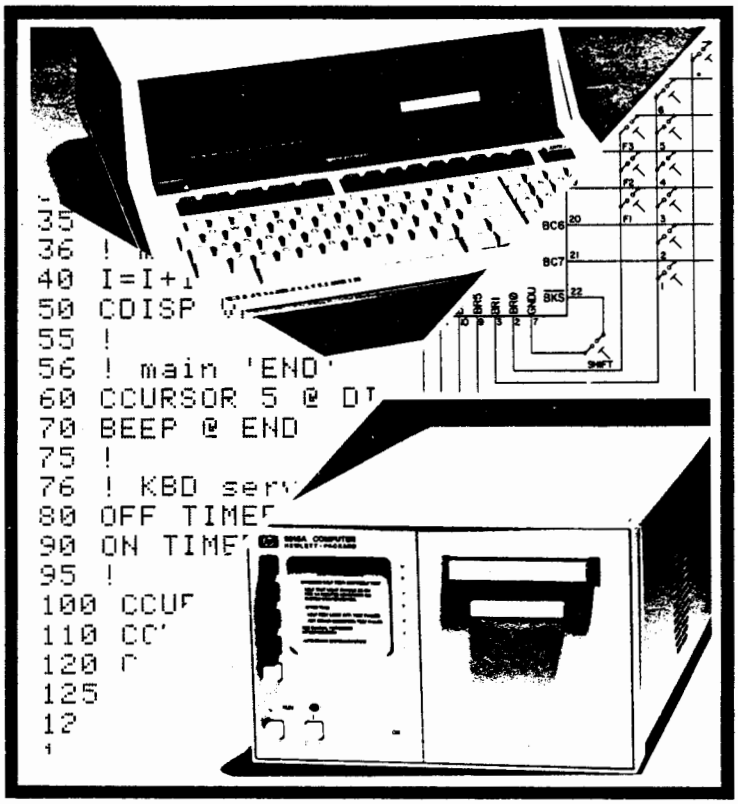
HP 9915 Networking

System Development Technical Supplement



HEWLETT PACKARD

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

HEWLETT PACKARD

Warranty Statement

Hewlett-Packard makes no expressed or implied warranty of any kind, including, but not limited to, the implied warranties of merchantibility and fitness for a particular purpose, with regard to the program material contained herein. Hewlett-Packard shall not be liable for incidental or consequential damages in connection with, or arising out of, the furnishing, performance or use of this program material.

HP warrants that its software and firmware designated by HP for use with a CPU will execute its programming instructions when properly installed on that CPU. HP does not warrant that the operation of the CPU, software, or firmware will be uninterrupted or error free.

Use of this manual and tape cartridge (or flexible disc) supplied for this pack is restricted to this product only. Additional copies of the programs can be made for security and back-up purposes only. Resale of the programs in their present form, or with alterations, is expressly prohibited.

Restricted Rights Legend

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b)(3)(B) of the Rights in Technical Data and Software clause in DAR 7-104.9(a).

HP 9915 Networking

System Development Technical Supplement

Part No. 09915-90022
Microfiche No. 09915-99022

Hewlett-Packard Desktop Computer Division
3404 East Harmony Road, Fort Collins, Colorado 80525

Table of Contents

The connection of two different computers can sometimes cause communication or protocol difficulties. An interface analyzer can be of great assistance during the development of a computer network. The HP 59401A HP-IB Bus Analyzer and the HP 1640 series Serial Data Analyzer were of considerable help in the development of the programs in this technical supplement.

Introduction

There are five ways to get a program into the HP 9915 Modular Computer:

1. Programs on tape via the Option 001 Tape Drive.
2. Programs in EPROM on the standard EPROM board.
3. Programs typed in by an operator on the optional 98155A Accessory Keyboard connected to the 9915 by the Option 002 Operator Interface.
4. Programs loaded from an external mass storage device (such as the 9895 or 82900-series disc drives) with the Mass Storage ROM (p/n 00085-15001).
5. Programs loaded into the 9915 from an external source (normally a computer) brought in through an optional 9915 interface card.

The purpose of this technical supplement is to discuss the last method of getting a program into the 9915 computer. This discussion will include the operations required of both the 9915 and of the "other" computer.

In addition to discussing the transmission of programs, the technical supplement will cover the transmission of data between a 9915 and some other computer.

Although a general understanding of computers will be adequate for most of this technical supplement, it is suggested that you have some familiarity with the HP-85 and the 9915 computers. The manuals to help you are:

- HP-85 Owner's Manual and Programming Guide (00085-90002)
- HP-85 I/O Programming Guide (00085-90142)
- HP 9915 System Development Manual (09915-90010)
- HP 9915 Operator Interface Technical Supplement (09915-90021)
- HP 9915 Tape Duplication and EPROM Programming Software Manual (09915-10011)
- HP-IB Peripheral Installation Instructions (82937-90001)
- HP-IB Installation and Theory of Operation Manual (82937-90007)
- Serial Installation and Theory of Operation Manual (82939-90001)

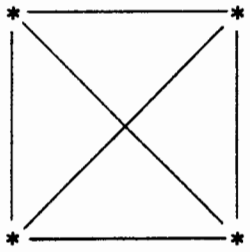
Since this technical supplement deals with networking, there is a very good chance that you will be connecting one or more 9915's to another computer. Before programming or configuring the system, you should read the appropriate manuals of this "other" computer. Some of the manuals that might be appropriate are:

- Introduction Manual
- Configuration Guide
- Operating and Programming Guide
- Data Communications or Terminal Control
- Interfacing or I/O Programming

There are several common networking configurations. The following configurations are only suggestions. Which of these configurations (or some other configuration) is appropriate depends on the actual application.

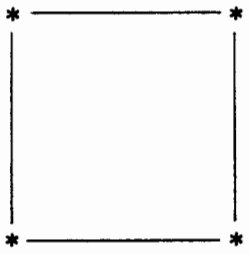
Complete interconnect: Every computer is connected to every other computer by a dedicated interface.

Advantage: Resistant to multiple failures. Efficient communication.
Disadvantage: High cable and interface costs.



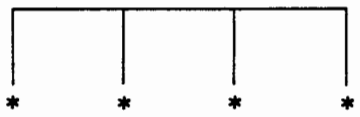
Loop: Every computer is connected to two other computers in a loop configuration.

Advantage: Lower cable and interface costs. Parts of system can survive failures.
Disadvantage: High communication overhead.



Bus: Every computer is connected to every other computer over a single communication path.

Advantage: Lowest cable and interface costs.
Disadvantage: Bus failure can cause system failure.



Chapter 1
Introduction to Networking

What is Networking?

A computer network is a collection of computers and communications equipment that allow a group of computers to cooperate. This group of computers can share items such as work, programs, printers, and plotters.

Some of the primary reasons that a computer network would be used instead of a single computer are:

Resource sharing: Better utilization of peripherals can occur by time-sharing them.

Program sharing: A set of programs can be more easily and quickly updated once to a central file rather than to dozens of different machines.

Fault-tolerance: If one computer in a network fails, the rest can continue functioning.

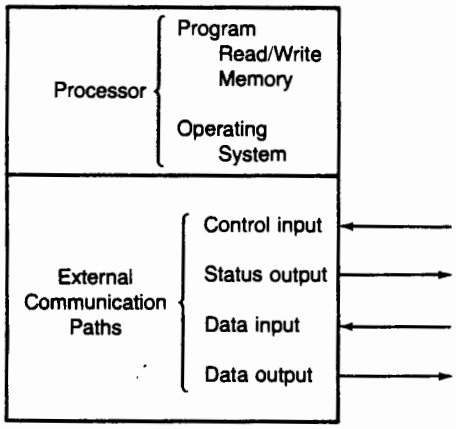
Cost: In many cases, several small computers can cost less than the single large computer required to perform the same set of tasks. (This depends on the tasks being divisible.)

Expandability: By using a modular approach to system design, system expansion is much easier.

Ease of design: In general, it is easier to design several small tasks rather than one large task. Documentation can be easier. Support problems can be minimized.

A Network Computer Model

There is a generalized model of a computer in a networking environment. This is an idealized model. An actual computer in a network can have more or less capability than what is shown here.



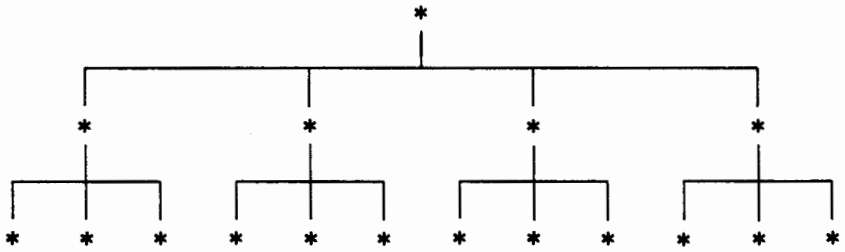
Networking Computer Model

The model is shown as two major sections: the processor or computer and the communication paths to and from the processor. The computer, to function as a node in a network, needs to have sufficient read/write memory for a program to be downloaded into memory by another computer in the network. The networking computer also needs some set of instructions to tell it what to do in the network. This set of instructions resides either in the Operating System of the computer or in an application program that the Operating System will execute automatically.

The communication paths are necessary to transmit data and programs within the network. The status and control communications paths allow another computer in the network to check on node and if the node is not performing properly the other computer can possibly reset or test the node computer.

The model does not show a separate program transmission path because most computers' data interfaces can transmit either programs or data. In general, the program and data are sent via data paths and status and control information is sent via a separate communication path.

The configurations just mentioned deal only with one level of a computer network. It is possible to connect levels of computers together in various ways. One approach is to have a hierarchical configuration; one level is similar to the next in organization.



The term **host** is used here when discussing a computer that controls one or more other computers. **Node** is the term associated with the computer being controlled. Note that in the hierarchical network shown above the middle row of computers have both node and host duties.

Most traditional uses of networking have been in computational and scientific applications. Most of the literature dealing with networking deal with this computational networking. The orientation of this technical supplement is on automation and instrumentation networking.

Automation Networking

Because the orientation of this document is automation networking, it is possible to identify a few operations that are fundamental to this type of networking. These operations are:

From Node to Host

Operation	Example Use
Data Upload	return test results
Program Upload	development phase - get the program working and send it to the host
Remote Status	check the computer for proper operation

From Host to Node

Operation	Example Use
Data Download	pass test parameters
Program Download	pass the program to the node for execution
Remote Control	restart the node if the machine has stopped running

Note that program uploading is normally only necessary during system or application development.

An Overview of the 9915 as a Networking Computer

The 9915 will not automatically download a program from an external computer. A program needs to be running in the 9915 to specify which program to get, where to get it, and in what protocol to receive it.

To start a program download is very simple. A version of it might be:

```
10 PROM IS 10              ! Specify select code 10
20                         ! S.C. 10 specifies the download protocol
30 PLOADGD "FILE"          ! Specify to load "FILE"
40 END
```

This program can easily fit on a single EPROM under the file name "Autost". When the 9915 is turned on it will go first to EPROM and hunt for an "Autost" file. Since there is an "Autost" file, the 9915 will load the EPROM "Autost" program and start running it.

Having this small program in the 9915 on EPROM will give the 9915 the effect of automatically downloading programs.

The previous program will generate a series of communications between the 9915 and the computer at the other end of the interface. The communications will consist of the following transactions:

1. The 9915 sends a request for the directory record.
2. The "other" computer sends the data which contains the 256 byte directory plus a 2 byte checksum.
3. If filename "FILE" exists in the directory the 9915 will send a request for the first record of the file.
4. The "other" computer sends the 256 byte record followed by a two byte checksum. Steps 3 and 4 are repeated for all the records in the file.

When the file is completely loaded, the 9915 starts running the program.

Although there has to be a program in the 9915 to start the downloading process, this can be a very useful approach to networking. Since there is a program running, the program could also contain some of the following:

System Self-test: By checking the "ST RESULT" function, the 9915 could test itself for hardware failures. The 9915 could also check the communication paths and other system hardware for proper operation.

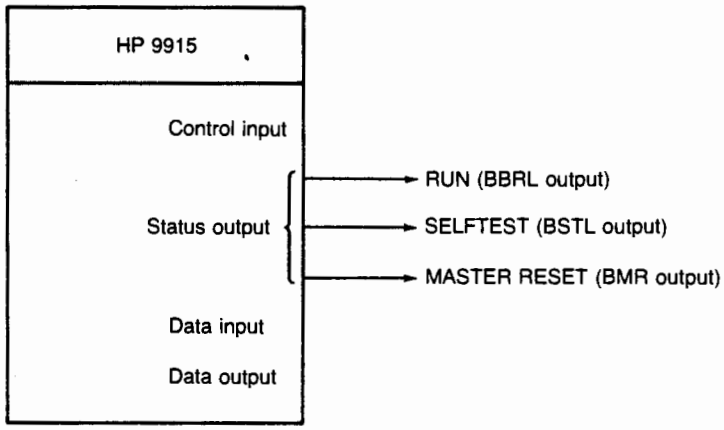
Fault-tolerant Operation: If no program gets downloaded, the 9915 could run a back-up program to keep the operator and equipment working.

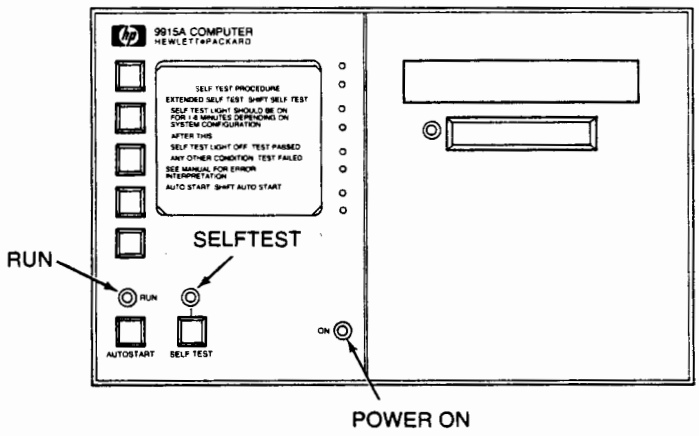
Self-configuration: The 9915 could communicate with the operator, the peripherals, and the "other" computer to determine the system configuration.

The remote status outputs from the operator interface are logically equivalent to the status lights on the 9915A front panel. These lights are:

- program RUNning,
- SELFTEST in progress,
- and power ON.

RUN and SELFTEST are high-true: when a positive voltage appears on these outputs, the condition is true. The complement of power ON, master reset, is brought out. Master reset is passive low when the computer is off.





The control inputs can be attached to switches or to TTL outputs. A status output can be attached through a resistor connected in series with an LED or to a TTL-style input. These lines are on the CONTROL connector of the Option 002 Operator Interface. The Operator Interface Technical Supplement (p/n 09915-90021) contains additional information on using the control and status lines.

Using the ENABLE AS-ST statement, it is possible to prevent the AUTOSTART and SELF-TEST inputs and keys from having any effect on the 9915A.

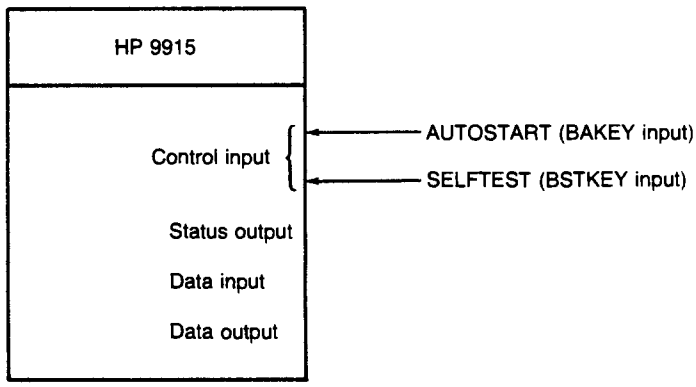
Chapter 2
Hardware Features

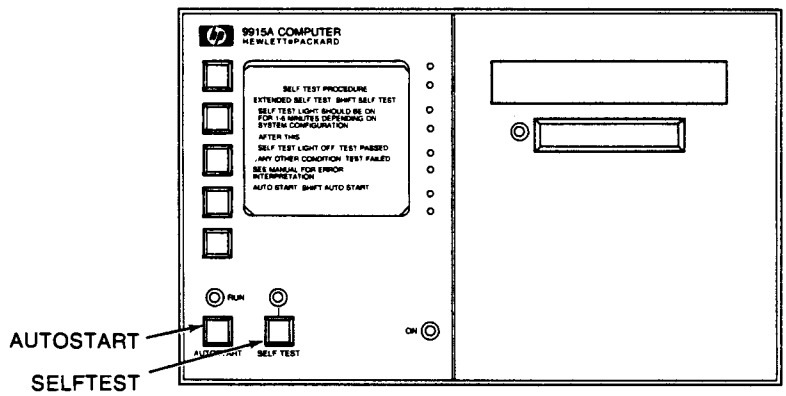
This chapter discusses the hardware features of the 9915A and its interface cards as those features apply to networking applications.

HP 9915A Hardware Features

The 9915A has certain hardware features that permit a system designer to perform remote control and remote status operations. These hardware features exist in the 9915A Option 002 Operator Interface.

The remote control inputs to the operator interface are totally equivalent to the AUTOSTART and SELFTEST buttons on the 9915A front panel. A zero voltage on these lines will cause the specified action (AUTOSTART or SELFTEST).



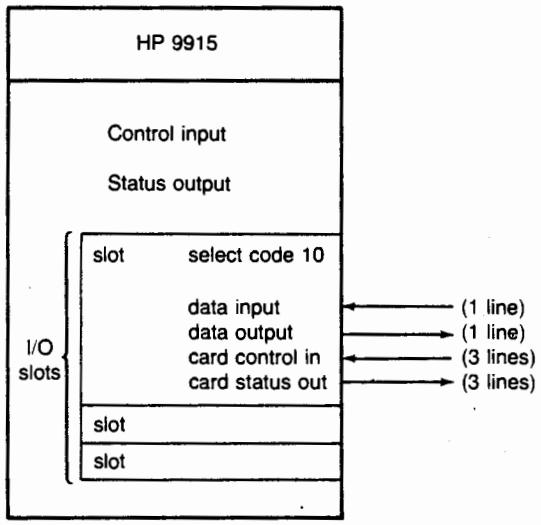


The interface cards are generally used in networking for program and data uploading and downloading. The interfaces are used to connect to instruments and peripherals in non-networking applications. The sections that follow discuss specific features of the interface cards that make them suitable for networking applications. For further information read the HP 85 I/O Programming Guide and the appropriate interface installation manual. The interface cards suitable for networking are:

- 82939A Serial: RS-232, current loop
- 82937A HP-IB: IEEE 488-1978
- 82940A GPIO: general parallel

Serial Interface

The serial interface card implements a bit-serial communication path. The 82939A Serial interface card implements both the voltage-level (EIA) and 20 ma current-loop drivers.



The EIA voltage level is generally used when connecting to various communications equipment a short distance away (tens of meters). This equipment, for networking applications, might be a **modem** (**mo**dulator **dem**odulator). The modem allows the serial data stream to be sent over standard telephone lines. The use of standard telephone lines means that the two devices, communicating via modems, can be large distances apart, limited only by the telephone lines.

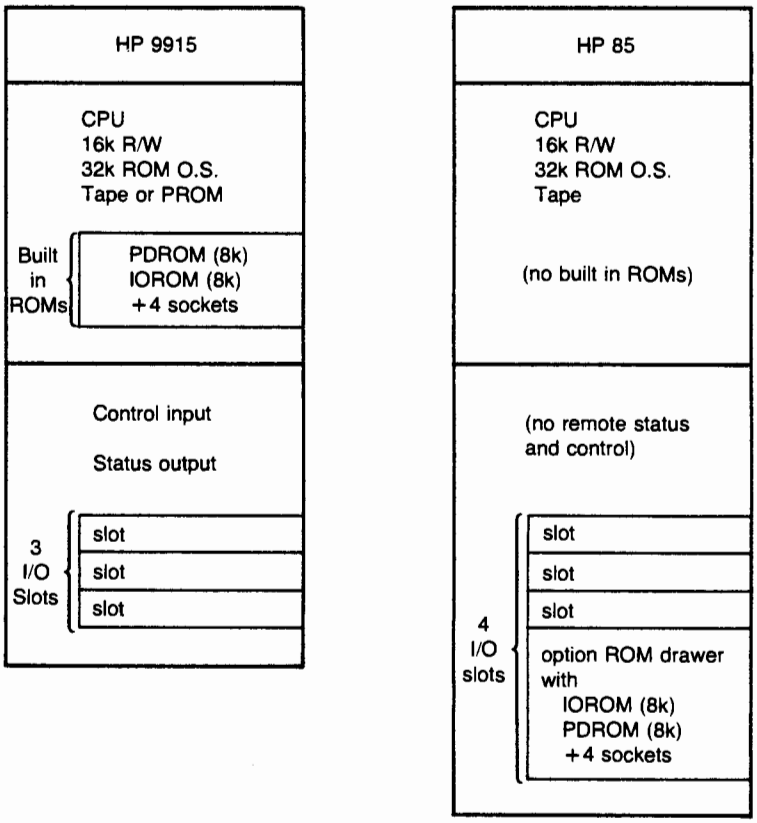
The 20 ma current loop mode is useful when communicating between computers that are in an electrically noisy environment because the signal depends on current flow (or the lack of it) and not on voltage levels.

There are multi-drop protocols for RS-232; schemes where several serial interfaces are connected to the same serial data line. The interface or computer can then decode if the data being sent is intended for this node. The serial interface for the HP-85 and 9915 does not have a "multi-drop" mode.

Interface Cards

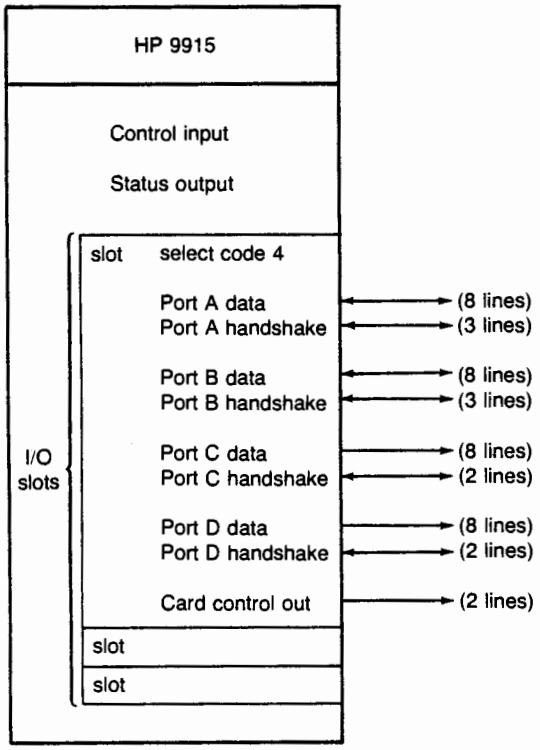
The generalized model of a networking computer showed a data input and a data output path. The optional interfaces available for the HP 9915A are the HP-85 interfaces: the Serial, HP-IB, GPIO (bit parallel), and Binary Coded Decimal interfaces.

The generalized model shows one I/O path. The HP-85 actually has four slots, one of which must contain an option ROM drawer with the I/O ROM. This leaves three slots for I/O cards and the 16k memory module. So, the HP-85 can have up to three interfaces when 16 kbytes of R/W memory is adequate or up to two interfaces when 32 kbytes of R/W is necessary. The 9915A has only three slots in the rear panel, but, the 9915A has the equivalent of a ROM drawer built in (which contains an I/O ROM, a Program Development ROM, and four empty ROM sockets). So, the 9915A has exactly the same number of slots available for interfaces and optional R/W memory as does the HP-85.



GPIO Interface

The HP-85 GPIO interface is a multi-port, general purpose 8-bit or 16-bit interface. It comes with two 8-bit bi-directional ports and two 8-bit output-only ports. The card allows a programmer to use the card as four 8-bit ports, two 16-bit ports or some combination or 8-bit and 16-bit.



The GPIO interface is a very flexible, powerful interface card for general interfacing applications which do not conform to the various interface standards. It is not particularly well suited for networking types of applications. The following table shows that, the HP-IB interface is a much better interface for networking.

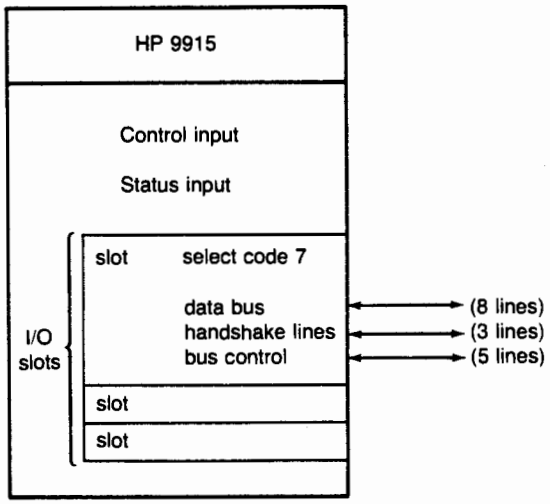
	GPIO	HP-IB
addressability:	4	31 (14 physical)
distance:	4 metres	20 metres
connector:	must be wired	standard, stackable
speed:	18 kbytes/sec	5 kbytes/sec

It is being mentioned for its uses in remote status and remote control. The GPIO interface could be used by one HP-85 or 9915 to control the remote status and remote control lines of up to eight HP 9915s. Using an HP-IB interface (for program and data upload and download) and a GPIO interface (for remote status and remote control) provides for good control over a network of HP 9915As.
The recommended use for the GPIO interface is for use as a status and control interface on an HP-85 or 9915 that is a host computer.

So, the recommended use of the serial interface is for point to point communication of programs and data.

HP-IB Interface

HP-IB is Hewlett-Packard's implementation of IEEE 488-1978. This IEEE 488 standard provides for the physical and electrical specifications for a general purpose, multiple device interface bus. So even though the interface only takes up one I/O slot, it can access 14 physical (31 logical) devices. This multiple device access is achieved through an addressed, bi-directional data bus.



For additional information on how the HP-IB bus works, read the HP-85A I/O Programming Guide.

One of the main features that lends the HP-IB interface to networking applications is the addressed nature of the bus. Several devices can be connected together.

One of the limitations of the HP-IB interface is that only 14 physical devices can be connected to a bus. This limitation is not peculiar to the 82937A interface but is a part of the IEEE 488 standard. Another limitation is that the entire bus cable length must be within 20 metres from end to end with no more than two metres per device. This is specified as part of the IEEE standard. It is possible to use optical or electrical bus extenders to circumvent the interface standard limitations, such as the HP 59403A Common Carrier Interface, the HP 12050A Fiber Optic Link, and the HP 37201A and 37203A HP-IB extenders.

So, the recommended use of the HP-IB interface is for multiple point communication of programs and data.

Program Development ROM

The 9915A Program Development ROM is standard in the 9915 computer and comes standard in the 98150A program development kit for the HP-85. This ROM provides various language enhancements including program downloading.

In a non-networking application, the PD ROM will load a program from the internal EPROM storage or from the optional tape drive. This program loading is very similar to normal HP-85 program loading. The 9915 will automatically try EPROM first and then the tape drive for an autostart program with the file name "Autost".

For the 9915 to download a proram from another computer you need to give the 9915 enough information about the program:

Interface select code: selects which interface is used to down load the program; selected by the PROM IS statement.

Download protocol: selected by the interface select code; odd selects binary protocol, even selects ASCII prototcol.

Program name: a six character name selected by the PLOADGO statement.

This information is communicated to the 9915 in the form of a program or a series of statement executions. To download a program, one of the following steps must occur:

1. The computer operator types in the statements to the 9915 via an optional keyboard attached via the Option 002 Operator Interface.
2. An "Autost" program on EPROM.
3. An "Autost" program on tape in the Option 001 Tape Drive.

Important

If there is no "Autost" program on EPROM or tape, the computer will halt with an error, "ERROR 48 ON LINE 9999: END". This indicates that there is no program for the 9915 to run. The 9915 will NOT automatically go to an interface card and try to download a program.

Chapter 3
Language Features

Introduction

This chapter discusses the language features of the 9915 that enable it to operate in a networking environment. This discussion includes an examination of the protocol and communication between the 9915 and the "other" computer in a network.

The three sets of language features that enhance the 9915's networking abilities are in the Program Development ROM, TAPDUP binary and the I/O ROM. The PD ROM[1] enables the 9915 to download a program from another computer through an interface card. The TAPDUP binary program allows access to 9915A or HP-85A tape files for program uploading. The I/O ROM allows for control over interface cards and is used for data upload and download.

The following table indicates what software or language capabilities are used to access each of the six networking operations:

Operation	ROM/Binary	Access
Control Input	PD ROM[1]	---None--- The PD ROM statement ENABLE AS-ST can disable these inputs.
Status Output	PD ROM[1]	---None--- The outputs cannot be controlled by the programmer.
Program Download	PD ROM	PLOADGO statement - will cause a program to be downloaded.
Program Upload	TAPDUP	READ RECORD IMAGE$ function - to read program File Information.
	TAPDUP	OPEN IMAGE statement - set-up for reading program files.
	I/O ROM	TRANSFER or OUTPUT statement - to send data to another computer.
	PD ROM	CHECKSUM$ function - can be done without the PD ROM, but it's slower.
Data Download	I/O ROM	TRANSFER or ENTER statement
Data Upload	I/O ROM	TRANSFER or OUTPUT statement

1 The PD ROM is necessary for control inputs and status outputs to work properly.

Statement	Action taken by the computer
PLOADGO	1. SCRATCH the current program, binary and common variables. 2. LOAD the specified program from the PROM IS device. 3. Perform a RUN, starting program execution.
PCHAIN	1. SCRATCH the current program. The binary program (if any) and the common variables (if any) are left intact. 2. The specified program is loaded into memory from the specified PROM IS device. 3. Perform a RUN, starting program execution.
PLOADBIN	1. The computer loads the specified binary program from the specified PROM IS device.

The Program Development ROM allows you to specify the PROM IS device by the PROM IS statement. At power on time, the PROM IS device is set to the internal EPROM. The syntax and parameters for PROM IS are as follows:

```
PROM IS device
```

where the device parameter can be:

0	To select internal EPROM; same as PROM.
PROM	To select internal EPROM; same as 0.
-1	To load from the optional tape drive; same as TAPE.
TAPE	To load from the optional tape drive; same as -1.
3 thru 10	To load from the specified interface.

Select codes 1 (CRT) and 2 (HP-85 internal printer) are not allowed.

Downloading

The download operation occurs through a series of data transfers between the node and host. The node will issue a series of requests for records to the host computer. The host will send the appropriate information after each request. The data that is sent from the host includes a file directory and the records that comprise an internal-form program.

The term **internal form** refers to the way the program information is sent. A program that is human-readable is called **source form**. A source-form program is a sequence of ASCII characters (what a programmer would type into a computer).

source form:

```
10 FOR I=1 TO 10 ᶜᴿ
20 DISP I  ᶜᴿ
30 NEXT I  ᶜᴿ
40 END  ᶜᴿ
```

ᶜᴿ means a carriage return or "END LINE" character

The internal-form version of this program is the "tokenized" or "parsed" program as it resides in computer memory or on a mass storage device (tape, flexible disc, PROM) as a PROG file. The terms tokenized and parsed refer to the way that the 9915A takes an ASCII-character program line and turns it into its machine-interpretable representation. The previous program segment might look like the following string of characters in memory or on the mass storage device:

Internal-form string:

```
MAIN.. '<..............
oVbsI@xeBrI@sdI "ew$*
"„ I@K×oI@
```

This internal form string consists of several different pieces of information. The "MAIN" indicates that the program segment is a main program and not a subprogram module. The nulls, shown here as the character ".", are the value area for the variables. This value area contains a pre-allocated storage area for the variables. The last section contains the actual program tokens. For further information on internal form programs, refer to the Assembly ROM documentation, (p/n 00085-90444).

As stated in the introduction to this chapter, the PLOADGO statement causes a program to be downloaded. There are actually three ways that the 9915 will download information:

The directory that gets downloaded looks just like the directory of the internal EPROM. The directory contains up to 32 file names of file types PROG and BPGM: program and binary program. The layout of the EPROM directory is:

File name:	6 characters
File type:	1 byte
Record number:	1 byte

The file type information byte contains a decimal 32 for stored programs and a decimal 8 for binary programs.

Important

The tape catalog and the PROM directory (and external PROM directory) are similar but NOT identical.

ASCII Protocol

An example program segment in the 9915A that will cause a program download might look like:

```
40 PROM IS 10  ! even select code selects ASCII protocol
50 PLOADGO  "MYFILE"!load the file "MYFILE"
```

The first request after execution of the PLOADGO is for the external directory. For the purposes of this example, MYFILE is a program file located immediately after the directory starting with record # 1. Because the program is so small, it takes only one record.

The program in MYFILE is the internal-form program version of:

```
10 DISP  "MYFILE GOT LOADED"
20 END
```

The ASCII download follows a handful of rules for both the node and the host computer. These rules are:

- A request from the node is indicated by an ASCII "R" (character 82) followed by the octal constant indicating the byte starting address of the record being requested.
- The data response from the host is preceded by an ASCII "D" (character 68) followed by 258 octal constants which comprise the record and the checksum. All characters preceding the "D" are ignored.
- The octal constant consists of any number of leading zeroes followed by up to three octal digits. An octal digit is defined to be the characters "0" through "7". The octal consant is terminated by a non-digit character. The octal termination character can be a comma, decimal point, line-feed, etc.

Protocols and Directories

When the PROM IS device has been set to an interface select code (via the PROM IS statement) one of two down-load protocols will be in effect; binary protocol or ASCII protocol, as determined by the select code. Even select codes (4, 6, 8, 10) will force ASCII protocol, while odd select codes (3, 5, 7, 9) force binary protocol.

The purpose for the different protocols is to allow easy connection to whatever computer you are networking with. The binary protocol is a fast and efficient protocol where each 8-bit byte of download information is sent as a single 8-bit byte. The ASCII protocol uses at least four 8-bit bytes to send each 8-bit byte of program information.

Independent of the protocol, the same type of information is exchanged. The node 9915 will generate requests for blocks of data by giving the starting byte address of the record to be requested. The host computer will respond by sending the appropriate data and a checksum.

Checksums are extra bytes sent with the data to insure that the message was sent properly. The node 9915 will perform a checksum algorithm on all the data up to the checksum and compare the transmitted checksum to the calculated checksum. If there is any difference, a transmission error has occurred and retransmission of the record is requested. See the Appendix for additional information on checksums.

The external PROM IS device is considered by the 9915 to be an external version of the internal EPROM. This means that the external device must respond to requests and look the same as the internal EPROM does. The internal EPROM (and therefore the external device) contains the file information in the following mapping:

Byte Address	Record Address	Data
0	0	directory
258	1	file data
516	2	file data
774	3	file data
•	•	•
•	•	•
258*n	n	file data
•	•	•
•	•	
•	•	•
32508	126	file data

The download firmware in the 9915 and the internal EPROM recognize 15 bits of address; so only 32 768 bytes are allowed in internal EPROM or external PROM IS devices. Note that each record contains 258 bytes: 256 bytes of actual data and 2 bytes of checksum information. This extra 2 bytes per record means that only 127 records fit into the 32 768 bytes.

```
REQUEST FOR RECORD 1
FROM NODE TO HOST

                                R000402 CR LF

RECORD 1
FROM HOST TO NODE

                                D115,101,111,116,000,000,140,074
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,020,000,027,126,005,021,115,131
                                ,106,111,114,105,040,107,117,124
                                ,040,114,117,101,104,105,104,360
                                ,354,016,040,000,002,212,016,231
                                ,251,002,031,018,002,000,040,040
                                ,040,040,040,040,040,377,012,011
                                ,040,040,040,040,040,040,040,377
                                ,212,020,022,000,022,000,377,377
                                ,040,040,040,040,040,040,040,040
                                ,040,040,040,040,040,040,040,040
                                ,040,040,122,040,322,303,046,005
                                ,011,040,117,116,040,114,111,116
                                ,105,040,046,323,303,046,005,021
                                ,040,104,125,122,111,116,107,040
                                ,104,117,127,116,114,117,101,104
                                ,040,046,007,016,160,011,004,132
                                ,043,003,016,231,251,002,031,016
                                ,312,000,212,015,100,000,100,000
                                ,377,377,040,040,040,040,040,040
                                ,040,040,040,040,040,040,040,040
                                ,040,040,040,040,040,040,040,040
                                ,040,040,040,040,040,040,040,040
                                ,040,040,040,040,040,040,040,040
                                ,040,040,040,040,040,040,040,040
                                ,040,040,040,040,040,040,040,040
                                ,040,040,212,006,022,000,022,000
                                ,377,377,040,040,040,040,040,040
                                ,040,040,040,040,040,040,040,040
                                ,265,307,
```

The carriage return/line-feed characters are the end-of-line terminators from the 9915 interface card. These characters can be changed into any character sequence (up to seven characters). Refer to the HP-85 I/O Programming Guide for further information on end-of-line sequences.

Note that there are 256 + 2 octal numbers sent for each record. These additional numbers are the checksum. The checksum consists of a two byte IBM SDLC checksum. The checksum is transmitted as 2 octal numbers. Refer to the Appendix for additional information on checksums.

So, given the preceding information, the communication between the host and node computers will look something like the following for ASCII protocol:

```
REQUEST FOR RECORD 0 FROM
NODE TO HOST
                                R000000 CR LF

RECORD 0
FROM HOST TO NODE
                                D115,131,106,111,114,105,040,000
                                ,040,001,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,000,000,000,000,000,000,000,000
                                ,317,112,
```

The communication between a host and node for a binary protocol download will look something like the following:

```
REQUEST FOR RECORD 0
FROM NODE TO HOST
                        00000001 b
                        00000000 b
                        00000000 b
                        00001101 b
                        00001010 b

RECORD 0
FROM HOST TO NODE
                        00000010 b
                        MYFILE.....................
                        ...........................
                        ...........................
                        ...........................
                        ...........................
                        ...........................
                        ...........................
                        ...........................
                        01100111 b
                        00100101 b

REQUEST FOR RECORD 1
FROM NODE TO HOST
                        00000001 b
                        00000001 b
                        00000010 b
                        00001101 b
                        00001010 b

RECORD 1
FROM HOST TO NODE
                        00000010 b
                        MAIN..'<.....................V..MY
                        FILE GOT LOADED... ........... ...
                             ....      ........
                                    R..&.. ON LINE &..&..
                        DURING DOWNLOAD &..P..Z*.......
                        ....@.@...
                                ........
                        01011010 b
                        01100011 b
```

This download protocol contains the exact same data records and checksums as the ASCII protocol. Notice that the directory is mostly empty. Also note that the record containing the program has some leftover data from some previous program (the ON LINE and DURING DOWNLOAD information). These bytes were left in memory when a new program was loaded. When the MYFILE program was stored, these extra bytes were also stored and are extraneous.

The request for record 0 caused a "R000000" but the record 1 request generated "R000402" request. Octal 402 is equivalent to decimal 258. What is happening is that the node is requesting the actual byte address in PROM of the requested record, where each record is 256 bytes long.

Note that the 82939A serial interface card comes from Hewlett-Packard with a select code switch setting of 10. The serial card can be just as easily set to 9 or 3, causing binary protocol to be used.

Binary Protocol

Odd select codes (3, 5, 7, 9) will cause the binary protocol for downloading to be used. The following example is very similar to the previous example. The only difference is in the PROM IS statement and in the data that represents the program. The final effect of both methods is the same: the specified program is loaded and execution begins.

The program segment in the node 9915A that causes a binary protocol download might look like:

```
40 PROM is 7   ! Odd select code causes binary protocol
50 PLOADGO "MYFILE"
```

The rules that the node and host follow for a binary protocol download are as follows:

- A request from the node is indicated by an ASCII "SOH" (character 1) followed by two bytes (most significant byte first) which are the absolute binary representation of address of the requested record.
- The data response from the host is preceded by an ASCII "STX" (character 2) followed by 258 bytes which are the actual data record and the 2-byte checksum. All bytes preceding the "STX" are ignored.

Tape Duplication Binary Program

The TAPDUP binary program comes with the 9915 Tape Duplication and EPROM Programming Software Pack (p/n 09915-10010).

The 9915A Tape Duplication binary program (TAPDUP) provides for the ability to upload programs from an HP 9915A or an HP-85A. The tape duplication binary is used in non-networking applications as a tape duplication facility. The TAPDUP binary is used to read program and binary program files on tape (only tape, not from flexible disc drives) as string information. This string data can then be transmitted or processed.

The Tape Duplication and EPROM Programming Software provides a utility program called IMAGE. This program takes the programs that you specify and puts them onto a series data file as PROM format records. The IMAGE program also takes care of generating the PROM directory in record number 0. Once these data records are created, the data files only need to be transmitted to the host for later downloading.

The program to transfer the data records from the 9915 to the host is called SEND and is in Chapter 5. The program to receive the data from SEND is called UPLOAD and is in the host computer chapters (Chapters 6, 7, etc.).

Tape Catalog vs. PROM Directory

As stated previously, the tape catalog and PROM directory are similar but contain differences. The following table illustrates the differences between the PROM directory and tape catalog:

	TAPE Catalog	PROM Directory
Total Bytes:	512	256
Number of Files:	42 maximum	32 maximum
File Types:	all PROG, BPGM, DATA, others	only PROG and BPGM
File Entry Format:		
name	6 bytes	6 bytes
file number	1 byte	
file type	1 byte	1 byte
bytes/rec	2 bytes	
# records	2 bytes	1 byte
Total Bytes/File:	12 bytes	8 bytes

As stated in ASCII protocol, the carriage return/line-feed characters can be changed to an arbitrary end-of-line sequence.

If a record gets garbled in transmission, the node 9915 will detect the error through the use of the checksum. When an error has occurred, the node will try 7 more times to get the bad record by requesting the same record. The 9915 will generate an `ERROR 116: PROM or I/O read` after eight bad record transmissions. This re-transmission will occur on both ASCII and binary protocol.

Why Bother with ASCII Protocol?

The question arises: why use ASCII protocol over BINARY protocol? There is no difference with regard to error detection. They both have the same checksum calculations. So what are the reasons for using one or the other?

The ASCII protocol should be used with a computer system that is incapable of easily generating or receiving arbitrary data streams. A typical record for downloading a program to the 9915 will contain a whole series of control characters. These characters will cause very unusual things to happen with the computer or its communication subsystems. The ASCII protocol can be used without transmitting a single control character. Note that ASCII protocol takes at least four characters for every 8-bit byte of download information.

Binary protocol should be used in all situations where the host computer can send and receive arbitrary control character sequences. Binary downloads will occur much faster since there is only one character per byte of download information (reducing transmission time) and since there is no need to convert from octal to binary representation.

Note that the previous examples were for a single numeric value. When OUTPUT is used to send several values, it will only place blanks between the values. For ENTER, however, the blank is not a numeric terminator. To send a large number of values, the following will work:

send:
```
100 OUTPUT 10 USING 110 ; A,B,C
110 IMAGE ",",K
```
or with IMAGE:
```
110 IMAGE K,/
```
receive: ENTER device selector; numeric variable list

The ENTER does not change because the OUTPUT is generating the non-numeric (a comma, a $^{C}_{R}$ $_{L}_{F}$, or whatever) btween the data. It is possible to use fixed field images to send the data such as:

send:
```
700 OUTPUT 4 USING 710 ; A, B, 12.34
710 IMAGE #,DDDD,8D,SDD.DDE
```

receive:
```
550 ENTER 4 USING 560 ; A, B, C
560 IMAGE #,DDDD,8D,SDD.DDE
```

These images need to match.

When sending arrays of numeric information, it will be necessary to use a FOR/NEXT loop. Since there will be an OUTPUT statement for each element, it is possible to use the default image (since it sends a $^{C}_{R}$ $_{L}_{F}$ after each one). An example of transferring an array might be:

send:
```
30 FOR I=1 TO 100
40 OUTPUT 10 ; A(I)
50 NEXT I
```

receive:
```
110 FOR J=1 TO 100
120 ENTER 10 ; Q(J)
130 NEXT J
```

All the restrictions described above about single numeric values still apply.

Input/Output ROM

The HP-85 I/O ROM provides many of the software features necessary for data transfers. The I/O ROM comes standard with the 9915.

The primary tools of the I/O ROM to be used in data transfers are the ENTER, OUTPUT statements. ENTER and OUTPUT are the formatted data transfer mechanisms and use normal handshake for the transfer. Another statement, TRANSFER, provides access to interrupt and fast handshake transfers to and from IOBUFFERs. Refer to the HP-85 I/O Programming Guide for details.

When dealing with mass storage transfers the programmer does not need to know exactly the form and type of data being sent. The mass storage firmware of the HP-85A and 9915A will automatically keep track of the data type, end-of-record and end-of-file information.

For I/O data transfers the programmer must know exactly the form and type of data being sent. The data typing and termination operations must be handled by the programmer when using the I/O ROM.

The sections that follow will show some typical approaches to common data transfers. For a better understanding of I/O, be sure to read the HP-85 I/O Programming Guide. In particular, read Section 2: Simple I/O Operations, and Section 3: Formatted I/O Operations for a discussion of ENTER and OUTPUT. For information on TRANSFER and end-of-transfer branching read Sections 8 and 9, Specialized Transfers and End Of Line Branching.

Transferring Numeric Data

The normal approach to sending numeric data is to use ENTER and OUTPUT. Using default formats (or USING with "K") the numeric values will be sent as a string of ASCII digits. The termination of the numeric value occurs when a non-numeric digit is received. So the statements to send and receive numeric data are as follows:

send: `OUTPUT` device selector `;` string expression, variable
receive: `ENTER` device selector `;` string variable

It is recommended to be consistent with regard to data types. For example, sending a REAL number −3.577 E +55 into an INTEGER variable will generate an error. Remember that sending a 9915A or HP-85A REAL variable +1.234 E +498 to a 9835 or 9845 REAL variable (with exponent precision of + 99) will also generate an error. (Note there is a special fixed field image, "e", for sending 2-digit exponent information; this allows the programmer of the 9915 or 85 to trap the error for out of range numbers.)

fixed field image, "e", for sending 2-digit exponent information; this allows the programmer

string expression allows for a program to determine the number of characters as it executes. The following program segments show how this can be done:

```
send:    100 S=LEN(B$) !       determine # characters in string
         110 OUTPUT 701 ; S  !    tell other computer how many chars
         120 OUTPUT 701 USING " #,K "  ; B$  !  this will send only the char-
             acters in B$
         130 !
receive: 300 ENTER 7 ; S !           get the number of characters
         310 I$= " #, " &VAL$(S)& " A "  !     build image using value of S
         320 ENTER 7 USING I$ ; B$ !    get the characters
```

Special Transfers

The I/O ROM allows for some advanced and high-performance data transfer techniques. These transfers are accessed by the TRANSFER statement. The TRANSFER statement allows for interrupt or fast handshake buffer transfers.

The term "buffer" means that the data must be formatted into an explicit I/O buffer by the IOBUFFER statement.

The interrupt type of TRANSFER allows for a buffer full of data to be sent or received while the program continues executing. This is the way that a 9915A or an HP-85 can achieve overlapped I/O and computation. An example of this might be:

```
10 DIM B$[1008] ! declare 1000 byte buffer
20 IOBUFFER B$ ! turn it into a buffer
30 B$[1000]="x" ! fill it with dummy data
40 ON EOT 7 GO TO 80 !  set up end of xfr branch
50 TRANSFER B$ TO 701 INTR !  send the data
60 DISP "WAITING FOR EOT" ! other computation and some I/O
70 GOTO 60 !  can happen where this loop is
80 DISP "TRANSFER IS DONE"
90 END
```

The fast handshake mode of TRANSFER provides the fastest method of data transfer. The HP-IB interface can go up to 25kbytes/second in TRANSFER fast handshake. Note that although the Serial I/O card does not support fast handshake, it can still handle 9600 baud with TRANSFER INTR.

Transferring Binary Data

The normal approach to sending binary data is to use ENTER and OUTPUT with a "B" or "W" image. An example of sending 3 bytes and 2 words (double bytes) is as follows:

```
send:      330 OUTPUT 720 USING "#,3(B),2(W)" ; A,B,255,D,32767
receive:   190 ENTER 7 USING "#,3(B),2(W)" ; R,X,Y,Z,W,
```

Note that the images have to match. The receiving computer does not have any idea whether the data is binary, string, or numeric. The "#" turns off the search for a line-feed at the end of ENTER and turns off the sending of $^{C}_{R}$ $_{L}F$ at the end of OUTPUT. It is used here because there is no need for the default termination conditions.

Transferring String Data

The normal approach to sending string data is to use ENTER and OUTPUT. Using default format (or USING with "K") the string is defined as being terminated with a carriage return and a line-feed. So the statements to send and receive strings (where the string does not contain a line-feed) of any dimensioned string length are:

```
send:      OUTPUT device selector ; string expression , variable
receive:   ENTER device selector ; string variable
```

The OUTPUT statement will send the string just as it exists in R/W memory; however, a carriage return and line feed will be sent at the end of the statement. The enter statement has several terminating conditions for the string. These conditions are:

- receipt of $_{L}F$
- receipt of $^{C}_{R}$ $_{L}F$
- string size reached

Note that these are the conditions that stop entry of data into the string; this does not necessarily stop the ENTER. If the string size was reached without a line-feed, the ENTER statement will continue hunting for a line-feed terminator.

When sending string data where the exact number of characters is known, it is possible to use an image. By doing this, it is possible to read binary data. An example of this is:

```
send:      OUTPUT device selector USING "#,nA" ; string expression , variable
receive:   ENTER device selector USING "#,nA" ; string variable
```

The n indicates the number of characters to be sent. This is fine when the number of characters is known at program development time. The fact that the USING image can be a

Get/Save Binary Program

The Tape Duplication and EPROM Programming Pack (p/n 09915-10010) that is included in the 98150A Program Development Kit, includes a Get/Save Binary Program. This binary program allows you to deal with programs on tape (and tape only) in source form. When you save a program it is stored in ASCII. You can create a program on another machine and then transfer it to the 9915, put it on tape in a data file, and then get it off of tape.

The question comes up: why not transfer a program from a host computer to the 9915 onto tape and then get the program? The reason that this CAN NOT be done is that the Get/Save binary program is not programmable; you cannot put a GET statement in a program. Since GET and SAVE commands can only be executed from the keyboard, you would have to have an operator and keyboard at the computer to download a program.

The other technique for more large programs is to tell the host which program is desired. The PLOADGO, PCHAIN, PLOADBIN statements do not tell the host which file is being loaded. They request the directory and check it for the file name. So the way to do this might look like:

```
450 •
460 •
480 •
490 OUTPUT S ; "File"
500 PROM IS S
510 PLOADGO "File"
520 •
530 •
540 •
```

Note that this technique gives you an arbitrary number of programs because the limitation is the requested file name. The host will take the file name and select which of several EPROM directories and EPROM data to send to the node.

Important

These techniques to get around the 32 768 byte limit are shown as examples and are not supported in the host and node example programs. You must modify the programs.

Chapter 4
Advanced Concepts

This chapter discusses some of the features and techniques on the 9915 that enable it to operate beyond what has been discussed in the preceding chapters.

Many Large Programs

The internal PROM and external PROM IS device can only directly access 32 768 bytes of program storage. This 32 768 bytes can be used as you desire; from one very large program up to 32 smaller programs. There are going to be applications where there is the demand for up to 32 (or more) large programs. There are ways the 9915 can be "tricked" into accessing larger memory.

One approach to more large programs is to build a directory for the external PROM IS device that has up to 32 program files on it. Independent of size, each program or binary is only allocated one record. So an example of the directory might look like:

Program Name	Type	First Record
PROG.1	prog	1
PROG.2	prog	2
PROG.3	prog	3
BPGM.1	bpgm	4
TEST.1	prog	5
•	•	•
•	•	•
•	•	•
LAST.9	prog	32

In actuality, PROG.1 contains 5 records, PROG.2 contains 99 records, and so on. The 9915 will request the appropriate starting record and then take the number of records specified in that starting record. It is necessary for the host to have the downloading program recognize that it needs to take the proper records and transmit them to the node. The host will be keying which program to down load off of the requested record. It then does not use the requested record number except as an error indication (a record can be re-requested if it was not properly received by the node). Note that this technique still limits you to 32 programs of any size.

The other technique for more large programs is to tell the host which program is desired. The PLOADGO, PCHAIN, PLOADBIN statements do not tell the host which file is being loaded. They request the directory and check it for the file name. So the way to do this might look like:

```
460 •
460 •
480 •
490 OUTPUT S ; "File"
500 PROM IS S
510 PLOADGO "File"
520 •
530 •
540 •
```

Note that this technique gives you an arbitrary number of programs because the limitation is the requested file name. The host will take the file name and select which of several EPROM directories and EPROM data to send to the node.

Important

These techniques to get around the 32 768 byte limit are shown as examples and are not supported in the host and node example programs. You must modify the programs.

I/O Card Features

It is possible to use some of the interface card lines for purposes other than what is normally intended.

HP-IB Interface

The HP-IB interface has several types of messages that have a pre-defined meaning. The two types of messages are **uni-line** (meaning that the message occurs when a single signal line is set) and **multi-line** (meaning that the message is a data byte transmitted on the HP-IB data bus with the ATN line true).

Some of these messages that might be of some use in networking applications are:

Message Name	Message Type	HP-IB Mnemonic	Possible Use
Interface Clear	Uni-line	IFC	Restarting node
Universal Device Clear	Multi-line	DCL	Restarting node
Selective Device Clear	Multi-line	SDC	Restarting node
Service Request	Uni-line	SRQ	Indicating need for service download or other needs

The technique for using the HP-IB messages to re-start the node computer is to generate a BASIC program interrupt on the specified condition and execute the AUTOSTART statement. An example program might look like:

```
    •
    •
    •
110 ON INTR 7 GOTO 900
120 ENABLE INTR 7 ; BTD("10000100")
130 !    enable for IFC and DCL/SDC interrupts
    •
    •
    •
900 STATUS 7,1 ; A
910 IF BIT(A,7)=1 THEN AUTOSTART
920 IF BIT(A,2)=1 THEN AUTOSTART
930 !   wasn't re-start
940 ENABLE INTR 7 ; BTD(''10000100'') @ RETURN
    •
    •
    •
```

Refer to the HP-85 I/O Programming Guide for more information about HP-IB and its capabilities. Refer to the general HP-IB device program in Chapter 5 for a skeleton program that can be customized for use in an HP-IB system.

Serial Interface

The serial interface for the 9915 has a set of control and status lines that are normally used for modem control. Just as the HP-IB interface lines can be used for networking control, the serial interface lines can be used. Some of the lines that are available are:

Name	Pin	Mnemonic	Direction
Request to send	4	RTS	into 82939
Clear to send	5	CTS	from 82939
Data set ready	6	DSR	from 82939
Carrier detect	8	CD	from 82939
Data terminal ready	20	DTR	into 82939
Data rate select	23	DRS	into 82939

These lines (and directions) are for the 82939A standard serial interface. The Option 001 reverses the directions. Refer to the 82939A Serial Installation and Theory of Operation Manual (p/n 82939-90001) for additional information.

The approach for generating the interrupts is identical to the HP-IB example. The only part of the program that needs modification is the ENABLE INTR mask and the select code.

The HP-85 as a Networking Computer

This technical supplement discusses the 9915 as the node networking computer. Fundamentally, the 9915 is an HP-85. So the HP-85 has all of the networking features of the 9915 with the following exceptions:

- The HP-85 does not have the control and status lines (AUTOSTART, SELFTEST, RUN light) that the 9915 has (in the Option 002 Operator Interface).
- The HP-85 does not have the internal PROM of the 9915. The 9915 does have a PROM drawer for use in designing customer Option ROMs. This PROM drawer is totally different in use and intent compared to the 9915 internal PROM. Note that this PROM drawer (used with the Assembly Language ROM) can be used on the 9915 in addition to the internal PROM.

With these exceptions, the HP-85 can download programs and use all of the networking programs described for the 9915 in this document.

Chapter 5
HP 9915 Networking Programs

Introduction

This chapter discusses and lists some networking related programs for the 9915. These programs are for an HP-85 or 9915 when it is the node computer.

The programs discussed are:

NODE	General purpose program for use as an autostart program on EPROM or tape.
IMAGE	Utility program to take program and binary program files and put them into EPROM format in data files on tape.
SEND	Utility program to take EPROM format data files and transfer them to the host computer.
GENDVC	General purpose program skeleton that functions as an HP-IB device.

NODE Program

The NODE program is a general purpose program that will perform the following operations:

- Determine if there is an HP-IB or Serial interface card and download from that card.
- If no card is found or there is an error in the download process, the program will load the Autost file from tape (for recent updates).
- If there is no tape drive or Autost file it will try to load a Backup program from EPROM.
- If there is no Backup program on EPROM, the 9915 will reset itself and try all over again.

Using NODE

The NODE program is intended for use on EPROM in a 9915. Use the IMAGE program to create an EPROM image of this program. When using the IMAGE program, be sure to rename NODE to "Autost". Then use the EPROM programmer program (such as DIOix or DIOxix) to burn the program in a 2716 or 2732 EPROM. The IMAGE and EPROM programmer programs are in the 9915 Tape Duplication and EPROM Programming Software Pack, (p/n 09915-10010).

Once the program is in an EPROM, put the EPROM on the 9915 EPROM board inside the 9915.

Modifications to NODE

There are several possible modifications to NODE that might be desirable for your application. Some ideas are:

- Turn on a front panel LED to indicate a download is in progress.
- Display a message on the CRT to indicate a download is in progress. (Note that the Operator Interface Option 002 and the display monitor do not need to be present for a DISP statement to execute without an error.)
- Use a SET TIMEOUT statement to give the host a specific time in which to respond to the node's request. If no response occurs, then the node could load another program off of tape or EPROM.
- Disable the AUTOSTART and SELFTEST keys and inputs via the ENABLE AS-ST statement.
- Require an operator log-in before system operation can occur. This log-in could consist of just the special function keys or a full alpha-numeric password.
- Try a second download path if the primary host computer fails. In some high-reliability or maximum up-time applications the cost of the extra interface card and computer are worthwhile.

```
1000 !
1010 !  PROGRAM    NODE
1020 !
1030 !  DATE       09/15/80
1040 !  UPDATE     04/04/81
1050 !
1060 !  COMPUTER   HP 9915A , HP 85A
1070 !
```

```
1080 !  PURPOSE   PROGRAM DOWNLOAD UTILITY FOR THE 9915A OR 85A
1090 !
1100 !
1110 CCLEAR
1120 DISP "DOWNLOAD UTILITY       HP 9915A"
1130 DISP
1140 F$="Autost" !                FILE TO BE DOWNLOADED
1150 T$="Autost" !                TAPE FILE TO BE LOADED IF NO DOWNLOAD
1160 B$="Backup" !                PROM BACKUP FILE TO BE LOADED OTHERWISE
1170 !
1180 ! FIND INTERFACE CARD
1190 !
1200 ON ERROR GOTO 1270 !         STATUS TO NON-EXISTANT CARD GIVES AN ERROR
1210 FOR S=3 TO 10
1220 STATUS S,0 ; A !             CHECK CARD I.D.
1230 IF A=2 THEN GOTO 1420 !      IF RS-232 THEN CARD IS FOUND
1240 IF A#1 THEN GOTO 1270 !      IF NOT HP-IB OR RS-232 THEN TRY ANOTHER
1250 STATUS S,4 ; B !             IF CARD IS HP-IB THEN MAKE SURE IT IS
1260 IF BIT(B,5)=0 THEN 1420 !    NOT SYSTEM-CONTROLLER
1270 NEXT S
1280 !
1290 ! NO CARD WAS FOUND
1300 !
1310 DISP "NO INTERFACE CARD WAS FOUND"
1320 PROM IS TAPE
1330 ON ERROR GOTO 1370
1340 PLOADGO T$
1350 DISP "TAPE FILE "&T$&" DID NOT LOAD"
1360 GOTO 1650
1370 DISP "NO TAPE DRIVE OR TAPE CARTRIDGE"
1380 GOTO 1650
1390 !
1400 ! CARD FOUND
1410 !
1420 OFF ERROR
1430 IF A=1 THEN C$="HP-IB" ELSE C$="SERIAL"
1440 DISP C$&" CARD ON SELECT CODE "&S
1450 IF BIT(S,0)=1 THEN P$="BINARY" ELSE P$="ASCII"
1460 DISP P$&" PROTOCOL WILL BE USED"
1470 !
1480 ! GET PROGRAM
1490 !
1500 ON ERROR GOTO 1620
1510 RESET S
1520 ! SET TIMEOUT S;32767 !      USE THIS LINE TO ONLY WAIT 32 SECONDS FOR DOWNLO
AD
1530 ! ON TIMEOUT S GOTO 1650 ! TRY BACKUP LOAD IF A TIMEOUT
1540 PROM IS S !                  SPECIFY CARD IS WHERE PROM DATA WILL COME FROM
1550 DISP "ATTEMPTING TO LOAD "&F$&" FROM INTERFACE "&S
1560 PLOADGO F$ !                 LOAD FILE FROM INTERFACE
1570 DISP "FILE "&F$&" DID NOT GET LOADED"
1580 GOTO 1650
1590 !
1600 ! ERROR DURING DOWNLOAD
1610 !
1620 OFF ERROR
1630 DISP "ERROR "&ERRN&" ON LINE "&ERRL&" IN ROM "&ERROM&" ON SC "&ERRSC&" DURI
NG DOWNLOAD "
1640 !
1650 ON ERROR GOTO 1700
1660 PROM IS PROM
1670 DISP "ATTEMPTING TO LOAD "&B$&" FROM PROM"
1680 PLOADGO B$
1690 DISP "BACKUP PROGRAM DID NOT LOAD"
1700 DISP "MACHINE WILL RESTART IN 10 SEC"
1710 WAIT 10000
1720 AUTOSTART
1730 END
```

IMAGE Program

The image program is the utility program to convert the application programs from the internal-form in PROG files on tape into DATA files usable for internal PROM or external downloading.

Refer to the Tape Duplication and EPROM Programming Software Pack (p/n 09915-10010) for more information on the IMAGE program and preparations for running the program.

Using IMAGE

After you have developed your applications programs and have them on tape, there are certain steps you need to go through before using the IMAGE program. These steps are:

1. Minimize program size. Refer to the Tape Duplication and EPROM Programming Pack for further information about reducing the program size. An important tool is the SHRINK program. By minimizing the programs' size, you can place more programs in your computer.
2. Determine how much memory will be used by your application programs. You can, however, just assume the maximum of 32 kbytes of program space. Assuming space of 32k will require the additional R/W memory module (p/n 82903A) on the HP-85 or 9915 running IMAGE.
3. Put all of the application programs on one tape that IMAGE will use to create the EPROM image on.

The steps to use IMAGE to create networking data files is as follows:

1. LOAD and RUN IMAGE.
2. The computer will ask for EPROM type. Even though you are not creating an EPROM, this will specify the file size used by the program.

   Position the cursor under 2716 or 2732 and then press ENDLINE.
3. The program will ask for the file where you want the data placed. The file name "EPROM" is commonly used.

   Type in the file name and press ENDLINE.

   Note that the file will be created if it does not already exist. If it does exist, the file will be overwritten.
4. The program will ask you for the number of EPROMs that you will be using. You can calcuate this exactly or you can merely type in 8.

   Type in the number of EPROMs and press ENDLINE.
5. The program will ask for a program file name. There are three responses to this request:

file name	to enter a program
/R file name	to enter a program under a new name
/Q	to indicate completion of the program

Modifications to IMAGE

There are no modifications that would greatly enhance the use of the IMAGE program.

```
10 !  EPROM IMAGE GENERATOR
20 !
30 ! Dimension R$ at least 2048 for 2716s, 4096 for 2732s, 8192 for 2764s
40 DIM C$[1024],D$[258],F$[6],T$[1],S$[1],X$[258],R$[4096]
50 ON ERROR GOTO 70 ! Need the T^PDUP binary
60 PLOADBIN "TAPDUP" @ OFF ERROR @ GOTO 100 ! Got it
70 OFF ERROR @ IF ERRN=25 THEN 100 ! Already loaded
80 LOADBIN "TAPDUP" @ OFF ERROR @ GOTO 100 ! Try tape
90 OFF ERROR @ IF ERRN#25 THEN 80
100 CCLEAR @ DISP @ DISP "EPROM type"
110 DISP 2716 @ DISP 2732 @ CCURSOR 42 @ INPUT T
120 T1$=VAL$(T) @ IF LEN(T1$)#4 THEN 100 ELSE T=VAL(T1$[3,4])*128
130 DISP "DATA file to use-" @ INPUT K$
140 ON ERROR GOTO 160 @ ASSIGN# 1 TO K$ ! Does it exist?
150 OFF ERROR @ ASSIGN# 1 TO * @ GOTO 180 ! Yes, don't CREATE
160 OFF ERROR @ DISP "Length in "&T1$&"s" @ DISP "     (0 to try again)"
170 INPUT F@ IF F<=0 THEN 130 ELSE CREATE K$,F,T+3
180 F=1 @ R9=259 @ C$=CATALOG$ @ D$=CHR$(0) ! Initialize
190 FOR I=1 TO 8 @ D$=D$&D$ @ NEXT I @ R$=D$&"MT"
200 PRINT @ PRINT "  GENERATING "&T1$&" IMAGEFILE"
210 PRINT "   filling "&K$ @ PRINT
220 FOR I=1 TO 32 ! Limit to 32 files
230 CLEAR ! Get a file
240 DISP USING "5X,K,DD,K,4Z" ; "Next free=ROM#",F-1,", loc ",R9-1
250 DISP "PROGRAM FILE /Q=quit /R=rename"!@ INPUT X$
260 R=0 @ X$=X$&" " ! Rename off
270 X=POS(X$,"/") ! Any switches?
280 IF X=0 THEN 330 ! No
290 F$=UPC$(X$[X+1,X+1])
300 IF F$="Q" THEN 680 ! done
310 IF F$="R" THEN R=1 ! Change name in EPROM image
320 X$[X]=X$[X+2] @ GOTO 270
330 ON ERROR GOTO 350
340 OPEN IMAGE X$ @ GOTO 370 ! find the file
350 OFF ERROR @ IF ERRN#67 THEN 340
360 DISP "*** File not found" @ GOTO 250 ! Try again
370 OFF ERROR @ C$[513]=CATALOG$ ! File found
380 L=INT(READTYPE/4)*4 @ IF L=8 OR L=32 THEN 400 ! PGRM or BPGM?
390 DISP "*** Invalid file type" @ GOTO 250
400 IF R THEN DISP "Name in EPROM image"!ELSE 420 ! Revised name
410 INPUT X$@ X$=X$&"       "
420 IF R THEN F$=X$[1,6] ELSE F$=READNAME$
430 L=0 ! File length
440 ON ERROR GOTO 470
450 X$=READ RECORD IMAGE$ ! Read to end
460 OFF ERROR @ L=L+1 @ GOTO 440
470 OFF ERROR @ IF ERRN#71 THEN 450 ! End of file?
480 PRINT USING "DD,X,6A,3D,11A,ZZ" ; I,F$,L," rec, type ",READTYPE
490 S$=CHR$(L) @ T$=CHR$(READTYPE) ! Size and type
500 D$[8*I-7,8*I]=F$&S$&T$ ! Add to directory
510 D$=D$[1,256]&CHECKSUM$(D$[1,256]) ! new checksum
520 X$=READNAME$ @ OPEN IMAGE X$
530 FOR J=1 TO L ! Convert to EPROM images
540 ON ERROR GOTO 560
550 X$=READ RECORD IMAGE$ @ OFF ERROR @ GOTO 610
560 OFF ERROR @ IF ERRN=243 THEN 580
570 IF ERRN#73 THEN 550
580 IF CATALOG$=C$[513] THEN 550
590 A7$=READNAME$ @ C=513 ! Get right tape
600 GOSUB 850 @ GOTO 540
610 X$=X$&CHECKSUM$(X$) ! Make into EPROM record
620 IF R9>T-257 THEN GOSUB 730 ! EPROM full
630 R$[R9]=X$ @ R9=R9+LEN(X$) ! Add to image
640 NEXT J ! Next record
```

```
650 PRINT USING "5X,14A,DD,6A,4Z" ; "Next free=ROM#",F-1,", loc ",R9-1
660 NEXT I ! Next file
670 DISP "DIRECTORY FULL" @ BEEP @ WAIT 500 @ BEEP
680 R$[T]=" " @ GOSUB 760 @ ASSIGN# 1 TO K$ ! Last image
690 READ# 1,1 ; R$@ R$[1,258]=D$ @ PRINT# 1,1 ; R$[1,T] ! Fill in directory
700 PRINT "      Total "&T1$&"s ="iF-1 @ PRINT @ PRINT
710 DISP "DONE"
720 END
730 R$[R9,T]=X$[1,T+1-R9] ! Finish current image
740 X$=X$[T+2-R9] ! Trim to next image's portion
750 ! Write R$ as EPROM F of image file.
760 ON ERROR GOTO 780
770 ASSIGN# 1 TO K$ @ OFF ERROR @ GOTO 800
780 OFF ERROR @ IF ERRN#67 THEN 770
790 A7$=K$ @ C=1 @ GOSUB 850 @ GOTO 760 ! Get the right tape
800 PRINT# 1,F ; R$[1,T] ! Add new record
810 ASSIGN# 1 TO * ! Update complete
820 R9=1 @ F=F+1 @ R$="" ! Init EPROM image
830 RETURN
840 END
850 DISP "Looking for tape with "&A7$
860 D=0 @ D1=0 @ BEEP 100,50
870 ON TIMER# 1,500 GOSUB 940
880 D=D+1 @ ON TIMER# 2,10000+50000*(D>6) GOSUB 930
890 IF D=6 THEN 880
900 IF D1 THEN 860
910 IF D#10 THEN 890
920 CLEAR @ RETURN
930 BEEP 100,50 @ D=D+(D>7) @ RETURN
940 ON ERROR GOTO 960
950 READ# 6 ; I1$ ! always errs
960 OFF ERROR @ IF ERRN#62 THEN RETURN
970 ON TIMER# 1,500 GOSUB 990
980 RETURN
990 ON ERROR GOTO 1050
1000 REWIND @ OFF TIMER# 1 @ OFF TIMER# 2 @ OFF ERROR
1010 ON ERROR GOTO 1040
1020 IF CATALOG$#C$[C,C+511] THEN 1040
1030 D=10 @ RETURN
1040 D1=1 @ OFF ERROR @ RETURN
1050 OFF ERROR @ RETURN
```

SEND Program

The SEND program is used to take the data produced by the IMAGE program and send it to an HP 9835/9845, HP 1000 or some other computer. It produces ASCII or binary data in 2 kbyte or 4 kbyte segments. The UPLOAD program is the program to run on the other computer.

The HP 9915 or HP-85, when used as a host, does not need to use the SEND program. The UPLOAD program will transfer the data from the EPROM file to a data file on a single HP 9915 or HP-85 computer.

Using SEND

The steps to use the SEND program are as follows:

1. Check the system configuration.
   Make sure that the HP-85 or 9915 that will be running SEND has an RS-232 or HP-IB interface card. If the interface is HP-IB, make sure that the other computer (running UPLOAD) and the SEND computer are not both system controller and that their bus addresses are different.
2. LOAD SEND on the HP-85 or 9915.
3. LOAD UPLOAD on the other computer.
4. Make whatever program modifications are necessary (select code).
5. RUN the programs on both computers.
6. Enter the file name of the EPROM data.
   This is the file you created in IMAGE. The file is usually called "EPROM".
7. Enter the EPROM type (2716 or 2732).
   This is the same as what you selected in IMAGE.
8. Enter the upload protocol; "A" for ASCII or "B" for binary.
   This selects whether the data is to be sent to the other computer as absolute binary data or as a series of octal constants. This is determined by what the other computer is configured to receive.
9. Follow the steps from the UPLOAD program on the other computer.

Modifications to SEND

In general, the only major modification necessary to SEND is to change the select code specified in the program to the select code of the desired interface card in the HP 9915 or HP 85 computer.

When using the HP 1000 over a asynchronous serial interface there are some additional changes that need to be made. Refer to Chapter 8 for additional information.

changes that need to be made. Refer to Chapter 8 for additional information.

```
1000 !
1010 !  PROGRAM   SEND
1020 !
1030 !  DATE      03/31/81
1040 !  UPDATE    04/25/81
1050 !
1060 !  COMPUTER  HP 9915A , HP 85A
1070 !
1080 !  PURPOSE   PROGRAM UPLOAD UTILITY FOR THE 9915A OR 85A
1090 !
1100 !
1110 DIM D$[4096] !              STRING VARIABLE FOR READING EPROM FILES
1120 CCLEAR
1130 DISP "SEND - UPLOAD UTILITY  HP 9915A"
1140 DISP
1150 S=5 !                       SELECT CODE FOR UPLOADING DATA
1160 !
1170 DISP "ENTER FILE NAME OF EPROM DATA"
1180 INPUT F$
1190 DISP "ENTER EPROM TYPE SPECIFIED IN IMAGE"
1200 DISP @ DISP "2716" @ DISP "2732"
1210 CCURSOR CCPOS-96
1220 INPUT Z
1230 IF NOT (Z=2716 OR Z=2732) THEN GOTO 1190
1240 IF Z=2716 THEN Z=2048
1250 IF Z=2732 THEN Z=4096
1260 DISP "ENTER UPLOAD FORM - A for ascii or B for binary"
1270 INPUT M$
1280 !
1290 !                           READ DATA AND TRANSMIT TO HOST
1300 !
1310 ASSIGN# 1 TO F$
1320 ON ERROR GOTO 1460
1330 !
1340 !                           READ DATA AND TRANSMIT TO HOST
1350 !
1360 READ# 1 ; D$
1370 OUTPUT S USING "4A,A,4D" ; "DATA";M$;Z
1380 IF M$="B" THEN OUTPUT S USING "#,K" ; ;D$[1,Z] @ GOTO 1360
1390 FOR I=1 TO Z
1400 OUTPUT S USING "K" ; DTO$(NUM(D$[I,I]))
1410 NEXT I
1420 GOTO 1360
1430 !
1440 !                           TELL HOST , UPLOAD IS DONE
1450 !
1460 IF ERRN=71 THEN GOTO 1510
1470 IF ERRN=72 THEN GOTO 1510
1480 DISP "AN ERROR HAS OCCURED"
1490 DISP "ERROR ";ERRN;" IN LINE ";ERRL
1500 STOP
1510 OUTPUT S USING "4A,X,4D" ; "DONE";0
1520 DISP "UPLOAD IS COMPLETE"
1530 END
```

GENDVC Program

GENDVC is a skeleton program; a general outline of a program that you can tailor to your needs. This program is not a networking program in the strict sense of the term. The program is intended for use in instrumentation applications where you want to have a smart instrument (the 9915) controlling a cluster of other instruments.

Using GENDVC

The GENDVC program is intended to be modified, so, use depends on the modifications. However, with no modifications, the GENDVC will operate as a type of bus analyzer. When you address it to listen, it will receive and display the message. When a bus message occurs (like IFC, GET, DCL, etc.) the computer will display the event.

Modifications to GENDVC

The GENDVC program can be extended in many ways. Some of the extensions might be:

- When a group execute trigger (or some specific data or command) appears on the HP-IB interface, download a new program to the instrument cluster.
- Use the secondary addresses to access the various instruments attached to the 9915 locally.
- Use the secondary addresses to access the various peripherals of the 9915. An example might be to have SCG 0 (secondary command group, address 0) be the alpha CRT display, SCG 1 is the keyboard, SCG 2 is the LED's, SCG 3 is the graphics CRT display, and SCG 4 is the audio oscillator.

```
1000 !
1010 !  PROGRAM    GENDVC
1020 !
1030 !  DATE       09/29/80
1040 !  UPDATE     03/31/81
1050 !
1060 !
1070 !  MACHINES   HP 9915A , HP 85A
1080 !
1090 !  PURPOSE    TO PROVIDE A SKELETON PROGRAM THAT WILL ALLOW A DESIGNER
1100 !             TO USE A HP 9915A OR 85A AS A GENERAL PURPOSE HP-IB DEVICE
1110 !
1120 !
1130 ! ----------------------- PROGRAM SETUP -------------------------
1140 !
1150 !
1160 DIM M$[256] !                MESSAGE BUFFER - COMING IN
1170 DISP "GENDVC                   HP 9915A"
1180 !
1190 ! FIND INTERFACE CARD
1200 !
1210 DISP "HUNTING FOR HP-IB CARD"
1220 ON ERROR GOTO 1280 !         STATUS TO NON-EXISTANT CARD GIVES AN ERROR
1230 FOR S=3 TO 10
1240 STATUS S,0 ; S1 !            CHECK FOR HP-IB CARD
1250 IF S1=1 THEN GOTO 1280
1260 STATUS S,4 ; S2 !            CHECK FOR SYSTEM CONTROLLER
1270 IF BIT(S2,5)=0 THEN GOTO 1340
1280 NEXT S
1290 DISP "NON-CONTROLLER HP-IB NOT FOUND"
1300 STOP
```

```
1310 !
1320 ! SET UP FOR INTERRUPT
1330 !
1340 OFF ERROR
1350 ON INTR S GOSUB 1520
1360 ENABLE INTR S;BTD("11110111")
1370 !                          SET UP FOR IFC,LA,CA,TA,DCL,SET,SCG INTERRUPTS
1380 !
1390 !
1400 ! ------------------------ MAIN PROGRAM IDLE LOOP ----------------
1410 !
1420 DISP "WAITING FOR AN INTERRUPTING CONDITION" @ CCURSOR CCPOS-64
1430 !                          USEFUL WORK COULD BE DONE HERE
1440 GOTO 1420
1450 !                          REMOTE/LOCAL STATUS MUST BE CHECKED IN THE
1460 !                          MAIN IDLE LOOP IF DESIRED
1470 !
1480 !
1490 ! ------------------------ INTERRUPT SERVICE ROUTINE -------------
1500 !
1510 !
1520 STATUS S,1 ; R !           GET REASON FOR INTERRUPT
1530 FOR R9=0 TO 7 !            CHECK FOR EACH CONDITION ONE AT A TIME
1540 IF BIT(R,R9)=0 THEN GOTO 1570
1550 ON R9+1 GOSUB 1670,1780,1840,1900,1980,2040,2090,2180
1560 !                          COMPUTED GOSUB FOR EACH CONDITION
1570 NEXT R9
1580 ENABLE INTR S;BTD("11110111") @ RETURN
1590 !                          RE-ARM INTERRUPTS & RETURN TO IDLE LOOP
1600 !
1610 !
1620 ! ------------------------ CONDITION SERVICE ROUTINES ------------
1630 !
1640 !
1650 ! SCG INTERRUPT            SECONDARY COMMAND GROUP
1660 !
1670 DISP "SCG INTERRUPT"
1680 STATUS S,6 ; V !           GET SCG ADDRESS BYTE
1690 !                          V IS AN EXACT COPY OF THE BUS WHEN THE SGC BYTE
1700 !                          WAS SENT INCLUDING BIT 7 ( WHICH CAN BE USER
1710 !                          DEFINED ) AND BIT 6&5 ( WHICH ARE 1's )
1720 DISP "SECONDARY ADDRESS WAS FOR ";V
1730 R=BINAND(R,BTD("10101111")) !   TO GET RID OF TALK/LISTEN AINTR BITS
1740 RETURN
1750 !
1760 ! GET INTERRUPT           GROUP EXECUTE TRIGGER
1770 !
1780 DISP "TRIGGER INTERRUPT"
1790 R=BINAND(R,BTD("10101111")) !   TO GET RID OF TALK/LISTEN AINTR BITS
1800 RETURN
1810 !
1820 ! DCL or SDC INTERRUPT     DEVICE CLEAR or SELECTIVE DEVICE CLEAR
1830 !
1840 DISP "CLEAR INTERRUPT"
1850 R=BINAND(R,BTD("10101111")) !   TO GET RID OF TALK/LISTEN AINTR BITS
1860 RETURN
1870 !
1880 ! SRQ INTERRUPT            SERVICE REQUEST
1890 !
1900 ! THIS SHOULD NEVER HAPPEN IN A NON CONTROLLER
1910 DISP "ERROR - SRQ INTERRUPTS SHOULD NOT HAPPEN"
1920 RETURN
1930 !
1940 ! TA INTERRUPT             TALK ADDRESSED
1950 !
1960 DISP "TALK INTERRUPT"
1970 STATUS S,5 ; V !           MAKE SURE I AM STILL ADDRESSED
1980 IF BIT(V,4)=0 THEN RETURN ! THIS MIGHT HAPPEN WHEN A SERIAL POLL OCCURS
```

```
1990 OUTPUT S ;" WHATEVER YOU WOULD LIKE TO SAY "
2000 RETURN
2010 !
2020 ! CA INTERRUPT              ACTIVE CONTROLLER
2030 !
2040 DISP "PASS CONTROL INTERRUPT"
2050 RETURN
2060 !
2070 ! LA INTERRUPT              LISTEN ADDRESSED
2080 !
2090 DISP "LISTEN INTERRUPT"
2100 STATUS S,5 ; V !            MAKE SURE STILL ADDRESSED TO LISTEN
2110 IF BIT(V,6)=0 THEN RETURN
2120 ENTER S ; M$ !              GET THE INCOMING MESSAGE
2130 DISP "THE MESSAGE WAS "&M$
2140 RETURN
2150 !
2160 ! IFC INTERRUPT             INTERFACE CLEAR
2170 !
2180 DISP "IFC INTERRUPT"
2190 RETURN
```

Chapter 6
HP 9915 Host Programs

Introduction

This chapter discusses and lists some networking related programs for the 9915. These programs are for an HP-85 or 9915 when it is the host computer.

The programs discussed are:

UPLOAD	Utility program to take the EPROM data file produced by IMAGE and convert into a form usable by HOST_S and HOST_M.
HOST_S	General purpose program that sends the data records requested by a single node computer.
HOST_M	General purpose program that sends the data records requested by several node computers.

Possible Configurations

The HP-85 and 9915 are general purpose desktop computers. Because they are general purpose, they can easily handle the task of being a host controller in a networking system. The particular advantages that the HP-85 and 9915 have over some other computers are:

- Low cost
- Small size
- Low power consumption

The HP-85 and 9915 have these advantages but are not high-performance (high-speed) computers compared to many other machines. When the primary task of a HOST is to bring up a system of NODE computers, however, the HP-85 and 9915 are excellent choices.

The HOST_S program supports a single node computer over a serial or HP-IB interface. This HOST program reads the download data off of a random access mass storage file. Because of tape movement, it can improve performance to put the file on a flexible disc, such as the HP 82900 or HP 9895 flexible disc systems.

The HOST_M program supports multiple node computers on a single HP-IB interface. This HOST program also uses mass storage files and would also benefit from a flexible disc system. The program is written for a single HP-IB interface card with the host as system controller. It is possible to modify the program to use several different interfaces, with some being HP-IB and some being serial interfaces.

Both programs, HOST_M and HOST_S, are written to expect only binary download protocol. The HP-85 and 9915 can handle ASCII protocol but this would slow down the download process.

UPLOAD Program

The UPLOAD program is a utility program that receives the EPROM data file created by IMAGE into a file usable by HOST_S and HOST_M programs.

Using UPLOAD

The steps to use UPLOAD are as follows:

1. Type in the UPLOAD program and store it on tape or disc.
2. If not already in the machine, load the UPLOAD program.
3. Press the RUN key.
4. When the program asks for the EPROM file type in the EPROM data file name, press the ENDLINE key.
   This is the file name you gave in IMAGE (generally "EPROM").
5. When the program asks for the destination file, type in the file name to be used by HOST_S and HOST_M (generally "FILES") and press the ENDLINE key.

Modifications to UPLOAD

There are no modifications that would greatly enhance the use of the UPLOAD program.

```
1000 !
1010 !  PROGRAM    UPLOAD
1020 !
1030 !  DATE       03/31/81
1040 !  UPDATE     04/06/81
1050 !
1060 !  COMPUTER   HP 9915A , HP 85A
1070 !
1080 !  PURPOSE    PROGRAM UPLOAD UTILITY FOR THE 9915A OR 85A
1090 !             THAT TRANSFORMS THE EPROM FILES INTO DATA FILES
1100 !             FOR USE BY HOST.S AND HOST.M
1110 !
1120 !
1130 DIM D$[4654] !              STRING VARIABLE FOR READING EPROM FILES ( 4096 +
 258 )
1140 DIM L$[258] !               STRING VARIABLE FOR LEFT OVER DATA
1150 CCLEAR
1160 DISP "UPLOAD UTILITY          HP 9915A"
1170 DISP
1180 !
1190 DISP "ENTER FILE NAME OF EPROM DATA"
1200 INPUT F$
1210 DISP "ENTER NAME OF DESTINATION DATA FILE"
1220 INPUT F2$
1230 DISP
1240 !
1250 !                           READ DATA AND TRANSMIT TO HOST
1260 !
1270 L$="" !                     CLEAR LEFTOVER STRING
1280 ASSIGN# 1 TO F$
1290 ON ERROR GOTO 1310
1300 CREATE F2$,128,261 !        CREATE DATA FILE ( SIZE + 3 BYTES OVERHEAD )
1310 OFF ERROR
1320 ASSIGN# 2 TO F2$
1330 J=1 !                       RECORD COUNTER FOR DESTINATION FILE
1340 ON ERROR GOTO 1500
1350 !
1360 !                           READ DATA AND RE-FORMAT FOR HOST
1370 !
```

```
1380 READ# 1 ; D$
1390 D$=L$&D$
1400 FOR I=1 TO LEN(D$) STEP 258
1410 IF I+258>LEN(D$) THEN GOTO 1440
1420 PRINT# 2,J ; D$[I,MIN(I+257,LEN(D$))]
1430 J=J+1
1440 NEXT I
1450 L$=D$[I-258,LEN(D$)]
1460 GOTO 1380
1470 !
1480 !                          TELL HOST , UPLOAD IS DONE
1490 !
1500 IF ERRN=71 THEN GOTO 1550
1510 IF ERRN=72 THEN GOTO 1550
1520 DISP "AN ERROR HAS OCCURED"
1530 DISP "ERROR "!ERRN!" IN LINE "!ERRL
1540 STOP
1550 PRINT# 2,J ; L$ !              LEFT OVER DATA
1560 ASSIGN# 2 TO *
1570 DISP "UPLOAD IS COMPLETE"
1580 END
```

HOST_S Program

The HOST_S program is a host program to service a single node computer.

Using HOST_S

The HOST_S program does not require operator interaction. The program needs to be loaded and run; however, it can be an Autost file on tape.

Modifications to HOST_S

There are several modifications that would enhance the use of HOST_S. Some of these modifications are:

- Put the data file on a flexible disc and modify the program to use the Mass Storage ROM and flexible disc. The random access nature of the download process lends itself to flexible discs.
- Let the HOST_S program search for the interface card by itself. This makes HOST_S self configuring.
- Rather than using ENTER/OUTPUT and the timeout facility, use TRANSFER interrupt. This allows the host computer more time to do other tasks.

```
1000 !
1010 !  PROGRAM    HOST_S
1020 !
1030 !  DATE       08/17/80
1040 !  UPDATE     04/04/81
1050 !
1060 !  MACHINES   HP 9915A , HP 85A
1070 !
1080 !  PURPOSE    TO DOWNLOAD THE PROGRAM FILE THAT A NODE
1090 !             REQUESTED
1100 !
1110 !
```

```
1120 ! ------------------------ PROGRAM SETUP ------------------------
1130 !
1140 !
1150 CCLEAR @ DISP "HOST.S UTILITY          HP 9915A"
1160 DIM B$[258] !                  REQUESTED RECORD BUFFER
1170 S9=7 !                         SELECT CODE
1180 S=720 !                        DEVICE SPECIFIER ( FOR HP-IB USE )
1190 !
1200 ! OPEN DATA FILE
1210 !
1220 ASSIGN# 1 TO "FILES" !         OPEN DATA FILE
1230 !
1240 !
1250 ! ------------------------ MAIN PROGRAM LOOP OF HOST -------------
1260 !
1270 !
1280 !
1290 !  SERVICE THE SINGLE NODE COMPUTER
1300 !
1310 GOSUB 1520 !                   GET REQUEST
1320 READ# 1,A+1 ; B$ !            READ RECORD FROM MASS STORAGE
1330 DISP "SENDING RECORD ";A
1340 GOSUB 1410 !                   SEND DATA
1350 GOTO 1310
1360 !
1370 !
1380 ! ------------------------ SEND FROM RECORD ----------------------
1390 !
1400 !
1410 SET TIMEOUT S9;5000 !          SET UP A TIMEOUT IN CASE THE DEVICE GOES DOWN
1420 ON TIMEOUT S9 GOTO 1690
1430 OUTPUT S USING "#,B,258A" ; 2,B$
1440 SEND S9 ; UNL UNT !            LEAVE THE BUS IN A CLEAN STATE
1450 RESUME S9 !                    BY UN-ADDRESSING EVERYONE
1460 RETURN
1470 !
1480 !
1490 ! ------------------------ GET NODE REQUEST ----------------------
1500 !
1510 !
1520 DISP "GET REQUEST FROM NODE"
1530 SET TIMEOUT S9;1000 !          SET UP A TIMEOUT IN CASE THE NODE GOES DOWN
1540 ON TIMEOUT S9 GOTO 1750
1550 ENTER S USING "B,W,X" ; R,A !   GET PROMPT AND ADDRESS
1560 IF R#1 THEN DISP "BAD PROMPT FROM NODE" @ GOTO 1550
1570 A=INT(A/258) !                 BECAUSE ADDRESS IS IN ABSOLUTE BYTE ADDRESS
1580 SEND S9 ; UNL UNT !            GET BUS INTO A CLEAN STATE
1590 RESUME S9 !                    BY UNADDRESSING EVERYONE
1600 RETURN
1610 !
1620 !
1630 ! ------------------------ TIMEOUT ROUTINES ----------------------
1640 !
1650 !
1660 !
1670 ! TIMEOUT FOR OUTPUT
1680 !
1690 RESET S9 !                     GET THE CARD INTO A CLEAN STATE
1700 CCURSOR CCPOS-64 @ DISP "OUTPUT TIMEOUT"
1710 RETURN !                       RETURN FROM SEND DATA ROUTINE
1720 !
1730 ! REQUEST TIMEOUT
1740 !
1750 CCURSOR CCPOS-64 @ DISP "REQUEST TIMEOUT"
1760 RESET S9 !                     GET THE CARD INTO A CLEAN STATE AFTER A TIMEOUT
1770 GOTO 1520 !                    TRY TO GET ANOTHER REQUEST
```

HOST_M Program

The HOST_M program is a host program to service a several node computers. The node computers are connected over an HP-IB bus with the host being system controller.

Using HOST_M

The HOST_M program does not require operator interaction. The program needs to be loaded and run; however, it can be an Autost file on tape.

Modifications to HOST_M

There are several modifications that would enhance the use of HOST_M. Some of these modifications are:

- Put the data file on a flexible disc and modify the program to use the Mass Storage ROM and flexible disc. The random access nature of the download process lends itself to flexible discs.
- Let the HOST_M program search for the interface card by itself. This makes HOST_M self configuring.
- Node computers on different HP-IB buses or on several serial interfaces could be included in the HOST_M service cycle. Modify the HOST_M program to set up the proper device addresses in the device array and set up the proper select code in the timeout setup.

```
1000 !
1010 !  PROGRAM    HOST_M
1020 !
1030 !  DATE       09/28/80
1040 !  UPDATE     04/04/81
1050 !
1060 !
1070 !  MACHINES   HP 9915A , HP 85A
1080 !
1090 !  PURPOSE    TO DOWNLOAD THE PROGRAM FILE THAT ANY ONE OF SEVERAL NODES
1100 !             HAS REQUESTED - USING A DATA FILE CREATED BY IMAGE
1110 !
1120 !
1130 ! ------------------------ PROGRAM SETUP ------------------------
1140 !
1150 !
1160 CCLEAR @ DISP "HOST.M UTILITY          HP 9915A"
1170 INTEGER A(30) !                 ARRAY OF NODE DEVICE ADDRESSES
1180 DIM B$[256] !                   REQUESTED RECORD BUFFER
1190 S=7
1200 !
1210 ! FIND ALL THE NODES
1220 !
1230 OFF ERROR
1240 DISP "ATTEMPTING TO FIND HPIB DEVICES"
1250 S8=S !                          SET S8 TO THE INTERFACE SELECT CODE
1260 C=0
1270 STATUS S8,4 ; S2
1280 FOR S8=0 TO 30 !                CHECK ALL DEVICES
1290 IF S2 MOD 32=S8 THEN 1380 ! SKIP IF IT IS THE HP-IB CARD'S ADDRESS
1300 SEND S ; UNL MTA LISTEN S8
1310 RESUME S !                      DROP ATN LINE
1320 STATUS S,2 ; H !                LOOK AT HANDSHAKE LINES
1330 H=H MOD 8 !                     LOOK FOR DAV=0 NDAC=1 NRFD=0
1340 IF H=2 THEN GOTO 1380 !         IF NO RESPONSE THEN GO ON
```

```
1350 !                              I FOUND A RESPONDING DEVICE
1360 C=C+1 !                        INCREMENT COUNT OF DEVICES
1370 A(C)=S*100+SB !                SAVE NODE ADDRESS
1380 NEXT SB
1390 IF C<>0 THEN GOTO 1420
1400 DISP "NO DEVICES WERE FOUND"
1410 STOP
1420 !
1430 ! OPEN DATA FILE
1440 !
1450 DISP "OPENING FILE"
1460 ASSIGN# 1 TO "FILES"
1470 !
1480 !
1490 ! ------------------------ MAIN PROGRAM LOOP OF HOST -------------
1500 !
1510 !
1520 FOR N=1 TO C !                 GO THROUGH ALL THE EXISTING NODES
1530 D=A(N)
1540 GOSUB 1600
1550 NEXT N
1560 GOTO 1520
1570 !
1580 !  SEE IF THE NODE WANTS SERVICE
1590 !
1600 GOSUB 1840 !                   GET REQUEST
1610 IF T=1 THEN RETURN !           WHEN TIMEOUT HAPPENS - GO ON TO NEXT NODE
1620 GOSUB 1720 !                   SEND B$
1630 GOTO 1600 !                    UNTIL THE READ REQUEST TIMES OUT
1640 !
1650 !
1660 ! ------------------------ DATA TRANSMISSION CODE ---------------
1670 !
1680 !
1690 !
1700 ! SEND DATA TO NODE
1710 !
1720 OUTPUT D USING "#,B,258A" ; 2,B$
1730 SEND S ; UNL UNT !             LEAVE THE BUS IN A CLEAN STATE
1740 RESUME S !                     BY UN-ADDRESSING EVERYONE
1750 RETURN
1760 !
1770 !
1780 ! ------------------------ DATA REQUEST CODE --------------------
1790 !
1800 !
1810 !
1820 ! GET REQUEST FROM NODE
1830 !
1840 DISP "GET REQUEST FROM NODE"
1850 T=0 !                          CLEAR TIMEOUT FLAG
1860 SET TIMEOUT S;1000 !           SET UP A TIMEOUT IN CASE THE NODE GOES DOWN
1870 ON TIMEOUT S GOTO 2040
1880 ENTER D USING "B,W,X" ; R,A !   GET PROMPT CHARACTER ONLY
1890 IF R<>1 THEN DISP "BAD PROMPT FROM NODE" @ GOTO 1880
1900 A=INT(A/258) !                 BECAUSE ADDRESS IS IN ABSOLUTE BYTE ADDRESS
1910 SEND S ; UNL UNT !             GET BUS INTO A CLEAN STATE
1920 RESUME S !                     BY UNADDRESSING EVERYONE
1930 DISP "TRANSMITTING RECORD "&A&" TO NODE "&D
1940 READ# 1,A+1 ; B$ !            READ RECORD + CHECKSUM FROM MASS STORAGE
1950 RETURN
1960 !
1970 !
1980 ! ------------------------ TIMEOUT ROUTINES ---------------------
1990 !
2000 !
2010 !
```

```
2020 ! REQUEST TIMEOUT
2030 !
2040 CCURSOR CCPOS-64 @ DISP "REQUEST TIMEOUT"
2050 RESET S !                    GET THE CARD INTO A CLEAN STATE AFTER A TIMEOUT
2060 T=1 !                        SET TIMEOUT FLAG
2070 RETURN
```

Chapter 7
HP 9835/9845 Host Programs

Introduction

This chapter discusses and lists some networking related programs for the 9835 and 9845 Desktop Computers. These programs are for a 9835/9845 as the host computer.

The programs discussed are:

UPLOAD	Utility program to take the EPROM data file produced by IMAGE and convert into a form usable by HOST_S and HOST_M.
HOST_S	General purpose program that sends the data records requested by a single node computer.
HOST_M	General purpose program that sends the data records requested by several node computers.

Possible Configurations

The 9835/9845 series of desktop computers are good, high-performance desktop machines. Because of their performance level, they can handle the job of being a host controller and other tasks as well. Some of the specific advantages that the 9835/9845 series of machines have over some other computers are:

- Good performance
- Very easy to program and use
- Large memory (up to 449 kbytes on a 9845)
- Graphics (on the 9845)

The HOST_S program supports a single node computer over a serial or HP-IB interface. This host program reads the download data off of a mass storage file only once per program execution. Once the program is running, the host program uses a large string (32 kbytes) as the download storage area. This is feasible since the minimum configuration of the 9835/9845 has 48 kbytes of user read/write memory.

The HOST_M program supports multiple node computers on a single HP-IB interface. This host program also reads the download data off of a mass storage file once per program execution and then uses a string for the storage area. It is possible to modify the program to use several different interfaces, with some being HP-IB and some being serial interfaces.

Both HOST_M and HOST_S programs are written to expect only binary prompts from the node computers. The 9835/9845 computers can handle ASCII protocol but it would slow down the download process.

The various mainframes that can be used and their attributes are:

Mainframe	Attributes
9835A	24 x 80 CRT 3 physical I/O slots (expandable to 14)
9835B	32 character LED display 3 physical I/O slots (expandable to 14) smallest of 9835/9845
9845B	24 x 80 alpha-numeric CRT mono-chromatic graphics 4 physical I/O slots (expandable to 12)
9845C	24 x 80 alpha-numeric CRT color graphics 4 physical I/O slots (expandable to 12)

There are several variations within each of the previous computers. Depending on your needs, these computers have a wide range of enhancements including assembly language, structured programming, data base management, etc. Refer to the system data sheet for some of the possibilities.

The 9835/9845 computer family all use the same interface cables. Some of the interfaces and their uses are:

98032 GPIO 16-bit General Purpose Interface - usable with the 9915 for remote status and control.

98034 HP-IB IEEE-488 Interface - usable for multi-point networking with 9915s.

98035 Real-Time Interface - Battery backed-up rea time interface.

98036 Serial Interface - RS-232 and current loop serial interface - usable for point to point networking with the 9915.

Reading Suggestions

The 9835/9845 series computers have extensive documentation. This list of suggested reading is intended for a beginner:

Manuals for the 9835 A/B Desktop Computer:

09835-90000 Operating and Programming
09835-90001 Beginner's Guide
09835-90005 Owner's Manual

Manuals for the 9845 B/C Desktop Computers:

09845-92000 Operating and Programming
09845-92001 Beginner's Guide
09845-92005 Owner's Guide

Manuals common to both the 9835 A/B and 9845 B/C:

09845-92060 I/O ROM Manual for the 9835A/B and 9845B/C
09835-90600 Basic Language Interfacing Concepts
09845-92070 Mass Storage ROM Programming

UPLOAD Program

The UPLOAD program is a utility program that receives the EPROM data file created by IMAGE into a file usable by HOST_S and HOST_M programs.

Using UPLOAD

The steps to use UPLOAD are as follows:

1. Determine the system configuration. Decide on what type of interface will be used for the communication and check the card for select code (and appropriate connector when using RS-232).
2. Type the program into the computer and SAVE (or STORE) the program on tape or disc.
3. GET (or LOAD) UPLOAD.
4. Press the RUN key.
5. Enter the file name to be used by HOST.S and HOST.M (generally "FILES"). (The input is terminated by pressing the CONTINUE key).
6. Run the SEND program on the HP 9915 or HP-85. Refer to Chapter 5 for additional details.

Modifications to UPLOAD

There are no modifications that would greatly enhance the use of the UPLOAD program.

```
1000 !
1010 !      PROGRAM      UPLOAD
1020 !
1030 !      DATE         09/17/80
1040 !      UPDATE       04/05/81
1050 !
1060 !      MACHINE      HP 9835 , HP 9845
1070 !
1080 !      PURPOSE      TO RECEIVE A SERIES OF PROGRAMS FROM AN HP 85 OR
1090 !                   HP 9915 IN EPROM FORM.  THESE PROGRAMS WILL BE PUT
1100 !                   ON A DATA FILE AS ONE LARGE STRING VARIABLE.  THIS
1110 !                   DATA FILE WILL BE USED LATER BY 'HOST.S' AND 'HOST.M'.
1120 !
1130 DIM Eprom$[32640]                !    EPROM DATA BUFFER
1140                                  !    NOTE THAT THIS STRING IS ONLY 32640
1150                                  !    BYTES LONG.   THE LAST 128 BYTES OF
1160                                  !    PROM ARE NOT A COMPLETE RECORD AND
1170                                  !    ARE NOT USEABLE - SO THEY ARE NOT
1180                                  !    EVEN DECLARED.
1190 DIM Temp$[4096]                  !    TEMPORARY READ BUFFER
1200 !
1210 ! PUT UP MESSAGES
1220 !
1230 PRINT PAGE
1240 PRINT "CREATE FILE UTILITY - UPLOAD";TAB(70);"HP 9835/45"
1250 PRINT
1260 !
1270 !    VARIABLE INITIALIZATION
1280 !
1290 Location=1
1300 S=7                              !    INTERFACE SELECT CODE
1310 Node=720                         !    DEVICE ADDRESS
1320 !
```

```
1330 ! CREATE DATA FILE
1340 !
1350 OFF ERROR
1360 F$="FILES"
1370 INPUT "ENTER EPROM DATA FILE NAME ( default = FILES )",F$
1380 DISP "CREATING DATA FILE"
1390 ON ERROR GOTO 1410
1400 CREATE F$,130                    !    SPACE FOR A 32K BYTE STRING
1410 OFF ERROR
1420 !
1430 !     START UPLOAD
1440 !
1450 Loop:      !
1460 DISP "GETTING PROMPT FROM NODE"
1470 ENTER Node USING "4A,A,4N";Op$,Protocol$,Size
1480 IF Op$="DONE" THEN GOTO Finished
1490 IF Op$="DATA" THEN GOTO Data
1500 PRINT "IMPROPER PROMPT RECEIVED"
1510 GOTO Loop
1520 !
1530 !     GET DATA
1540 !
1550 Data:      !

1560 IF Protocol$="B" THEN GOTO 1590
1570     PRINT "PROTOCOL OTHER THAN BINARY ATTEMPTED"
1580     STOP
1590 Image$="#,"&VAL$(Size)&"A"
1600 DISP "ENTERING RECORD STARTING WITH BYTE ";Location
1610 ENTER Node USING Image$;Temp$
1620 IF Location+Size<32640 THEN Eprom$[Location;Size]=Temp$
1630 IF Location+Size>=32640 THEN Eprom$[Location;Size-128]=Temp$
1640 Location=Location+Size
1650 GOTO Loop
1660 !
1670 !     FINISHED
1680 !
1690 Finished: !
1700 DISP "PLACING DATA ON TAPE"
1710 ASSIGN F$ TO #1                  !    OPEN THE FILE
1720 PRINT #1;Eprom$                  !    WRITE THE DATA TO THE FILE
1730 ASSIGN #1 TO *                   !    CLOSE THE FILE
1740 DISP "PROGRAM FINISHED"
1750 PRINT "PROGRAM FINISHED"
1760 END
```

HOST_S Program

The HOST_S program is a host program to service a single node computer.

Using HOST_S

The HOST_S program does not require operator interaction. The program needs to be loaded and run; however, it can be an autostart file on tape. Note that the 9835/9845 autostart file must be on the ":T15" tape drive with the file name AUTOST and must be a LOAD/STORE type file (not GET/SAVE). Note also that the 9835 will automatically load the AUTOST file if there is one and the 9845 requires the "AUTO ST" key on the keyboard to be in the "in" posifion.

Modifications to HOST_S

There are several modifications that would enhance the use of HOST_S. Some of these modifications are:

- Let the HOST_S program search for the interface card by itself. This makes HOST_S self configuring.
- Rather than using ENTER handshake and the timeout facility, use ENTER BINT transfers. This allows the host computer more time to do other tasks.

```
1000 !
1010 !  PROGRAM    HOST_S
1020 !
1030 !  DATE       09/17/80
1040 !  UPDATE     04/05/81
1050 !
1060 !  MACHINES   HP 9835 , 9845
1070 !
1080 !  PURPOSE    TO DOWNLOAD THE PROGRAM FILE THAT A NODE
1090 !             REQUESTED
1100 !
1110 !
1120 ! ------------------------ PROGRAM SETUP -------------------------
1130 !
1140 !
1150 PRINT PAGE;"HOST.S UTILITY ";TAB(70);"HP 9835/45"
1160 DIM Eprom$[32640]          !  REQUESTED RECORD BUFFER
1170 S9=7 !                     SELECT CODE
1180 S=720 !                    DEVICE SPECIFIER ( FOR HP-IB USE )
1190 !
1200 ! OPEN DATA FILE
1210 !
1220 ASSIGN #1 TO "FILES" !     OPEN DATA FILE
1221 READ #1;Eprom$       !     READ EPROM DATA
1230 !
1240 !
1250 ! ------------------------ MAIN PROGRAM LOOP OF HOST ------------
1260 !
1270 !
1280 !
1290 !  SERVICE THE SINGLE NODE COMPUTER
1300 !
1310 GOSUB 1520 !               GET REQUEST
1330 DISP "SENDING RECORD STARTING WITH BYTE ";A
1340 GOSUB 1410 !               SEND DATA
1350 GOTO 1310
1360 !
1370 !
```

```
1380 ! ------------------------ SEND PROM RECORD ---------------------
1390 !
1400 !
1410 SET TIMEOUT S9;5000 !       SET UP A TIMEOUT IN CASE THE DEVICE GOES DOWN
1420 ON INT #S9 GOTO 1690
1430 OUTPUT S USING "#,B,258A";2,Eprom$[A;258]
1440 SENDBUS S9;"?_"             !       LEAVE THE BUS IN A CLEAN STATE
1450 STATUS S9;Dummy  !                  BY UN-ADDRESSING EVERYONE
1460 RETURN
1470 !
1480 !
1490 ! ------------------------ GET NODE REQUEST ---------------------
1500 !
1510 !
1520 DISP "GET REQUEST FROM NODE"
1530 SET TIMEOUT S9;1000 !       SET UP A TIMEOUT IN CASE THE NODE GOES DOWN
1540 ON INT #S9 GOTO 1750
1550 ENTER S USING "B,W,X";R,A   !   GET PROMPT AND ADDRESS
1560 IF R=1 THEN GOTO 1570
1561    DISP "BAD PROMPT FROM NODE"
1562    GOTO 1550
1570 ! THIS PROGRAM HANDLES THE REQUESTS AS AN ABSOLUTE BYTE ADDRESS
1571 A=A+1
1580 SENDBUS S9;"?_"             !       GET BUS INTO A CLEAN STATE
1590 STATUS S9;Dummy  !                  BY UNADDRESSING EVERYONE
1600 RETURN
1610 !
1620 !
1630 ! ------------------------ TIMEOUT ROUTINES ---------------------
1640 !
1650 !
1660 !
1670 ! TIMEOUT FOR OUTPUT
1680 !
1690 DISP "OUTPUT TIMEOUT"
1710 RETURN !                    RETURN FROM SEND DATA ROUTINE
1720 !
1730 ! REQUEST TIMEOUT
1740 !
1750 DISP "REQUEST TIMEOUT"
1770 GOTO 1520 !                 TRY TO GET ANOTHER REQUEST
```

HOST_M Program

The HOST_M program is a host program to service several node computers. The node computers are connected over an HP-IB bus with the host being system controller.

Using HOST_M

The HOST_M program does not require operator interaction. The program needs to be loaded and run; however, it can be an autostart file on tape. The comments in the Using HOST_S section in this chapter on autostarting programs also apply to the HOST_M program.

Modifications to HOST_M

There are several modifications that would enhance the use of HOST_M. Some of these modifications are:

- Let the HOST_M program search for the interface card by itself. This makes HOST_M self configuring.
- Node computers on different HP-IB buses or on several serial interfaces could be included in the HOST_M service cycle. Modify the HOST_M program to set up the proper device addresses in the device array and set up the proper select code in the timeout set-up.

```
1000 !
1010 !  PROGRAM    HOST_M
1020 !
1030 !  DATE       09/28/80
1040 !  UPDATE     04/05/81
1050 !
1060 !
1070 !  MACHINES   HP 9835 , 9845
1080 !
1090 !  PURPOSE    TO DOWNLOAD THE PROGRAM FILE THAT ANY ONE OF SEVERAL NODES
1100 !             HAS REQUESTED - USING A DATA FILE CREATED BY IMAGE
1110 !
1120 !
1130 ! ------------------------ PROGRAM SETUP -----------------------
1140 !
1150 !
1160 PRINT PAGE!"HOST.M UTILITY"!TAB(70)!"HP 9835/45"
1170 INTEGER A(30) !            ARRAY OF NODE DEVICE ADDRESSES
1180 DIM Eprom$[32640] !        REQUESTED RECORD BUFFER
1190 S=7
1200 !
1210 ! FIND ALL THE NODES
1220 !
1240 DISP "ATTEMPTING TO FIND HPIB DEVICES"
1260 C=0
1270 STATUS S!Dummy,Dummy,SZ   ! GET HP-IB CARD ADDRESS
1280 FOR S8=0 TO 30            ! CHECK ALL DEVICES
1290 IF SZ MOD 32=S8 THEN 1380 ! SKIP IF IT IS THE HP-IB CARD'S ADDRESS
1300 CONFIGURE S LISTEN = S8   ! THIS CONFIGURE ASSUMES 9835/45 IS TALKER
1310 STATUS S!Dummy            ! DROP ATN LINE
1320 STATUS S!Dummy,Dummy,Dummy,H  ! LOOK AT HANDSHAKE LINES
1330 H=H MOD 8 !                 LOOK FOR DAV=0 NDAC=1 NRFD=0
1340 IF H<>4 THEN GOTO 1380    ! IF NO RESPONSE THEN GO ON
1350 !                           I FOUND A RESPONDING DEVICE
1360 C=C+1 !                     INCREMENT COUNT OF DEVICES
1370 A(C)=S*100+S8 !             SAVE NODE ADDRESS
1380 NEXT S8
1390 IF C<>0 THEN GOTO 1420
```

```
1400 DISP "NO DEVICES WERE FOUND"
1410 STOP
1420 !
1430 ! OPEN DATA FILE
1440 !
1450 DISP "OPENING AND READING EPROM FILE"
1460 ASSIGN #1 TO "FILES"
1461 READ #1;Eprom$
1470 !
1480 !
1490 ! ------------------------ MAIN PROGRAM LOOP OF HOST -------------
1500 !
1510 !
1520 FOR N=1 TO C !              GO THROUGH ALL THE EXISTING NODES
1530 D=A(N)
1540 GOSUB 1600
1550 NEXT N
1560 GOTO 1520
1570 !
1580 !  SEE IF THE NODE WANTS SERVICE
1590 !
1600 GOSUB 1840 !                GET REQUEST
1610 IF T=1 THEN RETURN !        WHEN TIMEOUT HAPPENS - GO ON TO NEXT NODE
1620 GOSUB 1720 !                SEND B$
1630 GOTO 1600 !                 UNTIL THE READ REQUEST TIMES OUT
1640 !
1650 !
1660 ! ------------------------ DATA TRANSMISSION CODE ----------------
1670 !
1680 !
1690 !
1700 ! SEND DATA TO NODE
1710 !
1720 OUTPUT D USING "#,B,258A";2,Eprom$[A;258]
1730 SENDBUS S;"?_"              !       LEAVE THE BUS IN A CLEAN STATE
1740 STATUS S;Dummy  !                   BY UN-ADDRESSING EVERYONE
1750 RETURN
1760 !
1770 !
1780 ! ------------------------ DATA REQUEST CODE ---------------------
1790 !
1800 !
1810 !
1820 ! GET REQUEST FROM NODE
1830 !
1840 DISP "GET REQUEST FROM NODE "D
1850 T=0 !                       CLEAR TIMEOUT FLAG
1860 SET TIMEOUT S;1000 !        SET UP A TIMEOUT IN CASE THE NODE GOES DOWN
1870 ON INT #S GOTO 2040
1880 ENTER D USING "B,W,X";R,A   !   GET PROMPT CHARACTER ONLY
1890 IF R=1 THEN GOTO 1900
1891    DISP "BAD PROMPT FROM NODE"
1892    GOTO 1880
1900 A=A+1                  !            ADDRESS IS HANDLED AS ABSOLUTE BYTE ADDR
1910 SENDBUS S;"?_"             !          GET BUS INTO A CLEAN STATE
1920 STATUS S;Dummy  !                     BY UNADDRESSING EVERYONE
1930 DISP "TRANSMITTING RECORD STARTING WITH BYTE "A" TO NODE "D
1950 RETURN
1960 !
1970 !
1980 ! ------------------------ TIMEOUT ROUTINES ----------------------
1990 !
2000 !
2010 !
2020 ! REQUEST TIMEOUT
2030 !
2040 DISP "REQUEST TIMEOUT"
2060 T=1 !                       SET TIMEOUT FLAG
2070 RETURN
```

Chapter 8
HP 1000 Host Programs

Introduction

This chapter discusses and lists some networking related programs for the HP 1000 minicomputer family. These programs are for the HP 1000 as the HOST computer.

The programs discussed are:

UPLOD	Utility program to take the EPROM data file produced by IMAGE and convert into a form usable by host.
HOST	General purpose program that sends the data records requested by a single node computer.
SEND2	This is an HP 9915/85 program showing the modifications necessary to the SEND program of Chapter 5.

Possible Configurations

The HP 1000 series of mini-computers are good, high-performance machines. Because of their performance level and multi-tasking nature, they can handle being a host controller and other many other tasks. Some of the specific advantages that the HP 1000 series of mini-computer have over some other computers are:

- Good performance
- Wide range of configurations and performance
- Large memory
- Multi-programming capabilities
- Multi-lingual (FORTRAN, BASIC, ALGOL, Pascal, etc.)

All the programs for the HP 1000 are written in FORTRAN. This was used over BASIC or any other language because the HP 1000 systems come standard with a FORTRAN compiler.

The host program supports a single node computer over a serial interface. This host program reads the download data off of a random-access mass storage file into an integer array. This technique will read a set of 258 integers into an array and send them to the node. It would be possible to use a single large integer array for the data (similar to the programs for the 9835/9845). This alternate technique would require the extended memory feature of the FORTRAN compiler (which is only available on RTE-IVB and its successors).

To run the HP 1000 programs in this chapter, the HP 1000 needs to have the following minimum configuration (or its equivalent):

Hardware:

- HP 1000 M, E or F (this includes processor, disc, memory and the system monitor.)
- 12792A 8-channel Asynchronous Multiplexer Subsystem.

Software:

- RTE-IVB with Session Monitor.
- FTN4 FORTRAN Compiler.

The HP 9915 or HP-85 serving as a node can use the HP 82939A Option 001 serial interface.

Reading Suggestions

The HP 1000 series of mini-computers has extensive documentation. This list of suggested reading is intended for a beginner to the HP 1000 mini-computers.

Manuals for system introduction:

92068-90001	Getting Acquainted With RTE-IVB
02716-90008	Getting Started With Your HP 1000

Manuals for system configuration:

5953-4200	HP 1000 Computer Systems Technical Data
5953-4202	HP 1000 Computer Systems Hardware Data
5953-4244	HP 1000 Computer Systems Peripherals Data
5953-4259	HP 1000 Computer Systems Communications Products Technical Data
5953-4256	HP 1000 Computer Systems Software Technical Data
5953-4279D	HP 1000 Computational Products Ordering Information
92068-90007	RTE-IVB On-Line Generator Reference Manual

Manuals for general programming

92068-90004	RTE-IVB Programmer's Reference Manual
92610-93003	RTE FORTRAN IV Reference Manual
09610-93003	ISA FORTRAN Extension Package Reference Manual
92068-90002	RTE-IVB Terminal User's Reference Manual

Manuals for using the 12792A Serial Multiplexer

12792-90001	HP 12792A 8-channel Asynchronous Multiplexer Subsystem Installation and Reference Manual
12792-90002	HP 12792A 8-channel Asynchronous Multiplexer Subsystem User's Manual
12792-90003	HP 12792A 8-channel Asynchronous Multiplexer Subsystem Configuration Guide

Certain manual numbers may change in the HP 1000 documentation, so when possible, refer to the manuals by their title or content.

UPLOD Program

The UPLOD program is a utility program that receives the EPROM data file created by IMAGE into a file usable by the host program. This file is a set of 256 random access records. Each record contains 258 integer numbers.

Using UPLOD

The HP 1000 will need to have a session terminal available to run the UPLOD program. This terminal will communicate with you. There will be the seperate 12792A serial interface that will communicate with the node computer. The steps to use UPLOAD are as follows:

1. Determine the system configuration. What is the LU (logical unit) of the 12792A serial interface that will be used to upload the data.
2. Type the program into a text file using one of the HP 1000 text editors (such as EDITR or EDIT).
3. Compile the program. Run a relocating loader on the output of the compiler which produces the executable program module. These steps can be performed by the CLOAD command. The command would look like "RU,CLOAD,&UPLOD,-,-". This will take source file &UPLOD and compile the FORTRAN program, send the list output to the session terminal and place the executable code in UPLOD.
4. Run the UPLOD program; type in "RU,UPLOD"
5. Run the SEND program on the HP 9915 or HP-85. There will be certain modifications necessary. Refer to the last section of this chapter for the modifications. Refer to Chapter 5 for additional information on running SEND.
6. The program will ask for the file name of the file where you want to put the uploaded data. Type in name and press return. If the file does not exist, the program will create the file.

Note that the program uses ENQ/ACK character handshakes to upload the data. This handshake is necessary to prevent overrunning the 12792A interface. A complete upload of 32 kbytes of EPROM data will take approximately one hour at 9600 baud.

Modifications to UPLOD

There are no modifications that would greatly enhance the use of the UPLOD program.

```
FTN4,L
      PROGRAM UPLOD (),HP 9915 NETWORKING TECHNICAL SUPPLEMENT
C
C           PROGRAM    UPLOD
C           DATE       04/16/81
C           UPDATE     05/08/81
C
C           MACHINE    HP 1000 SERIES
C
C           PURPOSE    TO RECEIVE A SERIES OF PROGRAMS FROM A HP 9915
C                      OR HP 85 IN EPROM FORM.  THESE PROGRAMS WILL BE
C                      PUT ON A DATA FILE AS A SERIES OF INTEGER ARRAYS.
C                      THIS DATA IS RANDOM ACCESS AND WILL BE USED LATER
C                      BY 'HOST'.
C
C           ------------------------------------------------------------
C
C                      THE PROGRAM USES A CALL TO EXEC TO READ DATA FROM
C                      THE NODE.  THE DATA IS PUT INTO A BUFFER AND THEN
C                      READ USING THE FORTRAN READ STATEMENT.  THE NUMBER
C                      OF BYTES READ IS DETERMINED BY THE LAST PARAMETER
C                      IN THE EXEC CALL - A POSITIVE # SPECIFIES # OF
C                      WORDS AND A NEGATIVE # SPECIFIES THE # OF BYTES
C                      TO TRANSFER.
C
C           ------------------------------------------------------------
C
C                      THE PROGRAM EXPECTS OCTAL DATA ( ASCII FORMAT ).
C                      THE DATA IS PLACED IN THE DATA FILE AS ACTUAL
C                      BINARY INFORMATION.  IT WOULD BE POSSIBLE TO
C                      MODIFY THE HOST PROGRAM TO HANDLE THE BINARY
C                      DOWNLOAD PROTOCOL.
C
C           ------------------------------------------------------------
C
C     TERM IS DEVICE 1
C     NODE IS DEVICE 11    ( ASYNC MULTIPLEX CONNECTOR J7 )
C
C     DECLARE VARIABLES
C
      INTEGER EPROM(258),BUFFER(32)
      INTEGER IDCB(144),IERR,INAM(3),ISIZE(2),ITYPE
      INTEGER LOC,LOCT,L,SIZE,DUMMY
      INTEGER NODE,ICNWD,IPRM1,PROT
      REAL    OP
C
C     SET UP PROGRAM
C
      NODE=11
      WRITE (1,10)
 10   FORMAT("UPLOAD UTILITY",T70,"HP 1000")
C
C     CONFIGURE 12792A SERIAL MUX
C     USE ENQ/ACK HANDSHAKE ON MUX SERIAL INTERFACE
C         THE ENQ/ACK HANDSHAKE IS NECESSARY TO INSURE NO
C         DATA LOSS
C     SET PORT CHARACTERISTICS TO
C         8 BITS , 1 STOP BIT , NO PARITY
C         9600 BAUD , PORT # 7
C
      CALL EXEC(3,30008+NODE,152337B)
C
C     CLEAR SERIAL MUX BUFFER
C
      CALL EXEC(3,23008+NODE)
C
C     LOC IS THE DATA COUNTER FROM THE NODE
C     J   IS THE COUNTER TO CHECK WHEN TO WRITE A RECORD
C     K   IS THE RECORD COUNTER
```

```
C
      LOC=1
      J=1
      K=1
C
C     FILE CREATION
C     IF AN ERROR OCCURS DURING FILE CREATION -
C     ASSUMPTION IS MADE THAT THE FILE ALREADY EXISTS
C
      WRITE (1,15)
 15   FORMAT("ENTER EPROM DATA FILE NAME")
      READ  (1,20) (INAM(I),I=1,3)
 20   FORMAT(3A2)
C                          ISIZE(1) SPECIFIES THE # OF 128 WORD BLOCKS
      ISIZE(1)=512
C                          ISIZE(2) SPECIFIES THE RECORD SIZE IN WORDS
      ISIZE(2)=258
C                          ITYPE=2  SPECIFIES FIXED SIZE RANDOM ACCESS
      ITYPE=2
      CALL CREAT(IDCB,IERR,INAM,ISIZE,ITYPE)
      IF (IERR.GE.0) GOTO 1000
      CALL  OPEN(IDCB,IERR,INAM)
      IF (IERR.GE.0) GOTO 1000
      WRITE (1,35) IERR,(INAM(I),I=1,3)
 35   FORMAT("ERROR ",I6," HAS OCCURED TRYING TO OPEN FILE ",3A2)
      STOP
C
C     START UPLOAD
C
 1000 WRITE (1,1005) NODE
 1005 FORMAT("GETTING PROMPT FROM NODE ",I2)
      CALL EXEC(1,NODE,BUFFER,-11)
      WRITE (1,1007) (BUFFER(I),I=1,6)
 1007 FORMAT("THE PROMPT FROM THE NODE WAS <",6A2,">")
      CALL CODE
      READ  (BUFFER,1010) OP,PROT,SIZE
 1010 FORMAT(A4,A1,I4)
      IF (OP.EQ.4HDONE) GOTO 9000
      IF (OP.EQ.4HDATA) GOTO 3000
      WRITE (1,1015)
 1015 FORMAT("IMPROPER PROMPT")
      GOTO 1000
C
C     GET DATA
C
 3000 IF (PROT.EQ.2HA ) GOTO 3010
      WRITE (1,3005)
 3005 FORMAT("PROTOCOL OTHER THAN ASCII ATTEMPTED")
      STOP
 3010 WRITE (1,3015) LOC
 3015 FORMAT("ENTERING RECORD STARTING WITH BYTE ",I6)
      LOCT=LOC+SIZE-1
      DO 3500 L=LOC,LOCT
 3020 CALL EXEC(1,NODE,BUFFER,-8)
      CALL CODE
      READ  (BUFFER,3025) EPROM(J)
 3025 FORMAT(O6)
      J=J+1
      IF (J.LT.259) GOTO 3500
      WRITE (1,3030) K
 3030 FORMAT("WRITING RECORD ",I4)
      CALL WRITF(IDCB,IERR,EPROM,0,K)
      IF(IERR.GE.0) GOTO 3200
      WRITE (1,3035) IERR,K,(INAM(I),I=1,3)
 3035 FORMAT("ERROR ",I6," OCCURED WRITING RECORD ",I6," IN ",3A2)
 3200 K=K+1
      J=1
```

```
 3500 CONTINUE
      LOC=LOCT+1
      GOTO 1000
C
C     FINISHED
C
 9000 WRITE (1,9005)
 9005 FORMAT("FINISHING THE LAST RECORD")
      IF (J.EQ.1) GOTO 9010
      CALL WRITF(IDCB,IERR,EPROM,0,K)
      IF (IERR.LT.0) GOTO 9500
 9010 CALL CLOSE(IDCB,IERR)
      IF (IERR.LT.0) GOTO 9500
      WRITE (1,9015)
 9015 FORMAT("PROGRAM FINISHED")
      STOP
C
C     ERROR HANDLER
C
 9500 WRITE (1,9505) IERR,(INAM(I),I=1,3)
 9505 FORMAT("ERROR ",I6," HAS OCCURED IN FILE ",3A2)
      STOP
      END
```

HOST Program

The host program is a host program to service a single node computer. This program takes requests from the node and then accesses the appropriate random access record on the disc file. Only one record at a time is available to the program. This is very similar to the HP 9915/HP-85 HOST programs.

Using HOST

The HOST program is written to be used from a session terminal. To use the HOST program, perform the following steps:

1. Determine the system configuration. What is the LU (logical unit) of the 12792A serial interface that will be used for the program download.
2. Type the program into a text file using one of the HP 1000 text editors.
3. Compile and relocate the program. Type in something like "RU,CLOAD,&HOST,-,-".
4. Run the HOST program. Type in "RU,HOST".
5. The program will ask for the file name of the file where you want the download data to come from. Type in the name and press return.
6. Start up the node computers.

Modifications to HOST

There are several modifications that would enhance the use of HOST. Some of these modifications are:

- Change the program to use a single large array to contain all of the EPROM data. This will require the use of the EMA extended memory facility. This would give the program better performance.

- Change the program to not ask for the file name of the EPROM data. If you already know the name, this would make the program easier to use by someone else.
- Make the program the HP 1000 equivalent of an autostart program. This could allow the computer to start execution of the HOST duties with no operator intervention. This automatic start up is in the HP 1000 STARTUP or WELCOM facility.
- Use HP-IB as the communication path between the 1000 and the NODE. This could allow the computer to use binary protocol for the program download.

```
      PROGRAM HOST  (),HP 9915 NETWORKING TECHNICAL SUPPLEMENT
C
C          PROGRAM   HOST
C          DATE      04/16/81
C          UPDATE    05/08/81
C
C          MACHINE   HP 1000 SERIES
C
C          PURPOSE   TO SEND THE REQUESTED PROGRAM RECORDS TO A NODE.
C
C          ------------------------------------------------------------
C
C                    THIS PROGRAM SUPPORTS ASCII DOWNLOAD PROTOCOL.
C
C          ------------------------------------------------------------
C
C     TERM IS DEVICE 1
C     NODE IS DEVICE 11    ( MULTIPLEXER PORT J7 )
C
C     DECLARE VARIABLES
C
      INTEGER EPROM(258),BUFFER(600)
      INTEGER IDCB(144),IERR,INAM(3)
      INTEGER REQ,ADDR,L,LEN
      INTEGER NODE,ICNWD,IPRM1
C
C     SET UP PROGRAM
C
      NODE=11
      WRITE (1,10)
   10 FORMAT("HOST UTILITY",T70,"HP 1000")
C
C     CONFIGURE 12792A SERIAL MUX
C     TURN OFF ENQ/ACK HANDSHAKE
C     SET PORT CHARACTERISTICS TO
C          9600 BAUD , 8 BITS , 1 STOP BIT , NO PARITY
C          PORT J7
C
      CALL EXEC(3,30006+NODE,152137B)
C
C     OPEN DATA FILE
C
      WRITE (1,15)
   15 FORMAT("ENTER EPROM DATA FILE NAME")
      READ  (1,20) (INAM(I),I=1,3)
   20 FORMAT(3A2)
      CALL  OPEN(IDCB,IERR,INAM)
      CALL READF(IDCB,IERR,EPROM,258)
      IF (IERR.GE.0) GOTO 1000
   30 WRITE (1,35) IERR,(INAM(I),I=1,3)
   35 FORMAT("ERROR ",I6," HAS OCCURED TRYING TO OPEN FILE ",3A2)
      STOP
```

```
C
C     GET REQUEST FROM NODE
C
 1000 WRITE (1,1005) NODE
 1005 FORMAT("GETTING REQUEST FROM NODE ",I2)
      CALL EXEC(1,NODE,BUFFER,-9)
      WRITE (1,1007) (BUFFER(I),I=1,5)
 1007 FORMAT("THE REQUEST FROM THE NODE WAS <",5A2,">")
      CALL CODE
      READ  (BUFFER,101)) REQ,ADDR
 1010 FORMAT(A1,06)
      IF (REQ.EQ.2HR ) GOTO 3000
      WRITE (1,1015)
 1015 FORMAT("IMPROPER PROMPT")
      GOTO 1000
C
C     READ RECORD FROM FILE AND SEND IT TO NODE
C
 3000 ADDR=(ADDR/258)+1
      WRITE (1,3005) ADDR
 3005 FORMAT("SENDING RECORD ",I3)
      CALL READF(IDCB,IERR,EPROM,258,LEN,ADDR)
      IF (IERR.LT.0) GOTO 30
      CALL CODE
      WRITE (BUFFER,3025) (EPROM(L),L=1,258)
 3025 FORMAT("D",258(O3,","))
      CALL EXEC(2,NODE,BUFFER,-1033)
      GOTO 1000
C
C     THE PREVIOUS LOOP IS CONTINUOUSLY EXECUTED
C
      END
```

Modified SEND Program

The HP 9915/HP-85 SEND program needs to be modified to use ENQ/ACK character handshake. This is necessary to prevent buffer overrun of the HP 1000 serial interface. Note that the host program does not use ENQ/ACK handshake.

The SEND2 program that is on the following page is the SEND program from Chapter 5 with the following modifications:

Program Change	Reason for the Program Change
`1150 S=10`	Set select code to RS-232 standard.
`1151 RESET S`	Put card into clean empty state.
`1152 CONTROL S,3 ; 15,3`	Set card to 9600 baud, 8 bit data 1 stop bit, no parity.
`1153 CONTROL S,16 ; 1`	Set the end of line sequence of the RS-232 card to carriage return only. The HP 1000 will generate a line feed in its buffer when it receives a carriage return.

`1361 GOSUB 9000`
`1391 GOSUB 9000`
`1510 GOSUB 90000`
	The GOSUB 9000 goes to a subroutine before every OUTPUT to perform the ENQ/ACK handshake.

`1511 OUTPUT S USING` `" 4A,X,4D" ; "DONE" ;0`	Has to be re-entered because of the GOSUB 9000 on line 1510.

`9000 ENTER S USING`
`"#,B" ; A9`
`9010 IF A9=5 THEN`
`SEND S ; DATA 6`
`9020 IF A9#17 THEN`
`GOTO 9000`
`9030 RETURN`
	Lines 9000 to 9030 is a subroutine that reads characters from the 1000. If the character was an ENQ then an ACK is sent. If the character received was a DC1, then the 1000 is ready to receive characters again.

```
1000 !
1010 !  PROGRAM   SEND2
1020 !
1030 !  DATE      03/31/81
1040 !  UPDATE    05/08/81
1050 !
1060 !  COMPUTER  HP 9915A , HP 85A
1070 !
1080 !  PURPOSE   PROGRAM UPLOAD UTILITY FOR THE 9915A OR 85A
1090 !            MODIFIED FOR USE WITH THE HP 1000 ( ENQ/ACK )
1100 !
1110 DIM D$[40961] !              STRING VARIABLE FOR READING EPROM FILES
1120 CCLEAR
1130 DISP "SEND - UPLOAD UTILITY  HP 9915A"
1140 DISP
1150 S=10 !                       SELECT CODE FOR UPLOADING DATA
1151 RESET S
1152 CONTROL S,3 ; 15,3 !           SET UP 9600 BAUD , 8 BITS , NO PARITY
1153 CONTROL S,16 ; 1 !             SEND ONLY CR - 1000 GENERATES THE LF
1160 !
1170 DISP "ENTER FILE NAME OF EPROM DATA"
1180 INPUT F$
1190 DISP "ENTER EPROM TYPE SPECIFIED IN IMAGE"
1200 DISP @ DISP "2716" @ DISP "2732"
1210 CCURSOR CCPOS-96
1220 INPUT Z
1230 IF NOT (Z=2716 OR Z=2732) THEN GOTO 1190
1240 IF Z=2716 THEN Z=2048
```

```
1250 IF Z=2732 THEN Z=4096
1260 DISP "ENTER UPLOAD FORM - A for ascii or B for binary"
1270 INPUT M$
1280 !
1290 !                          READ DATA AND TRANSMIT TO HOST
1300 !
1310 ASSIGN# 1 TO F$
1320 ON ERROR GOTO 1460
1330 !
1340 !                          READ DATA AND TRANSMIT TO HOST
1350 !
1360 READ# 1 ; D$
1361 GOSUB 9000
1370 OUTPUT S USING "4A,A,4D" ; "DATA"&M$&Z
1380 IF M$="B" THEN OUTPUT S USING "#,K" ; &D$[1,Z] @ GOTO 1360
1390 FOR I=1 TO Z
1391 GOSUB 9000
1400 OUTPUT S USING "K" ; DTO$(NUM(D$[I,I]))
1410 NEXT I
1420 GOTO 1360
1430 !
1440 !                          TELL HOST , UPLOAD IS DONE
1450 !
1460 IF ERRN=71 THEN GOTO 1510
1470 IF ERRN=72 THEN GOTO 1510
1480 DISP "AN ERROR HAS OCCURED"
1490 DISP "ERROR "&ERRN&" IN LINE "&ERRL
1500 STOP
1510 GOSUB 9000
1511 OUTPUT S USING "4A,X,4D" ; "DONE"&0
1520 DISP "UPLOAD IS COMPLETE"
1530 END
9000 ENTER S USING "#,B" ; A9
9010 IF A9=5 THEN SEND S ; DATA 6
9020 IF A9#17 THEN GOTO 9000
9030 RETURN
```

Possible Configurations

The 9826 computers are good, high-performance machines. Because of their performance level, they can handle the job of being a host and other tasks as well. Some of the specific advantages that the 9826 computer has over some other computers are:

- Good performance.
- Very easy to program and use.
- Graphics.
- Built in HP-IB interface.Multiple languages (BASIC, HPL, Pascal)

The HOST_S program supports a single node computer over a serial or HP-IB interface. This host program reads the download data off of a mass storage file only once per program execution. Once the program is running, the host program uses a large string (32 kbytes) as the download storage area.

The HOST_M program supports multiple node computers on a single HP-IB interface. This host program also reads the download data off of a mass storage file once per program execution and then uses a string for the storage area. It is possible to modify the program to use several different interfaces, with some being HP-IB and some being serial interfaces.

Both the HOST_M and HOST_S program are written to expect only binary prompts from the NODE computers. The 9826 series computers can handle ASCII protocol but it would slow down the download process.

The programs included here are for the BASIC language 9826. The HPL and Pascal systems could support host duties.

The 9826 comes standard with a built-in HP-IB interface. This is the interface that was used for the following programs. The 9826 computer has optional interfaces that can be added. The interfaces that might be of some use to a networking system are:

- 98624A HP-IB interface
- 98626A Asynchronous Serial interface
- 98622A GPIO 16-bit parallel interface
- 98620A DMA Controller (for use with the internal and external HP-IB and the GPIO interfaces

Chapter 9
HP 9826 Host Programs

Introduction

This chapter discusses and lists some network related programs for the 9826 Computer. These programs are for a 9826 as the host computer.

The programs discussed are:

UPLOAD	Utility program to take the EPROM data file produced by IMAGE and convert into a form usable by HOST_S and HOST_M.
HOST_S	General purpose program that sends the data records requested by a single node computer.
HOST_M	General purpose program that sends the data records requested by several node computers.

```
1000 !
1010 !      PROGRAM      UPLOAD
1020 !
1030 !      DATE         05/12/81
1040 !      UPDATE       05/12/81
1050 !
1060 !      MACHINE      HP 9826
1070 !
1080 !      PURPOSE      TO RECEIVE A SERIES OF PROGRAMS FROM AN HP 85 OR
1090 !                   HP 9915 IN EPROM FORM.  THESE PROGRAMS WILL BE PUT
1100 !                   ON A DATA FILE AS ONE LARGE STRING VARIABLE.  THIS
1110 !                   DATA FILE WILL BE USED LATER BY 'HOST_S' AND 'HOST_M'.
1120 !
1130 DIM Eprom$[32640]                    !    EPROM DATA BUFFER
1140                                      !    NOTE THAT THIS STRING IS ONLY 32640
1150                                      !    BYTES LONG.   THE LAST 128 BYTES OF
1160                                      !    PROM ARE NOT A COMPLETE RECORD AND
1170                                      !    ARE NOT USEABLE - SO THEY ARE NOT
1180                                      !    EVEN DECLARED.
1190 DIM Temp$[4096]                      !    TEMPORARY READ BUFFER
1200 !
1210 ! PUT UP MESSAGES
1220 !
1230 PRINT CHR$(12)
1240 PRINT "CREATE FILE UTILITY - UPLOAD";TAB(40);"HP 9826"
1250 PRINT
1260 !
1270 !    VARIABLE INITIALIZATION
1280 !
1290 Location=1
1300 S=7                                  !    INTERFACE SELECT CODE
1310 Node=723                             !    DEVICE ADDRESS
1320 !
1330 ! CREATE DATA FILE
1340 !
1350 OFF ERROR
1360 F$="FILES"
1370 INPUT "ENTER EPROM DATA FILE NAME ( default : FILES )",F$
1380 DISP "CREATING DATA FILE"
1390 ON ERROR GOTO 1410
1400 CREATE BDAT F$,130                   !    SPACE FOR A 32K BYTE STRING
1410 OFF ERROR
1420 !
1430 !    START UPLOAD
1440 !
1450 Loop:     !
1460 DISP "GETTING PROMPT FROM NODE"
1470 ENTER Node USING "4A,A,4D";Op$,Protocol$,Size
1480 IF Op$="DONE" THEN GOTO Finished
1490 IF Op$="DATA" THEN GOTO Data
1500 PRINT "IMPROPER PROMPT RECEIVED"
1510 GOTO Loop
1520 !
1530 !    GET DATA
1540 !
1550 Data:     !
1560 IF Protocol$="B" THEN GOTO 1590
1570    PRINT "PROTOCOL OTHER THAN BINARY ATTEMPTED"
1580    STOP
1590 Image$="#,"&VAL$(Size)&"A"
1600 DISP "ENTERING RECORD STARTING WITH BYTE ";Location
1610 ENTER Node USING Image$;Temp$
1620 IF Location+Size<32640 THEN Eprom$[Location;Size]=Temp$
1630 IF Location+Size>=32640 THEN Eprom$[Location;Size-128]=Temp$
1640 Location=Location+Size
1650 GOTO Loop
1660 !
1670 !    FINISHED
1680 !
```

```
1690 Finished: !
1700 DISP "PLACING DATA ON TAPE"
1710 ASSIGN @File TO F$                    !    OPEN THE FILE
1720 OUTPUT @File;Eprom$                   !    WRITE THE DATA TO THE FILE
1730 ASSIGN @File TO *                     !    CLOSE THE FILE
1740 DISP "PROGRAM FINISHED"
1750 PRINT "PROGRAM FINISHED"
1760 END
```

HOST_S Program

The HOST_S program is a host program to service a single node computer.

Using HOST_S

The HOST_S program does not require operator interaction. The program needs to be loaded and run; however, it can be an autostart file on the floppy.

Modifications to HOST_S

There are several modifications that would enhance the use of HOST_S. Some of these modifications are:

- Rather than using ENTER handshake and the timeout facility, use interrupt transfers. This allows the HOST computer more time to do other tasks.
- If there is not enough R/W memory for the Eprom$ string it is possible to change the data file to a random access file (similar to the HP 9915/85-HOST programs). The data would then be brought in 258 bytes at a time when requested.

```
1000 !
1010 !  PROGRAM    HOST_S
1020 !
1030 !  DATE       05/12/81
1040 !  UPDATE     05/12/81
1050 !
1060 !  MACHINES   HP 9826
1070 !
1080 !  PURPOSE    TO DOWNLOAD THE PROGRAM FILE THAT A NODE
1090 !             REQUESTED
1100 !
1110 !
```

```
1120 ! ------------------------ PROGRAM SETUP ------------------------
1130 !
1140 !
1150 PRINT CHR$(12);"HOST_S UTILITY ";TAB(40);"HP 9826"
1160 DIM Eprom$[32640]          ! REQUESTED RECORD BUFFER
1170 S9=7 !                     SELECT CODE
1180 S=723 !                    DEVICE SPECIFIER ( FOR HP-IB USE )
1190 !
1200 ! OPEN DATA FILE
1210 !
1220 ASSIGN @File TO "FILES" !  OPEN DATA FILE
1221 ENTER @File;Eprom$         ! READ EPROM DATA
1230 !
1240 !
1250 ! ------------------------ MAIN PROGRAM LOOP OF HOST ------------
1260 !
1270 !
1280 !
1290 ! SERVICE THE SINGLE NODE COMPUTER
1300 !
1310 GOSUB 1520 !               GET REQUEST
1330 DISP "SENDING RECORD STARTING WITH BYTE ";A
1340 GOSUB 1410 !               SEND DATA
1350 GOTO 1310
1360 !
1370 !
1380 ! ------------------------ SEND PROM RECORD ---------------------
1390 !
1400 !
1410 ON TIMEOUT S9,5 GOTO 1690 ! SET UP A TIMEOUT IN CASE THE DEVICE GOES DOWN
1430 OUTPUT S USING "#,B,258A";2,Eprom$[A;258]
1440 SEND S9;CMD "?_"           !  LEAVE THE BUS IN A CLEAN STATE
1460 RETURN
1470 !
1480 !
1490 ! ------------------------ GET NODE REQUEST ---------------------
1500 !
1510 !
1520 DISP "GET REQUEST FROM NODE"
1530 ON TIMEOUT S9,10 GOTO 1750  !  SET UP A TIMEOUT IN CASE THE NODE GOES DOWN
1550 ENTER S USING "B,W,X";R,A   !  GET PROMPT AND ADDRESS
1560 IF R=1 THEN GOTO 1570
1561    DISP "BAD PROMPT FROM NODE"
1562    GOTO 1550
1570 ! THIS PROGRAM HANDLES THE REQUESTS AS AN ABSOLUTE BYTE ADDRESS
1571 A=A+1
1580 SEND S9;CMD "?_"           !  LEAVE THE BUS IN A CLEAN STATE
1600 RETURN
1610 !
1620 !
1630 ! ------------------------ TIMEOUT ROUTINES ---------------------
1640 !
1650 !
1660 !
1670 ! TIMEOUT FOR OUTPUT
1680 !
1690 DISP "OUTPUT TIMEOUT"
1710 RETURN !                   RETURN FROM SEND DATA ROUTINE
1720 !
1730 ! REQUEST TIMEOUT
1740 !
1750 DISP "REQUEST TIMEOUT"
1770 GOTO 1520 !                TRY TO GET ANOTHER REQUEST
1780 END
```

HOST_M Program

The HOST_M program is a host program to service several node computers. The node computers are connected over an HP-IB bus with the HOST being system controller.

Using HOST_M

The HOST_M program does not require operator interaction. The program needs to be loaded and run; however, it can be an autostart file on tape. The comments in the Using HOST_S section in this chapter on autostarting programs also apply to the HOST_M program.

Modifications to HOST_M

There are several modifications that would enhance the use of HOST_M. Some of these modifications are:

- Node computers on different HP-IB buses or on several serial interfaces could be included in the HOST_M service cycle. Modify the HOST_M program to set up the proper device addresses in the device array and set up the proper select code in the timeout setup.
- Change the data file to a random access file (if there is not enough R/W memory).

```
1000 !
1010 !  PROGRAM    HOST_M
1020 !
1030 !  DATE       05/12/81
1040 !  UPDATE     05/12/81
1050 !
1060 !
1070 !  MACHINES   HP 9826
1080 !
1090 !  PURPOSE    TO DOWNLOAD THE PROGRAM FILE THAT ANY ONE OF SEVERAL NODES
1100 !             HAS REQUESTED - USING A DATA FILE
1110 !
1120 !
1130 ! ---------------------- PROGRAM SETUP -------------------------
1140 !
1150 !
1160 PRINT CHR$(12);"HOST_M UTILITY";TAB(40);"HP 9826"
1170 INTEGER Node(30) !          ARRAY OF NODE DEVICE ADDRESSES
1180 DIM Eprom$[32640]  !        REQUESTED RECORD BUFFER
1190 S=7
1200 !
1210 ! FIND ALL THE NODES
1220 !
1240 DISP "ATTEMPTING TO FIND HPIB DEVICES"
1260 C=0
1270 S2=21                       ! GET HP-IB CARD ADDRESS
1280 FOR S8=0 TO 30              ! CHECK ALL DEVICES
1281 ON TIMEOUT S,.2 GOTO 1371! SET UP TIMEOUT FOR SPOLL
1290 IF S2 MOD 32=S8 THEN 1380! SKIP IF IT IS THE HP-IB CARD'S ADDRESS
1300 Dummy=SPOLL(S8+S*100)       ! CHECK IF THE DEVICE EXISTS
1360 C=C+1 !                       INCREMENT COUNT OF DEVICES
1370 Node(C)=S*100+S8 !            SAVE NODE ADDRESS
1371 ABORT S
1380 NEXT S8
1390 IF C<>0 THEN GOTO 1420
1400 DISP "NO DEVICES WERE FOUND"
1410 STOP
```

```
1420 !
1430 ! OPEN DATA FILE
1440 !
1450 DISP "OPENING AND READING EPROM FILE"
1460 ASSIGN @File TO "FILES"
1461 ENTER @File;Eprom$
1470 !
1480 !
1490 ! ------------------------ MAIN PROGRAM LOOP OF HOST -------------
1500 !
1510 !
1520 FOR N=1 TO C !              GO THROUGH ALL THE EXISTING NODES
1530 D=Node(N)
1540 GOSUB 1600
1550 NEXT N
1560 GOTO 1520
1570 !
1580 !  SEE IF THE NODE WANTS SERVICE
1590 !
1600 GOSUB 1840 !                GET REQUEST
1610 IF T=1 THEN RETURN !        WHEN TIMEOUT HAPPENS - GO ON TO NEXT NODE
1620 GOSUB 1720 !                SEND B$
1630 GOTO 1600 !                 UNTIL THE READ REQUEST TIMES OUT
1640 !
1650 !
1660 ! ------------------------ DATA TRANSMISSION CODE ----------------
1670 !
1680 !
1690 !
1700 ! SEND DATA TO NODE
1710 !
1720 OUTPUT D USING "#,B,258A";2,Eprom$[A;2581
1730 SEND S;CMD "?_"             !       LEAVE THE BUS IN A CLEAN STATE
1750 RETURN
1760 !
1770 !
1780 ! ------------------------ DATA REQUEST CODE ---------------------
1790 !
1800 !
1810 !
1820 ! GET REQUEST FROM NODE
1830 !
1840 DISP "GET REQUEST FROM NODE "D
1850 T=0 !                           CLEAR TIMEOUT FLAG
1860 ON TIMEOUT 5,1 GOTO 2040        ! SET UP A TIMEOUT IN CASE THE NODE GOES DOWN
1880 ENTER D USING "B,W,X";R,A   !   GET PROMPT CHARACTER ONLY
1890 IF R=1 THEN GOTO 1900
1891    DISP "BAD PROMPT FROM NODE"
1892    GOTO 1880
1900 A=A+1                   !       ADDRESS IS HANDLED AS ABSOLUTE BYTE ADDR
1910 SEND S;CMD "?_"             !   GET BUS INTO A CLEAN STATE
1930 DISP "TRANSMITTING RECORD STARTING WITH BYTE "A" TO NODE "D
1950 RETURN
1960 !
1970 !
1980 ! ------------------------ TIMEOUT ROUTINES ----------------------
1990 !
2000 !
2010 !
2020 ! REQUEST TIMEOUT
2030 !
2040 DISP "REQUEST TIMEOUT"
2060 T=1 !                           SET TIMEOUT FLAG
2070 RETURN
2080 END
```

Appendix

Introduction

This appendix discusses and lists some networking related information that might be of use in developing networking applications:

- Computing an SDLC checksum
- Common errors in the NODE 9915
- Networking Bibliography

Computing an SDLC Checksum

The HOST and UPLOAD programs in this technical supplement do not check the incoming data for checksum errors. It would be useful to modify the programs to check the records. This section will briefly discuss what the checksum is and how to compute it.

SDLC stands for Synchronous Data Link Control which is a series of rules for transferring data using synchronous transmissions. The SDLC technique allows for point-to-point, multi-point, or loop arrangements. This system for communication includes a specific approach to error detection: the SDLC cyclic redundancy check checksum.

This checksum is implemented with a 16-bit register that is set initially to all ones. The register is a shift register that takes one bit at a time from the input character stream. The last bit that is shifted out is fed back through a group of exclusive-or operations into the register at various points.

The BASIC program on the next page calculates the 2-byte SDLC checksum. This program can be easily re-written in most langauges. The BINEOR function (binary exclusive-or) is used in this program. The HP-85/9915 and HP 9835/9845 desktop computers have this function in the I/O ROMs of the computers. For machines that do not have the exclusive-or capability, the following BASIC program will produce the same effect:

Error 116
- HP-IB is not active listener.
- If this occurs during a download operation, the node computer is not being allowed by the host to fully receive the PROM record. Check the format specifications (IMAGE or FORMAT) in the host and the node. Use a bus analyzer (HP 59401A or similar) to actually see how much data and in what form is being transferred. Also check the end-of-line characters being defined in the node.

From the I/O ROM and the serial interface:

Error 113
- UART receiver overrun; data has been lost.
- This will occur when the baud rate has been set too high for the interface to keep up with. Lower the baud rate.

Error 114
- Receiver buffer overrun ; data has been lost.
- This can occur when the baud rate has been set too high for the computer (HP-85/HP 9915) to keep up with. Lower the baud rate.
- This can also occur when data comes from the host to the node when the node is not expecting the data. (The internal card buffer can only contain 33 characters.) Check the program flow.

Error 115
- Automatic disconnect has occurred.
- This will occur when the host drops modem lines to the node. Check the modem control in the host program.

```
1000 ! A=OPERAND # 1
1010 ! B=OPERAND # 2
1020 ! Z=RESULT
1030 Z=A
1040 IF A=0 THEN Z=0
1050 RETURN
1060 !
1000 ! PROGRAM CHKSUM
1020 ! DATE 09/20/80
1030 ! UPDATE 04/27/81
1040 !
1050 ! MACHINES HP 85, HP 9915, HP 9835A/B, HP 9845A/B
1060 !
1070 ! PURPOSE TO SHOW HOW TO GENERATE AN SDLC CHECKSUM
1080 !
1090 DISP "CHECKSUM EXAMPLE"
1100 !
1110 ! MAIN PROGRAM
1120 !
1130 S$="12"
1140 GOSUB 1230
1150 DISP "CALCULATED SDLC CHECKSUM IS: ";IC$
1160 END
1170 !
1180 ! CHECKSUM OPERATION
1190 !
1200 ! S$ CONTAINS THE DATA STRING
1210 ! AT EXIT C$ CONTAINS THE 2 BYTE CHECKSUM
1220 !
1230 DIM S(16)
1240 FOR I=1 TO 16
1250 S(I)=1 ! INITIALIZE CRC SHIFT REGISTER
1260 NEXT I
1270 FOR I=1 TO LEN(S$) ! LOOP FOR CHARACTERS IN STRING
1280 FOR J=0 TO 7 ! LOOP FOR BITS IN CHARACTER
1290 B=BIT(NUM(S$[I,I]),J) ! GET NEXT BIT FOR CRC CALCULATION
1300 X=BINEOR(S(1),B)
1310 FOR K=2 TO 16
1320 S(K-1)=S(K) ! SHIFT THE DATA
1330 NEXT K
1340 S(16)=X ! PUT IN THE EXCL.ORed BIT
1350 S(4)=BINEOR(S(4),X) ! PUT IN THE EXCL.ORed BIT
1360 S(11)=BINEOR(S(11),X) ! PUT IN THE EXCL.ORed BIT
1370 NEXT J
1380 NEXT I
1390 V=0
1400 FOR I=1 TO 16 ! CONVERT FROM ARRAY TO DECIMAL VALUE
1410 IF S(I)=1 THEN V=V*2+0 ! NOTE COMPLEMENTING OPERATION
1420 IF S(I)=0 THEN V=V*2+1 ! THAT OCCURS HERE
1430 NEXT I
1440 C$=CHR$(INT(V/256))&CHR$(V MOD 256)
1450 RETURN
```

Common Errors

During the development of a networking system, there are some common errors that you might encounter. Some of these errors and how to approach correcting them are:

From the PD ROM:

`Error 114 No user PROM`

- The unit has no internal EPROM board.
- This problem generally occurs when attempting to load a program from EPROM in an HP-85. The program can be rewritten to do an ON ERROR branch and execute a PROM IS TAPE and then try the PLOADGO, PCHAIN, or PLOADBIN again.

`Error 116 PROM or I/O read`

- A checksum error has occurred.
- The failure could be in the actual transmission which means a slower baud rate on RS-232 must be used or there is inadequate shielding on HP-IB.
- The internal EPROM could be bad, due to bad EPROMs, a bad EPROM board (unlikely), EPROMs placed improperly in their sockets, the EPROM board improperly installed, or there was an error during the IMAGE/EPROM burning phase.

From the I/O ROM:

`Error 124 ISC`

- There is no interface at the specified select code.
- The interface might not be connected. Make sure that the interface is physically installed in the computer. The interface can appear to be in the machine without electrical connection occurring.
- The interface might be at the wrong select code. Either open up the interface and check the switches or perform a STATUS operation.

From the I/O ROM and the HP-IB interface:

`Error 114`

- HP-IB is not active controller.
- If the PROM IS specified an address (i.e., 713) then the card must be able to address the bus. Check the system configuration. OUTPUT and ENTER operations in the UPLOAD and host programs will generate a similar problem with the wrong configuration.

`Error 115`

- HP-IB is not active talker.
- If this occurs during a download operation, the node computer is not being allowed by the host to fully complete the request. Check the format specifications (IMAGE or FORMAT, in the host and the node. Use a bus analyzer (HP 59401A or similar) to actually see how much data and in what form is being transferred. Also check the end-of-line characters being defined in the node.
- If this occurs during an OUTPUT or TRANSFER, the 9915 is being un-addressed during the data output. Check the format specifications.

Networking Bibliography

This technical supplement does not attempt to fully cover all aspects of networking. The articles and books listed in the bibliography are intended to help introduce you to networking concepts. These references are also a very good source for additional materials because most have a good list of references.

1. **Technical Aspects of Data Communication** by John McNamara. Copyright 1977, 1978 by Digital Equipment Corporation. Published by Digital Equipment Corporation.
2. **A Review of Classification Schemes for Computer Communication Networks** by Maj. William Greene and Udo W. Pooch. November 1977 Computer. Volume 10 Number 11. Pages 12-21. Published by the Institute of Electrical and Electronics Engineers.
3. **Computer Interconnection Structures: Taxonomy, Characteristics and Examples** by G. A. Anderson and E. D. Jensen. December 1975 ACM Computing Surveys. Volume 7 Number 4. Pages 197-214. Published by the Association for Computing Machinery.
4. **Computer Communications Networks: Approaches, Objectives, and Performance Considerations** by S. R. Kimbleton and G. M. Schneider. September 1975 ACM Computing Surveys. Volume 7 Number 3. Pages 129-173. Published by the Association for Computing Machinery.
5. **Computer Communications: HP 9825 - HP 1000** Hewlett Packard Application Note 201-6, revised.