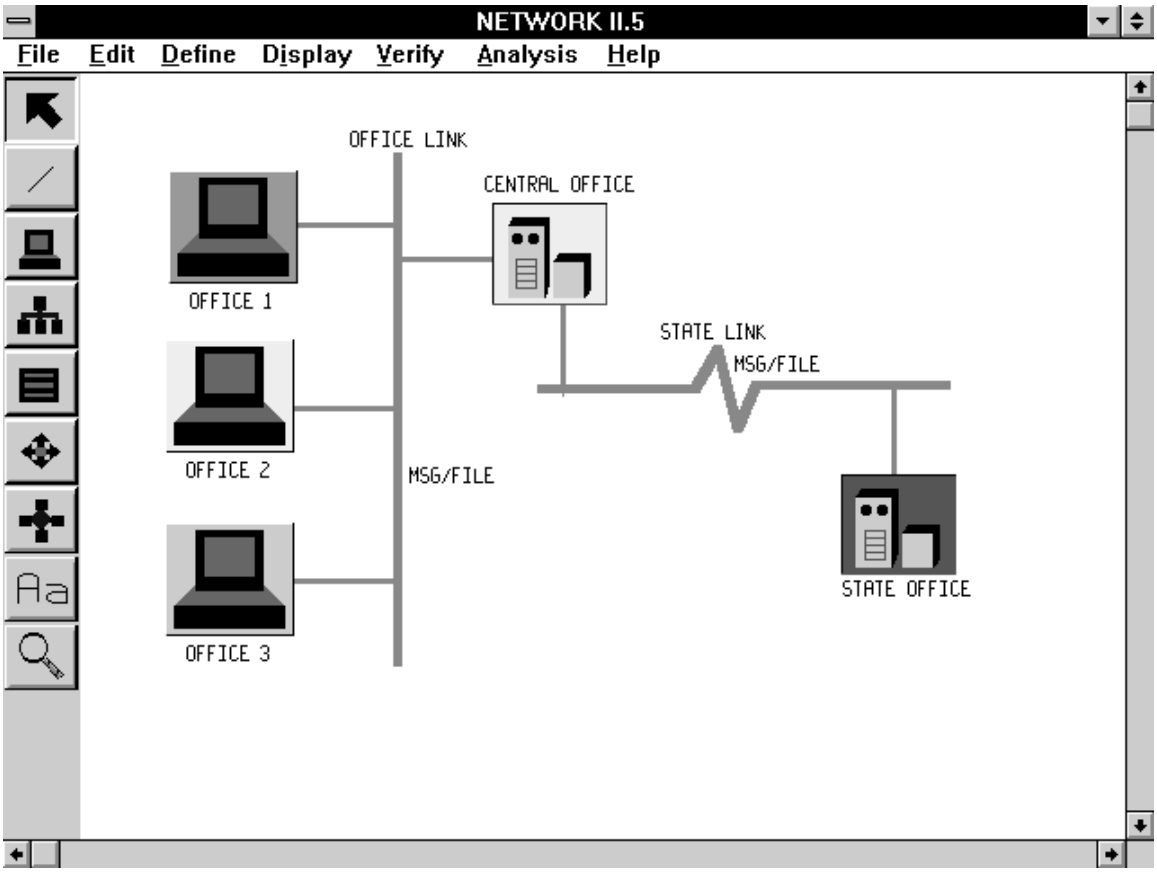
1. INTRODUCTION TO NETWORK II.5

1.1 NETWORK II.5 OBJECTIVES

NETWORK II.5 is a design tool which takes a computer system description you specify and provides measures of hardware utilization, software execution, message delivery times, response times and contention. It is intended for the person who designs or specifies computer systems and computer networks. NETWORK II.5 is used both to evaluate the ability of a proposed system configuration to meet the required workload and to evaluate competing designs. NETWORK II.5 is designed to model a wide variety of computer architectures, from a single processor to a complex system of processors and storage devices connected in a network. It is extremely flexible, allowing the portions of a computer system of special interest to be modeled at a detailed level while the rest of the system is modeled at a coarser level. Most importantly, NETWORK II.5 is a tool and not a computer language. It has an interactive graphical interface and can be used after a short period of training.



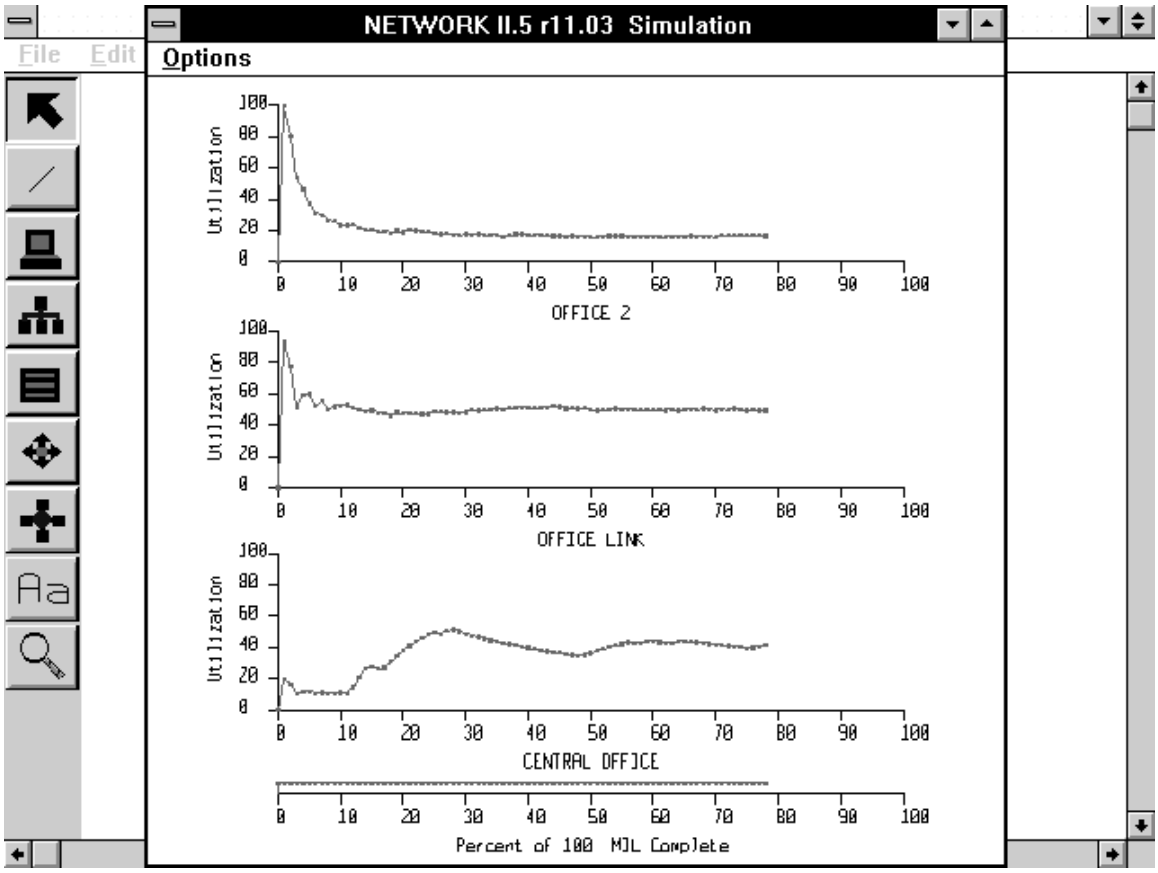
Building a Model

One of the main considerations in the design of a computer system is the effect of contention between devices on total system operation. This effect is not readily determined by the use of analytical models. Because NETWORK II.5 actually models the interactions between all devices in the system, the effects of contention are identified in NETWORK's report of total system performance.

1.2 OVERVIEW OF THE NETWORK II.5 SYSTEM

The NETWORK II.5 package provides three main functions; system description, system simulation and simulation analysis. System description is handled by graphical layout of the hardware elements and the use of graphics based forms to describe the component attributes. System simulation is simply a matter of invoking the simulation program for the system description that you have built. NETWORK II.5 does not require any programming, so there are never any programming delays. You can look at any of the reports as the simulation runs and/or follow an event trace that tells you exactly what is going on inside the simulation. NETWORK II.5 contains a rich repertoire of simulation

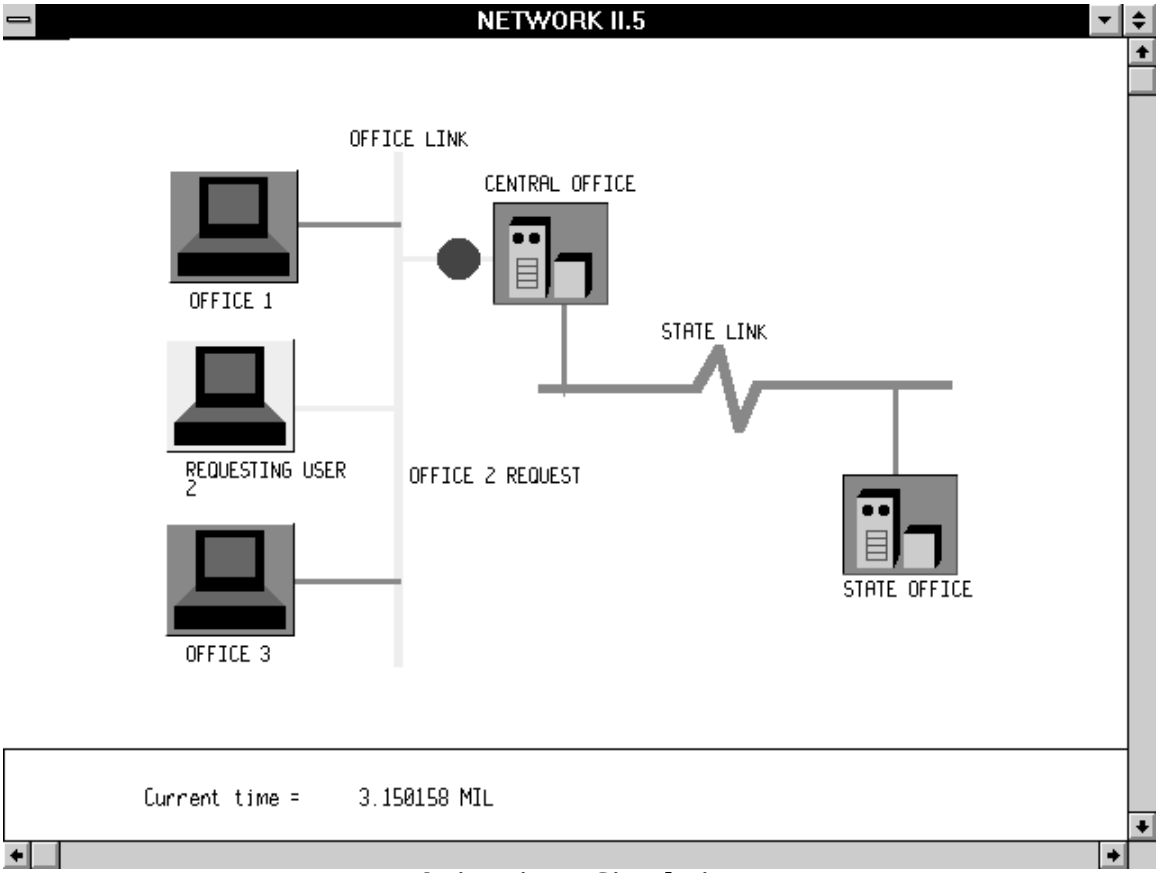
analysis tools. First and foremost is animation. You can actually see your system in operation, making many problems obvious by inspection. You can plot device status and utilization as a function of time. The plots are based on a database constructed during the simulation run. You may access that database directly for custom reports. Finally, there is a comprehensive set of tabular reports written at the end of the simulation. You can view this file with our built in browser, or print it out as documentation.



Running a Simulation

The computer system to be simulated is described in a data structure consisting of Processing Elements, Gateways, Transfer Devices, LANs, Storage Devices, Modules, and Files. Each of these building blocks has a series of attributes whose values are supplied by the user. For example, each Processing Element has a Basic Cycle Time attribute to represent its clock speed. The powerful NETWORK II.5 icon-oriented graphic user interface allows a user to build and maintain the requisite data structure by means of simple but powerful graphical operations. For example, a new Processing Element can be brought into existence by clicking on the Processing Element icon on the palette and dragging the icon to the display.

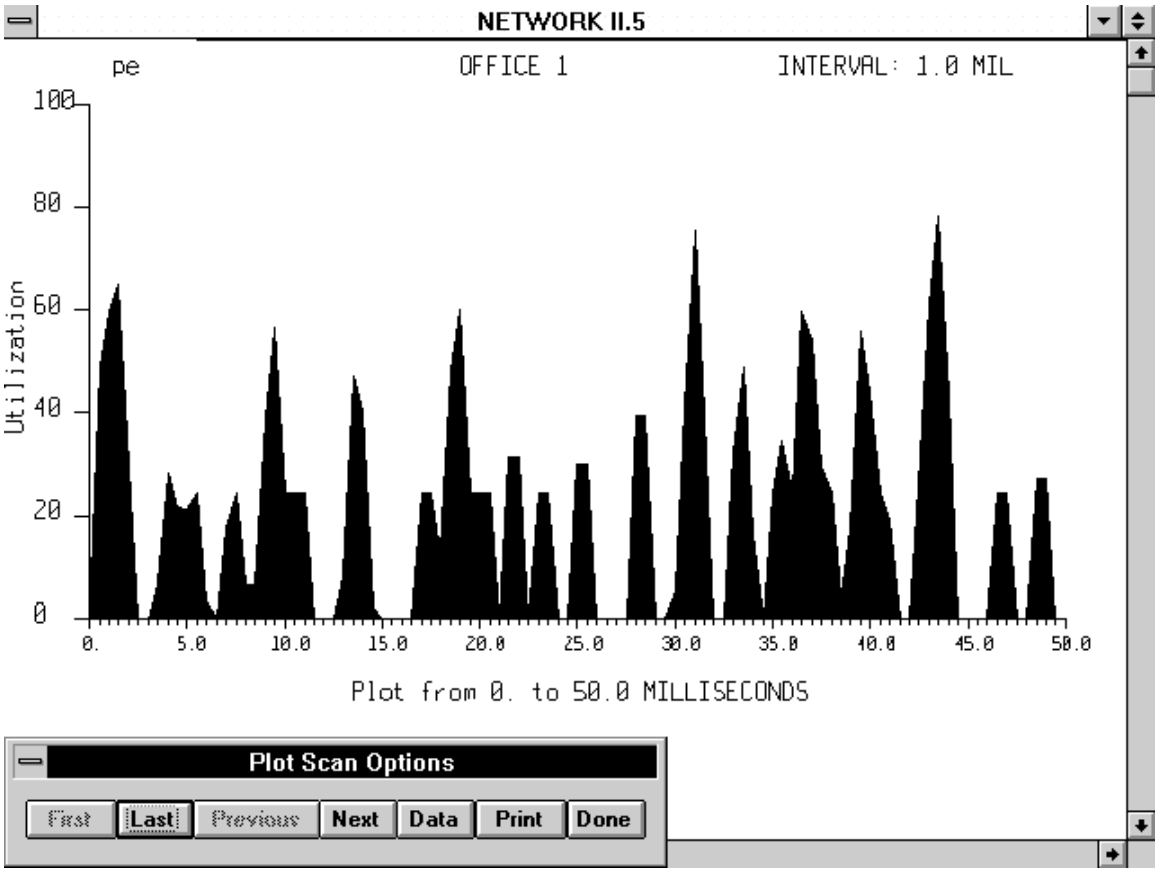
NETWORK II.5 stores the system description created graphically in a text network description data file which describes the hardware and software of the system to be simulated. This network description file is a concise English description of the system to be simulated that can serve as the definitive documentation of the system simulated. The format of this file is documented under separate cover and may be used as an interface to other databases and programs. After acquiring the runtime control parameters (such as simulation length, devices to graph) from you, NETWORK II.5 builds and executes the simulation. You can optionally monitor the simulation as it progresses through the use of trace and snapshot reports. Running NETWORK II.5 produces periodic and end-of-simulation reports. An event trace data file may also be requested. The event trace data file contains hardware and Semaphore status information that can be post processed by NETWORK II.5 and by custom report routines written by you.



Animating a Simulation

Animated displays of NETWORK II.5 simulations provide a view of the modeled computer system in operation. It is easy to define the location, color and icon of every device in a NETWORK II.5 simulation. Interfaces to network management software packages, such as HP OpenView, Netview for AIX, Digital PolyCenter and Cabletron SPECTRUM, allow users to import topology information automatically, if desired. The

animation may be advanced in single steps or continuously. Animation and event trace messages describing simulation activity can be displayed concurrently.



Plotting a Simulation

NETWORK II.5 can produce a graphical time line status report. This is a post-processed report that shows, for each Module, message and hardware device selected, the times at which the device was busy. In addition, the status and/or count of each of the Semaphores in the simulation is plotted as a function of time. It can show whenever any of these items exceeded a user defined threshold of activity. NETWORK II.5 can also produce detailed graphical utilization reports that will display the utilization of devices in a simulation as a function of time. The scale of these diagrams is adjustable. The diagram may cover the entire simulation (on a single screen) or you can examine a smaller period of particular interest in greater detail to study conflicts or dependencies, without rerunning the simulation.

1.3 NETWORK II.5 DATA STRUCTURES

NETWORK II.5 data structures fall into two primary groups: hardware and software. There are other data structures (Global Flags and Variables, Statistical Distribution Functions, etc.) that are used by these two groups.

Hardware devices are specified in NETWORK II.5 as one of three basic types: Processing Elements, Transfer Devices, or Storage Devices. Gateways are a special case of a Processing Element and LANs are a special case of a Transfer Device. Complicated real world devices may be simulated by a combination of these NETWORK II.5 building blocks.

Processing Elements are characterized by their Instruction set and cycle time. Transfer Devices are characterized by their data transfer rate, data transfer protocol, and connections. Storage Devices are characterized by capacity, access time, and access method. By using generalized building blocks, the user is not restricted to predefined hardware types. For example, a Transfer Device building block is used to model anything that transfers data from one device to another. This could represent a bus, a satellite communications link, a LAN, a WAN, etc.

There is no arbitrary limit to the number of hardware device building blocks that may be used in a NETWORK II.5 simulation. There is also no arbitrary limit to the number of Processing Elements and Storage Devices that can be connected to a single Transfer Device.

The software of the simulated system is presented to NETWORK II.5 in the form of software Modules. Each Module contains a list of the Processing Elements on which it may execute, a description of when it may run, what it is to do when running, and which other Modules (if any) to start upon completion. A Module may be assigned to a specific Processing Element, it may be allowed to run on any Processing Element on a list, required to run on every Processing Element in a list or it may be allowed to run on any Processing Element in the system. You can specify a limit to the number of copies of the same Module that may execute concurrently.

NETWORK II.5 provides you with the ability to specify many different kinds of conditions that must be met before a Module can start executing. These Module preconditions include time-, hardware-, message-, and Semaphore-based conditions. Time-based preconditions include starting a Module at a specific time during the simulation and/or automatically iterating at a rate specified by the user. Hardware-based preconditions include waiting until a particular set of Processing Element(s), Transfer Device(s), and/or Storage Device(s) are available at the same time. Message- and Semaphore-based preconditions include waiting until a given message or set of messages

are received or a given Semaphore is set. Semaphores can also be used to interrupt or cancel the execution of Modules. Messages and Semaphores have count values which can also be used in specifying preconditions.

1.4 NETWORK II.5 REPORTS

NETWORK II.5 offers a variety of different reports on system activity. These reports are:

- Graphical Reports
  - Automatic Hardware Layout
  - Software Module Diagram
  - Status Plot
  - Utilization Plot
  - Verify Report

- Tabular Reports
  - Processing Element Statistics
  - Instruction Execution
  - Transfer Device Statistics
  - Storage Device Statistics
  - Module Summary
  - Semaphore Report
  - Message Statistics
  - Received Message
  - Message Delivery
  - Snapshot Report

- Interactive Simulation Runtime
  - Narrative Trace
  - Snapshot Report

The Automatic Hardware Layout feature shows each device in the system and how it is connected to other devices. There are three layout algorithms to choose from. The Software Module diagram shows, for each Module, its preconditions, host Processing Element, instructions, message, Semaphore and File outputs, and successor Modules. The tabular reports show statistics on the operation of each hardware device, Module or Semaphore in the simulation. Average, maximum, minimum and standard deviation values are reported for execution, delay and queue sizes, as appropriate. The Narrative Trace produces an (optional) step-by-step trace of the execution of the simulation, showing, for example, when an Instruction begins execution, completes execution, is interrupted, and so on.

The interactive reports may be printed at your terminal while the simulation is progressing. The Snapshot Report shows the current status of all elements (both

hardware and software) in the simulation. The Snapshot Report may be printed periodically or interspersed with the Narrative Trace.

1.5 USER'S MANUAL ORGANIZATION

The organization of this manual is based on the parts of the NETWORK II.5 package. The parts are presented in the order in which they would be first encountered during a typical NETWORK II.5 session.

Chapter 1 - An overview and brief introduction to the NETWORK II.5 system.

Chapter 2 - How NETWORK II.5 interprets the world to be modeled.

Chapter 3 - Using the graphic user interface to develop and verify a system description.

Chapter 4 - How to run a simulation.

Chapter 5 - The reports produced by a simulation.

Chapter 6 - Using animation.

Chapter 7 - Plotting simulation results.

Chapter 8 - A complete example of a NETWORK II.5 simulation, from problem statement to analysis of simulation results.

Chapter 9 - Several methods and guidelines to debug a system description.

Chapter 10 - Using TRAFLINK to import topology and /or to drive a simulation with a LAN analyzer trace file.

Chapter 11 - How to convert a Simulation Plot File between binary and text format.

Appendices describing implementation-specific procedures on available hardware platforms are included. These currently include:

MS Windows (Windows 3.1, Windows 95 and Windows NT)
OS/2 Warp 3.0
Unix (HP 9000/700, SUN SPARC, IBM RISC/6000, Silicon Graphics)

NETWORK II.5 is written in a higher level language (SIMSCRIPT II.5), and is very portable. Implementations for other platforms will be added as required.

The final appendix in this manual describes using the SimGraphics II editor for the creation and modification of icons which can be added to your system description.

2. MODELING WITH NETWORK II.5

2.1 INTRODUCTION

In this chapter, all of the elements which can be used to describe the hardware and software of a computer system in NETWORK II.5 are defined. We are not concerned with the exact form of input preparation, which is described in Chapter 3.

2.1.1 Overview

Hardware devices are specified in NETWORK II.5 by using building blocks based on the functions of the device being modelled. There are three functions performed by the hardware elements in a computer system. These are:

- **Process data** – performed by a Processing Element building block
- **Transfer data** – performed by a Transfer Device building block
- **Store data** – performed by a Storage Device building block.

These building blocks are powerful enough to model any device that performs the given function. NETWORK II.5 uses an event driven simulation. In an event driven simulation, you only need to know the timing aspects of an operation, not its implementation. For example, to model a communications link, NETWORK II.5 only needs to know when to send the data and how long it takes to move the data from source to destination. Whether the technology involved is a satellite channel or a fiber optics link is immaterial.

Actual devices are modelled by defining one or more building blocks to perform the function of the device being modelled. Most devices can be modelled by a single building block. As an example, when modeling a personal computer connected to a local area network, the personal computer can usually be adequately modelled by a single Processing Element building block. However, if the internal operation of the personal computer were of interest, it could be modelled as a collection of one Processing Element (representing the microprocessor), one Transfer Device (modeling the internal bus), and one Storage Device (modeling the disk drive).

The software components of a system are characterized in NETWORK II.5 as modules, instruction mixes, macro instructions and files. Modules direct a NETWORK II.5 simulation by using hardware components as required to process, read, write, and send information through a network. Instruction mixes and macro instructions support modules by making random and/or repetitive simulation tasks easier to control and maintain.

2.1.2 Processing Element Overview

A Processing Element (abbreviated PE) is used to model anything in your simulation that executes instructions or makes decisions. A PE might be used to simulate a bus controller, a display, a sensor, or an entire arithmetic logic unit. A Processing Element is characterized by its Basic Cycle Time, Instruction Repertoire, Message List Size, I/O Setup Time, Time Slice, Interrupt Overhead and Input Controller. The Instructions listed in the PE's Instruction Repertoire define what a Processing Element can do.

Each instruction in the Instruction Repertoire is defined by using a generalized instruction building block. The defined instruction may be as detailed as desired. There are four kinds of generalized instruction building blocks; they are:

1. Processing Instructions
2. Read/Write Instructions
3. Message Instructions
4. Semaphore Instructions

The Instruction Repertoire of a Processing Element in NETWORK II.5 is not meant to be the literal instruction set of the machine being simulated (although it may be). Instead, it is more likely to consist of very high level pseudo-instructions. A database query might thus be listed as a single instruction, even though it consists of hundreds of machine-level instructions. This level of description may be used because the actual code of the algorithm is undecided, or because timing considerations for an entire system are being studied, and simulation at the machine instruction level would be unnecessarily detailed.

NETWORK II.5 is designed to simulate the effect of an instruction on the system being simulated, as opposed to computing a numerical result. Therefore, macro instructions (such as database query) can be mixed with assembler instructions in a Processing Element's Instruction Repertoire. This allows a simulation with items of special interest

modelled down to the machine instruction level, while the remainder of the system is modelled at a coarser level.

Gateways are a special case of the PE building block. A Gateway is a modeling convenience which allows you to easily model a device whose function is to transfer data from one Transfer Device to another. The name Gateway is used in a generic sense. The devices a NETWORK II.5 Gateway connects may be of the same protocol or different protocols.

Additional information about the operation of the PE building block will be covered in the Processing Element Details section, which appears later in this chapter.

2.1.3 Transfer Device Overview

Transfer Devices (TDs) are the links connecting Processing Elements and Storage Devices. They are used to move data between two or more Processing Elements or between a Processing Element and a Storage Device. They can connect as many of these devices as desired. Each Transfer Device has a user-defined specification giving the transfer speed, transfer overhead and protocol definition.

Data is moved between Processing Elements over a Transfer Device as the result of a Message Instruction, and between a Processing Element and a Storage Device as the result of a Read/Write Instruction.

Transfer Devices automatically organize every file or message transmitted into words and blocks. You specify word and block transfer and overhead times for the TD. Then, at run time, NETWORK II.5 computes the amount of time to send the actual amount of data a PE is sending or requesting. The TD protocol you select defines the method to resolve contention between Processing Elements for a Transfer Device. The TD protocol attribute may be set to model FIFO (First Come, First Served), Collision, Token Ring, Token Bus, FDDI, Priority, Aloha and others.

A LAN (Local Area Network) is a special case of a TD. A LAN is a standard TD building block with its parameters predefined to match industry standards. The advantage of the LAN building block is that the parameters for industry standard LANs are built in. The built in LANs types include Ethernet, 10BASE2, 10BASET, 4 Mb Token Ring, 16 Mb Token Ring, FDDI and others.

Additional information about the operation of the TD building block will be covered in the Transfer Device Details section, which appears later in this chapter.

2.1.4 Storage Device Overview

Storage Devices contain both user-named files and unstructured storage, called General Storage. Every Storage Device has a capacity measured in bits.

When a Read Instruction references a Storage Device, it checks to see if the requested file is available. If it is not, a warning message is printed and the simulation continues as if the file were available. If it is available, the file is read into the PE. You have the option of reading in the entire file or a specified portion. You may also specify if the Read Instruction will destroy the entire user-named file just read, or decrease the file size by the amount read, or leave the file size unchanged.

When a Write Instruction attempts to put a file in a Storage Device, it checks to see if there is enough space available. If there is, it accepts the file. If adequate space is not available, a warning message is issued to the user and the simulation continues as if there is enough room to transfer the entire file. Write Instruction options you may choose from are replacing an entire user-named file (if one by the same name existed before the write), or increasing the file size by the number of bits written, or leaving the file size unchanged.

For simulations where the file structure has yet to be determined or is not significant to the simulation, files can be read from or written to a Storage Device's General Storage. General Storage keeps track of the number of bits stored, but not individual file names. Commonly, for simplicity, some files are written to a Storage Device's General Storage while more significant ones are explicitly named and stored.

A Storage Device may be defined to serve more than one PE simultaneously. The user specifies that a Storage Device has *n* ports where *n* is the number of simultaneous accesses that can be handled by the Storage Device.

Additional information about the operation of the SD building block will be covered in the Storage Device Details section, which appears later in this chapter.

2.1.5 Module Overview

A Module is the specification of a task to be performed by a Processing Element. The Module description consists of four parts:

1. Scheduling conditions
2. Host Processing Element options
3. A list of instructions to execute
4. A list of Module(s) to execute when this Module completes

Each of these parts maps into a stage that every Module goes through in its "life." The stages in a Module's life are:

1. Checking preconditions
2. Requesting a host PE
3. Executing instructions
4. Choosing successors

When a Module checks its preconditions, it is really checking to see if the user-defined scheduling criteria are met. When they are, the Module then queues up for a host PE (i.e., the processor on which it will run). After obtaining a host PE, a Module begins executing by issuing instruction names from its Instruction List to its host Processing Element, one at a time. The PE executes the instruction it was given and then asks its host Module for another. After a Module has issued the last instruction in its Instruction List, it frees its host PE. The Module then chooses its successor Modules (if any were specified).

Additional information about the operation of the Module building block will be covered in the Module Details section, which appears later in this chapter.

2.1.6 Overview Summary

This overview section covered the major NETWORK II.5 building blocks at a fairly high level. It is now time to present the details. A description of a computer system may be developed beginning with either the hardware or the software. We will start with the hardware because it is, perhaps, the easier task. The software description will then follow.

2.2 NETWORK HARDWARE COMPONENTS

Hardware components consist of Processing Elements, Transfer Devices, Storage Devices, Routes, Echo PE Lists, LANs, Gateways, and Destination Mixes. Each is described in turn below.

2.2.1 Processing Element Details

As mentioned in the Processing Element Overview section, a Processing Element (abbreviated PE) is used to model a hardware device which is not merely a data source or sink. A PE might be used to simulate a bus controller, a display, a sensor, or an entire arithmetic logic unit. A Processing Element is characterized by its Quantity, Instruction Repertoire, Message List Size, I/O Setup Time, Basic Cycle Time, Time Slice, Interrupt Overhead and Input Controller. This section will cover the details of these characterizations.

The Quantity of a PE will determine if a single PE is defined (Quantity = 1) or if a number of identical PEs are defined (Quantity is greater than 1). If you select a number greater than 1 for a PE Quantity, NETWORK II.5 will recognize that number of separate, but identical PEs. NETWORK will allow you to clone a PE, but once clones are created, they are unrelated to the parent PE. If you edit a PE in NETWORK with a Quantity greater than 1, you are essentially editing the entire group at one time. This cannot be done if a group of cloned PEs is created. Each PE must be edited individually if all are to have the same changes.

The Basic Cycle Time of a PE is the basic time unit on which all of the processing instructions of a PE are built; its purpose is to give the user one "knob" which can be turned to speed up or slow down a PE for sensitivity studies (for example). Although the Basic Cycle Time may be the actual clock cycle time of the simulated PE, it need not be. As an example, if the minimum time to execute an instruction is 50 microseconds (mic) and all other instruction execution times are multiples of 50 mic, 50 mic would be an obvious choice for the Basic Cycle Time.

Every PE has an Interrupt Overhead specification. Switching from one Module to another due to an interrupt will always invoke the delay given by this specification. If you do not want to add any interrupt overhead, disable this feature by leaving Interrupt Overhead in the NR (No Response) state. An Interrupt Overhead of zero is not allowed because it adds to simulation execution time without an improvement in simulation accuracy.

Every PE has a Time Slice attribute. The Time Slice attribute specifies automatic swapping between equal priority Modules during their execution. Setting it to NR will disable this feature. Time Slice only applies to Modules whose Interruptibility Flag is set to YES. This means you can exclude certain Modules from being affected by Time Slicing.

The Time Slice feature works as follows. When a Module starts executing its first instruction, a timer is started which will expire in the amount of time specified by Time Slice. If the Module completes before the timer expires, the timer is canceled. If the timer expires before the Module completes, the Module is interrupted and put at the back of the queue for its priority class. Then the PE selects the next Module to execute according to standard rules. Normally, it will select another Module of equal priority. If there are no other Modules of equal or higher priority to execute, the PE will restart the interrupted Module. Interrupt Overhead applies to Modules interrupted by a Time Slice. Whenever an interrupted Module is restarted, a new timer is started to expire in one full Time Slice.

Every PE may have a Message List Size. This attribute defines the number of bits available for storing received messages. If Lose Overflow Messages is set to **YES**, this will cause any messages which overflow the PE's Received Message List to be lost. If set to **NO**, a warning flag is set, but the message is stored regardless.

Every PE has an I/O Setup Time, which applies to all Read, Write, and Message instructions. This attribute is used to add an additional amount of processing time before every Transfer Device access. This delay might represent various operations including checking Transfer Device status and breaking a message into packets.

Instructions form the link between hardware and software. The correspondence is established through a user-defined name. Each Module includes an Instruction List containing a sequence of instruction names. When the Module is "executed" on a particular PE, it issues the instructions it has in its Instruction List by name to the PE, in the order in which they appear. The PE is then responsible for examining its Instruction Repertoire to find an instruction definition by that name. The PE's instruction definition contains all of the information describing the instruction's type and characteristics. Thus, a Module executing an instruction on one PE (for example, a general-purpose processor) might require a long execution time whereas another (special-purpose) PE may execute the same instruction very quickly.

NETWORK II.5 supports a priority interrupt scheme. Modules have a priority associated with them. If a Module having a higher priority than a PE's current Module becomes available to execute, a PE will interrupt its current Module to work on the higher priority Module. You may disable priority-based interrupts of a Module by setting its Interruptibility Flag to **NO**.

When a Module executing a Processing Instruction is interrupted, the remaining execution time of the instruction is recorded. When the instruction execution is resumed, it will occupy the PE for that remaining length of time (unless interrupted again!). Message and Read/Write Instructions, if interrupted, are either resumed or restarted from their beginning based on a user-defined attribute of the instruction, called the Resume Flag. If a Module is interrupted during a Semaphore Instruction, the instruction will be completed and the Module will resume with the next instruction in its Instruction List.

2.2.1.1 Input Controller

Every PE may have an Input Controller. If a PE has an Input Controller, then it may accept messages and execute instructions simultaneously. Otherwise the PE must be in a non-busy state to accept messages. Messages may be received in parallel if more than one Transfer Device is sending a message to this PE. After a message has been received, it is put on the PE's Received Message List, where it may trigger the execution of a Module. A PE with an Input Controller which is receiving a message and is not executing a Module will be listed as idle.

A PE without an Input Controller must work the entire amount of time it takes to receive the message from the Transfer Device. The PE cannot execute a Module in parallel with the receipt of a message. A PE without an Input Controller that is executing a Module when another PE attempts to send a message to it over a Transfer Device will block the incoming message, thereby blocking the sending PE and the connecting Transfer Device. The sending PE and Transfer Device will remain blocked until the receiving PE becomes idle, having finished executing its scheduled Modules. A PE without an Input Controller will be listed as busy while receiving a message.

2.2.1.2 Keep Blocks Separate

The Keep Blocks Separate flag of a Processing Element is used to control the way in which messages are received and assembled by a PE. When a PE with Keep Blocks Separate = **NO** receives a message that arrives in blocks, the blocks will be combined as they are received to form a single message. If Keep Blocks Separate = **YES**, the blocks

will not be combined, but remain available as distinct blocks in the received message list of the PE. Keep Blocks Separate = **YES** will guarantee that a Module with a Wait For Block Message Status Requirement will respond to each individual block that is received for a message.

2.2.1.3 Queue Flag

The Queue Flag of a Processing Element is used to control the saving of messages in the Received Message List. If Queue Flag = **NO**, whenever a message arrives at the PE, if a message of the same name already is stored in the Received Message List, the new message is discarded. If Queue Flag = **Yes**, duplicate messages will be stored in the Received Message List if room is available. Message Instructions also have a Queue Flag to control saving messages in a destination PE, but the Queue Flag of the receiving PE takes precedence over the Queue Flag of the Message Instruction that originates the message. Thus, if a Message Instruction with Queue Flag = **YES** sends a copy of a message currently queued in a PE with Queue Flag = **NO**, the incoming message will not be stored in the Received Message List.

2.2.1.4 Processing Instructions

Processing Instructions are used to model instructions which execute on a Processing Element without any external references (such as reading or writing). They are commonly used to model arithmetic and logic instructions.

Every Processing Instruction has a name and a Number of Cycles. Names in NETWORK II.5 can be (almost) any combination of characters up to 40 characters in length. Embedded spaces are allowed; however, certain special characters are not allowed on some systems. The following characters are not permitted in names entered in NETWORK: “/”, “\”, “;”. If you attempt to enter a name using any of these characters, or, if for any reason a name you entered is considered invalid, you will be prompted to enter a new name.

The execution time for Processing Instructions is expressed in terms of a multiplier of the Basic Cycle Time. Therefore, a change in a PE's Basic Cycle Time automatically changes the execution time of all Processing Instructions in the Instruction Repertoire of that PE. The amount of PE time it takes to execute a Processing Instruction is simply the Number of Cycles of the instruction multiplied by the Basic Cycle Time of this PE.

The Number of Cycles for a Processing Instruction is always an integer. When a statistical distribution is used for a Number of Cycles calculation, the result is always rounded to an integer. Processing Instructions may use one of the Linear Distributions to compute a Number of Cycles based on the size of a received message, the size of a file that has been read, or a Semaphore count. See the section on Statistical Distributions later in this chapter for further details on Linear Distributions.

2.2.1.5 Read/Write Instructions

Read/Write Instructions move a file between a Storage Device and a Processing Element. Files have a name and a size, and reside on one or more Storage Device(s). Trying to read a file from a Storage Device where it does not reside, or trying to write a file to a Storage Device that does not have sufficient space available, produces a warning in NETWORK II.5.

A Processing Element will execute a Read/Write Instruction according to the following sequence of events:

1. Request a Transfer Device.
2. Request the Storage Device To Access.
3. Request the File Accessed.
4. Work the amount of time it takes to read/write the Number Of Bits To Transmit taking into account the Storage Device time and the Transfer Device time.
5. Relinquish the File.
6. Relinquish the Storage Device.
7. Relinquish the Transfer Device.

The time it takes to read or write a file depends on the speed of the Transfer Device and Storage Device used. Specifically, it will take the greater of either the time to transfer the file over the Transfer Device or to read/write the file from/to the Storage Device. Read Instructions which use a Transfer Device that is slower than the Storage Device used will add the time for the Storage Device to access the first word of the requested file to the Transfer Device time. A Read/Write Instruction may be delayed by contention for a Transfer Device, contention for a Storage Device, or contention for a file. It is these delays, plus the potential for interruption, that make the exact execution time of a Read/Write Instruction impossible to predict in advance (although a lower bound based on the transfer time/storage time does exist).

During file transfer, the PE will be listed as busy executing the Read or Write Instruction, even though it is not officially executing any "cycles." To simulate a PE which concurrently executes other instructions during a file transfer, a separate PE building block must be added to the system to handle the I/O operations for this PE. Because NETWORK II.5 uses one building block per function, it is quite reasonable to model one "real" device as a number of these building blocks when the "real" device performs more than one function simultaneously.

The TD and SD used by a Read/Write instruction will be listed as utilized for as long as the PE executing this instruction holds them. So, if a PE acquires a TD and then holds it while queueing for a SD, the TD will be listed as utilized during that time. Also, if a TD is much faster than a SD, the TD will still be listed as utilized for the entire time of the file transfer. The TD is counted as utilized because it is held by the PE for the duration of the entire transfer. The TD's ability to transfer bits faster than they are presented does not override the fact that the TD was held by the PE for the full length of the file transfer.

Sometimes it is not necessary to deal with specific files when describing input/output operations. In this case, the default "file" of **`General Storage`** may be specified for both read and write operations. General Storage is the name given to that part of memory with no specific file structure. General Storage is unique in that more than one user may simultaneously use this file. General Storage must be created like any other file, and will be removed from a Storage Device should its size ever fall to zero bits. The size of General Storage is monitored and reported separately.

Every Read/Write Instruction has:

- A name
- A Storage Device To Access
- A File Accessed
- A File Count
- A Number Of Bits To Read/Write
- File Modification Options
- A Resume Flag
- An Allowed Transfer Device List

The number of bits an instruction is to read or write may be an integer or a reference to a statistical distribution. The value given will be used to compute the length of time needed to access the file. For a Read Instruction, a NR ("No Response") to this attribute tells NETWORK II.5 to read all the bits the File Accessed contains. If an instruction tries to read more bits than a file contains or write more bits than the Storage Device To

Access has available, a runtime warning message will be issued. Execution continues as if the file was available or the Storage Device had sufficient space. This approach is taken because canceling the transfer would produce artificially low utilization reports. The Linear Distributions can be used to specify the Number Of Bits To Read/Write based on the size of a received message, another file or the count of a Semaphore, message, or file.

Read Instructions have three file modification options: Erase File, Decrement File, and Do Not Modify File. The Erase File option will release all of the storage occupied by a file upon completion of the instruction. Decrement File will only destroy the number of bits read from a file upon completion of the instruction. When Do Not Modify File is selected, the file size will remain unchanged at the end of the execution of the instruction.

Write Instructions have three file modification options: Append to file, Replace File, and Update File. Append to File adds the number of bits to write to the size of the file. If an instruction writes to a file that does not exist, a file will be dynamically created on the Storage Device To Access and will contain the number of bits to write at the completion of the instruction. The Replace File option replaces an existing file with the new data and the file will have a size equal to the number of bits written when this instruction completes. Update File does not affect a file it is greater than or equal to the number of bits written. In case you write more bits as an "update" than the file contains, it will grow in size to be the number of bits written by the "update" instruction.

The Resume Flag applies to both Read and Write instructions in instances where an interruption of the instruction occurs. If the Resume Flag = **NO**, this instruction will restart from the beginning if interrupted. If the Resume Flag = **YES**, this instruction will only read or write the number of bits not yet transmitted when resumed.

The Storage Device To Access is the name of the Storage Device that contains the File Accessed. Wild cards are allowed for the Storage Device To Access (see Section Wild Cards for a description of wild cards). For a Read Instruction, a wild card in the Storage Device To Access specification will cause a search for a Storage Device whose name matches the wild card and contains the File Accessed. The order of search is the order of appearance in the Connection List of the Transfer Device chosen. Failure to find a matching Storage Device name results in a runtime error and the instruction is aborted.

For a Write Instruction, a wild card in the Storage Device To Access specification will cause the file to be written to a Storage Device whose name matches the wild card and has enough room to store the entire file written. The order of search is the order of appearance in the Connection List of the Transfer Device chosen. If none of the Storage

Devices has sufficient room, the transfer is blocked until sufficient room becomes available.

The File Accessed is the name of the file this instruction will manipulate. Reading a file that does not exist produces a runtime error. The Read Instruction will then continue to execute as if the file is present. Writing to a file that does not exist will cause a file by this name to be automatically created. Files may be predefined in the SD (Storage Device) form when using NETWORK, so that they exist at the start of a simulation.

Files are distinguished not only by there names, but also by an optional File Count. A File Count can either be an integer value or a statistical distribution. When a Read/Write Instruction attempts to access a file, it will search for a file with the appropriate name. If the Read/Write Instruction also has a specified File Count, then the instruction will search for a file with a matching count, in addition to a matching name. If a Write Instruction dynamically creates a file during a simulation, and if a File Count is specified, the new file will then be given this value as its count. Writing to an existing may change its size, but its count will not be altered. The File Count can be used to create new "versions" of the same file and can also be a part of a Module's File Status Requirement

A Read/Write Instruction may optionally contain a list of Allowed Transfer Device(s). A file is transferred to/from a Processing Element via a Transfer Device. The default response of **ANY** presumes that any Transfer Device that connects the source and destination will be acceptable. Although NETWORK II.5 would never pick a Transfer Device that does not connect the source and destination, this feature allows the user to restrict the transfer of a large file to a high speed local data link, even though a low speed global bus also connects the two devices.

The first file read by a Module becomes its "original file." Original files are inherited, so that the original file for a successor Module becomes the first file read in by the predecessor Module. For example, assume Module K read in FILE 1, then FILE 2, and then chained to Module J. Module J would inherit FILE 1 as its original file. . If you do not want a Module to pass inherited files to successor Modules, you may set the Inhibit File Inheritance flag of the successor Module to NO. The "original file" concept is only used by the File Linear Distribution and the File Count Distribution. The File Linear Distribution performs an Ax + B calculation where A and B are user-defined coefficients and x is the number of bits read in the original file. The File Count Distribution performs a similar Ax + B calculation in which the count of the original file is substituted for x.

2.2.1.6 Message Instructions

Message Instructions are used to provide communication between two PEs. The effect of a Message Instruction is to deliver the Message Text of this instruction to the Destination Processor's Received Message List. The entries on the Received Message List can best be thought of as "tokens" which will remain in this list until "used" by a software Module which has this Message Text as a Message Status Requirement. Every PE has its own Received Message List. The same Message Text may appear more than once in the same list, and may appear simultaneously in more than one PE's list.

A PE executes one instruction at a time. The time it takes to execute a Message Instruction will be the time required to transfer the number of bits that make up this message across the chosen Transfer Device, plus any delays encountered. Every Message Instruction "executes" by the following sequence of events:

1. Request a Transfer Device.
2. Request the Destination Processor (only if the Destination Processor does not have an Input Controller).
3. Work the amount of time it takes to transfer the number of bits to send across the chosen Transfer Device.
4. Relinquish the Destination Processor (if one was requested).
5. Relinquish the Transfer Device.

If a Message Instruction is delayed for a Destination Processor, it will continue to hold its assigned Transfer Device. The Transfer Device will be listed as busy even though the transfer has not yet begun. These delays, plus the potential for interruption, make the exact execution time of a Message Instruction impossible to predict in advance (although a lower bound based on the data transfer time does exist).

During a message transfer, a PE is listed as busy executing this Message Instruction, even though it is not officially executing any "cycles." To model a PE that executes other instructions while sending a message, a new PE may be added to the system to handle the output operations.

A message may be sent by a Processing Element to itself. This operation does not require the use of a Transfer Device; therefore, it takes zero time for a PE to send a message to itself. This feature is useful in cases where Module coordination is handled by messages and Module-to-PE assignments are either dynamic or determined after the Modules are defined.

Every Message Instruction has:

- A name
- A Message Text
- A Message Count
- A Destination
- A Queue Flag
- A Resume Flag
- A Continue Message Flag
- An Inhibit Message to Self Flag
- An Inhibit Message Delivery Flag
- An Allowed Transfer Device List

The Message Text is the name that this instruction will put in the Received Message List of the Destination(s). It defaults to the name of the instruction that sends this message if no Message Text has been entered for the instruction. A Message Text can be a text string or a wild card. If the Message Text is not a wild card, this text string will be sent to the Destination(s). If the Message Text is a wild card, the message text is determined by the following two cases.

In the first case, a Message Text of ***** (i.e., an unqualified wild card) will retransmit the text of the "original message". The "original message" is the message that satisfied the initial Message Requirement of the Module executing this Message Instruction, or the predecessor to this Module. For example, consider a Module with this Message Status Requirement list:

```
PACKET ONE
PACKET TWO
PACKET THREE
```

The "original message" is PACKET ONE since it is first in the list. If a Module with the above Message Status Requirements list does not execute the Message Instruction, but chains to that Module which does, the "original message" remains the same because it is inherited by successor Modules. A Module will also inherit all of the messages that match the Message Status Requirements list of its predecessors. If you do not want a Module to pass inherited messages to successor Modules, you may set the Inhibit Message Inheritance flag of the successor Module to NO.

In the second case of a wild card used for the Message Text, the wild card will contain some text along with one or two asterisks. The Message Text will be selected by matching the wild card to the inherited message set of the Module that is executing the

Message Instruction. The first name in this list to match the wild card will be used. Messages satisfying Message Status Requirements will be added to the inherited message set of a Module chain.

In both cases of a wild card Message Text, if the Module has no inherited messages, then the Message Status Requirements list of the Module will be used to determine the Message Text. A search is made, starting at the top of the list, until a match with the wild card is found. When a match occurs, the name of the Message Requirement becomes the Message Text.

A Message Count can be specified in a Message Instruction as an integer value or a reference to a statistical distribution. A Message Count has various uses. A Message Count may be included to distinguish messages that have the same Message Text. A Module can be given a Message Status Requirement that keys on a message's count, as well as its text. The Message Count Distribution relies on the Message Count to generate Ax + B values in which the Message Count of the "original message" is substituted for x.

The length of a Message Instruction is the number of bits to send. It may be an integer, a reference to a statistical distribution, or NR ("No Response"). The length of a message is used to compute the amount of time to transfer the message over a Transfer Device. A length of **NR** will set the length of the message sent to the same number of bits as the "original message" received by the Module. A Message Linear Distribution will perform an Ax + B calculation for message length, where x is the length of the original message and A and B are user-defined coefficients. A File Linear Distribution will also perform an Ax + B calculation, where x is the length of the original file. Thus, to turn a file into a message, use a File Linear Distribution for message length.

The Queue Flag describes the course of action should another Received Message with the same Message Text appear in the Received Message List of a Destination. If the Queue Flag = **NO** and a copy of this Message Text already appears in the Destination's Received Message List, the new copy of the Message Text is not added to the list. The check of the list is made after the message has been completely transmitted. This means all devices will be utilized as if there were a message transfer, but no new copy of the message token will appear in the Received Message List. The message will be considered "lost" and recorded as such in the Received Message Report. This feature can be used to prevent multiple schedulings of a Module when the system loading causes a PE to fall behind. It also prevents cluttering the Received Message List if you send a PE messages which are not "used" by a Module.

If the Queue Flag = **NO** and no copy of this Message Text appears in the Received Message List, the newly transmitted message will be added to the Destination's Received Message List. If the Queue Flag = **YES**, the new copy of the message will always be added to the Destination's Received Message List.

If the Resume Flag = **NO**, an interrupted instruction will restart transmission of the message from the beginning when resumed. If the Resume Flag = **YES**, an interrupted instruction will only transmit the number of bits not yet sent when resumed.

The Continue Message Flag can be used if the Message Instruction has inherited an original message. If Continue Message = **YES**, the message is passed to its destination and the PE executing the instruction is considered to be an intermediary to the message, rather than the originator of the message. If this message is then echoed back from the destination, it will be returned to the origin of the original message. If Continue Message = **NO**, the PE executing the instruction is considered the originator of the message, even though an original message has been inherited.

If the Inhibit Message to Self Flag = **YES**, the message sent by this instruction is prevented from being received by the host PE executing this instruction. This allows the broadcast of a message to be received by every PE on a TD or LAN except the host PE. If a message is sent to a destination selected from a Destination Mix, when the destination selected is the Host PE for this instruction, a new selection will be made from the mix until a PE other than the host is selected. There is an upper limit of 100 attempts to find another PE to prevent an infinite loop.

If the Inhibit Message Delivery Flag = **YES**, the Processing Element will execute the Message Instruction, but no message will actually be delivered to the Destination(s) upon completion of the instruction. When this flag = **NO**, a message is delivered to the Destination(s) as controlled by the Queue Flag. The time required to execute a Message Instruction is not affected by the Inhibit Message Delivery Flag.

A Destination indicates where to send the Message Text. A Destination may be:

- the name of a PE
- the name of a Route
- the name of a Destination Mix
- the reserved phrase "**GLOBAL MESSAGE LIST**"
- the reserved word "**ECHO**"
- the reserved word "**NEXT**"
- a name using a wild card

Most Message Instructions will send a message to a Destination using a Transfer Device. This device must connect the sender and receiver, either directly or indirectly. To transfer a message over more than one Transfer Device from source to destination requires use of a Route. A Route defines a path from source to destination. The path consists of a list of PE names with an optional allowed TD list for each PE. Routes are covered as a separate topic in Section Routes.

Sending a message to the Destination **GLOBAL MESSAGE LIST** does not send it to any PE at all! Instead, it sends the message to a phantom list that all PEs have access to. Since it is not sending a message to an actual PE, this instruction does not use a Transfer Device and therefore executes in zero simulated time. This feature is very valuable when you would like to schedule a fixed number of tasks (perhaps a number randomly chosen) which have a choice of host processors. As an example, to model a parallel processor that spawns *n* copies of a task that run on the first available of *y* PEs, just send *n* messages to the Global Message List and define one Module that waits for that message and runs on any of the *y* PEs.

If the Destination is the reserved word **ECHO**, this instruction will send its Message Text to the Processing Element that sent the "original message" (as previously defined) via the same Transfer Device(s) on which it was received. If a message that arrives via the Global Message List is to be ECHOed, a Transfer Device will be dynamically selected from those TDs that link the source PE to the Destination. If no such TD exists, a runtime warning message will be issued and the message will not be delivered.

If the Destination is the reserved word **NEXT**, this instruction will send its Message Text to the Processing Element that appears next in the Route of the original message. If the original message is not traveling along a route, a runtime warning message is produced. If the Destination is a , this instruction will send the Message Text to every PE on the selected Transfer Device whose name matches the given wild card name. The Destination may also be the host PE executing this instruction. In this case, the PE is just sending the message to itself. This takes zero time, and does not require a Transfer Device.

Finally, if the Destination is a Destination Mix, the ultimate destination for the message is selected randomly according to the percentages specified in the mix.

If the Allowed Transfer Device List is set to **ANY** (the default), it will be presumed that any Transfer Device that connects the source and destination will be an acceptable path.

2.2.1.7 Semaphore Instructions

Semaphore Instructions are used to set or reset global flags called Semaphores and/or to modify the count of a Semaphore. These Semaphores are named by the user and may have a status of either **SET** or **RESET** and a count represented as an integer. The status (**SET**/**RESET**) and/or the count of a Semaphore can be examined or changed by any Processing Element or Module. Semaphores are commonly used to indicate device availability, to synchronize parallel activities, to record the duration of certain events, or to indicate the current phase of the simulation.

Executing a Semaphore Instruction takes zero simulated time. Therefore, Semaphore Instructions will never be interrupted in the midst of execution by another Module because they never can be "caught" executing. However, a Module might execute a Semaphore Instruction that changes a Semaphore to a state where the Module's own run conditions are no longer met. In this case of Module suicide, the Module will be interrupted (or cancelled) and listed as having this Semaphore Instruction as its current instruction. However, if the Module resumes, it will start with the next instruction in the Module's Instruction List.

Every Semaphore Instruction has:

- A name
- A Semaphore
- A Set/Reset Flag
- An assignment type
- A count

The Semaphore represents the name of the Semaphore this instruction is to modify. The Set/Reset Flag determines the effect the Semaphore Instruction will have on the status of the Semaphore. If the Set/Reset Flag = **SET**, then the status of the Semaphore will be SET upon completion of the Semaphore Instruction. If the Set/Reset Flag = **RESET**, then the status of the Semaphore will be RESET upon completion of the Semaphore Instruction. If the Set/Reset Flag = **TOGGLE**, then the status of the Semaphore will be reversed (from SET to RESET or from RESET to SET) when the instruction is finished. If the Set/Reset Flag does not equal **SET**, **RESET** or **TOGGLE**, then the Semaphore Instruction will have no effect on the status of the Semaphore.

The assignment type will determine if a Semaphore Instruction will modify the count of a Semaphore upon execution. Three different assignment types may be specified; "Increment By", "Decrement By", and "Equal To". The assignment type is used in

conjunction with the count of a Semaphore Instruction to modify a Semaphore's count. The effects of the assignment types are:

- Increment By — add the Semaphore Instruction count to the count of the Semaphore
- Decrement By — subtract the Semaphore Instruction count from the count of the Semaphore
- Equal To — set the count of the Semaphore equal to the count of the Semaphore Instruction

A Semaphore count can be used by a Module's Wait For and Chain If preconditions and Run When and Run Until conditions. See Section 2.3.1.1 Module Preconditions for more information on the use of Counter Semaphore preconditions. See Section 2.3.1.3 for information on Module run conditions. The count of a Semaphore can never be less than zero, so subtracting from the count of a Semaphore whose count is zero has no effect on the count and decrementing by a value greater than the Semaphore count will merely set the count to zero. Semaphores also can be used to create responses to time operations. See the Semaphore Report section in chapter 5, entitled Network Output Reports for an explanation of how to time operations using Semaphores to define a response.

2.2.2 Transfer Device Details

As mentioned in the Transfer Device Overview section, Transfer Devices (TDs) are the links connecting Processing Elements and Storage Devices. They are used to move data between two Processing Elements or between a Processing Element and a Storage Device. Data is moved between Processing Elements over a Transfer Device as the result of a Message Instruction and between a Processing Element and a Storage Device as the result of a Read/Write Instruction.

2.2.2.1 Words and Blocks

Transfer Devices organize all transmissions of data into groups called *words* and *blocks*. This allows the user to model up to two levels of a data packet structure by means of a user-specified word overhead time and block overhead time.

Words are the lowest level of a user-defined packet structure. Each is made up of a group of bits. The user-defined word overhead time can then be used for adding parity bits, start/stop bits, and so on. Fractions of a full word incur the full word overhead time. The term word is fairly standard across most implementations, so a NETWORK II.5 word generally represents a "word" in the modeled systems.

Blocks are the highest level of a user-defined packet structure. Each is made up of a group of words. The user-defined block overhead time can be used for adding source and/or destination headers to message packets, adding message checksums, and so on. Fractions of a block incur the full block overhead time. Blocks generally represent packets on an Ethernet and frames on a token ring. However, the term Block is intentionally generic. A Block can represent any structure that is a collection of words.

A Transfer Device will transmit data in one of two ways, depending on an attribute called Separate Blocks. If Separate Blocks = **YES**, the TD will transmit data one block at a time. The PE using the TD will have to contend for the TD once for every block sent. To send a message that required 10 blocks over an Ethernet, a PE would have to contend for the TD 10 times. If Separate Blocks = **NO**, the time to transfer the total number of words and blocks will be computed. However, the actual transmission will be made as one transfer. So, in the previous example, the PE would contend for the TD only once. Setting Separate Blocks to NO may reduce the execution time of your model by decreasing the number of TD requests. If the system that you are modeling actually transfers data one block at a time, this speedup will result in some loss of modeling accuracy.

2.2.2.2 Transfer Device Attributes

Every Transfer Device has:

- A protocol
- A Cycle Time
- A Bits Per Cycle
- A Cycles Per Word
- A Words Per Block

- A Word Overhead Time
- A Block Overhead Time
- A Minimum Bits to Send
- A Block Error Probability
- A Block Error Stream
- A Scale Error Probability flag
- A Block Retry Time
- A Separate Blocks flag
- A Connection List

The Protocol defines the method of resolving contention between Processing Elements for a Transfer Device. Your choices of Protocol are FCFS (First Come, First Served), Collision, Priority, Token Ring, Slotted Token Ring, Priority Token Ring, Aloha and Crossbar. Based on your selection, you may be required to specify additional attributes that are specific to the protocol chosen. As the protocols are all quite different from each other, each protocol will be covered in a separate section. Protocols not predefined in NETWORK II.5 may be modelled by the user by using Semaphores and messages to control the Modules that access a Transfer Device.

Cycle Time is a real number that specifies the "clock speed" of a Transfer Device. It is usually given as the time for one clock tick. If a Transfer Device is to send one bit at a time (i.e. serial), the Cycle Time is the inverse of the data rate. Bits Per Cycle is an integer, and specifies the number of bits transmitted in one cycle. Cycles Per Word is also an integer. The number of bits per word is obtained by multiplying Bits Per Cycle by Cycles Per Word.

For example, a serial bus with a transfer rate of 10 Mb (Megabits) per second translates to a Cycle Time of .1 microsecond per bit (the inverse of 10,000,000 bits per second). For parallel transfer devices, the Cycle Time is the inverse of the clock speed multiplied by the number of bits sent in parallel. As an example, take a 16 bit parallel transfer device running at clock speed of 64,000,000 cycles per second The Cycle Time is $(1/(64{,}000{,}000)) \times 16 = .25$ microseconds per cycle.

Words Per Block may be either an integer or NR ("No Response"). An integer value represents the number of words required to make one full block. A value of NR means that transfers are not to be broken up into blocks.

Word Overhead Time is a real number specifying the amount of time to add to each word transferred to account for parity bits, start/stop bits, and so on. This attribute should be set to zero if no overhead is desired. Block Overhead Time is a real number specifying

the amount of time to add to each block transferred to account for destination addresses, checksums, and so on. It should also be set to zero if no overhead is desired.

A Minimum Bits to Send may be specified to "pad" the data to a minimum size. This is especially useful for various LAN protocols that require a minimum frame size. This padding will apply only to the transfer. The number of bits received will always equal the number of bits sent.

The Separate Blocks Flag will determine if the transmission of a message or file will be handled as one transmission or as several transmissions of the component blocks of the message or file. If the flag is set to **YES**, messages or files will be transmitted one block at a time, with the controlling Processing Element releasing the Transfer Device and destination Processing Element/Storage Device between frame transmissions. If the flag is set to **NO**, the Transfer Device will be held by the Processing Element for the entire message or file. NOTE: setting the flag to **YES** may significantly increase the model simulation time due to the increased number of TD requests/relinquishes required to send every message or file.

The Connection List is a list of the names of the Processing Elements, Storage Devices and Gateways which connect to this Transfer Device. Using the reserved word in this list will connect this Transfer Device to every Processing Element, Storage Device, and Gateway in the simulation. Some of the TD protocols also require a key value, to be associated with some or all of the Connections. Generally, this list will be created graphically in NETWORK by drawing the connections from a TD to PEs and SDs.

2.2.2.2.1 Transfer Device Error Attributes

A Block Error Probability may be specified to model data transfers lost due to noise on a TD. The probability is a percentage, between 0 and 100 percent. The default is zero percent, which means no blocks are lost. A probability of 100 percent is allowed, meaning that no blocks are ever delivered.

The Scale Error flag is used if you set a non-zero Block Error Probability. Blocks have a maximum size. If you send less than a full block and set the Scale Error flag = **YES**, the probability of error will be scaled down since fewer bits are being sent. In effect, setting the Block Error Probability to **YES** turns it from a block error probability to a bit error probability. As an example, take the case of sending 512 bits over a TD with an Error Probability of .02 percent and a 1024 bit block size. If Scale Error = YES, the Error Probability used will be multiplied a factor of 1/2, resulting in a probability of .01

percent, since only 1/2 of a block is being sent. If Scale Error = **NO**, the probability of error would be fixed at .02 per cent per block.

The Block Retry Time specifies how long a PE is to wait before retransmitting a block lost due to a transmission error or message buffer overflow. It is only used if this TD's Separate Blocks flag is set to **YES**. By default, this attribute is zero and the PE will retransmit the lost block immediately.

2.2.2.3 Transfer Time Example

An example computation of the time needed to transfer bits over a Transfer Device follows.

To transfer 56 bits over a Transfer Device with:

9600 baud transfer rate	Basic Cycle Time = 104 mic
serial bus	Bits Per Cycle = 1
7 bits/word	Cycles Per Word = 7
5 words/block	Words Per Block = 5
1 bit per word overhead	Word Overhead Time = 104 mic
1 word per block overhead	

Block Overhead Time = $8 \times 104 = 832$ mic

Number of cycles = bits to transmit / bits per cycle
= 56 / 1
= 56 cycles

Number of words = number of cycles / cycles per word
= 56 / 7
= 8 words

Number of blocks = number of words / words per block
= 8 / 5
= 1.6 (which rounds up to 2 blocks)

Transmission time = (number of blocks × block overhead time) +
(number of words × word overhead time) +
(number of cycles × bus cycle time)
= $(2 \times (8 \times 104)) +$

$$
\begin{aligned}
&(8 \times 104) + \\
&(56 \times 104) \\
= \ &1{,}664 + 832 + 5{,}824 \\
= \ &8{,}320 \text{ microseconds}
\end{aligned}
$$

2.2.2.4 FCFS (First Come, First Served) Protocol

This is the simplest protocol and is therefore the default. In this protocol, requests are served in the order in which they are made. Once a PE is allocated a Transfer Device, it may use it once, regardless of how long that takes. No other PE may interrupt the current user of a Transfer Device. However, if the Module executing on the PE which is using Transfer Device is interrupted, the Transfer Device is released and the TD updates its statistics reporting an interrupted transfer.

2.2.2.5 Collision Protocol

The Collision Protocol does not use a central controller to arbitrate among contending Transfer Device users. Instead, each PE is responsible for checking the Transfer Device to see if it is busy before using it. If the Transfer Device is busy, each PE will wait until the requested device is idle and then wait an additional amount of time (the Contention Interval) before attempting to access the TD. A collision occurs if two or more PE's "see" a TD as idle and both try to use it. A TD is vulnerable to a collision for a period of time (the Collision Window) after a new user takes the TD. This value accounts for propagation delays, and delays between checking a TD status and actually beginning to transmit. In the event of a collision, all users must abort their current transmission and retry later (the Retry Interval).

In addition to all of the standard TD attributes listed in Section 2.2.2.2 Transfer Device Attributes, a collision TD has the following attributes:

- A Collision Window
- An Interframe Gap
- A Contention Interval
- A Retry Interval
- A Jam Time

A collision protocol Transfer Device has three states: idle , unsettled and busy. If the Transfer Device is idle, any PE on the Transfer Device may use it immediately. The

Transfer Device will then be considered unsettled from the time the PE starts using the Transfer Device until the user-specified time (the Collision Window) has elapsed.

If any PE attempts to access a Transfer Device while it is unsettled, a collision will be declared and the current user of the Transfer Device will be interrupted. Both the former user and the requesting PE will send a jamming signal for the length of time given in the Jam Time attribute. They will then each wait their randomly selected Retry Interval and then try to access the TD again. The number of collisions which occurred during a simulation will be reported in the Transfer Device report.

If any PE attempts to access a Transfer Device while it is busy (i.e., in use, but after its Collision Window is over), it will queue up for the Transfer Device. When the Transfer Device becomes available, the Interframe Gap (if specified) is imposed on all PEs waiting to use the TD. The Interframe Gap and the Contention Interval timers run concurrently. This means that a deferring PE waits the greater of the Contention Interval or the Interframe Gap before requesting the Transfer Device.

The Retry Interval must be a statistical distribution function (to avoid infinite loops). The Contention Interval, Jam Time and the Collision Window may be either statistical distributions or real numbers. Care must be taken in selecting the Retry Interval and Contention Interval to be used. If the Retry Interval or the Contention Interval return delay times for the two contending PEs that are within the Collision Window of a Transfer Device, another collision will occur. "Infinite loops" of collisions are possible!

If a Key Linear type statistical distribution function is used for the Retry Interval, Jam Time, and/or the Contention Interval of a collision Transfer Device, the additional entry of key values is required for every PE in the Transfer Device Connection List. These key values will be used as the x value in the Key Linear distribution's calculation. This value is a real number, greater than or equal to zero and unique within the Transfer Device Connection List in which it appears. Key values are only required for Transfer Device connections that are Processing Elements and are superfluous for other Transfer Device connections.

NETWORK II.5 supports the IEEE 802.3 standard Truncated Binary Exponential Backoff algorithm. It may be used for the Retry Interval only by selecting the Standard Retry Interval option. In brief, this algorithm uses the number of collisions to determine the length of time to wait before again requesting a Transfer Device after a collision. The algorithm is called in NETWORK and has the following parameters:

TYPE = IEEE BACKOFF

Slot Time – a multiplier. The distribution picks a random integer based on the number of collisions, then multiplies that integer by this value to compute the amount of time to wait before RETRYING to get the TD.

Retry Limit – After this number of collisions, the processor stops retrying for the TD. Instead, it waits the Limit Delay, then tries again. If you reach the Retry Limit, you are in real trouble! The number of times that you reach this limit is included in the TD report.

Limit Delay – After a PE reaches its Retry Limit, it waits this amount of time before trying for the TD again. When it retries, the collision count restarts at zero.

Offset – a parameter not in the IEEE standard. This value is added to the result of the IEEE algorithm. It is included merely to add flexibility. Set to zero if you do not want to modify what would be the standard result of the IEEE formula.

Stream – a random number stream for this distribution.

The algorithm uses the parameters as follows:

```
IF TYPE = "IEEE.BACKOFF"
   ''
   ''uses PE.COLLISION.COUNT to store the number of collisions
   ''COMPUTED.DELAY is the value to be used as the
   ''backoff delay
   ''RANDI.F(A, B, C) returns a random integer value
   ''between A and B.  C is the random number stream.
   ''
   ADD 1 TO PE.COLLISION.COUNT..
   IF PE.COLLISION.COUNT.. GT RETRY.LIMIT
         ''the maximum number of retries
         LET PE.COLLISION.COUNT.. = 0
         LET COMPUTED.DELAY = LIMIT.DELAY
   ELSE
         LET COMPUTED.DELAY = SLOT.TIME * RANDI.F(0,
               2**(MIN.F(10., PE.COLLISION.COUNT..)),
               ST.STREAM..) + OFFSET
   ENDIF
ENDIF
```

2.2.2.6 Priority Protocol

A Priority Transfer Device is allocated in the same way as in the FCFS protocol, with the exception that the request queue is ordered by PE priority, not by time of request. The PE priority is assigned by the Key associated with its the PE name in the Transfer Device Connection List. Because the priority definition is local to a particular Transfer Device, a PE can have a different priority on each Transfer Device. The Priority Protocol uses priority to resolve contention, not to provide for interruption. Therefore, if a higher priority request is received while a lower priority PE was using the Transfer Device, that new request would be served as soon as the lower priority PE completed its transfer. As with the FCFS protocol, if the Module executing on the PE which has the Transfer Device is interrupted, the Transfer Device is released and updates its statistics reporting an interrupted transfer.

Interestingly, the 802.6 MAN (Metropolitan Area Network) standard can be easily modeled with the Priority Protocol. The control bus in the 802.6 is used by all stations on the ring to compute a prioritized request queue for the ring. To model this in NETWORK II.5, you only need to model the data bus and let the Priority Protocol mechanism handle the ordering of TD requests.

2.2.2.7 Token Ring Protocol

A Token Ring Protocol views the Transfer Device Connection List as a ring with the last member of the list followed (at least in a logical sense) by the first member of the list. After a PE has used the Transfer Device an allowed number of times (given by the connection's Key), the Transfer Device is allocated to the next PE on this ring. PEs without pending requests will be skipped. It is possible that no other PE has a request and control will immediately return to the PE that was using the Transfer Device. Once a PE has control of a Transfer Device, it may not be interrupted by any other PE. It will release control of the Transfer Device when one of the following occurs:

1. It has finished the Key number of accesses of this TD. The Key is specified for a given PE in the Connection List. The default value for Key is one time. If the Key value is greater than 1, the PE must execute a continuous series of instructions that all access this TD to retain control. When a PE executes an instruction that does not access this TD, the TD is allocated to the next PE on the ring with a request. An exception to this rule is the Semaphore type instruction, which does not force the PE to give up the TD because it always takes zero time to execute.

2. It has no further requests.

3. The current Module is interrupted (the Transfer Device updates its statistics reporting an interrupted transfer) and the interrupting Module does not immediately use the TD.

A Token Ring TD also has a Token Passing Time attribute. It may either be, “NR” (No Response), a real number or a statistical distribution. The Token Passing Time specifies how long it takes to pass the token from one station to the next. The default Token Passing Time is NR (“No Response”) which disables adding a delay to pass the token. NOTE: If you specify a Token Passing Time, your TD will always be 100% busy. When not serving a PE it will be actually passing the token from station to station. This looks very good in the animation but can easily double (or worse!) the execution time of a model and cloud the actual data utilization of the TD.

2.2.2.8 Slotted Token Ring Protocol

A Slotted Token Ring Protocol also views the Transfer Device Connection List as a ring with the last member of the list followed (at least in a logical sense) by the first member of the list. It differs from a Token Ring Protocol by using a Slot Time to control Transfer Device access.

Every time a new PE is granted a TD, a timer is started. As long as the PE continually uses (except for Semaphore instructions) the allotted TD with an unbroken sequence of instructions, it maintains exclusive use of the TD (so long as it is still within its allowed slot). If a PE's Slot Time expires and it is still using the TD, it will finish its current instruction before control passes to the next PE along the ring.

Slot Time may be either a real number or a statistical distribution. If a Key Linear type statistical distribution is used, the key value associated with each PE in the Connection list will be used as x in the Ax + B computation. This allows giving a different Slot Time to each PE on a Transfer Device.

The Token Passing Time may be NR (“No Response”), a real number, or a reference to a statistical distribution. If set to NR, when a PE relinquishes a TD it is immediately allocated to the next PE on the ring with a request. If there are no pending requests, the TD will be idle. If a PE has many items to send but its Slot Time expires, it will

immediately continue with a new Slot Time if there are no other PEs with a pending request.

If a Token Passing Time is specified, it will give the delay in passing the token from PE to PE. If there are no pending requests, the token will still circulate around the ring at the stated rate. When a PE requests a TD, it must wait until the token arrives. While Token Passing Time adds realism to a model, it can dramatically increase model execution time. Even a TD which is never requested will be 100% busy passing the token! Use the Key Linear distribution to get a different Token Passing Time from each PE on the ring.

2.2.2.9 Priority Token Ring Protocol

The Priority Token Ring Protocol offers a generic capability that can be used to model both the 802.4 Token Bus and the Fiber Distributed Data Interface (FDDI) Token Ring. It is defined by a token rotation sequence, allocations for synchronous traffic and target token rotation times for asynchronous traffic. An unlimited number of priority classes may be specified for asynchronous traffic.

When a simulation starts, the first PE in the TD Connection List is given the token. When the token is released, it is passed to the next PE in the Connection List. After the token has been passed to each PE in the Connection List, it is once again passed to the first PE in the Connection List and this cycle will continue for the duration of the run. Both synchronous and asynchronous timers may be specified which will determine how long an individual PE will retain the token.

When a PE receives the token, it will first transmit using its synchronous allocation. A PE's synchronous allocation is a guaranteed amount of time that the PE may use the TD every time that PE receives the token. Each Connection within the TD Connection List that represents a PE will have its own synchronous allocation defined. Synchronous allocation may be a value greater than or equal to zero. Each of these Connections may also have a minimum synchronous priority defined; used to prevent low priority transfers from using the synchronous allocation of a PE. The priority of a transfer request is based on the priority of the Module executing the instruction invoking the transfer. The synchronous allocation is tested before the start of every transmission and is not used to interrupt a transmission. Therefore, a PE will continue to transmit after the expiration of the synchronous allocation for any transmission begun before the allocation expires.

After a synchronous allocation ends, a PE will begin its asynchronous traffic, which is traffic that is controlled by Target Token Rotation Times (TTRT). For each PE

Connection of the TD, a list of Target Token Rotation Times may be defined for any number of priority classes. The list of Target Token Rotation Times is used to provide a dynamic allocation for the PE based on the priorities of its transmission requests. Each TTRT will have a priority and a time value associated with that priority. The TD priority of a transfer request is based on the priority of the executing Module that is invoking the transfer. Multiple levels of priority are available to a PE to restrict the use of a Transfer Device when it is highly utilized.

Asynchronous traffic is regulated by the Target Token Rotation Times of a Connection and the Token Holding Time (THT). Whenever a token is received, the time between the current time and when the token was last received is recorded as the current Token Holding Time. When asynchronous traffic begins, the THT clock is started beginning with this assigned value. The time spent on synchronous traffic does not affect the THT clock. The priority of an asynchronous transmission will be used to select the TTRT whose priority is equal to or most nearly equals without exceeding the priority of the asynchronous transmission. If the lowest priority TTRT has a priority greater than the priority of the asynchronous transmission request, the token is passed to the next PE. Before each asynchronous transmission, The PE compares the Token Holding Time to the time of the selected TTRT. If the Token Holding Time is equal to or greater than the TTRT, the token is passed to the next PE in the Connection List. A Target Token Rotation Time will not interrupt a transmission, but is used to determine if a transmission should begin.

2.2.2.10 Aloha Protocol

The Aloha protocol can be described as a Collision protocol in which the status of the Transfer Device is unknown. Transmitting PEs are not aware of when the TD is busy nor do they recognize when a collision occurs. When a PE wants to use an Aloha TD, it immediately begins transmitting and continues even if a collision occurs. An instruction that calls an Aloha TD should not list any other Allowed Transfer Devices because succeeding Allowed TDs will never be selected due to the fact that there is never a delay for an Aloha TD.

An Aloha protocol TD has three attributes in addition to the normal TD attributes; Automatic Retry, Retry Interval, and Slot Width. Automatic Retry can be set to **YES** or **NO**. If Automatic Retry is set to **NO** and a collision occurs, the colliding instructions will continue until completed and then its host Module will move on to its next instruction. However, the message or file being transmitted will not have been sent or received. If Automatic Retry is set to **YES** and a collision occurs, the colliding instruction will

complete and then be rescheduled for another transmission attempt at the end of the collided transmission. The time the instruction will wait before retrying is derived from the Retry Interval which must be a statistical distribution function if it is specified. The IEEE BACKOFF statistical distribution function may be used for Retry Interval. Retry Interval is provided as a modeling convenience because the actual Aloha protocol uses acknowledge messages to detect undelivered transmissions. This feature permits modeling this mechanism without adding additional Modules and messages.

The Slot Width can be used to convert a pure Aloha TD (Slot Width = NR) into a slotted Aloha TD. When you provide a value for the Slot Width, a timer equal to the Slot Width is used to control the Transfer Device. PEs will only be allowed to transmit on the TD at the beginning of each time cycle defined by the Slot Width.

The Resume Flag of Message and Read/Write Instructions acts normally on an Aloha TD. This flag determines whether an interrupted instruction restarts completely or will resume and complete the remaining transmission. A collision on an Aloha TD does not interrupt a Module so the Resume Flag will not cause the retransmission of bits lost due to a collision because the sender is not interrupted. To automatically retransmit all lost transfers on an Aloha TD, set Automatic Retry to **YES**.

2.2.2.11 Crossbar Protocol

The Crossbar protocol permits simultaneous transmissions on the same Transfer Device at any time without any complications resulting from multiple transmissions. A Crossbar Transfer Device can be refined by using the Source Blocking Flag, the Destination Blocking Flag and the Number of Circuits attribute. If the Source Blocking Flag = YES, a PE will be blocked from transmitting to any other PE that is currently sending a message or a file. The blocked PE will delay until its intended destination has finished sending its transmission. If the Destination Blocking Flag = YES, a PE will be blocked from transmitting to any other PE that is currently receiving a message or a file. The blocked PE will delay until its intended destination has finished receiving the current transmission.

The Number of Circuits attribute can be used to limit the number of transmissions occurring on the TD at any given time. If Number of Circuits is NR (the default), the number of transmissions is limited only by the number of devices that are connected to the Transfer Device. If Number of Circuits is set to a number, then that number represents the maximum number of transmissions that can occur on the TD at any given time. A PE attempting to send a message or a file when the number of transmissions

equals the Number of Circuits will delay until the number of transmissions is less than the Number of Circuits.

2.2.3 Storage Devices

As mentioned in the Storage Device Overview section, Storage Devices contain both user-named files and unstructured storage (called General Storage). They have a capacity measured in bits. They are used to model anything that stores data either temporarily or permanently. This includes buffers, disk drives, tape drives and random access memory. Files are read or written analogous to the way real storage devices work.

2.2.3.1 Words and Blocks

Storage Devices automatically decompose all file reads and writes into words and blocks. Words are the smallest segment and usually correspond directly to the actual word size of the Storage Device being modelled (but they need not be). A word overhead time may also be specified.

Blocks are the largest segment of data storage and are made up of groups of words. Blocks are commonly used to model features of real Storage Devices. For example, one feature easily modelled by a block is the sector of a disk. Once a sector is located (Block Overhead Time models the disk seek time), the words in the sector are transmitted without delay.

When a read or write involves a block that is not completely “full,” the partial block will incur the full Block Overhead Time, but is not filled with blank words to pad out the transfer. The amount of time it takes a Storage Device to read or write a file depends not only on the speed of this device but may also depend on the speed of the requesting Transfer Device. If the Storage Device is faster than the Transfer Device, the read/write proceeds at the speed of the Transfer Device. The Storage Device is marked as busy for the entire length of the transfer. If, on the other hand, the speed of the Storage Device is the limiting speed, the read/write will take the amount of time it takes to move the data in or out of the Storage Device.

2.2.3.2 Storage Device Attributes

Every Storage Device has:

- A name
- A Capacity
- A Bits Per Word
- A Words Per Block
- A Number of Ports
- A Read Word Access Time
- A Write Word Access Time
- A Read Word Overhead Time
- A Write Word Overhead Time
- A Read Block Overhead Time
- A Write Block Overhead Time
- A Read Access Delay
- A Write Access Delay

Capacity is a real number specifying the total number of bits a Storage Device can hold. It should not include any overhead bits. Even though bits are usually integers, capacity is a real number to allow capacities greater than an integer's upper bound of slightly more than two billion. The upper bound of a real number depends on the word size of the host computer, but is generally at least 1.7 E+32.

The Number of Ports is the number of simultaneous accesses that can be made to the Storage Device. The value is independent of the actual number of Transfer Devices that connect to this Storage Device. If, for example, four TDs are connected to a Storage Device which had the Number of Ports = 2, the Storage Device would serve up to 2 Transfer Devices simultaneously. All other requests would be queued on a first-come, first-served basis. In addition, no more than one PE can access a given user named file at any given time. However, multiple users may access General Storage simultaneously.

Read Word Access Time and Write Word Access Time are real numbers, or references to a statistical distribution. They reflect the amount of time to read/write one word of data. Read Word Overhead Time and Write Word Overhead Time are real numbers or references to statistical distributions specifying an additional amount of time to add per word accessed. Bits Per Word is an integer specifying the number of bits in a word (not including overhead bits). Words Per Block is an integer specifying the number of words in one block. Read Block Overhead Time and Write Block Overhead Time are real numbers, or references to statistical distributions, specifying the amount of time to add to each block read or written to account for Storage Device overhead.

Read/Write Access Delay represents the initial amount of time before a read/write transfer can begin on the Storage Device, regardless of the length of the read/write. An example when this attribute is helpful is modeling the hard disk drive on a laptop computer. A hard disk request might be delayed waiting for a powered down hard disk to come up to speed. Once the hard disk is up to speed, the entire read/write transfer may proceed. The time for the transfer will then be based on the other SD speed and overhead attributes and the Transfer Device rate connecting the SD to the PE requesting the read/write.

2.2.3.3 Example — MODELING A DISK DRIVE

A sample calculation of the amount of time needed to move data from a Storage Device follows.

Disk Read Time Example:
To read 133 words from a Storage Device with a:

- 64 word sector
- Read Word Access Time = 500 mic
- average seek time = 10 mic (Read Block Overhead Time)

Word Time: 133 words × 500 mic/word = 66500 mic

Block Overhead Time: 133 words / 64 words per block = 2.08 blocks
3 blocks x 10000 mic = 30000 mic

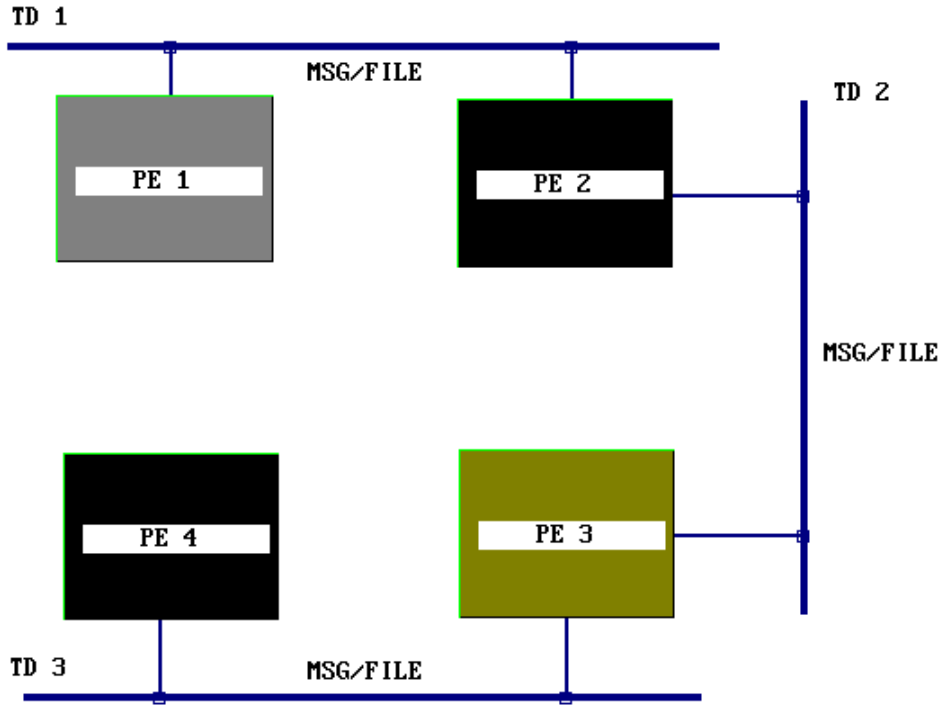
Total Transfer Time: Word Time + Block Overhead Time =
66500 + 30000 = 96500 mic

2.2.4 Routes

Message instructions let you specify a Route as a destination. A Route is defined at the top level by giving it a name and a list of PE, Gateway, and Destination Mix names, each of which may have an allowed TD/LAN list. If a message instruction uses a Route, the message sent will visit each PE or Gateway along the Route in the order they appear until it reaches the end of the Route. The last entry in a Route must be a PE or a Destination Mix that will produce a PE name. A Gateway is not a valid final destination for a message.

Gateways automatically include the ability to forward messages along a route. However, each PE found in a Route must have a user defined Module to pass along the message. This user defined Module must use the message as a Wait For type Message Status Requirement (or a wild card Message Status Requirement that matches the message) and then call a Message Instruction whose Message Text is * and whose destination is the reserved word **NEXT**. The message instruction will then use the destinations and allowed TD lists found in the Route. If no forwarding Module exists at the destination, no error is generated and the message is not forwarded.

For example, consider the case of a system of 3 TDs connecting 4 PEs as shown in the following diagram.



A Sample Simulation Using a Route

The easiest way to send a message from PE 1 to PE 4 is to define a route that consists of PE 2, PE 3 and PE 4. The destination of the message instruction in PE 1 would then be the name of this Route. PE 2 and PE 3 require a Module with a Message Requirement matching the name of the message to forward. This Module will execute a message instruction whose destination is NEXT. So when PE 1 sends its message, it is then

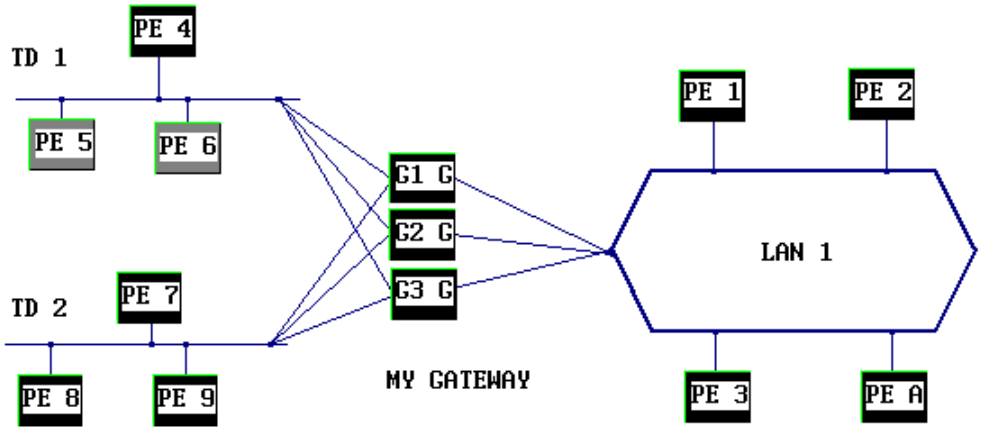
forwarded by PE 2 and then PE 3 until it reaches PE 4. If you use wild cards for the Message Requirement of the forwarding Modules, one Module is able to forward many different messages. Also note that the forwarding Module can work in any direction. It forwards messages regardless of source and destination. Therefore, you probably will not need more than 1 forwarding Module per PE.

```
***** ROUTES
HARDWARE TYPE = ROUTE
  NAME = PE 1 TO 4
    NEXT PE = PE 2
    NEXT PE = PE 3
    NEXT PE = PE 4
```

A Sample Route

2.2.5 Echo Processor Lists

Message Instructions allow you to reference an Echo Processor List. Echo Processor Lists are defined at the top level. The EPL (Echo Processor List) is given a name. It consists of a list of TD names, each of which has a PE name associated with it. This feature allows a PE to mark a message so that it ECHOs via a different PE than itself. By default, whenever a PE retransmits a message, it tags the message with its own name. If the message sent was returned using the ECHO feature, it would then return via this PE. If you specify an EPL for a message instruction, a PE can tag a message with a different PE's name. When an EPL is specified, the selection of which PE's name to use is based on the **incoming** TD's name.



A Sample Model of a Gateway

```
HARDWARE TYPE = ECHO PE LIST
  NAME = GATEWAY CLUSTER LIST
    TD = ETHERNET
      ECHO PE = G1/GATEWAY
    TD = TOKEN RING
      ECHO PE = G2/GATEWAY
    TD = TOKEN BUS
      ECHO PE = G3/GATEWAY
*
HARDWARE TYPE = PROCESSING
  NAME = G1/GATEWAY
    INSTRUCTION REPERTOIRE =
      INSTRUCTION TYPE = MESSAGE
        NAME ; RETRANSMIT ANYTHING
          MESSAGE ; *
          DESTINATION PROCESSOR ; NEXT
          ECHO PE LIST = GATEWAY CLUSTER LIST
  NAME = G2/GATEWAY
    INSTRUCTION REPERTOIRE =
      INSTRUCTION TYPE = MESSAGE
        NAME ; RETRANSMIT ANYTHING
          MESSAGE ; *
          DESTINATION PROCESSOR ; NEXT
          ECHO PE LIST = GATEWAY CLUSTER LIST
  NAME = G3/GATEWAY
    INSTRUCTION REPERTOIRE =
      INSTRUCTION TYPE = MESSAGE
        NAME ; RETRANSMIT ANYTHING
          MESSAGE ; *
          DESTINATION PROCESSOR ; NEXT
          ECHO PE LIST = GATEWAY CLUSTER LIST

SOFTWARE TYPE = MODULE
  NAME = GATEWAY/RETRANSMIT
    PRIORITY = 0
    INTERRUPTIBILITY FLAG = NO
    CONCURRENT LIMIT = 32000
    ALLOWED PROCESSORS =
      G1/GATEWAY
      G2/GATEWAY
      G3/GATEWAY
    REQUIRED MESSAGES =
        *
    INSTRUCTION LIST =
      EXECUTE A TOTAL OF ; 1 RETRANSMIT ANYTHING
```

A Partial Listing of an Echo PE List and Gateway Model

The Echo Processor List feature is useful when modeling devices like gateways. Presume that you model a gateway between 3 TDs by using 3 PEs. Each PE is dedicated to serve

as the output buffer for a single TD. If a PE serving as an output buffer forwards a message, it obviously wants that message (if ECHOed) to return back through the correct output buffer. Since that output buffer is a different PE than the forwarding PE, an EPL is required. A sample gateway model using the EPL follows.

In the above example, any message arriving from the Ethernet which is forwarded by any message instruction which references GATEWAY CLUSTER LIST will be returned via G1/GATEWAY (if returned through the use of the ECHO feature). Otherwise, the message would be returned via the forwarding PE, which would make dedicating a PE as an output buffer impossible.

2.2.6 LANs

The LAN building block is used to quickly specify common LAN implementations in NETWORK II.5. A LAN is really just a standard Transfer Device with its parameters automatically filled in to match industry standard LAN implementations. A LAN connects as many Processing Elements, Storage Devices, or Gateways as desired. Traffic on the LAN is in the form of messages or files. A LAN operates identically to a TD when the TD is specified with the same parameters as the LAN. Therefore, refer to the relevant TD section for the details on how a LAN operates.

Every LAN has a:

- Name
- Implementation
- Connection List

To specify a LAN, all you need to do is give it a name, select an implementation from the list of implementations presented and draw the connections. The following implementations are included in this release.

Collision LAN Implementations

Ethernet
IEEE 802.3 CSMA/CD 10BASE5
IEEE 802.3 CSMA/CD 10BASE2
IEEE 802.3 CSMA/CD 10BASET
IEEE 802.3 CSMA/CD 1BASE5 STAR
IEEE 802.3 TOP

Token Ring LAN Implementations
- IEEE 802.5 4Mb
- IEEE 802.5 16Mb
- FDDI

Token Bus LAN Implementations
- IEEE 802.4 1Mb
- IEEE 802.4 5Mb
- IEEE 802.4 MAP 10Mb

In addition to the above LAN implementations, three other LAN implementations are available; User Defined Collision, User Defined Token Bus and User Defined Token Ring. Although these three LAN implementations have predefined parameters, they can also be customized by changing any of the basic LAN parameters. The other LAN implementations have fixed parameters which can be viewed, but not altered.

2.2.7 Gateways

A Gateway is a device for multidirectional interconnection between two or more TDs and/or LANs. A NETWORK II.5 Gateway is the generic term for a repeater, router, bridge, gateway, or any type of hardware connecting two or more TDs and/or LANs. Gateways cannot generate traffic, they only forward messages sent to them by Processing Elements or other Gateways. A Gateway is made up of a number of output units equal to the number of TDs/LANs that the Gateway connects. These output units are modeled by standard NETWORK II.5 PEs and Modules automatically written for you when you save your model. Each output unit is dedicated to serve a single TD/LAN. It is responsible for properly requesting the TD/LAN and queueing the messages for that LAN. You may specify a Buffer Capacity for each of the output units of a Gateway.

When a Gateway receives a message, the output unit performs any required reformatting of the message. If the processing component of the output buffer is busy with another message, a message queue builds. If the size of the message queue exceeds the Buffer Capacity of the output unit, a runtime warning will be issued. However, the message will not be lost, but will always be forwarded. If the message arrives over a TD or LAN which has Separate Blocks = **NO**, the message will not be considered for processing and/or retransmission until the entire message is received at the Gateway. If the message arrives over a TD or LAN which has Separate Blocks = **YES**, the message will be considered for processing and/or retransmission block by block, as it arrives.

There is a set of reformatting parameters for processing time and message length for each TD/LAN connected to the Gateway. The processing time is calculated using the equation of the form **Ax + B** where **A**, referred to as the Message Multiplier, is any real number; **x** is the incoming message length (in bits), and **B**, referred to as the Message Additive, is any real number. The processing time is the time for any required translation and message length adjustment between the two TDs/LANs. The retransmitted message length is calculated using a different equation, also of the form **Ax + B**, where **A**, the Processing Multiplier, is any real number; **x** is the incoming message length (in bits); and **B**, the Processing Additive, is any real number. By default, the processing time is zero (**A** = 0, **B** = 0) and the outgoing message length is equal to the incoming message length (**A** = 1, **B** = 0).

The set of reformatting parameters for processing time and message length can be either identical for all output units or set individually. You set these parameters by clicking on the name of the TD/LAN that the output unit serves in the connection list of the Gateway form. There is no reason that the formatting time must be the same no matter what direction a message is going.

2.2.8 Destination Mixes

A Destination Mix is a list of names of Processing Elements, Routes, and Destination Mixes with a percentage associated with each. The percentages must sum to 100%. Each time a Destination Mix is referenced as a destination for a message instruction, a new probability is randomly selected from a uniform distribution. The corresponding destination from the mix is the selected and executed. You use a Destination Mix by giving its name as the destination for a Message Instruction. Because the names of Destination Mixes, PEs and routes must be unique, there will be no confusion.

```
***** DESTINATION MIXES
HARDWARE TYPE = DESTINATION MIX
  NAME = WORKSTATION MIX
    DESTINATIONS ARE ; 40 % ADMIN
    DESTINATIONS ARE ; 17 % FINANCE
    DESTINATIONS ARE ; 10 % GATEWAY TO WAN
    DESTINATIONS ARE ; 33 % PRINT SERVER
```

A Sample Destination Mix

Destination Mixes are recursive. Therefore, the name of a Destination Mix can be included as a "destination" in a Destination Mix, exactly as it could be inserted for a destination of a Message Instruction. To catch infinite loops, a limit of 100 levels of recursion for Destination Mixes has been built into NETWORK II.5. A message instruction attempting to find a destination from a Destination Mix that reaches the 100 level limit will be cancelled and a runtime warning will be issued.

2.3 NETWORK SOFTWARE COMPONENTS

The software components of a system are characterized in NETWORK II.5 as Modules, Instruction Mixes, Macro Instructions and Files. The role and use of each is described below.

2.3.1 Modules

A Module is the specification of a task to be performed by a Processing Element. The Module description consists of four parts:

1. Scheduling conditions
2. Host Processing Element options
3. A list of instructions to execute
4. A list of Module(s) to execute when this Module completes

Each of these parts maps into a stage that every Module goes through in its "life." The stages in a Module's life are:

1. Checking preconditions
2. Requesting a host PE
3. Executing instructions
4. Choosing successors

When a Module checks its preconditions, it is really checking to see if the user-defined scheduling criteria are met. When they are, the Module then queues up for a host PE (i.e., the Processing Element on which it will run). After obtaining a host PE, a Module begins executing by issuing instruction names from its Instruction List to its host Processing Element for execution. After a Module has issued the last instruction in its Instruction

List, it then chooses its successor Modules (if any were specified). Each of these four stages will be described in turn below.

2.3.1.1 Module Preconditions

There are a variety different conditions which may be used to schedule a Module. Every Module must have at least one of these conditions so that NETWORK II.5 knows when to "run" the Module. Sophisticated combinations of these preconditions for the same Module permit modeling real world scheduling criteria.

The different Module preconditions fall into four basic categories:

1. Time-based
2. Awaited
3. Checked once
4. System state

Each category will be described separately.

Time-based preconditions are the Module's Start Time, Stop Time and Iteration Period. Start Time, Stop Time and Iteration Period may be either real numbers or the name of a statistical distribution. If the specified Iteration Period is a Statistical distribution, a new pick from the distribution will be made each time the Module is scheduled to activate. The action that occurs at the start and iteration times is called a *scheduling*, because the Module may not immediately begin execution due to host PE availability or to other preconditions which must still be satisfied.

Start Time and Iteration Period may be specified either alone or in combination for the same Module. Because they interact, three cases are possible.

1. Only the Start Time is specified -- the Module will be scheduled to execute only once, at the given time.
2. Both the Start Time is specified and the Iteration Period is also specified -- the Module will be first scheduled for the given Start Time. Thereafter, the Module will schedule the next copy of this Module after a delay equal to the Iteration Period from the time when the last scheduling occurred.
3. Only the Iteration Period is specified -- a Start Time will be chosen randomly (using a uniform distribution) between 0 (zero) and one Iteration Period. If the Iteration Period is a statistical distribution, the uniform distribution used for Start Time will

use one selection from the Iteration Period statistical distribution as an upper bound. After that, the time between schedulings will be the specified Iteration Period.

If the Stop Time is specified for a module, the Module will continue to schedule executions until the Stop Time is reached. If there are Modules scheduled and queued when the Stop Time occurs, those queued Modules will execute. Once a Module Stop Time is reached, no new copies of the Module will be scheduled for the remainder of the simulation.

Awaited preconditions are the Module's Anded Predecessors and Ored Predecessors. If one Module lists another Module as a successor, it implies that the successor Module has a predecessor precondition. A given Module may have a number of other Module(s) which list it as a . When a Module completes and then notifies its successor(s) that it has completed, that notification is called chaining;. When any Module chains to another Module, the successor Module must determine whether its predecessor preconditions have been met. If a Module is defined as having Ored Predecessors, any Module chaining to it is sufficient to meet the predecessor precondition. If a Module is defined as having Anded Predecessors, then every Module which lists this Module as a successor must chain to this Module before its predecessor precondition is met.

Every new copy of this Module will have its own predecessor conditions which must be met. Therefore, if a Module has three Anded Predecessors, all three must chain to this Module once before this Module can be scheduled. Once the Module is scheduled, it creates a new copy of this Module, which starts by checking its preconditions. All three predecessor Modules must then chain to this new copy before it may be scheduled. Any "extra" chainings predecessor Modules made to the old copy will not be used to satisfy this new copy's Anded Predecessor preconditions.

It is often useful to specify a Module as its own successor, thus allowing a Module to be always ready to perform an action. When specifying a Module as its own successor, do not specify that it has a predecessor, thus avoiding a logical impossibility (Start only after you have completed). A scheduling precondition must still be provided for the Module. Commonly, the Module is given a Start Time of zero to bring it into existence at the start of a simulation. It will then chain to itself to remain active throughout the simulation.

Checked once preconditions are Chain If Semaphore, Chain If File, and Chain If Message Requirements. Chain If conditions are used to select a successor Module based on user defined conditions. Chain If conditions determine whether a Module can be Chained to at the time they are checked. The same Module may Chain If due to file,

Semaphore and message conditions. Chain If conditions are specified for a successor module and apply to all predecessor Modules which call this Module. Modules without predecessors cannot have Chain If preconditions, because this test is made only by a Module attempting to chain to a successor.

If a Module fails its Chain If test, it remains waiting for its predecessor precondition as if the predecessor never completed execution. The Chain If precondition is tested once by the predecessor when it is about to chain to a successor. If all Chain If requirements are met, the chaining occurs and the successor's predecessor condition for the Module that chained will be met. The successor Module will then check its remaining preconditions to see if they are satisfied. If any of the successor's Chain If requirements are not met, the successor Module remains inactive.

A Chain If file precondition allows you to select a Module based on the availability (or nonavailability) of a file either on a particular device or on any device. A Chain If Semaphore precondition; allows you to select a successor based on whether the Semaphore is set or reset or the Semaphore's count is greater than, less than, or equal to, a particular integer value. See Section 2.2.1.6 Semaphore Instructions. for an explanation of Semaphore counts. A Chain If message precondition allows you to select a Module successor based on the presence or absence of a particular message in the received message list of a specific Processing Element.

System state preconditions are Wait For Semaphore Status Requirement, Wait For File Status Requirement, Wait For Message Status Requirement, and Hardware Status Requirement.

After a Module has passed all of its time-based, awaited and checked preconditions, it then begins to check its system state conditions. When checking the system state conditions, a Module behaves as an impartial observer. The Module will remain in an observer state until every system state precondition is simultaneously true. When this happens, the Module then removes any messages it requires from the Received Message List of the host PE and moves to the next state where it waits its Delay (if any) and then queues for its host Processing Element.

Wait For Semaphore Status Requirement preconditions; allow a Module to require that one or more Semaphores have a specific setting (**SET** or **RESET**, or a count greater than, less than, or equal to, an integer. Wait For File Status Requirement preconditions allow a Module to require that a given file exist (or not exist) on a given Storage Device (or any Storage Device). A Wait For File Status may also be based on the count of a file. A Wait For precondition is a delaying precondition, not a scheduling precondition. Therefore, a

Module must have another precondition in addition to the Wait For precondition(s) to be considered for execution. For example, to start a Module every time a particular file was written, the Module could be given a Start Time of zero, and be listed as its own successor. This will meet the NETWORK requirement that a Module have a scheduling precondition (the zero Start Time) and cause a new copy of this Module to be made after the old one completes (because it is a successor to itself). BEWARE: This method requires that the Module, during its lifetime, do something to inhibit its next execution (such as to Read and Destroy the file). Otherwise, an infinite loop will result.

Wait For Message Status Requirements allow a Module to require that one or more messages be received at the Module's host Processing Element. When the Module leaves the checking preconditions state, each of the messages which satisfied the Module's Wait For Message Status Requirements is taken from its host PE's Received Message List. This effectively "uses up" the message. Another message must be received before the requirement would be satisfied again. If two or more Modules had the same message as a precondition, the first Module to execute would "use up" the message and the other Modules would have to wait until this message was received again.

Messages are only allocated to Modules which have all of their preconditions met. If two or more Modules with all other preconditions met were waiting for the same message, the message would be allocated to the highest priority Module. Within equal priority, it would go to the Module that waited the longest for this message. All else being equal, a random choice will be made.

Wild cards are permitted in a Module's Message Status Requirement. Therefore, a Module could wait for one of many different messages with only one wild card Message Status Requirement. See Section 2.3.7 for a complete description of wild cards.

The Wait For Message Status Requirement List is presumed to be an Anded List. This means that every message in the list must be received to satisfy this Module's Message Requirements. Wild cards are commonly used to effectively provide an Ored List, where any one of a group of messages would be sufficient to start this Module.

An additional Module attribute that relates to Wait For Message Status Requirements is the One Message At A Time flag. This flag is used in cases in which you may want only one copy of a Module that has a Wait For Message Status Requirement to exist at a host PE at any given time. When One Message At A Time = **NO** (the default), every time a new copy of the message which will satisfy the Module's Message Status Requirement arrives, a new copy of the Module is created and will contend for the host PE. This may lead to many Modules queueing for the host PE if many copies of the message are

received. If One Message At A Time = **YES**, a new copy of the Module will not be created in response to a message arrival until the currently executing copy of the Module is completed.

A Module may require any number of hardware devices to have a given status as a precondition to Module execution. These requirements specify that a PE, Storage Device, or Transfer Device be **busy** or **idle** before starting this Module. In addition, you may require **Collision** for Transfer Devices with a collision protocol and **Overflow** for a Storage Device or PE (if the PE has a Message List Size specified). If any device specified in the Hardware Status Requirements List does not have the required status, the Module execution will be deferred and retried when the monitored device's status changes. Appearance in this list does not allocate the device to this Module. Instead, it merely inhibits Module execution until the named device attains the requested status.

A Message Status Requirement must be defined in the Message Status Requirement form as one of four types; Global Message, Global Block, Local Message and Local Block. NETWORK II.5 allows you to specify if a message is to be sent across a Transfer Device entirely in one transmission, or in several transmissions with the message divided into blocks. If you want a Module to respond to the receipt of a part of a message, select either the Global Block or Local Block options for the Message Status Requirement of the Module. A Message Status Requirement with either of these options selected may be satisfied when a message block is received. If a message is sent from one PE to another in the form of five blocks, Modules with a Wait For Message Status Requirement of type Global Message or Local Message cannot begin execution until all five blocks of the complete message have arrived. However, a Module with a Wait For Message Status Requirement of type Global Block or Local Block could begin execution immediately after the first block is received.

During a simulation, messages may be sent to the Global Message List. Messages received in the Global Message List may satisfy Message Status Requirements for all Modules on any host PE. If a Message Status Requirement is designated either Local Message or Local Block, messages/blocks received in the Global Message List will cannot be used to satisfy the Message Status Requirement. In this case, only messages/blocks sent directly from one PE to another (or itself) or messages/blocks passed by means of a Route (i.e. - messages in a host PE's Received Message List) can be used to fulfill the Message Status Requirement.

2.3.1.2 Host Processing Element

Once all of the preconditions for a Module have been satisfied, it then waits its specified Delay. A Module will take its required messages (if any) before entering the Delay state and will not re-check its preconditions. When a Module's Delay has expired, it then requests a host PE. When a Module does not have a Delay specified, it requests a host PE immediately after its preconditions are met. To execute, a Module must acquire a host PE.

For Modules with a predecessor, the host PE will always be the host PE used by the predecessor Module. This is why Modules with predecessors may not have an Allowed Processor List or Resident Processor List.

For Modules without a predecessor, the host PE will be selected from the PEs available from among those in the Module's Allowed/Resident Processor List. Every Module which does not have a predecessor must have an Allowed/Resident Processor List defined. In the most common case, the "list" contains a single name. If an Allowed Processor List is specified, the first available PE in the list is used. If a Resident Processor List is specified, one copy of the Module will automatically be created for each PE in the list at the start of simulation. This allows you to define one Module and populate multiple PEs with independently running copies of that Module.

A Module queues up for a host PE only after all of its preconditions are met. Every Module has a priority and an Interruptibility Flag. The priority determines when it will execute relative to other Modules contending for the same Processing Element. A Module will be placed ahead of any other Modules with a lower priority in the queue of Modules waiting for a host Processing Element. A Module of higher priority will interrupt a Module of lower priority already executing in a PE only if the Interruptibility Flag of the executing Module is set to **YES**.

2.3.1.3 Module Execution

After a Module has acquired a host PE, it begins its execution stage. A Module executes by sequentially issuing all of the instructions in its Instruction List by name to its host PE. Each instruction name in the Instruction List contains a Number of Executions specification. The Number of Executions is the number of times to execute an instruction before going on to the next instruction in the Instruction List. Note that the name of an

instruction is the link to the Processing Element. An instruction with the name `MULT` on one PE may require a different number of basic cycles than on another PE. An instruction by the same name on a different PE may even have an entirely different instruction type. This is transparent to the Module.

Instructions can be of any degree of complexity. A user may specify Fast Fourier Transform as an instruction with an execution time of 200 cycles just as easily as a 2 cycle ADD instruction. The timing is, of course, specified in the Processing Element description. Only the names of instructions are given in the Module descriptions. During execution, an attempt to execute an instruction on a Processing Element which does not have that instruction in its Instruction Repertoire will cause a runtime warning message to be generated. The instruction execution will be abandoned and the next instruction will be attempted. The Verify feature of NETWORK will warn you about any Modules that may call an instruction which is not defined on any of its possible host PEs.

An instruction name in a Module's Instruction List may also refer to an Instruction Mix or a Macro Instruction, rather than to an instruction in a PE's Instruction Repertoire. Each of these features will be described in its own section. No special specification is required to indicate that a name represents one of these features. If a host PE has an instruction in its Instruction Repertoire with the same name as an Instruction Mix or Macro Instruction, the Instruction Mix or Macro Instruction will be executed in preference to the host PE's instruction.

A Module may be interrupted in three ways. If a Module with a higher priority requests a PE, and this Module's Interruptibility Flag is set to **YES**, this Module will be interrupted. In addition, for a Module with a Required Semaphore Status, if the required Semaphore status changes after a Module has its preconditions met, the Module will be interrupted. Also, if you specify a Time Slice for a PE, the PE will interrupt the Module if the Time Slice expires and this Module's Interruptibility Flag is set to **YES**. The Interruptibility flag takes precedence over Time Slicing. Statistics on the interruption will be reported in both the Module and the Processing Element reports (see Section 2.2.1 for a full description of what happens during an interrupt).

A Module may monitor the value of a Semaphore during its operation by using a Semaphore Status Requirement based on a Semaphore's state (**SET** or **RESET**), a Semaphore's count, or both. The Module will interrupt itself whenever the Semaphore does not have the required setting. When the Semaphore regains the required setting, the Module will resume according to standard interrupt/resume rules. The Run Until condition is similar to the Run When condition, except that if a Module fails any of its Run Until conditions, the Module is first interrupted (to give up its resources) and then

this copy of the Module is deleted from the simulation. Statistics on this action will be found in the reports.

Run When and Run Until *Semaphore* conditions differ from Wait For and Chain If *Semaphore* conditions, in that they are checked after a Module has its preconditions met. Wait For and Chain If conditions are only checked before a Module's preconditions are met. A Module may have any desired combination of these run, wait and chain conditions (including none). A Run Until *Semaphore* condition has an additional attribute, Completed If Run Cancelled. This attribute, if set to **YES**, means that a cancelled Module is "complete," and therefore chains to its successors (if any) and updates the completed Module statistics. If Completed If Run Cancelled = **NO**, this Module will not chain to successors if cancelled and will not update the completed Module statistics.

A Module may be defined such that more than one copy of it exists in the simulation at the same time. For example, more than one copy of the Module might be queued up for the same PE, or different copies of the same Module might be simultaneously running on different PEs. The "number of copies" specification is called the Concurrent Limit. The default Concurrent Limit is one. This limit includes all copies of this Module, whether executing, queued, or interrupted.

When a Module is scheduled, the Concurrent Limit is compared to the current number of copies of a particular Module, to see whether the limit is exceeded. If the limit is exceeded, the new copy of the Module fails the concurrent limit test. The failed copy of the Module is removed from the simulation, a warning message is issued, and failure statistics are updated. The preconditions of a Module determine when this test is made. For a Module with an Iteration Period, the test is made at the scheduled iteration time, regardless of any other preconditions. For a Module with Message Requirements but without an Iteration Period, the test is performed when all of its Message Requirements are met (regardless of any other preconditions). The failed Module still takes the messages that met its Message Requirements from its potential host PE's Received Message List. For a Module with predecessor Modules, the test is made when any predecessor chains to this Module.

The Concurrent Limit value can be helpful in debugging when Module executions should not overlap: If the system simulated cannot keep up with the workload, warning messages regarding too many copies of the same Module will be produced.

2.3.1.4 Module Completion

After all of the instructions in a Module have executed, the Module is marked as completed and any successor Modules are automatically considered for activation. Successor Modules will queue for the same PE as their predecessor.

A Module may be defined to select one of a list of Modules as a successor upon completion. The potential successor Modules are referred to as Statistical Successors or Anded Successors. The Module that has successors defined in its successor list is called a predecessor Module. Each statistical successor Module has an associated probability of selection. For example:

```
CHOOSE AS SUCCESSOR ;  30.00 % TRANSMIT
CHOOSE AS SUCCESSOR ;  70.00 % RECEIVE
```

At each completion of the predecessor Module, one of the successors is selected according to the specified probabilities, using a uniform distribution. You may specify a random number stream for use by this mechanism.

As an alternative to statistical selection, a Module may have Anded Successors. The Anded Successor list is a list of Modules, all of which are to be selected for activation when this Module completes. Each selection may be subject to a further condition. This condition is a “chain count” of two possible types: Iterate Then Skip Count and Iterate Then Chain Count.

If the Iterate Then Skip Count for a Module is *n*, a successor Module will be scheduled after the execution of the current Module (on the same PE) *n* times. At the $n + 1^{th}$ execution, the chaining attempt to the successor Module will be skipped. This sequence repeats indefinitely. Separate counts are kept for each of a Module's host PEs, to maintain the independence of the individual PE's.

If the Iterate Then Chain Count for a Module is *n*, a chaining attempt to a successor Module will be skipped *n* times and then activated after the $n + 1^{th}$ execution of the current Module (on the same PE). This sequence will then repeat indefinitely. Separate counts are kept for each of the Module's host PEs.

Skip and chain counts are used to control looping through a Module chain: they can cut iteration rates in half (for example). To have a Module always chain to its Anded Successors, select the Always Chain option.

2.3.2 Instruction Mixes

Instruction Mixes are "pseudo-instructions" included in the Instruction List of a Module. An Instruction Mix is a list of names of instructions, Instruction Mixes and Macro Instructions with a percentage associated with each. The percentages must sum to 100%. Each time an Instruction Mix is referenced, a new probability is randomly selected from a uniform distribution. The corresponding instruction from the mix is then selected and executed. The name of the Instruction Mix appears in a Module's Instruction List, exactly like the name of an instruction. NETWORK determines whether a name in an Instruction List specifies an actual instruction or an Instruction Mix at run time. Should a PE have an instruction of the same name as an Instruction Mix, the Instruction Mix will be executed.

An example of an Instruction Mix is:

```
NAME = MIX 1
   INSTRUCTIONS ARE ;  10.000 % PROCESSING INSTRUCTION
   INSTRUCTIONS ARE ;  10.000 % MESSAGE INSTRUCTION
   INSTRUCTIONS ARE ;  80.000 % MIX 2
   STREAM ; 22
```

The stream parameter is the number (1 through 999) of the random number stream to be used by the uniform distribution to pick the successor probability. In the absence of any user specification, unique streams will be automatically assigned to each Instruction Mix, assuring statistical independence. Further details on the use of random number streams are given in Section 2.3.8 Statistical Distributions.

Instruction Mixes are recursive. Therefore, the name of an Instruction Mix can be included as an "instruction" in an Instruction Mix, exactly as it could be inserted in a Module's Instruction List. To limit infinite loops, a limit of 10 levels of recursion for Instruction Mixes has been built into NETWORK II.5. A Module attempting to run an Instruction Mix that reaches the 10 level limit will stop executing and is considered completed. A runtime warning will be issued when this occurs.

2.3.3 Macro Instructions

Macro Instructions are collections of instructions which are referenced by a single name. A Macro Instruction can be thought of as an Instruction List that is not tied to any specific Module. Macro Instructions allow recursion; they can be made up of actual PE instructions, Instruction Mixes, or other Macro Instructions. To prevent infinite loops (if,

for example, a Macro Instruction directly or indirectly referenced itself), a limit of 10 levels of recursion for Macro Instructions has been built into NETWORK II.5. A Module attempting to run a Macro Instruction after 10 levels will stop execution and then is marked as complete, at which point a runtime warning will be issued.

The user is cautioned that the use of Macro Instructions can be very deceptive. It is quite possible to have a Module with only a "single instruction" that is a Macro Instruction that expands into literally hundreds of instructions to execute. Take, for example, a Module that has a Macro Instruction which is made up of 10 instructions which are really Macro Instructions, which are themselves made up of 10 PE instructions. This Module would execute 10 x 10 = 100 instructions before it is done!

Macro Instructions are useful in reducing the amount of work required to produce a simulation where many Modules have sections of their instruction lists in common. They are also valuable in top-down refinement of a simulation. For example, a Write Instruction in a preliminary simulation could be defined as `WRITE 600 BITS TO MEM 1`, but later refined into a series of 3 instructions that `SET STATUS FLAG`, `WRITE 600 BITS TO MEM 1` and `RESET STATUS FLAG` with one simple change. Macro Instructions are also used to create loops within a Module. If you have a series of instructions that you would like to loop through **n** times, put the instructions in a Macro Instruction and call it with a number of executions of **n**.

An example of a Macro Instruction is:

```
SOFTWARE TYPE = MACRO INSTRUCTION
    NAME = MACRO 1
        NUMBER OF INSTRUCTIONS ; UNIFORM BETWEEN 1 AND 10
          INSTRUCTION NAME ; PROCESSING INSTRUCTION
        NUMBER OF INSTRUCTIONS ; 1
          INSTRUCTION NAME ; SET SEMAPHORE 1
```

Notice the use of the name of a Statistical distribution instead of a number for the number of times to execute the first instruction.

2.3.4 Files

A File represents the organized storage of information in a Storage Device. It may be created by NETWORK when a model is being constructed, or a file may be dynamically created during a simulation by a PE executing a Write Instruction. A File may be the object of Read Instructions as well. Files defined by NETWORK (within the SD form)

are specified by name, count, size, where they initially reside, and whether they are Read Only Files.

A Read Instruction or a Write Instruction will search for a File on a given Storage Device by name, and also by count, if a count has been specified in the instruction. If a File has a matching name, but not a matching count, then the File cannot be used by the instruction. When a Write Instruction attempts to access a File which is not currently resident in the referenced Storage Device, the file is created with a size equal to the number of bits written to it. It remains resident until destroyed by a Read Instruction with the Erase File option selected or until the number of bits in the file is reduced to zero by one or more Read Instructions with Decrement File selected. If you write to an existing file, there are three options. If the Replace File option is selected, on completion of the instruction the file will contain the exact number of bits that the instruction writes. If the Append to File option is selected, the number of bits written to the file is added to the number of bits the file already contains. If the Do Not Modify File option is selected, the file size remains unchanged. You might use the Do Not Modify File selection to model updating records in a database. If you write more bits to a File than it currently contains and you selected Do Not Modify File, the File size is set to the actual number of bits written.

If a Read Instruction attempts to access a file which is not currently resident in the Storage Device, a runtime warning message is generated and the instruction continues as if the file exists.

General Storage is a special type of file. It has special characteristics and was intended to be used for modeling a Storage Device where explicitly defined files are not used. General Storage can best be considered a "bit bucket" from which anyone may pour or draw bits (up to the number of bits the bucket contains). The special characteristic of General Storage is that more than one user may access it at one time. Other than this special characteristic, General Storage is treated like any other file. It may be created by NETWORK or a Write Instruction and will be deleted from a Storage Device whenever its size becomes zero. General Storage may coexist with other named files within the same Storage Device.

2.3.5 Messages

Messages are commonly used to schedule and coordinate tasks. A message is sent from one Processing Element to another over a Transfer Device as the result of a Message Instruction. When a message is received, it is always filed in the receiving PE's Received Message List. Once in that list, Modules with that message name as a Wait For Message

Status Requirement will contend for that message. The highest priority Module that has all of its preconditions met will be assigned the message. In the event of a tie, the Module that waited the longest will get the message; if there is a time tie, a random assignment will be made.

There are two exceptions to the above message scenario. When a PE sends a message to itself, it does not use a Transfer Device and the “transfer” is completed in zero simulated time. The message sent still goes into the PE's Received Message List, just like any other message. The other exception occurs when a message is sent to the Global Message List. The Global Message List is essentially a system wide Received Message List from which any PE can take a message. Because it is a modeling convenience, not a real PE, no Transfer Device is required and the transfer takes zero simulated time.

A message can be viewed as a token. One message token can satisfy no more than one Module's Wait For Message Status Requirement precondition. Multiple copies of the same message token may reside in the same Received Message List (or different ones) until taken by a Module.

A message is defined implicitly through the definition of Message Instructions. In the instruction, a message is described by the Message Text (which defaults to the name of the issuing instruction), and a length in bits. The length is used in determining message transmission times over the various Transfer Devices. A Message Instruction has a Message Text separate from the Message Instruction name to allow more than one instruction to send the same message, and to allow the same instruction name to send different messages on different PE's.

For example, a Module issuing the message THIS PE FAILING might want to send a message that indicates who is failing. However, the Module that issues this instruction might be dynamically assigned to different PE's. Having an instruction of the same name on different Pes send a different message text from each PE (PE1 IS FAILING from PE1, PE2 IS FAILING from PE2, etc.) solves this problem. The Message Instruction's Message Text is the name that appears in the receiving PE's Received Message List and is the name that will be compared to a Module's Message Status Requirements.

2.3.6 Semaphores

Semaphores are also used to schedule and coordinate tasks. Semaphores are global flags with both a binary value (**SET** or **RESET**) and an integer count. They may be modified

by any PE in the simulation and examined by any Module in the simulation. Modifying a Semaphore does not take any simulation time.

Every Semaphore has a count associated with it. You may increment, decrement or assign a Semaphore's count by using a Semaphore Instruction. This count can be used by a Module's Wait For and Chain If preconditions. See Section 2.3.1.1 Module Preconditions for further details on the use of Semaphore counters.

Scheduling a task by a Semaphore differs from scheduling by message in that:

1. Scheduling is instantaneous (no transfer time).
2. All Modules monitoring the Semaphore will immediately take action (messages start only one Module).
3. Run When and Run Until *Semaphore* conditions may stop an executing Module (messages are strictly preconditions).
4. Semaphore actions may be timed.

Simulations which require message *and* Semaphore capabilities can easily use both. Sending a message (to properly load a Transfer Device) and then setting a Semaphore (to schedule many Modules) is very reasonable. Also, setting a Semaphore when a message is sent and resetting it when it is received is a common way to gather statistics on message traffic (see Section 6.4.6 for an explanation of how Semaphores are used to measure response time).

From the top level of NETWORK, you can select a semaphore using the Define/Semaphore command to edit its parameters in the Semaphore form. On this form you can give the semaphore a comment and set the initial status, initial count, Maximum Pending Responses and the Inhibit Response Time flag.

Every semaphore has an initial status and an initial count. The default is an initial status of **RESET** and an initial count of **zero**. Setting Inhibit Response Time to **YES** for Semaphores which are not used to measure response times will speed the execution of your model and reduce the clutter in the Semaphore Report. The Maximum Pending Responses value allows you to define boundaries for a Semaphore response computation. Every time you start a response by SETing a Semaphore, a few words of storage are allocated to store the attributes of this pending response. Every time you complete a response by RESETing the Semaphore, that storage is released. If you start thousands of responses without completing them, you will lose valuable memory.

A default of 999 maximum pending responses is presumed to warn users of potentially memory wasting Semaphores. If the specified maximum count is exceeded, a run time warning is issued and the new response is not started. All other features of a Semaphore instruction (SET/RESET/TOGGLE and INCREMENT/DECREMENT/ASSIGN COUNT) will work normally even in the event of a maximum pending response error.

Semaphore counts can be assigned from a statistical distribution. This will permit you to assign a value to a Semaphore count from a random distribution. Then you can use that selected value multiple times through the use of the Semaphore linear distribution.

2.3.7 Wild Cards

Wild cards are used to allow one text string to match more than one name. Wild cards can be used in Message Status Requirements and as Destinations for Message Instructions. For example, using a Wait For Message Status Requirement of NEW DATA* would allow a Module to be started by any message which starts with the text string "NEW DATA".

Wild cards in NETWORK II.5 are position dependent. Wild cards are indicated by the * character. The * may appear at the beginning, the end, or both sides of a text string. If the * is at the beginning of the text, the wild card will match anything that ends in the given text. If the * is at the end of the text, the wild card will match any text that begins with the given text. If the * is at both the beginning and the end, the wild card text will match any text string that contains, in whole or in part, the wild card text. Spaces between the * and the adjoining text will be ignored. For example,

*NEW DATA

Matches	Does not match
NEW DATA	NEWDATA
MORE NEW DATA	OLD DATA
SOMENEW DATA	NEW DATA COLLECTED

NEW DATA*

Matches	Does not match
NEW DATA	NEWDATA
NEW DATA FOUND	OLD DATA
NEW DATAHERE	COLLECTED NEW DATA

```
*NEW DATA*
```

Matches	Does not match
NEW DATA	NEWDATA
MORE NEW DATA	OLD DATA
SOMENEW DATAHERE	NEW COLLECTED DATA

2.3.8 Statistical Distributions

Many applications require choosing a value at run time from a user specified statistical distribution. Most NETWORK II.5 specifications will accept either a number or the name of a Statistical Distribution Function (SDF). NETWORK will indicate this option by offering in a dialog box all known SDF names and a prompt that allows you to enter a number or a new distribution name. Statistical Distributions are referenced by their user-defined name. The parameters of the distribution are defined separately at the top level of NETWORK.

Each Statistical Distribution Function has a name, an optional comment, and a distribution type (normal, beta, etc.). After you have selected a type, you will be able to access the parameter list containing a maximum of five attributes and possibly a stream to be used by the SDF.

NETWORK II.5 supports the following different types of distributions with the following attributes:

DISTRIBUTION	ATTRIBUTES	
Beta	K1 Lower Bound	K2 Upper Bound
Constant	Value	
Erlang	Mean	Variance
Exponential	Mean Lower Bound	 Upper Bound
File Count	A Lower Bound	B Upper Bound
File Linear	A Lower Bound	B Upper Bound
Gamma	Mean Lower Bound	K Upper Bound
IEEE BACKOFF	Slot Time	Offset

	Retry Limit Stream	Limit Delay
Key Linear	A Upper Bound	B Lower Bound
Log Normal	Standard Deviation Lower Bound	Mean Upper Bound
Message Linear	A Lower Bound	B Upper Bound
Message Count	A Lower Bound	B Upper Bound
Normal	Standard Deviation Lower Bound	Mean Upper Bound
Pattern	Table	
Pattern From File	File Name	
Random Linear	Table	
Random Step	Table	
Semaphore Linear	A Lower Bound Semaphore	B Upper Bound
Triangle	Mean Lower Bound	 Upper Bound
Uniform	Lower Bound	Upper Bound

Table of Statistical Distributions

The parameters for these distributions are in accordance with standard mathematical usage. The optional upper and lower bounds are extensions which allow bounding many of the distributions by truncation.

The Random Linear and Random Step Distributions allow defining a table of values selected based on a percentage from a Distribution. In the case of a Random Step Distribution, only those values given in the table will be returned. A Random Linear Distribution will perform a linear interpolation between the two nearest data points to determine a value.

A Pattern Distribution is a table of values. Every time a reference is made to this distribution, the next element in the table is returned. When the end of the table is

reached, the pattern is restarted at the first table element. Pattern Distributions may consist of one to 32000 elements.

A Pattern From File Distribution is a table of values with the table contained in a file external to the system description. This SDF behaves much like the Pattern SDF. Values are taken sequentially from the table of values contained in a file.

The Constant Distribution is obviously not a random distribution. This distribution type allows you to define named constants through the standard statistical distribution syntax.

The Key Linear, Message Linear, Semaphore Linear, File Linear Distribution, File Count and Message Count distributions perform a computation of the form Ax + B where x is chosen based on the type of distribution. For Message Linear Distributions, x is the number of bits in the "original" message that started the Module. (See Section 2.2.1.6 Message Instructions for how to determine which is the "original" message.) For Key Linear Distributions, x is the key value of the PE which is currently using this device. For File Linear Distributions, x is the number of bits in the "original" file (see Section 2.2.1.5 Read/Write Instructions for an explanation of which is the "original" file). For Semaphore Linear distributions, x is the count of the named Semaphore. For Message Count Distributions, x is the count of the "original" message. For File Count Distributions, x is the count of the "original" file.

The Message Linear Distribution may be used to base the Number of Cycles of a Processing Instruction on the size of the incoming message, to change the length of a message being retransmitted, and so on. The File Linear Distribution can be used to send a message of the same size as a file just read.

The IEEE Backoff Distribution is only acceptable as the Retry Interval for a Transfer Device with a Collision Protocol. As such, it is described in Section 2.2.2.5 Collision Protocol.

All random distributions in NETWORK II.5 have a random number stream attribute. The stream is an integer value between 1 and 999. If omitted, NETWORK automatically assigns a unique random number stream to each distribution. The stream is actually an index into an array of random number seeds, which are integers. An index is used instead of an actual seed value to preserve the machine independence of a NETWORK II.5 simulation. Using unique random number streams preserves the statistical independence of the distributions in a model. For instance, independence allows changing the iteration distribution of one Module without affecting the iteration periods statistically chosen for any other Module.

The user is given control over the starting random number seeds to be used so that the effect of different random number streams on the simulation can be studied. If a simulation contained 100 different random distributions, manually editing the simulation data file to alter them would be tedious.

At the top level of NETWORK, use the Define/Controls command to get to the form that allows you to define parameters which apply to all statistical distributions. On this form, you can set the values of Antithetic Variate, Randomizer and Minimize Random Seed Array.

The Randomizer and Antithetic Variate features are offered to make it easy to change every distribution's starting condition. The Randomizer is an integer between 0 and 9. It merely specifies which of 10 random number seed arrays is to be used by the simulation. The Antithetic Variate is either **YES** or **NO**. The Antithetic Variate takes advantage of the principle that all distributions can be derived by one or more selections from the uniform distribution on the interval 0 to 1. All of the distributions in NETWORK II.5 which are really statistical (i.e., not Pattern, Linear, etc.) are derived in this way. If the Antithetic Variate is **YES**, it subtracts the value returned by the uniform distribution from 1 and uses that value to compute a pick from a distribution. This may convert "lucky" to "unlucky," or vice versa; however, note that 1 - .5 = .5, so "middle of the road" choices remain unchanged.

Both the Randomizer and the Antithetic Variate are useful in studying the effect of the randomness on a particular simulation run. If changing one or both of these attributes produces a dramatic change in the final results, a longer simulation time is probably desirable. Long simulations will smooth out the perturbations caused by "luck" when drawing from a Statistical Distribution.

An additional global control, Minimize Random Seed Array, is offered. Minimize Random Seed Array is **YES** or **NO**. If **NO**, automatically assigned seeds are assigned starting at one greater than the largest seed value specified by the user. In addition, the last seed value is used for random start time generation for Modules with an iteration period but no start time. If **YES**, seed 1 is reserved for the distribution used by Modules with an iteration period but no start time. The remaining automatically defined seeds try to use the seeds not specified by the user, starting with seed 2. The most significant difference you will notice occurs when you add a SDF. If Minimize Random Seed Array is **NO**, the seed used to generate random start time changes, so the start times change. If **YES**, seed 1 is still used, and all start times remain the same. The default for Minimize

Random Seed Array is **YES**. This option is provided for compatibility with older releases, where Minimize Random Seed Array was always **NO**.

2.3.9 Global Controls

There are three categories of Global Controls: Run Logic Options, Statistical Options, and Miscellaneous Options. All global flags are defined at the top level of NETWORK with the Define/Controls command.

Run Logic Options

Iterate By Priority
Maximum Message Delivery Combinations
Maximum Module Messages

The Iterate By Priority Flag is used to control the situation in which copies of one or more Modules scheduled by an Iteration Period are scheduled to execute at the same time. If Iterate By Priority = **NO**, the Module with the longest Iteration Period will run next. No consideration of the priority of the Modules is made. Sometimes a lower priority Module will run in preference one with a higher priority, because it was scheduled first. If Iterate By Priority = **YES**, all copies of Modules scheduled by an iteration period for execution at the same time will execute in priority order. This results in a slightly longer run time, but is useful in modeling a smart scheduler. The default for Iterate By Priority is **NO**.

Maximum Message Delivery Combinations can be set to limit the size of the Message Delivery Report. By default, the Message Delivery report includes statistics on every combination of message, source and destination that occurred during the simulation. However, this can sometimes result in a very large report. Presume that you have a 100 node network where any station might send a message to any other station. If there is only one message type, that could lead to 1 * 100 * 99 = 9,900 combinations. Printing a maximum of 12 combinations per page, the report would require 9,900 / 12 = 825 pages. You can limit this by either turning reports off for specific messages or by setting Maximum Message Delivery Combinations to some reasonable value. Any combinations that occur after the maximum is reached will not be reported. This also saves computation time and memory space. The default Maximum Message Delivery Combinations is 999.

Maximum Module Messages is provided mostly as a debugging tool. By default, a module gives its messages to its successor unless the Inhibit Message Inheritance Flag of

the successor = **YES**. If you have a Module in a chain of Modules that loops back, it is possible that this inherited list will continue to grow without bounds. By default, Maximum Module Messages is set to 999. This means that if a Module is given more than 999 messages by its predecessor, a warning message will be issued and only the first 999 will be taken. If you would like to catch this condition sooner, you could set the Maximum Module Messages to a lower value.

Statistical Options

Antithetic Variate
Randomizer
Minimize Random Seed Array

These options are described in detail in Section 2.3.8 Statistical Distributions.

Miscellaneous Options

Autosave
Full Verify
Batch Run Simulation

NETWORK has the ability to automatically save your work at fixed time intervals. If Autosave is set to `N` (a positive, non zero integer), your work will be automatically saved every `N` minutes. The default Autosave is zero, which turns off the Autosave feature.

The Full Verify flag can provide more information when running a NETWORK Verify. The standard Verify only reports errors or warnings with a high likelihood of being an error. If Full Verify = **YES**, less significant warnings and notes will be included in the Verify report. The additional messages may help you track down a problem, but are generally not presented because they are often warnings and notes about situations that will not preclude running the simulation. The default is Full Verify = **NO**.

The Batch Run Simulation flag is useful for those systems on which you can build a script to execute a NETWORK II.5 simulation as a batch job. If Batch Run Simulation = **YES**, interactive prompts for a simulation are not provided. TEXTWORK, the text-based simulation provided on certain platforms, will use the run parameters saved within a model when Batch Run Simulation = **YES**. If Batch Run Simulation = **NO**, the default, TEXTWORK will ask for run parameters before starting a simulation. SIMWORK, the graphics-based NETWORK II.5 simulation engine, is never run as a batch job. Complete descriptions of SIMWORK, TEXTWORK and all run parameters are given in Chapter 4.

2.4 SIMULATION PARAMETERS

The preceding simulation building blocks have been described qualitatively in order to give an overview of all of the elements of a NETWORK II.5 simulation. The interactive input processor, NETWORK, provides a graphic screen-based input environment for describing the system to simulate. The method of entering building block descriptions is covered in Chapter 3.

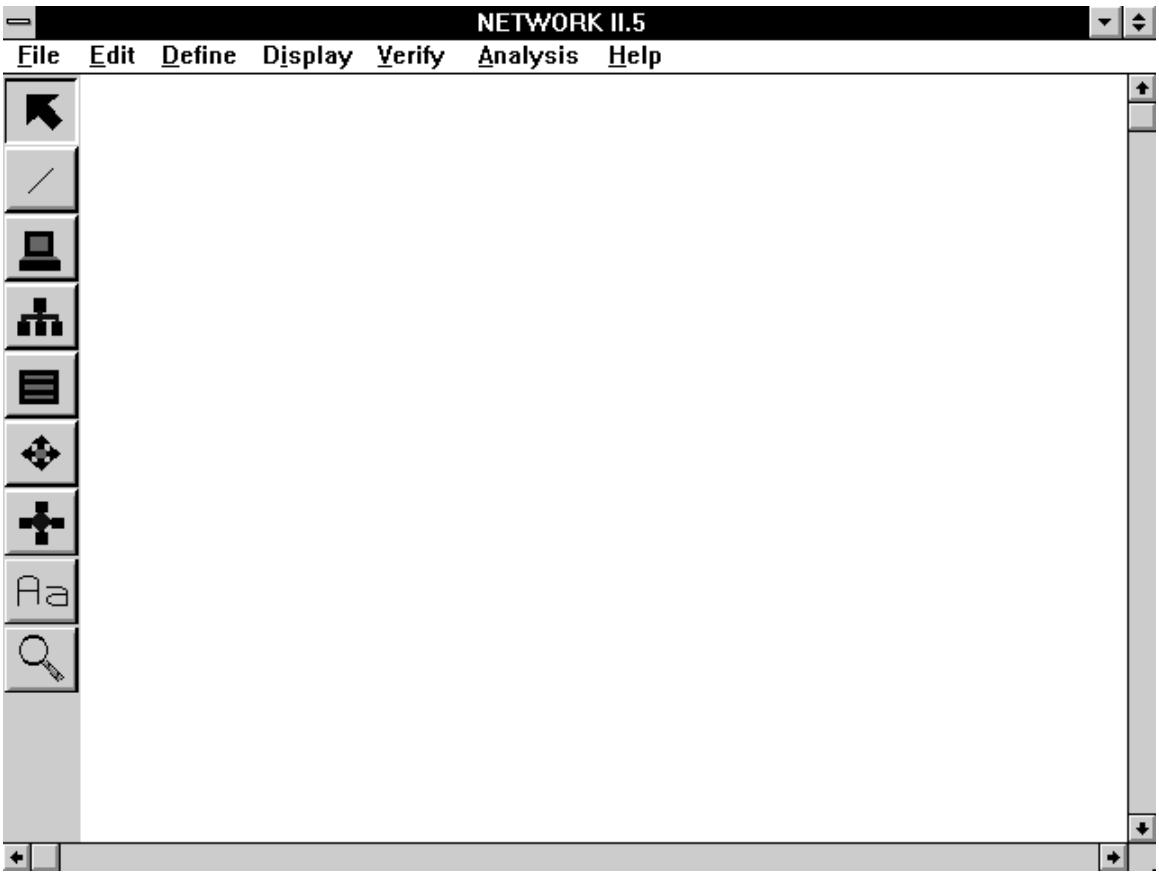
After this step is complete, the only additional inputs are the simulation control parameters relating to the run length (simulated time) and reporting options. A description of how to run a NETWORK II.5 simulation and a complete discussion of these options are given in Chapter 4.

Some consideration should be given to the selection of run length versus the smallest time increments. For example, if the Basic Cycle Time is one microsecond, there may be excessive activity in a one second simulation period (and a correspondingly excessive expense in computer use). However, it may be reasonable to "upgrade" the basic instructions to larger units. For example, if Module descriptions include executions of 100 ADDs, followed by 100 MULTs, it might be more reasonable (and equally accurate) to define a LARGE ADD instruction which represents the time of 100 ADD instructions but executes as a single instruction. Since the only effect of a processing instruction is to keep the processor busy for a period of time, the effect is the same but the simulator will complete the simulation an order of magnitude faster. Further information on how long to run a NETWORK II.5 simulation and efficiency considerations are included in Chapter 4. A sample problem is presented in Chapter 8.

3. THE NETWORK II.5 GRAPHICAL USER INTERFACE

3.1 INTRODUCTION

NETWORK is the graphical user interface to NETWORK II.5 that is used to create, modify, and analyze a model. The Network Description File (.NET) that NETWORK produces serves as an input to the simulation program.



The NETWORK II.5 User Interface

3.1.1 OVERVIEW

The primary focus of this chapter is to describe how to create, edit, and manipulate a model. A detailed description is given on how to use the menubar, toolbar and associated forms to create model components and edit their attributes. This chapter does not focus

on the underlying modeling purposes of these attributes, these topics are covered in Chapter 2. The major data and graphics editing functions found in NETWORK will be discussed in this chapter. The other major capabilities provided by NETWORK II.5 are described in the following chapters: running a simulation - Chapter 4, animation of a model - Chapter 6, plotting simulation results - Chapter 7, using external traffic files in a simulation - Chapter 10, and file conversion - Chapter 11.

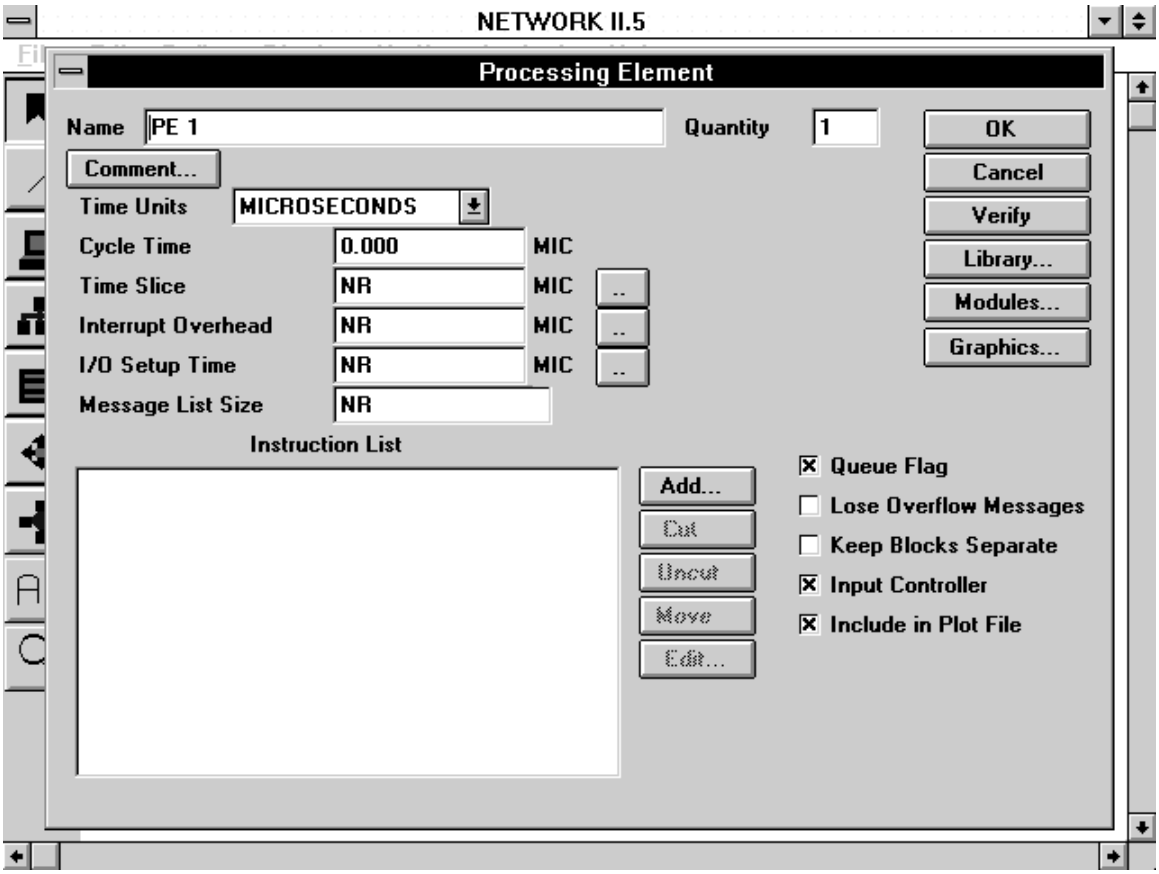
NETWORK is started by double-clicking on the NETWORK program icon, or by typing NETWORK at the command prompt, depending on the computer and windowing system in use. Please refer to the appropriate appendix for further information.

3.1.2 The NETWORK Display

When NETWORK is started, an introductory form is displayed followed by the presentation of the main display. The display is the window in which a graphic layout of a model is developed and the various forms of NETWORK are displayed. At the top of the display is a menubar. Selecting an option on the menubar will present a menu from which related commands may be selected. A complete description of all the menubar menus and commands is found in section 3.4.

On the left side of the NETWORK display is a vertical toolbar from which other commands may be invoked by selecting a toolbar button. The NETWORK toolbar is described in section 3.2.1. Scrollbars are positioned at the bottom and right edges of the NETWORK display to enable you to move the display to a different area of the current graphical layout. The use of these scrollbars is described in section 3.3.1.

3.1.3 Working With NETWORK's Forms



The PE Form

Many of the tasks involved in model building are performed in NETWORK by the presentation of various forms in which data may be entered or edited and other options may be selected. These forms range from the very simple, requiring a single choice, to more complex forms with multiple fields for data entry, check boxes, list boxes and radio buttons. The forms displayed by NETWORK share common features which are to be manipulated according to the same conventions for all of the forms.

Dialog Boxes

Dialog boxes define the background and borders of the forms used in describing the model and controlling NETWORK operations. A dialog box appears as a solid-colored rectangular area on the screen upon which the options of a form are arrayed. When a dialog box is displayed, fill in the parameters contained in the dialog box, and click on a control button (usually OK, CANCEL or RETURN) to exit.

When a dialog box is displayed, all other fields outside the dialog box are nonselectable. This includes the menubar, the palette, display icons, and all other underlying dialog

boxes. Dialog boxes may be displayed on top of other dialog boxes, in which case, the most recently displayed dialog box is active.

Scrolling List Boxes

List Boxes display a list of text items (usually names) or numeric values. Moving through the list box is accomplished by using the scrollbar on the right hand side of the list box. The list boxes provide the means for selecting an item from a list, and in some cases, performing additional operations on the item selected.

There are several ways to add an item to a list box, depending upon the form containing the list box. An ADD button may appear next to the list box for this purpose. Click on the ADD button, enter the new name in the resulting dialog box, and click on OK. The new entry will then be added to the list box. A text box by which a new list entry may be added may also appear above a list box. Click on the text box, add a new list item, and press the Enter/Return key. The new item will then be added to the list.

Entries in the list box may be selected by clicking on the appropriate line of the list, which will highlight the line. In most cases, selecting a list item will activate buttons associated with the list, such as the EDIT, CUT, and MOVE buttons. These buttons will allow you to edit a list item (EDIT - display an attributes form for a list item), or edit the list itself (MOVE - change an items position in the list), or do both (CUT - remove an item from the list and from the model). To deselect a list item that has been highlighted, simply click on it again and it is no longer selected/highlighted. To retrieve an entry which had been previously cut, click on the UNCUT button that appears next to the list box. If no items appropriate to the list had been deleted, the UNCUT button will remain deactivated, it cannot be selected and appears "grayed-out."

Buttons

Buttons are primarily used to transfer control from one form to another, or to initiate an operation. Examples of buttons include the OK, CANCEL and RETURN buttons, or the Set Button (double-dots). Selecting OK means that all of the values appearing on the form have been accepted. If CANCEL is selected, any changes made to values on the form will be rejected. The primary function of the Set Button is to enter the name of a Statistical Distribution Function (SDF) in an attribute field, e.g. to enter a distribution for the Iteration Period of a Module. The Set Button may also be used to display an additional form in which values may be presented and edited.

Text Boxes, Value Boxes

Text and value boxes display current text and numeric attributes and allow the entry of new text or values. In many cases, the initial text or value displayed is NR ("No Response") to indicate that the default value is to be used and no user selected text or value has been entered. A new value can be entered directly by clicking within the box and changing the text/number displayed there.

Combo Boxes

A combo box is used to make a selection from a fixed list of choices, such as setting time units. Selecting the down arrow on the right side of the combo box displays the list of choices and selecting a list item will remove the list and put the selected list item in the combo box.

Check Boxes

Check boxes represent a YES/NO value. A check mark in the box is equivalent to a YES value, while no check mark is equivalent to a NO value.

Radio Buttons

Radio buttons are used in situations where only one option from a series of options is valid at a given time. Selecting a radio button from a radio button group will deselect all of the other radio buttons in the group. The selected radio button is represented as a filled circle while deselected radio buttons are shown as empty circles

3.1.4 Chapter Organization

Section 3.2 explains the basics of creating a model using NETWORK's menubar, toolbar, and forms. Section 3.3 describes how to manipulate the graphical display. Section 3.4 provides a reference for the menubar pulldown items. Section 3.5 describes

NETWORK's attribute forms. Section 3.6 describes the Library function of NETWORK that is available for several NETWORK II.5 data types.

3.2 BASICS OF CREATING A MODEL

The basics in building a model (system description) are as follows:

1. Establish the network topology by creating PEs, SDs, Gateways, TDs, and LANs. Connect the PEs, SDs, and Gateways to the TDs and LANs as desired.

2. Define the network operation by creating Modules, instructions, statistical distribution functions, Semaphores, Routes, Echo PE Lists, Destination Mixes, Instruction Mixes, and Macro Instructions and by setting the attributes of the PEs, SDs, Gateways, TDs, and LANs.

3. Set global model attributes via the Define/Controls commands.

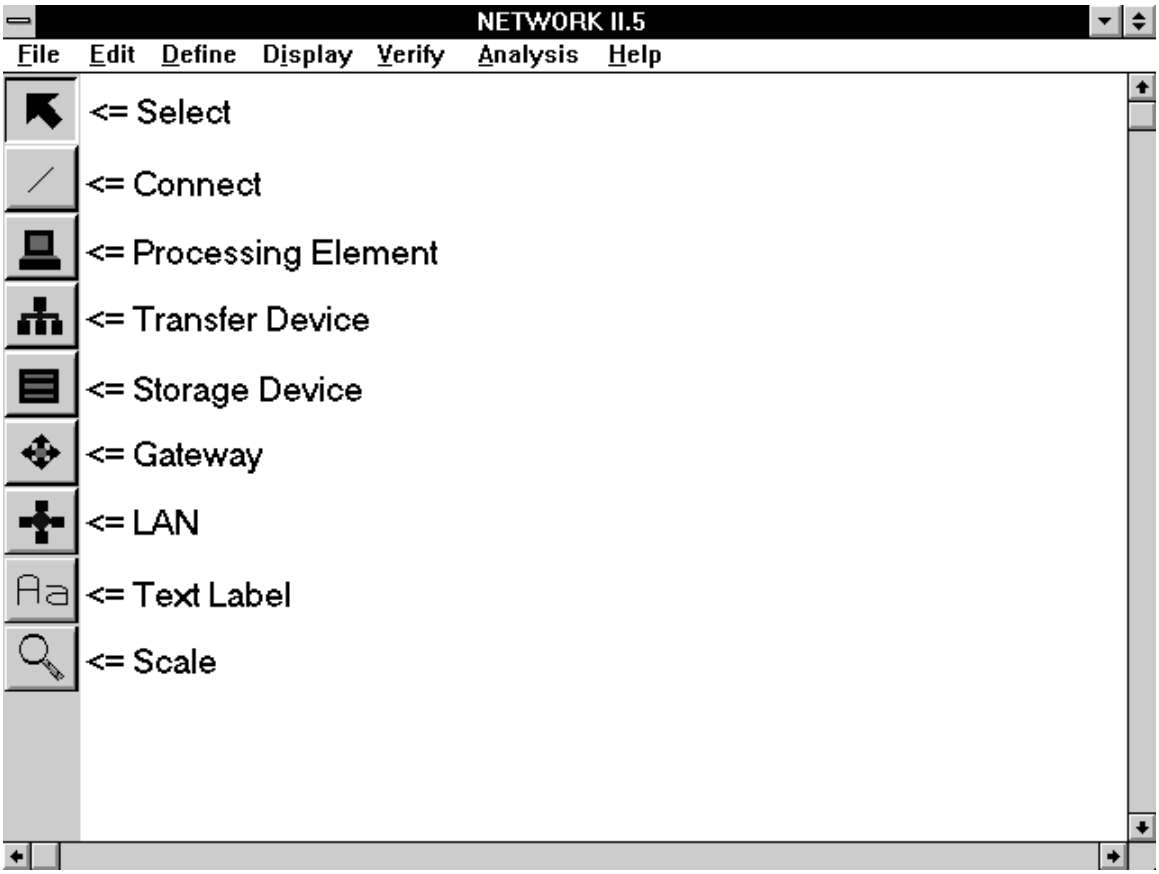
4. Verify the model by selecting Verify from the menubar.

5. Set up the run parameters with the Analysis/Start Simulation command.

6. Save the model by clicking on the File/Save command.

3.2.1 Defining the Network Topology

The NETWORK toolbar on the left side of the display contains buttons for creating and manipulating the network topology and display. The toolbar buttons used in creating the network topology include the Connect Button, Processing Element Button, Transfer Device Button, LAN Button, Storage Device Button, and the Gateway Button. The Select Button is used to select building blocks for editing and by default is the active button. The Label Button allows you to position text labels throughout the display. The Scale Button is used to change the magnification of the display.



The NETWORK Toolbar

To create a new hardware structure (PE, TD, SD, LAN, Gateway), select the appropriate toolbar button by clicking on it. A new building block icon is automatically created and the cursor will be displayed as the building block's icon. Position the icon at the desired location, and click on the mouse once again to anchor it.

Now the PEs, SDs, and Gateways can be connected to the TDs and LANs. PEs, SDs, and Gateways may not be connected directly to each other (i.e., no direct PE to PE, PE to SD, PE to Gateway connections are allowed). PEs, SDs, and gateways may only connect to each other through TDs and LANs. Similarly, no direct TD/LAN to TD/LAN connections are permitted. TDs/LANs may only be connected to other TDs/LANs through PEs and Gateways.

To create a connection, click on the toolbar's Connect Button. Position the cursor on a TD or LAN, and click the mouse. This anchors the first connection point to the TD/LAN. Next, move the mouse to a PE, SD or Gateway and click the mouse again. A connection has been established. Continue making connections in this manner until the network topology has been established. The order in which a connection is created is

important. The first connection point must be on a TD or LAN, with the second point on a PE, SD, or Gateway.

When the toolbar's Connect Button has been selected, you may create as many connections as you desire. After you have finished creating connections, deselect the Connect Button by clicking on the display background. If you wish to redraw a connection between to devices, make a new connection between them and the old connection between the two devices will be removed.

3.2.2 Defining Network Operations

In order to define the network operations, you must specify the characteristics of each PE, SD, Gateway, TD, LAN, and Module in your model. For a more complex model, you may need to create Semaphores, SDFs, Instruction Mixes, Destination Mixes, Routes, or Macro Instructions.

When a new hardware data structure is created, it is given a default set of attributes (name, icon, model parameters). To add meaning to these structures, you must edit the structure's attributes so that they reflect the system under consideration.

To edit a hardware structure's attributes, double click on the hardware structure's icon. The hardware structure's attribute form will then be displayed. Make the desired changes and click on OK to save the changes to the structure and remove the form. Alternatively, you may access the hardware structure's attribute form by selecting Define from the menubar, selecting the appropriate hardware type from the menu that appears, selecting the name of the specific hardware structure from the list of names, and then selecting the EDIT button.

To create a non-graphical building block, the software data structures, select Define from the menubar and then the appropriate entity type. When the name list for the entity type selected appears, enter a new name for the building block by selecting the ADD button. The name will be added to the list and a new building block has been created. To edit the building block's attributes, select the name in the list and then select the EDIT button which will be enabled as a result of highlighting a list item. You may create and edit Modules, statistical distribution functions, Semaphores, Routes, Echo PE Lists, Destination Mixes, Instruction Mixes, Macro Instructions, and messages in this manner.

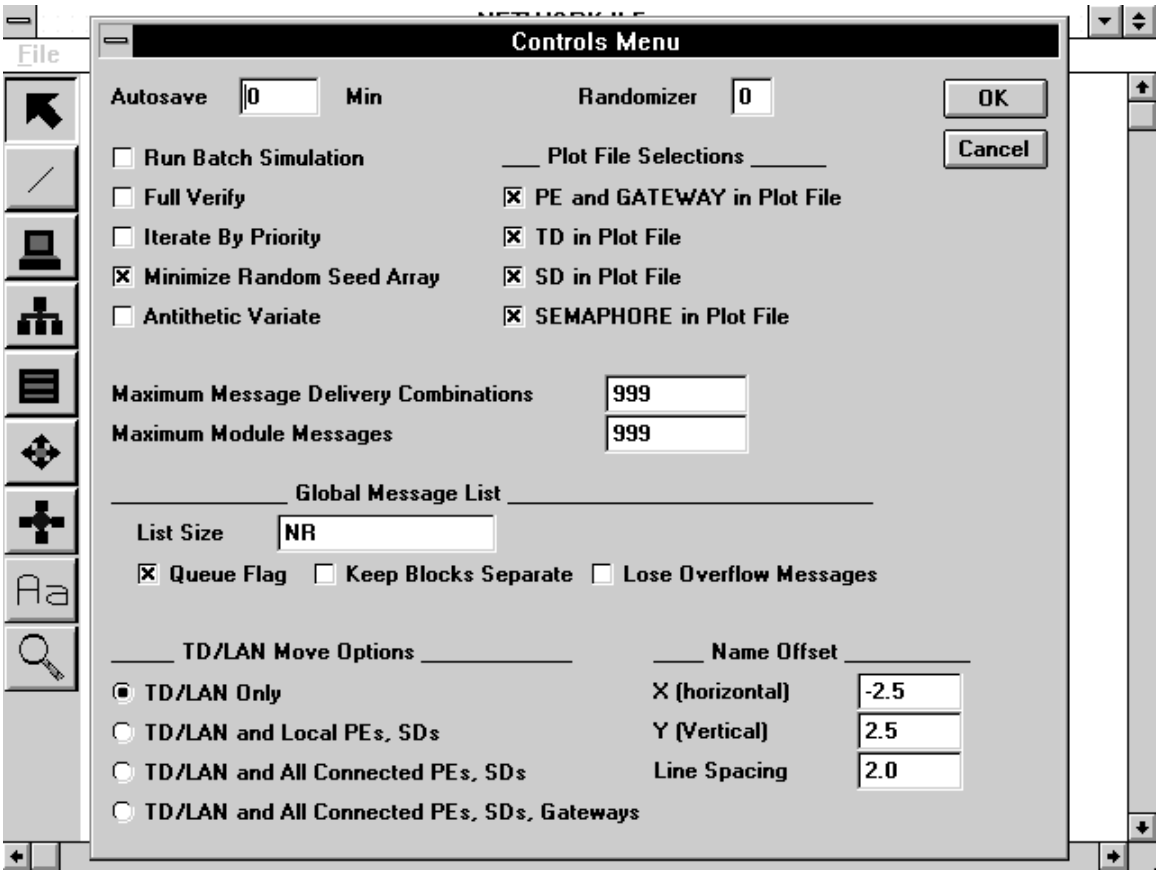
Modules may be defined by using on the Define/Module menubar command which will display a list of existing modules. You may add a new Module by selecting the ADD

button and entering a new name. Modules may also be accessed and created when within the PE form. Click on the Modules button found in the PE form and a list of Modules that run on this PE will be displayed. Modules may be selected from the list to be edited and new Modules may be added.

Several means of directly adding hardware and software data structures to your model have been described above. NETWORK will also indirectly create new data structures when you enter a name of a data structure that does not exist in an attribute field of a current building block. As an example, if you are editing a Module and enter the name of a Statistical Distribution Function for the Iteration Period of the Module and a SDF of the given name does not exist, a new one is automatically added to the model. If you verified the model immediately after editing the Module, an error message would be generated because the new SDF has not been defined with a type. Be aware that adding new names in attribute fields may automatically create new data structures which must be defined to be valid.

3.2.3 Defining Global Model Attributes

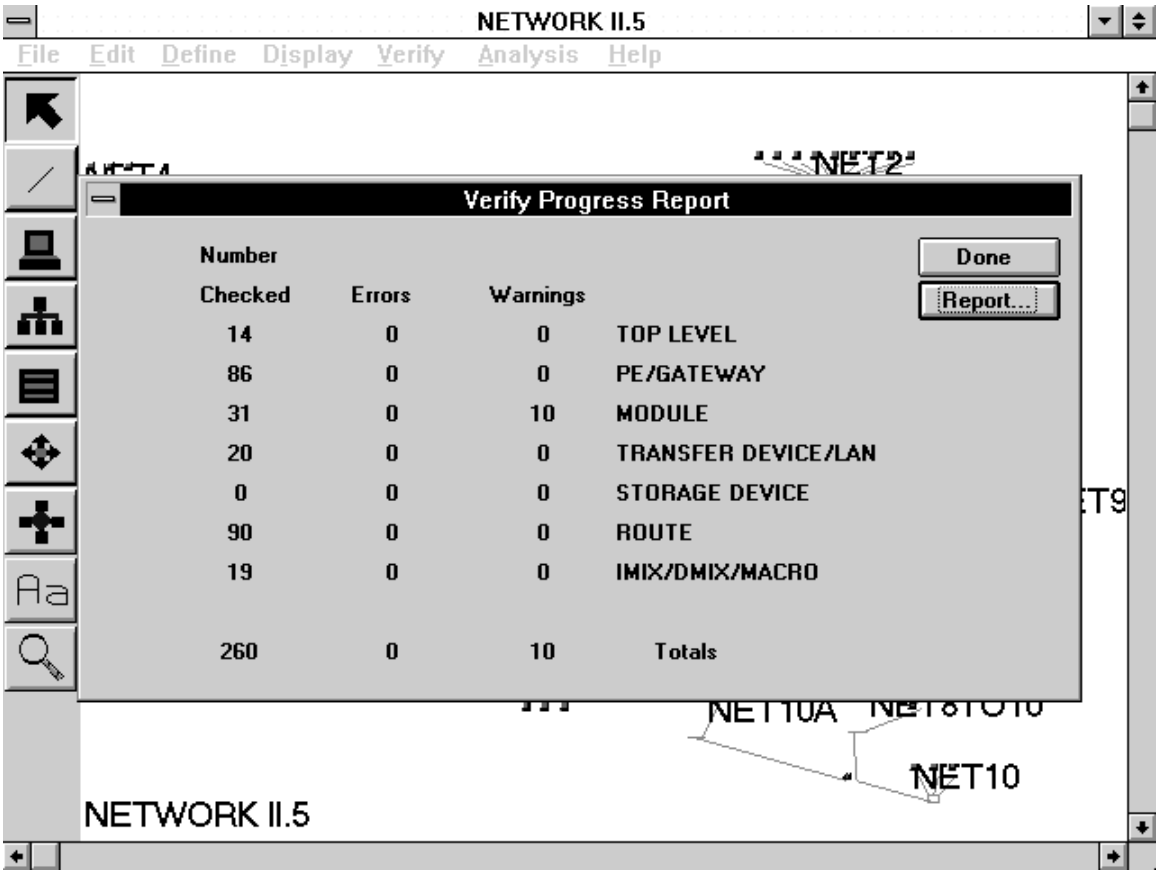
Global Model Attributes can be used to influence the behavior of a model to achieve varying simulation results. Changing Randomizer and Antithetic Variate will produce different numbers in a simulation without changing the attributes of the model components. Choose the Define/Controls command from the menubar to access the Controls form which contains various global attributes.



The Controls Form

3.2.4 Model Verification

Before attempting to run a simulation, you should verify the model to ensure that it has no errors. To verify the entire model, select the Verify/ALL command on the menubar. If errors are found, they can be corrected prior to running a simulation.



The Verify Progress Report

During the model verification process, a Verify Progress Report is displayed. When the verification process is complete, two buttons: REPORT and DONE, are displayed at the bottom of the form. If verify errors are found, click the REPORT button to view a description of the errors. You can then save this description to a file to use as a reference later. Selecting the DONE button removes the Verify Progress Report form.

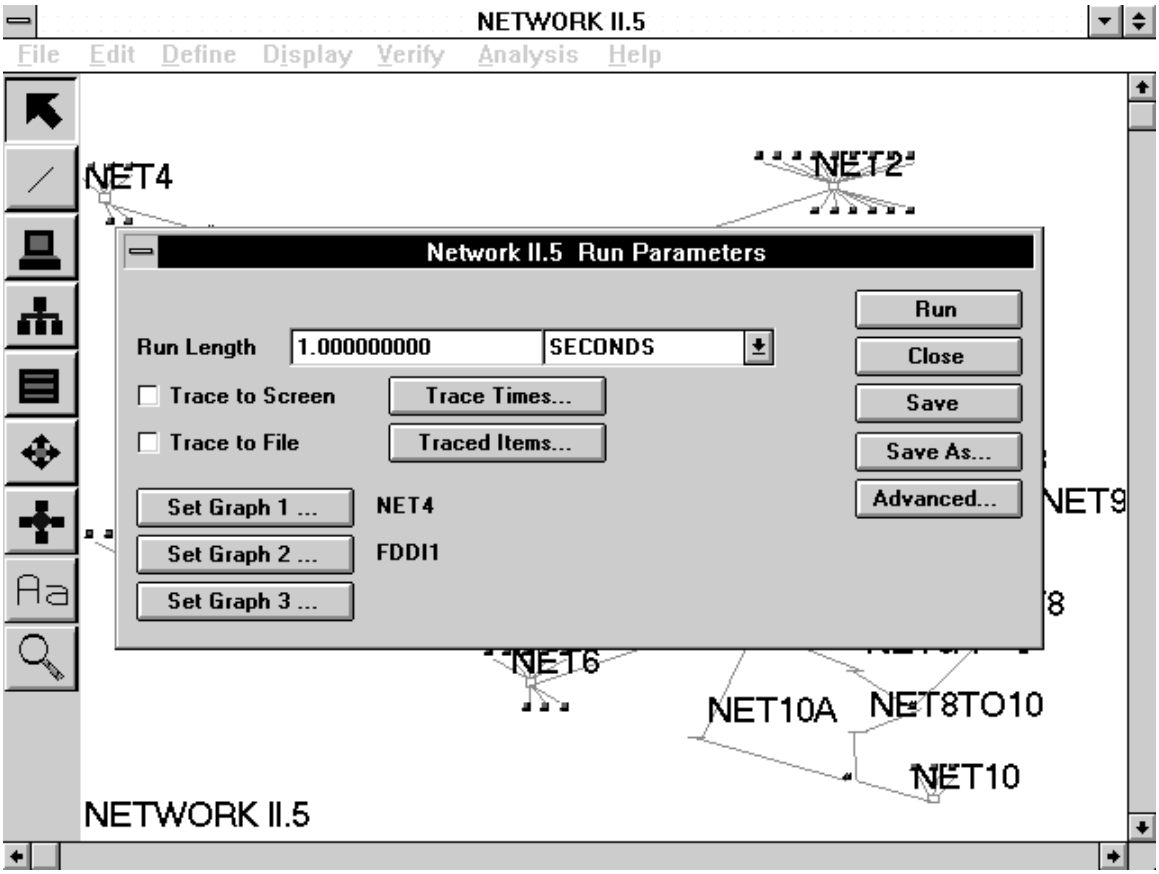
In the Verify Progress Report, the first column, Number Checked, notes the number of structures of each type found. The second column, Number With Errors, notes the number of data structures that have errors.

3.2.5 Saving The Model

A model may be saved by selecting the File/Save command from the menubar. You are now ready to run a simulation.

3.2.6 Running A Simulation

When you are satisfied that your model is sufficient to test, use the Analysis/Start Simulation command on the menubar to display the Run Parameters form. After selecting a Run Length and setting any other desired optional attributes, start the simulation by selecting the RUN button. A full Verify is always performed on a model just before the simulation starts.



The Run Parameters Form

3.3 MANIPULATING THE DISPLAY

3.3.1 Scrolling

Scroll bars are positioned on the display window to allow you to focus on different parts of your model, if the layout is larger than the display window. The vertical scrollbar on the right side of the display provides the up and down movements for the display window

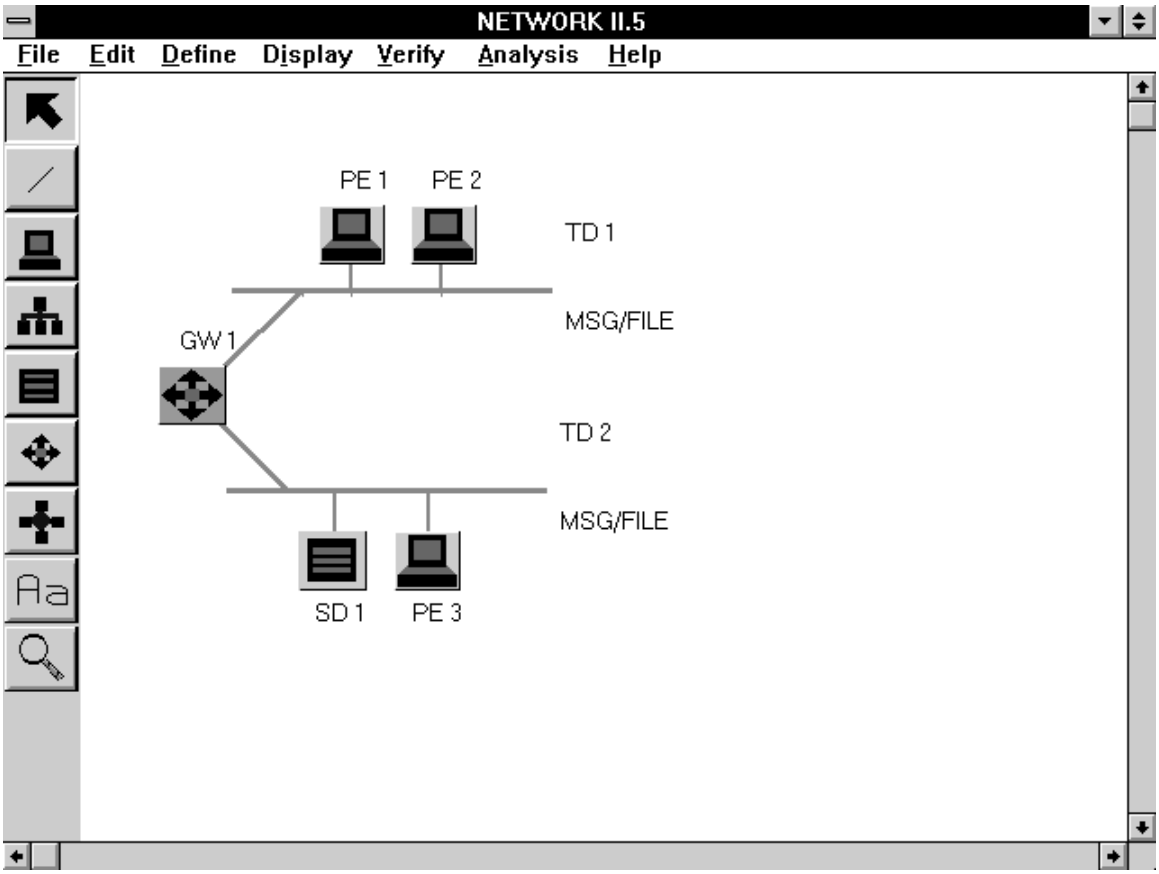
and the horizontal scrollbar on the bottom of the display provides left and right movements. To examine an object out of range of the display and to the right of the currently displayed objects, click on the right arrow of the bottom scrollbar, or simply drag the scroll box to the right. When you begin NETWORK, the working area size, which is the area on which you may position hardware icons, is equal to 1.5 times the display window, expanded to the right and bottom of the display. To increase the size of the work area for large layouts, select the Display/Work Area Size command from the menubar.

3.3.2 Selecting Objects On the Display

Objects on the display (PE, TD, SD, LAN, Gateway) may be selected for editing, moving, cutting, etc., by clicking once on the object's icon. When an object has been selected, its icon is highlighted. Multiple objects may be selected at a given time. To deselect all selected objects, click on the display background. To immediately edit a single object on the display, double click on its icon and its corresponding attributes form will be displayed.

3.3.3 Positioning Objects On the Display

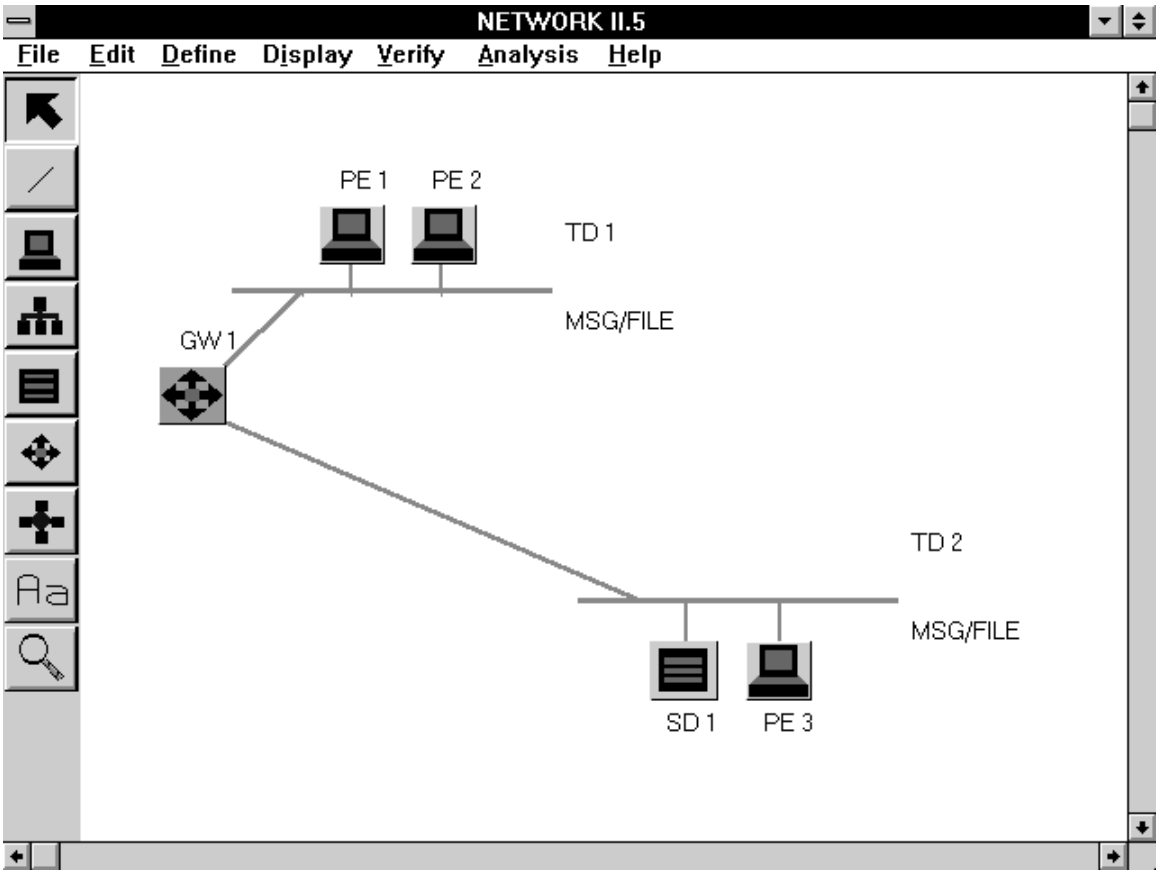
PE icons, SD icons, Gateway icons and labels may be positioned on the display by dragging these objects with the cursor. Click directly on an object and move the mouse without releasing the mouse button. As long as the mouse button is not released you may reposition the object on the display. Release the mouse button when the object is in the desired position and the object will remain in its new position.



The Display Prior to a TD/LAN Move

If you position a TD or LAN icon, the manner in which the TD/LAN icon and the icons of connected PEs/SDs move depends on NETWORK's TD/LAN Move Mode (four are available in the Controls Form). If the mode is *TD/LAN* only, the TD/LAN icon moves independent of all connected devices. If the mode is *TD/LAN and Local PEs, SDs*, all Pes and SDs that are connected solely to this TD/LAN will move relative to the TD/LAN icon. If the mode is *TD/LAN and All Connected PEs, SDs*, all Pes and SDs connected to this TD/LAN will move relative to the TD/LAN icon. The fourth move mode, *TD/LAN and All Connected PEs, SDs and Gateways*, will also move any connected Gateways relative to the TD/LAN icon.

Multiple objects may be positioned as a group. To do so, click once on several icons (PE, TD, SD, LAN, Gateway, label) on the display to highlight them. Then, click and drag a single icon (PE, SD, Gateway, or label but not a TD or LAN) and reposition it. Each highlighted icon will move relative to the repositioned icon.

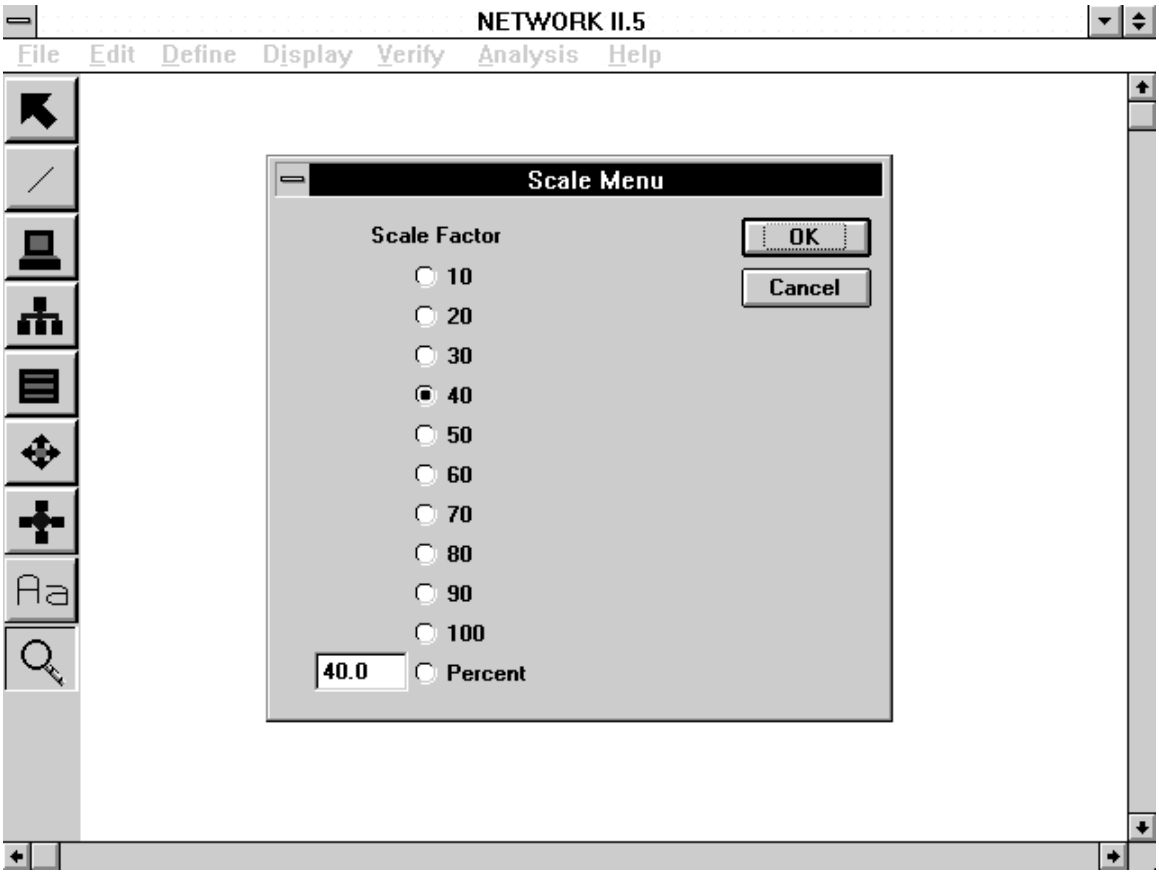


The Display After a TD/LAN Move

3.3.4 Scaling the Display

The Scale command allows you to change the diagram magnification and is invoked by selecting the Scale button on the toolbar which will display the Scale Form.

The Scale form has a set of radio buttons which will allow you to change the diagram scale in increments of 10%. The diagram scale ranges from the platform minimum (Windows and UNIX - 1%, OS/2, DOS - 10%) to 100%. A value box is also provided for fine-tuning of the diagram scale. The default scale is 40 percent. To choose the diagram scale, click on the appropriate radio button and click on OK.



The Scale Selection Form

3.3.5 User-Created Icons

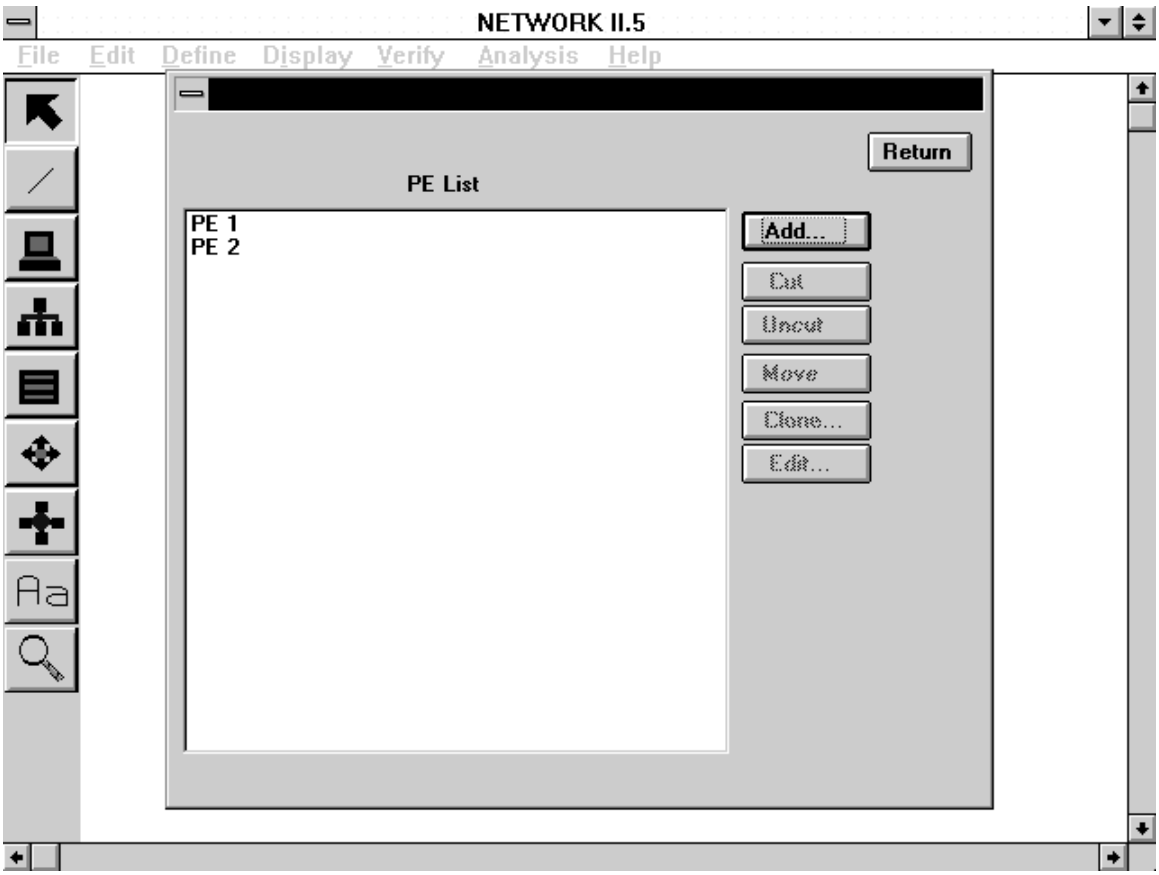
You may create your own hardware icons using the SIMGRAPHICS II editor. After you create an icon, it must be included in the proper NETWORK II.5 graphics library file to be available for your models. Refer to Appendix G for details on running the SIMGRAPHICS II editor and saving your new icons.

3.3.6 Managing Complex Displays

Large models may become quite cluttered and confusing on the display, especially if the diagram is scaled to view all model components. For highly meshed networks, the topology may also become quite messy. To produce a more coherent network layout, it is often helpful to hide individual hardware icons on the display.

Icons may be hidden individually or as a group. Single click on one or more hardware icons to select/highlight the items to be hidden. Then use the Edit/Hide command on the

menubar to remove the selected icons from the display. Although a hidden item is not found on the display, the data structure still exists in the model and it can still de accessed with the Define command on the menubar.



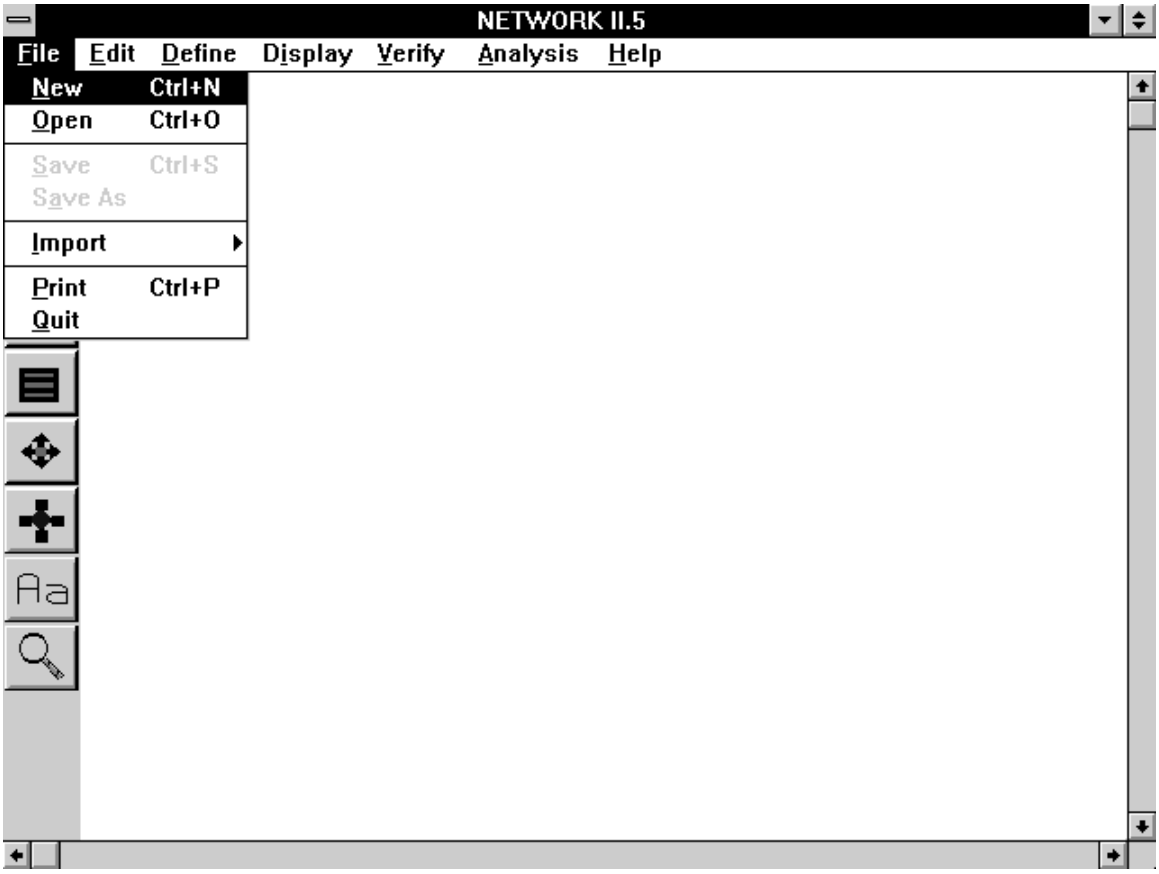
The PE List

Individual icons may also be hidden by selecting the GRAPHICS button from within the attributes form of the hardware entity. The Hide check box on the form is used to control the hiding of icons. Marking this box with a check mark will hide the item if the OK button is selected in the Graphics form. To redisplay a hardware icon, it is necessary to enter the Graphics form of the hidden hardware element. The Define command will be required because there is no display icon to highlight. Once in the Graphics form, click on the Hide check box to remove the check mark and OK the form. The hardware icon will then be restored to its previous position. Thus, to redisplay a PE that is hidden, you must select Define/PE from the menubar, select the PE name from the list, select the EDIT button to get to the PE attributes form, select the GRAPHICS button, and finally remove the check mark from the Hide check box and OK the from.

3.4 NETWORK MENUBAR COMMANDS

This section describes the commands available through the NETWORK menubar. The items on each of the pulldown menus are listed from top to bottom.

3.4.1 The File Menu

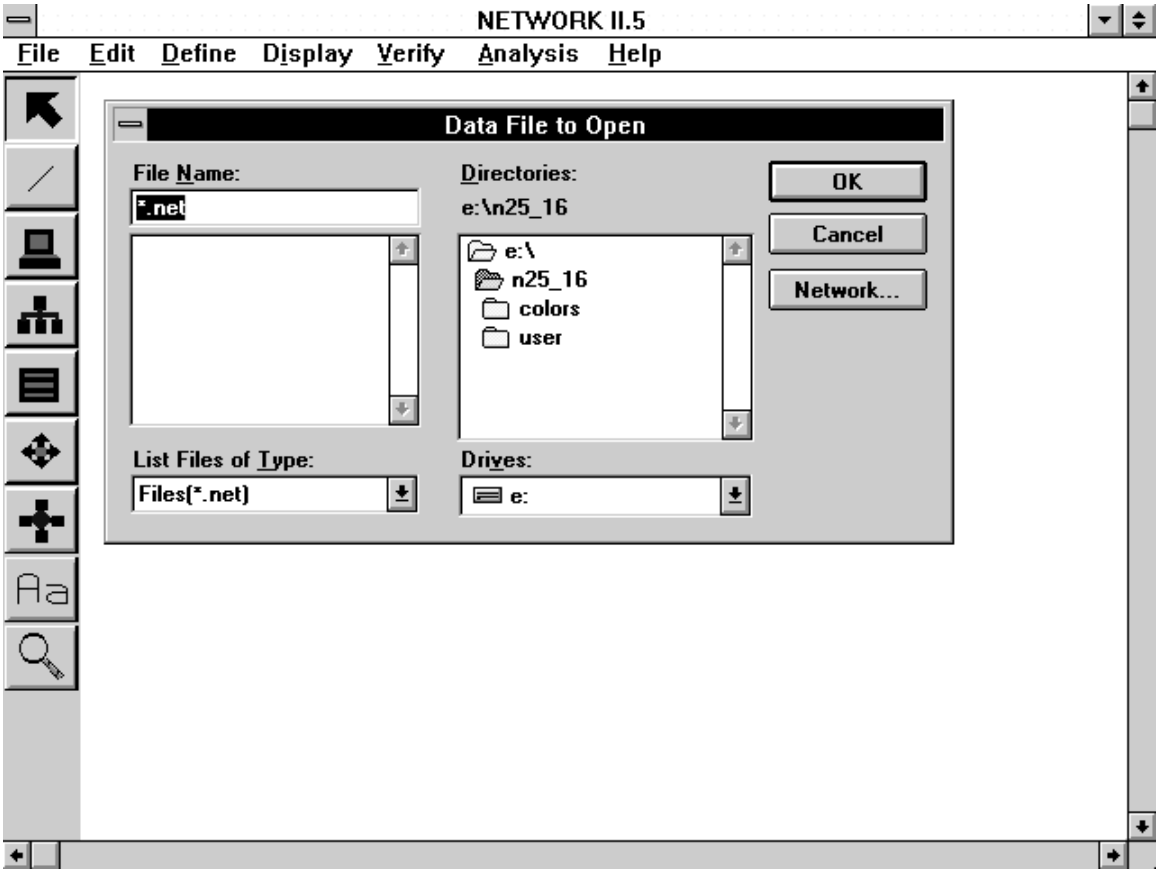


The File Menu

File/New
The File/New command clears the current model from NETWORK. By using this command, you do not need to leave and restart NETWORK to begin building a new model.

File/Open
The File/Open command retrieves a previously saved Network Description File (.NET). If an existing model has already been loaded in NETWORK or data structures have already been created, you have the option to either merge the data from the file to be

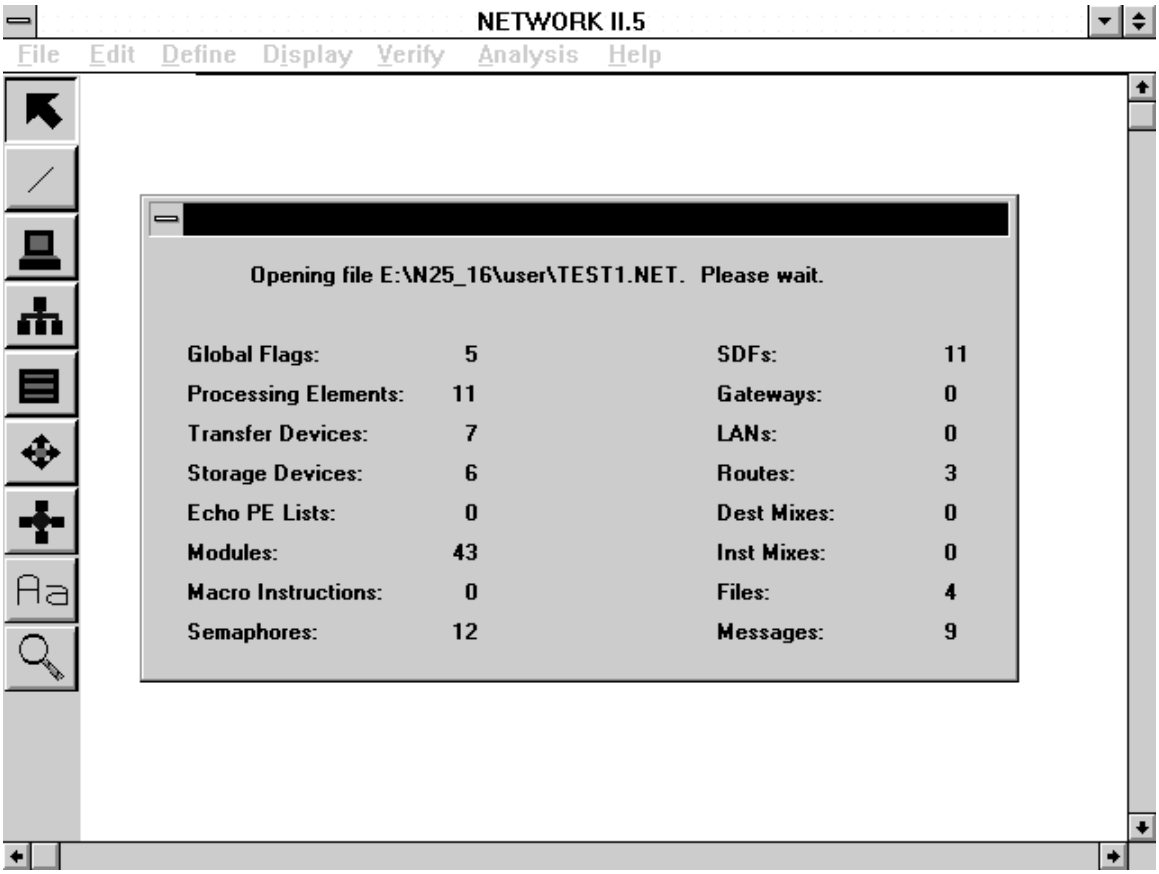
opened with the current description or to replace the current data with the file to be opened.



The Open Form

After the File/Open command has been entered, a form is presented to permit you to enter the name of a Network Description File (.NET). A file name may be entered directly into the File Name box, or selected from the list of names below this box. The list of file names displayed is based on the current directory selected and contains all Network Description Files found in that directory. Depending upon the platform you are using, options should be available to select a different directory or drive.

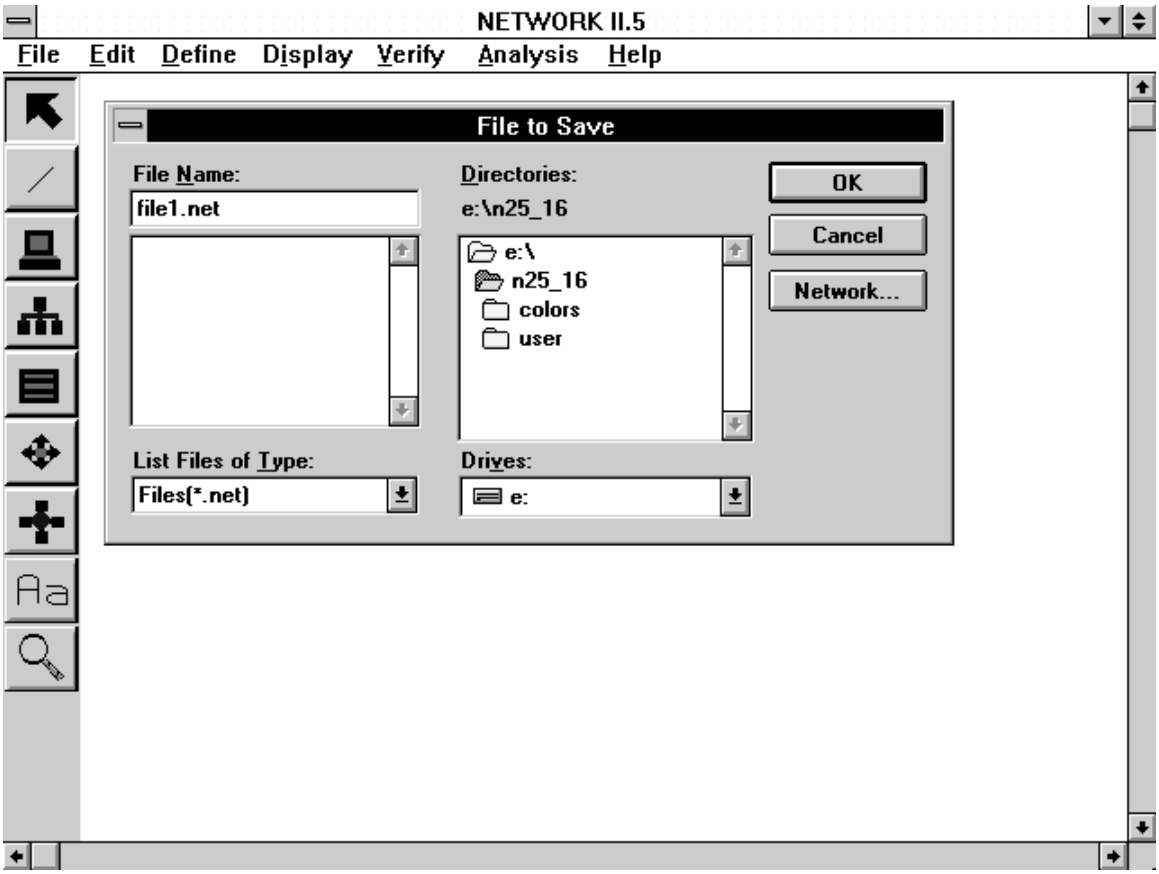
A form is displayed during the process of loading the model, in which NETWORK displays the number of individual model components read in from the Network Description File. When the model is completely loaded, this form is erased and all model components with drawing information are displayed.



The Load Progress Form

File/Save

The File/Save command stores the current model in a Network Description File (.NET) using the current file name. If you wish to save a model using a different name, use the File/Save As command (see below).



The Save Form

File/Save As

The File/Save As command allows you to save a model using a new file name or file path. The save form used by this command has a file name box where you can directly enter a file name. The current file name will appear in this box when the form is first displayed. Depending upon the platform in use, you may be able to select which directory or drive your Network Description File (.NET) is saved to.

File/Import

The File/Import command is used to make calls to the TRAFLINK III program. A side menu is displayed which has two types of options. The first option External Traffic allows you to import an external traffic file which can be used to help "drive" a simulation. The second group of options will import a topology produced by a network management system to provide the initial hardware layout of a model. Please refer to Chapter 10 for a complete description of TRAFLINK III and how to use it.

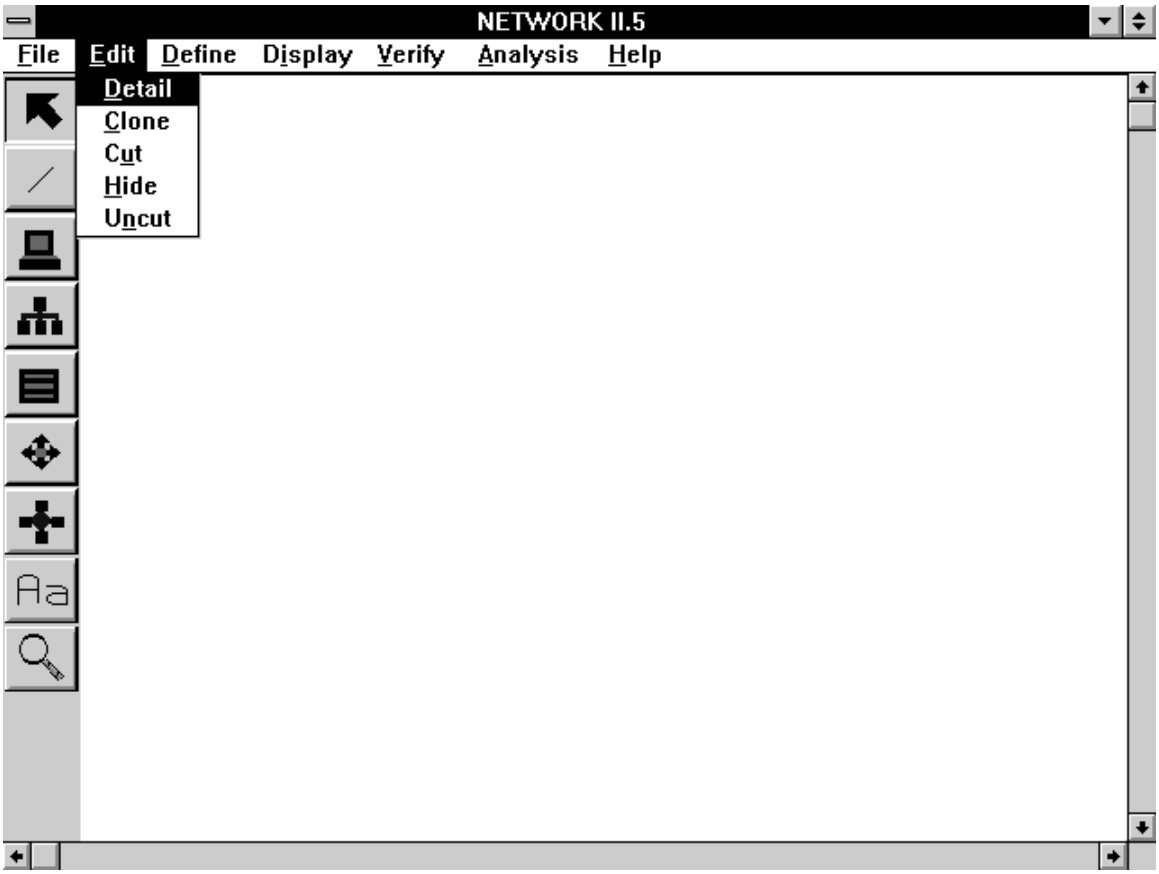
File/Print

The File/Print command is used to access system print functions. This command will display a form which will permit you to print the NETWORK display directly, or to save the screen shot to a postscript file.

File/Quit
The File/Quit command terminates a NETWORK session. If you have modified a model and have not saved the changes, NETWORK requests confirmation before ending the session. If you elect to save the changes, the Save Form is displayed. After the save operation has completed, the session is terminated.

3.4.2 The Edit Menu

Edit/Detail
After selecting (highlighting) an object, or a group of objects (PEs, TDs, LANs, SDs, and Gateways), the Edit/Detail command accesses the forms which contain the object's attributes. When multiple items are selected, PEs are edited first, followed by Gateways, TDs, LANs, and SDs. If you have not selected an object, the Edit/Detail command will not be selectable.



The Edit Menu

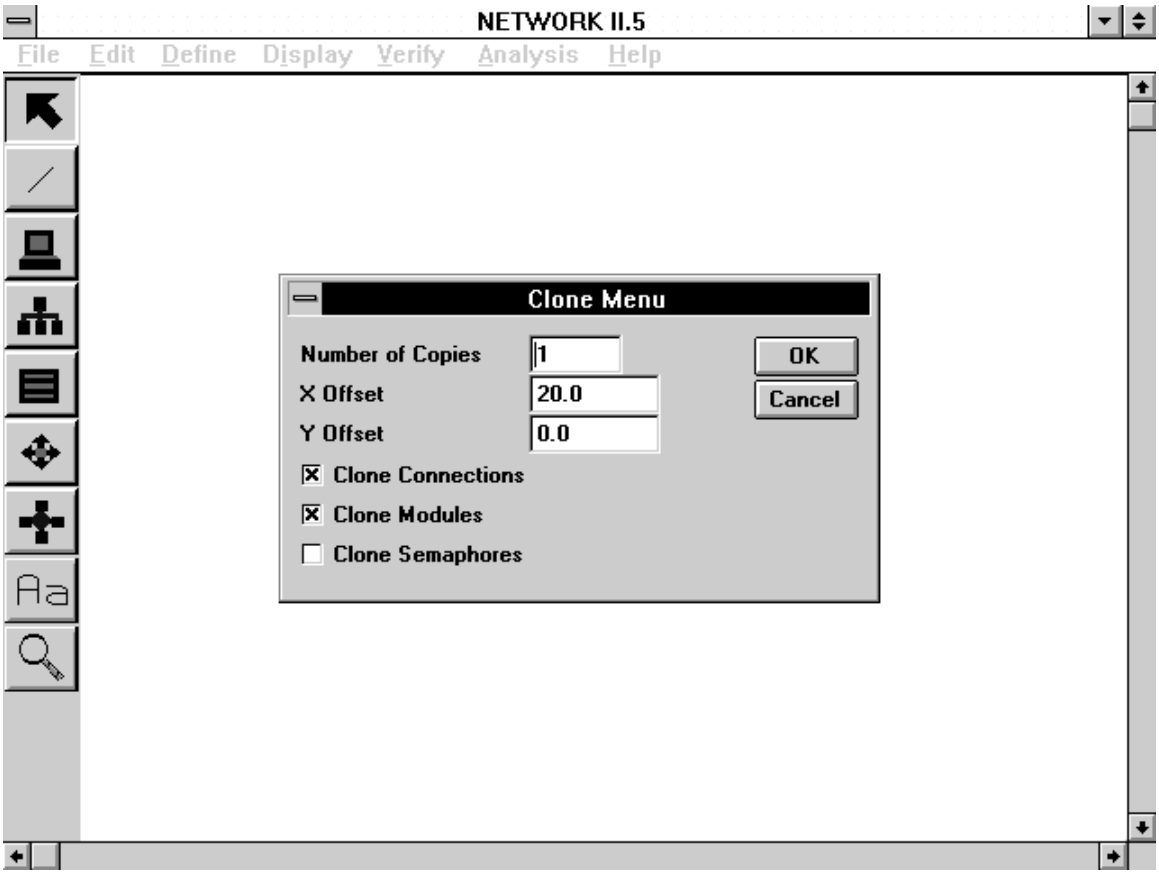
Edit/Clone

After selecting an object, or a group of objects, use the Edit/Clone command to produce another object with identical attributes. Multiple copies of an object can be created, connections may be duplicated, Modules specific to a cloned PE may be copied and Semaphores specific to a cloned PE may be cloned. If you have not selected an object on the display, the Edit/Clone command will not be available.

Clones are positioned next to each other in a line. The X Offset (default = **20**) and Y Offset (default = **0**) values determine the position of the newly created objects relative to the original cloned object. Clones are automatically named by adding a copy number to the name of the parent object. For example, if two clones are created from the PE named WORKSTATION, the resulting clone names are WORKSTATION.1 and WORKSTATION.2.

PE, SD, and Gateway connections to TDs and LANs may be replicated by selecting the Clone Connections check box. An example application of Clone is to produce multiple identical PEs connected to the same TD or LAN. For this application, it is helpful to set the attributes and draw options of the PE, and to connect the PE to the TD or LAN before performing the Clone.

The Clone Modules check box is only used when PEs are cloned. Setting this check box to yes may have two effects on the cloning process. Any Modules that run exclusively on a PE that is cloned (i.e. those Modules with a single Resident or Allowed PE name identical to the cloned PE) will be duplicated and renamed for each clone. For example, the Module called SEND MESSAGE runs on the PE WORKSTATION. When WORKSTATION is cloned, the new Module WORKSTATION.1 - SEND MESSAGE will be created if Clone Modules was selected. Also, those Modules that may run on several PEs including the PE being cloned will be will have their Resident/Allowed PE lists altered by the addition of clone names in these lists. In this case, no new Modules are created, but current Modules may now run on the clones. If you do not select Clone Modules, new Modules will not be created, and the existing Modules will not be altered.



The Clone Form

The Clone Semaphores check box is only used if PEs are to be cloned. If Clone Semaphores = YES, for any cloned PE, a check is made to see if this PE has Semaphore Instructions that access Semaphores that are not accessed by any other Pes in the model. If such a PE is to be cloned, a newly named Semaphore is created for each of the Semaphores specific to the PE and the Semaphore Instructions of the cloned copies of the PE will have the new Semaphore names. This will allow you to retain PE/Semaphore groupings so that statistics gathered for Semaphores will relate to one PE only.

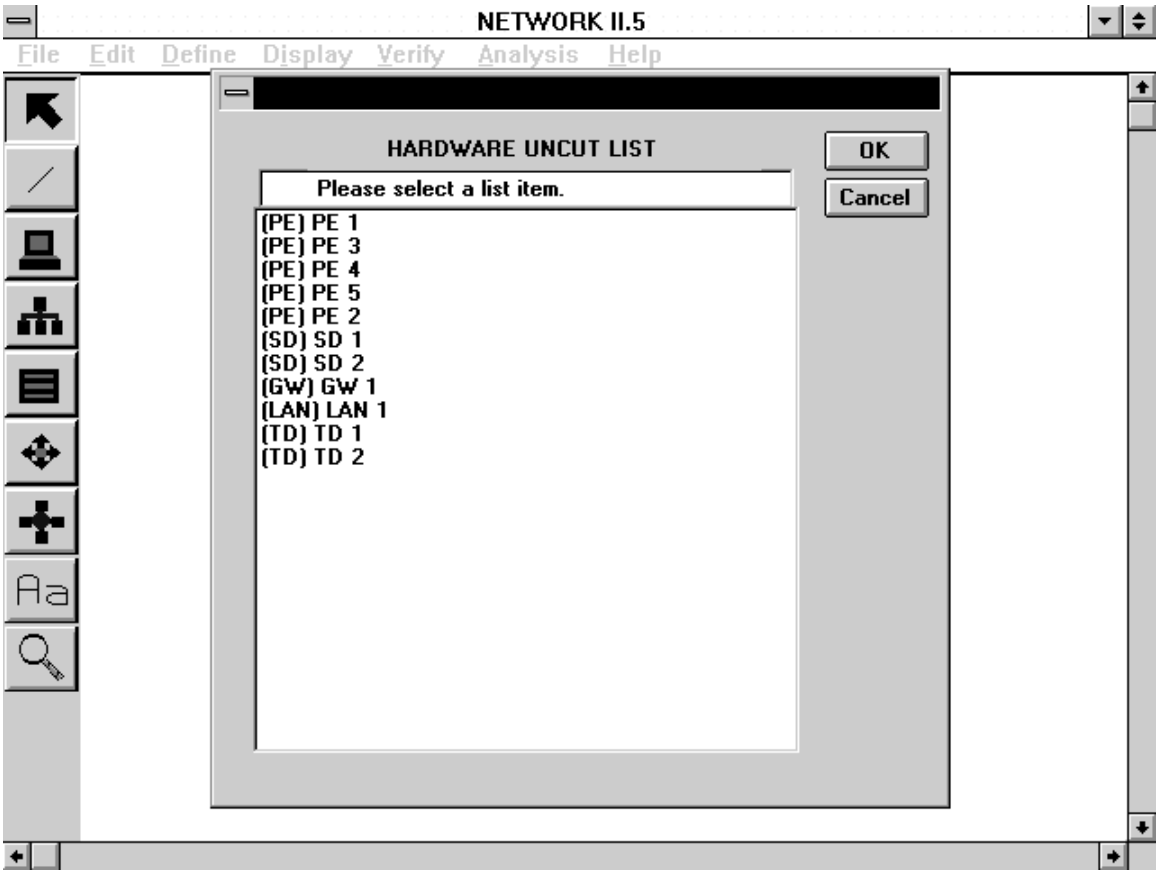
Edit/Cut

The Edit/Cut command removes objects (TD, LAN, PE, SD, Gateway) which have been selected (highlighted) from the model. Cutting a PE, SD, or Gateway will also delete any TD/LAN connections. The Edit/Cut command is only available when one or more objects have been selected on the display.

Edit/Hide

After selecting (highlighting) an object, or a group of objects (PEs, TDs, LANs, SDs, and Gateways), use the Edit/Hide command to hide these objects from the display. Once hidden, an object is not visible, but it is still present in the model and will be included in

your .NET file if a save is requested. To redisplay a hidden object, you must use the Edit/Select command to access the hidden object by its name and set the Hide check box of the object to NO in the appropriate Draw Options Form. The Edit/Hide command is only available if at least one object on the display has been highlighted.

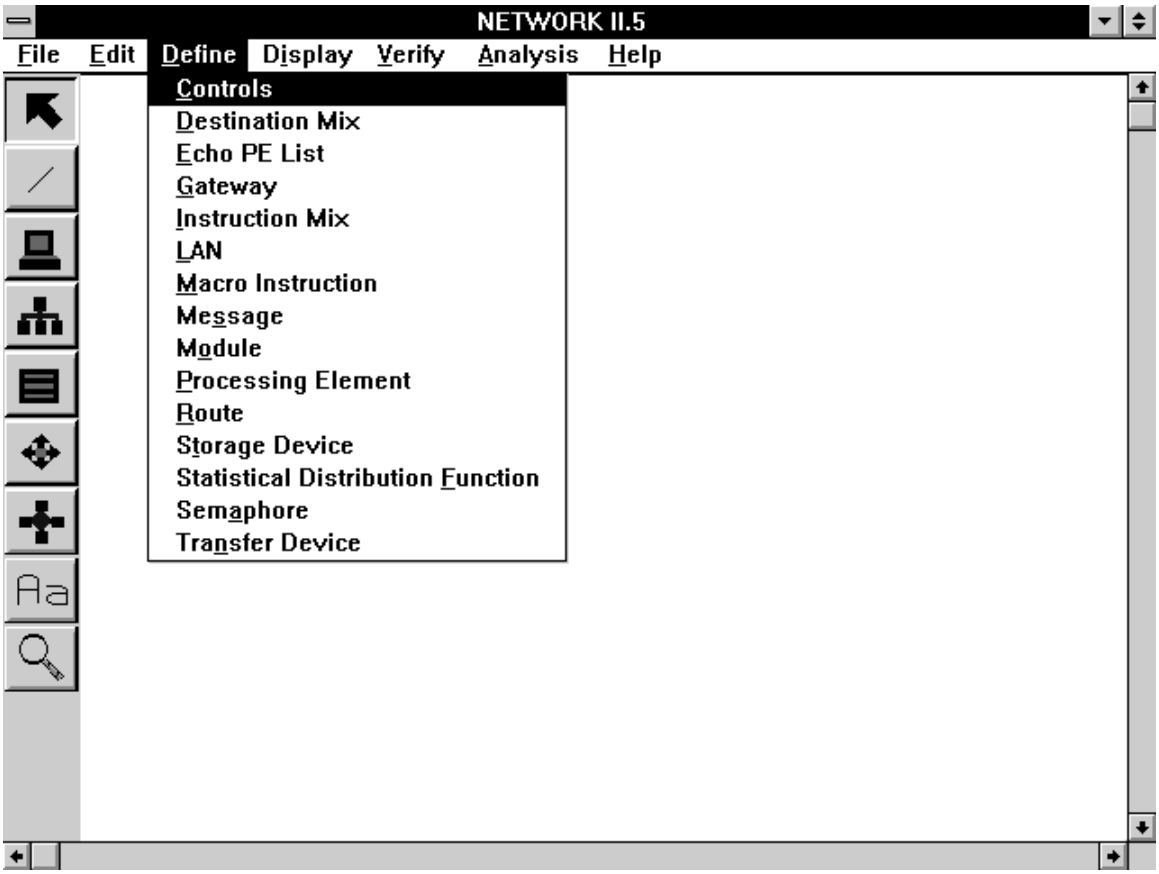


The Uncut Form

Edit/Uncut

The Edit/Uncut command allows you to restore objects which have been previously cut from the model. After selecting this command, a list of cut data structures will be displayed. Selecting names from the list of cut data structures and then clicking on the OK button will restore the Cut data structures. TD/LAN connections to PEs, SDs, and Gateways will not be restored when you return a PE, SD, or Gateway to the model. The Edit/Uncut command is selectable only when a hardware data structure has been cut from the model.

3.4.3 The Define Menu



The Define Menu

The Define Menu allows you to create, edit, cut, restore and clone the data-structures of your model in a list-oriented fashion. You may also edit program and simulation control parameters by selecting Controls from the Define Menu. The Define Menu is an alternate means for working with the hardware data structures of NETWORK II.5 which can be created and selected directly on the display. For most of the software data structures, the Define Menu is the only way to access them within NETWORK.

When the Define command is selected, a menu with the names of all the types of building blocks of NETWORK II.5 is displayed with the additional choice of Controls. Selecting Define/Controls will present the Controls Form which is described in section 3.5.17.

Selection of any of the Define Menu options other than Controls (Destination Mix, Echo PE List, Gateway, Instruction Mix, LAN, Macro Instruction, Message, Module, Processing Element, Route, Storage Device, Statistical Distribution Function, Semaphore and Transfer Device) will display the Define Form which has a name list containing the names of the current data structures for the type selected. To the right of the name list is a group of buttons that will let you edit data structures and perform other tasks.

The Add button is always selectable from the Define Form and is used to introduce new data structures to your models. When the Add button is selected, you will be prompted for a new name. If a valid name is entered, a new data structure is created and its name is then placed at the bottom of the name list in the Define Form.

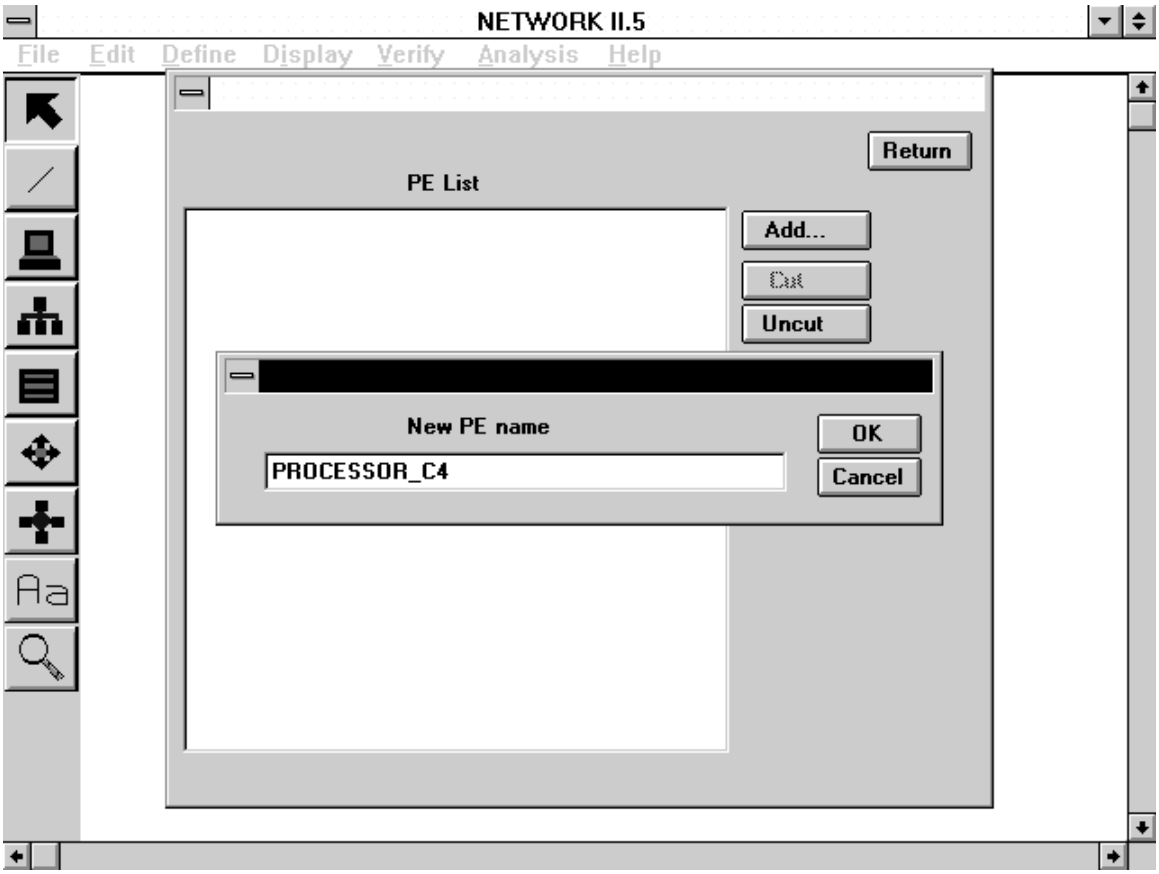
The Cut button is used to remove data structures from a model. The Cut button is only selectable when one or more names have been highlighted in the name list. When entities are removed with the Cut button, their names are removed from the name list. If a PE, Gateway, SD, TD or LAN is cut in this way, its icon will also be removed from the display.

The Uncut button will restore cut data structures. If any data structures of the type selected in the Define menu have been cut during your current NETWORK session (unless you select File/New), the Uncut button will be selectable. The Cut button will display a list of the cut data structures from which names may be selected to restore the cut data structures to the model.

The Move button will reposition a name within the name list and is selectable only when 1 name has been selected/highlighted in the list. This is useful not only to position those items in the list that are frequently edited in a more convenient location, but in some cases, the list position of a data structure can influence the model. The order of PEs is important, as an example, when a Module with an Allowed PE List of “any” is searching for a host PE. The name to be moved will be removed from the list, and list position numbers will be inserted between the remaining list names to assist you in repositioning names. Once a Move is started, it must be finished before any other actions can be performed.

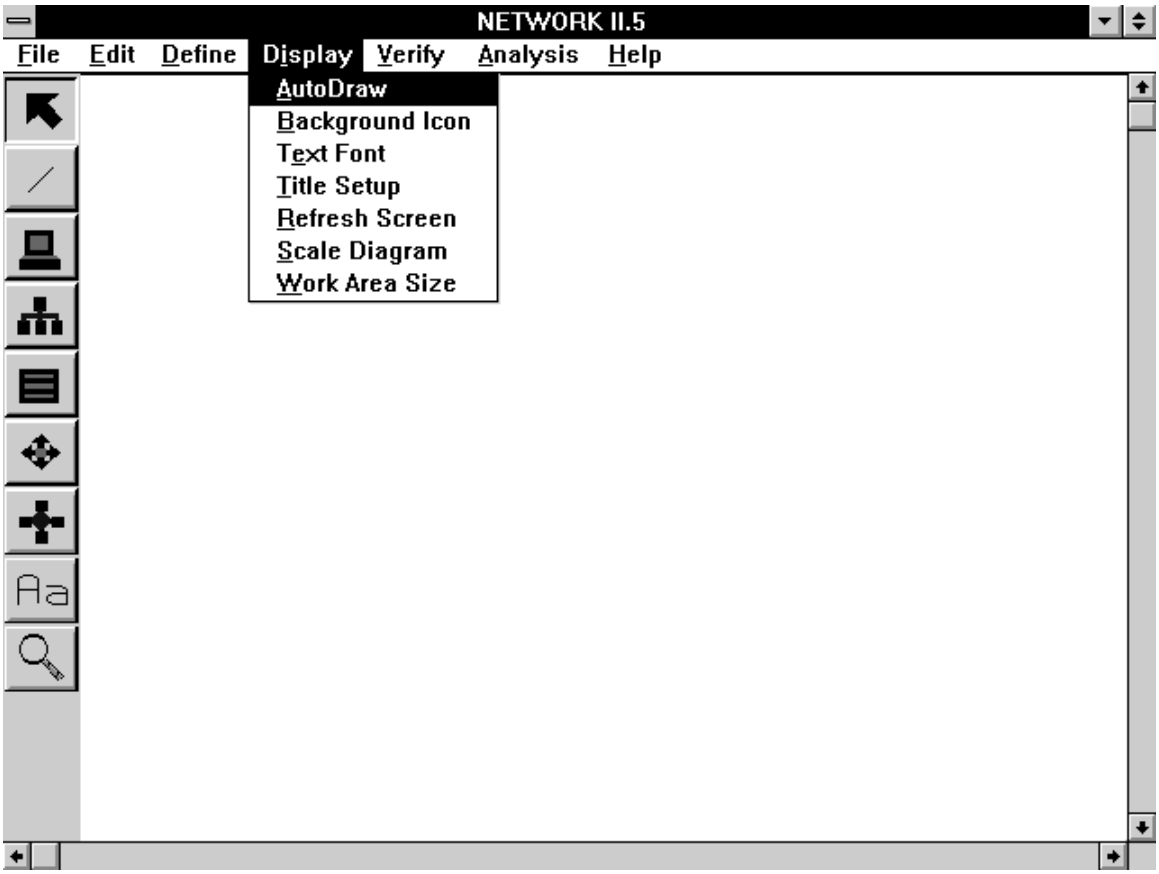
The Clone button is selectable when a list name is highlighted and will create new copies of the selected data structure. If a hardware data structure is being cloned, the standard Clone Form will be displayed to allow multiple clones. If the Clone button is used for a software data structure, you will be asked merely to supply a new name and only one copy is made. The names of the clones will then be displayed in the name list.

The Edit button will display the standard attributes form for each of the selected list names so that you can examine/modify your data structures.



Adding a New Data Structure

3.4.4 The Display Menu



The Display Menu

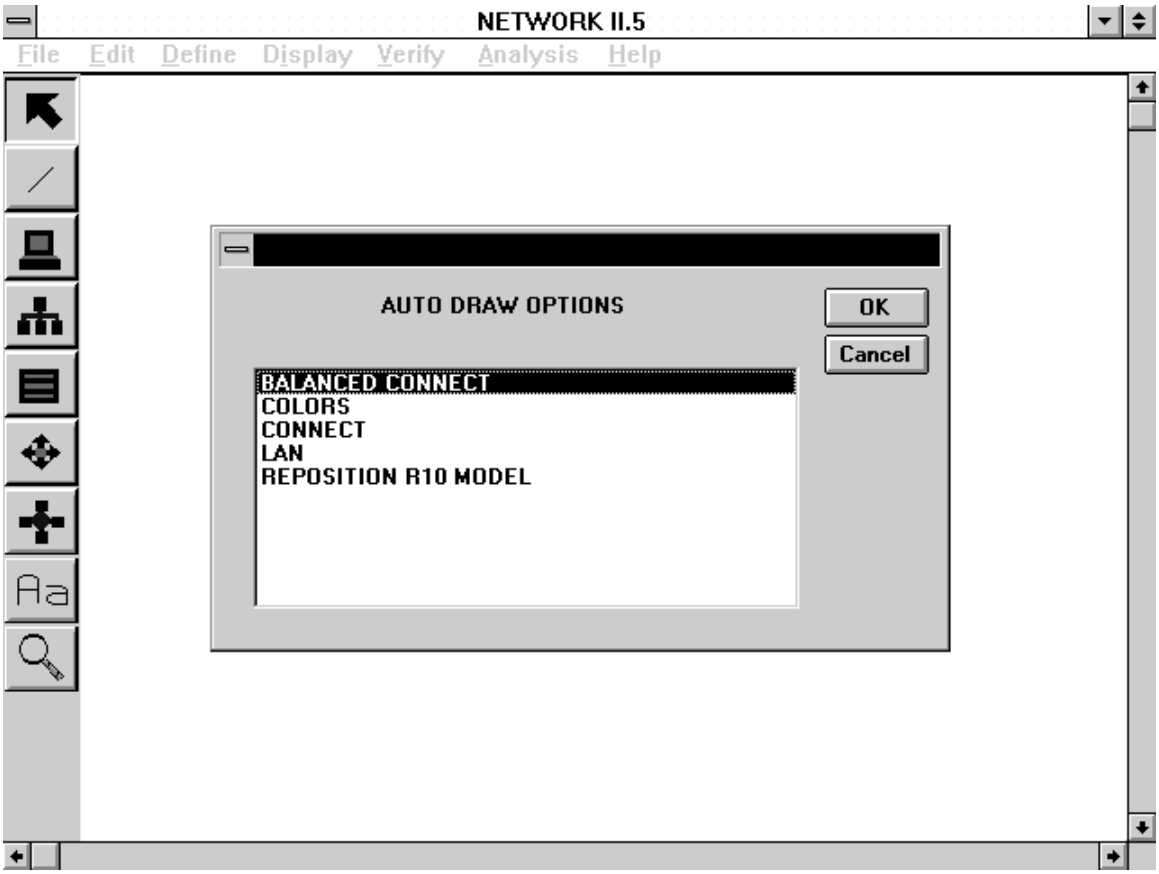
Display/AutoDraw

The Display/AutoDraw command is used to automatically draw the components of your model on the display. After selecting this command you may choose one of the five options from the AutoDraw Options form. Three of the options, CONNECT, BALANCED CONNECT, and LAN, are used to reconfigure the layout of objects on the display. The COLORS option allows you to decide which colors will used in the display. Click on the desired option and click on the OK button. If CONNECT, BALANCED CONNECT or LAN is chosen, NETWORK requests confirmation since all current drawing information will be lost when the auto draw begins.

The CONNECT option draws all Processing Elements in a row from left to right across the top of the diagram. All of the TDs and LANs are drawn as straight lines underneath the Processing Elements. All the Storage Devices are drawn in a row from left to right below the Transfer Devices. Gateways are drawn to the right of any Processing Elements. This results in a hardware “sandwich”.

The BALANCED CONNECT option draws a diagram in the same format as CONNECT except that PEs, SDs and Gateways may appear either above or below TDs and LANs. This algorithm minimizes the width of the resultant diagram at the expense of a less structured diagram.

The LAN option will attempt to draw each TD and LAN as a separate entity, and then connect all PEs, SDs and Gateways found in the TD or LAN Connection List. If the PE, SD or Gateway has already been connected to another TD or LAN, a connection is drawn from the TD or LAN to the PE, SD or Gateway from wherever it has been placed on the screen.



The Auto Draw Options Form

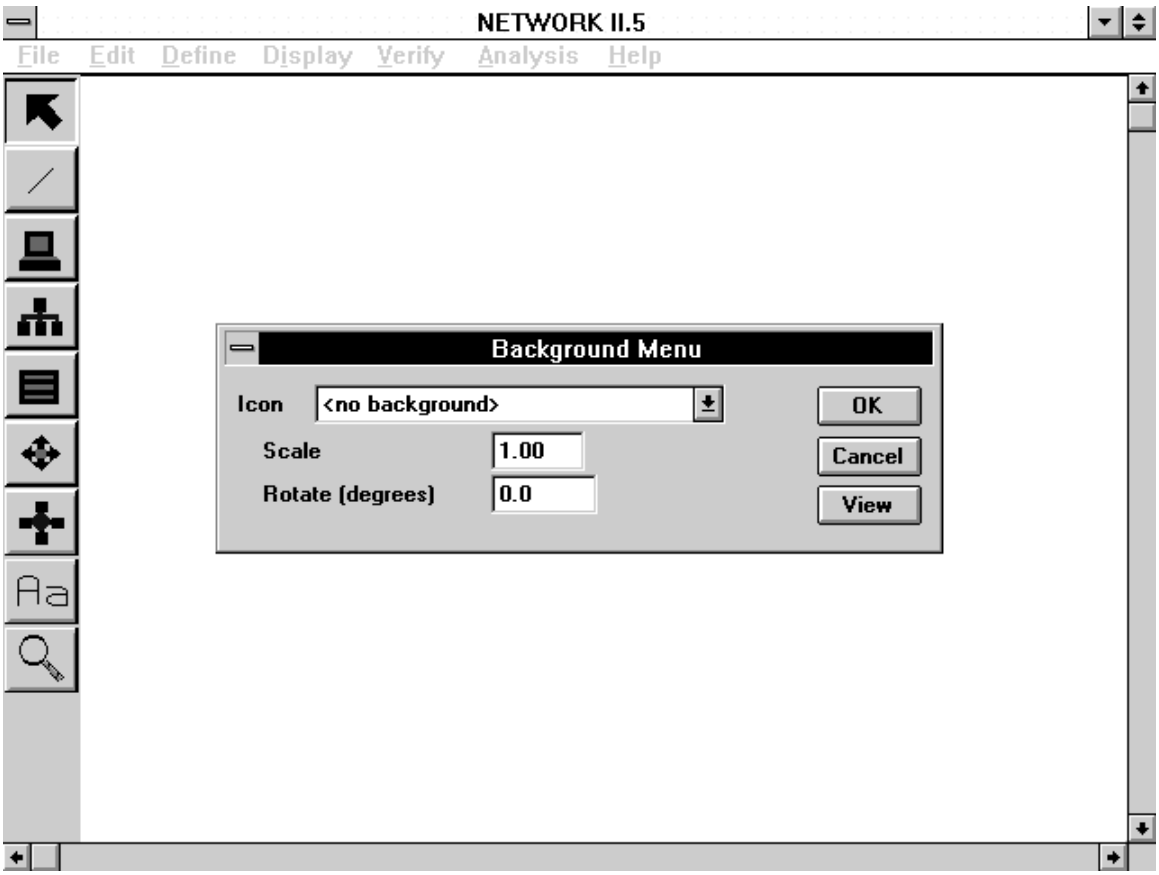
The COLORS option allows you to select the colors that will be seen on the display. A default set of color choices is used when NETWORK is invoked. The current color choices are indicated by a check next to the color. To select (or deselect) a color, click on the check box next to the desired color. When all color choices have been made, clicking on the OK button accepts the color changes. The display will be adjusted so that the only

colors shown are the current color choices. To leave without adjusting any colors, click on the CANCEL button.

The Reposition R10 Model option is useful if you an older model that is read into NETWORK is displayed with connections from TDs/LANS to hardware icons not completely connecting to the hardware icons. This offset of hardware icons is a result of the newer NETWORK graphics system defining an icon's origin in the center of the icon rather than the upper left corner. Simply select this option to "adjust" your layout if the hardware icons seem to be out of position.

Display/Background Icon

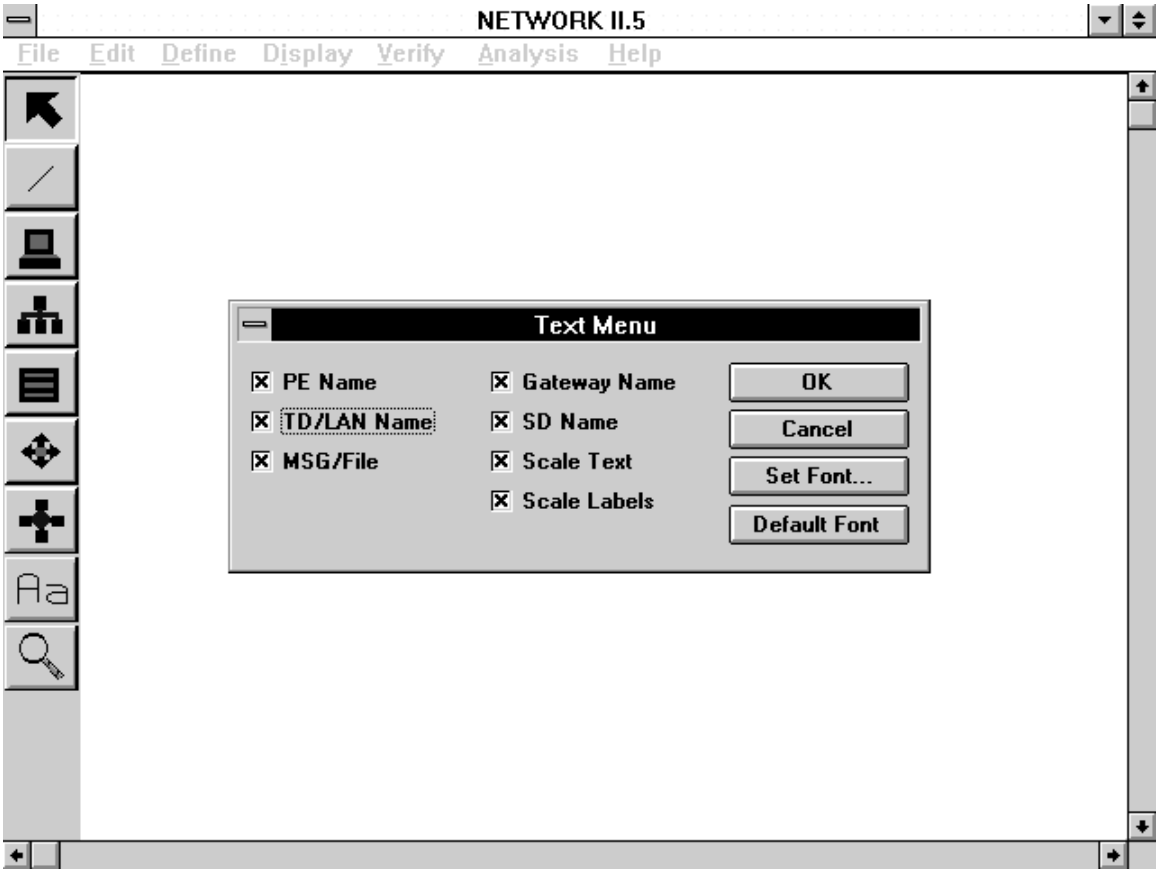
If you wish to add a background icon to your model, use the Display/Background Icon command. An icon may be selected from the list of icons in the combo box. Background icons can be scaled and rotated as well. A set of background icons is provided with NETWORK II.5. You may create your own icons by using the SIMGRAPHICS II editor supplied with NETWORK II.5. Background icon file names must start with the letters BK, end with a file extension of .ICN and must be stored in the appropriate graphics library file. A further description of the SIMGRAPHICS II editor and creating your own background icons is given in Appendix G.



The Background Icon Form

Display/Text Font

The Display/Text Font command can be used to set the text font used on the NETWORK display (to display icon names and labels) and also determines if the PE Names, TD/LAN Names, TD/LAN Msg/File, SD Names, and Gateway Names are shown on the display. To turn the name display on or off, set the check box next to the desired item. The Scale Text and Scale Labels check boxes are used to scale text equal to the display scale. The Set Font and Default Font buttons are used to change the display text font or restore it to the default. Click on the OK button to accept changes and redraw the display.

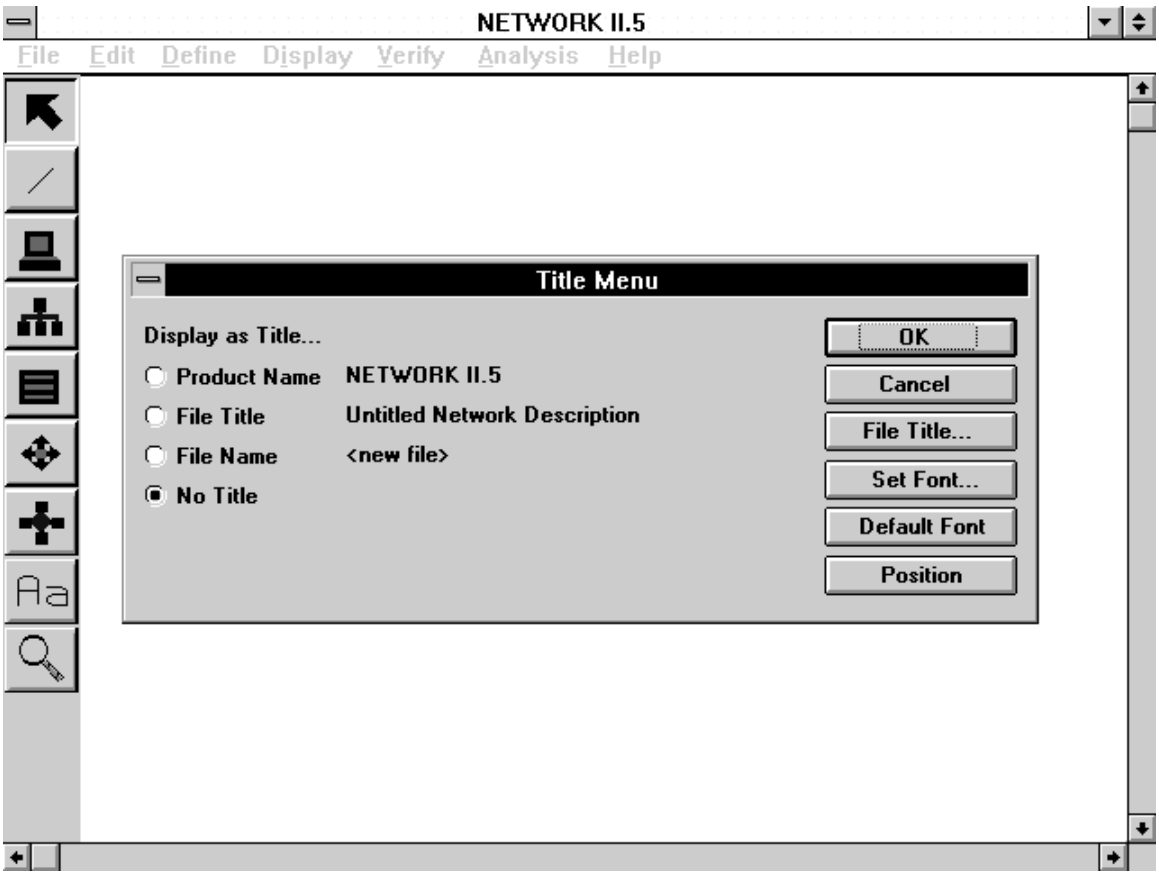


The Text Font Form

Display/Title Setup

The Display/Title Setup command allows you to select a title to be displayed. Available selections include the Product Name, the File Title, the File Name, or No Title. The File Title is the first line of the top model comment of a Network Description File. The top

model comment may be entered by clicking on the File Title button. The Position button will let you change the position of a title on the display. Use the Set Font and Default Font buttons to change the title's text font or restore it to the original default font.



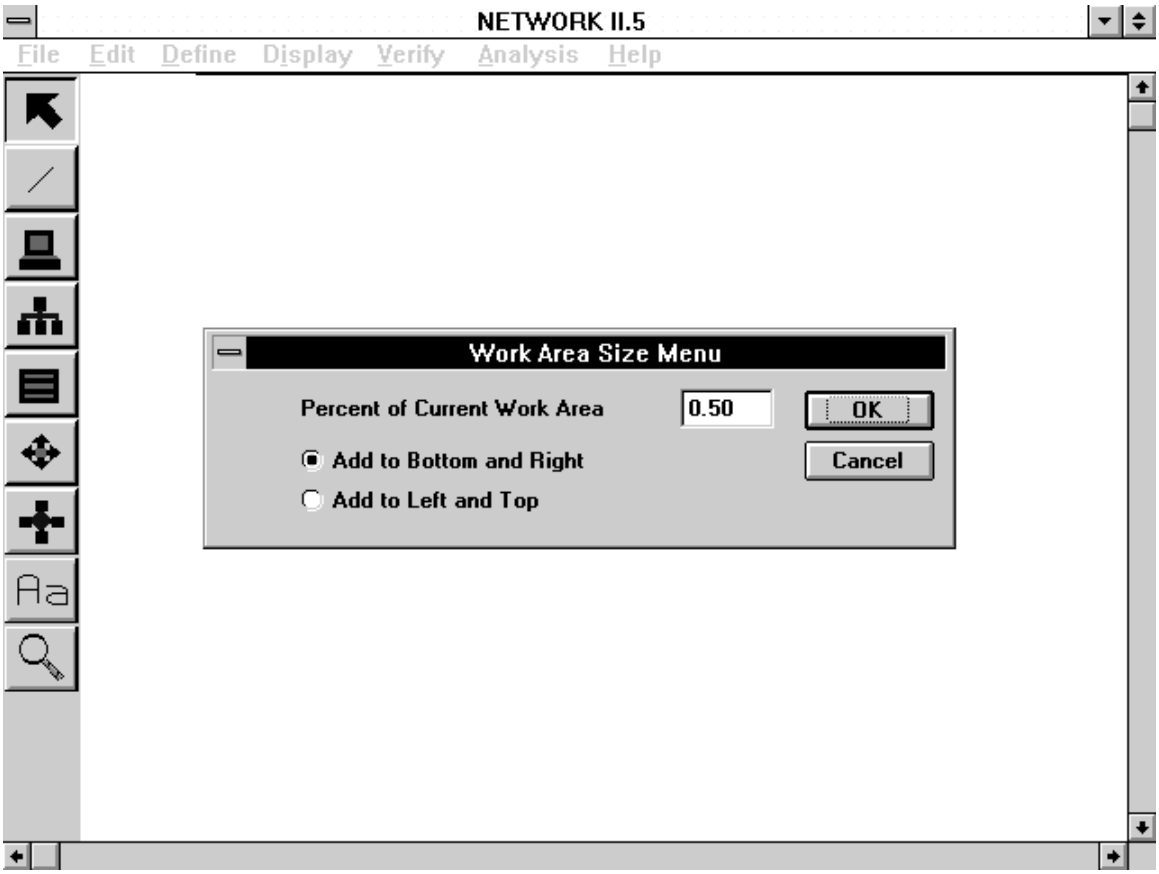
The Title Setup Form

Display/Refresh Screen

The Display/Refresh Screen command redraws the entire display with the current model drawing information. This command may be useful in cleaning up the display.

Display/Scale Diagram

The Display/Scale Diagram command allows you to change the magnification of objects on the display. Selecting the Display/Scale Diagram command has the same affect as clicking on the toolbar's Scale Button. After selecting this command, the Scale Selection form is presented. The diagram scale ranges from the platform minimum (Windows, UNIX- 1%, OS/2, DOS - 10%) to 100%. A value box is also provided for fine-tuning of the diagram scale. The default scale is 40 percent. To choose the diagram scale, click on the appropriate radio button and click on OK.

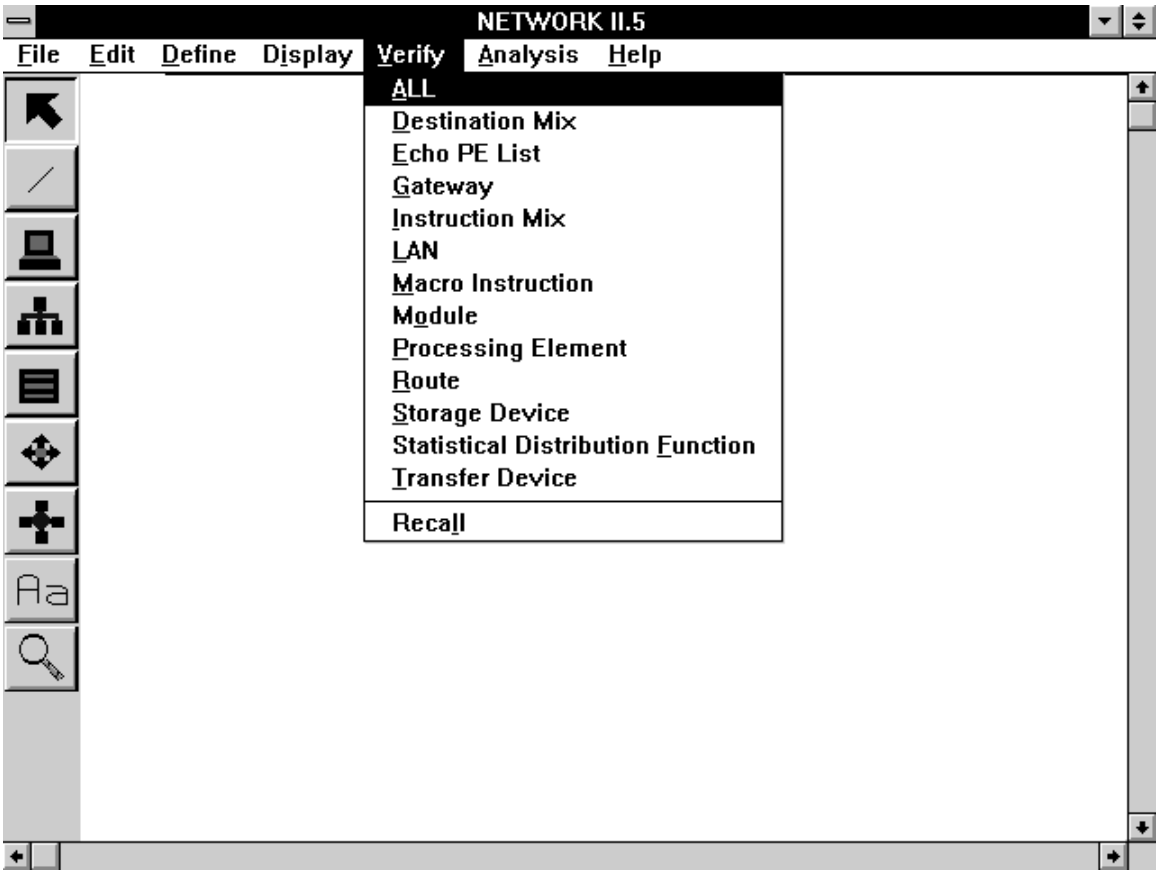


The Work Area Size Form

Display/Work Area Size

The Display/Work Area Size will let you expand the size of the canvas on which you can position a hardware layout. When you start NETWORK, the default size of the work area is 1.5 times the size of the display window, with the extra space defined to the right and below the display. Initially, the scroll box of the vertical scrollbar is at the extreme top and the scroll box of the horizontal scrollbar is at the extreme left. This means that if you scroll all the way to the right, you have shifted the display one half screen to the right, and if you scroll all the way to the bottom, you have shifted the display down one half screen. To increase the available work space, enter a larger number in the Percent Of Work Area Space box. You have two options of increasing the size of the work area, extra space may be added to the right and bottom of the display, or extra space may be added to the top and left of the display.

3.4.5 The Verify Menu

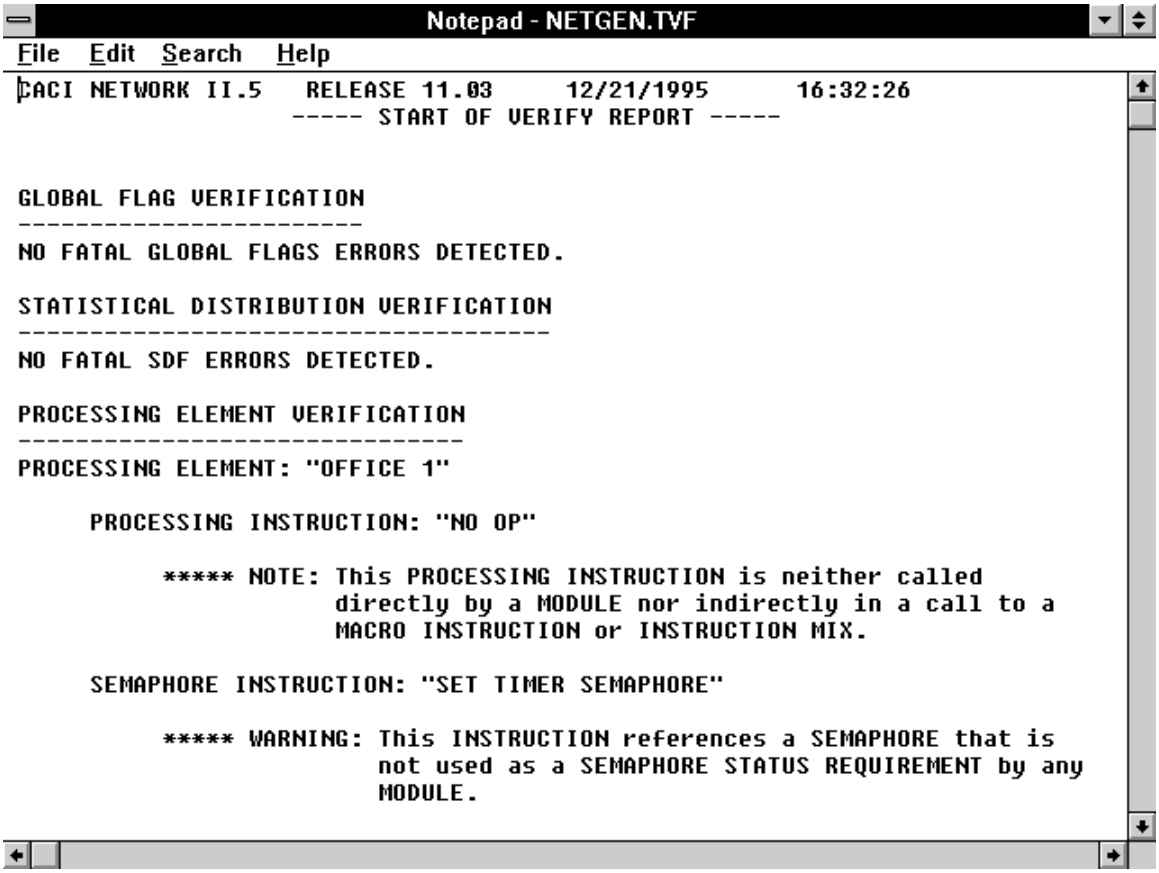


The Verify Menu

Before executing a simulation, you should verify your model to ensure that the model does not contain logic errors. During the process of verifying the model, NETWORK displays the Verify Progress Report. The first column in the report, Number Checked, notes how many data structures of the given type have been verified. The second column in the report, Errors, notes how many data structures of the given type contained errors. The third column in the report, Warnings, notes how many data structures of the given type contained warnings. The last row of the report presents the cumulative totals for number of data structures checked, number of data structures with errors, and number of data structures with warnings. The counts for SDFs and Echo PE Lists are included in the Top Level counts.

If errors or warnings were found in the model, select the Report button to view the verification report. If no verify errors were found, you may select the Done button to remove the Verify Progress Report. Errors should always be removed from a model before running a simulation. Warnings are produced in a verify report to note conditions

in the model which do not have to be changed before running a simulation. Warnings serve to alert you of these conditions which you may have overlooked, such as a Module having only a Start Time defined, which means that it will only execute once during a simulation.



A Verify Report

The Verify Report lists the status of the model components which have been verified. You will be notified of the names and types of the data structures that have been examined and any errors associated with each individual data structure. The Verify Report will be displayed in an editor native to the platform on which you are running NETWORK (Windows - Notepad, unix - vi).

Verify/ALL

The Verify/ALL command requests verification of the entire model. The Verify/ALL command is the only way to perform the top-level verify checks.

Verify/Destination Mix

Verify/Echo PE List

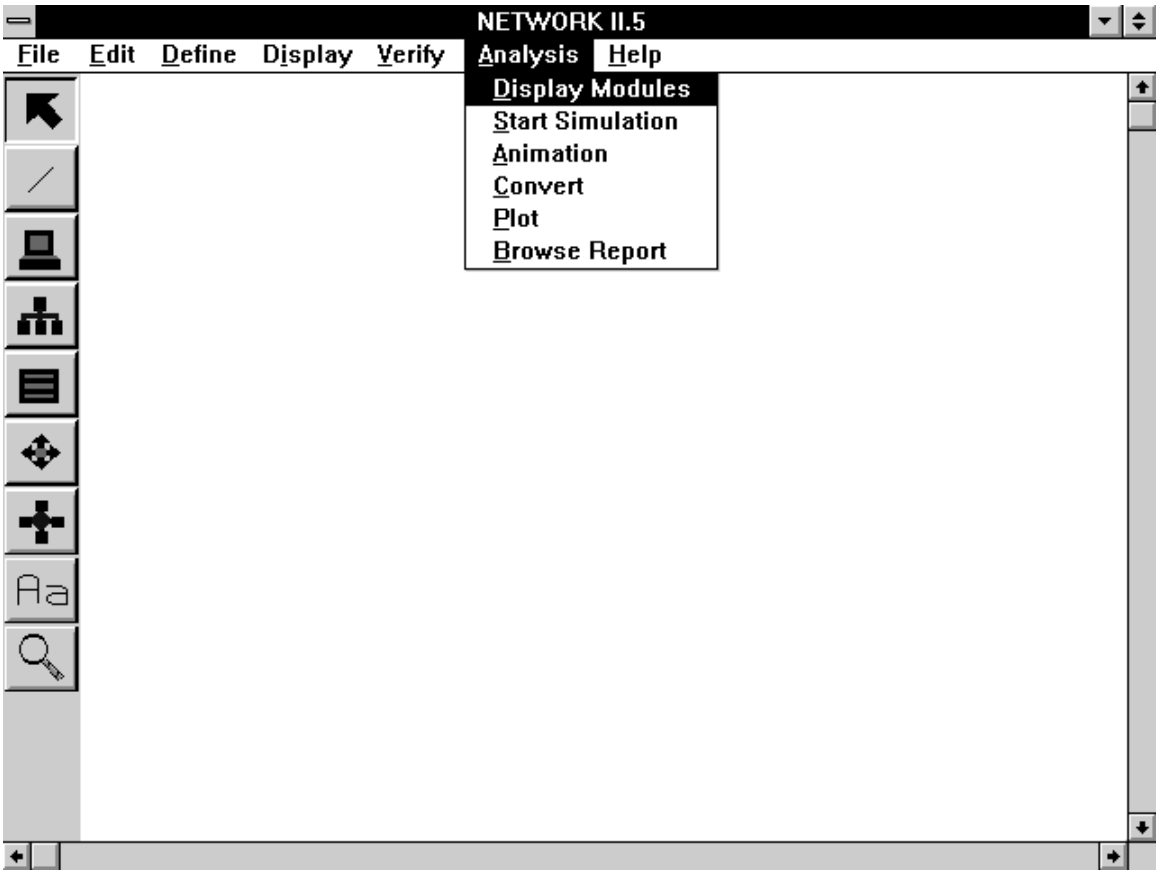
Verify/Gateway
Verify/Instruction Mix
Verify/LAN
Verify/Macro Instruction
Verify/Module
Verify/Processing Element
Verify/Route
Verify/Storage Device
Verify/Statistical Distribution Function
Verify/Transfer Device

To verify only those data structures of the same type, select that type from the Verify Menu. Verification will then proceed for all data structures of the given type. Thus, selecting the Verify/Module command will perform a verification on all Modules in your model. If three are no data structures of any of the given types, that type will not be selectable on the Verify Menu.

Verify/Recall

The Verify/Recall command redisplays the most recent verification report and is not available if verify has not been previously called.

3.4.6 The Analysis Menu



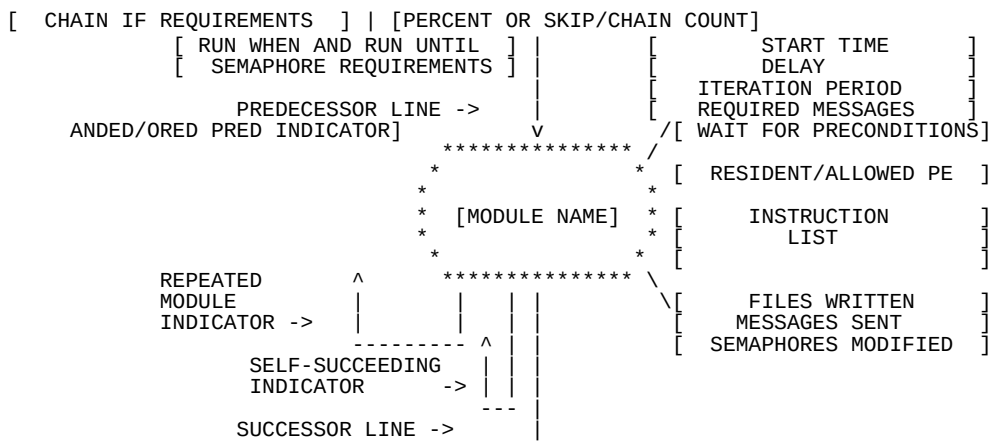
The Analysis Menu

Analysis/Display Modules

The Analysis/Display Modules command will create text diagrams of the Modules in your model. Four diagram options are available: Display All, Display by Message, Display by Semaphore, and Display by Name. Display All will generate a diagram of all Modules currently in your model. Display by Message will present a list of all current messages from which you may select a name. The Module diagram created will show all Modules that may transmit the selected message and all Modules that have this message as a Message Status Requirement, as well as all successors to the second group of Modules. Display by Semaphore will present a list of all Semaphores in the model from which you may select a name. The module diagram created will show all Modules that affect the status or count of the selected Semaphore and all Modules that have a Semaphore Status Requirement using this Semaphore, as well as all successors to the second group of Modules. Display by Name presents a list of all Module names from which you may pick a name to create a diagram of that Module only.

Each of the four choices will provide a diagram of one or more Modules. The diagram includes the Module's attributes and a chaining schematic. See the chart below for more

information on the contents of the Module diagram. The Module diagram will be presented in an editor which will allow you to scroll through the diagram or save it to a file.



Software Module Diagram Organization

Because of the limited space available for these text illustrations, some abbreviations are used in the module diagrams and text is frequently truncated. The Compress Display check box can be used to reduce the size of Module diagrams by eliminating unimportant diagram lines. The following table of abbreviations and conventions can be consulted when examining your Module diagrams.

Module Activation

ST: - Start Time IP: - Iteration Period

CI: - Chain If

A Semaphore, message or file name may appear after CI:.

RU: - Run Until RW: - Run When

A Semaphore name follows either RU: or RW:.

Message Status Requirements

The message name appears after the colon.

M: - Message LM: - Local Message

B: - Block LB: - Local Block

Semaphore Status Requirements

A Semaphore name appears before the colon.

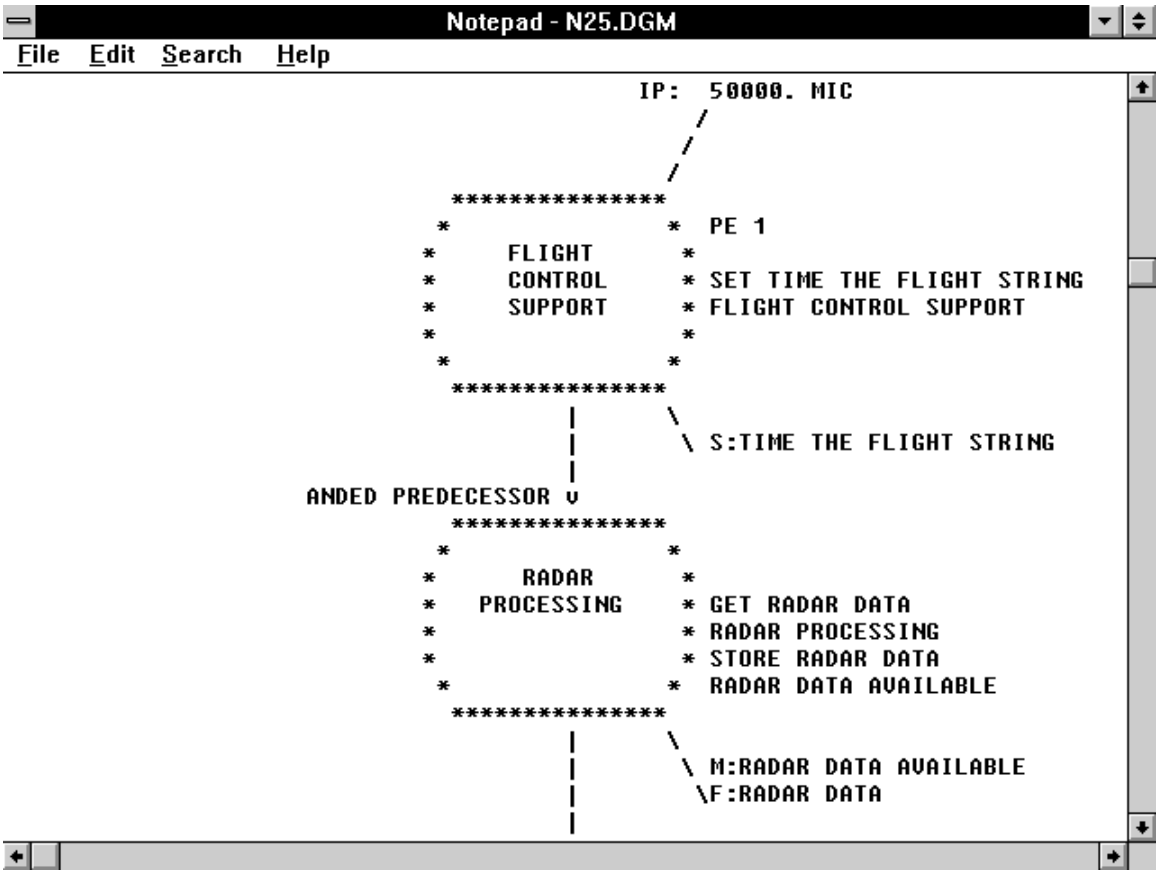
:S - Set :R - Reset
:INT or :I - Integer :(a number) - Semaphore Count

File Status Requirements
A file name appears before the colon.
:O - On :N - Not On

Hardware Status Requirements
Hardware data structure name appears before the colon.
:IDL - Idle :BSY - Busy
:COL or :COLLIS - Collision :OVR or OVRFLW - Overflow

Module Outputs
M: - Message, the name of the message sent by the Module.
F: - File, the name of the file written to by the Module.

A Module may affect the status and count of a Semaphore. Thus a letter or symbol or both may appear before a Semaphore name. (e.g. S+:)
S: - Set +: - increment Semaphore count
R: - Reset -: - decrement Semaphore count
T: - Toggle =: - set Semaphore count



A Sample Module Diagram

Analysis/Start Simulation

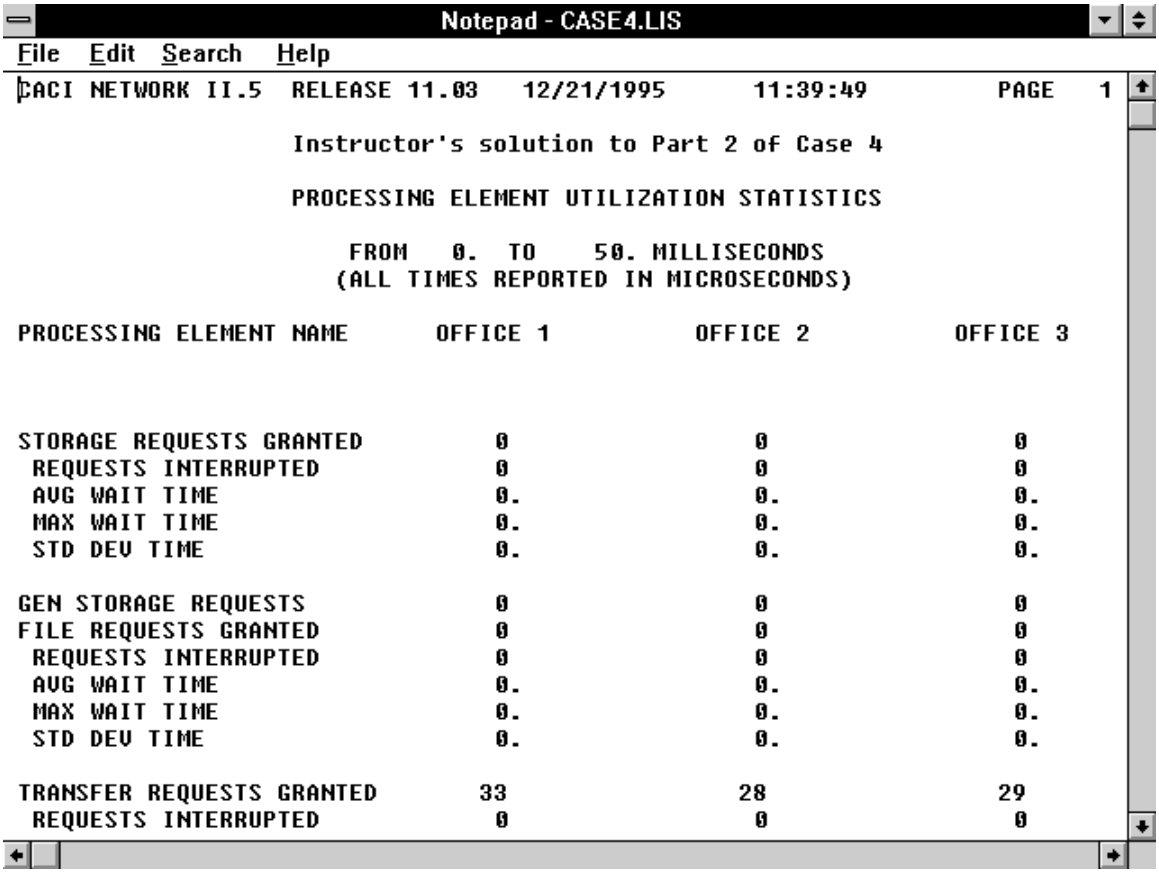
To run a simulation from NETWORK, select the Analysis/Start Simulation command which will display the Run Parameters form. Once you have set the run parameters to your satisfaction, select the Run button to actually start a simulation. The Run Parameters form and running simulations are described in Chapter 5.

Analysis/Animation

Click on the Analysis/Animation command to start an animation. To run an animation, fill in the animation parameters as described in Chapter 6.

Analysis/Convert

The Analysis/Convert command converts the Simulation Plot File (.PIN) between binary and text format. For more information, see Chapter 11 in this manual.



A Simulation Report File

Analysis/Plot

The Analysis/Plot command is used to generate graphical utilization Plots. When the Analysis/Plot command is selected, the Plot Menu is displayed. Fill in the plot

parameters as described in Chapter 7. Analysis/Plot is only selectable after a simulation has been run if a Simulation Plot File had been requested for the simulation.

Analysis/Browse Report
The Analysis/Browse Report command displays the Simulation Report File (.LIS) which was generated by the simulation engine for the current model. To use the Analysis/Browse Report command, you must have previously run a simulation for the current model. For example, if the model EXAMPLE.NET is the current model read in by NETWORK, the Analysis/Browse Report command will allow you to examine the contents of the file EXAMPLE.LIS. If a Simulation Report File (.LIS) does not exist for the current model, the Analysis/Browse Report command will not be selectable.

3.5 ATTRIBUTE FORMS

To completely define the data structures that comprise your NETWORK II.5 model, it will be necessary to provide values for the various attributes of the data structures. These attributes describe the behavior of an object during a simulation. Each data structure type has its corresponding attributes form which presents these attributes for examination and editing. Many of the attributes, such as a name, are common to all or most of the data structures. These comment attributes will be described in the first part of this section along with the most common forms control buttons. The remainder of this section will detail the attribute forms for each of the NETWORK II.5 data structures and how to edit these attributes.

The hardware data structures also have a set of graphics attributes which defines the appearance of an object on the display. The attributes and controls of these forms will be described after the section detailing all of the attributes forms of the various data structures.

3.5.1 Common Attributes and Controls

Name
A name identifies a data structure within a model. To change a name, enter a new name in the name text box. The name may be up to 40 characters long, and must be unique within the model. If a name is changed, any occurrences of the name will automatically be changed. For example, if you change the name of a PE, any Message Instructions which had the old PE name as a Destination will be changed so that they use the new name.

Comment

Each data structure may have a comment that describes it in further detail or holds any type of note you may wish to include. To enter or change a comment, select the Comment button and then make your changes in the Comment form. A comment may be as many lines long as you wish. If a data structure is given a comment, the first line of the comment will be displayed on the attributes form as a label to the left of the Comment button.

Time Units

Many of the attributes of data structures are related to time. You may change the time units used in a form by selecting new time units from the Time Units combo box. Time units may be set to seconds, milliseconds, microseconds, or nanoseconds. Any time values displayed on a form will be scaled to the new units. If you have specific time values to enter into a form given in a certain time unit, it is easier to convert the time units on the form than it is to convert the numbers to a time unit.

Include In Plot File

This check box is provided so that you can determine if an individual item is to be included in or excluded from the plot file. To include a data structure in the plot file, make sure the check box is marked. The Controls Form also has similar check boxes which can be used to mark whether or not an entire group of data structures will be included in the plot file. These check boxes are useful if you wish to ignore plot data on less important modeling objects and to restrict the size of the plot file.

OK

To accept changes that have been entered in a form, click on the OK button. When you OK the form, NETWORK may check the validity of your entries, such as checking a new name to make sure that it conforms to NETWORK II.5 naming rules. If an invalid entry is found, NETWORK will display a form to explain what new entries must be changed. In this case, your OK will be cancelled and you will be returned to the attributes form to make any corrections.

Cancel

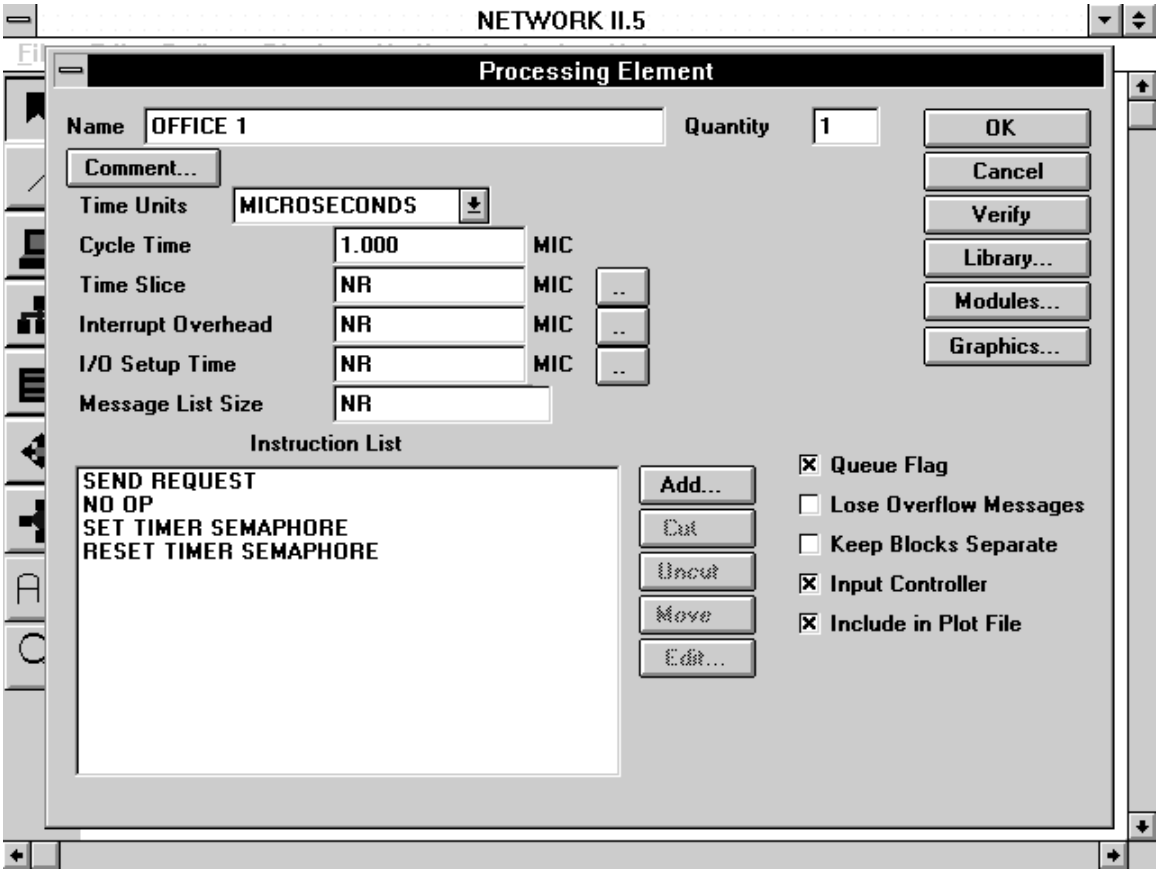
To reject changes that have been entered in a form, click on the CANCEL button. If you cancel from the PE Form, changes made to the PE's drawing information and to Modules that may run on the PE will be retained.

Verify

A data structure may be verified individually by clicking on the Verify button.

The Library button will display the Library Form which has a variety of functions related to the creation, use and maintenance of data structure library files. The Library command is available for Modules, Processing Elements, Storage Devices and Transfer Devices only. The last section of this chapter describes how to use the Library command.
To make changes to the drawing information of a hardware data structure (PE, Gateway, SD, TD, or LAN) select the Graphics button from within the initial attributes form. The Graphics form will then be displayed from which you can change the graphics characteristics of an object, such as changing color or selecting a new display icon.**Processing Element Form**

The Processing Element form lists a PE's current attributes. To modify the attributes, enter new parameters and select OK.



The Processing Element Form

Quantity

The PE quantity determines if there is a single PE (PE quantity is 1) or a group of identical PEs (PE quantity is greater than 1). The PE quantity can be any integer value from 1 to 999.

Cycle Time
The Cycle Time must be a real number greater than or equal to zero.

Time Slice
The Time Slice may be a real number or may be chosen from a SDF. To use a SDF, either enter the name of a SDF in the text box, or click on the double-dots button next to the text box.

Interrupt Overhead
The Interrupt Overhead may be a real number or may be chosen from a SDF. To use a SDF, either enter the name of a SDF in the text box, or click on the double-dots button next to the text box.

I/O Setup Time
The I/O Setup Time may be a real number or may be chosen from a SDF. To use a SDF, either enter the name of a SDF in the text box, or click on the double-dots button next to the text box.

Message List Size
The Message List Size must be a real number or NR (No Response). If Message List Size = NR, it has an infinite capacity.

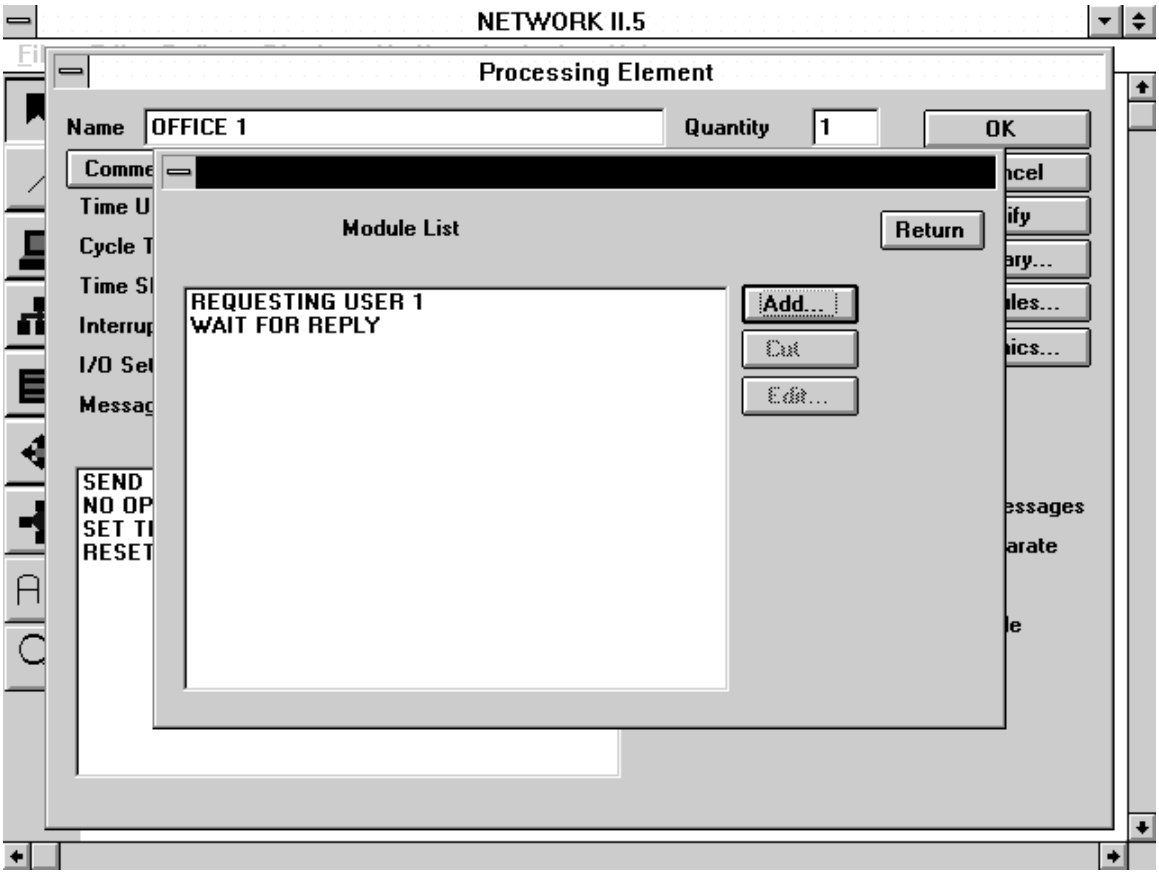
Queue Flag
A PE's Queue Flag is set to YES when the check box is selected.

Lose Overflow Messages
A PE's Lose Overflow Messages flag is set to YES when the check box is selected.

Keep Blocks Separate
A PE's Keep Blocks Separate flag is set to YES when the check box is selected.

Input Controller
A PE's Input Controller flag is set to YES when the check box is selected.

Modules
You may modify Modules that operate on a PE, or create new Modules by selecting the Modules button. The current list of Modules that may run on the PE is shown when the Modules button is selected. This list includes Modules that have the PE as an Allowed or Resident PE, and all the successors of these Modules. Modules may be edited from this list or created and added to the list. If a Module that appeared in the list is edited so that it no longer runs on the PE, its name is removed from the list and any successors it called may be removed from the list.



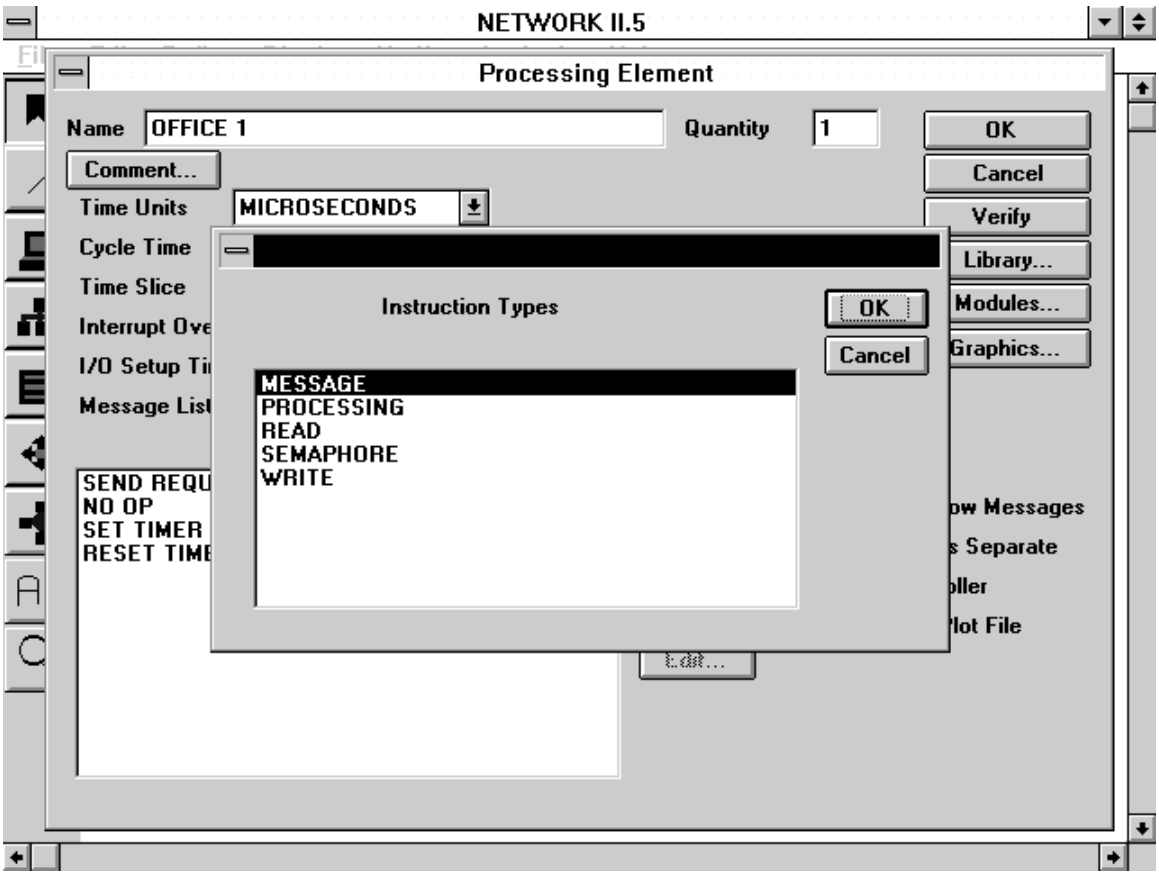
The Modules List Form

Instruction List

The Instruction List contains the names of instructions used by a PE. You may use the Instruction List to create, modify and delete PE instructions. Four control buttons are provided for the Instruction List; Add, Cut, Uncut, Move and Edit.

To add a new instruction to a PE, click on the Add button. You will be asked to enter a new name or select from a list of instruction names. This name list includes Module instruction names taken from Modules that may run on the PE and for which a corresponding instruction name does not yet exist in the PE Instruction List. After you have chosen a name, you will be asked to select the instruction type for the new instruction (either read, write, message, semaphore or processing) from the Instruction Types Form. If you cancel from the Instruction Types Form, the new instruction is discarded. When a new instruction is given a type, it will be entered in the Instruction List of the PE.

To complete the definition of a new instruction, or to examine or change the attributes of any PE instruction, highlight the name in the Instruction List and then select the Edit button. The attributes form for the selected instruction will then be displayed.



The Instruction Types Form

To delete an instruction from the PE Instruction List, highlight the name in the list and select the Cut button. Cut instructions can be restored to a PE Instruction List by selecting the Uncut button and then selecting names from the list of cut instructions. The Cut button will be selectable if instructions have been cut from the PE in the current NETWORK session.

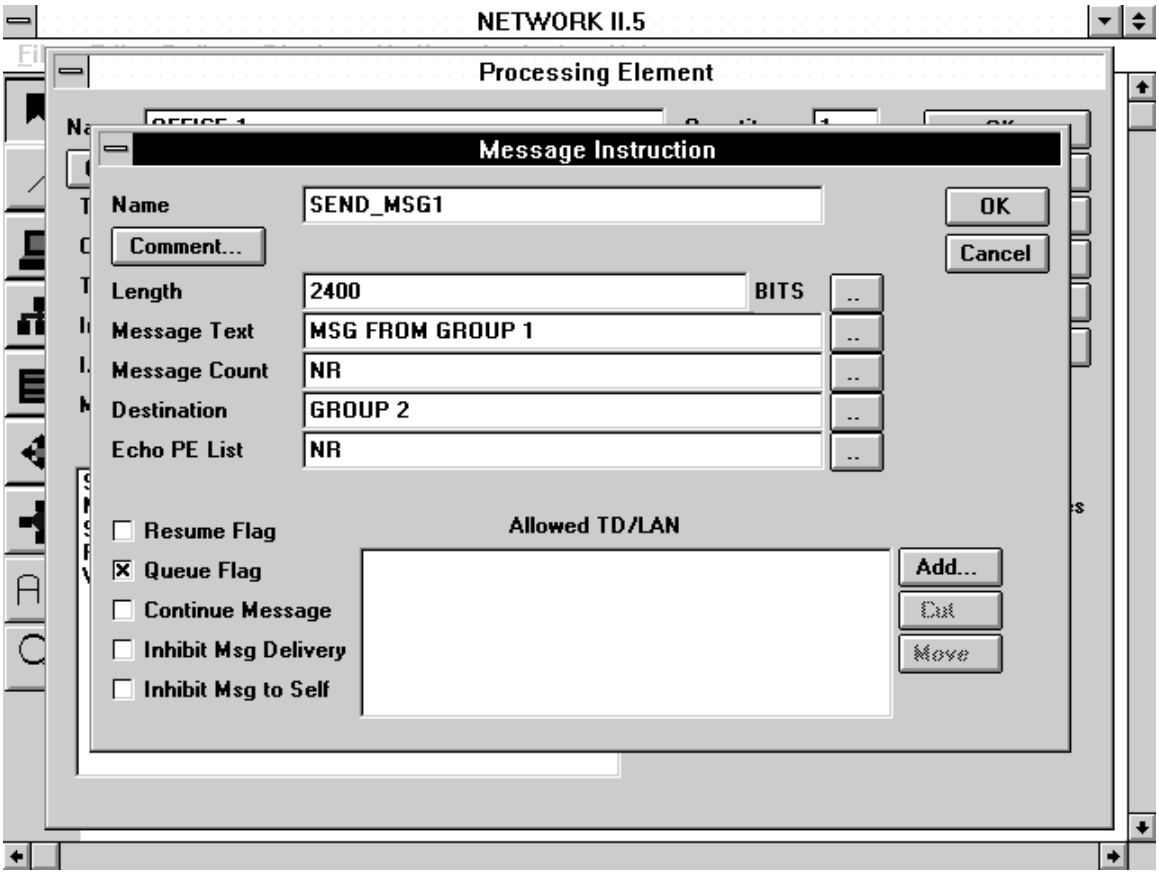
3.5.2.1 Instructions

Processing Elements may use five types of instructions: processing, message, read, write, and semaphore. The attribute forms for each type are described next.

3.5.2.1.1 Message Instruction Form

Length

The message length may be an integer number or may be chosen from a SDF. To use a SDF, either enter the name of a SDF in the text box, or click on the double-dots button next to the text box.



The Message Instruction Form

Message Text

The message text may be up to 40 characters long. If the value NR is selected, the message text will default to the message name in the simulation. To choose from a list of current messages in the model, click on the Message Text double-dots button.

Message Count

The Message Count can be an integer value or the name of a Statistical Distribution Function. Select the Message Count double-dots button to choose a current SDF name.

Destination

You may enter the message destination directly, or choose from the list of available destinations by clicking on the Destination double-dots button.

If the destination does not exist in the model, a new destination will be created. You will be asked to select the destination type (PE, Route, or Destination Mix) when you click on the OK button.

Echo PE List
You may enter the name of an echo PE list directly, or choose from the list of available destinations by clicking on the Echo PE List double-dots button.

Resume Flag
A Message Instruction's Resume Flag will be set to YES when this check box is selected.

Queue Flag
A Message Instruction's Queue Flag will be set to YES when this check box is selected.

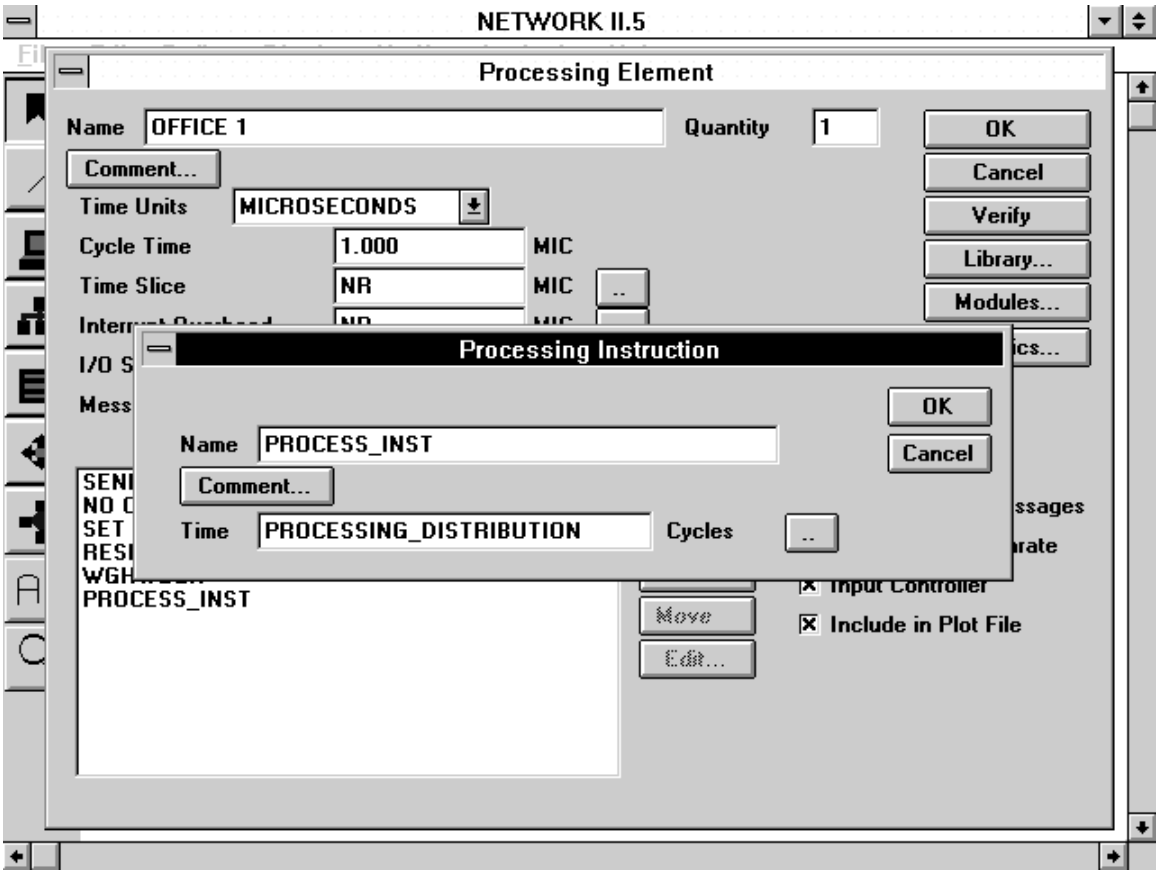
Continue Message Flag
A Message Instruction's Continue Message Flag will be set to YES when this check box is selected.

Inhibit Msg to Self
A Message Instruction's Inhibit Message to Self Flag will be set to YES when this check box is selected.

Inhibit Msg Delivery
A Message Instruction's Inhibit Message Delivery Flag will be set to YES when this check box is selected.

Allowed TD/LAN
You may use the Allowed TD/LAN List to add and delete allowed TDs/LANs for the Message Instruction. The creation, deletion and positioning of entries in the Allowed TD/LAN List is controlled by the Add, Cut and Move buttons to the right of the list.

3.5.2.1.2 Processing Instruction Form

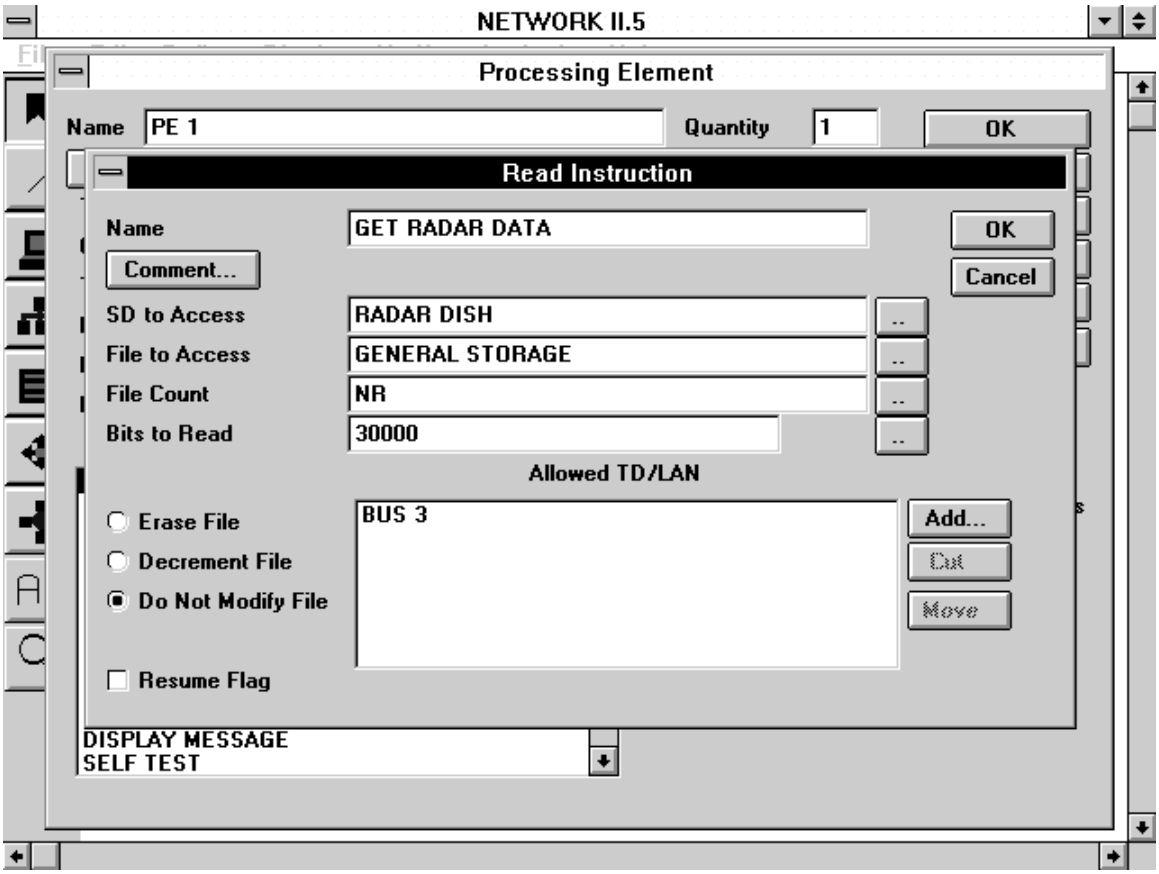


The Processing Instruction Form

Time

The processing time may be a real number or chosen from a Statistical Distribution Function. To use a SDF, either enter the name of a SDF in the text box, or click on the double-dots button next to the text box for a list of SDFs.

3.5.2.1.3 Read Instruction Form



The Read Instruction Form

SD to Access

The Storage Device name may be up to 40 characters long. To change the SD name, either enter a name in the text box, or click on the double-dots button next to the text box to choose from the list of current SD names. If the name of a nonexistent SD is entered, a new one will be created when you OK the form.

File to Access

The file name may be up to 40 characters long. To change the file name, either enter a name in the text box, or click on the File to Access double-dots button to choose a name from the list of current files. If you enter the name of a nonexistent file, a new one will be created when you OK the form.

File Count

The File Count can be an integer value or the name of a Statistical Distribution Function. Select the File Count double-dots button to choose a current SDF name.

Bits to Read

The number of bits read may be an integer number or may be chosen from a SDF. To use a SDF, either enter the name of a SDF in the text box, or click on the double-dots button next to the text box.

File Modification Method

One of three file modification methods may be chosen from the group of radio buttons. These are Erase File, Decrement File, and Do Not Modify File.

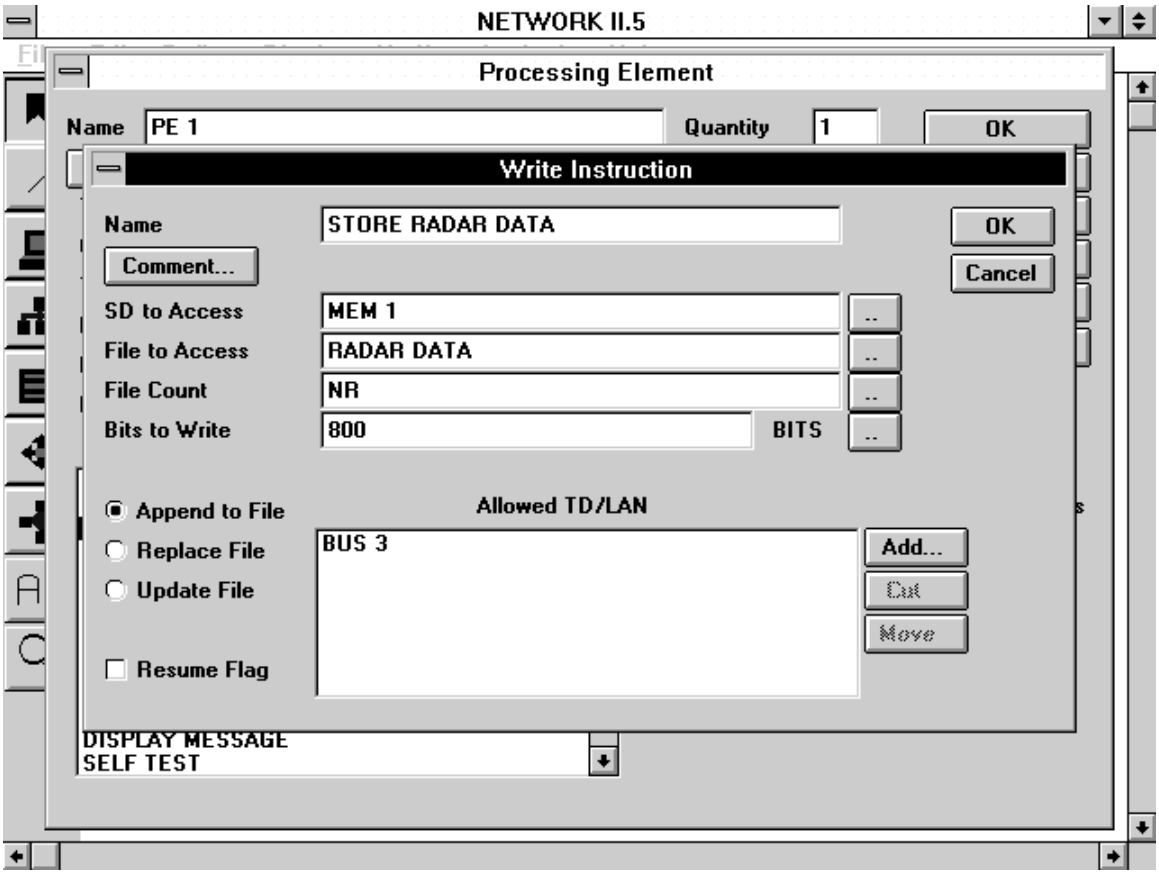
Resume Flag

A Read Instruction's Resume Flag will be set to YES when this check box is selected.

Allowed TD/LAN

You may use the Allowed TD/LAN List to add and delete allowed TDs/LANs for the Read Instruction. The creation, deletion and positioning of entries in the Allowed TD/LAN List is controlled by the Add, Cut and Move buttons to the right of the list.

3.5.2.1.4 Write Instruction Form



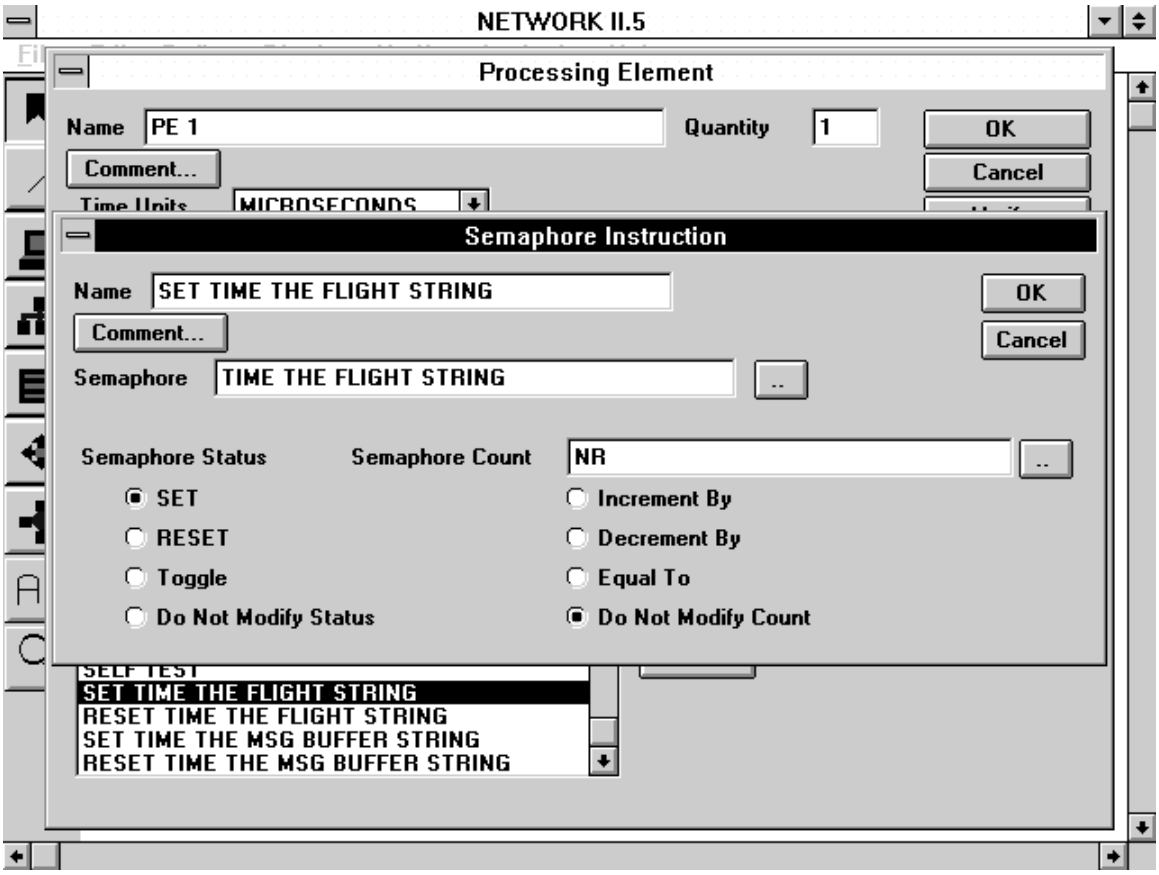
The Write Instruction Form

The attributes on the Write Instruction form are nearly identical to the attributes of the Read Instruction (see directly above) with one exception, the options for modifying files are different. All other attributes are the same.

File Modification Method

One of three file modification methods may be chosen. These are Replace File, Update File, and Append to File.

3.5.2.1.5 Semaphore Instruction Form



The Semaphore Instruction Form

Semaphore

The name of the Semaphore to be accessed by this instruction may be up to 40 characters long. To change the Semaphore name, either enter a name in the text box, or click on the double-dots button next to the text box. If the name of a nonexistent Semaphore is entered, a new one will be created when OK the form.

Semaphore Status

One of four Semaphore status modification methods may be chosen. These are Set, Reset, Toggle, and Do Not Modify.

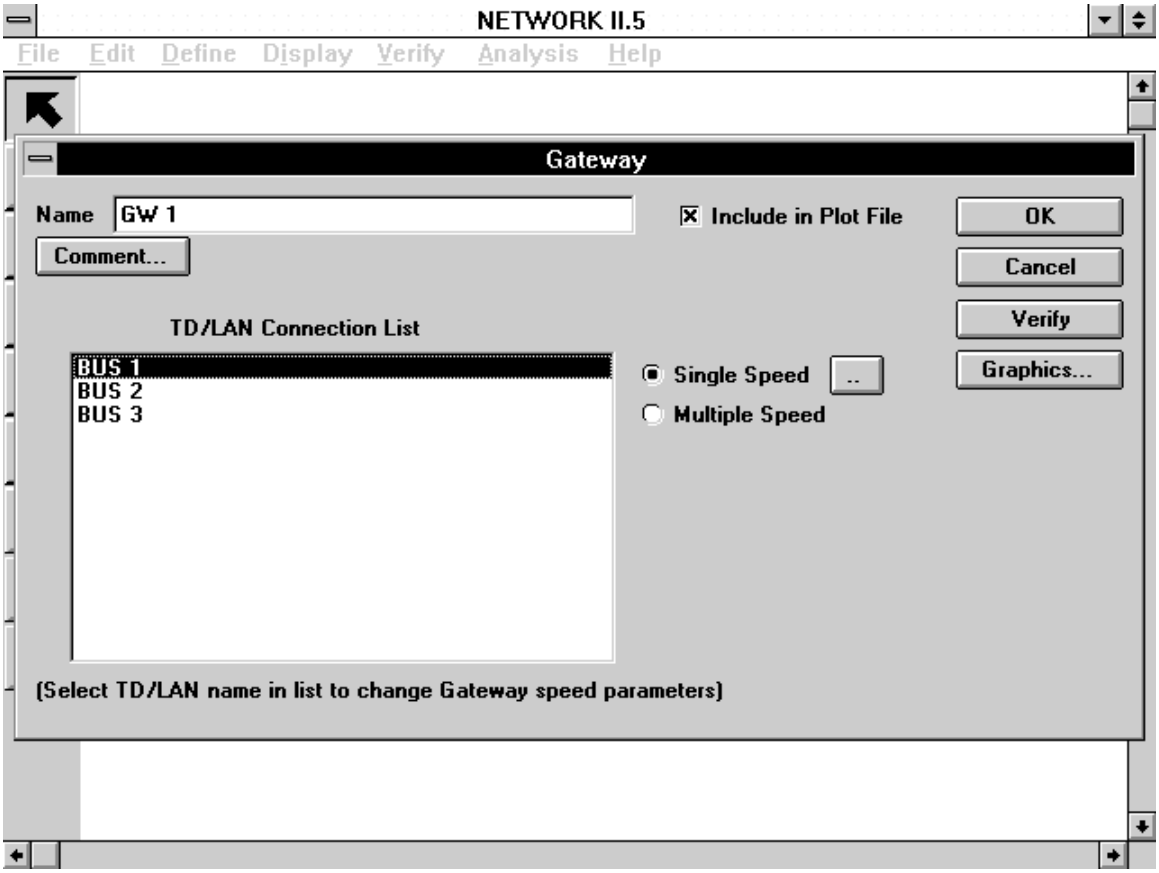
Semaphore Count

To modify the Semaphore count, you need to specify the modification value in the Semaphore Count text box, and the Semaphore modification value. Four modification methods are available. These are Increment By, Decrement By, Equal To, and Do Not Modify. The Semaphore modification value is used to alter the semaphore count,

depending on the modification method. The Semaphore modification value may be a real number or may be derived from a SDF. To use a SDF, either enter the name of a SDF in the text box, or click on the double-dots button next to the text box.

3.5.3 Gateway Form

The Gateway Form lists a Gateway's current attributes. To modify the attributes, enter new parameters and select OK.



The Gateway Form

TD/LAN Connection List

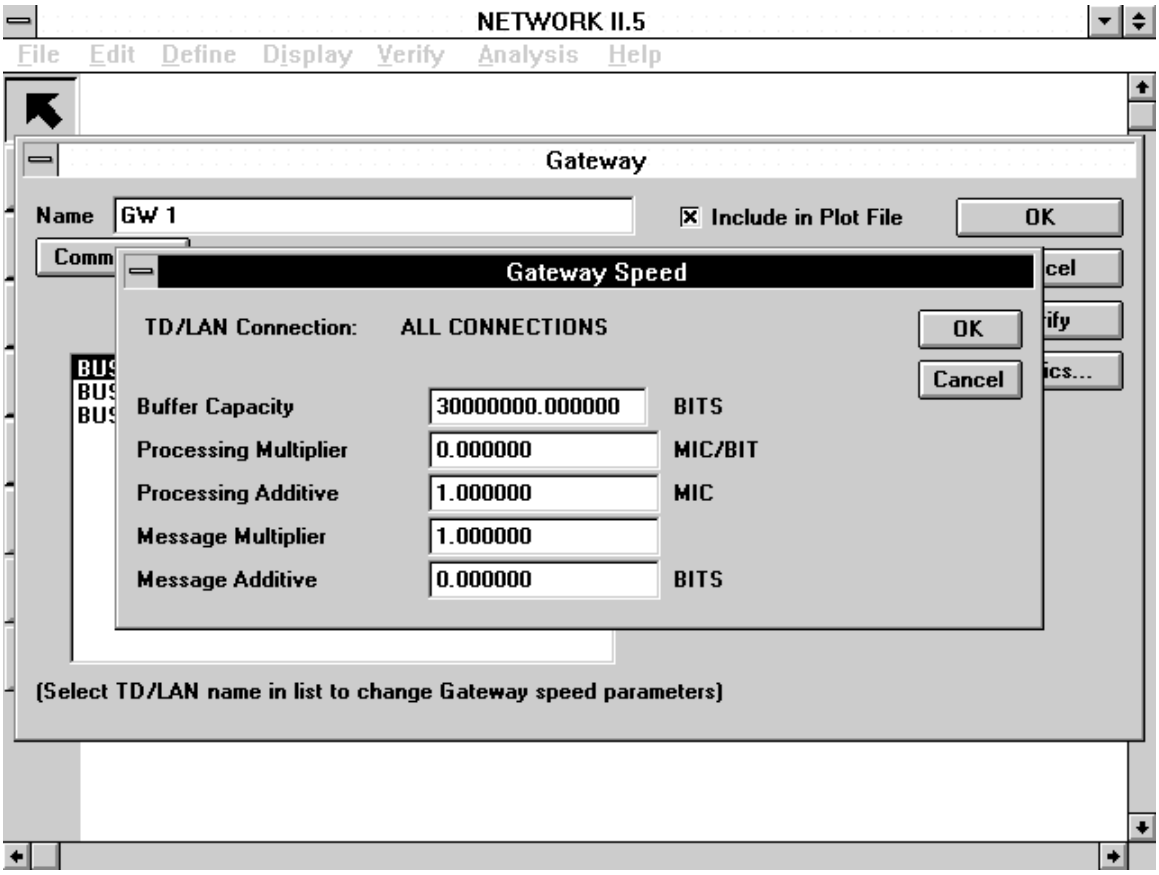
The TD/LAN connection list identifies the TDs and LANs that the Gateway is connected to. Names may only be added to this list by creating LAN connections, either graphically on the display, or else by adding the name of this Gateway to the Connection List of a TD/LAN. To edit the Gateway speed parameters, select a name in the list.

Gateway Speed

The Gateway speed determines how the processing overhead, message reformatting overhead, and buffer capacity are applied to each TD/LAN connection. If the Gateway is single speed, the overhead and buffer parameters are the same for all TD/LAN connections If the Gateway is multiple speed, the overhead and buffer parameters may be specified individually for each LAN connection. To edit the Gateway speed

parameters, click on a name in the TD/LAN Connection List. Alternatively, for a single speed Gateway, you may edit the Gateway speed parameters by clicking on the double-dots button next to the Single Speed radio button.

3.5.3.1 Gateway Speed Parameters Form



The Gateway Speed Parameters Form

Buffer Capacity
The Buffer Capacity must be a real number. The unit of the Buffer Capacity is bits.

Processing Multiplier
The Processing Multiplier must be a real number. The unit of the Processing Multiplier is microseconds per bit.

Processing Additive
The Processing Additive must be a real number. The unit of the Processing Additive is microseconds.

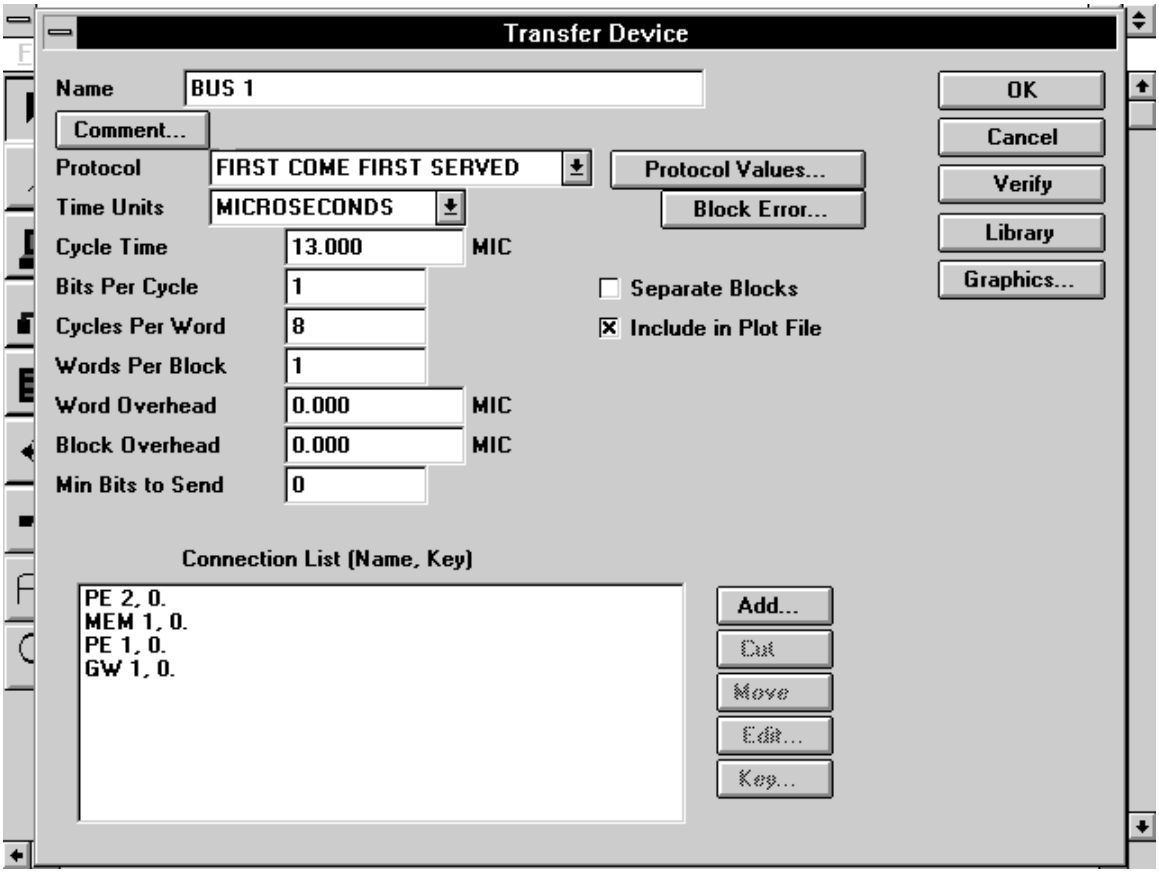
Message Multiplier
The Message Multiplier is a unitless quantity that must be a real number.

Message Additive

The Message Additive must be a real number. It is measured in bits.

3.5.4 Transfer Device Form

The Transfer Device form is used to display and change the attributes of a selected Transfer Device. Protocol independent TD attributes are displayed on the TD form and protocol specific attributes are displayed in subforms for the various protocols.



The Transfer Device Form

Protocol
To establish a protocol for the Transfer Device, click on the down-arrow of the Protocol combo box and select one of the protocol choices from the list.

Cycle Time
The cycle time must be a real number.

Bits per Cycle
The bits per cycle must be an integer value.

Cycles per Word
The cycles per word must be an integer value.

Words per Block
The words per block must be an integer value.

Word Overhead
The word overhead parameter must be specified as a real value.

Block Overhead
The block overhead parameter must be specified as a real value.

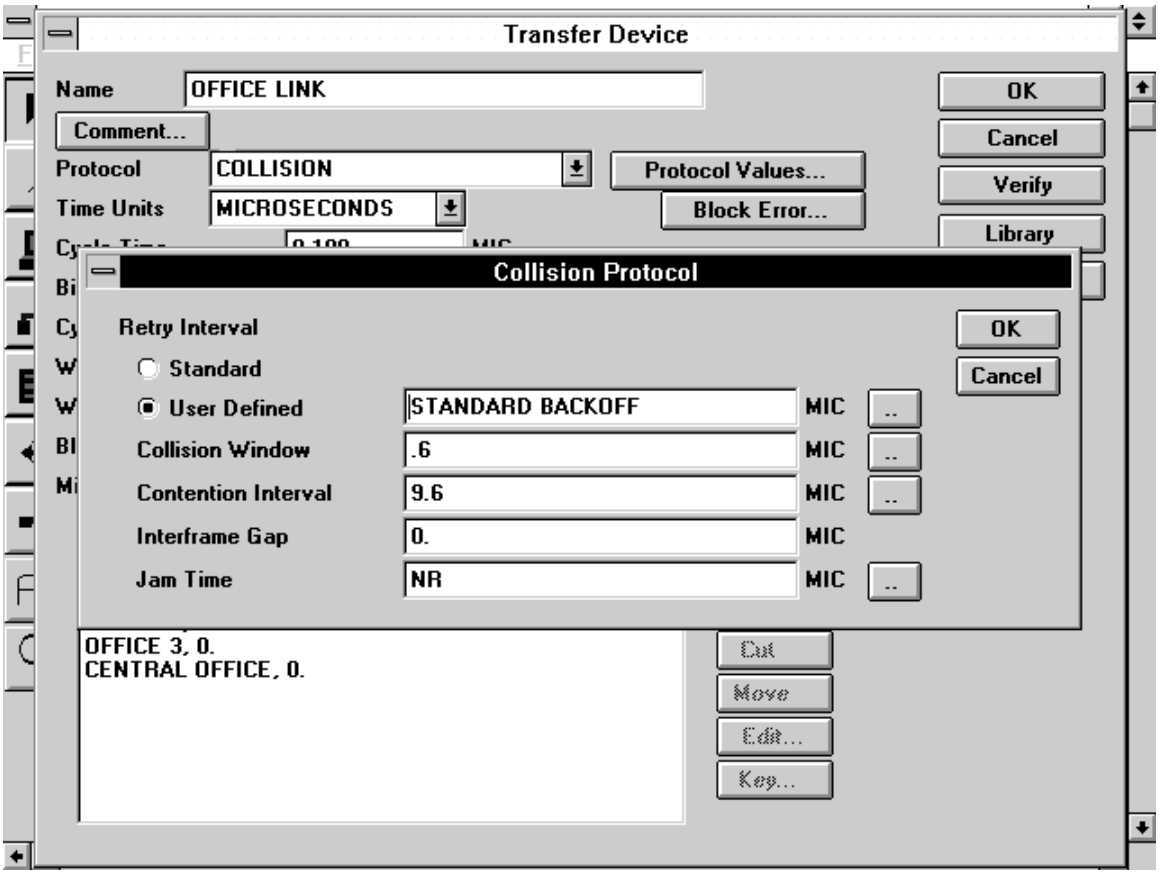
Min Bits to Send
The minimum bits per transfer (min bits to send) parameter must be specified as an integer value.

Separate Blocks
A TD's Separate Blocks flag is set to YES when this check box is selected.

Connection List
The Connection List displays all of the hardware data structures (Pes, Gateways and Storage Devices) that are connected to the TD. Use the Add, Cut and Move buttons to add connections, cut names from the Connection List and to position names in the Connection List. The Edit button is only selectable if the TD protocol is Priority Token Ring because connections of a TD having this protocol are given additional attributes. The Key button will let you change the Key values of the connections.

3.5.4.1 Protocol Specific Attributes

Select the Protocol Attributes button to edit protocol specific attributes. A protocol must have been selected for the TD before attempting to edit the attributes.



A Protocol Specific Attributes Form

First Come First Served

The First Come First Served protocol has no protocol specific attributes.

Collision

Retry Interval

The Retry Interval must be specified as a Statistical Distribution Function. Either enter a SDF name in the text box, or click on the double-dots button to select from a list of current SDFs.

Collision Window, Contention Interval, Jam Time

These parameters may be a real number or may be chosen from a SDF. To use a SDF, either enter the name of a SDF in the text box, or click on the double-dots button next to the text box.

Interframe Gap

The interframe gap must be specified as a real number.

Token Ring

Token Passing Time

This parameter may be a real number or may be chosen from a SDF. To use a SDF, either enter the name of a SDF in the text box, or click on the double-dots button next to the text box.

Slotted Token Ring

Slot Time, Token Passing Time

These parameters may be a real number or may be chosen from a SDF. To use a SDF, either enter the name of a SDF in the text box, or click on the double-dots button next to the text box.

TD Connection Key Values

If a key linear distribution is used for the slot time parameter, the TD connection key values are used as x in an Ax + B calculation. To enter a key value, click on the key number next to the appropriate connection and enter a new value.

Priority

TD Connection Key Values

In a priority TD, the key values are used to assign the PE priority. To enter a key value, click on the key number next to the appropriate connection and enter a new value.

Priority Token Ring

Token Passing Time

These parameters may be a real number or may be chosen from a SDF. To use a SDF, either enter the name of a SDF in the text box, or click on the double-dots button next to the text box.

The following attributes of a Priority Token Ring TD are found in the TD Connection Menu, which is accessed by selecting a name from the TD Connection List and then selecting the Edit button.

Synchronous Allocation, Target Token Rotation Times

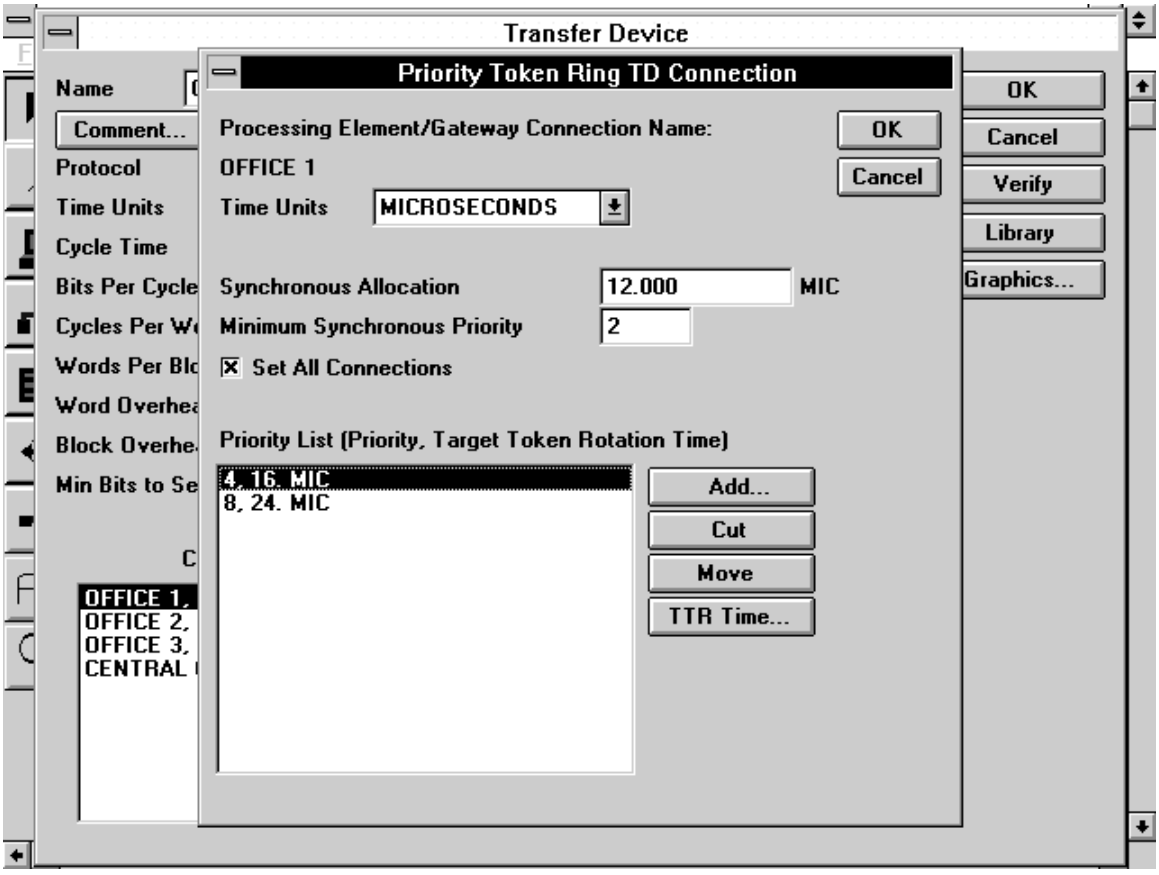
These parameters must be specified as real numbers.

Minimum Synchronous Priority, Priority

These parameters must be specified as integer numbers.

Set All Connections

If this flag is selected, the synchronous allocation, minimum synchronous priority, priority, and target rotation time parameters will be used for all TD connections.



Priority Token Ring TD Connection Form

Aloha

Retry Interval

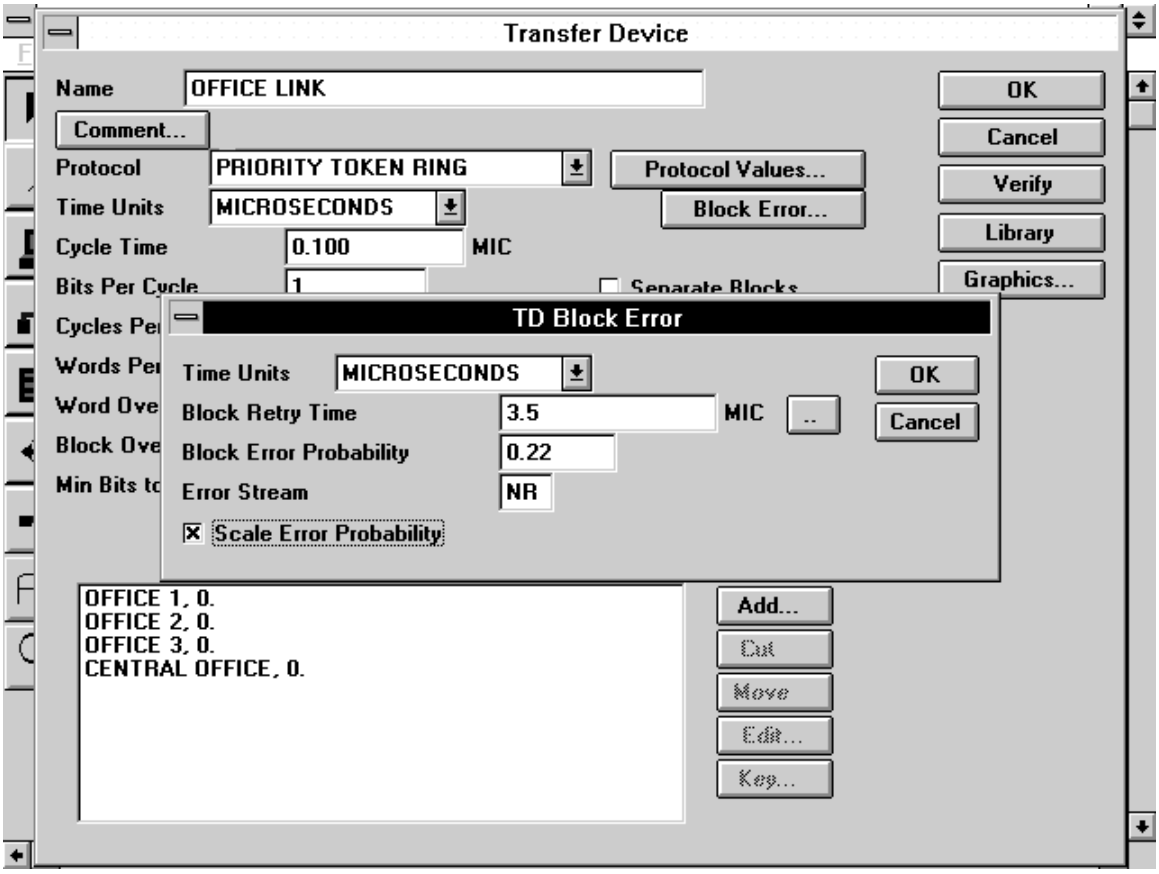
The retry interval must be specified as a statistical distribution function. Either enter a SDF name in the text box, or click on the double-dots button to select from a list of existing SDFs.

Automatic Retry

An Aloha TD's Automatic Retry Flag will be set to YES when this check box is selected.

3.5.4.2 Block Error Form

You may set several values related to transmission errors for the TD. To do this, select the Block Error button and then make changes on the Block Error Form.



The Block Error Form

Block Retry Time

The block retry time parameter must be specified as a real value.

Block Error Probability

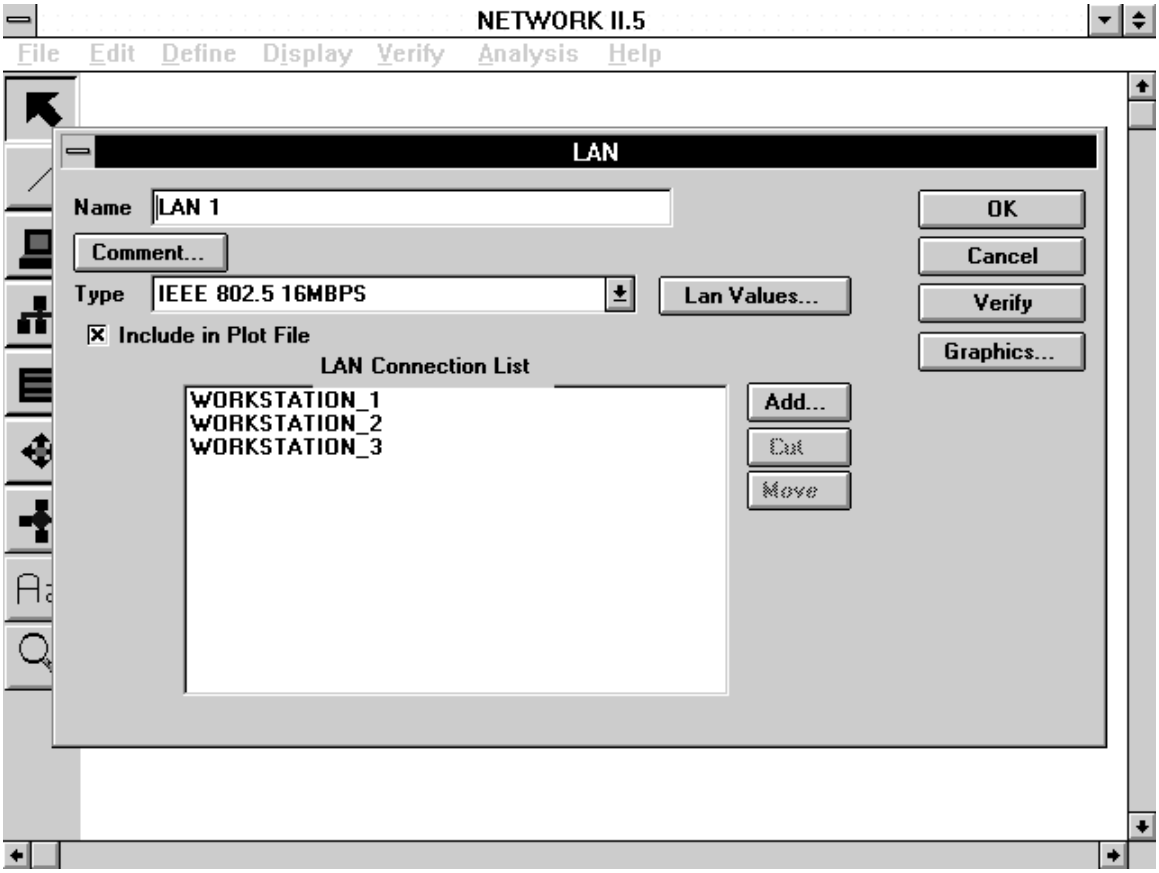
The block error probability parameter must be specified as a real value.

Error Stream
The error stream is an integer value that must be in the range from 1 to 999. If you leave the error stream in the NR state, an error stream value will automatically be chosen by the simulation program.

Scale Error Probability
A TD's Scale Error Probability flag is set to YES when this check box is selected.

3.5.5 LAN Form

The LAN Form is used to display and change the attributes of a selected LAN.

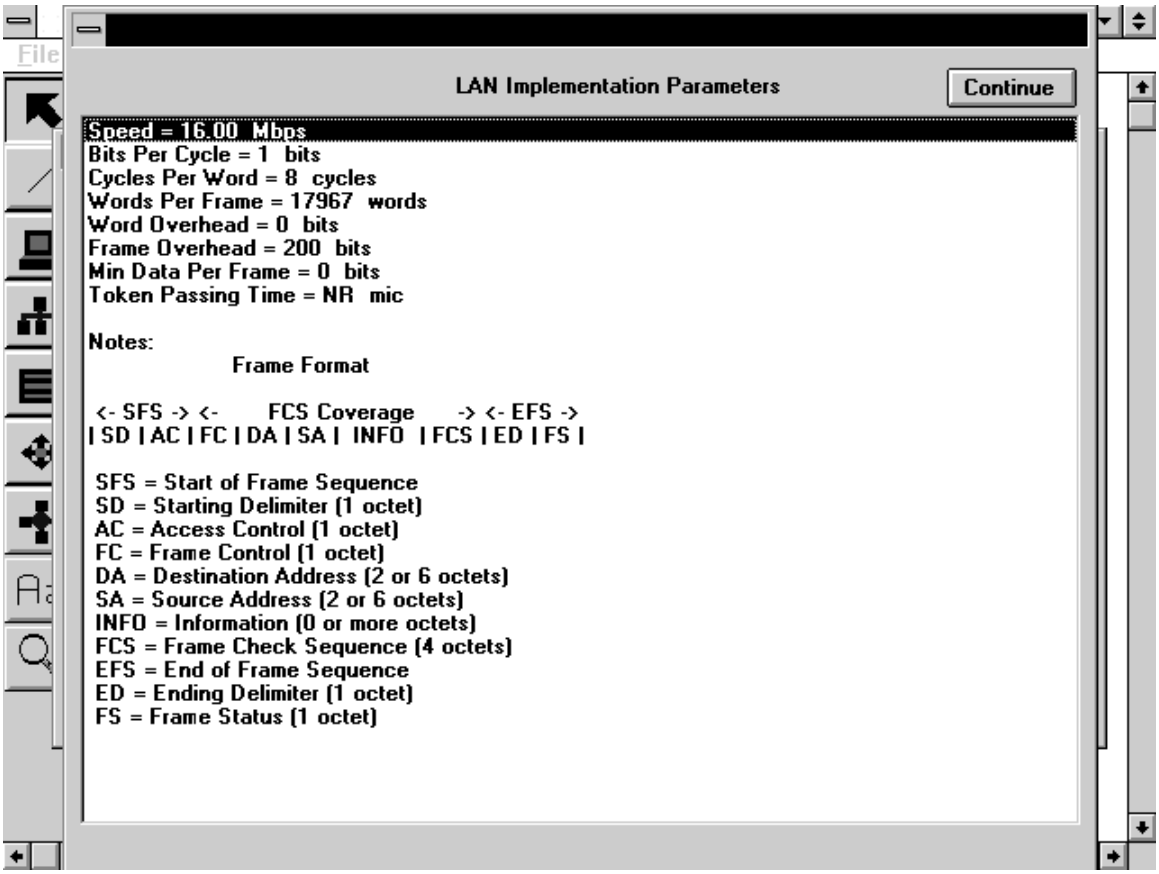


The LAN Form

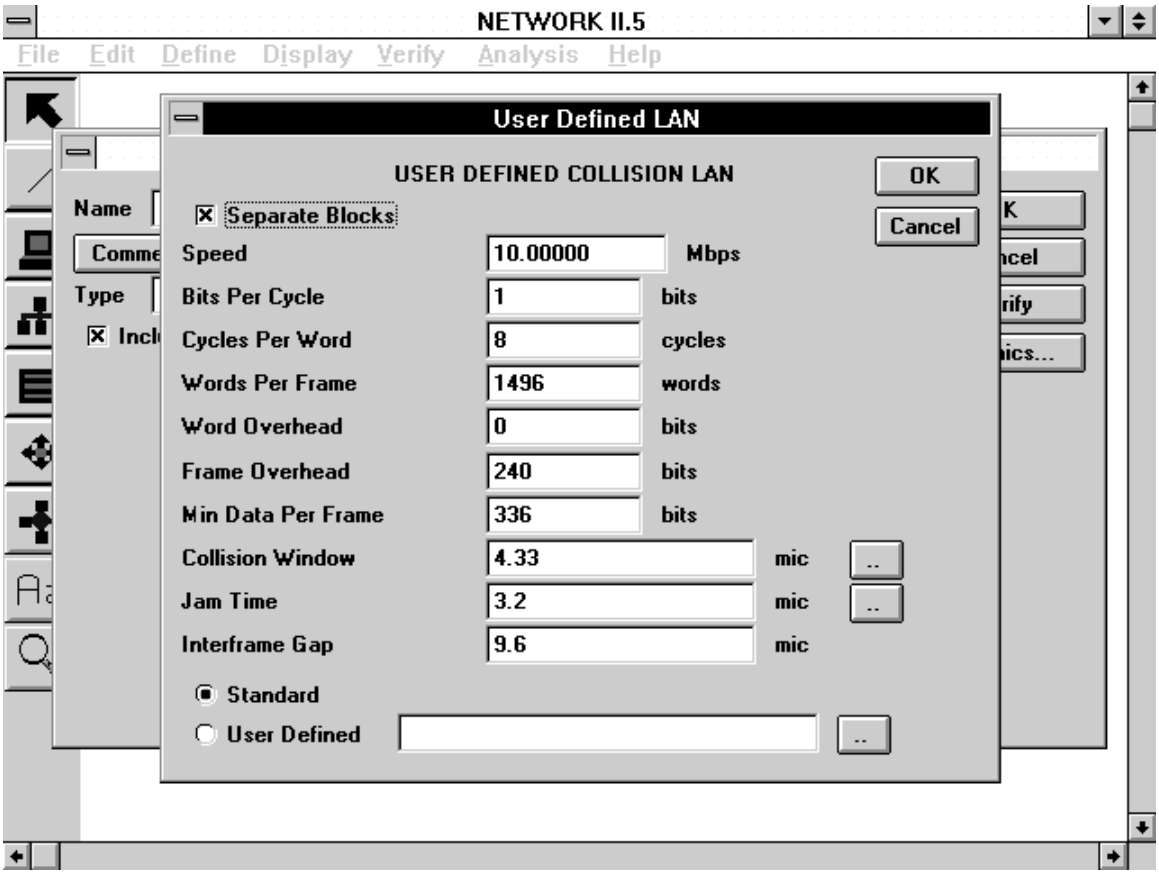
Type
Use the Type combo box to select a LAN Type. A list of available implementations is displayed by clicking on the down arrow of the Type combo box. In addition to the predefined implementations, user defined LANs are also available which will allow you to customize the LAN parameters.

Values

To view the LAN implementation parameters, click on the Values button. A list box with the implementation values will be presented. Notes regarding how the LAN implementation parameters were derived may also be displayed at the bottom of the list box. If you have chosen a user-defined LAN Type, you will be able to edit the parameters in the form.



Predefined LAN Implementation Values

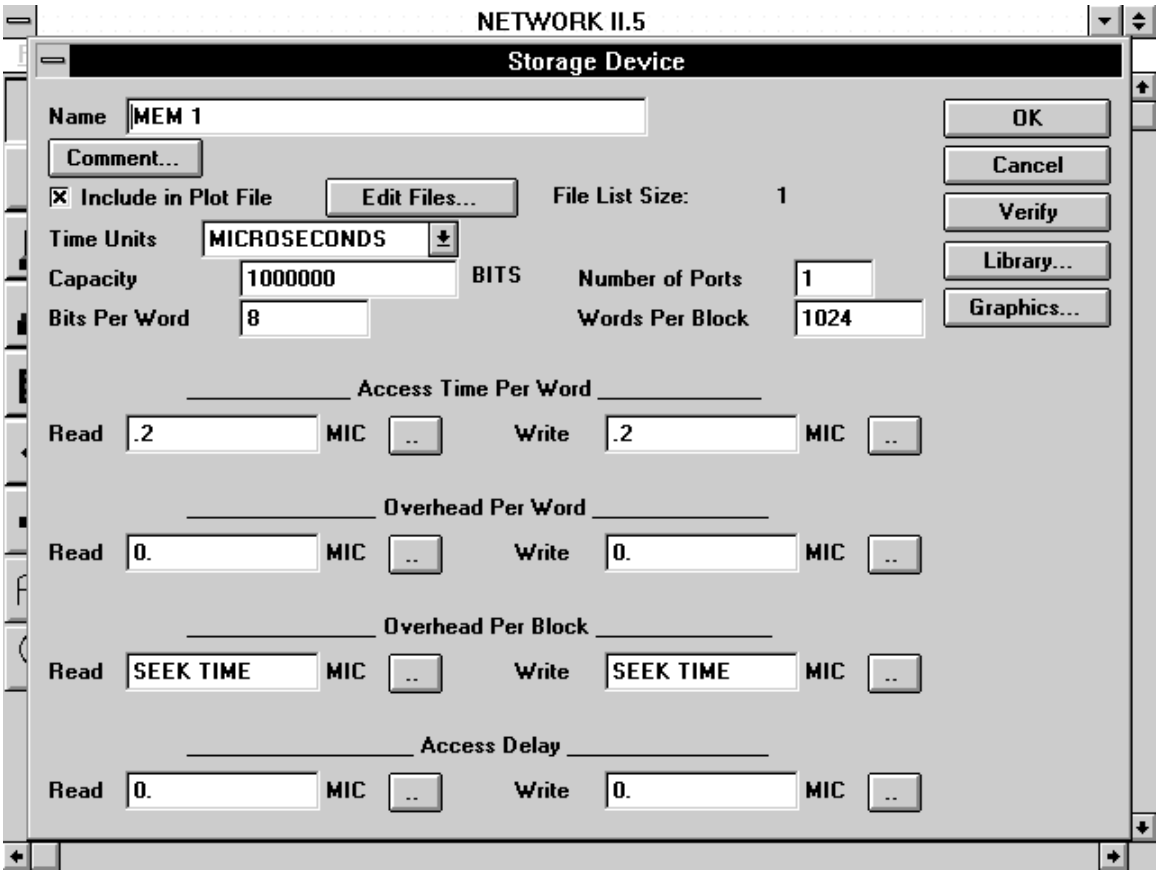


A User-Defined LAN Form

LAN Connection List

The LAN Connection List provides a listing of the PE's, Gateways, and SD's connected to the LAN. The order of the list determines the token passing order for the token LAN implementations. Connections may be added, deleted or moved in the list by using the Add, Cut and Move buttons next to the list box.

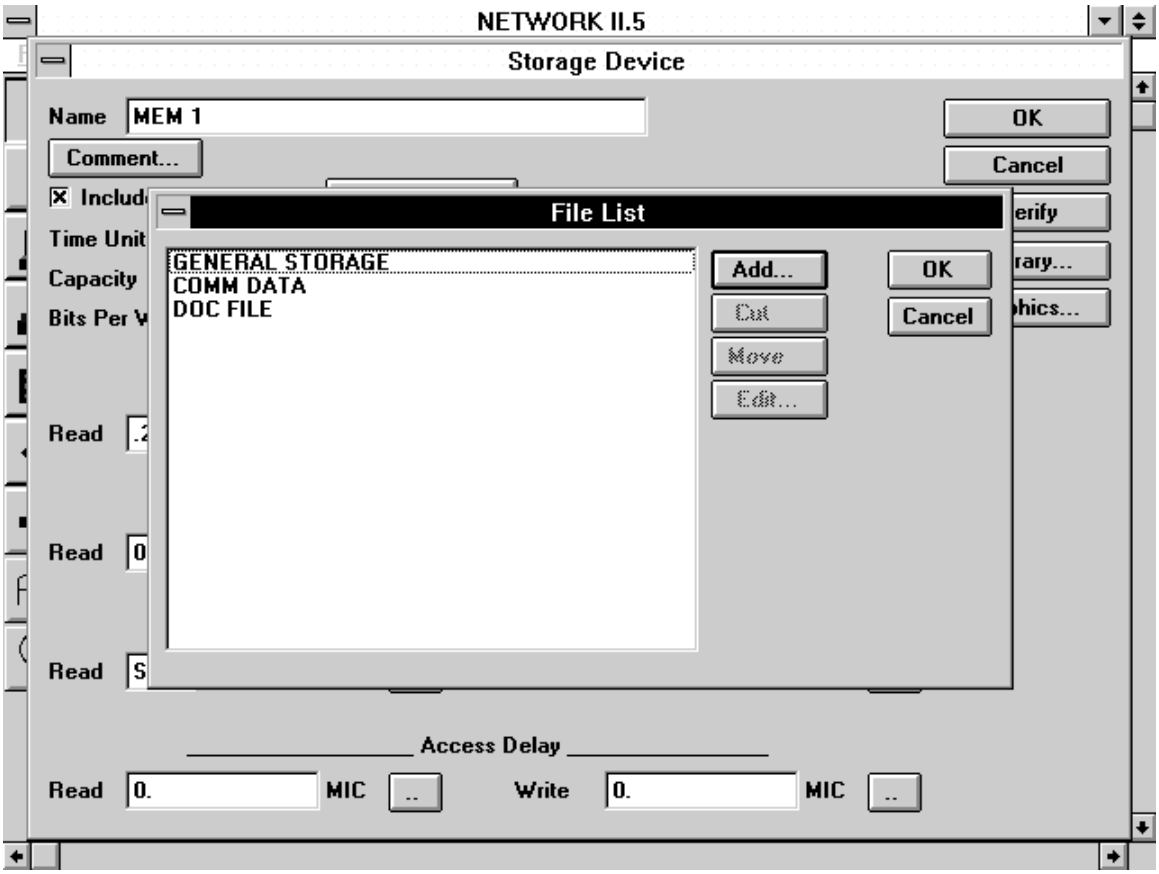
3.5.6 Storage Device Form



The Storage Device Form

Capacity
A Storage Device's capacity must be entered as a real value.

Number of Ports, Bits per Word, Words per Block
These parameters must be entered as integer values.



The File List Form

Access Time per Word, Overhead per Word, Overhead per Block, Access Delay
These parameters may be a real number or may be chosen from a SDF. To use a SDF, either enter the name of a SDF in the text box, select the double-dots button next to the text box and choose a name from the list of current SDFs.

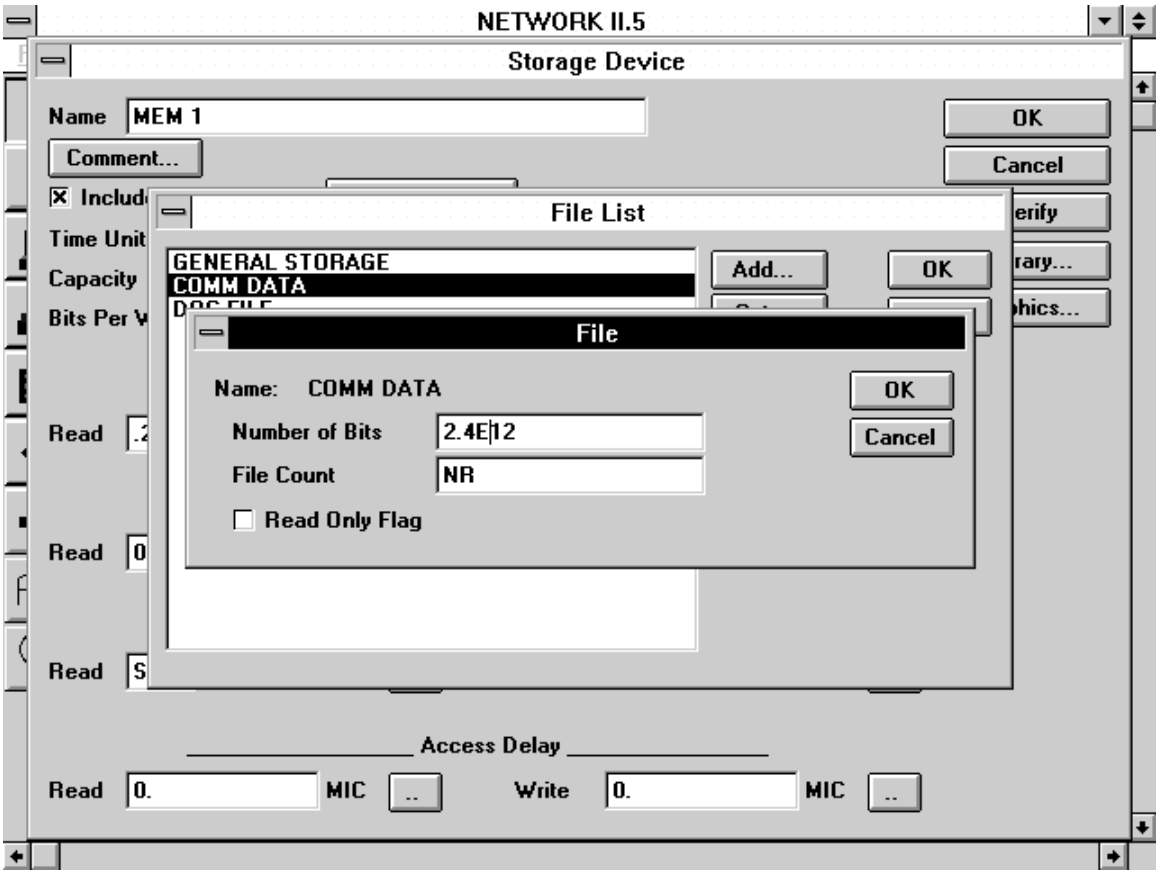
File List
To edit the Storage Device's File List, click on the Edit Files button. The File List Form will then be displayed with a list of all Files currently resident on the SD. Use the Add, Cut and Move buttons to add Files, cut them from the list or change their list positions. To edit a File's parameters, highlight its name in the list and select the Edit button to display the File Form. A count is displayed on the SD Form of the current number of Files in the File List.

3.5.7 File Form

Number of Bits
The number of bits must be specified as a real number.

File Count
An integer only (or NR) may be entered for the File Count.

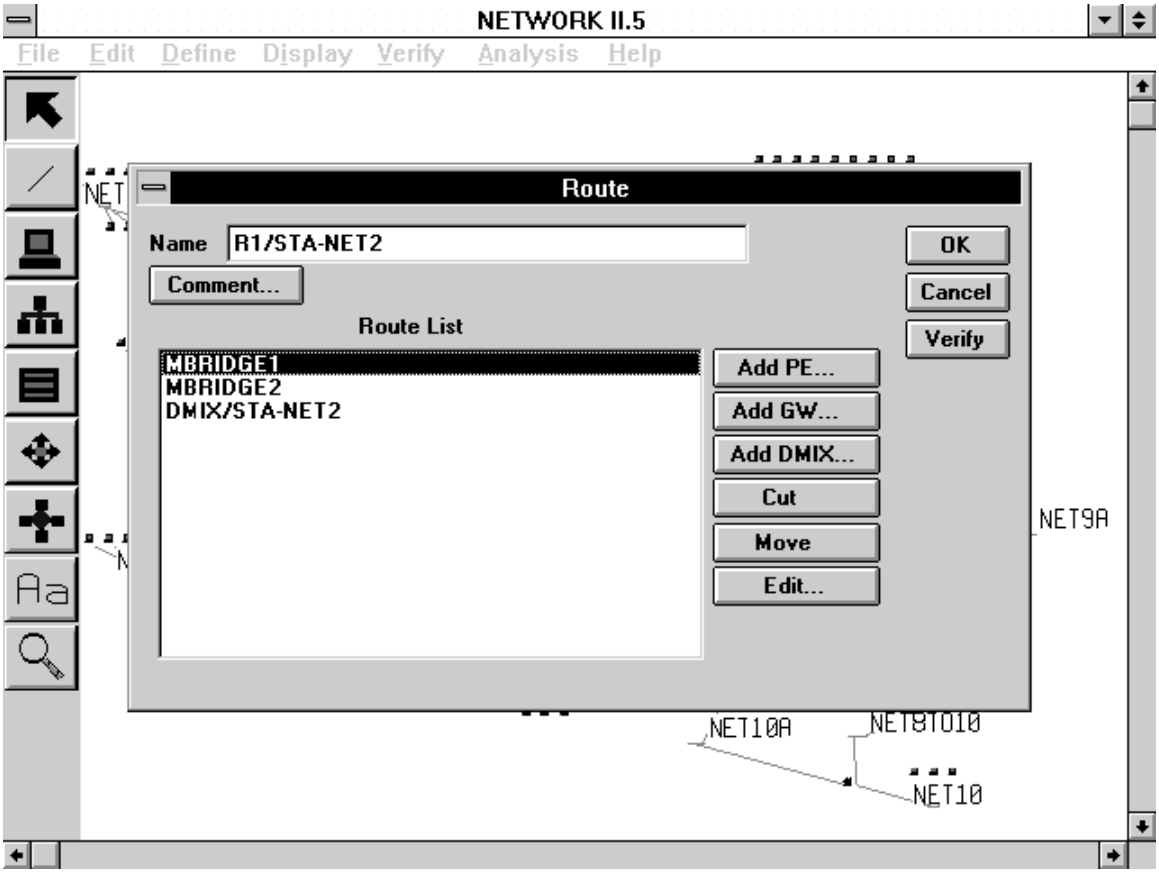
Read Only Flag
A file will be marked as read-only when this check box is selected.



The File Form

3.5.8 Route Form

Routes define the order in which a message will be passed through a network. The Route Form is used to construct a path for messages.



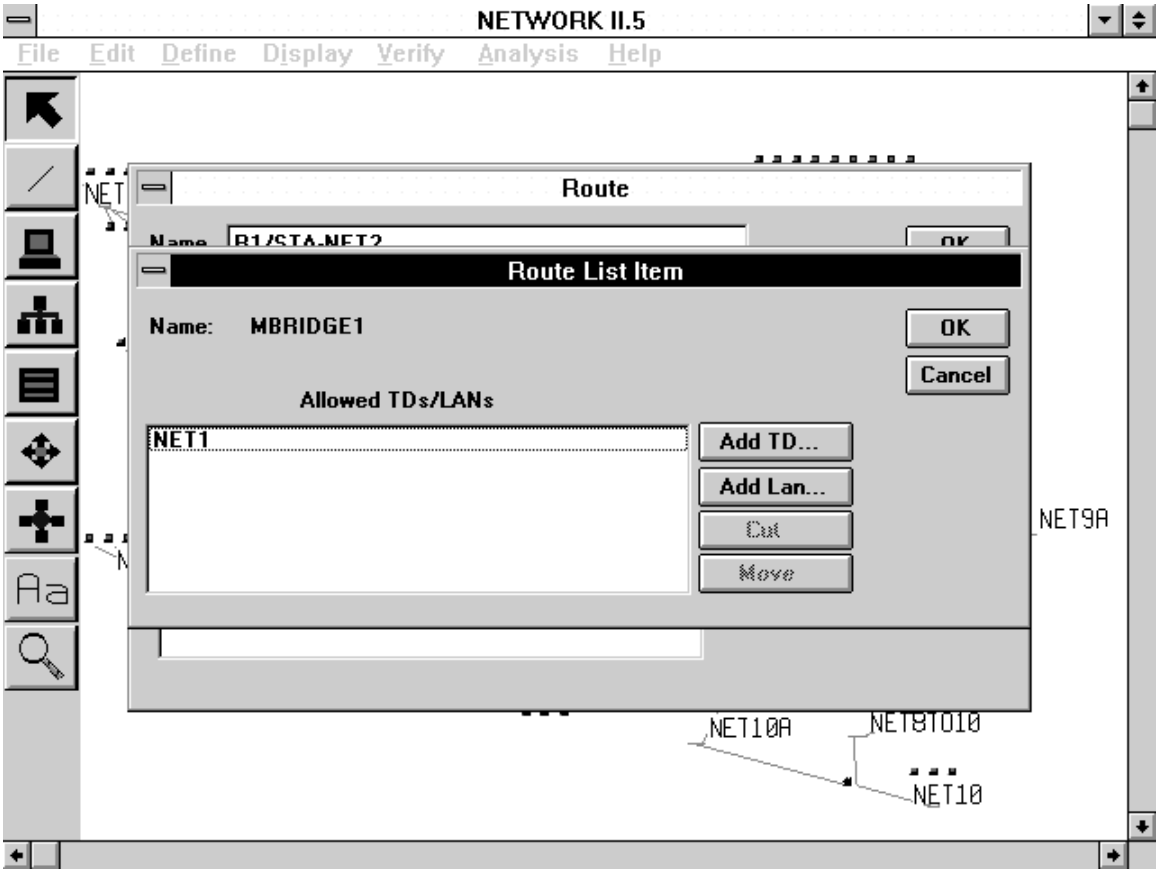
The Route Form

Route List

The Route List describes the order of hardware elements along which a message using this route will be passed. To add items to the list, use the Add PE, Add Gateway, and Add DMIX (Destination Mix) buttons. The final item in the list must be a PE or Destination Mix.

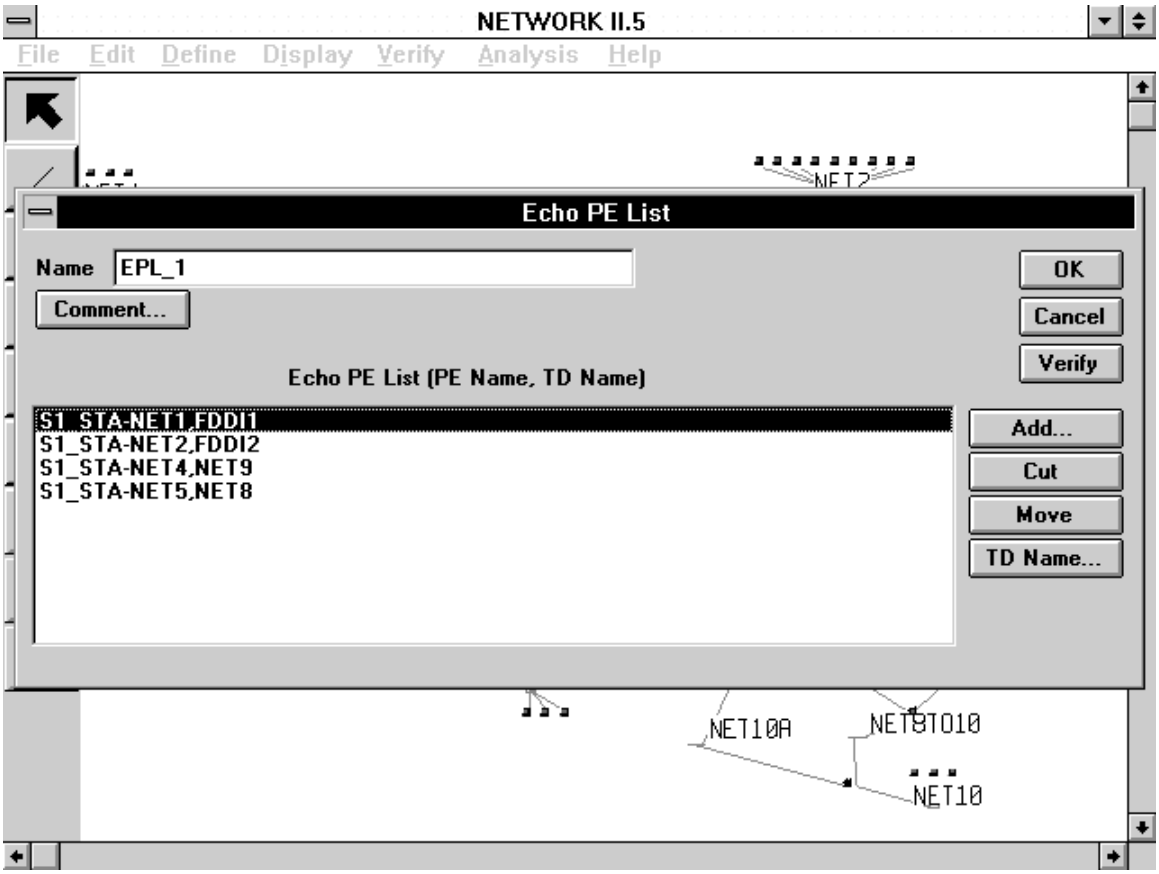
You may cut, move, or edit individual items in the Route List by using the Cut, Move and Edit buttons. Each intermediate hop in the route may have an Allowed TD/LAN list. If a processing element or Gateway is connected to multiple TDs and/or LANs, the Allowed TD/LAN list determines which Transfer Device or LAN may be used to pass a message. If the Allowed TD/LAN list is empty, any valid TD or LAN (connected to origin and

destination) may be used at runtime. The Edit button will allow you to set up the Allowed TD/LAN list for an entry in the Route List.



The Allowed TD/LAN Form

3.5.9 Echo PE List Form



The Echo PE List Form

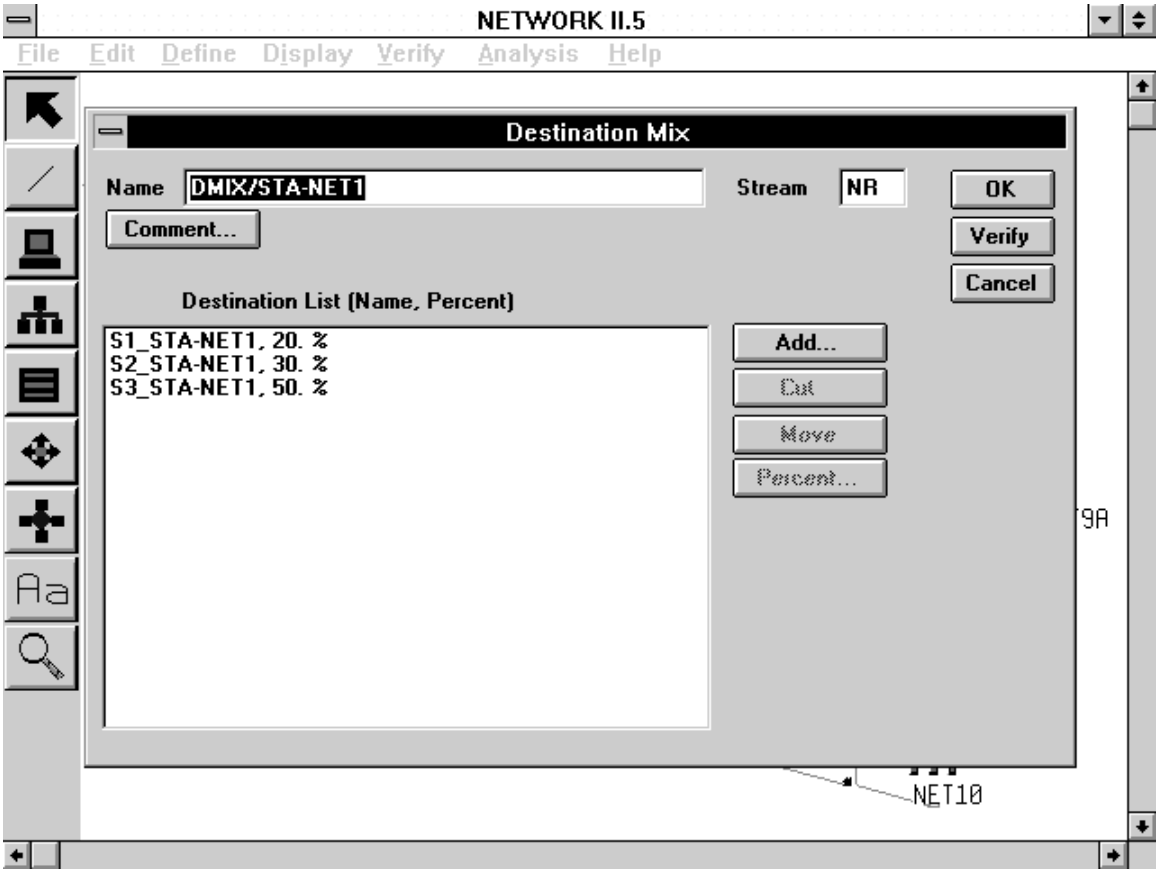
Echo PE List

The Echo PE List Form consists of a single list box with control buttons to its right. An Echo PE List is essentially a list of linked PE and TD/LAN names. The Add button will let you enter a new PE name in the Echo PE List. The TD Name button will permit you to attach a TD/LAN name to a PE name in the list. Use the Cut and Move buttons to remove list entries or change their positions.

3.5.10 Destination Mix Form

Stream

The stream is an integer value that must be in the range from 1 to 999. If you leave the stream in the NR state, it will be selected by the simulation program. The stream is used to control the random sequence in which destinations are chosen.



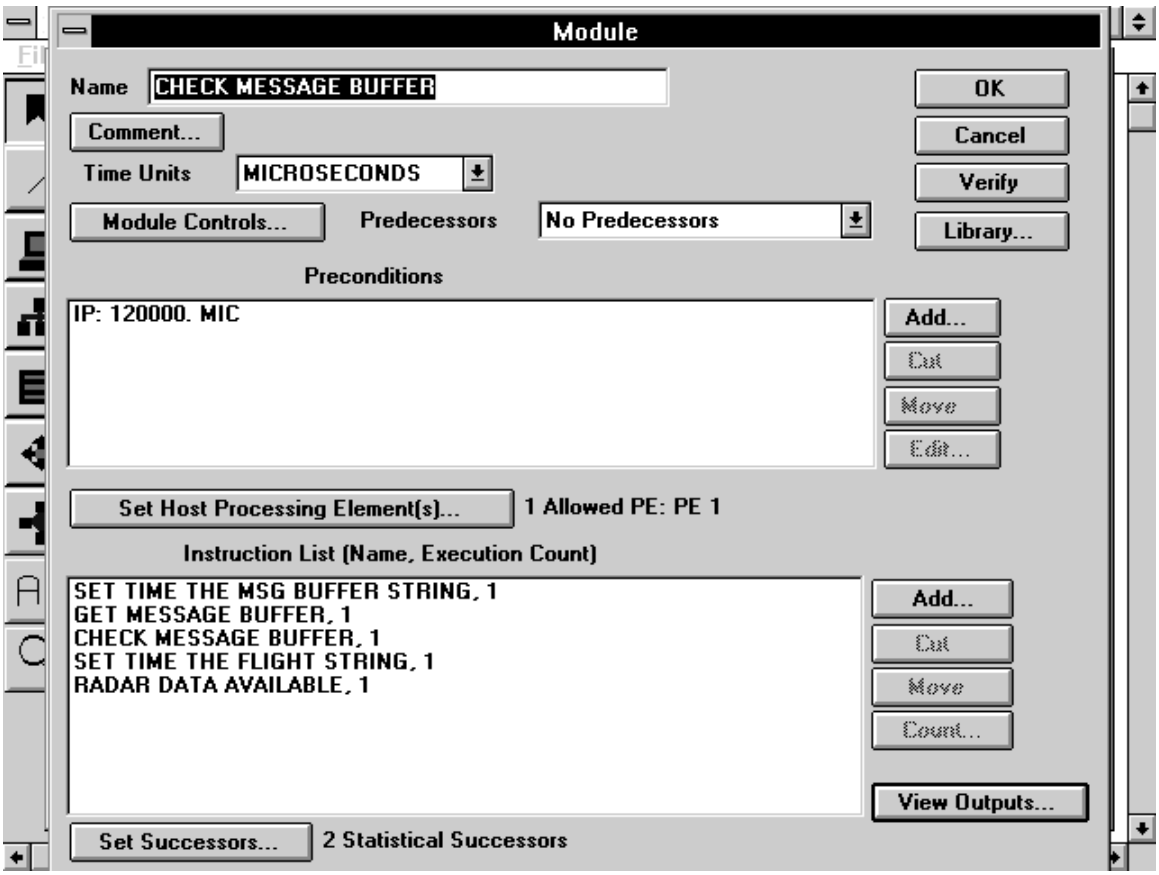
The Destination Mix Form

Destination List

The Add button is used to add destinations to a Destination Mix. Valid destinations include Processing Elements, other Destination Mixes, Routes , Global Message List, Next, and Self. The Cut and Move buttons may be used to cut or move entries in the list. The destination's percentage of selection can be set by highlighting the destination name in the list and selecting the Percent button. The Destination List list box will display a destination name followed by its percentage of selection. The sum of the percentages in the list must be 100.

3.5.11 Module Form

Owing to the large number of attributes a Module possesses, the Module Form is one of the more complex NETWORK forms. All of the attributes of a Module are accessed from within this form or from subforms emanating from the Module Form.



The Module Form

Instruction List

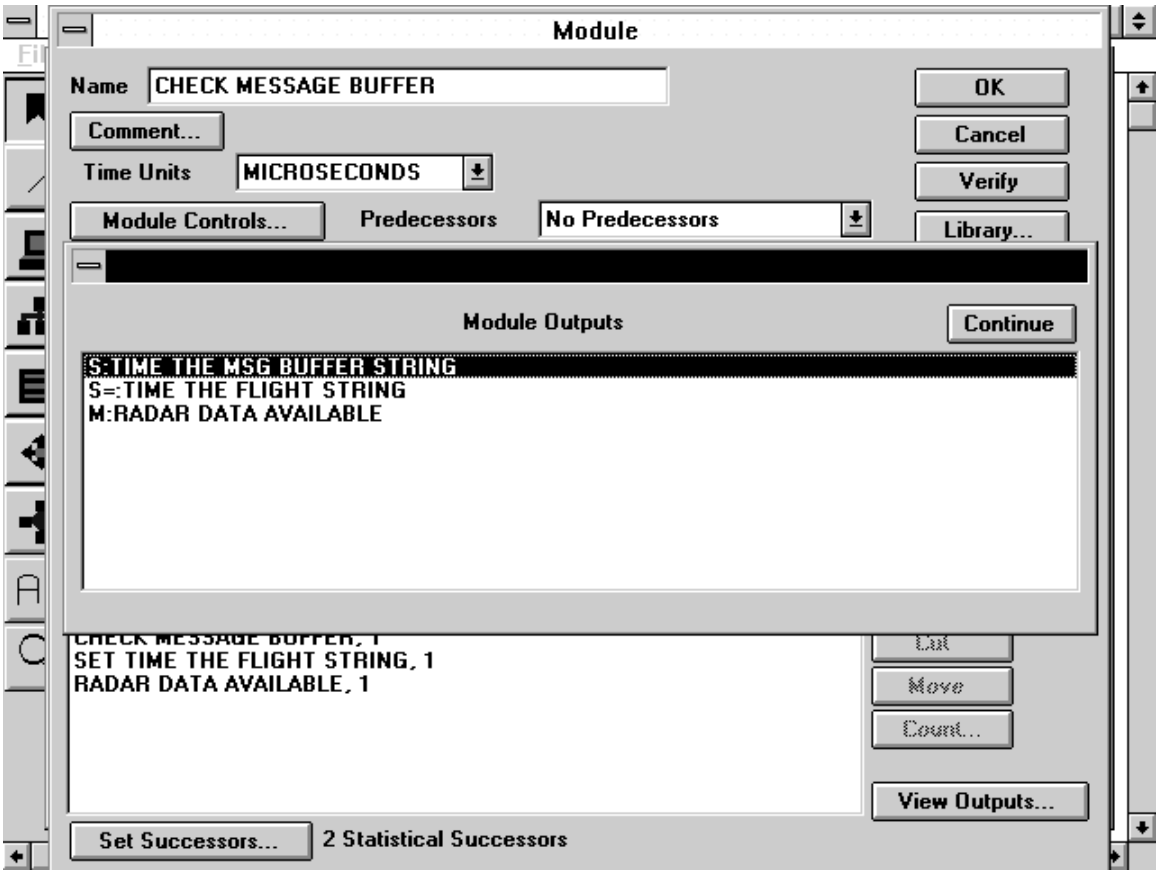
The Instruction List of a Module is displayed in the large list box on the bottom of the Module Form. This list holds the names of Module instructions and the number of times each is to be executed. If more than one instruction is listed, the instructions will be executed in the order they appear in the list. The Add, Cut, and Move buttons are used to create, remove and reposition instructions in this list. The Count button is used to change the number of executions attribute of a Module instruction. To change the execution count of a Module instruction, select its name in the list and then click on the Count button. A form will appear which will allow you to set the instruction count to an integer or to the name of a Statistical Distribution Function.

Predecessors

The Predecessors combo box is used to select a predecessor type for this Module. Click on the down arrow to select either No Predecessors, Anded Predecessors, or Ored Predecessors. If either of the latter two choices are made, the Allowed/Resident Processors list will be emptied.

View Outputs

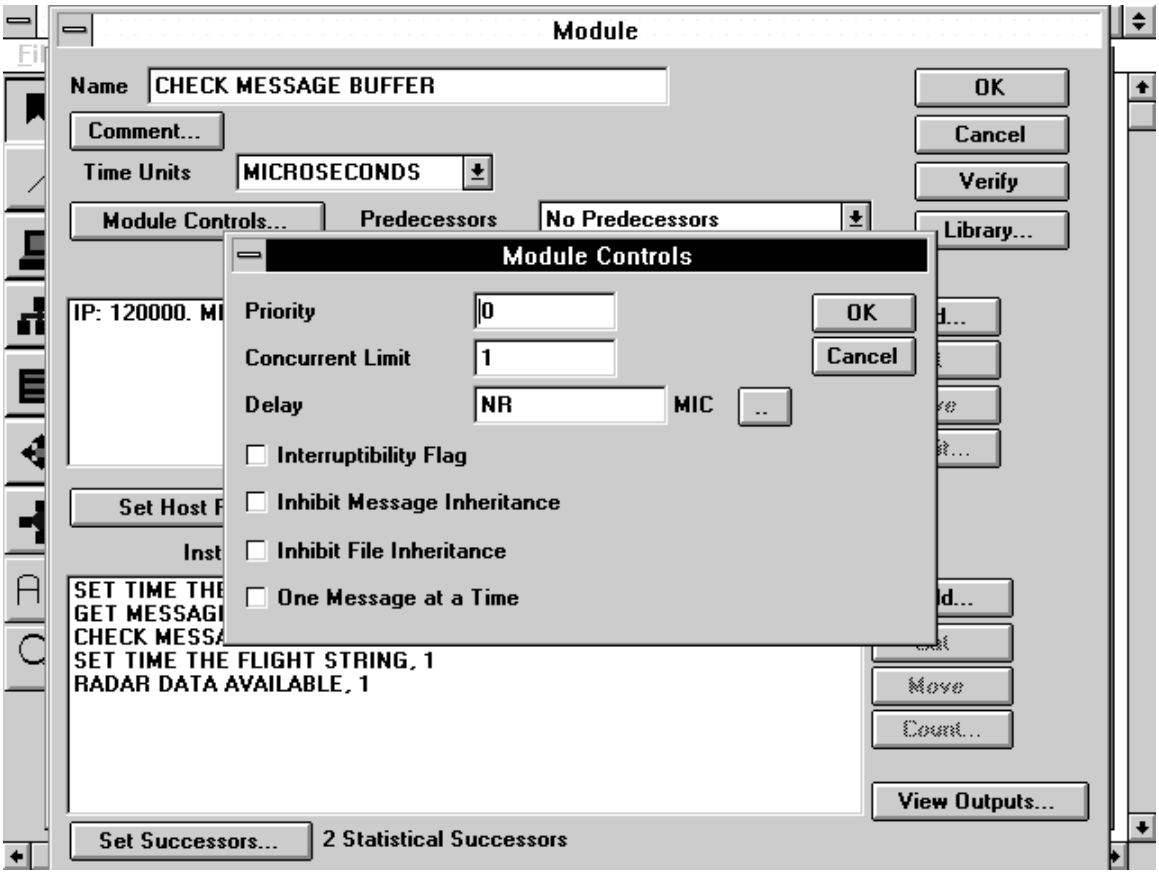
Clicking on the View Outputs button at the bottom of the Module Form will display the names of any Files that will be written, Messages sent, or Semaphores modified as a result of the Module's execution. An example is shown below. In the example, a semaphore named "TIME THE FLIGHT STRING" is SET and its count is incremented. This is indicated by the text "S+:" which appears before the Semaphore name.



The Module Outputs Form

Module Controls

Select the Module Controls button to access the Module Controls Form. The Module Controls form contains seven control parameters for a Module. Priority and Concurrent Limit can be set to an integer value. The Delay attribute may be set to a real value or given the name of a SDF. Use the Delay double-dots button to select the name of a current SDF. The remaining four attributes; Interruptibility Flag, Inhibit Message Inheritance, Inhibit File Inheritance and One Message at a Time, are check boxes which can be set to YES or NO.



The Module Controls Form

Preconditions

The Preconditions list box is used to display, create and edit a Module's preconditions. The Add, Cut, Move and Edit buttons to the right of the list box will allow you to work with the Preconditions list. Use the Add button to enter a new precondition for a Module. You will then be asked to select a precondition type.

Preconditions that you may specify include:

- Start Time
- Stop Time
- Iteration Period
- Message Status Requirement
- Semaphore Status Requirement
- Hardware Status Requirement
- File Status Requirement

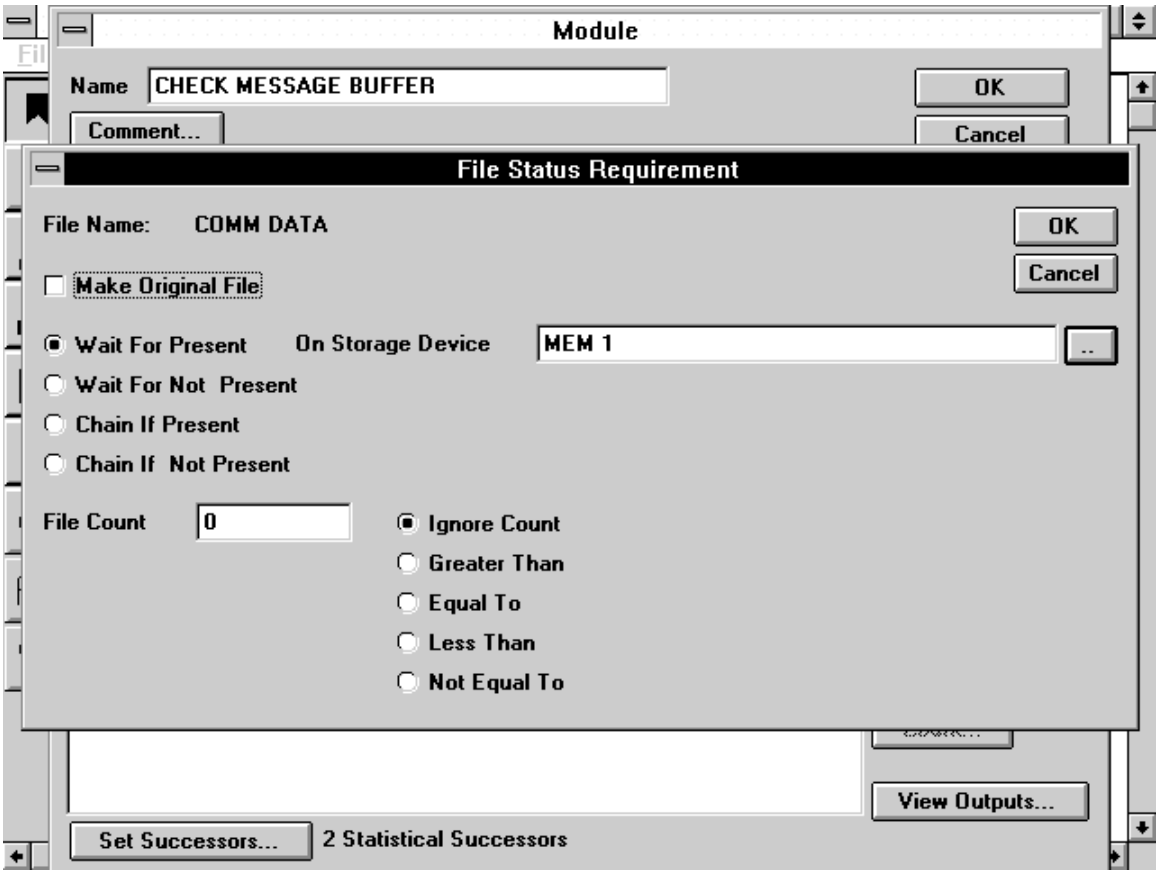
Once a Start Time, Stop Time or Iteration Period is added to the precondition list, it will no longer appear in the list of precondition types when you add a new precondition, because a Module has only one Start Time, Stop Time or Iteration Period, but may have

several of the other precondition types. Preconditions may be removed from the list (Cut button) or repositioned in the list (Move button).

The Preconditions list is actually several lists combined into a single list. Preconditions selected for a Module will be displayed in the Preconditions list in the following order: Start Time, Stop Time, Iteration Period, Message Status Requirements, Semaphore Status Requirements, Hardware Status Requirements and File Status Requirements. The first three preconditions cannot be moved in the list, but will always appear at the top. The remaining preconditions (Message, Semaphore, Hardware and File Status Requirements) can only be moved if two or more of a particular type exist. The movement will be confined to that portion of the Preconditions list that held the selected type of precondition to be moved. For example, if you wish to move a File Status Requirement in the Preconditions list, it may only be moved to the top, bottom or another part of the sublist that contains only File Status Requirements.

To change a current precondition, highlight it in the list box and then select the Edit button. For Start Time, Stop Time and Iteration Period, you will be asked merely to enter a new value or SDF name. The other preconditions will have their attributes displayed in a separate form in which you may make changes. The Hardware Status Requirement Form is simply a group of four radio buttons to determine the required status of the selected hardware data structure (busy, idle, collision and overflow). The remaining Module precondition forms are described below.

3.5.11.1 File Status Requirement Form



The File Status Requirement Form

Make Original File

The Make Original File flag can be set to YES/NO by marking and unmarking the check box. It can be used to reset the "original file" of a Module.

File Status Requirement Type

A group of radio buttons is used to determine the if the File Status Requirement will be a Wait For or Chain If type and if the File is to be present or not present on a specified Storage Device. The choices are Wait For Present, Wait For Not Present, Chain If Present and Chain If Not Present.

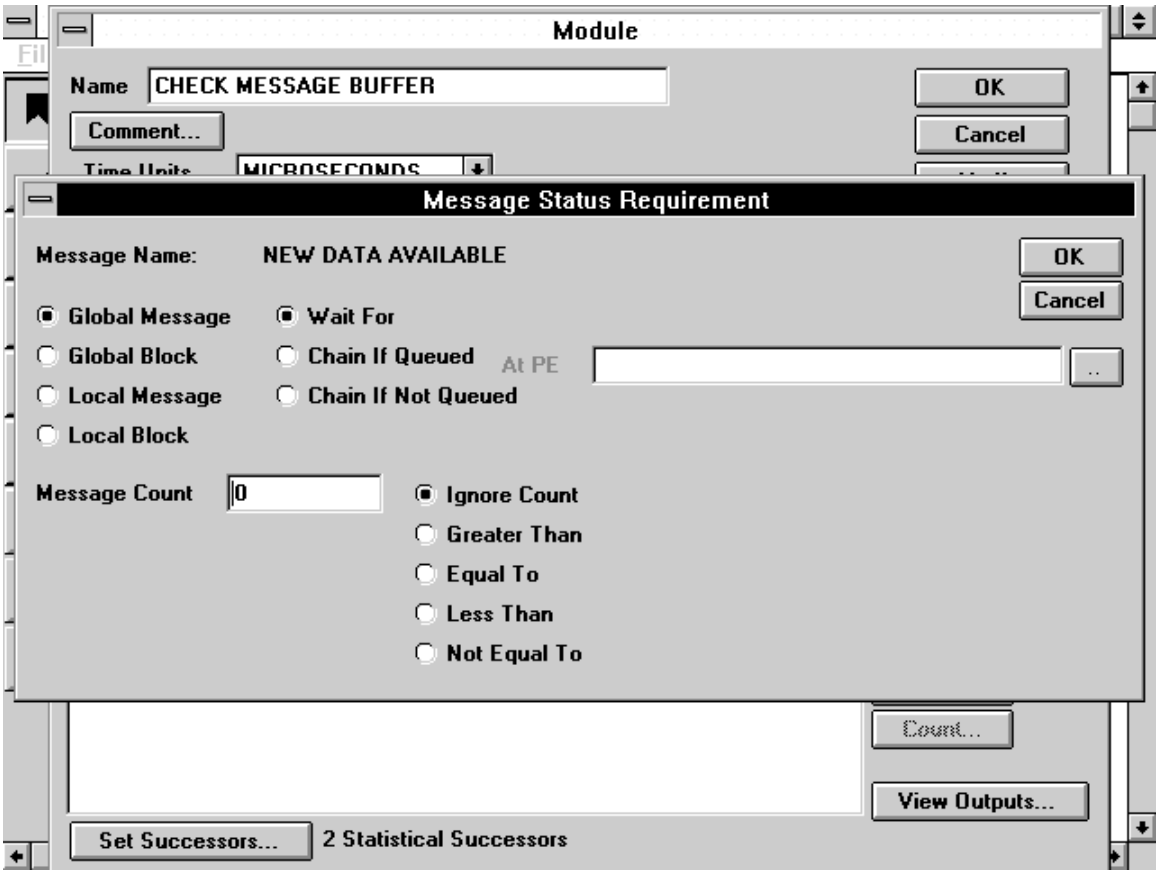
On Storage Device

Enter the name of a Storage Device directly in this text box, or use the double-dots button to select from the list of current SD names. This is used in conjunction with the File Status Requirement Type to determine when the Module may execute.

File Count

A File Status Requirement may additionally monitor the count of a File. Enter an integer value in the File Count text box and then select one of the File Count conditions from the group of radio buttons next to the File Count text box.

3.5.11.2 Message Status Requirement Form



The Message Status Requirement Form

Message Type

Select one of these radio buttons to determine if the message that meets this Message Status Requirement may be a global message or must be a local message and if a message block is satisfactory or the complete message is required.

Message Status Requirement Type

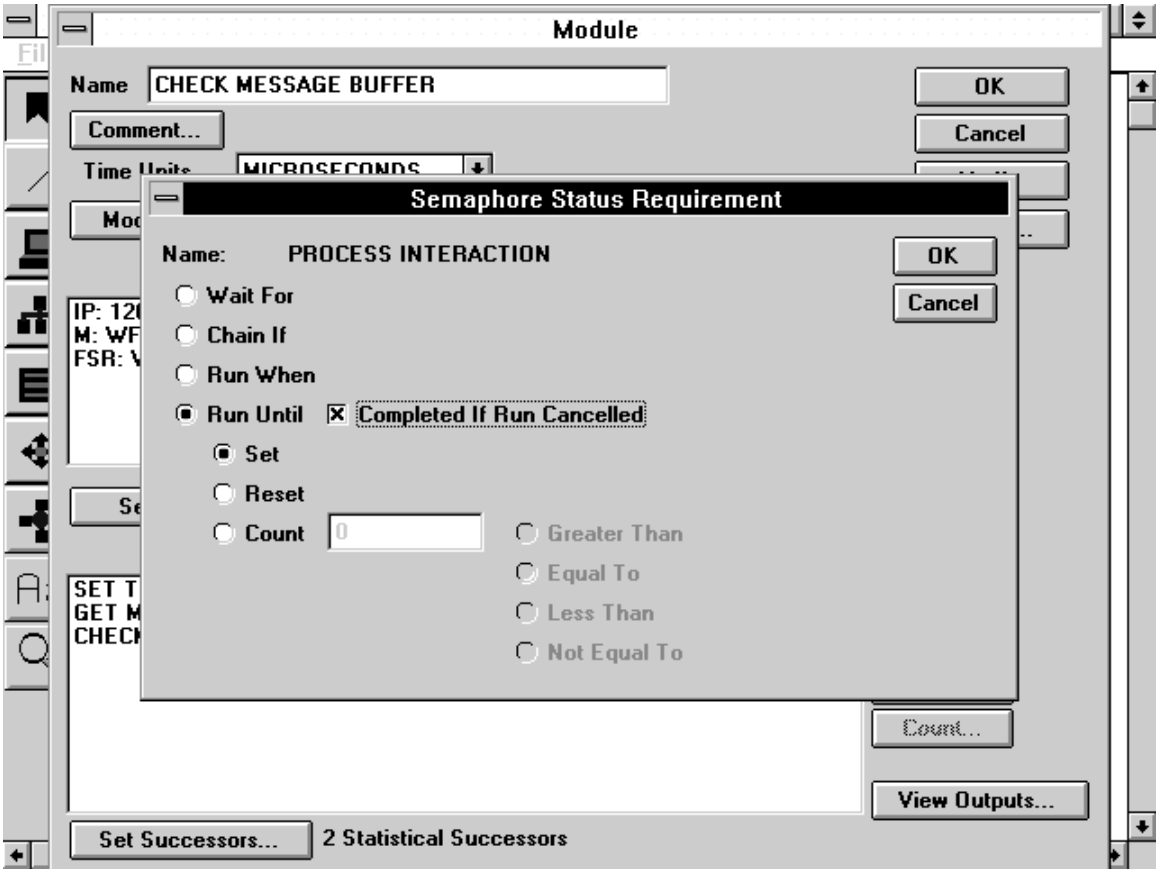
A Message Status Requirement may be a Wait For type (the message must be present at the host PE) or one of two Chain If types (a Module will not be chained to unless the message is present or not present at a specified PE). If the type selected is Chain If Queued or Chain If Not Queued, the text box to enter a PE name and its double-dots

button will be selectable. Enter a PE name directly or select the double-dots button to choose from the list of current PE names.

Message Count

A Message Status Requirement may additionally check the count of messages. Enter an integer value in the Message Count text box and then select one of the Message Count conditions from the group of radio buttons next to the Message Count text box.

3.5.11.3 Semaphore Status Requirement Form



The Semaphore Status Requirement Form

Semaphore Status Requirement Type

This group of radio buttons is used to select a type for the Semaphore Status Requirement. The four choices are Wait For, Chain If, Run When and Run Until.

Completed If Run Cancelled

If Run Until has been selected as the type of the Semaphore Status Requirement, this check box will be selectable and can be set to YES/NO. If set to YES, a Module cancelled by the Run Until condition is considered to have completed its execution and this will be included in the completed Module statistics, otherwise the Module cancellation statistics will be updated.

Required Status or Count

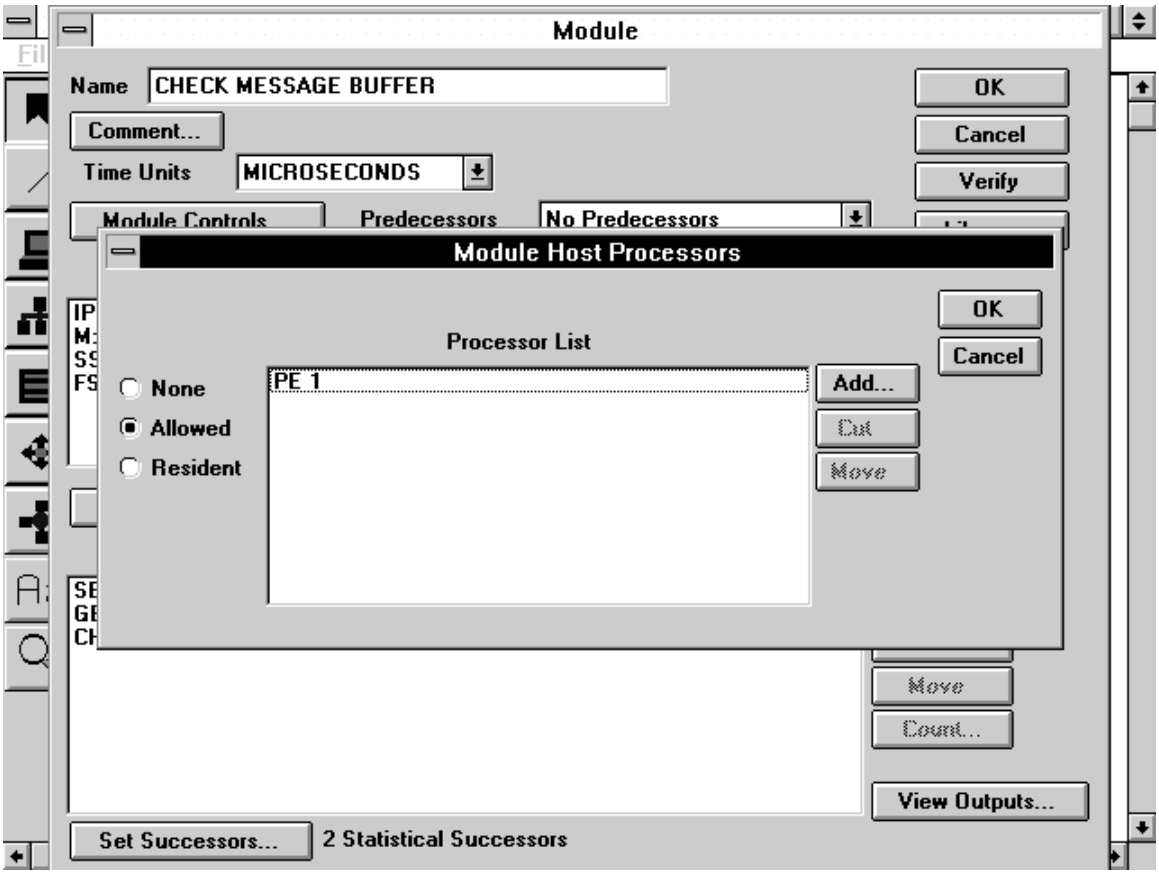
The group of radio buttons appearing below the Semaphore Status Requirement Type radio buttons is used to determine if the status of the selected Semaphore is to be monitored (select either Set or Reset) or if the Semaphore's count is to be monitored (select the Count radio button). If Count is selected as the Semaphore attribute to be monitored, the Semaphore Count fields will be selectable.

Semaphore Count

Enter an integer value in the Semaphore Count text box and then select one of the count checks from the group of radio buttons to the right of the Count box. You may set up a Semaphore Status Requirement to be true if the Semaphore's count is greater than, less than, equal to, or not equal to the count entered in the Semaphore Count box.

3.5.11.4 Set Host Processing Elements

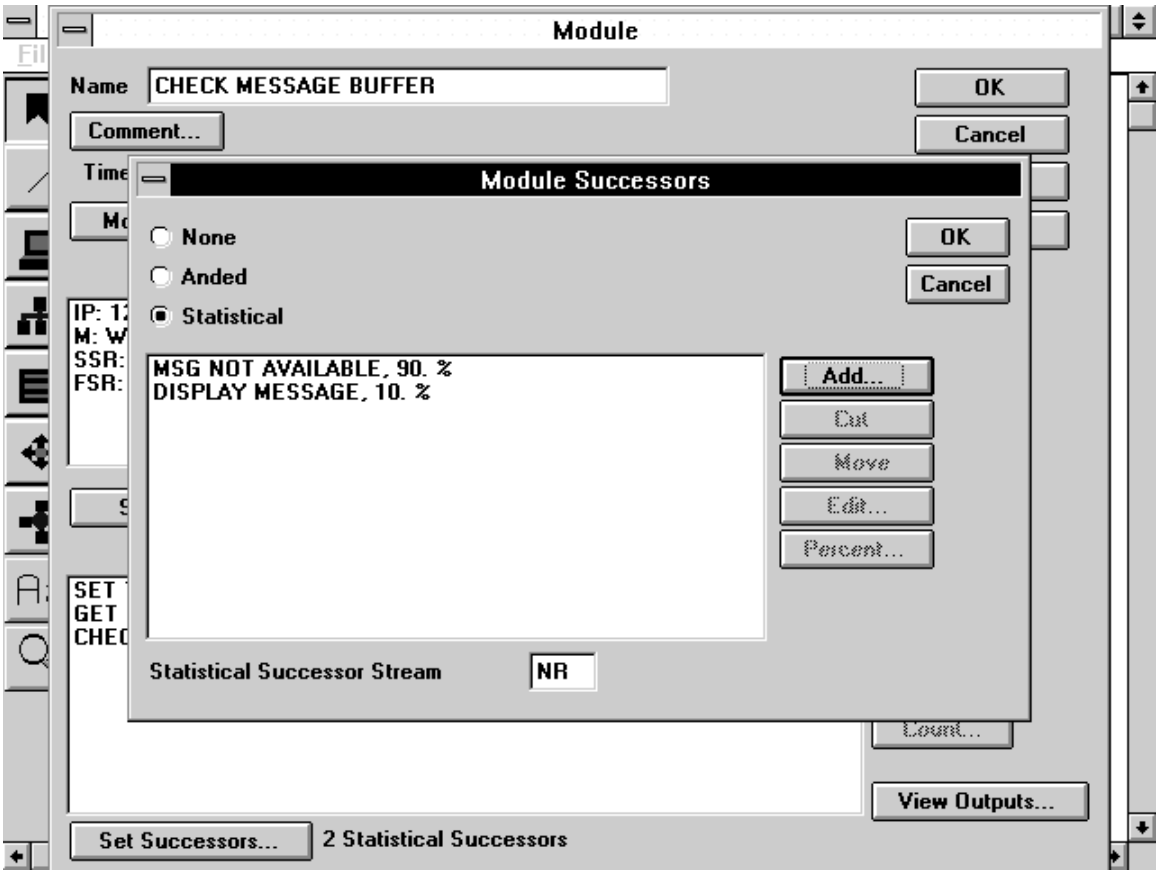
The Set Host Processing Elements button will display the Module Host Processors Form to select a list of host Pes for a Module and to select their type (Allowed or Resident). If any Allowed/Resident are added to a Module, the number and type of host Pes will be displayed on the Module form next to the Set Host Processing Elements Button.



The Module Host Processors Form

3.5.11.5 Set Successors

The Set Successors button will display the Module Successors Form which controls the successor list of a Module. This form has a set of radio buttons to determine the successor type, None, Anded or Statistical. If None is selected, the Module Successors list will be emptied. If Anded is selected, all listed successor Modules will be scheduled to run at the completion of this Module. If Statistical is selected, only one successor Module will be selected and the selection is determined by the percentage attribute of the successor Modules. Use the Add, Cut and Move buttons to create, remove and reposition successors in the list. The percentages of Statistical Successors are displayed in the list box, just after their names. The Percent button can be used to change the percentage of Statistical Successor names highlighted in the list box. These percentages must sum to 100. The Edit button is selectable only for Anded Successors and is used to associate a Skip Count or a Chain Count with an Anded Successor.

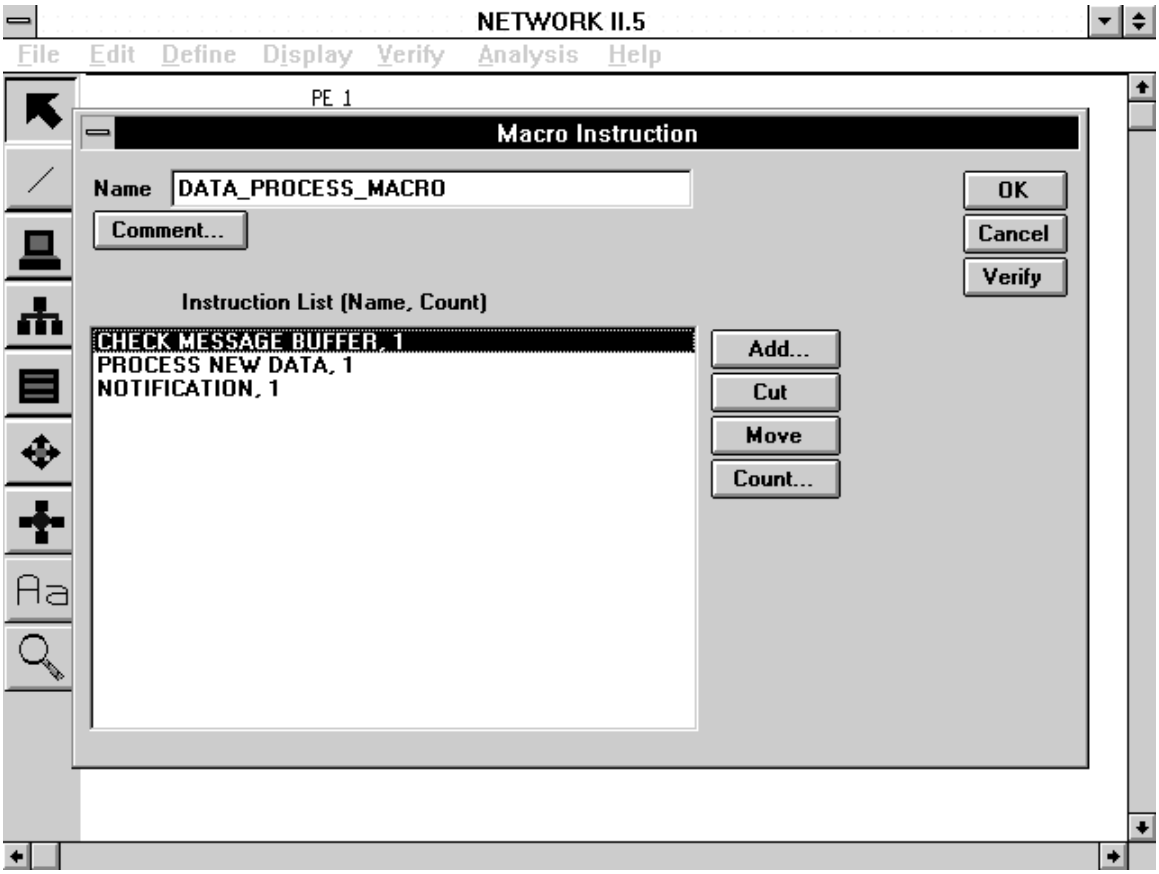


The Module Successors Form

3.5.12 Macro Instruction Form

Instruction List

Click on the Add button to add names of individual instructions, Macro Instructions and Instruction Mixes to the Instruction List of a Macro Instruction. To change the count of an individual item in the list, highlight its name and then select the Count button.



The Macro Instruction Form

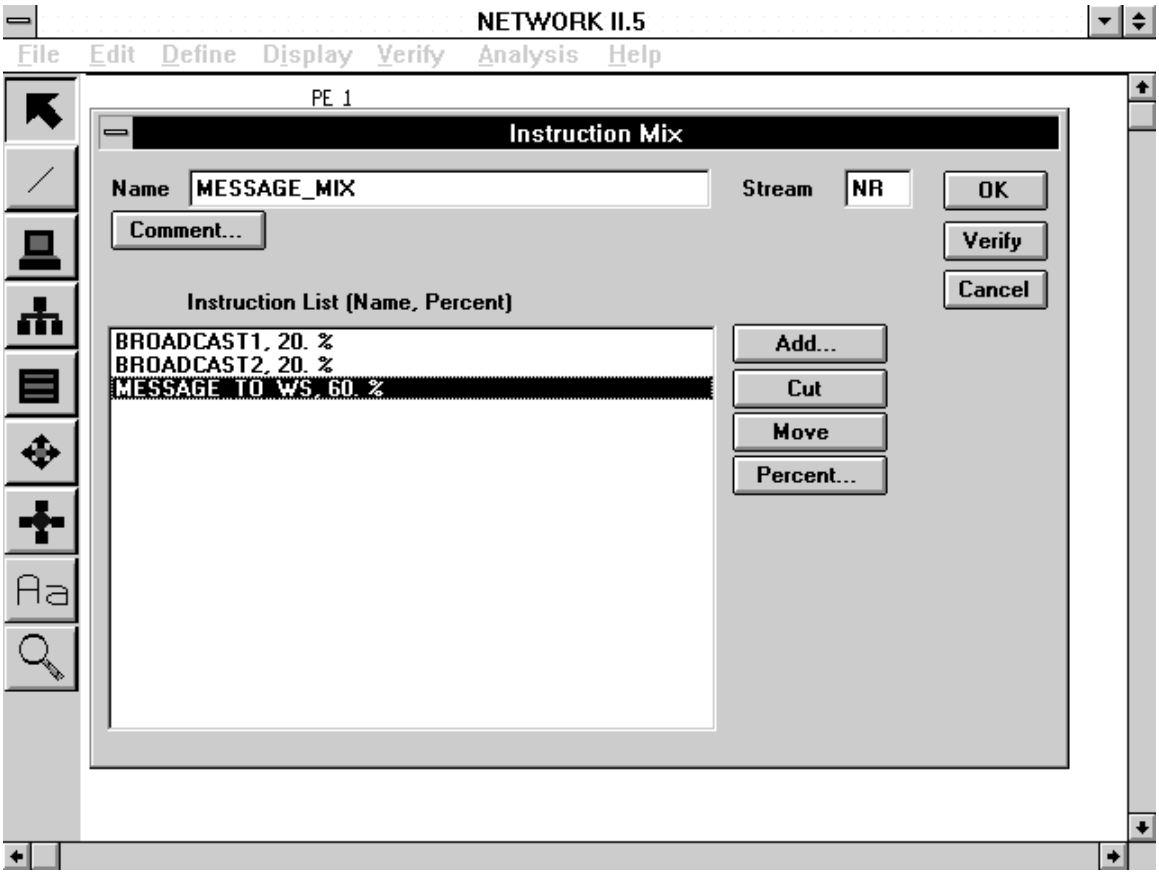
3.5.13 Instruction Mix Form

Stream

The stream is an integer value that must be in the range from 1 to 999. If you leave the stream in the NR state, a value will be chosen by the simulation program. The stream is used to control the random selection from the mix list.

Instruction List

To add individual instructions, Instruction Mixes or Macro Instructions to an Instruction Mix, click on the Add button. The percentage of selection of a list item can be changed by highlighting its name and selecting the Percent button. The sum of the percentages in the list must be to 100.



The Instruction Mix Form

3.5.14 Statistical Distribution Function Form

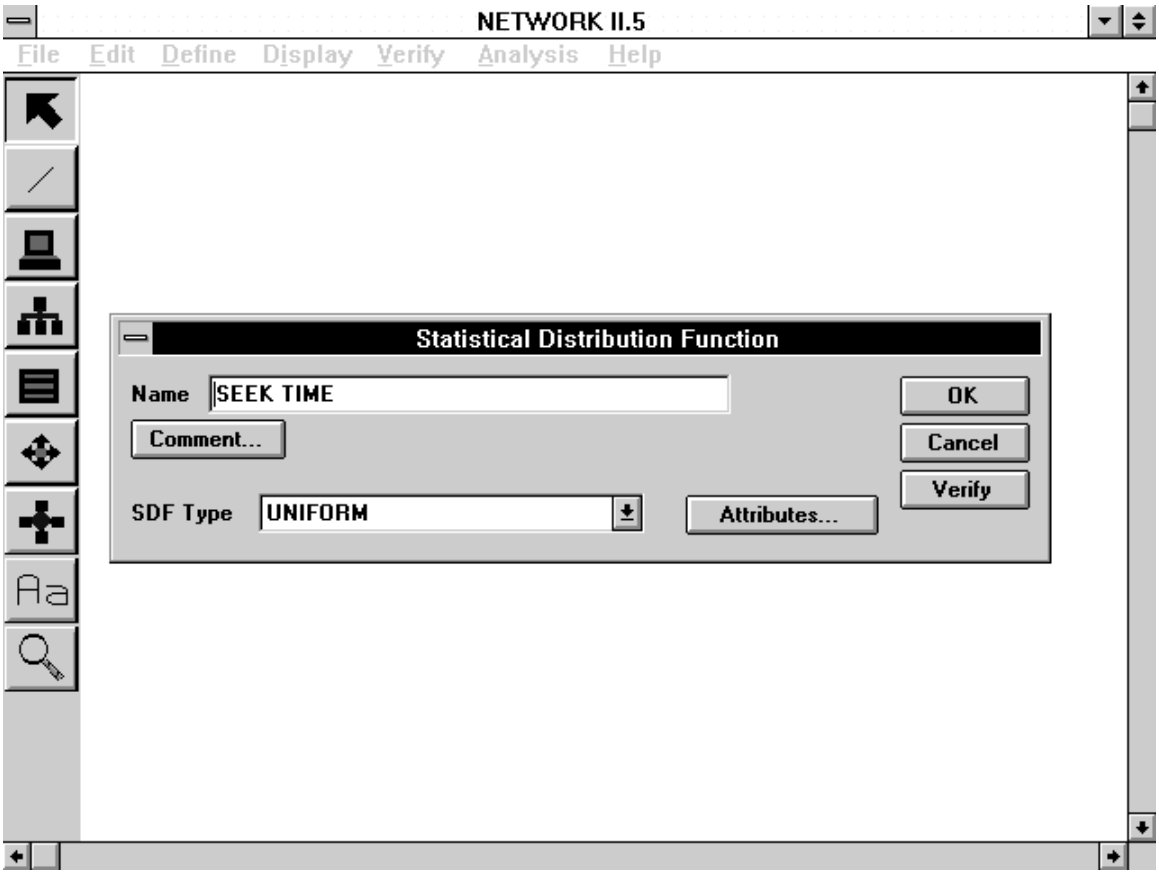
The SDF Form is used to display and manipulate the attributes of Statistical Distribution Functions.

Type

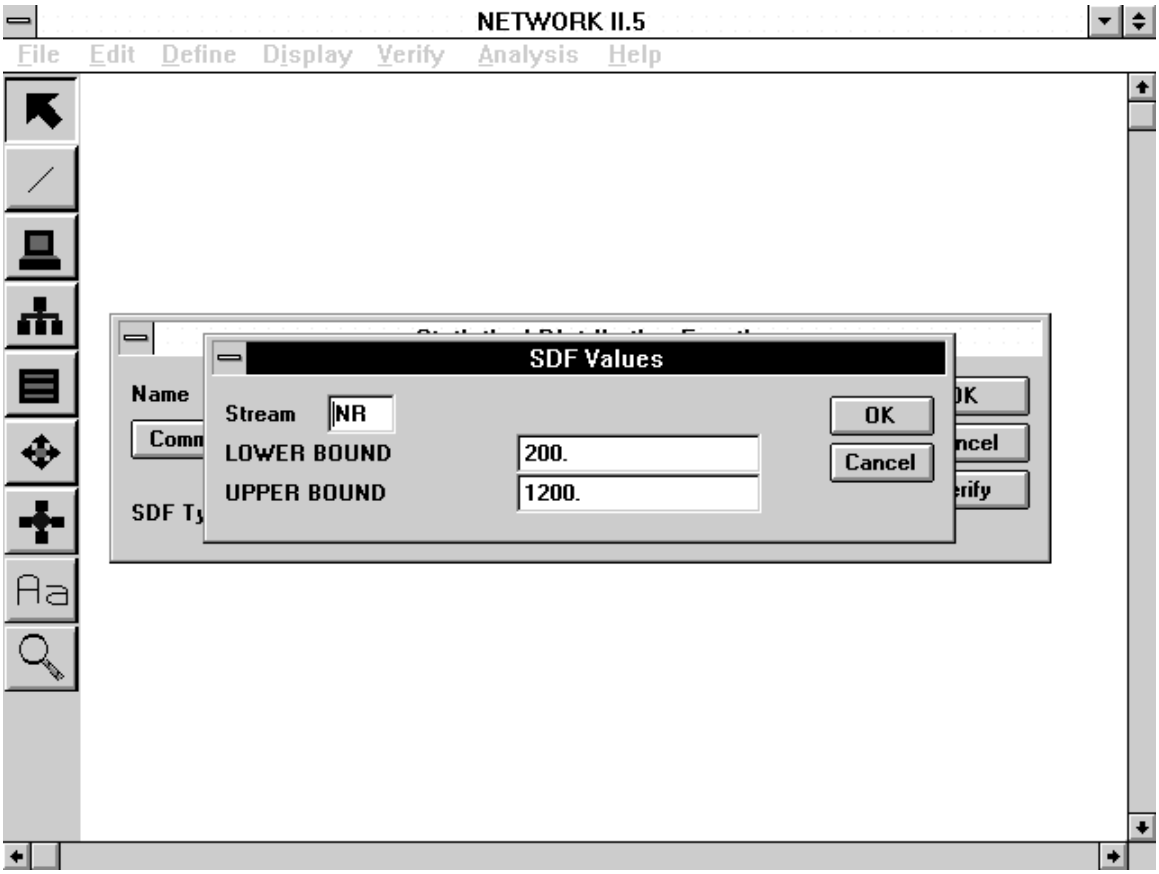
To select a SDF type, click on the down arrow of the Type combo box and then select a type from the list.

Attributes

The attributes of a SDF are determined by the SDF type that has been selected. After a choice of SDF type is made, clicking on the Attributes button will display the SDF attributes form. SDF parameters are displayed and edited within this form.


NETWORK II.5
File Edit Define Display Verify Analysis Help
Statistical Distribution Function
Name
SEEK TIME
Comment...
SDF Type
UNIFORM
Attributes...
OK
Cancel
Verify

The SDF Form



Example SDF Attributes Form

The following types of SDF parameters may be found on the Set Attributes forms of various SDF types.

Stream

The stream is an integer value that must be in the range from 1 to 999. If you leave the stream in the NR state, a value will be chosen by the simulation program. The stream is used to control the random sequence in which numbers are chosen. The stream value is not displayed for all distribution types.

Non-Table Distribution Parameters

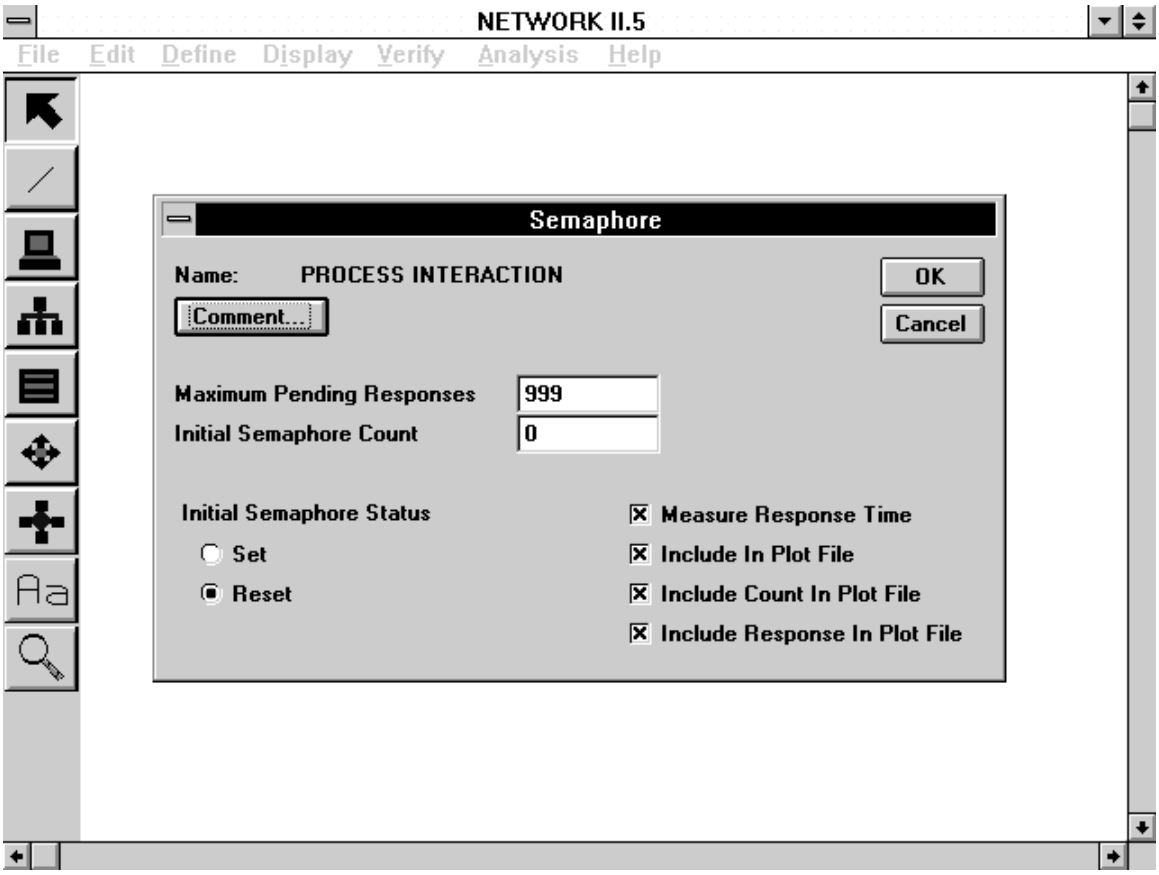
In general, the parameters for non-table distributions may be edited directly on the SDF Form. In most cases, the parameters will be entered in the form of real values. The number of parameters varies based on the distribution type. Non-table SDF types include beta, constant, erlang, exponential, file count, file linear, gamma, IEEE backoff, key linear, log normal, message count, message linear, normal, pattern from file, semaphore linear, triangle, and uniform.

Table Distribution Parameters

The table distribution types include pattern, random linear, and random step. The pattern distribution has a simple table of values which is controlled by Add Cut and Move buttons.

The random linear and random step distributions also have a percent probability associated with each value entry, so their tables consist of rows of value and percent boxes. To enter a new value/probability percent row to the bottom of the table, enter a value and percent in the Value and Percent boxes to the right of the table, and then click on the Add to Table button. To change either the value or percent of a row in the table, click directly on the value or percent box of the row in the table. The Value and Percent boxes to the right of the table will then display the current values for the row. Enter a new value for either or both of these boxes and select the Update Table button. To remove a table row, highlight the value or percent box of the row to be deleted and then select the Remove button. The probabilities may be displayed either cumulatively or discretely by selecting the appropriately named radio button. A table of cumulative probabilities will show the probability of a particular table value with the probabilities of all previous values added to it. The first probability in this table should be 0 and the last should be 100. A table of discrete probabilities displays a table value's probability independent of all other table values. The probabilities in such a table should sum to 100.

3.5.15 Semaphore Form



The Semaphore Form

Maximum Pending Responses
The semaphore's maximum pending responses must be input as an integer value.

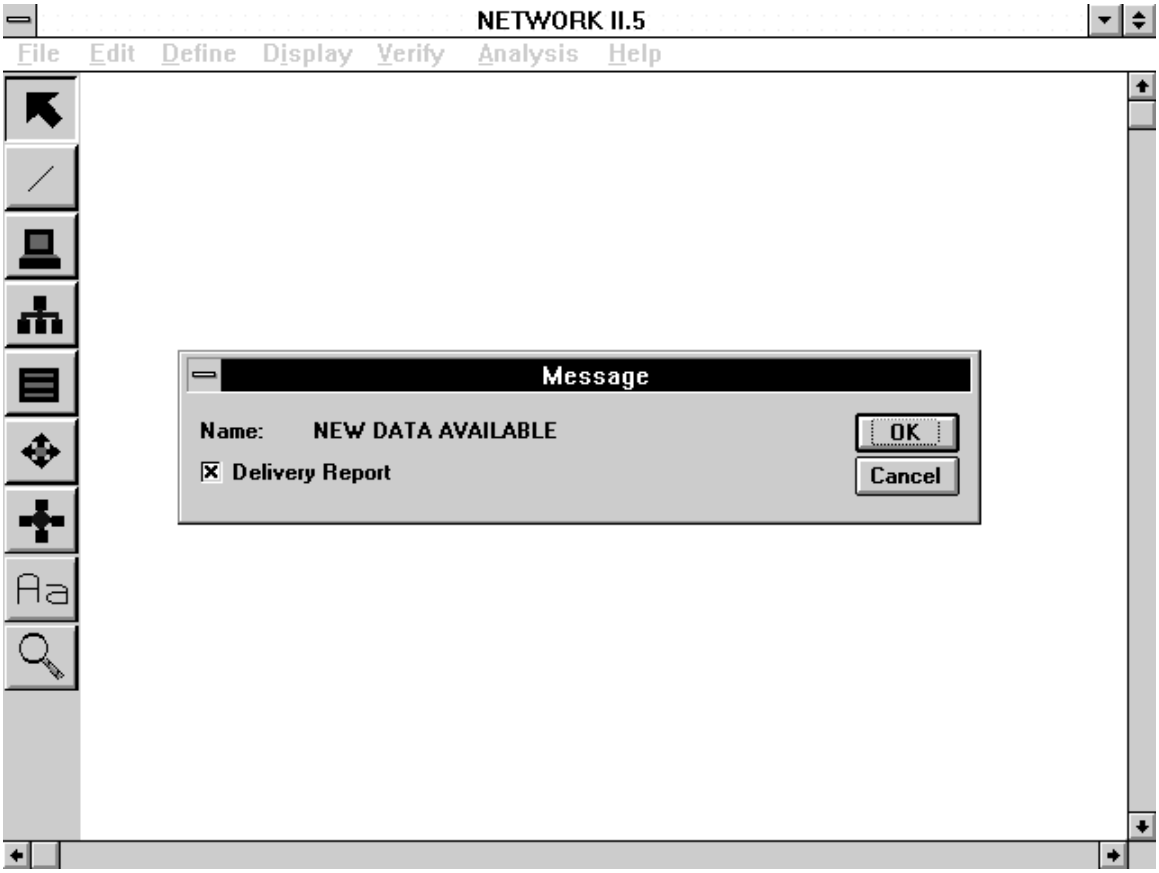
Initial Semaphore Count
The semaphore's initial count must be input as an integer value.

Initial Semaphore Status
Either Set or Reset at simulation startup.

Measure Response Time
Response times will be computed for this semaphore when this check box is selected.

Include Count in Plot File, Include Response in Plot File
These check boxes determine if data on the Semaphore's count and response will be added to the plot file during a simulation.

3.5.16 Message Form



The Message Form

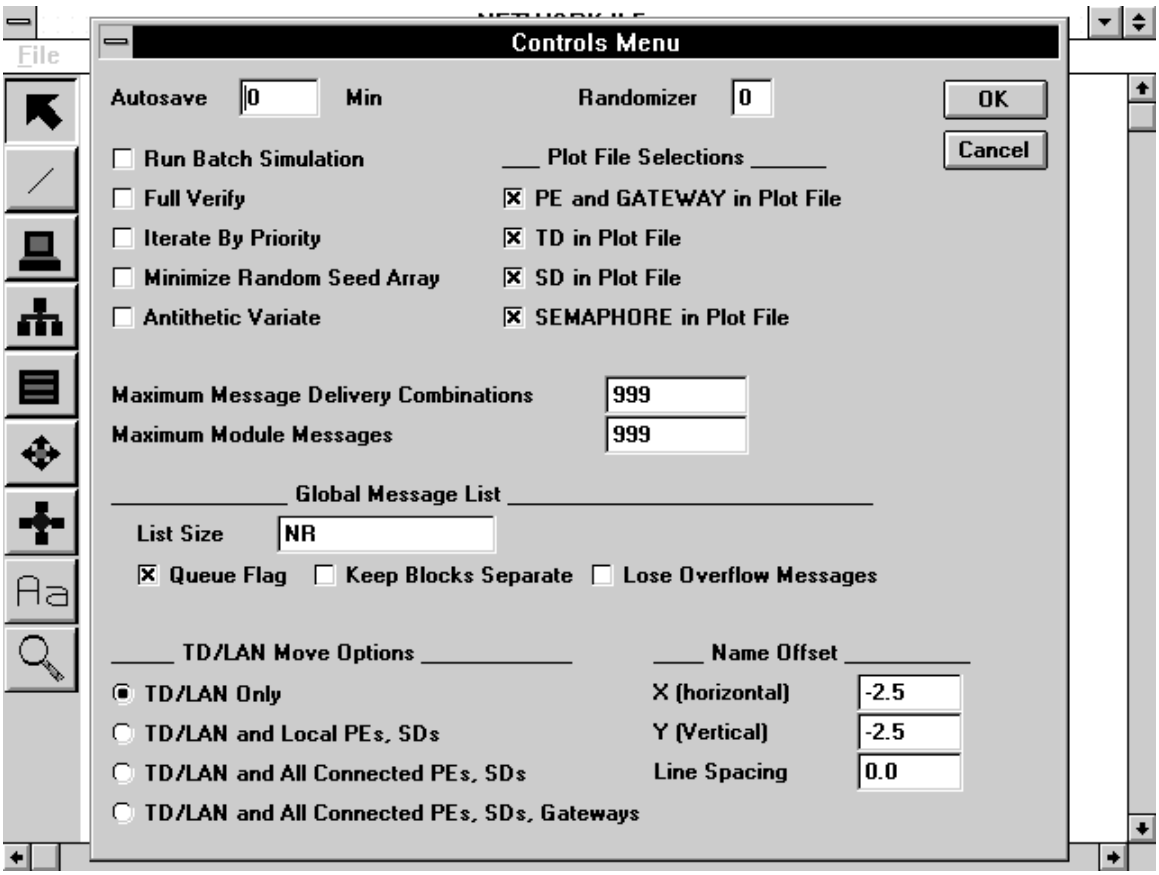
Delivery Report
The Delivery Report flag will be set to YES when this check box is selected and determines if the message will appear in the Delivery Report.

3.5.17 Controls Form

The Controls Form allows you to manage NETWORK (Full Verify, Autosave, TD/LAN Move Options) and change certain simulation parameters (Iterate by Priority, Minimize Random Seed Array, Antithetic Variate, Max Message Delivery Combinations, Maximum Module Messages, and Randomizer).

Full Verify
The Full Verify check box determines the type of verification report produced when Verify is invoked either at the top level or from a data structure form. If the Full Verify

check box is not selected, a brief description of verification errors is given. If the Full Verify check box is selected, additional notes on default parameters and model characteristics are given in the verification report.



The Controls Form

Antithetic Variate, Randomizer

The Randomizer and Antithetic Variate parameters provide short cuts in defining the random number seeds. For example, if a model contained 100 different SDFs, manually editing each SDF would be tedious.

The Randomizer is an integer between 0 and 9. It specifies the ordering of the random number seeds in the array generated when the simulation is being initialized.

The Antithetic Variate is a YES/NO value. It takes advantage of the principle that all distributions can be derived from a uniform distribution whose values range from 0 to 1. All distributions are derived in this manner.

If the Antithetic Variate flag is set to YES, the value picked from the uniform distribution is subtracted from 1. That value is used to calculate the value of the SDF in question.

Iterate by Priority

If the Iterate by Priority check box is selected, Modules scheduled by iteration period at the same time will be scheduled in the order of highest priority. If the Iterate by Priority check box is not selected, a Module with a lower priority than another Module scheduled at the same time may be activated first.

Run Batch Simulation
This check box is intended for those platforms in which a non-interactive simulation can be launched. If the box is marked YES, SIMWORK will use the run parameters saved in the model description to control the simulation.

Minimize Random Seed Array
If the Minimize Random Seed Array check box is selected, random seed array assignments will use as few seeds as possible, as close to each other as possible. If the Minimize Random Seed Array check box is not selected, random seed array assignments are made as they were prior to NETWORK II.5 R6.0.

Maximum Message Delivery Combinations
The Maximum Message Delivery Combinations control allows you to specify the number of delivery combinations a message will traverse before the simulation program generates a runtime warning. The runtime warning notifies you of potential memory wasting conditions. A message delivery combination is uniquely identified by the source PE, message text, and destination PE (either interim or final). The default value is 999.

Maximum Module Messages
The Maximum Module Messages control allows you to specify the number of messages a Module may inherit from a predecessor before the simulation program generates a runtime warning. The runtime warning notifies you of potential memory wasting conditions. The default is 999.

TD/LAN Move Options
The manner in which a TD or LAN is moved on the display when you double click on its icon is dependent on the setting of the TD/LAN Move Option. The TD/LAN may be moved independent of all connected devices, with all local PEs/SDs (i.e. PE/SD only has connection to one TD/LAN), with all connected PEs/SDs, or with all connected PEs/SDs/Gateways.

Autosave
The Autosave value box indicates the number of minutes that will elapse in a NETWORK session before an autosave is invoked. When an Autosave occurs, NETWORK saves the current model to the file AUTOSAVE.NET in the current directory. When NETWORK begins execution, the default Autosave value is 0 minutes. Autosave can be set from 0 (no autosave will occur) to 9999 minutes.

Plot File Selections

Check boxes exist which allow you to include (box is marked YES) or exclude a type of data structure from the Simulation Plot File. The choices are PE and Gateway, TD, SD and Semaphore.

Global Message List

This section of the Controls Form permits you to set the characteristics of the Global Message List. The List Size box will accept real or integer values. Real values are permitted to allow values greater than the maximum integer, and will be truncated because this field is measured in bits. The Global Message List is the global equivalent of a PE's Received Message List and it has similar check boxes to define it; Queue Flag, Keep Blocks Separate and Lose Overflow Messages can be set to YES or NO.

Name Offset

Three attributes are grouped under the Name Offset label to control how names of new data structures are shown on the display. The attributes can be set for an individual data structure in its graphics form. The X (horizontal) value controls how far the starting position of a name is offset from the center of an icon. A negative value positions names to the left, a value of zero will start a name at the icon's center, and a positive value moves names to the right. The Y (vertical) value controls the vertical offset of names in relation to the center of an icon. Positive values will move the name above the icon's center, while negative values move the name downward. The Line Spacing value is used to control the spacing of lines for names that are broken into more than one line.

3.6 Graphics Forms

Each of the hardware data structures of NETWORK II.5 has graphical attributes to describe its appearance on the display. The graphics form of a data structure is accessed from within the attributes form of the data structure. The attributes and controls common to all or most of the graphics forms will be described next, followed by illustrations and descriptions of each of the specific graphics forms found in NETWORK.

3.6.1 Common Graphics Forms Attributes and Controls

Name
The name is included for informational purposes and is not selectable. If you want to change the name of a data structure, you must enter a new name in the appropriate Attributes Form.

Position
Select the Position button to change the placement of an icon on the display. The icon will move with the cursor until you press the mouse button to anchor the icon in a new location.

View
You may view how changes made in a graphics form will affect the a data structure's appearance on the display by selecting the View button. After you are finished viewing the changes, click on the Continue button to return control to the current graphics form. If the graphical changes made to an icon are unsatisfactory, merely cancel from the graphics form and the old graphics attributes will be retained.

Hide
If the Hide flag is set to YES (the check box contains a mark), the data structure will not be shown on the display. The hidden item will still be available for editing through the Define/(data structure type) command. To display a hidden icon, click again on the Hide check box to set the Hide flag to NO (mark is removed from the check box).

Icon
The Icon combo box is used to select the icon used to represent a hardware data structure on the display. The list includes a variety of icon names, as well as the default icon a hardware data structure has when created with the toolbar (shown as <default icon> in the name list) and a group of basic shape icons. The basic shape icons are listed as <rectangle icon> (available only for PEs), <hexagon icon> (available only for SDs), and

<octagon icon> (available only for Gateways). The special basic shape icons cannot be scaled or rotated.

Color

The Color combo box is available on the PE and Gateway graphics forms only. To change the color of an icon, select the down arrow of the Color combo box and choose a new color from the list.

Scale

The icon scale controls the size of a hardware data structure's icon on the display. The scale value ranges from 0.01 to 10. The default scale of 1 presents a file icon at the size it was created. Increasing the scale above 1 increases the size of the icon. Decreasing the scale below 1 decreases the file icon size. The special basic shape icons rectangle (PE), octagon (Gateway) and hexagon (SD) cannot be scaled. Because a TD/LAN is drawn using line segments, it has Horizontal and Vertical Scale parameters.

Rotate

The icon rotation controls the angle at which a hardware data structure's icon is displayed. The rotation value may range from -359.9 degrees to 359.9 degrees. The default rotation is 0 degrees. Only a file icon may be rotated. Attempting to rotate an internal icon will have no effect on the icon.

Name Offset

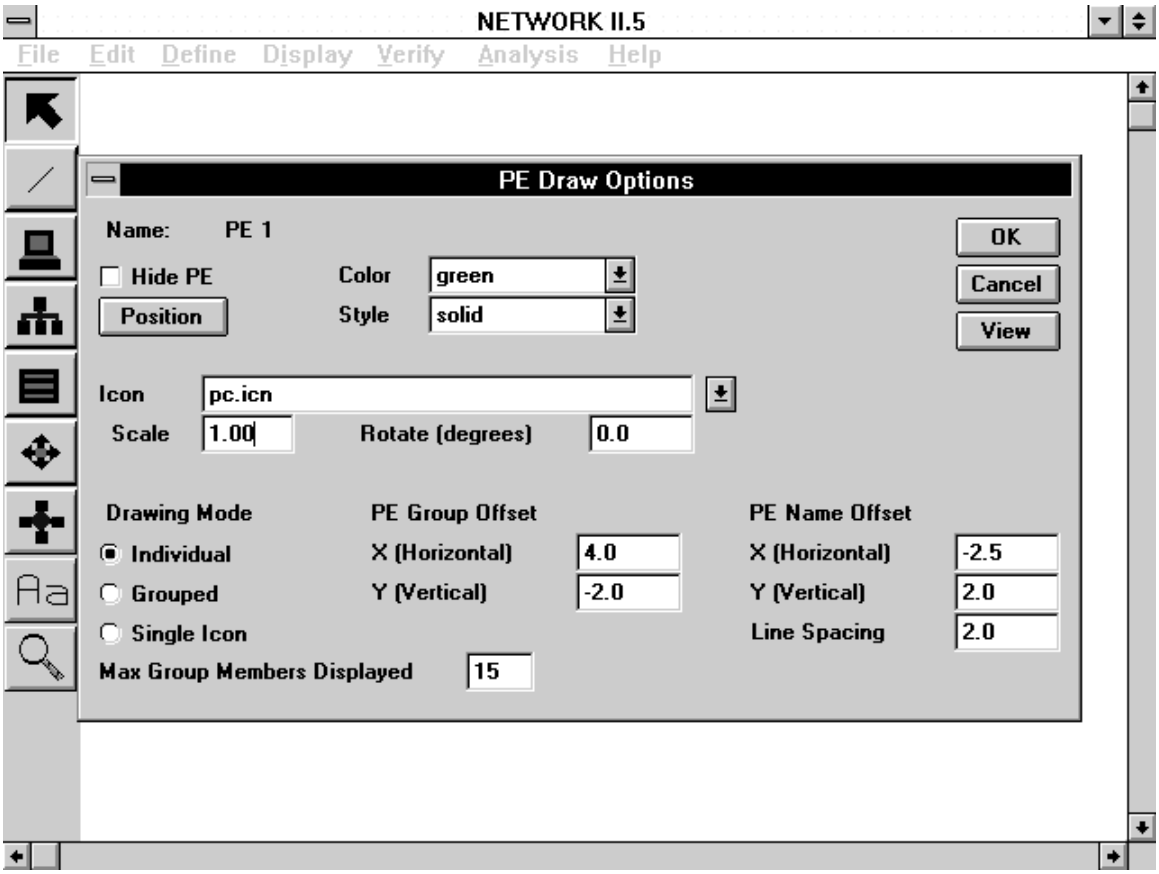
The PE, Gateway and SD graphics forms have the three Name Offset fields that were described above in the Controls Form. Setting the X, Y, and Name Space attributes will allow you to reposition an icon's name in relation to the center of the icon.

Style

You may change the fill style of any of the basic shape icons (rectangle, hexagon, octagon) by selecting the down arrow of the Style combo box and choosing a style from the available choices (solid, diagonal, small diagonal, dash diagonal, cross hatch, large hatch and dash hatch).

3.6.2 Processing Element Graphics Form

In addition to the graphics forms attributes mentioned above, you will find attributes related to the Quantity field of a PE.



The PE Graphics Form

Drawing Mode

There are three PE Drawing Mode options you may select in the PE Draw Options Form. Individual mode will create a PE icon for each copy of the PE group. Each of these copies can be positioned independently. Grouped mode will also create a PE icon for each copy of a PE group, but each PE icon will be positioned relative to the group. Single Icon mode will create a single PE icon to represent an entire PE group, regardless of the quantity of the PE group.

Max Group Members Displayed

Max Group Members Displayed will limit the number of copies of a PE group that can be shown on the display. If the quantity of a PE group exceeds this number, it will be shown in Single Icon mode on the display.

PE Group Offset

The X (horizontal) and Y (vertical) values of PE Group Offset are used to position copies of a PE group displayed in Individual or Group mode. You may change either of these values by clicking in the box and entering a real value. When a PE group is displayed,

the first icon is presented and then succeeding group icons are displayed by adding the offset values to the coordinates of the previously displayed icon.

3.6.3 Gateway Graphics Form

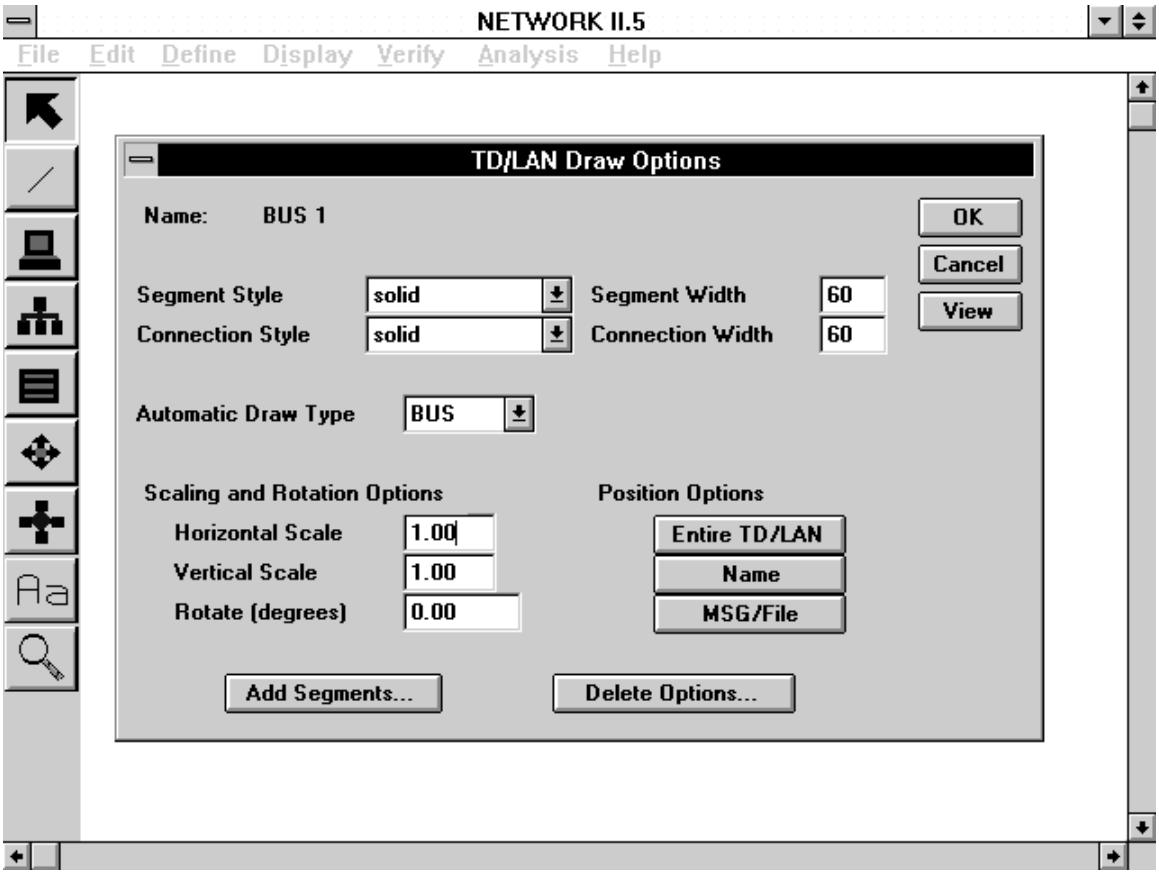
The Gateway Graphics Form is nearly the same as the PE Graphics Form. There is no Drawing Mode or Group Offset controls on the Gateway's form because Gateways do not have a Quantity attribute.

3.6.4 Storage Device Graphics Form

The Storage Device Graphics Form has all of the fields found on the Gateway Graphics Form with one exception. There is no Color combo box on the SD Graphics Form because Storage Devices do not have a color attribute.

3.6.5 Transfer Device/LAN Graphics Form

Because Transfer Devices and LANs are drawn as line segments in NETWORK II.5, the TD Graphics Form is unlike the other graphics forms described above.



The TD/LAN Graphics Form

Segment Style

The Segment Style combo box controls the line pattern used for the Transfer Device line segments. Five different line styles are available. To change the line style, select the down arrow of the combo box and make a selection from the list.

Segment Width

The segment width controls the line width used for the Transfer Device. The segment width may range from 0 to 999. The default segment width is 120. Also, it is worth noting that the increase/decrease in segment width is relative to the model scale. For example, changing the segment width from 120 to 190 with a model scaled at 30% will not be noticeable. However, if the model scale is 80%, you will be able to perceive the difference.

Connection Style

The Segment Style combo box controls the line pattern used for the Transfer Device connections to PEs, SD and Gateways. Five different line styles are available. To change the line style, select the down arrow of the combo box and make a selection from the list.

Connection Width

The connection width controls the line width used for the Transfer Device connections. The segment width may range from 0 to 999. The default segment width is 120.

Automatic Draw Type

You may change the drawing type of the Transfer Device by selecting the down arrow of the Automatic Draw Type combo box. The available selections include Bus, Ring, Star, and User.

Position Options

Three buttons are available for positioning Transfer Devices and LANs. Select the Entire TD/LAN button to move the Transfer Device or LAN on the display. The TD/LAN icon will move with the cursor until you click on the mouse again to anchor the TD/LAN icon in a new position. The Name and MSG/File buttons will move only the TD/LAN name and TD/LAN Msg/File locations respectively.

Horizontal Scale

The Horizontal Scale controls the horizontal size of the TD icon on the display. The horizontal scale value ranges from 0.01 to 10. The default scale of 1 presents the TD icon at its original size. Increasing the scale above 1 increases the horizontal size of the TD icon. Decreasing the scale below 1 decreases the horizontal TD icon size.

Vertical Scale

The Vertical Scale controls the vertical size of the TD icon on the display. The vertical scale value ranges from 0.01 to 10. The default scale of 1 presents the TD icon at its original size. Increasing the scale above 1 increases the vertical size of the TD icon. Decreasing the scale below 1 decreases the vertical TD icon size.

Rotate

The rotation controls the angle at which a TD icon is displayed. The rotation value may range from -359.9 degrees to 359.9 degrees. The default rotation is 0 degrees.

Add Segments

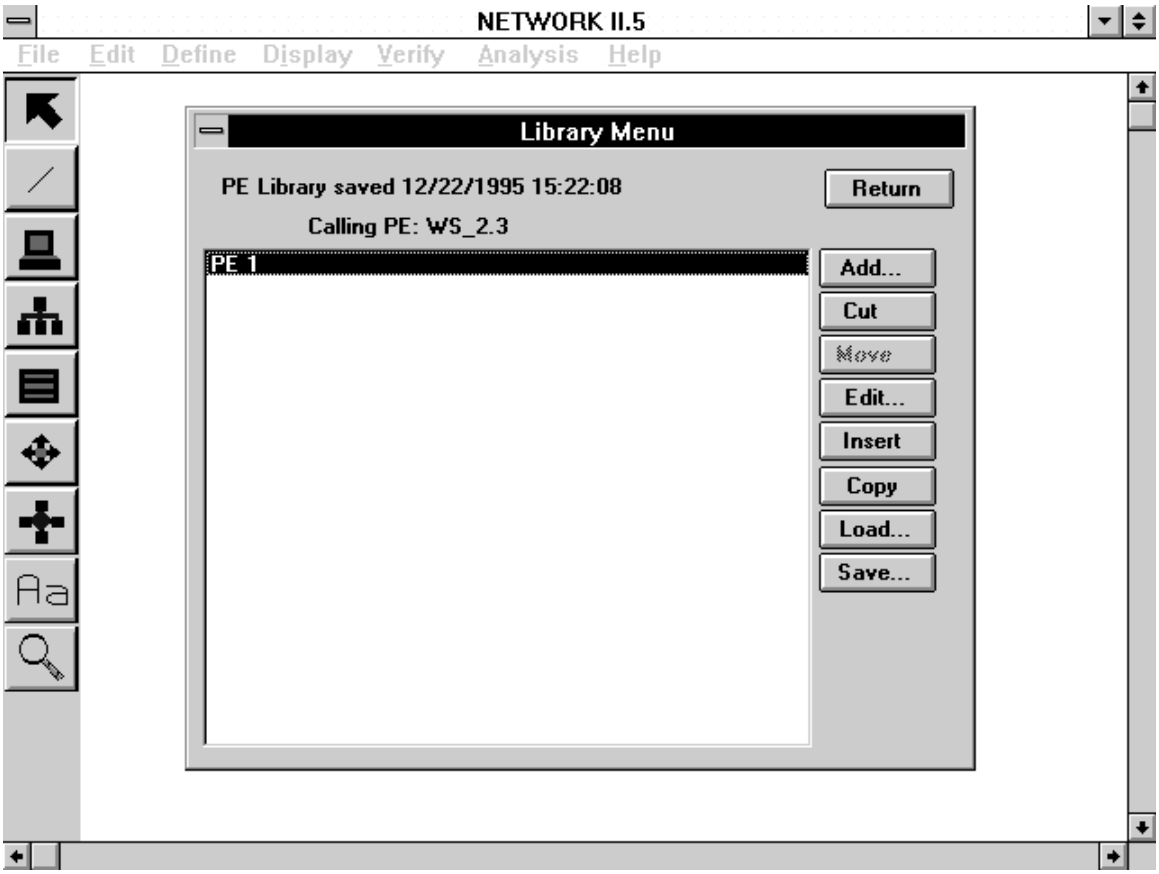
By selecting the Add Segments button, you may add line segments to the TD icon. After selecting this command, a ribbon cursor that is attached to the last TD segment point appears. Move the cursor to a new position and click once to anchor a new segment. You may create as many segments as you need. To return control to the TD/LAN Draw Options Form, double click the mouse at the same location. The Automatic Draw Type of the Transfer Device/LAN will be changed to user when you add a segment.

Delete Options

To remove all or part of the graphical description of a TD/LAN, select the Delete Options button. Options are available using this command to delete the Entire TD/LAN, the Name label, the MSG/File label, connections, and the last segment of the TD/LAN icon.

3.7 LIBRARY

NETWORK provides a library command for PEs, TDs, SDs, and Modules. Each of the attribute forms of these data types will permit access to the library command by selecting the Library button. You may save data elements to a library file, examine and edit the attributes of items saved in library files, and load the attributes of library items into your current model. When the library command is called, NETWORK displays the Library Form from which all of the library operations proceed.



The Library Form

The Library Form has a list box in its center which is used to display the names of all current library items, and a group of control buttons to the right of the list box. You may load a library file by selecting the Load button. When a library file is loaded, the names of all items in the file are then displayed in the list box.

The four different types of library files will have a specific file suffix attached to the names you give them. PE library files are designated by a .PEL suffix. TD library files are designated by a .TDL suffix. SD library files are designated by a .SDL suffix. Module library files are designated by a .MDL suffix. You are not limited to only one library file for each data type. You may create as many library files for each data type as you desire by specifying a different name for each file.

You may save a library file by selecting the Save button. NETWORK will save all items found in the list box to the library file. A new file may be created by entering a new name or an old library file can be updated by selecting its name.

The Add button is used to add the current data structure being edited to the list of library items. If the item added is a PE, you may also save all Modules associated with the PE to the PE library. If the item added to the library name list is to be saved in a library file, you must use the library Save command before leaving the library form.

You can remove items from the list by highlighting a name and then selecting the Cut button. However, if the file is not saved after this is done, the item will still reside in the library file because Cut merely removes the name from the list, but the Save button saves the current library list.

You may position items in the library name list by highlighting a name and then selecting the Move button. A new list position is then given by selecting one of the numbers inserted between the names.

If you wish to examine or edit a library item, use the Edit button after you have highlighted the name of the item. The attribute form for the data structure type of the library item selected will be shown with the current attributes of the library item. Most of the attributes of the library item may be changed, including the name. Again, these changes are not retained unless you use the Save button to save the new attributes in a file.

You may copy the attributes of a library element into the data structure of your current model that is being edited by using the Insert button. To do this, highlight a name in the name list of library items and then select Insert. The attributes of the data structure will then be the same as those of the library item when you return from the library form.

The Copy button allows you to copy a library element from the current list of library elements directly to a different library file. After the name of an item is highlighted and the Copy button is selected, NETWORK will request a library file name in which the selected item will be saved.

4. RUNNING A SIMULATION

4.1 INTRODUCTION

A NETWORK II.5 simulation is normally launched directly from the menubar in NETWORK by selecting `Analysis`, `Start Simulation`. It also can be started by invoking the SIMWORK or TEXTWORK programs directly. SIMWORK is the graphics based simulation program which is activated by NETWORK. TEXTWORK (not available for Windows 3.1) is the same simulation program with a text based user interface. The purpose of providing a text based interface to the modeling engine is to allow batch simulation runs. TEXTWORK also allows running a simulation from a remote computer which only has command line access to a host computer. An example of using remote command line access would be to run a simulation over the internet using a telnet session.

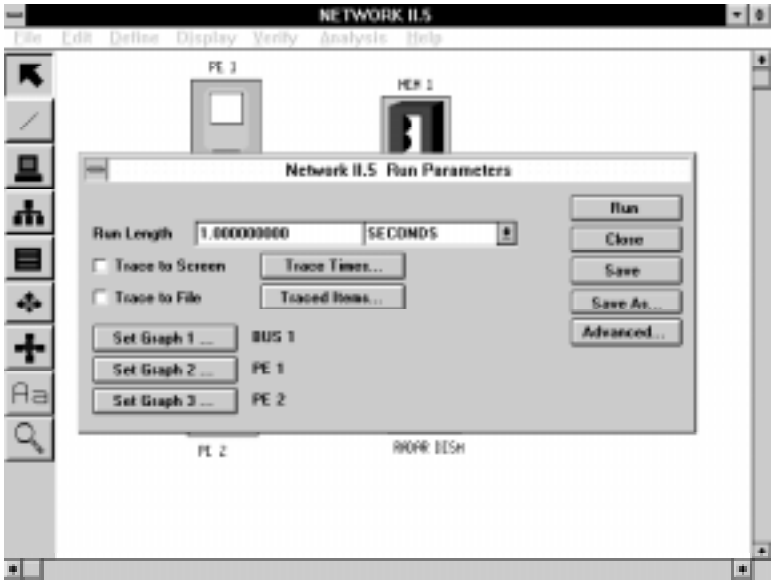
During the simulation, reports and trace messages are available. In addition, you will be notified of simulation warning and error conditions as they occur throughout the simulation run. Output produced by the simulation program includes the simulation Report File (file type .LIS) and the simulation Plot File (file type .PIN). The simulation Report File is a text file which contains summary statistics from the simulation run. This is the file presented when you `Browse, Reports` from within NETWORK. The file consists of plain text and can be printed and viewed with an editor. The simulation Plot File is a binary file which is used by NETWORK to generate plots and animation. The Plot File is not normally examined by the user.

4.2 SIMWORK

SIMWORK is the simulation program with a graphics based user interface. SIMWORK is started from within NETWORK by selecting `Analysis, Start Simulation` from the menubar. Prior to starting SIMWORK, you should have opened or created a model containing at least one Module to be executed by a Processing Element.

When you invoke SIMWORK, it begins execution by displaying the Run Parameters form where you may set the values that control your simulation. The `Run Length` is the only run parameter that must be supplied. If no other parameters are entered, the simulation will produce a progress graph, a simulation Plot File (file type .PIN) for the entire simulation and a simulation Report File (file type .LIS) which contains every

applicable report. Fill in the run parameters as appropriate, and then click the Run button to begin the simulation. Additional run parameters can be accessed on the Advanced Options Form when you click the `Advanced...` button from the Run Parameters form.



The Run Parameters Form

4.2.1 Run Parameters Form

Run Length

The Run Length represents the amount of simulated time to run the simulation. Run Length is a real number greater than zero. Use the combo box to change the simulation time units to seconds, milliseconds, microseconds or nanoseconds. If the Run Length time unit is changed, all other time values will be scaled to the new unit. In addition, the trace and warning messages will be produced using the Run Length time units.

Trace to Screen, Trace to File

You can generate a narrative trace report of simulation by selecting either `Trace to Screen` or `Trace to File` (or both). When you select either (or both) of these, the Traced Items List must also be specified. The Trace Start Time will default to 0 and the Trace Stop time will default to the current Run Length. Click on `Trace Times...` to select a different interval.

Trace Times

To display the Time Menu in order to set new trace start and stop times, click on the `Trace Times...` button. The Trace Start Time must be greater than or equal to zero and less than the simulation Run Length. The default Trace Start Time is zero. The

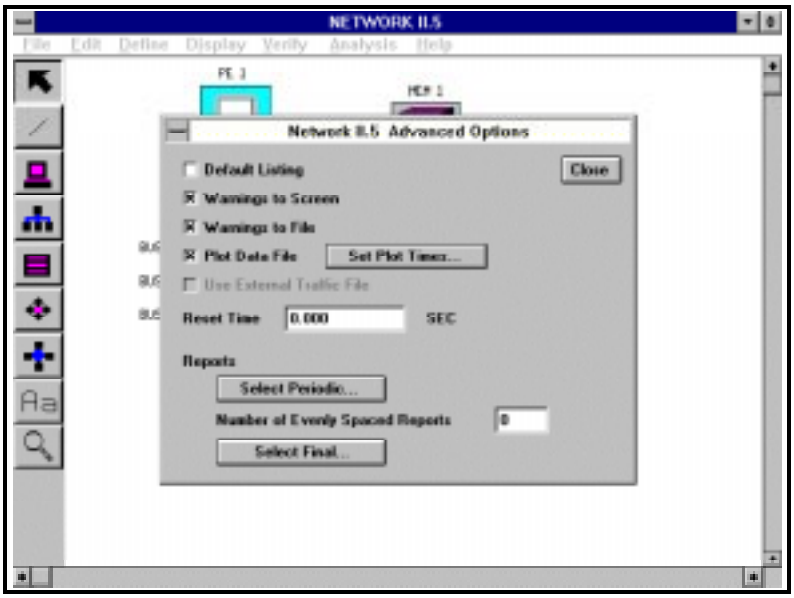
Trace Stop Time must be greater than the Trace Start Time and less than or equal to the simulation Run Length. The default Trace Stop Time is the simulation Run Length.

Traced Items

The Traced Items List may be displayed and modified by clicking on the `Traced Items...` button. Use the `Add` and `Cut` buttons to enter and remove names of items that are to be traced during the simulation. Everything in your model can be traced by selecting ALL and entire classes may be selected by clicking on ALL PES, ALL TDS, ALL SDS, ALL INSTRUCTIONS, and ALL MODULES. External traffic which was captured with a LAN Analyzer may be traced by selecting <External Traffic>.

Set Graph 1..., Set Graph 2..., Set Graph 3...

Three separate utilization graphs may be displayed during the simulation. To select the name of a Processing Element, Storage Device, or Transfer Device, click the Set Graph buttons and then select a device name.



The Advanced Options Form

4.2.2 Advanced Options Form

Default Listing

By selecting the `Default Listing` option, a copy of the internal model database used in the simulation will be saved in the simulation Report File. Any required

parameters omitted in the Network Description File will have their default values displayed in this listing.

Periodic Reports

If the number of Periodic Reports is greater than zero, the simulation program will generate the specified number of summary reports evenly spaced over the course of the simulation. The number of reports does not include the final report at the end of the simulation. The Periodic Reports generated cover from start of simulation (or time of statistics reset, if applicable) until the time of the report. The same reports offered at the end of simulation are offered as Periodic Reports. The Periodic Reports List may be edited by clicking on the `Select Periodic...` button located on the Advanced Options Form. For example, if the simulation run length is one second and you would like reports every quarter second, request three periodic reports. Periodic Reports will be produced at 0.25 seconds, 0.50 seconds, and 0.75 seconds. The final report will be produced at 1.0 second.

Reset Time

Simulation statistics can be reset by specifying a `Reset Time` greater than zero. If a reset time is specified, all simulation activity that occurred prior to the reset time is not included in the summary reports. This feature is valuable to warm up a simulation whose start-up activities are different from its steady-state activities. If a Periodic Report is generated before the reset time is reached, it will include statistics from the start of simulation to the time of the report.

Runtime Warnings

The `Warnings to Screen` and `Warnings to File` check boxes determine how runtime warnings are handled. Runtime warnings notify you of illogical and/or impossible conditions encountered during the simulation. Not all problems can be identified by the static test performed by Verify. Some problems only turn up during the dynamics of a simulation. The runtime messages may be sent to the simulation Report File and/or your display. If runtime warnings are directed to your display, the simulation will halt after each message to await user interaction. By default, runtime warnings go to both the display and the simulation Report File. You might want to ignore runtime messages either to let the simulation continue regardless of warnings and/or to reduce the size of the simulation Report File.

If `Warnings to File` is checked and `Warnings to Screen` is not, the simulation will not halt when a runtime warning is encountered. Any warnings that occur during the simulation will be written in the simulation Report File.

If neither the Warnings to Screen option nor the Warnings to File option is selected, the simulation will only count the number of runtime warnings encountered during the simulation. The count will then be displayed in the Simulation Completed message box when the simulation concludes.

A complete description of the runtime warnings that may be encountered during a simulation is given in Chapter 9, Debugging a NETWORK II.5 Model.

Plot Data to File

If Plot Data to File is checked, the simulation program will generate a simulation Plot File (file type .PIN). The simulation Plot File is a binary file which is used by NETWORK's plotting and animation features.

The Plot Start Time and Plot Stop Time may be entered in the dialog box that appears after clicking on Set Plot Times.... The Plot Start Time must be greater than or equal to zero and less than the simulation Run Length. The Plot Stop Time must be greater than the Plot Start Time and less than or equal to the simulation Run Length.

Final Reports

The reports included in the final report at the end of the simulation may be selected by editing the Final Reports List. To access this list, click on the Select Final... button in the on the Advanced form. By default, all of the available reports will be printed (Processing, Instruction, Transfer Device, Storage Device, Module, Semaphore, Message Statistics, Received Message, Message Delivery, and Snapshot). The order that reports appear in this list is the order that they will be put in the simulation Report File. This allows you to both choose which reports are presented and the order in which they are presented.

4.2.3 Run Parameters Buttons

Save

The Save command saves the simulation run parameters in the Network Description File (file type .NET). The next time you load the saved file into NETWORK, these parameters will be used to initialize the NETWORK Run Parameters Form. Because NETWORK performs a Save every time that you Run a simulation, it is generally not necessary to perform a Save unless you want to set up the run parameters without actually running a simulation.

Save As...

The `Save As...` button allows you to save your model under a new name. The current name of your file will appear in this box as the default. To save to a different name, click in the text box and enter a new name.

Close

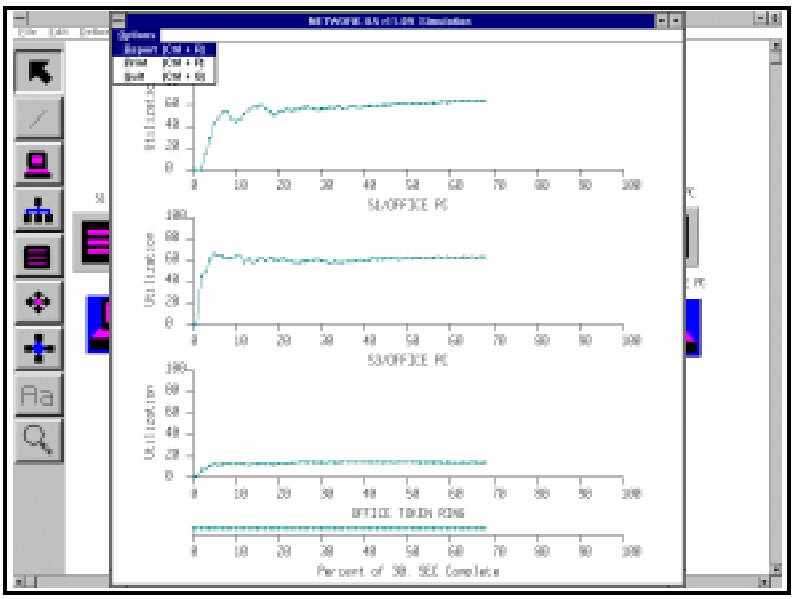
The Close command terminates this simulation session, returning you to NETWORK if simulation was invoked from NETWORK.

Run

The simulation is started by selecting the `Run` button. All Run Parameters are checked for errors at this time and you will not be allowed to start the simulation until any errors noted are corrected.

4.2.4 The Simulation Runtime Menubar

You may interrupt a running simulation by clicking on the `Options` field of the menubar present during the simulation. The pulldown offers to generate a `Report`, to `Print` what currently is on the screen or to `Quit`. The simulation pauses while you make your selection. If you select `Report`, a dialog box will open up from which you can select which report(s) you would like to see. The simulation will remain paused until you have finished viewing the requested reports.



The Run Time Menubar

Your choice of reports is the same as you have for Periodic and Final Reports with the addition of one other report, Memory Utilization. The Memory Utilization Report lists those NETWORK II.5 data structures currently in memory, giving the number and size of each type. This report is provided to allow a user to estimate the amount of memory a simulation is using. All of the other summary reports are described in chapter 5.

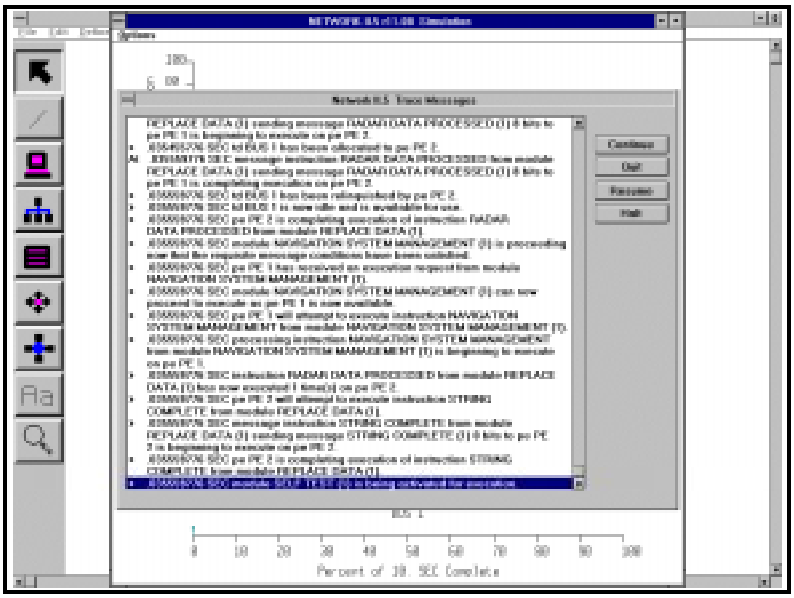
If you select Print, a dialog will open up that allows you to direct what is on the screen out to a printer or a file. The exact dialog depends on the host system running NETWORK. If you select Quit, the simulation will be halted and final reports written to the Listing file as of the current simulation time.

4.2.5 Simulation Progress Graph

The Simulation Progress Graph measures the percentage of the simulation which has been completed and is displayed along the bottom of the screen. The Simulation Progress Graph is updated at an interval of 1/50th of the simulation run length. The runtime menubar picture shows a progress graph.

4.2.6 Trace Reports

The runtime trace report allows you to monitor simulation events. The menubar commands are the same when a runtime trace report is displayed as when it is not. The menubar commands were described in the preceding paragraphs.



A Sample Runtime Trace

Trace Report Button: `Halt`

The Halt command discontinues the simulation trace to both the display and the simulation Report File (if specified). Halt merely stops the generation of trace messages. The simulation will continue until the specified stop time. A confirmation is requested before the Halt command is accepted.

Trace Report Button: `Resume`

The Resume command temporarily discontinues the simulation trace from the current time to a user specified time in the future. You will be asked to enter a Resume Time which must be a value greater than the current simulation time and less than the simulation Run Length. When the simulation time reaches the Resume Time, the simulation trace will be continued.

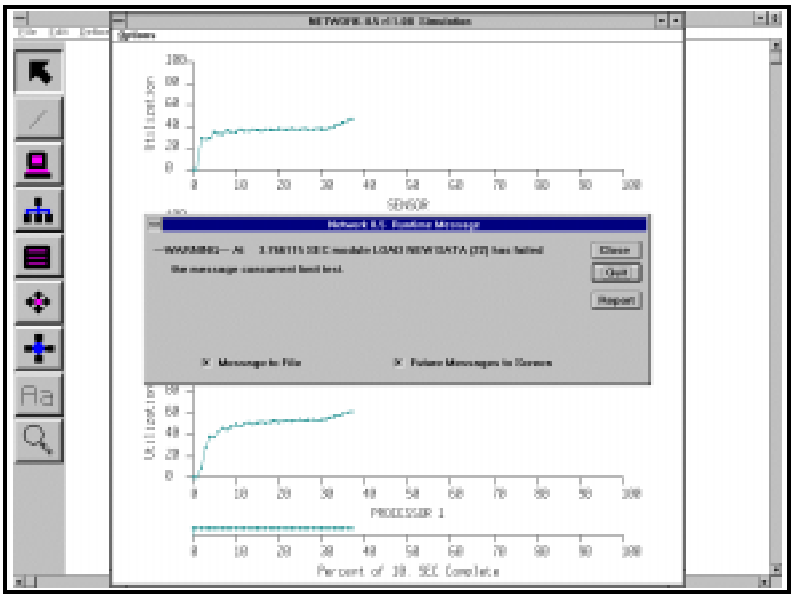
Trace Report Button: `Continue`

The Continue command advances the simulation to the next listing of traced events. The list of events is updated only when there is enough to fill the list box. This is normally about 20 lines which represents about 5 events.

4.2.7 .i.Runtime Warnings

Runtime warnings notify you that a potential problem has occurred in the simulation. If a runtime warning occurs and the Warnings to Screen option has been selected, the Runtime Warning Form will be displayed.

The Runtime Warning Form provides several reporting and course-of-action options including Close, Quit, Report, Message to File, and Future Messages to Screen. In addition, the Print option from the menubar remains active. These options allow you to determine the state of the simulation at the time of the runtime warning, and to prescribe how to handle future runtime warnings. Each of these options is described below.



A Sample Runtime Error

Runtime Warning Button: `Report`

Summary reports containing simulation statistics may be generated by clicking on the Report button of the Runtime Warnings Form. Select the name of a desired report from the list that is presented or use the Cancel command to return immediately to the simulation. Your choice of reports is the same as you have from the menubar.

Runtime Warning Button: `Quit`

The Quit button terminates the simulation run. A confirmation will be requested prior to exiting. The Final Reports are produced as of the time the simulation was canceled.

Runtime Warning Button: `Continue`

Selecting the Continue command will resume the simulation from the point at which it was interrupted. The action that the simulation will take in response to the runtime warning will be included in the runtime warning message.

Runtime Warning Check Boxes:

`Message to File`

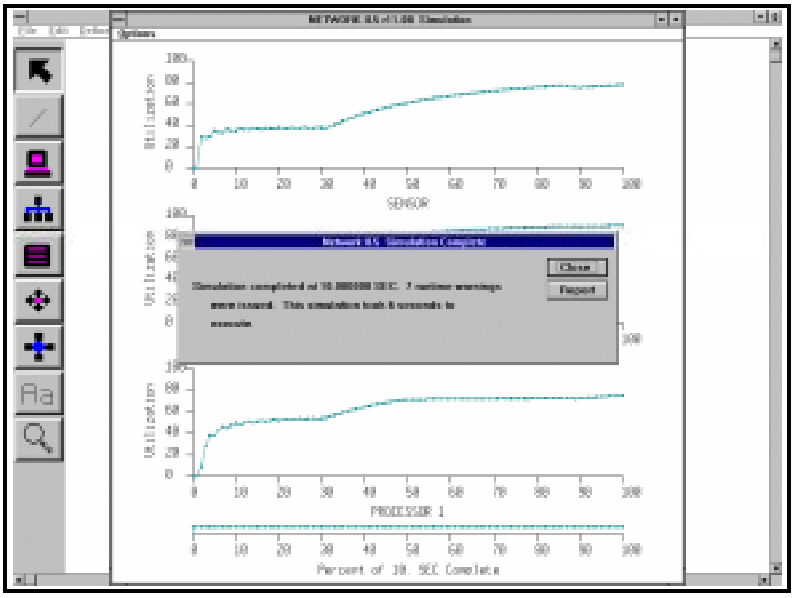
`Future Messages to Screen`

The Message to File and Future Messages to Screen options allow you to enable/disable the notification of future runtime warning messages. If the same type of runtime

warnings are being generated frequently, these options allow you to speed up the simulation execution and reduce the size of the simulation Report File.

4.2.8 The Simulation Completed Message

At the end of the simulation, the simulation completed message form is displayed. This message includes the actual time to run the simulation and the number of runtime warnings encountered. Click on the View button to observe the runtime utilization graphs (if they have been created). Final summary reports can also be viewed at this time by selecting the Report button.



The End of Simulation Message

4.3 .i.TEXTWORK

TEXTWORK is the text-based user interface simulation program. The program is started by typing `TEXTWORK` at the command prompt (`textwork` on UNIX platforms). TEXTWORK is available on all platforms with the exception of Windows 3.1.

After starting TEXTWORK, you will be asked to enter the model name, and the simulation run parameters. When all of the prompts from TEXTWORK have been answered, the simulation begins. During the run, you may monitor the simulation progress by interactively generating summary reports, snapshot reports, and trace reports.

The main advantage of TEXTWORK over SIMWORK is that you do not need graphics to run a simulation. Running TEXTWORK over the internet via a telnet session is one example where you will not have graphics capabilities. Also, it is easier to run TEXTWORK as a batch or background job because it does not try to open a graphics window. Because TEXTWORK does not incur the overhead of SIMWORK's graphical interface, a TEXTWORK simulation may run faster than a SIMWORK simulation, depending on your model.

4.3.1 Simulation Setup

The following paragraphs describe the steps necessary to load a file into TEXTWORK and run a simulation. The prompts you will encounter during a TEXTWORK run appear in the bordered sections of the text. Sample replies to the prompts are presented below these prompts and are indicated by ">".

4.3.2 Loading a File

Entering a File Name

TEXTWORK will prompt you for the name of the input file. The file name may include a path (e.g., d:\net\user\case1.net). If you do not specify a path in the file name, TEXTWORK assumes that the file is located in the default directory. The default directory is established by the N25DEV.CFG file.

```
Welcome to CACI NETWORK II.5, Release 11.00.

Please enter the name of the NETWORK II.5 input data file. (or ABORT)
> \case1

Input File = \CASE1.NET; Listing File = \CASE1.LIS
Title = CASE 1A - A 2 COMPUTER, 1 BUS ARCHITECTURE
```

TEXTWORK Setup Parameters: Input Listing

If the Input Listing option is selected, a copy of the Network Description File (.NET) used in the simulation will be inserted in the Simulation Report File (.LIS). This allows you to preserve the model that produced the simulation results.

```
Do you want a copy of your input file in the listing file?  (Y/N/ABORT)
> y
```

TEXTWORK loads the Model and conducts Pass One Error Check

As your input file is read in by TEXTWORK, the load progress report is displayed on your screen indicating the number of entities that have been loaded.

```
  8  Global Flags initialized.
  2  Processing Elements initialized.
  1  Transfer Devices initialized.
  2  Modules initialized.
```

If there are any pass one errors in the model, the resulting error messages will be written to the display and to the Simulation Report File (.LIS). Pass One checks the model for syntax and completeness. TEXTWORK terminates execution when a pass one error is encountered. If a pass one error occurs, load the model into NETWORK, verify the model, and correct the identified errors.

If an input listing was requested, the pass one error messages will be written below the line that caused the error. If the error resulted from an incomplete description, the error messages will be written at the end listing of the offending Module, PE, Transfer Device or Storage Device.

```
PASS ONE found FATAL ERRORS in the input file.  Simulation stopped.
Refer to the listing file for error messages.
```

If no pass one errors are found in the model, TEXTWORK continues the simulation setup dialog.

```
NETWORK II.5 took 1 seconds to read the 93 line data file
```

TEXTWORK Setup Parameters: Default Listing

After TEXTWORK has read in the input file, you are given the opportunity to send the default listing to the simulation Report File. The default listing provides a copy of the internal model database to be used in the simulation with all default and omitted values filled in. See Chapter 5 for further information.

```
Do you want a report in the listing file showing the input file complete
with all assumed default values?  (Y/N/ABORT)
> y
```

TEXTWORK conducts Pass Two Error Check

After building the internal model database, TEXTWORK conducts the pass two error check. Pass Two identifies model logic errors (e.g., a module with an allowed PE that does not exist). If an error is encountered, the resulting error message will be written to the simulation Report File. TEXTWORK terminates execution when a pass two error is encountered. If a pass two error occurs, load the model into NETWORK, verify the model, and correct the identified errors.

```
FATAL ERRORS found by PASS TWO in the input file.
Refer to the listing file for error messages.  Simulation stopped.
```

When the model has been successfully loaded by TEXTWORK, the following message is displayed.

```
NO FATAL INPUT ERRORS HAVE BEEN DETECTED.
```

4.3.3 Entering TEXTWORK Run Parameters

TEXTWORK Run Parameters: Time Units

You will be requested for the time units to be used for all the time-based simulation run parameters (Run Length, Reset Time, Plot Start Time, Plot Stop Time, Trace Start Time, Trace Stop Time). Your choices for time units are seconds, milliseconds, microseconds, and nanoseconds.

```
Enter the time unit which you wish to use for input.
It must be SECONDS, MILLISECONDS OR MICROSECONDS.  (SEC/MIL/MIC/NAN)
> s
```

TEXTWORK Run Parameters: Run Length

The Run Length tells TEXTWORK how long to run the simulation. The Run Length parameter is specified in simulated time, not actual time. Large numbers may be entered in scientific notation (e.g., 56,000 could be entered 5.6E4).

```
Enter length of simulation (IN SECONDS).
> 1000
```

Default Run Parameters

If you do not wish to specify additional run parameters (Reset Time, Periodic Reports, Final Reports, Plot Data File, LAN Analyzer File, and Runtime Warnings) before starting the simulation, answer NO to the question below. The default run parameters will then be used in the simulation (Reset Time = none, Periodic Reports = none, Final Reports = ALL, Plot Data File = yes, Plot Interval = entire simulation, LAN Analyzer File = none, Runtime Warnings = send to both Listing file and Terminal).

```
Do you want to define any additional run parameters? (Y/N/ABORT)
> n

Do you wish to trace the event flow? (Y/N/ABORT)
> n

Simulation begins.
```

TEXTWORK Run Parameters: Reset Time

The statistics collected by TEXTWORK may be reset at some point during the simulation by specifying a Reset Time. If a Reset Time is entered, the simulation activity which occurs before the Reset Time will be ignored in the simulation Report File's summary reports. This feature allows you to warm up a simulation whose start-up activities may be different from its steady-state activities.

```
Do you want the statistics reset during the run? (Y/N/ABORT)
> y

Enter the time to reset statistics  (IN SECONDS)
> 10
```

TEXTWORK Run Parameters: Periodic Reports

The periodic summary reports provide simulation statistics for the selected items in the Periodic Report List (PE, instruction, TD, SD, module, semaphore, message, snapshot). The reports are written to the Simulation Report File.

```
Do you wish to have periodic reporting? (Y/N/ABORT)
> y
```

Report Scheduling

If evenly-spaced reports are requested, TEXTWORK asks for the number of reports to be produced during the simulation. For example, if the simulation run length is one second and reports are desired every quarter second, three periodic reports should be requested.

```
Do you want evenly spaced periodic reports? (Y/N/ABORT)
> y

How many periodic reports do you want?
> 4
```

Otherwise, TEXTWORK asks when the reports should first be printed, the time interval for succeeding sets of reports, and the time for the last set of reports to be printed.

```
Do you want evenly spaced periodic reports? (Y/N/ABORT)
> n

When do you want the first report? (IN SECONDS)
> 100

What interval do you want between reports? (IN SECONDS)
> 100

When do you want the last report? (IN SECONDS)
> 900
```

Report List

After the scheduling method for Periodic Reports has been determined, TEXTWORK will prompt you for the specific list of Periodic Reports to print.

```
Which report do you want?  (? for choices, LIST for current selections)
> ?

Choices:
  Processing Element ("P")   Storage Device        ("SD")   Module ("M")
  Transfer Device    ("TD")  Instruction Execution ("IE")   Semaphore ("SE")
  Snapshot           ("S")   Message Statistics    ("MS")
  Message Delivery   ("MD")  Received Message      ("RM")   All
```

```
Which report do you want? (? for choices, LIST for current selections)
> p

Do you want to add another report? (Y/N/ABORT)
> n
```

TEXTWORK Run Parameters: Final Reports

You have the opportunity to choose those reports to be included in the Final Reports at the end of the simulation by specifying the Final Reports List. If you do not specify the Final Reports List, all of the available reports will be printed (PE, instruction, TD, SD, module, semaphore, message, and snapshot).

```
Do you want to specify the list of final reports? (Y/N/ABORT)
>y

Which report do you want?  (? for choices, LIST for current selections)
> LIST
> No reports are selected

Which report do you want?  (? for choices, LIST for current selections)
> ?

Choices:
  Processing Element ("P")   Storage Device        ("SD")   Module ("M")
  Transfer Device    ("TD")  Instruction Execution ("IE")   Semaphore ("SE")
  Snapshot           ("S")   Message Statistics    ("MS")
  Message Delivery   ("MD")  Received Message      ("RM")   All

Which report do you want? (? for choices, LIST for current selections)
> p

Do you want to add another report? (Y/N/ABORT)
> n
```

TEXTWORK Run Parameters: Plot Data File

The simulation Plot File (file type .PIN) is a binary file which is used by the NETWORK's post-simulation processing functions. It records simulation events as they occur. For long simulations, this file may grow quite large. Therefore, you may specify a plot file for only a portion of the simulation or you may choose to have no Plot File created during the simulation.

```
Should the plot data file cover All, Part, or None of the simulation?
(A/P/N)
> p

When do you want the plot file to start? (IN SECONDS)
> 100

When do you want the plot file to stop? (IN SECONDS)
> 900
```

If All or None is entered in reply to the first question, the Plot Start Time and Plot Stop Time are not requested because the plot interval will be the entire simulation (All) or no plot is being generated (None).

The Plot Start Time and Plot Stop Time determine the plot interval. The Plot Start Time must be greater than or equal to zero and less than or equal to the simulation Run Length. The Plot Stop Time must be greater than the Plot Start Time and less than or equal to the simulation Run Length.

TEXTWORK Run Parameters: LAN Analyzer File

The LAN Analyzer File parameter specifies the name of an External Traffic File (file type .EXT) that you wish to use in the simulation. The External Traffic File is generated by the TRAFLINK program.

If you choose to enter a LAN Analyzer file name, TEXTWORK attempts to locate the file. If the file cannot be found, TEXTWORK will inform you that it is unable to find the file and then again ask if you wish to specify a LAN Analyzer File.

```
Do you want to specify a LAN Analyzer Name? (Y/N/ABORT)
> y

Please enter the LAN Analyzer file name
> case1.ext
```

TEXTWORK Run Parameters: Runtime Warnings

Runtime Warnings notify you of potential error conditions in the simulation. Runtime Warnings may be sent to the simulation Report File and/or to your display.

```
Do you want to specify how to handle runtime errors now? (Y/N/ABORT)
> y
```

```
Would you like runtime messages directed to your Terminal, the Listing
file, Both places or Neither place? (T/L/B/N).
> b
```

By default, Runtime Warnings will be sent to both your terminal and the simulation Report File. The simulation will halt when a Runtime Warning is encountered and you will then be prompted to take a course of action. If you wish to run a simulation unattended, send Runtime Warnings only to the simulation Report File or to neither the listing file nor your display.

A complete description of the Runtime Warnings which may be encountered during a simulation is given in Chapter 9 of this manual.

TEXTWORK Run Parameters: Trace

If a narrative trace is requested, TEXTWORK writes a brief description of simulation events to your display and/or to the simulation Report File.

```
Do you wish to trace the event flow? (Y/N/ABORT)
> y
```

If you choose to trace simulation events, you must specify the names of model components to be traced, the Trace Start Time, and the Trace Stop Time. Everything in your model can be traced by selecting ALL and entire entity classes may be selected by clicking on ALL PES, ALL TDS, ALL SDS, ALL INSTRUCTIONS, and ALL MODULES. External traffic which was captured with a LAN Analyzer may be traced by selecting <External Traffic>.

```
Enter the name of a PE, Instruction, Transfer Device, Storage Device or
Module to trace.  Type LIST to display all traceable names.  (ALL, ALL
PES, ALL INSTRUCTIONS, ALL MODULES, ALL TDS, ALL SDS and <EXTERNAL
TRAFFIC> may be specified)
> all

Do you want the trace to be printed on your SCREEN, on the LISTING FILE,
or BOTH? (T/L/B)
> t
```

The Trace Start Time and Trace Stop Time determine the trace interval which is the period during the simulation that trace messages will be generated. The Trace Start Time must be greater than or equal to zero and less than or equal to the simulation Run Length.

The Trace Stop Time must be greater than the Trace Start Time and less than or equal to the simulation Run Length.

```
Enter time to start trace (IN SECONDS) or All to trace
entire simulation.
> 0

Do you want the trace to stop at the end of every page? (Y/N/ABORT)
> y

Simulation begins.
```

4.3.4 Runtime Reports

Trace Report

The trace report provides a brief description of simulation events. An example of a trace report is given below.

After a page of trace messages is printed on your display, TEXTWORK offers you the option to continue the trace (C), resume the trace at a later time (R), halt the trace (H), print a Snapshot Report (S), print a summary report (O), or stop the simulation (A). The number of lines displayed may be adjusted by editing the terminal height field in the N25DEV.CFG File.

```
At   233.142163 SEC module REQUEST "DATA" FILE (8) can now proceed to
execute as pe COMPUTER A is now available.

>    233.142163 SEC pe COMPUTER A will attempt to execute instruction
ASK FOR "DATA" FILE from module REQUEST "DATA" FILE (8).

>    233.142163 SEC message instruction ASK FOR "DATA" FILE from module
REQUEST "DATA" FILE (8) sending message ASK FOR "DATA" FILE (8) 256 bits
to pe COMPUTER B is beginning to execute on pe COMPUTER A.

>    233.142163 SEC td BUS 1 has been allocated to pe COMPUTER A.

Continue/Resume/Halt trace, Snapshot/Other report, Abort?  (C/R/H/S/O/A)
> s
```

Continue (C)
The simulation will proceed and a new page of simulation events are displayed when the Continue command is selected.

Resume (R)
The Resume command puts the simulation trace into sleep mode. TEXTWORK will ask you for the wake up time when the trace shall be resumed. The simulation proceeds without producing any trace messages until the simulation clock reaches the wake up time. The trace is then resumed at this time.

```
Enter time to resume trace.  (IN SECONDS)
> 50

Do you want the trace to stop at the end of every page?
(Y/N/ABORT)
> n

Enter the time to stop trace. (IN SECONDS)
> 60
```

Halt (H)
The Halt command discontinues the simulation trace to both the display and the simulation Report File. Once the Halt command has been executed, it will not be possible to start a trace again during the current simulation.

Snapshot Report (S)
Selecting the Snapshot command will generate a Snapshot Report. An example of a snapshot report is given below.

```
                    S N A P S H O T   R E P O R T
                       AT     23.142163 SECONDS

pe COMPUTER A is BUSY executing instruction ASK FOR "DATA" FILE
    resident module is REQUEST "DATA" FILE (1)
pe COMPUTER B is idle

td BUS is IDLE

module REQUEST "DATA" FILE (1)
    instruction ASK FOR "DATA" FILE iteration  1 of 1
    start time =   23.142163 SEC  host pe = COMPUTER A
```

Summary Report (O)

A summary report containing simulation statistics may be generated by entering the Summary Report command. The available reports include Processing, Instruction, Transfer Device, Storage Device, Module, Semaphore, Message Statistics, Received Message, Message Delivery and Snapshot. A sample of a summary report is given below. Chapter 5 provides a full description of the report contents.

```
                    COMPLETED MODULE STATISTICS

                  FROM     10. TO   500. SECONDS
               (ALL TIMES REPORTED IN MICROSECONDS)

MODULE NAME               REQUEST "DATA"   TRANSMIT "DATA"
                               FILE             FILE

HOST PE                     COMPUTER A       COMPUTER B

COMPLETED EXECUTIONS           16               16

CANCELLATIONS DUE TO
 CONCURRENT LIMIT               0                0
 RUN UNTIL SEMAPHORES           0                0

NUM PRECONDITION TIME          16               16
 AVG PRECONDITION TIME          0.               0.
 MAX PRECONDITION TIME          0.               0.
 MIN PRECONDITION TIME          0.               0.
 STD DEV TIME                   0.               0.

AVG EXECUTION TIME        244069.000       3900106.000
 MAX EXECUTION TIME       244069.000       3900106.000
 MIN EXECUTION TIME       244069.000       3900106.000
 STD DEV TIME                   0.               0.

RESTARTED INTERRUPTS           0                0
 AVG INTERRUPTED TIME          0.               0.
 MAX INTERRUPTED TIME          0.               0.
 STD DEV TIME                  0.               0.
```

Stop the Simulation (A)

The simulation may be terminated by issuing the Abort command. TEXTWORK will confirm that you really wish to end the program before terminating. The Simulation Report File (.LIS) and the Simulation Plot File (.PIN) are produced and will contains statistics and events from the start of the simulation or the Reset Time up to the time the simulation was canceled.

```
Continue/Resume/Halt trace, Snapshot/Other report, Abort?  (C/R/H/S/O/A)
> a

Do you really want to Abort now? (Y/N)
> y
```

4.3.5 Runtime Warnings

Runtime Warnings notify you that a potential error has occurred in the simulation. If a runtime warning occurs and you have elected to send runtime warnings to your screen, the warning message will be displayed immediately. You will then have the opportunity to generate reports and to direct the destination of future runtime warnings. A complete list of runtime warnings is included in Chapter 9.

```
---WARNING--- At 0.00308989 SEC module TEST has failed the iteration
concurrent limit test.

A runtime warning has been encountered.  Would you like to stop at every
runtime message for a chance to interact with NETWORK II.5? (Y/N/ABORT)
> y
```

Future Messages

TEXTWORK will allow you to direct Runtime Warnings and future messages to the display, listing file, both or neither to enable/disable the notification of future runtime warning messages. If the same type of runtime warnings are being generated frequently, these options allow you to speed up the simulation execution and reduce the size of the Simulation Report File (.LIS) by having runtime warning messages directed to neither destination.

```
Would you like runtime messages directed to your Terminal, the Listing
file, Both places or Neither place? (T/L/B/N)
> t
```

Simulation Control

When a runtime warning is generated, you are given the option to stop the simulation (A), print a snapshot report (S), print a summary report (O), or continue the simulation (C).

```
Do you want to Abort, print Snapshot/Other report or Continue?
(A/S/O/C).
```

```
> o

Which Report do you want?  (? for choices, LIST for current selections)
> ?

Choices:
Processing Element ("P")  Module     ("M")  Storage Device   ("SD")
Transfer Device   ("TD")  Semaphore ("SE")  Instruction Execution ("IE")
Snapshot           ("S")  Message   ("ME")  All

Which Report do you want?  (? for choices, LIST for current selections)
```

4.3.6 The Simulation Completed Message

```
Current Time is 1000.000000 SECONDS
Simulation complete.  No runtime warning or error messages were issued.
This simulation took 2 seconds to execute.
```

At the end of a TEXTWORK simulation, the simulation completed message is displayed. This message includes the actual amount of real time elapsed during the simulation and the number of runtime warnings encountered.

4.4 HOW LONG TO RUN A NETWORK II.5 SIMULATION

One of the more difficult decisions to make when conducting a simulation is the length of time to simulate. The longer the simulation is run, the greater the confidence you may have in the answers, and the higher the cost in computer time. Therefore the length of the simulation run can be a tradeoff between confidence and cost. Choosing simulation lengths is an art, and this section simply presents qualitative concepts that you may find useful.

In many simulations, there is an underlying time basis for scheduling most of the workload. Modules may be iterating in terms of seconds, microseconds, etc. If the longest iteration period for any module is T, then running the simulation for 10 times T might be a good starting point. As a minimum, a simulation length that will allow all modules to run at least once should probably be chosen.

An important reason for using NETWORK II.5 is that it simulates the interaction between various hardware and software elements. An ideal simulation would run until every possible combination of interactions has occurred. Unfortunately, in a complex

system this could take days, weeks, or years. One approach is to use the periodic reports to help determine the run length. By running the simulation with periodic reports, you may determine if the simulation statistics approach either a constant value or follow a pattern. When the difference between the reports is insignificant or when there appears to be a pattern to the results, it probably is time to stop the simulation.

Another technique is to explore a worst case scenario. Sometimes, the main concern driving the simulation is how well the simulated system will perform under stress. The scenario for a failure or some other worst case condition may be fairly well defined, with a definite beginning and end. This would make choosing a simulation length considerably easier.

The choice of simulation run length should influence the simulation design. If a module executes at a very low rate (e.g., once a year), you should either restrict the time period for that event so that it occurs within the projected simulation run length, or ignore the module if it has an insignificant impact on the simulation. This will eliminate the need to run a simulation for an excessive length of time just to see what happens when this module executes.

4.5 .i.EFFICIENCY CONSIDERATIONS

NETWORK II.5 is a very efficient program. However, some users have limited computer resources, while others have enormous problems to solve. This section should help these users minimize execution costs and decrease run times.

Because NETWORK II.5 is a commercial product, a number of runtime optimization features are already incorporated into the package. They are, however, invisible to you and are automatically invoked. The main way that you can reduce the CPU time needed to execute a NETWORK II.5 simulation is to limit the amount of work to be done. Reducing the amount of work performed does not necessarily diminish the quality of the simulation.

Run Length

One major efficiency consideration is the simulation run length. Running a simulation twice as long as necessary will cost nearly twice as much and will not improve the results.

Number of Instructions

Another major efficiency consideration is the number of instructions executed. The amount of work required to set up an instruction and execute it is approximately fixed (within an instruction type). Having a message instruction send 600 bits takes NETWORK II.5 approximately the same amount of CPU time as sending 6 bits. Thus, it pays to combine instructions when there is no (significant) impact on the simulation results. Make one 400 cycle processing instruction instead of 100 4 cycle processing instructions. The NETWORK II.5 CPU time to execute this workload will go down by a factor of 100!

Extreme care also must be employed in the use of Macro Instructions. Used properly, they contribute to an easy-to-read module definition, and they save time when specifying a module. However, they can also be deceptive, in that each Macro Instruction is mapped into two or more real instructions. In addition, Macro Instructions can be made up of Macro Instructions. Therefore, what looks like a single instruction in a module definition may turn out, in reality, to expand to 100 or 1000 actual instructions to execute.

Instruction Type

Different types of instructions take different amounts of NETWORK II.5 CPU time to execute. The hierarchy of instruction CPU execution times is:

Highest	>	Read/Write Instruction
		Message Instruction
		Semaphore Instruction
Lowest	>	Processing Instruction

Unfortunately, instruction type choices are generally dictated by the job that needs to be done. However, when designing a simulation, execution time considerations may help a you decide the instruction type to choose when there is a choice. In general terms, a Processing Instruction takes half the CPU time of a Semaphore Instruction , and a Semaphore Instruction takes half the time of either a Message or a Read/Write Instruction. Therefore, using a semaphore instead of a message to synchronize two modules could result in a modest saving at runtime.

Preconditions

Combining modules will cut down on the simulation execution time because it will reduce the amount of precondition checking NETWORK II.5 will be required to perform. It takes approximately the same amount of CPU time to set up a module with 6 instructions in its instruction list as it takes to set up a module with a single instruction (exclusive of the actual instruction execution times). This is because NETWORK II.5 spends a considerable portion of the CPU time required to set up a module checking the module's preconditions.

Level of Detail

Make sure that your model provides the proper level of detail. If a function is of no real importance to a simulation, leave it out! Don't model every PE at the micro instruction level if you only need that much detail on one PE. Don't add more preconditions than are absolutely necessary. Sometimes, once a function is modeled in very fine detail, conclusions can be drawn based on simulation results that allow remodeling that function at a much higher level in future simulation runs, at a significant saving in CPU time.

4.6 Summary

The techniques most likely to produce a payoff in reduced execution time are reducing the simulation run length and aggregating instructions and modules. Cutting the NETWORK II.5 simulation time in half by going through an existing simulation and combining instructions and modules is sometimes possible.