INTRODUCTION

The Ring allows up to 255 Memotech MTX and FDX series of micro-computers to exchange programs and data, and is an example of a local-area network. The network is constructed in the form of a ring, with the output of one computer, or node, connected to the input of the next. The links between the nodes carry RS232 signals along single-core coaxial cable at a rate of 19200 bits per second. The maximum data transfer rate is approximately 30000 bytes every minute.

A computer connected into the Ring appears to the user to behave no differently to a normal machine, since the Ring operates in the background, under the control of the processor interrupts. The top line of the text screen is used to show information on the operation of the Ring and a real-time clock. This display may be turned off when not required.

Each node in the Ring is identified by a twelve character name, which can be changed by the user at any time. Information can be transferred by typing BASIC instruction lines beginning with the keyword NODE. The NODE commands can be used to send BASIC programs, Noddy pages and electronic mail, and send or request BASIC string variables and blocks of memory. Transfers can be addressed to individual nodes or to all nodes in the Ring. A node is not obliged to receive all transmissions and specific categories may be rejected by using a NODE instruction. NODE instructions are also used to give a name to a node, to list the names of other nodes in the Ring, and for various other functions that are described in detail in the following chapters.

Information is moved from one node to another in the form of data packets which are transmitted around the Ring from the source node to destination node. A data packet consists of a string of ASCII characters with a maximum length of 255. Only one node is able to transmit a data packet at any time and transmission times are allocated to each node by using a token passing system. The token is a three character group which is passed from one node to the next. When a node receives the token, it can transmit a data packet containing upto 64 bytes of data. The destination node sends back an acknowledgement and the source node passes on the token. A node with a large block of data to transmit splits the data into groups of 64 bytes and transmits one group each time it receives the token.

When the physical continuity of the Ring is established and the ROM packs that contain the Ring operating software are installed in all of the nodes, the Ring is started by the first node to enter a name. The initialising node is called the master and is assigned a node number of 1.
The numbers of the other nodes are determined by counting round the Ring from the master; this happens automatically and the user need not be aware of the number assigned to the each node. The master nodes creates the token and transmits it to the next node

in the Ring. The other nodes are divided into two categories, senders and repeaters. A sender node is identified by a name and can transmit and receive data. A repeater node can neither transmit nor receive data, but simply passes on any packets it receives; a repeater is converted into a sender by entering a name at the node or remotely, by an instruction executed at one of the other nodes in the Ring.

A RING ERROR is a condition that arises when the token is no longer being passed round the Ring. It can occur for several reasons, such as a break in the cable, a computer in the Ring being reset or electrical interference along the data links. When the cause of the error is removed, the Ring will be re-initialised by the master node. If the master is reset or physically removed from the Ring, the sender node with the first name in an alphabetic list of the senders will become the master and initailise the Ring. All of the nodes that were senders before the error occured will continue to be senders after the Ring recovery.

If a node instruction to transfer a large block of data is entered from the keyboard, or is encountered whilst running a program, it will be found that the execution time for the command is very short. It may take several seconds to completely transfer the data, but the BASIC interpreter is free to do other instructions while the transfer is in progress. If a ring error happens during the transfer, the remaining data will not be sent after the Ring is re-initialised. In some circumstances, for instance in applications programs, this may be undesirable. To ensure that data is successfully transferred, the error trapping mode can be selected by using a NODE instruction. In error trapping mode, a NODE instruction will not pass control to the next program line until the data transfer is complete. If an error occurs during the transfer, program control will be passed to a BASIC line specified in the instruction that selected error trapping mode. An error handling routine at this line number could then decide what action to take, which might be a jump back to the data transfer command to re-transmit the data.

Assembling the Ring Hardware

The Ring hardware is simple to assemble. Each Memotech computer that is to be included in the Ring should have:

1. A Ring ROM.
2. RS232 communications board.
3. RS232 cable to connect adjacent computers together.

The RS232 communications board should already be inside your Memotech computer. If it is not, consult your dealer. The Ring ROM attaches to the expansion slot on the end of the computer, and the RS232 connector wires plug into the RS232 socket at the back of the computer. The Ring is physically completed by inserting the male plugs on the wire to the female plugs on the

computer that is to be next in the Ring.

The Ring software

The Ring should now be ready to be started up. Reset all of the computers in the Ring so that the standard Memotech BASIC screen is displayed. A clock should appear on the top line of the screen at each computer. (This is the Ring Clock, which gives the same time as the normal BASIC clock.) If the clock does not appear, check that all necessary elements of the Ring as given above are present.

To send or receive information via the Ring, each node must be given a name. (A node is simply the short-hand name for a computer connected in the Ring, which will be used throughout the remainder of this document.) This name may be chosen by the operator at each node, or assigned automatically from another node. To give a node a name, enter the following command

```
NODE NAME,<node_name>
```

where <node_name> is a string expression that begins with a capital letter and is truncated to twelve characters, e.g.

```
NODE NAME,"A"
```

or equivalently

```
LET NODENAME$="A": NODE NAME,NODENAME$
```

(Note. The NAME command is just one of a number of extended BASIC commands that are used with the Ring, each prefixed by the word NODE. These new commands can be treated like any other BASIC command, with the exception that they must be the last command on a line, like REM.)

The message

```
Ring initialised
```

should appear on the top line of the screen at the node at which the NAME command was executed. The 'top line of the screen' is called the node display screen, or just node screen. This screen is constantly re-written, which means that it overwrites any characters printed at the top of the screen, and also that the CLS command, for instance, will clear the node screen only temporarily.

At each of the other nodes

```
Ring in operation
```

should be displayed. If this has not happened then check that all

nodes are connected together in the proper manner and try again. Do not be concerned about any other messages that might be displayed - these will be discussed later.

Should the correct messages still not appear after checking the connections between nodes, there is likely to be a hardware problem. The following procedure ought to be adopted to isolate the error. Reset all machines, connect the output from each computer into its own input, and enter a NODE NAME command. Each machine should display the

Ring initialised

message, and will be passing the token back to itself. Any computer that fails to do this is almost certainly the source of the problem. Now join the nodes together in pairs and repeat the NODE NAME command. If any pair fails to operate, the problem is likely to be faulty connectors. Continue connecting the machines into bigger groups until they are all in the same Ring.

Hopefully, the Ring has now been successfully initialised, which means that information can be transferred around the Ring. At this stage, however, there is only one node 'in the Ring' (only one node has been given a name. Remember that in order to send or receive data a node must have a name). This first-named node is the most important node in the Ring, and is called the master node. The person operating the master node has ultimate control of all other nodes in the Ring, if he chooses to exercise it. This will described in more detail later.
All of the other nodes, which are currently nameless and therefore 'not in the Ring', can be forced to enter the Ring by typing the following command at the master node

NODE ENTER

The message

Node is in Ring

should appear at each node. The name given to a node that has been entered by another node is its ring number. The master node has, by definition, the ring number 1; thus node number 1 is the master node. The node that receives the transmissions of the master is number 2, the one that receives the transmissions of number 2 is number 3, and so on.

To find out which nodes are in the Ring, and their names (and ring numbers), the command

NODE LIST

is available, which prints on the screen a list of all nodes in the ring. The use of this command is best explained by examples. Supposing the master node, named A, is the only node with a name and there are two other nodes. The NODE LIST command will then produce the following screen output

(i) at the master node

<Node name> <Ring No.>

A 01 on the main display screen

Node is in Ring on the node screen

(ii) at node number 2

2 02

A 01 on the main display screen

Node is not in Ring on the node screen

(iii) at node number 3

3 03

A 01 on the main display screen

Node is not in Ring on the node screenAt each node, the name (if it has one) and ring number of the node is displayed first, then a line is skipped and the other nodes in the Ring are listed. Therefore, at the master node, it can be seen that the master is the only node in the Ring. In other words, node A is the only sender node (the only node that is capable of sending and receiving data). At the other two nodes, it can be seen that the node is a repeater node (it is not in the Ring), and that the only node that is in the Ring is the master, A.

If the NODE ENTER command is now typed at the master node, the following screen output will be produced by a NODE LIST command

(i) at the master node

A 01

2 02
3 03 on the main display screen

Node is in Ring on the node screen

(ii) at node number 2

2 02

A 01
3 03 on the main display screen

Node is in Ring on the node screen

(iii) at node number 3

```
               3              03

               A              01
               2              02       on the main display screen

               Node is in Ring         on the node screen
```

This time all three nodes are in the Ring, and at each node the names of the other two are given.

An important point regarding names of nodes is that a node may be given a name, using the NODE NAME command at that node, at ANY time, even if it already has a name or was forced to enter the Ring with a name equivalent to its ring or node number.

This means that an operator at a node can change the name of his node whenever he wishes, which might lead to trouble if, for example, an important message destined for FRED fails to be delivered because FRED has been re-named BERT.

Fortunately, this problem can be solved with the aid of the command

```
               NODE DIR,<string>,<node number>
```

where <string> is the name of a string variable that has been defined and can hold at least twelve characters, and <node number> is a numeric expression that corresponds to the number of a node (this must be in the range 1 - 255). This command is used to discover the name of a node that has a particular number, e.g.

```
               NODE DIR,A$,1
```

With the node names as listed above, this will assign A to string variable A$, which will then have length 1. If the node specified by the number either does not exist because the number is greater than the number of nodes physically connected into the Ring, or the node is not in the Ring, then the string variable is assigned the null string, e.g.

```
               NODE DIR,NAME$,6
```

Here the variable NAME$ is a null string (it has length 0) since the parameter 6 is greater than the largest node number. The NODE DIR command supplies the same information as NODE LIST, but in a different manner and is intended for use mainly inside programs. (NODE commands in programs will be dealt with later.)

A node, although part of the ring, can be treated as a standard Memotech computer if desired. All Ring facilities will still be available to the operator, but Ring-independent programs may be written and run. The node screen is turned off by

```
NODE OFF
```

Although still part of the ring, neither messages nor the Ring clock will be displayed, and the computer can be treated (almost) as if there were no Ring at all. The differences occur with the cassette commands, SAVE/LOAD/VERIFY (this is explained in Chapter 6).

```
NODE ON
```

switches the node screen on, re-displaying node messages and the clock.

To clear the node screen of any old messages or unwanted information, use the command

```
NODE CLS
```

This chapter has covered starting the Ring, but so far no information has been sent around the Ring. This will be dealt with in Chapter 2.

Sending messages

The simplest form of communication between nodes in the ring is by sending messages with the message

```
NODE MESSAGE,<nodename>,<message>
```

where <nodename> is a string expression that is the name of the destination node, and <message> is a string expression that is the message (truncated to 30 characters) which will be displayed on the node screen of the destination node, e.g.

```
NODE MESSAGE,"2","Hallo Number Two!"
```

or equivalently

```
LET NODENAME$="2": LET MESSAGE$="Hallo Number Two!":
NODE MESSAGE,NODENAME$,MESSAGE$
```

At the node 2, the node screen will display

```
Hallo Number Two!
```

If the name of the destination node is given as *, then the message will be sent to all nodes that are in the Ring. (N.B. This star convention can be used with all node commands that require a destination node name except NAME, RCV and MRCV. These last two commands will be discussed later.)

```
NODE MESSAGE,"*","Hallo everyone!"
```

It should be noted that any information sent with the NODE MESSAGE command will be over-written by the next node message that is displayed. It does have the advantage of being displayed immediately on receipt so that an operator at the node can see it. Information that requires to be kept, or is larger than thirty characters, should be sent as post.

Sending Post

Each node in the Ring has a mail-box which can receive and store mail until the operator of the node wishes to look at it. (The words post and mail are used here interchangeably.) The command

```
NODE POST,<nodename>,<string_variable>
```

will send the contents of <string_variable> to the node specified by <node_name>.
E.g.

```
NODE POST,"JOHN",INFO$
```

which sends string INFO$ to the mail-box of node JOHN. The star name * may be used to send post to all nodes in the Ring.

When a piece of mail is sent successfully, the message

```
Mail sent
```

appears on the node screen of the sending node. At the receiving end, the message displayed is

```
Mail received
```

If the node specified by <node_name> does not exist, or does not begin with a capital letter, then the node message

```
Bad node name
```

will result. Attempting to use the name of another node in a NODE NAME command will also give this message. (This is an example of a node error, which is explained later.)

If the post item is too large to fit in the available space in the mail-box of the receiving node, or the mail-box is full, then

```
Data too large
```

will appear on the node screen of the sender. This is another node error message. The post item could be made smaller and the NODE POST command repeated, or

```
NODE MESSAGE,"JOHN","Delete some mail please"
```

could be sent, followed some time later by the post, after JOHN

has made some room in his mailbox (see below).

Looking at the Mail-Box

To display the contents of a mail-box use the command

NODE MAIL

Each item of post is displayed with the name of the node that sent it and also the time at which it arrived. (Remember, this time is just the normal BASIC clock and can be set using the BASIC command CLOCK.) The item that arrived earliest is displayed first.

Unwanted items of post can be selectively removed from the mail-box by

NODE CLEARQ

This command displays each mail item separately and asks the question

Delete?

If the Y key is pressed then the item will be deleted from the mailbox. If the BREAK key is pressed the command will be terminated, and if any other key is pressed the next item (if there is one) will be displayed.

All of the items in the mail-box can be deleted in one go by using the command

NODE CLEAR

There is an optional parameter to this command that specifies the size of the mail-box, i.e.

NODE CLEAR,<mail-box size>

where <mail-box size> is a numeric expression which is less than 8192 (8192 bytes is the maximum size that the mail-box can be set to). For example

NODE CLEAR,1000

This will remove all items of post from the mail-box and create a new mail-box of size 1000 bytes. If this parameter is missing, then the size of the mail-box will remain unchanged. The initial size of the mail-box is approximately 6000 bytes.

After any NODE CLEAR command, the node screens displays

Mailbox empty

This message will also appear if NODE MAIL or NODE CLEARQ is attempted when there is nothing in the mail-box.

Printing Post

The contents of the mail-box may be sent to a printer that is connected to the node by

```
NODE PRINT,ON
```

This causes the first item of mail to be printed, and then the next item, etc. An important point to note is that as a post item is printed it is deleted from the mail-box. Therefore, if no mail arrives whilst the printer is operating, the mail-box will become empty when the last item of mail has been printed. Post that arrives in the middle of printing is stored in the mail-box as normal and will in its turn be spooled (printed). Post arriving when the mail-box is empty will be stored, spooled and deleted as it is received.

The commands NODE MAIL, NODE CLEAR and NODE CLEARQ are not allowed while the spooler is 'ON'. The node message

```
Mailbox in use
```

will result if these commands are tried. Incidentally, this message is also produced if post is sent to a node where NODE CLEAR or NODE CLEARQ is currently being done, or when NODE CLEAR or NODE CLEARQ are tried at a node which is in the process of receiving post.

The spooler is turned off by

```
NODE PRINT,OFF
```

This command can be entered at any time and will have no effect if the spooler is not on. (The default state for the spooler is 'OFF'.) NODE MAIL, NODE CLEAR, or NODE CLEARQ will all now operate as usual. (The NODE MAIL and NODE CLEARQ commands may show that the first item in the mail-box is one which has been half-printed.) Assuming that nothing has been deleted from the mail-box by the node operator, a NODE PRINT,ON command will set the spooler going again from where it left off, with no loss of mail-box information.

A node may act as a telex machine by leaving it in the PRINT,ON mode.

This chapter deals with the transfer of BASIC string variables and memory blocks.

Sending and receiving strings

To send a string variable from one node to another use the

command

```
NODE SEND,<node_name>,<string_1>,<string_2>
```

This statement means that string variable <string_1>, which exists at the sending node, is to be sent to string variable <string_2>, which exists at the receiving node <node_name>.
E.g.

```
10 LET A$="ABCDEF"
20 NODE SEND,"JOHN",A$,B$
30 GOTO 10
```

In this program, the contents of string variable A$ (namely ABCDEF) will be sent to the node JOHN and put into string variable B$. Note that the receiving string variable MUST exist before the NODE SEND command is issued, unless the destination node is given as *, in which case any nodes receiving the string simply ignore it. (Remember that all NODE commands are like any other BASIC instruction and can be used either directly or within programs. However, any command following a NODE command in the same BASIC line will be ignored.)

Node messages that the NODE SEND command may produce are listed below:

(i) At the sender node

Variable does not exist	The receiving string is undefined (only if destination not *)
Data too large	String is too big to fit into receiving string variable
Variable sent	if success

(ii) At the receiving node

Variable received	if success

To request that a string variable be sent from a node, there is the command

```
NODE RCV,<node_name>,<string_1>,<string_2>
```

This works in a similar way to NODE SEND, with the difference that string variable <string_2> is received from node <node_name> into string variable <string_1> at the requesting node. Both string variables must exist, and <node_name> cannot be * (a string variable cannot be requested from all nodes).

Supposing that the example for NODE SEND is followed by

```
LET A$="": NODE RCV,"JOHN",A$,B$
```

The LET statement will make A$ a null string, and the NODE RCV will cause B$ (which is ABCDEF) to be sent from JOHN and put into A$. Therefore, A$ will be ABCDEF once again. This illustrates that the Ring provides true two-way communication; sending data and receiving data requested.

Node messages that the NODE RCV command may produce are listed below:

(i) At the sender node

Variable does not exist	The string that is being requested does not exist
Data too large	The requested string is too big to fit into receiving string variable

(ii) At the receiving node

There are no messages.

Some time after a NODE RCV command is successfully executed, the message

Variable received

will be displayed at the requesting node, indicating that the string variable has been transferred.

Sending and receiving blocks of memory

The transfer of a block of memory is done in much the same way as the transfer of a string variable. The commands NODE MSEND and NODE SEND are analagous to NODE SEND and NODE RCV.

NODE MSEND,<node_name>,<p1>,<a1>,<p2>,<a2>,<size>

This sends a memory block from RAM page <p1> at address <a1> to RAM page <p2> at address <a2> in destination node <node_name>. The number of bytes transferred is given by <size>.
As an example, the BASIC real-time clock, which consists of seven bytes at location 64855 decimal, can be sent from one node to all the rest. This has the effect of synchronizing all clocks in the Ring.

```
10 CLOCK "000000"
20 NODE MSEND,"*",0,64855,0,64855,7
```

A check is made to see whether the block to be sent actually exists at the sending node. If not, the BASIC error Out of range is generated. If there is insufficient memory at the receiving node to store the memory block, then the node message

Memory does not exist

is displayed at the sender. If the command is successful, the messages shown are

```
Memory block sent        at the sender node
Memory block received    at the receiver node
```

In the example below, the block of memory specified will not be sent if the node JOHN has only one RAM page, since the destination address of the block is in RAM page 1.

```
NODE MSEND,"JOHN",0,32768,1,40000,256
```

For any address greater than or equal to C000 hexadecimal or 49152 decimal, the address is in the top 16K of RAM and therefore always switched in, and so the value of the source page <p1> is unimportant, but it is advised that the value 0 be used to avoid confusion (this is done in the command that sends the clock).

To request a block of memory use

```
NODE MRCV,<node_name>,<p1>,<a1>,<p2>,<a2>,<size>
```

This command operates in exactly the way one would expect, e.g.

```
NODE MRCV,"JOHN",0,50000,0,64855,7
```

This gets the clock bytes from node JOHN and stores them at address 50000 decimal. As with NODE RCV, a node name of * is not allowed. If the block of memory requested does not exist

```
Memory does not exist
```

is displayed at the requesting node. Memory block received

is given at the requesting node when the block of memory arrives. No messages are produced at the node that received the NODE MRCV request.

The NODE MSEND and NODE MRCV commands are designed of transferring blocks of RAM memory; blocks of ROM memory cannot be sent or received using these two commands.

Sending programs

BASIC programs may be sent from one node to another by

```
NODE PROGRAM,<node_name>
```

<node_name> is the node to which the program is sent, e.g.

```
10 FOR X=32 TO 128
20 PRINT ASC(X);
30 NEXT X

NODE PROGRAM,"JOHN"
```

The program will be sent to the node JOHN, overwriting any existing program. Two useful variations to this command are

```
NODE PROGRAM,<node_name>,RUN
```

which causes the program to run on loading at the destination node, and

```
NODE PROGRAM,<node_name>,LLIST
```

which causes the program to be listed to a printer at its destination. (The options RUN and LLIST cannot be used at the same time.)

Sending Noddy Pages

Before any Noddy pages can be sent, it is necessary to make room for them at the receiving node with the command

```
NODE RESERVE,<no_of_bytes>
```

where <no_of_bytes> is a numeric expression that gives the size of space to allocate for incoming Noddy pages. This command destroys any BASIC variables at the node at which the command is entered, and is probably best done shortly after starting to use a node, but it may be done at any time and any number of times.

```
NODE RESERVE,1000
```

The above statement will set aside 1000 bytes for receiving Noddy pages.

To send a Noddy page use

```
NODE SNODDY,<node_name>,<Noddy_page>
```

For this command, <Noddy_page> is the name of the Noddy page to be sent from the sending node, and is also the name that the Noddy page will be assigned at the receiving node. There should not already be a Noddy page with the same name at the receiving node, otherwise the node message

```
Noddy page exists
```

is displayed at the sending node and the Noddy page is not accepted. If there is insufficient space at the destination node for the Noddy page, the node screen at the sender will show

```
Data too large
```

Assuming success, the node messages will be

```
Noddy page sent          at the sending node
```

```
Noddy page received      at the receiving node
```

A Noddy page that has travelled around the Ring can be treated in identical fashion to Noddy pages created at the node which received the Noddy page. It can be edited, displayed, deleted, etc.

To effectively prevent any Noddy pages appearing unexpectedly,

```
NODE RESERVE,0
```

will do the trick. However, this is a rather crude way of rejecting Noddy pages. In Chapter 6, the concepts of accepting data sent from another node or not, and disabling NODE commands will be discussed.

There are three NODE instructions which enable the computer to perform subroutines on given conditions: NODE FLAG, NODE GOSUB, and NODE RETURN. At every node, there exists always a GOSUB flag, which is normally set to zero. The value of this flag can be changed by another node using the NODE FLAG command, and if it becomes non-zero and a program being run encounters a NODE GOSUB command, then the BASIC subroutine starting at the line number specified in the NODE GOSUB command will be performed. The command NODE RETURN sets the GOSUB flag equal to zero.

The GOSUB flag can be looked at using the NODE STAT command, e.g.

```
10 DIM T(20)
20 NODE STAT,T
30 PRINT T(13)
```

The GOSUB flag at a node may be given a value by

```
NODE FLAG,<node_name>,<value>
```

<Value> is a numeric expression, which evaluates to an integer in the range 0 to 255. This value is assigned to the GOSUB flag at node <node_name>. It should be noted that this GOSUB flag has nothing whatsoever to do with the ordinary Memotech BASIC command GOSUB; it only has an effect on the NODE GOSUB command. The NODE FLAG instruction produces the node messages

```
Flags sent               at the sending node
Flags received           at the receiving node
```

As an example, suppose that whenever a GOSUB flag is given a value, that value is equal to the number of the node that sent the flag. The next program prints the name of the last node to send a GOSUB flag.

```
10 DIM T(20): LET M$=""
20 LET A$="GOSUB flag received"
30 NODE GOSUB 100
40 GOTO 30
```

```
50 REM
100 NODE STAT,T
110 NODE DIR,M$,T(13)
120 PRINT "Flag set by ";M$
130 NODE SEND,M$,A$,B$
140 NODE RETURN
150 REM Clear GOSUB flag
160 RETURN
```

As mentioned before, NODE RETURN puts the value 0 in the GOSUB flag. However, this may also be done by sending a GOSUB flag which is zero, eliminating the need for a NODE RETURN command to clear the GOSUB flag.

E.g.

```
10 NODE GOSUB 100
20 PRINT "Gosub flag is zero"
30 GOTO 10
100 PRINT "Gosub flag is non-zero"
110 RETURN
```

In the above program, if a GOSUB flag is received with a value greater than 0, the subroutine at line 100 will be executed, and it will continue to be done until a GOSUB flag of 0 is received.

It is always interesting to 'spy' on a node and see what it is doing, and this can be done very simply with the commands NODE FLAG, NODE GOSUB and

```
NODE SCREEN,<string>
```

For the latter, <string> is an already existing string variable that must be capable of holding at least 960 or 1920 characters. The effect of the NODE SCREEN command is to copy the contents of the entire text screen into the specified string. If the default BASIC screen is 40 columns wide (which it will be for MTX machines), then 40 * 25 (960) characters will be read. If the default BASIC screen is 80 columns wide (which it generally will be for FDX machines), then 80 * 25 (1920) characters will be read.

The program below shows how NODE SCREEN is used.

```
10 INPUT "Enter size of text screen: ";SIZE
20 IF SIZE<>960 AND SIZE<>1920 THEN GOTO 10
30 DIM TEXT$(SIZE): DIM T(20)
40 REM TEXT$ is for text screen, T is for NODE STAT
50 NODE GOSUB 10000

<Main part of program>

10000 REM Find out who sent GOSUB flag
10010 NODE STAT,T
10020 REM T(13) is number of spying node
10030 NODE SCREEN,TEXT$
10040 REM Read text screen into TEXT$
```

```
10050 NODE DIR,SPY$,T(13)
10060 REM SPY$ is name of spying node
10070 NODE SEND,SPY$,TEXT$,A$
10080 REM Send screen in TEXT$ to string A$ at node SPY$
10090 NODE RETURN
10100 RETURNWhen this program is run and a GOSUB flag received, the
```
subroutine at line 10000 will be executed. Assuming that the value of the flag is the number of the node that sent the flag, then a NODE STAT command is done to get this value, which is used in a NODE DIR command to get the name of the node. The text screen is read into the string variable TEXT$ and this is sent to the string A$ at the node which sent the GOSUB flag.

At this node, it is a trivial matter to display the contents of A$ and thus see what was on the text screen of the node to which the GOSUB flag was sent, i.e.

```
CLS: PRINT A$;
```

However, there is one small complication that arises from the fact that for a 40-column screen all 40 columns are read into A$, including the left-hand column which is purposely left blank and not printed to. Therefore, displaying A$ on the normal 39-column text virtual screens will produce a staggered effect. This can be overcome by

```
10 CRVS 6,0,0,0,40,24,40
20 REM Virtual screen 6 is a 40-column screen
30 VS 6:CLS
40 PRINT A$;
```

This problem does not arise with 80-column screens.

Calling machine-code routines

It is possible to call machine-code routines at other nodes with

```
NODE CALL,<node_name>,<page>,<address>,<parameter>
```

This command will cause the machine-code routine in the page given by the numeric expression <page> at the address given by the numeric expression <address> at node <node_name> to be called, with the HL register pair containing the two-value <parameter>. It is the responsibility of the person using the NODE CALL command to ensure that the machine-code subroutine contains a RET instruction, and also that it does not last too long since it is called during a processore interrupt.

Note The NODE CALL instruction has been included for the benefit of those who understand machine-code, and is potentially an extremely destructive command if used incorrectly. It is also very powerful, especially in view of the fact that <page> supplies full page information for the address of the routine, allowing calls to any ROM or any RAM page.

Example

```
10 REM Call to sound bell at a node
20 NODE LIST
30 INPUT "Ring the bell of node: ";NODE$
40 NODE CALL,NODE$,0,2387,0
50 GOTO 30
60 REM Bell routine is located at decimal
70 REM address 2387 in ROM page 0.
80 REM Input parameter not needed.
```

Saving, loading and verifying programs on cassette

In order to save, load or verify programs on cassette tape, it is first necessary to suspend the Ring so that these operations can be performed. This is required because the cassette routines, like the Ring routines, use the interrupts. To suspend the ring, type

```
NODE SUSPEND
```

Supposing this command is entered at node JOHN, then assuming that the command is enabled, the node message

```
Ring suspended
```

will be displayed at JOHN.

At every other computer connected in the Ring, the node message will be

```
Ring suspended by JOHN
```

This indicates to other nodes that node JOHN has issued the NODE SUSPEND command. Saving, loading and verifying can now be done without affecting the Ring. It is extremely important to realise that whilst the Ring is suspended no information whatsoever is able to travel around the Ring so each node is isolated.

When the saving, loading or verifying has finished, the Ring can be re-activated by the command

```
NODE CONT
```

which can only be issued by the node that suspended the Ring, otherwise the node error message

```
Suspend error
```

is produced. (This will also occur if the Ring is already suspended when a NODE SUSPEND command is attempted.)
Due to to the effect that these two commands have on the Ring, a NODE SUSPEND command should be done just before the cassette is needed, and a NODE CONT command be done as soon as the cassette is no longer required. When using a disc system to load or save a

program, the ring need not be suspended.

RS232 Instructions

The RS232 channels can be controlled with the following set of instructions:

1. NODE BAUD,<channel_no>,<baud_rate>

This command sets the baud rate (data transfer rate in bits per second) for the RS232 channel specified by the numeric expression <channel_no> equal to the numeric expression <baud_rate>. The values that <channel_no> can be are 0 or 1, and the values that <baud_rate> can be are 50, 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600 or 19200.

Example

NODE BAUD,1,1200

This sets the baud rate for channel 1 to 1200 baud.

2. NODE FORMAT,<Data_bits>,<Stop_bits>,<Parity>

This command specifies the format for data transmitted or received on RS232 channel 1. <Data_bits> is a numeric expression specifying the number of data bits, which can be 5, 6, 7, or 8. <Stop_bits> is a numeric expression specifying the number of stop bits, which can be 1, 1.5, 2. <Parity> specifies either even parity if +, odd parity if -, or no parity if 0.

NODE FORMAT,8,2,0

The above statement will set the channel 1 data format to 8 data bits, 2 stop bits and no parity. N.B. The Ring uses channel 0, for which the data format cannot and must not be changed.

3. NODE IN,<string>,<no_of_chars>

This command will take in the number of characters given by the numeric expression <no_of_chars> from RS232 channel 1 and put them into the string variable <string>, e.g.

NODE IN,A$,10The receiving string variable, A$ in this instance, must be large
enough to hold 10 characters, otherwise the BASIC No space error results.

4. NODE OUT,<string>,<no_of_chars>

This command works in an opposite way to NODE IN, e.g.

```
NODE OUT,A$,10
```

This outputs to channel 1 the first 10 characters of string variable A$, which must be at least 10 characters long.

The NODE IN and NODE OUT commands allow a node to communicate with any other device equipped with an RS232 socket, such as different make of computer or a terminal, and also allows two or more Rings to be linked, with a node in one Ring connected to a node in another Ring via RS232 channel 1. It would not be too difficult to get any node in one Ring to communicate with any node in a second Ring, through the 'link nodes'.

Node reset instructions

When the ring is in operation, the BASIC command NEW should never be used. To perform the equivalent of a BASIC NEW, there is the command

```
NODE NEW
```

which should always be used in place of NEW. This resets the BASIC program pointers, clears all variables, and restores the sound buffers.

The direct node equivalent of resetting the computer is the BASIC command

```
ROM 7
```

This resets all internal node variables, and also destroys the mail-box amongst other things. It should, therefore, be used with great care.

Each node has a set of parameters which tells the node whether to accept or reject various data sent from other nodes. There is also a set of parameters that tells the node which NODE commands it can do. These parameters are listed below.

Data Accept Flags

- Mail and messages
- Ring mail
- Strings and memory blocks requested by other nodes
- Strings and Noddy pages sent by other nodes
- Memory blocks sent by other nodes
- Calls
- Programs
- External parameter set

Command Enable Flags

```
POST,MESSAGE
```

```
RCV,MRCV,FLAG
SEND,MSEND,SNODDY
Ring packets
SET
PROGRAM,CALL
ENTER,SUSPEND,CONT
EXT
```

Notice that each set of flags contains eight separate groups.

To change these operating parameters, type

```
NODE SET,<accept_flags>,<enable_flags>
```

where <accept_flags> is a numeric expression that corresponds to the desired data accept parameters, and <enable_flags> is a numeric expression that corresponds to the desired command enable parameters, e.g.

```
NODE SET,135,31
```

To understand what the numbers 135 and 31 mean in terms of the accept and enable flags, it is necessary to think of them as two 8-bit numbers, each bit of which indicates whether a certain group of data will be accepted or not, or a certain group of commands will be enabled or not. (The groups have been already been mentioned above.) If a bit is set (1), then the data will be accepted or the command enabled (the flag is ON). If a bit is reset (0), then the data will be rejected or the command disabled (the flag is OFF).
For the accept flags, the binary equivalent of 135 decimal is

```
Bit:     7     6     5     4     3     2     1     0
Value:   1     0     0     0     0     1     1     1

      (128  +  0  +  0  +  0  +  0  +  4  +  2  +  1  =  135)
```

It is now quite simple to see what this means:

Bit	Group	Accept?
0	Mail and messages	Yes
1	Ring mail	Yes
2	Strings and memory blocks requested	Yes
3	Strings and Noddy pages sent	No
4	Memory blocks sent	No
5	Calls	No
6	Programs	No
7	External parameter set	Yes

Thus the node will accept only mail and messages (of any sort), requests for strings and memory blocks, and external operating parameters. If any other data is sent to the node, then the node message

```
Data rejected
```

will be displayed at the node which sent the data, and the data will NOT be accepted at the receiving node. Note that messages are ALWAYS accepted from the master node, regardless of the accept flags.

For the enable flags, the binary equivalent of 31 decimal is

Bit:	7	6	5	4	3	2	1	0
Value:	0	0	0	1	1	1	1	1

(0 + 0 + 0 + 16 + 8 + 4 + 2 + 1 = 31)

It is now quite simple to see what this means:

Bit	Group	Enabled?
0	POST,MESSAGE	Yes
1	RCV,MRCV,FLAG	Yes
2	SEND,MSEND,SNODDY	Yes
3	Ring packets	Yes
4	SET	Yes
5	PROGRAM,CALL	No
6	ENTER,SUSPEND,CONT	No
7	EXT	No

Thus the node is allowed do all NODE commands, except PROGRAM, CALL, ENTER, SUSPEND, CONT and EXT. If these commands are attempted, the node message

```
Command disabled
```

will appear, and the command will NOT be executed.

The command for setting the operating parameters of another node is

```
NODE EXT,<node_name>,<accept_flags>,<enable_flags>
```

where <node_name> is the name of the node to which the flags are sent, and can be * for all nodes. This command works in the way that is to be expected.

```
NODE EXT,"JOHN",255,255
```

turns ON all operating parameters for the node JOHN. For this command to be successful, the node at which the command is entered must be able to do NODE EXT (bit 7 of the command enable flags must be 19, and also the receiving node must be accepting external parameters (bit 7 of the data accept flags must be 1). If both conditions are met, the following node messages will result

```
Flags sent              at the sending node
Flags received          at the receiving node
```

When the ring is initialised, the master node has all of its operating parameters turned ON, which means the master node is in a position to alter not only his own operating parameters, but also those of any other node. All other nodes have all their parameters ON, with the exception of the EXT flag. This means that they can do everything but set the operating parameters of another node. They can accept everything.

It is not possible, by using the NODE SET command, to change the values of the external parameter set accept flag or the EXT command enable flag. Therefore

```
NODE SET,255,255
```

is interpreted as

```
NODE SET,127,127
```

These two flags can only be changed by a NODE EXT command entered at another node.

Status of the Ring

It is very useful to know what is happening with the Ring at any given time. The command that supplies information about the status of the Ring is

```
NODE STAT,<numeric_array>
```

The numeric array variable <numeric_array> must consist of one dimension of at least 20 elements, since there are 20 status values. The following program illustrates how the NODE STAT command operates.

```
10 DIM T(20)
20 NODE STAT,T
30 FOR I=1 TO 20
40 PRINT I,T(I)
50 NEXT I
```

The list of numbers produced by running this program is explained fully below.

Array Element	Meaning
T(1)	Ring Status: 0 = Ring not initialised 1 = Ring in operation 2 = Ring suspended 3 = Ring recovery in progress
T(2)	Number of nodes in ring (Range 1-255)
T(3)	Number of sender nodes (Range 1-255)
T(4)	Node type: 0 = Repeater (not in Ring) 1 = Sender (in Ring)

2 = Master (in Ring)

T(5)	Data accept flag byte
T(6)	Instruction enable flag byte
T(7)	Number of instructions being transmitted
T(8)	Number of instructions being received
T(9)	Last data received sender number
T(10)	Last data received type
T(11)	Error trapping mode status: 1 = ON 0 = OFFT(12) Error number
T(13)	Value of GOSUB flag
T(14)	Default baud rate for channel 0
T(15)	Channel 0 baud rate
T(16)	Number of items in mailbox
T(17)	Instruction received flag
T(18)	New name flag
T(19)	Non-token packet counter (modulo 256)
T(20)	Token counter (modulo 256)

Notes

1. The baud rate is given by dividing 19200 by the the baud rate number.

2. The instruction received and new name flags are set to non-zero values when an instruction is received or the name list is changed. They are set to zero after a NODE STAT command is executed.

The following example programs show how these status values may be examined.

Example 1

```
100 REM *** Ring Status Flag ***
105 REM
110 DIM T(20)
115 NODE STAT,T
120 CSR 2,2
```

```
125 ON T(1) GOSUB 145,150,155,160
130 IF INKEY$="" THEN GOTO 130
135 CLS: GOTO 115
140 REM
145 PRINT "Ring is not initialised": RETURN
150 PRINT "Ring is in operation": RETURN
155 PRINT "Ring is suspended": RETURN
160 PRINT "Ring recovery in progress": RETURN
```

Example 2

```
200 REM *** Number of nodes, node types ***
205 REM
210 DIM T(20)
215 NODE STAT,T
220 PRINT "There are";T(2);" nodes in the ring"
225 ON T(4) GOSUB 245,250,255
230 IF INKEY$ = "" THEN GOTO 230
235 CLS: GOTO 215
240 REM
245 PRINT "This is a repeater node": RETURN
250 PRINT "This is a sender node": RETURN
255 PRINT "This is the master node": RETURN
```

Example 3

```
300 REM *** Last instruction ***
305 REM
310 DIM T(20): LET M$=""
315 NODE STAT,T
320 NODE DIR,M$,T(9)
325 PRINT "Last instruction received from ";M$
330 PRINT "Last instruction type received: ";
335 ON T(10)-1 GOSUB 353,356,359,362,365,368,371,
374,377,380,383,386,389,392
340 IF INKEY$="" THEN GOTO 340
345 CLS: GOTO 315
350 REM
353 PRINT "CALL": RETURN
356 PRINT "EXT": RETURN
359 PRINT "FLAG": RETURN
362 PRINT "MESSAGE": RETURN
365 PRINT "POST": RETURN
368 PRINT "MSEND": RETURN
371 PRINT "SEND": RETURN
374 PRINT "SNODDY": RETURN
377 RETURN
380 RETURN
383 PRINT "PROGRAM": RETURN
386 PRINT "MRCV": RETURN
389 PRINT "RCV": RETURN
392 PRINT "DISC": RETURN
```

Example 4

```
400 REM *** Mail-box items ***
405 REM
410 DIM T(20)
415 NODE STAT,T
420 PRINT "There are ";T(16);" items in your mailbox"
425 STOP
```

From Example 3, it can be seen that each type of data received has an associated number. This number is the number of the NODE command that caused the data to be sent. These numbers and their NODE commands are

Code	Meaning
1	CALL
2	EXT
3	FLAG
4	MESSAGE
5	POST
6	MSEND
7	SEND
8	SNODDY
9	
10	
11	PROGRAM
12	MRCV
13	RCV
14	DISC

Codes 9 and 10 will never occur.

An explanation of the error trapping mode and the error number status variables is given in the next chapter.

A node error occurs when a NODE command fails, for some reason, to do what is expected of it. This may be due to a machine in the Ring being reset causing a RING ERROR, or data being rejected or a variable not existing, and so on. It is vital that a program using the Ring knows whether a node command has been successfully executed or not, so that when a node error occurs appropriate action can be taken.

The command

```
NODE ERROR,<line_no>
```

turns on error trapping mode (which by default is OFF). In error trapping mode, a number of NODE commands do NOT pass program control onto to the next BASIC line until it is known that the command was successful. If there is a node error, then program control is passed to the BASIC line specified by the <line_no> parameter in the NODE ERROR instruction. Each node error has an associated error number, which is the number of the node error

message that is displayed on the node screen when the error occurs.

From the previous discussion of the NODE STAT command, it can be seen that status variables 11 and 12 refer to errors. The eleventh variable indicates whether error trapping mode is enabled or not, e.g.

```
10 DIM T(20)
20 NODE STAT,T
30 PRINT "Error trapping ";
40 IF T(11)=1 THEN PRINT "ON" ELSE PRINT "OFF"
```

The twelfth variable gives the number of the last node error. There is a total of 18 distinct node errors and these are listed below.

Number	Node Error Message
1	RING ERROR

A node has been reset; the token or part of a data packet has been lost; a node has not acknowledged receiving data within a certain time limit. The master node will recover the Ring automatically.

2	MASTER RESET

Internal node variables have been changed at a node, so that they are no longer the same at each node. The master node has been forced to reset the Ring, making all other nodes repeaters. This seldom happens.

3	Data transmission error

Data sent to a node or an acknowledgement sent back has been corrupted, due to bad electrical transmission. With good connections between nodes, this error should never occur.

4	Node is not in Ring

A NODE command involving sending data to a node or requesting data from a node has been entered at a repeater node.

5	Bad node name

For NODE NAME: a name has been typed in that either does not begin with a capital letter, or is the name of another node. Generally: the node name specified is not the name of a node in the Ring.

6	* not allowed

A node name beginning with * has been entered in NODE NAME or NODE RCV or NODE MRCV.

7 Command disabled

The command enable flags do not allow this command.

8 Data rejected

The data sent has been rejected by the receiving node.

9 Node busy

Happens occasionally with NODE RCV or NODE MRCV. A request to send a string or a block of memory has been received by a node which is the middle of a NODE command that sends data. The request should be made again.

10 Receiver buffer full

When a node receives data such as a string, memory block, post or a program, an entry is made in the command receiver buffer. This entry tells the node how much data to expect and where to put it, so that when the next packet of data arrives (if there is one) it knows what to do with it. The entry is deleted when all the data has been received.This error indicates to the node that is sending the data that
the receiving node is unable, at the moment, to accept the data because its command receiver buffer is full. The user should keep trying to send the data until there is room in the receiver buffer. (Note. This error can also occur when a request for data is sent to a node that has no room in its command transmitter buffer. See directly below.)

11 Transmitter buffer full

The command transmitter buffer operates in a similar way to the command receiver buffer, except that this buffer is used to hold entries containing information about data to be transmitted. This error occurs when any command that involves sending data is entered at a node which has no free space in its command transmitter buffer. The user should try the command again until there is room in the transmitter buffer.

12 Data too large

NODE POST, NODE SEND or NODE SNODDY. The data being sent cannot be accomodated at the receiving node. Also occurs with NODE RCV if the requested string is bigger than the maximum size of the receiving string.

13 Memory does not exist

NODE MSEND or NODE MRCV.

14 Variable does not exist

NODE SEND or NODE RCV.

15 Noddy page exists

NODE SNODDY.

16 Program already loading

NODE PROGRAM. Results if a program is sent to a node to which a program is already being sent.

17 Mailbox in use

NODE POST. Post is sent to a node that is in the middle of a NODE CLEAR or NODE CLEARQ command.

18 Suspend error

NODE SUSPEND: The Ring is already suspended. NODE CONT: The Ring was not suspended by this node.

If a command is successful in error trapping mode, then the error number produced is 0. To turn off error trapping mode use

```
NODE RESTORE
```

Example

```
10 NODE ERROR,100
20 REM Error trapping mode on
30 INPUT "What name would you like? ";I$
40 NODE NAME,I$
50 NODE RESTORE
60 REM Error trapping mode off
70 STOP
100 PRINT "Please enter a proper node name"
120 GOTO 30
```

The following is list of all NODE commands and the errors (and numbers) that may be trapped in error trapping mode.

Node Command	Error Number	Error Message
NAME	1	RING ERROR
	5	Bad node name
	6	* not allowed
CLS		None
DIR		None
MAIL		None
CLEARQ		None

RESERVE		None
SUSPEND	1	RING ERROR
	4	Node is not in Ring
	7	Command disabled
	18	Suspend error
ENTER	1	RING ERROR
	4	Node is not in Ring
	7	Command disabled
FORMAT		None
IN		None
SCREEN		None
SET	4	Node is not in Ring
	7	Command disabled
STAT		None
ERROR		None
CALL	1	RING ERROR
	2	MASTER RESET
	3	Data transmission error
	4	Node is not in Ring
	5	Bad node name
	7	Command disabled
	11	Transmitter buffer full
EXT	1	RING ERROR
	2	MASTER RESET
	3	Data transmission error
	4	Node is not in Ring
	5	Bad node name
	7	Command disabled
	11	Transmitter buffer full
FLAG	1	RING ERROR
	2	MASTER RESET
	3	Data transmission error
	4	Node is not in Ring
	5	Bad node name
	7	Command disabled
	11	Transmitter buffer full
MESSAGE	1	RING ERROR
	2	MASTER RESET
	3	Data transmission error
	4	Node is not in Ring
	5	Bad node name
	7	Command disabled
	11	Transmitter buffer full

POST	1	RING ERROR
	2	MASTER RESET
	3	Data transmission error
	4	Node is not in Ring
	5	Bad node name
	7	Command disabled
	10	Receiver buffer full
	11	Transmitter buffer full
	12	Data too large
	17	Mailbox in use
MSEND	1	RING ERROR
	2	MASTER RESET
	3	Data transmission error
	4	Node is not in Ring
	5	Bad node name
	7	Command disabled
	10	Receiver buffer full
	11	Transmitter buffer full
	13	Memory does not exist
SEND	1	RING ERROR
	2	MASTER RESET
	3	Data transmission error
	4	Node is not in Ring
	5	Bad node name
	7	Command disabled
	10	Receiver buffer full
	11	Transmitter buffer full
	12	Data too large
	14	Variable does not exist
SNODDY	1	RING ERROR
	2	MASTER RESET
	3	Data transmission error
	4	Node is not in Ring
	5	Bad node name
	7	Command disabled
	10	Receiver buffer full
	11	Transmitter buffer full
	12	Data too large
	15	Noddy page exists
PROGRAM	1	RING ERROR
	2	MASTER RESET
	3	Data transmission error
	4	Node is not in Ring
	5	Bad node name
	7	Command disabled
	10	Receiver buffer full
	11	Transmitter buffer full
	16	Program already loading
MRCV	1	RING ERROR
	2	MASTER RESET
	3	Data transmission error
	4	Node is not in Ring

	5	Bad node name
	7	Command disabled
	9	Node busy
	10	Receiver buffer full
	11	Transmitter buffer full
RCV	1	RING ERROR
	2	MASTER RESET
	3	Data transmission error
	4	Node is not in Ring
	5	Bad node name
	7	Command disabled
	9	Node busy
	10	Receiver buffer full
	11	Transmitter buffer full
	12	Data too large
	14	Variable does not exist
BAUD		None
CLEAR		None
CONT	1	RING ERROR
	4	Node is not in Ring
	7	Command disabled
	18	Suspend error
GOSUB		None
LIST		None
NEW		None
OFF		None
ON		None
OUT		None
PRINT		None
RESTORE		None
RETURN		None

Notice that for commands like NODE CLEAR or NODE CLEARQ, error trapping mode does not apply, even though messages that are node error messages can occur, such as

Mailbox in use

if the spooler is operating for example. This is not considered to be a true node error (most node errors take place with commands that send or request data).

There are a total of 40 node screen messages, of which 18 are error messages. For the sake of completeness, the remaining non-error messages are as follows:

Message Number	Message
0	Clears node screen
19	Ring
20	Ring initialised
21	Ring in operation
22	Ring suspended by
23	Node is in Ring
24	Code executed
25	Flags received
26	Flags sent
27	Message sent
28	Mailbox empty
29	Mail received
30	Mail sent
31	Memory block received
32	Memory block sent
33	Variable received
34	Variable sent
35	Noddy page received
36	Noddy page sent
37	Program loading
38	Program received
39	Program sent

RING INSTRUCTION SET SUMMARY

1. Definitions

I$ is a string variable which is truncated to 12 characters.
S$ is a string expression which is truncated to 40 characters.
M$, A$, B$ are string variables.
J, K, L, P are numeric variables with integer values in the range 0 - 255.
N, X, Y are numeric variables with integer values in the range 0 - 65535.
T is a numeric array with dimension >= 20
n is a BASIC line number.
< > denotes an optional field in an instruction line.

All instructions (except ROM) are preceded by the BASIC reserved word NODE. A NODE instruction cannot be followed by another instruction on the same line. If the name of the destination node I$, in any instruction except NAME, is replaced by "*", then the instruction will be sent to all nodes in the ring.

2. Node Name Instructions

NAME, I$

Enter name for node. This instruction causes the node to initialise the ring or, if ring is in operation, to enter the ring when the next token is received. If the node is already in the ring, the instruction changes the name of the node. A name must begin with an upper case alphabetic character.

LIST

List names of nodes in ring.

DIR, M$, J

If J is the number of a node in the ring, then M$ is equated to the 12 character string which is the name corresponding to J. If J is not the number of a node in the ring, M$ is equated to the null string.3. Mail Instructions

POST, I$, A$

Send string A$ to mail-box of node I$.

MAIL

Display contents of mail-box.

CLEAR <,N>

Clear mail-box. N sets size of mail-box, where N < 8192.

CLEARQ

Step through mail-box with option to clear individual items.

PRINT,ON

Print contents of mail-box and thereafter print items of post as they are received. Entries are deleted from mail-box after printing. The maximum length of an entry is 8192 characters.

PRINT,OFF

Stop printing of mail.

MESSAGE,I$,S$

Send message S$ to node screen in node I$. Does not return to BASIC until message is sent.

4. Memory Transfer Instructions

MSEND,I$,P1,X1,P2,X2,Y

Send memory block with length Y and start address X1 on RAM page P1 to start address X2 on RAM page P2 in node I$.

MRCV,I$,P1,X1,P2,X2,Y

Receive memory block with length Y and start address X2 on RAM page P2 in node I$ and write to start address X1 on RAM page P1.

5. String Transfer Instructions

SEND,I$,A$,B$

Send string A$ to string B$ in node I$.

RCV,I$,A$,B$

Receive string B$ from node I$ and write to string A$.

6. Program Transfer Instruction

PROGRAM,I$ <,RUN> <,LLIST>

Transfer program to node I$. The first optional field causes program to run after loading. The second causes the program to be listed to a printer at the destination node.

7. Ring Operating Instructions

SUSPEND

Suspend ring operation when next token received.

CONT

Continue ring operation.

ENTER

Cause all repeater nodes to enter ring with identifiers equal to their ring numbers.

8. Node Screen Instructions

ON

Switch node screen on.

OFF

Switch node screen off.

CLS

Clear node screen.

9. BASIC Subroutine Instructions

FLAG,I$,N

Set GOSUB flag N in node I$.

GOSUB n

Call subroutine at line number n if GOSUB flag N <> 0.

RETURN

Clear GOSUB flag.

10. Subroutine Call Instruction

CALL,I$,P,X,Y

Call machine code routine on page P at address X in node I$ with parameter Y in register pair HL. Bits 0 -3 of P select the RAM page and bits 4 - 6 select the ROM page. If bit 7 is set, the CP/M RAM configuration is selected.

11. Node Parameter Set Instructions

SET,J,K

Set operating parameters for node. This instruction cannot set bit 7 of the flag bytes below.

EXT,I$,J,K

Set operating parameters for node I$. J and K are 8-bit numbers. J gives the data accept flags and K gives the instruction enable flags.

Accept flags: Bit J 0 = reject 1 = accept Default = FFH

0	Mail and messages
1	Ring mail
2	Strings and memory blocks requested by nodes
3	Strings and Noddy pages sent from other nodes
4	Memory blocks sent from other nodes
5	Calls
6	Programs
7	External parameter set

Enable flags: Bit K 0 = disable 1 = enable Default = 7FH (FFH for master)

0	Post and messages
1	Receive and flag
2	Send strings, memory blocks and Noddy pages
3	Ring packets
4	Set
5	Programs and calls
6	Enter, suspend and continue
7	External parameter set

12. Error Trapping Instructions

ERROR,n

Enable error trapping mode. If an error occurs during a node instruction, a BASIC program jump to line number n is executed.

In error trapping mode, all node instructions which cause data to be transmitted do not pass control to the next program line until the transmission is complete or an error is detected.

RESTORE

Clear error trapping mode.

13. Ring Status Instruction

STAT,T

Write status parameters to one-dimensional numeric array T. The contents of T are as follows:

T index

1	Ring status:	0 = Ring not initialised 1 = Ring in operation 2 = Ring suspended 3 = Ring recovery in progress

2	Number of nodes in ring (1 - 255)
3	Number of sender nodes (1 - 255)
4	Node type: 0 = repeater 1 = sender 2 = master
5	Data accept flag byte
6	Instruction enable flag byte
7	Number of instructions being transmitted
8	Number of instructions being received
9	Last instruction received: sender number
10	Last instruction received: instruction type
11	Error trapping mode ON (1) or OFF (0)
12	Error number
13	GOSUB flag
14	Default baud rate number for channel 0
15	Channel 0 baud rate number
16	Number of items in mailbox
17	Instruction received flag
18	New name flag
19	Non-token packet counter (modulo 256)
20	Token counter (modulo 256)

The baud rate is given by dividing the baud rate number into 19200. The instruction received and new name flags are set to non-zero values when an instruction is received or the name list is changed. They are cleared after a STAT command is executed.

14. RS232 Instructions

BAUD,0,r

Set channel 0 baud rate to r (75 to 19200).If the ring is in operation, the baud rate will not be changed until the node suspends the ring.

BAUD,1,r

Set channel 1 baud rate to r (75 to 19200).

FORMAT,b,s,p

Set data format for channel 1. b is the number of data bits (5,6,7,8), s is the number of stop bits (1, 1.5, 2) and p is the parity (-, 0, +).

IN,A$,N

Read N characters to string A$ from channel 1 input. The length of A$ is set equal to N.

OUT,A$,N

Write the first N characters of string A$ to channel 1 output.

15. Noddy Instructions

RESERVE,X

Reserve X bytes of Noddy program space for Noddy pages sent from other nodes.

SNODDY,I$,"page name"

Send specified Noddy page to node I$.

16. Screen Instruction

SCREEN,A$

Causes 40 * 24 bytes to be read from the video RAM into an already existing string A$. These bytes are the characters currently held in the full text screen.

17. Node Reset Instructions

NEW

Resets the BASIC program variables and pointers, and the sound buffers. This instruction should be used instead of the BASIC instruction NEW when the ring is in operation.

ROM 7

Reset node variables (Node cold-boot).

18. Error Numbers

The error numbers are used in error trapping mode. If an instruction is successfully executed the error number is 0. The error number is given by the STAT command.

Number:	Error Type:
1	RING ERROR
2	MASTER RESET
3	Data transmission error
4	Node is not in ring
5	Bad node name
6	'*' not allowed
7	Command disabled
8	Data rejected
9	Node busy
10	Receiver buffer full
11	Transmitter buffer full
12	Data too large
13	Memory does not exist
14	Variable does not exist
15	Noddy page exists
16	Program already loading
17	Mailbox in use
18	Suspend error

19. Instruction Types

The instruction type code is given by the STAT

command. It indicates the type of the last instruction received by the node.

Code:	Instruction:
1	CALL

19. Instruction Types (cont.)

Code:	Instruction:
2	EXT
3	FLAG
4	MESSAGE
5	POST
6	MSEND
7	SEND
8	SNODDY
9	
10	
11	PROGRAM
12	MRCV
13	RCV
14	DISC