The Transputer Network Tester

Neil Miller

Given perfect parts, a hardware system may not work properly because of wrongly connected cables, poor connections, electrical noise, poor design, etc. All Inmos products are, of course, completely tested before being shipped. However, in the real world of bad tracks, static discharges, and transputers designed into experimental systems, there is a need for diagnostic software.

Over the last year, a number of 'worm' programs have been developed, for use in exploring, testing and debugging various transputer systems. Some of these have now been put into a single program, for use as a general purpose diagnostic tool. At the same time, the opportunity is being taken to include tests for T800s, and provide a uniform convention for describing the network, and reporting errors. An algorithm for relating the physical network found to the one which the user specifies has also been added.

This document describes the new transputer network test program, gives interpretations of the error messages, and describes some of the more common problems encountered in running multiple transputer networks.

1 What the 'Network Tester' Does

This document describes the use of the transputer network test program, both for checking the configuration in which a number of transputers have been connected, and for pinpointing any hardware problems. The program is run as an **EXE** on the *master* transputer, typically a B004 (a version is also available for a B002), running on an IBM or similar pc. It is assumed that the master transputer is booted on link 0 from the host computer. Any of the remaining 3 links may be connected into a network of transputers, with a reset cable from the subsystem socket of the B004 controlling the rest of the network.

The only restriction on the network to be tested is that there is no path into the control link of a C004 (crossbar switch) - for otherwise the state of the C004 will be corrupted. However, the program can test a network of transputers whose connections are controlled by a C004.

The network debug program executes by sending a worm into a network of transputers. The worm explores the network, reaching every single transputer that is connected, no matter what configuration, and reports back the configuration which it finds. No initial assumption is made about the network. This should be contrasted with the loading of a network of transputers with a program of fixed configuration, which is the normal approach to developing programs using the transputer development system.

The worm algorithm is described in INMOS Technical Note 24, which forms a useful background to this documentation. It is important to realise that the worm numbers transputers in the order in which it finds them, which may bear absolutely no relation whatsoever to any conceptual order which the user has in mind.

For ease of use, however, the network test program can compare the network it has found against a user's **PROGRAM** specification, and give results in terms of the user's numbering, together with a statement on whether or not the two networks match.

The worm proceeds in two phases. Firstly, each transputer in the network is loaded up one by one with a copy of the worm program. As this happens, information about each new transputer found - the *loading data* - is relayed back to the master and displayed. At the same time, the error flag is briefly set high on the newly found transputer (halt on error has been set to false!) which lights up the error light, and is detected by the master in order to determine that error signals are being propagated back correctly.

Once the entire network has been explored, any further tests are performed on all transputers in the network in parallel. The *network test data* thus found, including a complete list of link connections, is reported.

Having completed testing, the program starts again by resetting or analysing the system and sending in a fresh worm.

The program is got and run as an **EXE**. If you want the results to be filed, then run the program while pointing at an ordinary text fold. When the program starts testing, a message 'Filing Output' is displayed on the screen, and results will appear in a new file at the end of the bundle. Otherwise, a message 'WARNING: NOT filing output' is displayed.

If you want to match results against your own network definition, run the program while pointing at your **PROGRAM** fold. If the matched results are to be filed, then pick the **desc** fold from your **PROGRAM** fold, and put it inside a text fold.

You are then prompted for an option. These are listed in the next section. Different options are appropriate to different circumstances, but for a quick check, try option 'C'.

After selecting an option, you will be prompted for a link from which to send the worm. This will be link 1, 2 or 3, according to which link is connected to the system to be tested. When the master transputer controls a module mother board, this will be link 2. Link 0 cannot be selected, since this is assumed to be connected back to the host computer. A link which is used to control C004 settings must not be used.

Some options allow all links of the master to be tried. Usually only one connection is made from the master into the rest of the system, but it is often useful to be able to explore a network from two different directions, in order to pin-point an error.

Finally, you will be asked whether you want results displayed in brief or in full. Brief mode simply presents a summary of results - whether a hardware error has been found, and whether the network found will match the one specified by the user (if, indeed, one was specified). Full mode presents all results available, and is described in the following sections.

After a 3 second delay, testing is repeated, until a key is pressed. However, the display can be held in the usual way, by typing cntrl/S to suspend, cntrl/Q to resume. If results are being filed, a new file will appear for each run.

2.1 A note on Matching

The problem of matching two networks is not trivial.

The worm uses its own numbering as it explores the network of transputers, and matches what it has found against the description of the user's **PROGRAM** configuration (if given). So long as the two networks match, the worm gives results using the user's numbering scheme. If, however, no match is found (which is always the case if no **PROGRAM** configuration is given), the worms own number will be given, suffixed by a *.

The master transputer is never included in a **PROGRAM** description, but is reported by the worm as **MT***.

The matching algorithm is as follows. While the worm loads the network, each time it finds a new transputer it consults the configuration specified in the **desc** fold to see whether it matches. However, if the physical network contains more transputers, or more links, than the network specified, the matching may be incomplete; consider for example the situation when an extra connection is present in the physical network, and the worm loads through it onto a transputer which does have a counterpart in the **PROGRAM** descriptor. The matching algorithm cannot know immediately that the worm has found a transputer which is described in the descriptor.

As the worm returns the network test data, together with a complete map of the link connections, results are reported using the (possibly incomplete) numbering equivalence discovered above.

However, once all the results are returned, and a complete list of link connections established, the link map is used to discover any new equivalences between the network found and the network specified. If new equivalences are indeed found, then the network test data and link map is again displayed, using the user's numbering scheme as far as possible.

2.2 Limitations of Use

If a network contains T414 revision A transputers, then only option A should be used.

If a worm comes across a C004 control link, the state of the C004 will be corrupted, and an error may result. Care must also be taken if any other peripheral is connected via a link-adaptor.

3 Options Available

The following options are available:-

3.1 A - Check T4/T8 network using internal ram

The network is explored, but no testing is performed, by a program which only requires 2k of memory on each transputer (except, of course, the master). The network may consist of T414 revs A and B, and T800 transputers. This is the only option which can explore networks containing T414's of revision A and is used both to locate such devices (which should ideally be upgraded to revision B) and to check configurations in which one or more transputer has no external memory.

3.2 B - Check T2 network using internal ram

As option A, but for T212s. This option also requires only 2k of memory on each transputer - it fits inside internal ram.

3.3 C - Check M2/T2/T4/T8 networks

Again, the network is explored, but no testing is performed. However, this program requires 6k of memory on each transputer. It is the only option which will explore networks of mixed wordlength (16 and 32-bit transputers), indeed it is the only option which is safe to use with M2 disc controller chips.

3.4 4 - Test T414 / T800 networks

A network of T414 and T800 transputers is explored, and all devices found are tested. The parameters used for testing can be varied, or certain suitable defaults used. These are outined below. Appendix A describes the testing in more detail. The worm requires 16k of memory to operate, and will only work on T414 revision B, and T800 transputers. There is no option available to test a network with both T212 and T414 transputers.

3.5 2 - Test T212 networks

A network of T212 transputers is explored, and all devices found are tested. The bottom 10k of the first T212, however, is not tested. The worm requires 10k of memory to operate. The following sections outline the different modes of testing both T2 and T4 networks.

3.6 D - Development Mode

This mode allows you to explicitly set certain testing values. These are described in Appendix A.

3.7 E - Error light testing

This mode proceeds slowly, flashing the error light on and off on each transputer as it is found. All modes, infact, toggle error, but this one holds the light on for long enough for you to see it clearly. Thus, the progress of the worm may be followed, so that if it mysteriously fails, its last known address will be obvious.

3.8 F - Full Testing

This mode tests all the memory it finds on each transputer, with pauses to test for data retention. This means that it loads up the network very slowly, for it is testing the bottom 16k (T414/T800) or 10k (T212) of memory by writing data, pausing for a second, and reading it back, for 4 different sets of data, before loading each transputer.

In the case of the T414/T800, there is an algorithm which determines the size of memory (to a resolution of 16k). The remainder of the memory, and the links, are thoroughly tested once the whole network of transputers have been loaded. Memory is not tested on the master transputer, so there may be a pause after results come back from the master transputer, before data from the rest of the network is returned.

When testing a network of T212s, however, no further memory is tested unless explicitly set using option D. Also, the memory of the first T212 in the network is not tested. However, all links in the network, including the link from the master transputer to the first T212, are tested, as above.

3.9 L - Link Test

This mode loads up the network, and then tests all the links in the network in parallel. Because the network is loaded quickly, and a lot of power is drawn due to all links and the processor working flat out in parallel, this mode is useful to 'warm up' a network if a temperature-dependent problem is suspected.

3.10 M - Memory Test

As in mode F, 16k (T414/T800) or 10k (T212) of memory is tested on each transputer before loading, and the remainder is tested at the end. However, there is no pause for data retention. This makes this mode much faster than mode F. Links are not tested.

3.11 Q - Quit

Yes, you guessed it. The program terminates without exploring the network. However, it does reset all input links of the master transputer, and also resets the subsystem.

This section covers the table of data which appears as the network is loaded. Each new entry in the table corresponds to a new transputer which has been found. The first entry is from the host transputer.

Typically, the table looks like this:-

```
      Boot    Booted by                      Analysed    Error
Id    Link    Id   Link     Type     Speed   or Reset    Line
--    ----    --   ----     ----     -----   --------    -----
MT*    0      --    --      Host`    ---      Reset       ok    ..
 0     1      MT*   2       T414b    -15       ok         ok    ..
 1     0       0    3       T800c    ---       ok         ok    ..
```

A classic problem is that a network is found on the first run of the worm, but not on subsequent runs. This indicates that the reset cable is not connected to the subsystem socket of the master transputer correctly. The network of transputers always powers up in a reset state, ready to run a program, but if the reset cable is not connected correctly, then the network cannot be reset for another run.

4.1 Id

As far as possible, the numbering scheme is as specified in the user's **PROGRAM**. However, if no **PROGRAM** descriptor fold is found, or the networks don't match exactly (see the note on matching, above), then the worm's own numbering will be given, suffixed by a *. In certain cases, such as a module mother board or B003, the worm's numbering will conveniently match the standard numbering of transputers. **MT*** is the master transputer.

4.2 Link

Links are numbered 0, 1, 2, 3. The boot link is the link on which the transputer was booted by the worm. (Note that this is not necessarily the same as the way a **PROGRAM** would be booted.)

4.3 Booted by Id, Link

These entries indicate which parent loaded the transputer. The master was loaded by no-one. So, in the example given above, link 2 of the master transputer booted transputer 0 on link 1, and link 3 of transputer 0 booted transputer 1 on link 0.

4.4 Type

T212, T414, or T800. Note that, at this stage, the worm does not distinguish a M212 from a T212 (see section 5.1). The different revisions of transputers are also distinguished. The current products are T212a, T414b, and T800c.

4.5 Speed

The speed of the part is reported. This test has not yet been calibrated for the T800, so the speed is not reported for that device.

Because the test for speed is sensitive to the speed of external ram, there is a possibility that this test will give the wrong speed when very fast, or very slow, external ram is used. In case of descrepency, the modes which use internal ram (A, B) should give the correct speed - assuming internal ram hasn't been disabled !

4.6 Processor Analysed

Every transputer has an internal flag, called ProcessorAnalysedNotReset, which indicates whether the transputer was most recently reset or analysed. The worm reads this flag, and sends the results back. The purpose of this is to check that both the reset and analyse control signals are correctly propagated through the system.

On alternate runs, the master will reset, then analyse the system. If the flag matches what was expected, then the message **`ok`** is given. Otherwise, a message **`Reset not Analysed`** or **`Analysed not Reset`** is given, and a fault in the reset/analyse chain should be suspected.

The flag on the master transputer, however, is either **`reset`** or **`analysed`**, but should not change during repeated testing.

4.7 Error Line

The error line is tested each time the worm finds a new transputer. If it seems to be working, the message **`ok`** is reported. If the line is broken, the message **`Not set`** will appear.

On the master transputer, however, it is expected that the error line is clear. If this is not so, then the message **`Not clear`** will appear. The same message may also appear the first time that the worm is run - this is quite normal, and is due to the fact that error may have been set on one of the transputers (transputers power up with the error flag in a random state).

Since the worm leaves the error flag low, option C is useful when a user wants to clear error (and, in the case of a T800, the fpu error flag) on all transputers in a network.

Having completed loading the network, further tests may be performed, according to the option selected. Sometimes, the program may appear to pause while returning results. This is because it is still testing some transputers. Results are then returned, together with a complete list of link connections.

Here is an example of some results returned (using option F), from a particularly bad network:

```
                   Memory    Link:   0          1          2          3
Id  Type    Tested    Error     Id Link    Id Link    Id Link    Id Link
--  ----    ------    -----     -------    -------    -------    -------
MT* Host      --        --        ----       ----       0  1       ----
 0  T414b   256 k       ok        oooo       MT  2      oooo       1  0
 1  T800c    64 k #80008014      0  3    Link Err 17    oooo    #80000800
```

Memory is tested in two phases. Firstly, as the network is loaded, a transputer tests a section of its neighbour's memory before loading the program. Then, once all the transputers have been loaded, each transputer tests the remainder of its own memory. If a memory error is found at the first stage, it is reported under the link which was doing the testing. At the second stage, the error is reported under **Memory Error**.

5.1 Type

M212 (which is now distinguished from the T212), T212, T414, or T800 as appropriate.

5.2 Memory Tested (Applies to options D, F, M)

If an option is selected which tests memory, it will first find out how much memory (to a resolution of 16k) is present, and then test that amount of memory. This is not true when testing a T212 network. A given memory size can be explicitly selected by using option D (see appendix A). The amount of memory which has been tested is indicated, in this case 256 kbytes for transputer 0, 64 kbytes for transputer 1. The memory of the master transputer is never tested.

Testing memory can take some time, (up to 30 seconds per megabyte), and the program will pause while this is happening.

5.3 Address of Memory Error (Applies to options D, F, M)

If an error is found, it will be reported as a hexadecimal address. This address should give the actual byte which is at fault. In the example given, there is a problem in the memory of transputer 1 at address **#80008014**.

5.4 Network Connections

Each transputer has four entries, corresponding to its four links. Each entry may be

----	indicating that the link has not been tried (applies to master transputer);
oooo	indicating that the link is unattached;
x y	indicating that the link is attached to link y of transputer x;

or an error message. Such a message indicates either a problem on the link, or on a transputer attached to that link. It does not necessarily imply a problem on the transputer being reported. Read appendix A for a background to the error messages.

The error messages are listed below. Some of the messages may refer to a particular 'stage', which is the stage of testing at which the error occurred. Certain errors tend to be revealed at particular stages, but for completeness the stages are listed in appendix B.

5.5 #8000abcd - Options D, F, M

Before a neighbouring transputer is loaded with code, part of its memory is tested using the peek and poke facility of the transputer. This error indicates that a neighbouring transputer has indeed been found, but that its memory is faulty (or does not exist) at the address given. If the value is in the range `#80000000` to `#800007FF` (T414) or `#80000FFF` (T800), then the problem lies in internal ram. Otherwise, up to a highest tested address of `#80003FFF` (16k), the problem lies in external ram. Don't forget that a transputer must have 16k of ram for this test to succeed.

In the example, the entry `#80000800` indicates that the first byte of external ram is faulty on the transputer which is connected to transputer 1 link 3. The most likely explaination in this example is that there is no external memory attached!

5.6 Token Error x - All options

When waiting for a reply on a link under test, an unknown token has been returned, at stage x. This may indicate a problem on the links (eg they are communicating at different speeds, or noise) - this usually appears as `Token Err 1`.

If an option is selected which doesn't check memory, a Token Error may indicate a memory problem on the adjacent transputer; when a transputer is first booted, it returns a copy of the program code for confirmation, which may have been corrupted. This usually occurs as `Token Err 9`.

5.7 Time Out Error x - All options

When waiting for a reply on a link under test, no reply has been received within a reasonable time, at stage x. For some reason, the neighbour died before it was properly loaded.

If an option is selected which doesn't check memory, `Time Out 9` may indicate a memory problem on the adjacent transputer; the transputer has been loaded, but does not run. This might happen, for example, if option C is used on a network which includes a transputer with no external memory.

`Time Out 18` indicates that a link, which was expecting to pass back results from further down the chain, has not received anything. This error will usually be part of a line of results which is otherwise a repeat of a previous line - the transputer has reported results, but now wishes to revise its report to indicate an error. It means that, although the worm was successfully loaded, it has subsequently died somewhere down the chain from the link indicated.

5.8 Alt Error y - All options

While waiting for a reply on link y, an unrecognised token was input on this link. This frequently occurs when two links are communicating at different speeds. It can also occur if a link is unconnected and floating, i.e. is not pulled down using an appropriate resistor. Check link y on the same transputer, as well as the link which reported this error.

5.9 Link Error x - Options D, F, L

When testing the links, corrupted data was transmitted at stage x (probably stage 17, which is when the links are tested exhaustively). This may be due to noise on the line, because of insufficient decoupling, or strong electical interference, for example. Or it might indicate a problem with the transputer link at either end, though this is rare.

More often, this error indicates a fault in the section of memory where test data is prepared. This occurs when the links but not the memory are being tested, i.e. option L. Try again with option M or F.

In the example, the entry **`Link Err 17`** indicates a problem on link 1 of transputer 2. Since option F was used in the example, this implies that data was being corrupted during transmission between transputer 2 link 1 and the attached neighbour.

5.10 Output Error x - All options except A, B

The worm has failed to output data on a link, despite the fact that it has already discovered the link to be attached to another transputer. This implies that the neighbouring transputer has died, for some reason.

5.11 ? #z

An unknown error message has been returned to this link. The hexadecimal error value, **`z`**, may or may not be useful. If an option has been used which performs no testing, then try again with e.g. option F.

To understand how a worm works, it is essential that Technical Note 24 is read. The program explores the network, using one of five worms, corresponding to options A, B, C, 2, and 4. These have been called the 'Skinny' (T2, T4), 'Mixed Network', and 'Fat' (T2, T4) worms. All five will exercise the reset, analyse and error lines, but the first three do not have any means of testing memory or links.

The fat worm, on the other hand, performs tests on memory and links using parameters supplied to it. The various options E, F, M, L, set defaults which have found to be suitable, while option D allows the user to alter the defaults himself. The sections below which describe the testing of memory and links refer to the fat worm - default values corresponding to option F are written inside (brackets).

A.1 Order of Loading

The worm explores the transputer links in order of priority 2, 3, 0, 1. I.e. from a particular transputer, it first tries to explore any network off link 2, then off link 3, etc. This contrasts to the order given in technical note 24. The order of exploration means that the numbering scheme which the worm uses matches the actual numbering of transputers when module mother boards and B003s are explored.

A.2 Size of System

The limit on the size of system which can be explored is displayed by the program. At the time of writing it is 5000 transputers.

A.3 Speed of Part

Much useful information can be deduced from the time taken at low priority to perform the loop

```
SEQ I = 0 FOR 1000
  INT x :
  SEQ
    x := -1
    x := x TIMES x
```

This will depend on the speed of the part, and type of the part. It may be used to distinguish T414 revision A from revision B, and T800 revision A from revision B. However, if internal Ram is disabled, the timings will be affected, and the speed of the part may be wrongly stated.

The following possible values for the Tick Rate assume that the program and workspace are both inside internal ram - ie option A/B:
- T212a: For 20MHz parts the value is 30 or 31;
- T414a: For 12.5 MHz parts, the value is about 2800;
- T414b and T800a: For 20 MHz parts, the value is 43 or 44;
- T800b and c: For 20 MHz parts, the value is 20.

If external memory is used (all options other than A, B), the value will be slightly larger. On the master transputer, which may be performing other work at the same time, the value given will be substantially larger, and depends on what else (displaying results, etc.) that transputer doing.

A.4 Type

By combining a test for **MemStart** (see the transputer reference manual) with the above tests for speed, the various different types of transputer can be distinguished. A 16-bit transputer is easily distinguished from a 32-bit transputer, for example by using the transputer **BCNT** instruction (see the compiler writer's guide).

A.5 Reset and Analyse

On alternate runs, the master will either reset or analyse the subsystem. This makes no difference to the function of this program, but a register exists on the transputer which is read to tell whether the transputer was reset or analysed, and hence confirm that these signals have been propagated correctly.

A.6 Error Line

As each transputer (except the master) is loaded, its error flag is set, the master reads the subsystem error line, and the error flag is then cleared before the worm proceeds to the next transputer. Thus, the Error Line should be TRUE for every transputer, except for the master, when it should be FALSE.

Note that, on the first run, the error line may be TRUE for the master transputer - error flags may have been set by a previous program, and not yet cleared. Indeed, when transputers power up, the state of the error flag is not initialised. If, however, the value is TRUE for the master transputer on subsequent runs, or FALSE for any other transputer, then a fault on the error line should be suspected.

A.7 Memory

Before loading a daughter with code, the parent tests the lowest (16 - T4, 10 - T2) kBytes of memory using peek and poke. This allows it to verify that the space which will be occupied by the worm, when it is loaded onto that transputer, is indeed safe. An error which occurs at this stage is reported in the network test data as an entry under the link which was performing peek and poke. It is possible to increase this value using option D.

When testing T212s, note that the master (T414) does not peek and poke the memory of the first T212.

Once loaded, the program uses an algorithm to determine how much memory it has. (This is not true for the T212). This algorithm has a 16 kbytes resolution, and may viotate parity. In development mode (D), the user may specify the amount of memory to be tested on each transputer, to 1k resolution, with no risk of violating parity.

The remainder of memory, up to the largest memory address found, is tested on all transputers in parallel, once all the transputers in the network have been loaded. Any error will be reported in the network test data under the memory error column.

In both cases, the memory is tested as follows:- Firstly, the address is written, as a word, to each word in the block to be tested. After a pause of (1000) milliseconds, each word is read back, and checked, in turn, being replaced by the BITNOT conjugate word. After a further pause, each word is checked, and replaced by the value **#55555555** (**#5555** for the T212). After a further pause, the words are checked and replaced by **#AAAAAAAA** (**#AAAA** for the T212). Finally, after a pause, the words are read and checked. Option M does not use any pause. A different pause may be specified using option D.

The memory of the master transputer is not tested. The program does not perform detailed tests on the memory (e.g. march tests, etc.) except as described above. If you are building your own boards, standard dynamic ram tests should be tried on a board by board basis as appropriate.

A.8 Links

Each link is tested for the existence of a neighbour by attempting to output a probe sequence, and waiting (50) milliseconds for a reply (modes A, B wait just 10 milliseconds). This default is more than ample. Communication takes place using a byte protocol, and if at any stage incorrect data is returned, the link is assumed to be bad, and no further communication takes place on that link. If a communication with incorrect protocol takes place on the link which is being probed, the error is reported as a Token Error, the entry being made against the link which was doing the probing. If some unrecognised data appears on a different link, it is reported as an Alt error.

When a daughter is loaded, it immediately returns the program for checking. If the program has been corrupted in transmission, this will show up as Token Error 9.

After all transputers in the network have been loaded, on receipt of a synchronise token, all links in the whole network are tested in both directions in parallel. A test block of data, which is 256 words long and consists of a section of the orginal program, is transmitted in both directions on each link. The input is checked, and the exchange is repeated (350 - T414/T800, 700 - T212) times on each link, independently. As far as possible, the constructs OutputOrFail.t and InputOrFail.t (see technical note 1) are used, so that the program can recover from, and report the communication of bad data. An error appears as the entry Link Error 17 against the link which discovered that its input data was corrupted.

Any links of the master transputer which are found to be connected into the rest of the network are tested in the same way as the other links in the network.

In technical note 24, the worm algorithm is described with reference to a number of different stages. These stages are also useful in telling when an error was detected. The following list of stages refers to the fat worm. Other worms may use a subset of these stages. The meaning of the tokens is described in technical note 24.

1	Send a probe sequence from a link, to determine whether there is another transputer connected.
2	Set the bottom 10k/16k of memory of the neighbour to word addresses. Pass back a `GreenLight.t` token to the master. Pause for one second.
3 - 6	Read back and check data, writing a new word as we go. After each stage, pass back a `GreenLight.t` token, and pause for one second.
9	Having determined that there is an unbooted neighbour with at least 10k/16k of good data, boot that transputer with a copy of the worm program. The neighbour will return the program for checking.
11	Send down a set of initalisation data to the newly booted daughter. The daughter will return a set of `loadingData`. Pass this back to parent, and synchronise with the master.
12	The neighbour, or someone further down the chain, is now testing its links. Pass back `GreenLight.t` tokens. Do not time-out the link at this stage.
12	Also be prepared to pass back `loadingData`, and forward a `Synchronise.t` token.
12	When the neighbour sends back `ReturnControl.t` and the number of transputers found so far, it is assumed that the branch off that link has been completey explored. Try another link.
14	Once all links have been explored, return control to parent.
15	Synchronise the whole network, prior to final testing.
17	Test all links and memory in parallel.
18	Send results of testing, `networkData`, back to parent. Forward `networkData` from each link in turn, reading from a link until `NoMoreData.t` is encountered. When all links have been read, return `NoMoreData.t` to parent.

The dots `..` which appear while the worm is loading indicate the return of the token `GreenLight.t`.

INMOS Limited
1000 Aztec West
Almondsbury
Bristol BS12 4SQ
UK
Telephone (0454) 616616
Telex 444723

INMOS Corporation
PO Box 16000
Colorado Springs
CO 80935
USA
Telephone (303) 630 4000
TWX 910 920 4904

INMOS GmbH
Danziger Strasse 2
8057 Eching
Munich
West Germany
Telephone (089) 319 10 28
Telex 522645

INMOS Japan K.K.
4th Floor No 1 Kowa Bldg
11 - 41 Akasaka 1-chome
Minato-ku
Tokyo 107
Japan
Telephone 03-505-2840
Telex J29507 TEI JN
Fax 03-505 2844

INMOS SARL
Immeuble Monaco
7 rue Le Corbusier
SILIC 219
94518 Rungis Cedex
France
Telephone (1) 46.87.22.01
Telex 201222

February 1988
72 TRN 146 00