AD-A040 926 ELECTRONICS ENGINEERING GROUP (1842ND) RICHARDS-GEBAU--ETC F/G 9/2
DESIGN SPECIFICATION FOR AFCS UNIVERSAL PERFORMANCE ASSESSMENT --ETC(U)
MAR 77 D R WORTENDYKE, R J STEELE

UNCLASSIFIED 1842 EEG/EEIMM-TR-77-17 NL

| OF |
AD
A040 926

END
DATE FILMED
7 - 77

(15)

1842 EEG/EEIMM TR 77-17

AFCS TECHNICAL REPORT

DESIGN SPECIFICATION FOR AFCS UNIVERSAL PERFORMANCE ASSESSMENT AND CONTROL SYSTEM

EMC/TEST AND COMPUTER BRANCH
1842 ELECTRONICS ENGINEERING GROUP (AFCS)
RICHARDS-GEBAUR AIR FORCE BASE, MISSOURI 64030

15 JUNE 1977

1842 ELECTRONICS ENGINEERING GROUP

MISSION

The 1842 Electronics Engineering Group (EEG) is organized as an independent group reporting directly to the Commander, Air Force Communications Service (AFCS) with the mission to provide communications-electronics-meteorological (CEM) systems engineering and consultive engineering for AFCS. In this respect, 1842 EEG responsibilities include: Developing engineering and installation standards for use in planning, programming, procuring, engineering, installing and testing CEM systems, facilit.es and equipment; performance of systems engineering of CEM requirements that must operate as a system or in a system environment; operation of a specialized Digital Network System Facility to analyze and evaluate new digital technology for application to the Defense Communications System (DCS) and other special purpose systems; operation of a facility to prototype systems and equipment configurations to check out and validate engineering-installation standards and new installation techniques; providing consultive CEM engineering assistance to HQ AFCS, AFCS Areas, MAJCOMS, DOD and other government agencies.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. N/A	3. RECIPIENT'S CATALOG NUMBER N/A
4. TITLE (and Subtitle) Design Specification for AFCS Universal Performance Assessment and Control System		5. TYPE OF REPORT & PERIOD COVERED Final rept.
		6. PERFORMING ORG. REPORT NUMBER N/A
7. AUTHOR(s) David R. Wortendyke Richard J. Steele		8. CONTRACT OR GRANT NUMBER(s) EIIIM-10
9. PERFORMING ORGANIZATION NAME AND ADDRESS U.S. Dept. of Commerce, Office of Telecommunications, Inst. for Telecom. Sciences, Boulder, CO 80303		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS N/A
11. CONTROLLING OFFICE NAME AND ADDRESS 1842 EEG/EEIM Richards-Gebaur AFB MO 64030		12. REPORT DATE March 17, 1977 13. NUMBER OF PAGES 28
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 1842 EEG/EEIM Richards-Gebaur AFB MO 64030		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)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Communication system monitoring, distributed computer networks, IEEE 488-1975 controllers, UPACS, minicomputer measurement system		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) A prototype data acquisition system that utilizes commercial software and hardware design concepts, for use in the development of performance monitoring systems of communication networks, has been designed, assembled, and checked out for the use of the USAF/AFCS. This document provides the design specifications for this universal performance assessment and control system (UPACS). Features emphasized are the use of a real-time software operating system, distributed processing, and the IEEE 488-1975 Digital		

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT (contd)

Interface for Programmable Instrumentation. A centralized minicomputer controls multiple distributed satellite processors by means of a multipoint asynchronous telemetry channel. All data acquisition takes place at the satellite terminals by controlling data acquisition devices connected to the IEEE 488 interface. A detailed description is given of hardware and operating system software. No specific data acquisition software is discussed.

APPROVAL PAGE

This report has been reviewed and is approved for publication and distribution.

Gerald T. Harris, Chief
Electronic and Base Systems Engineering Division

Amos J. Hardy, Chief
EMC/Test and Computer Application Branch

David A. Lindberg, Technical Area Mgr.
Electronics Testing

David R. Wortendyke, Electronics Engr.
OT/ITS, Author

Richard J. Steele, President
Minicomputer Systems Company,
Author

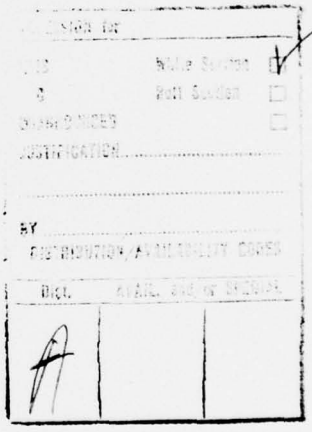


TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY OF TERMS

MNEMONIC ACRONYM	MEANING
ACK	Positive Acknowledgment
AFCS	Air Force Communication Service
ASCII	American Standard for Communication Information
BACI	Buffered Asynchronous Communication Interface
BCS	Basic Control System
CE	Communication Executive
CPU	Central Processing Unit
CRT	Cathode Ray Tube
EIA	Electronic Industries Association
ENQ	Enquiry
EOT	End of Transmission
ETX	End of Text
GPIB	General Purpose Interface Bus
HP	Hewlett-Packard
IEEE	Institute of Electrical and Electronic Engineers
IOC	Input-Output Control
I/O	Input-Output
MOS	Metal Oxide Semiconductor
MCU	Master Control Unit
NAK	Negative Acknowledgment
OCM	Operator Command Module
OT/ITS	Office of Telecommunications/Institute for Telecommunication Sciences
RTE-C	Real Time Executive, Core Based
RTE-II	Real Time Executive, Version 2
SCEGN	Satellite Communication Executive Generator
SCE/5	Satellite Communication Executive, Version 5
SOH	Start of Heading
STU	Satellite Terminal Unit (Remote minicomputer processor)
STX	Start of Text
SXL	System Cross Loader

Certain commercial equipment, instruments, and materials are identified in this paper to adequately specify experimental procedure. These commercial products were acquired for the USAF through competitive procurement procedures and are currently in use. This availability dictated their use for this study. In no case does such identification imply recommendation or endorsement by the USAF or the Office of Telecommunications, nor does it imply that the material or equipment identified is necessarily the best available for the purpose.

DESIGN SPECIFICATION FOR AFCS UNIVERSAL PERFORMANCE ASSESSMENT AND CONTROL SYSTEM

D. R. Wortendyke* and R. J. Steele**

A prototype data acquisition system that utilizes commercial software and hardware design concepts, for use in the development of performance monitoring systems of communication networks, has been designed, assembled, and checked for use by the USAF/AFCS. This document provides the design specifications for this universal performance assessment and control system (UPACS). Features emphasized are the use of a real-time software operating system, distributed processing, and the IEEE 488-1975 Digital Interface for Programmable Instrumentation. A centralized minicomputer controls multiple distributed satellite minicomputer processors by means of a multipoint asynchronous telemetry channel. All data acquisition takes place at the satellite minicomputers by controlling data acquisition devices connected to the IEEE 488 interface. A detailed description is given of hardware and operating system software. No specific data acquisition software is discussed.

Key Words: Communication system monitoring, distributed computer networks, IEEE 488-1975 controllers, UPACS, mini-computer measurement system.

1. PURPOSE OF THIS DOCUMENT

This document describes the final specifications for the initial AFCS Universal Performance Assessment and Control System (UPACS). This report defines the operation of the Master Control Unit (MCU) and the Satellite Terminal Units (STU's)† in sufficient detail to provide a semi-technical description of the hardware and software systems.

* The author is with the Institute for Telecommunication Sciences, Office of Telecommunications, U. S. Department of Commerce, Boulder, Colorado 80302.

**The author is with Minicomputer Systems Company, P.O. Box 3031, Boulder, Colorado 80303.

† The terminology of "Satellite" is used within this report to mean a slave computer, and does not refer to an object in orbit.

2. OBJECTIVES OF THE UPACS SYSTEM

The Universal Performance Assessment and Control System (UPACS) is a distributed measurement network, controlled by minicomputers at several radio communication sites for the purpose of collecting and analyzing data on the performance and operation of the radio equipment, link reliability, and digital transmission. The measurement equipment will initially be standard test equipment utilizing the IEEE 488-1975 (1975) General Purpose Interface Bus (GPIB)* for remote programming and data acquisition. The minicomputers controlling the GPIB will be polled on a serial party line (initially a "voice" channel) by a master computer for transmission of data to a central location. The master computer (MCU) will also have the capability of analyzing the data and developing new programs. An operator using simple commands at the master computer can cause newly developed programs to be loaded into the remote satellite minicomputers replacing the previous software measurement routines.

3. SYSTEM HARDWARE

3.1 MCU Hardware Configuration. The Master Control Unit (MCU) is responsible for monitoring one or more Satellite Terminal Units (STU's). It is a large real-time system with 32 k words of memory, magnetic disc, magnetic tape, paper tape reader and punch, a line printer, a plotting scope and a system console. Figure 1 provides a diagram of the system. Several manufacturers' equipments are utilized in the MCU, but the principal supplier is Hewlett-Packard (HP) Corporation.**

Listed below is a brief description of each unit in the MCU hardware.

1. Central Processing Unit (CPU)
   The CPU is an HP 2100S with 32 k words of memory. Built-in features include extended and floating point arithmetic, dual direct memory access ports, power fail interrupt and memory protect. It also has microprogramming capabilities.

**Certain commercial equipment, instruments, or materials are identified in this paper to adequately specify the experimental procedure. In no case does such identification imply recommendation or endorsement by the Office of Telecommunications, nor does it imply that the material or equipment identified is necessarily the best available for the purpose.

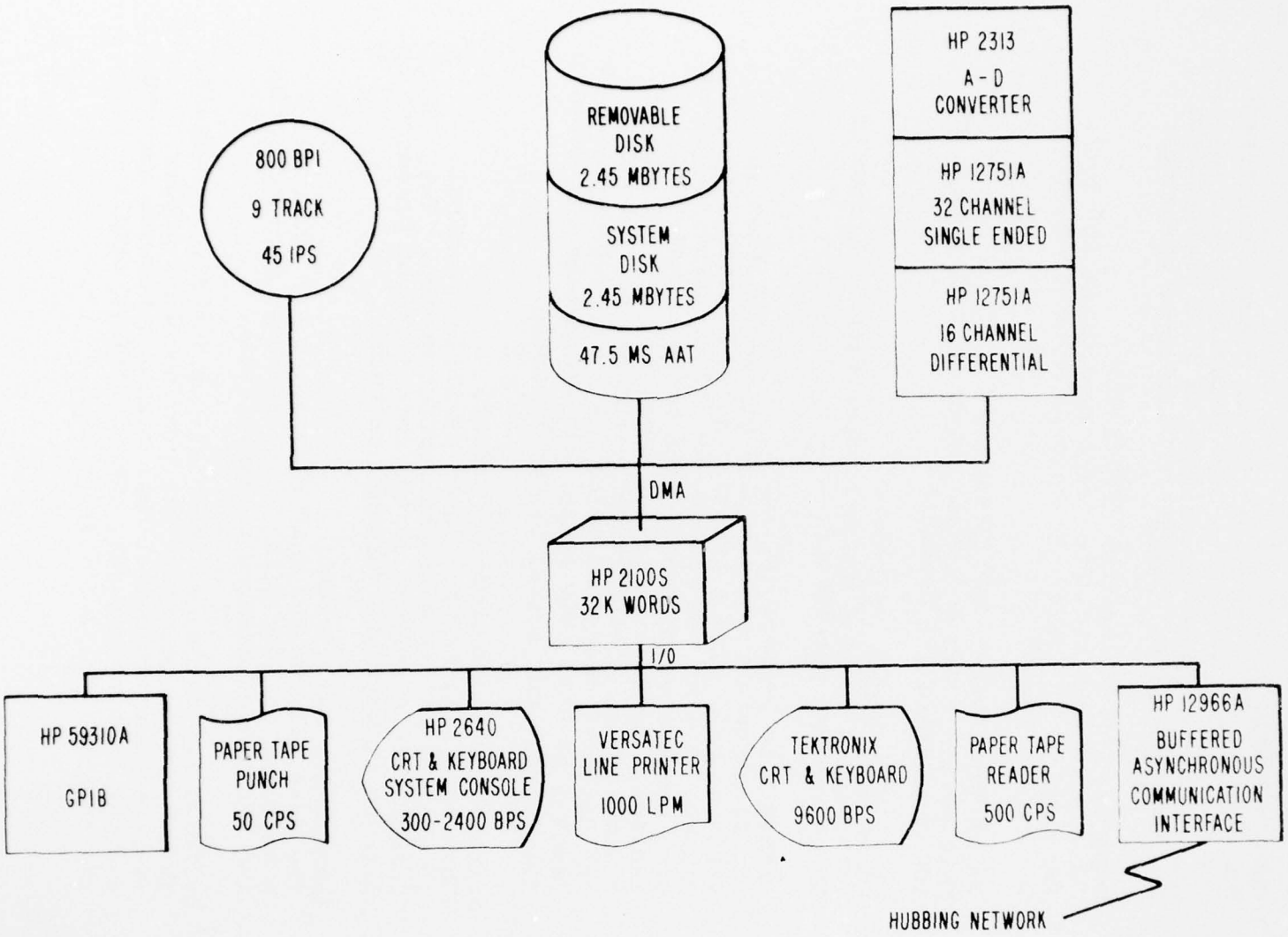


Figure 1. Master Control Unit hardware configuration.

2. Disc Subsystem (HP12960A)
The disc subsystem consists of two discs - one fixed and one removable with a total capacity of 4.9 million bytes. The average access time is 47.5 milliseconds. The removable disc uses an IBM 2315-type cartridge.

3. Magnetic Tape Subsystems (HP12970A)
The single magnetic tape drive records at 800 bytes per inch in 9 tracks. It records at a speed of 45 inches per second.

4. Paper Tape Reader (HP12925A)
This device reads 8 channel paper tape at a rate of 500 characters per second.

5. Paper Tape Punch (HP12926A)
This device reads 8 channel paper tape at a rate of 50 characters per second.

6. MATRIX Line Printer (D1110A)
The electrostatic matrix line printer/plotter is a Versatec unit with a printing speed of 1000 lines per minute. A line can have up to 132 characters. It can also produce graphs and charts using the plotting package. Resolution is 100 dots per inch arranged in a dual array. The Tektronix cathode ray tube (CRT) and keyboard described below are hardwired to the matrix printer so hard copy output of information generated on the graphic CRT screen can be recorded.

7. Graphic CRT and Keyboard (TEK4012)
This unit will be used on an auxiliary terminal to the CPU and as a graphic output device. The information displayed on the screen can be preserved by dumping it onto the Versatec printer.

8. Interactive CRT (HP2640)
The 2640 will be the system console. It includes a keyboard and CRT with a 5 k character storage buffering and a line drawing character set.

9. Buffered Asynchronous Communication Interface (HP12966A)
This hardware interface kit provides half-duplex asynchronous bit-serial data transfer between the CPU and the hubbing network. The interface complies with EIA Standard RS232-C (1969).

10. General Purpose Interface Bus (HP59310A)
A two-way digital communication structure for one or more instruments with ASCII-compatible interface is provided by this hardware feature.

11. A/D Subsystem
The A/D subsystem consists of HP2313B and two HP12751A High-Level Multiplexers. One 12751A card can input 32 single-ended channels and the other card can input 16 differential channels.

3.2 STU Hardware Configuration. The three STU units differ slightly in configuration. All three STU's have a time base generator, and a General Purpose Interface Bus (GPIB) (the IEEE 488-1975 standard), a Buffered Asynchronous Communication Interface and a system console interface. Two units have a Tele-Tec CRT and keyboard. The third unit has a Teletype® ASR 33 for a system console.

Figure 2 provides a diagram of all three satellite hardware configurations.

Listed below is a brief description of each unit in the STU.

1. Central Processing Unit
The CPU is an HP21MX with 16 k words of MOS memory. Standard features include extended and floating point arithmetic, memory parity, power failure detection and microprogramming capability. A two-hour battery backup unit is provided for the MOS memory. Each unit is also equipped with memory-protect hardware to maintain the integrity of the system software.

2. System Console
Each STU has a system console consisting of a Teletype® ASR 33 or a Tele-Tec CRT unit. The teleprinter includes paper tape reader/punch, keyboard and a 10 character per second printer. The CRT includes a keyboard and a screen capable of holding 24 lines of 80 characters each. The transfer rate is 240 characters per second.

3. Buffered Asynchronous Communication Interface (HP12966A)
This hardware interface kit provides half-duplex asynchronous bit-serial data transfer between the CPU and the hubbing network. The interface complies with EIA standard RS232-C.

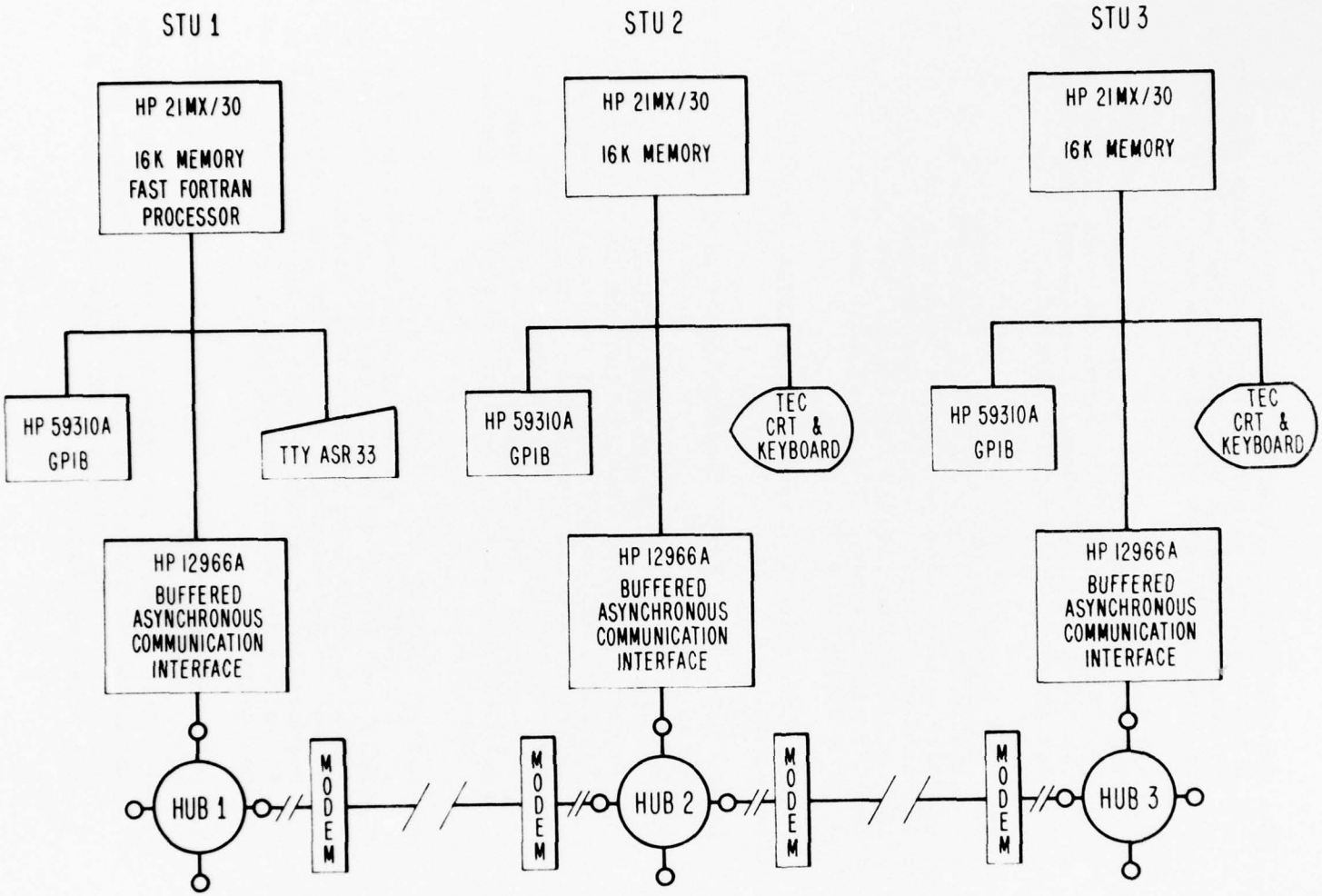


Figure 2. Satellite Terminal Unit hardware configuration.

4. General Purpose Interface Bus (HP59310A)
A two-way digital communication structure for one or more instruments with an IEEE 488-1975 compatible interface is provided by this hardware feature.

3.3 Communication Subsystem. The communication subsystem consists of the interface kits within each CPU and the hubbing networks mounted in each of the single bay racks for the STUs. The configuration also includes two 1200 baud modems in each bay for transmission on the voice channels of the communication subsystems. Each CPU communicates via an EIA RS232-C interface (25 pin connector) by means of the Buffered Asynchronous Communication Interface (BACI) (HP12966A). Each STU BACI is connected to S1 port of its hubbing network, which provides a digital communication path back to the MCU through the P1 port and a route to the next downstream STU via the P2 port. Figure 3 provides a diagram of the overall system including the communication subsystem.

4. MASTER CONTROL UNIT SYSTEM SOFTWARE

4.1 Introduction. The MCU uses the HP Real Time Executive Software, version 2, (RTE-II) operating system. It is a comprehensive, multiprogramming system which fulfills all the requirements of the UPACS system. Nearly all of the system software at the MCU is provided by HP. Three system software programs are provided by OT/ITS.

The first program is a driver for the Buffered Asynchronous Communication Interface. Since HP does not have an RTE driver for this device, project personnel wrote a driver and incorporated it into the RTE-II system.

A Communication Executive (CE) comparable in function to the STU communication executive is the second program. It is used to interface the user software with the drive for the Buffered Asynchronous Communication Interface. A copy of the CE program resides with each program communicating with an MCU. However, a common table capable of supporting 8 STU's resides in the operating system.

The third program is the Operator Command Module (OCM). It is responsible for providing the MCU operator with a means of controlling the STU.

The first two programs are written in Assembly language. The operator command module is coded in FORTRAN IV.

AFCS user programs may be written to run in resident or non-resident foreground and/or in non-resident background. The background area will be used primarily for program development and analysis.

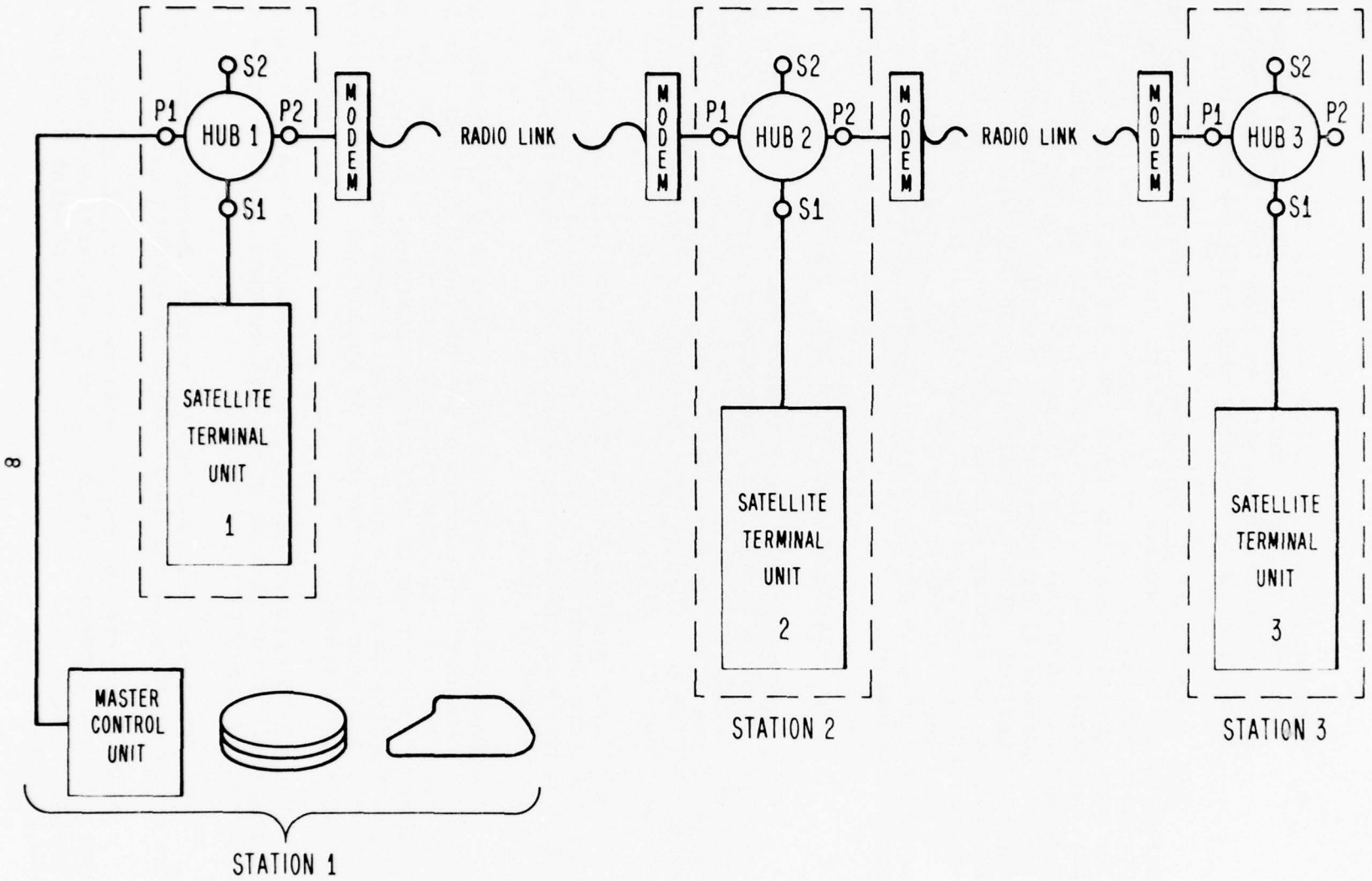


Figure 3. Universal Performance Assessment and Control System (UPACS) diagram.

A good general description of the HP RTE-II system may be found an article by Anzinger and Gadol (1975).

4.2 Operator Commands Available. A disc-resident background program called the Operator Command Module (OCM) allows the MCU operator to exercise control over the operations of any STU. Specifically he can perform any one of these five functions:

1) Load a program into an STU;
2) Start a program in an STU;
3) Stop a program in an STU;
4) Check the status of an STU, and obtain the current time;
5) Send time to the STU; and
6) For RTE-C based STU's, the operator can use any of the normal RTE-C commands.

Program loading requires the STU communication executive or the STU system loader to be in control. The STU communication executive operates under the Basic Control System (BCS) or the core based Real Time Fxecutive (RTE-C) system. It can load an absolute program into the user program area. If control resides with the system loader, only a BCS or RTE-C system with the supporting system software can be loaded. However, a user program could be included as part of the system files.

Once a program is loaded, the MCU operator can give the command to start execution of a user program. The STU communication executive remains in control until the start command is given.

The stop program command terminates a user program and returns control to the STU communication executive. A new user program can then be sent to the STU. In addition, another command is also available which causes the system loader to gain control, so a new system can be transmitted to the STU.

The STU's time of day, day of year, and year can be updated using the set time command. A small time offset may be specified in the command to compensate for the transmission time and processing time in the MCU and STU.

The status of an STU can be checked with the OCM. An STU can be under control of a user program, the STU, CE, or the system loader.

4.3 Satellite Program Loader. Before a user program can be transmitted to an STU, it must be processed by a loader which resolves all external references, links the user program to the required library routines, and generates absolute program code. The HP System Cross Loader (SXL) is used for this purpose in BCS systems.

Initially a BCS system with the drivers, communication executive, and clock routines is loaded to produce a cohesive system called the STU System Software. The STU System Software is a single file in absolute format which is loaded by the system loader in the STU. A snapshot of memory at the STU after the System Software is loaded is maintained at the MCU.

Whenever a user wishes to prepare his program for loading, he will use SXL and the snapshot file to produce an absolute program with the proper links to the STU System Software. This program is saved in a file at the MCU, and loading is initiated using the operator command module. When the module is loaded, the MCU operator can start the STU user program.

If RTE-C is chosen as the STU operating sytem, the loading is performed by the HP Satellite Communication Executive, version 5 (SCE/5) satellite generation program called SCEGN from the HP Distributed Systems Library in place of the SXL loader.

5. SATELLITE TERMINAL UNIT SYSTEM SOFTWARE

5.1 System Software Under BCS. The STU system software consists of a collection of programs which the user program accesses to communicate with the various peripherals. Included are the device drivers, the BCS system, the clock routines, the GPIB handler, and the communication executive.

Device drivers operate at the lowest level. They communicate directly with the hardware devices - they are responsible for initiating the I/O operation, handling interrupts, and cleanup when the I/O transfer is complete. Four device drivers are required. They are:

1) The HP time base generator driver called D.43;
2) The HP GPIB driver D.37;
3) The HP multiterminal driver D.00 for the teleprinter or the CRT's; and
4) A specially developed driver for the HP buffered asynchronous communication interface card.

All four drivers will operate under BCS - the HP Basic Control System. BCS is really just an interface to these drivers.

The next level of system programs communicate downward with the drivers via BCS and upward via the user program. They include:

1) The communication executive which handles all data transfers with the MCU;
2) The GPIB interface which provides a convenient means for the user to perform I/O and control the GPIB and the devices attached to it; and
3) The Time Base Generator routines which maintain the time, date, and user timers.

All of these programs are written in Assembly language. They represent the STU resident software system and are loaded only once. All of the system software is protected using the HP memory protect feature. Without some form of protection, a user program could inadvertently damage the system software which might require operator intervention at the STU to get the system restarted.

The STU system software is loaded from the MCU using the system loader. This loader is a small compact program which resides in upper memory of the STU and could eventually be micro-coded into a read-only memory (ROM).

A block diagram of the STU system software is shown in figure 4. Figure 5 presents a memory map of the system software with approximate memory requirements.

5.1.1 Timekeeping Under BCS. Timekeeping functions at the STU are handled by several clock routines and a modified version of the HP clock driver (D.43). The following functions are performed by the clock routines:

1) Maintain the time of day in terms of 10's of milliseconds, seconds, minutes, and hours;
2) Maintain the day of the year and the year;
3) Provide a user program interface for accessing the time and date, and setting it; and
4) Provide a user program interface for defining count-down timers which are automatically decremented.

Time can be set through the user program or from an MCU message. Once set, the time will be maintained automatically.

The count-down timers are useful in handling time-dependent functions within the user program. A user program will call a clock routine with the initial time and a location for counting. The clock routine will decrement the count until it reaches zero and hold it there. The user program need only test for zero and then reset the timer after the necessary processing is complete. This timer operates in conjunction with the real-time

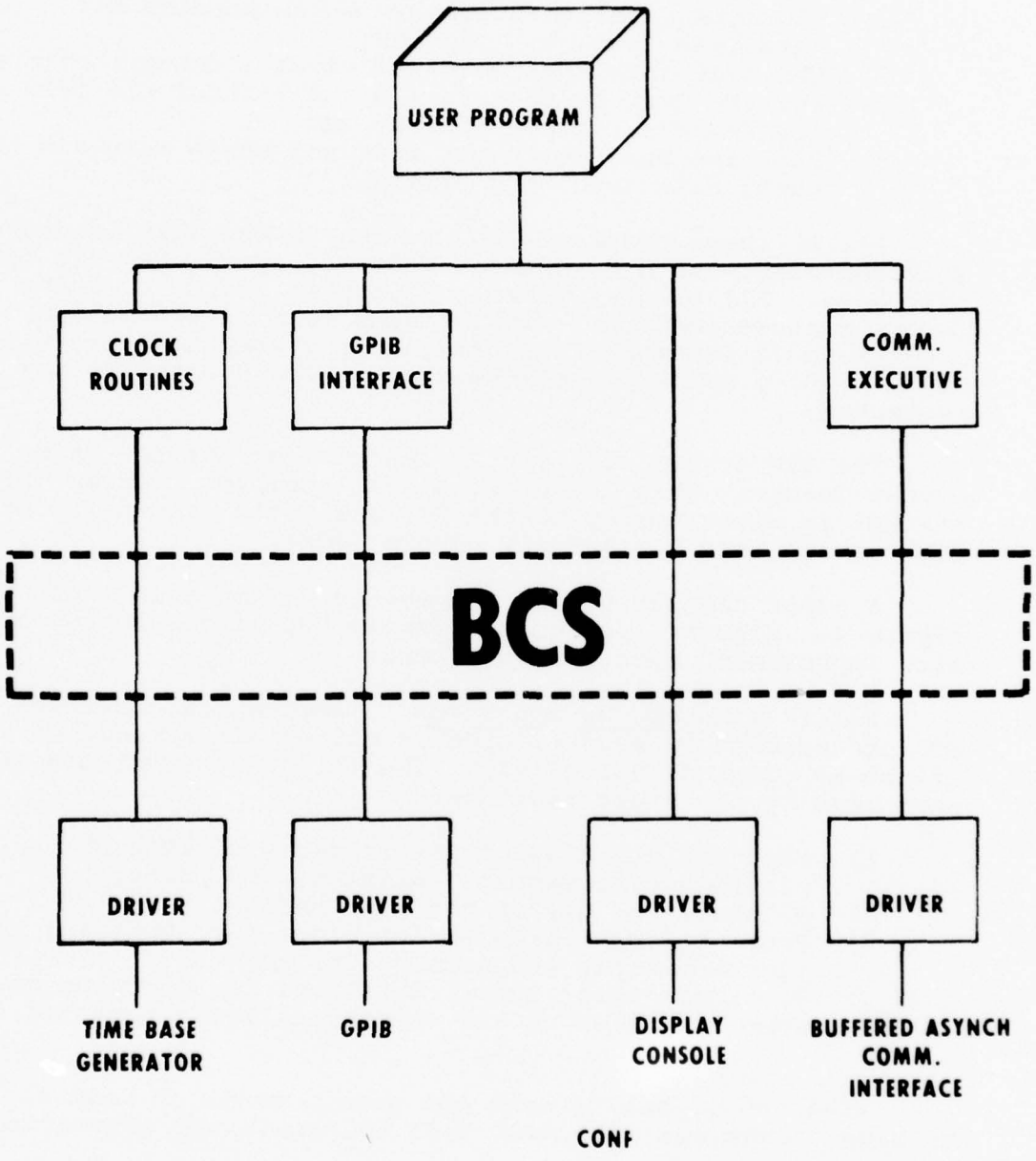


Figure 4. BCS STU system.

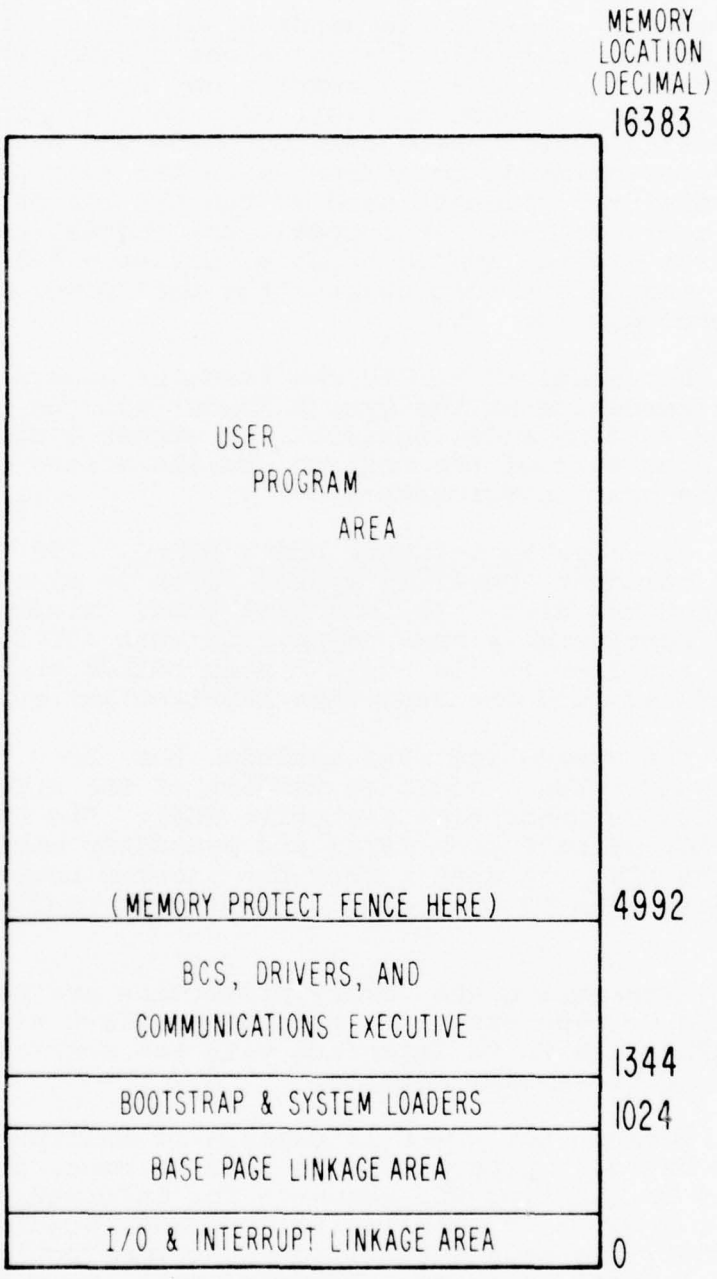


Figure 5. BCS STU memory map.

clock so that events scheduled every hour for example, will always occur at the beginning of each hour.

5.1.2 Operator Commands Available Under BCS. Since the STU is basically a stand-alone system, that is, normally unattended, it does not require any system commands. Further, the system software consists of a simple, single-tasking operating system which does not have the capability to process operator commands concurrent with the user program. Therefore, any operator commands used at the STU are implemented in the user program. An operator can request access to a user program via the system console (either a teleprinter or a CRT) and as a result control the user program or request information from it.

Personnel at an STU can transfer control to the communication executive or the system loader via the CPU console switches or by forcing a CPU restart. In either instance the MCU would be told of the restart and the status of the STU system on the next interrogation.

5.2 System Software Under RTE-C. RTE-C is a standard HP core resident operating system which is similar to RTE-II without the disc. It is a real-time, multiprogramming system with considerably more capability than a BCS. There are three main routines in the RTE-C system called the executive, the scheduler, and the real-time input/output control program.

The system software includes the above three modules of RTE-C, plus a modified version of the absolute loader, and the communication executive (CE). The CE is responsible for MCU message processing and preparing messages for transmission to the MCU. It also routes the program code to the absolute loader. The CE is a scheduled program operating under RTE-C.

Timekeeping and memory protection are the responsibility of RTE-C. The executive program of RTE-C has been modified to allow the CE to interface with the commands normally used through the system console.

Because the RTE-C is capable of multiprogramming, more than one user program can be running concurrently. One program could be handling MCU requests for information while the other program is handling all the data acquisition responsibilities. In fact, the data acquisition functions could be separated into several programs. However, only one program can communicate with the CE at a time.

Figures 6 and 7 provide a block diagram of the system under RTE-C and a memory map, respectively.

5.2.1 Operator Commands Under RTE-C. The system console under RTE-C can be used to communicate with the user program or RTE-C. There are 13 RTE-C commands available from the system console. Any of these commands could be used concurrently with user program execution.

5.3 Power Failure Detection/Automatic Restart. Each STU central processor is equipped with a power monitor which generates an interrupt to the CPU when power fails. The CPU saves the current system state and performs an orderly system shutdown. The batteries maintain the MOS memory contents for as long as two hours. When power is restored, the automatic restart software initializes the system and returns control to the interrupted routine.

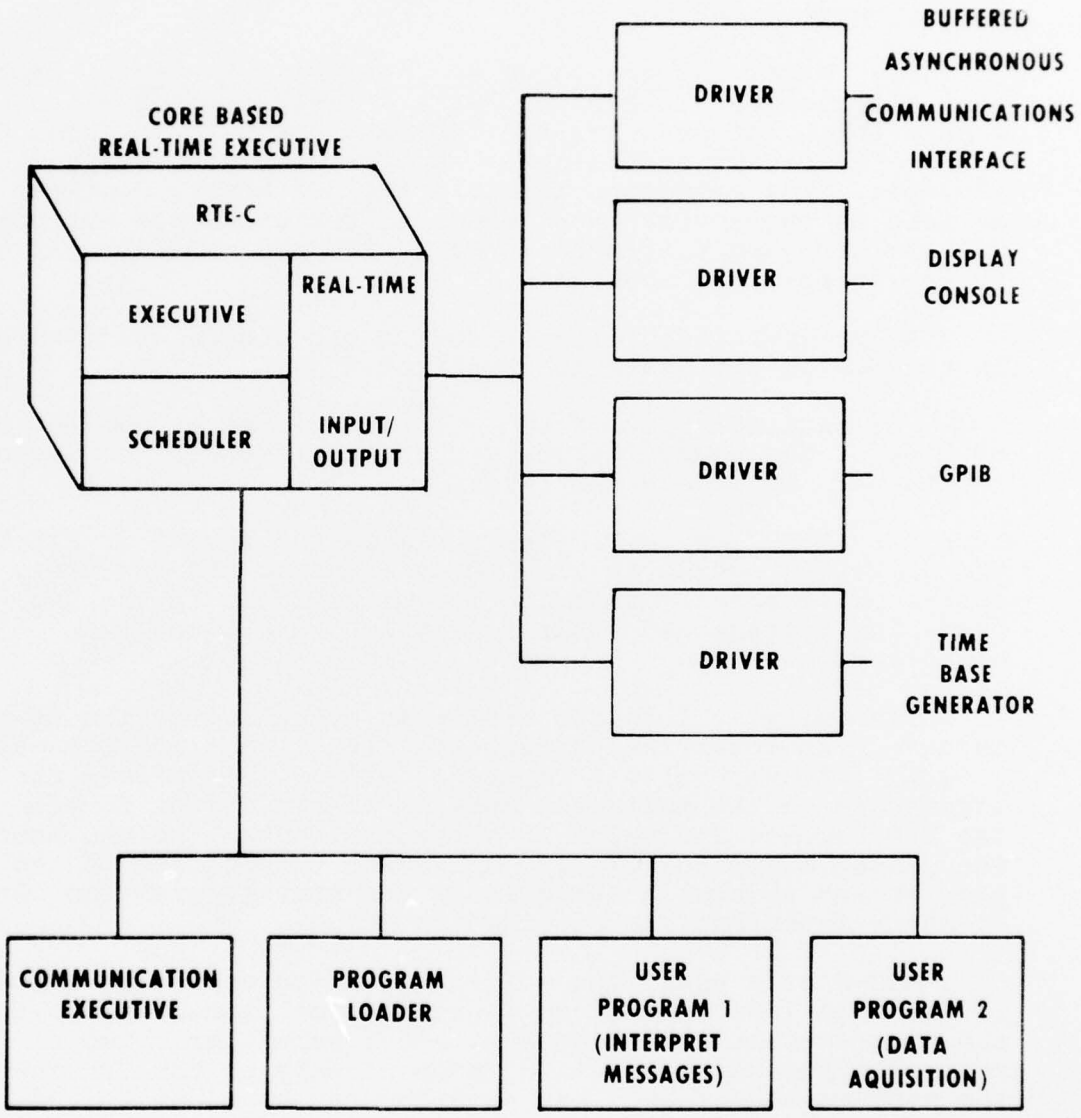
A power-failure flag is set in the status and transmitted to the MCU on the next interrogation.

The internal time of the STU will be off by the amount of time of the power failure. Therefore, it will be necessary for the MCU to update the time.

5.4 RTE-C vs. BCS. The decision to use BCS or RTE-C as the operating system in the STU is an option of the user. This section will briefly describe the differences in the two operating systems and their respective advantages and disadvantages.

Basically, BCS is not really an operating system at all. It is merely an interface between the I/O drivers and the user program. A user calls a routine, "IOC", which passes the parameters to the initiator section of the driver to set up the I/O request and which then returns control to the user. The driver completes the I/O operation independent of the program and updates a table which the user program can check. No other functions are provided in BCS.

RTE-C is a real-time multiprogramming operating system. It includes I/O control, an executive for processing operator- and program-initiated requests, and a scheduler. Two or more programs can be running independently in the CPU sharing the system's resources. New programs can be added to the system at the same time other programs are being executed. When a program makes an I/O request via an executive call, the RTE-C system suspends the program until the I/O operation is complete. Other executive calls include scheduling other programs, scheduling the calling program, getting the time, setting the time, and checking the status of I/O devices. In addition, RTE-C has 13 operator commands that can be used via the system console.



EXEC

Figure 6. RTE-C STU system.

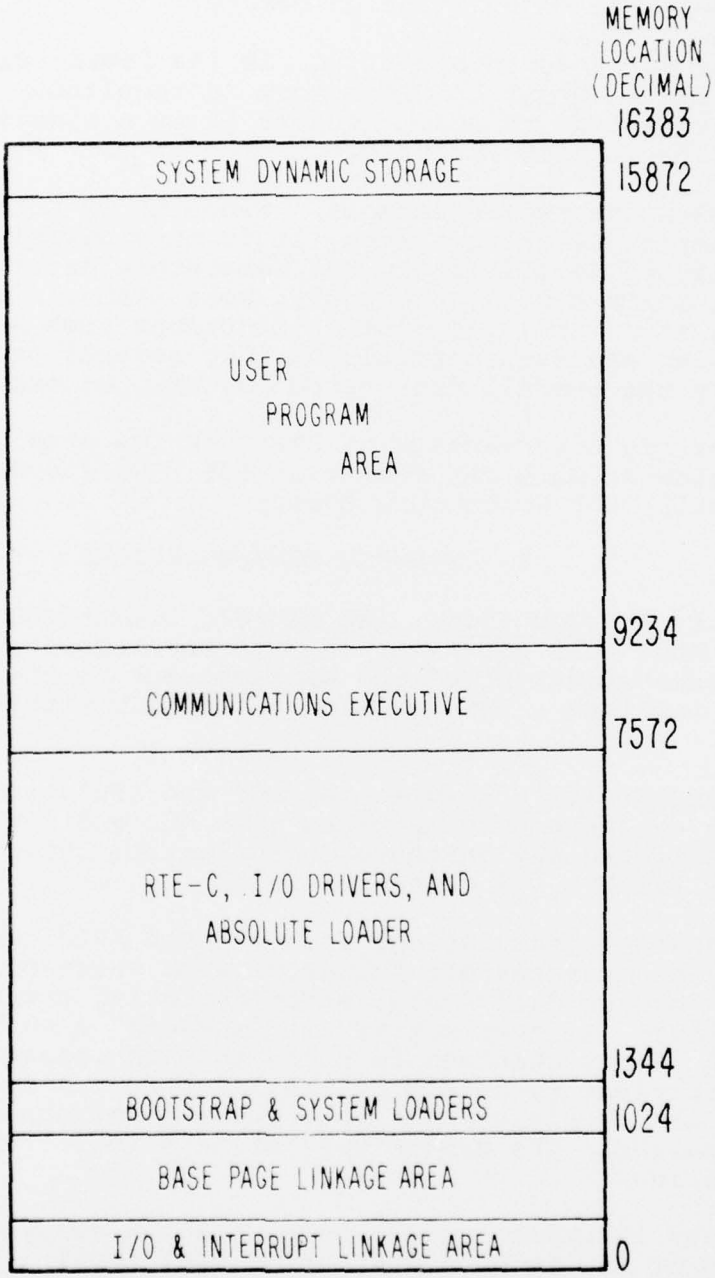


Figure 7. RTE-C STU memory map.

To summarize, BCS is an interface to the I/O drivers. RTE-C is a complete operating system. As might be expected, BCS requires less than half the CPU memory of RTE-C. For UPACS, a BCS system uses approximately 5,000 words of memory. RTE-C requires 9,300 words of memory.

The major advantage of BCS is its lower hardware cost, since only a small amount of CPU memory is required. There is one disadvantage to BCS: because it is a simple operating system, it is also inflexible.

RTE-C has two advantages. Since it is a large multi-programming operating system, it is considerably more flexible, which may be beneficial in the laboratory environment of the initial UPACS system. Secondly, more than one program can be running in an STU. Separating functions such as MCU message processing and data acquisition into several programs should simplify the overall user system as well as each program.

The big disadvantage of RTE-C is its size. It requires almost twice as much CPU memory as BCS. Consequently, each STU may require 8 k words more memory.

6. MCU/STU COMMUNICATIONS

6.1 Introduction. The MCU/STU communication is controlled by the MCU. The STU responds with the requested information or an acknowledgment of the message sent by the MCU. The STU cannot initiate a transmission to the MCU without first being polled.

Communication between the MCU and STU's can be up to 9600 baud in the asynchronous mode. The MCU and STU's each contain an HP Buffered Asynchronous Communication Interface (BACI card #12966A) with a 128 character buffer.

Message transmissions include both ASCII and binary information. The messages are formatted with start-of-message and end-of-message delimiters, error-detecting characters and address characters. An even parity bit is added to each 8 bit character. A 16-bit checksum is added to each message. Error checking is sufficient to detect multibit errors. A transmission error causes the receiving unit to request a retransmission of the last message. The system will attempt to retransmit a message three times.

User interface to the communication subsystem from either the MCU or STU is on a subroutine call level. To transmit a message, a user calls a communication executive (CE) subroutine with the appropriate parameters. The message is formatted by the MCU CE and sent to the STU. If the user message is too long for a physical message, the CE will send multiple messages to the STU.

Each message is acknowledged by the STU. The STU CE reassembles the message and passes it to the user program.

The STU user program must have made a call to the CE prior to receiving an MCU message. If the STU CE receives a message for a user program, but the user program has not issued a call to the CE, the CE will set a status flag indicating a user buffer (supplied via the call) was not available and then the CE will acknowledge the message.

In the MCU, the user program or OCM transmitting a message is suspended until a return message is received. However, the user program in the STU is not suspended when it calls the CE. A return code, supplied in the call to the CE, is set to minus one by the CE before returning control to the calling program. When a user message is received, the CE will change the return code to a number indicating the message status. The STU user program must periodically check the return code to determine if a message is waiting in the user's buffer.

The CE's in both the STU and MCU handle all the message formatting, error checking and message acknowledgments. The STU user program, upon receiving a message from the MCU, simply calls the CE with a return message. The CE formats it and sends it back as a reply to the MCU user message. If the STU user message is too long for one physical message, the STU CE will send multiple messages to the MCU. Each message is acknowledged by the MCU CE.

With the RTE-C operating system in control at the STU, several user programs can be running concurrently. Therefore, one user program should be responsible for handling information requests from the MCU while the others are responsible for data acquisition. The user message handling program is scheduled whenever the CE receives a valid message from the MCU. It extracts the requested information from the data base and returns it to the MCU.

6.2 Communication Protocol. Communication between the MCU and STU's is controlled by a rigid set of rules. First, only the MCU can initiate a communication sequence. An STU cannot send a message to the MCU without being asked for one by the MCU. Second, every message sent from one subsystem to the other must have a reply. The reply can be a response message, an acknowledgment (ACK)* or a negative acknowledgment (NAK). The absence of a reply from an STU within the timeout limit for the physical unit generates a message to the MCU operator. However, an STU which fails to get a response from the MCU cannot do anything but wait for the MCU to poll it again. The STU always retains the last message transmitted in case retransmission is required.

*All characters used in this section are taken from the ASCII character set.

Third, an MCU can talk to only one STU at a time. For example, a user program can poll an STU and wait for a reply. If no reply is received within the allotted time, a message should be sent to the operator and/or logged; then the user program should go to the next STU.

Finally, the message is in a precise format which is verified by the receiving system. The first ten characters of a message are always the same. Also, the last six characters of the message are always the same. Between the header and trailer are the message characters. The header consists of addressing and message content information.

Table 1 provides a summary of message format for each byte sent. The first six characters are reserved for the MCU and STU drivers which interface with the HP Buffered Asynchronous Communication Interface card which is called the BACI card in subsequent text. Since all BACI cards are set to a listening mode, a character transmitted down the communication link by the MCU or an STU is received by all BACI cards. The special character buffer of the BACI card contains the address character for the MCU or STU. The driver is notified via interrupt if the address character is detected. It also gets an interrupt from the BACI card on buffer full and half-full. However, these interrupts are ignored. When the driver receives a special character interrupt, it reads characters until it detects a start of message character (SOH). All subsequent characters are stored in the CE buffer.

The last four characters of the header, in an MCU-to-STU message, which are stored in the CE buffer contain the address character of the message destination, the address character of the message source, the message type code and the message block number. Presently, the characters A through J are reserved for addressing. Address characters A and B are reserved for the primary and secondary MCU. Characters C through J are assigned to STU's 1 through 8.

The type code identifies the message content and format. There are 14 different message types; 13 message types are used by the system and the 14th is reserved for user messages.

The block number is used to detect missing messages and to ensure message sequence integrity. Block numbers start at 6 and increase numerically to 63. Then they start over at 6 again. Both the MCU and STU's maintain separate block number registers for transmitted and received messages. All CE's verify the block number of received messages. An error causes the error statistics of the block number to be updated.

Table 1. Message Formats

MCU Char #	STU Char #	Description of Character	Comments
1	1	Destination or To Address Character	Used by Driver
2	2	Secondary MCU Address Character	
3	3	Fill Character for Timing	
4	4	Fill Character for Timing	
5	5	Fill Character for Timing	
6	6	Start of Message Character (SOH)	
7	7	Destination or To Address Character	Included in Error Code
8	8	Source or From Address Character	
9	9	Message Type Code	
10	10	Message Block Number	
	11	Status Character 1 } STU Message	
	12	Status Character 2 } Only	
11	13	Message Text Character 1	
12	14	Message Text Character 2	
.	.	.	
.	.	.	
.	.	.	
10 + n	12 + n	n	
11 + n	13 + n	End of Text {STX = more to follow; ETX = no more}	
12 + n	14 + n	Error Code Character 1	
13 + n	15 + n	Error Code Character 2	
14 + n	16 + n	Error Code Character 3	
15 + n	17 + n	Error Code Character 4	
16 + n	18 + n	End of Message (EOT)	

Maximum message length is 540 characters.
Maximum test message length, n is 524 characters.
Minimum message length, n is 16 characters.

The message text follows the block number. It can contain ASCII characters, binary numbers, or any bit patterns. The maximum message text length is 524 characters. Any text character which matches a special character (binary numbers 1 through 5 and address characters A through J) is separated into two characters, and preceded by an encoding character. For example, a text character A (bit pattern 01000001) in the original message becomes three characters in the transmitted message whose 8 bit patterns are 00000101, 01100100, and 01100001. The left four bits of the second and third characters (0110) represent encoding bits to produce a bit pattern which does not match any special characters. The text message is decoded into its original character string at the receiving terminal. This technique requires only 6% of the possible 256 bit patterns to be encoded.

The character immediately following the message text is either an end of address (STX) or end of message (ETX). An STX character indicates the message is not complete and more text will follow in the next physical message. The receiving terminal formats the message into the user buffer and acknowledges the message. A user buffer can be longer than one physical message can handle. All system messages are less than 524 characters in length. After the sending terminal receives the acknowledgment, it sends the next message segment. This sequence is repeated until all the message text characters are sent. The last message has an ETX which indicates to the receiving terminal that the last message block has been received.

Following the STX or ETX character are four error code characters. They represent a 16 bit checksum of the message characters beginning with the "TO" or destination character (following the SOH - the sixth character in the message) and ending with, but including, the STX or ETX character. The 16 bit error code is separated into four characters to ensure a bit pattern that does not match any special characters. The left four bits of each character contain the encoding pattern 0110. If the calculated sum and the error code sum do not match, the receiving terminal returns a negative acknowledgment (NAK) and waits for a new message.

The last character of each physical message is an end of transmission character (EOT). Once a receiving terminal driver detects the EOT, it sets a flag indicating the operation is complete. The CE does not process the message until the EOT is received or a timeout occurs.

A message sent from the STU to the MCU has two additional characters in the header following the block number. These two characters represent the STU status - a 16 bit word which includes current STU status and the status of the last message received. The status word is separated into two 8 bit

characters. Labeling the left-most bit as bit 15 and the right-most bit as bit 0; bits 15 and 7 are always set to 1, so there is no danger of a bit pattern matching one of the special characters. Table 2 defines the status word and the meaning of each of the bits which are defined.

A message acknowledgment can be a return message, a positive response, or a negative response. A message sent to an STU for information will be acknowledged by a response message containing the requested information. However, a message containing a command for the STU to get ready to accept a program would cause the STU to send back an ACK message signaling that the STU got the message and was ready to accept a program. In all instances a message from the MCU must be answered with a message from the STU.

When an STU has sent back an information message, the MCU will acknowledge receipt of the message with an ACK if there are multiple message blocks, an NAK, or a new message. A valid response which has only a single message block will not be acknowledged.

A negative acknowledgment will cause a retransmission of the last message. For example, the STU returns an information message to the MCU, the MCU detects an error and sends a negative acknowledgment, the STU then sends the last transmitted message again. This dialogue will be repeated three times, and then the MCU will set a return code and terminate the dialogue.

Many system message types do not require any message text. For example, the positive and negative acknowledgment (ACK and NAK) do not have any text. The type code distinguishes one from the other. Table 3 provides a definition of each message type, its message length, the received message length, the valid responses, and who uses the message type. Table 4 provides a sample of communication sequences.

7. USER PROGRAMS

7.1 User Program Languages. The MCU and STU application software languages include FORTRAN IV and Assembler. The HP BASIC is not suitable for this system. Since the STU system software requires program loading in absolute format, a compiler must be available to convert the language into Assembler and/or into absolute machine code. Assembly language produces machine code. The standard HP FORTRAN IV compiler does this also. However, the HP BASIC does not generate machine code. It cannot be compiled into an absolute program; and it also shuts off interrupts during statement execution. Such cancellation of interrupts cannot be tolerated in a real-time environment. The Kearfott Division of the Singer Company has a BASIC language which meets the minimum system requirements. It could be

Table 2. STU Status Word

Bit	Usage
0	System Loader In Control
1	CE In Control
2	User Program In Control
3	Power Failure Occurred
4	Memory Parity Error
5	Memory Protect Error
6	Address Error On Load
7	
8	
9	
10	
11	
12	Parity Or Error Code Error
13	Block Number Error
14	Word Count Error
15	

purchased from Singer and used within the system if there is a requirement to develop programs in BASIC. It is anticipated that the majority of all user program requirements may be satisfied with the Fortran IV provided.

7.2 User Program Interface to the MCU/STU. A user program converses with the MCU or STU through the respective Communication Executive (CE) which talks with the Buffered Asynchronous Communication Interface. If BCS is used at the STU, a user program will check for an MCU request periodically by testing a flag given the CE by a call. The CE sets the flag when a message has been received from the MCU. The STU user program sends a message back to the MCU by a program call to the CE giving the message buffer address and its length. The CE is designed to transmit the message and to retransmit it if necessary.

An MCU user program upon requesting a message from a particular STU, is suspended until the message is received. Then the MCU is given control and processes the message. When the MCU program sends a message it makes a call to the CE giving the message buffer address and the length. The CE takes the message and transmits it to the STU.

If RTE-C is used at the STU, the same procedure is followed for communication as in a BCS based STU. However, with RTE-C, multiprogramming is available which gives the user the capability to have one program dedicated to processing MCU requests and another program for the data acquisition.

7.3 Typical Development Cycle For a User Program. All user program development occurs at the master station. The user will also test his program at the master station using simulated input and/or the MCU's GPIB. The program could be written in Assembler or FORTRAN IV.

After testing is completed, the user generates a relocatable binary file using either the FORTRAN IV compiler or Assembler. Next he runs the cross loader which will be SXL for BCS based STU's or the satellite generation program SCEGN if RTE-C is selected as the STU operating system. This process includes linking the user program with the STU resident operating system and the FORTRAN IV and system library. The end product is the absolute program code stored in a disc file.

The user program is ready for loading at an STU. The MCU operator, using the operator command module stops the current user program running at the STU. He may first request status from the STU. Then the operator sends the user program to the STU. Upon successful transmission, the operator commands the STU to start executing the program. The user monitors the STU program via the message content returned to the MCU.

Table 3. Message Function

MESSAGE TYPE (Decimal)	MESSAGE DESCRIPTION	TRANSMITTED TO STU		RECEIVED FROM STU		VALID RESPONSE MESSAGE TYPE	USED BY		AVAILABLE TO	
		TEXT LENGTH IN CHAR	MSG LENGTH IN CHAR	TEXT LENGTH IN CHAR	MSG LENGTH IN CHAR		BCS BASED SYSTEM	RTE-C BASED SYSTEM	OCM ONLY	USER
0	System Loader for loading BCS or RTE-C System	VAR	VAR	0	0	10,11	X	X	X	
10	Positive Acknowledgment	0	16	0	18	10,11	X	X	X	X
11	Negative Acknowledgment	0	16	0	18	10,11	X	X		
12	Request STU Status	0	16	60	78	10,11,12	X	X	X	X
13	Clear STU Statistics	0	16	0	18	10,11	X	X	X	
14	Set STU Time	14	30	0	18	10,11	X	X	X	X
15	Setup to Load a User Program	12	28	VAR	VAR	10,11,22	X	X	X	
16	Load a User Program	VAR	VAR	VAR	VAR	10,11,22	X	X	X	
17	Start a User Program	6	22	0	VAR	10,11,22	X	X	X	
18	Continue a User Program	6	22	0	VAR	10,11,22	X	X	X	
19	Stop a User Program	6	22	0	VAR	10,11,22	X	X	X	
20	Go to System Loader	0	16	0	16	10,11	X	X	X	
21	User Program Message	VAR	VAR	VAR	VAR	10,11,21	X	X		X
22	RTE-C System Message	VAR	VAR	VAR	VAR	22		X	X	

VAR = VARIABLE LENGTH

Table 4. Sample Communication Sequences

	MCU	STU
Request Information:	Message →	← Response
Request Information (with error):	Message → NAK →	← Response ← Response
New Program Sequence:	Message to get ready to receive a program → Program Msg 1 → Program Msg 2 → Program Msg n → Start Program →	← ACK ← ACK ← ACK ← ACK ← ACK (Program starts running)
Request Information Requiring Multiple Messages:	Message → ACK → ACK →	← Response 1 ← Response 2 ← Response n
Request without Response:	Message → (time delay) Message → (time delay) Message sent to MCU Operator	

8. REFERENCES

Anzinger, G.A., and Gadol, A.M. (1975), A real-time operating system with multi-terminal and batch/spool capabilities, Hewlett-Packard Journal, December, pp. 12-20. (Hewlett-Packard Company, 1501 Page Mill Road, Palo Alto, Calif. 94304.)

Electronics Industries Association (EIA), Standard RS-232-C (1969), EIA Engineering Department, 2001 Eye St., N.W., Washington D.C. 20006.

IEEE (1975), IEEE Standard Digital Interface for Programmable Instrumentation, 488-1975, IEEE, 345 East 47th St., New York, New York.

DISTRIBUTION LIST

	NO OF CYS
DCA/470 Washington DC 20305	3
DCEC/R220 1860 Wiehle Avenue Reston VA 22070	4
Naval Telecommunications Command ATTN: 2633 Massachusetts Avenue, NW Washington DC 10390	
Naval Electronics System Command ATTN: 41014, 41025 Washington DC 20360	
US Department of Commerce Office of Telecommunications Institute for Telecommunications Sciences ATTN: Mr. Walt Johnson Boulder CO 80302	20
NSA/S254 Ft Meade MD 20755	2
USACEEIA/CCC-CED	2
USACEEIA/CCC-TED	2
USACEEIA/CCC-CED-XEM Ft Huachuca AZ 85613	1
AFLC/MMWE Wright-Patterson AFB OH 45433	1
AFCD/MMCY Kelly AFB TX 78241	1
AFSS/SREE Kelly AFB TX 78241	1
ESD/DCF Hamscom AFB MA 01730	3
RADC/DCLD	2
RADC/DCLF Griffiss AFB NY 13440	2
NCA/EPE Griffiss AFB NY 13440	2
ECA/EPE APO New York 09633	1
SCA/EPE Oklahoma City AFS OK 73145	1
PCA/EPE APO San Francisco 96515	1

Defense Documentation Center
Cameron Station
Alexandria VA 22314. 15
1839 EIS
EMC Measurements Division/EPEF
Keesler AFB MS 39534 1
HQ AFCS
DOY. 1
LGX. 1
LGMEE. 1
LGMSR. 1
OA 1
XPQC 1
1842 EEG
EPEC 1
EPECR/DEB. 1
EPEU 1
HQ AFCS
DAPL. 10