared, allowing all D.D. modules to begin attempting to transmit and receive the message specified. Also, all normal DEC/X11 modules (if selected) will be started. The N2A? module will then drop itself because it is no longer needed.

If this group of modules runs successfully for at least 8 passes each, you should then select all modules and attempt to run again. Location 164 in module N1A? can be MOD'ed to a 000001 if you want to skip the question and answer dialogue regarding multi-drop or not and full-duplex or not. If you later want to change these parameters, this location should be set back to zero before running. Also, when answering these questions for multi-drop and half-duplex, a "@" and carriage return indicates you do not wish to modify any more of these types of parameters for this module.

3.3 Legal Network Node Configurations and Modes

Any one line of a communications option may be connected to many different "ends" and this line may be run in many different modes. Furthermore, any one node running an exerciser can have many such lines and each line can be running in different modes and connected to different type "ends". As you can see, this can become quite complicated. To make it any simpler envolves taking away features and options, which is not desirable. I strongly recommend that a Network map be drawn showing each Node, what it is connected to each other node by (device type, address, vector, modem type, data link type, baud rate, and half or full duplex), and what software the Nodes will be running. If this is done carefully, many problems, such as connecting asynchronous devices to synchronous modems or connecting devices with unlike baud rates, can be avoided.

3/3.1 What Can a Line be Connected To?

Any one line from a communication device can be connected to:

a. Any simple "loop back" device which turns the transmit signals around to the receive side. Synchronous devices do not have a clock to strobe the data bits out with, so such a loop back device must have a clocking mechainism for synchronous devices. If this loopback device does contain a buffer to allow an entire message to be captured and THEN echoed back to the communications device, the communication device can be run in half or full duplex. If the loopback device does not buffer the message, but rather, just ties the transmitted data line directly to the receive data line, then the communication device must be run in full duplex, Master mode. Below is a list of known loopback devices which will work:

    None Yet

   This list will be added to as new loopback devices are tested and confirmed.

b. Another line on the same system subject to 1 restriction. If the 2 lines involved are being controlled by seperate D.D. modules, there is no problem, but if they are both being run by one D.D. module, then you must insure that both lines are being being run by the module at the same time. Later sections of this document will explain more about what this means and how to properly connect and select the software options to assure this. Although any one D.D. module generally handles 16 lines, it only maintains active I/O on 2 of these lines at any one instant. It does one send/receive iteration simultaneously on 2 lines, then does the same on the next 2 different lines, and so on, until it has done all 16 lines (if selected), then it does the first pair again, etc. If you want to tie one line back on another line in the same system and both lines are being controlled by the same module, you must make sure these 2 lines are one of the "pairs" that get run together. For more on this, read the section on "2 Lines at a Time Considerations".

c. Another computer system (Node). This other node can be another PDP-11 running DECNET DEC/X11 or it can be ANY type of processor by any company running any software (including its normal operating system) subject to 2 restrictions. First the other end of this line must be a compatible modem and communications device and second, the software must be using the DDCMP protocol (In particular, the Maintenance "loop back" message) or else be capable of accepting a message and echoing it back exactly as received. This other side must not send unsolicitated messages to our node. So, some typical nodes which can be connected to our node would be:

- A PDP-11 running RSX-11M (if it supports DDCMP)
- A PDP-11 also running DECNET DEC/X11
- A PDP-10 running an operating system supporting DDCMP
- A brand X computer running any software that will exactly echo messages which we send to it.

3.3.2 What Modes Can a Line Be Run In?

A line can be run:

a. Run or not run as selected by the bit map in location 14 of the module. Bit 0 on means test line 0.

b. Master or Slave as selected by the bit map in location 16 of the module. Bit 0 on means, if testing line 0, treat it as the Master. The only difference between Master and Slave is which side of the line expects to transmit first (Master) and which side expects to receive a message (Slave) before it transmits. Both sides transmit, receive, transmit, receive, etc., over and over, but one line (the Slave) waits to receive the first message before it starts the test sequence. One side of a line must be Master and the other Slave. It is purely arbitrary which side is which, as long as they are not both the same. If a line is connected to itself through some loop back device, it must be set to be Master. Likewise, if a line is connected to another line anywhere which is not running DECNET DEC/X11, it must also be set to Master.

c. Half or Full Duplex. By answering the questions which are asked when the run command is typed, any line can be set to either, provided the hardware and the modem can do either, otherwise, you must set the line what the hardware and modem dictate.

d. Point-to-Point or Multi-Drop. Again, this is set according to how you answer the questions at run time. There is a seperate section dealing with Multi-drop. Point-to-Point is the normal case of 1 device connected to 1 and only 1 other device.

e. One way only. If you have isolated a problem to a particular device, you may want to verify if the receiver or transmitter, or both are bad. For this reason, one-way-receive and one-way-transmit (OWR, OWT) are provided. To use this mode, in general, all other devices both communication and other would be deselected, and within the D.D. moule which is controlling the line in question, all lines but one would be deselected.(Of course, N1A?, N2A?, and the clock module must stay selected to support this D.D. module.) This would be mainly for simplicity of operation, and other modules can be left running if desired, Within the module which controls the line to be tested, location 176 contains an address (absolute, not relative to the module start as the 176 is) which contains a 401. If the location which 176 points to is patched from 401 to a 240 (a NOP), and the line being tested is set up to be half-duplex, point-to-point, Slave, then that line will only receive messages, and never attempt to transmit. It will check the received message for correct data. The other end of this line, be it another system, or another module within the same system, or even another line under the same module, should then be set to OWT. This is done by patching the absolute address pointed to by location 200 of the module from a 401 to 240, selecting the line to half-duplex, point-to-point, and Master. This line will now transmit only, never waiting to receive. Now if these 2 lines are run, one will transmit over and over, regardless of what the receiving side does, while the receive side, will attempt to receive and check only. If both sides of the link run error free, the same patches and line specifications should be reversed, allowing verification of the opposite path. Some points to have in mind when running OWR or OWT:

* If either locations pointed to by 176 (OWR pointer) or 200 (OWT pointer) in a module is patched from 401 to 240, and then the line is to be tested back in normal 2 way mode, these locations must be patched back to their original value (401).

* When a line is selected for OWT, that line is in effect "running free", i.e., it has to wait for nothing, and scoping can be done with no other considerations. However, when running OWR, this line will do nothing (except complain of the TIME-OUT errors) unless it is connected to another line which is doing OWT. So, it makes good sense to run a line OWT all by itself, but no sense to run a line OWR unless it is connected to a transmitting line.

* If other modules are selected while trying OWT or OWR on another line, it will work but the module will not have the complete system to itself, so it, on occasion, will have to wait for monitor service and queing, so scoping would be inconvenient.

* It is legal to run one line OWT and another OWR out of the same module, either looped to each other or to 2 other lines.

* It is even legal to run 3 to 16 lines out of a module with some of them set for OWT and/or OWR, but the resulting testing mode is complicated to understand or interpute. This method is not recommended for any but the expert user of D.D. For those that want to attempt this, read the following. Once the patch for OWT is made, all lines selected to run within that D.D. module as half-duplex, point-to-point, Master, will be run as OWT. If the patch is made for OWR, all lines selected to run within that D.D. module as half-duplex, point-to-point, Slave, will run as OWR. Any lines selected as multi-drop or full-duplex are unaffected by other lines in the same module running OWT or OWR. Half-duplex, point-to-point lines ~~selected as Slave are unaffected~~ by patches made to run a OWT line, as are half-duplex, point-to-point lines selected as Master unaffected by patches made to run a OWR line. Also, with multiple lines running (3 or more) all lines are not running at exactly the same instant, while 2 are running, the others are waiting their turn, so scoping is difficult and care must be taken not to connect a line of a module to another line of the same module if the 2 do not get run together as a pair. (See section on "2 Lines at a Time Considerations".)

3.4 Running Options

Besides the various modes of running (Section 3.3.2) and the various terminations (Section 3.3.1) there are other options within each D.D. module. These options are made by patching locations within the module before running. The addresses for any of these option patches are standard in every D.D. type module. The options include:

a. Location 164 is a switch which indicates whether or not all errors should be reported as they happen. If location 164 is non-zero (equal to n), then receiver errors are totalled (on a line by line basis); (rather than being typed out as they happen) every n passes this total number of errors for the previous n passes will be reported. The value of n is the value of location 164. The maximum number of errors that can be talleyed is 256 per line. Therefore if a line is not known to be of very good quality, the value of location 164 times the value of location 166 should be less than 128. (location 166 is defined next.) This means the number of messages per pass times the number of passes per summary report should be less than 128. In this way, if every message attempted has an error on both transmit and receive, the error tally will not overflow. If the line is of known good quality, this restriction may be relaxed. The default value is to report all errors, and not give error summary reports.

b. Location 166 indicates how many iterations per line should be done before an end of pass is called. An iteration is normally one transmit and one receive, in either order. The default value is 8.

c. Location 170 is a time scaling factor. There are many time out loops in the modules, some should logically be longer than others. The value in location 170 is the number of seconds to wait for the smallest time outs. Some time outs will be a multiple of this factor. Default is 30 seconds.

d. Location 172 is an error toleration level, on a single message basis. So if in an attempt to do one iteration (usually a transmit then recieve or vice-versa) the number of errors reaches the value in location 172, that line is dropped from testing. If before this level is reached, an iteration is correctly completed, then the error count which is compared to location 172 is reset to zero. The default is 10.

e. Location 174 contains 2 identical bytes for use as sync characters. The default is 113226 which is 2 bytes of 226. If this is changed, both bytes must be identical to each other. This location is a don't care for asynchronous devices, because they "sync" on the first byte of the actual message, not on preceding characters as synchronous devices do.

NOTE

The NQA? module and the NVA? module, if both configured in one exerciser, require that the same sync character be used even if they are not connected to each other. Since the DV-11 sync character is set by hardware, this location must be patched to match the DV-11 hardware. If you do have a DV-11 and a DQ-11, then the DV and DQ's sync characters must both be patched to match the DV-11 hardware (even if they are not connected to each other)

f. Location 176 is for One-way-receive. See Section 3.3.2, e.

g. Location 200 is for One-way-transmit. See Section 3.3.2, e.

h. Locations 322 through 341, the Baud Rate Byte Table, contain a coded baud rate byte for each corresponding bit in the DVC, location 14. This table is only present for asychronous device modules with programable baud rates. Refer to the individual module listing for what code corresponds to which baud rate.

3.5 Modems

Modems, or at least modem simulators, are always required for synchronous communications devices. Asynchronous devices do not for short cable distances. All D.D. modules are written expecting modems and modem control lines connected to the devices being tested. For asynchronous devices, the listing for individual D.D. module types may contain information on how lines can be tested without modems. The following modems have run D.D. modules successfully:

Synchronous	Asynchronous
Bell 201A	Bell 101A
Bell 208B	
Com Link II	

3.6 Multi-drop (Not Yet Tested, so Unsupported)

```
*******************************************
Multi-drop is complicated and unless  your
configuration   requires   it,   I   would
recommend not reading this section, as  it
may just add unneccessary confusion to the
understanding of DECNET DEC/X11.
*******************************************
```

Multi-drop, or Multi-point, is a hardware configuration in which more than 2 communication devices are connected to a single line. In the more common mode, point-to-point, 1 device has its transmitter connected to 1 other device's receiver and that device's transmitter is connected to the first's receiver. In Multi-drop there is one device (called Master Stations) connected to 2 or more devices (called Slave Stations). The slave stations can never "speak" (transmit) until "spoken to" by the Master. The D.D. exerciser which is running as Master will transmit one message on the line which is exactly like the message used in all point-to-point cases except the header's 6th byte (and the 7th and 8th which are the CRC-16 characters for the header). This 6th byte is the "address" byte. Every slave device on this multi-drop line receives this message and compares its own unique station address with this 6th byte. If they do not match, the message is ignored. If they do match, then the slave station is allowed to transmit one message on the line. This message is exactly what it received, and since the address byte is set for this transmitting slave, all other slaves must ignore this message also. The Master, which started this process, will be expecting this "echo". If it gets it ok, it will select the next slave station and send the message with a new unique address byte. If the Master does not get the echo, it will retry, up to the limit specified by location 172 of the module. (See section 3.4, d.)

When the D.D. exerciser is started, for every line selected to test, the operator is asked if this line is Multi-drop or not. If the answer is yes, (Y<cr>) then he will be asked for a station address for this line if it was selected to be a Slave (by bit map in SR1, location 16 of the module). If this line was selected to be Master and Multi-drop, then the operator will be asked for a series of addresses, one for each slave on the Multi-drop line. This table of Slave's addresses has room for 15 Slaves. If there are less, typing a "-" (minus sign<cr>) will terminate the questioning for addresses. These addresses should be any octal number from 0 to 177. Any one module can only run 1 line as Multi-drop Master. If you attempt to answer Y<cr> for a second time for a line which is Master, an error will be reported, and you will be asked the questions over.

3.7 Errors

There are 3 classes of errors reported by D.D. modules.

3.7.1 Set-up Errors

When specifying parameters or run time conditions, operator errors can be made. If detected because of incorrect or contradictory responses, the operator will be notified. This error conditions will be self explanatory.

3.7.2 Data Errors

When a message is received by a D.D. module, a common subroutine located in the N1A? module is called to check the data in the message. All DECNET DEC/X11 module data compare errors look like the standard DEC/X11 data compare error message except:

a. The module name typed is FFXX, where FF is filled in to be the name of the DECNET DEC/X11 module which called the N1A? module to do the compare. In other words FF represents the module that "had" the error.

b. The error number is a count of the data compare errors in the current message being checked, it is therefore reset to zero every time a new message is to be checked.

c. ACSR contains the address of the DECNET DEC/X11 module which called this routine to have its message checked.

d. SBADR contains the line # of the device which had the data error.

e. WASADR contains the count (in octal) of which character in the message this bad character was.

f. The "should be" and "was" items are normal.

NOTE, because the N1A? module keeps having its named changed by D.D. modules doing data compares, you cannot expect to know what name it is using at any one time. When first loaded, it is called N1XX, but it often gets changed while running. So do not be alarmed if in run summarys and maps, the name has changed to NQXX0, NLXX0, or whatever. If you want to deselect it or use the MOD command on it, use the name which last appeared in a run summary or MAP command.

3.7.3 Running Errors

Other running errors are reported using the normal error call in DEC/X11 CSRA and ACSR are standard. STATC contains a coded error number, with the code being consistent in all DECNET DEC/X11 modules. The low order 2 digits of STATC contain which line had the error the other digits contain this code:

0010XX	rec time out
0020XX	tmt clock loss
0030XX	header CRC error
0040XX	msg CRC error
0050XX	tmt time out
0060XX	tmt non-existent memory
0070XX	tmt latency
0100XX	rec data parity error
0110XX	rec non-existent memory
0120XX	rec clock loss
0200XX	rec framing error
0400XX	rec overrun or latency error

The first time the run time exerciser is run after loading, the NET1 module will type this error code list out to the system console for the operators subsequent use. (Unless switch 0 is up on the switch register)

3.8 Altering the Order of Line Testing

The module contains a table of bytes from location 212 to 231. Location 212 corresponds with bit 0 of DVC (location 14 in module), location 213 with bit 1, etc, up to location 231 with bit 15. These bytes contain the line number to test in the low order 4 bits. The upper 4 bits are unimportant. Modules as shipped, contain a 0 in location 212, 1 in 213, 2 in 214, etc., up to a 17 in location 213. This table in conjunction with the DVC location, tell the module which lines to run and in which order. The module takes DVC, starting with bit 0, and for each bit set to a 1, runs the line number specified by the corresponding byte, 212 to 231. Given the default values in 212 to 231 and DVC set to 177777, the module would test line 0, then line 1, then 2, etc. up to line 17. If for any reason, you would want to run the lines in any different order, just modify *BYTES* 212 thru 231 to the order you want, making sure that every number from 0 to 17 (octal) appears only once, in this byte table. For example, the default values of this table look like:

Location	Value	In Bytes		DVC	bit
212	000400	001	000	1	0
214	001402	003	002	3	2
216	002404	005	004	5	4
220	003406	007	006	7	6
222	004410	011	010	11	10
224	005412	013	012	13	12
226	006414	015	014	15	14
230	007416	017	016	17	16

This will test lines 0 thru 17 in ascending order. (Provided the corresponding bit in DVC indicates the line is to be tested.) If you wanted to test the lines in the following progression, 3, 11, 2, 17, 1, 7, 16, 6, 14, 0, 4, 10, 5, 12, 13, 15, you should modify the module to the following:

Location	Value	In Bytes		DVC	bit
212	004403	011	003	1	0
214	007402	017	002	3	2
216	003401	007	001	5	4
220	003016	006	016	7	6
222	000014	000	014	11	10
224	004004	010	004	13	12
226	005005	012	005	15	14
230	006413	015	013	17	16

If when you go to modify these locations the upper 4 bits are non-zero, do not bother writing these bits, they are set up at run time, just patch the lower 4 bits to what you want. Make sure no two bytes contain the same line number and remember that no matter what is done to this table, DVC, location 14, has the bit map of which lines will run. If a bit is not on in DVC, then no matter what is in the corresponding byte in this table, the line will not be run.

3.9 2 Lines at a Time Considerations

Due to program size constraints, D.D. modules will only maintain active message flows on 2 lines at any one instant. The program will, according to the lowest order bit which is on in DVC, location 14, take the first line to be tested, get it started, and then start the second line. These 2 lines will be doing simultaneous message transfers until either has done the required number of iterations (per location 166). Then that line that just finished will be dormant and the 3rd line (as selected by the 3rd lowest order bit in DVC) will become active. One of the original 2 lines may still be active with this 3rd line. Next this last line of the first 2 will finish and go dormant and the 4th line will be started. It will be running simultaneously with the 3rd line. This process repeats until the last line selected is complete.

Then an End-of-Pass call is made and the whole process starts over. If all lines were running in the same test mode and at the same baud rate, then lines 0 and 1 would run and finish, lines 2 and 3 would then run, etc. up the line 16 and 17 (octal). If however, line 0 was running 20 times slower than the other 15 lines (say because of different baud rates) then line 0 would start, and each line 1 thru 17, would be started and finished in turn, and still line 0 would not have finished. Why do you care about this? Because it is both possible and easy, to connect lines to lines such that a running line cannot finish because it must wait on another line which will not even get started till the first is finished. For example, suppose your system has 4 DL11-E's all at 9600 baud and you run all 4 using 1 D.D. module, NLA?. If you wanted to just loop your lines on each other for a standalone test you have 3 choices of connections. Two of them will not work. If you connect line 0 to line 1 and line 2 to line 3, it will run fine, when started, the module will run line 0 and 1 to completion, then lines 2 and 3, and then do an End-of-Pass. If you instead, connect line 0 to line 2, and line 1 to line 3, it will not work. The module will start line 0 and line 1, but the lines they are connected to are "asleep". Line 0 will transmit over and over and get no response from line 2 because line 2 is not actively running, it will not be started till line 0 is done! If you really wanted to test line 0 to 2 and lines 1 to 3, you have 2 choices, either configure 2 NLA? modules and let each run 2 lines at a time, in which case you will truly have 4 lines running at once, or else, using only 1 module, you can modify the order in which lines are tested (See section 3.8). By doing this, line 0 and line 2 will be selected for testing first, and then 1 and 3.

The same type of problems can occur between systems. Consider 2 systems, call them A and B, each with 4 DL11-E's. If lines A0 is connected to B2, A1 to B3, A3 to B0, and A4 to B1, the same type problem will happen. System A will make lines 0 and 1 active but they will get no action from B's line 2 and 3, because they won't get started until lines B0 and B1 are done. In this case, as in most interprocessor configurations, these mismatched situations will usually straighten themselves out after a "rocky" start. Either System A or B would likely have both its lines "time out" and give up, and go on to the next pair of lines. These would then run smoothly, and from then on, System A would be in the middle of a pass while System B would be at the end of a pass. There would, however, be a long time and about 30 time out error messages (8 on each of 2 lines on one system, and 7 on each of 2 lines on the other system) before the systems run.

So, to avoid these problems, always connect line 0's to line 0's, 1's to 1's, etc. BETWEEN modules whether the modules are running both on 1 system or on 2 seperate systems. If the lines are looped back onto lines from the same module, connect adjacent (0 to 1, 2 to 3, etc.) pairs together, evenly from the first line. In both cases, I mean line 0 to be the 1st line selected by the lowest order bit in DVC, the 2nd line would be the 2nd lowest order bit in DVC, etc.. For example, if you had a DVC of 010450, meaning lines 14, 10, 5 and 3 should be tested, connect 3 to 5 and 10 to 14.

You can connect 3, 5 or 25 processors together in rings or stars or whatever pattern you like. In all cases it will be possible to configure D.D. exerciser to get the entire network to play, but care must be used not to connect lines to lines such that line's other end cannot get started until this side is done.

4.0 The Future of DECNET DEC/X11

Modules were done for the DL, DU, DQ, and the DUP devices, but it does not appear that other devices will be supported at this time. Current thinking is that standard DEC/X11 testing along with individual line testing with ITEP will detect all errors.