DATE: August 16, 1984

TO: Prime RD&E Personnel

FROM: Clarise E. Patton

SUBJECT: Bronto (New PNC)

REFERENCE: RingNET, PNC, Bronto

KEYWORDS: RingNET, PNC

## ABSTRACT

This document will describe the plans for the Software Support for a modified PrimeNET Node Controller (PNC), which will be referred to as PNC Booster. The full project is known as Bronto. There is a description of the current RingNET operations in Appendix B which is recommended for anyone who is unfamiliar with the product.

Table of Contents

# 1 Introduction to PNC Booster

There will be a modified PNC board developed to enhance the RingNET product for the near term future (1984, 1985). The new board is an Intel 80186 based network controller with 8K words of PROM and 256K words of RAM. When it is initialized, the board will run from the code in the PROM. This document describes how the PNC Booster will operate from a PRIMOS software point of view and the software modifications needed to support the new functionality.

## 2 Goals

The goals of the Bronto project are:

To eliminate receive congestion bottlenecks;

To gather additional statistics which will enable the location of soft failures in the ring;

To do transmit timeout and token recovery in a fast and consistent basis from the PNC Booster.

## 3 Justification

### Receive Congestion

Various types of receive congestions have caused throughput degradation in the ring. Whenever the PNC is unable to accept a packet, it sets the WACK (Wait Acknowledge) bit in the ACK Byte of the packet. The statistics program has shown this to run as high as 50 percent of all transmits to a busy node. The assumed causes of this congestion have been: the single buffer for receive on the PNC board; the time delay in assigning a new buffer to the PNC; and the delay in freeing buffers for reuse by the PNCDIM.

The enhancement project is attacking this in the following manner. The PNC Booster will have multiple receive buffers. It will be able to receive in advance of a buffer assignment. The use of ring buffers for broadcast node id packets will be eliminated.

### Soft Fail

There is currently no information available to enable the network administrator to locate a failure in the ring which does not totally disable the ring. If the signal is being degraded, but making it around the ring, it can be almost impossible to locate the failure point. In order to make it possible to locate the failure, it is necessary to collect information which was not previously available. The enhancement project is attacking this by having the PNC Booster generate the information necessary. This knowledge is the information as to which PNC in the ring first noted the degraded data and set the NAK bit in the ACK BYTE.

### Timing on the Board

Transmit timeout and token recovery have been done by using software timers with a 1/10 second granularity and no knowledge of what has been happening on the ring. This has caused these errors to take longer to detect than is necessary, and to make the transmit timeout an ambiguous situation (there are two different kinds). The enhancement project will have the timing done by the board in a consistent manner.

# 4 Modes

## 4.1 Emulation Mode

The PNC Booster will be able to operate in 'unmodified' mode which will totally emulate the PNC. This will be the default mode. When the board is initialized it will be in 'unmodified' mode until it is told to go into 'booster' mode.

While it is in emulation mode the PNC Booster will be totally compatible with the software now running in Primos (rev 19.3). The current software will not be able to differentiate the PNC and the PNC Booster.

There will be three transparent differences in the interaction of the PNC Booster with Primos even in emulation mode.

1) Network Status Word Additions

   There are currently three unused bits in the Network Status Register (3,4,5). Bit 3 is currently held low and the other two are undefined. Since bit 3 is a known quantity, this will be used to indicate that the board is a PNC Booster.

   Bit 3 will be set to a 1 if the board is a PNC Booster. Bit 4 will then have significance.
   If bit 4 is a 0 the PNC Booster is emulating unmodified mode.
   If bit 4 is a 1 the PNC Booster is running in booster mode.
   (Bit 5 wil have significance in booster mode. It will be a 1 if the board has failed self verify. After getting through that initialization, it will be a 1 if the board has received an invalid command and cannot continue.)

2) Burst mode DMA will be used when applicable.

   This can be done without the intervention or knowledge of the software. This will allow us to speed up the DMA transfers even while in emulation mode.

3) PNC Booster will receive up to 4 data packets.

   The unmodified PNC can accept a packet only after having been assigned a DMA channel number with which to move it into Primos memory. Once the packet has been accepted, the PNC WACKs all other packets addressed to it until it has moved the previous packet into Primos memory and the PNCDIM has assigned it another DMA channel number (actually the same one, reused).

   The PNC Booster may receive up to four packets in advance of the assignement of a DMA Channel number. This will allow the PNC Booster to receive a new packet while it is DMAing a previous

packet into Primos and while Primos is handling the buffer.

This will increase receive throughput in two ways. First, four consecutive packets for this controller can be accepted, although the subsequent ones will be WACKed until the first one is processed in Primos. This will smooth sporatic bursts of traffic. Second, the reception of a packet will be performed concurrently with the DMA transfer and Primos processing of the previous packet.

## 4.2 Default Booster Mode

Unless specifically enabled, most of the new capabilities will not be utilized. There is a subset of capabilities which are default when the PNC Booster is put into booster mode.

Token Timeout and Recovery by the PNC Booster

WACK transmit retry by the PNC Booster

Transmit timeout by the PNC Booster

## 4.3 Full Booster Mode

There are additional capabilities which will be utilized only when specifically enabled by software.

DMT transfer of Status and Counter information

DMT transfer of Node Id Packet into circular buffer

DMT transfer of regular data packets

## 5 Mode Control

### Emulation Mode

The PNC Booster will remain in emulation mode unless specifically told to activate its new capabilities. It will ignore any of the commands which are specific to booster mode except the command to go into booster mode. An OCP 6 (currently unused) will be used to tell the PNC Booster to go into booster mode. This will be issued by the software only if bit 3 of the Network Status Register is a 1. If it is issued erroneously, it is a command which is ignored by the current PNC.

### Booster Mode

Once the PNC Booster receives the command to go into booster mode, it will do self-verify testing. When it has past its internal diagnostics, it will set bit 4 of the Network Status Register to a 1. It is now able to accept all commands, including those specific to the modified features. Without having received any specific commands, the PNC Booster will operate in default booster mode. It will do token recovery and transmit retry for WACKs (see more details below). All other transmits and receives will continue as before until more of the new capabilities are specifically enabled.

If the PNC Booster fails self-verification, it will set bit 5 of the Network Status Register to a 1. The PNC Booster will also stop accepting OTA's. Anytime a 'fatal' error has occurred, the PNC Booster will stop accepting OTA's. It will not stop accepting INA's. If an INA command does not skip, the board is totally dead.

There will not be a 'return to emulation mode' command designed. If networks are stoped and then restarted the OCP Initialize (17) which is part of the initialize code will cause the board to come back up in the default state of emulation mode.

# 6 New Features

## 6.1 Burst Mode DMA

This is an automatic feature which will be present whether the PNC Booster is in unmodified mode or booster mode. It is transparent to the software and does not have to be taken into account in the software design.

## 6.2 Counters and Status

The PNC Booster will maintain the following counters:

1) # packets WACKed by this board

1A) Maximum # packets WACKed in a row

2) # packets with NAK set by this board

3) # packets with NAK seen by this board

4) # tokens seen by this board (in thousands)

5) # tokens inserted by this board.

The PNC Booster will be maintaining these counts while in unmodified mode. The counts are cumulative and will be cleared only by an OCP Initialize (17) or an OCP Clear Counters (OCP 5).

The WACK counter will allow us to determine if there is receive congestion, and how often it has occurred. This counter will be incremented each time the board is forced to WACK a packet. For the PNC Booster this will be when the board has received all the packets allowed (4), and until the PNC Booster has transfered one of the packets into Primos memory and been given a new DMA channel. (On the current board the time from the issuing of a Receive Interrupt until the issuing of an OTA Receive Channel varies from .5 milliseconds to 1.7 milliseconds.) The number of sequential WACKs will give us an idea of the maximum amount of time we remain unable to receive.

The NAK counters will enable us to determine the probable location of a soft failure on the ring. A soft failure is one in which the data is degraded but not totally lost. If this PNC Booster is the first node to note that the data is bad, then the failure lies between this node and the previous active node.

The board will maintain a status word with the following information:
transmit retry count
node id
Flag if ring is down
TBS

The PNC Booster will move the counters and status information into Primos memory once a per second after having been told to do so (see DMT Transfers for Special Information).

## 6.3 Retransmit Capability

There will be two types of transmit retrying done directly from the board. The first (WACK Retry) will be a default booster option, and will be enabled automatically when the booster mode is entered. The second (Retry Command) will be done only upon a specific command from Primos.

### 6.3.1 WACK Retransmission

When a packet whose destination node is specific (not broadcast) has a WACK as its only error condition, the PNC Booster will retransmit the packet automatically. This will be done N times. N will be a default of 10. This number can be changed by a OTA 0 command any time after booster mode has been enabled. The packet will be retransmitted 500 microseconds after the transmission of the previous packet was initiated. (On a maximum sized ring, the round trip delay would be slightly under 500 microseconds.) On a PNC it would take 1 to 2 milliseconds to process a buffer and be ready for the next. On a PNC Booster it will take about 350 microseconds to do so. In either case, the 10 times retry should succeed if it is normal congestion.

o If the packet is WACKed 10 times, the PNC Booster will interrupt Primos with a transmit interrupt. The transmit status word will show that the attempt failed (WACK bit set) and the count will show 10.

o If the packet is transmitted successfully after fewer than 10 tries, the PNC Booster will issue a transmit interrupt. The transmit status word will show success, and the count will show how many times it had previously been WACKed.

o If the packet has an error other than a WACK, the PNC Booster will issue a transmit interrupt. The transmit status word will show the type of error encountered, and the count will show the number of previous WACKs.

### 6.3.2 Retransmit Command

If Primos is informed of an error on a transmitted packet and wishes to have the packet resent, it will issue the Retry Transmit Command (an OCP 7 - currently unused). This will tell the PNC Booster to retransmit the packet which is already in its local buffer. This will mean than the transmit buffer does not need to be set up and moved over the I/O bus again.

## 6.4 Board Timers

The PNC has no timers on the board and all timing is done in the software in Primos. When Primos gives a transmit buffer to the PNC, the software sets a timer for 6/10ths of a second. This is a time longer than the delay before the token is seen if the maximum number of nodes were sending the maximum sized packet. If the PNC has not responded with a transmit complete within this time, the software checks to see if the PNC has seen a token (this would imply that it had sent or was now sending the transmit packet). If the PNC has seen a token, the software waits another 1/10th of a second for the transmit complete interrupt (this is the smallest granularity of software timing). If no transmit interrupt has been seen, the 'Packet did not Return' is assumed, and the transmit attempt is stopped.

If the token has not been seen at the end of the 6/10ths second period, the software goes into a staggered timing algorith. It multiplies the low 3 bits of the node id by 1/10 of a second. If a token still has not been seen, it tells the PNC to issue a token. (Appendix B has a detailed description of how the current software works.)

The timing done by software is limited in a couple of respects. First the granularity is much larger than called for by the speed of the ring. Second, the software (and the PNC) has no way to differentiate a ring with data packets going by, and an idle ring. The timeout for the two situations cannot be separated, so the longer time must be used.

The PNC Booster will do transmit timing on the board. The board will know:

1) if the ring has activity or is idle (whether there are data packets going by);

2) if the token has been seen within maximum rotation time of the ring (taking into account the data traffic if any);

3) if the transmitted packet has returned within maximum rotation time of the ring.

### 6.4.1 Transmit Timeout

If a packet is sent, and is not seen by the PNC Booster within the expected time period (500 microseconds), the PNC Booster will issue a transmit interrupt with the Packet Did Not Return Error set and go into token recovery. This will also occur if a token is seen within the time period during which the PNC Booster is waiting for the packet to return.

Currently the Packet did not Return error is set only if a token is seen instead of the packet. If no token is seen, the error is caught

by a timeout in the Primos code. The PNC Booster will be aware of the error much more quickly than Primos could be since it is aware exactly when it put the packet out on the ring.

These two errors have an identical cause - the packet sent out did not return. If a token is seen, it is called Packet Did Not Return. If a token is not seen, it is called Packet Timed Out. These two are currently handled differently, which can delay the realization by the software that the ring is broken. These will be merged into one error - Packet Did Not Return. If this error is encountered, the PNC Booster will be told to retry the transmit (OCP 7). If it fails up to 5 times in a row, the packet will be aborted with the PKT_TMT (Packet Timed Out) error. This will be the indication to RNGRCV that the ring is probably broken.

### 6.4.2 Token Recovery

The PNC Booster will be able to do token recovery much more quickly than it was done by the software. If the token has not been seen within the maximum time (500 microseconds), the PNC Booster will back off using the complete node-id number as a stagger factor. The node-d will be multiplied by 1 millisecond. If the PNC Booster has not seen the token within the back-off period, it will issue a token. If that token does not come back, it will issue another one at the same stagger period.

At the time that the second token is issued the PNC Booster will internally mark that the ring is down. This will be one of pieces of status information which will be passed into Primos every second.

The PNC Booster will keep a count of the number of times it has inserted the token. This will also be passed in with the counters and status information. This count is currently kept in software, since the software tells the PNC when to issue a token.

The PNC Booster will not discard the pending transmit packet when it goes into token recovery. The only thing which will cause it to discard the packet would be the Stop Transmission command from Primos. This will allow us to do regular token recovery without having to kill and reissue the transmit.

## 6.5 DMT Transfers for Special Information

The PNC Booster will be able to do DMT transfers at approximately the same speed as normal mode DMA. Initially DMT will be used only to transfer the specific information described below from the PNC Booster to Primos memory.

### 6.5.1 Status and Counters

The PNC Booster will move a set of counters and status information into Primos memory once a second using a DMT transfer. This ability will be initialized by the receipt of an OTA 0 function call with the start address and length of the data area into which it is to be moved. Each second this area will be overlaid with new information.

The format of this buffer will be as follows:

Word 1,2 - # of packets WACKed by this board

Word 3 - Maximum # of packets WACKed in a row

Word 4 - # of packets NAKed by this board (NAKs Sent)

Word 5 - # of NAKed packets seen by this board (NAKs Seen)

Word 6 - # of tokens inserted

Word 7,8 - # of tokens seen (in thousands)

Word 9 - TBS

Word 10 - Status word defined as follows:

Bits 14-16 - Transmit retry count

Bits 6-13 - Node Id

Bit 5 - Ring is Down Flag (1 = down)

Bits 1-5 - TBS

### 6.5.2 Node-Id Packets

Primos Rev 19.3 and later send the node identification packet as a broadcast packet. The earlier revs sent individual packets to each node in the network configuration. This node identification packet is used to inform the other nodes that this node has entered the ring, or remains in the ring. This packet is only five words long, but each packet is transfered into Primos using a ring buffer assigned for

receive.

The PNC Booster will be able to accept another OTA 0 function call for node-id DMT transfers. This call will contain the start address and length of a buffer in Primos memory to be used in a circular manner for broadcast node ids. Once this function call has been accepted by the PNC Booster, it will differentiate broadcast node identification packets from other received packets. All broadcast node id packets will be transfered into this circular buffer in Primos memory.

No other action will be taken on this receive. This means that the PNC Booster will not issue a receive interrupt. The PNC Booster will not attempt any checking to confirm that data has been removed from the buffer, it will use the buffer as an endless circular buffer. If the node id packet is not a broadcast packet, it is sent via the regular data transfer like any other receive packet.

The RNGRCV module will check the DMT buffer after it has flushed the receive queue for the DIM. In order to protect against a situation where nothing but node-id packets are being received, the RNGRCV module will be called every second based on a timer. The RNGRCV module will process the node-id packet, use it to modify the line definition, and then zero the buffer locations so that it will know that they have been processed. The RNGRCV module will maintain a read pointer in order to know where it is in this buffer.

This feature will avoid tying the ring buffers up for node id packets, and will avoid having the DIM handle these packets at all.

## 6.6 Multiple Packet Receives

The PNC Booster will be able to accept four buffers in advance of the DMA channel. In booster mode, the PNCDIM will be able to issue multiple receive DMA channel numbers to the PNC Booster. These will be used in a FIFO ordering. In order to be certain, the PNCDIM will do an INA Receive channel to pick up the channel number for the current buffer to be handled.

The PNC Booster will be able to receive 4 back to back packets before it will need to WACK an incoming packet.

If DMT data packet transfers are being used (see Futures), the INA channel number will pass the offset address of the end of the data back in the A register. This should be as specific for DMTs as the channel address is for DMAs.

The initial receive buffer size will be 1024 words. This is compatible with unmodified PNCs. The ring buffers are currently allocated with the specification that they cannot cross a page boundary. Once the software design is able to efficiently handle larger buffers, the buffer size can be increased. This would be done using an OTA 0 special function command before any receive channels are assigned.

# 7 Futures

Some features may not be implemented in the initial software support. These will have the necessary hooks in the PROM on the board, so that when we do want to take advantage of them they will be available.

## 7.1 Down Line Load

If the board has received an OCP 11 - Go into Diagnostic Mode, it will accept the ROIPQNM Down Line Load command. It will follow the format for down line load files already established for ICS1 and ICS2. This will allow us to update the software on the board in conjuntion with software changes in Primos. When it receives the final data transfer which contains the command to 'transfer to this address' it will begin to process from the down line loaded code instead of the PROM. At this point it will be interfacing into Primos via the ROIPQNM interface as an intelligent controller.

## 7.2 Up Line Dump

If the board is believed to be malfunctioning, it will be sent an OCP 11 - go into diagnostic mode. This will cause it to begin executing from some PROM code. It will then begin watching for specific commands. If the board receives an OTA 0 function code telling it to do an Up Line Dump, it will know that the information to follow will be the beginning address and count for a DMT transfer for pre-arranged information.

## 7.3 DMT Data Transfers

The PNC Booster will initially be set up to do data transfers with DMAs, since it is necessary to do it this way for unmodified mode. Once the board is in booster mode, it can be told to begin using DMT transfers instead of DMA. This would be done by an OTA 0 function code.

Once it had been told to go to DMT transfers, the board would ignore any DMA channel commands.

There are some restrictions for transmit buffers to be passed by DMT. All DMT windows for transmit and receive packets must be from the same segment (either 0 or 1). This does not mean that the special information DMT window needs to be from the same segment.

The transmit buffer must be either < 16 words or modulo 16 words. This size restriction is on the buffer size, and not on the data size within the buffer. This restriction is required by the method the PNC Booster

will use for DMT transfers. It will either specify a length less than 16 words to the hardware handling the transfer, or it will be given transfers of 16 words apiece. In order to prevent the hardware from accessing past the boundary of the buffer and potentially crossing a page boundary (causing a page fault), the buffer must be allocated modulo 16. Currently, the only sizes allowed are 256, 512, and 1024, so no problem is anticipated.

On DMT receives, when the PNCDIM does an INA Receive Channel it will be given the offset address of the end of the data within the buffer. This will allow it to verify that the correct buffer is being used, and also give it the length of the receive data.

## 8 Dependencies

There will be few dependencies for this modfied PNCDIM. Most of the dependencies would be similar to those for the multiple PNC project. This modified device must be considered to be able to be used in conjunction with the unmodified PNC.

### 8.1 Device Addresses

Additional device addresses need to be assigned to the PNCs and PNC Boosters.

### 8.2 I/O Windows

In order to take full advantage of the receive enhancements, we will need to have additional segment 0 or 1 windows available. We would use four windows for receive, one for transmit, and one for special information transfer.

At this time the PNCDIM allocates 4 windows, 2 for transmit and 2 for receive, This is done to protect against going over a page boundary with a buffer. Well, at this time the buffers are allocated so that they will not cross a page boundary. Until and unless we went with buffers larger than 1k words, we could take advantage of the current situation. The 4 windows could be used for 2 receives, 1 node id DMT (also counters and status area), and 1 transmit, without increasing our usage of windows.

## 9 Impact

The current PNC can handle about 400 packets a second. This apparently caused the PNCDIM to take about 20 percent of the CPU and NETMAN to take about 30 percent (this test was run on s37). If this number is similar for the PNC Booster I would not expect that there would be any perceivable impact on the rest of the system. If the throughput is faster on the modified board, the impact on the system of the increased interrupt load would need to be checked.

# 10 Software Support Summary

The current software should be able to run the PNC Booster in unmodified mode without being aware that there is a modified device on the system. This will be extensively tested.

## 10.1 Modified Functions

In order to take advantage of the modified board we will need to modify the PNCDIM and RNGRCV. This would be sufficient for what I have described as the initial support. When we went to more complete support (using ROIPQNM and putting most of level 2 on the board perhaps) the Primos modifications will be more extensive. For the initial support, the modifications are comparatively clear cut.

### 10.1.1 Initialization

The initialization code will need to check to determine which board is being run. If it is a modified board, there are additional commands which will have to be given.

### 10.1.2 Receives

If the software is running a modified board in booster mode, it will want to have a method to assign multiple buffers for receive. It will, therefore, be necessary to do an INA channel number after a receive interrupt to confirm which buffer is being used.

If the DMT window has been given to the PNC Booster for node-id packets, RNGRCV will have to handle this buffer after it has drained the receive queue for this device. It will have to be able to differentiate a PNC from a PNC Booster for this (remember there may be a mix of the two types of devices on the system).

### 10.1.3 Transmits

The error recovery for transmit will be the largest change in this area. There is no longer any transmit timeout being done by the software for the PNC Booster. The Packet Did Not Return error must be handled somewhat like the timeout used to be.

The retransmits for errors will be done with the new OCP instead of the OTA transmit channel.

The WACK retry code will need to be aware of the automatic retry from

the board.

## 10.2 New Functions

### 10.2.1 DMT Data Transfers

If DMA channels remain in short supply, we will probably want to do the transmits and receives using the DMT data transfer. This will need to be checked into more thoroughly for speed considerations. The commands to allow this will be implemented in the Booster, but will probably not be taken advantage of by the first software release.

## 10.3 Future Functions

The initial software does not need to support DLL and Up Line Dump, but the design of these does need to be done. The hooks to allow future software utilization of these abilities need to be built into the PROM.

## 10.4 The World

Once Down Line Load capabilities exist for the PNC Booster the functionality can be expanded to the limits of imagination (and the hardware).

## 11 Commands

All the existing commands will operate as they did in the PNC. There will be some new commands which will operate for the PNC Booster. Until a Set Modified Mode command has been received, the PNC Booster will ignore any new command given. The PNC Booster has an Intel 80186 on it.

### 11.1 Interrupt Structure of Booster

The CPU has 4 programmable interrupt lines. Two will be unused at the present. The existing interrupts will be:

#### 11.1.1 DMA Device EOR Interrupt

One of the DMA channels has reached an completed a transfer. By reading an eight bit wide FIFO, the channel can be identified. The FIFO also insures that these interrupts are received in the correct order. All pending EOR interrupts will be serviced until the FIFO is empty. The DMA channels are:

A. Receiver (5 channels)
B. Transmitter
C. Backplane DMx
D. Transmitted Packet Returned

#### 11.1.2 OTA,OCP Received

A PIO output instruction has been received. This interrupt handler will determine the instruction type by reading the function code register. If the instruction type has a data word (OTA), that is also read. At this point, the instruction is acted upon. Since the function code register is actually a FIFO, upon completion the FIFO must be shifted and tested for more data. All OTA's and OCP's in the FIFO will be handled immediately.

### 11.2 INAs

The INA instruction is handled by the hardware rather than by the Intel CPU. It does not get put into the FIFO.

## 11.3 New Commands

The following commands will be ignored by the PNC, but accepted and used by the PNC Booster.

### 11.3.1 Set Modified Mode (OCP 6)

If the Network Status Register shows that this is a PNC Booster (Bit 3 set), the modified software can take advantage of the newer capabilities. The PNC Booster must be in Disconnect mode. The software will issue an OCP 6 to the PNC Booster. This will cause it to go into booster mode, using the default options. These are:

retransmit WACKs 10 times;

do token recovery;

do transmit timeout;

accept booster mode commands;

accept multiple receive buffer commands.

### 11.3.2 Retransmit Buffer (OCP 7)

In booster mode, the software can streamline transmit retries by using the OCP 7 command. This will cause the transmit buffer to be resent from the controller.

### 11.3.3 New Functions (OTA 0, OTA 1, OTA 2)

All other new functions will work off of the OTA 0 command. This will have a function field in the A register which will inform the PNC Booster which of the new functions is being done. Based on the function, there may be more information to follow. The additional information is sent by OTA 1 and OTA 2 commands.

If the function requires an OTA 1 (or 2) the PNC Booster will not execute that function until the necessary following command is received. If the command is not received, and another command is seen instead, it is a 'fatal' error and the PNC Booster will set bit 5 of the Network Status Register and stop accepting OTAs. The refusal of the device to skip on an OTA has always been a 'fatal' error, and will cause us to go into error recovery.

OTA 0 - this is currently planned to indicate the following functions:

Change retry count.

If we want the PNC Booster to retry other than 10 times for a WACKed transmit packet, we can tell it any number from 0 to 15.

Change receive buffer size.

Clear Counters

This will zero the counters being held by the board. These counters are cleared only on this command or by an initialize. The OCP 6 to set Booster Mode does not reset these coutners.

Node-ID DMT. This tells the PNC Booster to begin sending the broadcast node-id packets to Primos via the DMT window circular buffer. The A register with the OTA 0 will contain the size of the buffer and the first two bits of the beginning address. The next command will be an OTA 1 with the rest of the beginning address.

Count and Status DMT. This tells the PNC Booster that it should begin sending the counters and status information to Primos once a second. OTA 1 will be used as above.

Receive DMT. This gives the PNC Booster a receive buffer DMT address. OTA 1 will be used as above. It will not contain the size, since this is specified once for all buffers.

Transmit DMT. This gives the PNC Booster a transmit buffer DMT address. OTA 1 is used as above. OTA 2 will be used to hold the count.

# 12 Appendix A Detailed Description of all Commands

This section is here for clarity and documentation. It is not necessary for an understanding of the software support for the PNC Booster. It is necessary for the coding of the software.

PNC BOOSTER PIO Instructions

| | OCP | INA | OTA |
|---|---|---|---|
| 00 | Disconnect | | Special Funcs. |
| 01 | Connect | | Reserved |
| 02 | Transmit Token | | Reserved |
| 03 | Simulate Token | | |
| 04 | Ackn Transmit Int | | |
| 05 | Add Delay | | |
| 06 | Set New Mode | | |
| 07 | Retransmit Pkt | | |
| 08 | Stop Transmission | | |
| 09 | Stop Receive | Node ID | Diagnostic Output |
| 0A | Set Normal Mode | Receive Status | |
| 0B | Set Diagnostic Mode | Transmit Status | |
| 0C | Ackn Receive Int | Receive DMx Channel | Receive DMX Channel |
| 0D | Set Int Mask | Transmit DMx Channel | Transmit DMX Channel |
| 0E | Clear Int Mask | Diagnosic Reg | Int Vector |
| 0F | Initialize | Network Status | Node Number |

## 12.1 OCP Instructions

OCP $00 - Disconnect

Receipt of this command causes the node controller to be disconnected from the network. This is done by writing to a specified port. Also, upon receipt of this command, all on board data buffers will be flushed. When in the disconnect state, the PNC will not respond to DMX Channel OTA's. The PNC will accept a Change Receive Buffer Size OTA.

OCP $01 - Connect

This command causes the PNC to become connected into the network. Physically this is done by writing to a specified port, and the PNC will now actively look at network data. However, no data can be received until an DMX Receive Channel OTA is received. Any data received after the Connect, but prior to the DMX Receive channel OTA will be WACKED, or NAKED. If a connect is received prior to reception of a NODE ID OCP, the connect will be ignored.

OCP $02 - Transmit Token

Upon receipt of this command, the PNC hardware is instructed to transmit a token.

OCP $03 - Simulate Token

This instruction is basically for debug use. It instructs the PNC to transmit a packet without waiting for a token, and to transmit a token following the packet. Note that packet collisions can be expected in this mode.

OCP $04 - Ack Transmit Interrupt

This command causes the transmit interrupt request bit in the Network Status Register to be cleared. The PNC can now accept a XMIT DMx Channel OTA.

OCP $05 - Add Delay

Add delay was intended to alleviate a problem with extremely small rings. On the PNC BOOSTER, the delay will always be enabled. Therefore, this command will be ignored.

OCP $06 - Set New Mode

The board will now accept all new mode commands, all variables and states (except for the counters) will be reinitialized.

OCP $07 - Retransmit Packet

Causes the PNC to retransmit the current transmit packet.

OCP $08 - Stop transmission

This causes the PNC to cancel transmission of the packet if the token has not yet been seen. In this case, the packet will be discarded. If the token was already seen, the transmission will proceed normally. By reading the network status register after issuing this command, one can infer what took place. If the Token Detected Status bit is asserted, the transmission of the packet has already begun (but is not neccessarily completed). If the bit is zero, then the packet was not transmitted, and has been discarded.

OCP $09 - Stop Receive

In old mode, if reception of a packet has not yet started, it may be cancelled via a Stop Rcv command. Packets received after this will be WACK'ed, until a RCV DMX Number is received. If reception of a packet has already started, it will proceed normally.

In new mode, this command is not used.

OCP $0A - Set Normal Mode

If the PNC is in diagnostic mode, it will revert to normal mode. If the PNC is already in normal mode, the command will be ignored.

OCP $0B - Set Diagnostic Mode

If the PNC is in normal mode, it will switch to diagnostic mode.

OCP $0C - Ack Receive Interrupt

Receipt of this OCP causes the receive interrupt bit in the Network Status Register to be cleared. In emulation mode this OCP is required before another DMX Receive Channel OTA is accepted. In booster mode, this will be used to indicate when it is possible to set up the receive status word and issue a receive interrupt.

OCP $0D - Set Int Mask

This OCP is a hardware function, and is not seen by the software.

OCP $0E - Clear Int Mask

This OCP is also a hardware function, and is not seen by the software.

OCP $0F - Initialize

All hardware is reset and initialized the same as at power up, except for the node ID register.

## 12.2 OTA Instructions

OTA $00 - Special Functions

Rather than trying to set aside a separate OTA for each new function, many of these functions have been placed as functions within the OTA $00 command.

These next commands use the OTA $00 by itself. They are as follows:

### Change retry count

This is used to alter the transmit retry count. The default value is 10 decimal.

OTA $00

| 1-4 | 5-13 | 14-16 |
|---|---|---|
| b'0000' | Unused | 0 - 15 |
| Function Code | | Retry Count |

### Change receive buffer size

OTA $00

| 1-4 | 5,6 | 7-16 |
|---|---|---|
| b'0001' | | Buffer Size (8K Max) (10 MSB's) |
| Function Code | | |

Note: This command will be ignored unless it is issued while the board is in 'Disconnect mode'. If not sent, the default buffer size will be used. The default buffer size is 1024 words.

### Reset Counters

OTA $00

| 1-4 | 5-16 |
|---|---|
| b'0010 | Unused |

All counters are reset.

The following commands require that a second and perhaps third word follow directly.

Periodic Status Channel

```
OTA $00
   1-4                          5-14                              15,16
*****************************************************************
*  b'1001 *                 Word Count                          * seg *
*         *                                                     *  #  *
*****************************************************************
   Function
  Code

OTA $01
           1-16
*****************************************************************
*          Buffer Address                                       *
*                                                               *
*****************************************************************
```

This is a 2 word sequence, with OTA $00 containing the command type, the word count, which is the size of the buffer reserved in the host for the periodic status information. The PROM based code has 10 words of status to send. If the word count is <10, the message will be truncated. If the word count is >10, only 10 words will be sent. In addition, the OTA $00 will contain the 2 MSB's of the starting address.

It must be followed by an OTA $01, containing the starting address of the buffer reserved in the host for the periodic status information. After reception of this sequence, the PNC will begin transmitting periodic status to the host, at 1 second intervals. Note that this message will always be put into the same place, and the earlier information will be overlaid.

Receive DMT Channel #

```
OTA $00
   1-4                          5-14                              15,16
*****************************************************************
*  b'1010' *                 Unused                              * seg  *
*          *                                                     *  #   *
*****************************************************************
   Function
   Code

OTA $01
           1-16
*****************************************************************
*          Buffer Address                                       *
*                                                               *
*****************************************************************
```

This is a two command sequence, the format being OTA $00 contains the command type and 2 MSB's of the starting address, and OTA $01 is the starting address of the buffer.

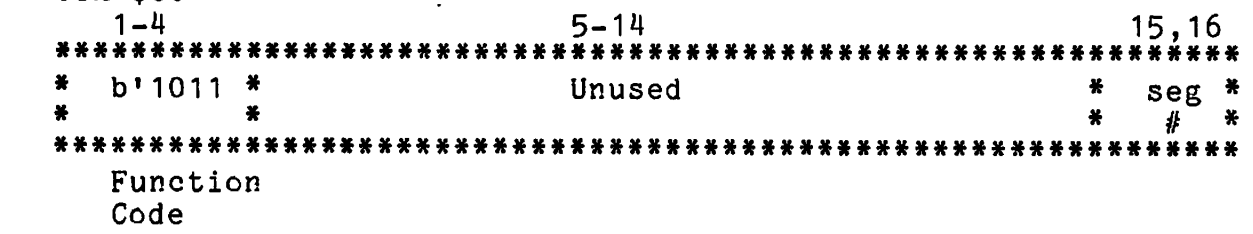
Note: RX & TX Channel buffers must all reside in the same segment. Therefore, upon receipt of the first RX or TX DMT Channel #$ command, the segment # will be stored, and any future RX or TX DMT Channel # command referencing a different segment number will be treated as a fatal error, causing the PNC to stop accepting OTA Instructions.

Note that this only applies to RX & TX DMT channels. DLL, Node ID, and Periodic status buffers may reside in different segments.

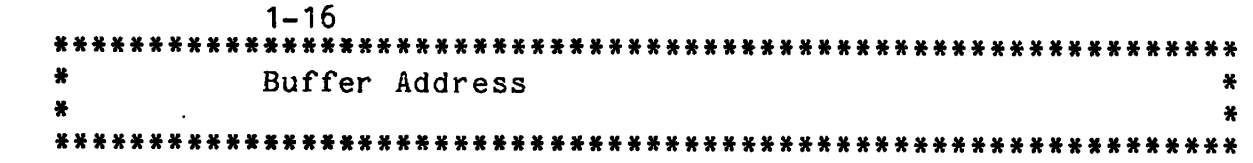
The size does not need to be specified since it is either the default of 1k, or has been specified by a separate OTA 00 call.

Transmit DMT Channel #

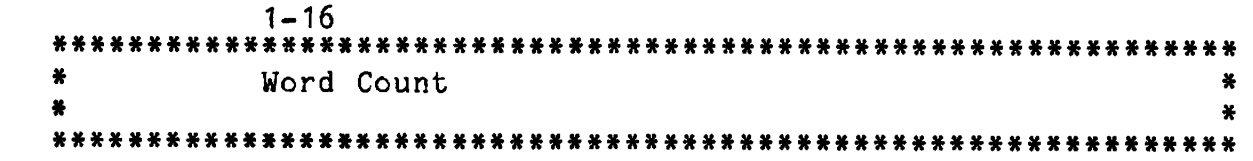
OTA $00




OTA $01




OTA $02



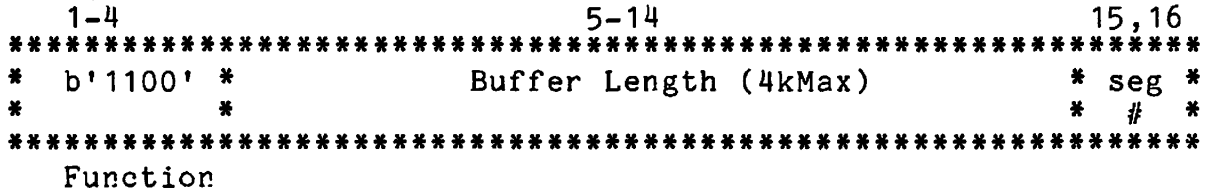

A three command sequence, the format being OTA $00 contains the command type, as well as the 2 MSB's of the buffer starting address. OTA $01 is the remainder of the starting address, and OTA $02 is the word count.

Note: RX & TX Channel #'s must all reside in the same segment. Therefore, upon receipt of the first RX or TX DMT Channel #$ command, the segment # will be stored, and any future RX or TX DMT Channel # command referencing a different segment number is a fatal error, causing the PNC to set bit 5 of the Netowrk Status Register and stop accepting OTA Instructions.

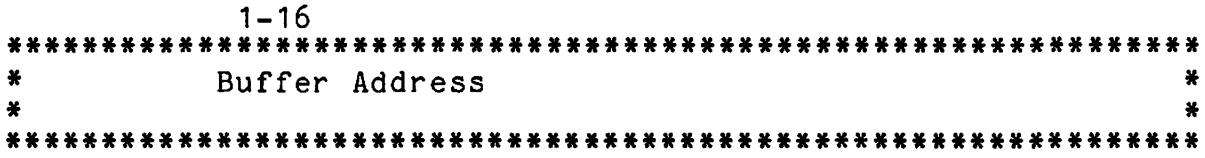
Note that this only applies to RX & TX DMT channels. Node ID, and Periodic status buffers may reside in different segments.

Node ID Channel

OTA $00




OTA $01

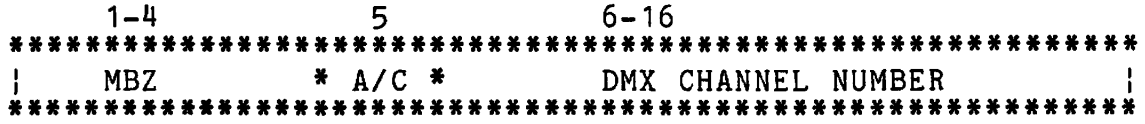



This command sets up a DMT channel for passing node ID packets to the host. Node ID Packets will be put into a circular buffer in the host, with the PNC controlling the wrap. If a packet will not fit into the remaining buffer space, the whole packet will be inserted at the top of the buffer.

OTA $0C - Receive DMx Channel

This command is used to issue a DMx channel number to the PNC. The PNC will use this channel to DMX a receive packet to the host. The format is as follows:

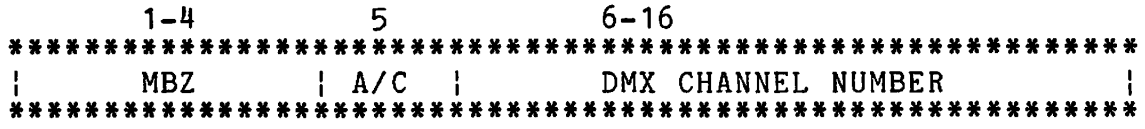
Word 2 -




Bit 5 - 0 = DMA, 1 = DMC

OTA $0D - Transmit DMx Channel

This command is used to issue a DMx channel number to the PNC. The PNC will use this channel to obtain a transmit packet from the host. This command will be accepted only if there is no other transmit buffer currently outstanding. Chaining is not implemented. The format is as follows:



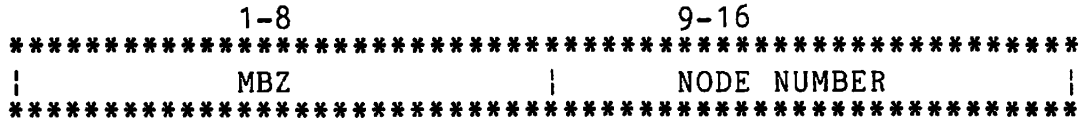

Bit 5 - 0 = DMA, 1 = DMC

OTA $0E - Interrupt Vector

A 16 bit interrupt vector is sent from the host. This vector, with the LSB set to 1 will be the transmit vector, and with the LSB set to 0 will be the receive vector. These vectors are used alert the host to reception of data, or completion of a transmission.

OTA $0F - Node Number

This command assigns a node number to the PNC. The format is as follows:

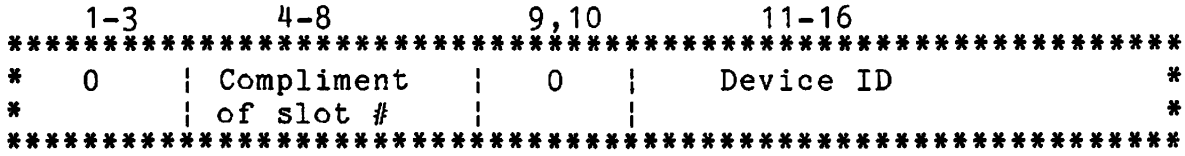



The node number is an 8 bit quantity, unique to that particular node. Node numbers range from 0 to 255 decimal. Some node numbers are reserved. Node 0 is the address that a PNC is initialized to, prior to a node number command. Node numbers 248 thru 255 are reserved, with node number 255 being the broadcast address.

## 12.3 INA Instructions

INA $09 - Input ID

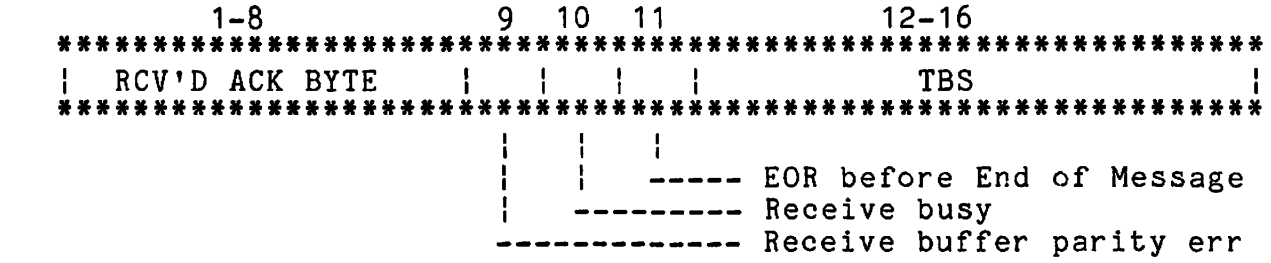
This inputs the Node ID into the A register. The format is as follows:




The Device ID is 31H for both the old and new PNC board.

INA $0A - Receive Status

The contents of the Receive Status Register are returned. This register reflects the status of the last packet received. When data is received from the network, the PNC BOOSTER interface puts it into memory. It then appends one word to this message, the receive status word. This word is then written into the Receive Status Register. It is cleared to zero's by a Master Clear, an OCP Initialize, or an OCP Ack Receive Interrupt. This INA will always respond ready. Its format is as follows:

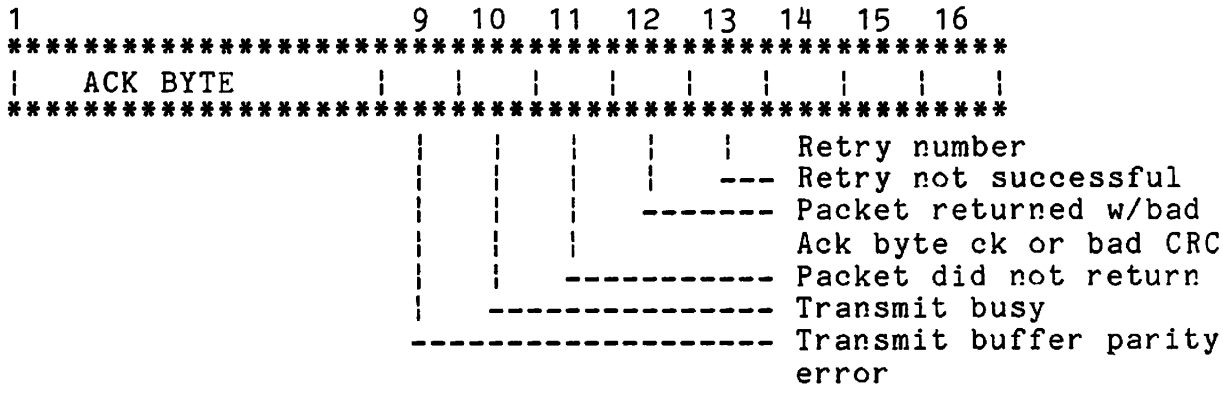



The Ack byte formatted as follows:

Bit1 - ACK'd by a previous node
Bit2 - Multiple Ack
Bit3 - WACK'd by a previous node
Bit4 - NAK'd by a previous node
Bit5 - TBS
Bit6 - TBS
Bit7 - Parity Error on Rcv'd Ack byte
Bit8 - Check Error on Rcv'd Ack byte

INA $0C - Transmit Status

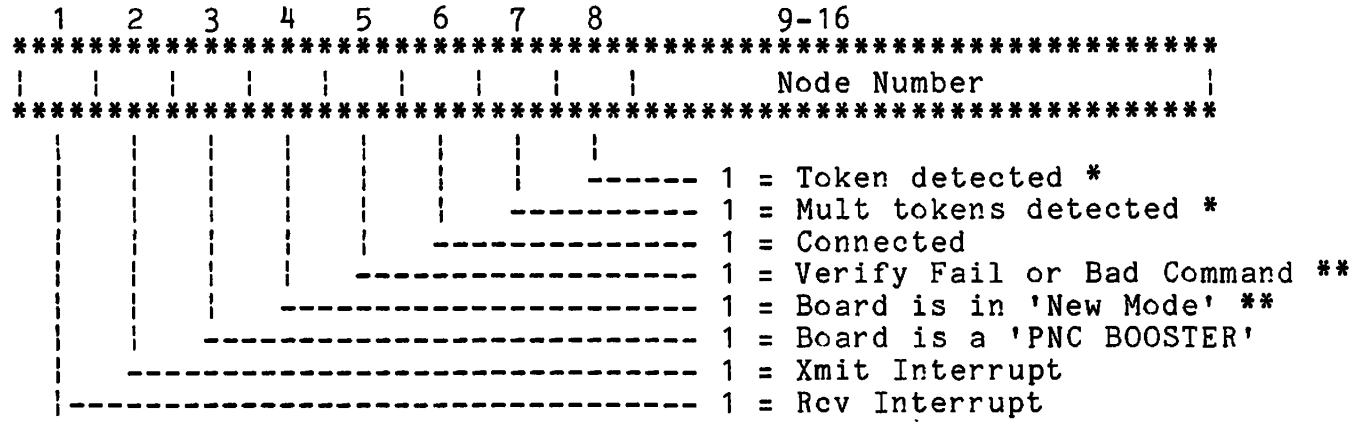
This INA moves the contents of the PNC Transmit status register to the CPU A Register. It reflects the status of the last packet transmitted. The transmit status register is cleared to zeros by a master clear, OCP $0F (initialize), or OCP $04 (ack transmit interrupt), and at power on. This INA will always respond ready. The 16 bit register is defined as follows:




Bits 1-8 - Ack Byte. (See Rcv Status INA)
Bit 9 - Transmit buffer parity error
Bit 10 - Transmit busy; The PNC BOOSTER is currently transmitting.
Bit 11 - Packet did not return; A packet which was transmitted did not come back.
Bit 12 - Packet returned with an ACK Byte Check, or bad CRC.
Bit 13 - Retry Unsuccessful; Transmission was retried the maximum number of times (see retry count) without success.
Bits 14-16 - Retry count; # of retries attempted; see bit 13 for outcome.

INA $0F - Network Status

This INA moves the contents of the PNC
Network Status Register to the CPU A Register. The network status register
is cleared to zero's on power -on, Master Clear, or reception of OTA $0F
(Initialize). This INA will always respond ready.
The register is 16 bits, formatted as follows:




* Only updated upon completion of a transmit

** Bit only valid if board = 'PNC BOOSTER'

# 13 Appendix B Introduction to RingNET

The following pages will decribe the operations of the current ring and how error conditions are now handled.

## 13.1 Token Ring Description

A token ring is a communications method pretty well defined by its own name. The medium over which the communications packets are sent is configured in a ring. Everything which is sent out will return to the sender. And the signal to a node that it is allowed to send is the receipt of a token.

A token is a special bit pattern which can be recognized by the Primenet Hardware. (Slide 1) The hardware for our ring is a twin axial cable over which we transmit at 10mgb, a junction box, and a node controller. The junction box is inserted into the ring. When it is not active, the junction box acts as if it were part of the cable itself. When the junction box is active, it takes the signals on the ring and sends them down to the Node Controller. The Node Controller does all of the work for the ring.

The node controller will usually be transceiving. This means that it receives the signals from the ring, and transmits them back out on the ring. There is a six bit time delay in this process, and the signal is physically regenerated by each PNC (Primenet Node Controller).

While the PNC is transceiving it is watching the data going by. It is checking for the presence of the special characters which will indicate data on the line. If the PNC is waiting to transmit it will be looking for the presence of the Token on the line. At all times it will be looking for the presence of a leading frame. (slide 2)

The data is transmitted and received in packets. A packet (slide 3) will consist of the packet protocol and data - supplied by the higher level of software, the ringnet header - supplied by level 2 software, and the ring protocol words supplied by the PNC. A packet can be 256, 512 or 1024 words in length. This length currently includes the header overhead.

The leading frame character alerts the PNC that it is seeing a data packet. The PNC will check the To node address to see if the packet is being sent to this node. The PNC will verify the CRC (Circular Redundancy Check) on the data. It will make any necessary modifications to the ACK Byte, and will transceive the packet on the ring.

## 13.2 Normal Transmit and Receive

If we take a five node ring (slide 4), we can watch a series of normal operations on the ring. Node E is physically connected to the ring, but it is powered down and the Junction Box is inactive. It will not effect or react to anything on the ring.

In the first slide there is a token on the ring by itself. This 12 bit pattern is unique and cannot be duplicated by data. If no one has anything to transmit, this token will continuously circle the ring. In the slide, Node A has a packet it wants to transmit to Node B. Node B is to its left, or upstream of it. The only way that a packet can be sent to Node A is to send it around the ring. Traffic on the ring is counter-clockwise.

When the PNC is told to transmit, it moves the data into the transmit buffer on the PNC. So the PNC for Node A watches for a token. When it sees the token, it strips the token off of the ring and begins to transmit its data block. The PNC will put out the special leading frame character, the packet given it by the software, and then the rest of the hardware utilized information - The CRC word, the Ack Frame, the Ack Byte and the Trailing Frame. It will then put the token back out on the ring directly behind the packet. (slide 5) Actually the token is changed into a leading frame character while it is within the delay cycle (6 bits) of the PNC.

Node D also has a data packet which it wants to send to Node C. Its PNC is watching for the token. The PNC will see the leading frame of the BA packet, and check for the node number. Since this packet is not for Node D, it will continue to transceive and watch for the token. Node D does, however, do CRC validation on the data, and will indicate in the ACK Byte if it sees a CRC failure in this packet.

When the token arrives, Node D's PNC strips it off (converts it), puts its packet on the ring, and reissues the token. We now have two data packets on the ring. (slide 6) Node C will transceive the first packet, and then notice that the second packet is intended for his node. The PNC will continue to transceive the packet, but it will also read the packet into the receive buffer in the PNC. The PNC will do the CRC validation checking on the data, and if everything is ok, it will set a bit in the ACK byte to indicate that the packet was received OK. (slide 7) The PNC will do a DMA to move the data into the host memory, to a preassigned buffer. The PNC will interrupt the host to inform it that there has been an action completed. The host will get the status from the PNC which will tell it that the buffer within the host memory now contains the received data.

We still have both packets on the ring, even though one of them was received by the intended node. The packets can only be taken off of the ring by the node which put them on the ring. All packets make the complete circle of the ring.

Node B sees the first packet, recognizes that it is for his host, and

receives it into the receive buffer. The ACK byte will be modified to show a correct receive. Node B will transceive the following packet and the token. Since Node B has nothing it wants to transmit at this time, the PNC just transceives the token.

When the data gets back to Node A, the PNC recognizes that its node id is the sender of the first packet. It will strip this packet off of the ring, and see what the ACK Byte says happened. The PNC on node A will issue a transmit interrupt to the host to let it know that the transmit has completed. The host will do an INA (input a register) to pick up the transmit status. The PNC for node A will transceive the next packet and the token. (slide 8)

When the remaining data packet reaches Node D, it will strip its packet from the ring and inform its host of the completion of the transmittal. The PNC will transceive the token. (slide 9)

When I have spoken of there being a packet on the ring, or multiple packets on the ring I am speaking of the apparent reality seen by a node watching data go by. The actual speed of the ring is great enough, that it is necessary to have at least 16 nodes in the ring before the smallest transmitted packet will fully fit 'on' the ring. However, a node transceiving the data will see (in the above case) two data packets go by and then the token.

## 13.3 Logical and Physical Nodes

In the slides, I have shown a 5 node ring. This means that there are 5 nodes physically connected to the cable. Any host which is physically connected to the ring cable is obviously a physical node.

I am using a term 'logical node' to indicate a node that is physically on the ring and logically visible to a user. A node is logically visible to a user if the Network Configuration for his host has the other node defined. For example, if Node A has configured Node B and Node B has configured Node A, the two hosts can communicate and the nodes are logical nodes. If Node A has Node B configured, but not Node C or Node D, then it is on a ring with two logical nodes (itself and node B), but 5 physical nodes.

The ability to communicate between two nodes is a cooperative effort, however. If Node A has configured Node B, but Node B has not configured Node A, the two hosts cannot send data to each other. Node A would not be able to send data to Node B even though Node B is a logical node for it. Node A would attempt to send data, and node B would discard it as unacceptable because it does not think of Node A as a logical node. It cannot 'see' Node A.

It is very possible to have a very large physical ring with many physical nodes which has a set of varying and overlapping logical rings. If Node A sees only Node B and Node B sees Node A they form a two node logical ring. If Node B sees Node C and Node D in addition to Node A, it is part of a larger logical ring also. The production ring at the R & D facility at Prime is an excellent example of this, since there are no nodes which are accessible by all other nodes.

In the main body of the discussions, I will be using the assumption that the nodes on a ring are all on the same logical ring as well as the same physical ring. In those situations where there is a good reason, I will differentiate explicitly.

## 13.4 Nodes Up and Down

In order for a host to know that another node is up, a couple of things have to happen. First the host has to have configured the other node. Then it has to have received a packet from the other node. It doesn't matter what kind of packet it is, although it is usually a timer driven node identification packet. These are sent out on a timed interval by each node. In earlier versions of RingNET each node sent a node id packet to each other node for which it was configured. It did this once every 30 seconds. Now the node sends out a broadcast packet once every 10 seconds.

So that if a new node connected into the ring, it could be about 30 seconds before it would see the other nodes as Up.

Once a node is marked as Up, it will remain so marked until a transmit to that node fails or until we time out on receive (we have not received a node id packet from them for 3 minutes). This is another reason for the sending of the node id packet at a timed interval. Even if there are no logical connections set up to the other node, we are sure to send it a packet every 10 or 30 seconds.

Thus our vision of the other nodes on the ring is this:

If we have never heard from them they are Down.

If we can receive from them they are Up.

If we cannot transmit to them they are Down.

If we have not heard from them for 3 minutes they are Down.

## 13.5 Error Recovery

In the example given there were no problems with the communications. This is the normal situation, but errors do inevitably occur. I will be talking about level 2 errors and error recovery most of the time. Level 2 is the ring and ring protocol level. It is this level which guarantees the correct physical delivery of communications data. In other words, when level 2 passes data up to the higher levels, it is promised that the data has been received correctly. For RingNET we interface into a standard X25 level 3 protocol.

The usual type of error is an interference of some kind. This would indicate that something physically went wrong with the attempt to communicate.

### 13.5.1 Data Degradation

There are numerous things which can cause the data to be corrupted as it traverses the ring. If the distance between active nodes is over 750 feet the signal will degrade. If there is a loose connector into a PNC the signal can pick up noise. And a new node entering the ring will cause an amount of noise on the circuit.

A NAK is a Negative Acknowledgement. It says that something in the data packet was corrupted and the data could not be received correctly. This indication would be set by the PNC if the data did not pass the CRC validation, or if the ACK Byte did not pass parity checking. When the transmitting host sees a NAK, it will attempt to get the data across by retransmitting the packet. It will try 20 times, and then give up. The CRC failure bit in the ACK Byte can be set by any PNC on the ring.

On the receiving side, a packet is received because it has this node's number in the destination field. If the packet shows a CRC failure or a ACK Byte Parity error, the packet is never given to the host. Since the node number is within the area covered by the CRC, if we cannot be sure of the CRC we cannot be sure the packet was really intended for this node.

### 13.5.2 Receive Congestion

The PNC has a single receive buffer currently. If the PNC does not have this buffer free or a Primos buffer is not assigned to receive the data from the PNC, it will not attempt to accept the data from the ring. However, it is necessary for the PNC to indicate that it has seen the packet. A WACK is a Wait Acknowledge. It indicates that the node to whom the packet was sent has seen the packet and that the data was clean. The node, however, does not have a buffer to receive the packet into at this time. The packet would be resent by the

transmitter host.

The packet is sent 20 times, and if it has not been accepted it is placed in a Linked List structure. This structure is maintained in node id order, and transmittal sequence within node it. This allows us to put the packet which is not being able to get through on the Linked List and continue transmitting to other nodes on the ring. The packet on the list is retried another 20 times after a second has gone by. This retry sequence is done 5 times. If the packet still cannot get through it must be assumed that the node to whom it is addressed is in a halted state. The node is marked down.

### 13.5.3 Node Leaves Ring - Non Acknowledge

A Non Acknowledge is the lack of any modification to the ACK Byte at all. It indicates that the node to whom this packet was sent is not physically in the ring at this time. If the node were present, it would have either Acked, NAKed, or WACKed. The attempt to transmit is dropped.

### 13.5.4 Packet Lost or Ring Down

A packet is considered to have Not Returned if a token is seen before the packet is seen. As was seen in the example earlier, the token is always inserted back onto the ring after the data packet. When that node next sees the token, it should have already stripped the packet from the ring. If it does not see the packet before the token, it must assume that the packet was so badly corrupted that it was not recognizable. It will attempt to recover by retransmittal 20 times.

A Packet can time out This mean the PNC sent the packet, but neither it nor a token was seen before a timeout. This would indicate that the ring might be broken and no traffic is getting through. It could be a ring going through token recovery, also. The node waits for a token and attempts one more transmit with this packet. If the second attempt to transmit does not succeed, the attempt to transmit is dropped.

## 13.6 Token Recovery

It is possible for the token to be totally lost from the ring due to noise or interference. This would have the effect that no one could transmit at all and all communications would cease.

Since RingNET is a peer ring (it does not have one node which 'controls' the ring) all nodes are able to recover the token. A token is recovered by having the PNC transmit a token onto the ring. The question in a peer ring is, of course, who gets to recover the token and how do they know they are supposed to do it.

A node determines that the token has been lost only if it has a transmit pending. When there is a transmit pending the PNC specifically watches for a token. There is a timer set in the host, which expires if we have not heard back from the PNC within a logical amount of time. The logical amount of time would be the maximum token circulation time. This is the time it would take the token to circle the ring if every node was sending a maximum sized packet.

If this timer expires, the host checks to see if the PNC has seen the token - i.e. has initiated the transmit. If it has, the host will set a shorter timer and wait again. This second timer is how long it would take a single packet to circle the ring. This second timer is the one which would trigger the Packet Timed Out error.

If the PNC has not seen a token, the host goes into a staggered timeout to determine if it should be the node to issue a token. This is based on a timer interval multiplied by the node number (which has been masked). The first time we back off and then issue a token we use only the low order 3 bits of our node number. If there is a collision and no token is seen, we use the low order 6 bits, then 8 bits. Unless the ring is physically broken the token is recovered very quickly.

## 13.7 Ring Break Detection and Localization

One of the primary needs in order to support large rings is to be able to determine quickly when problems have occurred and to be able to find the problem. This can be down for ring breaks in RingNET. If there is a problem with the ring which is severe enough that data cannot make the entire loop, the break can be found.

The code takes advantage of the fact that the aborted transmit packets are returned and that they contain the reason that they were aborted. The level II code for the ring checks these packets. If we begin to get packets aborted because of a Packet Timed Out, it goes into a checking sequence. If we get three consecutive packets timed out (which means six attempts at transmit), without having had a successful transmission, the ring can be stated to be down.

The routine prints out on the control console ***RING MAY BE DOWN*** and puts on a date and time stamp. The routine will not reissue this message unless we begin to get successful transmissions, and then the ring breaks again. If there is a yoyoing ring, then, you will get multiple messages, but you will get only one message for any given break down. It is not necessary for the routine to log the probable break as an event, since the nodes being marked down are labled as being down because of the broken ring.

Once the system administrator has reason to believe that the ring is broken, there is a program that can be run to determine the location of the break. This is called Find_Ring_Break. The program will describe the apparent state of the ring and tell which nodes can be received

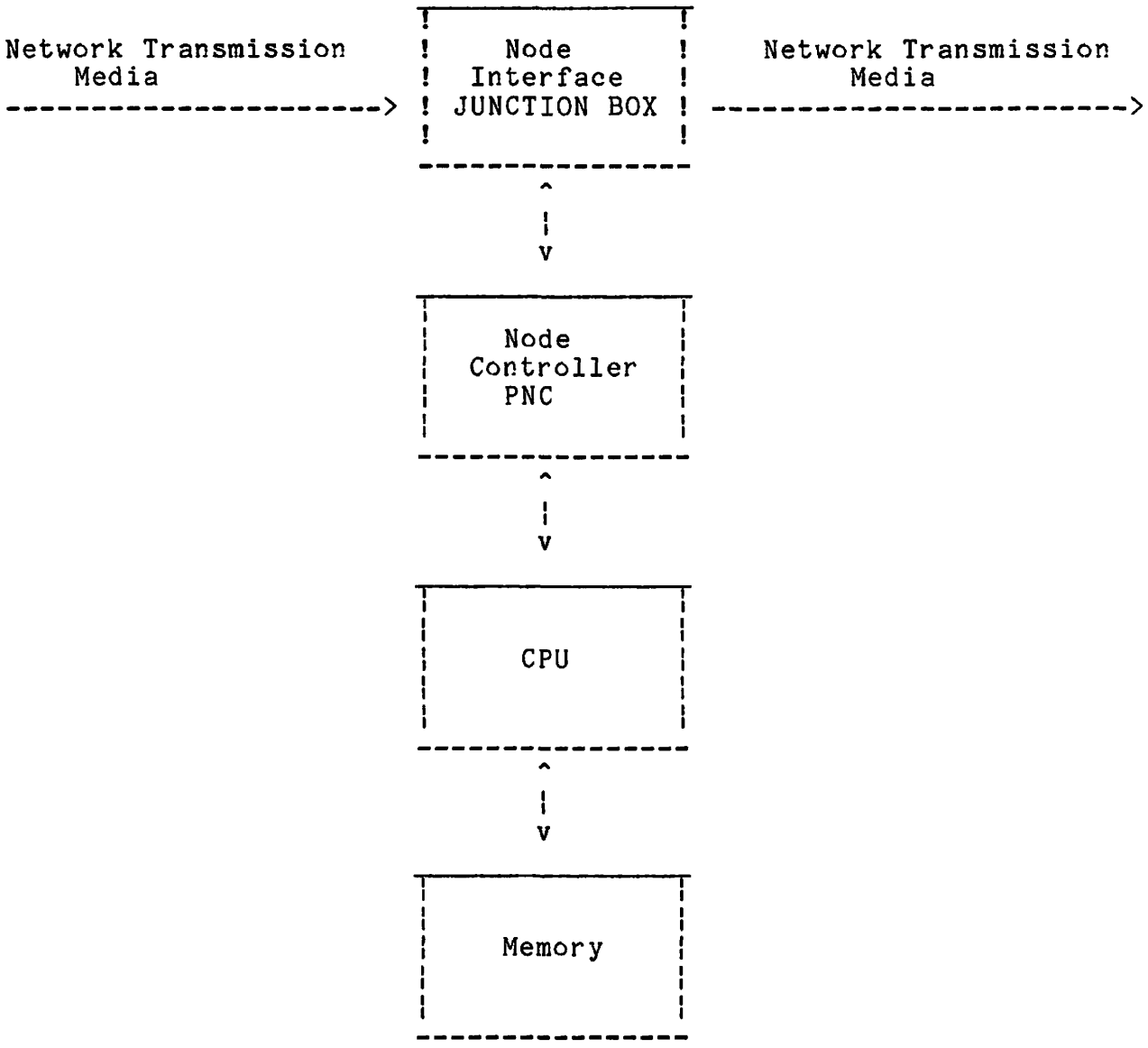
from and which cannot. It will then attempt to state where the break specifically is. It can do this only to the extent that nodes have been active on the ring. It will localize the break to between two active nodes.

The program requires that the user provide a file containing the information as to the physical sequence of the nodes on the ring. If this information is not available, the program will print out the nodes in node-id sequence and inform the user which are up and which are down. Given a knowledge of the ring protocol, the break can be located.

## 13.8 Statistics and Error Reporting

A second tool for the user of the ring is a program which will display (and archive into a disc file if desired) the information as to what is happening in this node on the ring. It will display throughput and error information and do some calculation of peak throughputs. This program is called Monitor_Ring_Node.

Slide 1


Network Transmission Media
Node Interface JUNCTION BOX
Network Transmission Media
Node Controller PNC
CPU
Memory

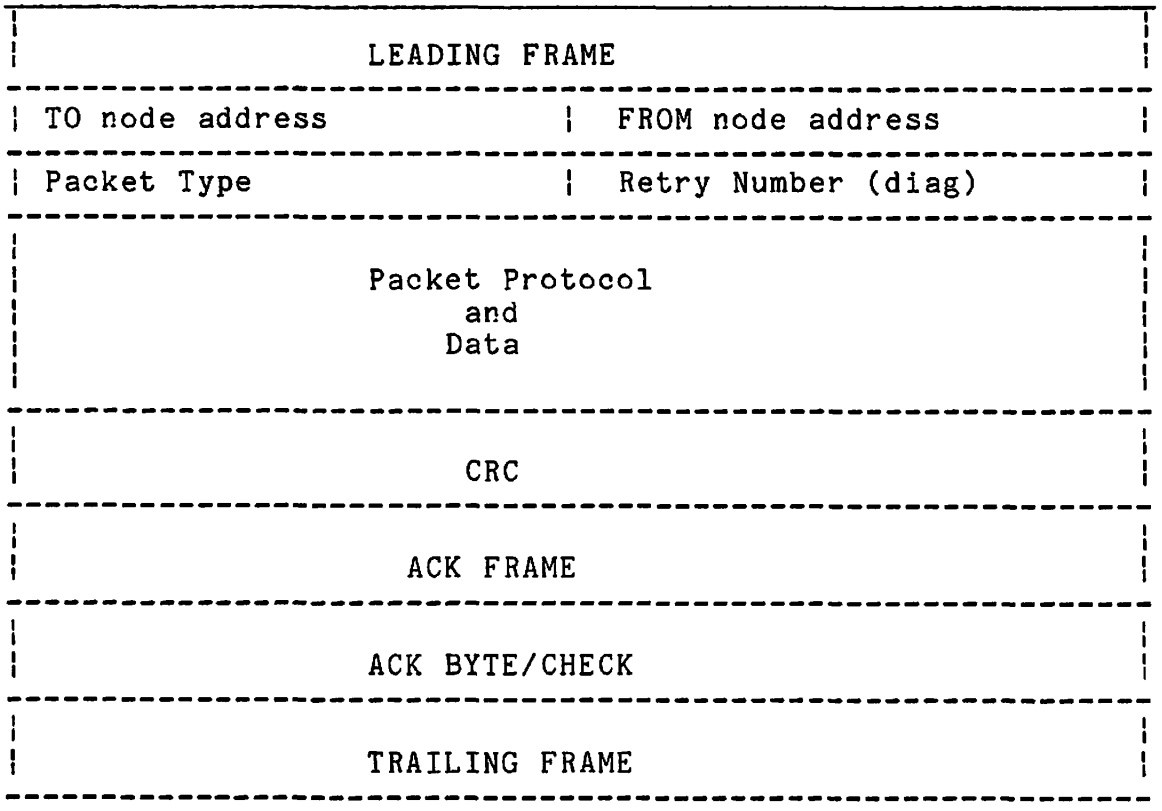
## SPECIAL CONTROL CHARACTERS

Slide 2

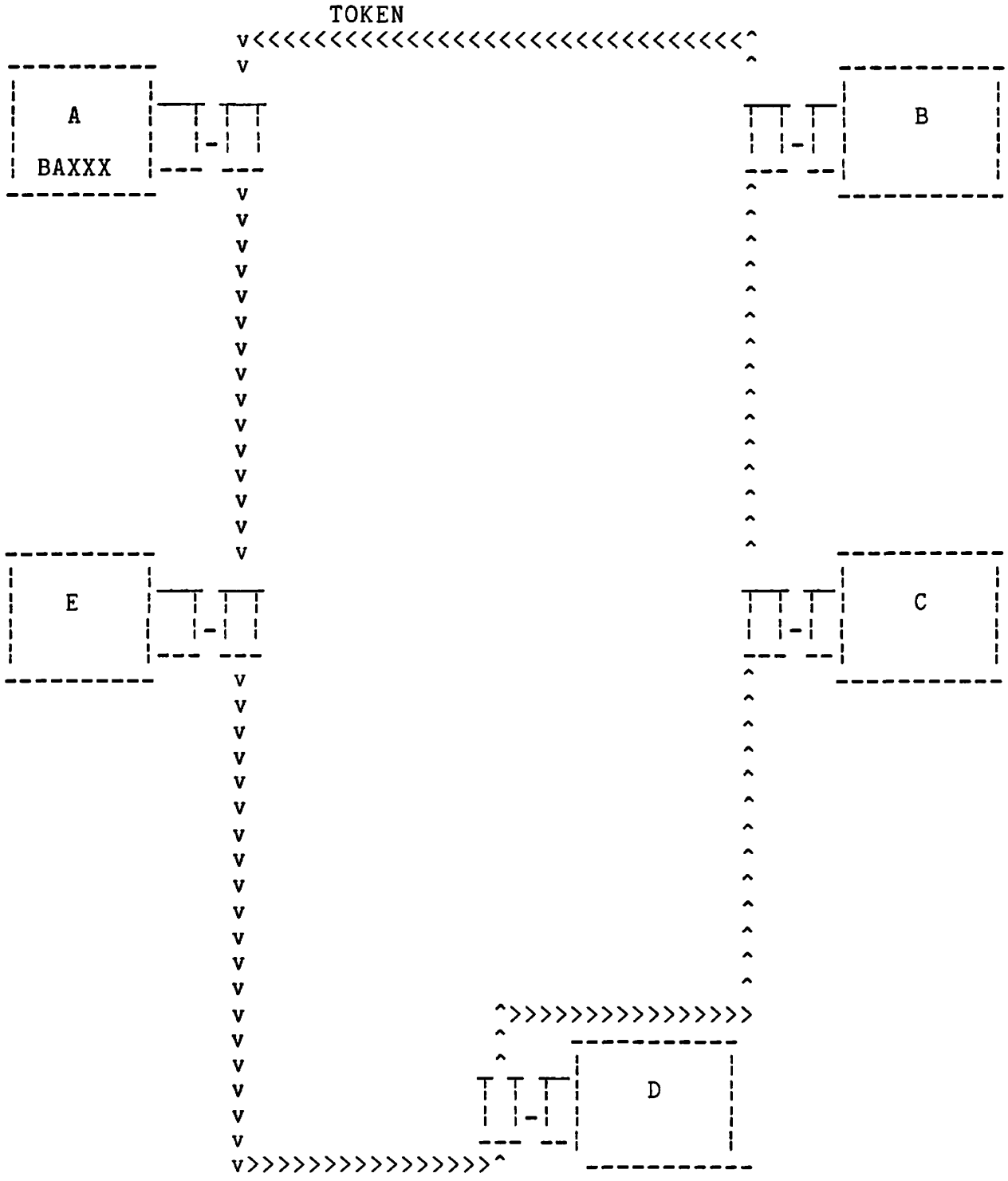
| Function | As Transmitted In Normal Mode | As Received In Shift Register |
|---|---|---|
| Token | 1000001 00101 | 10001 00101 |
| Leading Frame | 1000001 00110 | 10001 00110 |
| ACK Frame | 1000001 00111 | 10001 00111 |
| Trailing Frame | 1000001 11111 | 10001 11111 |

PACKET FORMAT

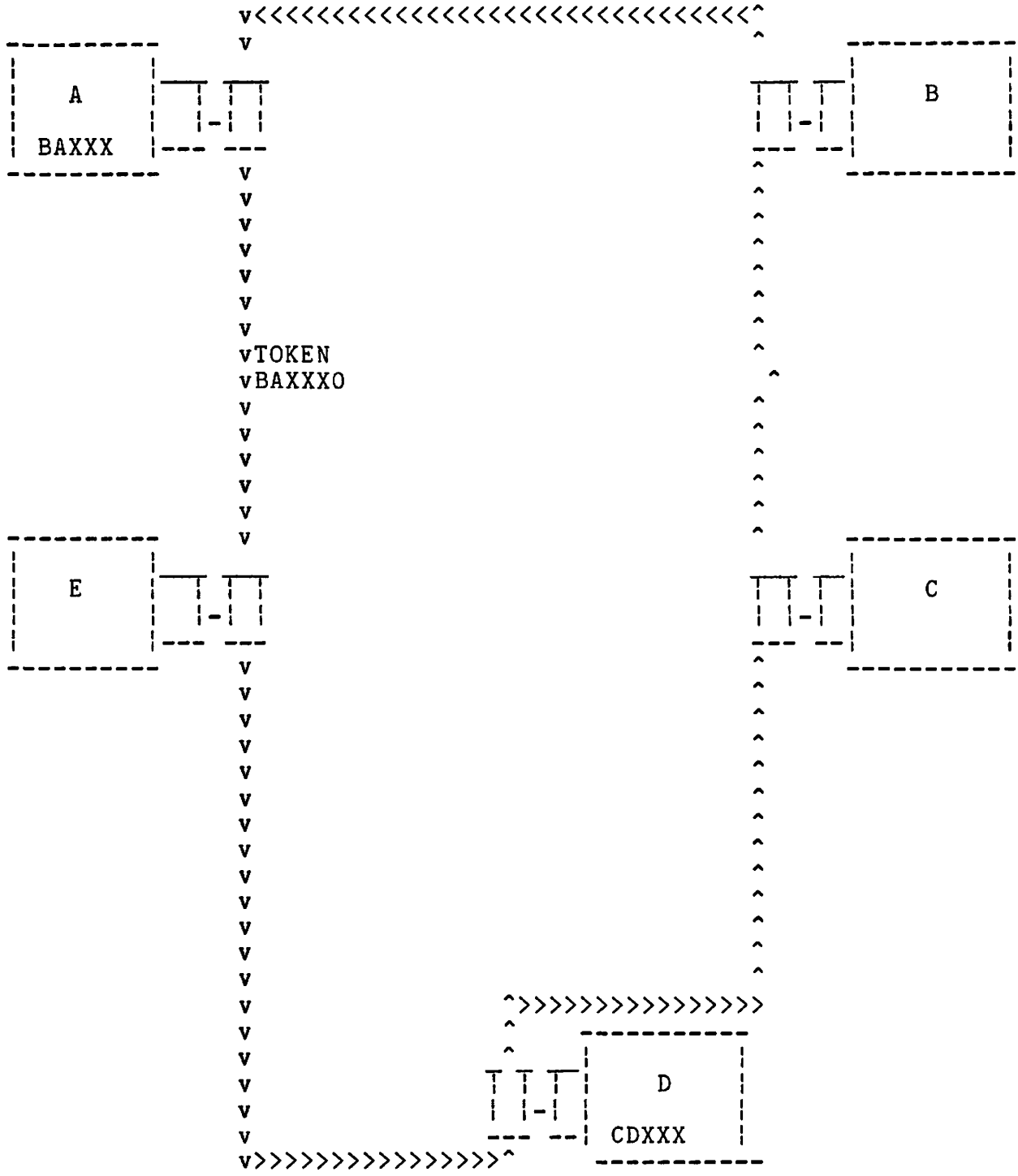
Slide 3



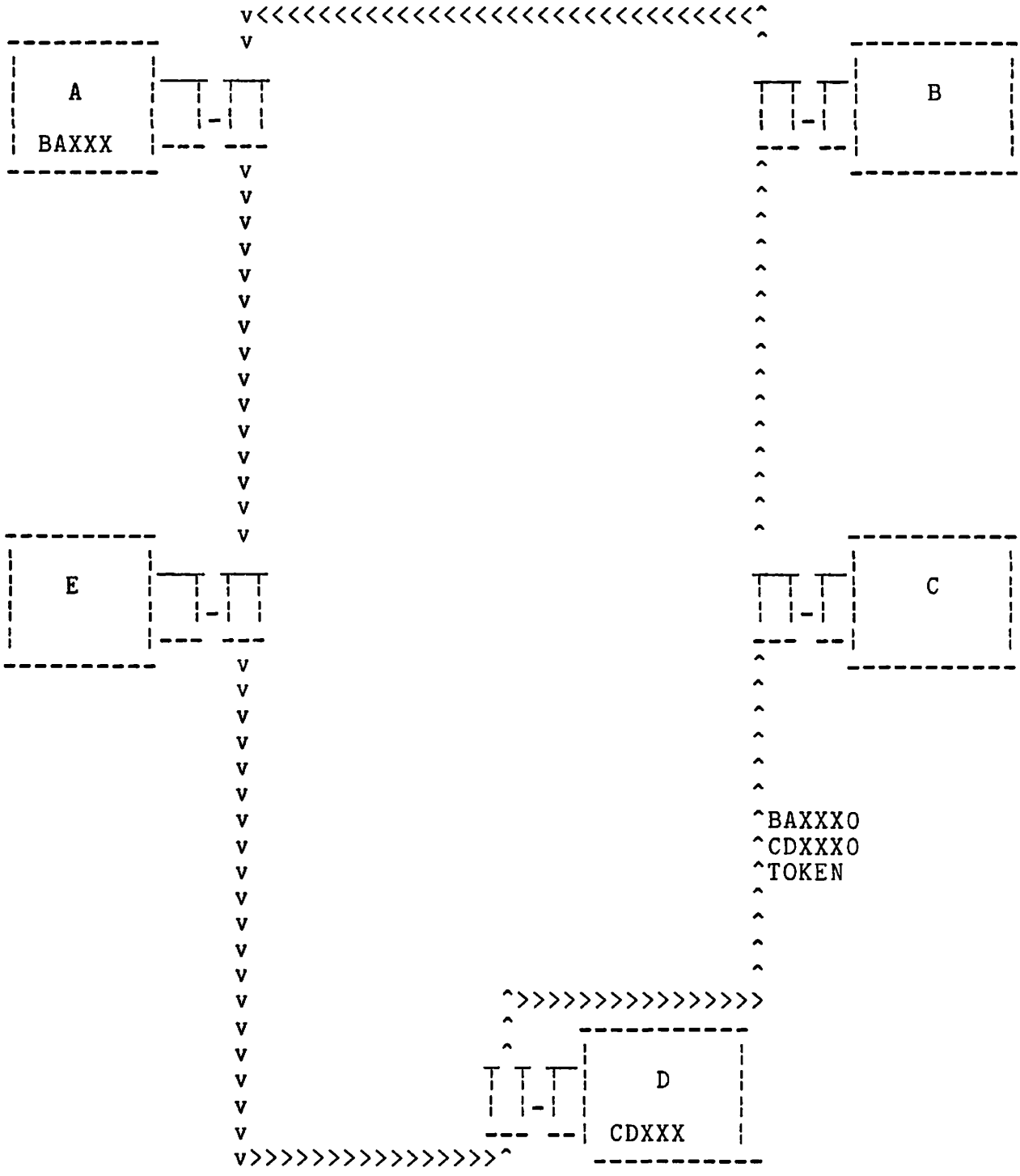
Slide 4



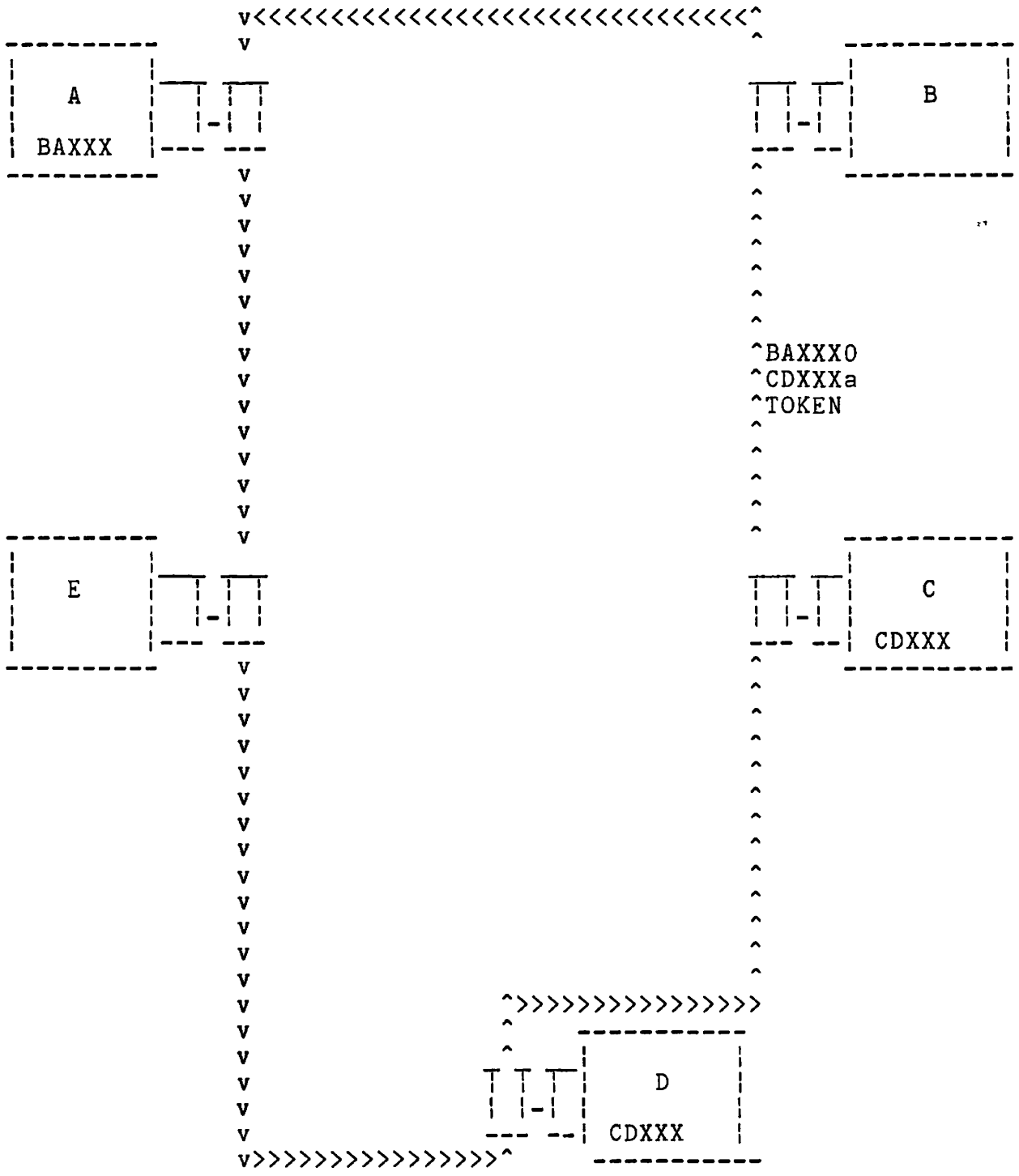
Slide 5



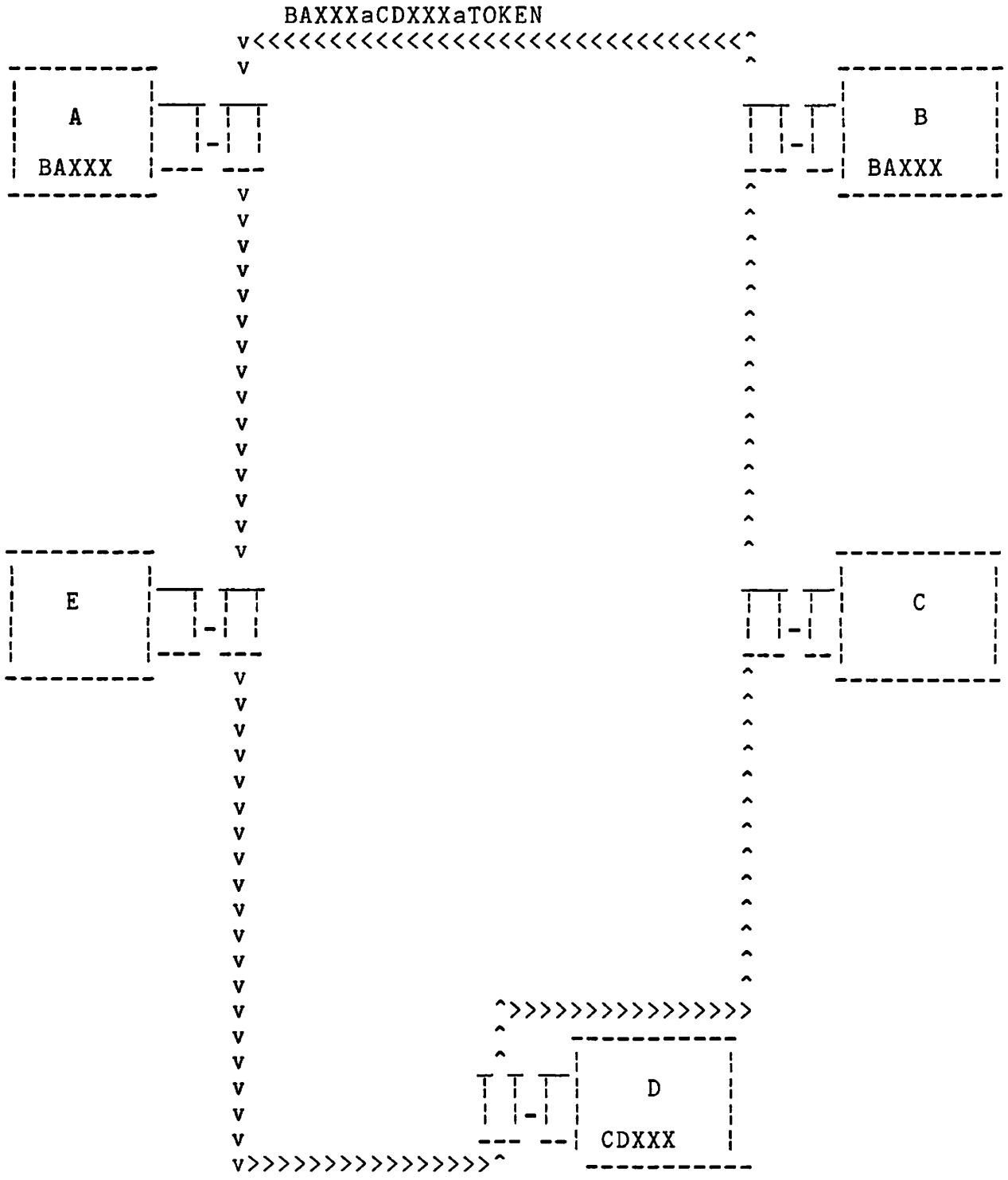
Slide 6



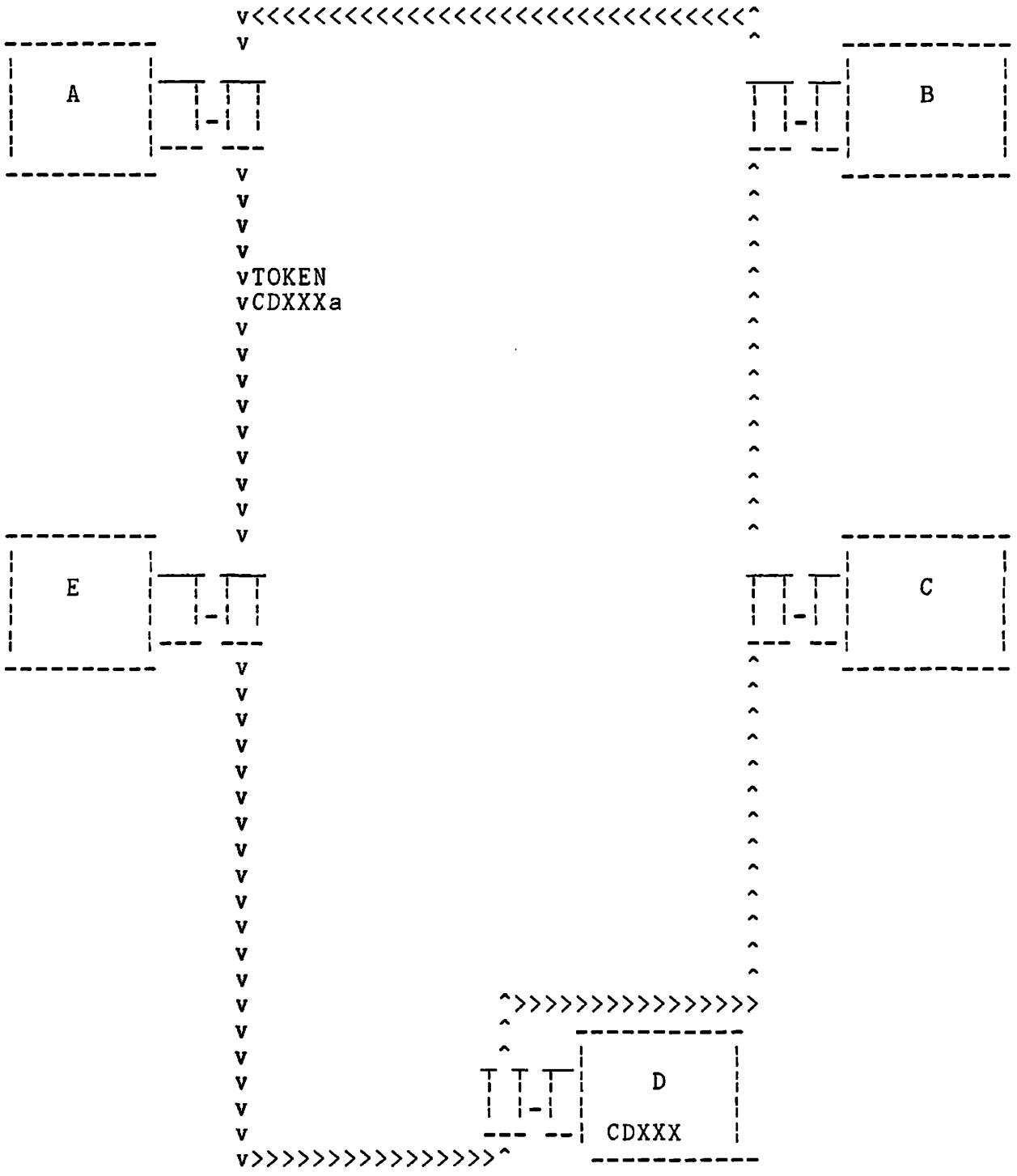
Slide 7



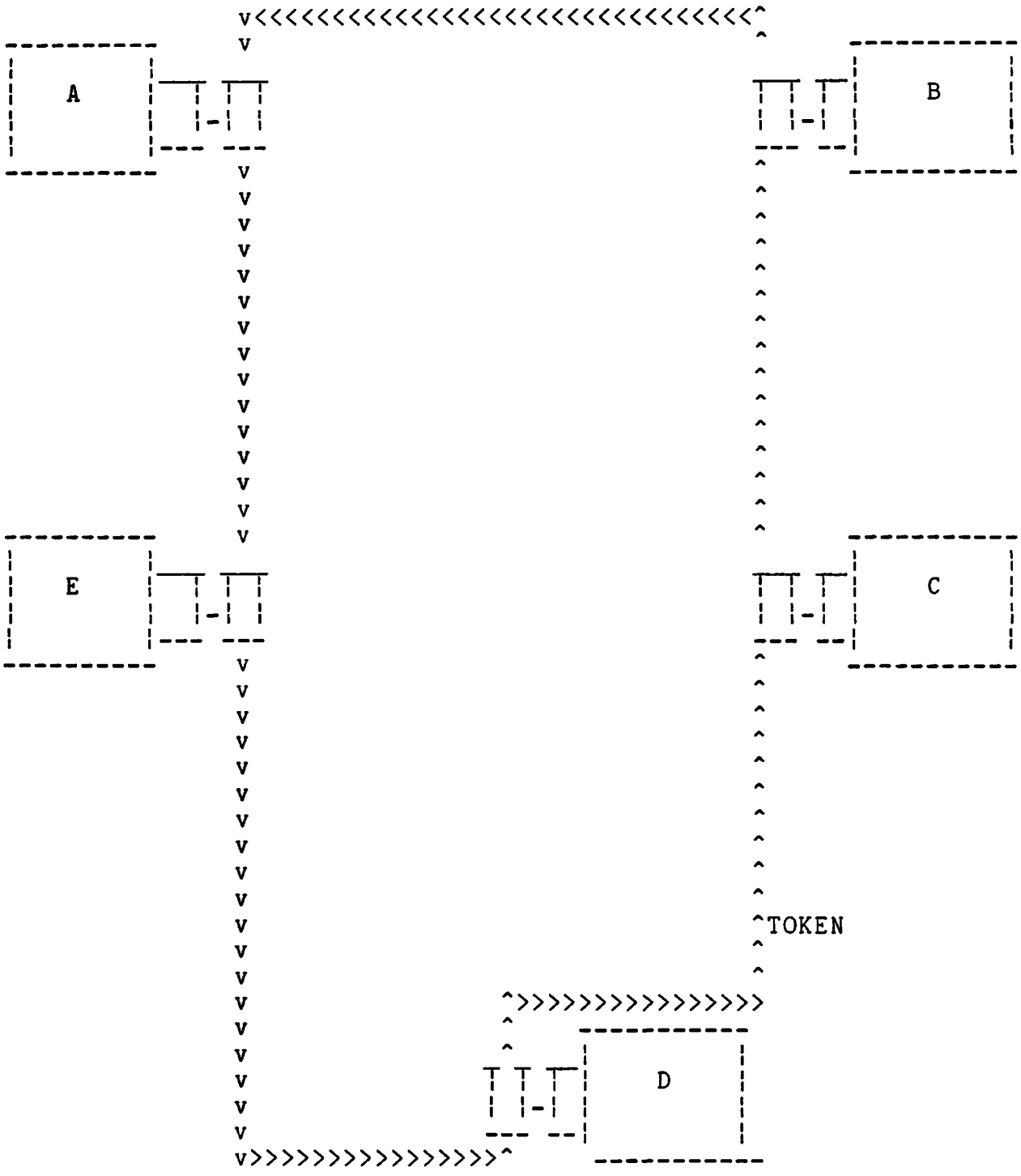
Slide 8



Slide 9



Slide 10



14 Reference Documents

PE-TI-307 Primenet Node Controller Specification, Farr, 1978