AD-A038 433 COMPUTER SCIENCES CORP FALLS CHURCH VA F/G 9/2
DESIGN PLAN FOR GCOS/ARPANET INTERFACE.(U)
MAR 77
F30602-76-C-0199
UNCLASSIFIED RADC-TR-77-87 NL

1 OF 2
AD
A038433

12 B.S.

RADC-TR-77-87
Final Technical Report
March 1977

DESIGN PLAN FOR A GCOS/ARPANET INTERFACE

Computer Sciences Corporation

ROME AIR DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
GRIFFISS AIR FORCE BASE, NEW YORK 13441

This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public including foreign nations.

This report has been reviewed and is approved for publication.

APPROVED: Thomas F. Lawrence

THOMAS F. LAWRENCE
Project Engineer

APPROVED: Robert D. Krutz

ROBERT D. KRUTZ, Col, USAF
Chief, Information Sciences Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: RADC-TR-77-87	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): DESIGN PLAN FOR GCOS/ARPANET INTERFACE	5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report, 12 Mar 76 - 12 Dec 76
	6. PERFORMING ORG. REPORT NUMBER: N/A
7. AUTHOR(s)	8. CONTRACT OR GRANT NUMBER(s): F30602-76-C-0199
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Computer Sciences Corporation, 6565 Arlington Blvd, Falls Church VA 22046	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 63728F, 55500828
11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (ISCP), Griffiss AFB NY 13441	12. REPORT DATE: March 1977
	13. NUMBER OF PAGES: 196
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Thomas Lawrence (ISCP)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

ARPANET
NSW
Networking
GCOS
Network Front-End

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report contains a specification of the hardware/software interface between the Honeywell H635/6180 GCOS System and a PDP 11/35 ELF System for the purpose of providing ARPANET/NSW access.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

405 717

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

Section 11 - Cont'd

LIST OF ILLUSTRATIONS

Figure

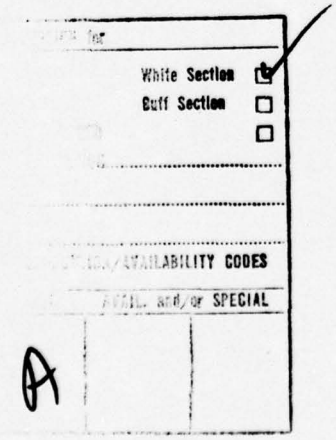
LIST OF TABLES

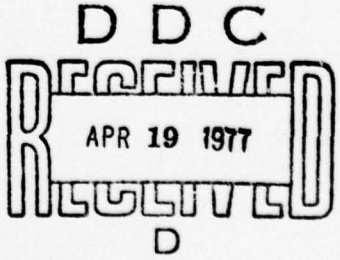
Table

EVALUATION

This report will be used in support of the on-going National Software Works Project at RADC. The purpose of NSW is to produce a software production environment by making various software production tools existing on the ARPANET readily available to software production personnel. This report will be used as the basis of an effort to incorporate resources that exist under GCOS as tools within the NSW environment.

THOMAS F. LAWRENCE
Project Engineer





SECTION 1 - INTRODUCTION

1.1 PURPOSE

This document defines functional requirements to interface the Advanced Research Projects Agency Network (ARPANET) with a Honeywell 6000 Computer (H6000) using the General Comprehensive Operating Supervisor (GCOS). The interface design includes intermediate hardware and software, primarily located in a PDP-11 minicomputer acting as a Network Front-End (NFE).

An H6000 interfaced with the ARPANET will provide many benefits. These include providing remote user access to existing GCOS software facilities and establishing the means for Department of Defense (DOD) and Air Force (AF) access to many high-quality Rome Air Development Center (RADC) software programs. The ARPANET interface is an initial phase in a long range program aimed at an ultimate interface of GCOS with the National Software Works (NSW).

1.2 SCOPE

This document includes all functions outlined in the Statement of Work (SOW). The GCOS to ARPANET and a subsequent NSW interface will provide the following capabilities for users with appropriate access permissions:

1. Terminal access to GCOS. ARPANET terminal users will be directly supported by GCOS and appear to GCOS as local terminals.
2. File transfer facility. Any ARPANET host computer user will be able to transfer files with the RADC GCOS file system.
3. Remote Job Entry (RJE). An ARPANET user will be able to submit RJE jobs to GCOS.
4. NSW compatibility. The ARPANET interconnection and software design will facilitate a future connection to the NSW as a Tool Bearing Host (TBH).

The GCOS/ARPANET interface analysis and design has been conducted in two phases. During the first phase, four hardware connection alternatives were analyzed

by Computer Sciences Corporation (CSC). An interim report was prepared which detailed the benefits and limitations of each approach. The interim report recommended a specific hardware connection rationale (identified in Section 2). The second phase of the effort was initiated when RADC, as project sponsor, approved the selected hardware configuration.

This document addresses those software issues required to specify the recommended interface methodology. The primary data contained herein focuses on the modification and development requirements for the H6000, GCOS, and the PDP-11 software.

The NFE will provide a capability for the RADC GCOS system to appear as an ARPANET host. During the period of this design effort, it became obvious that there was a fluid issue involved in the interface of any host to the NSW. At issue is the choice of the best operating system for the NFE minicomputer. Under the scope of the contract, we have been furnished substantive Real Time Operating System (ELF) data, but we have not limited our research to ELF alone. We have determined that there are a number of on-going NFE projects using a variety of executive structures. Many of these related projects are under intensive evaluation, even as this document is being prepared. This effort looks at the NFE as a resource, against which certain functions may be allotted. No recommendation is formulated for a specific NFE Operating System. A discussion of the GCOS/NFE facility, in light of its future role as an NSW participating host is included.

1.3 BACKGROUND

RADC is a participant in an effort to accomplish research in, development of, and user access to high quality software programs, referred to as tools. Studies indicate that the cost-spiral for new software may reach as high as 90 percent of certain Electronic Data Processing (EDP) budgets within the next decade. The problem is magnified in most cases because software tools remain native to the host

computer installation for which they were created. There is no current method for easily disseminating this technology so that other users may benefit.

Tools which offer widespread benefits to the AF and DOD include compilers, editors, simulators, and debuggers. The cost benefits of sharing these tools may be viewed as twofold: first, the higher quality design and implementation will provide a quicker, more reliable software tool. Second, there are long range benefits associated with the life-cycle utility of these tools.

To achieve the goal of providing a wide base of user access to software tools, RADC is involved in a research activity called the NSW. The NSW is an ARPANET based facility which will provide technical and managerial support in the development and use of software tools. Primarily, NSW provides the means to communicate requests for and access to requested tools and files.

The H6000 computer, utilizing the GCOS operating system, provides many potential tools which would benefit the goals of RADC's research and development program. Further, these tools would be of benefit to the AF and DOD communities. Therefore, the H6000/GCOS facility is a likely candidate for future connection to the NSW network as a TBH.

The immediate consideration is to develop a non-disruptive interface between GCOS and the ARPANET. A non-disruptive interface is defined as one which requires no modification to existing user programs, and only minimal modifications to GCOS. The selected strategy is to use a PDP-11 minicomputer which performs necessary networking functions for GCOS. This is referred to as an NFE.

It is anticipated that the design and subsequent implementation of the GCOS/NFE/ARPANET interconnection will be an important step toward a future NSW interface.

1.4 SUMMARY OF CONCLUSIONS

Section 11 of this report presents an in-depth summation of findings surrounding the GCOS interface design. Results include the following findings:

1. The interconnection of GCOS and the ARPANET via an NFE is a desirable, feasible interface.

2. Software development is required in both the H6000 and the PDP-11 to support this interface.

   a. Modification of GCOS to intercept network Input/Output (I/O) is achievable with minimal impact, and is best accomplished by implementing a new program to simulate Datanet 355 (DN355) terminal processes.

   b. Most networking, overhead to the H6000, can be off-loaded by implementing ARPANET protocols in a PDP-11, acting as the NFE.

1.5 ORGANIZATION OF INTERFACE DOCUMENT

An overview of this document is provided below for project sponsors or other readers who may have parochial interest in isolated areas of this paper.

There are ten sections in this report, exclusive of this introduction. Section 2 discusses Project Activities. A review of the interim report is provided for two purposes. First, the selected and approved hardware interface requirements are briefly reviewed. Next, information is presented which addresses how certain earlier hardware and software recommendations have been influenced by further analysis. Of particular note is modification of the interface approach - that is, the specific point of interception of network output from GCOS.

Section 3 discusses functional requirements of the study; it provides background, scenario and technical requirements for Teletype Network (TELNET), File Transfer Protocol (FTP), RJE and local terminals. These functions are discussed in a dual dimension (where applicable), i.e., in terms of the present ARPANET protocols and in terms of any H6000 peculiarities relative to the functions.

The Design Overview contained in Section 4, begins to isolate and assign the previously discussed requirements into the H6000/PDP-11 framework. It is here that the TELNET requirement is decomposed into USER and SERVER roles and

similar data are presented for the FTP and RJE requirements. The last part of the Design Overview presents GCOS interface requirements. The basic building block structures of the existing GCOS and the modified GCOS are presented. This point leads to a precise technical discussion of software modules required to support the interface.

Sections 5 through 8 are the most definitive areas of the document. They define specific modification and development requirements. The existing Input/Output Supervisor program (IOS) will be minimally modified to gain entrance into new GCOS software. This is discussed in Section 5 and introduces the Pseudo-soft DN355 (PS355), a DN355 simulator program. The specification then defines a new protocol handler which is assigned responsibility for H6000 to PDP-11 data transfers. The final H6000 consideration is the Asynchronous Bit-Serial Interface (ABSI) channel module. Here the lowest level of the interface (e.g., physical data exchange) is specified.

Front-end (FE) processor functional requirements are defined in Section 9. This part of the paper analyzes and provides functional definition for the NFE executive, standard ARPANET software, and the GCOS interface software.

Section 10 reviews the operational considerations of the GCOS interface to the ARPANET. The analysis discusses a number of issues such as operator interface and configuration factors.

Finally, Section 11 summarizes the findings contained in this document. A management level implications section narrates the advantages and, where appropriate, any disadvantages surrounding the H6000 and PDP-11 software interfaces.

There are five appendices to this report. These include a glossary, a bibliography, and three technical appendices. The first technical appendix documents key assumptions of the design approach. The other technical appendices are those which analyze the PDP-11 operating systems and the H6000 as an NSW TBH.

SECTION 2 - PROJECT ACTIVITIES

This section broadly summarizes all study activities and analysis areas. Included is a review of the interim report, a tutorial on the functional areas supporting the study, a review of interface alternatives, a recommended interface approach within General Comprehensive Operating Supervisor (GCOS), and a discussion of activities and analysis surrounding the PDP-11 to be used as a Network Front-End (NFE).

2.1 REVIEW OF INTERIM REPORT (RADC-TR-76-228)

The interim report published in July 1976 addressed the problem of connecting an H6000, operating under the control of standard commercial release GCOS, to ARPANET for the purpose of supporting National Software Works (NSW) activities. Four hardware configurations, all capable of supporting this requirement, were evaluated using the following selection criteria.

1. Hardware availability
2. Functional applicability
3. Capability to support throughput and response
4. Expandability
5. Maintainability
6. Risk
7. Cost.

The results of this evaluation, outlined in Section 4 of RADC-TR-76-228, were to recommend the hardware configuration shown in Figure 2-1. The H6000 will logically interface the ARPANET through a PDP-11 which functions as a NFE for the H6000. The physical connection is accomplished using an Asynchronous Bit Serial Interface (ABSI) developed by Massachusetts Institute of Technology (MIT) to interface the MIT Multiplexed Information and Computing Service (MULTICS) system to the ARPANET. The ABSI is designed to connect two Input/Output Multiplexor

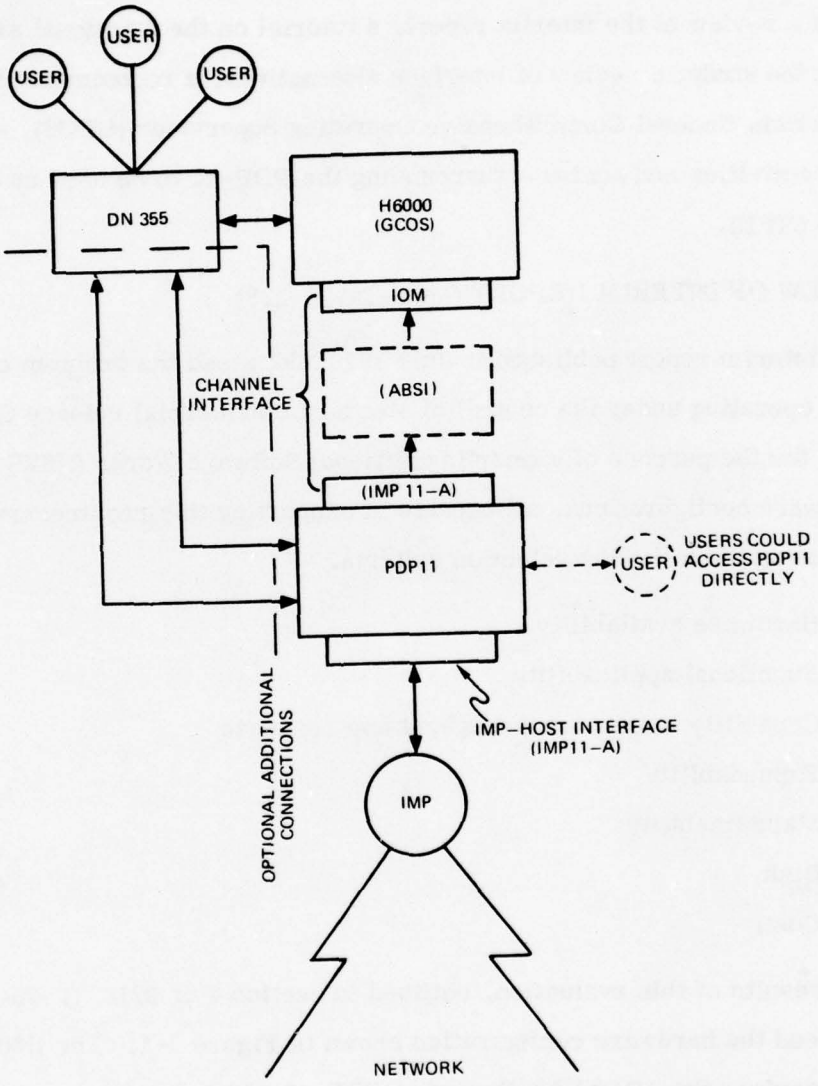


Figure 2-1. Selected Hardware Configuration

(IOM) common peripheral interface channels on the H6000 side to two IMP-11A lines on the PDP-11 side, thus providing a full duplex data path from the PDP-11 to the H6000.

This hardware configuration was subsequently approved and endorsed by Rome Air Development Command (RADC) and additional investigation and study has served to confirm this approach.

In addition to evaluating the various hardware configurations, the interim report addressed the various software approaches available for achieving the desired connection and functionality for a GCOS connection to ARPANET. Certain of these approaches were obviated by the selection of the hardware configuration.

In the Interim Report, the option of making the PDP-11 look like a known device (the Datanet 355 (DN355)) to GCOS, was deemed to be an unsatisfactory approach due to the inability of the H6000 to be a passive partner with the PDP-11 in a data transfer operation. There were two problems which were considered to be unsolvable in achieving the DN355 simulation. The first is the inability of the PDP-11 to directly manipulate the H6000 mailboxes. The second is the inefficiency and difficulty of duplicating the H6000/DN355 protocol over an ABSI controlled channel (multiple hardware interrupts in both the H6000 and DN355). Continued detailed investigation into solution of these problems has yielded a technique whereby DN355 capabilities are made available to network software. The solution is embodied in an H6000 resident privileged slave program which controls data flow to and from GCOS by manipulating data structures and generating interrupts in the same way as the DN355. In addition, flow to and from the network is controlled via normal H6000 Input/Output (I/O) procedures utilizing a new ABSI device handler. Since this approach represents a departure from the alternatives presented in the interim report, many of the conclusions regarding software considerations presented in that report are no longer valid. The advantages to this new approach will become apparent as succeeding, more detailed sections are presented.

2.2 SECOND PHASE ANALYSIS

The second phase of the effort began when the Government approved the interim report hardware connection scheme. This section discusses second phase H6000, GCOS, and PDP-11 analysis.

2.2.1 Second Phase H6000/GCOS Connection Analysis

The Statement of Work (SOW) identifies the host computer as a Honeywell 635 or 6180 running the GCOS operating system. Subsequent conversation with the Government indicates that compatibility with the H635 is not required. Instead, the generic name H6000 will be used to indicate the several compatible models which support GCOS.

CSC herein provides a design document for software interfaces to provide three basic services to remote ARPANET users. These services are referred to as Telecommunications Network (TELNET), File Transfer Protocol (FTP), and Remote Job Entry (RJE). They will allow ARPANET users to access interactive processes running on the H6000, to move files between the H6000 and a remote host, to enter a batch job to the H6000 from a remote host and, after execution, return the output it produces to an appropriate remote host.

The remainder of this section discusses GCOS interfaces required to support TELNET and RJE.

2.2.1.1 H6000/PDP-11 Interface

In order to provide any of the required capabilities, a basic known communications path must be provided between H6000 and the PDP-11. The existing structure of GCOS provides for isolation of device dependent I/O processing from global I/O processing. The software modules which contain device dependent processing are referred to as Channel Modules.

Examination of GCOS Input/Output Supervisor (IOS) software and available ABSI documentation has validated the assumption that GCOS could be modified to

perform I/O with a PDP-11 via the ABSI device. The basic structure of GCOS supports the interfacing of new devices. It is feasible to develop a Channel Module to handle the ABSI device such that normal GCOS I/O procedures can be used without major modification. Some non-standard techniques will be recommended in Section 8 for software configuration of the ABSI device and Channel Module but these will not require any significant modification to existing GCOS routines.

Given that physical connection of the computers is feasible, and that a non-disruptive GCOS/ABSI interface can be developed, it is appropriate to discuss the functional requirements for TELNET, FTP, and RJE interfaces within GCOS.

2.2.1.2 Server TELNET

A primary design requirement of a Server TELNET is to interface with interactive processes, so that they believe they are communicating with a local terminal. This requirement allows any existing interactive software to be accessed from the ARPANET without modification to that existing interactive software. It also ensures that new interactive software will be accessible to local users (as well as remote ARPANET users).

To make a remote ARPANET user appear to an interactive process as if he were using a local terminal, the interactive process terminal I/O requests must be "switched" somewhere within GCOS terminal handling to network software for dispersal over the ARPANET. Using this approach an interactive process issues its normal terminal I/O requests and gets back normal results which makes the process believe it is talking with a local terminal. Even so, several alternatives were considered as to where the "switch" should be installed. These alternatives can best be illustrated by explaining GCOS terminal handling in conjunction with the following Figure 2-2.

Figure 2-2 shows the major functional components involved with interactive terminal communications on the H6000 computer with the GCOS. Before proceeding to network implications, a definition of the components will be given. (The circles A, B, and C are discussed later).

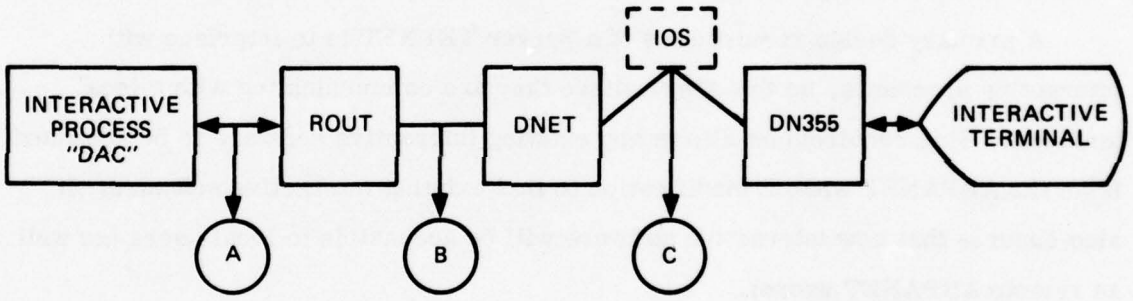


Figure 2-2. GCOS Input/Output Flow

1. Interactive process - GCOS terminology often refers to this as a Direct Access (DAC) program. A DAC program is any H6000 program which performs I/O to terminals. This definition includes portions of GCOS such as Time Sharing System (TSS), and Transaction Processor (TRAX), and VIDEO as well as user written programs. Since a different system call is required for terminal I/O (MME GEROUT) than is used for other I/O (MME GEINOS), DAC programs are easily identified.

2. ROUT - This is a core resident portion of GCOS. It is responsible for the initial processing of a terminal I/O request (MME GEROUT). It validates all terminal I/O requests, manipulates appropriate control structures, and initiates (via a subroutine in DNET) the initial interaction between the H6000 and the DN355.

3. DNET - This is a core resident portion of GCOS. It is responsible for all communication with the DN355. It also is invoked after a communication with the DN355 has completed. Based on the state of H6000 control tables and the type of communication which has terminated, it completes the processing of that portion of a terminal I/O request by updating GCOS control tables and initiating the next communication until the terminal I/O request has finished. Since four to six steps are involved in each terminal I/O request, DNET is slightly larger than ROUT and performs most of the H6000 to DN355 protocol.

4. IOS - This is a core resident portion of GCOS which handles all I/O operations. The DN355 is a separate piece of hardware while the other boxes in the above diagram are software modules within the H6000. Since the DN355 is a peripheral device, communication with it is performed through the IOS. IOS actually has a very small role in H6000 terminal handling and is shown primarily for completeness.

5. DN355 - A Datanet model 355 is a stand-alone small computer which acts as a front-end (FE) processor for interactive terminals connected

locally. All terminals are connected to the DN355 which supports teletype-like (half duplex, character-at-a-time) devices as well as RJE stations and page-at-a-time terminals. Data flow between the H6000 and the DN355 is buffered to at least one line but may include several lines. Data flow for all terminals is half duplex with the interactive process controlling line turn-around.

6. INTERACTIVE TERMINAL - Standard GCOS terminology refers to this as a remote terminal because it may be connected to the DN355 over telephone lines and therefore does not require physical proximity. Networking terminology uses the term remote to mean a terminal which is not directly connected to the host family computer (i.e., it is connected through the network).

One of the required functions for an ARPANET host is Server TELNET which allows remote terminals (connected to other hosts) access to local interactive processes running under GCOS. The goal is to make the remote terminal appear to the user as if it were locally connected to the H6000.

Somewhere in the existing communication path shown previously, a software "switch" must be inserted to sidetrack the normal data flow between an interactive program and local terminal and reroute data flow to the ARPANET. Several constraints exist which will be used to evaluate the location of the "switch" required for Server TELNET. These constraints include:

1. A user program must not require modification to be used through the network
2. The switch must be installed in the H6000 rather than the DN355
3. Installation of the switch should require a minimum modification to existing GCOS software
4. The switch should be installed so that GCOS maintenance and future releases will have a minimal impact

5. Insofar as the other constraints allow, the Server TELNET function should require the minimum amount of new software development, H6000 overhead, and it should be easily maintained.

Looking back at the GCOS I/O Flow, there are three reasonable places for installing the Server TELNET switch, labeled A, B, and C. Each of these alternatives are analyzed for impact using the constraints mentioned above.

A. This alternative, labeled A, involves "switching" terminal I/O requests (MME GEROUT) issued by an interactive process almost immediately after they are issued, and before any significant processing by ROUT. This approach will not require any modification to DAC processes and very little modification to existing GCOS software. This approach will require duplication of ROUT and DNET functions and will involve a large new software development effort. Because it will be involved in interpreting MME GEROUT requests and controlling the execution of interactive processes (as ROUT currently does), it will be sensitive to GCOS changes in those areas. While the long-range format and meaning of MME GEROUT requests should be fairly stable, the format and interpretation of data structures used to control a program's execution, as well as the techniques used to manipulate these data structures are subject to change with each new GCOS release.

B. The second approach, labeled B, requires "switching" to network software after initial processing by ROUT but before DNET becomes intimately involved. While this does not require any changes to interactive processes, it will probably involve inserting 20-40 separate switches in ROUT because the calls to DNET cannot be easily intercepted upon entry to DNET. This approach will involve simulating DNET functions and therefore require moderate new software development.

   Because this approach involves controlling the execution of interactive processes and data structures used for communication between ROUT

and DNET, it will be sensitive to changes in those areas. The internal operation of ROUT and DNET has been evolving, so the approach will probably be quite sensitive to new GCOS releases. Also, this area of GCOS is not well documented or well structured, so debugging and maintenance of software to duplicate DNET's operation will involve significant technical risk.

C. The third approach ("C" in the diagram) involves "switching" to network software when DNET is ready to communicate with the DN355. The "switch" will occur at a special entry point in IOS called only by DNET. Networking software will then simulate the actions normally taken by the DN355. Because the DN355 is a separate computer, it does not directly manipulate H6000 internal data structures. Although the protocol between the H6000 and the DN355 is not well documented, it has been relatively stable and is expected to have little change until the entire GCOS terminal handling software is rewritten. One drawback is that this approach would configure a separate DN355 and IOM channel number for use by networking software. While no new hardware is required to do this, it reduces the number of real DN355s which can be attached to the H6000 from three to two.

Option C is the recommended approach. Option C will require no greater software development than the other approaches. It requires a minimum of GCOS modifications and will provide the easiest maintenance and compatibility with new GCOS releases. If Honeywell continues its current plans to completely redesign GCOS terminal handling software, Option C will be the easiest option to reimplement. Option C will also provide an automatic GCOS RJE capability. The functional aspects of GCOS RJE and file transfers are discussed in the following paragraph.

2.2.1.3 Server RJE and File Transfers

A GCOS interface is capable of providing an RJE capability to remote ARPANET users. The RJE capability is described in the ARPANET protocol specification and is shown in Figure 2-3.

Unlike most ARPANET standard software, the RJE protocol is nonsymmetrical. Normally, the responsibility to provide a capability is evenly split between a User side process and a Server side process. RJE design does not require a User RJE; instead all RJE operation is centralized at each Server host with RJE interfaces to remote hosts using components of other ARPANET functions namely, User TELNET and Server FTP.

The software structure shown above is recommended for ARPA but not required for compatibility. It does show the three basic capabilities required of an RJE implementation; user interface, FTP interface, and local system interface.

User interface is provided through RJE Server TELNET and Server RJE. The former is a TELNET like communication mechanism which performs network I/O to pass lines of text between the local Server RJE and a remote user or user process. The Server RJE interprets the text as RJE commands, takes appropriate action, and reports the results of that action as text to be sent to the user.

An FTP interface is used to retrieve a remote job file for local execution and to return output files produced by that execution to some remote host. Server RJE has three choices in the FTP interface. Either it could not use any local FTP modules and handle both control and data connections to remote Server FTP modules. Or, Server RJE could interface with the local User FTP to request a file transfer. Server RJE would not be involved in the file transfer but would be informed when the file transfer had completed. The third choice is that Server RJE could interface with User FTP in some special manner such that User FTP would handle the control connection to a remote Server FTP, but the data connection would be handled by Server RJE.

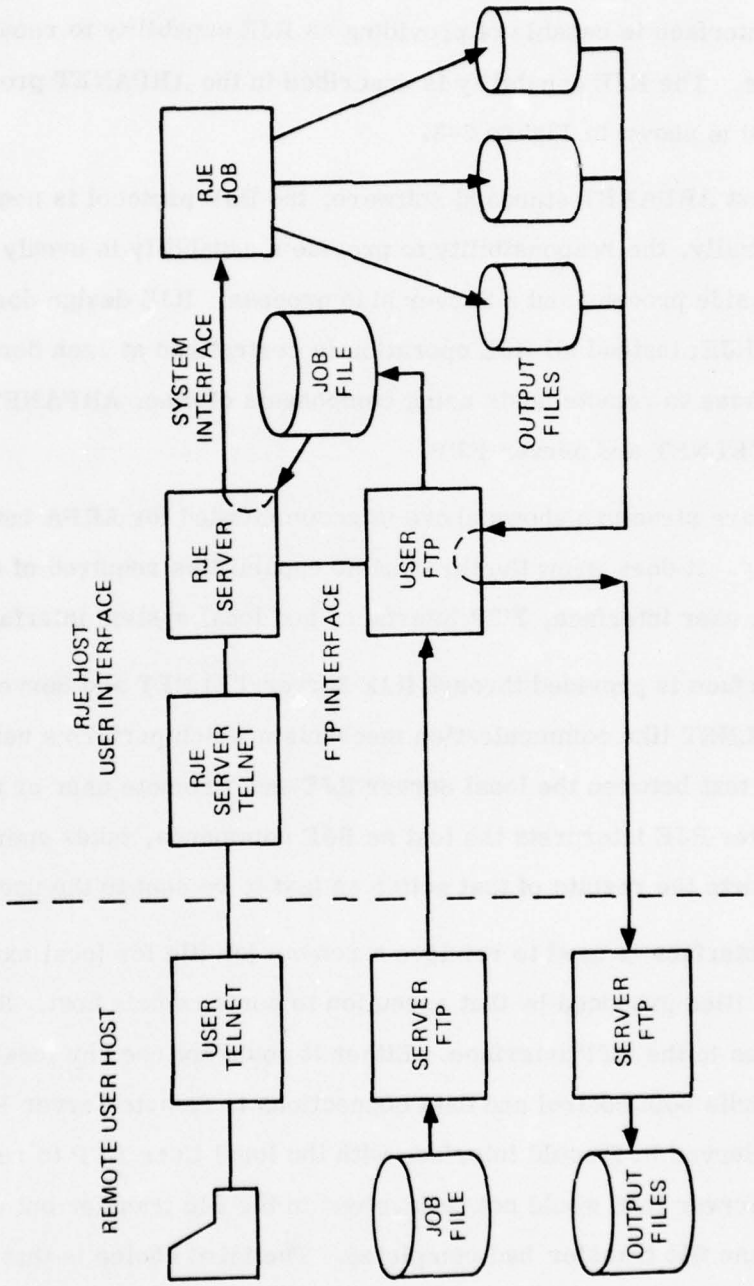


Figure 2-3. RJE Flow Diagram

With this definition of RJE capabilities in mind, one may examine GCOS interfaces which might be used to support ARPANET RJE. GCOS provides a mechanism which any job may use to initiate another job (this mechanism is referred to as MME GENEWS or spawning). Unfortunately, this mechanism is not fully suitable because it does not allow the initiating job to control the execution of the spawned job and it does not allow the initiating job to retrieve output produced by the spawned job.

GCOS has another mechanism which will provide the required functionality. This mechanism is used by two GCOS programs, TSS and RBATCH. These programs allow interactive terminals to submit batch jobs and retrieve printed output produced by the jobs. Remote network users can use these programs directly through TELNET, but while this provides the functionality of RJE, it is not an ARPANET standard RJE implementation. Unfortunately, the mechanism used by TSS and RBATCH is only available to those programs and cannot be used by Server RJE without significant GCOS modification.

In addition to the job entry mechanisms described above, GCOS supports Remote Computers (RMC) and Remote Network Processors (RNP). These are minicomputers with a card reader and printer which are connected to the H6000 through the DN355. Despite the use of the term remote, RMC and RNP are standard Honeywell products and have nothing to do with a computer network (such as ARPANET). Server RJE software will be able to use the capabilities for job entry provided within GCOS for RMC/RNP through a special interface which will be implemented to support Server TELNET (see Paragraph 2.2.1.2). This DN355 interface, PS355, consists of H6000 resident networking software performing the DN355 side of the H6000 to DN355 protocol, such that H6000 GCOS believes that it is talking to a real DN355. Different commands of H6000 to DN355 protocol are used for remote job entry than are used for regular terminals. As a result, the implementation of Server TELNET will not automatically provide Server RJE. The implementation of Server TELNET will provide GCOS modifications and interface points which will be a framework for Server RJE development but only a

small amount of software can be shared between Server TELNET and Server RJE.

Some difficulties exist in implementing the output dispersal functions of ARPANET standard RJE. GCOS makes a general distinction between output directed to files and print or punch output. Print or punch output is normally collected (sometimes referred to as output spooling) by a portion of GCOS named SYSOUT. After the job has finished execution, SYSOUT holds the output until it can be dispersed. Dispersal may be to either H6000 peripheral devices (line printers and card punches) or to RMC/RNP devices connected through the DN355. GCOS control cards define where portions of the job's output should be sent. The only output which will be available to Server RJE without GCOS modification is that output which is normally sent to the DN355. The OUT command of RJE would therefore, not apply to output which the remote ARPANET user has explicitly directed to local devices via GCOS control cards contained in the submitted job stream. Major GCOS modification will be required to have Server RJE commands override GCOS control cards; this is a reasonable limitation.

GCOS makes a clear distinction between temporary and permanent files in dispersal of output directed to a file. Temporary files are created to support the execution of a job and are destroyed when they are deaccessed such as during end-of-job processing. Without GCOS modification, output dispersal of temporary files cannot be accomplished. Implementation will probably involve developing a GCOS capability to pass files between jobs and modification of GCOS file deaccessing modules to recognize an ARPANET RJE job and pass temporary files to Server RJE instead of destroying them. Permanent files may be created and deleted by the execution of a batch job but are usually presumed to exist independent of any particular file. Permanent files may be accessed by several jobs either serially or concurrently. Dispersal of permanent files can be supported by an H6000 Server RJE design but is primarily a user convenience. Because permanent files remain in existence after the execution of a remote job, the user can manually disperse those files through FTP rather than have Server FTP perform the dispersal.

2.2.2 Second Phase PDP-11 Analysis

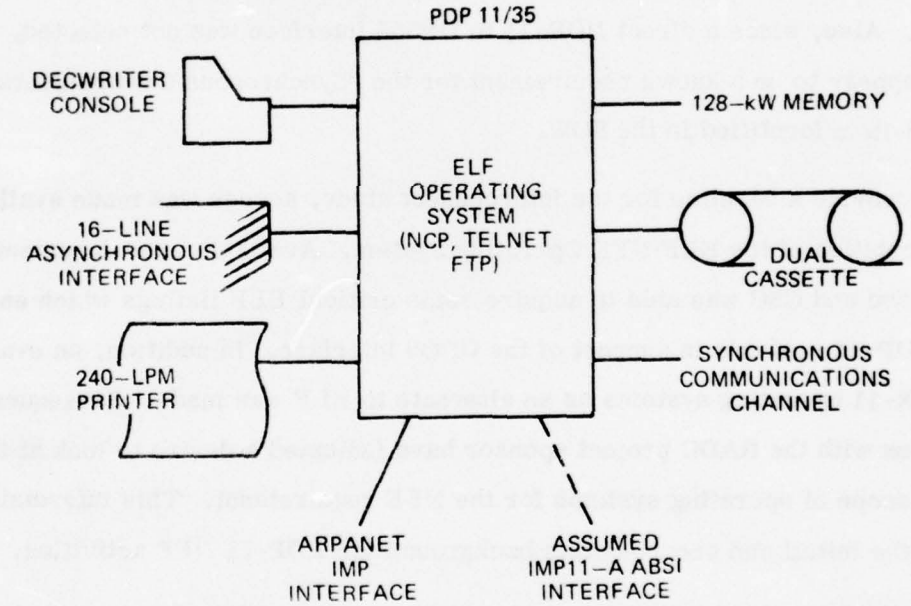
The SOW identified the NFE hardware and software environment. The Government provided configuration study is shown in Figure 2-4. The hardware items used for the study included the basic processor, console, printer, dual cassettes, and other interfaces. Annex 1 to the SOW identified an "ARPANET Interface Message Processor (IMP) Interface Unit". However, through the entire post interim report period, it has been mutually assumed that the intent of the SOW was to indicate two of these interface units (IMP11-A's). With respect to the Government selected interface configuration, one IMP11-A is required for the H6000/ABSI side of the PDP-11, and another is required for the ARPANET/IMP interface. Also, since a direct PDP-11 to DN355 interface was not selected, there does not appear to be a known requirement for the "Synchronous Communication Port" end-item identified in the SOW.

To provide a baseline for the interconnect study, access was made available to documentation of the ELF NFE Operating System. Available user documentation was included and CSC was able to acquire some critical ELF listings which enhanced PDP-11 analysis in support of the GCOS interface. In addition, an evaluation of the RSX-11 operating systems as an alternate to ELF was made. Subsequent discussions with the RADC project sponsor have indicated a desire to look at the broadest scope of operating systems for the NFE requirement. This information provides the initial and second-phase background for PDP-11 NFE activities.

The fact that the initially constructed PDP-11/35 architecture will be modified prior to any implementation of the GCOS interface led to a point where it was felt impractical to specify the exact software requirements for the NFE. Coupling that with the results of the NFE operating system survey only strengthened the previous conclusion.

During the period of the second phase PDP-11 activities, CSC:

1. Continued ELF analysis
2. Conducted ELF user site visits



NOTE:
PROVIDED BY GOVERNMENT TO SUPPORT GCOS/NFE/ARPANET INTERFACE STUDY

Figure 2-4. NFE Model Configuration

3. Commenced UNIX Operating System Analysis

4. Visited with UNIX networking site implementors

5. Formed an RSX-11 Operating System analysis sub-task

6. Discussed on-going RSX-11M and RSX-11D NFE efforts with Government and contractor personnel.

From all of these, CSC has developed an operating system composite profile which is presented in Section 9. We have included a first-level tradeoff discussion of the apparent NFE contenders as an appendix item.

There are a plethora of partially completed and on-going NFE projects. For purposes of this document, we address the NFE processor as a support function to the GCOS/ARPANET connection - one which is geared towards the general NFE goal of off-loading the maximum networking requirements from the host processor. We believe that the NFE is not to be constrained by the baseline architecture; however, we do envision that the NFE construct will include:

1. A mini-computer of the PDP-11 family

2. An ELF-like executive and NFE structure (subsequently addressed)

3. Developed and proven peripheral support capabilities. These include character and block I/O interfaces, medium and/or large disk storage, 9 track tape, and medium speed printer.

In addition, it is desirable to have on-line program development and higher language capabilities. While these latter issues are not mandatory, a reasonable case may be made for quicker implementation and more responsive maintenance posture if they are found in the selected NFE. For example, the ELF's MACRO file (e.g., MAC.UML), cannot be assembled under the DOS operating system because all user source expansion statements are physically brought into core memory during an attempted assembly. The combination of the MAC.UML and certain source module files causes an inability to assemble the ELF on a 28K PDP-11 which may be used as the NFE. The result is that a PDP-10 (TENEX-type

Host) must be used to cross-assemble and link (bind) the ELF object. This object is then bootstrapped across the network to the particular location which is to run the PDP-11 NFE system. Although there is a perfectly valid background to support the "why" of this situation, it is obvious that no long range justification exists. We noted that certain users have circumvented the described situation - others are satisfied to use the ARPANET as described.

The summation of our investigations into the ELF Operating System are: First, the ELF represents an investment; one, which unfortunately is mostly shelved. It is surprising that so few years have passed since the first ELF was implemented, and once implemented, technology and user needs caused the ELF to be almost outmoded. This is not to say there are not satisfied users and current development efforts based on ELF. Rather, the desires for on-line program development, higher level language, and a respectable file system caused numerous offshoots in NFE technology. *Most widely recognized* is the NFE development effort under UNIX. In addition, the Massachusetts Computer Associates, Inc. (MCA) is developing an RSX-11M based NFE Operating system. There are other examples.

Second, ELF is not formally supported. There is the apparent continuing attention of ELF's author (D. Retz) and a user's group, loosely defined as the contents of the [SRI-AI] file <PEOPLE> FOLK.ELF-GROUP. In a small community of users, support is not really of the magnitude that causes one to categorically reject ELF because there is no formal telephone number with 24-hour-a-day answering service. Yet, support is an issue which cannot be dismissed. Key people who regularly contribute to ELF's health and maintenance do travel, get other assignments, etc.

There is an effort within the Government Naval Research Laboratory (NRL) to pick up, enhance and maintain the ELF Operating System as NRL is interconnected via the ELF/ARPANET technology. While we have discussed some of the ELF's benefits and shortcomings with personnel from this area of activity, we are cautious in suggesting this as an alternative to the support issue. First, there is the argument that this is still in a development and analysis arena. Second,

there is the possibility that the NRL-moulded ELF may in fact become that - slightly NRL unique. It is premature to speculate (and beyond the effort of this paper).

On balance, the lack of support postulate may be viewed that whether the selected Operating System is supported by a large well known organization, or a fraternity of friendly sympathizers is not the strongest factor. Any executive structure will have very sensitive areas such as device handlers, kernel-to-user interfaces, and virtual memory considerations. If local users are given the capability to maintain or develop new code in these sensitive areas, the possibility exists for the "whose problem is it?" syndrome. It is safe to state that resources will be expended trying to answer that question, should a problem arise. This is a well known arid point in any development activity.

Third, while we have been impressed with some of the coding technology and documentation surrounding ELF, it is not well documented for use as a development base for a NFE. Much of the information gained about ELF comes from visiting other users who have already experienced the problems and sorted the various, sometimes conflicting statements, versions of files, and similar problems surrounding trying to become an ELF'er. We understand that D. Retz has been most generous with users who have visited or called him with questions or problems. However, there apparently are no flow charts, test outlines, and similar expectations that we were able to find during the period of this study.

Finally, ELF is written entirely in Assembly-level language. It is written to be edited, assembled and linked in a Digital Equipment Corporation's (DEC) Disk Operating System (DOS) type environment. DOS supports only BASIC and FORTRAN. DOS supports only a single user at a time (doing program development). As suggested above, DOS, like ELF, represents a technology which has been surpassed by the last few years of commercial, scientific, educational, and governmental advances. There is also some question of the reliability of some of the networking software contained in ELF.

While the ELF may have utility for a very interim solution, at best it can be thought of as probably a non-recoverable development. It is therefore not recommended as the prima facie selection for the GCOS interconnection to the Network. This is in contrast to our interim report findings in which we indicated ELF might be suitable for the NFE.

SECTION 3 - FUNCTIONAL REQUIREMENTS

This section establishes the identification and definition for the functions required of the GCOS-ARPANET interface. Included are discussions of:

1. TELNET
2. File Transfer
3. Remote Job
4. PDP-11 Local Terminals
5. NSW Compatibility.

With regard to the TELNET description, it is noted that while the Statement of Work (SOW) identified only a server TELNET capability, subsequent Government/CSC discussions indicated the desirability of a User TELNET for the H6000. The discussion that follows presents functional requirements supporting a User and Server TELNET design.

3.1 TELNET

The ARPANET TELNET capability allows terminals to be used as if they were locally connected to any ARPANET host. Figure 3-1 shows TELNET is divided into a User TELNET which translates user terminal interactions into the TELNET protocol and Server TELNET which translates process interactions into TELNET protocol. TELNET is based on the communication facility provided by Network Control Program (NCP) and the IMP subnetwork.

The TELNET protocol is based on the concept of a Network Virtual Terminal (NVT). Both ends of a TELNET connection appear to the other end to be a NVT. In other words, the function of User TELNET is to make his local terminals act like NVTs, regardless of how they function or how they are interfaced to that host. The function of Server TELNET is to make the NVT appear to his local interactive processes to be a local terminal. This releases the various hosts in ARPANET from a requirement to handle large numbers of different types of terminals since all remote terminals and interactive processes appear to be the same.

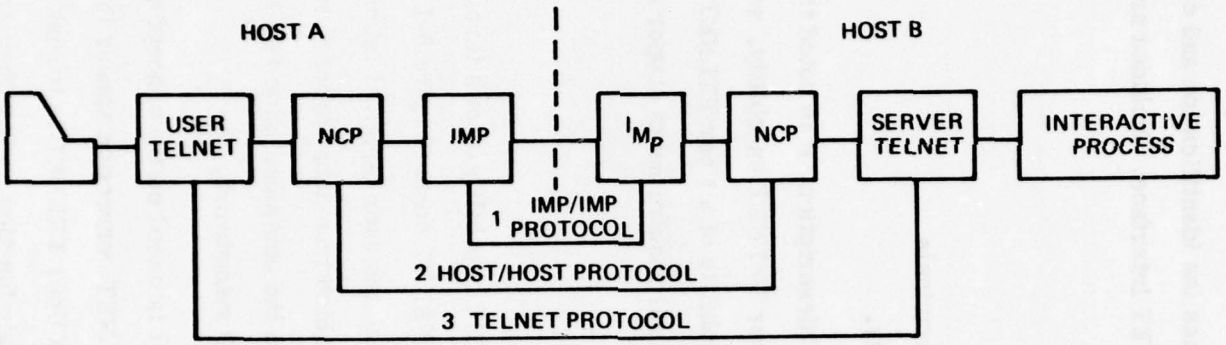


Figure 3-1. ARPANET TELNET

The basic NVT is a half-duplex, line at a time, American Standard Code for Information Interchange (ASCII) device similar to a teletype. All TELNET implementations must support the basic NVT. However, procedures exist in the TELNET protocol to change portions of the characteristics of the NVT providing both User and Server TELNET are willing to accept the change. This procedure is referred to as option negotiation. The options which a particular host may wish to accept are dependent on the characteristics of the terminal interface at the host.

To appreciate the significance of a GCOS implementation of TELNET, it is first necessary to discuss normal H6000 terminal support.

3.1.1 H6000 Terminal Handling

All interactive terminals on an H6000 are connected through a stand-alone front-end (FE) computer known as a Datanet 355 (DN355), as shown in Figure 3-2. The DN355 performs line buffering for the H6000 mainframe. It and the GCOS Operating System define the nature of terminal interfaces on an H6000. Since a User and Server TELNET implemented on an H6000 will be mostly limited to normal terminal operation, an examination of these characteristics is relevant. NOTE: H6000 terminology refers to interactive terminals as remote terminals. The use of the term "remote" in this discussion indicates a device attached through a DN355 and has no relation to ARPANET's use of the term "remote."

All terminal input to an H6000 is line buffered in the DN355 and is not sent to the interactive process until a carriage return character is struck. In addition, input is only buffered for delivery when the interactive process has requested input and once a carriage return is depressed, all subsequent input is discarded until the interactive process requests another line of input. The interactive process can issue any number of multi-line output requests without requesting input. All input requests, however, must include output (i.e., two GCOS functions are available for remote terminal INPUT/OUTPUT (I/O): OUT and OUT/IN). The output portion of an OUT/IN request is usually used to position the terminal's carriage since the terminal is assumed to not have "new line" capability (combined carriage return and line feed).

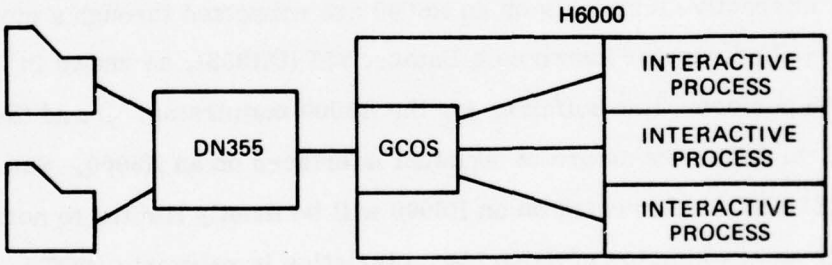


Figure 3-2. H6000 Terminal Handling

Two keys on the teletype keyboard are active at all times, Break and Control-C. The Control-C key causes the terminal to be logically disconnected from the interactive process it is "linked" to. The Break key causes any I/O operation in process to be aborted and returns a status to the interactive process which is sometimes used to cancel the current activity, it may be interpreted in any manner or ignored. The terminal is a teletype-like device using the ASCII character set and is assumed to be half-duplex in that no echoing is performed and no input is collected unless the keyboard has been "unlocked" by an input request.

3.1.2 NVT on an H6000

The NVT is defined as having six control functions: Go Ahead (GA), Interrupt Process (IP), Abort Output (AO), Are You There (AYT), Erase Character (EC), and Erase Line (EL).

GA is used to signal User TELNET to "unlock" the user's keyboard and accept input. This command should be generated by Server TELNET immediately following the output text of an OUT/IN request. This command will be required by User TELNET to issue an OUT/IN request to receive terminal input.

IP is not normally available on an H6000. The Break signal is sometimes interpreted as an interrupt by H6000 software because it is the only signal which the DN355 will give to the H6000 without a request for input. The TELNET protocol specification states that Break and IP are not synonymous. However, it also notes that IP may be required by processes which use the TELNET protocol as a lower level communications mechanism. At this time only TELNET has access to the TELNET protocol and new processes presumably use their own protocol. Server TELNET therefore, will ignore the IP command and expect the user to send the Break command as he would at a local terminal. If at some future time a requirement to support IP is identified, IP will be treated by Server TELNET as a Break command.

Although a mechanism will be provided in User TELNET to generate the remaining commands of NVT, they cause no action to be taken at the user host. Therefore, only Server TELNET's treatment of these commands will be discussed in this section.

AO is not normally available on H6000. Server TELNET may simulate this function by returning successful statuses to OUT requests of an interactive process without sending the output text to User TELNET. An OUT/IN request can reset the AO, but only the last line of the output text will be sent to User TELNET. Subsequent OUT and OUT/IN requests are handled normally until another AO command is received. The interactive process will be totally unaware of AO. Currently, no requirement exists to support this command.

AYT is not normally available to H6000 users. Rather than modify GCOS to provide this function, Server TELNET could provide an appropriate response. Although AYT is not a required capability, it seems useful and should be provided.

EC function on an H6000 is provided by the DN355 while a line of input is being collected. On a line-at-a-time system (such as the H6000), this function will be performed at the User host and therefore will not be seen by Server TELNET. To support remote User TELNET implementations which do not provide this capability, it should be performed solely by Server TELNET. Some implementation restrictions will be necessary. EC commands will apply only to the current line of input since the previous line has already been sent to the interactive process. This means that any EC commands received after an end-of-line and before any printable characters are received would be ignored. In addition, the ARPA definition of a "print position" which the EC command is supposed to delete may be very difficult to implement. If an EC command were to follow a simple character string such as "A ⟨BS⟩ X", both the A and the X should be deleted since they occupy the same "print position." If the preceding character string should include multiple BS characters or other format effectors such as Vertical Tab (VT) or Horizontal Tab (HT), the correct interpretation of EC is difficult. For example, the character strings "AAA ⟨BS⟩ ⟨BS⟩⟨BS⟩ XXX", "A ⟨CR⟩ ⟨NULL⟩ X", or "A ⟨HT⟩ ⟨BS⟩ X" are confusing since interpretation of print position is dependent on the terminal hardware involved.

EL function on an H6000 is provided by the DN355 while a line of input is being collected. This function would be more efficiently provided by User TELNET so that Server TELNET is never given an EL command. Even so, this command should

be implemented by Server TELNET because of the unknown limitations of the remote User TELNET. EL commands received after an end-of-line and before any characters of a new line are ignored. Also, multiple occurrences of the EL command are treated as a single occurrence. As with the EC command, these restrictions are based on the assumption that an end-of-line will be used to initiate delivery of the line to an interactive process and once delivery has been initiated it is too late to modify or delete the line.

In addition to the six commands discussed above, the TELNET protocol supports the Break key as a 129th character in the ASCII character set and a SYNCH procedure to clear the data path between User and Server TELNET.

Both User and Server TELNET are required to perform the SYNCH procedure although the interactive process or the terminal user will be unaware that a SYNCH has occurred when the other side initiates it. Since Server TELNET may not implement the AO command, it should not need to generate a SYNCH. User TELNET will allow the user to send a Break character and Server TELNET will treat Break just as if it had been generated at a local terminal.

Although Carriage Return (CR) and Line Feed (LF) are standard ASCII characters, some terminals treat an LF as a new line indicator and also perform a CR. As a result, the TELNET standard representation of a new line is "CR-LF." Server TELNET must scan output text produced by interactive processes and convert contiguous sequences of CR, LF, DELETE (DEL), and NULL into "CR-LF" sequences (deleting DEL and NULL characters).

Any single occurrence of CR must be converted to "CR NULL" to avoid ambiguity. User TELNET must convert "CR-LF" and "CR NULL" sequences to whatever is required by the users terminal to perform new line and carriage return functions, including insertion of NULL fill characters to control timing. User TELNET must scan user input and convert CR and/or LF to "CR-LF" or "CR NULL" as appropriate. Server TELNET will buffer user input received from the network until either "CR-LF" or "CR NULL" is received. Server TELNET will then give the line of input accumulated to the interactive process (to satisfy an OUT/IN request). Since on an H6000

all lines of input are assumed to end with the first CR character, Server TELNET will delete the LF or NULL following a CR.

The NVT is assumed to be responsive to the following carriage control characters: NULL, LF, CR, BELL, Back Space (BS), HT, VT, and Form Feed (FF). Normally these characters are sent directly to the user's terminal. When program specifications for User TELNET are being formed, the capability should be considered to allow the user to inform TELNET that the terminal cannot handle one or more of these control characters. User TELNET can then simulate the effects of these characters by replacing them with either printable graphics (such as ↑L for FF) or a combination of other control characters which the terminal can handle (such as several LF characters for FF).

3.1.3 Negotiated Options

The ARPA TELNET protocol supports the concept of negotiated options. This allows the characteristics of NVT to be modified provided both User and Server TELNETs agree to the change. If either TELNET rejects an option, the characteristic of the NVT remains unchanged. The negotiation mechanism facilitates the implementation of new options because a TELNET is not required to know about an option to reject it. The following is an analysis of the existing TELNET options and their applicability to an H6000 Server TELNET and User TELNET supporting H6000 terminals. User TELNET support of terminals connected directly to the NFE will not be addressed because the terminal processing environment is presumably less restrictive in the NFE and because the NFE software has not precisely been identified. If the NFE software selected provides a User TELNET capability, the TELNET options it allows will continue to be supported.

3.1.3.1 TELNET Binary Transmission Option

This option allows transmission of 8-bit binary characters over the TELNET connection. The H6000 TELNET implementation does not include explicit program to program communication using TELNET as a lower level protocol. In addition, since NVT does not have explicit paper tape capabilities (especially where the parity

channel is used as a data channel to record 8-bit bytes), this option is not to be supported by H6000 TELNET. Because the H6000 terminal handling allows handling of 8-bit bytes of data in conjunction with paper tape, this option can be implemented if a requirement is identified.

3.1.3.2 TELNET Echo Option

This option requests the other TELNET (normally a Server TELNET) to transmit back (echo) data characters it receives as input. This option is useful on character-at-a-time computers where interactive processes normally echo input. This option is not practical on line-at-a-time computers such as the H6000 and should not be implemented.

3.1.3.3 TELNET Reconnection Option

Since interactive processes are assumed to be unaware of the network, it is unlikely that they would initiate a reconnection. Initially, this does not appear to be a required option. If a need to support this option where identified, Server TELNET could accept a reconnection from a remote process and User TELNET could allow a reconnection to take place. So that the user does not become confused by switches between server processes, he should be informed of a reconnection.

3.1.3.4 TELNET Suppress Go Ahead Option

Both User and Server TELNET are required by H6000 terminal handling software to operate in a half-duplex mode. The GA signal is required for half-duplex operation. Therefore this option should always be rejected.

3.1.3.5 TELNET Approximate Message Size Negotiation Option

Even after negotiation, messages of any size may be sent by either TELNET. The goal of this option appears to be to reduce network overhead by reducing the number of single character transmissions over a TELNET connection. In a line-at-a-time, half-duplex mode of operation, presumably each transmission would involve a line of text. This option is therefore unnecessary. It if is determined that remote

sites are sending single character transmissions, then this option will become highly desirable because a high network overhead is incurred to transfer a single character.

If this option were implemented, then the maximum input line length specified by the interactive process should be used. Normally this is 120 ASCII characters. In practice, users seldom type 120 characters of input per line, but using the maximum should insure a line-at-a-time transmission of text over the network.

3.1.3.6 Revised TELNET Status Option

Presumably, both sides of a TELNET connection know what options are in effect at any point in time. This option should be fairly easy to implement and may be useful for debugging. The decision for implementation should be based on the amount of resources available in the NFE after required functions are provided.

3.1.3.7 TELNET Timing Mark Option

This option appears to be primarily useful to full-duplex, character-at-a-time computers. It is not required for H6000 TELNET.

3.1.3.8 Remote Controlled Transmission and Echoing TELNET Option

Since this option is only useful to full-duplex hosts, it is not required for H6000 TELNET.

3.1.3.9 TELNET Output Line Width Option

Since the H6000 does not provide a low-level mechanism for an interactive process to determine the line width of a terminal, there is no reason for Server TELNET to accept this option. If a TELNET user specifies his line width, User TELNET should attempt to negotiate this option since the remote host may support it.

3.1.3.10 TELNET Output Page Size Option

Since the H6000 does not provide a low-level mechanism for an interactive process to determine the page size of a terminal, there is no reason for Server

TELNET to accept this option. If a TELNET user specifies a page size to User TELNET, it should attempt to negotiate this option since the remote host may support it.

3.1.3.11 TELNET Options for Terminal Characteristics

The TELNET protocol provides a number of options for disposition of carriage control characters including: CR, HT, FF, VT, and LF. Different terminal models react differently to these control characters. Some terminals will perform the specified action, others will ignore the control character, and some will indicate an error condition.

It is User TELNET's responsibility to correctly handle local terminals. Although a simple implementation would pass the control characters directly to the terminal and hope for correct action, a more elegant approach would allow the terminal user to specify whether each control character should be ignored, converted to a printable graphic, sent to the terminal, or simulated by replacement with other control character strings. This interaction between the user and his local TELNET does not require any negotiation of TELNET protocol options.

Some hosts provide a general purpose mechanism for specifying terminal characteristics, i.e., interactive processes are provided a query terminal characteristic function. The information thus obtained may be used by the interactive process to produce "good looking" output. For example, a process which produces summary tables may use line length and page size information to avoid line "folding" and insure a symmetrical, well-grouped report. If a host normally provides this information to interactive processes, then some provision is required to pass these terminal characteristics within the TELNET protocol. That is exactly the type of function performed by these negotiated options. In addition, the server host may normally provide the kind of control character processing mentioned as a User TELNET responsibility. When this is the case, these options allow User TELNET to pass some of the local terminal control responsibilities to the server host.

H6000 terminal handling does not provide a low-level mechanism for specifying terminal control characteristics. Server TELNET should therefore reject these options. Because a remote server host may support these options, User TELNET should allow the user to request negotiation of these options.

3.1.3.12 TELNET Extended ASCII Option

This option is used to support special terminal hardware capable of generating and printing a 9-bit character set. This option can be handled very effectively on an H6000 because of its 9-bit byte size. No special software or hardware exists on an H6000 to make use of extended ASCII, so there is no requirement at this time to implement this option.

3.1.3.13 TELNET Extended Options - List Option

This option provides a mechanism for supporting more than 256 options. When an extended option negotiation becomes desirable, this procedure will be followed. For now, this option is to be rejected.

3.2 FILE TRANSFER PROTOCOL (FTP)

File Transfer Protocol (FTP) is an ARPANET function provided to enable the sending and receiving of data files between two host computers at the initiation of either, or at the request of a third host. There is a functional dichotomy characterized by the terms "user" and "server." These indicate the location of the initiating or destination terminal or process. This gender is imposed on functional elements of FTP and is represented by the relationship shown in Figure 3-3. For the purposes of this document, most user FTP protocol and negotiation functions are presumed to exist in the chosen FE system. The emphasis here is placed on H6000 and GCOS characteristics available to support the server FTP implementation in the H6000.

There are two factors which influence the design of the FTP process. The first is the location of the user with respect to the H6000, i.e., local to the H6000 or local to a remote network host.

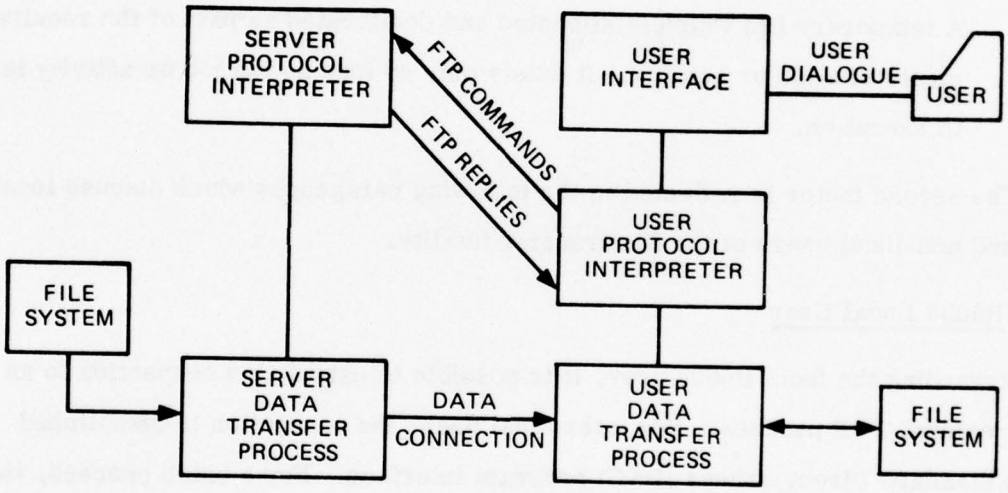


Figure 3-3. File Transfer Protocol

The second factor is the category of file in the H6000 which is being affected by the file transfer. Within GCOS there are several categories of files which FTP could deal with:

1. A cataloged or permanent file which is always on-line and no device need be allocated to allow access
2. A cataloged disc file which resides on a removable disk pack and must be mounted on an allocated device before access
3. A tape file, either cataloged or not
4. A temporary file which is allocated and deallocated as part of the requirements of a job or activity. It exists only so long as the job or activity is in execution.

The second factor is reflected in the following paragraphs which discuss local users and non-local users of the file transfer facility.

3.2.1 H6000 Local User

Regarding the local H6000 user, it is possible to establish a connection to an H6000 resident FTP process. For a terminal user, the connection is established using a standard Direct Access (DAC) program interface. For a batch process, the connection is established through use of GCOS Intercom I/O interface.

ARPANET standard FTP commands will be presented, via the established connection, to define the specific file transfer requirement. H6000 FTP will use the derived information to gain access to the required file. It will also establish the NFE connections and cause initiation of the file activity. All H6000 and NFE connection establishments and file activities occur subsequent to the FTP initiation by the user.

The significant consideration for the FTP design is found in the accessability of a given file to the FTP process. For a terminal user connected via the DAC interface, any on-line, catalogued file will be available for FTP access using standard GCOS file system access methods (MME GEFSYE). Due to timing considerations,

other types of files must be treated differently. These differences are reflected in subsequent paragraphs.

It should be noted that in the case of a terminal user connected directly to the H6000 FTP process, there are no implicit or explicit temporary files attached to the FTP process, so it is unnecessary to support transfer of a temporary file at the initiation of a terminal user.

3.2.2 File Transfer for Catalogued Files

Whenever the terminal user desires to transmit a file which is cataloged but off line, such as a cataloged removable disc file or a cataloged tape, the problem becomes more complex. The fact that access to these files also requires allocation of a device means that a considerable delay may take place for the program requesting the allocation. As shown in Figure 3-4, a different approach is required because the FTP process cannot afford to be disabled or delayed for an extended period of time. In this case, the FTP process will spawn a "dummy" job which in turn gains allocation and access to the desired file, be it tape or removable disc. Once in execution, the dummy job notifies the FTP process through an Intercom I/O. The FTP process picks up the file information necessary to perform the file transfer from the slave, and puts the dummy job to sleep (set urgency to zero). When the file transfer has completed, the FTP will enable the dummy program which then terminates releasing the files and the allocated devices.

Once the FTP to "dummy" job interface is established, the concept may be extended for any user slave program executing in the H6000. By providing a general purpose subroutine which can be called by any slave program, and which provides the same functionality as the dummy job mentioned above, a user process may invoke a network file transfer. Any of the four categories of files may be transferred via this facility. Even temporary files attached to a slave program can be transferred because access and allocation can be guaranteed for the duration of the file transfer. It is probable that the only restriction placed on a slave program using this interface will be that this subroutine call is the last thing performed, since FTP will suspend the dummy's execution until the file transfer is completed. This implementation technique

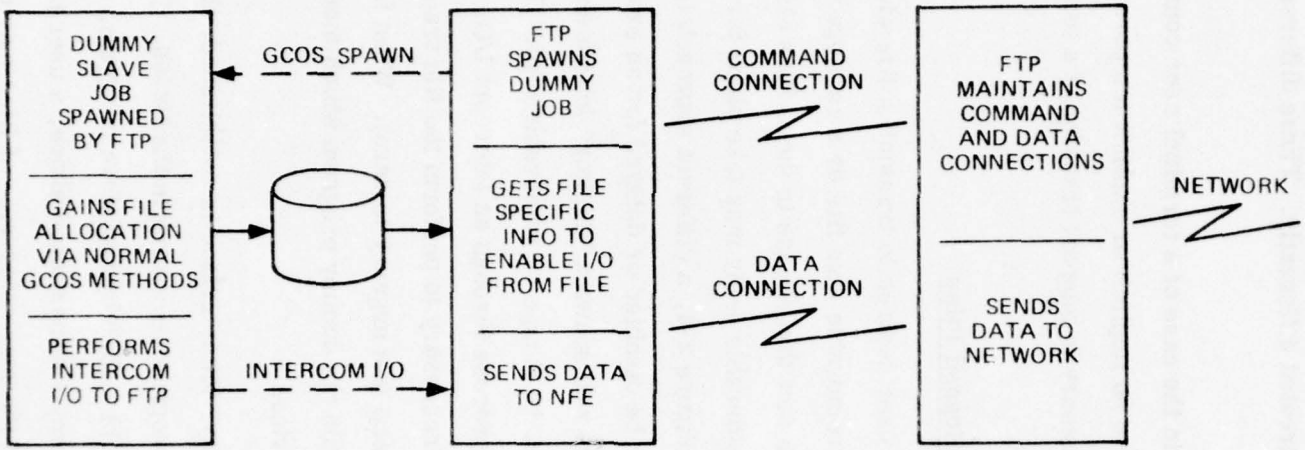


Figure 3-4. The H6000 File Transfer

does not require any GCOS modifications; only existing interfaces are used or required. There is however one performance characteristic which should be considered. Within the context of GCOS, the spawning of a job can be severely dependent upon several factors. The most significant factor is the load characteristic of GCOS at the time of the spawn. If the load is heavy and contention for resources is great, then allocation of the slave job can take some significant length of time.

For the local H6000 user this does not impact the network because no network connection is established until after the spawned job goes into execution. However, for a distant user, particularly an interactive user, where the connection is opened and exists before the spawn request, network time out characteristics may be an important consideration. There are several techniques within the GCOS environment which can aid in diminishing the delay in job spawning such as express scheduling, high priority, and limiting resource requirements. However, under extreme conditions minimal duration of time delays cannot be guaranteed. This situation can be handled by denying the FTP request and indicating that it should be attempted later.

3.2.3 Non-Local Users of File Transfers

Thus far the major emphasis of this presentation has been towards local H6000 users and files local to the H6000. When considering non-local users where the gender of the FTP process becomes "server," several new characteristics must be considered. In particular, the destination of the file being transferred to the H6000 becomes relevant to the H6000 resident FTP process. If the destination is to be an on-line cataloged file and that file already exists, then no additional functions are involved and the above described mechanisms prevail without notable difficulty. The same is true for well-known removable media destinations, i.e., (removable permanent or tape files). If however the desired destination is an as yet unknown cataloged file, then the additional logic to allow for on-the-fly cataloged file creation is needed. This doesn't cause any real functional or interface problems because GCOS has existing mechanisms for file creations; however additional information is required from the non-local user which is not part of standard ARPA FTP. File system specific information such as permissions, file type and protection options are

not general ARPA options although default values could be assumed. Because of network timing considerations, it may be desired to implement a separate FTP function to create new files before, and independent of, the actual file transfer. Such a function may be considered unnecessary when the TELNET connection is to GCOS time sharing (TSS), because the TSS access subsystem allows independent cataloged file creation and deletion. Removable media files as H6000 destinations are supportable under the above techniques and subject to the same timing considerations implied by job spawning, device allocation, and file access. Support of temporary files as the destination of a non-locally initiated file transfer appears to be unnecessary because, as noted before, this type of file exists only as long as an activity to which it is attached is in execution. It seems unlikely that a non-local user will have enough intimate knowledge of such job's status within the H6000 to be able to initiate and control such a transfer.

3.2.4 FTP Data Storage and Transmission

The data storage conventions of the H6000 are dissimilar from most other ARPANET host computers data storage formats, so transmission modes present a problem. From the standpoint of usefulness within the H6000, only 9 bit byte ASCII type or binary image type data seem necessary to be supported. All other data formats could be stored in one of these two modes without transliteration. The storage convention for ASCII data will be H6000 standard system format. For binary data the convention will be a contiguous bit stream with zero filler to the next H6000 word boundary at the end of each byte. It should be noted that for the binary mode convention to be efficient or useful, transmission byte size should normally equal some multiple of 36 bits, or record length, where possible.

To support these data format limitations some transmission mode restrictions are implied. Because ASCII data will be stored as a structured, record oriented, file within the H6000, it appears reasonable to restrict the transmission mode for this type file to a stream of bytes with or without the two byte End of Record (EOR) control code at the end of each record. Except for the EOR indicator, no block or record information characteristics of a particular file structure

will be sent in this mode, because the H6000 FTP will produce the H6000 standard system format structure when storing the file. Initially, at least, it seems reasonable to limit transmission byte size to the ARPANET standard 8 bit byte for files of this type. This byte size will require a conversion to the 9 bit byte when passing the data from the front end to the H6000. For ASCII files being sent from the H6000, the front end will do the conversion from 9 bit to 8 bit bytes.

Binary files will also be transmitted in byte stream mode. No limitation on byte size shall be implied or imposed. However as previously indicated, zero padding will be done on byte boundaries and judicious choice of byte size is presumed when transmitting in this mode. No EOR or End of File (EOF) bytes are included in this transmission because the structural characteristics are presumed to be encoded in the data or a function of byte size. The FE FTP will not manipulate data.

3.3 REMOTE JOB ENTRY (RJE)

Remote Job Entry RJE is a standard GCOS capability which supports remote minicomputers, Honeywell 716s, attached to the H6000 via an intermediate DN355 processor. The H716 computers are referred to as Remote Network Processors (RNP), and allow the submission of batch jobs and retrieval of output produced by the jobs. Typically, card readers and line printers are attached to an RNP in support of job submission and output.

The GCOS use of RJE is not to be confused with ARPANET RJE. Although they both serve a common function, there are some subtle differences. Honeywell's use of the term 'remote' only refers to the fact that the connection of an RNP and H6000 is not limited to a precise foot-length cable. The H716, as implemented, is a stand-alone computer. Another difference is found in the area of the format of H6000-entered jobs. H6000 RJE demands very specific formats for command and job data elements. These are supported even by the DN355 FE to GCOS. In effect, GCOS RJE is an outward extension of the H6000. In the design to be presented in this document, the GCOS extension will be distributed between new software in the H6000 and in the NFE.

At a point in GCOS processing of RJE, an interface to network software is feasible, so that the job entry and output retrieval capabilities of GCOS can be used to implement ARPANET RJE. The specific design approach is discussed in Section 4. At this point, it is beneficial to discuss the functional aspects of H6000 RJE.

3.3.1 H6000 RJE

RNP is connected to the DN355 via a normal communications path such as a telephone line. There is a low-level protocol between the two computers which is implemented through the use of control cards read in at the remote card reader. These control cards are used to control the remote batch operations of the remote computer, switching from I/O (which never occurs simultaneously), requesting the status of jobs which have been entered at the remote station, and establishing and breaking connection between the remote computer and the DN355.

Within the DN355, data blocks received from RNP are separated into card images and passed to the H6000 employing the normal H6000 to DN355 protocol. Up to this point, remote computer input and terminal input are treated in the same manner. However, a departure occurs within the H6000 in processing direct access or terminal data and remote computer data. In DAC processing the H6000 modules DNET and ROUT act like a pipeline to pass the data directly to the DAC program to which it is destined. In remote batch processing, the idea is for the remote computer to be made to appear like a system card reader and on-line printer. There are two special slave programs which accomplish this function on the H6000 side.

Batch input is accomplished normally by the module GEIN. This is a slave program, many copies of which may be executed simultaneously. Each copy of GEIN consists of two parts: a main processing part, which interprets GCOS control cards (Note: These are not the remote batch control cards), builds special system temporary files, and enters the job for execution, and an I/O overlay which reads the input from a specific device. This input device may be a card reader, a disk, or a tape file. Because remote devices are much slower than other devices,

it is inconvenient to keep a copy of GEIN in core during the entry of a remote batch job. Therefore another module (RGIN) handles the reading of all remote jobs simultaneously, spooling them off to disks. When an entire job has been spooled to disk, a copy of GEIN is started to read the remote job from disk and begin its execution.

Dispersal of printed output to the system line printer is performed on the H6000 by a GCOS module called GEOT. It is a privileged slave program which is afforded special privileges by the rest of GCOS. It is, therefore, rather natural that it also performs the output dispersal function to RNP as well as to on-line printers. The print image files are exactly the same in the remote case as in the on-line case, and in fact, the output may be switched from one to the other. There is a great deal of special software in GEOT to perform the remote output function. It involes special interfaces with the ROUT/DNET modules and data structures which can be manipulated directly by all concerned.

The above is a brief narrative of the data flow and processing associated with RJE from H6000 RNPs. For ARPANET RJE, an analogous path and equivalent processing will be furnished to allow network-connected users to present RJE Job streams to the H6000 for processing and subsequent dispersal of printed output. The DN355 simulator technique, discussed in the Server TELNET section, will provide the GCOS interface for the ARPANET RJE. This interface will use standard "remote computer opcodes," which are part of the standard H6000 to DN355 protocol to enter jobs, retrieve output, and obtain job status. A residual RJE component will be required in the H6000.

3.4 LOCAL TERMINAL SUPPORT

Given a GCOS/NFE/ARPANET interface, it has been indicated by the project sponsor that a terminal support capability be provided in the requirements specification. This is for those terminal users directly connected to the PDP-11 NFE and is over and above any H6000 terminal capability provided to the DN355 users. This is demonstrated in Figure 3-5, Local Terminal Support.

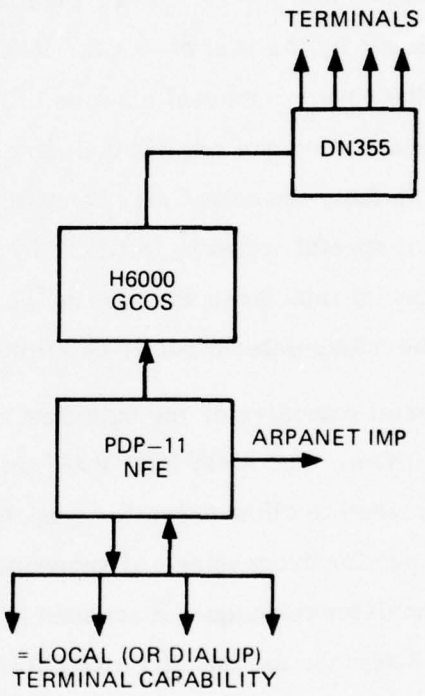


Figure 3-5. Local Terminal Support

The broad goals of connecting terminals to and supporting them in the NFE include:

1. Provide a Terminal Interface Processor (TIP) type of interface to the ARPANET
2. Provide an alternate terminal access to the ARPANET in the case of H6000 failure
3. Possible program development support during relatively idle networking periods.

There are several pertinent factors regarding specifying a terminal support facility such as indicated above. The standard asynchronous terminal functions, as detailed in Terminal Interface Message Processor -- User's Guide (BBN#2183) are considered as existent capabilities in an ELF-like NFE system. Therefore, the dialogue supporting User TELNET activities such as OPEN, CLOSE, ECHO, and RESET are present and require no special development activities or specification.

In fact, although it is not easy to use, the capability of a PDP-11 local keyboard-to-keyboard dialogue is possible. As noted in the TIP Users Guide, this is accomplished in the general format of:

HOST <Numeric>
SEND TO SOCKET <Numeric>
RECEIVE FROM SOCKET <Numeric>
PROTOCOL BOTH.

Since a main thrust of the GCOS interface is to provide remote user (e.g., users on the ARPANET) access to the facilities of GCOS, and appear to GCOS as if they are local terminals, there is little additional effort to provide this to local PDP-11 supported terminals. Given that there are going to be constraints imposed on any remote users (e.g., line-at-a-time dialogue, knowledge of H6000 DAC program names, etc.) these constraints can also be dictated to the NFE terminals. Alternatively, for a slight addition to the User TELNET and device handler for these terminals, it is believed reasonable to anticipate software support of character-at-a-time dialogue with the terminal users (e.g., the typical TIP-type mode of

operation) while passing data to the H6000 on a line-at-a-time basis (given that the user is willing to establish a special prompt to indicate a logical end-of-line sequence). The prompt transmitted to the H6000 must be a <CR>.

It is beyond the scope of this paper to provide an exhaustive survey of, and recommendations for, the types of local terminals that might be connected to the NFE processor. Certainly a main factor will reside in an analysis of the primary application(s) these terminals are used for. Presupposing standard TIP-like user functions such as editing, file manipulations and mail activities, it is believed that a mix of cathode ray devices (CRTs) and hard copy devices (teletypes) are appropriate. These asynchronous devices will minimally offer selectable baud rates (in the range of from 10-30 CPS through 1.2-2.4.Kbps), local/remote echo operation and standard ASCII character set.

These local terminals can be interfaced through a standard multi-line interface for the Digital Equipment Corporation (DEC) FE. Interface examples include the DJ11 and DH11 multiplexors, (DEC Peripherals Handbook, 1975). Auto-answer, dial-up interface may be indicated if users are to operate in a (semi) remote environment.

Depending on the finally configured NFE, the use of a standard operating system which supports the widest range of existing DEC peripheral interfaces is recommended. This will ensure existing software driver/handler support and probable upward compatibility as new devices and/or operating system versions are introduced.

The operating profile that is established tends to profile a TIP user environment much like that generated when an ELF configuration is established. The largest single difference with the GCOS NFE is the user capability to, in effect, have a message switching capability. That is, traffic may be vectored in either of three directions: local user to local user, local user to H6000, and local user to the ARPA Network.

SECTION 4 - DESIGN OVERVIEW

4.1 INTRODUCTION

A major goal of the General Comprehensive Operating Supervisor (GCOS) connection specification is to minimize the impact of the interface on the system and provide a networking environment which imposes no modifications or restrictions to existing user process interfaces. The existing GCOS remote Input/Output (I/O) functions (e.g., MME GEROUT) are used to support the Server aspect of network communication. Thus, existing processes such as Time Sharing (TSS) require no modifications.

A network connection may be envisioned as a connection between the Network Front-End (NFE) and a remote user (process). All communications are managed from the NFE through its Network Control Program (NCP). In this front-end (FE) environment, the H6000 software uses the NFE to perform connection and similar network activities. The capability to distinguish network from local I/O was initially discussed in Section 2--Second Phase Analysis and Design for GCOS. Basically, a network interface point in GCOS has been isolated. As network I/O is recognized (based on the Datanet (DN355) Number used by GCOS), an entry is made to appropriate network pre-process software resident in the H6000. This is shown in Figure 4-1.

When the I/O request (as shown) reaches the intercept point, all standard GCOS I/O validation, table and data structure manipulation, and similar processing has been accomplished. In fact, in the non-network (e.g., prior to the ARPANET interface) environment, the final I/O processing is basicaliy determination of the DN355 number and the initiation of the physical I/O.

In the case where the DN355 number indicates a network I/O activity, control is transferred to network preprocesses (through a DN355 simulator) such as TELNET File Transfer or Remote Job. These preprocesses are minimum residual requirements supporting the formal ARPANET Protocols. The actual protocols are

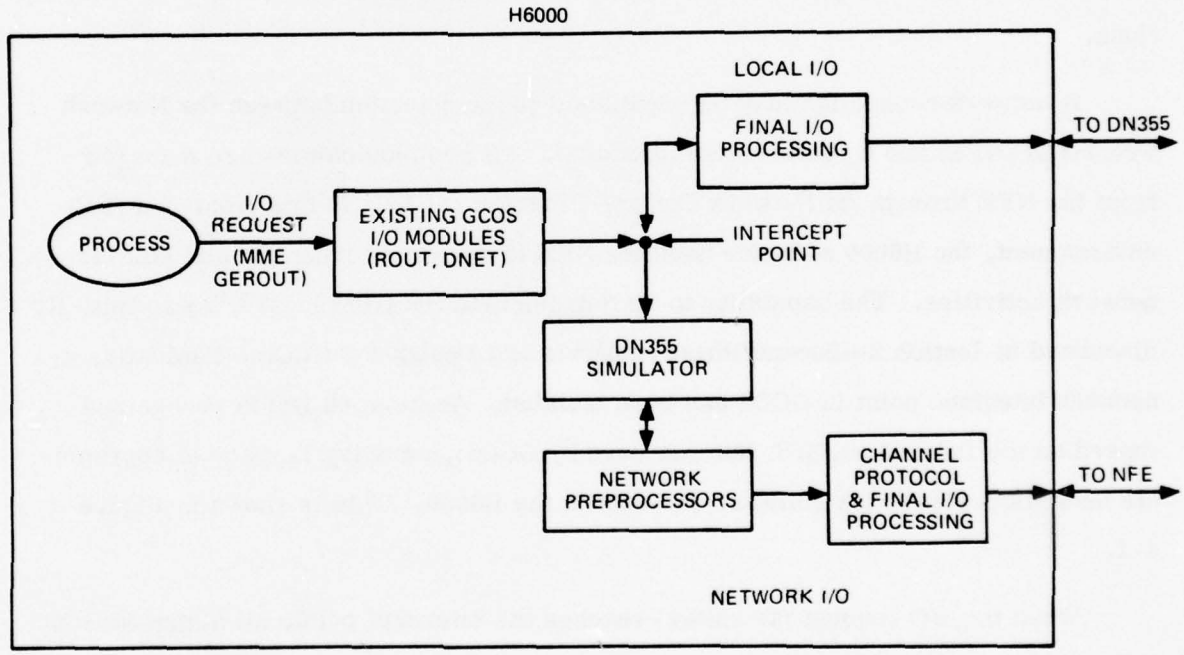


Figure 4-1. Interface I/O Approach

represented by larger segments of coding in the NFE. When the network preprocesses have performed their requirements (for example--framing I/O data, building a process leader, etc.), a call is made to the channel protocol and final I/O processing module. Here, a connection to and actual data transfer to the NFE is accomplished.

It is evident then, that the breach in normal GCOS I/O processing is minor. It may be categorized as being only a few instructions (discussed in Section 5-IOS modification) of code to vector data from its previous path to the network portion of the interface. There is one additional requirement which surfaced in the previous paragraph--a DN355 simulator. This software provides the capability to pass commands and data with existing GCOS I/O modules. These commands and data elements are contained in GCOS structures called mailboxes. The result of DN355 simulation is that all existing GCOS and H6000 user processes are unaware of the existing network interface, i.e., full transparency is achieved.

4.1.1 Scenario

The existing H6000 program inquiry function is issued in a passive manner. If no current request has been received from a remote (e.g., network) user, the inquiry is stored for a future match. The remote user must specify a direct access (DAC) program name to establish user-to-process communication. With this background, we may now explore a design scenario for the GCOS/ARPANET interface.

A DAC program in the H6000 communicates with GCOS via MME GEROUTs when it wishes to communicate with remote users. First, it may issue an inquiry function to GCOS (MME GEROUT, Function 5). If no user (local or network) has requested access to the DAC, the inquiry is stored in a GCOS table to await future terminal connectivity.

A remote ARPANET user will establish a User to Server TELNET connection, much as is presently done across the ARPANET now. For example, a remote Terminal Interface Processor (TIP) user wishing to communicate with some GCOS program would initially enter an "@O<DECIMAL HOST NUMBER>." The

decimal host number is the assigned RADC number which is unspecified as of this paper. When connected to the Server TELNET at RADC, the Server would respond with "ENTER PROGRAM ID." The user then enters the program identifier matching the DAC name with which he desires to communicate.

When the Server TELNET receives the Direct Access (DAC) name, it is passed to the DN355 Simulator (again, as viewed in Figure 4-1). It is the responsibility of the DN355 Simulator to interface with the existing GCOS I/O modules to pass two "arguments" to GCOS. These arguments are:

1. Connect to new terminal
2. Connect to DAC program.

If the DAC name is valid, GCOS notifies the DAC program of the terminal connection and the DAC-to-user activity is started. Existing implementations of remote I/O by GCOS requires the unique identification of the distant end point by a two character terminal identifier (ID). Tables internal to GCOS allow conversion of significance from ID to the specific DN355. Hence, as the DAC program communicates with the network user, the DAC will specify a ID (e.g., the DAC is unmodified) and at the intercept point (actually in a GCOS I/O module called Input/Output Supervisor (IOS), the significance of the Pseudo-Soft (PS355) is interpreted to allow communication from the H6000 to the NFE and ultimately to the remote user.

Connection closing may be initiated by either end of the network. A close by the remote user will generate a close request from the NFE to the H6000. Networking software within the H6000 will translate the request into a "line disconnect" status for GCOS. The DAC program will then determine the specific action required. Connection shutdown initiated by the process is indicated by "line disconnect" remote I/O request. Networking software in the host will generate a close to the NFE which in turn will pass this status to the remote user process.

This scenario discussed major actions in an error-free environment. It envisioned a terminal-oriented user but the same methodology is valid if the scenario presented a Remote Job Entry (RJE) environment. That is, the H6000 remote job

support uses the same I/O flow concepts as discussed here for terminal users. The additions of small file transfer and user TELNET capabilities to the network pre-process programs enables the connection to satisfy all program goals.

4.2 SERVER AND USER TELNET

4.2.1 Server TELNET

Some H6000 resident software is required for Server TELNET to perform the PS355 interface which appears to be a group of terminals to interactive processes. We recommend that all ARPANET TELNET protocol be handled in the NFE. A new protocol is defined to interface H6000 and NFE resident Server TELNET modules. This protocol is referred to as the TELNET Backend Protocol and is shown in Figure 4-2, Server TELNET Software Structure. This protocol will be layered above HFP, is oriented toward terminal operations and should be fairly simple. This structure minimizes the responsibilities of H6000 resident Server TELNET and it becomes essentially an interface between the real Server TELNET in the NFE and H6000 resident interactive processes.

The TELNET Backend Protocol is discussed in a subsequent paragraph. The functional requirements of the ARPANET Server TELNET (discussed in Section 3) include:

1. Go Ahead (GA) control function following the text transmission in an OUT/IN request.
2. Abort Output (AO) control function response, e.g., Abort Output until last line of output text.
3. Are You There (AYT) control function. Respond to "Are You There?"
4. Erase Control (EC) control function. Delete character in current line.

In addition, Server TELNET will support SYNCH and BREAK processes.

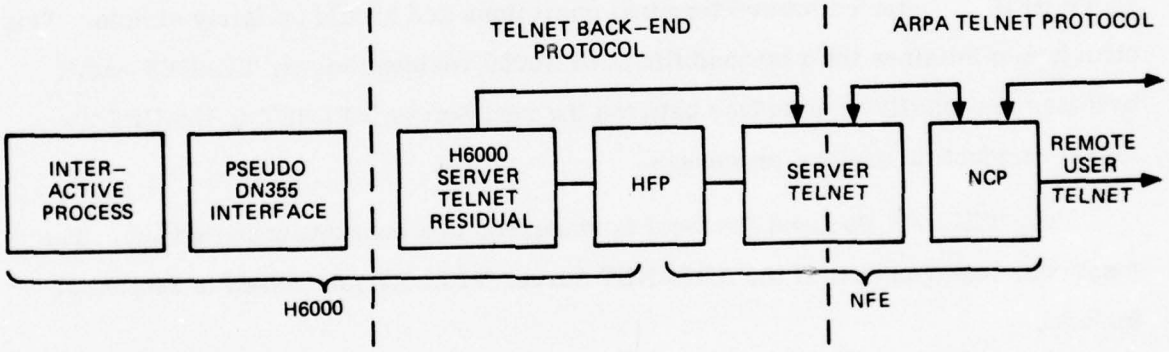


Figure 4-2. Server TELNET - Software Structure

4.2.2 User TELNET

The minimum implementation of a server host does not require a User TELNET capability. A User TELNET component in the H6000 will allow locally connected terminals to access the ARPANET. This capability expands the number of terminals available for networking. If local H6000 terminals are dispersed throughout the facility while network terminals are arranged in a cluster, this arrangement also allows more convenient network access. If both local H6000 terminals and network terminals use dial-up telephone lines, an H6000 based User TELNET capability provides only a small increase in convenience of access to the network.

If a User TELNET capability is identified by the government as required for H6000 terminals, then a structure similar to Server TELNET is possible and is shown in Figure 4-3, Software Structure for a User TELNET.

Here, a fairly simple H6000 resident User TELNET is provided to pass lines of data between H6000 connected terminals and NFE resident User TELNET. Communication between H6000 connected terminals and the H6000 resident User TELNET component will use standard GCOS terminal interface mechanisms. Communication between the H6000 and NFE resident User TELNET components uses Host Front-End Protocol (HFP) to support a fairly simple, terminal I/O oriented protocol known as the TELNET Backend protocol. Regardless of whether an H6000 User TELNET is provided, the NFE will provide User TELNET capabilities to terminals connected directly to the NFE. This allows access to the network when the H6000 is down or dedicated to non-networking activities.

Terminals connected directly to the NFE will have character-at-a-time capabilities in addition to line-at-a-time operation. This character-at-a-time capability increases the facilities available when accessing character-at-a-time hosts. Finally, User TELNET will allow terminals connected to NFE direct access to local H6000 interactive programs. That is, a direct interface between User and Server TELNET capabilities is provided so that NCP and Interface Message Processor (IMP) resources are not wasted when terminals connected to the NFE are accessing the local H6000.

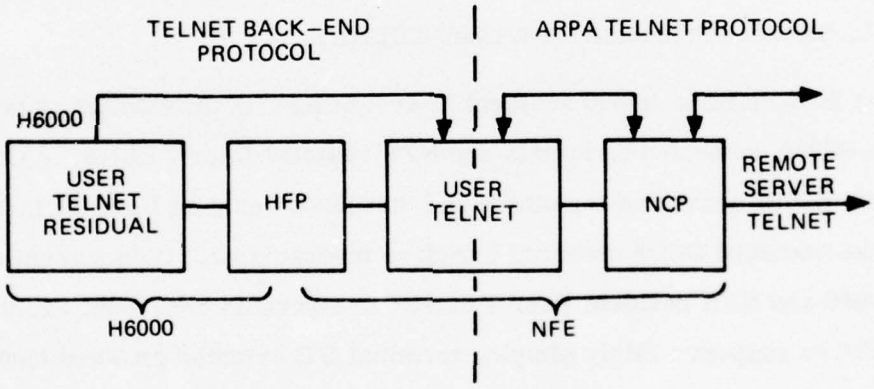


Figure 4-3. User TELNET - Software Structure

4.2.3 TELNET Backend Protocol

This protocol is designed to provide a simple, function oriented interface between H6000 and NFE resident components. This protocol is adapted for easy H6000 implementation and its flow control, like H6000 terminals is half-duplex and line (or block) buffered.

The TELNET Backend protocol is not symmetrical, i.e., commands are either sent "to the terminal" or received "from the terminal." Since this protocol is between processes, the question of where the terminal is may be confusing. For the purposes of this protocol, the Server TELNET component resident in the NFE is the "terminal" to the H6000 resident Server TELNET component. Actually, the user's terminal is connected to a User TELNET at some remote host. Even so, the NFE resident Server TELNET is closer to the terminal and is therefore treated as a Network Virtual Terminal (NVT) terminal handler. If a User TELNET is implemented in the H6000, then it is the "terminal" to NFE resident User TELNET because it controls the real terminals.

TELNET Backend protocol commands from the terminal include: connect terminal to program, disconnect terminal, set break status, output printed, output printed with input, and status query. Commands to the terminal include: print output, print output and return input, terminal connected, disconnect, break acknowledged, and retransmit lost input.

4.2.3.1 TELNET Backend Protocol, Terminal Log

Terminal log-on is normally initiated from the NFE. It establishes an HFP path to H6000 Server TELNET and sends a Connect-Terminal command. The H6000 Server TELNET by using the PS355, will interface with the specified interactive program. This interface with H6000 Server TELNET will appear to the interactive process and GCOS as a normal terminal log-on. Once the interface is established, a Terminal Connected command will be sent to NFE Server TELNET. Any further log-on required by the interactive process will be performed as normal terminal I/O and presumably would be answered by the user. If it later becomes

desirable, NFE Server TELNET could answer additional log-on parameters, provided NFE Server TELNET knows what questions will be asked, and has already obtained the answer (i.e., User Identification (USERID) and PASSWORD) from the network. If H6000 Server TELNET is unable to establish communication with the specified interactive process, a Disconnect command will be issued instead of Terminal-Connected. Unless the NFE Server TELNET wishes to try again, presumably with a different interactive process, the NFE Server TELNET will close the HFP path.

4.2.3.2 TELNET Backend Protocol, Data Flow

Although bi-directional data flow may be occurring on the ARPANET TELNET connection, data flow between H6000 and NFE will be half-duplex and controlled by the interactive process. The interactive process will issue either Print Output or Print Output and Return Input requests.

When the interactive process issues a Print Output request, H6000 Server TELNET will issue a Print Output command to NFE Server TELNET. When NFE Server TELNET is ready for the next output, it will return an Output Printed command.

Instead of Output Printed command, TELNET might return Break or Disconnect commands. In this case the interactive process will be returned a break or disconnect status to his output request rather than a normal status.

When an interactive process issues a Print Output and Return Input request, the H6000 Server TELNET will issue a Print Output and Return Input command to NFE Server TELNET. When NFE Server TELNET is ready for the next output and has accumulated a line (terminated by CR) of input, it will acknowledge the output and return input by issuing an Output Printed with Input command. Instead, NFE Server TELNET may issue Break or Disconnect commands which will not contain any input.

NFE Server TELNET must not acknowledge a Print Output command with an Output Printed with Input or acknowledge a Print Output and Return Input command

with an Output Printed command. Either action will be interpreted as a protocol error.

After a line of input has been sent with an Output Printed with Input, NFE Server TELNET will save it until the next command is sent from H6000 Server TELNET. This allows H6000 Server TELNET to issue a Retransmit Last Input command. When input is received by H6000 Server TELNET and the Interactive process is swapped, the input will be discarded. After the interactive process is back in execution, H6000 Server TELNET will issue a Retransmit Last Input command. NFE Server TELNET should then respond with the normal Output Printed with Input command and the original line of input.

4.2.3.3 TELNET Backend Protocol, Terminal Disconnect

Either NFE Server TELNET or an interactive process may cause a Disconnect command to be sent. Once a Disconnect is issued the interface with the interactive process will be broken. Any outstanding I/O requests will be terminated and any further requests will be denied until a successful terminal log-on sequence has been executed.

4.3 USER AND SERVER FILE TRANSFER

The FTP is divided into two physical entities, one which resides in the H6000 and one which resides in the front-end (FE). The separation is based on a logical division of functionality where those activities required to interface with existing H6000 characteristics are H6000 resident and those interfacing with the network reside in the FE.

4.3.1 Characteristics of H6000 FTP

The H6000 FTP must be capable of performing those functions outlined in Paragraph 3.2 for multiple and simultaneous file transfers. Since these transfers can be initiated locally, or from a distant user, the FTP must be capable of initiating (CONECT) a network connection or accepting (LISTEN) one initiated by someone else. Using the HFP outlined in Section 7 of this document, a logical command path can be established between the H6000 FTP and the FE FTP whenever the H6000 FTP

begins execution. This path will be used to pass commands between the host and FE FTP.

Whenever an FTP is required, appropriate commands and responses over the command path will cause the establishment of a data path between the host and FE FTPs. This path will be used to transfer the file data to or from the H6000 depending on the direction of the transfer. Each data path will be assigned a Path Identifier (PATHID) which uniquely identifies each file transfer and maps it to a unique network connection. All command messages between the host and FE FTPs, will contain this PATHID to indicate which file transfer is to be affected by the command. Other than this simplified HFP no initmate knowledge of ARPANET characteristics will be presumed for the H6000 resident FTP.

It is necessary to maintain an open command path connection between the H6000 and NFE FTP processes. This requires the continuous presence of the H6000 FTP as long as the network is actively connected to the H6000. To provide this continuity of execution the H6000 FTP must be embodied in a privileged slave program which is capable of preventing swapping or moving of itself as long as the command path is active.

The provision for handling multiple concurrent file transfers requires a multitasking, event-driven program discipline. There must also be provisions for internal shared buffer management to allow for efficient core utilization in the multiple file transfer environment.

Functionally, access to four types of files is required by the H6000 FTP: (1) cataloged on-line disk files, (2) cataloged off-line disk or tape file, (3) non-cataloged tape files, and (4) temporary on-line disk files. All information necessary for doing I/O to or from any of these files is totally contained in the Peripheral Assignment Table (PAT) and PAT pointer. This information is contained in a programs Slave Service Area (SSA) and is generally accessible via a unique file code assigned to each allocated file regardless of its type. If a general interface is established in which any slave process can identify a file by file code to the H6000 FTP, the FTP can obtain the PAT and PAT pointer from the user slave and gain access to the required file.

This interface is implementable through a generalized subroutine which will do an Intercom I/O to the FTP process. The FTP process will retrieve the PAT information from the slave programs SSA, disable the slave to allow swapping, establish a suitable PAT in its own SSA and perform the necessary I/O to the file under the slave's control. When the file transfer is completed, the FTP would enable the slave which would then do a second Intercom I/O to get status and accounting information from the FTP. Characteristically, this requires the FTP to have an open Intercom file with a read outstanding at all times.

The local terminal user support introduces another element into the H6000 resident FTP. The function required is to allow a local terminal user to initiate a file transfer between any two network host computers. This requires that the H6000 resident FTP be a DAC within the GCOS environment. When the FTP process begins execution it will initialize it's DAC program name through normal MME GEROUT interfaces allowing a terminal user to obtain direct access to the FTP at any time. Once a terminal has logged onto the FTP, the user may enter the necessary parameters to define the desired file transfer. It is envisioned that the probable implementation scheme used to provide the terminal conversation will be a separate DAC program which is spawned by the FTP. Once in execution the spawned program will establish its DAC name via MME GEROUT. The FTP can then switch the terminal user line to the spawned DAC and the conversation can proceed under the control of the spawned DAC. Once the file transfer is well defined, the spawned DAC program will do any file allocation necessary and issue the Intercom I/O to the FTP to initiate the actual file transfer.

4.3.2 File and Data Structures

It is required that the H6000 resident File Transfer process is involved with file structures. When a segment of records which make up a file are received, the H6000 FTP storage processing will be a function of the transmission mode and data type.

H6000 FTP will scan for end-of-record (EOR) bytes in ASCII type data to produce an H6000 standard system format file. The EOR bytes will be eliminated from the data stream, and be transposed into a GCOS record control word. This word is then appended to the front of each stored H6000 record. The reverse process is required for H6000 files transferred to the network in ASCII byte stream mode.

Binary image mode files present a different consideration. An algorithm is required to provide an efficient storage pattern and enable maximum file space utilization. This algorithm will produce a storage convention which minimizes file space utilization as a function of the negotiated byte size. It is necessary that the algorithm be invertible for H6000-to-network binary mode file transfers.

4.3.3 FE FTP Functions

The selected NFE will provide an ARPANET-standard, file transfer methodology. An interface between the NFE and H6000 file transfer processes will be fabricated and use HFP as the communications vehicle.

The FE takes on the form of a transformer where ARPANET FTP protocol is transformed into the sequence of commands and data path instructions for the H6000 FTP. The NFE will reverse this processing for H6000 file transfers which are to be sent to the network.

Although no data manipulation is anticipated for binary image mode data, the NFE will be required to accomplish 8 and 9 bit, data-byte conversions on behalf of the H6000 for ASCII data.

4.4 REMOTE JOB ENTRY (RJE)

Section 2 presented various alternatives for implementing the GCOS interface to support networking activities. The recommendation and subsequent design is based upon the DN355 software simulation. In part, this recommendation was formulated because it provides access to existing GCOS RJE technology.

The following paragraphs address the design aspects of implementing the ARPANET RJE using the recommended GCOS interface.

4.4.1 DATANET RJE Function Simulation

The problem of supporting server type RJE from the network is reduced to a simple one given the ability to simulate H6000 to DN355 protocol utilizing an H6000 resident PS355 module. As described above the remote job data streams are treated the same as terminal data except for the special interfaces with the RGIN and GEOT modules within the H6000. These interfaces are invoked by the use of special remote computer opcodes presented to the H6000 by the DATANET in the mailboxes used to pass data to and from the H6000. The DATANET knows to use these remote batch mode codes as a function of receiving the special $*$ control cards from the remote batch terminal. To accomplish the network RJE interface, an H6000 process capable of receiving these remote batch control cards and translating them into appropriate PS355 commands, (i.e., causing the PS355 module to use appropriate DATANET to H6000 opcodes when initializing and manipulating mailbox information) must be written such that data from the network can be handed to GCOS and treated as a remote job stream as if it were coming from an H6000 attached RJE station. Since the PS355 is required for Server TELNET, no further modifications need be made to any GCOS modules to achieve this interface.

It seems appropriate and indeed useful to give brief descriptions of the remote job control cards whose functions must be simulated in the H6000 RJE implementation. These job control cards are:

1. $*$ID - identifies remote terminal; associates a logical two character line ID with this line. This ID is unique while the line is connected, and is used with the MME GEROUTs and elsewhere to identify the line.

   "Accept New Terminal" sent to the H6000

   H6000 checks ID for duplicate, then sends

   "Terminal rejected" to the DATANET

   or assigns a line table (T.TAB) entry and sends

   "Terminal accepted" to the DATANET.

2. $*$RCD - initiates the reading of remote batch jobs. Followed by one or more job decks. Puts the line in "batch mode" in the DATANET.

   "Accept input" sent to the H6000

   H6000 sets up more tables in RGIN (prompted by DNET placing an entry in RGIN's program queue, and waiting for RGIN to read the queue entry and issue a GEROUT "Input accepted")

   "Input accepted" sent to the DATANET.

   These are repeated until an entire job is read and spooled to disc by RGIN.

3. SNUMB - not really a $*$ remote batch control card, but still looked at by the DATANET, as well as being sent to the H6000. Begins each new job, and must be there. When one is found, it signals the end of the previous job to the DATANET.

   "Accept end of current job" sent to the H6000

   RGIN wraps up the current job, closes disc file, invokes GEIN, etc.

   "Input accepted" sent to the DATANET.

4. $*$OUT - initiates the printing of output destined for this remote station (i.e., the line ID specified on the $*$ID card). This printed output may be from several jobs, submitted at any time in the past from a remote computer signed on as this line ID (by default), or from the central or other remote computer by specifying this station as the destination of the output at the time the job was submitted.

   "Send initial output" sent to the H6000

   GEOT checks to see if there are any reports for this ID.

   If not,

   "Output Not Available" sent to the DATANET, which sends an

informative message to the remote computer. If there is output, the first block of output is sent with an

"Accept output" sent to the DATANET.

This is repeated, except that for subsequent blocks, the DATANET sends "send output" instead of "Send initial output," until there is no more output for this ID. When the last block of output is sent, it is sent with "Accept final output" to the DATANET.

5. $*$STS - obtain status. This control card is used to obtain status of jobs by SNUMB or of all jobs submitted by this or another station ID. It can also be used to direct jobs destined for this station to another station or to an online printer.

   "Send status" sent to H6000

   If status of a job is wanted, this line is connected to RGIN, and a status request is sent to RGIN through its program queue. If it is a redirect command, GEOT processes the request.

6. $*$ABT - the batch job specified (by SNUMB) on this card is aborted, if it is in the system.

   "Abort" sent to H6000.

7. $*$DIS - causes the DATANET to disconnect this line.

   "Disconnected line" sent to the H6000

   The H6000 (in DNET) sets a bit in the line table, so that in due time RGIN will notice it and stop processing for this line. In the meantime, DNET/ROUT sees the bit is on and doesn't pass any more messages on to the DATANET until the line table is cleared.

These control cards are analogous to the ARPANET RJE commands used to define and control the transmission of a remote job over the network. If a FE resident RJE process transformed ARPANET commands into $*$ control cards, and presented them to a H6000 RJE process, over a command path, the normal H6000

job facility could be essentially implemented for a network user. To the extent that ARPANET RJE commands are equivalent to the analogous $*$ control cards, compatibility and functionality can be achieved. Any inequivalence between network RJE philosophy and H6000 philosophy, will result in a functional limitation, because, as described above, the H6000 view of RJE is rigid and deeply imbedded with GCOS structure. Any attempt at variance can cause extensive GCOS changes.

4.4.2 FE Remote Job Process

A requirement for an FE resident server RJE exists. The FE resident process must be capable of accepting TELNET connections with non-local users in order to accept commands for defining and controlling a remote batch job. The FE RJE would establish a command connection to the H6000 RJE process over which to pass $*$ control cards equivalent to the ARPA RJE commands.

An interface must be provided between the FE RJE and FTP, which will allow a file transfer from the remote host to the H6000 RJE process. A data path between the FE FTP and H6000 RJE, will be used to pass the job stream data which will be converted to H6000 6-bit Binary Coded Decimal (BCD) and handed to GCOS via the PS355. This represents a slight departure from normal FTP operation, however, the difference is transparent to the FE FTP process. The command and data connections between the FE FTP and the remote server FTP are the same as described for the normal FTP procedure.

Once the RJE job stream has been passed to the H6000 RJE all FTP related connections can be closed. The RJE command connections will be maintained to allow for continued control of the RJE process.

It is necessary to accept both 8-bit ASCII files and binary bit stream files. The conversion of 8-bit ASCII to H6000 compatible 9-bit ASCII will be done by the FE FTP. The H6000 RJE will convert 9-bit ASCII to H6000 6-bit BCD card images. Binary bit stream files are presumed to be H6000 compatible mixed binary and BCD data. While no conversion is being done, the H6000 RJE process will unpack the data stream into binary or BCD card images as required. The data connection between the H6000 RJE and FE FTP will be closed at the end of the input file.

4.4.3 H6000 Remote Job Output Retrieval

Within the RJE environment of GCOS, the distribution of output to the remote user is primarily station oriented. The $*$OUT control card in its most general form could cause the output of all SYSOUT being held at the H6000 for that station independent of any user association. Any output destined to that station, either implicitly by virtue of being generated by a job received from that station, or explicitly through the use of a $REMOTE control card specifying that station, would be sent to that station at the receipt of the $*$OUT card specifying that station ID.

Alternatively, an $*$OUT SNUMB control card specifying the particular GCOS SNUMB associated with a particular job could be used to limit the output to that of the particular job. In this case any output, either implicitly directed to SYSOUT via the P* file code, or explicitly through the use of the $REMOTE control card, by the specified job would be sent as a result of the $*$OUT card specifying the associated SNUMB.

An ARPANET OUT command requesting the SYSOUT of a particular job to be sent via an FTP to a remote host would result in all the output being transmitted to that file. Since the SYSOUT file is a mixed media and multiple report file, some separation of output would be required to be done by the RJE process. Individual reports are separated by SYSOUT however mixed media output is multiplexed to the different output devices. This means that one might get card images interspersed with print line images. The RJE separation process would have to collect punch output and send it to the remote host destination after all printed output has been transmitted. The remote user would have to be aware that his single SYSOUT destination file would contain multiple print media reports and punch media output appended to the end of the file.

Print media data format would be NVT ASCII code in blocked format with forms control appended as column 1. Card image would be NVT ASCII in blocked format or binary bit stream depending on the mode of the data.

When the explicit use of separate file cards such as $TAPE or $PRMFL is used, a one to one relationship can be achieved for output files destined to a remote host. In this case separate FTP processes can be used for transmitting these files to the remote host. The only limitation is that temporary files cannot be transferred as a result of an ARPANET RJE OUT command. These files would have to be transferred as a result of the RJE job itself initiating the FTP.

In addition, some ARPANET RJE commands are not supportable within the available GCOS capability. The RJE OUTPUT commands to control output during transmission are not generally supportable by current SYSOUT capabilities in the remote job output environment. An example would be that GCOS SYSOUT does not support restart or repositioning of printed output as a general capability. Some minimal positioning is available through simulation of printer control interfaces which do exist, however these interfaces are extremely limited and are not general in nature. Hence, without significant GCOS modification the support of ARPANET OUTPUT control commands would be minimal in the RJE environment.

4.4.4 RJE From H6000 to a Remote Host

An RJE from the H6000 batch environment to a remote network host requires several functional capabilities. Some of the required capabilities will be presumed to exist by virtue of having been implemented. The others will be additional requirements to support this function.

If we presume the existence of the following facilities, the additional functions required for implementation of RJE from the H6000 batch environment can be specified.

Presumed to exist are:

1. ARPANET compatible RJE server in the remote host
2. FTP capability between H6000 and remote host
3. General TELNET connectability from H6000 Front End Interface Program (FIP) to a remote RJE server.

Needed in addition are:

1. Network RJE utility program
2. H6000 Resident user RJE process (part of the FIP)
3. An interface between H6000 resident FTP and user RJE.

4.4.5 Network RJE Utility Program

The network RJE utility program is a slave process, executable in H6000 batch mode, and has three files attached to it. The first file would contain a set of ARPANET RJE commands which completely identify a remote job to a server RJE process. The second file contains the job stream to be run on the remote host. The third file is a temporary disk file of sufficient size to contain the SYSOUT produced by the execution of the remote job. There may be more than one of these temporary SYSOUT files if more than one distinguishable output file is to be returned to the H6000.

Functionally this utility program simply does an intercom I/O to the FIP resident user RJE process. Through this Intercom I/O the user RJE can retrieve the file information required to build PAT entries within the FIPs SSA, thus enabling I/O to or from any of the files by the FIP resident RJE and FTP processes.

User RJE process reads the RJE commands from the appropriate file, establishes a TELNET connection to the remote server RJE process, and passes these RJE commands over the TELNET connection to the specified remote server RJE. During this process, I/O file names are retrieved from the associated command entries and recorded for use by the H6000 Server FTP. A flag is set to indicate that table entries exist, and for FTP to scan these entries for possible filename matches whenever initiating an FTP to or from the H6000.

Through normal remote Server RJE processing, an FTP request will be initiated to pass the job stream file to the remote host for execution. When the H6000 Server FTP gets the request specifying one of the unique file names logged by the User RJE, the PAT data is retrieved from the User RJE table and used to perform the necessary I/O, to read the job stream for transmission to the network, or to write the output as it comes from the network.

The addition of the special RJE and FTP interface is necessary to facilitate the remotely initiated FTP to temporary files within the H6000, a functionality which is not generally supported by the proposed H6000 FTP process.

The collection of output returned to the H6000 as a result of the remote job execution can be handled in several ways. If returned to a temporary file, that file can be handed to GCOS SYSOUT for output to local online devices via normal procedures. When returned to a permanent file, output can be provided through a separate, locally initiated, GCOS utility.

This implementation will provide a nearly complete ARPANET RJE capability from the H6000. No GCOS changes are required to support this implementation. Figure 4-4 depicts the User RJE flow diagram.

4.5 LOCAL TERMINAL DESIGN OVERVIEW

The NFE-supported local terminals are to be provided with multi-directional connectivity, i.e., the local terminals may:

1. Become users of ARPA Network host resources
2. Become users of H6000 resources
3. Connect to other local terminals.

The services provided to local terminals are to be derived from two non-network software modules and from one network type software module. Respectively, these are: (1) Terminal Driver, (2) Terminal Handler, and (3) TELNET.

Since the actual, Government-provided NFE configuration does not specifically address the type of terminal line interface, this section will only address the functions served by the above software modules, i.e., no detailed hardware or software specifications are delineated.

4.5.1 Terminal Driver Software

As found in a typical Digital Equipment Corporation (DEC) software architecture, an I/O driver is one which supports user-provided software in the accomplishment of physical data transfers. The driver is "argument driven" and, in fact, is completely unaware of the application (e.g., the reason) for data I/O.

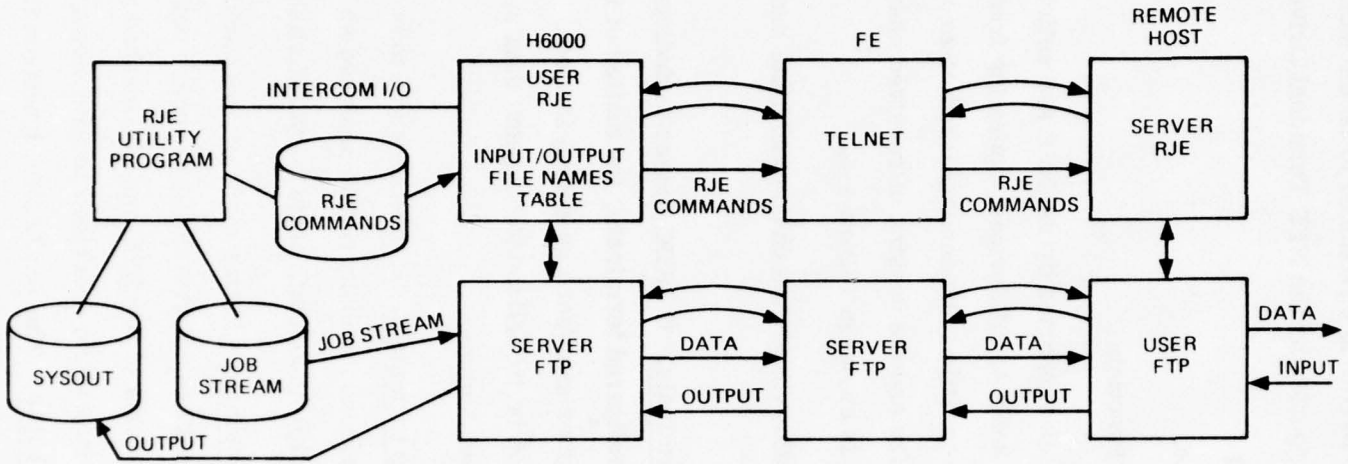


Figure 4-4. User RJE Flow Diagram

The NFE Terminal Driver is envisioned to be the same genus as above. The two primary functions supported by the NFE Terminal Driver are:

1. Terminal Read
2. Terminal Write.

4.5.1.1 Terminal Read Function

Because all terminals supported by the NFE are anticipated to be of the same category (asynchronous, 8-bit ASCII devices, operating from 10-30 cps to as high as 1.2Kbps range), there is little to be accomplished other than determination of the device's characteristics and the passing of received characters "up" to the Terminal Handler. This is shown in Figure 4-5.

Terminal identification (hardware characteristics) can be handled by any of the following methods:

1. Hard-wired terminals. If RADC wishes to dedicate local hard-wired, permanently configured terminals, the matter of identification of the terminal characteristic (such as speed) is moot. Terminal Driver table entries can be initialized to "expect" data in the speed and mode of the configured devices.

2. Isolated dial-up terminals. If the users are to be afforded a dial-up capability, telephone numbers can be assigned so that terminals are isolated to specific incoming, previously initialized ports on the NFE.

3. Full dial-up terminals. In this mode of operation, the terminal users are non-local and are not restricted to access ports. Here, the hardware/software capabilities of the NFE must be meshed so as to support a "hunt" mode, like the hunt mode described in BBN Report 2183 - User's Guide to the Terminal IMP. The RS232 NFE interface is such that a software controlled line speed adjustment is required. Dial-up users will be required to enter a special prompt character for the initial transmission to the NFE. The prompt character enables initialization code resident

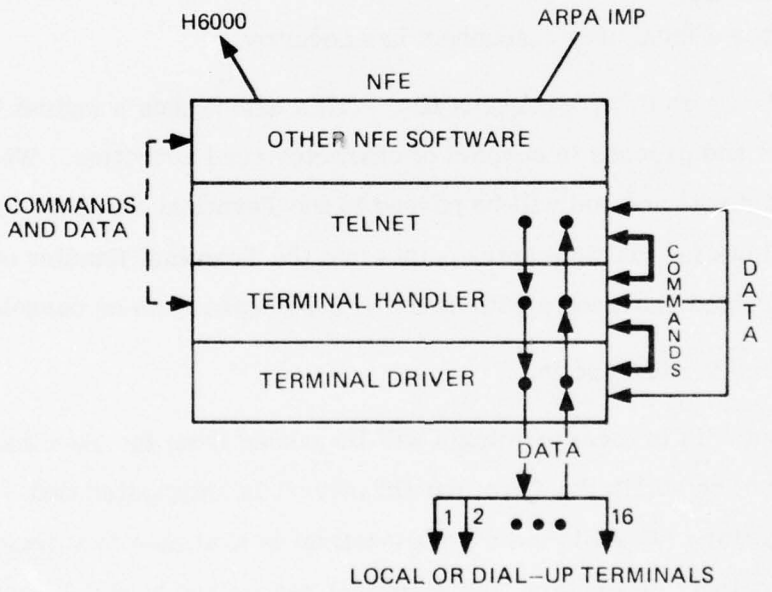


Figure 4-5. Software Support for Local Terminals

in the Terminal Driver to decode and adjust the line interface to the device characteristics. Several terminal hunt algorithms are available, some of which will operate on a PDP-11 acting as an NFE.

The terminal read function of the Terminal Driver supports the character-by-character accumulation and passage to the Terminal Handler. All data and command transfers occur via executive supported primitives. It is expected that the Terminal Driver will maintain a queue of terminal read requests. Commands that the Terminal Driver must support include:

1. RESTART <All ports or specific port ID> - This causes a reset of all or specified port ID table entries. This is useful at system initialization or after a local user disconnect has occurred.

2. READ <port ID>, <process ID> - This establishes a logical link between a port and process in support of character read activities. While most characters received will be passed to the Terminal Handler, the Process ID allows for multiple entry points into the Terminal Handler or connection of received characters with other NFE software such as console routines.

4.5.1.2 Terminal Write Function

Most output data to local terminals will be passed from the Terminal Handler to the Write Function within the Terminal Driver. It is anticipated that a memory buffering architecture (dynamic memory allocation) is available to support the handler/driver output. Therefore, the argument recognized by the Terminal Driver is:

- WRITE <Port ID>, <Address>, <Count>, <Return> - This causes the Terminal Driver to write an amount (Count) of Data (at Address) to the specified terminal (Port ID) with a return to the specified process.

The Terminal Write function will also be able to return status information to the process executing the write function.

4.5.1.3 Multiple Terminal Drivers

Figure 4-5 depicts a single Terminal Driver. If there is to be only one type of interface on the system (e.g., - a DEC DH-11, 16 line multiplexer), most of the Terminal Driver/Terminal Handler functions could be absorbed into a single software structure.

However, if a number of different asynchronous line interfaces are configured by the Government into the NFE, multiple Terminal Drivers may be required (e.g., combinations of such as DECs DL-11, DJ-11 and DH-11 interfaces). With a multiplicity of interfaces, it is easy to appreciate the need for a common Terminal Handler - described in the next paragraph.

4.5.2 Terminal Handler Software

The Terminal Handler software provides a single consolidated interface between the vagaries of device handling on one hand (the Terminal Driver's arena) and the networking or systems considerations on the other hand (TELNET and other NFE software, respectively). As stated above, a design factor in requiring a Terminal Handler to be incorporated in the NFE is the desire to minimize the requirement for TELNET and other software to have device specific knowledge prior to actually accomplishing I/O requests.

A process such as TELNET, that wishes to communicate to a local user terminal, addresses I/O requests to the Terminal Handler specifying a port identifier. The Terminal Handler moves the data from the process task area to intermediate staging buffers, builds table entries associated with the I/O request, schedules a second phase of the Terminal Handler, and returns control to the requesting process. The process is not "blocked" by I/O requests, i.e., in this phase of the Terminal Handler, the process which calls upon the Terminal Handler is only blocked from execution long enough to have the Terminal Handler move output data from the process area to a pool of memory buffers which the Terminal Handler and Terminal Driver use for physical output to terminals. This is shown in Figure 4-6 - Process-to-Terminal I/O Flow.

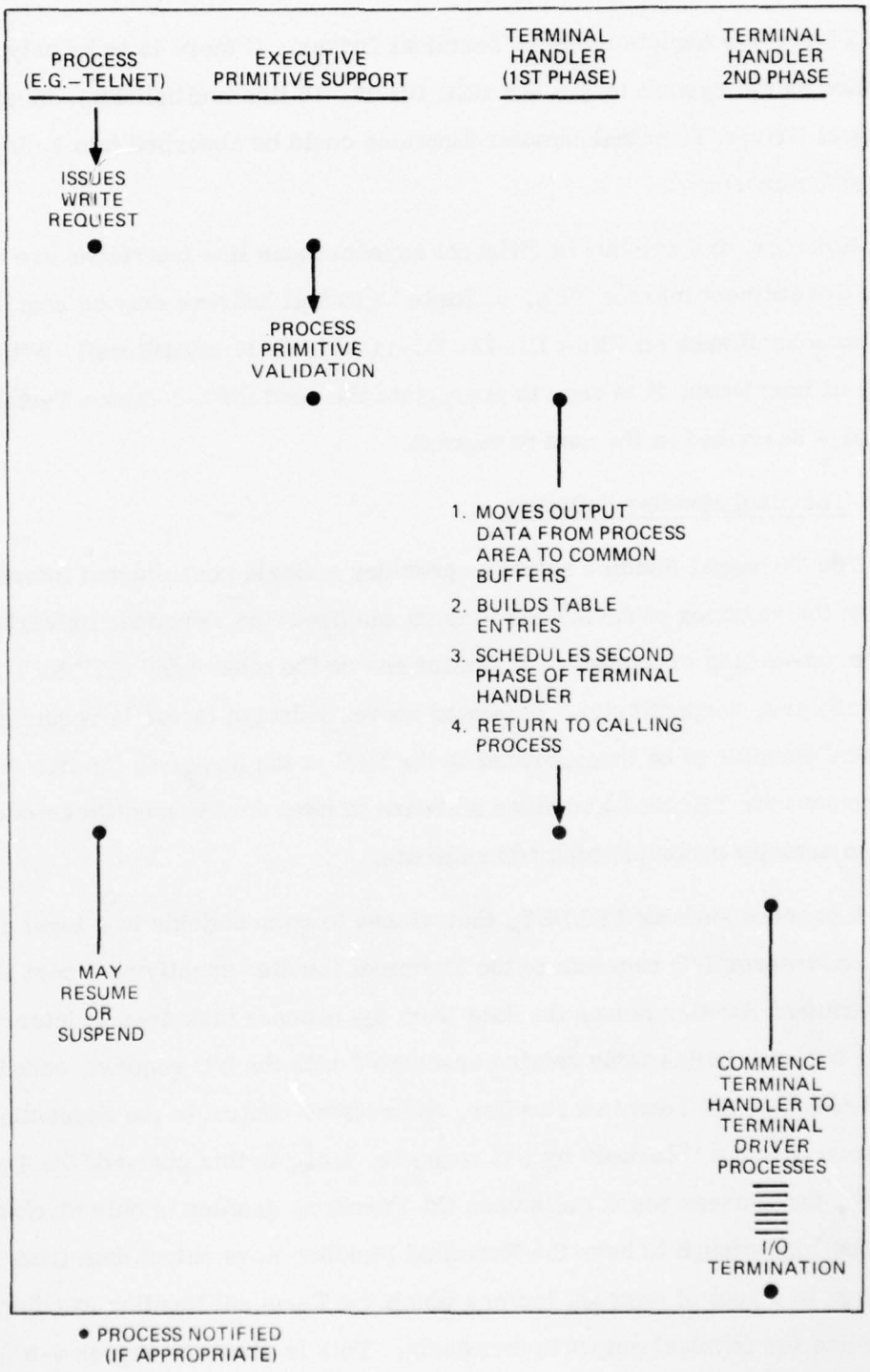


Figure 4-6. Process-to-Terminal I/O Flow

The second phase of a Terminal Handler arbitrates the scheduling and initialization of the various queued I/O requests. It passes read, write, and write/read requests to the Terminal Driver via executive kernel routine calls. This segment of software also maintains an amount of core memory in the form of allocatable buffers. As output physical I/O terminates, this software frees buffers to make them available for subsequent I/O. It may be noted that the Terminal Handler is the selected place to isolate device-dependent processes. For example, if special CRTs were used for local NFE terminals and the CRT's required cursor positioning, video highlighting and similar functions, the Terminal Handler would be the place to isolate the responsibilities, as opposed to requiring them to be placed in TELNET or in the actual device drivers.

4.5.3 NFE-Resident TELNET Software

There is presumed to be a viable TELNET protocol structure resident in the selected NFE software. It is not the intent of this section to specify the TELNET command functions such as IP, AO, AYT, EC and EL. Instead, it is discussed here due to the unique "switching" nature of the NFE TELNET. Also, it is raised as a part of the local terminal support issue.

User TELNET must accommodate NFE local terminal requirements not only in support of ARPANET connections, but also in the establishment of local terminal to local terminal connections and, more important, allow local terminal access to GCOS. The TELNET/NCP repertoire must be rich enough to support this three dimensioned communications environment. With regard to the GCOS access by NFE-local terminals, the same restrictions imposed by the GCOS TELNET are indicated. Examples of this include half-duplex, line-at-a-time operation, and the lack of IP and AO Server TELNET commands.

Within the NFE, a direct User/Server TELNET interface should be established to support local H6000 access by NFE terminals. This interface would avoid the overhead of using an NCP network connection for NFE to H6000 communication. In no case, should communication between co-resident TELNET functions involve traffic through the IMP.

In its interface with NFE terminal software, TELNET must maintain socket/port identification table entries. TELNET will use the port identifier in its executive calls. At initialization (or restart), the Terminal Driver will have a device initialization vector established to support device startup (e.g., - the case of device speed arbitration). Thereafter, TELNET will have a read "posted" for the established local terminals. When a user disconnects from a TELNET dialogue, TELNET will be required to "flush" the significance of the terminal relationship by a kernel-level call to cause the Device Driver to "restart" a specified port identifier table entry.

4.6 GCOS INTERFACE REQUIREMENTS

4.6.1 Discussion of Interface Software

The project activities presented in Section 2 discussed the various options for intercepting certain GCOS I/O and diverting it to the network. A synopsis of benefits and limitations for each approach was provided.

The recommended intercept point in GCOS, is at an entry point in the IOS. The selected approach allows the use of a significant amount of existing GCOS remote I/O processing, which creates a natural Server TELNET entry point within GCOS, and provides the capability to use the existing GCOS RJE facilities.

The ease with which GCOS is breached does not mean that new software is not required. The intercept is executed by patches to the IOS module; this is discussed in the following section - IOS Modification.

Section 6 identifies the software mechanism used to allow the convenient interface with GCOS. The software process functions as a DN355 simulator and is called a PS355. This module translates networking I/O into commands and data structures which are familiar to GCOS.

Section 7 presents the functional requirements for a general purpose H6000 to PDP-11 communications protocol. This program is identified as HFP. HFP allows the establishment of command and data paths between communicating processes in the H6000 and PDP-11.

Standard GCOS I/O structures require programs called channel modules which accomplish the lowest level of protocol, the physical data transfers. Since the interface between the host and the FE is supported by a new device (e.g., new to GCOS) - the ABSI, a final new H6000 module is discussed in Section 8 - the ABSI Channel Module.

These new interface software modules are shown in Figure 4-7, GCOS Interface modules. Some points are worth noting regarding this figure.

First, it may be seen that H6000 DAC processes are not "disturbed" by the addition of the new networking interface software. A DAC process, for example TSS, may communicate unmodified with either network users or DN355 terminal users. In the case of TSS, it may support simultaneous local and network terminals.

Next, the dotted line between the PS355 and HFP is intentional. While the modules previously discussed support the GCOS interface and subsequent data communication path to the FE, we must consider the H6000 residual software impacts. These impacts are discussed below.

Finally, the referenced figure shows a vertical line descending at a mid point between the GCOS modules DNET and IOS. Functionally this is correct. Specifically, the "trap" to the PS355 module occurs within the IOS module.

4.6.2 Residual Networking Software

We have defined the GCOS interface technique. Separate sections specify the requirements for each new software module. However, it is appropriate to complete the H6000 software picture by looking at Figure 4-8. This figure shows the addition of minimum requirement software structures which support the various networking requirements. These residual functions support the H6000 in its response to the NFE's need for TELNET, File Transfer, and RJE. These were discussed in Section 3 - Functional Requirements, and in previous paragraphs of this section. It is upon this software isolation (displayed in Figure 4-8) that the following paragraphs discuss the mechanics of the GCOS interface.

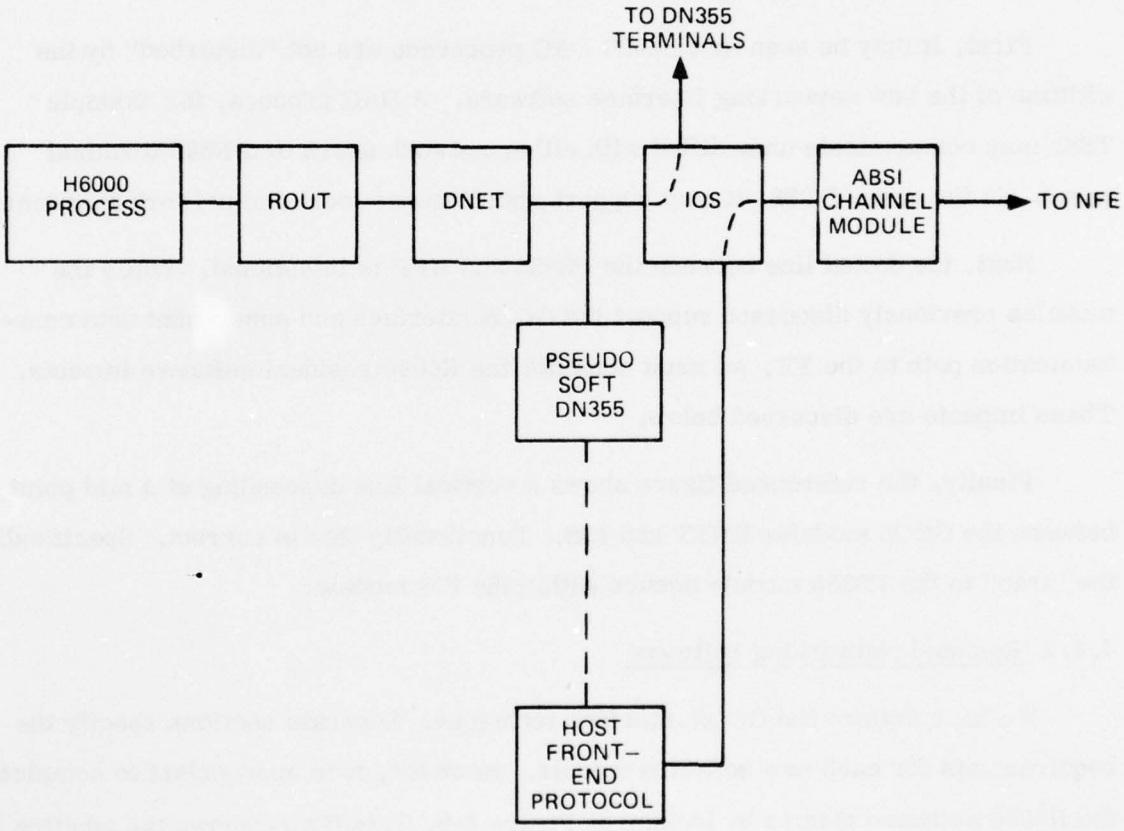


Figure 4-7. GCOS Interface Software

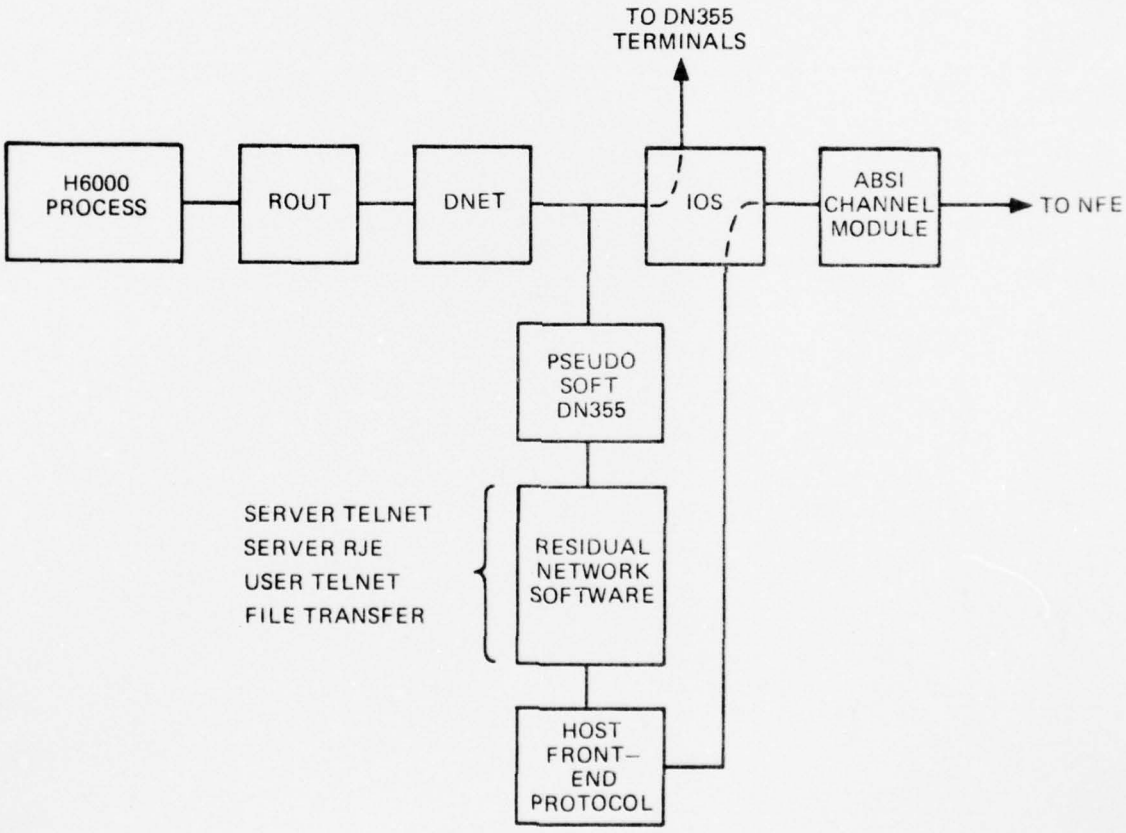


Figure 4-8. Total GCOS Interface Software Requirements

SECTION 5 - INPUT/OUTPUT SUPERVISOR (IOS) MODIFICATION

As previously indicated, the design approach is based on a desire to minimize the General Comprehensive Operating Supervisor (GCOS) changes necessary to support the required functionality. The Datanet 355 (DN355) simulation approach accomplishes this aim with two interfaces required within the Input/Output Supervisor (IOS) module.

The first interface is required to give control to the Pseudo-soft 355 (PS355) module just before an Input/Output (I/O) is to be connected to what GCOS "sees" as the highest numbered and configured DN355. This is, in fact, the network front-end (NFE). The interface occurs in entry point 31 of MIOSO (DCCON). The code required consists of comparing the DN355 index with that assigned to the NFE via the dummy configuration of an extra DN355. If the index indicates that the current I/O to be connected is for the dummy DN355, then a GOTO or direct transfer is made to the PS355 module. Note that the trap would be initialized at startup time as a function of whether or not PS355 NFE was configured. If the network is not configured the trap would be inoperative.

The second IOS interface is required in the interrupt handler (IOTRM) of MIOSO. The interface is necessary to make sure that an interrupt set by the PS355 module is processed. Functionally, not all I/Os normally done between the H6000 and the DN355 will be done between the H6000 and the front-end. However, to simulate H6000/DN355 protocol, it is necessary to generate actual hardware interrupts from within the H6000. These interrupts will be generated by the PS355 module using the Set Memory Controller Interrupt Cells (SMIC) instruction to set the appropriate interrupt cell in the system controller. In addition to setting this cell in the system controller, the Input/Output Multiplexer (IOM) also sets an Interrupt Memory Word (IMW) within the H6000 memory to indicate channel and type of interrupt. There is no way through an executable instruction, to ensure that the processor and IOM are not changing the IMW word at the same time. Conflict between the processor and the IOM could cause an interrupt to be lost so some other interface is needed. Within

the I/O Termination logic the hardware IMW word is picked up and moved into a software IMW word (i.e., used only by software). This software IMW word is then used to process all indicated interrupts. Because we cannot safely modify the hardware IMW word to indicate the PS355 software generated interrupt, it must be factored into the software IMW word during the interrupt processing. Hence, if the PS355 module sets its interrupt information in a different hard core location, then whenever the interrupt handler is moving the hardware IMW bits to the software IMW word, the PS355 interrupt bits may safely be factored in at that point. At symbol IOM16 of .MIO0, the PS355 interrupt bits will be OR'ed to the hardware IMW bits and allow the simulated interrupt to be processed. This then allows the generation of interrupts which look to IOS as if they were coming from a DN355 and thus permits the simulation of the H6000 to DN355 protocol.

SECTION 6 - PSEUDO-SOFT 355 (PS355)

6.1 INTRODUCTION

Having established interfaces in Input/Output Supervisor (IOS) to allow interception of Input/Output (I/O) destined to the network front-end (NFE), and a means of generating interrupts which appear to be the result of successful completion of that I/O, the protocol between the H6000 and DATANET 355 (DN355) may now be simulated.

6.2 H6000/DN355 PROTOCOL

This protocol consists of the H6000 and DN355 passing information back and forth using a set of pseudo-channel mailboxes. There is one set of mailbox areas for each configured DN355. Thus if the NFE is configured as a DN355, it also will have a set of these mailboxes. The mailbox area consists of control information and seven pseudo-channel mailboxes. This allows for the asynchronous, simultaneous processing of up to seven messages between the DATANET and H6000. The mailboxes are located in the H6000's core and are directly addressable by the DN355 via its hardware interface the Direct Interface Adapter (DIA). They are allocated and de-allocated by the H6000, so when DN355 wants to send a message to the H6000, it first must request a mailbox from the H6000. This is done by the DN355 setting some information in the mailbox control area and then interrupting the H6000. The H6000 assigns the DN355 a pseudo-channel mailbox and notifies the DN355 via an interrupt (i.e., connect). The DN355 then uses the mailbox to set up a data transfer to or from the H6000 memory and via the DIA connection executes the data transfer. When the data transfer is completed, the DN355 interrupts the H6000 notifying it of that fact. The H6000 can then de-allocate the pseudo-channel mailbox and enable its use for another I/O.

6.3 PSEUDO-SOFT 355 (PS355) SIMULATION

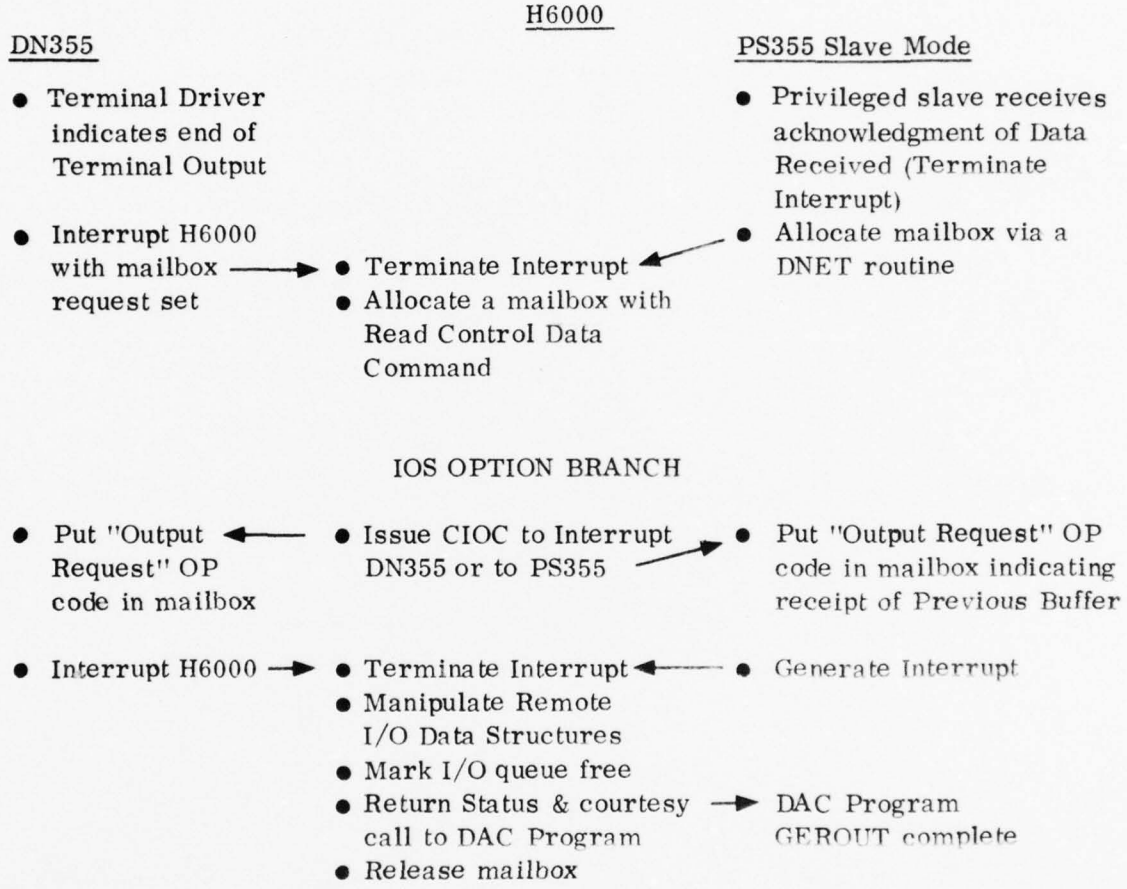
To simulate this interaction, the Pseudo-soft 355 (PS355) module needs to be responsive to an H6000 interrupt of a DN355. Because this interrupt is generated

by issuing a Connect Input/Output Controller (CIOC) instruction on the DN355 channel (which in turn notifies the DN355 that activity is required), a direct transfer to the PS355 just prior to the connect will notify the PS355 that it has activity requirements. Since the PS355 is H6000 resident, it may directly interrogate the mailboxes to determine what function is required. In the case of a data transfer from the H6000 to the NFE, the PS355 module would perform a core to core move of the data, from the user program buffer to an ABSI channel output buffer (the buffer is contained in the privileged slave network program), queue the network program to do an I/O to the front-end, manipulate the mailbox as if it were a DN355, and generate an interrupt using the IOS interrupt process described in Section 5. At this point, IOS and DNET will update their data structures as if the I/O had been completed by a real DN355. To illustrate the simulation process, a flow diagram, Paragraph 6.3.1, depicts what happens for a typical MME GEROUT destined to the network as opposed to one destined to a real DN355.

Functionally the PS355 module is analogous to a DN355 and appears to be just that to GCOS. All processing and data structure manipulation related to remote I/O continues to be done by existing GCOS modules with the exception of the mailbox manipulations which would normally be done directly by the DN355.

These manipulations will be performed by the PS355 module in the case of network I/O. The actual data transfer to the network will be performed independently by the privileged slave network control program using its own data structures and the ABSI channel module. Once the data has reached the front-end, normal ARPANET processes take over and the data is routed over the network to the remote user. The front-end will notify the privileged slave network program that the data has reached its destination, at which point the Datanet 355 simulation process continues as shown in Paragraph 6.3.1.

6.3.1 Flow Diagram



It should be noted that the only GCOS interfaces required to achieve this DN355 simulation are the two IOS modifications specified in Section 5. In addition, the only GCOS data structure which needs to be manipulated is the mailbox area belonging to the network's pseudo DN355. Finally, it is felt that changes in the H6000/DN355 protocol are less likely to occur than changes impacting any other alternative discussed in Section 2. The recommended interface will be maintainable across future GCOS releases.

SECTION 7 - HOST TO FRONT-END PROTOCOL (HFP) HANDLER

This section discusses characteristics of the Asynchronous Bit Serial Interface (ABSI) interface affecting host to front-end protocol (HFP) communications. It also includes recommended HFP, description of HFP handling software, and a description of interfaces between HFP handling software and higher level processes.

7.1 CHARACTERISTICS OF ABSI AFFECTING PROTOCOLS

The ABSI is a full-duplex device which allows simultaneous asynchronous read and write operations. This eliminates the need for line turnaround protocol, which might be required if the ABSI only supported half-duplex operation. To make best use of the ABSI, the protocol should also be asynchronous, i.e., the condition where a write cannot be initiated until a response to the previous write has been read is undesirable because the idle period will significantly reduce throughput.

The ABSI is expected to provide 500 Kbps throughput rate on each channel allowing for a nominal bi-directional data exchange of 1,000 Kbps. Actually the effective data throughput will be less than this due to processing delays in the H6000 and Network Front-End (NFE). Even so, the ABSI is significantly faster than the data flow requirements between the network and the NFE which is unlikely to exceed 100 Kbps effective data throughput. This allows protocol design to be based on criteria other than data throughput efficiency, because as much as 90 percent overhead will still provide acceptable data throughput. The important design criteria for HFP include: ease of implementation, separation from higher level protocols, and the ability to support new higher level protocols.

Normally, the only operations available through the ABSI are read and write. This excludes, for example, the use of multilevel interrupts as part of the HFP. Also, to cause an Input/Output (I/O) operation to occur, one side must have a read request outstanding and the other side must have a write request outstanding. This implies that both the H6000 and the NFE must keep a read request outstanding at all times in case the other machine desires to perform a write.

AD-A038 433 COMPUTER SCIENCES CORP FALLS CHURCH VA F/G 9/2
DESIGN PLAN FOR GCOS/ARPANET INTERFACE.(U)
MAR 77 F30602-76-C-0199
UNCLASSIFIED RADC-TR-77-87 NL

2 OF 2
AD A038433

END

DATE FILMED
5-77

All I/O between the H6000 and NFE will appear to H6000 HFP software to be word oriented. The hardware status words returned with the completion of an ABSI I/O operation, cannot be used to determine the actual amount of data transmitted, because of fill bits inserted by the ABSI hardware. As a result H6000 HFP software must read one word more than the maximum input expected in order to absorb fill bits. Variable length messages must contain a count of actual data in order to account for fill bits.

Conversations with Rome Air Development Center (RADC) and Massachusetts Institute of Technology (MIT), which use an ABSI to connect an Advanced Research Protects Agency (ARPA) Interface Message Processor (IMP) to an H6180 using the Multiplex Information and Computing Service (MULTICS) operating system, indicate that the ABSI is a reliable device. The interface does not use software data validation techniques and treats all hardware detected errors as non-recoverable fatal errors. This is an easily implemented strategy and is apparently successful, therefore, it will be adopted for HFP. If reliability is subsequently determined to be a problem, a more sophisticated error detection and correction strategy can be implemented.

7.2 HFP

The HFP is a low level protocol which allows processes in the H6000 and processes in the NFE to communicate. It is a layered protocol, such that it is ignorant of network connections and other higher level functions. This provides a simple mechanism, easily implemented, that will not be impacted by future development of higher level functions (such as replacing Network Control Program (NCP) or adding National Software Works (NSW) capabilities). The protocol should be asynchronous to make use of the ABSI full-duplex capability although efficiency of channel utilization is not an overriding concern because of the speed of the ABSI.

The proposed protocol is based on the ARPA suggested Front-End (FE) Protocol. It has been modified due to specific characteristics of GCOS and the proposed interconnection; it has been simplified to facilitate implementation. Because the RADC

NFE is not required to support different types of hosts, the number of protocol options specified in the general protocol can be reduced to those desirable for H6000 operation. Further simplification has resulted from the removal of NCP specific information from the protocol. For example, BEGIN and LISTEN commands in the ARPA-suggested protocol contain NCP specific parameters, and are assumed to result in NCP CONECT and LISTEN operations. This is not chosen for the RADC NFE, which contains a variety of processes in addition to NCP.

7.2.1 HFP Commands

The BEGIN command causes a logical communication path between a process in the H6000 and a process in the NFE to be established. This function is not directly related to an NCP CONECT function. If a process in the H6000 wishes to issue a CONECT or LISTEN, it will first establish a path to NCP in the NFE and then issue CONECT or LISTEN functions via HFP data messages.

The BEGIN RESPONSE command indicates the success (or failure) of a BEGIN command. If the BEGIN RESPONSE indicates success, then the path is open and communication may begin.

The END command breaks a logical communication path established by a BEGIN. It is not required if the BEGIN RESPONSE indicated failure. When an END command is received any queued I/O requests will be flushed.

The END RESPONSE command indicates that end processing is complete. It is only a convenient stimulus to clean up control information because an END command is always successful.

The MESSAGE command is used to transmit text between the H6000 and NFE. Each MESSAGE command will send one block of text with a maximum size restriction. The text field may carry higher level protocol as well as data but HFP will treat it all as text.

The MESSAGE RESPONSE command indicates that a previous message command was accepted or rejected. Rejections may be temporary because of buffering

constraints so this command is also used to request retransmission of a MESSAGE.

The INTERRUPT command causes the receiving process to be interrupted and optionally have I/O aborted.

The PARAM command may be used to change or request parameter information for the specified path. It may also be used to verify that the host to FE interface is still operational.

The PARAM RESPONSE is used to return parameter information or to indicate that a requested parameter change has occurred.

7.2.2 Data Representation Between Host and NFE

Compatibility between an H6000 with its 36 bit words, 6 or 9 bit characters and the PDP-11 with its 16 bit words, 8 bit characters presents some difficulty. All I/O between the H6000 and NFE will appear to H6000 HFP software to flow in multiples of 36 bits. The NFE will be responsible for manipulating this data. H6000 input buffers which accept data directly from the ABSI must contain an extra word into which fill bits may be inserted by the hardware.

Each HFP command will be 72 bits in length as this is an even byte boundary in both machines. Each field within the command is a multiple of 9 bits for H6000 convenience. Actual values in these fields will not exceed 255 for 9 bits, so that the NFE will not need to be concerned with handling 9 or 18 bit numbers.

Several predefined data transformations to allow byte boundary alignment will be available to facilitate communication between higher level components in the host and NFE. Probably, data reformatting and character transliteration should be handled by software at a level above HFP. The HFP software in the NFE can insert or delete high order fill bits to allow values to be stored on convenient boundaries. For example, ASCII characters are only 8 bits but are stored in the H6000 in 9 bit bytes. Since the high order bit (2^8) is always zero, it can be discarded. In addition to 8/9 bit conversions, 16/18 and 32/36 bits will be useful. These transformations will be performed in the NFE regardless of the direction of data flow.

7.2.3 Path Addressing

Each logical communication path between a process in the H6000 and a process in the NFE will be known by a Path Identifier (PATHID). The PATHID is a 16 bit number assigned by the H6000 and is unique for as long as that path is in use. A separate generic process identifier will be used to distinguish between known processes in the host and NFE during PATH establishment. After the path is established, the PATHID is the only identifier which will be used.

7.2.4 Flow Control

HFP design will not dictate any intermediate buffering of data by the protocol handling software. This means that write buffers must be held after a write operation has completed until an acknowledgement has been received. A positive acknowledgement allows the write buffer to be released while a negative acknowledgement would require that the write buffer be held until retransmission is requested.

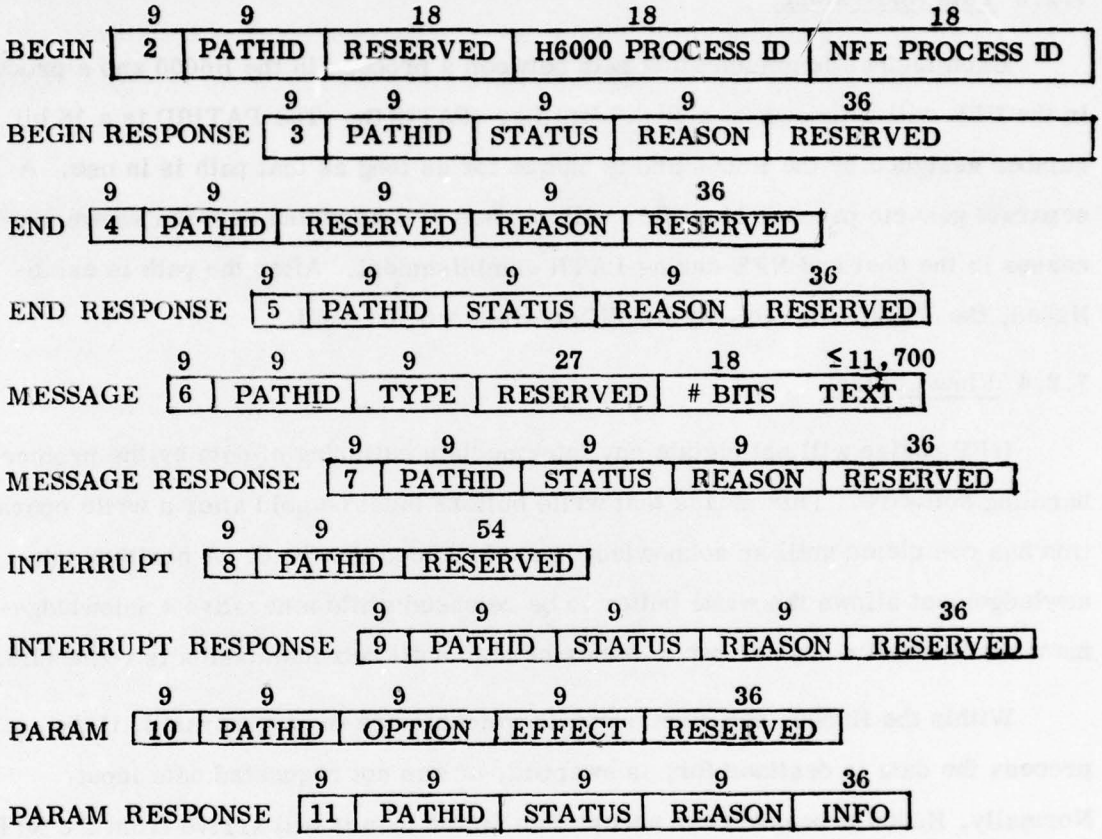
Within the H6000, negative acknowledgements are to be generated, if the process the data is destined for, is swapped, or has not requested data input. Normally, H6000 processes can accept data faster than it will arrive from the NFE, and would be ready for input whenever it is received. However, processing characteristics may not be presumed when designing low-level protocols. Therefore, in order to prevent frequent retransmissions, in the case where a process is not accepting data as fast as it is being received from the network, HFP will include a "ready for data" indication.

HFP flow control is on a per path basis, so that a blockage on one path will not impact other paths. For each path, only one command at a time of each type is allowed in each direction. In other words, when an HFP command is sent on a path, another command of that type must not be sent until a response is received.

7.2.5 HFP Command Format

HFP commands are formatted for H6000 convenience. For the convenience of the NFE, values of 9 bit fields do not exceed 255 and values of 18 bit fields do

not exceed 65,535 (2^{16}-1). This format is as follows:



OP CODE - A 9 bit field indicating the type of HFP command. Opcode zero is illegal. The least significant bit may be treated as a command or command response indicator.

PATHID - The identifier of a logical communication path. For BEGIN commands from NFE to H6000, PATHID will be zero. A PATHID is assigned by H6000 and returned in a BEGIN RESPONSE. This implies that only one BEGIN command from NFE to H6000 may be pending and that PATHID zero is reserved.

PROCESS ID - An identifier of H6000 and NFE processes. A processes ID consists of two 9 bits fields, a generic process identifier and an instance identifier used to differentiate when multiple copies of the same generic process are in execution.

STATUS - Bit flags indicating the success or failure of the command being responded to. 2^0 - 0 = ACK, command was acceptable but not necessarily successful

1 - NAK, command was not acceptable for the reason specified in REASON

2^1 - 0 = OK, command has been successfully performed

1 = HOLD, command could not be processed at this time

2^2 0 = Ready, another command of this type may be sent

1 = Not ready, do not send another command of this type until another response with ready = 0 is received.

REASON - A binary value indicating the reason for a NAK.

0 - not used

1 - invalid command

2 - invalid PATHID

3 - path not established

Also a binary value indicating the reason for an END

<15 protocol error as specified above.

16 - process requested END

17 - process terminated

TYPE - Transformation type to be applied to text of MESSAGE

0 - none

1 - compress 9 bits to 8 bits

OPTION and EFFECT are not immediately implemented

SECTION 8 - ASYNCHRONOUS BIT SERIAL INTERFACE (ABSI) CHANNEL MODULE

8.1 DEFINITION OF H6000 CHANNEL MODULE

It is appropriate to review the purpose, structure and flow of normal General Comprehensive Operating Supervisor (GCOS) Input/Output (I/O) processing using channel modules.

As shown in Figure 8-1, the main body of the I/O processing is contained in the Input Output Supervisor (IOS). IOS contains four separate functional areas (shown as four separate boxes even though IOS is actually a single module). Requests for I/O from processes (MME GEINOS requests), are validated and queued to be issued by the user interface routine. After the user interface routine has processed the I/O request, it calls the start I/O routine in an attempt to initiate the I/O operation as quickly as possible. The start I/O routine calls a device dependent routine referred to as a Channel Module. Channel Modules are separate from IOS and perform device specific processing. A separate Channel Module exists for each different type of peripheral device. This structure nicely separates the general I/O control in IOS from device control in the Channel Modules.

When called from the Start I/O routine, a Channel Module determines if the specified hardware device is busy with a previous request. If so, the request will be queued and control returned to the user process. If the device is available, the requested I/O operation will be initiated and control is returned. Returning to the process is possible because H6000 I/O is performed in parallel with processing.

When an I/O operation completes, a Terminate Interrupt occurs and the IOS interrupt processor receives control. A segment of coding in the Channel Module provides the identity of the interrupting devices. In addition, the Channel Module will perform any device specific processing required and return to IOS. IOS will then signal the process that the requested I/O operation is complete. If additional I/O requests are queued for the same device, the Start I/O routine is entered again.

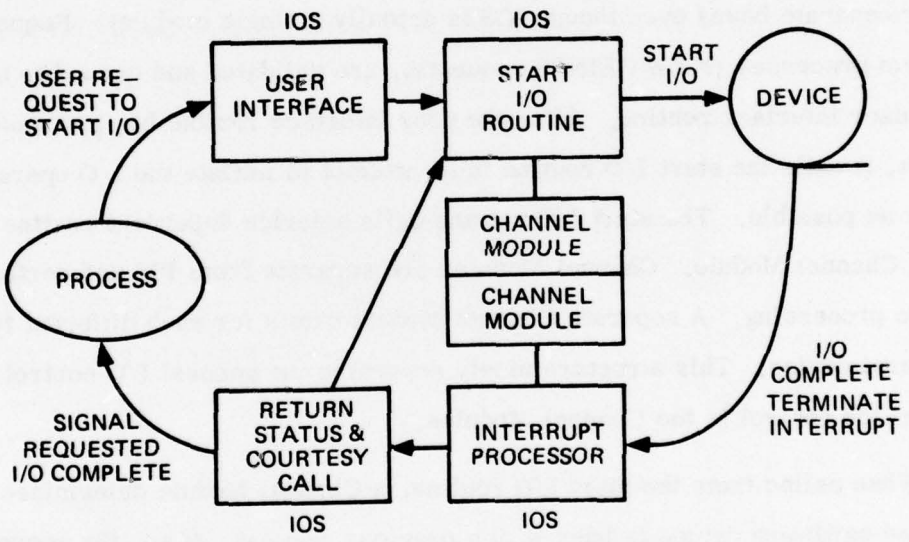


Figure 8-1. Input/Output Processing

Because the Asynchronous Bit Serial Interface (ABSI) device which will be used to interface with the Network Front-End (NFE) is not a standard GCOS device, no channel module currently exists for it. The interfaces between IOS and Channel Modules are sufficiently flexible to allow the development of an ABSI Channel Module. Due to the structure of I/O processing in GCOS, no significant IOS modifications will be required to perform ABSI I/O, just the implementation of an ABSI Channel Module.

8.2 CONFIGURING THE ABSI

Normally, a description of the types of peripheral devices attached to Input/Output Multiplexor (IOM) channels is given to the GCOS startup program (INIT) in the boot deck (CONFIG section). INIT decodes this information and builds GCOS tables describing the hardware configuration. During startup procedures, IOS is loaded and its initialization code examines the primary configuration table and requests appropriate channel modules be loaded to handle the configured devices.

To use this technique to load the ABSI channel module, GCOS device types must be expanded to include the ABSI. INIT will require modification to recognize ABSI configuration cards and to build configuration table entries. IOS will require modification to recognize the ABSI device and request that the ABSI Channel Module be loaded. This is straightforward and will be easily understood by operations personnel. This approach is somewhat sensitive to release changes in that an amount of re-implementation will be required with each GCOS source release.

An easier, although less straightforward approach is available. A single MASK card could be used to change the definition of the ABSI Channel Module in a table in INIT. When networking capabilities are desired, a MASK card is used to make the ABSI Channel Module a "required, Hard Core Module." This will automatically cause the ABSI Channel Module to be loaded without intervention by IOS. When networking capabilities are not required, a different MASK card will be used to make the ABSI Channel Module "not required;" it would, in this case not be loaded. It should be noted that this change, like any other system reconfiguration, requires a system boot (warm boot) to be effective. Since INIT would not be building ABSI configuration tables, the

ABSI Channel Module, during its initialization would be required to configure the ABSI. Fortunately, space is reserved in the important tables (.CRMB1, .CRI01,. CRCT1) for all 32 IOM channels regardless of whether or not a device is currently configured on that channel. The ABSI Channel Module will fill in these open slots with appropriate information which will later be required by IOS.

8.3 ABSI CHANNEL MODULE AND HOST FRONT-END PROTOCOL (HFP) INTERFACE

The ABSI Channel Module is strictly a device driver and is completely unaware of Host to Front-End Protocol (HFP). Assuming that HFP will be implemented by a privileged slave program which is the only program allowed to perform ABSI I/O directly, then the ABSI Channel Module is straightforward. The HFP will be implemented in a program known as the Front-End Interface Program (FIP) and will perform most of the work associated with ABSI I/O.

To minimize the impact of new GCOS releases on networking software, the FIP will not use an MME GEINOS to issue ABSI I/O. Instead of modifying the I/O request processor, which verifies an MME GEINOS request and manipulates the GCOS data structures (particularly I/O queues) to cause the requested I/O operation to be performed, FIP will issue its own ABSI I/O via GCOS QUEUE and LINK functions. ABSI I/O will be implemented in this manner so as not to require the modification of the I/O request processor. The I/O request processor verifies MME GEINOS requests, manipulates I/O queues and other GCOS data structures.

The FIP will issue its own requests for I/O, including specification of ABSI hardware commands. It will be responsible for processing status returned after an I/O operation has completed. The FIP will be responsible for ABSI initialization, error detection, and declaration of "host down" conditions (except when H6000 failure occurs). The ABSI Channel Module and IOS will only be concerned with signaling the hardware to start an I/O operation, (filling mailboxes and issuing a CIOC instruction), and signaling FIP that an I/O operation is complete (returning status and courtesy call).

Within the responsibilities of starting I/O and I/O termination, the ABSI Channel Module must provide special processing for read requests. Due to the nature of the ABSI hardware, the H6000 must keep a read outstanding in case the NFE desires to pass to the H6000. Although this read operation may only require a few milliseconds (ms) to complete, during periods of idleness the read may remain outstanding for longer periods of time. If the I/O is issued in the normal manner (with data from the ABSI being placed directly into a buffer within FIP), the FIP will be "locked in core" for as long as the read remains outstanding. GCOS cannot swap or terminate a job while an I/O request is outstanding. Normally an I/O request completes in a few MS, so this presents no problem. To reduce system degradation by the FIP having an I/O "in transmission", the ABSI Channel Module will issue read commands into a core resident buffer instead of within FIP. During this time, read requests issued by FIP will be left in a "linked" status which usually would indicate that a previous request is outstanding. When the read completes, the ABSI Channel Module will move data read from the core resident buffer into FIP's buffer. The ABSI Channel Module will then process the I/O termination normally, i.e., return status and courtesy call to FIP.

To maximize ABSI channel utilization, the ABSI Channel Module will issue a read request into core resident buffer even if FIP has not yet issued another read request. It is assumed that by the time this read completes, FIP will have requested another read. When the read completes, if FIP does not have a read request queued, the ABSI Channel Module will enable FIP to "wake it up" to issue another read. Until FIP accepts the data in the core resident buffer, another read operation cannot be started. This condition will occur infrequently because FIP will be using a sufficiently high urgency to keep it running.

8.4 ABSI INITIALIZATION AND TERMINATION

The ABSI contains a relay which is used to indicate host up and host down conditions. A similar relay exists for "NFE up" and "NFE down". Before the ABSI will accept read or write commands, both host and NFE must be "up." FIP will issue the special commands to set "host up" and to wait for "NFE up." The ABSI Channel Module will be ignorant of these conditions. In the event of a fatal protocol error or

if the console operator requests, FIP will set "host down" to signal the NFE. If an H6000 system failure occurs, FIP will not be able to issue "host down" and the console operator should set "host down" by pushing initialize buttons on the ABSI IOM or system console. This will signal the NFE so that it can reset its tables.

SECTION 9 - FRONT-END (FE) PROCESSOR REQUIREMENTS

9.1 NETWORK FRONT-END (NFE) EXECUTIVE REQUIREMENTS

The use of a PDP-11 as a Network Front End (NFE) imposes certain functional requirements on the operating system. Three possible operating systems for the NFE were examined, ELF II, UNIX, and RSX-11M, and a composite set of functional and general requirements for the operating systems selected. These functional requirements are divided into three areas:

1. Processor Management
2. Storage Management
3. Input/Output (I/O) Management.

Each of these areas are discussed below, as well as the general requirements of the NFE executive.

9.1.1 Processor Management

The NFE must be able to support simultaneous connections between the H6000 and ARPANET, as well as users connected directly to the PDP-11. Consequently, the executive must be able to support a multi-task, multi-user environment. This involves sharing the processor time between several competing tasks and maintaining the status of each task to determine if it is ready to execute. As appropriate to available facilities, the executive schedules the next task to be executed, according to priority, or some similar arbitration scheme. It shall have a mechanism to allow tasks to be created, terminated, and executed, and to allow inter-task communication and synchronization.

To support multiple users, the executive will support multiple terminals and probably require login and logout commands. To protect users it is desirable to be able to establish and identify passwords and maintain accounts of individual users.

A real time clock is necessary to permit task synchronization and proper accounting. The tasks shall be able to access the current time and to be activated

after a selected time interval or at a particular time of day. To maintain the real time clock, commands shall be available to users to set and/or get the time of day.

In addition, commands shall be available to initiate and terminate tasks, and provide status of the active tasks. The system shall also include some debug capability; at least the ability to examine and modify memory locations in the user's virtual address space.

The executive must be able to recover from processor error traps, which are due to conditions such as an odd address on a word access, an access to nonexistent memory, and an attempt to execute a non-existent instruction. It shall also be able to save the status of the system in the event of a power failure, and resume execution, upon restoration of power with reasonable process and data integrity.

9.1.2 Storage Management

In a non-memory management environment, the PDP-11 can support only 28K words of memory. This is insufficient for a system that will serve multiple users as well as serve as an NFE for the H6000. Consequently, the executive must be able to support memory management, which allows up to 124K words of core to be accessed.

To implement a suitable NFE, a directoried disk file system is essential. This file system can also support local program development. Since there will be multiple users, a file protection scheme is also required, to prevent one user from destroying other users' files. Further, because of the volume of data that will be passing through the NFE (e.g., from/to the H6000), a dynamic storage allocation capability is needed in the executive. This is important because the NFE may have several connections transmitting data at different rates. It is not known a priori what the buffering requirements will be for each connection; dynamic storage allocation is thus required in the executive.

9.1.3 I/O Management

The NFE executive is responsible for managing the various I/O devices connected to the PDP-11. It shall support both character and block I/O operations, and it will certainly include device drivers for terminals, disks, synchronous and asynchronous communication lines such as DH11, DL11, IMP-11A, line printers, and dual cassette units. A desirable feature is an easy method to write and install drivers for additional devices not initially supported by the executive.

The executive must also provide a link between user programs and the device drivers. It shall support commands such as read and write data, open and close files and devices, attach and detach devices to specific tasks, and abort operations already in progress. A time-out feature on I/O operations is also necessary to prevent the NFE from waiting indefinitely for a response from pending I/O.

The executive is also responsible for queueing I/O requests to a particular device and buffering data as needed. It shall allow an I/O request to be performed synchronously or asynchronously with respect to the issuing task, and in the latter mode, the executive will have a mechanism to inform a task when its I/O is completed. The ability to redirect I/O requests is useful in debugging situations, and also when normally configured devices are inoperative.

9.1.4 General Requirements

In addition to the above functional requirements, there are several other desirable characteristics of an NFE executive. It is desirable to have a program development capability, which includes support for editors, assemblers, compilers, linkers, and loaders. It should be both disk-bootable and bootable via a down-line load. The executive should have low execution time overhead and occupy as little core as possible, leaving maximum memory for applications programs and buffers.

The executive shall be modular, so that it is relatively easy to expand or modify. It shall be debugged and reliable, and include maintenance support from the supplier.

9.2 STANDARD ARPANET SOFTWARE

Earlier sections defined the NFE processor as a "resource" against which certain functions may be allotted. The broad categories of functional support that the NFE will provide include: H6000 (GCOS) interface, local terminal support (including operator interface) and finally, standard ARPANET software.

The categories of ARPANET software are:

1. ARPANET interface
2. Host-to-host protocol
3. Applications protocols.

These categories are briefly discussed in the following paragraphs and are assumed to be existing capabilities of a Network Front-End Processor (NFEP) - whether manifest or latent.

9.2.1 ARPANET Interface

The PDP-11 NFE minicomputer interface to the ARPANET is anticipated to use an IMP-11/A type device. The IMP 11-A is a dual Digital Equipment Corporation (DEC) DR11-B direct memory access (DMA) arrangement configured on the PDP's UNIBUS. Our interim report identified some minimal risks in the area of this interface device, however electrical compatibility appears inherent. There will be a premium overhead caused by the software requirement for bit transliteration on most H6000 to NFE to Network data transfers. (See the section of this report entitled "Implications of PDP-11 Requirements.")

The actual software required for the ARPANET interface is composed of an IMP 11-A Driver (the same driver as is required for the NFE-to-H6000 interface) and that portion of the Network Control Program (NCP) (see paragraph following), referred to as Host-to-IMP (HI). The HI function is specified in BBN Report 1822 - Specifications for the Interconnection of a Host and an IMP. Actual program implementation is dependent upon the selected NFE operating system. Basically, the HI function is required to maintain messages (segments) and control data which are queued for transmission to the IMP or which have been received from the IMP.

If the IMP is non-local, (defined as Host-IMP distances greater than 2,000 feet) the HI function must be modified to include the Reliable Transmission Package (RTP) an implementation of the Very Distant Host Protocol. The RTP requires the transmission of packets which are multiples of 16 bits in length across a radio supported interface. The RTP is also defined in Appendix F of BBN Report 1822.

9.2.2 Host-to-Host Protocol

The NFE processor will appear to be a Host to the ARPA Network's IMP. The NFE is required to have implemented the second layer of protocol (that is, a protocol above and feeding the HI discussed previously). This layer is know as the Host-to-Host protocol, or as implemented in the NFE, NCP.

The NCP is a network communications protocol which masks the often found differences in host word lengths, command formats, and data transfer characteristics. Primarily, NCP is responsible for process connection establishment, maintenance and flow control. NCP is fully specified in the ARPANET Protocol Handbook (NIC7104), April 1976.

There are a number of existing PDP-11 supported NCP implementations. Most immediate systems from which a viable NCP could be considered include the UNIX networking executive, the ELF-II system, Massachusetts Computer Associates (MCA) RSX-11/M system and the Haskins Lab RSX-11/D system. It is safe to reason that a DOD-wide search would reveal even more PDP-11 type NCPs which are based on the protocol specifications. The point at issue with regard to the NCP (and unlike the probable development in Paragraph 9.2.1 above) is significant, i.e., under which criteria is the remaining system development to be performed. If the NCP is to be constructed or borrowed from a current generation operating system, the field is probably limited to two choices - UNIX or RSX. This paper addresses only the required functions for the NCP.

The NCP functional requirements, as defined in NIC7104 include:

1. Connection Establishment. Sender to Receiver, (STR) and Receiver to Sender, (RTS) messages
2. Connection Termination. Close, (CLS)
3. Flow Control. Allocate, (ALL); Give Back, (GVB); Return, (RTN) messages
4. Interrupts. Interrupt, (INR); Interrupt by Sender, (INS)
5. Miscellaneous. Queue maintenance, time-outs, illegal conditions, and miscellaneous commands such as ECHO, (ECO); ECHO reply, (ERP); Reset, (RST); and NOP.

Those processes in the NFE or H6000 which communicate with the network will use executive level primitive calls specifying socket numbers, host addresses, priority and similar argumentation. The process calls will typically appear as I/O calls with parameters requesting some of the above services of NCP. Examples include: (1) establish connection, (2) listen, (3) close and get/set NCP parameters.

9.2.3 Applications Protocols

Previous sections have discussed software protocol requirements which are passive in nature, in that they only support and isolate other system components from the networking aspects of this interface. Before discussing the reason or application aspects of the GCOS/NFE/ARPANET capability, a review of the supporting software structures is presented.

9.2.3.1 Software Support Structures

The software support structures are comprised of:

1. IMP 11-A Driver. This software supports a PDP-11 device and performs the lowest level (e.g., physical data transfer) protocol. An IMP 11-A Driver is to be used in support of the NFE's communication with the H6000 and the IMP

2. Terminal Driver. This software supports a selected PDP-11 multi-line interface for local NFE terminals

3. Terminal Handler. This optional software is intended to support communications between any number of NFE components and local terminals. The Terminal Handler is intended to mask the possible use of different terminal interfaces on the PDP-11

4. Host Front-End Protocol (HFP). This software supports PDP-11 to H6000 communications. It is a general purpose communications mechanism which has no awareness of applications requirements for the H6000 ARPANET interconnection

5. TELNET Protocol. TELNET software supports local and remote terminal access to remote processes. Differences in terminal characteristics are made transparent because TELNET creates a virtual terminal for network communications. The Server aspects of TELNET are supportive only of the options and capabilities of H6000 terminals

6. NCP. NCP is discussed above. It supports process to process connections and flow control. Depending upon implementation technique, a portion of NCP will contain HI subprotocol.

The preceding defines the interface specific support software structures. There may also be a requirement for operator interface, statistics, and similar ancillary support items.

9.2.3.2 File Transfer Application

The basis for a File Transfer application is discussed in Section 3.2. There are any number of scenarios envisioned for file transfers serving H6000 or remote users. The transfers may occur in either direction.

The requirement for a File Transfer necessitates an ARPANET compatible protocol, resident in the NFE which support the capabilities and limitations specified in Section 3.2.

The basis for the interaction and support for file transfers is contained in the FTP specification dated August 12, 1973. This is Network Information Center (NIC) document number 17759, and is also available as RFC 542 via the ARPANET or a member of the Network Working Group.

Since there are presumed nuances of network implementation techniques, and certain growths in file transfer set-up, negotiations and implementation, the NIC 17759 is only considered to form the approach upon which the GCOS front-end file transfer is to be implemented.

The user interface to FTP will appear as TELNET strings and support the GCOS file command entries such as USER NAME, PASSWORD, ACCOUNT, and others. The data storage and transmission capabilities include 9 bit H6000 standard ASCII and binary.

The service commands originated by the File Transfer include RETRIEVE, STORE, RESTART, and STATUS. The use of H6000 Time Sharing (TSS) via TELNET will allow for the allocation and deletion of H6000 catalogued files.

9.2.3.3 Remote Job Application

The SOW has defined a requirement for application support so that network users may enter GCOS job streams. RJE is fully supported within GCOS, and the selected interface scheme in this document fully utilizes it. The fact that the users are distributed across the ARPA Network is completely transparent to GCOS.

The NFE will share a subset of the remote job requirements--mostly in the area of the job set-up (any required negotiations or file transfers) and the transliteration on output to or input from the H6000.

Links between the NFE and the host H6000 will be established to support command and data transfers. As in the File Transfer above, the basis for an RJE concept rests in the capabilities and limitations for H6000 RJE (discussed in Section 4.4) and as presented in the RJE Protocol document-NIC 12112, dated October 16, 1972. This document is also available as RFC 407 via the network.

The entry of remote jobs presupposes a verbose exchange between the network user and the NFE (which in turn mandates NFE to H6000 interactions). Whether the user selects a "one step at a time" approach using a TSS interface to the H6000, or whether RADC selects a more automated approach, as in the RJE protocol, is not at issue here. This section deals with the remote job facility as a set of functions which must be satisfied. If the network user is to use the TELNET/TSS interface, many of the requirements to enter network jobs are latent in other capabilities. This anticipates a knowledge and willingness of a remote user to go through several steps to create necessary H6000 files, spawn the job, retrieve status, and output.

As specified in the RJE protocol, a more basic, general purpose net-language using simplistic commands is also feasible (and may be recommended as a long-range implementation option dependent on a network user profile). Here, the user addresses a TELNET-to-RJE connection and passes user identification, password, source, and output type commands.

If the source or destination imposes a file transfer requirement, it is performed automatically. The user is required to have only minimal understanding in this approach (excepting the GCOS remote job file and command formats) but does have access to a rich inventory of job, status, and output manipulating alternatives. The simplicity of this approach should not be construed to mean that an alternating dialogue between the server RJE and the user does not take place; it's just less H6000 specific.

The more formal (yet possibly operator simplistic) remote job capability enlarges the role the NFE plays in dialogue, negotiation, file transfers, job execution, and output disposition. The importance here of an RJE implementation which is NFE supported is that it may easily be excised for replacement by NSW functional support.

9.3 GCOS INTERFACE SUPPORT SOFTWARE

The H6000 will communicate with the ARPANET via the NFE, which will be a PDP-11. The NFE will have an executive, networking software to communicate with

ARPANET, and software to interface with the H6000. This section describes the software required to implement the interface with the H6000.

This interface consists of a two-level layered protocol. A low-level protocol controls the connection between a process in the H6000 and a process in the PDP-11. The higher level directs the PDP-11 in its communications with ARPANET.

The information being transferred in the lower level protocol consists of 72 bit protocol messages and data messages of up to 16,700 bits. In one transmission one block is sent, which may contain up to three protocol messages, followed by one data message, as illustrated in Figure 9-1. A block has a fixed maximum size of 11,916 bits. This is composed of up to 325 - 36 bit H6000 words plus up to 3 - 72 bit protocol messages.

There are five types of protocol messages; each has a corresponding response.

1. BEGIN - establish H6000 to PDP-11 connection
2. END - terminate H6000 to PDP-11 connection
3. DATA - transmit data--must precede data message
4. INTERRUPT - cause software interrupt
5. PARAM - used to change or request path parameters.

The software in the NFE that handles the lower level protocol must perform the following functions on transmissions initiated by the H6000 as illustrated in Figure 9-2:

1. Input the transmission block to an allocated buffer. To accomplish this, the interface to the H6000 is normally kept in a "listening" state
2. Examine and interpret the 72-bit protocol messages, calling appropriate routines depending on the type of message
3. Perform appropriate memory allocation and process initiation to create a process as required in a BEGIN command

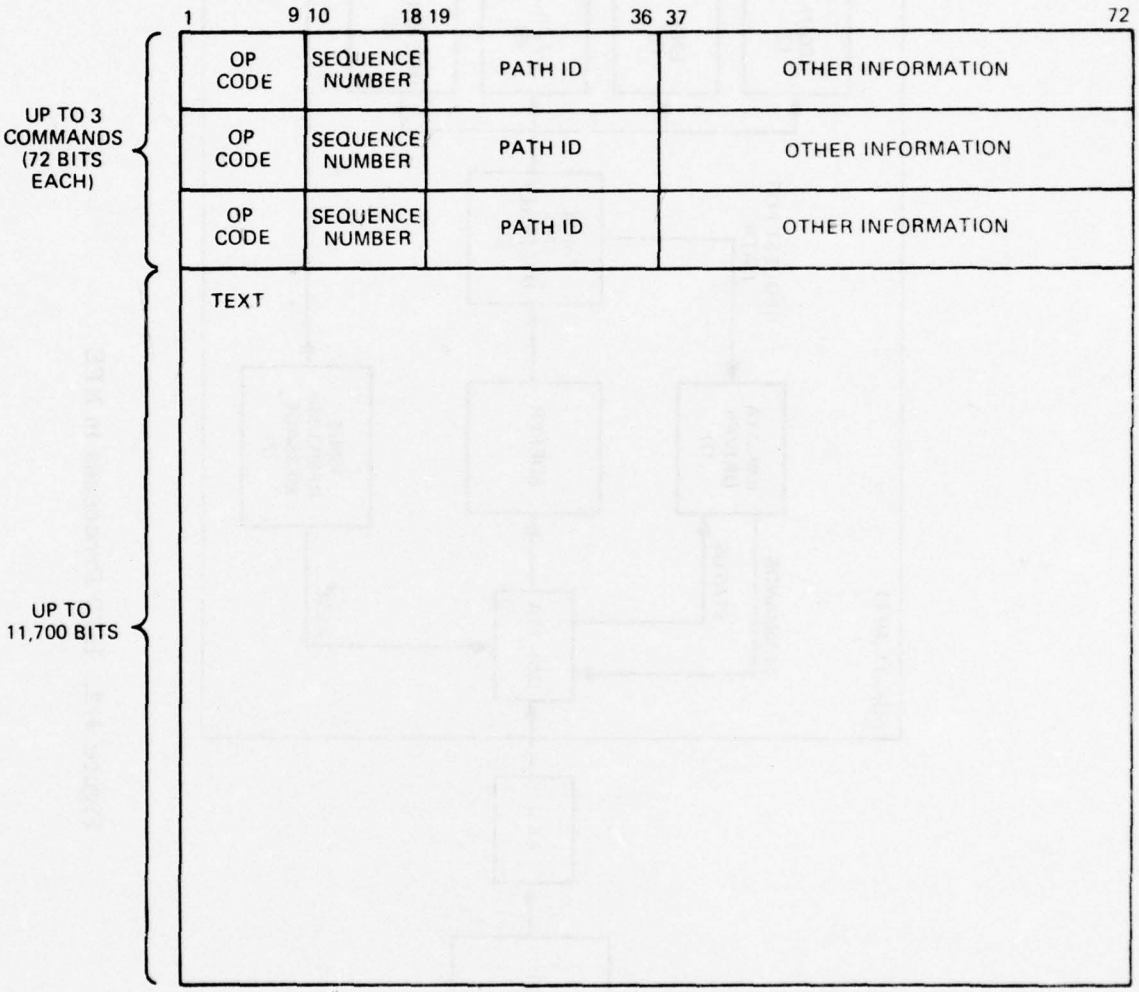


Figure 9-1. H6000/NFE Transmission Block

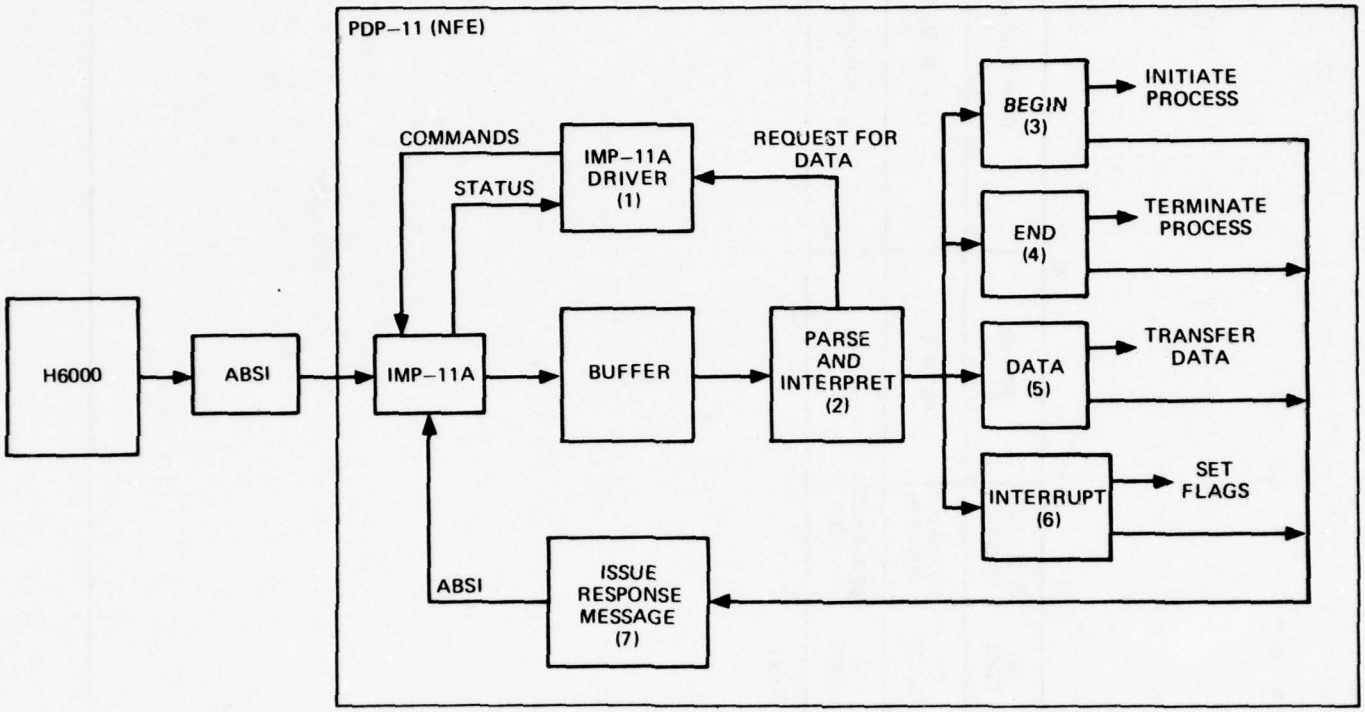


Figure 9-2. HFP Processes in NFE

4. Perform appropriate memory deallocation and process termination in response to an END command

5. Transfer the data to the selected process for further interpretation and transmission as required by a DATA command

6. Set appropriate flags or communicate with the process specified in an INTERRUPT command

7. Format the proper RESPONSE message and transmit it to the H6000.

The software which initiates messages from the NFE must perform the following functions as illustrated in Figure 9-3:

1. Format the appropriate messages according to the requirements of the higher level protocol software

2. Enter the messages into an output buffer

3. Transmit the buffer via the IMP-11A to the H6000

4. Listen for a response packet when appropriate.

The higher level protocol software translates instructions contained in the data messages into commands for the network programs such as NCP, TELNET, FTP, and RJE. This paper is unable to define these functions in any detail until the NFE operating system has been selected and the network programs have been well defined. At this point, only the general functions of this software may be reviewed:

1. Examine the data supplied by the lower level protocol interface to determine the desired function

2. Supply the proper data and commands to the appropriate module to accomplish the desired function. (e.g., a "logical" line of terminal data is passed to TELNET to establish a network connection)

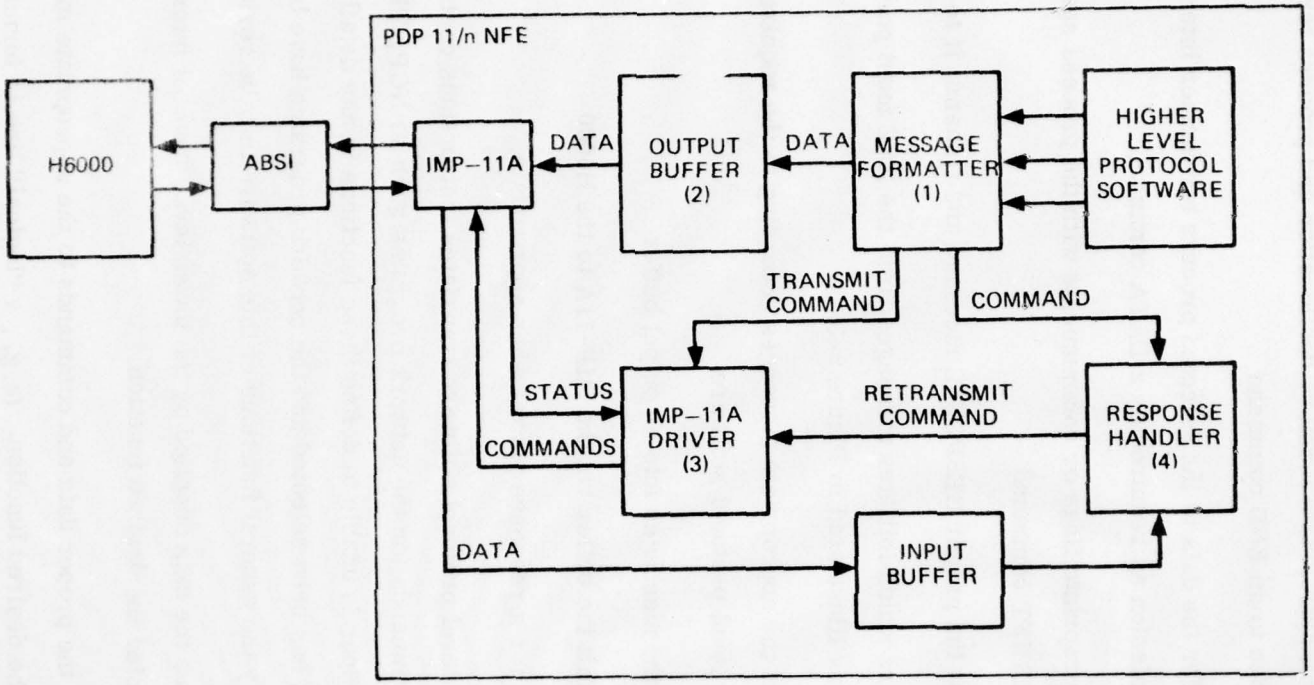


Figure 9-3. NFE to H6000 Interface Software

3. Perform the appropriate function required by network programs. This includes establishing or terminating connections with the H6000, or transmitting data messages to the H6000.

The GCOS interface software will require support from the operating system. A block-oriented device driver is required to control the IMP 11-A interface to the H6000. The operating system will be required to queue write commands to the IMP 11-A to transmit messages and responses to the H6000. It will also allow a task to issue a read, and then inform that task when the read is satisfied.

Multi-tasking is necessary to allow several connections between the NFE and H6000 to exist concurrently. It must also be possible for tasks to create and terminate other tasks, to implement commands such as BEGIN and END. Inter-task communication is necessary to enable the higher level protocol software to communicate with the Message Formatter and the DATA and INTERRUPT commands.

I/O timeout is a useful feature of the operating system. It is needed to determine if a message response from the H6000 takes longer than expected, indicating that exception processing is required.

The buffer associated with the IMP-11A driver can be a fixed size because the maximum transmission block size is fixed. Dynamic memory allocation is required by the higher level protocol software which receives and must store the text associated with a DATA command, before transmission to ARPANET. Because there is no limit to the number of DATA commands sent over a connection, buffers must be dynamically allocated to accommodate text.

SECTION 10 - OPERATIONAL CONSIDERATIONS

By its very nature commercial General Comprehensive Operating Supervisor (GCOS) encourages operational flexibility and control. The goal of this is to allow sufficient control and variation of an operational environment and enable maximum utilization of resources under variable load conditions. The design approach presented in this document allows a degree of flexibility in determining the operational environment for best handling varying network load requirements.

As currently designed, the majority of network required H6000 software is embodied in a single privileged slave program, which must be placed into execution before network connectivity is possible. Operationally, the execution of the network interface program may be operator controllable in much the same fashion as GCOS time-sharing (TSS) is controllable. Alternatively, a more rigid approach may be assumed where the Network Interface Program (NIP) may be roll-called (at system startup time) in the same manner as the file management system executive program (FSEX).

Since the design approach is modular, it is feasible to allow limited network functions via console operator verbs. Although not included in this design, it may be desirable to provide sufficient resources to allow either TELNET service, TELNET plus File Transfer Protocol (FTP), or TELNET, FTP, and Remote Job Entry (RJE). Although some modification to GCOS may be required to support these functions, the operational flexibility may be beneficial.

Another operational consideration implicit with the proposed design concerns the configuration of the network. Since the network is configured to appear to be a DATANET 355 (DN 355), terminal configuration is required at startup time. While this configuration is the actual hardware configuration for a DN 355 and affects space allocation for data structures within the H6000, the same configuration data will be required when adding a network front-end to the system. The terminal configuration data does not represent a physical limitation to the network, but does however represent a logical limit as to the number of concurrent terminal connections which can be

maintained (i.e., one terminal for each entry in the GCOS line table). This configuration data can have an impact on response and thus should be engineered carefully.

Another point we have raised in this paper is the possibility of RADC's interest in limiting or weighting the number of network users in favor of local H6000 users. There are a number of methods of establishing this throttling. An easy, interim solution might be to meter the network connections managed by the Network Control Program (NCP) in the NFE. Subsequent refinements might include NFE/H6000 dialogue where actual H6000 resource measurements are requested, and the results used as a barometer of network connection support.

SECTION 11 - SUMMARY

11.1 IMPLICATIONS OF H6000 SOFTWARE

This document provides the recommended design for the implementation of primarily new software to perform networking functions. The approach has been to modularly design new components rather than attempt to significantly modify the existing executive structure. Since the RADC site uses the "Commercial" version of GCOS, every effort has been made to minimize future release incompatibilities. The resultant design will reduce (although not eliminate) conflicts between GCOS maintenance and network software maintenance activities.

Obviously, major GCOS design changes will have impact on a network interface which is oriented to the GCOS structure of today. The converse is not necessarily so. As functions are added to or excised from the proposed network interface design, they will typically require no further GCOS modifications, although this is not categorically true.

The structure of this design is modular. Functional separation has been provided in that there is a basic GCOS interface module (the DN355 Simulator), a residual H6000 network program, a HFP and finally, a channel module to support the ABSI device. Even in what may appear to be a monolithic structure, the residual H6000 network program, there is a functional isolation between the component software responsibilities.

The addition of networking software is an important first step towards long range RADC goals. This interface will, however, increase system overhead due to the additional memory, control structure, and user loading. This overhead will predictably relate to a decrease in throughput for subsystems, when compared to the pre-interface period.

The size of core resident GCOS (called Hard Core Monitor) will increase; in addition, at least one system program which is always in execution will be added. Activities supporting remote users will be added to the existing job load.

For the tremendous benefits to be derived from extending GCOS facilities to the network, it is unfortunate that the network may be seen as the slowest device connected to the H6000. Processes performing network I/O will run longer than processes performing local I/O. Another obvious example may be seen in that a network file transfer will be slower than a local disk copy.

The requirement to extend the H6000 access assumes that there is no difference (to GCOS) between local and remote users. This, in turn means that the same attitude extended to local users must be implied for the remote user. We anticipate a non-malicious environment. Depending upon RADC's desires to "weight" the goal of satisfying local or net users, RADC may desire to implement certain throttling alternatives, possibly implemented in the NFE.

Finally, as complementary as the network interface scheme is to the existing GCOS architecture, it appears best to endorse a staged implementation. That might take the form of implementing the User or Server TELNET interface first, followed by increments of functions. In effect, this will create a 'try before fly' and allows the emerging user profile to shape subsequent detailed design.

11.2 IMPLICATIONS OF PDP-11 REQUIREMENTS

The GCOS connection to the ARPANET using a PDP-11 NFE has been designed around goals of (1) minimal impact to existing GCOS facilities, and (2) off-loading maximum network requirements to the front end.

An example of an implication to be found in the PDP-11 can be seen at the lowest level--the actual bit transfer from the H6000 to NFE. Here, the NFE must convert all ASCII data from (and to, if output to the H6000 is discussed) the H6000 nine bit format to the PDP-11's eight bit format. For the maximum block size discussed in this design, the transformations exceed 1,320 iterative excursions through the received data. While there are instructions to handle this transliteration, it is cumbersome at best, and a factor which may have a throughput impact. Preliminary timing estimates on a 144 bit Host to NFE exchange indicate a factor of approximately 130 microseconds to perform byte and bit conversions (e.g., - a 91 μs per bit overhead).

Other implications may be seen in that the large measure of ARPANET protocol negotiation, connection maintenance and flow control is to occur by software resident in the NFE. Precise definition of that software is not currently possible due to considerations raised earlier in this paper. Intuitively, it is obvious that the NFE must be a virtual memory, fast minicomputer. That a PDP-11/35 is an optimum selection is unclear. Residual issues such as the final hardware peripheral selections and the NFE Operating System will be of assistance. For example, if UNIX were to be selected as the NFE Operating System, that would almost certainly preclude the use of a PDP 11/35.

The local user support factor will necessitate an additional buffering requirement as well as some additional NFE software. The selection of different line interfaces to support a variety of local users will add new line driver requirements.

The size and complexity of this NFE, i.e., supporting the functions required of this GCOS interface, are much larger than any known existing NFEs. The closest parallel appears to be the MCA interface to the ARPANET for CDC host computers. The MCA effort is under development and cannot be used as a precise yardstick against which to measure this NFE requirement. The RSX-11 M NFE development effort by MCA certainly appears to be based on a more substantive configuration than the baseline hardware we have used as a model for this study.

While the core memory provided for this analysis seems to be quite adequate, there is a need for peripheral mass storage. An overlay support type of disk is indicated--something of the DEC RK05 variety at a minimum, and a DEC RP type being an optimum selection. Here again, the UNIX type of operating system is intensively disk-bound and fast disks are mandatory. Any tree structured file system will also be mass-store constrained. The peripheral mass storage will support program loads in addition to the above.

It is anticlimatic to state that the most intelligent approach to the NFE question is the classic one. Namely, define the need, define the functions, select the NFE hardware and executive scheme, and prepare a program design. Unfortun-

ately, this is not totally a Research and Development (R&D) effort. Due to the numerous NFE efforts, it is probably best to stick with the DEC family of 16 bit processors. Next, due to the current development and evolution of NFE operating systems, there are three alternatives: (1) pick an existing operating system and develop the necessary network software within the framework of the chosen executive structure, or (2) wait until the current crop of NFE executives have been thoroughly evaluated and select the best candidate, or (3) develop a new NFE to match the need.

Given the RADC project goal of providing (expeditious) user access to the many existing RADC software tools, alternative one, above, is recommended. It is realized that the NFE choice of today may not be the optimum choice six months from now, vis-a-vis the ELF; our rationale here is that the NFE is only the "vehicle" to gain access to the H6000 tools, it is not the focal point of the connection project.

11.3 CONCLUSIONS

There are four pertinent factors in building a summation to the GCOS connection design.

1. Does it meet "Program Goals"
2. How is the "GCOS Breach" accomplished?
3. Evaluation of "New H6000 Software"
4. NFE Support.

These factors are briefly discussed below.

11.3.1 Program Goals

The previous sections introduced and then specified how minimal H6000 software and NFE software will support the program goals for User and Server TELNET, a File Transfer capability, and a Remote Job facility. In addition, a section of the report addressed the functions and specifications necessary to support PDP-11 based local terminals.

The constraints imposed upon these goals included minimal impact on existing GCOS software, and the appearance to GCOS as if network (ARPA) originated requests were local requests.

The fact that all existing GCOS remote I/O is accomplished via MME GEROUT code was an early awareness point. H6000 programs such as TSS (the access to TSS by network users was a specific goal), and RJE are prime examples. In our design approach, the continued, unmodified use of MME GEROUT techniques is still afforded. The File Transfer capability to be implemented under this design will also use existing, unmodified GCOS supervisory calls. Any new networking software (HFP, ABSI, Channel Module, etc.) will also use current compatible GCOS technology.

To conclude, the network interface will be accomplished satisfying all program goals and totally within the constraints of the program sponsors.

11.3.2 The "GCOS Breach"

A GCOS breach was required at some selected point in order to recoup existing Remote Job and Server TELNET, latent within GCOS. The breach was finally selected in the IOS module of GCOS.

At the specified point in IOS (see Section 5), a simple compare/jump type sequence will be "patched" in to effect a transfer of control to appropriate network software. The approach chosen by the design team is to simulate a DN355 by the use of a PS355 software module. The DN355 simulation technique allows the non-impacting use of a substantive amount of GCOS I/O coding involving two major software programs, ROUT and DNET. Other alternatives were rejected because they involved modifications extending across these two modules.

The IOS breach location is considered optimum due to the amount of existing GCOS code which remains unmodified and useful, the ability to recover Server TELNET and RJE, and finally, the relative safety from future GCOS enhancement conflicts.

11.3.3 New H6000 Software

The new requirements for H6000 software fall into two categories; networking software and GCOS interface software.

The networking software, in all but one case, is residual requirement software necessary to accomplish virtual data transfers between the H6000 and the full networking functions found in the NFE. These residual requirements include Server and User TELNET and File Transfer. Individually, these are not expected to be large programs (each less than 3,000 lines of code) to develop. The largest networking software component is the HFP. This is a general purpose communications protocol which is independent of the ARPANET type of functions which it supports.

The GCOS interface software is comprised of the DN355 simulation program, discussed above, and an ABSI channel module. This channel module is a software program which addresses and utilizes the existing control structures (mailboxes, tables, etc.) in support of the physical H6000 to NFE communication.

As in most development experience, once a concept becomes reality, many of the lessons learned are after-the-fact. CSC's previous H6000 Host software development efforts lead us to fully endorse the Host software presented in this connection design.

11.3.4 NFE Support

Previously, this paper discussed the "resource" aspect of the NFE, i.e., to the extent the H6000 may be off-loaded of networking implications, the NFE becomes a resource. The most clear illustrations of NFE resource assignment include the NCP, protocol negotiations in support of file transfers, RJE, and local terminal support.

When a comfortable NFE hardware and software environment is selected, it is felt that new requirements, such as the NSW will be mainly supported by modularity and growth in the NFE. Additions to the preliminary file transfer design presented here may also be supported by NFE processing.

The probable growth of NFE services virtually mandates a local development capability. This thought has been explored in Sections 2 and 9 of this paper.

APPENDIX A - TERMINOLOGY FOR THE GCOS/ARPANET CONNECTION DESIGN

ABSI	-	Asynchronous Bit Serial Interface. Device designed at MIT Electronic Systems Laboratory to interface an H6000 host to an IMP.
ACK	-	Acknowledgement. Response used to signal successful reception of data or commands. ASCII code 006 (Octal).
AO	-	Abort Output. TELNET command which terminates output from a process.
ARPA	-	Advanced Research Projects Agency. A Department of Defense Agency.
ARPANET	-	ARPA network. Large network which supports heterogeneous computer communications.
ASCII	-	Amercian Standard Code for Information Interchange. A standard code used for communications.
AYT	-	Are You There. A TELNET command which provides the user with visible evidence that the system is operational.
BBN	-	Bolt, Baranek and Newman, Inc. A company which contributed to the design of ARPANET.
BCD	-	Binary Coded Decimal. Normally refers to the 6 bit binary representation of Hollerith characters within the H6000.
BELL	-	ASCII code that causes an audible sound at the receiving terminal, ASCII code 007 (Octal).
BMC	-	Bulk Media Conversion is a GCOS utility program which allows the independent printing or punching of files under control of imbeded media and report codes.
BOOTDECK	-	A deck of punched cards which is used to provide configuration and other data at system start-up.

BPS	-	Bits Per Second. Measure of the speed of a digital communications link.
Break	-	Special signal provided by a teletype normally used for synchronization.
BS	-	Back Space. A teletype type control function. ASCII code 010 (Octal).
CDC	-	Control Data Corporation. A computer manufacturer.
CIOC	-	Connect Input/Output Controller. An H6000 processor instruction used to initiate I/O operations. Even though Input/Output Controller hardware does not exist on an H6000 but has been replaced by Input/Output Multiplexor hardware, the mneumonic of this instruction has not been changed.
CONFIG	-	A section of H6000 bootdeck which defines the hardware configuration to the GCOS startup module.
CPS	-	Characters Per Second. Measure of the speed of a character-oriented digital communications link, or speed of a device such as a teletype.
CR	-	Carriage Return, ASCII code 015 (Octal).
.CRCT1	-	A global GCOS symbol which is used to refer to the Primary Configuration table. This table describes the hardware configuration and also contains information used by GCOS to control I/O operations.
.CRMB1	-	A global GCOS symbol which is used to refer to the Mailbox Table. This table is located at a preset address in main memory and is used as a communication area between the H6000 Processor and Input/Output Multiplexor hardware.

.CR101	-	A global GCOS symbol which refers to a table used by GCOS software to control I/O operations.
CRT	-	Cathode Ray Tube. Any interactive terminal which uses a CRT for a visual display rather than a printer mechanism.
CSC	-	Computer Sciences Corporation.
DAC	-	Direct Access. A general designation for any H6000 process capable of handling interactive terminals. Since a separate I/O mechanism is provided by GCOS to handle terminal operations, the distinction between DAC processes and other processes is easily made.
DATANET 355	-	A stand alone minicomputer which is used as a front-end processor for the H6000. All interactive terminals and any devices which can be connected via telephone lines are interfaced to H6000 through one or more DN355s.
DCA	-	Defense Communications Agency. An agency within Department of Defense responsible for global defense communications.
DEC	-	Digital Equipment Corporation, Maynard, Mass. Manufacturers of PDP-10 and PDP-11 computer systems. DEC is a trademark name.
DECwriter	-	Low-speed, hard copy terminal built by DEC.
DEL	-	Delete, ASCII code 377 (Octal).
DH11	-	Digital Equipment Corporation's programmable multiplexor used to connect a PDP-11 with up to 16 asynchronous serial communications lines.
DIA	-	Direct Interface Adapter.

DMA	-	Direct Memory Access. Type of peripheral interface that accesses memory without processor intervention. Primarily used for block transfers of data.
DJ 11	-	Digital Equipment Corporation's multiplexed interface between 16 asynchronous serial data communications channels and a PDP-11.
DL 11	-	Asyncrhonous interface between a single serial communications channel and a PDP-11.
DNET	-	A GCOS module which handles the majority of DN355 interface processing. DNET performs the H6000 side of H6000 to DN355 protocol.
DN355	-	DATANET 355.
DOS	-	Disk Operating System. A PDP-11 Operating System.
EC	-	Erase Character. TELNET command that erases the last typed character.
EDP	-	Electronic Data Processing.
EL	-	Erase Line. TELNET command to erase current line.
ELF	-	Real-time Operating System designed to run on a PDP-11 and support ARPANET interface programs.
ELFII	-	Later version of ELF.
EOF	-	End of File.
EOR	-	End of Record.
FE	-	Front-end. See NFE.
FF	-	Form Feed, ASCII code 014 (Octal).

FIP	-	Front-End Interface Program. An H6000 resident privileged slave program which translates network protocol into GCOS compatible functions.
FM	-	Foreman. The foreman is a local software management function found at host computer sites which are performing as Tool Bearing Hosts of the National Software Works.
FP	-	File Package. The File Package is a local software support function found at host computer sites of NSW participants. The FP supports user file activities.
FTP	-	File Transfer Protocol. A protocol on ARPANET hosts that permits files to be transferred between hosts.
GA	-	Go Ahead. A TELNET command to allow a terminal to transmit characters.
GCOS	-	General Comprehensive Operating Supervisor. A Honeywell operating system used on H6000 family computers. GCOS is a batch operating system which also supports interactive terminal access.
GEIN	-	H6000 GCOS program which provides the batch input function for job streams to be executed under GCOS control.
GEOT	-	GCOS module which reads the SYSOUT file and disperses job output to appropriate media (i.e., printer or punch).
HCM	-	Hard Core Monitor. That portion of GCOS which is loaded into core during system startup and resides there for the life of the system. Due to extended memory hardware restrictions, all GCOS tables and HCM must reside in the first 64K words of memory. Hard Core Module, a GCOS module which is part of the Hard Core Monitor.

HFP	-	Host Front-End Protocol. Communications procedures between the Host (H6000) and Front-end (PDP-11).
HI	-	Host-to-IMP. A software segment of the Network Control Program which is responsible for interface to the ARPA Network. The HI is located in the Network Front-End in the RADC GCOS connection project.
Host	-	A computer attached to ARPANET which provides services to local and remote users.
HT	-	Horizontal Tab, ASCII code 011 (Octal).
H600	-	The predecessor series to H6000. The H6000 instruction set includes all H600 instructions to provide forward compatibility for H600 software. Backward compatibility is not as simple since H6000 processors support additional instructions not available on H600 processors.
H6000	-	A family of 36-bit computers built by Honeywell.
H6180	-	The fastest processor model in the H6000 family. It is equivalent to an H6080 processor but contains additional hardware to support virtual memory operation. GCOS will run on a H6180 but does not use any virtual memory capabilities.
ID	-	Identifier. Depending on context, ID may be a user or program parameter.
IMP	-	Interface Message Processor. Packet switching computer which provides the basic data transmission capability for the ARPANET.
IMP11-A	-	Interface between PDP-11 and an IMP. Also the IMP11-A will be used to connect a PDP-11 used as an NFE to an H6000.

IMW - Interrupt Multiplexor Word. The H6000 hardware provides up to 32 vectored interrupts. There are eight different types of interrupts for each of four possible IOMs. When an interrupt occurs, a bit flag is set in the corresponding IMW to indicate which channel initiated the interrupt. If several channels request the same type of interrupt within a very short time period, multiple bits will be set in the corresponding IMW and only one interrupt taken.

INIT - A GCOS module which provides system startup processing. INIT is bootloaded into a "dead" machine and causes GCOS to be loaded and initialized to begin normal operation.

I/O - Input/Output. A hardware operation which causes data movement between high speed main memory and slower peripheral devices such as tapes, disks, etc.

IOM - Input/Output Multiplexor. The H6000 hardware architecture allows up to four of these special I/O processors which handle all H6000 I/O operations. This allows I/O operations to be performed concurrently with program processing and a minimum interference between processors and I/O.

IOS Input/Output Supervisor. The GCOS module which controls all I/O operations within the multiprocessing, multiprogramming environment provided by GCOS.

IOTRM - A routine within IOS which handles interrupts. Normally these are Terminate Interrupts signaling the completion of an I/O operation.

IP - Interrupt Process. TELNET command to immediately halt an executing process in a host computer.

K	-	Unit of 1000 or 1024, depending on context. When referring to words of storage, K equals 1024. When referring to bits per second, K equals 1000.
KB	-	Kilo bits. On thousand bits. KB is a measure of data transmission speed.
KW	-	Kilo words. Unit of 1024 computer words.
LF	-	Line Feed, ASCII code 012 (Octal).
LPM	-	Lines Per Minute. Measure of line printer speed.
MACRO	-	Assembly language symbol which can be directly replaced with expanded assembly language statements. Also DEC's name for the DOS Assembly language.
mailbox	-	A GCOS data structure which provides input/output control between the H6000 and the input/output multiplexor (IOM).
MCA	-	Massachusetts Computer Associates, Inc., Wakefield, Mass. Private company currently engaged in interfacing a network of CDC omputers to the ARPANET using a PDP-11 based RSX-11M operating system. MCA is also engaged in the development of the National Software Works
MCR	-	Monitor Console Routine. A process in RSX-11D and RSX-11M that interacts with the console.
MME	-	Master Mode Entry. An H6000 processor instruction which unconditionally generates a processor fault. Processes use this fault to pass control to GCOS when requesting GCOS perform some service for them. Usually associated with each MME is a symbolic code which is used to specify the type of service being requested.

MME GEFSYE - A master mode entry used by processes to request an I/O operation be performed. MME GEINOS are not used for I/O operations involving equipment connected through a DN355.

MME GEROUT - A master mode entry used by processes to request an I/O operation with equipment connected through a DN355, primarily interactive terminals.

(M) ROUT - The GCOS module which processes MME GEROUT requests. ROUT verifies a process request and interfaces with DNET to initiate the requested operation.

MSG - A component of NSW which is implemented at each NSW Tool Bearing Host. MSG provides flexible interprocess communication between NSW processes.

MULTICS - MULTiplexed Information and Computing Service. A Honeywell operating system which requires the virtual memory hardware of an H6180.

NAK - Negative Acknowledgement response used to signal erroneous reception of data or commands. ASCII code 025 (Octal).

NCP - Net Control Program. A software program used by hosts. NCP includes Host-IMP Protocol and Host-Host Protocol.

NFE - Network Front-End. A computer which interfaces the ARPANET on behalf of a host computer. In particular, a PDP-11 interfaced between an H6000 and the ARPANET.

NIC - Network Information Center. NIC provides and maintains information about ARPANET and ARPANET protocols.

NPR - Non-Processor Request. DEC's term for DMA.

NRL - Naval Research Laboratory. Engaged in connecting PDP-11's running ELFII operating system with ARPANET.

NSW	-	National Software Works. Nationwide computer utility to facilitate software development. NSW is essentially a distributed network operating system using ARPANET for communication.
NULL	-	ASCII code 000 (Octal), signifying no operation.
NVT	-	Network Virtual Terminal. A virtual terminal concept used for terminal to process and some process to process communications.
ODT	-	Online Debugging Technique. A debugging program for PDP-11.
Op Code	-	Operation Code. Binary code that determines the operation indicated by a command or instruction.
OUT	-	Command to GCOS to perform output.
OUT/IN	-	Command to GCOS to perform output followed by input.
PASSWORD	-	Secret character string unique to a particular computer user that allows the user to access a system and specified privileges.
PATHID	-	Path Identifier. Bit or character string that uniquely identifies a particular path or connection.
PAT	-	Peripheral Assignment Table. The PAT is a GCOS table which contains device relevant information and status.
PDP-10	-	32-bit medium-size computer manufactured by DEC.
PDP-11	-	16-bit minicomputer manufactured by DEC. It will be used as an NFE between an H6000 and the ARPANET.
PDP-11/35	-	A model of PDP-11.

PS355		Pseudo-Soft DN355. The PS355 is a name devised to describe the functional requirement of an H6000 software program. Namely, simulating the DN355 protocol in the H6000.
RADC	-	Rome Air Development Center, Griffiss AFB, New York.
RFC	-	Request For Comments. Technical notes published by ARPANET Network Working Group or other users soliciting comments on protocols or designs.
RGIN	-	GCOS module which receives remote job input streams and spools them to disk for presentation to GEIN.
RJE	-	Remote Job Entry. ARPANET protocol allowing a user to submit batch jobs to be executed on a remote host.
RK05	-	Cartridge disk drive and control.
ROUT	-	The GCOS module supporting Remote Input and Output processes.
RP	-	Magnetic disk pack drive and control.
RSX-11	-	Family of PDP-11 real-time operating systems supplied by DEC.
RSX-11D	-	PDP-11 real-time multi-user operating system.
RSX-11M	-	PDP-11 real-time, multi-tasking operating system supplied by DEC.
RS232	-	Terminal Interface.
RTP	-	Reliable Transmission Package. This is a software protocol used to support host to IMP communications over very distant lines (i.e., greater than 2,000 ft.). If required in the GCOS connection project, RTP would be implemented in the NFE.

SCRL	-	Speech Communications Research Laboratory, California.
SERVER	-	A *software protocol* supplying a service to another host via the ARPANET.
SMIC	-	Set Memory Controller Interrupt Cells. An H6000 processor instruction which causes an interrupt to occur. GCOS does not normally use this instruction since interrupts are expected from the IOMs rather than processors.
Socket	-	Software Input/Output port belonging to a process in an ARPANET host.
SOW	-	Statement of Work.
SSA	-	Slave Service Area. An area of memory immediately preceeding each program running under GCOS. It is outside of the program's address space. GCOS uses this area to store control information about the program. GCOS also pages SSA modules into this area to provide various services to the program.
SYNCH	-	Signals passed between TELNET hosts to resynchronize their processes.
SYSOUT	-	Generic term referring to the collection and dispersal of *output under control* of the GCOS operating system.
TBH	-	Tool Bearing Host. Computer connected to NSW containing software tools (programs) which can be used by remote users.
TELNET	-	Teletype Network. ARPANET communications facility allowing a terminal at one host to act as if it were connected to another host.

TENEX	-	A Digital Equipment Corporation PDP-10 Time Sharing executive.
TIP	-	Terminal Interface Processor.
TRAX	-	Transaction Processor. A privileged DAC program provided by Honeywell. It receives transaction requests from many terminals and funnels the requests through an appropriate processing module known as a TPAP.
TSS	-	Time Sharing System. A privileged DAC program which is provided by Honeywell. TSS divides processor and core resources among numerous interactive terminal users. It allows the remainder of GCOS to remain a batch operating system while providing interactive facilities. The facilities allow the entry and editing of text files and an environment for compiling and executing user programs interactively in several languages.
T.TAB	-	A GCOS data structure which contains control information as a function of active terminals. It contains information such as terminal ident, terminal type, and line status and is used to control data flow between the H6000 and the DATANET 355 on a per terminal basis.
UNIX	-	A PDP-11 timesharing operating system developed by Bell Laboratories.
USER	-	A software protocol which acts on behalf of a user requesting a service supplied by a host on the ARPANET.
USERID	-	User Identification. Means of a user providing identification to a computer.
VIDEO	-	A privileged DAC program provided by Honeywell. It accumulates information concerning system activity,

		formats it, and prints it on Honeywell CRT terminals known as Visual Information Projection Systems (VIPS).
VT	-	Vertical Tab, ASCII code 013 (Octal).
WM	-	Work Manager. The Works Manager is a software management function of the National Software Works. WM controls user access, file transfer and software tool execution in NSW host computers.
WMO	-	Works Manager Operator. A subset of the functions provided by the Works Manager.
WWMCCS	-	World Wide Military Command and Control System.

APPENDIX B - BIBLIOGRAPHY

IN SUPPORT OF AN NSW-COMPATIBLE GCOS/ARPANET
CONNECTION DESIGN

ARPANET Protocol Handbook, NIC 7104, Elizabeth Feinler and John Postel, Network Information Center, Stanford Research Institute, Menlo Park, CA 94025, April 1976

Asynchronous Bit Serial Interface (ABSI) for Connecting the Honeywell 6180 to an ARPA IMP, John E. Ward, Electronic Systems Laboratory, MIT, March 3, 1975

Campus Computing Network - An IP Server for NSW, R.T. Braden and H. C. Ludlam, UCLA, April 1, 1976

Communication with the PDP-11/B4700 (Chapter 4)

ELF Debugger User's Guide, B. W. Shafer, D. L. Retz, J. R. Miller, J. L. McClurg, SCRL, Inc., Sept. 30, 1975

ELF File System Manual, B.W. Schaffer, D.L. Retz, J.R. Miller, J.L. McClurg, SCRL, Jan. 12, 1976

ELF II TELNET Program Listing, Feb. 20, 1975

ELF Kernal Modules, Feb. 21, 1975

ELF Kernal Programmer's Guide, D. L. Retz, J. R. Miller, J. L. McClurg, B. W. Schafer, SCRL, Jan. 1, 1976

ELF NCP Programmer's Guide, D. L. Retz, Stanford Research Institute, Menlo Park, CA, Nov. 1, 1975

ELF Program Listings, May, 1976

ELF System Documentation, July 12, 1974

ELF User's Guide, B. W. Schafer, D. L. Retz, J. R. Miller, J. L. McClurg, SCRL, Inc., Jan. 1, 1975

End-to-End Protocol, INWG General Note #95, V. Cerf, A. McKenzie, R. Scantlebury, H. Zimmerman, Sept. 4, 1975

Establishing a Connection, INWG Protocol Note # 14, Yogen K. Dalal, Digital Systems Laboratory, Stanford, CA, March 1975

The Evolution of Message Processing Techniques in the ARPA Network, J. M. McQuillan, BBN, Infotech State of the Art Report, Network Systems and Software, 1975

The F Format, Supplement to: Experiment in Inter-Networking Basic Message Format, IFIP WG6.1 Protocol Committee Document 2, Vint Corf, Feb. 10, 1975

File Transfer Protocol, ARPA NIC # 17759, (RFC # 542, 354, 454, 459, NIC # 7104), N. Neigus, BBN-NET, July 12, 1973

The Foreman: Providing the Program Execution Environment for the National Software Works, BBN Report No. 3266, MCA Document No. CA00-7604-0111, Richard E. Schrantz, BBN, and Robert E. Millstein, MCA, Preliminary, March 31, 1976

Front End Protocol, G. W. Bailey, K. McCloghrie, Draft, June 6, 1975

Front End Protocol, July 28, 1976

A Front-End Strategy for Attaching Host Computers to the WWMCCS Inter-Computer Network, MTR-5245, Michael A. Padlipsky, June 1975

Function-Oriented Protocols for the ARPA Computer Network, SJCC AFIPS Proceedings, Stephen D. Crocker, ARPA, John F. Heafner, RAND, Robert M. Metcalf, MIT, Johnathan B. Postel, UCLA, 1972

A High Throughput Packet-Switched Network Technique Without Message Reassembly, Roy Daniel Rosner, Raymond H. Bittel, Donald E. Brown (DCA), IEEE Transactions on Communications, August 1975, pages 819-828

Honeywell GCOS Software (Series 60 (Level 66) /6000), Honeywell Information Systems, April 1974

Host-Front End Protocol (Draft), G. R. Grossman, M. A. Padlipsky, D. M. Grothe, J. D. Day

Host/Host Protocol for the ARPA Network, ARPA NIC #8246, Alex McKenzie, BBN, Jan. 1972

IMP 11-A PDP-11 Host to IMP Full Duplex NPR Interface, Digital Equipment Corp., Maynard, Mass., May 1975

Interface Message Processors for the ARPA Computer Network, BBN Quarterly Technical Report #1 (Report 3063), Jan 1, 1975 to March 31, 1975

Interface Message Processor - Specifications for the Interconnection of a Host and an IMP, BBN 1822, Jan., 1976

Introduction to RSX-11M, Digital Equipment Corp., May, 1974

An Introduction to XOFF, A Document Generation System, Joseph Newcomer, Carnegie-Mellon Univ.

MAILSYS, Jan 6, 1975

MCA Semi-Annual Technical Report CADD-7603-0411, March 4, 1976

MCA to CSC Correspondence, including MCA internal memos relating to MCA's analysis of ELF II, UNIX, and RSX-11M operating systems; and also MCA plans to interface RSX-11M to ARPANET

More Joy of TENEX - Some of the Refinements, compiled by NOLG at ISIB, Nov., 1975

MSG Manual, John Vittal, USC-ISI

The National Software Works: A New Method for Providing Software Development Tools Using the ARPANET, Stephan D. Crocker, USC/Information Sciences Institute, June 16-19, 1975

NSW GCOS Connection, RADC-TR-76-228, Interim Report, Computer Sciences Corp., July, 1976

NSW/GCOS Connection Project Task #4423 Project Review (CSC), Contract # F 30602-76-C-0199, Oct., 1976

National Software Works Protocols, Jonathan B. Postal, Augmentation Research Center, Stanford Research Institute, Menlo Park, CA 94025

National Software Works Second Semi-Annual Report, CA00-7608-1611, Massachusetts Computer Associates, Inc., Wakefield, Mass., Aug. 16, 1976

NSW GCOS Connection Technical Proposal, CSC, in response to RFP No. F 30602-76-R-0110, Nov. 11, 1975

Network Front End Development Plan, Sept. 1976

The Network UNIX System, Gregory L. Chesson, University of Ill., Urbana, Ill. 61801

NLS-8 Primer, SRI-ARC, #32954, July 11, 1975

A Note on Some Unresolved Issues for an End-to-End Protocol, INWG General Note 98, John Day, Aug. 1975

An Overview of the ELF Operating System, David L. Retz, Speech Communication Research Lab., Inc., Santa Barbara, CA 93109, May 6, 1975

PDP-11 Peripherals Handbook, Digital Equipment Corp., 1973

Preliminary Report on Attachment Strategies for a Prototype Front End, R. A. Mikelkas, M. A. Padlipsky, R. K. Trehan, The Mitre Corp., Washington Operations, Nov. 26, 1975

The Procedure Call Protocol, Version 2, James E. White, Augumentation Research Center, Stanford Research Institute, Menlo Park, CA 94025, Jan. 1, 1975

Proposal for an Internetwork End-to-End Protocol, IFIP WG6.1, INWG General Note 95,96, V. Cerf, A. McKenzie, R. Scantlebury, H. Zimmerman, July 1975

Prototype WWMCCS. Intercomputer Network Functional Description - Alternative Network Interface Architectures prepared for Joint Technical Support Agency, DCA, Contract DCA100-72-C-0035, Report R417700014-1-1, Dec. 6, 1974

Prototype WWMCCS Intercomputer Network - Network Front End Design Analysis Host Software Conversion Aspects, Command and Control Technical Center, DCA, Washington, D.C., Contract DCA100-75-C-0029, Report R493700041-1-1, March 29, 1976

Remote Job Entry Protocol, ARPA RFC #407, NIC #12112, Bob Bressler (MIT), Rick Guida (MIT), Alex McKenzie (BBN-NGT)

A Resource Sharing Executive for the ARPANET, Robert H. Thomas, BBN, Reprint - Proceeding NCC, 1973 AFIPS

Revised Statement of Work for NSW GCOS Connection, PR No. B-6-3223, Rome Air Development Center, Grifiss AFB, N.Y., Sept. 18, 1975

Revised TELNET Status Option, ARPA RFL #651, D. Crocker (UCLA-NXC), Oct. 25, 1974

RSX-11M Executive Reference Manual, Digital Equipment Corp., 1974

RSX-11M Operator's Procedures Manual, Digital Equipment Corp., 1974

Study and Analysis of the Technical Problems, Alternatives, and Approaches for Developing a Network Operating System (NOS) for Heterogeneous Distributed Computer Networks, Statement of Qualifications, CSC, for RADC, April 1976

TELNET Approximate Message Size Negotiation Option, ARPA NIC #15393

TELNET Binary Transmission Option, ARPA NIC # 15389

TELNET Echo Option, ARPA NIC # 15390

TELNET Extended Options, List Option, ARPA NIC #16239

TELNET Option Specifications, ARPA NIC # 18640, Aug. 1973

TELNET Output Carriage-Return Disposition Option, ARPA RFC # 652, D. Crocker, UCLA-NXC, Oct. 25, 1974

TELNET Output Form Feed Disposition Option, ARPA RFC # 855, NIC # 31158, D. Crocker, Oct. 25, 1974

TELNET Output Horizontal Tab Disposition Option, ARPA RFC # 654, D. Crocker, Oct. 25, 1974

TELNET Output Horizontal Tabstops Option, ARPA RFC # 653, D. Crocker, UCLA, Oct 25, 1974

TELNET Output Line-Feed Disposition, ARPA RFC # 658, NIC # 31161, D. Crocker, Oct. 25, 1974

TELNET Timing Mark Option, ARPA NIC # 16238

TELNET Output Vertical Tab Disposition Option, ARPA RFC # 659, D. Crocker, Oct. 25, 1974

TELNET Protocol Specification, ARPA NIC # 18639, August 1973

TELNET Reconnection Option

TELNET Suppress Go Ahead Option, ARPA NIC # 15392

TELNET User Guide

Terminal Interface Message Processor User's Guide, Report No. 2183, Bolt, Berqnek, and Newman, Inc., for ARPA, Aug. 1975

The UNIX I/O System, Dennis M. Ritchie, Bell Telephone Laboratories

The UNIX Time Sharing System, Dennis M. Ritchie, Ken Thompson, Bell Laboratories, Murray Hill, N.J. 07974, CACM, Vol. 17, No 7 (1974), pp. 365-375

XED - Experimental Editor, Short Reference Manual, Ronald Tugender, Donald Oestreicher, Sept. 25, 1975

APPENDIX C - ASSUMPTIONS SUPPORTING THE SELECTED INTERFACE APPROACH

APPENDIX C - ASSUMPTIONS SUPPORTING THE SELECTED INTERFACE APPROACH

C.1 INTRODUCTION

There are any number of ways of fabricating a modification and set of enhancements to an existing software system. The emphasis, direction, and interpretation of basic project guidelines have formed the basic constraints upon which to measure the effectiveness of this interface schema.

These constraints and goals are discussed below and form the basis of assumptions in presenting this design.

C.2 GENERAL DESIGN PARAMETERS

Certain phrases have appeared throughout the document. The "off-loading of the H6000" is a prime example. Off-loading the H6000 of all possible networking implications is inherent in this design. Given luxuries such as unrestricted access to GCOS modifications and non-concern for supporting the existing user profile, it is highly likely that the interface design would be quite different.

Another satisfied goal is found in that there are to be no modifications to existing GCOS facilities, such as TSS, to support the new network user. Further, the interfaces between new software and GCOS have been kept to a minimum. The interim report identified a different interface approach as probable. It was subsequently discarded because of the amount of interfaces into the existing (and future) GCOS structure.

The use of the NFE as a "resource" has been frequently indicated by the project sponsor, and is reflected throughout this document. The point to stress here is that the off-loading of the H6000 to the NFE has tended to tailor some of the NFE towards the H6000. So, a basic assumption rests in the premise that the NFE is <u>not</u> to be picked up in-toto from some other effort, and used virtually unmodified.

Last, the current effort is an interim design, albeit one which is moulded by long range goals. As an interim solution, it is expected that user needs and subsequent analysis will shape and add non-apparent capabilities. For example, there is little mention herein of debugging, instrumentation, or accounting. Yet, experience or even a casual reading of NSW capabilities indicate these as probable.

C.2 SPONTANEOUS DESIGN PARAMETERS

Through the period of analysis and design, certain design criteria develop, seemingly in a spontaneous manner. These are evidenced by the enthusiasm for a project which offers a "light at the end of the tunnel" for participants. Certain of these considerations are presented here as extensions beyond explicit guidance, yet as factors which have influenced this design document.

First, a phased implementation is anticipated. It is probably in the government's interest to sequence a GCOS to ARPANET interconnect implementation. The first step would be the decision as to the functions to be activated first. Next, detailed programming specifications would be developed. Upon approval of these, actual development of the selected functions would begin. The sequence of developed software would include all dedicated machine environment activities first. These include drivers, hardcore, and operating system segments. It is clear that the accomplishment of all of this category of software first will require a concentrated amount of dedicated machine time. However, once this is accomplished, all subsequent development and testing could be performed without a significant impact on existing users. The ratio of time spent in the dedicated machine environment is anticipated to be 25 percent versus 75 percent for shared development.

Another spontaneous concept is the development of a user baseline profile. The various user agencies that would benefit from this connection of GCOS to the ARPANET could also form the nucleus of contributors to subsequent design and implementation activities. The use of selected addressee RFCs (Request for Comments sent over the ARPANET) could form the basis for adding additional features to this initial design document.

Another point regards the PDP-11 NFE. As is clear from activities supporting the PDP-11 analysis and design, there is no imminent system recommendation which contains all needed functions. Development is indicated no matter which artiface is selected. This results in an attitude to select the closest match to the needs expressed in this document. The NFE is a means to an end - it is not an objective.

Finally, the issue of NSW compatibility. The NSW effort is one which constrained early design efforts due to a precise definition of how GCOS would assume a TBH role. Because many of the user and NSW controlling software concepts are emerging even now, we began to address the issue of NSW compatibility being satisfied in either of two manners. Either the FE would assume significant new NSW functional responsibilities or a second, post ARPANET implementation, design iteration such as this document would detail the new development/modification requirements. The design related herein does not contravene subsequent NSW development for the H6000.

C.3 DIFFERENCES IN COMMERCIAL WORLD WIDE MILITARY COMMAND AND CONTROL SYSTEM (WWMCCS) GCOS

A point related to our design approach is found in the differences in the commercial version of GCOS as opposed to the Government's World Wide Military Command and Control System (WWMCCS) version of GCOS. In the WWMCCS' GCOS, the largest functional difference is in the area of security.

This functional difference is manifested in a significant amount of coding distributed between the H6000 and the DN355 software structures. The areas impacted by the WWMCCS security additions include:

1. The addition of security levels for all GCOS processing. This includes files, file accesses, terminals, jobs, job outputs, and users
2. Terminal log-on validation

3. Modified DN355/H6000 protocol to support terminal log-on activities

4. Operator line, terminal, and DN355 control processing.

The internal and external effects of security implemented for the WWMCCS GCOS is seen in the User Security Matrix (USM), Terminal Security Matrix (TSM), Security Classification Code (SCC), and Caveats.

The USM and TSM binary fields represent maximum security levels and access permissions for specific users and terminals. The SCC is a 3 character code which may be transliterated into the 23 bit form used for the USM and TSM representation. Caveats are text fields used at the top and bottom of print pages to denote the security level. The processing significance for these security parameters is distributed throughout the WWMCCS GCOS system. These security provisions and others (e.g., clearing of mass store during file allocation) clearly differentiate the two GCOS executives.

As is obvious from the preceding, there is a wide gulf between the two operating system versions - albeit there may be a long range goal for reconciliation between them. If we were considering the ARPANET interface design using a military GCOS version, the proposed scheme may have been somewhat different. There is a networking interface to WWMCCS GCOS found in the Prototype WWMCCS Intercomputer Network (PWIN). Also, there is a PWIN-based R&D effort to build an H6000/PDP-11 ARPA Network interface.

Considering the urgency of RADC's need to implement a network interface, and the fact that RADC uses the commercial GCOS, the design proposed in this document is an optimum solution.

If a WWMCCS version of GCOS were to be installed at RADC, the same networking scheme proposed here could be implemented. However, a subsequent study would be necessary to identify any remote user or H6000 software impacts.

APPENDIX D - PDP-11 OPERATING SYSTEM TRADEOFF ANALYSIS

APPENDIX D - PDP-11 OPERATING SYSTEM TRADEOFF ANALYSIS

There are three operating systems being considered as the NFE Executive:

1. ELF II
2. UNIX
3. RSX-11M.

ELF II is a special-purpose operating system written by the Speech Communications Research Laboratory at Santa Barbara, California. Its primary function is to provide a multi-user interface to ARPANET. UNIX is a time-sharing operating system written at Bell Laboratories, Murray Hill, New Jersey. It has been interfaced to ARPANET at the University of Illinois, Urbana, Illinois.

RSX-11M is a real-time operating system written by the manufacturer of the PDP-11, Digital Equipment Corp., (DEC) Maynard, Massachusetts. It is similar to RSX-11D. RSX-11M is currently being interfaced to the ARPANET by Massachusetts Computer Associates (MCA), Wakefield, Massachusetts.

These three operating systems will be evaluated using criteria developed in Paragraph 9.1, NFE Executive Requirements. Table D-1 summarizes the requirements and indicates how well the three operating systems meet each requirement.

D.1 PROCESSOR MANAGEMENT

All three operating systems support multi-tasking. They all use a real-time clock, and include mechanisms for inter-task communications. UNIX and RSX-11M allow priority to be assigned to a user's tasks; ELF II does not.

ELF II and UNIX are set up for multi-user operation which includes Log-in/Log-out commands, passwords, and accounting. RSX-11M is not designed for multi-user operation, although it has limited multi-user capability. Although MCA is incorporating many of the following capabilities into RSX-11M, the standard operating system does not have Log-in/Log-out commands, passwords nor accounting.

Users at different 11/M terminals cannot simultaneously use the same systems programs such as the editor, assembler, or Monitor Console Routine (MCR). RSX-11D does fully support multi-user programming.

All three operating systems allow user access to the time of day, and all allow the user to modify the time of day. In all three operating systems, the user can get the status of the system and tasks from a terminal. Appropriately privileged tasks also have access to the system and task status.

Some sort of debug facility is common to all three operating systems. For example, RSX-11M has minimal debugging facilities which allow the user to examine and modify any location from the terminal, and also supports DEC's On-line Debugging Technique (ODT). ELF has incorporated limited debugging facilities, which allow the user to set breakpoints and examine and modify registers and storage.

All three operating systems handle error traps which occur on an illegal odd address word FETCH, an access to nonexistent memory, and an attempt to execute a nonexistent instruction. RSX-11M has the capability to store the volatile registers upon power failure, and to recover when the power is restored. UNIX does not have this feature.

D.2 STORAGE MANAGEMENT

All three operating systems support memory management, which allows the PDP-11 to access up to 124K words of core, but the ELF II version which supports memory management does not appear to be fully debugged. All three operating systems include a disk file system. RSX-11M and UNIX both have an on-line directoried file management system with file protection. ELF II's file system is much more primitive than the others; it does not include directories, disk formatting, nor any file protection features. Because of the file system, it is possible to do program development on both UNIX and RSX-11M, but not on ELF.

On the other hand, ELF II has dynamic memory allocation. RSX-11M has limited dynamic storage allocation used by executive calls, but this is useable only by privileged user tasks.

D.3 INPUT/OUTPUT MANAGEMENT

ELF, UNIX, and RSX-11 have similar I/O support. They all queue I/O requests and support character and block read and write functions, allow devices to be attached to a user, job, or task, and provide a mechanism to abort an I/O operation.

Only ELF and RSX-11M permit non-blocking, or asynchronous, I/O operations. In UNIX, when a process requests an I/O operation, it is blocked from executing until that I/O operation is completed. UNIX and RSX-11M provide a time-out associated with an I/O operation. ELF and RSX-11M provide the user with the ability to redirect I/O requests from one device to another.

All three operating systems provide most of the device drivers needed for an NFE. RSX-11M, being manufacturer supplied and supported, has drivers for all standard DEC devices, and drivers will be supplied as new devices become available. ELF is weakest in its variety of device drivers. For example, it lacks device drivers for the larger disks. Nevertheless, all three operating systems have the drivers required for a NFE, and they all have the facility for adding new drivers to the system.

D.4 GENERAL REQUIREMENTS

There are several other criteria beside functional requirements that are used to evaluate operating systems. These include bootstrap capabilities, core usage, execution overhead, modularity and ability to be modified, reliability, and maintenance support.

All three operating systems may be booted via a down-line load from another computer or a network. ELF currently exists in a form that can only be booted via a down-line load; however, it could be converted to a disk-bootable medium. Both RSX-11M and UNIX may be booted from a disk.

The amount of core required by each operating system is listed in Table D-1. The numbers given refer to the core required by the operating system only, exclusive of network or applications programs. In ELF, the virtual memory (memory management) version is used. As can be seen, RSX-11M uses the least core and UNIX requires the most core. The actual amount of core required by each operating system will vary depending on which drivers and optional functions are included.

Exact figures for the amount of execution overhead required by each operating system are not available. It is probable that UNIX requires the most overhead because of its extremely general purpose nature, while ELF requires the least overhead in a network application, because it was designed specifically to interface with ARPANET.

UNIX is the easiest operating system to modify because it is written in a high level language (C) and has a highly modular design. RSX-11M also features modular design, and is relatively easy to modify. ELF is the most difficult to modify.

RSX-11M and UNIX have both been used in the field for some period and are essentially error free. UNIX has a reputation of being extremely reliable. ELF, on the other hand, is still being developed, the virtual memory version cannot currently be considered reliable, and the file system is also not appropriate to the goals of the NFE.

RSX-11M has full maintenance support from DEC, as it is one of DEC's standard operating systems. Neither UNIX not ELF were developed by commercial vendors, but rather by research groups as in-house tools; consequently, they do not have formal maintenance support.

D.5 SUMMARY

All three of the operating systems under consideration, ELF, UNIX, and RSX-11M, have most of the features required for the NFE executive. The following paragraphs summarize the advantages and disadvantages of each of the operating systems.

TABLE D-1. COMPARISON OF OPERATING SYSTEMS

	ELF II	UNIX	RSX-11M
Multi-Tasking	Yes	Yes	Yes
Real-Time Clock	Yes	Yes	Yes
Inter-Task Comm.	Yes	Yes	Yes
Task Priority	Yes	Yes	Yes
Multi-User	Yes	Yes	No [1]
Login/Logout	Yes	Yes	No [1]
Password	Yes	Yes	No [1]
Accounting	Yes	Yes	No [1]
Get/Set Time of Day	Yes	Yes	Yes
System Status	Yes	Yes	Yes
Debugging	Yes-limited	Yes	Yes
Error Traps	Yes	Yes	Yes
Power Failure/Restart	Yes	No	Yes
Memory Management	Yes	Yes	Yes
Disk File System	Yes-limited	Yes	Yes
Directory	No	Yes	Yes
File Protection	No	Yes	Yes
Create/Delete File	No	Yes	Yes
Program Development	No	Yes	Yes
Dynamic Memory Allocation	Yes	No	No
Queue I/O Requests	Yes	Yes	Yes
Character and Block	Yes	Yes	Yes
Read and Write	Yes	Yes	Yes
Attach and Detach Devices	Yes	Yes	Yes
Abort I/O Operation	Yes	Yes	Yes
Asynchronous I/O	Yes	No	Yes
Time-out I/O Operation	Yes	Yes	Yes
Re-direct I/O Requests	Yes	Yes	Yes
Device Drivers	Most	All	All [2]
Add User-Written Drivers	Yes	Yes	Yes
Boot Strap	Down-Line	Disk/ Down-Line	Disk/ Down-Line
Core Usage [3]	16K	26.5K	8K
Execution Overhead	Least	Most	Average
Modularity	Average	Good	Good
Reliability	Poor	Good	Good
Maintenance Support	No	No	Yes

NOTES:

1. RSX-11D has multi-user capability, including Login/Logout commands, passwords, and job accounting. These features are being added to RSX-11M by MCA.
2. DEC will supply drivers for new devices for RSX-11M.
3. This refers to the virtual memory executive systems, exclusive of network programs.

D.5.1

ELF II has sufficient multi-tasking and multi-user facilities and has a limited debugging facility.

ELF supports memory management, but its file system is weak and not yet reliable. It cannot support program development, but it does have dynamic memory allocation. It has satisfactory input/output peripheral support, but has a limited selection of available device drivers.

ELF is not set up to be booted from a disk, although it could be transferred to a proper medium. It has average core requirements (16K), and low overhead. Its major drawbacks are that it is not fully developed nor reliable, it is difficult to modify and it has no maintenance support.

D.5.2 UNIX

UNIX has excellent multi-tasking, multi-user facilities, and an excellent file system. It is lacking a powerfail/restart capability and dynamic memory allocation. It also has good I/O peripheral's support, with a good selection of device drivers. Its main problem is that it cannot have a task's I/O execute asynchronously with that task; a task must wait for its I/O to complete before resuming execution. Also, UNIX does not offer the capability to redirect I/O requests.

The main drawbacks of UNIX are high overhead and large core requirements. Lack of maintenance support could be a problem, except that UNIX appears quite reliable, and rarely requires maintenance.

D.5.3 RSX-11M

RSX-11M has full multi-tasking capability, but it lacks an unrestricted multi-user development capability. However, this is included in RSX-11D, and it is being incorporated into RSX-11M by MCA. RSX-11M has full memory management facilities, a directoried disk file system with file protection, and can be used for program

development. It does not have dynamic memory buffer allocation for user programs (although this capability may easily be provided).

RSX-11M has excellent I/O support, including drivers for most DEC-supplied interfaces and peripherals, and the capability of installing user-written drivers. It can be bootstrapped from disk or via a down-line load. RSX-11M requires the least amount of core of the three operating systems, and has average overhead. It is modular, debugged and reliable, and it has full maintenance support from DEC.

APPENDIX E - AN ANALYSIS OF THE H6000
AS A NATIONAL SOFTWARE WORKS TOOL BEARING HOST

E.1 INTRODUCTION

The National Software Works (NSW) is an ARPANET-based, distributed network operating system capability. NSW is being developed to reduce the cost of software. Three features are expected to reduce software costs. First, NSW is expected to provide a congenial software development environment by shielding the developer from incompatibilities between different types of hosts in the network and by providing a number of automated development aids.

NSW software will be dispersed throughout the various hosts involved in NSW. Within each host, NSW will make use of the existing operating system but will provide standard NSW services to remote users. This will give the illusion of host to host compatibility. Between hosts, NSW will use existing ARPANET hardware and HOST/HOST, HOST/IMP protocols for communication. Second, multiple implementations of a software package on several different types of computers will no longer be required since NSW will provide convenient access to any software package from any host. Third, by providing a variety of automated management tools, NSW should improve the management of software development projects.

E.1.1 NSW Functional Components

Figure E-1 depicts logical relationships between NSW components.

1. Works Manager. This component is responsible for centralized control functions normally associated with an operating system. It maintains a network file system, validates NSW users, controls access to NSW resources and provides management tools. Because it provides centralized functions, the Works Manager appears to be a single entity even if it is partially distributed to improve reliability.

2. MSG. This component provides a flexible interhost, interprocess communication capability. It is the prime method used for communication between NSW components in different hosts. Initially, MSG will use the ARPANET NCP for interhost communication. MSG is the only component which will directly access ARPANET software. This

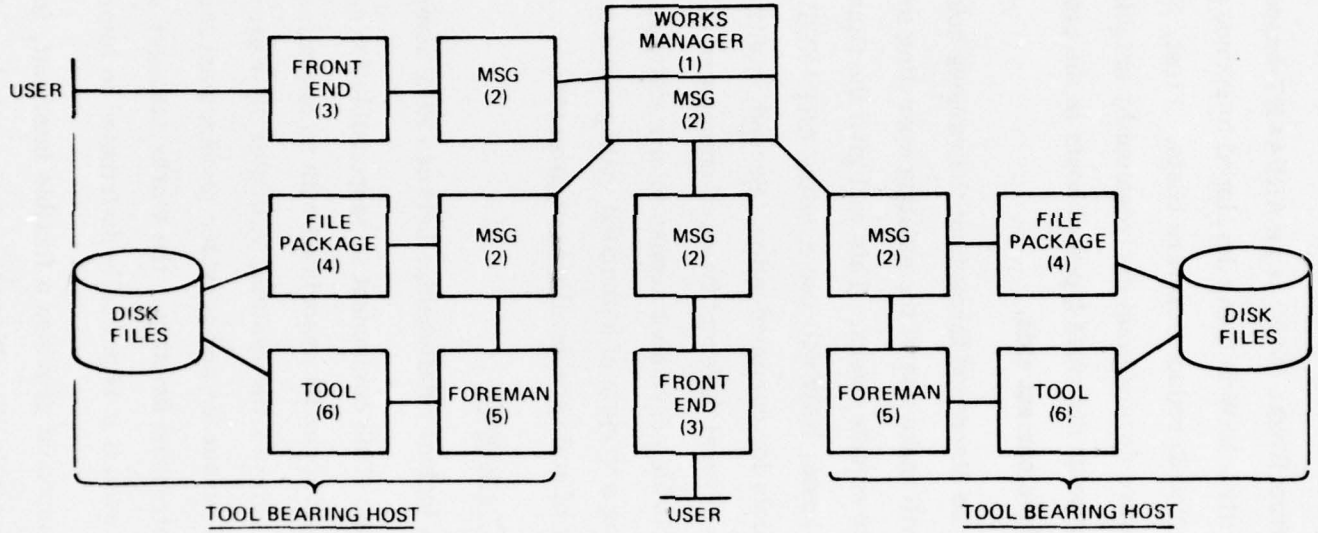


Figure E-1. NSW Functional Components

allows NSW to be independent of ARPANET; this will facilitate future NSW development.

3. Front-end (FE). This is a software function which provides user interface to NSW. While a FE is not required at any host, it must exist if a host requires user interface. In this context, FE is a software component of NSW, and although the name is similar, this should not be confused with minicomputer hardware connected as a NFE processor to a large computer. Because the government is developing FE computers for networking, the NSW FE software component might reside in a NFE computer.

4. File Package. This component is responsible for moving NSW files. It moves NSW files to and from local file systems, even though NSW files are also stored within local file systems. It is also responsible for moving NSW files from one host to another. When a file is moved, it is expected to be immediately usable, even if the movement is between different types of hosts. The File Package is therefore responsible for translating files to local standards during file movement. Initially, the only translation defined is for ASCII text files.

5. Foreman. This component provides the interface between tools and NSW. The Foreman is the most complex, distributed NSW component. Foreman responsibilities include: tool startup and termination, tool control, tool access to NSW resources, tool debugging, error recovery and tool encapsulation. The Foreman is expected to appear to a tool as part of the local operating system. In addition, tool encapsulation requires the Foreman to "trap" tool requests to the local operating system and translate these requests to NSW requests.

   It is expected that the Foreman implementation will involve minimum modification to the local operating system and existing tools. The Foreman implementation is general to allow for future development of NSW and tools.

6. Tools. Tools are not actually considered to be NSW components because NSW is normally thought of as connecting users and tools. A tool can be defined as any piece of software which a user might run. In particular, a tool is not part of an operating system but includes utility, application or user written software. Tools may be grouped into two general categories, those that are aware of NSW and those that do not discriminate between a local operating system and the NSW.

   New tools which are designed to run in an NSW environment are aware of NSW in that they issue NSW primitive calls to the Foreman. These tools will probably be well behaved and follow NSW conventions. For example, a well behaved tool would tell the Foreman when it is finished execution rather than turning the user over to the local operating system.

   Existing tools, developed before NSW, are presumably not well behaved, because they are likely to make extensive use of local operating system facilities. This includes those facilities, which in an NSW environment should only be available through the Foreman. Existing tools may either be modified to issue NSW primitives or encapsulated such that requests for local operating system services are "trapped" and handled by the Foreman.

E.2 AN INTERIM NSW/GCOS CONNECTION

Design and implementation of a fully compatible NSW/GCOS interface is anticipated to represent a significant effort and is discussed below. However, it is possible to provide a subset of NSW services to accomplish an interim NSW/GCOS connection.

An interim software discipline, referred to as Interface Protocol (IP) has successfully been used to interface a Burroughs B4700 with an IBM 360/91. This interface provides an interim NSW capability without requiring the following NSW components: MSG, File Package, and a Foreman.

Instead, an IP Server Message Processor at the host location provides a portion of these services, and a Work Manager process called Works Manager Operator, performs several batch-Foreman type functions.

Although IP does not provide a complete NSW capability, the FOREMAN specification states "... batch tools in NSW will be handled by the IP protocol for the immediate future ...". Because IP was initially designed as a "back end" protocol between a Network Front End and a B4700, it is particularly compatible with the ARPANET connection design presented in this document. Two alternative configurations may be envisioned as shown in Figure E-2.

The first alternative shows the IP Server implemented in the NFE. It communicates with the remote Works Manager (not shown) via NCP and uses IP. In turn, it communicates with the H6000 resident Server FTP and Server RJE, using their respective backend protocols. It should be fairly easy to use the same H6000 software for ARPANET FTP, RJE, and for the NSW IP Server. In fact, concurrent operation would not be too difficult.

Depending upon the ARPANET/GCOS, NFE loading, it is possible to implement IP in the H6000. This is shown in alternative 2. As shown, IP Server would communicate with the network using NCP, and would make use of H6000-resident FTP and RJE capabilities.

This software design would be somewhat complicated since IP Server would not use the HFP interface to communicate with Server FTP and RJE. Possible solutions include Server FTP, Server RJE, IP Server, and a HFP interpreter as separate tasks in a single H6000 process.

The first alternative is preferred since it realizes the H6000 off loading design goals.

IP Server does not provide full NSW capabilities. Only well behaved batch tools would be available. Although this may include most of the NSW tools currently envisioned for installation at RADC, no interactive tools may be provided. In

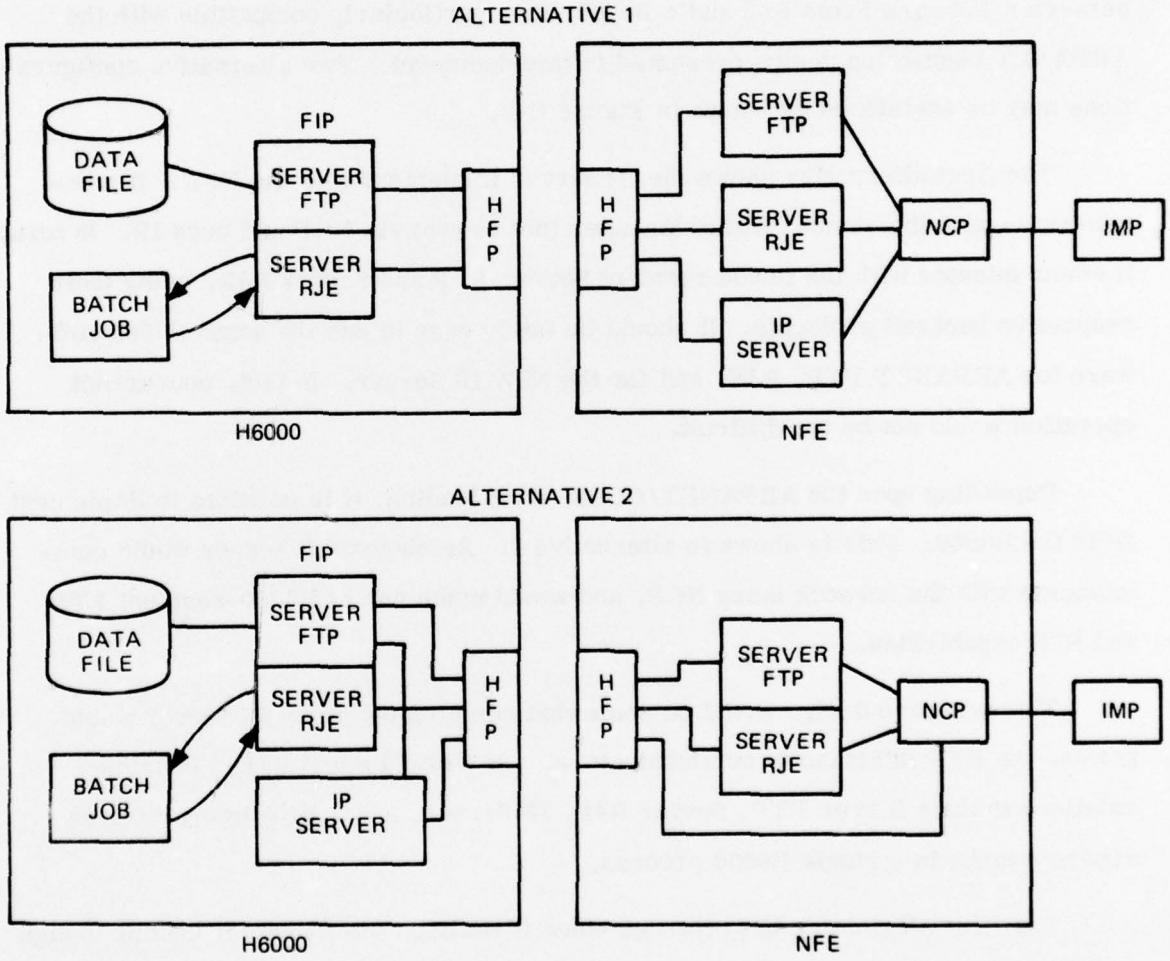


Figure E-2. NFE Alternative Configurations

addition, the lack of tool encapsulation would tend to exclude user programs, because NSW resource controls could not be enforced.

GCOS modifications may be designed which prevent a batch tool from accessing permanent files not specified in its JCL or other unauthorized resource requests. However, NSW file manipulations would not be available under IP.

In keeping with the philosophy of IP, file movement would occur before and after the execution of a batch tool and would be restricted to permanent files or tapes, as specified in the tool's JCL. Working copies of files would not be used since read only permissions are sufficient to avoid inadvertent destruction of input files. The batch tool's print/punch file, which is collected by a GCOS module known as SYSOUT, is available for dispersal as through RJE. To facilitate recovery, it may be desirable to copy SYSOUT collected output to a permanent file, but this is not required.

E.3 TOTAL NSW/GCOS CAPABILITY

A complete NSW TBH implementation would include more capabilities than provided in the interim approach. Major additions include the NSW file system, tool execution environment, and NSW software structure.

The NSW file system includes the concept of global file space which is never read or written by tools, and local work-space which is always assigned to a particular tool. A file update under NSW first involves copying the NSW file from global file space to local work-space assigned to a particular tool. The tool then updates the work file. After the update is completed, the work file is copied back to the global file space replacing the original file. Because GCOS does not support this type of file handling, it must be performed entirely by new H6000 NSW software.

Both NSW global files and local work files will probably be permanent files catalogued under a User Master Catalogue reserved for NSW. Most of the NSW software including NSW tools refer to these files by NSW-defined file names. One function of NSW software is to translate the NSW file names to local GCOS files names in a transparent manner. The NSW file name is syntactically incompatible with

GCOS file names. This may present some conversion problems for old tools, even when encapsulated, because GCOS expects individual processes to parse file names.

A detailed specification of NSW software structure for an H6000 is beyond the scope of this document. Certain design issues will be presented here which must be addressed during NSW design. Although NSW documents identify components such as Foreman, File Package, and MSG, these components are actually logical constructs with specific functional capabilities. The actual structure of NSW software for an H6000 may be quite different. This may occur because of the nature of GCOS which is considerably different than UNIX and TENEX operating systems. Most of the NSW software will probably be configured as system software.

GCOS contains three basic types of system software: Hard Core Monitor, SSA, and Privileged Slave. Each type has unique capabilities and limitations and require different kinds of interprocess communication. These characteristics must be carefully considered during design of NSW software for the H6000.

NSW components require close interaction. The ability to distribute NSW function between H6000 and NFE will be constrained by interaction requirements and must be carefully considered. Off-loading portions of NSW to the NFE is feasible, however new interfaces would be required, because interface characteristics are considerably different in components which are bound within the same process.

The NSW software is expected to provide a controlled environment to NSW tools. NSW primitives for initiation, termination, file manipulation and interprocess communication should be available to NSW tools. NSW design specifies that these primitives be available as "operating system-like" calls. This may best be implemented under GCOS as a series of MME functions. Under this approach, tools written for NSW operation are provided controlled access to NSW resources. In addition, it would be necessary to restrict the use of other MME functions. For example, NSW tools could not use all MME functions available for file catalogue activities, because all file operations are required to be performed through NSW software using NSW file names.

An alternative strategy, encapsulation, is suggested in the NSW design. With encapsulation the tool believes that it is operating in a GCOS environment. Requests for GCOS functions are trapped when issued by the tool and translated into NSW requests. A successful encapsulation allows the installation of existing software as NSW tools with minimal modification, possibly none at all. Regarding encapsulation, one important characteristics of GCOS structure must be considered. This can be depicted as shown in Figure E-3.

GCOS is primarily a batch-oriented operating system. The HCM portion of GCOS supports multiprocessing and multiprogramming of up to 63 batch programs within the constraints of available hardware resources. A large portion of GCOS processing is actually performed by special batch programs which are sometimes referred to as System programs or Privileged Slave programs. User run programs are called slave programs. Any batch program has the capability to communicate with one or more interactive terminals. In practice, the requirements for efficient batch operation are significantly different from requirements for efficient, interactive, time sharing operation.

GCOS addresses this by adding another layer of control for timesharing operation. The previous figure shows a batch program labeled TSS. TSS is a system program which handles multiple interactive users. TSS is functionally divided into an Executive and Subsystems. The TSS Executive controls the execution of subsystems, including the division of available core and processor resources. TSS is usually core resident. The executive, however, is continuously swapping subsystems such that they appear to be simultaneously running even though insufficient core resources are available for them all to be simultaneously core resident within TSS. Each subsystem is run on behalf of a user, and may be user written.

The execution environment provided to TSS Subsystems by the TSS Executive is significantly different than the batch execution environment. Subsystems are not allowed MME interfaces; instead, a different set of services is available through another interface.

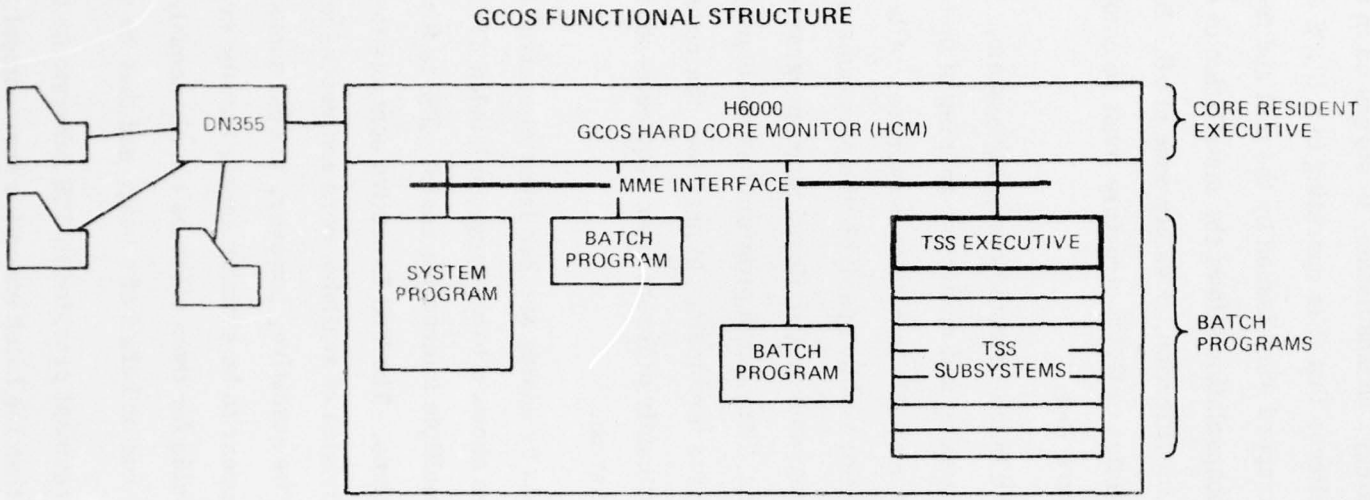


Figure E-3. GCOS Functional Structure

An essential question for NSW is the definition of an NSW tool. Are batch slave programs or TSS subsystems to be considered as tools or can both be tools? TSS Executive modifications and expansion are required in either event. It is probably not feasible to treat the TSS Executive as a tool because of its special relationship to the rest of GCOS, and because of the difficulty in differentiating TSS resource requests (for local users) from resource requests (for NSW users). If batch programs are to become tools, their MME requests must be trapped, and some mechanism provided to distinguish between local batch programs and NSW tools.

Finally, other privileged programs similar to TSS exist. Honeywell provides TPE and TPS for transaction-oriented, interactive systems. While this analysis has not dealt with TPE, TPS or any similar system software, it is mentioned here to show that TSS is not a unique case.

☆U.S. GOVERNMENT PRINTING OFFICE: 1977-714-025/170

METRIC SYSTEM

BASE UNITS:

Quantity	Unit	SI Symbol	Formula
length	metre	m	...
mass	kilogram	kg	...
time	second	s	...
electric current	ampere	A	...
thermodynamic temperature	kelvin	K	...
amount of substance	mole	mol	...
luminous intensity	candela	cd	...

SUPPLEMENTARY UNITS:

plane angle	radian	rad	...
solid angle	steradian	sr	...

DERIVED UNITS:

Acceleration	metre per second squared	...	m/s
activity (of a radioactive source)	disintegration per second	...	(disintegration)/s
angular acceleration	radian per second squared	...	rad/s
angular velocity	radian per second	...	rad/s
area	square metre	...	m
density	kilogram per cubic metre	...	kg/m
electric capacitance	farad	F	A·s/V
electrical conductance	siemens	S	A/V
electric field strength	volt per metre	...	V/m
electric inductance	henry	H	V·s/A
electric potential difference	volt	V	W/A
electric resistance	ohm		V/A
electromotive force	volt	V	W/A
energy	joule	J	N·m
entropy	joule per kelvin	...	J/K
force	newton	N	kg·m/s
frequency	hertz	Hz	(cycle)/s
illuminance	lux	lx	lm/m
luminance	candela per square metre	...	cd/m
luminous flux	lumen	lm	cd·sr
magnetic field strength	ampere per metre	...	A/m
magnetic flux	weber	Wb	V·s
magnetic flux density	tesla	T	Wb/m
magnetomotive force	ampere	A	...
power	watt	W	J/s
pressure	pascal	Pa	N/m
quantity of electricity	coulomb	C	A·s
quantity of heat	joule	J	N·m
radiant intensity	watt per steradian	...	W/sr
specific heat	joule per kilogram-kelvin	...	J/kg·K
stress	pascal	Pa	N/m
thermal conductivity	watt per metre-kelvin	...	W/m·K
velocity	metre per second	...	m/s
viscosity, dynamic	pascal-second	...	Pa·s
viscosity, kinematic	square metre per second	...	m/s
voltage	volt	V	W/A
volume	cubic metre	...	m
wavenumber	reciprocal metre	...	(wave)/m
work	joule	J	N·m

SI PREFIXES:

Multiplication Factors	Prefix	SI Symbol
1 000 000 000 000 = 10^{12}	tera	T
1 000 000 000 = 10^{9}	giga	G
1 000 000 = 10^{6}	mega	M
1 000 = 10^{3}	kilo	k
100 = 10^{2}	hecto*	h
10 = 10^{1}	deka*	da
0.1 = 10^{-1}	deci*	d
0.01 = 10^{-2}	centi*	c
0.001 = 10^{-3}	milli	m
0.000 001 = 10^{-6}	micro	μ
0.000 000 001 = 10^{-9}	nano	n
0.000 000 000 001 = 10^{-12}	pico	p
0.000 000 000 000 001 = 10^{-15}	femto	f
0.000 000 000 000 000 001 = 10^{-18}	atto	a

* To be avoided where possible.