113 pages

NASA Grant NCC 2-41

Technology Research For Strapdown Inertial Experiment and Digital Flight Control and Guidance

FINAL TECHNICAL REPORT

(NASA-CR-176907) TECHNOLOGY RESEARCH FOR STRAPDOWN INERTIAL EXPERIMENT AND DIGITAL FLIGHT CONTROL AND GUIDANCE Final Technical Report, 1 Oct. 1979 - 31 Dec. 1984 (University of Southern Colorado, Pueblo.) G3/04

N86-29800

Unclas 43271

# School of Applied Science and Engineering Technology

UNIVERSITY OF SOUTHERN COLORADO

2200 North Bonforte Boulevard Pueblo, Colorado 81001

NASA Grant NCC 2-41

# Technology Research For Strapdown Inertial Experiment and Digital Flight Control and Guidance

FINAL TECHNICAL REPORT

NASA Grant NCC 2-41

# Technology Research For Strapdown Inertial Experiment and Digital Flight Control and Guidance

## FINAL TECHNICAL REPORT


| | |
|---|---|
| Principal Investigator: | Ralph A. Carestia |
| Co-Investigator: | Don E. Cottrell |
| Research Assistants: | Arthur A. Beckman<br>Tim R. Johnson<br>Peter J. Holub<br>Richard W. Parker |
| Period Covered By Report: | October 1, 1979 to December 31, 1984 |
| Institution: | University of Southern Colorado<br>2200 Bonforte Blvd.<br>Pueblo, Colorado 81001 |
| NASA Technical Officer: | George Callas |

# TABLE OF CONTENTS


| | Page No. |
|---|---|
| General Summary | 1 |
| Cumulative References | 6 |
| Appendices | 7 |

NASA Grant NCC2-41

# Technology Research for Strapdown Inertial Experiment and Digital Flight and Guidance

## FINAL REPORT

## General Summary

The University of Southern Colorado research engineers were involved in: (1) Investigating strapdown inertial sensor systems and (2) Studying the system requirements of digital avionics systems in rotorcraft. This included feasibility examination of digital data acquisition techniques in aircraft and the use of digital systems in rotorcraft for the purpose of specifying design criteria for integrated guidance, navigation and control. Results from the research produced three literature publications and several working reports for NASA. A chronologically ordered cumulation of these publications is provided following this summary.

The main objectives of the University of Southern Colorado's cooperative agreement at NASA-Ames were to test a Strapdown Inertial Navigation System and to provide system criteria required to design reliable cockpit avionics systems for rotorcraft. The applied research conducted was divided into three areas: (1) To develop a digital data acquisition system and flight test a Strapdown Ring Laser Gyro Tetrad Inertial Navigation System in a VH-1B helicopter. (2) To develop, design and build an operational interface for the Rotorcraft Digital Advanced Avionics System (RODAAS) and (3) To integrate and test the navigational capabilities of a Differential Global Positioning Satellite System.

The first area of research involved a helicopter flight-test program to evaluate the performance of Honeywell's Tetrad - a strapdown, laser gyro, inertial navigation system. The results of 34 flights showed a mean final navigational velocity error of 5.06 knots, with a standard deviation of 3.84 knots; a corresponding mean final position error of 2.66 n.mi., with a standard deviation of 1.48 n.m.; and a modeled mean-position-error growth rate for the 34 tests of 1.96 knots, with a standard deviation of 1.09 knots. Tetrad's four-ring laser gyros provided reliable and accurate angular rate sensing during the test program and no sensor failures were detected during the evaluation.

The second area's objective of the cooperative agreement was to develop criteria suitable for investigating cockpit systems in rotorcraft. This criteria led to the development of two basic simulators.

The first was a standard simulator which could be used to obtain baseline information for studying pilot workload and interactions. Figure 1 is a block diagram of this system. The instrument panel used was typical of a UH-1H helicopter. The mathematical model for the UH-1H was implemented in the Dec 11/23 minicomputer, while all instrument controls were implemented in the microcomputer data adapter.

The second was an advanced simulator which integrated the RODAAS developed by Honeywell into this simulator. Unlike the standard system, the RODAAS integrates an advanced set of avionics functions into hardware. Figure 2 is the microcomputer system which was used to interface to the RODAAS instrumentation.

The second area also included surveying the aerospace industry to determine the level of use and impact of microcomputers and related components on avionics systems. It was determined that the useage of microcomputers and related components is very extensive. A report was written pertaining to this and is a part of the appendix.

Figure 1 — *Functional Diagram of System*

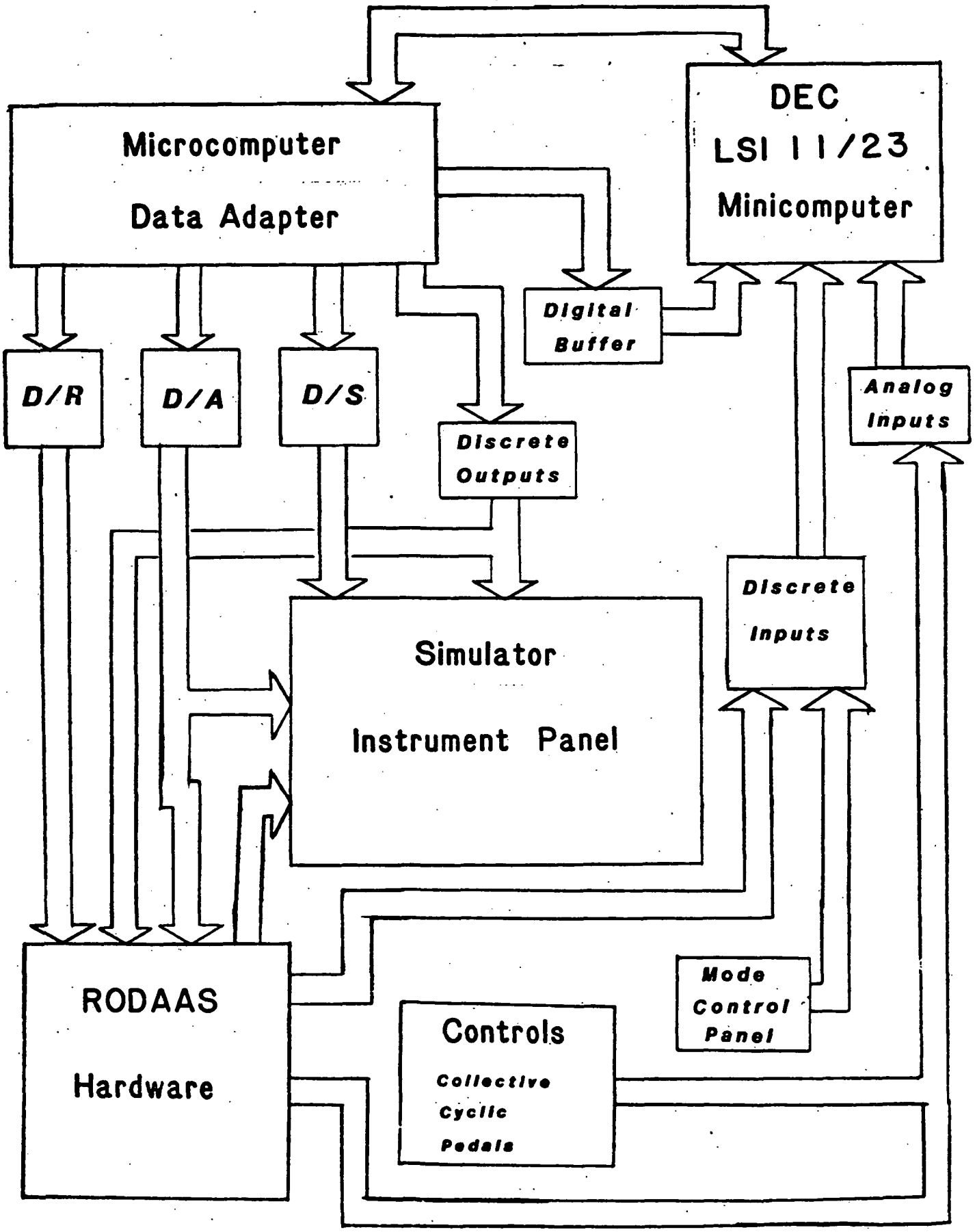
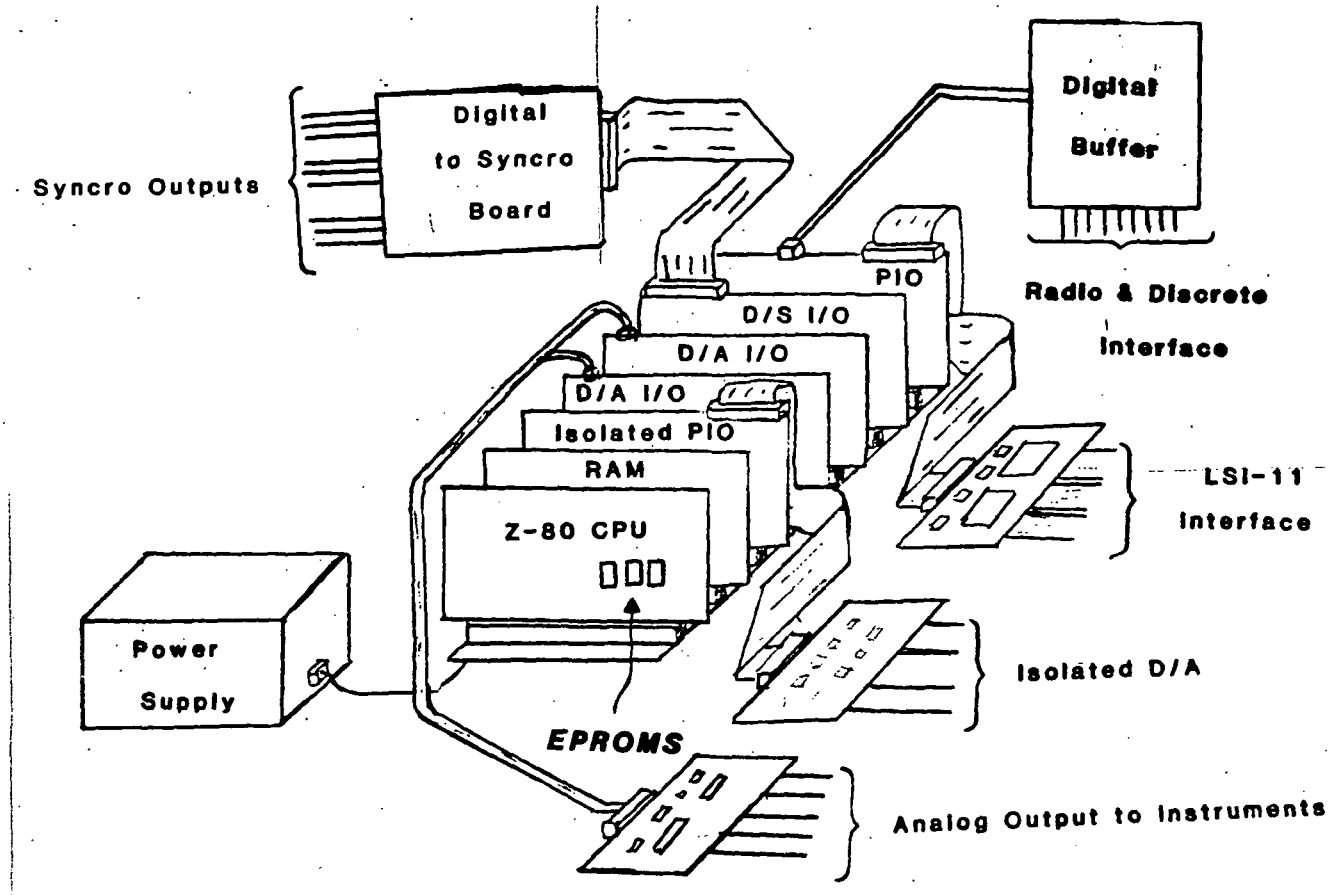


FIGURE 2



# Microcomputer System



The third area of investigation was to integrate and test the navigational capabilities of a Differential Global Positioning Satellite (Differential GPS) System, which is a prototype space-based radio navigation system developed by the Department of Defense. Work for NASA in support of the Differential GPS fell into the following areas: literature search, hardware installation in the SH3 helicopter and laboratory, technical visits to Magnavox to verify performance of a NASA interface processor (NIP) and final acceptance testing of the NIP, design and testing of software used in the NASA computer to gather data and send the data via telemetry to a ground station for recording, and support of the preliminary Differential GPS flight experiment. This report is accordingly embodied in the appendix.

## Cumulative References

1. "Guidance and Flight Control Demonstration in a Helicopter Flight Environment Using a Laser-Gyro Inertial Navigation System", S.F. Schmidt, R.A. Carestia, R.J. Hruby and L.D. Corliss, National Aerospace Meeting, NASA Ames Research Center, Moffett Field, California, March 24-25, 1982.

2. "Performance of a Strapdown Ring Laser Gyro Tetrad Inertial Navigation System in a Helicopter Flight Environment:, R.A. Carestia, R.J. Hruby, and W.S. Bjorkman, National Aerospace Meeting, NASA Ames Research Center, Moffett Field, California, March 24-25, 1982.

3. "Flight Test Results of the Strapdown Ring Laser Gyro Tetrad Inertial Navigation System", R.A. Carestia, R.J. Hruby, and W.S. Bjorkman, NASA Technical Memorandum 84358, October, 1983.

The following consists of working papers written by university researchers during the grant reporting period:

| Contents | Title |
|---|---|
| Appendix A | "A Survey of the Use and Trends of Microprocessors/Computers in the Avionics Industry", D. Cottrell, R. Carestia and P. Holub. |
| Appendix B | "Differential Global Positioning Satellite Project", A. Beckman. |
| Appendix C | "Report of Contract Work for University of Southern Colorado", T. Johnson. |
| Appendix D | "NASA Summer Project", R. Parker. |

# APPENDIX A

# A SURVEY OF THE USE AND TRENDS OF MICROPROCESSORS/COMPUTERS IN THE AVIONICS INDUSTRY

October 1984

by Dr. D. Cottrell
R. A. Carestia
P. Holub

# A SURVEY OF THE USE AND TRENDS OF MICROPROCESSORS/COMPUTERS IN THE AVIONICS INDUSTRY

October 1984

by

Don Cottrell
Ralph Carestia
Peter Holub

## ABSTRACT

In 1983, a nationwide survey was conducted of the avionics use of microprocessors and microcomputers, including their associated memories and peripheral devices. In this survey, it was determined that microprocessors and microcomputers have made a great impact upon the avionics industry. They are being used in numerous applications such as control, communications, guidance, navigation, and system monitoring. Based on this information and in light of the current state-of-the-art in computer hardware and software, this report suggests some philosophical design considerations which could influence hardware selection and attempts to predict trends in computer hardware applications for avionics systems.

TABLE OF CONTENTS


| | Page |
|---|---|
| Introduction | 1 |
| Results of Survey | 3 |
| Section I - Microprocessors | 4 |
| Section II - Memory | 12 |
| Section III - I/O Ports, MPU vs. Non-MPU and IC's | 18 |
| Section IV - Microprocessor Systems | 19 |
| Summary and Conclusions | 23 |
| Microprocessors | 23 |
| Memories | 25 |
| I/O Ports, IC's, and MPU's vs. Non-MPU's | 26 |
| Microprocessor Systems | 27 |
| References | 29 |
| Appendix (Survey Form) | 30 |

LIST OF FIGURES AND TABLES


| Figure | | Page |
|---|---|---|
| 1. | First Microprocessor Used | 4 |
| 2. | First Year of Microprocessor Use | 5 |
| 3. | MPU System Classification | 6 |
| 4. | MPU Data Size Bus | 6 |
| 5. | MPU Process Type | 7 |
| 6. | MPU Procurement | 7 |
| 7. | Assembled Module Type | 8 |
| 8. | MPU Applications | 8 |
| 9. | Microcomputer System Application | 9 |
| 10. | MPU Buying Decision Factors | 10 |
| 11. | Microprocessor Architecture | 10 |
| 12. | Advanced Architecture Use | 11 |
| 13. | Maximum System Memory Access Time | 13 |
| 14. | Memory System Requirements | 14 |
| 15. | Memory System Requirements | 15 |
| 16. | Memory Buying Decision Factors | 16 |
| 17. | External Data Storage | 17 |
| 18. | Key Discrete Devices | 19 |
| 19. | System Development Tools | 19 |
| 20. | High Level Languages | 20 |
| 21. | System Throughput | 21 |
| 22. | Maximum System Power | 21 |
| 23. | Redundancy Type | 22 |

| Table | | |
|---|---|---|
| 1. | I/O Ports Supported by the Microprocessor | 18 |
| 2. | % ICS Eliminated | 18 |

## INTRODUCTION

Many aircraft functions including control, navigation, guidance, communications, and system monitoring are being implemented reliably and accurately with the aid of the digital computer. The use of digital computer-based avionics equipment represents a potentially high cost for modern aircraft, thus a desirable goal for avionics systems is to find ways to minimize cost and still retain high system reliability. With the advent and increasing use of the microprocessor in recent years, this goal is being met.

Due to the advanced nature of the NASA/AMES flight research program, the selection of any avionics equipment must consider the types of systems which will be representative of the post 1985 time period. Selection is complicated by the fact that both digital systems technology, and the applications of microprocessors in these systems, are changing rapidly.

In late 1982, the University of Southern Colorado decided to conduct a nationwide survey of avionics engineers on their use of microprocessors and the associated peripheral devices, such as memory, in avionics systems. Since NASA had recently demonstrated the use of microprocessors in an advanced avionics system, the University and NASA concurred it would be desirable to know what level of microprocessor integration exists in the avionics industry. Hence, this survey report considers current state-of-the-art avionics computer hardware and microprocessors, suggests some philosophical design considerations which could influence computer hardware selection, and attempts to predict trends in computer hardware applications for avionics systems in the middle and late 1980's.

A questionnaire was developed by the University in late 1982 and early 1983. It was mailed and received by 168 avionics engineers identified as being directly involved with avionics system design in the spring of 1983 and again as a follow-up in the fall of 1983. To allow time for the maximum number of responses, analysis of the data did not begin until January of 1984. The number of questionnaires returned was 64, which constituted a 38% response.

This report should be valuable to the systems engineer in designing avionics systems because it includes detailed data on the types of microprocessors, memories, microprocessor applications, and programming languages presently used or anticipated. A copy of the survey form is included in the Appendix.

## RESULTS OF SURVEY

The data, graphs, and tables are presented in this section according to the survey form given in the Appendix. There are twenty-eight questions arranged in the following outline form:

Section I: Microprocessors

- A. First type used
- B. When first used
- C. Plans for purchase
  - 1. Type (Standard, dedicated or custom)
  - 2. Bit size (4, 8, 12, 16, over 16)
  - 3. IC type (CMOS, NMOS, etc.)
- D. Procurement and purchasing
- E. Applications
- F. Architecture
  - 1. Performance
  - 2. Standard or advanced

Section II: Memory

- A. Type
  - 1. Semiconductor
  - 2. Bubble
- B. Access time
- C. Total memory required
  - 1. 1982
  - 2. 1983
- D. Purchasing Factors
  - 1. Speed
  - 2. Performance
  - 3. Price
  - 4. Others
- E. EPROM to mask, ROM capability
- F. External data storage

Section III: Miscellaneous

- A. I/O ports
- B. Critical discrete devices and IC's for MPU
- C. Does MPU system replace a current non-MPU based system?

Section IV: Microprocessor Systems

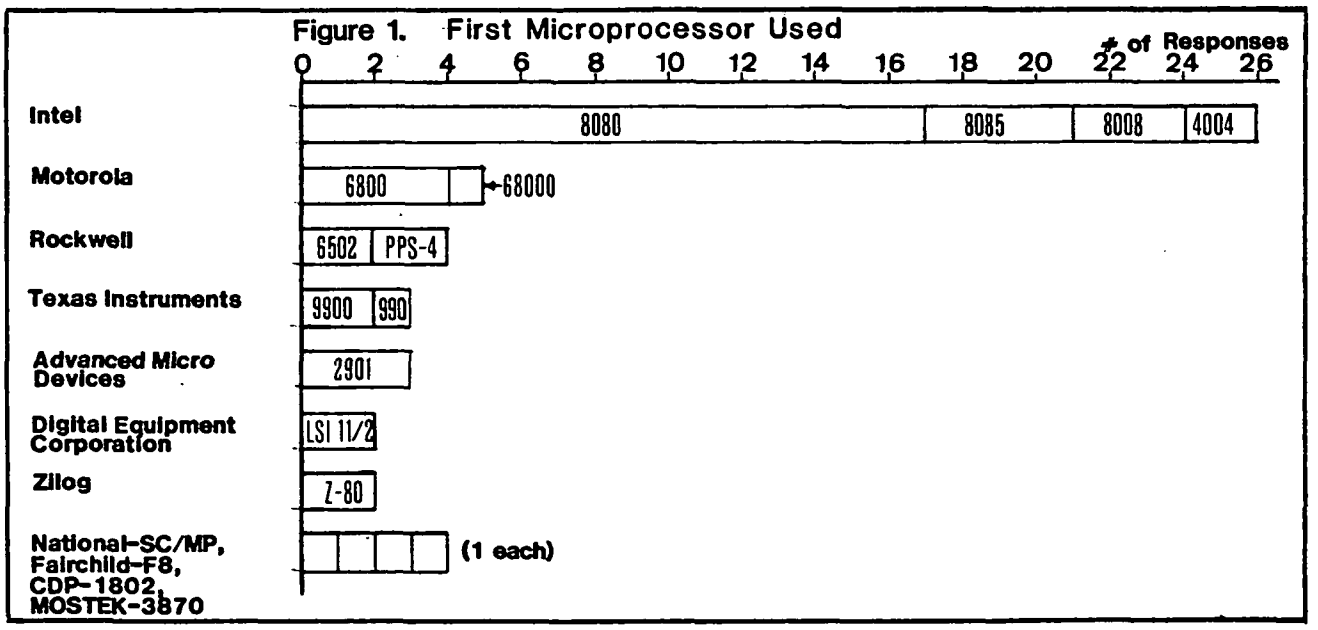
A. Will MPU system use system development tools and which ones?
B. Importance of high level languages
C. Throughput and maximum power of microcomputer system.
D. Does MPU use "Built-In-Test or redundancy?

1. What type?
2. Time critical?
3. Synchronous or asynchronous?
4. Failure detection threshhold?

## SECTION I - MICROPROCESSORS

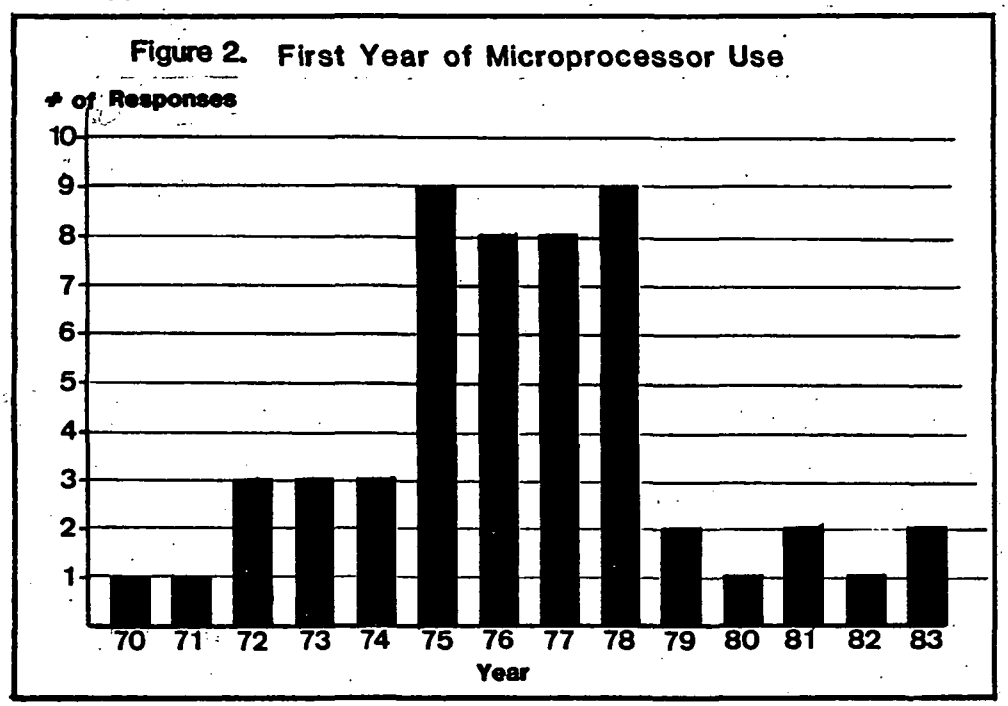
Q 1 - What was the first microprocessor you used?

The bar graph of Figure 1 shows the results. The greatest number of respondents (26) selected Intel's microprocessors with the Intel 8080 having the highest frequency of use in avionics possibly because it was one of the first available. Mororola's 6800 was second, with five responses. Other types of microprocessors used include those manufactured by Rockwell, Texas Instruments, Advanced Micro Devices, Digital Equipment Corporation, Zilog, National, Fairchild, and Mostek.



Q 2 - What year did you first use microprocessors?

This data is given in Figure 2 and indicates that most avionics specialists first began using the microprocessor between the years 1975-1978. This period coincides with the introduction of 8 bit microprocessors. Responses ranged from as early as 1970 to as late as 1983.




Q 3 - In 1983, have you purchased or do you plan to purchase MPU systems?

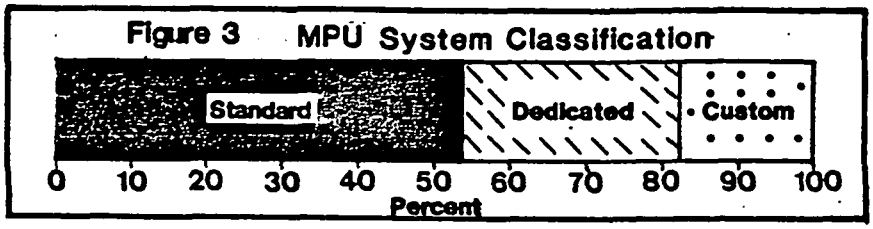
91% answered yes, 7% answered no, and 2% of the respondents did not answer the question.

Q 3A - If yes, will they be standard, dedicated, or custom?

Figure 3 shows the results:

1. 55% are standard MPU's.
2. 29% are dedicated MPU's
3. 16% are custom MPU's.

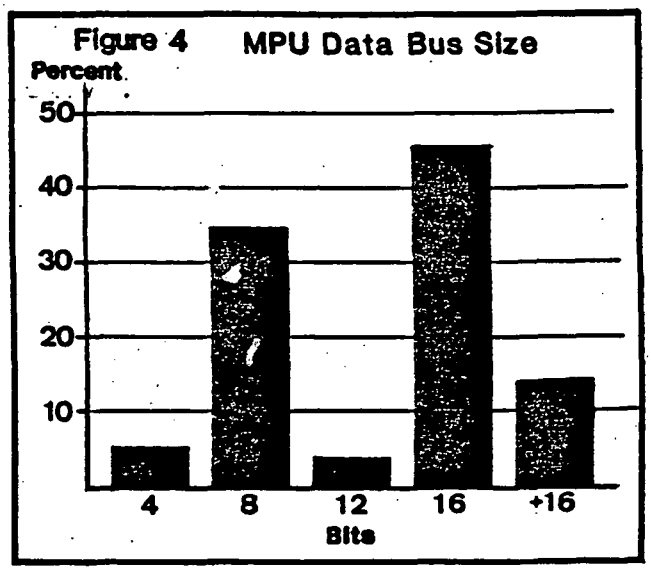
It is likely that the majority use standard MPU's because of their flexibility for system design.




Q 3B - Have they been or will they be 4 bit, 8 bit, 12 bit, 16 bit, or over 16 bit?

The results of this question are given in Figure 4. The 16 bit MPU is the most popular, followed by the 8 bit MPU. The over 16 bit MPU is at 13% of the total, which might indicate its growing popularity.

ORIGINAL PAGE IS
OF POOR QUALITY




Q 3C - Will the MPU's be IIL, Schottky, CMOS, NMOS, or PMOS?

As shown in Figure 5, NMOS and CMOS lead the field at 36% and 34% respectively; the others range from around 5% to 9%. NMOS devices have been popular and are in high production by manufacturers. They are also usually faster than PMOS MPU's. CMOS may be slower than NMOS or PMOS; they are low power devices, and are relatively unaffected by variations in power supply voltage (NMOS MPU's require well-stabilized power supplies). The IIL and Schottky devices are normally chosen where high speed is essential. Their high power requirements may be the reason they have limited use in avionics systems.

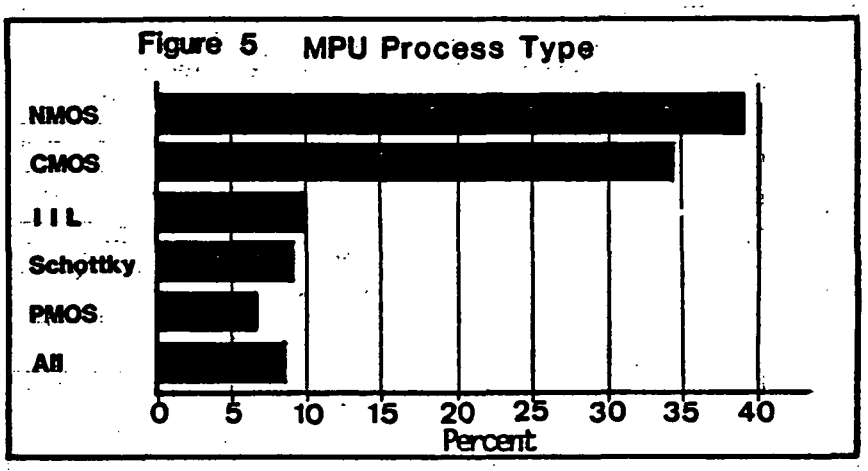



Figure 5 MPU Process Type

Q 4A - How will you procure your MPU'S?

As illustrated in Figure 6, this question breaks down as follows:

| | |
|---|---|
| As chips alone for in-house design (components) | - 39% |
| As chips on boards (sub systems) | - 20% |
| As assembled modules | - 20% |
| Complete MPU System (includes memories and power supplies) | - 21% |

It appears that the greater percentage of respondents preferred to do their own in-house design using MPU chips. This corresponds with the results of Question 3A.

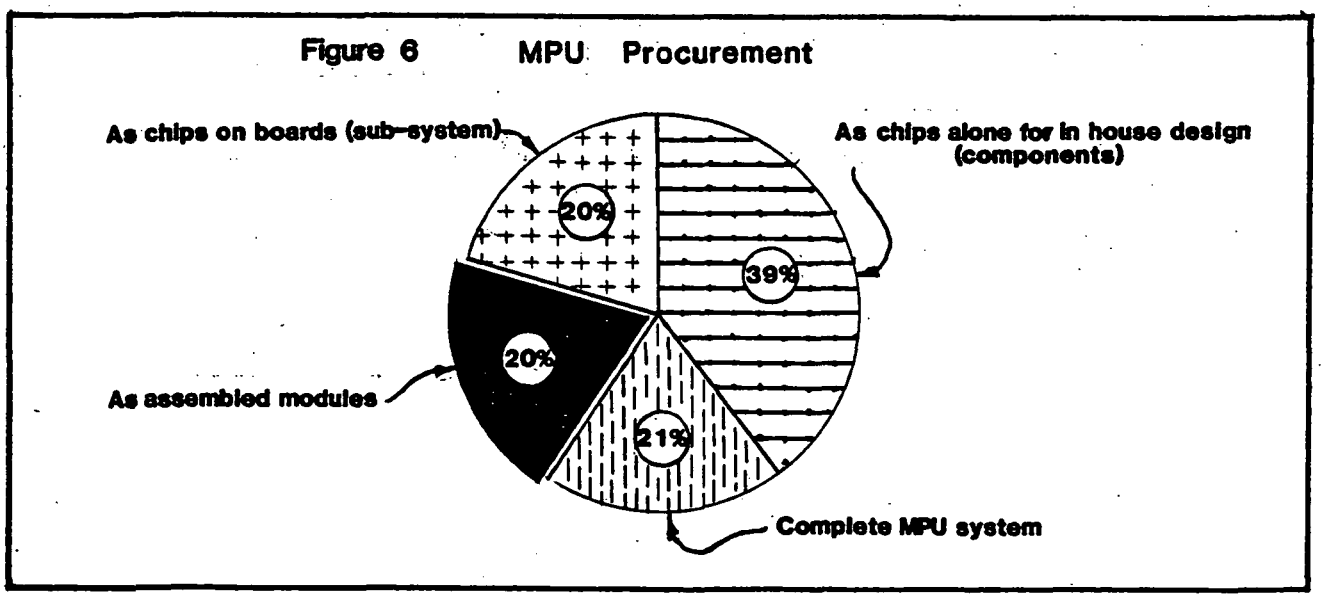


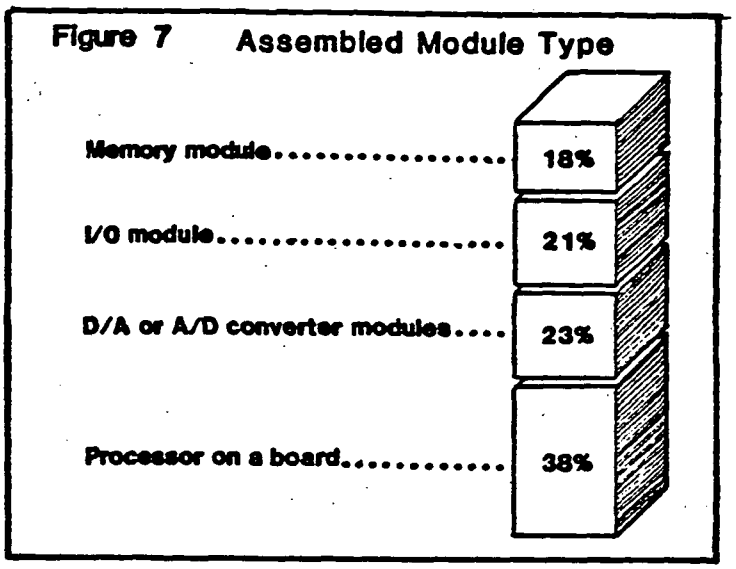

Figure 6 MPU Procurement

ORIGINAL PAGE IS
OF POOR QUALITY



Q 4B - If assembled modules, which ones do you contemplate buying?

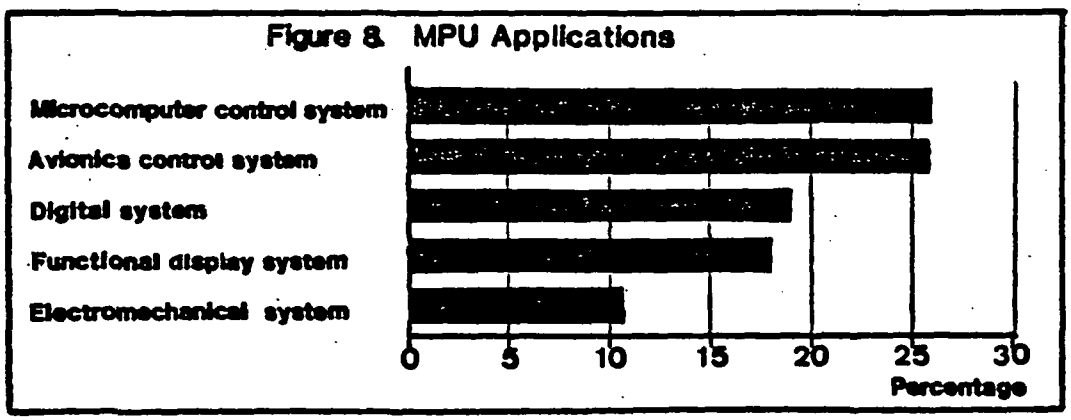
As shown by Figure 7, the largest percentage of designers, 38% contemplated buying the processor-on-a-board.




Q 5 - Does your MPU application relate to:

Digital System?
Electromechanical System?
Microcomputer Control System?
Avionics Computer System?
Functional Display System?

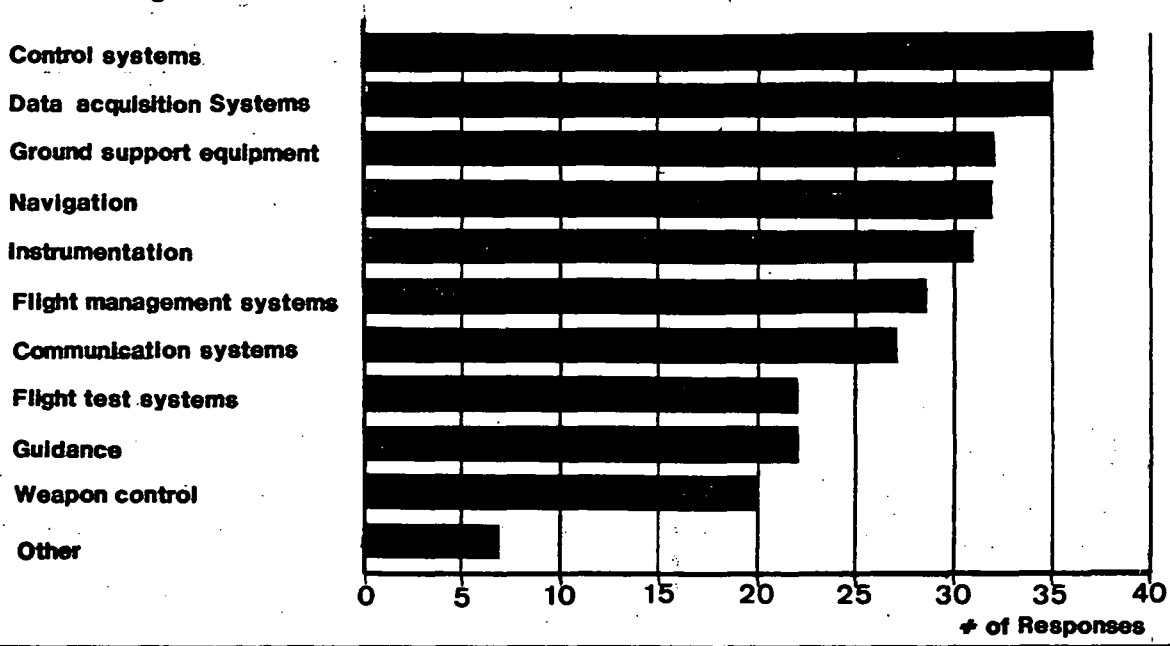
The results, given by Figure 8, indicate that microcomputer control systems and avionics control systems have the highest application of MPU's at 26% each.



Q 6 - In what applications do you use your microcomputer systems?

Figure 9 shows a fairly even distribution of applications. Control systems and data acquisition systems had the highest number of responses at 37 and 35 respectively. Most of the other applications were fairly closely grouped, ranging from 20-32 responses. Other applications had only 7 responses.

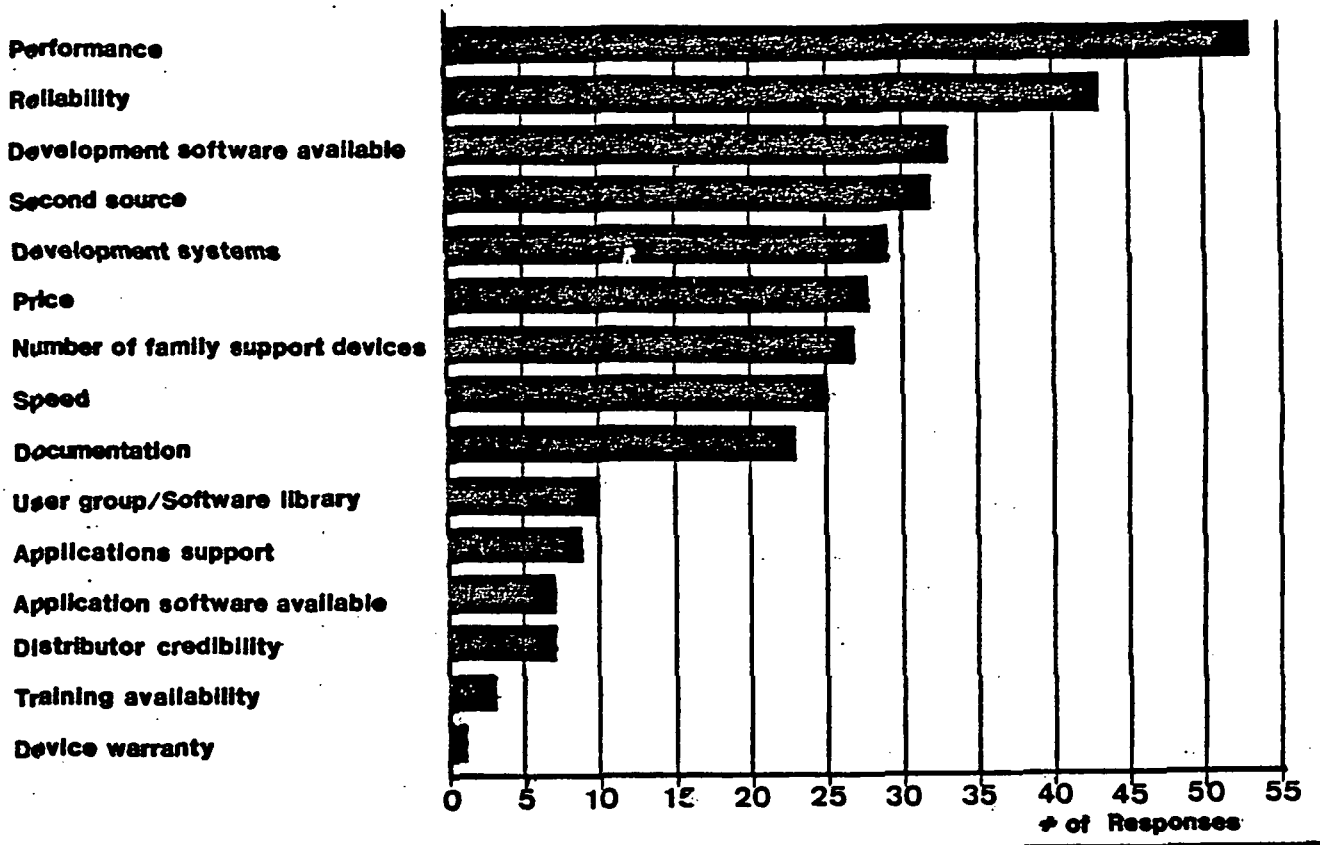
Figure 9 Microcomputer System Application




Q 7 - Which of the following factors do you feel are the most important in your company's MPU buying decisions?

As indicated by Figure 10, 15 different factors, such as price, performance, speed, etc., were considered. Performance was the factor most cited, followed by reliability and development software availability, as well as a second source. The other factors fell in a range of 1-28 responses.

Figure 10 MPU Buying Decision Factors



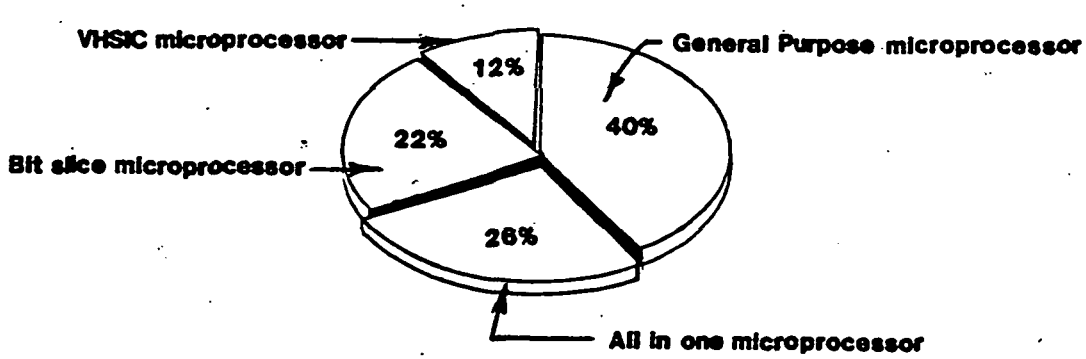

Q 8 - Is the microprocessor architecture a:

general purpose microprocessor?
all in one microprocessor?
bit slice microprocessor?
VHSIC microprocessor?

ORIGINAL PAGE IS
OF POOR QUALITY

The pie chart of Figure 11 points to the general purpose microprocessor at 40% as the most popular architecture. The other less popular types of architectures range from 12% to 26%, but comprise a significant percentage of the total.

Figure 11 Microprocessor Architecture



Q 9 - Did the microprocessor perform as expected?

89% answered yes, 9% indicated no, and 2% gave no answer.

Q 10 - If not, tell why.

Generally, the MPU's did perform as expected; if they did not, then the major reason given was that the MPU was not fast enough:

| | |
|---|---|
| Not fast enough | - 67% |
| Limited I/O ports | - 17% |
| Limited interrupt capability | - 17% |
| Limited memory addressing capability | - 0% |
| Other | - 0% |

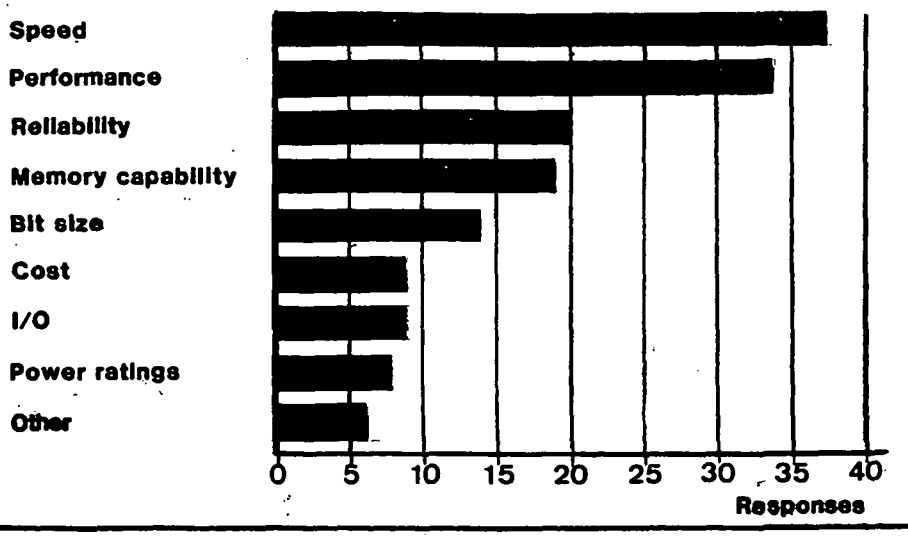
Q 11A- Will you be using advanced architectures such as bit slice, Z8000, 68000, etc., in the future?

91% of all respondents said yes to this question, while 7% said no, and 2% gave no answer.

Q 11B- If yes, what general application and why?

41% of all respondents planned to use advanced architectures for avionics flight control and management systems; the rest did not indicate their application. The reasons for use of advanced architectures included speed, performance, reliability, memory capability, etc., in that order. See Figure 12.

Figure 12 Advanced Architecture Use



Q 11C - If no, why not?

Only 3 respondents cited cost and 2 cited power ratings as reasons for not using advanced architectures. None of the respondents selected any of the other reasons listed in the survey.

## SECTION II MEMORY

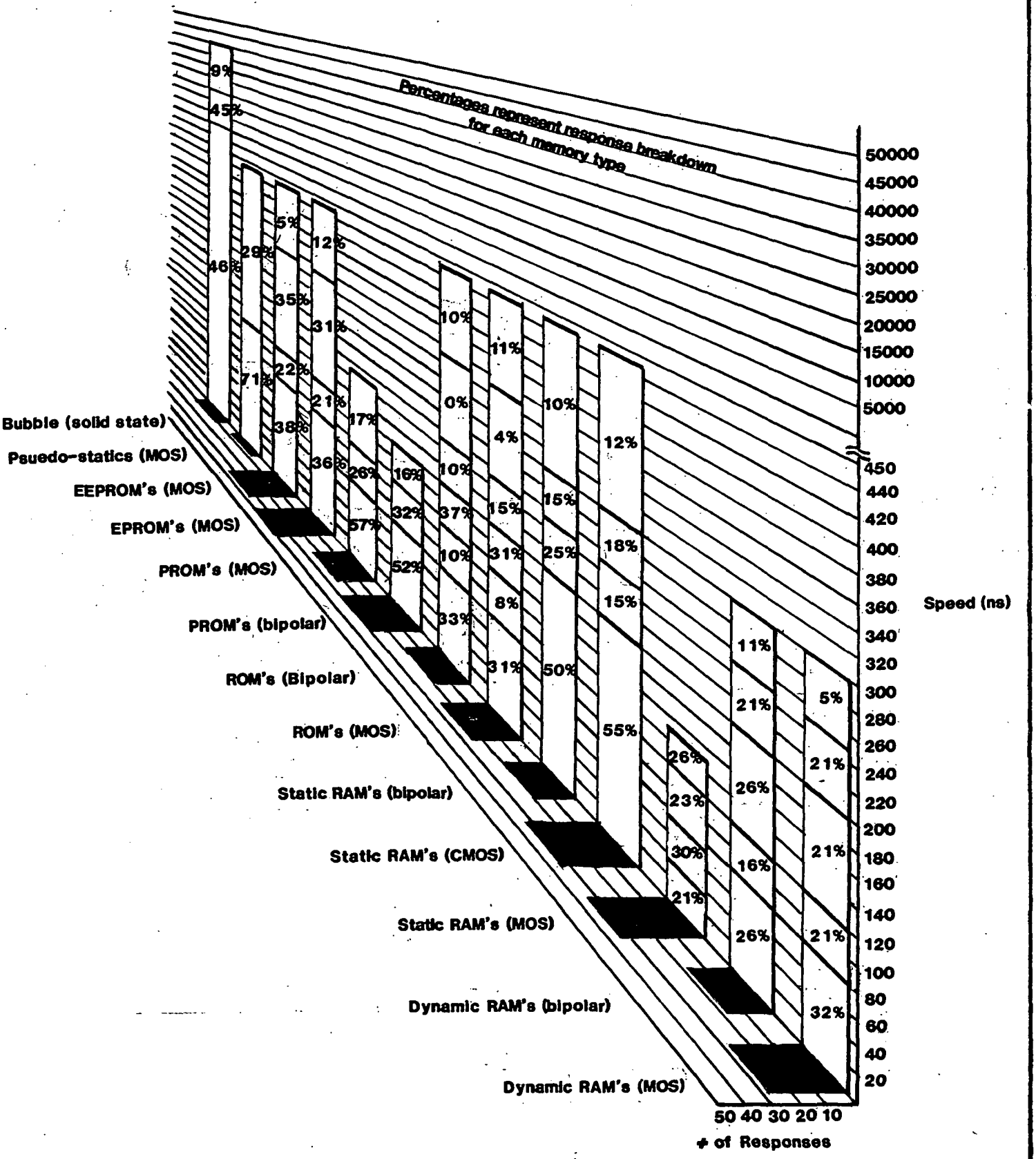
Q 12 - Have you used or do you plan to use semiconductor memories and/or bubble memories during 1982 or 1983?

The largest percentage by far, 84%, planned to use semiconductor memories, and only 14% planned to use bubble memories. Two percent of all those surveyed planned to use neither.

Q 13 - What speed (maximum access time in ns) is best suited for your system? (Check all that apply.)

For the various kinds of memories, such as dynamic RAM's, static RAM's, ROM's, PROM's, EPROM's, and bubble, the results are shown in Figure 13. For example, for CMOS static RAM's, 55% of the respondents indicated 20 to 200 ns speeds as being most suited to their system. Fifteen percent desired speeds of 200 to 250 ns, eighteen percent wanted speeds of 250 to 300 ns, and twelve percent felt 300 to 450 ns speeds would be best. The higher speeds of 20 ns to 200 ns are the most necessary in most situations, while the lower speeds ranging from 300 ns to 5000 ns are the least suited.

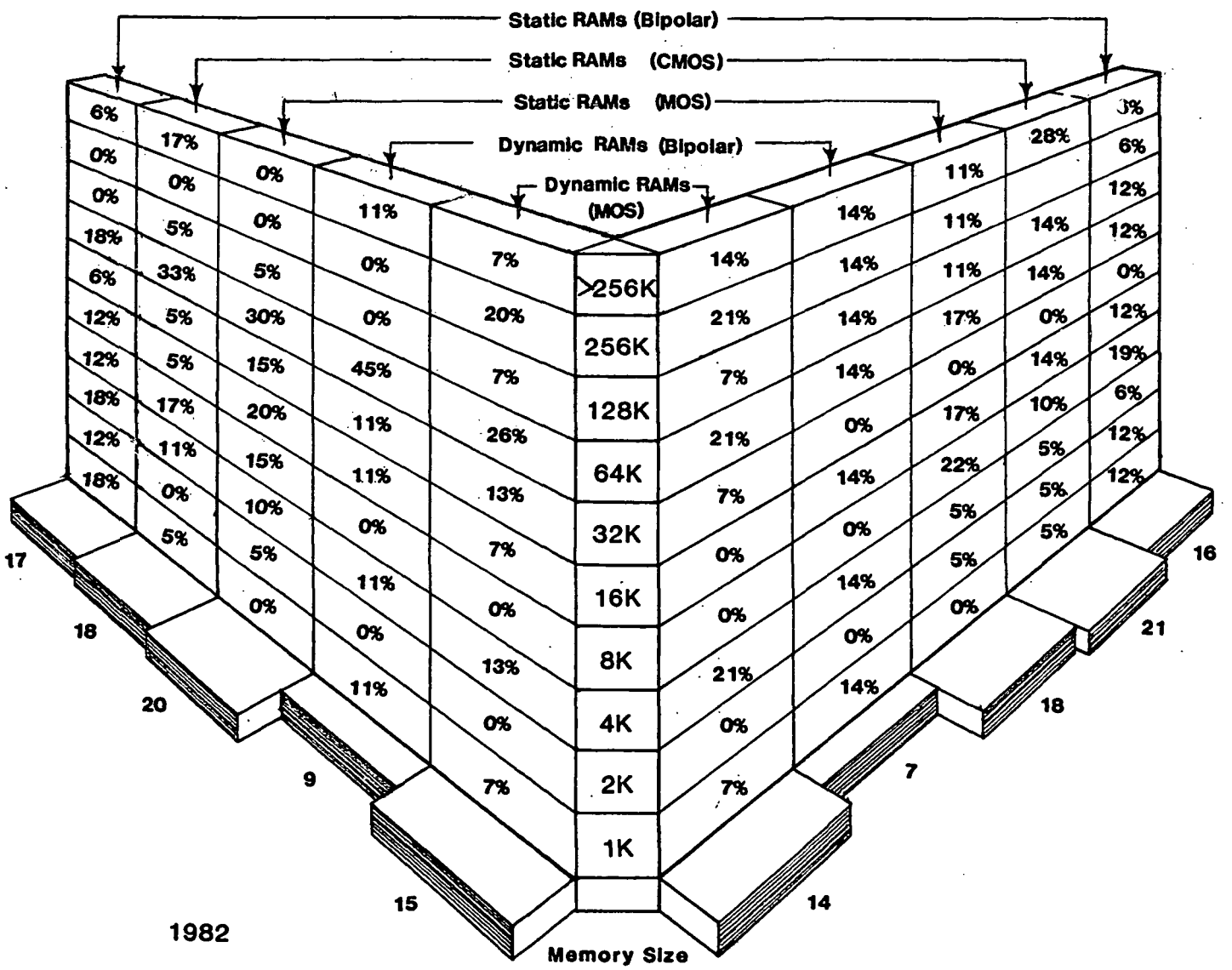
Figure 13. Maximum System Memory Access Time



Q 14 - How much total memory will your system require in 1982 and 1983? (Bytes)

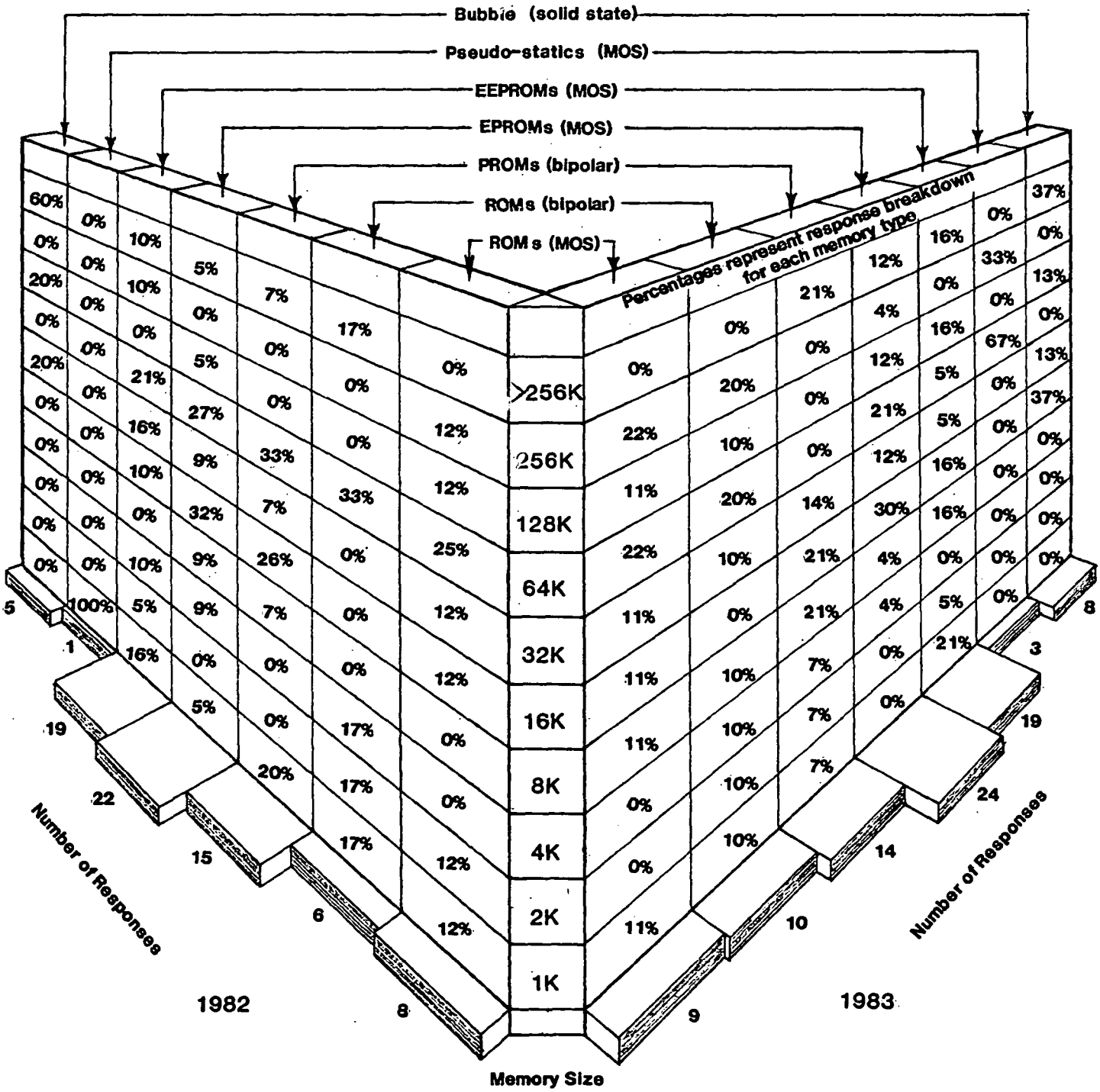
As shown by Figures 14 and 15, the 64K memory has the largest percentage of use in most categories, including Dynamic RAM's, Static RAM's, ROM's, PROM's, etc., in both 1982 and 1983. There is a slight trend in the usage of more memory (128K, 256K, and > 256K) between 1982 and 1983. EPROM's (MOS), and Static RAM's (MOS) had the greatest number of responses in 1982, whereas in 1983, EPROM's (MOS) and Static RAM's (CMOS) generated the largest number of responses.

Figure 14. Memory System Requirements




-Note: Percentages represent response breakdown for each memory type

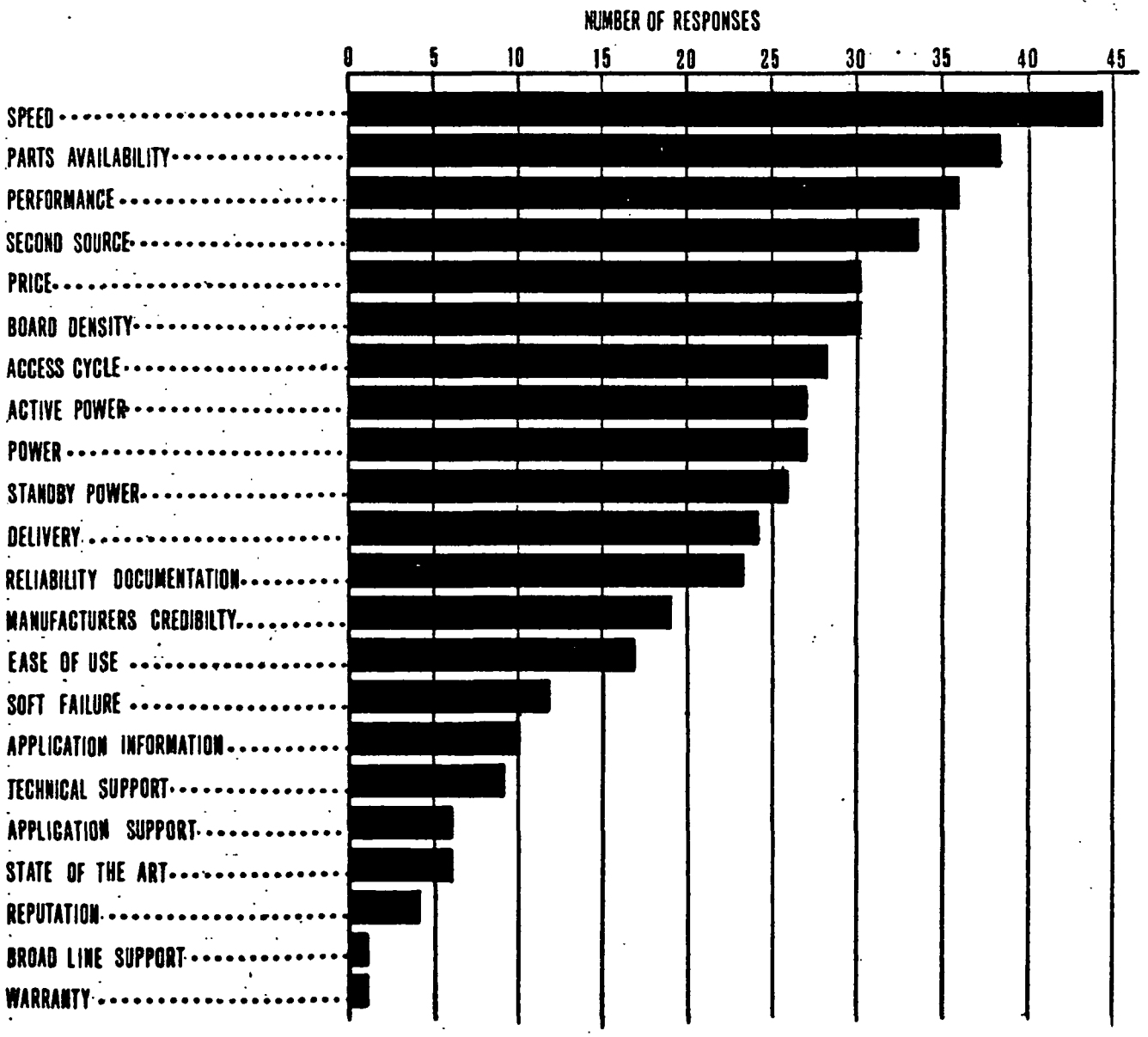
Figure 15. Memory System Requirements



Q 15 - Which of the following factors do you feel is important in your memory buying decisions? (Check all that apply.)

All factors are listed on the bar graph of Figure 16. The response distributions range from 1 to 44 with speed of the memory at 44 responses being the factor considered to be most important, followed by parts availability and performance, which are nearly equal. State-of-the-art, application support, warranty, broad line support, and reputation had the fewest responses and were thought to be the least important.

**Figure 16. Memory Buying Decision Factors**



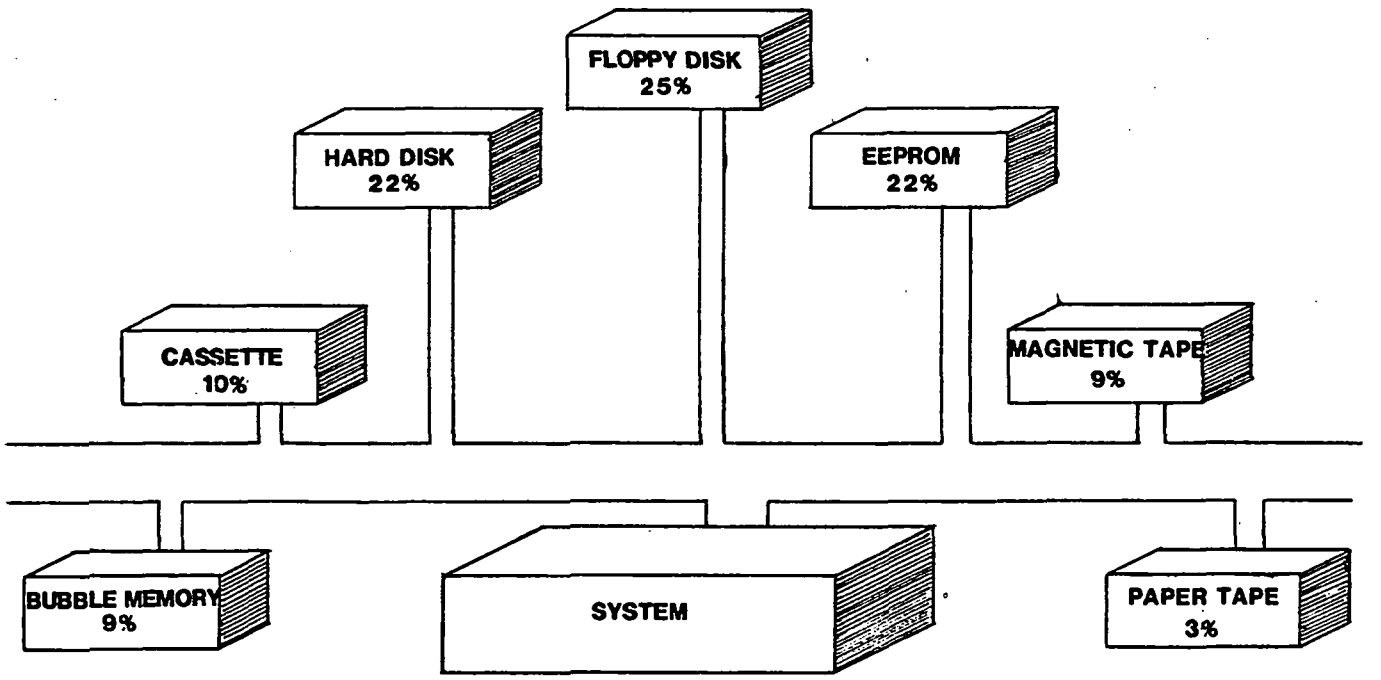
Q 16 - In your system, how important is EPROM to Mask ROM capability?

35% said they will not use EPROM without pin compatible mask ROM. However, a larger number, 47%, indicated compatibility was not important. Eighteen percent of the respondents gave no answer.

Q 17 - Does the system incorporate any external data storage?

The results of this question are presented in Figure 17; it shows that the floppy disk (25%) closely followed by the hard disk (22%) and EEPROM (22%) are the most often used for external data storage. These three comprise 69% of the total and cassettes (10%), paper tape (3%), magnetic tape (9%), and bubble memory (9%) comprise the remainder (31%).

Figure 17. External Data Storage



SECTION III - MISCELLANEOUS - I/O Ports, etc.

Q 18 - How many I/O ports does the microprocessor support?

The respondents indicated the following in Table 1:

| No. of Ports | % of Total Respondents |
|---|---|
| 0 - 10 | 55 |
| 10 - 100 | 27 |
| Over 100 | 18 |

TABLE 1
I/O PORTS SUPPORTED BY THE MICROPROCESSOR

So, the majority of respondents used ten or fewer ports in their applications.

Q 19A - Does your MPU-based system replace a current non-MPU based system?

39% of those surveyed indicated yes and 61% said no.

Q 19B - If yes, how many discrete IC's or other semiconductor elements have been eliminated?

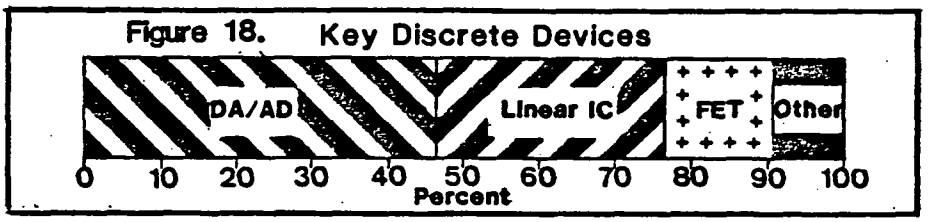
Table 2 indicates that of the respondents (13) who answered this question, 61% said they eliminated over 200 IC's, 32% said they eliminated 20-100 IC's, and 7% indicated only 0-10 IC's. The other categories had none.

| No. of IC's Eliminated | % Respondents |
|---|---|
| 0 - 10 | 7 |
| 10 - 20 | 0 |
| 20 - 100 | 32 |
| 100 - 200 | 0 |
| Over 200 | 61 |

TABLE 2
% IC's ELIMINATED

Q 20 - What discrete devices or IC's are critical or key to your MPU systems?

As shown by Figure 18, the most critical is the DA/AD converter (46% of total), followed by linear IC's (30% of total), FET's (15%), and other (9%).



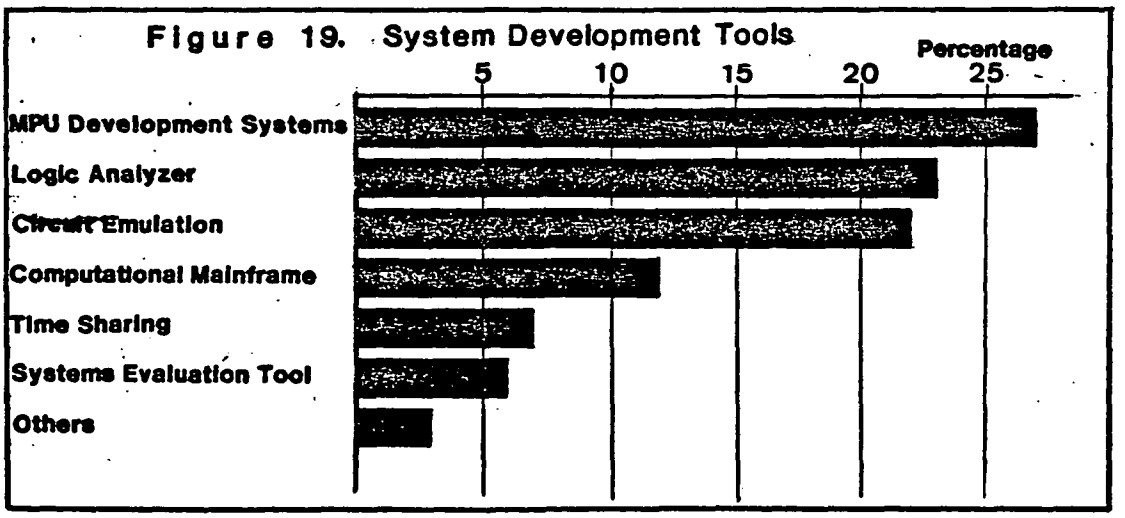

SECTION IV - MICROPROCESSOR SYSTEMS

Q 21A - Will you use system development tools that are marketed by microprocessor manufacturers?

By far, the majority (88%) said yes while only 12% said no.

Q 21B - What specific development tools do you/will you use?

Figure 19 shows that 27% of the respondents plan to use a MPU development system, 23% will use a logic analyzer, and 22% will utilize circuit emulation. Use of other tools, such as computerized main frame, time sharing, etc., ranged from 3% to 12% of the total.

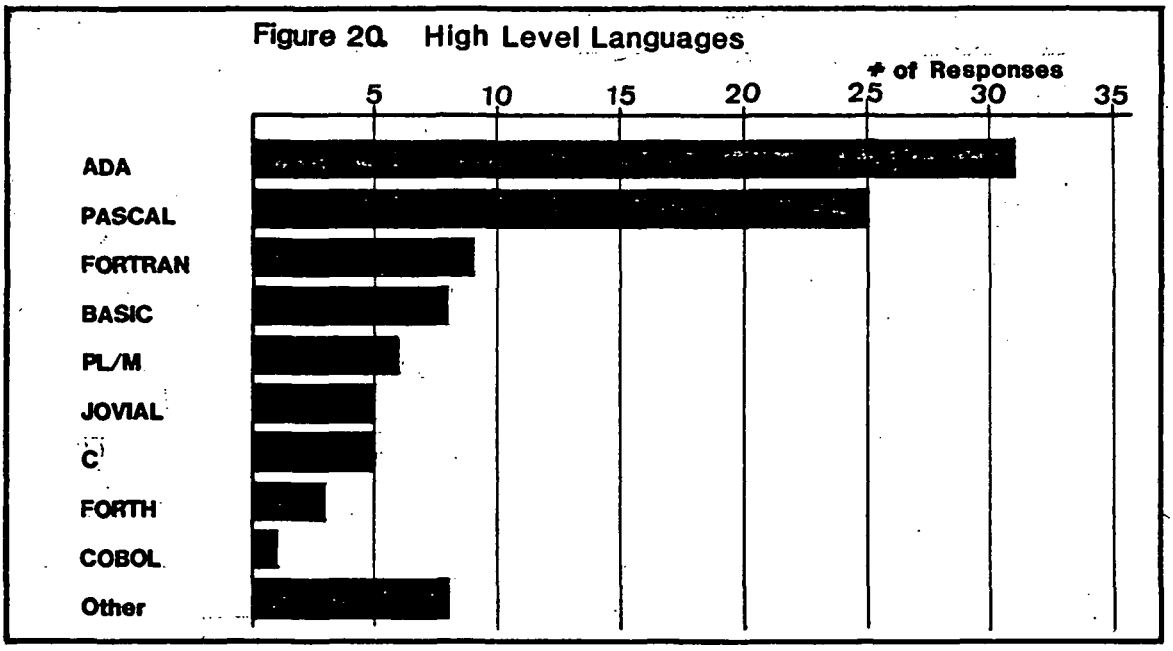


Q 22A - How important is availability of high level languages?

The majority, 66%, felt it was very important to have high level languages available, 26% said it was somewhat important, and only 8% indicated it was not important.

Q 22B - If a high level language is important, which do you favor?

As illustrated in Figure 20, ADA (the high level language suggested as a government standard) had the greater percentage of popularity, PASCAL was a strong second, FORTRAN and BASIC were nearly identical, while the others (JOVIAL, C, etc.) were less popular.



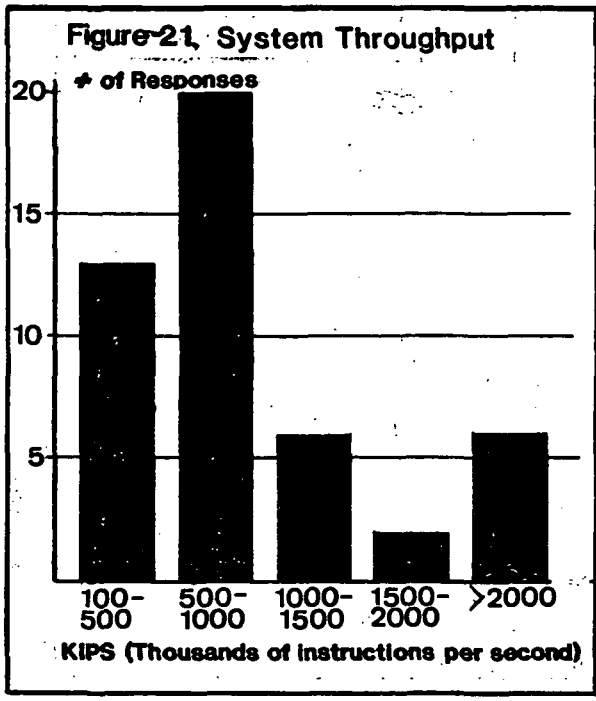

Q 23 - What is the throughput and the maximum power of your microcomputer systems? (Please specify the general application of the system.)

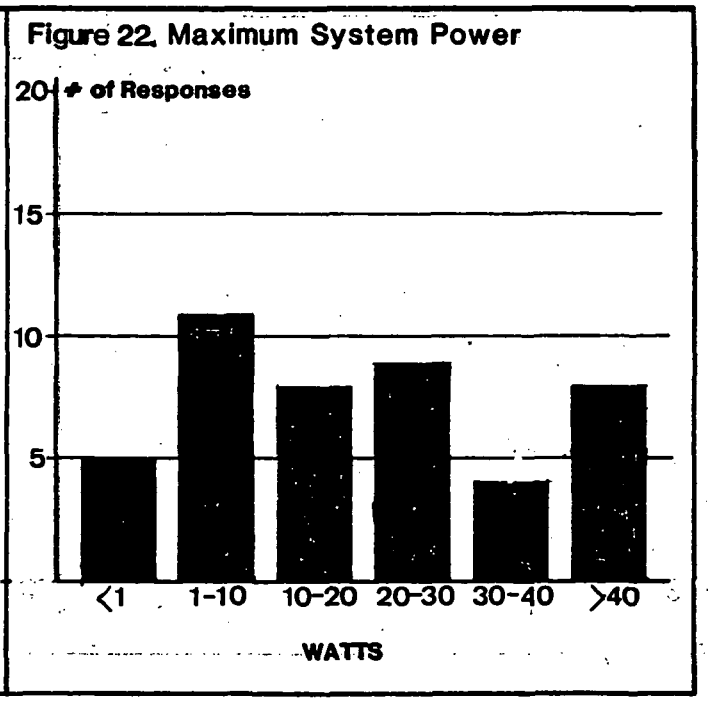
Of the 32 respondents answering this question, 19 or 59% were using their MPU systems for applications directly related to flight control systems. The majority of the remaining respondents' applications were indirectly related to avionics.

Figures 21 and 22 indicate the KIPS (thousands of instructions per second) and maximum power in watts for the various applications.

The largest percentage of KIPS was in the 500-1000 range at 41% of all responses, and the next highest percentage of KIPS was in the 100-500 range at 28% of the total responses.

Power requirements were distributed fairly evenly from less than 1 to greater than 40 watts with the 1-10 watt range having the highest value at 25% of all responses.




Q 24 - Does the microcomputer system have built-in-test fault?

72% of the respondents answered yes, while only 28% answered no. This apparently indicates that most designers prefer this feature.

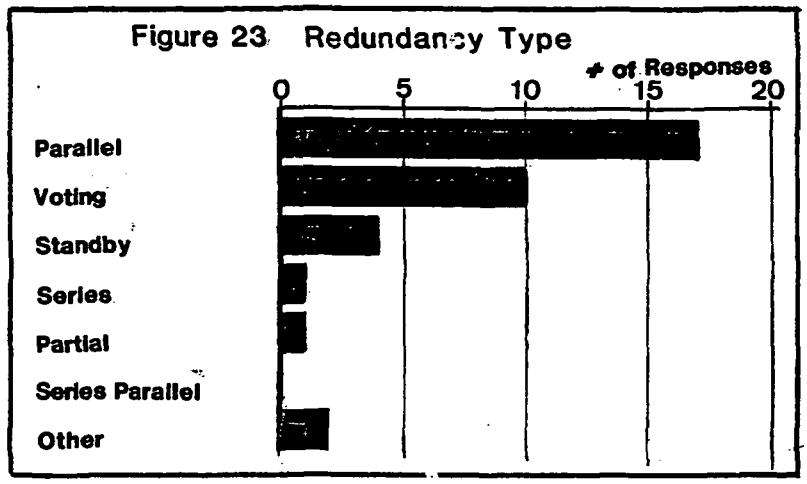
Q 25 - Do you use redundancy in the microcomputer systems you are designing?

The answer yes and no were approximately at the same percentage as follows: 53% said yes and 47% said no.

Q 26 - If yes, what types of redundancy do you use? (Please specify the general application of the system.)

As noted in Figure 23, 17 of those responding chose parallel redundancy. The second largest was voting redundancy with 10 responses.

[Most applications were listed for flight control systems and avionics.]




Q 27 - Are the microcomputers used in flight systems where failures are time critical?

74% said yes and 26% said no to this question.

Q 28 - In multiple microcomputer systems, are the microcomputers operated synchronously or asynchronously in your redundant application?

The answers to this question were distributed almost evenly: 48% indicated synchronous operation and 52% said the operation was asynchronous.

## SUMMARY AND CONCLUSIONS

### Microprocessors:

The results of this survey definitely support the assertion that microprocessors/microcomputers have a high frequency of use in avionics systems and have already replaced many of the minicomputers of the 1970's. This is not surprising since microprocessors have been available since the early 1970's. Due to their versatility, low cost, small size, and low power consumption they have been widely accepted by avionics engineers. In line with this, probably because they were the first to make microprocessors commercially available, Intel's microprocessor chips are the most often used in the avionics industry. The next most often used were Motorola's 6800 and 68000. Most avionics specialists started using the microprocessor between the years 1975-1978, but some started as early as 1970 and others as late as 1983.

The majority of the respondents (55%) purchased standard microprocessors and nearly everyone (91%) planned to buy microprocessor systems in 1983. There is also a trend to use microprocessors of 16 bits or more.

NMOS and CMOS (MPU) microprocessors are the most popular in avionics. The NMOS MPU device is usually faster than the PMOS MPU. The NMOS MPU's require well-stabilized power supplies, whereas the CMOS MPU is a low power device and relatively unaffected by variations in power supply voltage. The IIL and Schottky devices were normally chosen where high speed is essential, but require more power, which may explain why their use is limited in avionics systems.

The greater percentage of respondents (39%) preferred to do their own in-house design using MPU chips. Of those who would choose to purchase already assembled modules (20%), most of them would select the processor-on-a-board.

Applications of the MPU to avionics vary greatly, but the microcomputer control system and avionics computer system were generally the most prevalent. More specifically, control systems, data acquisition systems, navigation, ground support equipment, instrumentation, flight management systems, communication systems, guidance, flight test systems, etc., are the applications listed in order of highest use.

Performance and reliability appear to be the highest priorities in company MPU buying decisions. Secondary buying considerations include price, speed, documentation, availability of a second source, development systems, and software availability.

89% of the respondents indicated that their microprocessors typically performed as expected. If they did not, the reason most cited was that the MPU was not fast enough. With respect to MPU architecture, the general purpose MPU has the most popular usage, but the all-in-one MPU (which is a computer on a chip) and the bit slice MPU (MPU's connected in cascade to allow larger size words) together comprise the larger percentage. The VHSIC (very high speed IC) usage at this time is limited, although this will surely change with greater availability in the future.

Advanced architectures are certainly on the horizon for avionics with new chip designs such as the Z8000 and M68000 becoming available. Most

avionics specialists (91%) said they would be using these in applications relating to flight control and flight management systems.

Microprocessor manufacturers have been able to design a large variety of microprocessors and these permit many avionics applications. Also, the MPU is well-suited to the modular concept, which allows greater flexibility of system design. At present though, there is not enough standardization of most microprocessor units between manufacturers and more needs to be done.

In recent years, the cost distribution of computer system components, such as the CPU, memory, and peripheral devices, has changed drastically. In the early 1970's, the cost of minicomputer systems was in the $10K-20K range with the CPU being the major expense. Later, system prices for computers based on LSI technology were in the $1K-5K range and memory and peripheral devices had taken over as the major cost factors. Today that has been accelerated by the microprocessor, which is very inexpensive compared to memory and peripheral devices.

Memories:

Memories are a major portion of the cost of avionics computers, but there is a trend toward lower cost. Currently memories are basically of two types: (1) semiconductor storage and (2) magnetic storage. The semiconductor type is high speed and low capacity (although capacities have improved) when compared to magnetic storage which is low speed and high capacity. A typical semiconductor memory might have 64K bytes of storage capacity with a 200 ns access time, while a magnetic memory might have 1 megabyte of storage, but only 5000 ns access time (as in a bubble memory).

Speed was cited as very important in memory buying decisions. Probably because of this, semiconductor memory was chosen as most popular for avionics systems. 84% planned to use semiconductor memories, but only 14% planned to use bubble memories (2% planned to use neither). The higher speeds (maximum access times) of 20 ns to 200 ns are the most prevalent in avionics, while the lower speeds ranging from around 300 ns to 5000 ns are the least utilized.

The total memory that avionics systems require presently centers around the 64K capacity which had the largest usage, but there is a trend towards the larger memories of 128K, 256K, and greater.

If a system incorporates any external data storage, it would typically be a floppy disk, hard disk, or EEPROM. Cassettes, magnetic tape, and bubble memories are not typical for external data storage.

I/O Ports, IC's, and MPU's vs. Non-MPU's:

Slightly more than half of the designers used ten or fewer I/O ports in their applications, whereas around one-quarter used 10-100 I/O ports and less than 20% used over 100 I/O ports.

It was determined that 39% of all respondents indicated they replaced a current non-MPU based system with one that is MPU-based; 61% said they did not. If they had replaced the non-MPU based system, it was found that around two-thirds of the designers had eliminated over 100 discrete IC's; the other one-third eliminated 20-100 discrete IC's. This is a significant reduction which means fewer interconnections, thus increased system reliability. But this also increases the difficulty of error detection and

isolation. The discrete devices, or IC's, most critical to MPU systems were the DA/AD converter, linear IC's, FET's, etc., in that order.

Microprocessor Systems:

Strong trends in microprocessor systems for avionics indicate: (a) System development tools marketed by microprocessor manufacturers will have a great number of applications. The tools having the largest use will be MPU development systems, logic analyzers, and circuit emulators. (b) The availability of high level languages is quite important to at least two-thirds of the experts surveyed. The high level language that had the greatest popularity was ADA, followed by PASCAL, FORTRAN, and BASIC. It is not surprising that ADA was the most popular, since it is the one which has been adopted as the government standard. (c) Most MPU systems have built-in-testing which indicates a strong preference for it by most designers (nearly three-quarters of the total). (d) More than half of the respondents used redundancy in their microcomputer systems. When redundancy was used, the most popular method was parallel redundancy, followed by voting redundancy. (e) Failures in flight systems using microcomputers are typically time critical, according to nearly three-forths of the respondents.

In conclusion, avionics systems incorporate microprocessors to a large extent in their designs and handle a wide range of applications, including flight management systems, flight control systems, navigation and guidance, data acquisition, communications, etc. There has been a reduction in the cost, size, weight, and power due to the microprocessor and VLSIC's. Use

of BIT (Built-In-Testing), redundancy and fewer interconnections has brought about greater reliability in airborne electronics systems.

The design philosophy of avionics systems has changed over the last decade. In the 1970's the concept was to use one large central computer to perform the desired functions. Now, however, most designers use the distributed techniques in which several or more microprocessors are used to perform the necessary functions. This idea has resulted in greater performance and reliability.

## REFERENCES


1. Seacord, Charles L. and Vaughn, Darrel of Honeywell, Inc., "Computer Technology Forecast Study for General Aviation," NASA CR-137889, Contract No. NAS2-8971, June 20, 1976.

2. Blatt, Paul E. and Yechout, Thomas R., "Digital Flight Control Technology Roadmap," Technical Report AFFDL/FGL-TM-73-65, Wright Patterson AFB, Ohio, January, 1977.

3. Grgurich, John and Bradbury, Peter of Sperry Flight Systems, "Stoland Final Report," NASA CR-137972, Contract No. 2-6567, November, 1976.

4. Frost, Cecil R., "Military CPU's," Datamation, July 15, 1970.

5. Bursky, Dave and Barnes, Dave, "Choosing a $\mu$p By Its Capabilities is a Growing Family Affair," Electronic Design, July 5, 1977.

6. Tsolis, Steve, "IMP-16 Helps Small Planes Fly a Straight Course," Electronics, June 9, 1977.

7. Klass, Phillip J., "Civil Needs Spur Technological Advances," Aviation Week & Space Technology, August 16, 1976.

8. Money, S. A., Microprocessor Data Book, McGraw-Hill, 1982.

9. Lerner, E., "The Automated Cockpit," IEEE Spectrum, February, 1983.

10. Schmid, C. F., and Schmid, S. E., Handbook of Graphic Presentation, John Wiley & Sons, 1979.

# APPENDIX

# SURVEY FORM

(1) What was the first microprocessor that you used? _____

(2) What year did you first use microprocessors? _____

(3) In 1983, have you purchased or do you plan to purchase MPU systems?

_____YES _____NO

A. If yes, will they be:

_____Standard _____Dedicated _____Custom

B. Have they been or will they be:

_____4 bit _____8 bit _____12 bit _____16 bit _____Over 16 bit

_____All of these

C. Will they be:

_____IIL _____Schottky _____CMOS _____NMOS _____PMOS

_____All of these

(4) How will you procure your MPUs? (check all that apply)

_____As chips alone for in-house design (Components)
_____As chips on boards (Sub-System)
_____As assembled modules
_____Complete MPU system (includes memories and power supplies)

If assembled modules, which ones do you contemplate buying?

_____Processor-on-a-board _____Memory module, size _____
_____I/O module _____D/A or A/D converter modules
_____Other _____

(5) Does your MPU application relate to: (check all that apply)

_____Digital system (discretes)
_____Electromechanical system
_____Microcomputer control system
_____Avionics computer system
_____Functional display system

(6) In what applications do you use your microcomputer systems? (check all that apply)

_____Ground Support Equipment _____Navigation
_____Flight Management Systems _____Guidance
_____Flight Test Systems _____Control Systems
_____Data Acquisition Systems _____Instrumentation
_____Communications Systems _____Weapon Control Systems
_____Other: _____

(7) Which of the following factors do you feel are the most important in your company's MPU buying decisions?

_____Price
_____Performance
_____Second src.
_____Distributor credibility
_____Users group/Software library
_____Device warranty
_____Reliability
_____Documentation
_____Training avail.
_____Applications support
_____Development systems
_____Devel. software avail.
_____Applic. software avail.
_____Speed
_____Number of family support devices avail.

(8) Is the microprocessor architecture a: (check all that apply)

_____General purpose microprocessor
_____All-in-one microprocessor
_____Bit slice microprocessor
_____VHSIC microprocessor (Very High Speed IC)

(9) Did the microprocessor perform as expected?

_____Yes _____No

(10) If not, tell why?

_____Not fast enough
_____Limited I/O ports
_____Limited interrupt capability
_____Limited memory addressing capability
_____Other _____

(11) Will you be using advanced architectures such as bit-slice, Z8000, 68000, etc. in the future?

_____Yes _____No

If yes, what general application, and why?

Application _____,

_____Speed
_____Memory capability
_____Bit size
_____Cost
_____Other _____
_____Power ratings
_____Performance
_____Reliability
_____I/O

If no, why not?

_____Speed
_____Memory capability
_____Bit size
_____Cost
_____Other _____
_____Power ratings
_____Performance
_____Reliability
_____I/O

(12) Have you used or do you plan to use semiconductor memories and/or bubble memories during 1982 or 1983?

_____Semiconductor _____Bubble _____Neither

(13) What speed (maximum access time in ns) is best suited for your system? (check all that apply)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Dynamic RAMs (MOS | ___80 | ___120 | ___200 | ___250 | ___300 | ___Other _____ | | |
| Dynamic RAMs (Bipolar) | ___80 | ___120 | ___200 | ___250 | ___300 | ___Other _____ | | |
| Static RAMs (MOS) | ___20 | ___35 | ___45 | ___55 | ___70 | ___120 | ___150 | |
| Static RAMs (CMOS) | ___200 | ___250 | ___300 | ___450 | ___Other _____ | | | |
| Static RAMs (Bipolar) | ___200 | ___250 | ___300 | ___450 | ___Other _____ | | | |
| ROMs (MOS) | ___100 | ___150 | ___200 | ___250 | ___350 | ___450 | ___Other _____ | |
| ROMs (Bipolar) | ___100 | ___150 | ___200 | ___250 | ___350 | ___450 | ___Other _____ | |
| PROMs (MOS) | ___25 | ___80 | ___100 | ___120 | ___150 | ___200 | ___Other _____ | |
| PROMs (Bipolar) | ___25 | ___80 | ___100 | ___120 | ___150 | ___200 | ___Other _____ | |
| EPROMs (MOS) | ___150 | ___200 | ___250 | ___350 | ___450 | ___Other _____ | | |
| EEPROMs (MOS) | ___150 | ___200 | ___250 | ___350 | ___450 | ___Other _____ | | |
| PSEUDO-Statics (MOS) | ___200 | ___250 | ___300 | ___450 | ___Other _____ | | | |
| Bubble (Solid-State) | ___5000 | ___10000 | ___20000 | ___40000 | ___Other _____ | | | |

(14) How much total memory will your system require in 1982 and 1983? (bytes)

| 1982 | 1K or Less | 2K | 4K | 8K | 16K | 32K | 64K | 128K | 256K | > |
|---|---|---|---|---|---|---|---|---|---|---|
| Dynamic RAMs (MOS) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| Dynamic RAMs (Bipolar) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| Static RAMs (MOS) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| Static RAMs (CMOS) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| Static RAMs (Bipolar) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| ROMs (MOS) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| ROMs (Bipolar) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| PROMs (Bipolar) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| PROMs (MOS) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| EPROMs (MOS) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| EEPROMs (MOS) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| PSEUDO-Statics (MOS) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| Bubble (Solid-State) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| _____ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |

| 1983 | 1K or Less | 2K | 4K | 8K | 16K | 32K | 64K | 128K | 256K | > |
|---|---|---|---|---|---|---|---|---|---|---|
| Dynamic RAMs (MOS) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| Dynamic RAMs (Bipolar) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| Static RAMs (MOS) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| Static RAMs (CMOS) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| Static RAMs (Bipolar) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| ROMs (MOS) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| ROMs (Bipolar) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| PROMs (Bipolar) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| EPROMs (MOS) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| EEPROMs (MOS) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| PSEUDO-Statics (MOS) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| Bubble (Solid-State) | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |
| _____ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ | ___ |

(15) Which of the following factors do you feel are important in your memory buying decisions? (Check all that apply)

_____Speed
_____Parts availability
_____Ease of use
_____Board density
_____Price
_____Reliability documentation
_____State-of-the-art
_____Second source
_____Application support
_____Warranty
_____Manufacturers credibility
_____Performance
_____Broad line support
_____Technical support
_____Access cycle
_____Active power
_____Standby power
_____Reputation
_____Application information
_____Delivery
_____Power
_____Soft failure

(16) In your system, how important is EPROM to mask ROM compatibility?

_____Will not use EPROM without pin compatible mask ROM
_____Compatibility not important

(17) Does the system incorporate any external data storage?

_____Floppy disk _____Cassette _____Paper tape _____EEPROM

_____Magnetic tape _____Hard disk _____Bubble memory

(18) How many I/O ports does the microprocessor support? _____

(19) Does your MPU-based system replace a current non-MPU based system?

_____Yes _____No

If yes, how many discrete ICs or other semiconductor elements have been eliminated?

_____0-10 _____10-20 _____20-100 _____100-200 _____Over 200

(20) What discrete devices or ICs are critical or key to your MPU systems?

_____Linear ICs
_____DA/ADs
_____FETs
_____Other: _____

(21) Will you use the system development tools that are marketed by microprocessor manufacturers?

_____Yes _____No

What specific system development tools do you/will you use? (check all that apply)

_____MPU development system
_____Computational mainframe
_____Logic analyzer
_____Circuit emulation
_____Time sharing
_____Systems evaluation tool
_____Others: _____

(22) How important is availability of high-level languages?

_____Very _____Somewhat _____Not important

If a high-level language is important, which do you favor?

_____Basic _____Fortran _____Other: _____
_____PL/M _____Pascal
_____Cobol _____Ada

(23) What is the throughput and the maximum power of your microcomputer systems? (please specify the general application of the system)

Application 1 _____
KIPS (thousands of instructions per sec)

_____100-500 _____500-1000 _____1000-1500 _____1500-2000 _____>2000

Watts

_____<1 _____1-10 _____10-20 _____20-30 _____30-40 _____>40

Application 2 _____
KIPS (thousands of instructions per sec)

_____100-500 _____500-1000 _____1000-1500 _____1500-2000 _____>2000

Watts

_____<1 _____1-10 _____10-20 _____20-30 _____30-40 _____>40

Application 3 _____
KIPS (thousands of instructions per sec)

_____100-500 _____500-1000 _____1000-1500 _____1500-2000 _____>2000

Watts

_____<1 _____1-10 _____10-20 _____20-30 _____30-40 _____>40

(24) Does the microcomputer system have Built-in-Test fault?

_____Yes _____No

(25) Do you use redundancy in the microcomputer systems you are designing?

_____Yes _____No

(26) If yes, what types of redundancy do you use? (please specify the general application of the system)

Application 1 _____

(26) Continued

_____Parallel _____Series _____Series-parallel
_____Standby _____Partial _____Voting
_____Other _____

Application 2 _____

_____Parallel _____Series _____Series-parallel
_____Standby _____Partial _____Voting
_____Other _____

Application 3 _____

_____Parallel _____Series _____Series-parallel
_____Standby _____Partial _____Voting
_____Other _____

(27) Are the microcomputers used in flight systems where failures are time critical?

_____Yes _____No

(28) In multiple microcomputer systems are the microcomputers operated synchronously or asynchronously in your redundant application?

_____Sync _____Async

Thank you!

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

OPTIONAL:

NAME:_____

ORGANIZATION:_____

ADDRESS:_____

_____

TELEPHONE NO:_____

# APPENDIX B

ORIGINAL PAGE IS
OF POOR QUALITY

# UNIVERSITY OF SOUTHERN COLORADO
# DIFFERENTIAL GLOBAL POSITIONING SATELLITE PROJECT
# FINAL REPORT

ARTHUR A. BECKMAN

## INTRODUCTION

The NAVSTAR Global Positioning System (GPS) is a prototype space-based radionavigation system developed by the Department of Defense (DOD). The system is managed by the Joint Program Office (JPO) in the Air Force System Command (AFSC). When the system is fully deployed and operational in 1987 it will consist of 18 satellites in three planes of orbit.

Phase I of the deployment schedule verified that it is feasible to determine highly accurate user time and positioning by tracking four satellites placed in 10,900 nautical mile radius, 12-hour orbits. There are currently four functioning satellites in two circular orbital planes at 55 degree inclination with respect to the earth's equatorial plane, providing periodic three to four hour intervals of four satellite coverage over selected geographical areas.

Current DOD goals for NAVSTAR GPS accuracy call for 10 meter three-dimensional accuracy for military missions, and 100 meter accuracy for civil aviation. The satellites broadcast on two frequencies, L1 and L2. The signals on these two frequencies will consist of ephemeris data from the satellite data store and one of two pseudo-random codes. The coarse aquisition (C/A) code has a very short period and will be used by the military to aquire the much longer precision (P) code. The military will use the P code to navigate with, whereas civilian aviation will use the less accurate C/A code.

ORIGINAL PAGE IS
OF POOR QUALITY

The NAVSTAR GPS system consists of three segments. The satellite segment consists of the space-based satellites which broadcast ephemeris data and current satellite time. Each satellite has a highly accurate atomic clock on board. The ground segment consists of a number of ground-based stations to monitor satellite accuracy and status and another ground-based station to collect reports and transmit current ephemeris and correction data to each of the satellites. The final segment is the user segment which receives the satellite signals and uses them to navigate.

Four satellites are normally required for navigation. Ranges to the four satellites are determined by scaling the signal transit time with the speed of light. If the user clock could be maintained precisely with GPS system time, only three satellites would be required for navigation. The fourth satellite is required to resolve the fixed bias ambiguity between the imprecise user's clock and the GPS system time. User position is determined by simultaneous solution of four equations involving four unknowns- the three dimensional position estimates and user clock bias.

DIFFERENTIAL GPS

To prevent unfriendly nations from benefiting from the navigation capabilities of the GPS system, the DOD currently plans to corrupt the C/A code to degrade its' capabilities to no better than 100 meters (2 DRMS). This capability is sufficient for RNAV/IFR enroute, terminal, and non-precision approach and landing requirements but is not sufficient for RNAV/IFR Categories 1 or 2 precision approach and landing conditions.

ORIGINAL PAGE IS
OF POOR QUALITY

One of the major potential benefits of the NAVSTAR GPS system is the capabiltiy to provide enroute and landing capability in areas which are not covered by conventional navigation systems. These areas include large portions of Alaska and offshore oil drilling rigs.

The use of GPS in a differential mode has the capability to provide not only this remote navigation benefit but also allow category II and III approaches and landings if the FAA will modify its' regulations regarding helicopter approaches. Differential navigation is currently in use with Omega, Loran-C, and the TRANSIT satellite system.

The differential technique significantly improves position accuracy by utilizing a surveyed ground site which receives navigation signals, applies corrections to the signals received, and broadcasts these corrections to any aircraft in the local area to apply to their solutions. The errors for all local users are normally highly correlated. The errors also tend to change more slowly than the time required to transmit them to nearby users. For Omega, Loran-C, and Transit, the technique provides improved position accuracy over distances up to a few hundred miles.

There are three basic differential GPS concepts:

1. Data link type.
2. Pseudolite type.
3. Translator type.

The data link type uses a surveyed GPS receiver which receives GPS signals, computes its' location, and computes the

ORIGINAL PAGE IS
OF POOR QUALITY

corrections based on its' known location. These corrections are then transmitted to local users. It has the advantage of requiring little modification to airborne or ground GPS sets. The major disadvantage is that it requires a separate data link.

The pseudolite type also computes corrections in the same way as the data link type but instead of broadcasting just the corrections, it also generates its' own code and navigation message. It has a number of advantages but also has numerous disadvantages, mostly involving cost.

The translator type receives the GPS signal and retransmits them on an offset frequency. The airborne GPS set corrects for the errors in each signal by knowing the location of the transponder. The major advantage is the low cost of the ground equipment. The major disadvantage is that the cost of the airborne set is much higher.

The goal of the research being conducted at NASA Ames is to provide a data base of infomation regarding differential GPS. Major research prior to University of Southern Colorado involvement was an initial test flight series to establish the operation of the Z-set under the current satellite configuration, and modeling of the different concepts of differential GPS to determine the type which will be pursued in flight testing.

UNIVERSITY OF SOUTHERN COLORADO CONTRIBUTIONS.

Work for NASA in support of differential GPS fell into the following areas: literature search, hardware installation in the SH3 helicopter and laboratory, technical visits to Magnavox to verify performance of the NASA interface processor (NIP) and final acceptance testing of the NIP, design and testing of

ORIGINAL PAGE IS
OF POOR QUALITY

software used in the NASA computer to gather data and send the data via telemetry to a ground station for recording, and support of the preliminary differential GPS flight experiment.

The literature search involved a DIALOG based search for all published GPS related work. Pertinent articles were then read and analysed to insure that NASA has access to applicable previous research, and that research already performed will not be duplicated by NASA.

The helicopter used for NASA's GPS research is a SH3 Sea Stallion. In the previous experiment an antenna and pre-amp were mounted in tail of the aircraft and an antenna lead was run to the Magnavox Z-set receiver in the experimenter's rack. The Z-set displays its' output on a control display unit (CDU), also mounted on the rack. The Z-set receiver communicates with the CDU via a serial RS-232C link. In the previous experiment, an interface had been installed between the two devices to also transmit the data to a HP computer to record.

For the next set of experiments, a new interface was designed and built by Magnavox which connects directly to the Z-set's computer bus and transfers data via direct memory access (DMA), a method which allows the gathering of much more data. This interface includes a Digital Equipment Corporation (DEC) LSI computer to process requests for data from NASA's Norden PDP 11/34 computer, collect this data from the Z-set, do limited processing of this data, and then send it to the NASA computer using a serial RS-232C line.

Due to the volume of the data collected, the available on-

ORIGINAL PAGE IS
OF POOR QUALITY

board storage was deemed insufficient and all data is sent to a telemetry transmitter via a parallel port to be sent to the ground where it is recorded on magnetic tape. The Norden computer connects to a floppy disk drive for operating system and applications program storage. An HP terminal is used for operator input and to monitor data flow (see figure 1).

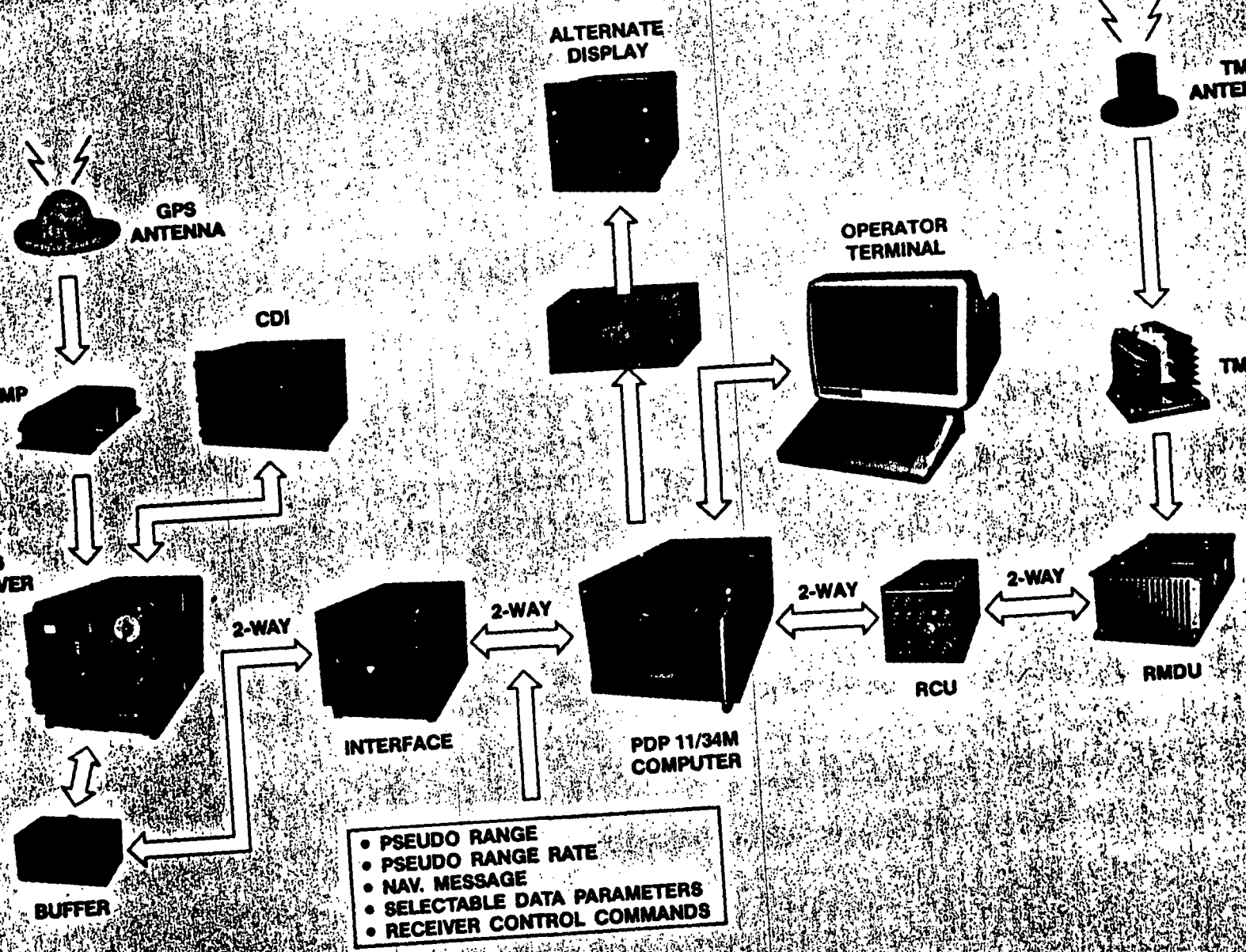
Equipment installation consisted of installing racks for the Z-set, NIP, Norden computer, and HP terminal, designing and building cables between the telemetry transmitter and Norden computer, and between the Norden computer and the HP terminal, and routing power to the various components. USC's input was to specify the layout for the racks, verify the design of the cabling, and test all cabling and overall operation of the system.

A second installation is in building 210. This laboratory installation is used for static operation to determine error characteristics, allow side-by-side comparisons with other sets, and allow testing of the various components. The antenna for the lab is on the building's roof as is the pre-amp. For previous work cabling had been installed for both a GPS antenna and a Loran antenna. The remaining equipment from the aircraft is installed in the lab itself.

To allow the operation of two sets, a second antenna was required. USC's role was to procure the cabling and connectors, coordinate the modifications to the roof antenna tower to accept a second antenna and pre-amp, coordinate the installation of the cabling, and test the system.

In order to extract information from the Magnavox Z-set

# DIFFERENTIAL GPS AIRBORNE SYSTEM




ORIGINAL PAGE IS OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

receiver for processing by the NASA computer, a contract was let to Magnavox's Advanced Products and Systems Company to build an interface processor which would receive requests for blocks of information from the Norden and then get this data from the receiver. Both the receiver and interface processor use DEC LSI computers with the Q-bus. The interface processor would communicate with the receiver via the bus and with the Norden using serial RS-232C.

Magnavox began work on the interface in January 1983 and was scheduled to finish in mid-October 1983. The author was assigned the task of monitoring progress, making liason trips to observe progress, and represent NASA during final acceptance testing.

Magnavox underestimated both the complexity of the project and the lead times for parts. By October 1983, the original software team had been replaced and a new start was scheduled for the software. The hardware was substantially complete in November and an attempt was made to write enough software to fly the originally scheduled experiments in December 1983. The author made two trips to Magnavox's location in late November to test the software. Unfortunately it was not possible to extract enough information to make the flights useful and the experiments were rescheduled for April 1984.

A meeting at NASA in December resolved the last of the conflicts with the design specifications and development then proceeded on the new schedule. Final acceptance testing at Magnavox's location in mid-March 1984 was successfully completed and the equipment was shipped to NASA.

ORIGINAL PAGE IS
OF POOR QUALITY

The purpose of the initial set of flight experiments was to provide a base of information to roughly define the actual performance capabilities of differential corrections. Since NASA's actual ground GPS station wasn't due to be finished until late 1985, Sierra Nevada Corporation was contracted to provide their services to locate a mobile second GPS set with data collection capabilities. This set would then be located on a surveyed site at the Navy's Crows Landing facility where the flight tests were scheduled to be conducted.

During a flight, both GPS sets would collect data at the same time. The airborne system would send the data to the ground station using telemetry from where it would be transmitted back to Ames for recording on magnetic tape. The ground system would send its' data via a high speed modem to Ames where it would be likewise recorded. A laser tracker at Crows would be utilized to provide a highly accurate record of where the aircraft was in space at all times during the flight. (See figure 2).

Subsequent to the flight the laser tracking data and data from the aircraft GPS set provide a record of the error experienced throughout the flight. The data from the ground based GPS set provides the errors experienced by that set by comparing the computed position and subtracting the known coordinates. This error data can then be applied to the airborne data to give a rough approximation of the benefit of differential GPS.

The author was responsible for all software running on the Norden computer on board the SH3. Three major routines were written to accomplish the tasks of communicating with the NIP and

# PSEUDO-DIFFERENTIAL GPS FLIGHT EXPERIMENT

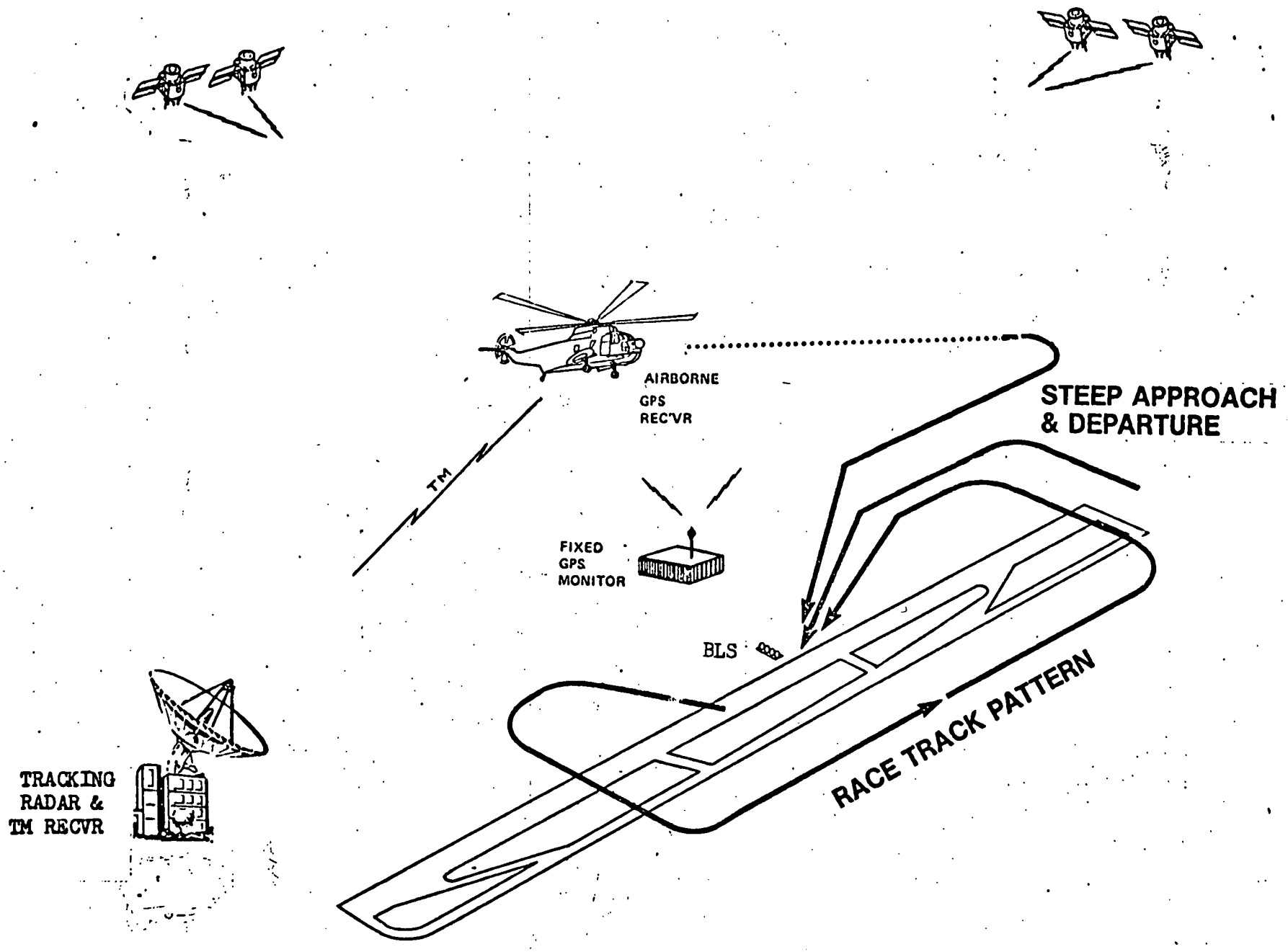



Figure 2

ORIGINAL PAGE IS
OF POOR QUALITY

the telemetry hardware. Two routines are actually collections of subroutines which handle all communications with a specified hardware device. The main routine handles interaction with the operator, prepares requests for transmission to the NIP, and buffers data for the telemetry link.

A Norden DZ-11 device is used to communicate with the NIP. It is a multiplexed eight port serial card. The DZ subroutines handle initialization, data transmission and receiving, and error handling. The transmission subroutine is called by non-time critical portions of the main routine and sends the message without returning to the calling routine until complete. The receiving subroutine on the other hand is called by a time critical portion of the main routine and must operate using interrupts. It uses double buffering to insure that no data is lost due to slow response time by the main routine. The incoming message's checksum's are verified to insure that a complete message is received.

A Norden DR device is used to communicate with the telemetry hardware. This is a 16-bit parallel port with one input port and one output port. The DR subroutines handle initialization of the DR card and output to the card. The output subroutine is called by a time critical part of the main routine and is therefore interrupt driven. It is single buffered and the main routine waits if it gets ahead of the output stream.

The main routine calls all of the initializations subroutines and then presents the operator with a menu of

ORIGINAL PAGE IS
OF POOR QUALITY

choices. This menu can be recalled at any time by hitting a key on the keyboard. When the operator makes a choice the routine then prepares the appropriate messages to be sent and calls the subroutines. Several of the selections request repetive data and once selected the software automatically collects the incoming data and sends it to the telemetry hardware with no further operator intervention. The routine contstantly checks a series of flags from the subroutines when not in the menu mode. These flags inform the main routine of the status of the various buffers so that data can be copied at the appropriate time to prevent loss of data. Portions of the collected data are displayed on the operator console so that correct operation of the program can be verified.

The program was specifically written to allow for a wide variety of different configurations to allow for different flight experiments in the future. Only the messages to the NIP and the menu need be changed to request new data. For the initial flights a set of requests were prepared for ephemeris data and another set for range and pseudo-range data. The pseudo-range and range data were collected throughout the flight except when new ephemeris data was being collected.

Ephemeris data was collected just prior to beginning the experiment and whenever there was a satellite changeover during the flight (no more than one changeover). A communications link with the ground GPS station insured that they collected the ephemeris at the same time.

Support for the flight operations involved installing and testing the equipment prior to each flight, operating the

ORIGINAL PAGE IS
OF POOR QUALITY

equipment during the flight and then removal of those portions of the equipment required back in the lab after the flight. Several flights were made to insure that enough data was collected. Each flight lasted at least an hour and involved approaches and take-offs as well as a racetrack pattern (see figures 3 and 4).

This initial set of flights were finished in early June. The results appear to be useful and will be published in early 1985.

---

Appendices

1 - Checklists for installation and operation of equipment

2 - Software listings.

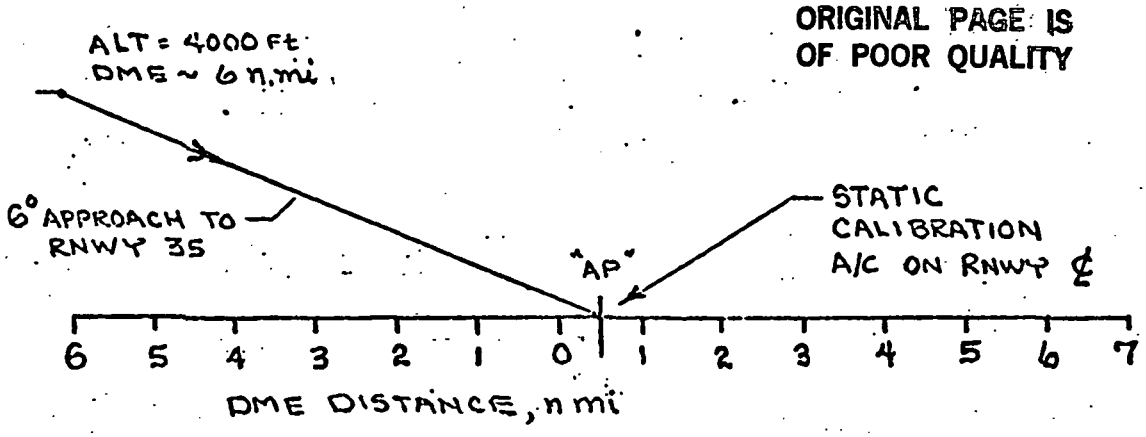
# DIFF. GPS SH-3G FLIGHTS

## CROWS LANDING OPERATIONS

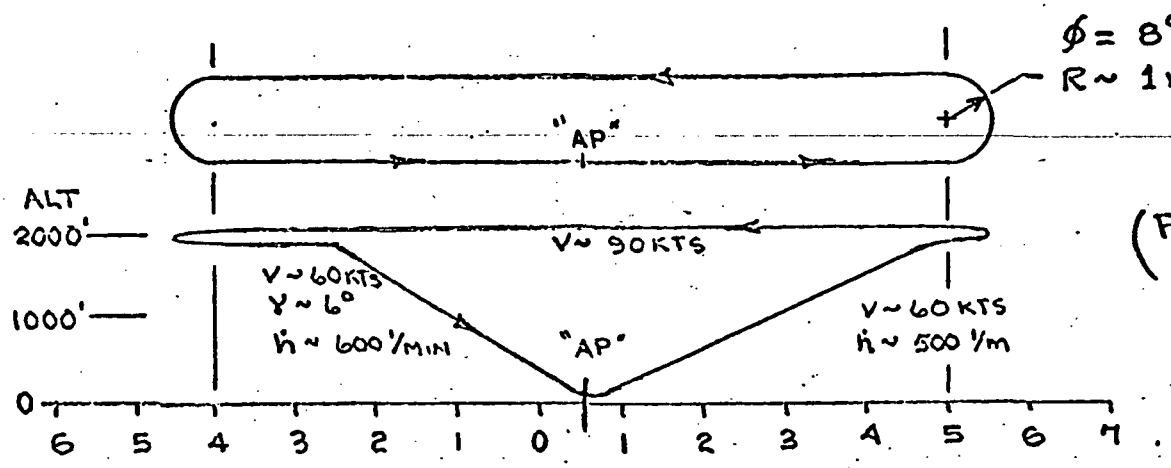
JUN 21 1984

ORIGINAL PAGE IS OF POOR QUALITY

T.O. MOFFETT 16:00
T.D. CROWS 17:15




RUN A 17:15 ⇓ 17:3

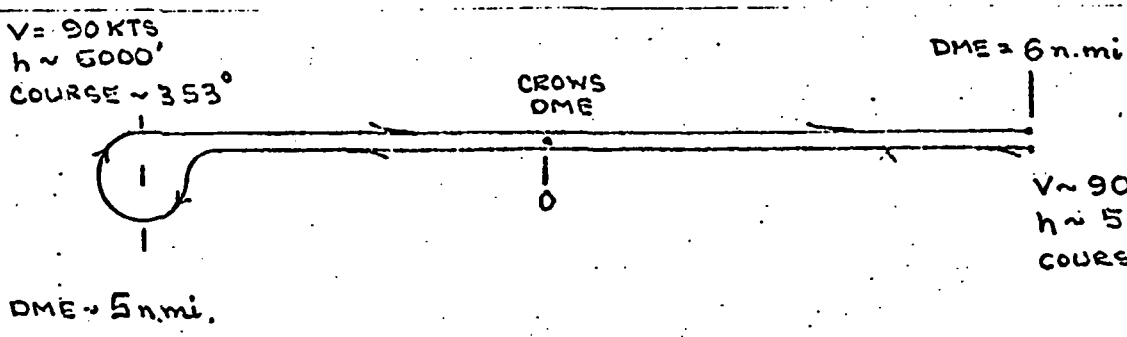



17:3

RUN B

(PERFORM TWICE)

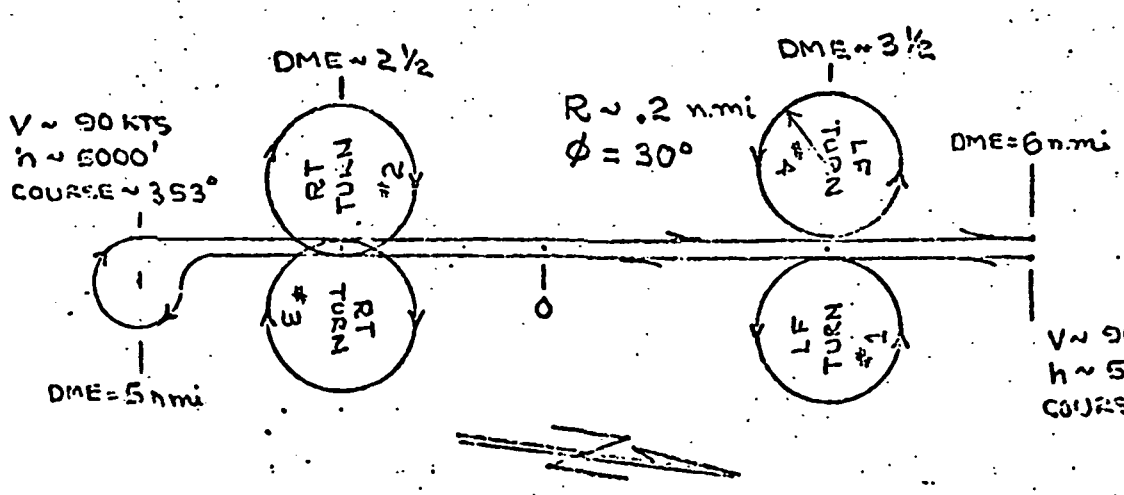
18:0




18:1

RUN C

18:2




18:2

RUN D

18:4

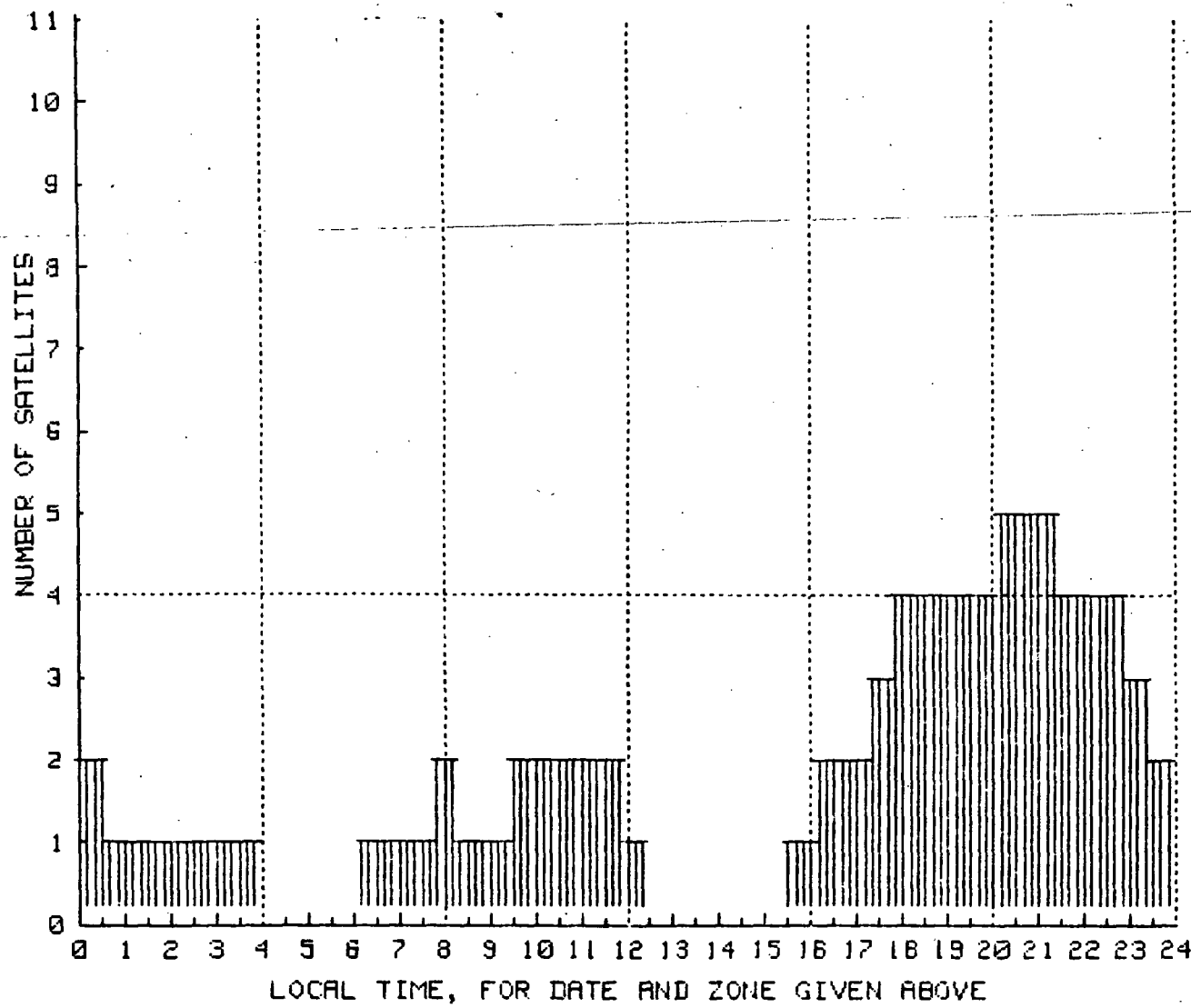
RETURN TO MOFFETT 19:

AMES RESEARCH CENTER NASA NAVSTAR

ORIGINAL PAGE IS
OF POOR QUALITY

ALMANAC AGE IS 5 / 3 / 1984
DATA VALID FOR 5 / 24 / 1984
37.412 DEGREES LATITUDE
-121.1058 DEGREES LONGITUDE
5 METERS ABOVE WGS72 ELLIPSOID
SATELLITES
9 4 11 6 8

SATELLITE AVAILABILITY SUMMARY



# GPS SH3 INSTALLATION
## EQUIPMENT INSTALLATION

WARNING! INSURE THAT RACK POWER IS OFF.

1. Insure that pre-amp has been installed in tail of SH3 by aircraft maintenance technician.

2. Insure that RMDU is connected properly (coordinate with aircraft manager).

3. Secure HP 2382A terminal to rack using straps and screws (baseplate). Connect DB-25 connector to rear and plug power cord into power strip at rear of rack.

4. Install CDU in cutout. Connect cable marked "CDU" to J-1 on back of CDU.

5. Insure that jumper cable is in place between J-1 of CDU and J-3 of Z-Set (old interface used to connect to this break).

6. Install Norden in full ATR rack. Connect power cable to J-4. Connect cable to terminal to J-1. Connect cable from NIP to J-6. Connect UNIBUS cable to J-5. Verify that all pins are solid on cable to RMDU and connect to J-3 (one pin is held in place by glue).

7. Install Miltope floppies and controller in their rack (if flying) or hook up on top of rack. Connect UNIBUS cable to J-1 of controller. Plug in power cords to power strip at rear of the rack.

8. Install Z-set in right hand 3/4 ATR rack. Connect antenna lead to J-4. Connect power cable to J-2. Connect CDU cable (from CDU J-1) to J-3.

9. Install NIP in left hand 3/4 ATR rack. Connect power cable to J-2. Do not plug in. Connect cable from J-6 of Norden to J-3. Connect thick cable from Z-set to J-1.

10. Have wiring inspected and tied down.

To remove equipment, reverse procedures. Note that all cables that connect to the NIP are also used in the lab.

Tools required: Flat head and phillips screwdrivers and wire cutters (to free cabling from tie-downs).

# GPS SH3 INSTALLATION
# PRE-FLIGHT INSTRUCTIONS

WARNING! - THESE STEPS TO BE ACCOMPLISHED IN SEQUENCE.

1. Insure that power is available to SH3 and that experimenters station power switch (to right of righthand pilot) is on.

2. Insure that all equipment is turned off. Turn brightness switch on CDU fully counter-clockwise (click).

3. Flip two right hand switches up on small orange box at upper left of rack. Verify that green lights come on.

4. Push in CB-18, 21, 22, 19, 7, 8, 10, and 12 (GPS, rack and RMDU power). Push in CB-4 and 5 for AIMS altimeter (if used) and CB-3 if radar tracking will be utilized.

5. Apply power to terminal (push switch in rear).

6. Apply power to disk controller and drives.

7. Push in CB-16 to turn on computer.

8. Turn brightness switch on CDU clockwise to adjust brightness of display.

9. Plug NIP power cord into power strip at the rear of the rack.

10. Pre-flight Z-set per page 5-12 of Z-set user's manual.

11. Insert aircraft floppy disk in DY0:

12. Hold down shift key on terminal and hit break key.

13. Hit caps key.

14. Type L<sp>171000<cr> followed by S<cr>. System should boot.

15. After system boots type INIT VM:<cr> followed by Y<cr>. The next 6 instructions load virtual memory with a minimal system and the program.

16. Type COPY<cr> *.SYV<cr> VM:*.SYS<cr>.

17. Type COPY<cr> RT11VM.SYS<cr> VM:RT11SJ.SYS<cr>.

18. Type COPY<cr> SWAP.SYS,NORINT.SAV<cr> VM:<cr>.

19. Type COPY<cr> STARTV.COM<cr> VM:STARTS.COM<cr>.

20. Type COPY/BOOT<sp>VM:RT11SJ<sp>VM:<cr>.

21. Type BOOT<sp>VM:<cr>. What boots is a minimal system running on a virtual disk (in core memory). The only usable

command is R (not RU or RUN).

22. Type R<sp>NORINT<cr> to verify program operation.

23. When NORINT comes up type <sp> to begin.

24. Type 2<cr> to send a test pattern to the ground station. Verify with the ground station that they are receiving the pattern.

25. Type <sp> to get menu back. Type 6<cr> to exit program.

26. Verify that the almanac has been gathered successfully. Page 4-10 of Z-set User's Manual.

27. Remove the aircraft disk.

28. Pull CB-16 out. Computer must be turned off before peripherals to prevent core memory from being trashed.

29. Turn off the floppy drives and the terminal.

30. Unplug the NIP.

31. Turn brightness knob on CDU fully counter-clockwise (click).

32. Pull CB-12, 10, 8, 7, 18, 19, 21, and 22. Pull any other circuit breakers still in.

33. Flip switches at orange box down.

34. Remove UNIBUS cable from front of Norden. Take UNIBUS terminator off of the floppy drive controller and screw into J-5 on Norden.

35. Remove floppy drive from aircraft if required.

GPS SH3 INSTALLATION
FLIGHT OPERATIONS

WARNING! - THESE STEPS TO BE ACCOMPLISHED IN SEQUENCE.

1. Wait until engines have started and pilots give go ahead for removing seatbelts.

2. Flip two right hand switches up on small orange box at upper left of rack. Verify that green lights come on.

3. Push in CB-18, 21, 22, 19, 7, 8, 10, and 12. Push in CB-4 and 5 for AIMS altimeter (if used) and CB-3 if radar tracking will be utilized.

4. Plug NIP power cord into power strip at rear of the rack.

5. Apply power to terminal (push switch in rear).

6. Push in CB-16.

7. Turn brightness switch on CDU clockwise to adjust brightness of display.

8. See page 5-12 of Z-set user's manual to prepare for take off.

9. Hold down shift key on terminal and hit break key.

10. Hit caps key.

11. Type L<sp>0<cr> followed by S<cr> to boot system in memory.

12. After system boots type R<sp>NORINT<cr>.

13. When NORINT comes up, type <sp> to begin.

14. Type 1<cr><sp>1<cr> to get accurate status of Z-set.

15. Verify that status of Z-set is healthy and that the set is navigating (see Z-set user's manual page 5-12).

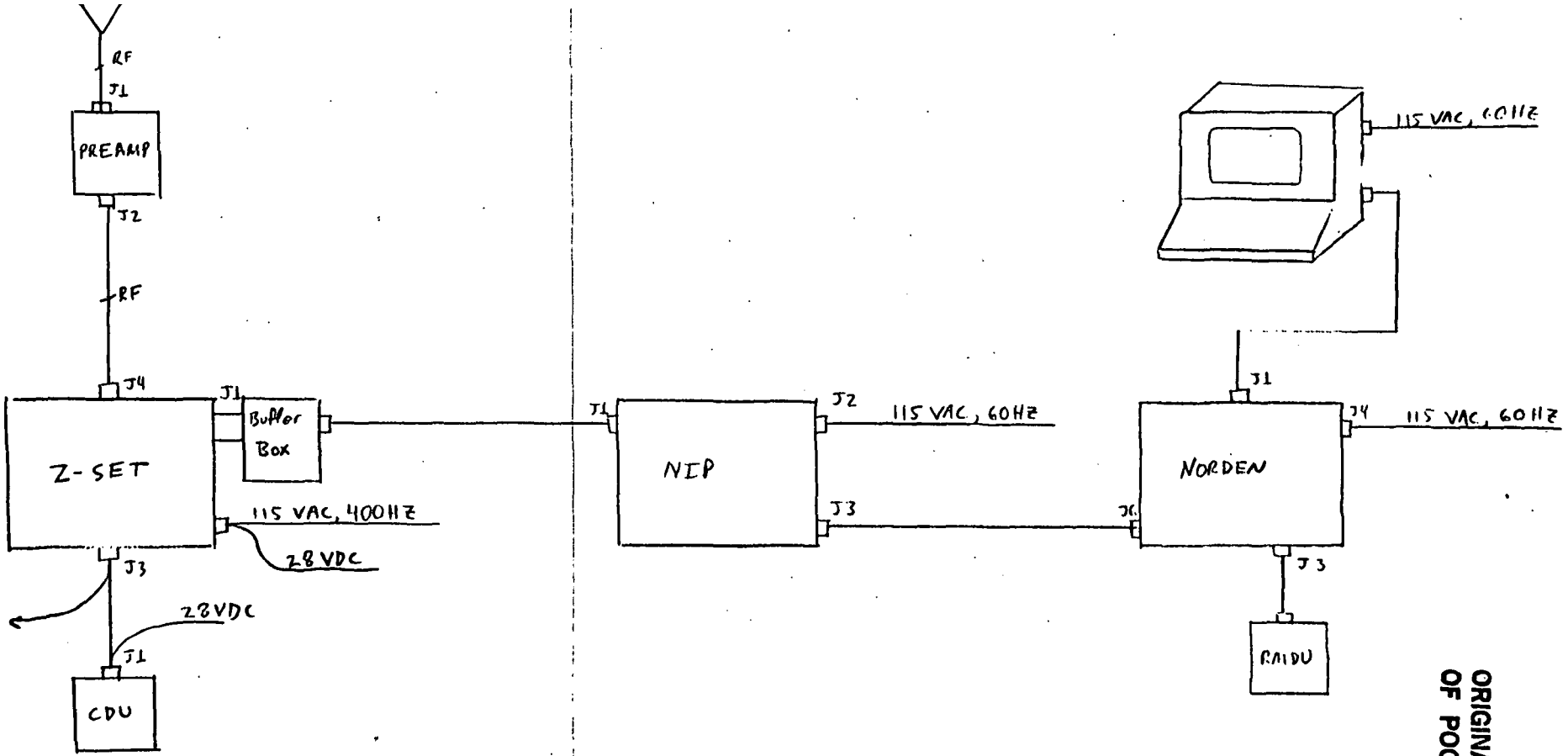
16. Type <sp>5<cr> to begin data collection.

17. Next three instructions gather ephemeris. Type <sp>3<cr>, wait for 5 seconds, then type <sp>4<cr>. Block 34 should be collected immediately.

18. Turn selector switch on CDU to EPE position, push 5A key, wait until A appears in upper left on display, then push CLR key. "A" should be cleared from display. Z-set re-gathers ephemeris.

19. Wait for all blocks to be collected. Completion indicated when Z-set is again navigating on four satellites.

20. Type <sp>5<cr> to restart data collection.




ORIGINAL PAGE IS
OF POOR QUALITY

Differential GPS Airborne Equipment

# GPS SH3 INSTALLATION
# SHUTDOWN INSTRUCTIONS

WARNING! - THESE STEPS TO BE ACCOMPLISHED IN SEQUENCE.

1. Type (sp)6(cr) to exit program.

2. Pull CB-16 out.

3. Turn off the terminal.

4. Unplug the NIP.

5. Turn knob on CDU fully counter-clockwise (click).

6. Pull CB-12, 10, 8, 7, 18, 19, 21, and 22. Pull any other circuit breakers still in.

7. Flip switches at orange box down.

8. See equipment installation instructions for SH3 for removal of equipment.

# GPS LAB INSTALLATION
## EQUIPMENT INSTALLATION

1. Insure that all breakers are out on power distribution box.

2. Connect CDU (J-1) to merged cable coming from J-3 of Z-Set.

3. Connect AIMS altimeter (if used) to cables coming from J-3 of Z-Set and power distribution box.

4. Connect cable from CDU to J-3 of Z-Set.

5. Connect J-2 of Z-Set to cable coming from power distribution box.

6. Connect J-1 of Z-Set to buffer box. To simplify installation, thick cable leading to NIP should already be connected to buffer box. Connect cable to J-1 of NIP.

7. Connect antenna lead to J-4 of Z-Set (marked as GPS #1).

8. Connect ground wire from 28 volt power supply to one of the lower righthand screws on the face of the Z-Set.

9. Insure that the 28 volt leads from the power distribution box are connected to the 28 volt power supply. The two ground wires which parallel the antenna wires from the crawl space should be connected to the 28 volt power supply.

10. Loran antenna cable is being used to power preamp on roof and should be connected to 15 volt power supply. Antenna has dual banana plug attached. Notch on plug is ground side.

11. Connect cable (bring over from aircraft) between J-3 of NIP and J-6 of Norden.

12. Connect console cable from breakout panel on rack to J-1 of Norden. Connect terminal to panel (J1B). Cable should have wires 2 & 3 straight through (verify with ohm meter).

13. Connect UNIBUS cable to MILTOPE floppies or extender box (See separate instructions for extender box) and J-5 of Norden.

14. Connect power cable to J-2 on NIP. Do not plug into power.

# GPS LAB INSTALLATION
## POWER ON

WARNING! - THESE STEPS TO BE ACCOMPLISHED IN SEQUENCE.

1. Fill out GPS log.

2. 115 VAC, 400HZ power cord connected to power distribution box.

3. Power cords to disk drives, terminal, and power supplies plugged in (60HZ).

4. GPS brightness control on CDU full counter-clockwise (click).

5. CB-1 on power distribution box in. Then CB-2 in.

6. CB-3 (AIMS PUR), and CB-4 (AIMS VIB.) in if altimeter used.

7. Turn on power to antenna pre-amp. Verify approx. 0.5 amp current and 15 volts.

8. Turn on +28 volt power supply to CDU (two switches). WARNING! Do not touch the 28 VDC rheostat adjustment. It is preset on 80 for proper operation.

9. Turn GPS brightness control on CDU clockwise to adjust display brightness.

10. Plug in NIP to 115 VAC, 60HZ.

11. Follow GPS operations checklist for proper system operations.

12. Apply power to terminal.

13. Apply power to disk drives.

14. Apply power to Norden computer.

15. Follow software operation instructions.

GPS LAB INSTALLATION
POWER OFF

WARNING! - THESE STEPS TO BE ACCOMPLISHED IN SEQUENCE.

1. GPS brightness switch full counter-clockwise (click).
2. CB-1 out.
3. CB-2, CB-3, CB-4 out.
4. +28 VDC output power switch (right) off.
5. +28 VDC power switch (left) off.
6. Turn off power to antenna pre-amp.
7. Exit any software running on Norden.
8. Unplug NIP.
9. Turn off Norden.
10. Turn off disk drives.
11. Turn off terminal.

# EXTENDER BOX
# CONNECTING TO NORDEN

1. Unload heads of RL02s (load switches out).

2. Turn power switch on commercial PDP 11/34A to OFF.

3. Pull extender box drawer out and open top.

4. Remove UNIBUS cable leading to SIGMA 9 interface (left rear cable). Note orientation of connector.

5. Remove terminator card (card in back with purple top) from SIGMA 9 interface and place in slot emptied by UNIBUS cable in extender box.

6. If additional memory board is needed (for RSX-11M), verify proper memory address using attached sheet (NORDEN has 64K words internally). Place memory board between main UNIBUS cable and RX211 card after removing grant jumper.

7. Close extender box and push it back into place.

8. Connect UNIBUS cable from NORDEN to connector behind extender box.

9. Pull top off of commercial computer and remove UNIBUS cable. NOTE LOCATION OF NOTCH!

10. Insert UNIBUS cable into slot next to where the cable from the NORDEN screws into extender box. Note that the tab which prevents the cable from being inserted backwards is broken. Insure that notch is over broken portion.

11. Apply power to NORDEN and terminal.

12. Apply power to disks and extender box by flipping silver switch above and right of RX02s up.

13. Push load buttons on RL02s in.

Reverse procedure to reinstall commercial computer. Remember grant jumper after removing memory.

```
C+-----------------------------------------------------------------------
      PROGRAM NORINT
C
C     INTERFACE SOFTWARE BETWEEN NIP AND SMDU TO RUN ON NORDEN
C
C     CALLS TO:  DZINIT, DZOUT, DZSTOP, IAND, INTTRD, NIPSTA, INTTRD,
C                RXINT, DRINIT, DRINT, DROUT, DRSTOP
C
C     CALLS FROM:  NONE
C
C     PARAMETERS:  NONE
C
C     COMMONS:  NONE
C
C     ENVIRONMENT:  NORDEN PDP-11/34A  RT-11  FORTRAN IV
C
C     WRITTEN BY:  ARTHUR A. BECKMAN, UNIV. OF SO. COLORADO, 1984
C
C     RESTRICTIONS:  NONE
C
C     REFERENCES:  USER IS EXPECTED TO BE FAMILIAR WITH THE Z-SET
C                  AND NIP USER MANUALS.
C
C     NOTE:  NORMAL COMPILATION RESULTS IN A VERSION WHICH WORKS
C            WITHOUT THE DR CARD OUTPUT.  TO PRODUCE THE AIRBORNE
C            VERSION USE THE DEBUG SWITCH TO INCLUDE STATEMENTS
C            WITH D IN FRONT OF THEM.
C
C     MODIFICATIONS:
C
C-----------------------------------------------------------------------
C
C DATA CONSTANTS
C
      INTEGER * 2  MSLLOC !LOCATION OF LENGTH WORD IN MESSAGE ARRAY
      INTEGER * 2  RSTLOC !LOCATION OF RESPONSE MESSAGE STATUS BYTE
      INTEGER * 2  RSPLOC !LOCATION OF RESPONSE TYPE BYTE
      DATA         MSLLOC,RSTLOC,RSPLOC/2,5,2/
      INTEGER * 2  OUTTYP(16)     !ARRAY TO MATCH RESPONSE TYPE TO OUTPUT
      DATA         OUTTYP/1,7,2,5,3,7,4,4,4,2,4,4,4,6,2,2/
```

ORIGINAL PAGE IS OF POOR QUALITY

```
0008        INTEGER * 2  CMDL(16)        !CONSTANTS TO MATCH OUTPUT TYPE
      C                                   TO RESPONSE TYPE
0009        DATA         CMDL/3,3,3,3,3,0,4,4,3,3,4,4,3,3,3,3/
0010        REAL * 4     CMDR(16)        !CONSTANTS TO MATCH RESPONSE TYPE
      C                                   TO NAME
0011        LOGICAL * 1  CMD(4,16)
0012        EQUIVALENCE  (CMD,CMDF)
0013        DATA         CMDR/"CCK","CAN","CON","DAT","ENQ",0.0,"REZ0",
           .."REZ1","RIP","RST","SEZ0","SEZ1","SIP","DFB","ACK","NAK"/
0014        LOGICAL * 1  LPROM(4),LINE(80)      !INPUT BUFFER AND PROMPT FOR CRT
0015        DATA         LPROM/10,13,">","20)/
      C
      C     THE FOLLOWING ARE MESSAGES TO THE NIP AND MESSAGE LENGTHS
      C
0016        LOGICAL * 1  RSTMSG(5),ENQMSG(6),B32MSG(7),CANMSG(6)
0017        LOGICAL * 1  DFB34M(14),DFB33M(14),B23MSG(7),B34MSG(6)
0018        LOGICAL * 1  B6MSG(6),B7MSG(6),B8MSG(6),B9MSG(6),B4MSG(7)
0019        INTEGER * 2  DFB33L,DFB34L,B33MSL,B34MSL,B6MSL,B7MSL,B8MSL
0020        INTEGER * 2  B9MSL,B4MSL,CANMSL,RSTMSL,ENQMSL,B32MSL
0021        DATA         RSTMSL,ENQMSL,B32MSL/5,6,7/
0022        DATA         DFB33L,DFB34L,B33MSL,B34MSL,B6MSL/14,14,7,6,6/
0023        DATA         B7MSL,B8MSL,B9MSL,B4MSL,CANMSL/6,6,6,7,6/
      C     BLOCK 12 IS DECLARED INFREQUENTLY SO THE VARIABLES WHICH IT CONSISTS
      C         OF HAVE BEEN TIED TO BLOCK 34 BELOW.  IF IT IS DESIRED TO USE
      C         BLOCK 33 INSTEAD, USE THE DEFINITIONS COMMENTED OUT.
      C      DATA      DFB33M/"252,14,14,"72,33,39,37,26,22,21,42,28,41,"337/
      C      DATA          B33MSG/"252,4,7,"113,33,"R","215/
0024        DATA      DFB33M/"252,14,14,"72,34,39,37,26,22,21,42,28,41,"336/
0025        DATA         B33MSG/"252,4,7,"113,34,"R","214/
      C     TWO BLOCKS BELOW REQUEST SATELLITE HEALTH DATA
0026        DATA          B34MSG/"252,4,6,"114,34,"336/
0027        DATA        DFB34M/"252,14,14,"72,34,39,2,35,38,40,1,36,37,"372/
      C     THE FOLLOWING BLOCKS REQUEST EPHEMERIS DATA
0028        DATA         B6MSG/"252,4,6,"114,6,"372/
0029        DATA         B7MSG/"252,4,6,"114,7,"371/
0030        DATA         B9MSG/"252,4,6,"114,8,"370/
0031        DATA         B9MSG/"252,4,6,"114,9,"367/
      C     BLOCK 4 IS USED TO TEST THE NIP SINCE IT IS SENT FROM THE Z-SET
      C         BOTH BEFORE AND DURING NAVIGATION.
0032        DATA         B4MSG/"252,4,7,"113,4,"R","252/
```

ORIGINAL PAGE IS OF POOR QUALITY

```
C       RST RESTARTS THE NIP (WARM START)
0033    DATA          RSTMSG/"252,10,5,"107,0/
C       ENQ SENDS A REPETITIVE REQUEST FOR STATUS
0034    DATA          ENQMSG/"252,5,6,"113,"R","256/
C       B32 REQUESTS THAT BLOCK 32 BE SENT REPETIVELY
0035    DATA          B32MSG/"252,4,7,"113,32,"R","216/
C       CAN CANCELS ALL REPETITIVE MESSAGE REQUESTS IN NIP
0036    DATA          CANMSG/"252,2,6,"116,"377,1/
C
C  DATA VARIABLES
C
0037    LOGICAL * 1   BUFFB(500),BUFFA(500)  !BYTE MESSAGE BUFFERS
0038    LOGICAL * 1   OUTBUB(500)            !BYTE OUTPUT BUFFER
0039    INTEGER * 2   BUFWA(250),BUFWB(250)  !WORD MESSAGE BUFFERS
0040    INTEGER * 2   OUTBUW(250)            !WORD OUTPUT BUFFER
0041    EQUIVALENCE (BUFFB,BUFWB),(BUFFA,BUFWA)
0042    EQUIVALENCE (OUTBUB,OUTBUW)
0043    INTEGER * 2   ERRSAV,ERROA,ERROB     !ERROR FLAGS FOR BUFFER A AND B
C                                            AND TEMPORARY STORAGE
0044    INTEGER * 2   REQ                    !REQUEST TYPE VARIABLE
0045    LOGICAL * 1   DONEA,DONEB            !BUFFER FILLED FLAGS
0046    LOGICAL * 1   READ                   !FLAG TO CONTROL COMMAND ENTRY
0047    LOGICAL * 1   MORE                   !FLAG TO CONTROL OUTER LOOP
0048    LOGICAL * 1   FLAG                   !FLAG TO INDICATE OR COMPLETION STATUS
0049    INTEGER * 2   OLENB,OLENW  !LENGTH OF MESSAGE VARIABLES
0050    INTEGER * 2   RSPNSE !OUTPUT TYPE VARIABLE
0051    INTEGER * 2   I,J                    !TEMPORARY VARIABLES
C       REMAINING VARIABLES ARE FOR TIME (TOD), LATITUDE AND LONGITUDE
C           (LAT8,LON8), ALTITUDE (ALT4), SPEED (SPD4), ESTIMATED
C           POSITION ERROR (EPE4), GROUND TRACK (TRK4), AND LOGICAL
C           SATELLITE IN USE (LSAT).
C       OTHER VARIABLES AND EQUIVALENCE STATEMENTS SERVE TO LOAD THE
C           DATA FROM THE MESSAGES INTO THE MAIN VARIABLES.
0052    INTEGER * 2   TIME(2),TIME1,TIME2
0053    REAL * 4      TOD
0054    INTEGER * 2   LSAT
0055    INTEGER * 2   DAY,HOUR,MIN,SEC
0056    INTEGER * 2   LAT(4),LAT1,LAT2,LAT3,LAT4,ALT1,ALT2,TRK1,TRK2
0057    INTEGER * 2   LON(4),LON1,LON2,LON3,LON4,SPD1,SPD2,EPE1,EPE2
0058    INTEGER * 2   ALT(2),TRK(2),SPD(2),EPE(2)
```

ORIGINAL PAGE IS OF POOR QUALITY

```
0059          REAL * 8      LAT8,LON8
0060          REAL * 4      ALT4,SPD4,TRK4,EPE4
0061          INTEGER * 2   LATD,LATM,LATS,LOND,LONM,LONS
0062          EQUIVALENCE (TIME1,TIME(1),TOD),(TIME2,TIME(2))
0063          EQUIVALENCE (LAT1,LAT(1),LAT8),(LAT2,LAT(2)),(LAT3,LAT(3))
0064          EQUIVALENCE (LAT4,LAT(4)),(LON1,LON(1),LON8),(LON2,LON(2))
0065          EQUIVALENCE (LON3,LON(3)),(LON4,LON(4))
0066          EQUIVALENCE (ALT1,ALT(1),ALT4),(ALT2,ALT(2))
0067          EQUIVALENCE (TRK1,TRK(1),TRK4),(TRK2,TRK(2))
0068          EQUIVALENCE (SPD1,SPD(1),SPD4),(SPD2,SPD(2))
0069          EQUIVALENCE (EPE1,EPE(1),EPE4),(EPE2,EPE(2))
      C
      C  PROGRAM START
      C
0070          TYPE *,'          *** Norden Communications Software ***'
0071          CALL DZINIT                                  !INITIALIZE DZ CARD
      D       CALL DRINIT                                  !INITIALIZE DR CARD
0072          CALL NIPSTA                                  !INITIALIZE NIP
0073          CALL IPOKE ("44,"50100.OR.IPEEK("44))        !ENABLE TERMINAL INPUT
0074          FLAG = 1                                       !DR NOT IN USE FLAG SET
0075          MORE = .TRUE.                                  !NOT FINISHED FLAG SET
0076          CALL DZOUT (RSTMSG,RSTMSL)
0077          TYPE *,'A RESTART REQUEST HAS JUST BEEN SENT TO THE NIP.'
0078          CALL ISLEEP (0,0,1,0)                        !ONE SECOND DELAY
0079          CALL INITRD(BUFFA(1),DONEA,ERROA,BUFFB(1),DONEB,ERROB)
0080          TYPE *,'HIT ANY KEY TO BEGIN'
      C       DO WHILE (MORE)
0081  100     IF(.NOT.(MORE))GO TO 220
      C          IF (DONEA)
0083             IF(.NOT.(DONEA))GO TO 130
      C             THEN
      C             BUFFER A IS FILLED AND NEEDS TO BE COPIED
0085                ERRSAV=ERROA                           !SAVE COPY OF ERROR
0086                OLENB = BUFWA(MSLLOC)                  !COPY LENGTH
      D110           IF (FLAG.LE.0) GO TO 110              !WAIT FOR DR TO FINISH
0087                DO 120 I = 1,OLENB                     !COPY BUFFER
0088                   OUTBUB(I) = BUFFA(I)
0089  120           CONTINUE
0090                DONEA = 0                              !RESET DONE FLAG
0091                ERROA = 0                              !RESET ERROR FLAG
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
0092                GO TO 500                                  !CALL OUTPUT
0093                GO TO 160
      C           ELSE
      C             IF (DONEB)
0094  130           IF (.NOT.(DONEB)) GO TO 170
      C               THEN
      C               BUFFER B IS FILLED AND NEEDS TO BE COPIED
0096                    ERESAV = ERROR                         ! SAVE COPY OF ERROR
0097                    OLEN = BUFWB(MELLOC)                   !COPY LENGTH
      0140              IF (FLAG.LE.0) GO TO 140               !WAIT FOR OR TO FINISH
0098                    DO 150 I=1,OLEN                        !COPY BUFFER
0099                      OUTBUF(I)=BUFFB(I)
0100  150               CONTINUE
0101                    DONEB = 0                              !RESET DONE FLAG
0102                    ERROB = 0                              !RESET ERROR FLAG
0103                    GO TO 500                              !CALL OUTPUT
0104  160             GO TO 210
      C               ELSE
      C               NO MESSAGE AWAITING ATTENTION, CHECK KEYBOARD INPUT
      C               IF (ITTINR() .GE. 0)
0105  170             IF(.NOT.(ITTINR().GE.0))GO TO 210
      C                THEN
0107                      READ = 1
      C                   REPEAT
0108  180                    TYPE 1000,
0109  1000                   FORMAT (' RECOGNIZED COMMANDS:',/,
                           '      1) ENQUIRE',/,
                           '      2) TEST PATTERN',/,
                           '      3) CANCEL REQUESTS',/,
                           '      4) EPHEMERIS COLLECTION',/,
                           '      5) BLOCKS 32 & 34',/,
                           '      6) EXIT PROGRAM',/,
                           '      7) RESTART NIP')
0110                         CALL GTLIN(LINE,LPROM)
0111                         DECODE(1,1010,LINE,ERR=180) REQ
0112  1010                   FORMAT (I2)
      C                      CASE REQ OF
0113                         GO TO (300,310,320,330,340,350,360),REQ
0114                         GO TO 370
      C                         DOENQ:
```

ORIGINAL PAGE IS OF POOR QUALITY

```
0115  300                 CALL DZOUT (ENQMSG,ENQMSL)
0116                      READ = 0
0117                      GO TO 380
      C                 DOTST:
0118  310                 CALL DZOUT(B4MSG,B4MSL)
0119                      READ = 0
0120                      GO TO 380
      C                 DOCAN:
0121  320                 CALL DZOUT (CANMSG,CANMSL)
0122                      READ = 0
0123                      GO TO 380
      C                 DOEPH:
0124  330                 CALL DZOUT(DFB34M,DFB34L)
0125                      CALL DZOUT(B6MSG,B6MSL)
0126                      CALL DZOUT(B7MSG,B7MSL)
0127                      CALL DZOUT(B8MSG,B8MSL)
0128                      CALL DZOUT(B9MSG,B9MSL)
0129                      CALL DZOUT(B34MSG,B34MSL)
0130                      READ = 0
0131                      GO TO 380
      C                 DOBLO:
0132  340                 CALL DZOUT(DFB33M,DFB33L)
0133                      CALL DZOUT(B32MSG,B32MSL)
0134                      CALL DZOUT(B33MSG,B33MSL)
0135                      READ = 0
0136                      GO TO 380
      C                 DOEXE:
0137  350                 MORE=.FALSE.
0138                      READ = 0
0139                      GO TO 380
      C                 DORES:
0140  360                 CALL DZSTOP
0141                      CALL NIPSTA
0142                      CALL DZOUT (RSTMSG,RSTMSL)
0143                      TYPE *,'RESTART SENT, WAITING...'
0144                      CALL ISLEEP (0,0,1,0)
0145                      CALL DZINIT
0146                      CALL INITRD(BUFFA(1),DONEA,ERROA,
                          BUFFB(1),DONEB,ERROB)
0147                      TYPE *,'HIT ANY KEY TO BEGIN'
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
0148                                    READ = 0
0149                                    GO TO 380
      C                              ELSE:
0150  370                               GO TO 380
      C                           ENDCASE
      C                        UNTIL READ = 0
0151  380                      IF (READ.EQ.0) GO TO 210
0153                           GO TO 180
      C                     ENDIF
      C                 ENDIF
      C             ENDIF
      C          ENDWHILE
0154  210       GO TO 100
0155  220       RXENAB=.FALSE.          !TURN OFF CHARACTER CATCHING
0156            CALL DZSTOP
      D         CALL DRSTOP
0157            CALL EXIT
      C+-----------------------------------------------------------------------
      C         SUBROUTINE OUTPUT (CALLED BY GO TO STATEMENT TO SAVE TIME)
      C
      C         OUTPUTS DATA TO CRT AND TELEMETRY.  TYPES ERROR MESSAGES TO CRT.
      C
      C         CALLS TO:  DROUT
      C
      C         CALLS FROM:  NORINT
      C
      C         PARAMETERS:  NONE
      C
      C         COMMONS:  NONE
      C
      C         ENVIRONMENT:  NORDEN POP 11/34A  RT-11  FORTRAN
      C
      C         WRITTEN BY:  ARTHUR A. BECKMAN, UNIV. OF SO. COLORADO, 1984
      C
      C         RESTRICTIONS:  JUMPED TO BY GO TO STATEMENT TO SAVE TIME.
      C
      C-----------------------------------------------------------------------
0158  500       IF (.NOT.(ERRSAV.GT.0)) GO TO 550
      C            THEN
      C            ERROR PRESENT, IDENTIFY AND HANDLE.
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C       EVEN THOUGH THE ERROR FLAG IS SET THE BUFFER IS RETURNED ONLY WHEN
C        FINALLY FILLED WITH GOOD DATA.
0160      IF (.NOT.(ERRSAV.EQ.1)) GO TO 510
C          THEN
0162          TYPE *,'SYNC'
0163          GO TO 550
C       ELSE
0164  510   IF (.NOT.(ERRSAV.EQ.2)) GO TO 520
C          THEN
0166          TYPE *,'HEADER'
0167          GO TO 550
C          ELSE
0168  520     IF (.NOT.(ERRSAV.EQ.4)) GO TO 530
C              THEN
0170              TYPE *,'TRAILER'
0171              GO TO 550
C              ELSE
0172  530         IF (.NOT.(ERRSAV.EQ.8)) GO TO 540
C                  THEN
0174                  TYPE *,'COLLIS'
0175                  GO TO 550
C                  ELSE
0176  540             TYPE *,'MULT ERR'
C       ENDIF
0177  550 RSPNSE=OUTBUB(RSPLOC)        !SAVE RESPONSE TYPE
0178      OLENW=OLENB/2
C       IF (IAND(OLENB,1).EQ.1)       !ADJUST FOR ODD MESSAGE LENGTH
0179      IF(.NOT.(IAND(OLENB,1).EQ.1))GO TO 560
C       THEN
0181          OUTBUB(OLENB+1)=0
0182          OLENW=OLENW+1
C       ENDIF
C       IF ((RSPNSE.GE.1).AND.(RSPNSE.LE.16))
0183  560 IF(.NOT.((RSPNSE.GE.1).AND.(RSPNSE.LE.16)))GO TO 870
C       THEN
0185          TYPE 2000,OUTBUB(RSTLOC),
              (CMD(I,RSPNSE),I=1,CMDL(RSPNSE))
0186  2000    FORMAT('$STATUS:',O4,2X,3A,:,A)
0187          J=OUTTYP(RSPNSE)        !MAP RESPONSE TYPE TO OUTPUT TYPE
C             CASE J OF
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
0188            GO TO (700,710,770,780,790,830,850),J
0189            GO TO 860
      C         CCK:
0190  700           TYPE 2010,(OUTBUB(6+I),I=1,OLENB-7)
0191  2010          FORMAT(' ,',80A)
0192                GO TO 860
      C         ACK,CON,NAK,RST:
0193  710           RSPNSE=OUTBUW(4)
      C             IF (RSPNSE.LE.10)
0194                IF(.NOT.(RSPNSE.LE.10)) GO TO 750
      C                THEN                                    !RST, CON
0196                    TYPE 2020,(CMD(I,RSPNSE),I=1,CMDL(RSPNSE))
0197  2020              FORMAT('$,',3A,:,A)
      C                 IF (OLENB.GT.9)
0198                    IF(.NOT.(OLENB.GT.9))GO TO 730
      C                    THEN
      C                       IF (RSPNSE.EQ.10)
0200                          IF(.NOT.(RSPNSE.EQ.10)) GO TO 720
      C                        THEN                           !RST
0202                              TYPE 2030,OUTBUW(5)
0203  2030                        FORMAT('$,',O6)
0204                              GO TO 730
      C                        ELSE                           !CON
0205  720                         TYPE 2040,OUTBUW(5)
0206  2040                        FORMAT('$,',I2)
      C                       ENDIF
      C                 ENDIF
0207  730               GO TO 760
      C             ELSE                                      !ACK, NAK
0208  750               TYPE *,'???'
      C           ENDIF
0209  760         TYPE *,''
0210              GO TO 860
      C         ENQ:
0211  770         TYPE 2050,OUTBUW(4),OUTBUW(5),(OUTBUB(10+I),I=1,15)
0212  2050        FORMAT(X,/,' Zsts:',O7,10X,'Nipsts:',O7,/,
                  ' Evmsk: ',5(X,O3),/,' Evmskr:',5(X,O3),/,
                  ' Setid: ',I2,3X,'Setver:',I3,'.',I2,3X,
                  'Nipver:',I3,'.',I2,/)
0213              GO TO 860
```

```
C                 REZ0,REZ1,RIP,SEZ0,SEZ1,SIP:
0214  780            TYPE 2060,(OUTBUW(3+I),I=1,OLENW-4)
0215  2060           FORMAT(' ,',O6,':',/,9(X,8(O6,:,',',),/))
0216                 GO TO 860
C                 DAT:
0217  790            CONTINUE
D                    CALL DROUT (OUTBUB(1),FLAG)
C                    IF (OUTBUW(4).EQ.32)
C                     IF(.NOT.(OUTBUW(4).EQ.32))GO TO 810
C                        THEN
C                           TIME1=OUTBUW(9)
C                           TIME2=OUTBUW(10)
C                           TOD=TOD+OUTBUW(11)/10.
C                           DAY=TOD/86400.
C                           TOD=TOD-(DAY*86400.)
C                           DAY=DAY+1
C                           HOUR=TOD/3600
C                           TOD=TOD-(HOUR*3600)
C                           MIN=TOD/60
C                           SEC=TOD-(MIN*60)
C                           DO 800 I=1,4
C                              LAT(I)=OUTBUW(13+I)
C                              LON(I)=OUTBUW(17+I)
C800                        CONTINUE
C                           LATD=LAT8
C                           LAT8=ABS(60.*(LAT8-LATD))
C                           LATM=LAT8
C                           LATS=60.*(LAT8-LATM)
C                           LOND=LON8
C                           LON8=ABS(60.*(LON8-LOND))
C                           LONM=LON8
C                           LONS=60.*(LON8-LONM)
C                           ALT1=OUTBUW(22)
C                           ALT2=OUTBUW(23)
C                           ALT4=3.2808*ALT4
C                           SPD1=OUTBUW(24)
C                           SPD2=OUTBUW(25)
C                           SPD4=3.6*SPD4
C                           TRK1=OUTBUW(26)
C                           TRK2=OUTBUW(27)
```

ORIGINAL PAGE IS OF POOR QUALITY

```
      C                    EPE1=OUTBUW(28)
      C                    EPE2=OUTBUW(29)
      C                    EPE4=5.3968-4*EPE4
      C                    TYPE 2070,OUTBUB(RSTLOC),DAY,HOUR,MIN,SEC
      C2070                FORMAT(' Status:',I4,2X,'DAT.32',3X,'Time=',
      C                    [illegible])
      C                    [illegible],LATS,LOND,LONM,LONS,
      C                    [illegible]
      C2080                [illegible]
      C                    [illegible]
      C                    [illegible]
      C                    'Ground Speed:  ',F7.2,' Kt',5X,
      C                    'Ground Track:  ',F6.2,5X,
      C                    'Epe:  ',F7.3,' nm',/)
      C                    GO TO 820
      C                 ELSE
0218  810                  TIME1=OUTBUW(9)
0219                       TIME2=OUTBUW(10)
0220                       TOD=TOD+OUTBUW(11)/10.
0221                       TYPE 2090,OUTBUW(4),TOD
0222  2090                 FORMAT(' ,',I2,3X,'TIME=',F8.1)
      C                    IF (OUTBUW(4).EQ.34)
0223                       IF (.NOT.(OUTBUW(4).EQ.34)) GO TO 820
      C                       THEN
0225                            LSAT = OUTBUW(56)
0226                            TYPE 2095, LSAT
0227  2095                      FORMAT (' ,SAT = ',I2)
      C                    ENDIF
      C                 ENDIF
0228  820              GO TO 860
      C              DEB:
0229  830              TYPE 2100,OUTBUW(4)
0230  2100             FORMAT('$,',I2)
      C                IF (OLENB.GT.0)
0231                   IF(.NOT.(OLENB.GT.0))GO TO 840
      C                   THEN
0233                       TYPE 2110,(OUTBUB(8+I),I=1,OLENB)
0234  2110                 FORMAT('$,',65(I2,:,',',))
      C                ENDIF
0235  840              TYPE *,''
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
0236                    GO TO 860
      C               CAN,UNDEFINED:
0237  850                 GO TO 860
      C               ELSE:
      C                   GO TO 860
      C             END CASE
0238  860           GO TO 880
      C         ELSE:
0239  870           TYPE *,'ILLEGAL RESPONSE CODE'
      C       ENDIF
0240  880     GO TO 160
0241          END
```

ORIGINAL PAGE IS
OF POOR QUALITY

FORTRAN IV  Storage Map for Program Unit NORINT

Local Variables, .PSECT $DATA, Size = 003630 (  972. words)

| Name | Type | Offset | | Name | Type | Offset | | Name | Type | Offset | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ALT1 | I*2 | 003444 | Eqv | ALT2 | I*2 | 003446 | Eqv | ALT4 | R*4 | 003444 | Eqv |
| B32MSL | I*2 | 003476 | | B33MSL | I*2 | 003504 | | B34MSL | I*2 | 003506 | |
| B4MSL | I*2 | 003520 | | B6MSL | I*2 | 003510 | | B7MSL | I*2 | 003512 | |
| B8MSL | I*2 | 003514 | | B9MSL | I*2 | 003516 | | CANMSL | I*2 | 003522 | |
| DAY | I*2 | 003566 | | DFE33L | I*2 | 003500 | | DFE34L | I*2 | 003502 | |
| DONEA | L*1 | 003544 | | DONE6 | L*1 | 003545 | | ENQMSL | I*2 | 003474 | |
| EPE1 | I*2 | 003460 | Eqv | EPE2 | I*2 | 003462 | Eqv | EPE4 | R*4 | 003460 | Eqv |
| ERROA | I*2 | 003536 | | ERROB | I*2 | 003540 | | ERRSAV | I*2 | 003534 | |
| FLAG | L*1 | 003550 | | HOUR | I*2 | 003570 | | I | I*2 | 003560 | |
| J | I*2 | 003562 | | LATD | I*2 | 003576 | | LATM | I*2 | 003600 | |
| LATS | I*2 | 003602 | | LAT1 | I*2 | 003424 | Eqv | LAT2 | I*2 | 003426 | Eqv |
| LAT3 | I*2 | 003430 | Eqv | LAT4 | I*2 | 003432 | Eqv | LAT8 | R*8 | 003424 | Eqv |
| LOND | I*2 | 003604 | | LONM | I*2 | 003606 | | LONS | I*2 | 003610 | |
| LON1 | I*2 | 003434 | Eqv | LON2 | I*2 | 003436 | Eqv | LON3 | I*2 | 003440 | Eqv |
| LON4 | I*2 | 003442 | Eqv | LON8 | R*8 | 003434 | Eqv | LSAT | I*2 | 003564 | |
| MIN | I*2 | 003572 | | MORE | L*1 | 003547 | | MSLLOC | I*2 | 003464 | |
| OLENB | I*2 | 003552 | | OLENW | I*2 | 003554 | | READ | L*1 | 003546 | |
| REQ | I*2 | 003542 | | RSPLOC | I*2 | 003470 | | RSPNSE | I*2 | 003556 | |
| RSTLOC | I*2 | 003466 | | RSTMSL | I*2 | 003472 | | RXENAB | R*4 | 003612 | |
| SEC | I*2 | 003574 | | SPD1 | I*2 | 003454 | Eqv | SPD2 | I*2 | 003456 | Eqv |
| SPD4 | R*4 | 003454 | Eqv | TIME1 | I*2 | 003420 | Eqv | TIME2 | I*2 | 003422 | Eqv |
| TOD | R*4 | 003420 | Eqv | TRK1 | I*2 | 003450 | Eqv | TRK2 | I*2 | 003452 | Eqv |
| TRK4 | R*4 | 003450 | Eqv | | | | | | | | |

Local and COMMON Arrays:

| Name | Type | Section | Offset | Size | | Dimensions |
|---|---|---|---|---|---|---|
| ALT | I*2 | $DATA | 003444 | 000004 | (  2.) | (2) |
| BUFFA | L*1 | $DATA | 001450 | 000764 | (  250.) | (500) |
| BUFFB | L*1 | $DATA | 000464 | 000764 | (  250.) | (500) |
| BUFWA | I*2 | $DATA | 001450 | 000764 | (  250.) | (250) |
| BUFWB | I*2 | $DATA | 000464 | 000764 | (  250.) | (250) |
| B32MSG | L*1 | $DATA | 000337 | 000007 | (  4.) | (7) |
| B33MSG | L*1 | $DATA | 000410 | 000007 | (  4.) | (7) |
| B34MSG | L*1 | $DATA | 000417 | 000006 | (  3.) | (6) |
| B4MSG | L*1 | $DATA | 000433 | 000007 | (  4.) | (7) |
| B6MSG | L*1 | $DATA | 000425 | 000006 | (  3.) | (6) |

ORIGINAL PAGE IS
OF POOR QUALITY

FORTRAN-IV  Storage Map for Program Unit NDRINT

| | | | | | | |
|---|---|---|---|---|---|---|
| B7MSG | L*1 | $DATA | 000433 | 000006 | ( 3.) | (6) |
| B8MSG | L*1 | $DATA | 000441 | 000006 | ( 3.) | (6) |
| B9MSG | L*1 | $DATA | 000447 | 000006 | ( 3.) | (6) |
| CANMSG | L*1 | $DATA | 000346 | 000006 | ( 3.) | (6) |
| CMD | L*1 | $DATA | 000100 | 000100 | ( 32.) | (4,16) |
| CMDL | I*2 | $DATA | 000040 | 000040 | ( 16.) | (16) |
| CMDR | R*4 | $DATA | 000100 | 000100 | ( 32.) | (16) |
| DF833M | L*1 | $DATA | 000372 | 000016 | ( 7.) | (14) |
| DF834M | L*1 | $DATA | 000354 | 000016 | ( 7.) | (14) |
| ENQMSG | L*1 | $DATA | 000331 | 000006 | ( 3.) | (6) |
| EPE | I*2 | $DATA | 003460 | 000004 | ( 2.) | (2) |
| LAT | I*2 | $DATA | 003424 | 000010 | ( 4.) | (4) |
| LINE | L*1 | $DATA | 000204 | 000120 | ( 40.) | (80) |
| LON | I*2 | $DATA | 003434 | 000010 | ( 4.) | (4) |
| LPROM | L*1 | $DATA | 000206 | 000004 | ( 2.) | (4) |
| OUT8U8 | L*1 | $DATA | 002434 | 000764 | ( 250.) | (500) |
| OUT8UW | I*2 | $DATA | 002434 | 000764 | ( 250.) | (250) |
| OUTTYP | I*2 | $DATA | 000000 | 000040 | ( 16.) | (16) |
| RSTMSG | L*1 | $DATA | 000324 | 000005 | ( 3.) | (5) |
| SPD | I*2 | $DATA | 003454 | 000004 | ( 2.) | (2) |
| TIME | I*2 | $DATA | 003420 | 000004 | ( 2.) | (2) |
| TRK | I*2 | $DATA | 003450 | 000004 | ( 2.) | (2) |

Subroutines, Functions, Statement and Processor-Defined Functions:

| Name | Type | Name | Type | Name | Type | Name | Type | Name | Type |
|---|---|---|---|---|---|---|---|---|---|
| DZINIT | R*4 | DZOUT | R*4 | DZSTOP | R*4 | EXIT | R*4 | GTLIN | R*4 |
| IAND | I*2 | INITRD | I*2 | IPEEK | I*2 | IPOKE | I*2 | ISLEEP | I*2 |
| ITTINR | I*2 | NIPSTA | I*2 | | | | | | |

ORIGINAL PAGE IS
OF POOR QUALITY

```
        .TITLE  DZGPS   DZ CARD INTERFACE SOFTWARE
        .IDENT  /01/
;
;
; AUTHOR: AAB ON 5/30/74
;
;*****
; CHANGES:
;
;
; *****
; MODULE FUNCTION
;       THESE ROUTINES HANDLE ALL COMMUNICATIONS WITH THE DZ-11 BOARD.
;       DZINIT INITIALIZES THE DZ AND SETS UP THE INTERRUPT VECTOR.
;       INITRD SETS UP PARAMETERS FOR DZ READ OPERATIONS.
;       RXINT IS THE INTERRUPT DRIVEN INPUT ROUTINE ENABLED BY INITRD.
;       DZOUT HANDLES OUTPUT TO THE DZ.  DLSTOP TURNS OFF RXINT.
;       IAND LOGICALLY ANDS TWO VALUES.  NIPSTA PUTS NIP INTO ODT AND THEN
;       CAUSES A COLD START (JUMP TO ADDRESS ZERO).
;
;       THIS CODE IS POSITION INDEPENDENT AND WILL RUN UNDER THE FB MONITOR.

        .PAGE                   ; BREAK PAGE FOR PREFACE
        .NLIST  BEX             ; SUPPRESS BIN EXTENSION
;*****
; LOCAL SYMBOL DEFINITIONS:
        LNKCSR  = 160100
        LNKTCR  = LNKCSR + 4
        LNKRD   = LNKCSR + 2
        LNKLPR  = LNKCSR + 2
        LNKVEC  = 400
        TLM     = 252           ; MESSAGE START CONSTANT
        MSLOFF  = 2             ; MESSAGE OFFSET IN ARRAY
        HDRSIZ  = 6             ; MESSAGE HEADER SIZE
        PR0     = 0             ; PRIORITY ZERO
        PR7     = 340           ; PRIORITY SEVEN
        ASCZER  = 60            ; ZERO IN ASCII
        G       = 107           ; LETTER G IN ASCII
;
;
;*****
; LOCAL MACROS:

; *****
;
```

```
; *****
        .MACRO  FILL BUF,ERROR,MSL,SAV,?G0,?G01,?G02,?G03,?G04,?G05,?G06
G0:     MOVB    @#LNKR0,R1      ; READ THE BYTE
        BIC     #177400,R1      ; CLEAR THE UPPER BYTE
        TSTB    ENAB            ; IF ROUTINE NOT ENABLED
        BEQ     G03
        TST     I               ; WAIT FOR FIRST BYTE OF A BLOCK
        BNE     G01             ; IF NOT THEN BYPASS
        CMP     #TLM,R1         ;       ELSE TEST SYNC PATTERN
        BEQ     G01             ; BYPASS IF VALID PATTERN FOUND
        BISB    #1,@ERROR       ;       ELSE SYNC ERROR
        MOV     PC,R3
        ADD     #TERM-.,R3
        JMP     (R3)
G01:    MOVB    R1,@BUF         ; SAVE BYTE IN BUFFER
        INC     BUF             ; BUMP POINTER
        INC     I               ; BUMP BYTE COUNT
        ADD     R1,CKSM         ; ADJUST CHECKSUM
        CMP     #HDRSIZ,I       ; WAIT FOR END OF HEADER
        BNE     G02             ; IF NOT THEN BYPASS
        TSTB    CKSM            ; ELSE, IS CKSM1 ZERO?
        BEQ     G05             ; YES, CONTINUE
        BISB    #2,@ERROR       ;       ELSE CHECKSUM ERROR
        MOV     PC,R3
        ADD     #TERM-.,R3
        JMP     (R3)
G05:    MOV     SAV,R0          ; SAVE LENGTH
        ADD     #MSLOFF,R0
        MOV     (R0),MSL
G02:    CMP     MSL,I           ; TEST FOR COMPLETION
        BEQ     G04
G03:    MOV     PC,R3           ; IF NOT LAST BYTE THEN BYPASS
        ADD     #END-.,R3
        JMP     (R3)
G04:    TSTB    CKSM            ; CHECK FINAL CHECKSUM
        BEQ     G06             ; BRANCH IF GOOD
        BISB    #4,@ERROR       ;       ELSE ERROR
        MOV     PC,R3
        ADD     #TERM-.,R3
        JMP     (R3)
G06:    INC     GOODBU          ; SET GOOD BUFFER FLAG
        MOV     PC,R3           ; TERMINATE
        ADD     #TERM-.,R3
        JMP     (R3)
        .ENDM   FILL
;
; *****
; LOCAL DATA BLOCKS:
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
R1SAV:  .WORD   0               ; REGISTER SAVE AREA
R2SAV:  .WORD   0
R3SAV:  .WORD   0
R4SAV:  .WORD   0
ENAB:   .WORD   0               ; INTERRUPT ENABLE FLAG
BUFA:   .WORD   0               ; BUFFER A ADDRESS
DONA:   .WORD   0               ; DONE FLAG ADDRESS FOR A
ERRA:   .WORD   0               ; ERROR FLAG ADDRESS FOR A
BUFB:   .WORD   0               ; BUFFER B ADDRESS
DONB:   .WORD   0               ; DONE FLAG ADDRESS FOR B
ERRB:   .WORD   0               ; ERROR FLAG ADDRESS FOR B
CKSM:   .WORD   0               ; CHECKSUM
I:      .WORD   0               ; BYTE COUNTER
SAVBUA: .WORD   0               ; BUFFER A ADDRESS SAVE AREA
SAVBUB: .WORD   0               ; BUFFER B ADDRESS SAVE AREA
MSL:    .WORD   0               ; MESSAGE LENGTH STORAGE
MSLA:   .WORD   0               ; BUFFER A LENGTH
MSLB:   .WORD   0               ; BUFFER B LENGTH
ABUF:   .WORD   0               ; BUFFER IN USE FLAG
GOODBU: .WORD   0               ; GOOD BUFFER FLAG
DZAREA: .BLKW   3               ; EMT ARG BLOCK
DZLIST: .WORD   LNKCSR          ; LIST FOR .DEVICE
        .WORD   0               ; CLEAR CSR
        .WORD   LNKCSR + 4      ;       AND TRANSMIT CONTROL
        .WORD   0
DZVSAV: .WORD   0,0             ; RESTORE PREVIOUS VECTOR
        .WORD   0,0             ; RESTORE PREVIOUS INTERRUPT ADDRESS
        .WORD   0               ; END-OF-LIST
OFFVAL: .WORD   0               ; OFFSET VALUE FOR RELOCATION
TIME:   .WORD   0,0
EMTBLK: .WORD   0,0
; ASCII MESSAGE AREA

VECMSG: .ASCIZ  /?? RX-DRIVER-W-VECTOR ALREADY PROTECTED ??/
ERRMSSG:        .ASCII  /WAIT ERROR.../<15><12><200>
        .EVEN
;
;
; *****
; FUNCTION DETAILS:

;       INPUTS:         DZINIT HAS NO INPUTS.
;                       INITRD RECEIVES FROM THE CALLING FORTRAN ROUTINE:
;                               BUFA, BUFB: ADDRESSES OF DATA BUFFERS.
;                               DONA, DONB: ADDRESSES OF FLAGS TO CALLING
;                                           ROUTINE THAT BUFFERS FILLED.
;                               ERRA, ERRB: ADDRESSES OF FLAGS TO CALLING
;                                           ROUTINE OF ERROR CONDITIONS.
;                       BYTE BUFFER ADDRESS AND BUFFER SIZE ARE
```

```
                    CALLING ROUTINE.
                DZSTOP HAS NO INPUTS.
                IAND RECEIVES TWO VALUES FROM THE CALLING FORTRAN
                        ROUTINE TO BE LOGICALLY ANDED.
                NIPSTA HAS NO INPUTS.

OUTPUTS:        DZINIT HAS NO OUTPUT.
                INITRD HAS NO OUTPUT.
                RXINT RETURNS THE DONE FLAGS AND ERROR FLAGS.
                        THE ERROR FLAGS CONTAIN THE FOLLOWING:
                        BIT 0- SYNCHRONIZATION ERROR, MISSED DATA
                        BIT 1- ERROR IN HEADER CHECKSUM
                        BIT 2- ERROR IN TRAILER CHECKSUM
                        BIT 3- BUFFER NOT READ IN TIME. OVERWRITTEN.
                        BIT 4- MULTIPLE ERRORS.
                DZOUT SENDS DATA TO THE DZ-11.
                DZSTOP HAS NO OUTPUTS.
                IAND RETURNS ONE OR ZERO IN R0.
                NIPSTA HAS NO OUTPUT.

SIDE EFFECTS:   DZINIT, RXINT, DZOUT, IAND, NIPSTA AND DZSTOP HAVE
                        NO SIDE EFFECTS.
                INITRD CHANGES ALL OF THE PARAMETERS SENT BY THE
                        CALLING ROUTINE.


;*****
;START CODE HERE

        .PAGE
        .MCALL  .TTYOUT,.DEVICE,.PROTECT,.PRINT,.TWAIT,.RSET
        .GLOBL  IAND,NIPSTA,INITRD,RXINT,DZINIT,DZOUT,DZSTOP
;*****

;DZINIT INITIALIZES THE DZ CARD AND SETS UP THE VECTORS.

;*****
DZINIT:
        MOV     PC,R0           ; PROCESS FOR POSITION INDEPENDENCE
OFFSET:
        MOV     R0,OFFVAL       ; SAVE OFFSET VALUE
        MOV     #20,@#LNKCSR    ; CLEAR THE DZ11
10$:    BIT     #20,@#LNKCSR    ; CHECK THAT THE DZ11 CLEARED
        BNE     10$
        MOV     #17030,@#LNKLPR ; SET UP PORT PARAMETERS:
                                ; 9600 BAUD, NO PARITY, ONE STOP BIT,
                                ; EIGHT BITS
        BIS     #40,@#LNKCSR    ; ENABLE PORT 0 RECEIVER INTERRUPT
```

ORIGINAL PAGE IS OF POOR QUALITY

```
        BCC     1$                      ; BRANCH IF NO COLLISION
        CALL    WARN                    ; ELSE TAKE THEM AND WARN USER
1$:     MOV     #LNKVEC,DZVSAV          ; SAVE THEM IN THE DEVICE BLOCK
        MOV     @#LNKVEC,DZVSAV + 2
        MOV     #LNKVEC + 2,DZVSAV +4
        MOV     @#LNKVEC + 2,DZVSAV + 6
        .DEVICE #DZAREA,#DZLIST ; ENABLE TO CLEAR ON EXIT
        MOV     #RXINT-OFFSET,@#LNKVEC ; LOAD RECEIVE ROUTINE
                                        ; ADDRESS AT INTERRUPT VECTOR
        ADD     OFFVAL,@#LNKVEC ; AND ADD OFFSET VALUE
        MOV     #PR7,@#LNKVEC + 2 ; SET PSW FOR INTERRUPTS
        RTS     PC

WARN:   MOV     #VECMSG-OFFSET,R0       ; GET MESSAGE ADDRESS OFFSET
        ADD     OFFVAL, R0              ; ADJUST ADDRESS
        .PRINT
        RTS     PC

;*****
;
;  IAND LOGICALLY ANDS TWO VALUES
;
;*****
IAND:                                   ; LOGICAL AND FUNCTION
        MOV     @2(R5),R0
        MOV     @4(R5),R1
        COM     R1                      ; COMPLEMENT R1
        BIC     R1,R0
        RTS     PC

;*****
;
;  NIPSTA COLD STARTS THE NIP BY SENDING A BREAK WHICH PLACES THE NIP
;       INTO ODT, THEN A OG TO BRANCH TO ADDRESS ZERO.
;
;*****
NIPSTA:                                 ; RESET NIP
        MOV     R0,R0SAV                ; SAVE REGISTERS
        MOV     R1,R1SAV
        BIS     #400,@#LNKCSR + 6       ; SEND NIP A BREAK
                                        ; PUTS NIP INTO ODT
        MOV     #120.,R1                ; WAIT
        MOV     R1,TIME + 2
        .TWAIT  #EMTBLK,#TIME
        BCC     10$
        .PRINT  #ERRMSG
10$:    BIC     #400,@#LNKCSR + 6       ; TURN OFF BREAK
20$:    BIT     #100000,@#LNKCSR
```

ORIGINAL PAGE IS OF POOR QUALITY

```
        [illegible]     30$
        MOVB    #G,@#LNKCSR + 4 ; SEND ASCII G TO NIP
;                               ; NIP COLD STARTS
        MOV     R0SAV,R0        ; RESTORE REGISTERS
        MOV     R1SAV,R1
        RTS     PC

; *****
;
;  INITRD COPIES ADDRESSES AND DATA SENT BY THE CALLING ROUTINE,
;      SETS UP THE LOCAL VARIABLES, AND ENABLES INTERRUPTS.
;
; *****
INITRD:
        MOV     2(R5),BUFA      ; SAVE PARAMETER ADDRESSES, BUFFER A
        MOV     4(R5),DONA      ;       DONE FLAG FOR BUFFER A
        MOV     6(R5),ERRA      ;       ERROR FLAG FOR BUFFER A
        MOV     10(R5),BUFB     ;       BUFFER B
        MOV     12(R5),DONB     ;       DONE FLAG FOR BUFFER B
        MOV     14(R5),ERRB     ;       ERROR FLAG FOR BUFFER B
        CLRB    @DONA           ; CLEAR VARIABLES
        CLR     @ERRA
        CLRB    [illegible]
        CLR     [illegible]
        CLR     [illegible]
        CLR     MSL
        CLR     I
        CLR     CKSM
        MOV     BUFA,SAVBUA     ; SAVE BUFFER ADDRESSES
        MOV     BUFB,SAVBUB
        INC     ACUF            ; SELECT BUFFER A
        MOV     #1,ENAI         ; SET ENABLE INTERRUPTS FLAG
        BIS     #100,@#LNKCSR   ; ENABLE PROT 0 RECEIVER INTERRUPT
;                               ;       AND MASTER SCAN
        RTS     PC

; *****
;
;  RXINT RECEIVES INTERRUPTS FROM THE DZ CARD, READS THE INCOMING BYTE
;     HANDLES ERROR CHECKING, AND LOADS MESSAGE BUFFERS.
;
; *****
RXINT:
        MOV     R0,R0SAV
        MOV     R1,R1SAV                ; SAVE REGISTERS
        MOV     R2,R2SAV
        MOV     R3,R3SAV
        MOV     R4,R4SAV
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
        ADD     #FILLB-.,R3
        JMP     (R3)                    ; BRANCH IF BUFFER B
FILLA:  FILL    BUFA,ERRA,MSLA,SAVBUA   ; LOAD BUFFER A
FILLB:  FILL    BUFB,ERRB,MSLB,SAVBUB   ; LOAD BUFFER B
TERM:   TST     ABUF                    ; CHECK BUFFER IN USE
        BEQ     CHNGB                   ; BRANCH IF BUFFER B
        TST     GOODBU                  ; CHECK FOR ERRORS
        BEQ     CLRVAR                  ; DON'T SWITCH BUFFERS IF FOUND
        CLR     ABUF                    ; SWITCH BUFFERS
        CLR     GOODBU                  ; RESET BUFFER FULL FLAG
        INCB    @DONA                   ; SET DONE FLAG FOR BUFFER A
        TSTB    @DONB                   ; CHECK TO VERIFY THAT BUFFER B CLEARED
        BEQ     CLRVAR                  ; KEEP GOING IF NO COLLISION
        BIS     #10,@ERRB               ; COLLISION ERROR
        BR      CLRVAR
CHNGB:  TST     GOODBU                  ; CHECK FOR ERRORS
        BEQ     CLRVAR                  ; DON'T SWITCH BUFFERS IF FOUND
        INC     ABUF                    ; SWITCH BUFFERS
        CLR     GOODBU                  ; RESET BUFFER FULL FLAG
        INCB    @DONB                   ; SET DONE FLAG FOR BUFFER B
        TSTB    @DONA                   ; CHECK TO VERIFY THAT BUFFER A CLEARED
        BEQ     CLRVAR                  ; KEEP GOING IF NO COLLISION
        BIS     #10,@ERRA               ; COLLISION ERROR
CLRVAR: TST     ABUF
        BEQ     10$
        MOV     SAVBUA,BUFA             ; RESTORE START ADDRESS
        BR      20$
10$:    MOV     SAVBUB,BUFB             ; RESTORE START ADDRESS
20$:    CLR     MSL                     ; CLEAR MESSAGE LENGTH
        CLR     CKSM
        CLR     I                       ; CLEAR COUNT
END:    MOV     R0SAV,R0
        MOV     R1SAV,R1                ; RESTORE REGISTERS AND RETURN
        MOV     R2SAV,R2
        MOV     R3SAV,R3
        MOV     R4SAV,R4
        RTI

;*****
;
; DZOUT OUTPUTS A STRING TO THE DZ OUTPUT BUFFER (AND SO TO THE NIP)
;
;*****
DZOUT:
        MOV     2(R5),R1                ; STORE BUFFER'S ADDRESS
        MOV     @4(R5),R2               ; # OF BYTES TO SEND
10$:    BIT     #100000,@#LNKCSR        ; READY TO SEND?
        BEQ     10$                     ; NO: THEN LOOP
        MOVB    (R1)+,@#LNKCSR          ; YES: OUTPUT BYTE
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
; *****
;
; DZSTOP TURNS OFF THE INTERRUPT ROUTINE AND THE DZ CARD
;
; *****
DZSTOP:
        DEC     ENAB            ; TURN OFF INTERRUPT SERVICE ROUTINE
        BIC     #100,@#LNKCSR   ; TURN HARDWARE INTERRUPTS OFF
        RTS     PC

        .END
```

```
.TITLE  DRGPS  CONTROL DR CARD
.IDENT  /01/

;AUTHOR:        AAB ON 4/13/84

;*****
;CHANGES:

;*****
;MODULE FUNCTION
;       THIS SET OF ROUTINES CONTROLS COMMUNICATIONS TO THE PARALLEL PORT
;       CONNECTED TO THE RMOU
;
;       CODE IS POSITION INDEPENDENT AND CAN BE USED WITH EB MONITOR
;
;       DR DETAILS:
;               DRCSR   167770
;               DROUTBUF 167772
;               DRINBUF 167774
;               PRIORITY 5 (USER SELECTABLE ON BOARD)
;               VECTORS 300 (REQUEST A) DR 304 (REQUEST B)
;               BIT 6 IN CSR SETS IEA TO ALLOW INTERRUPTS WHEN REQA IS SET BY
;                       DEVICE
;
;
.PAGE                   ; START PAGE FOR PREFACE
.NLIST  BEX             ; SUPPRESS BIN EXTENSION
;*****
;LOCAL SYMBOL DEFINITIONS:
        DRVEC   = 300
        DRCSR   = 167770
        DROUTBUF= DRCSR + 2
        DRMSLOFF= 2             ; LENGTH OFFSET IN BUFFER
        PR0     = 0             ; NO PRIORITY
        PR7     = 340           ; MAXIMUM PRIORITY
        ZERO    = 0
        DROVER  = 0             ; OVERHEAD
;
;*****
;LOCAL MACROS:
;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
R0SAVDR:  .WORD   0
R1SAVDR:  .WORD   0                    ; REGISTER SAVE AREA
R2SAVDR:  .WORD   0
OFFVAL:   .WORD   0
USEBUF:   .WORD   0                    ; BUFFER IN USE FLAG
DRBUE:    .WORD   0                    ; OUTPUT BUFFER ADDRESS
DRENAB:   .WORD   0                    ; INTERRUPT ENABLE FLAG
DRLEN:    .WORD   0                    ; LENGTH OF BUFFER
DRDONE:   .WORD   0                    ; ADDRESS OF DONE FLAG
DRAREA:   .BLKW   3                    ; EMT ARG BLOCK
DRLIST:   .WORD   DRCSR                ; LIST FOR .DEVICE
          .WORD   0                    ; CLEAR CSR
DRVSAV:   .WORD   0,0                  ; RESTORE PREVIOUS VECTOR
          .WORD   0,0                  ; RESTORE OLD INTERRUPT ADDRESS
          .WORD   0                    ; END-OF-TABLE
;
;
; *****
; FUNCTION DETAILS:

;         INPUTS:        CALLING ROUTINE SENDS BUFFER ADDRESS AND DONE FLAG
;                        ADDRESS TO DROUT.  DRSTOP AND DRINIT RECEIVE NO
;                        INPUTS.  DRINT IS INTERRUPT DRIVEN.
;
;         OUTPUTS:       DROUT RETURNS DONE FLAG TO FORTRAN ROUTINE.
;                        DRINIT OUTPUTS CONTROL WORD TO DR DEVICE.
;                        DRSTOP HAS NO OUTPUTS.
;                        DRINT OUTPUTS EITHER ZERO OR DATA FROM THE BUFFER
;                        RECEIVED FROM THE FORTRAN ROUTINE TO THE DR.
;
;         SIDE EFFECTS:  DONE FLAG CHANGED.
;
; *****
; START CODE HERE

          .PAGE
          .MCALL  .DEVICE,.PROTECT
          .GLOBL  DRINIT, DROUT, DRINT, DRSTOP
; *****
;
; DRINIT INITIALIZES THE DR BOARD AND SETS UP THE VECTOR
;
; *****
DRINIT:
          MOV     R0,R0SAVDR
          MOV     PC,R0
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
        MOV     #DRVEC,DRVSAV
        MOV     @#DRVEC,DRVSAV + 2
        MOV     #DRVEC + 2,DRVSAV + 4
        MOV     @#DRVEC + 2,DRVSAV + 6
        .DEVICE #DRAREA,#DRLIST
        MOV     #DRINT-OFFSET,@#DRVEC   ; LOAD INTERRUPT ROUTINE ADDRESS
                                        ;       AT INTERRUPT VECTOR
        ADD     OFFVAL,@#DRVEC
        MOV     #PR7,@#DRVEC + 2        ; SET PSW FOR PRIORITY 7
        MOV     #100,@#DRCSR            ; ENABLE INTERRUPTS
        INC     DRENAB                  ; ENABLE INTERRUPT ROUTINE
        MOV     R0SAVDR,R0
        RTS     PC

;*****
;
; DROUT HANDLES SET UP TO OUTPUT A BUFFER WHEN CALLED BY THE FORTRAN
;       ROUTINE.  DRINT WILL OUTPUT THE BUFFER.
;
;*****
DROUT:
        MOV     R1,R1SAVDR              ; SAVE REGISTER
        MOV     2(R5),DRBUF             ; SAVE ADDRESSES OF BUFFER
        MOV     4(R5),DRDONE            ;                      DONE FLAG
        CLRB    @DRDONE                 ; CLEAR DONE FLAG
        MOV     DRBUF,R1                ; FETCH LENGTH WORD TO DRLEN
        ADD     #DRMSLOFF,R1
        MOV     (R1),DRLEN
        ADD     #DROVER,DRLEN           ; CORRECT FOR OVERHEAD
        BIT     #1,DRLEN                ; CHECK IF ODD # OF BYTES
        BEQ     10$                     ; BRANCH IF NOT
        MOV     DRBUF,R1                ; ZERO LAST BYTE
        ADD     DRLEN,R1
        MOVB    #ZERO,(R1)
        INC     DRLEN                   ; COUNT LAST BYTE
10$:    ASR     DRLEN                   ; CHANGE BYTE COUNT TO WORD COUNT
        MOV     R1SAVDR,R1              ; RESTORE REGISTER
        INC     USEBUF                  ; SET USE BUFFER FLAG
        RTS     PC

;*****
;
; DRSTOP DISABLES DRINT AND LOADS THE DR OUTPUT BUFFER WITH ZERO
;
;*****
DRSTOP:
        CLR     DRENAB                  ; DISABLE INTERRUPT ROUTINE
        MOV     #ZERO,@#DROUTBUF        ;       ; SEND ZEROS TO RMOU
```

```
;
;  DRINT OUTPUTS ONE WORD TO THE DR OUTPUT BUFFER AT EACH INTERRUPT.
;        WHEN NO DATA IS AVAILABLE, A ZERO IS OUTPUT TO SEPARATE MESSAGES.
;        MESSAGE FORMAT IS THE SAME AS DESCRIBED IN THE NIP USER'S MANUAL.
;
;*****
DRINT:
        MOV     R2,R2SAVDR          ; SAVE REGISTERS
        TSTB    DRENAB              ; VERIFY ROUTINE ENABLED
        BNE     10$                 ; BRANCH IF TRUE
        BR      DREND
;       MOV     PC,R2               ; OTHERWISE EXIT
;       ADD     #DREND-.,R2
;       JMP     (R2)
10$:    TST     USEBUF              ; CHECK FOR BUFFER IN USE
        BNE     30$                 ; BRANCH IF BUFFER PRESENT
        MOV     #ZERO,@#DROUTBUF    ; OTHERWISE OUTPUT ZERO,
        BR      DREND
;       MOV     PC,R2               ; AND EXIT
;       ADD     #DREND-.,R2
;       JMP     (R2)
30$:    MOV     @DRBUF,@#DROUTBUF   ; OUTPUT NEXT WORD
        INC     DRBUF               ; INCREMENT ADDRESS TO NEXT WORD
        INC     DRBUF
        DEC     DRLEN               ; DECREMENT COUNTER
        TST     DRLEN               ; TEST IF LAST WORD SENT
        BEQ     40$                 ; BRANCH IF BUFFER FINISHED
        BR      DREND
;       MOV     PC,R2               ; OTHERWISE EXIT
;       ADD     #DREND-.,R2
;       JMP     (R2)
40$:    INCB    @DRDONE             ; SET DONE FLAG
        CLR     USEBUF              ; CLEAR USE BUFFER FLAG
DREND:  MOV     R2SAVDR,R2          ; RESTORE REGISTERS
        RTI
        .END
```

ORIGINAL PAGE IS
OF POOR QUALITY

# APPENDIX C

REPORT OF CONTRACT WORK

FOR UNIVERSITY OF SOUTHERN COLORADO

JULY 1983 - AUGUST 1983

TIM R. JOHNSON

My main task during this two-month period was to install the RSX11M operating system on the Helicopter Group Lab's LSI 11/23 computer.

The LSI 11/23 computer system is connected to various peripherals (see diagram) including a Ramtek-9400 color graphic system, voice input and output devices, digitizer tablet, and touch panel. The system is used as a general purpose graphics workstation for design of interactive color displays and concepts. The computer had been running under the RT-11 operating system, but we decided that the RSX11M operating system (multi-user, multi-tasking) would make better use of the hardware resources for the following reasons:

1) timesharing - two or more people could be developing, editing, compiling, or doing other tasks at the same time.

2) bigger program space - RSX11M makes better use of the 128 kwords of memory present on the system, allowing programs up to 32 kwords in size (larger if overlayed).

3) multi-programming - more than one process can run simultaneously, allowing more overall throughput, better real-time response for those programs that need it, and new programming options such as inter-task communication and parallelism.

4) improved security and organization - files and programs can be logically grouped into separate directories, and files can be protected against accidental or unauthorized deletion or alteration.

The work of converting to RSX11M consisted mainly of writing, modifying, and installing new system software drivers for the various peripherals, testing these, and writing new I/O routines in the graphics support library to use the new drivers. In this way, many RT-11 programs that were written for the color graphics system and other devices would

need very few changes made to them in order to run under the new RSX11M operating system. In addition, some new features offered by the RSX11M drivers could be used by all programs, present and future, in a simple, general way.

The following drivers were installed:

a) serial drivers for the touchpanel and Voterm - I decided to use the RSX11M terminal driver (TTDRV) to communicate with our two serial devices, the touchpanel and Voterm voice recognition unit. The use of the system terminal driver allows a separate interrupt-driven 'typeahead' buffer for each device, which can then be queried by a program when it wants input from the device. The buffer can be checked to see if any characters are in it, and the program can specify whether to wait until some are (hang mode) or return to the program and continue (nohang mode). In this way, both touchpanel and Voterm can be used as dual, either/or input devices, which provides flexibility in programming and built-in backup should one or the other device be unavailable.

b) graphics printer - The Hewlett-Packard graphics printer was connected to a dedicated IBV11 interface and cable (general purpose interface bus, IEEE-488 standard), and I modified the RSX11M standard line printer driver (LPDRV) to run the HP printer. The main changes were that the new HP driver has to address the HP printer as a GPIB device on startup as well as when the printer goes offline, since the printer will become un-addressed if powered off during printing (to clear a paper jam, for instance). Also, the new HP driver must pass all characters to the printer, since the driver will also be used to print from the Ramtek screen. This will involve sending binary data to the printer (with the printer in graphics mode) so tabs and other control characters must cause no special action on the part of the new HP driver.

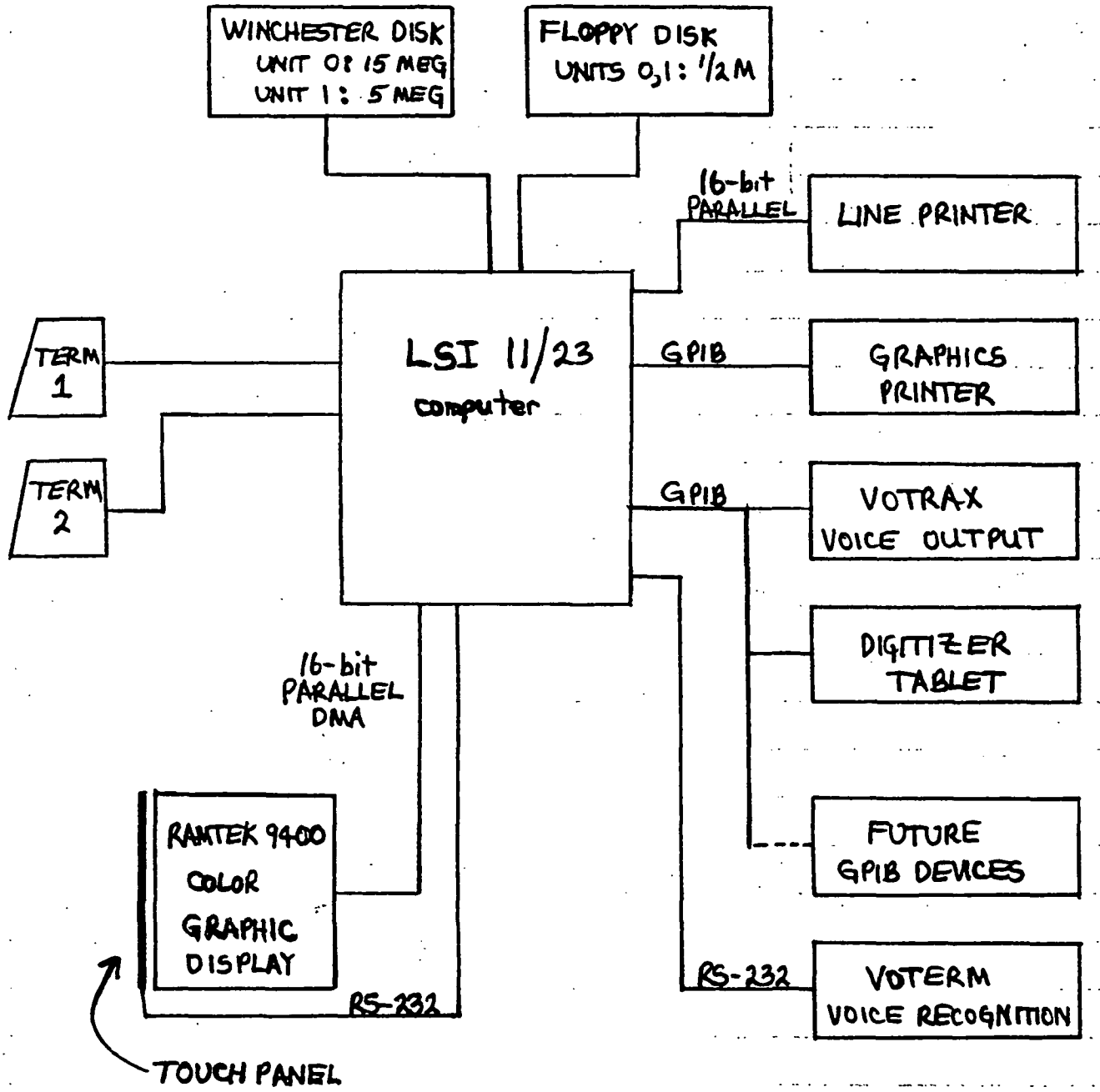
c) Ramtek driver - A device driver for the Ramtek-9400 was obtained and installed, and new I/O routines for the Ramtek graphic support library (Ramlib) were written to use the new driver.

The driver uses direct-memory access to communicate with the Ramtek device, so I/O to the Ramtek is faster now than under RT-11, which used programmed I/O.

d) GPIB driver (for Votrax and digitizer tablet) - A new driver, IBDRV, was obtained and installed to communicate with the Votrax voice synthesizer and the HP digitizer tablet, both devices on the GPIB (IEEE-488) bus. Future devices can be simply plugged onto the external cable for this bus, set to a unique GPIB bus address, and programmed through the IBDRV driver. Two new I/O routines, IBREAD and IBWRITE, were written and included in the Ramtek support library to commmunicate with GPIB devices. In addition, the routine IBSETMODE allows the IB: driver to be set up in four different read and write modes, as well as specifying the end-of-data character for a read of a GPIB device.

The conversion of old RT-11 graphic programs to the new RSX11M system was fairly straightforward. Some procedure names had to be changed, ( ie from Writedriver, Readdriver to Ibwrite, Ibread ), and the touchpanel and Voterm routines have a new parameter 'hangmode', a boolean, to specify whether to wait for input or return immediately if there is none. In addition, a new routine, Flushbuffer, was added to the support library. This routine flushes the touchpanel or Voterm 'typeahead' buffer, since there will be some cases where a program does not want any previous input from the device but only input that occurs after a specific request for input from that device. The usual case when this happens is during successive menu box selection using the touchpanel, where two or three 'hits' for the same menu box will be stored in the typeahead buffer, but all those after the first hit are redundant and possibly wrong if taken to be the input for the next series of menu box tests on a new display frame, for instance.

# SYSTEM CONFIGURATION



# APPENDIX D

NASA

SUMMER

PROJECT

June-August, 1983

by

Richard W. Parker

to

Dr. Don Cottrell

During the summer of 1983 at NASA Ames Research Center, located at Moffett Field in California, I performed various technical and development tasks. The tasks that were performed and required of me were to assist Ralph Carestia in the development of a few boards used on the project and to do technical tasks on projects previously started.

I assembled two Intel SDK-86 single board computers and trouble-shot another one to be used for training purposes by Ralph. The SDK-86's were very enlightening to assemble, because they gave me an understanding of a 16 bit computer system and some familiarity with 8086 assembly language.

The boards that were assembled and developed were couple port I/O boards, a 16 port DAC board, a digital to syncro board, a digital to analog discrete board, and an analog to digital discrete board. With regard to the later two boards, I was instrumental in their design. The digital to analog changed a digital word from a S-100 interface to 28vdc and 0vdc for the simulator use, the analog to digital changed the 28vdc input to a logic 5vdc level used in constructing words utilized by the computer.

The digital to analog board used a DG162 for a switching device for operating a transistor, used as a current sinking device, switching developing the 28vdc used as an input to the simulator.

The analog to digital board used a comparator (LM 311) to change a 28vdc input to a logic 5vdc level. I used a debounce circuit to eliminate unwanted spikes caused by the switches bouncing when the operator used them. A storage register (D flip/flop) is used to latch the information until the computer requests the data. A signal from the computer resets the register after it receives the data to enable sampling of the word again. The signal used for the reset function was the Data Transmitted line.

The summer employment was very enjoyable and a good experience for students to have. The expense was a little rough if the students have some financial responsibility elsewhere. But I enjoyed the experience and thank the university for selecting me to participate in the program and I hope the program will be able to continue.