NASA Conference Publication 2158

Aerospace Applications of Microprocessors

Preprint for a workshop
held in Greenbelt, Maryland
November 3-4, 1980

NASA Conference Publication 2158

Aerospace Applications of Microprocessors

Preprint for a workshop sponsored by
NASA Goddard Space Flight Center,
Greenbelt, Maryland, and the
American Institute of Aeronautics
and Astronautics, New York,
and held in Greenbelt, Maryland,
November 3-4, 1980

NASA
National Aeronautics and
Space Administration

**Scientific and Technical
Information Office**

1980

FOREWORD

NASA/Goddard Space Flight Center (GSFC), in cooperation with the AIAA Technical Committee on Computer Systems, sponsored this workshop on Aerospace Applications of Microprocessors. The rapidly increasing capabilities and decreasing costs of digital computing systems in general, and microprocessors in particular, have meant orders of magnitude increases in their use in aerospace systems, particularly onboard satellites and aircraft.

The objectives of the workshop were to assess the state of microprocessor applications and to identify current and future requirements and associated technological advances which allow effective exploitation of this rapidly advancing technology. There were four sessions in the workshop:

I. Air/Space Applications of Microprocessors;
II. Ground Based Aerospace Microprocessor Applications;
III. Microprocessor Software Technology; and
IV. Microprocessor Hardware Technology.

This document contains only a synopsis and key figures of each presentation. The synopses and figures were submitted as camera-ready copies prior to the workshop. Only minor editorial changes have been made.

In addition to the formal presentations, the workshop was structured to provide time for audience interaction. On the evening of November 3, a panel discussion on "Are Microprocessor Trends and Aerospace Requirements Heading in the Same Direction?" was held. The panelists were:

Terry Straeter, General Dynamics Data Systems Service, (Moderator);
Rocky A. Evans, Military Products Manager, Intel Corporation;
Adrian Hooke, Jet Propulsion Laboratory;
Charles Husson, Langley Research Center; and
John Shea, Vice-President Integrated Circuit Electronics.

The workshop was organized by a subcommittee of the AIAA technical committee on computer systems. Co-chairmen were:

John Sos, Goddard Space Flight Center
Terry Straeter, General Dynamics Data Systems Services.

Other committee members were:

M. Kelly, Sperry Flight Systems;
T. McTigue, McDonnell Douglas Aircraft;
R. Schwartz, McDonnell Douglas Astronautics; and
T. Smith, NAVAIR Systems Command; and

NASA/GSFC members were:

E. Connell
R. Nelson.

Use or identification of commercial products in this document does not constitute an official endorsement of such products or their manufacturers, either expressed or implied, by NASA.

John Y. Sos
Program Co-chairman

Page Intentionally Left Blank

CONTENTS

SESSION I

AIR/SPACE APPLICATIONS OF MICROPROCESSORS

Preceding Page Blank

AN IMAGING INFRARED (IIR) SEEKER USING A MICROPROGRAMMED PROCESSOR

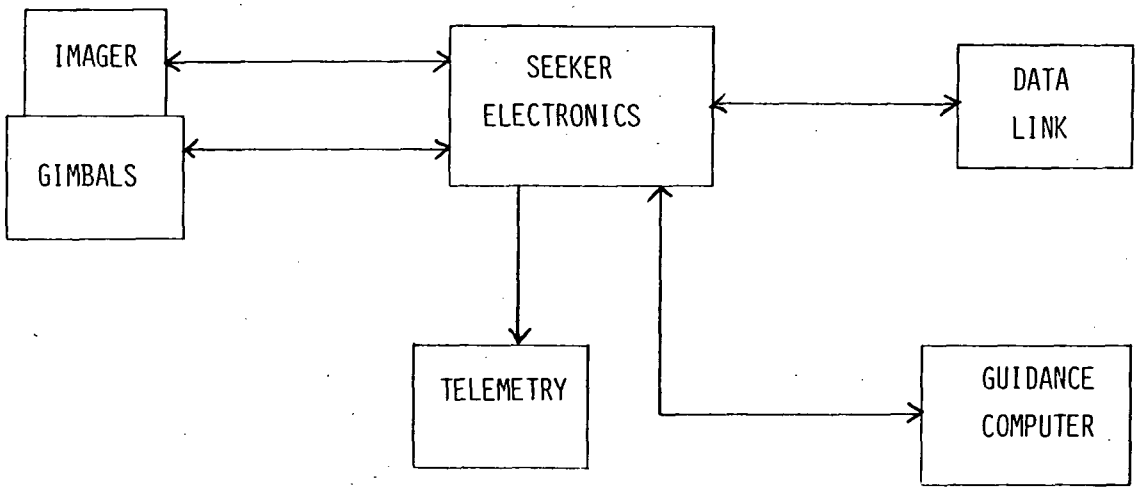
Kerry V. Richmond
McDonnell Douglas Astronautics Co.
St. Louis, Missouri

A recently developed IIR seeker uses a microprogrammed processor to perform gimbal servo control and system interface via a MIL-STD 1553 port while performing the seeker functions of automatic target detection, acquisition and tracking. Although the acquisition and centroid tracking are relatively low computation load seeker modes, the automatic detection mode requires up to 80% of the available capability of a high performance 2900 based microprogrammed processor. With the high speed processing capability available it is possible to implement a digital servo in the same processor using only 5% of the computation capacity. This digital servo includes six modes of gimbal control at the basic processor 60 Hz computation loop plus a 200 Hz rate loop, the latter being transparent to the main seeker functions. These two asynchronous timing loops plus a 50 Hz system interface loop driven from the 1553 port are implemented in the one processor. The fast response required by the rate loop for the rate sensor demodulator inputs also requires an interrupt driven analog data acquisition system. A 4K microcode program driven by eight interrupts implements these functions as well as the other operator and system interfaces.

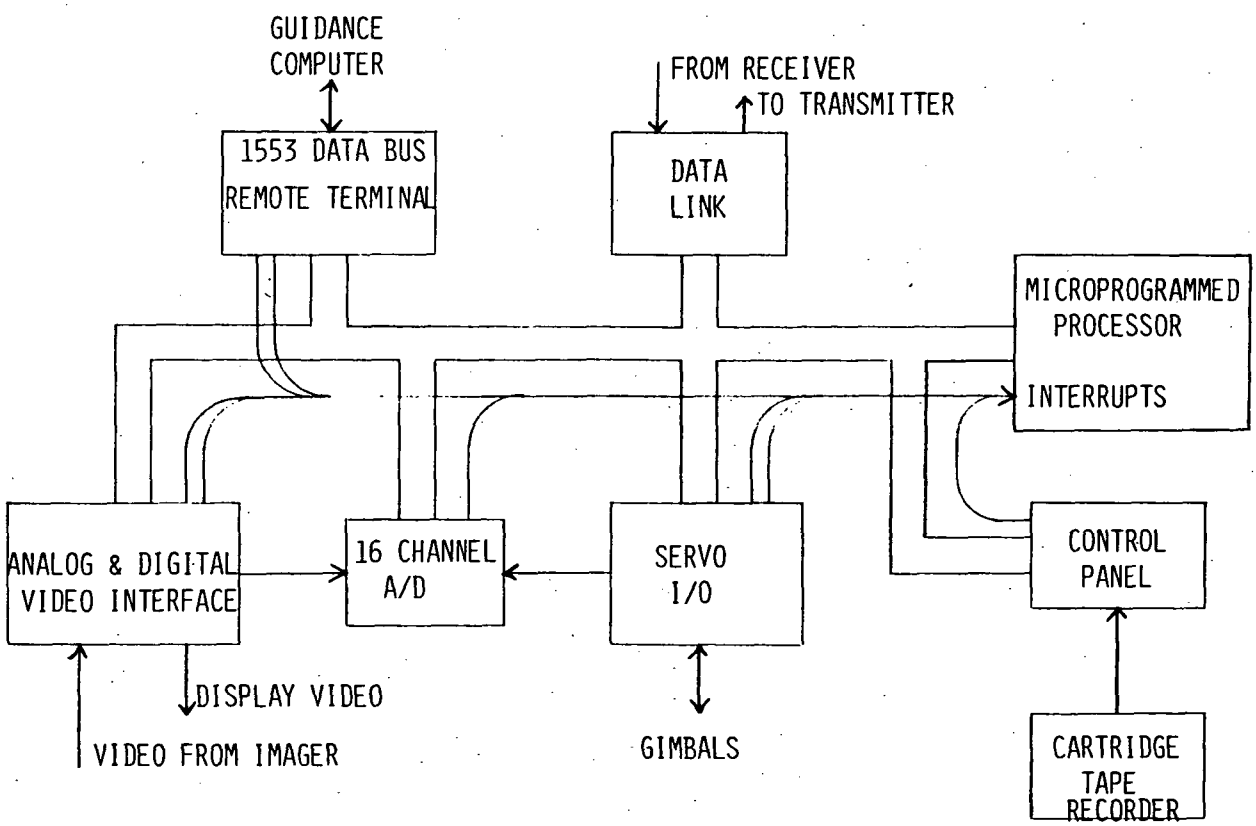
The eighth interrupt is used to force the processor into special "front panel" code which suspends all other interrupt processing and saves the state of the processor to allow the programmer to view the contents of all registers and memory as well as enter new values and resume normal processor execution at the interrupted location or any other selected location. This programmer debug aid in the hardware coupled with a set of support software including a symbolic cross assembler and a software simulation of the 2900 based processor allow efficient program development and checkout.

This system developed around the microcoded processor required by one of the system tasks has been designed, checked out and flown successfully. Although system complexity was increased significantly by adding the additional functions this approach can be cost effective when the basic computation capacity is already available.

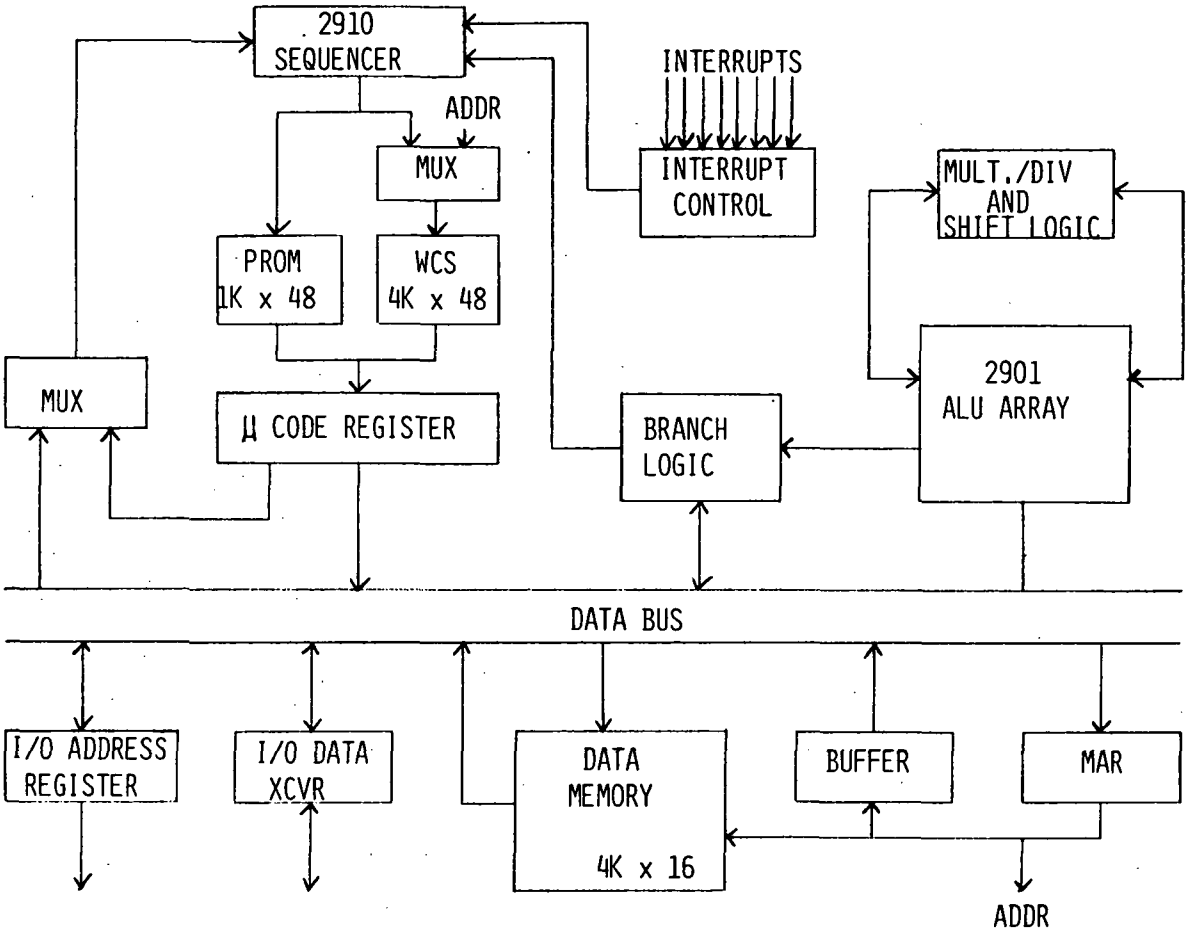
IIR SEEKER MISSILE INTERFACES



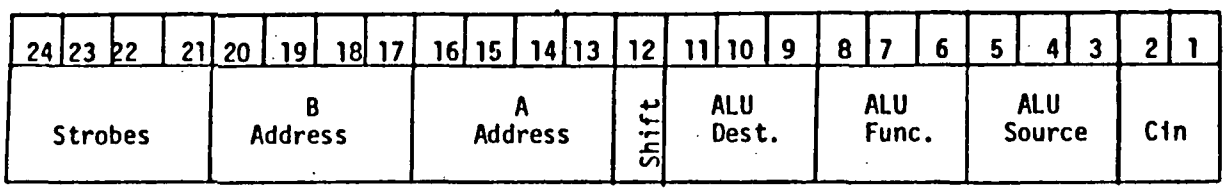
SEEKER BLOCK DIAGRAM

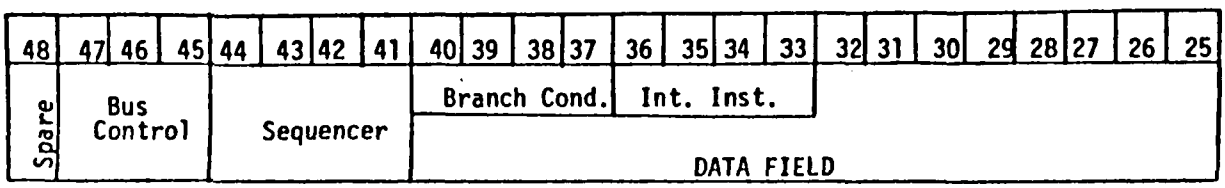


CPU BLOCK DIAGRAM



MICROCODE WORD





CPU PERFORMANCE/REQUIREMENTS

- 2900 BIT SLICE MICROPROGRAMMED PROCESSOR
- 48 BIT WIDE MICROCODE WORD
- 267 NANOSECOND CYCLE TIME
- 4K PROGRAM MEMORY
- 2K SCRATCH PAD MEMORY
- 8 INTERRUPTS

SEEKER INTERRUPTS

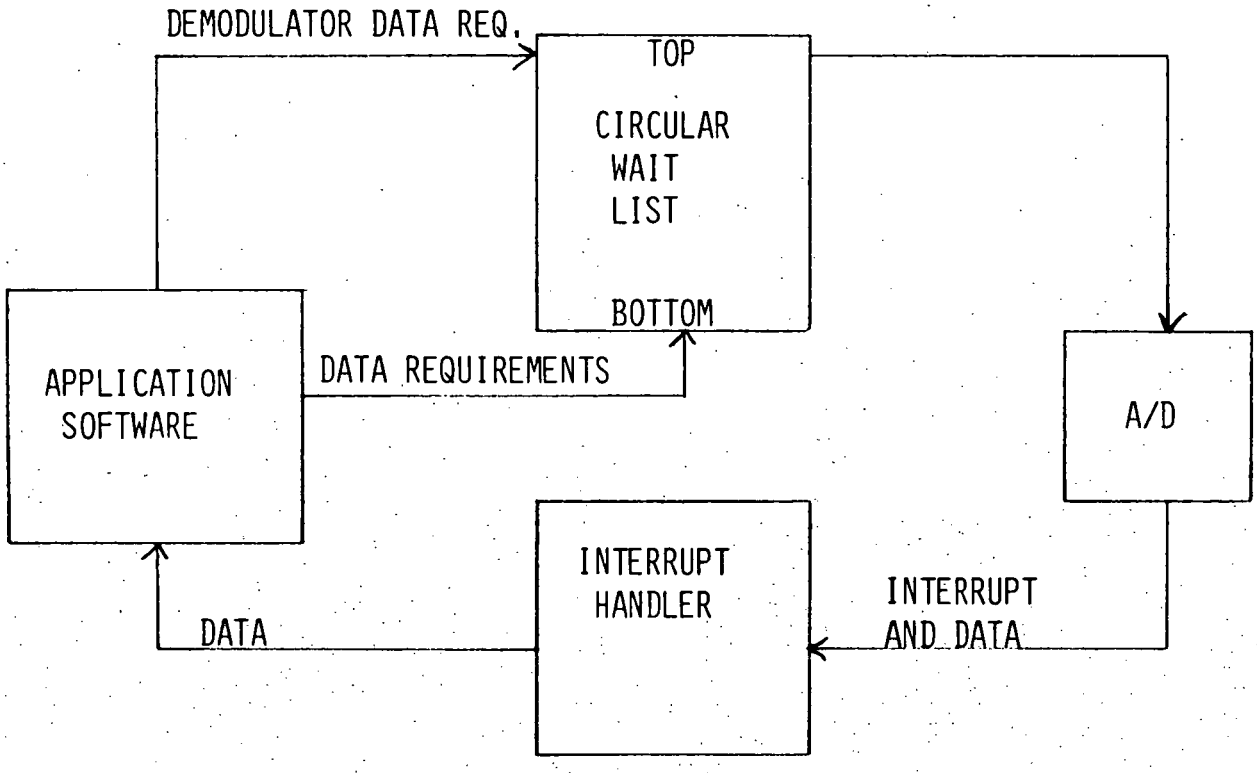
SYSTEM INTERRUPTS
- CONTROL PANEL
- A/D COMPLETION

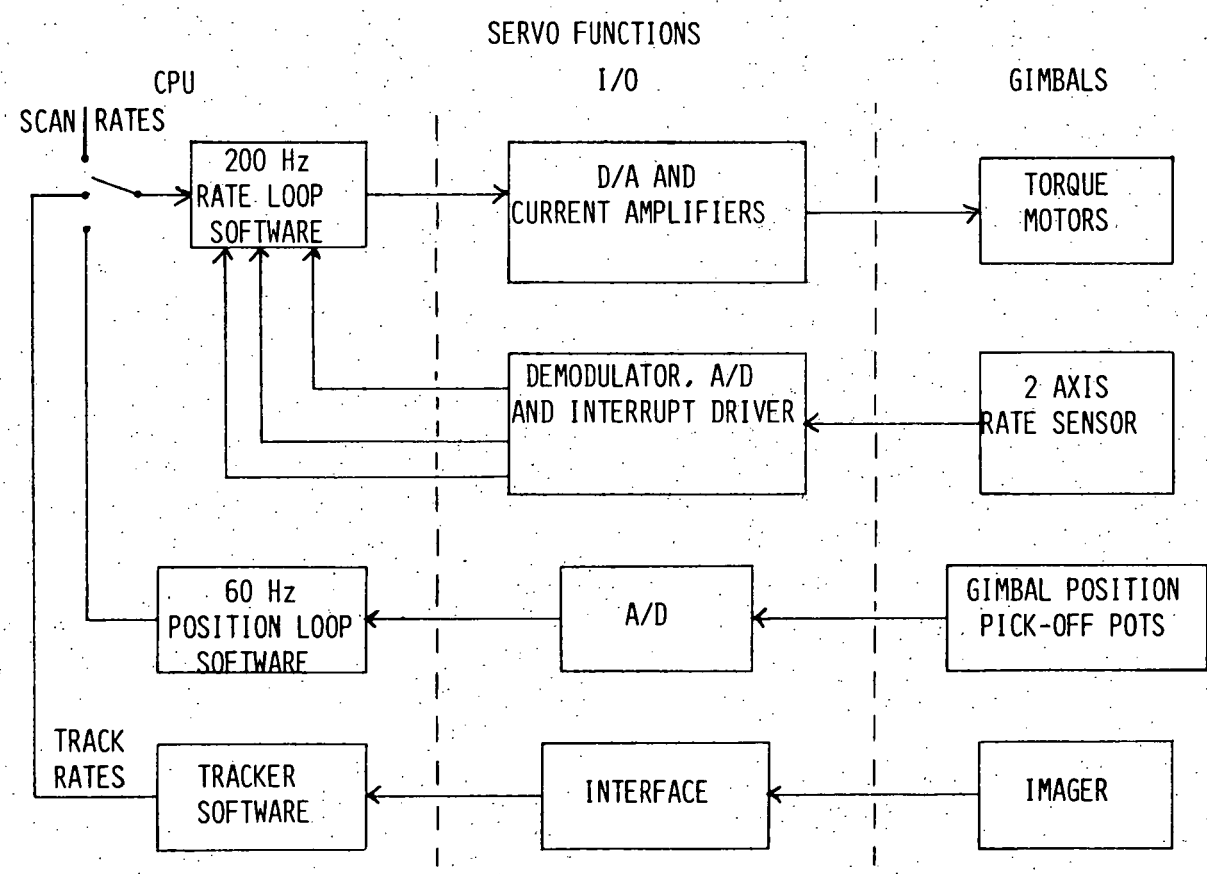
60 Hz MAIN LOOP
- END OF GATE
- END OF FIELD

200 Hz SERVO RATE LOOP
- SAMPLE AZIMUTH DEMODULATOR
- SAMPLE ELEVATION DEMODULATOR

50 Hz GUIDANCE COMPUTER TIMING LOOP
- 1553 INPUT DATA READY
- 1553 OUTPUT DATA READY

ANALOG DATA ACQUISITION

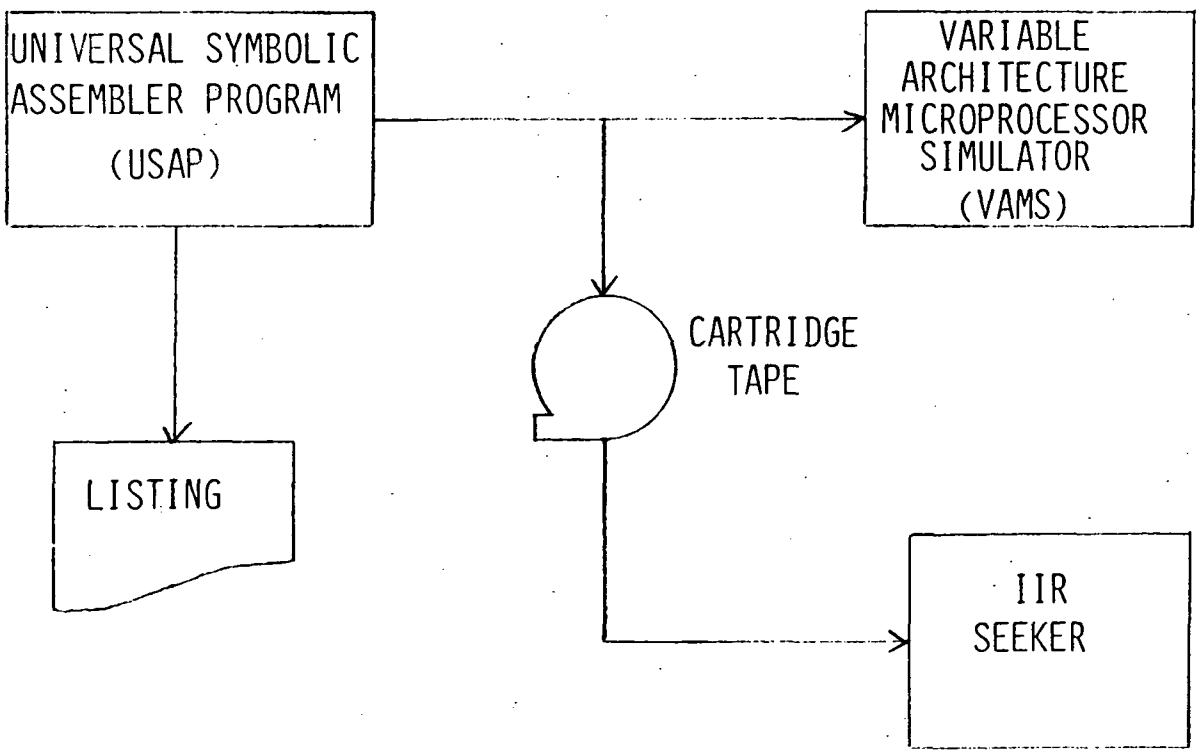




FRONT PANEL FUNCTIONS

- READ/LOAD DATA MEMORY
- READ/LOAD REGISTERS
  - CPU
  - I/O ADDRESS
  - MEMORY ADDRESS
  - CONDITION CODE
- READ/LOAD INTERRUPT STATUS
- EXIT/RETURN TO PROGRAM

SOFTWARE SUPPORT



EIGHT MICROPROCESSOR-BASED INSTRUMENT DATA SYSTEMS IN THE GALILEO ORBITER SPACECRAFT*

Robert C. Barry
Jet Propulsion Laboratory
Pasadena, California

The Galileo Orbiter spacecraft carries nine scientific instruments, all but one of which are controlled by individual microprocessors. Scientific investigations include interplanetary measurements of charged atomic particles, magnetic and electric fields, and dust. In orbit, Galileo will investigate Jupiter's magnetosphere and atmosphere, and surface properties of the four largest satellites. Launch is scheduled for early in 1984.

While the complexity of the instruments and their data systems varies widely, all utilize components from the RCA 1800 microprocessor family, and all perform the same basic functions. The decisions to utilize microprocessors in the instruments were heavily influenced by the spacecraft distributed Command and Data System (CDS) design which uses this same LSI family.

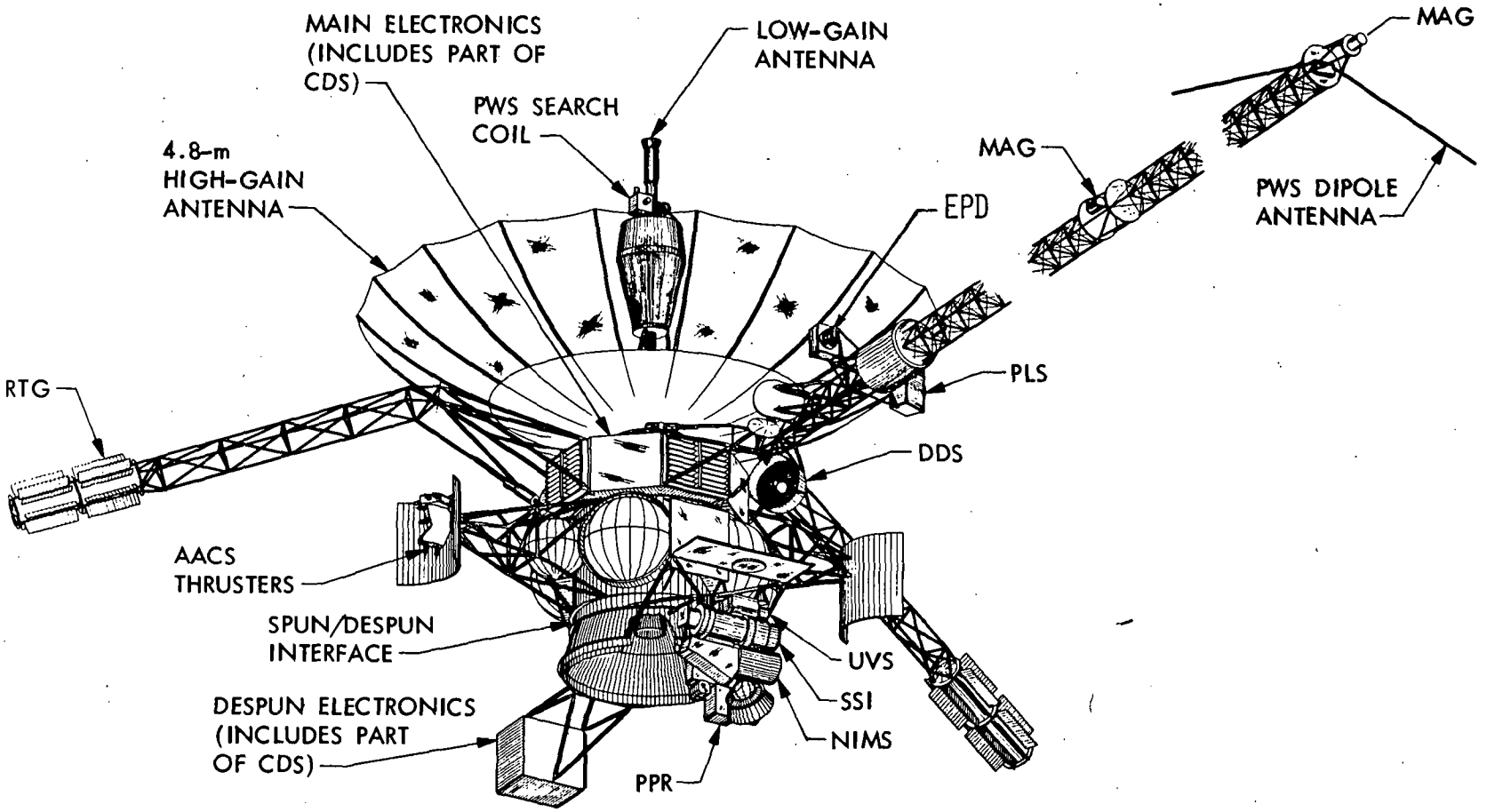
A typical instrument data system consists of a microprocessor, 3K Bytes of Read Only Memory (ROM) and 3K Bytes of Random Access Memory (RAM). It interfaces with the spacecraft data bus through an isolated user interface with a direct memory access bus adapter. Microprocessor control and data lines provide interrupts, serial, and/or parallel data from instrument devices such as registers, buffers, analog to digital converters, multiplexers, and solid state sensors. These data systems support the spacecraft hardware and software communication protocol, decode and process instrument commands, generate continuous instrument operating modes, control the instrument mechanisms, acquire, process, format, and output instrument science data.

The approach has resulted in many specific improvements over past missions. Some of the most important include: increased instrument autonomy, functional commanding, and macro mode generation; enhanced telemetry output from both operational and scientific points-of-view; and additional flexibility for in-flight optimization, problem work-arounds, and instrument generalization for support of multiple missions.

There was a significant but manageable underscoping of the microprocessor development tasks and costs. This was related to difficulty in establishing firm requirements at an early date, interfacing complexity, parts acquisition problems, and a general lack of extensive experience in microprocessor hardware and software design.

While the Galileo entry-level introduction into instrument microprocessor appears to be proceeding well, additional effort is needed for standard use of microprocessors in science instruments to achieve a significant part of its high potential benefit. This includes generation and clarification of spacecraft system level requirements in concert with the objectives of the end-to-end information system design, improved use of microprocessor development tools and practices, and justification for increased instrument funding compatible with the increase in capability and cost.

*This work was performed for the Jet Propulsion Laboratory, California Institute of Technology, sponsored by the National Aeronautics and Space Administration under Contract No. NAS7-100.


MAIN ELECTRONICS (INCLUDES PART OF CDS)
LOW-GAIN ANTENNA
MAG
PWS SEARCH COIL
4.8-m HIGH-GAIN ANTENNA
MAG
EPD
PWS DIPOLE ANTENNA
RTG
PLS
DDS
AACS THRUSTERS
SPUN/DESPUN INTERFACE
UVS
SSI
DESPUN ELECTRONICS (INCLUDES PART OF CDS)
NIMS
PPR

INTRODUCTION

- HISTORICAL ASPECTS OF INSTRUMENTS ON JPL SPACECRAFT
  - INCREASING COMPLEXITY
  - HIGHLY INTEGRATED SPACECRAFT
  - HIGHLY INTEGRATED MISSION OPERATIONS & FLIGHT TEAM
  - EXTENSIVE DATA PROCESSING & CORRELATION THE NORM

- WHAT ROLE DO INSTRUMENT MICROPROCESSORS PLAY IN THIS?
  - NOT A DRAMATIC CHANGE, A NATURAL EVOLUTION
  - VERY FLEXIBLE, EXPANDABLE CONCEPT
  - GALILEO IS THE STARTING POINT

- LIMITATIONS OF THIS PRESENTATION
  - RESTRICTED TO A HIGH-LEVEL, BRIEF REVIEW
  - TIME RESTRICTIONS FORCE GENERALIZATION
    - ACCENTUATE COMMON ELEMENTS
    - LITTLE DISCUSSION OF UNIQUE IMPLEMENTATIONS

MATERIAL TO BE COVERED

- REASONS FOR USE OF MICROPROCESSOR-BASED DATA SYSTEMS IN THE INSTRUMENTS
- FUNCTIONS PERFORMED BY THE μPs
- SUMMARY OF THE INSTRUMENTS
- SELECTED HIGHLIGHTS FROM THE GALILEO APPLICATIONS
- PROBLEM AREAS ENCOUNTERED
- EVALUATION OF THE GALILEO APPROACH

WHY USE MICROPROCESSORS IN THE GALILEO INSTRUMENTS?

- SUPPORT THE GENERALIZED SPACECRAFT SYSTEM INTERFACE (BUS)
- PROVIDE INCREASED INSTRUMENT AUTONOMY
  - REDUCE REQUIREMENT FOR SPACECRAFT SERVICES
  - INCREASE INSTRUMENT DESIGN CONTROL
- UTILIZE SEMICONDUCTOR INDUSTRY ADVANCES
  - AVAILABLE, PROVEN LSI PRODUCTS (RCA CDP1800 SERIES)
  - DECREASE POWER AND MASS REQUIREMENTS (CMOS LSI)
  - INCREASE RELIABILITY (REDUCE PARTS COUNT)
  - SIMPLIFY DESIGN (REPLACE DISCRETE, MSI LOGIC)
- EXTEND INSTRUMENT CAPABILITY (FALLOUT, NOT A REQUIREMENT)
  - ADD FLEXIBILITY TO ACCOMODATE CHANGING REQUIREMENTS (SOFTWARE)
  - PROVIDE ENHANCED MODE GENERATION AND CONTROL (MINIMAL H/W)
  - GENERALIZE INSTRUMENT
    - MODIFY OR UPGRADE FOR FUTURE USE ON OTHER SPACECRAFT

INSTRUMENT DATA SYSTEM FUNCTIONS

- SUPPORT SPACECRAFT INTERCOMMUNICATION BUS AND PROTOCOL
- DECODE AND PROCESS INSTRUMENT COMMANDS
- GENERATE INSTRUMENT OPERATING MODES
- CONTROL MECHANISMS
- PROCESS SCIENCE DATA FOR OUTPUT

DATA SYSTEM FUNCTIONS

- SUPPORT OF SPACECRAFT BUS AND PROTOCOL
  - SERIAL, SYNCHRONOUS DATA AT 403.2 KBPS
  - ISOLATED USER INTERFACES (TRANSFORMERS)
  - MEMORY TO MEMORY TRANSFER USING DIRECT MEMORY ACCESS
  - S/C COMMAND DATA SYSTEM (CDS) INITIATES AND CONTROLS ALL ACTIVITY ON A TIME MULTIPLEXED BASIS

- DECODING AND PROCESSING OF INSTRUMENT COMMANDS
  - ACCOMODATE VARIOUS INPUT DATA
    - COMMANDS
    - MEMORY LOAD
    - S/C TIME AND SPIN DATA
  - PROCESS COMMANDS
    - ERROR CHECKING & VALIDATION
    - UPDATE OF INSTRUMENT STATE DATA

DATA SYSTEM FUNCTIONS (CONT'D)

- GENERATION OF INSTRUMENT OPERATIONAL MODES
  - ALLOW FUNCTIONAL LEVEL COMMANDING (TYPICALLY 3 TO 8 MAJOR MODES)
  - REDUCE INTER-SUBSYSTEM COMMUNICATION
  - SYNCHRONIZE WITH SPACECRAFT TIMING
  - PROVIDE MORE MODE GENERATION FLEXIBILITY TO THE INSTRUMENT

- MECHANISMS CONTROL FUNCTIONS (SENSORS, FILTER WHEELS, MULTIPLEXERS, ETC.)
  - SENSE MECHANISM STATUS
  - GENERATE CONTROL SIGNALS
  - ACQUIRE AND BUFFER DATA

- SCIENCE DATA PROCESSING
  - APPLY ALGORITHMS TO PROCESS DATA (COMPRESSION, STATISTICS, DE-SPIN, ETC.)
  - FORMAT DATA (CONTROL SUBCOMMUTATION, ADD ENGR & STATUS, ETC.)
  - OUTPUT TO BUS UPON REQUEST

INSTRUMENT SUMMARY

ABBR.	NAME	PRINCIPAL INVESTIGATOR	INSTITUTION
SSI	SOLID STATE IMAGING	DR. J.S. BELTON, (TEAM LEADER)	KIT PEAK NATIONAL OBSERVATORY, TUSCON, AZ
NIMS	NEAR INFRARED MAPPING SPECTROMETER	DR. R. CARLTON, (TEAM LEADER)	JET PROPULSION LABORATORY, PASADENA, CA
PPR	PHOTOPOLARIMETER RADIOMETER	DR. J.E. HANSEN	GODDARD INSTITUTE FOR SPACE STUDIES, NEW YORK, NY
UVS	ULTRAVIOLET SPECTROMETER	DR. C.W. HORD	LABORATORY FOR ATMOSPHERIC AND SPACE PHYSICS, BOULDER, CO
EPD	ENERGETIC PARTICLE DETECTOR	DR. D.J. WILLIAMS	NOAA SPACE ENVIRONMENT LABORATORY, BOULDER, CO
PLS	PLASMA SUBSYSTEM	DR. L.A. FRANK	UNIVERSITY OF IOWA, IOWA CITY, IOWA
MAG	MAGNETOMETER	DR. M. KIVELSON	UCLA LOS ANGELES, CA
DDS	DUST DETECTOR SUBSYSTEM	DR. EBERHARD GRÜN	MAX PLANCK INSTITUT FÜR KERNPHYSIC, HEIDELBURG, WEST GERMANY
PWS	PLASMA WAVE SUBSYSTEM (NO μP)	DR. D.A. GURNETT	UNIVERSITY OF IOWA, IOWA CITY, IOWA

INSTRUMENT COMPLEXITY COMPARISON

INST	ROM KBYTE	RAM KBYTE	TELM DATA (KBPS)	DATA MODES	NO COMMAND PARMS.	NO MECHANISMS	NO SENSORS	MASS (KG)	PWR (WATT)
SSI	3	3.5	768. TO 0.02	3	24	9	800 x 800 CCD	28.0	25.
NIMS	3	1.75	11.52	6	31	9	17	18.1	16.
PPR	4	0.25	0.18	5	12	10	3	4.3	12.
UVS	--	0.75	1.0	3	20	6	3	4.2	4.5
EPD-1	4	2.66	0.92	15	150	50	17	8.5	8.6
EPD-2	2	0.25	--	--	50	3	--	(INCL ABOVE)	
PLS-1	4	4	0.6	7	140	31	20	10.7	9.5
PLS-2	(SAME AS PLS-1)								
DDS	3	2	.024	3	33	9	4	4.0	1.8
PWS (NO μP)	--	0.25	645. TO 0.2	2	7	11	3	5.3	5.6

INSTRUMENT DATA SYSTEM HIGHLIGHTS
(SELECTED FROM AMONG THE EIGHT INSTRUMENTS)

1) FUNCTIONAL COMMANDING AND MACRO MODE GENERATION

2) TASK ALLOCATION BETWEEN MICROPROCESSOR AND OTHER INSTRUMENT HARDWARE
   - HIGH RATE DATA TRANSMISSION
   - SPECIALIZED PROCESSORS (FORMATTERS, MULTIPLIER, ENCODER/COMPRESSOR, ETC.)
   - SOFTWARE OVERALL CONTROL
   - HYBRID INSTRUMENT OPERATION (WITH OR WITHOUT μP)

3) ENHANCED DATA ACQUISITION & TELEMETRY OUTPUT
   - SUBSYSTEM UNIFORMITY
   - ADDED ENGINEERING VISIBILITY (SPECIAL HIGH-RATE MODES, SUBCOMS, ETC.)
   - SOME ADDITIONAL ON-BOARD PROCESSING & BUFFERING (DATA DISCRIMINATION, DATA SEARCH, OPTIMUM AVERAGING, COMPRESSION, ETC.)

4) MEMORY RE-ALLOCATION AND REPROGRAMMABILITY FOR SCIENCE MODE OPTIMIZATION AND RESPONSE TO SUBSYSTEM OR SPACECRAFT FAILURES
   - RAM MARGIN
   - RAM REPLACEMENT OF ROM VIA BUS COMMAND
   - ROM LINKAGES TO RAM FOR SUBROUTINE REPLACEMENT

INSTRUMENT DATA SYSTEM HIGHLIGHTS (CONT'D)

5) HIGHLY FLEXIBLE APPROACH
   - ALLOWS SUBSYSTEM OPTIMIZATION
   - WIDE RANGE OF DATA RATES
   - 8 UNIQUE DESIGNS

6) CLEAR BENEFIT IN LOGIC REPLACEMENT OBSERVED

7) ADVANCED DATA SYSTEM TECHNIQUES
   - MULTIPLE MICROPROCESSORS
   - POWER HSARING, FAILURE ISOLATION
   - AUTO-CALIBRATION
   - BACKGROUND PROCESSING, HIGH LEVEL LANGUAGE (FORTH)

8) COMPATABLE WITH LONG-RANGE DEEP SPACE EXPLORATION DESIGNS AND GOALS
   - PACKET TELEMETRY
   - ADDED AUTONOMY
   - EEIS

PROBLEM AREAS ENCOUNTERED

1) HARDWARE DESIGN LIMITATIONS AND CONSTRAINTS
   - SPACECRAFT MASS, POWER, AND RELIABILITY REQUIREMENTS
     - LIMITED OPTIONS IN μP AND CHIP FAMILY SELECTION (1802, 1852, 1856, 1834, TC244)
     - LIMITED MEMORY SIZE (RAM IS 256 x 4 BITS)
   - JUPITER MISSION REQUIRED HIGH RADIATION TOLERANCE
     - PARTS DEVELOPMENT AND ACQUISITION PROBLEMS
     - CAUSED INCREASED COST, SCHEDULE PROBLEMS

2) USE OF READ ONLY MEMORY (ROM) AS PRIMARY PROGRAM MEMORY
   - INCREASED COSTS AND SCHEDULING PROBLEMS
   - DECREASED FLEXIBILITY
   - INCREASED NEED FOR EARLY SYSTEM-LEVEL VALIDATION

3) UNDERSCOPING OF MICROPROCESSOR DEVELOPMENT TASKS AND COSTS
   - SOFTWARE, SOFTWARE MANAGEMENT AND DOCUMENTATION
   - DEVELOPMENT SYSTEMS AND SUPPORT EQUIPMENT
   - INTERFACE REQUIREMENTS STABILITY
   - ADDED TESTING COMPLEXITY

PROBLEM AREAS ENCOUNTERED (CONT'D)

4) BUS DESIGN
   - BUS ADAPTER COMPLEXITY
   - LOW ERROR REQUIREMENT
   - OPEN-LOOP PROTOCOL (NO HANDSHAKE)

5) SYSTEM LEVEL REQUIREMENT IMMATURITY
   - END-TO-END INFORMATION SYSTEM (EEIS) GOALS REDUCED
   - EARLY INTERFACE REQUIREMENT STABILITY AND DETAILED DESCRIPTION

6) PERSONNEL EXPERIENCE AND TRAINING
   - μP
   - SOFTWARE

EVALUATION OF THE GALILEO APPROACH

- CURRENT STATUS
  - MOST INSTRUMENTS HAVE AN OPERATIONAL BREADBOARD DATA SYSTEM NOW
  - EXPECT ON-TIME DELIVERY OF ADVERTISED CAPABILITY
- EXPECT TO SIGNIFICANTLY ENHANCE THE SCIENCE VALUE OF THE MISSION
  - EACH INSTRUMENT PROVIDES SCIENCE OPTIMIZATION FOR ITS INVESTIGATORS
  - EACH HAS PROVIDED FOR INCREASED IN-FLIGHT FLEXIBILITY
- ADDITIONAL EFFORT MUST BE EXPENDED TO SOLVE PROBLEMS ASSOCIATED WITH:
  - SYSTEM REQUIREMENTS DEFINITION
  - IMPROVED USE OF MICROPROCESSOR DEVELOPMENT TOOLS
  - FUNDING CONSISTENT WITH THE ENHANCED CAPABILITY AND COST

FUTURE PROJECTIONS

- TECHNOLOGY IMPROVEMENT
  - SEMICONDUCTOR TECHNOLOGY (ADVANCED μP, DENSE MEMORY, HIGHER SPPED, ETC.)
  - ADVANCED ARCHITECTURE AND SOFTWARE DESIGN (MULTIPLE MICROPROCESSORS, HIGH LEVEL LANGUAGES, ETC.)
- APPLICATION ADVANCES
  - INSTRUMENT AUTONOMY
  - HIGHLY FUNCTIONAL COMMANDING
  - EXTENSIVE ON-BOARD PROCESSING (ESPECIALLY FRONT-END APPLICATIONS)

Page Intentionally Left Blank

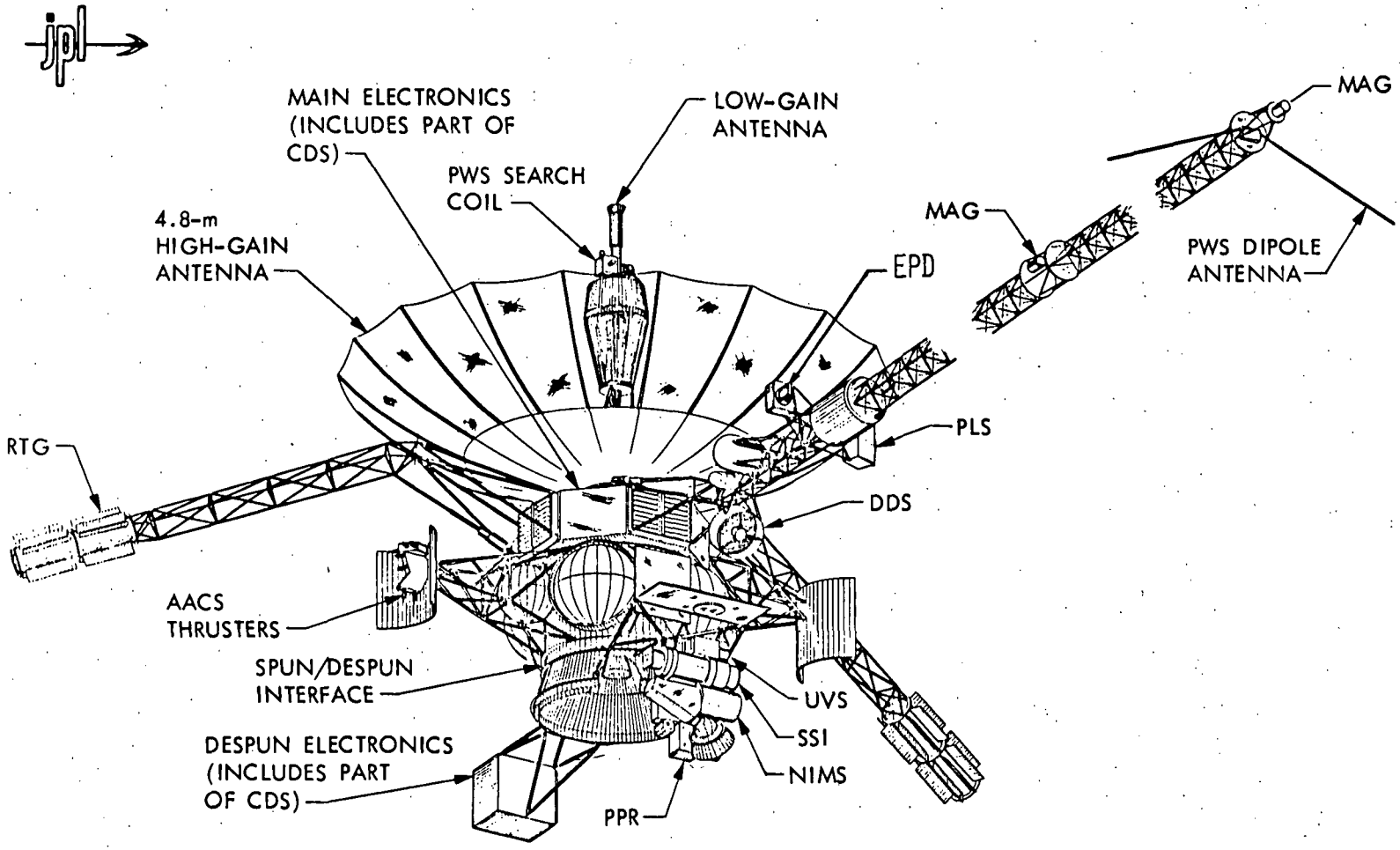
A COMMAND & DATA SUBSYSTEM FOR DEEP SPACE EXPLORATION BASED ON THE RCA 1802 MICROPROCESSOR IN A DISTRIBUTED CONFIGURATION

Jack S. Thomas
California Institute of Technology
Jet Propulsion Laboratory
Pasadena, California

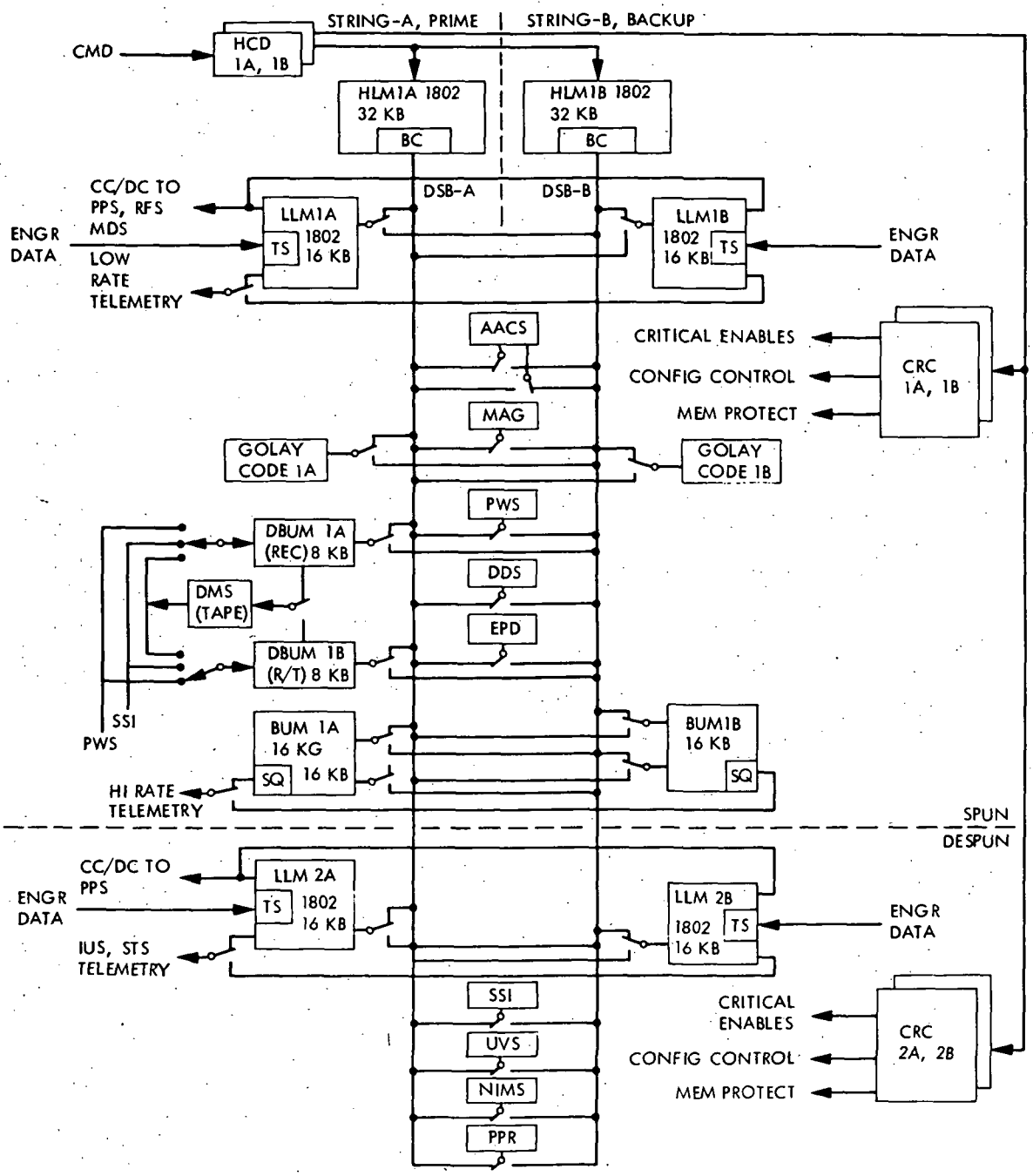
The Command and Data Subsystem (CDS) is an RCA 1802 CMOS microprocessor-based subsystem that acts as the central nervous system for the Galileo Orbiter Spacecraft. All Communication between the ground and spacecraft flows through the CDS. The CDS also distributes commands in real time, algorithmetically expanded from a data base loaded from the ground and in response to spacecraft alarms.

The distributed microprocessor system is configured as a redundant set of hardware with three microprocessors on each half. The microprocessors are surrounded by a group of special purpose hardware components which greatly enhance the ability of the software to perform its task.

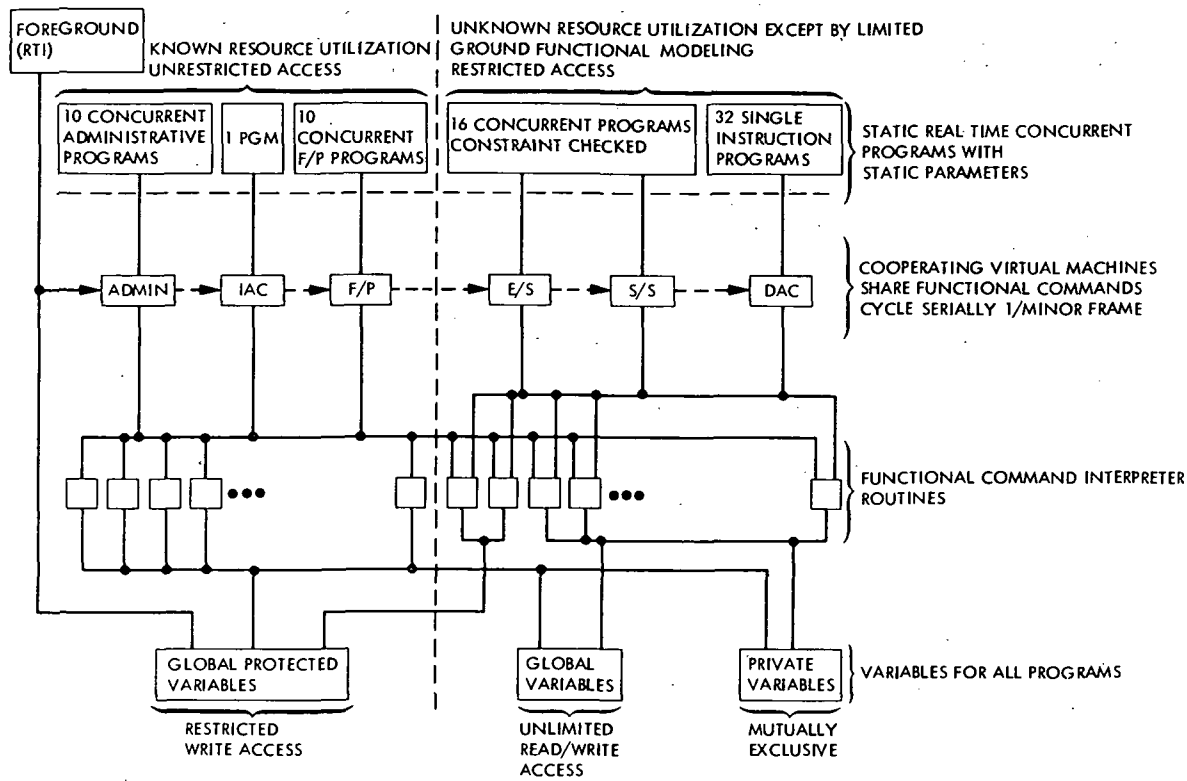
The presenter shows how the software architecture makes a distributed system of six microprocessors appear to each user as a single virtual machine, and collectively as a set of cooperating virtual machines that prevent the simultaneous presence of the several users from interfering destructively with each other.


jpl
MAIN ELECTRONICS (INCLUDES PART OF CDS)
LOW-GAIN ANTENNA
MAG
PWS SEARCH COIL
4.8-m HIGH-GAIN ANTENNA
MAG
PWS DIPOLE ANTENNA
EPD
PLS
RTG
DDS
AACS THRUSTERS
SPUN/DESPUN INTERFACE
UVS
SSI
DESPUN ELECTRONICS (INCLUDES PART OF CDS)
NIMS
PPR

GALILEO CDS FLIGHT HARDWARE CONFIGURATION

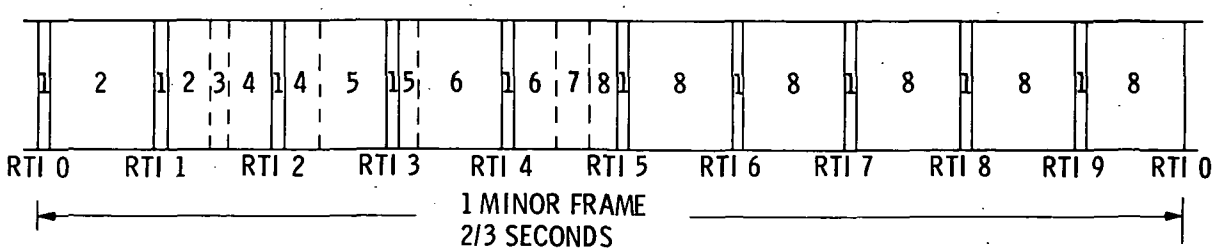


GALILEO CDS FLIGHT SOFTWARE ARCHITECTURE



HLM PROCESSOR TIME ALLOCATION

1. FOREGROUND EXECUTIVE
2. ADMINISTRATIVE PROGRAMS
3. IMMEDIATE ACTION COMMAND PROGRAM
4. FAULT PROTECTION PROGRAMS
5. ENGINEERING SEQUENCE PROGRAMS
6. SCIENCE SEQUENCE PROGRAMS
7. DELAYED ACTION COMMAND PROGRAMS
8. PROCESSOR TIME MARGIN



SYNERGISTIC INSTRUMENT DESIGN

Dale E. Winter
Jet Propulsion Laboratory
Pasadena, California

I. The Synergistic Approach

A. Do a functional design

1. Block-out all system functions.
2. Identify all areas that are exclusively analog.
3. Identify all areas that are exclusively digital.
4. Identify any analog/digital hybrid areas.

B. Design hardware to promote efficient software.

1. Supply task-efficient timing.
2. Supply task efficient I/O structures.
3. Design a task/code efficient system architecture.

C. Design software to promote efficient hardware.

1. Structure software to minimize hardware.
2. Customize coding to be task efficient.
3. Directly replace hardware functions wherever possible.
4. Use time and memory space wisely.

D. Completed design yields bonuses.

1. Additional features can be included with nominal hardware increases.
2. Design changes can be made easily.
3. Less hardware means less power, less mass and fewer failures.

II. The Galileo Television Camera

A. Taking pictures.

1. Filter selection and shuttering with software timed pulses directly to mechanism drive amplifiers.
2. CCD readout H1 rate timing executed in hardware.
3. CCD readout L0 rate timing and video/data system time syncronization executed in software.
4. Software timing is precision synchronized with system clock to assure exposure accuracy.

B. Telemetry acquisition.

1. Software controlled ADC and Mux.
2. Precision sample times.
3. Software can position sample times anywhere within the camera cycle to monitor specific activities.

C. Communications

1. Non-immediate bus adapter does most work in software.
2. Software sequencing sync's up with time broadcast.
3. Software rate buffers telemetry for transmission.

D. In-flight problem solving.

1. Programmable telemetry can profile electrical activity.
2. Multi-mode memory switching and mixing.
3. In-flight re-programming capability.
4. Diagnostic software reports and time tags errors.

SSI Timing

SSI image parameter control and timing signal generation is based on application of microcomputer technology. In addition to controlling serial pixel shifting and pixel analog-to-digital conversion, all timing, sequencing, mechanism control, engineering and status data acquisition, and buffering shall be performed under programmed microcomputer control. SSI data rates and formats shall be as specified in GLL-3-280, Telemetry Measurements and Data Formats. Additional SSI rates and timing intervals are presented in Table 1. Figures 2A, 2B and 2C present the relationship between the various SSI timing parameters for SSI imaging modes of 8 2/3, 30 1/3 and 60 2/3 seconds respectively.

TABLE 1. SSI TIMING PARAMETERS

	8 2/3 Second Mode	30 1/3 Second Mode	60 2/3 Second Mode
a. Pixel bit rate	806.4 KBPS	806.4 KBPS	806.4 KBPS
b. Pixel rate	100.8K Pixels/s	100.8K Pixels/s	100.8K Pixels/s
c. Line time	8 1/3 m sec	33 1/3 m sec	66 2/3 m sec
d. Read frame time	6 2/3 sec	26 2/3 sec	53 1/3 sec
e. Frame repetition time	8 2/3 sec	30 1/3 sec	60 2/3 sec
f. Prepare time	2.0 sec	3 2/3 sec	7 1/3 sec
g. Filter steps allowed	2	3	7
h. Maximum normal exposure	800 m sec	800 m sec	800 m sec
i. Maximum extended exposure	6400 m sec	25600 m sec	51200 m sec
j. SSI reply data rate	403.2 KBPS	403.2 KBPS	403.2 KBPS
k. CDS sync	806.4 KBPS	806.4 KBPS	806.4 KBPS
l. Real-time interrupt	15 Hz	15 Hz	15 Hz

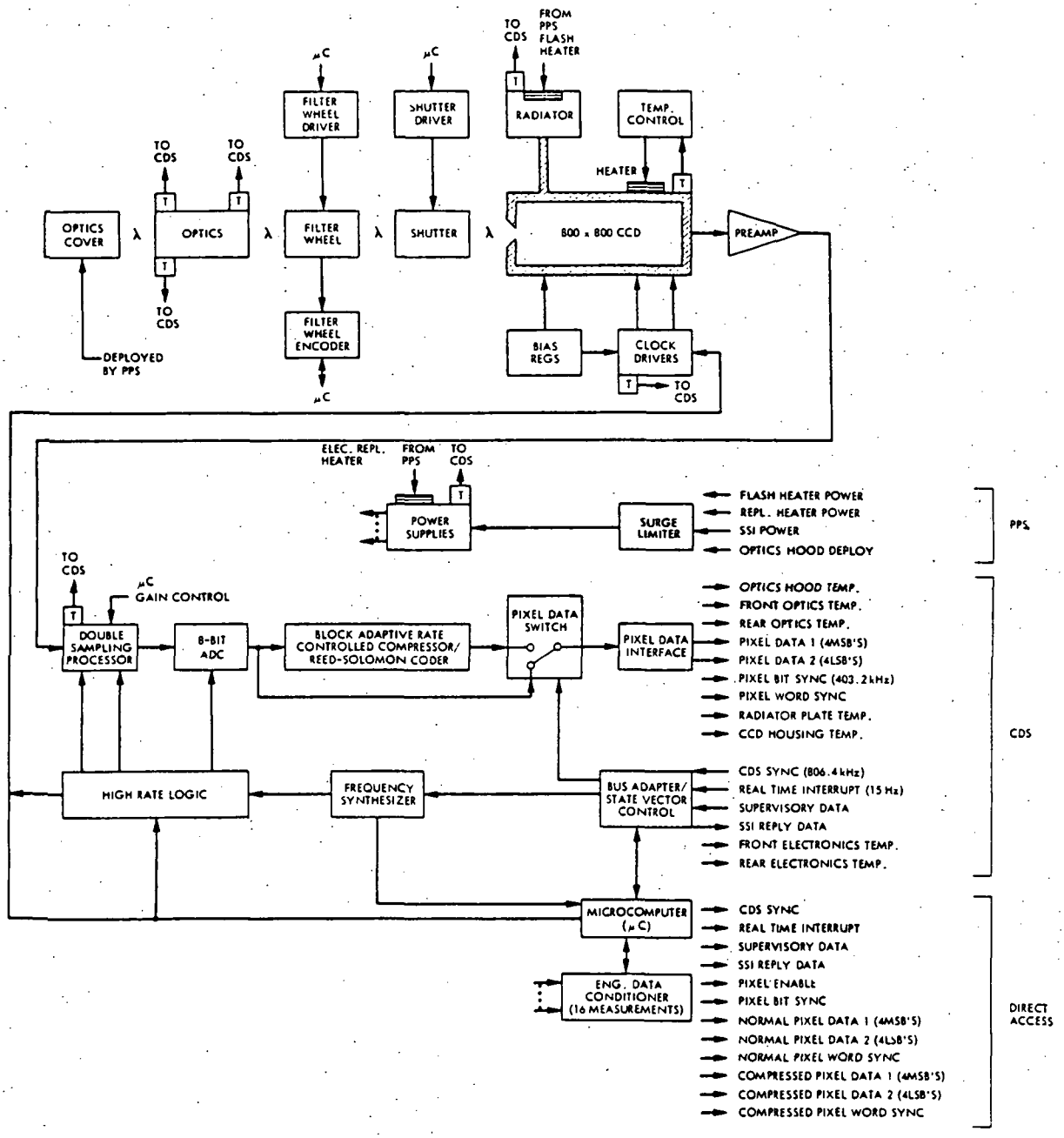


FIGURE 1. SSI FUNCTIONAL BLOCK DIAGRAM

Galileo

REQUIREMENTS

- COMMUNICATE VIA CDS BUS PROTOCOL
- MEET CAMERA FUNCTIONAL OBJECTIVES
- PREVENT HAZARDOUS CONDITIONS
- PROVIDE CAMERA HEALTH DATA
- PROVIDE FOR BACK-UP MODES
- PROVIDE FOR POST LAUNCH REPROGRAMMING
- PROVIDE DIAGNOSTIC TOOLS

Galileo

DESIGN CRITERIA

- FUNCTIONAL REQUIREMENTS
- CIRCUIT STIMULATION REQUIREMENTS
- COMMUNICATIONS REQUIREMENTS
- TIMING CONSIDERATIONS
- HARDWARE/SOFTWARE TRADEOFFS
- DIAGNOSTICS
- FAULT DETERMINATION
- REPROGRAMMING TECHNIQUES

Galileo

DESIGN APPROACH

- EFFICIENT PROGRAM ARCHITECTURE
  - OPTIMAL MEMORY USAGE AND EXECUTION TIMES
- ERRONEOUS COMMAND PROTECTION
  - PARITY ERRORS/ILLEGAL COMMANDS
- CONTINUOUS DIAGNOSTICS
  - CHECKSUMS/SCRATCH-PAD WRITE-READ
- FAULT DATA IN TELEMETRY
  - PARITY ERROR, COMMAND TRAFFIC, ILLEGAL COMMAND COUNTS
  - DIAGNOSTIC RESULTS/FAULT TIME TAG
- SPECIAL FAULT ANALYSIS TOOLS
  - PROGRAMMABLE ENGINEERING READOUTS
  - PROGRAMMABLE MEMORY MONITOR
- BACK-UP MEMORY CONFIGURATIONS
  - EXECUTE CODE FROM RAM, ROM + RAM, ROM/RAM + SCRATCH-PAD

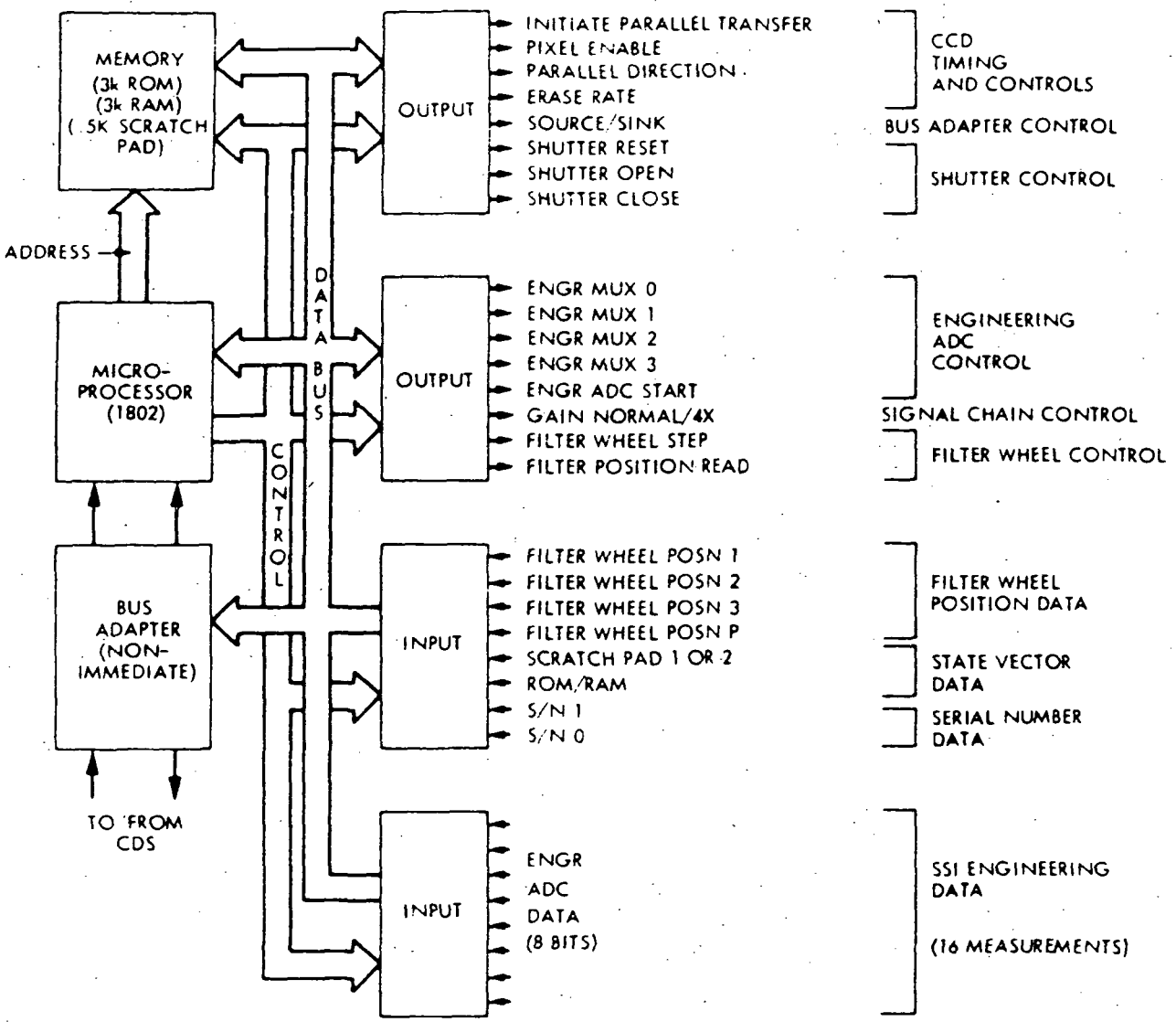
    USE SPARE SCRATCH-PAD FOR CODE OR DATA

Galileo

DATA SYSTEM ARCHITECTURE

- OUTPUT PORTS SUPPLY LOW AND MEDIUM RATE PULSES AND SIGNALS TO CAMERA ELECTRONICS
- SOFTWARE DISPATCHES AND TIMES OUTPUTS IN ACCORDANCE WITH COMMAND, FUNCTIONAL AND ELECTRICAL REQUIREMENTS
- INPUT PORTS SUPPLY ENGINEERING, FILTER POSITION AND STATUS DATA TO THE SOFTWARE
- FLAGS SUPPLY RTI, PROGRAM LINK MODE, SRTI PHASE AND CDS BUS PARITY ERROR DATA TO THE SOFTWARE
- SOME OUTPUTS ARE RE-CLOCKED WITH SRTI OR SRTI PHASE TO ASSURE SYSTEM SYNCHRONISM

Galileo
SSI MICROCOMPUTER



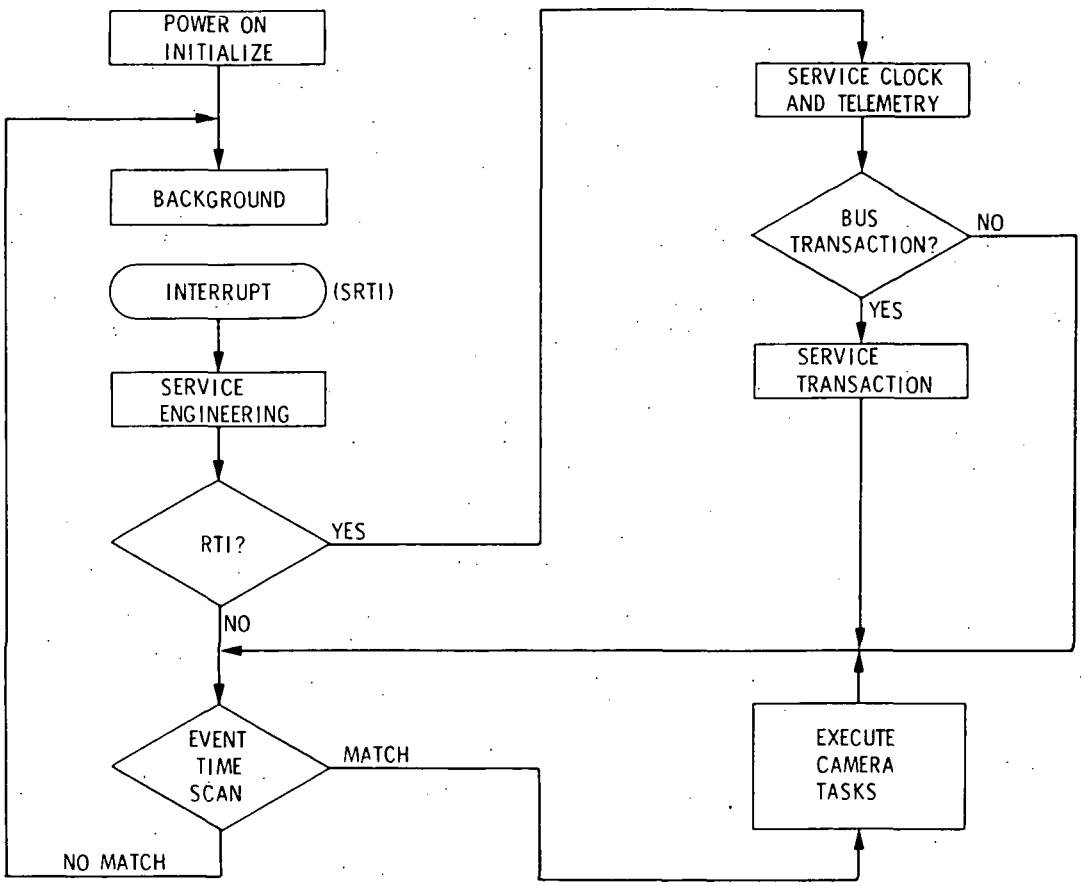
Galileo

SOFTWARE STRUCTURE

- REAL TIME INTERRUPT DRIVEN
- FOREGROUND/BACKGROUND OPERATION
- INTERNAL SPACECRAFT TIME CLOCK
- TIME DISPATCHED EVENTS
- SYNCHRONOUS OUTPUTS

Galileo

BASIC PROGRAM FLOW



Page Intentionally Left Blank

APPLICATION OF MICROPROCESSORS TO INTERPLANETARY SPACECRAFT DATA SYSTEMS

Samuel G. Deese
Jet Propulsion Laboratory
Pasadena, California

The Jet Propulsion Laboratory has committed to the use of a microprocessor based distributed data system in the 1984 Galileo mission to Jupiter. There has been an evolution of this commitment following the advances in component and device technology. Early spacecraft were very simple with subsystems very much single function oriented. Our understanding was very high and the need for design and analysis tools very low.

As technology grew, so did the complexities of the systems. Step by step, subsystems and functions were combined thus increasing their capability as well as complexity. Missions became more ambitious and the returns were high. Costly design and analysis tools were developed to support system test and operations. With these tools we were able to some small degree analyze and/or predict the performance of the spacecraft.

The Galileo Command and Data Subsystem (CDS) evolved from the combination of two special purpose computers from previous spacecraft: the Flight Data Subsystem and the Computer Command Subsystem. The CDS architecture utilizing concepts investigated in the development of the Unified Data Subsystem (UDS) takes advantage of the microprocessor technology and serves as the core of the distributed microprocessors interconnected by a high speed data bus.

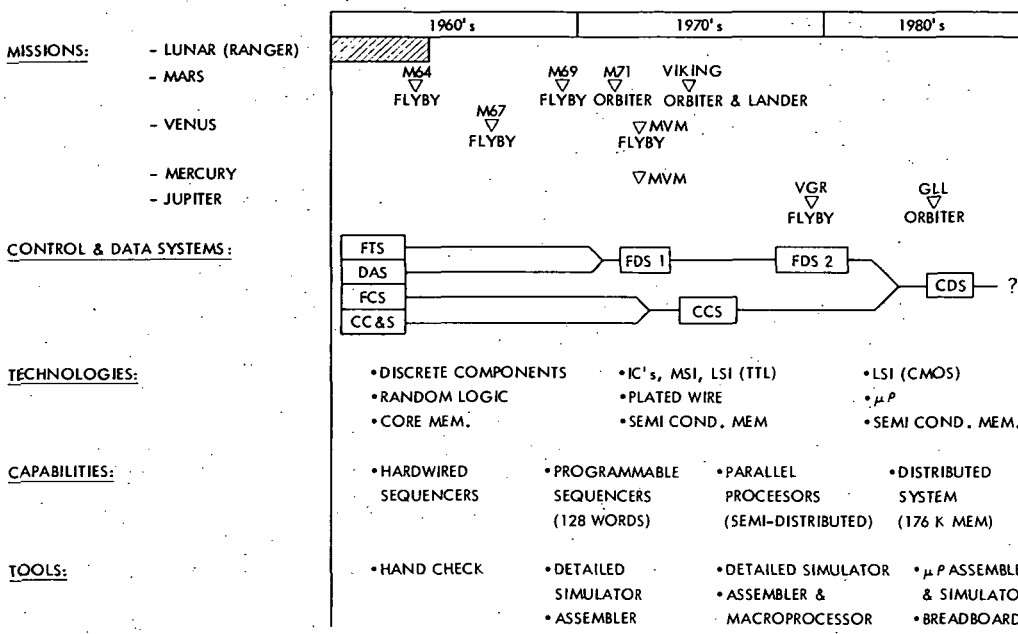
The CDS design is complete and breadboard integration and test are in process. The flight software is in the requirements and "prototype" design phase. Many obstacles have been encountered and overcome. Some worthy of mention are:

1) Choice of a microprocessor architecture based primarily on its power and radiation hardness qualifications.
2) High speed operation of CMOS logic.
3) Adaptation of a Higher order Language to a microprocessor and in particular to a processor with an architecture not well suited for the CDS application.
4) The difficulties in obtaining quantities of qualified parts that are very complex and have difficult requirements, i.e. radiation hardening.
5) Availability of design and analysis tools for understanding and validating distributed systems with concurrent processing.

Ongoing advanced development and preproject studies are primarily based on data system designs having the same requirements as the CDS. We are committed in the future to the continued application of microprocessors to distributed data systems; solutions to the above problems; and to continue to follow advances in technology with the incorporation of VLSI into modular fault tolerant building blocks.

In summary, technology and complexity have very rapidly advanced since the first Ranger spacecraft in the 1960s. The design and analysis tools have sadly lagged this progress leaving our ability to "best" design and understand what we have designed less than optimum. Along with our use of the new technologies of the future, we must also attack this deficiency.

THE CHRONOLOGICAL PATH TO MICRO-PROCESSOR APPLICATION



THE BASELINE - UDS

THE UDS, A DEVELOPMENT SPONSORED BY THE NASA UNDER CONTRACT NAS7-100 WITH THE CALIFORNIA INSTITUTE OF TECHNOLOGY AT JPL.

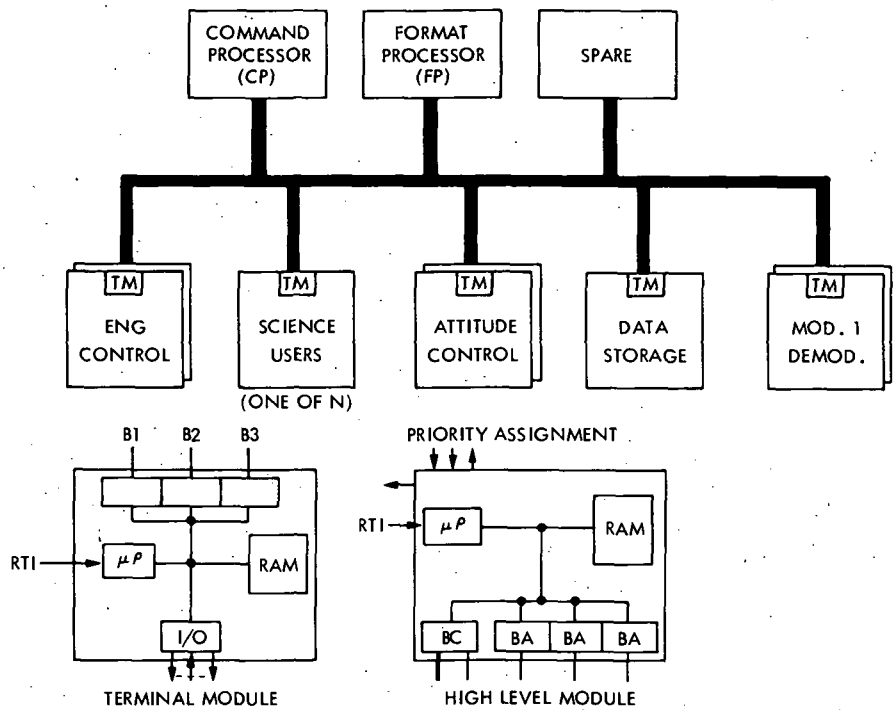
SALIENT FEATURES:

- REAL TIME CONTROL - PRECISE TIMING
- DISTRIBUTED ARCHITECTURE
  - + DISTRIBUTED FUNCTIONS (HI-LEVEL CONTROL, LOW-LEVEL EXECUTION)
  - + INTERACTION MINIMIZED
  - + HIERARCHICAL CONTROL
  - + COMPUTER INDEPENDENCE
- STANDARDIZED SOFTWARE AND SUPPORT EQUIPMENT
- STANDARD INTERFACES

FEASIBILITY DEMONSTRATED VIA:

- BREADBOARD OF BASIC SYSTEM UTILIZING NAKED MINI'S
- ONE REMOTE TERMINAL UNIT (RTU) INCLUDING 8080 MICROPROCESSOR
- DEVELOPED CONCEPT OF UDS DESIGN LANGUAGE (UDL)
- DESIGNED AND IMPLEMENTED "TYPICAL" APPLICATION SOFTWARE
- EXPLORED CONCEPTS OF DEBUGGING A DISTRIBUTED SYSTEM

THE UDS DESIGN
(MARINER CLASS S/C)



THE RTU

INITIALLY DEVELOPED AS A PART OF THE UDS.

- UDS I/O TYPE INTERFACES
- UDS BUS INTERFACE
- MICROPROCESSOR DRIVEN

PROPOSED FOR DEVELOPMENT AS A NASA STANDARD FOR POSSIBLE USE WITH:

- MMS (DHCS - NSSC-1)
- GALILEO
- VARIOUS PREPROJECT STUDIES

CONCEPT DIED

- BURDENED WITH UNIVERSAL I/O
- SALE OF THE CONCEPT - NO VOLUNTARY FIRST USER
- FEW SUPPORT FACILITIES

MAIN INHERITANCE FROM THIS EFFORT - RCA 1802 MICROPROCESSOR, IMPLEMENTATION CONCEPTS OF BA AND BC.

GALILEO

A COMMITMENT TO A DISTRIBUTED DATA SYSTEM UTILIZING THE MICROPROCESSOR TECHNOLOGY IN A FLIGHT DEVELOPMENT ENVIRONMENT

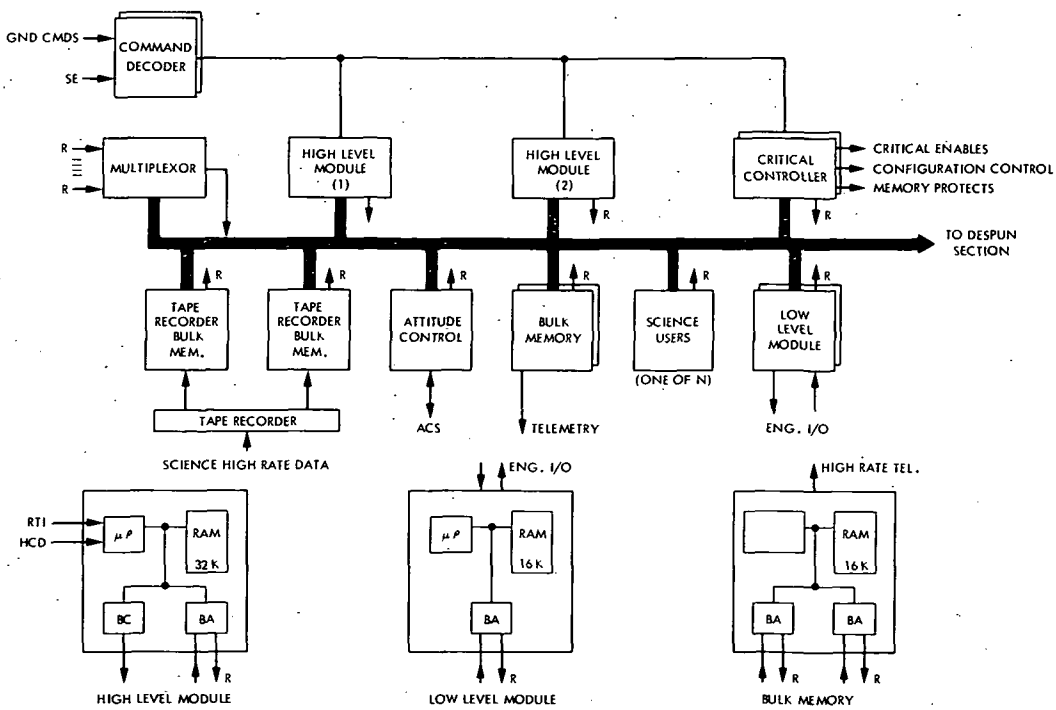
DRIVERS ON DEVELOPMENT

- UDS AS A BASELINE
- LOW POWER AND RADIATION DICTATE ACCEPTANCE OF THE RCA 1802
- COMBINING FDS AND CCS INTO SINGLE SUBSYSTEM
- CHEAPER OPERATIONS
- VOYAGER AS A BASELINE
- CMOS (4000 SERIES) SUPPORT LOGIC
- MUST USE HOL (SPECIFICALLY, HAL-S)
- NOTION THAT THE MORE YOU DO ON BOARD - THE CHEAPER ON GROUND

STATUS

- DESIGN COMPLETE (HARDWARE)
- ARCHITECTURAL DESIGN OF FLIGHT SOFTWARE IN PROGRESS. SOME PROTOTYPE/ BREADBOARD DESIGNS. (J. THOMAS PRESENTATION)
- BREADBOARD AND SUPPORT EQUIPMENT INTEGRATED DESIGN VERIFICATION IN PROCESS
- HAL-S HAS BEEN REMOVED AS A REQUIREMENT.
- DESIGNS FOR SOFTWARE AND OPERATIONS SUPPORT TOOLS IN PROGRESS.

GALILEO DESIGN



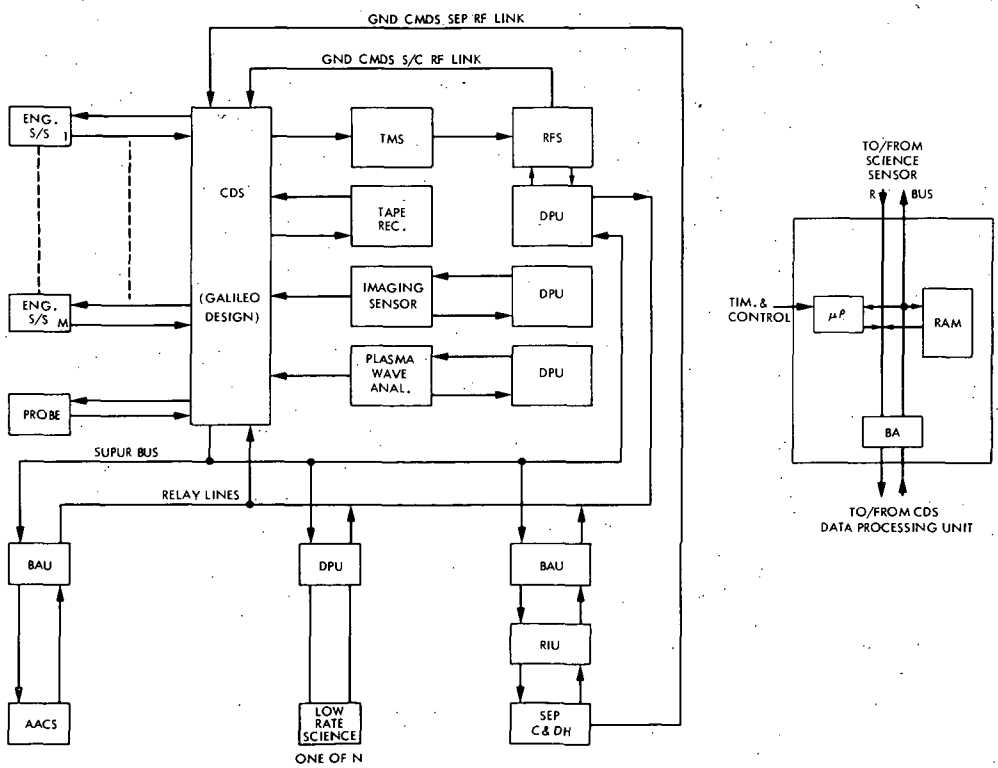
OTHER PROPOSED APPLICATIONS

COMET RENDEZVOUS

- MANY IMPLEMENTATIONS PROPOSED DEPENDING ON CHARACTER OF MISSION AT ANY GIVEN TIME AND ECONOMIC ENVIRONMENT.
  + BASED ON "CORE" DISTRIBUTED DATA SYSTEM
  + USERS NEED COMPUTING POWER
  + COMPUTING POWER DERIVED AS A STANDARD FOR THE S/C AND WOULD SIMPLY BE INCORPORATED INTO THE USER DESIGN (RTU SANS THE I/O).

OTHER PROPOSALS HAVE BEEN MADE, PRIMARILY BASED ON THE DISTRIBUTED SCHEME.

COMET RENDEZVOUS PROPOSED DESIGN



THE CHRONOLOGICAL PATH OF PROBLEMS

EARLY DESIGNS (M69, M71, M73)

- SIMPLE DESIGNS EASILY UNDERSTOOD
- COMPONENT COMPLEXITY LESS; EASILY TESTED AND SCREENED
  + PROCESS PROBLEMS (PURPLE PLAGUE, CORROSION, ETC.)
- HARDWARE AND SOFTWARE DESIGN AIDS AVAILABLE EARLY
  + DETAILED SIMULATOR
    - MEMORY SIZING
    - TIMING
    - TEST SOFTWARE DEVELOPMENT AND VALIDATION
  + ASSEMBLER/LOADER
- SIMPLER SOFTWARE (128/500 words)

RECENT DESIGNS (VIKING, VOYAGER)

- DESIGNS MORE COMPLEX
- INCREASED COMPONENT COMPLEXITY
- HARDWARE AND SOFTWARE DESIGN AIDS AVAILABLE EARLY
  + DETAILED SIMULATOR
  + ASSEMBLER/MACRO PROCESSOR

THE CHRONOLOGICAL PATH OF PROBLEMS (CONT.)

RECENT DESIGNS (VIKING, VOYAGER) CONT.

- INCREASED COMPLEXITY IN SEQUENCING
- ON-BOARD FAULT MANAGEMENT (LIMITED)
- MORE COMPLEX SOFTWARE
- DECREASING R&AD FUNDS

GALILEO AND FORWARD

- LACK OF EARLY DEVELOPMENT TOOLS
- VERY COMPLEX COMPONENTS
- INCREASED COMPLEXITY IN SEQUENCING, ON-BOARD FAULT MANAGEMENT
- MORE SEVERE ENVIRONMENTS
- LACK OF CONTINUITY OF MISSIONS
- FURTHER DECREASE IN R&AD FUNDS
- MUST BE CHEAP

SPECIFIC PROBLEMS

- DIFFICULTY IN APPLYING HOL (HAL-S) TO RCA 1802
- RCA 1802 ARCHITECTURE PROBABLY NOT BEST SUITED FOR CDS TASK.

THE CHRONOLOGICAL PATH OF PROBLEMS (CONT.)

SPECIFIC PROBLEMS (CONT.)

- RADIATION HARDENING PROBLEMS
- ARCHITECTURAL EVALUATION
  - MEMORY SIZING
  - BUS TRAFFIC
  - TIMING MARGINS
- DEVELOPMENT AND TEST OF TEST SOFTWARE
- COMPETITION IN LABOR MARKET

Page Intentionally Left Blank

THE ROLE OF THE MICROPROCESSOR IN ONBOARD IMAGE PROCESSING FOR THE INFORMATION ADAPTIVE SYSTEM

W. Lane Kelly, IV, and Barry D. Meredith
Langley Research Center
Hampton, Virginia

The Information Adaptive System (IAS) is an element of the NASA End-to-End Data System program and is focused toward high speed onboard data processing for NASA missions in the 1980's. Particular emphasis is placed on multispectral-image data processing since the speed and quantity of that variety of data places the greatest burden on the current NASA data system. Some of the image processing functions planned for the IAS include sensor nonuniformity correction, geometric correction, data editing, formatting, packetization and adaptive system control.

The design of the IAS is intended to apply to a variety of future missions; therefore, architectural flexibility is a key design feature. The programmability of the microprocessor lends this required flexibility to the system, allowing it to accommdate new processing functions and interface with a variety of sensor configurations. The high throughput rate required for multispectral image data processing prohibits the use of conventional computer software approaches without significant increases in the speed of the central processing unit. Hence, a combination of high speed special purpose hardware and microprocessors for control and computational support, appears to offer the best technical approach for the near term. In addition, a sophisticated microprocessor will serve as the overall system supervisor interfacing with commands from the spacecraft and the ground.

This paper presents the preliminary design of the Information Adaptive System and discusses the role of the microprocessor in the implementation of the individual processing elements.

THE CURRENT NASA DATA SYSTEM PROBLEM

- EVER INCREASING DEMAND MET WITH PROBLEM BY PROBLEM SOLUTIONS.
- CURRENT DATA LOAD - 10^{11} bits/day.
- DATA PROCESSING DELAYS ARE EXCESSIVE.
- DATA PROCESSING COSTS ARE TOO HIGH.
- FORTHCOMING PROJECTS WILL INCREASE DATA LOAD BY AN ORDER OF MAGNITUDE.
- SHUTTLE CAPABILITY WILL BOOST LAUNCH RATE BY FACTOR OF 6.

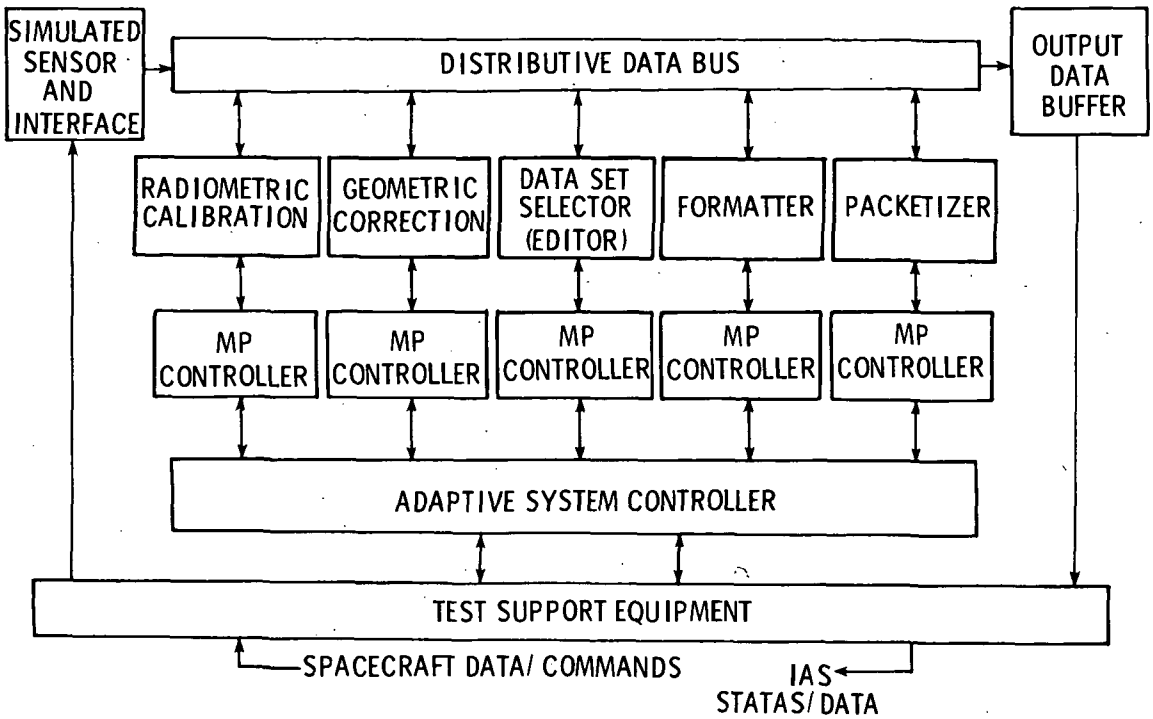
NEEDS II-INFORMATION ADAPTIVE SYSTEM

GOAL: DESIGN, DEVELOP AND DEMONSTRATE IN EARLY 1983 A SYSTEM ARCHITECTURE THAT UTILIZES ADVANCED TECHNOLOGY FOR HIGH-SPEED MULTISPECTRAL IMAGE DATA PROCESSING.

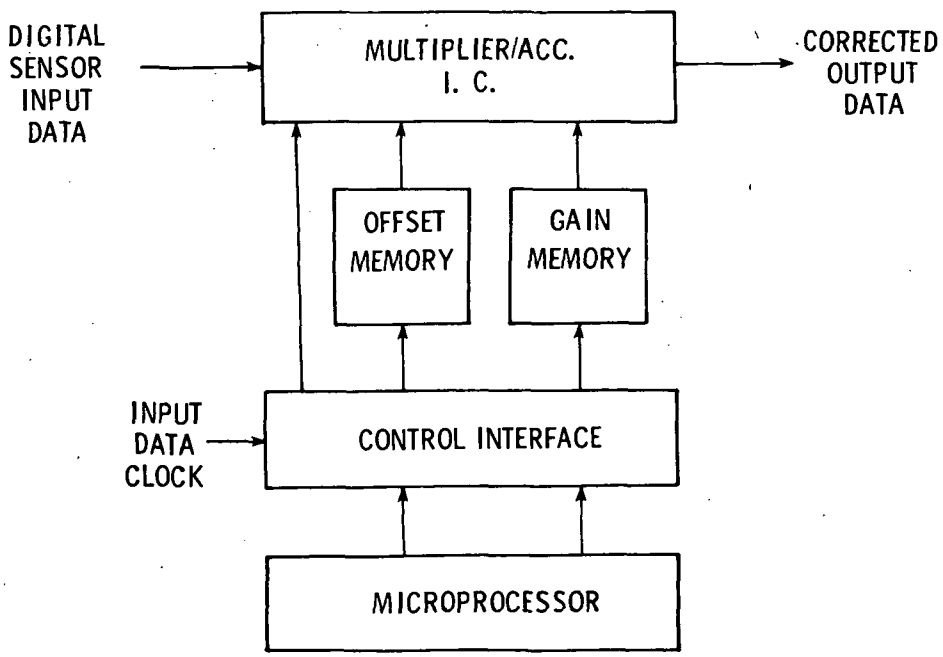
DESIGN FEATURES:

- HIGH DATA THROUGHPUT RATE
- PROGRAMMABILITY
- FLEXIBLE ARCHITECTURE
- ADAPTABILITY

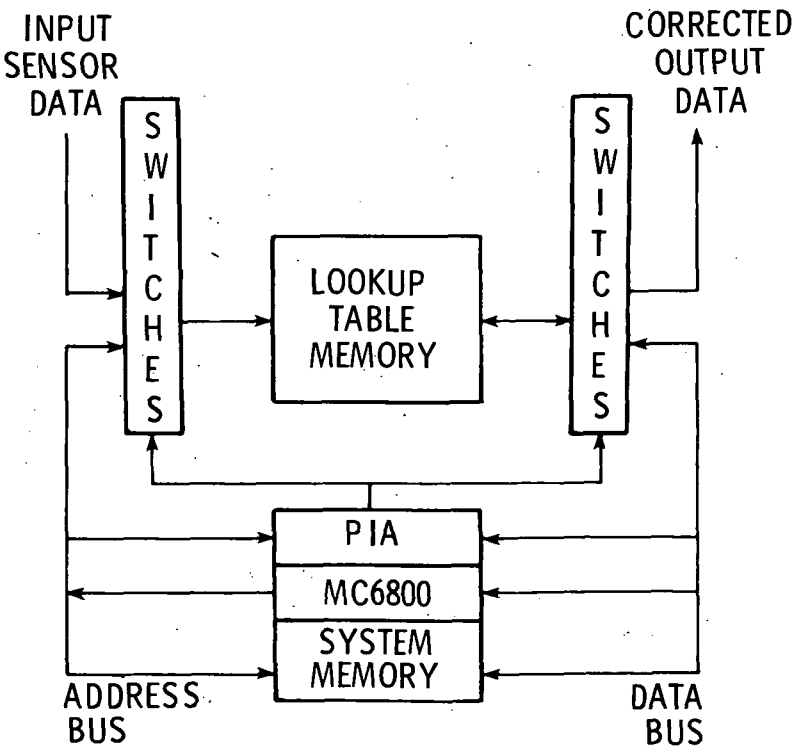
IAS DEMONSTRATION SYSTEM BLOCK DIAGRAM



RADIOMETRIC CORRECTION - LINEAR CURVE FIT APPROACH



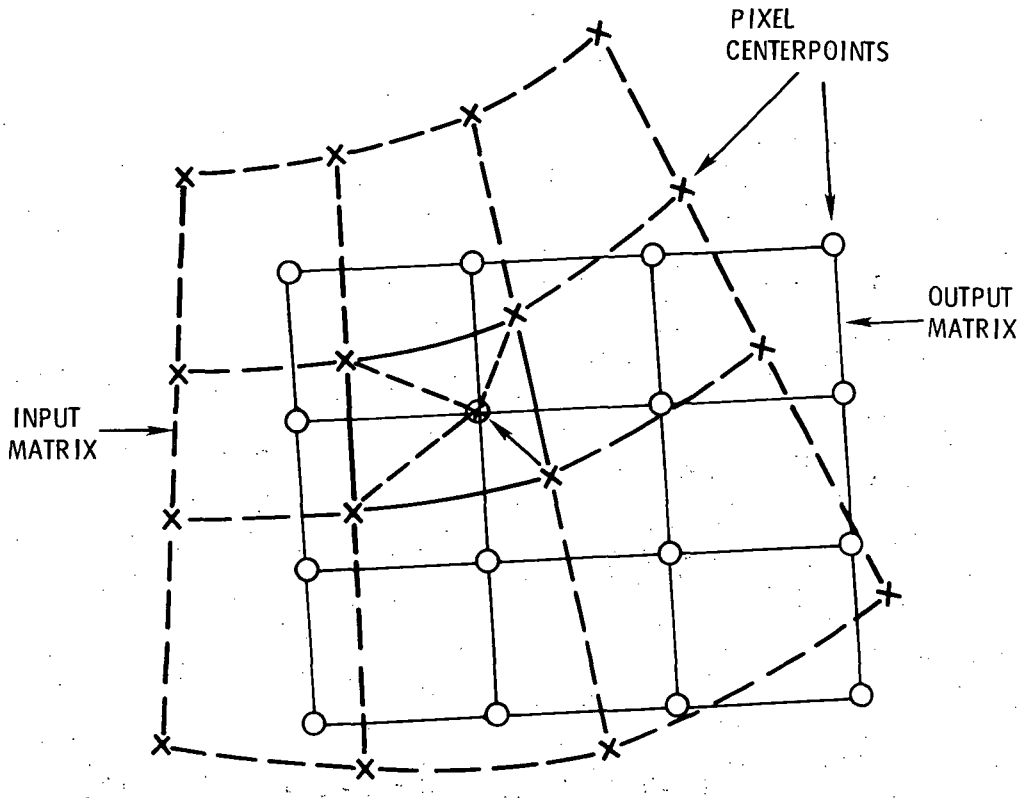
LOOKUP TABLE DESIGN FOR RADIOMETRIC CALIBRATION



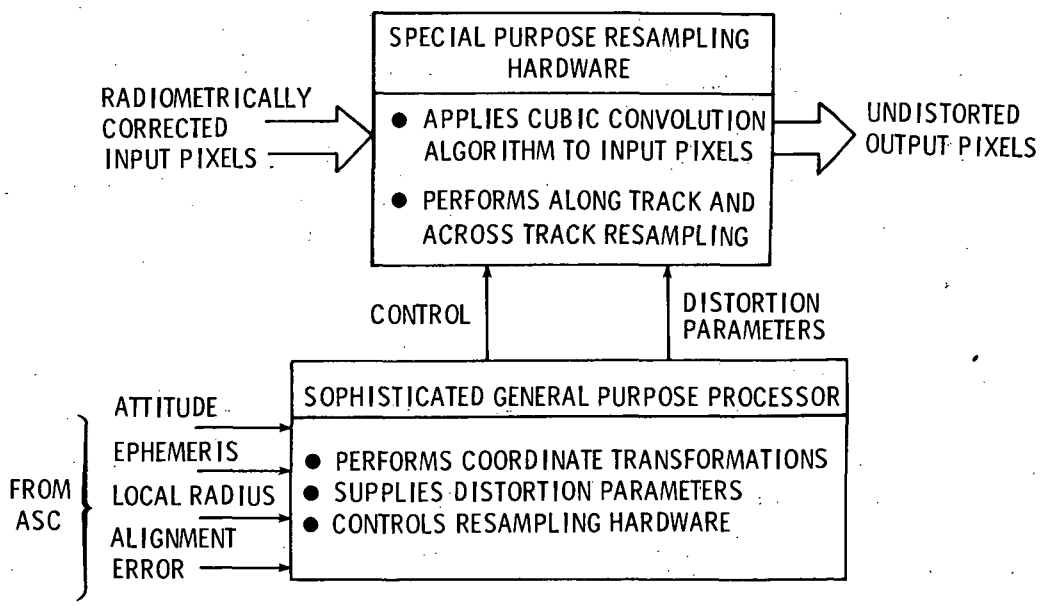
SOURCES OF DISTORTION IN IMAGE DATA AND THEIR CORRESPONDING ERROR MEASUREMENT TECHNIQUES

EPHEMERIS VARIATIONS	→ GLOBAL POSITIONING SYSTEM (GPS)
SPACECRAFT ATTITUDE VARIATIONS	→ ADVANCED STAR TRACKER
SENSOR MISALIGNMENT	→ PERIODIC GROUND CALIBRATION
EARTH CURVATURE	→ GPS WITH LOCAL EARTH RADIUS INFORMATION

GEOMETRIC CORRECTION



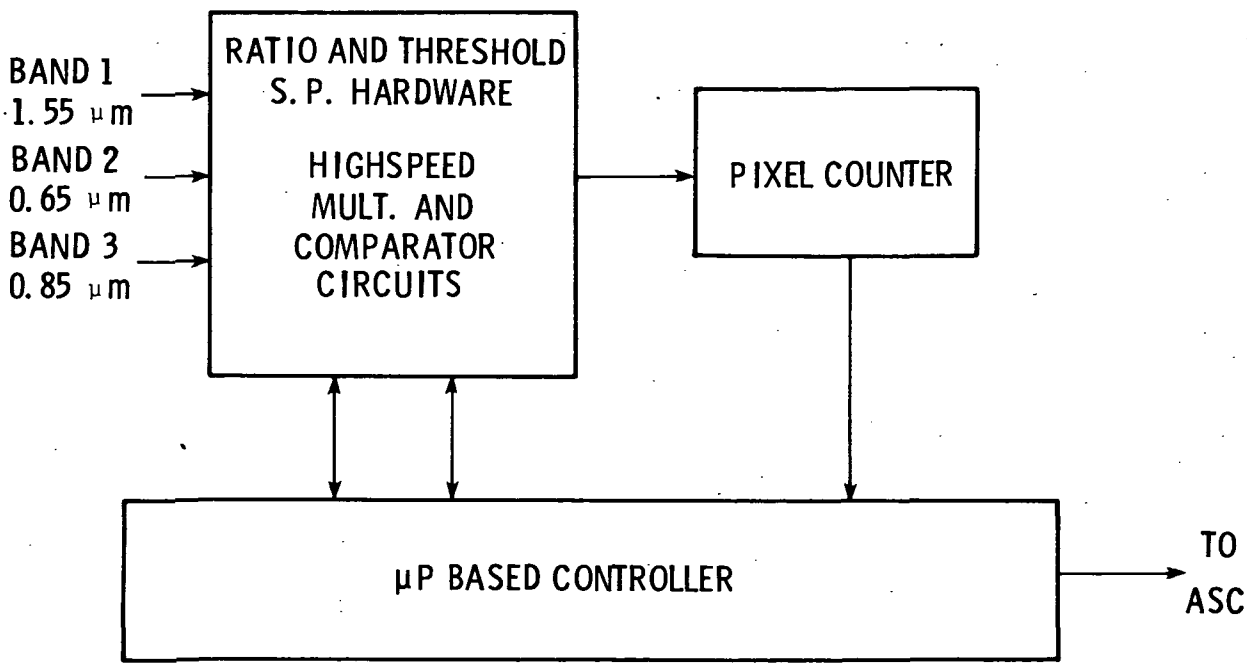
GEOMETRIC CORRECTION PROCESSING



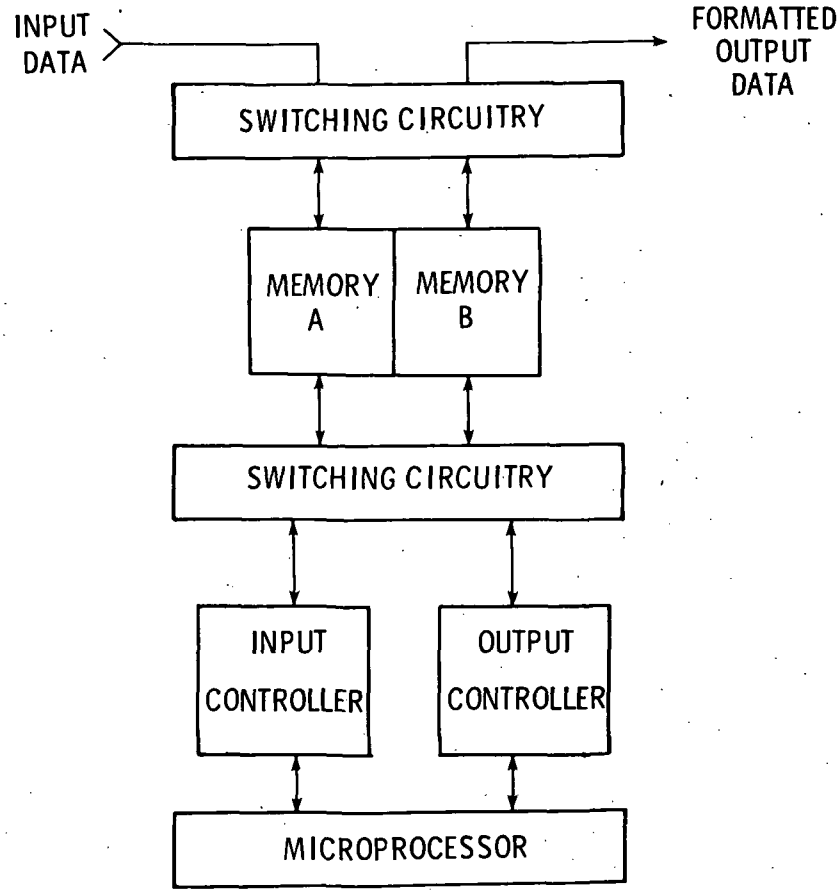
EDITING CRITERIA UNDER INVESTIGATION FOR THE IAS

- TIME
- SPACECRAFT POSITION
- INFORMATION FROM OTHER EXPERIMENTS
- CLOUD COVER

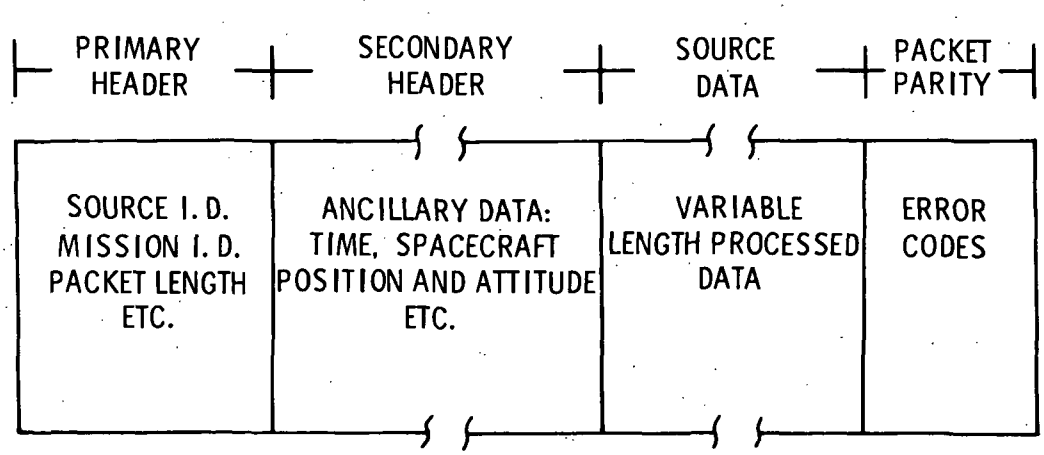
IMPLEMENTATION OF CLOUD DETECTION ALGORITHM



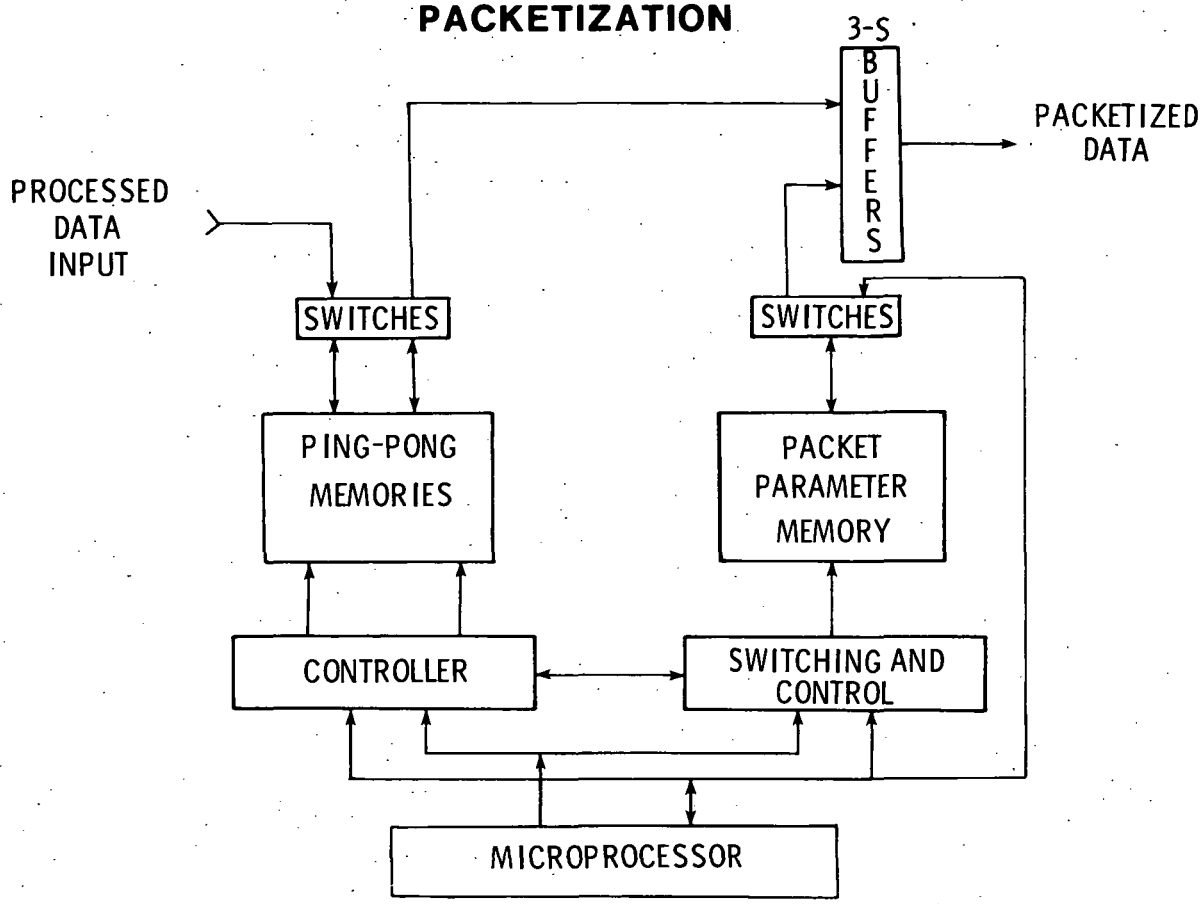
BUFFER MEMORY SYSTEM FOR DATA FORMATTING



NASA DATA PACKET FORMAT



HIGH SPEED PING-PONG MEMORY APPROACH TO DATA PACKETIZATION



ADAPTIVE SYSTEM CONTROLLER TASKS

- INITIALIZE INDIVIDUAL CONTROLLERS
- ASSIST CONTROLLERS IN INITIALIZATION OF IAS COMPONENTS
- ESTABLISH AND MAINTAIN OPERATING MODE
- MONITOR STATUS OF ALL IAS COMPONENTS
- FORMULATE ERROR MESSAGES
- MAINTAIN COMMUNICATION WITH SPACECRAFT AND GROUND
- PROVIDE COMPUTATIONAL SUPPORT TO IAS MODULES

ADAPTIVE SYSTEM CONTROLLER DESIGN FEATURES

- SOPHISTICATED MICROPROCESSOR ARCHITECTURE
- ADAPTABLE
- EXPANDABLE
- COMPATABLE WITH HIGH ORDER LANGUAGE

Page Intentionally Left Blank

APPLICATION OF A MICROPROCESSOR TO A SPACECRAFT ATTITUDE CONTROL

D. H. Brady and F. W. Hermann
TRW Defense and Space Systems Group
Redondo Beach, California

The space-qualified TDRSS attitude control system (ACS) microprocessor development work spanned three main design areas: hardware and instruction set, ACS firmware, and hardware/firmware verification testing.

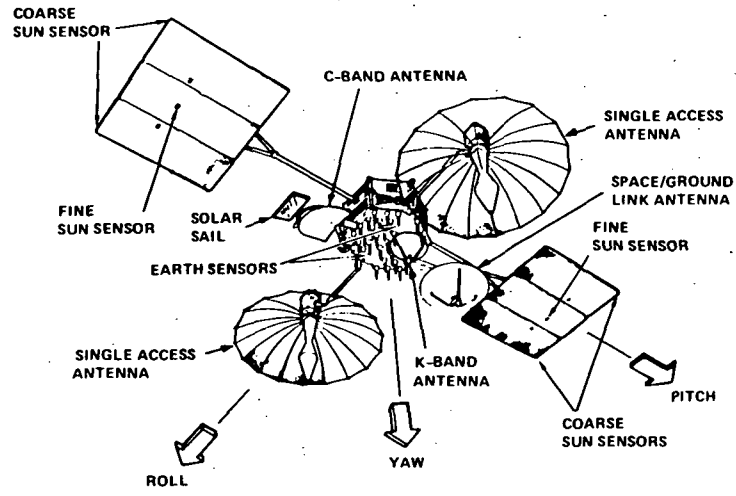
The Control Processor Electronics (CPE) hardware utilizes two parallel AM2901 4-bit microprocessors, with a microprogrammed instruction set tailored to the TDRSS controls application. Fourteen special purpose I/O instructions interface with the ACS hardware, each transferring data for one sensor while meeting timing and data-handling requirements. The ACS firmware resides in 5120 bytes of ROM with 1024 bytes of RAM for scratch data storage. The 16-bit add, subtract, and divide operations are overflow-protected and multiplies are done in hardware.

The firmware includes data processing for five sensors, four attitude control laws, and telemetry and commands. The main design limitations were: 16-bit word length, 1024 8-bit byte RAM, and computation speed/task sharing tradeoffs. It performed ACS computations quickly and without significant data degradation. The word length limitation motivated the careful selection of control filter topology and sacrifice of dynamic range in favor of null performance.

The flight program design was tested with three tools: Varian V-73 minicomputer, CDC Cyber computer, and the CPE development station. The V-73 simulation tested source (assembly) programs prior to development station availability. The CDC Cyber simulation was used primarily for accuracy analysis. The development station included commercial versions of the flight hardware which were run at full speed. All formal verification tests were run on the development station to verify timing, accuracy, and interface requirements.

From this development experience, additional hardware and software requirements were identified:

- Rapid Memory Examine/Change
- Floating point hardware
- High level language



TDRSS SPACECRAFT CONFIGURATION

TRACKING AND DATA RELAY SATELLITE (TDRSS) MISSION:

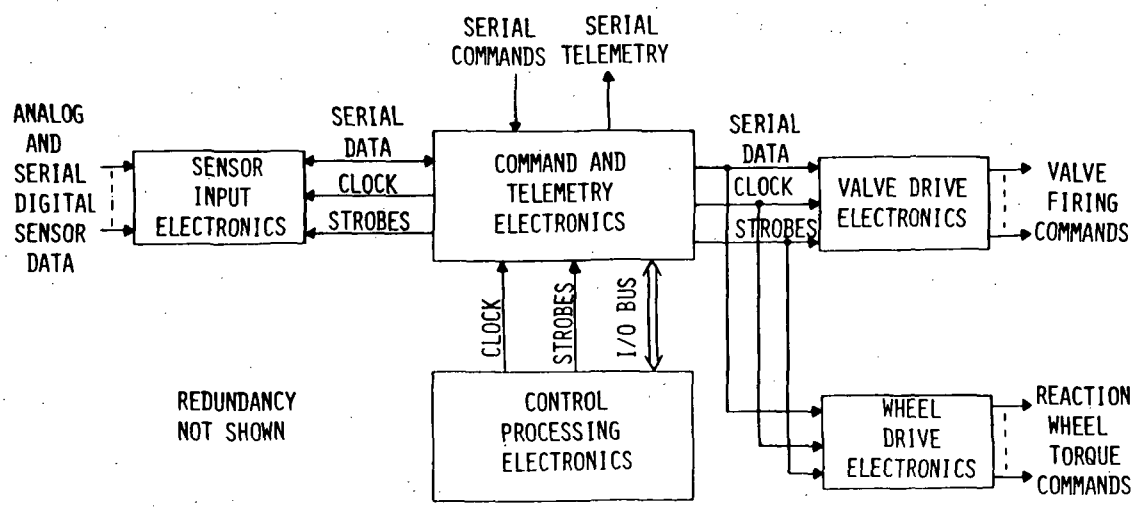
- COMMERCIAL COMMUNICATION FOR WESTERN UNION
- SATELLITE TRACKING AND DATA RELAY FOR NASA

- TDRSS IS ONE RECENT TRW APPLICATION OF THE USE OF A MICROPROCESSOR TO PERFORM SPACECRAFT ATTITUDE CONTROL SYSTEM COMPUTATIONS
- DEVELOPMENT WORK SPANNED THE FOLLOWING AREAS:
  - HARDWARE DESIGN OF THE CONTROL PROCESSING ELECTRONICS (CPE), A SPECIAL-PURPOSE MICROCOMPUTER BASED ON THE AMD 2901 BIT-SLICE MICROPROCESSOR
  - DEFINITION OF THE INSTRUCTION SET
  - DESIGN AND CODE THE FIRMWARE FLIGHT PROGRAM
  - VERIFICATION TESTING OF THE FIRMWARE

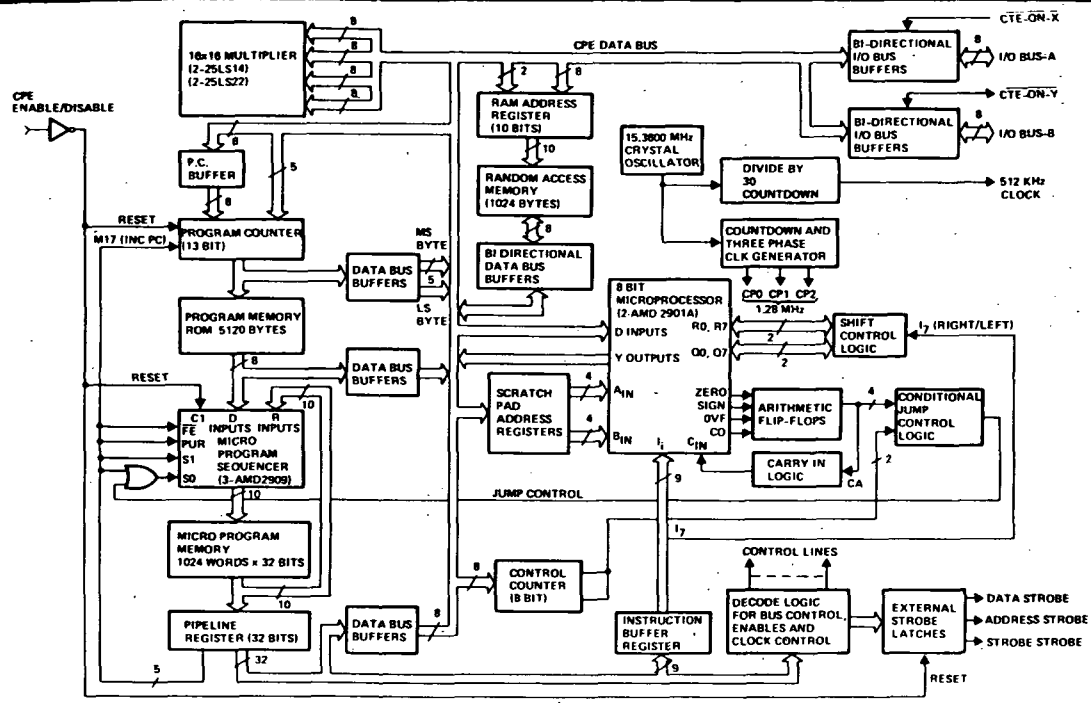
TDRSS ATTITUDE CONTROL HARDWARE

- CONTROL ELECTRONICS ASSEMBLY
- SENSORS
  - EARTH SENSOR
  - COARSE SUN SENSORS
  - FINE SUN SENSORS
  - GYROS
  - REACTION WHEEL TACHOMETERS
  - SOLAR ARRAY DRIVE RESOLVERS
- ACTUATORS
  - THRUSTERS
  - REACTION WHEELS

TDRSS CONTROL ELECTRONICS ASSEMBLY



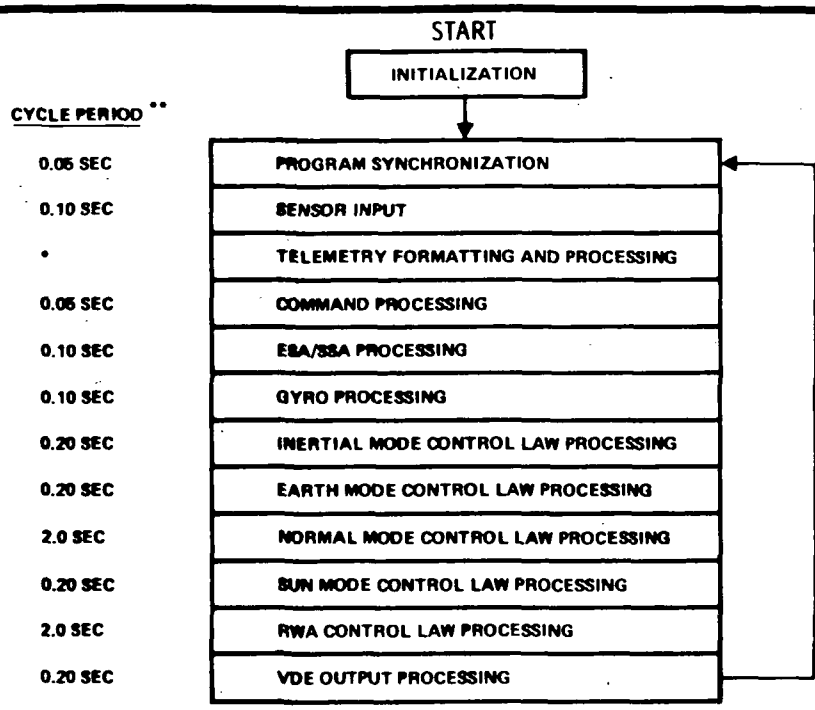
CPE BLOCK DIAGRAM



CPE INSTRUCTION SET

- SPECIFIED BY CPE CONTROL FIRMWARE DESIGNERS TO BE OPTIMUM FOR TDRSS APPLICATION
- INSTRUCTION SET MICROPROGRAMMED BY HARDWARE DESIGNERS
- 108 TOTAL INSTRUCTIONS INCLUDING:
  - SINGLE- OR MULTIPLE-BIT SET, RESET AND TEST
  - 16-BIT FRACTIONAL FIXED-POINT ARITHMETIC
  - 16-BIT ADD, SUBTRACT AND DIVIDE CLAMPED AT ± 1.0 WHEN OVERFLOW OCCURS
  - MULTI-BIT LOGICAL AND ARITHMETIC SHIFTS (SINGLE AND DOUBLE-PRECISION)
  - FOURTEEN INPUT/OUTPUT INSTRUCTIONS DEDICATED TO PARTICULAR SENSORS OR ACTUATORS WITH ALL HARDWARE TIMING BEING TAKEN CARE OF IN THE MICROCODE. FOR EXAMPLE, INSS INSTRUCTION INPUTS ALL DATA ASSOCIATED WITH THE COARSE AND FINE SUN SENSORS TO PAGE 0 OF RAM; OUWTQ OUTPUTS TORQUE COMMANDS TO THE REACTION WHEELS FROM PAGE 0 OF RAM.

PROGRAM STRUCTURE



*TELEMETRY BIT RATES OF 250, 1000, AND 4000 BITS/SECOND ARE SELECTABLE FOR THE 512 BIT MAIN FRAME. THESE PRODUCE TELEMETRY CYCLE TIMES OF 2.048, 0.512, AND 0.128 SECONDS RESPECTIVELY.

**EXECUTION FREQUENCY (NOT EXECUTION TIME) FOR EACH PROGRAM MODULE.

FIRMWARE DEVELOPMENT/VERIFICATION PHILOSOPHIES

DESIGN SCOPE -

- MICROPROCESSOR DEDICATED TO ATTITUDE CONTROL FUNCTION
- INSTRUCTION SET TAILORED TO CONTROL NEEDS (ARITHMETIC LIMITING, CONTROL HARDWARE I/O)

DESIGN PERSONNEL

- FIRMWARE DESIGNED CODED, AND TESTED BY CONTROL SYSTEMS ENGINEERS
  - IMPROVED COMMUNICATION FOR MECHANIZATION OF CONTROLS REQUIREMENTS IN FIRMWARE
  - INTERACTION IMPROVED FOR ACHIEVING ADEQUATE PERFORMANCE IN THE FIRMWARE

INDEPENDENT VERIFICATION

- ACHIEVED BY SWAPPING MODULE RESPONSIBILITY AT VERIFICATION TIME
- VERIFICATION TEST PLAN INDEPENDENT OF MODULE DESIGNERS
- REVIEW OF TEST RESULTS BY CONTROL LOOP ANALYSTS

CONSTRAINTS AND THEIR EFFECTS ON FIRMWARE DESIGN

TRW

16 BIT WORD LENGTH MAXIMUM, 8-BIT BYTES

- CAREFUL MAGNITUDE SCALING FOR COMPUTATIONS
- QUANTIZATION EFFECTS ON CONTROL FILTER TOPOLOGIES
- LOW LEVEL PERFORMANCE/DYNAMIC RANGE TRADEOFFS

1024 8-BIT BYTES ADDRESSABLE IN RAM

- ATTITUDE CONTROL FILTER COMPLEXITY
- FILTER STATE ACCURACY/MULTIPLE PRECISION ARITHMETIC

DATA PROCESSING

- COMPUTATION COMPLEXITY/TIME DELAYS
- COMPUTATION COMPLEXITY/TASK SHARING

FILTER IMPLEMENTATION TOPOLOGIES

TRW

WORD LENGTH EFFECTS -

- ACCURACY, QUANTIZATION

FILTER TOPOLOGIES -

- POLYNOMIAL FORM (REJECTED)
- CASCADED FORM (ACCEPTED)

OTHER ALTERNATIVE FORMS

- PARALLEL FORM
- MATRIX FORM
- DFT, FFT

SCALING CONSIDERATIONS

- TOPOLOGY AND GAIN DISTRIBUTION FOR INTERMEDIATE STATES
- DYNAMIC RANGE, ETC.

FIRMWARE DEVELOPMENT & VERIFICATION TOOLS

TRW

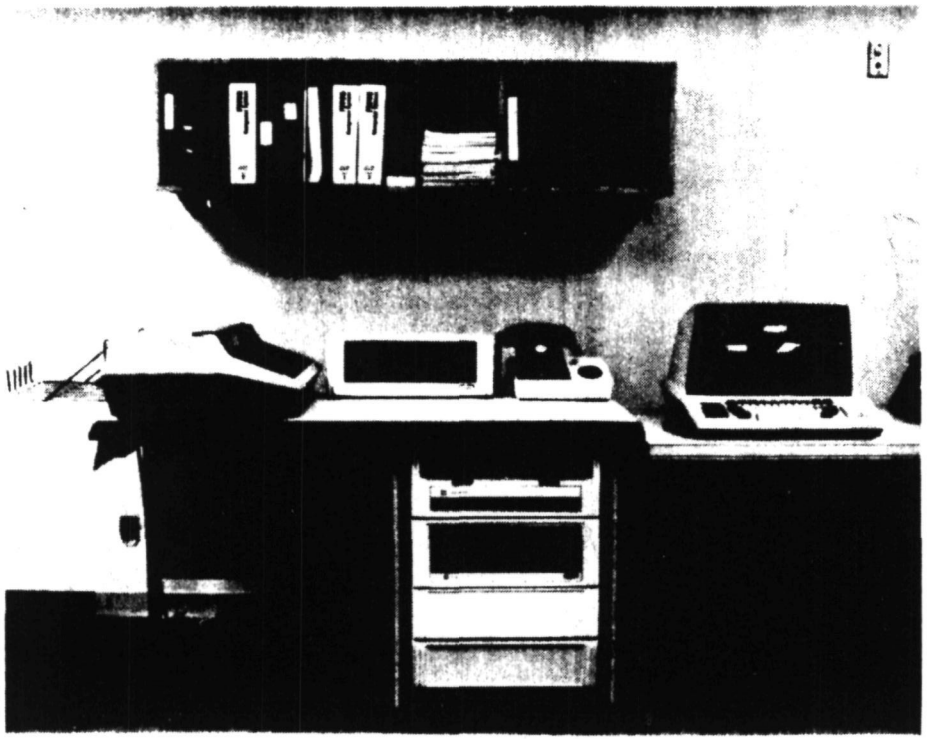
- VARIAN V-73 MINICOMPUTER
  - 8 AND 16 BIT ARITHMETIC
  - FORTRAN FUNCTIONS SIMULATING INDIVIDUAL CPE INSTRUCTIONS
  - PROVIDES INTERFACE IN ENGINEERING UNITS FOR MAXIMUM VISIBILITY
  - INPUT SAME AS ASSEMBLER INPUT
- CYBER 74 TIMESHARE
  - SCIENTIFIC SIMULATION OF ARITHMETIC INSTRUCTIONS
  - PROVIDES TOOL FOR CONTROL LOOP DESIGNERS
    - ASSESS LIMITED WORD LENGTH EFFECTS
    - FILTER DESIGN
  - RESIDENCE FOR CONTROLLED CPE SOURCE PROGRAM
  - RESIDENCE FOR CPE ASSEMBLER PROGRAM

FIRMWARE DEVELOPMENT & VERIFICATION TOOLS (CONTINUED)

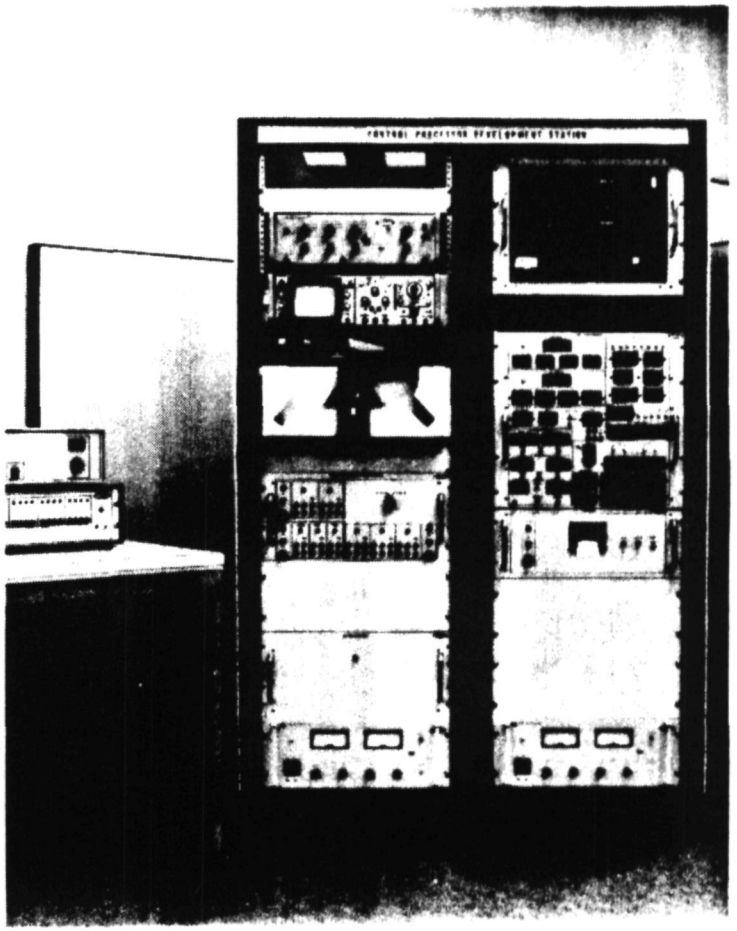
TRW

- TI 990/10 MINICOMPUTER
  - VERIFICATION TOOL
  - USED AS DRIVER FOR VERIFICATION TESTS
  - DATA COLLECTION AND PRINTOUT
  - PROVIDES DYNAMICS SIMULATION FOR DYNAMIC ENVIRONMENT
- CPE DEVELOPMENT STATION
  - BREADBOARD CPE
  - FRONT PANEL FOR PROGRAM ENTRY, EDIT, AND EXECUTION
- INTERFACE BOX
  - HARDWARE LINK OF TI 990/10 AND BREADBOARD CPE
  - GROUND STATION COMMAND SIMULATOR

DEVELOPMENT STATION EQUIPMENT



TF 990/10 AND PROM PROGRAMMER



INTERFACE BOX AND CONTROL
PROCESSOR DEVELOPMENT STATION

REVIEWING THE FIRMWARE DESIGN

TRW

DESIGN IMPROVEMENT, FUTURE DESIGNS ARE ENHANCED BY DESIGN REVIEW AT SIGNIFICANT MILESTONES -

- INITIAL DESIGN
- COARSE PROGRAM FLOWS
- DETAILED PROGRAM FLOWS
- CODE CHECKOUT, MODULE LEVEL
- TOTAL INTEGRATED PROGRAM VERIFICATION
- SUBSYSTEM TESTING
- SPACECRAFT OPERATIONS DESIGN
- FLIGHT EXPERIENCE

DESIGN REVIEW AREAS -

- HARDWARE DESIGN
- MICROCODE DESIGN
- INSTRUCTION SET DESIGN
- GROUND STATION COMMANDS SET
- TELEMETRY DATA AVAILABLE
- PROGRAM STRUCTURE
- DESIGN PHILOSOPHIES

REVIEWING THE FIRMWARE DESIGN

TRW

SOME DESIGN IMPROVEMENTS UNCOVERED -

- RAM MEMORY LOAD AND MOVE DATA INSTRUCTION
- EXECUTIVE PROGRAM AND SUBROUTINES STRUCTURE
- GROUND COMMANDS TAILORED TO SPACECRAFT OPERATIONS
- MICROPROCESSOR DEVELOPMENT STATION SOFTWARE CHECKOUT FEATURES
  - RAM
  - SCRATCHPAD VISIBILITY
  - RAPID MEMORY EXAMINE/CHANGE
- CONTROL FILTER STATES WORD LENGTH $\geq$ 24 BITS
- FLOATING POINT HARDWARE
- HIGH LEVEL LANGUAGE

Page Intentionally Left Blank

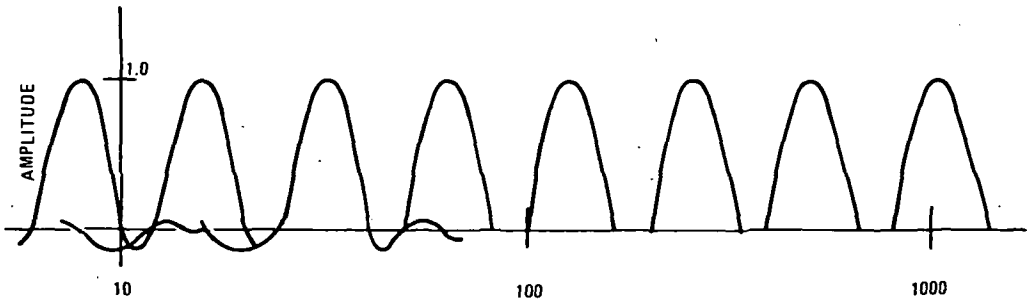
A PLASMA WAVE FOURIER TRANSFORM PROCESSOR EMPLOYING 1802 MICROCOMPUTERS FOR SPACECRAFT INSTRUMENTATION

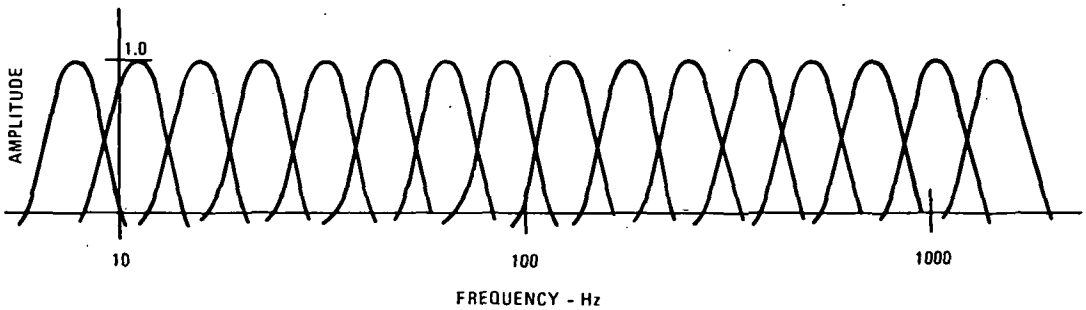
Donald C. Lokerson and James N. Caldwell
Goddard Space Flight Center
Greenbelt, Maryland

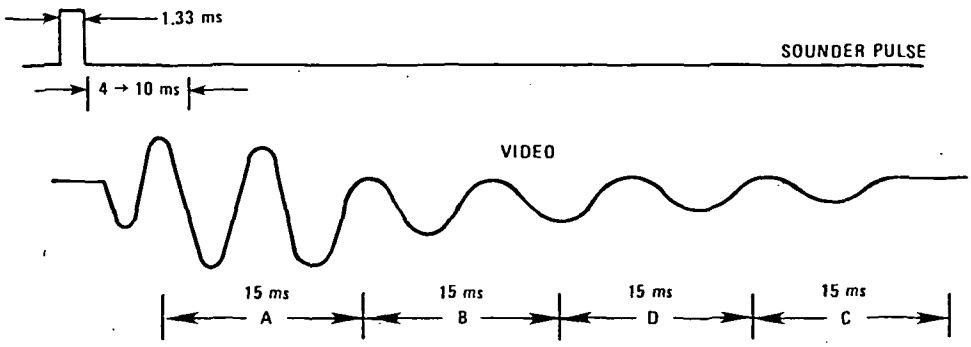
The requirements of low power, small space, low weight, and high data compression limit the capabilities of interplanetary plasma wave processing. The processing of plasma wave signals in spinning spacecraft in the range from D.C. to 500Hz are described. Another signal source from the intermediate frequency of a sounder transponder is processed by the same hardware, but with linear Fourier transform software.

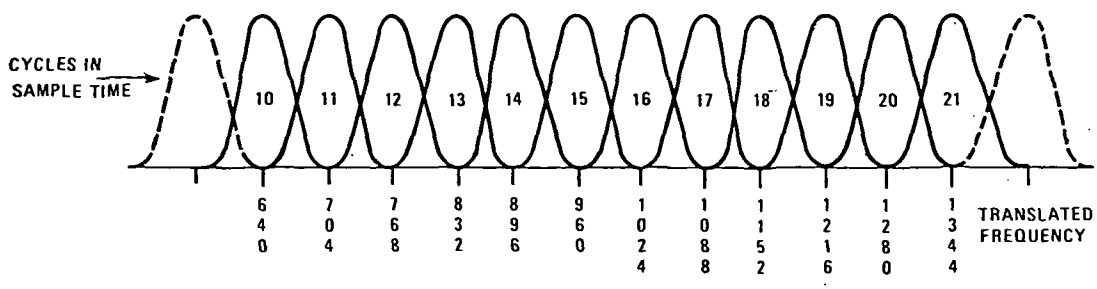
Three antenna inputs are filtered to prevent aliasing above 500Hz. A 92db range of data is multiplexed to an analog-to-digital converter, and then to three RCA 1802 microcomputers. Groups of this data are collected under DMA control by each microcomputer. A logarithmically-spaced Fourier transform is computed. Each computer requires 2 kilowords of read-only-memory and a half kiloword of random access memory.

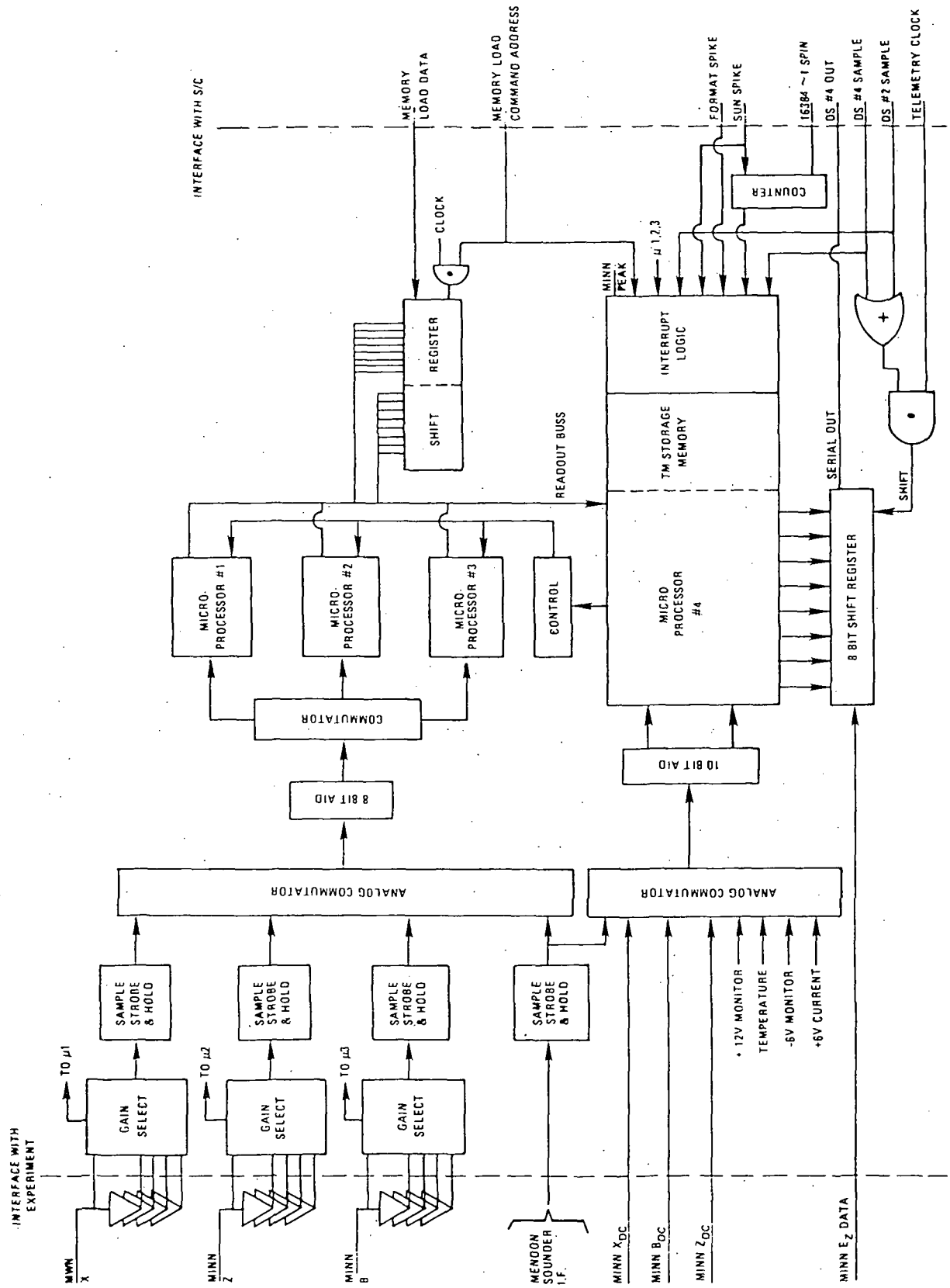
Data at lower frequencies is spin modulated by the spacecraft. For these signals, data is sampled 512 times in one spin and also processed by logarithmically-spaced Fourier transform techniques. A fourth microcomputer performs this task and coordinates all operations. This microcomputer requires 5 kilowords of read-only-memory and 3.5 kilowords of random address memory. The computer also performs data averaging, peak detection, data formatting, output compression, and ground commanded tasks.


AMPLITUDE
1.0
10
100
1000
FREQUENCY - Hz


AMPLITUDE
1.0
10
100
1000
FREQUENCY - Hz

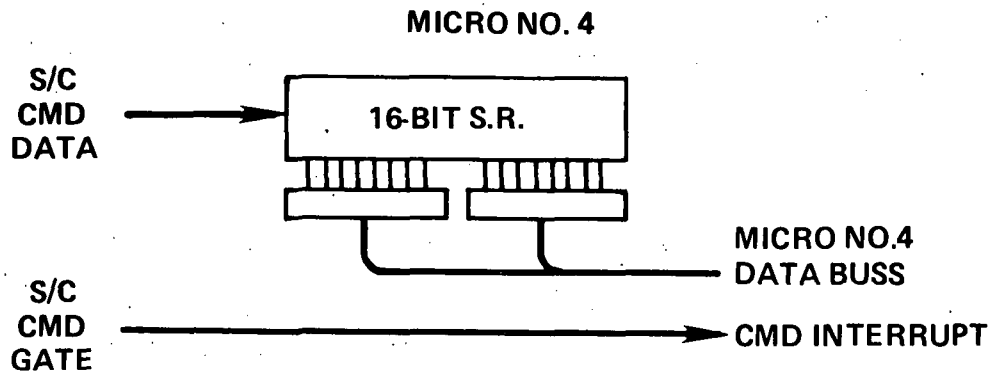

1.33 ms
SOUNDER PULSE
4 → 10 ms
VIDEO
15 ms
A
15 ms
B
15 ms
D
15 ms
C

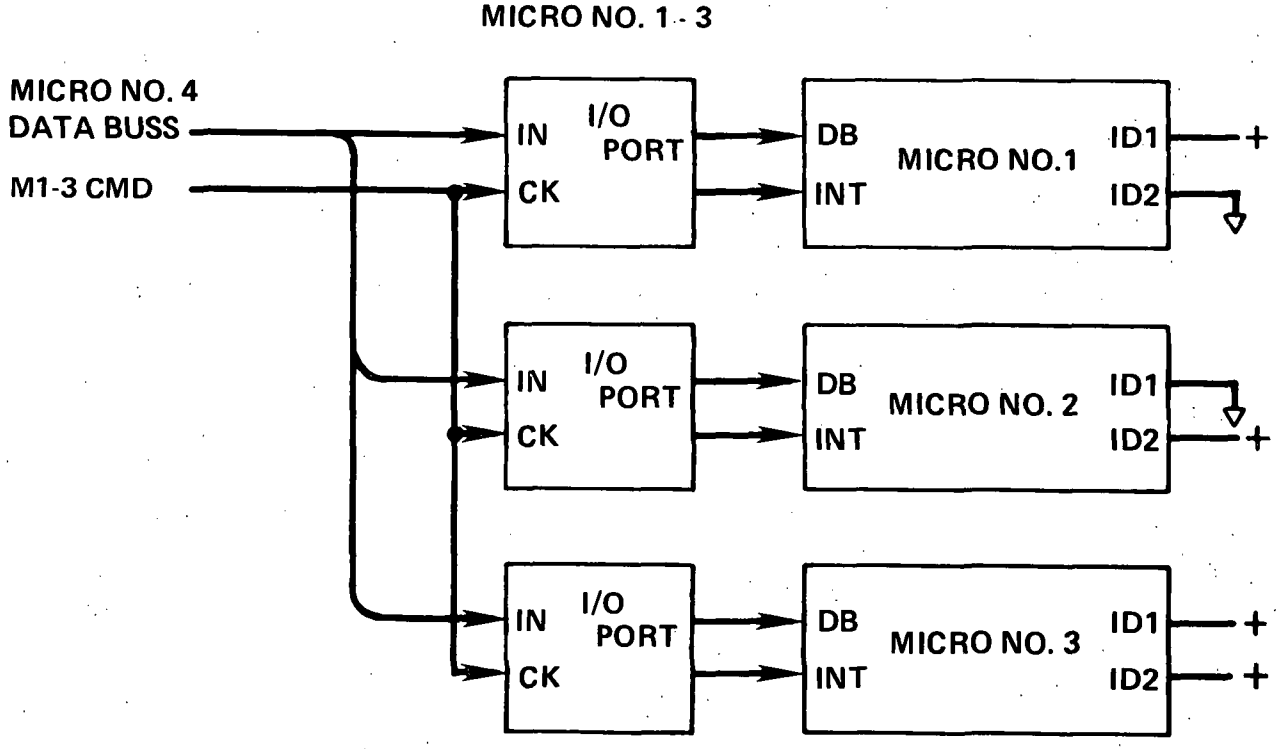

CYCLES IN SAMPLE TIME
10
11
12
13
14
15
16
17
18
19
20
21
640
704
768
832
896
960
1024
1088
1152
1216
1280
1344
TRANSLATED FREQUENCY



Functional block diagram of SPM-FFT Processor

COMMANDS


MICRO NO. 4
S/C
CMD
DATA
16-BIT S.R.
MICRO NO.4
DATA BUSS
S/C
CMD
GATE
CMD INTERRUPT


MICRO NO. 1 - 3
MICRO NO. 4
DATA BUSS
M1-3 CMD
IN
CK
I/O
PORT
DB
INT
MICRO NO.1
ID1
ID2
+
MICRO NO. 2
MICRO NO. 3

INPUT

STATE 0

- GAIN RANGE ALL INPUT GROUPS
- STROBE ALL SAMPLE AND HOLD AMPLIFIERS

STATE 1

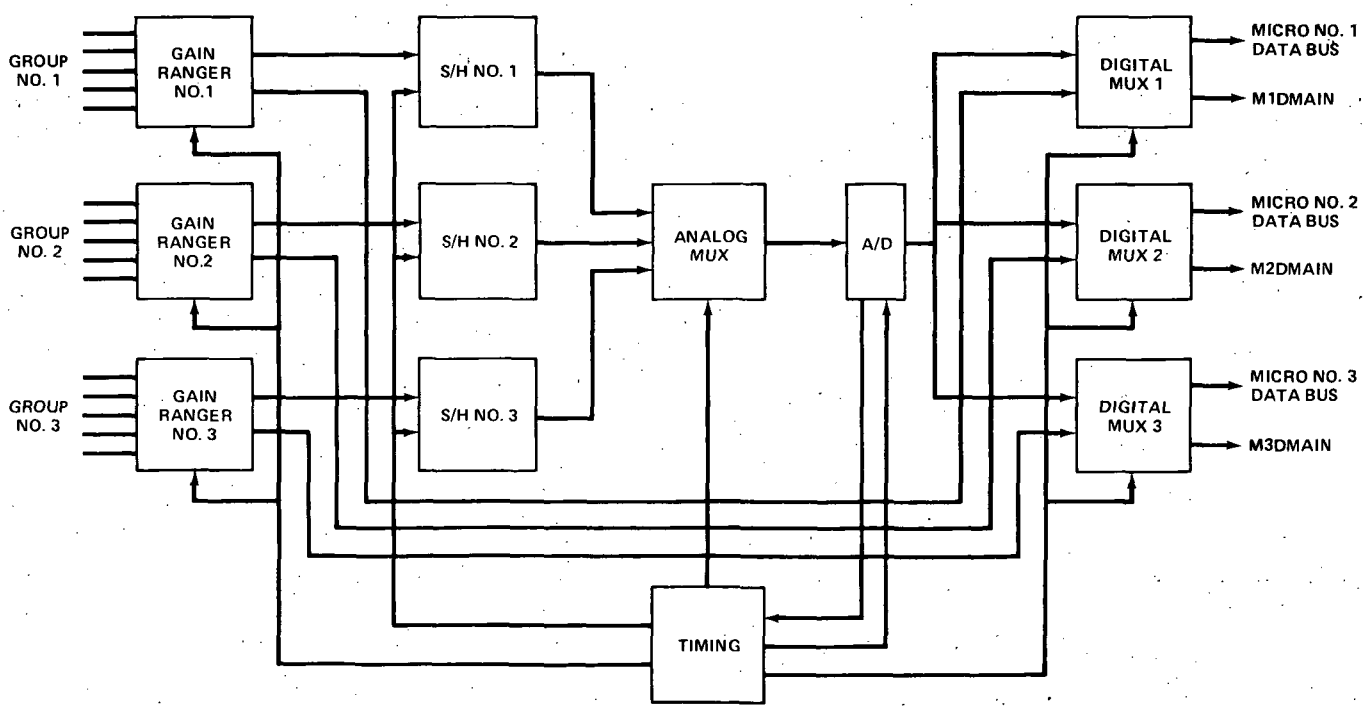
- A/D CONVERT S/H NO. 1
- DMA GAIN RANGE AND A/D DATA FOR GROUP NO. 1 TO MICRO NO. 1

STATE 2

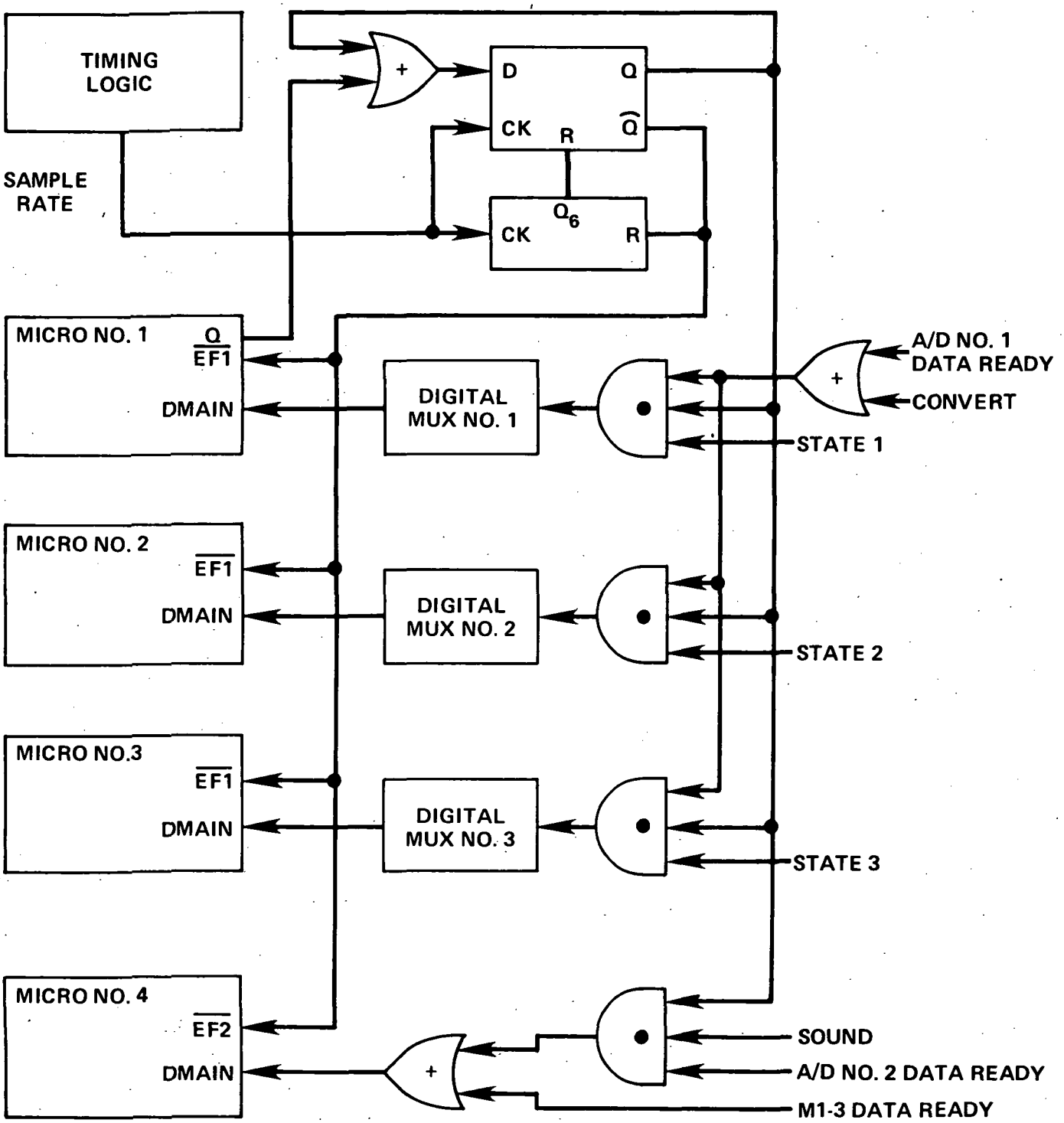
- A/D CONVERT S/H NO. 2
- DMA GAIN RANGE AND A/D DATA FOR GROUP NO. 2 TO MICRO NO. 2

STATE 3

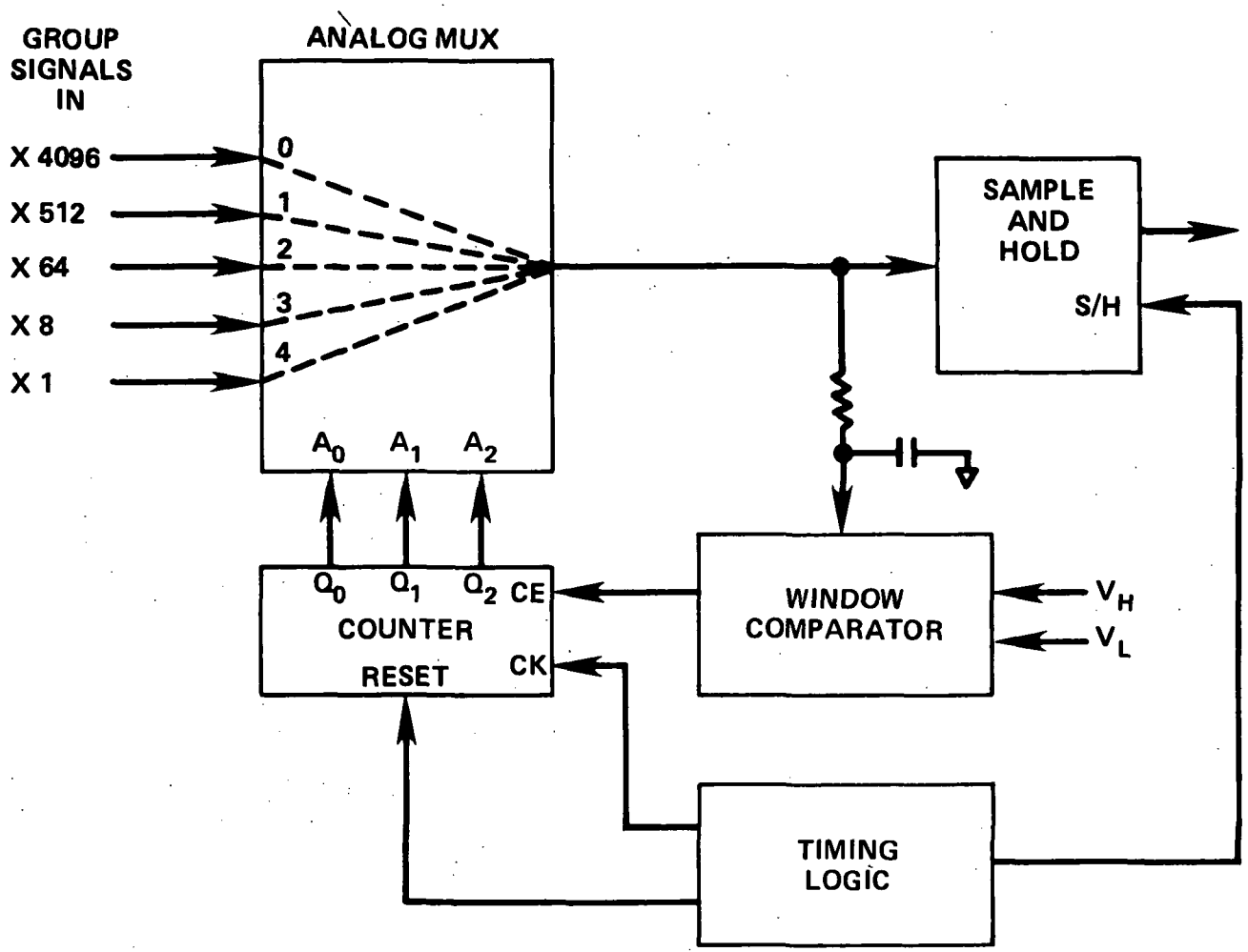
- A/D CONVERT S/H NO. 3
- DMA GAIN RANGE AND A/D DATA FOR GROUP NO. 3 TO MICRO NO. 3

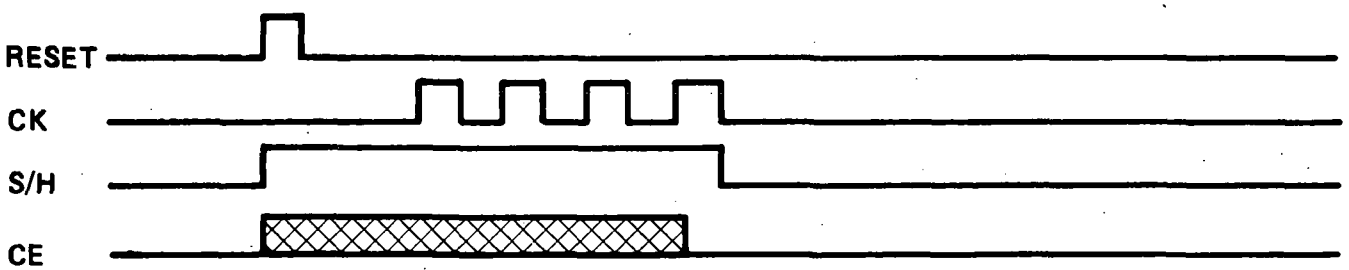


DMA CONTROL

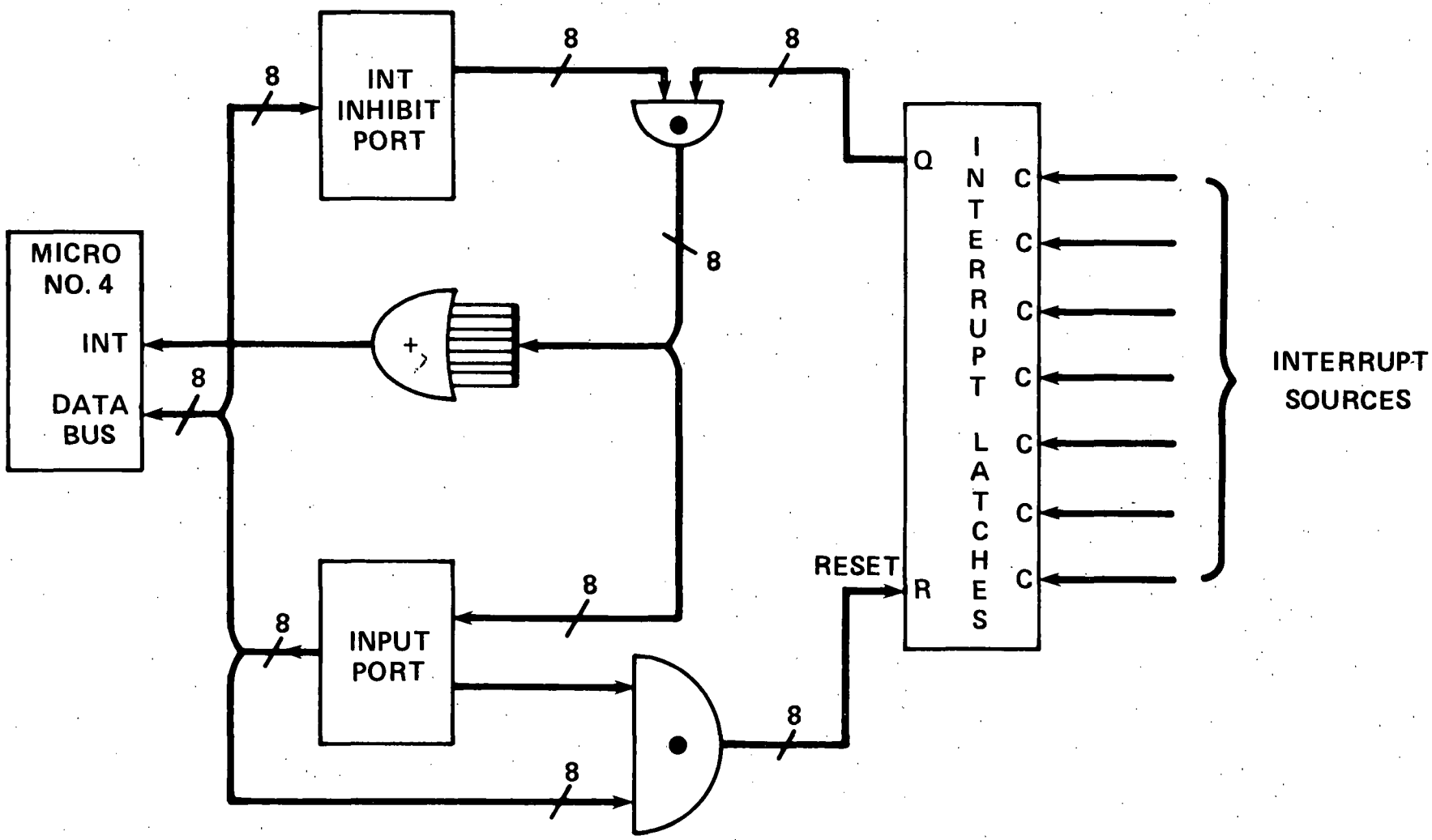

TIMING LOGIC
SAMPLE RATE
+
D
Q
CK
R
Q
Q6
CK
R
MICRO NO. 1
Q
EF1
DMAIN
DIGITAL MUX NO. 1
A/D NO. 1 DATA READY
CONVERT
STATE 1
MICRO NO. 2
EF1
DMAIN
DIGITAL MUX NO. 2
STATE 2
MICRO NO.3
EF1
DMAIN
DIGITAL MUX NO. 3
STATE 3
MICRO NO. 4
EF2
DMAIN
SOUND
A/D NO. 2 DATA READY
M1-3 DATA READY

GAIN RANGING





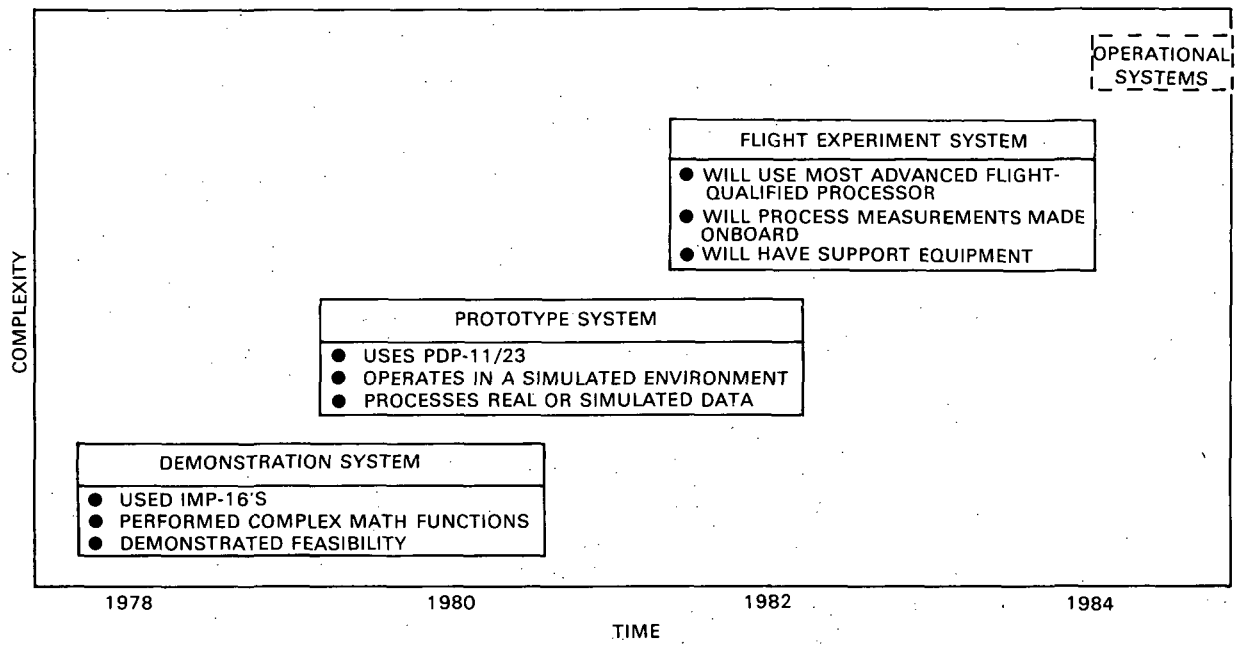
INTERRUPT LOGIC



PROTOTYPE DEVELOPMENT OF A MICROPROCESSOR-BASED ONBOARD ORBIT DETERMINATION SYSTEM

Keiji K. Tasaki and Rose S. Pajerski
Goddard Space Flight Center
Greenbelt, Maryland

DEVELOPMENT STAGES OF ONBOARD ORBIT DETERMINATION SYSTEMS



OBJECTIVES

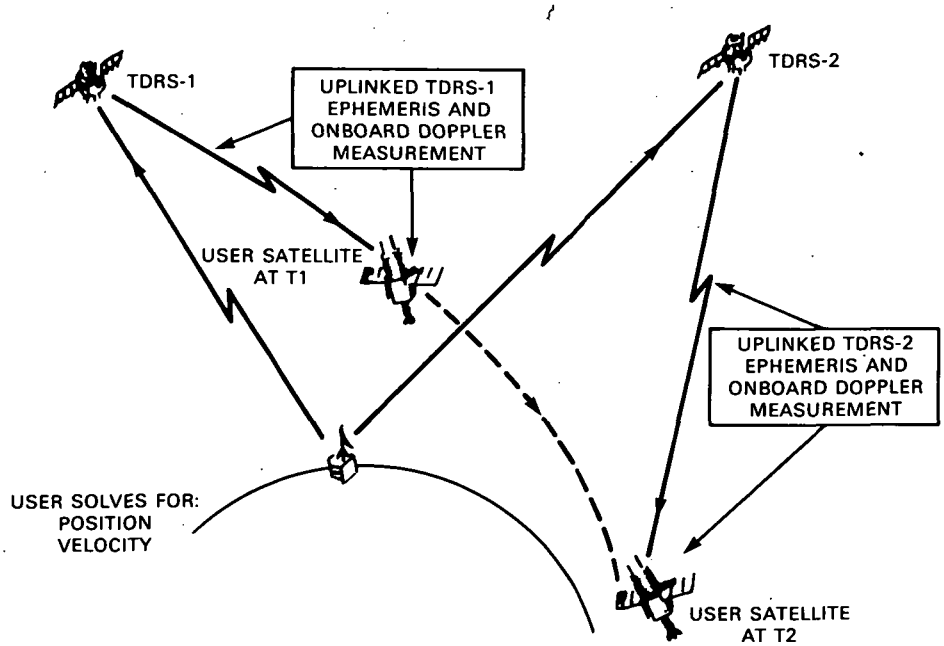
1. DEVELOP A MICROPROCESSOR-BASED AUTOMATED ORBIT DETERMINATION SYSTEM (AODS) USING:

   - PDP-11/70 AS THE DEVELOPMENT MACHINE
   - PDP-11/23 AS THE TARGET MACHINE
   - HIGH-LEVEL LANGUAGE - FORTRAN

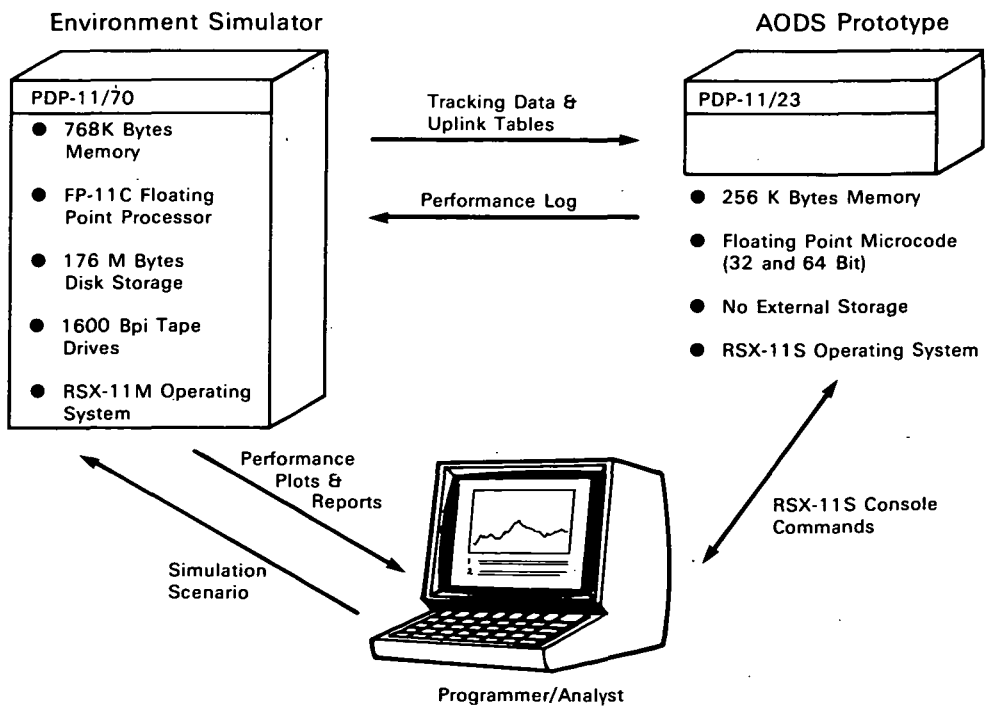
2. EXERCISE THE SYSTEM IN CONJUNCTION WITH A SIMULATOR.

3. REFINE SOFTWARE TO ACHIEVE HIGHER EFFICIENCY.

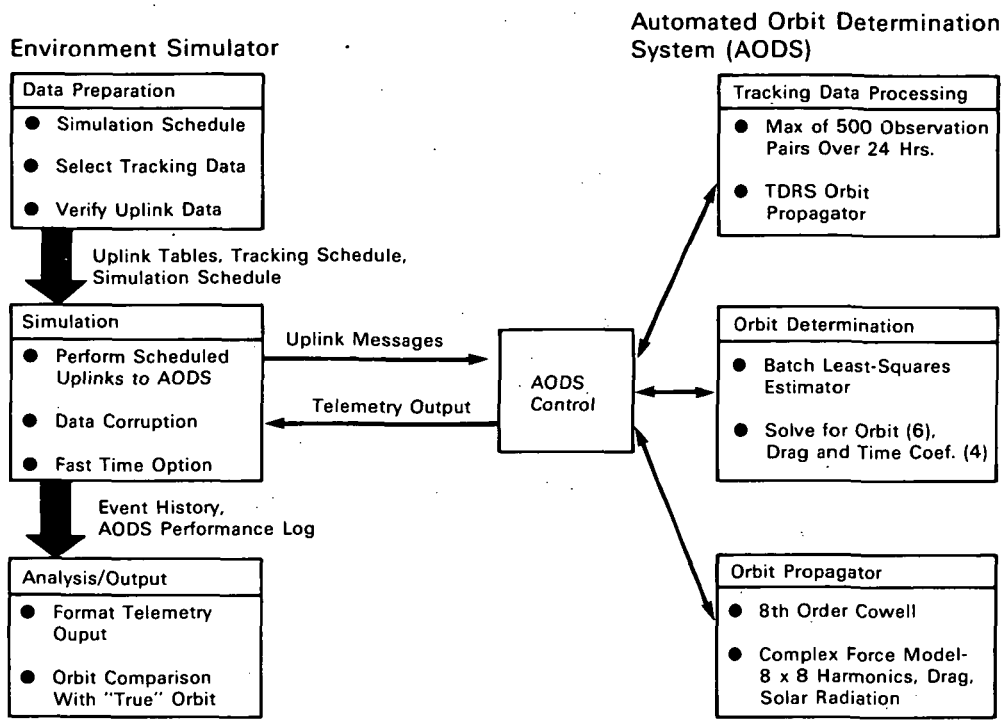
ONBOARD NAVIGATION WITH TDRSS



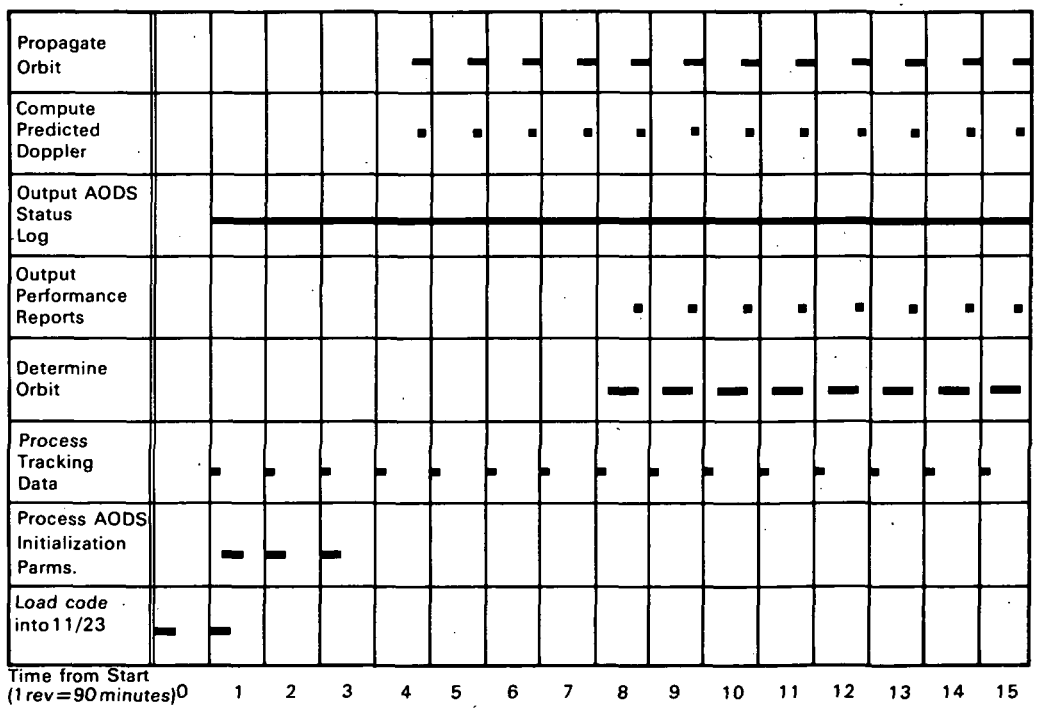
HARDWARE OVERVIEW



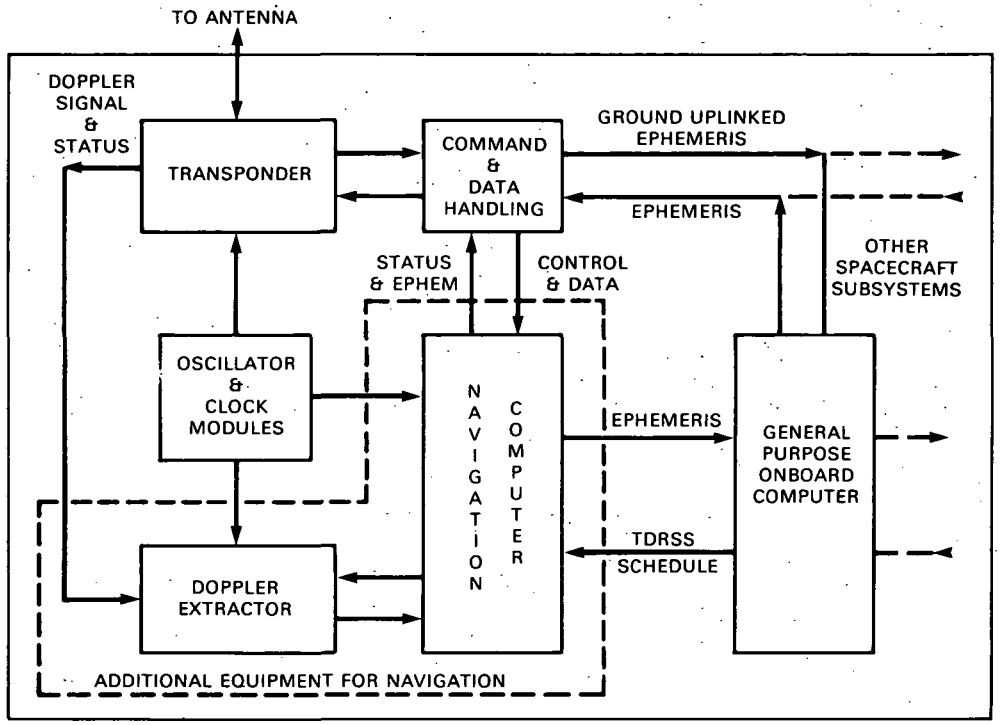
SOFTWARE OVERVIEW



TYPICAL AODS SIMULATION SCENARIO



FLIGHT EXPERIMENT CONFIGURATION



FLIGHT EXPERIMENT CONSIDERATIONS

1. SELECTION OF A FLIGHT-QUALIFIED PROCESSOR

   - 64-BIT FLOATING POINT ARITHMETIC
   - LARGE ADDRESSABLE MEMORY (BEYOND 64K)
   - MULTITASKING OPERATING SYSTEM

2. HARDWARE INTERFACE

   - RECEIVER - AODS - COMMAND AND DATA MODULE
   - AODS - OTHER ONBOARD PROCESSORS

SESSION II

GROUND BASED AEROSPACE MICROPROCESSOR APPLICATIONS

THE REMOTE COMPUTER CONTROL (RCC) SYSTEM

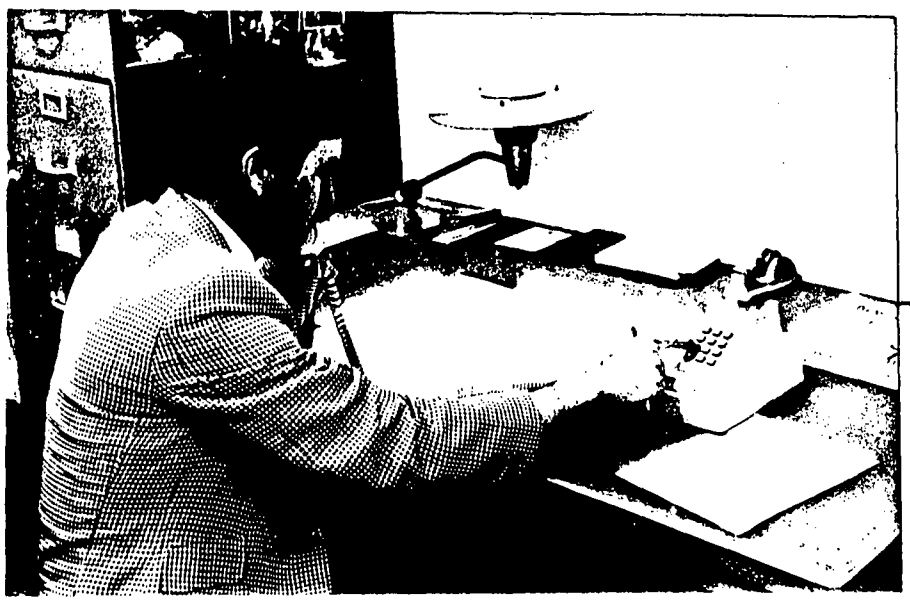
William Holmes
Goddard Space Flight Center
Greenbelt, Maryland

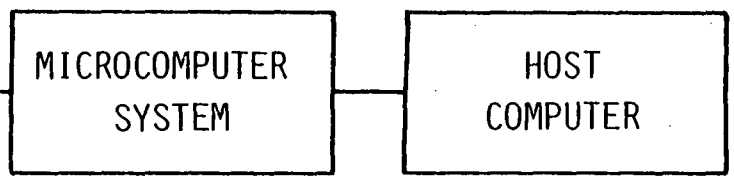
A system to remotely control job flow on a host computer from any touchtone telephone is currently being developed at Goddard Space Flight Center (GSFC). Using this system a computer programmer can submit jobs to a host computer from any touchtone telephone. In addition the system can be instructed by the user to call back when a job is finished. Because of this system every touchtone telephone becomes a conversant computer peripheral. This system known as the Remote Computer Control (RCC) system utilizes touchtone input, touchtone output, voice input, and voice output. The RCC system is microprocessor based and is currently using the INTEL 80/30 microcomputer. Using the RCC system a user can submit, cancel, and check the status of jobs on a host computer. A user can also have the RCC system call when a specified condition is fulfilled. For example, a user could have the RCC call when a specific job has been successfully completed on a host computer.

The peripherals used for communication with the user over the telephone are the MH88205 DTMF Receiver/Decoder by Mitel for touchtone input, the MC14410P integrated circuit by Motorola for touchtone output, the Voice Recognition Module (VRM) by Interstate Electronics for voice input and the ML-I Multi-Lingual Voice System by Federal Screw Works for voice output. The RCC system peripherals consist of a CRT for operator control, a printer for logging all activity, mass storage for the storage of user parameters, and a PROM card for program storage.

This RCC system enables a user to communicate with a host computer and control job flow on a host computer from any touchtone telephone at any time. The use of this system can decrease turnaround time on a host computer by minimizing the time between job termination and user notification of job termination. This system can help distribute the work load of a host computer to off hours by enabling a user to control the host computer job flow from any remote touchtone telephone.

Remote Computer Control





- CONTROL JOB FLOW
- HAVE HOST COMPUTER GENERATE TELEPHONE CALLS UPON REQUEST

Pilot System

- BUY THE DEVICES
- BUILD THE INTERFACES
- PROGRAM THE MICROPROCESSOR

SYSTEM COMMANDS / DEMONSTRATION

- LOGON NUMERIC ID'S CORRELATED WITH VALID HOST COMPUTER ID'S
- SUBMIT JOBS PRE-STORED ON DISK IN HOST MACHINE
- STATUS BY NUMERIC JOB ID
- CANCEL BY NUMERIC JOB ID
- NOTIFY TO ANY TELEPHONE NUMBER
- TERMINATE SESSION

A MICROPROCESSOR APPLICATION TO A STRAPDOWN LASER GYRO NAVIGATOR

C. Giardina and E. Luxford
The Singer Company-Kearfott Division
Little Falls, New Jersey

This paper is concerned with replacing analog circuit control loops for laser gyros (path length control, cross axis temperature compensation loops, dither servo and current regulators), with digital filters residing in microcomputers. The object of using this type of design is to improve on system reliability (through part count reduction), reduce size and power requirements, and therefore, improve on system performance. Consistent replication in the design is a further benefit derived by replacing analog components with digital software.

In addition to the control loops, a discussion will be given on applying the microprocessor hardware to compensation for coning and skulling motion where simple algorithms are processed at high speeds to compensate component output data (digital pulses) for linear and angular vibration motions.

Highlights are given on the methodology and system approaches used in replacing differential equations describing the analog system in terms of the mechanized difference equations of the microprocessor. Here standard one for one frequency domain techniques are employed in replacing analog transfer functions by their transform counterparts. Direct digital design techniques are also discussed along with their associated benefits. Time and memory loading analyses are also summarized, as well as signal and microprocessor architecture utilized to do the "best job".

Trade offs in algorithm, mechanization, time/memory loading, accuracy and microprocessor architecture are also given.

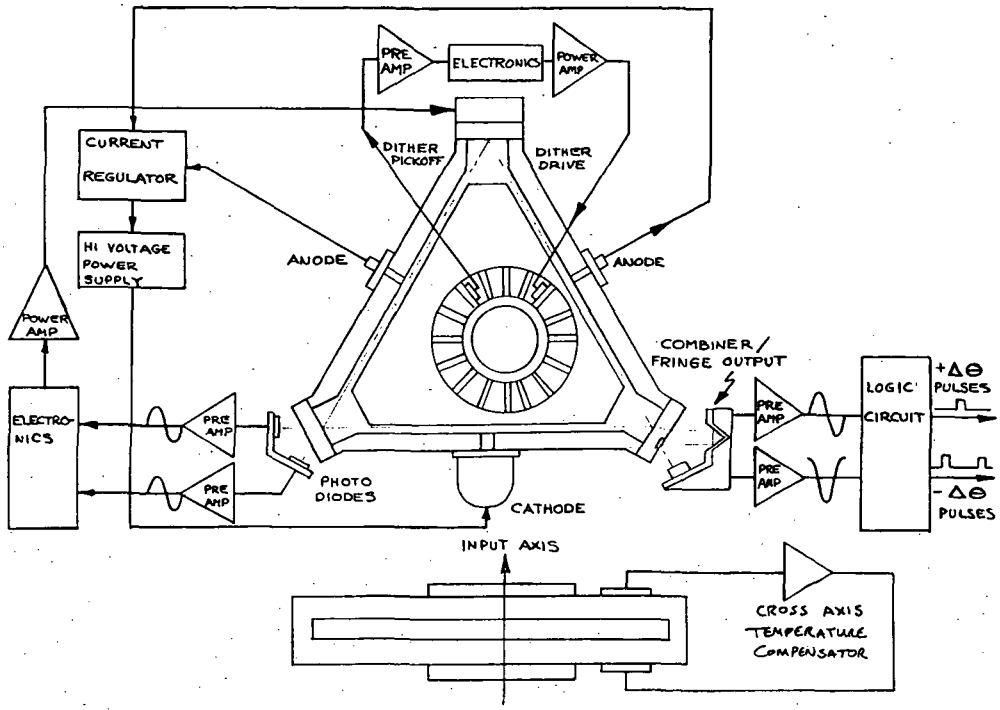
RLG CONTROL LOOPS - PERFORMANCE INFORMATION

Loop	Purpose of Loop	Method of Improving Performance	Required Control Loop Bandwidth
Dither	Eliminate damping in dither spring mechanism	Overcome backscattered light between two beams due to mirror (reflector) imperfections and thereby circumvent lock-in	Moderate when compared to microcomputer speeds
Current Regulator	Balance anode currents	Minimize drift due to gas flow in laser cavity	Long when compared to microcomputer speeds
Path Length Control	Maintain path length in cavity at an integral number of wave lengths	Minimize drift due to temperature variation of block	Long when compared to microcomputer speeds
Cross Axis Temperature Compensator	Center beam in cavity	Minimize drift due to temperature variation of block	Long when compared to microcomputer speeds
Variable Beam Intensity Corrector	Locate mirrors to minimize total backscatter "vector" from the three mirrors (thereby reducing effective lock-in level)	Minimize output random noise resulting from dither	Long when compared to microcomputer speeds

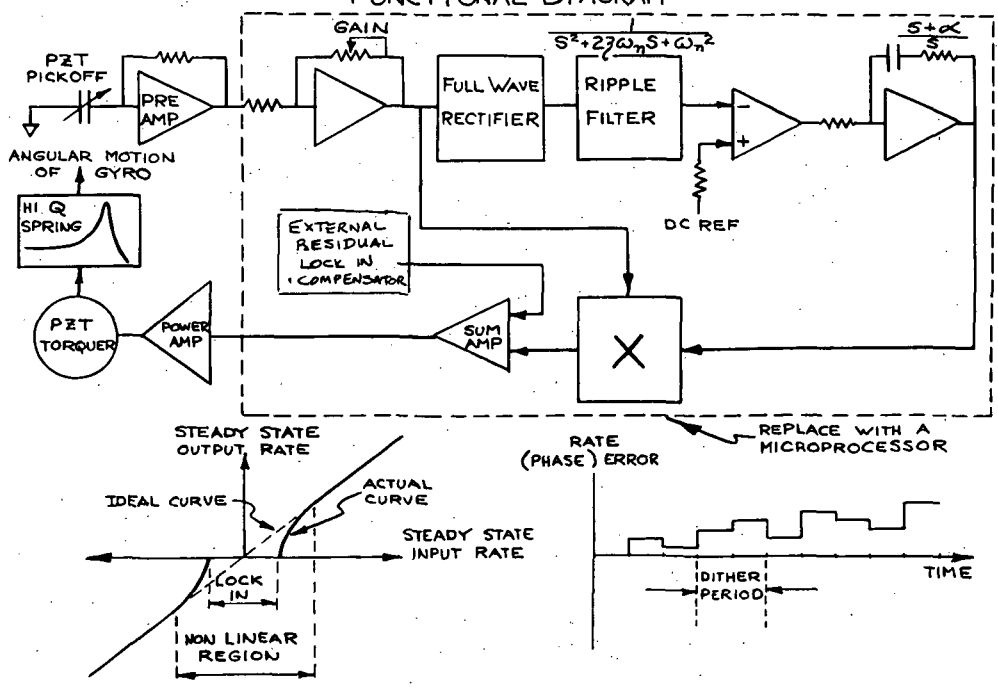
RLG CONTROL LOOPS - OBSERVABLES AND METHOD OF CONTROL COMPARISON

Loop	Observed Signal	Control Signal	Method of Observing Signal	
			Analog	Digital
Dither	Dither Amplitude	Force applied to gyro dither spring through voltage applied to PZT device	Full wave rectified dither amplitude	Measurement of peak dither amplitude through successive measurement of amplitude
Current Regulator	Difference in anode currents	Base voltage applied to control transistor	Output voltage from difference amplifier representing difference current	Same as analog
Path Length Control	Beam Intensity	Force applied to movable mirror through voltage applied to PZT device	Rectification of a carrier signal modulating beam intensity	Measurement of beam intensity variation with path length variation through successive iterations of path length
Cross Axis Temperature Compensator	Beam Intensity	Force applied to gyro block through voltage applied to PZT device	Rectification of a carrier signal modulating beam intensity	Measurement of beam intensity variation with force applied to the block through successive iterations of that applied force
Variable Beam Itensity Corrector	Variation in beam intensity (distortion) as gyro periodically locks in during dither reversals. Variation usually occurs only with a given gyro turn-on.	Force applied to a set of two movable mirrors through voltages applied to PZT devices	Rectification of a carrier signal modulating "winking" amplitude	Measurement of "winking" signal through successive iterations of position of two movable mirrors. The signal amplitude will be determined through direct integration.

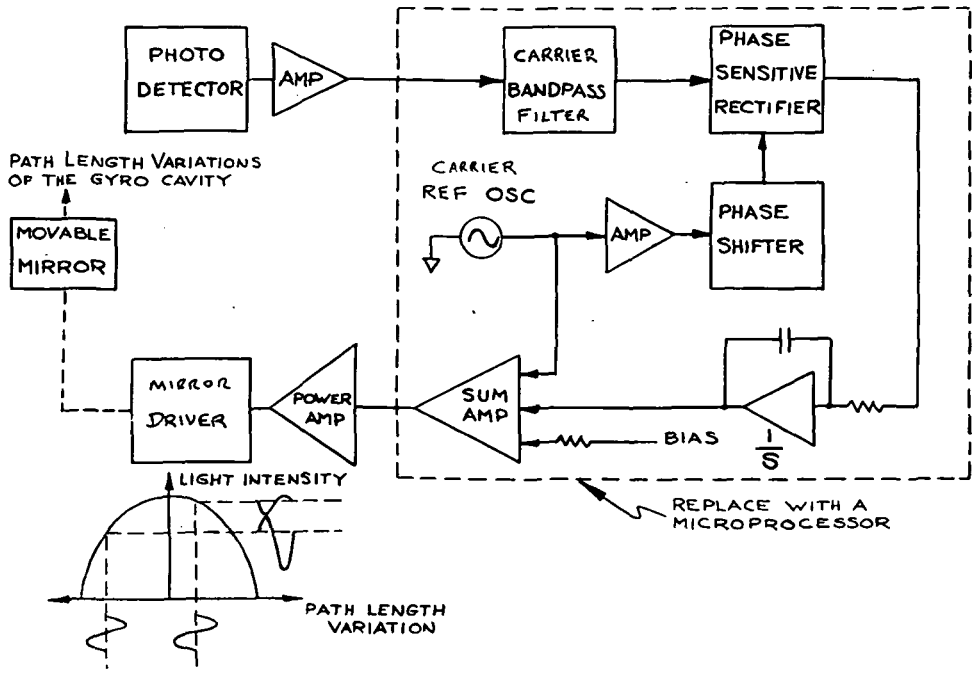
RLG Functional Block Diagram



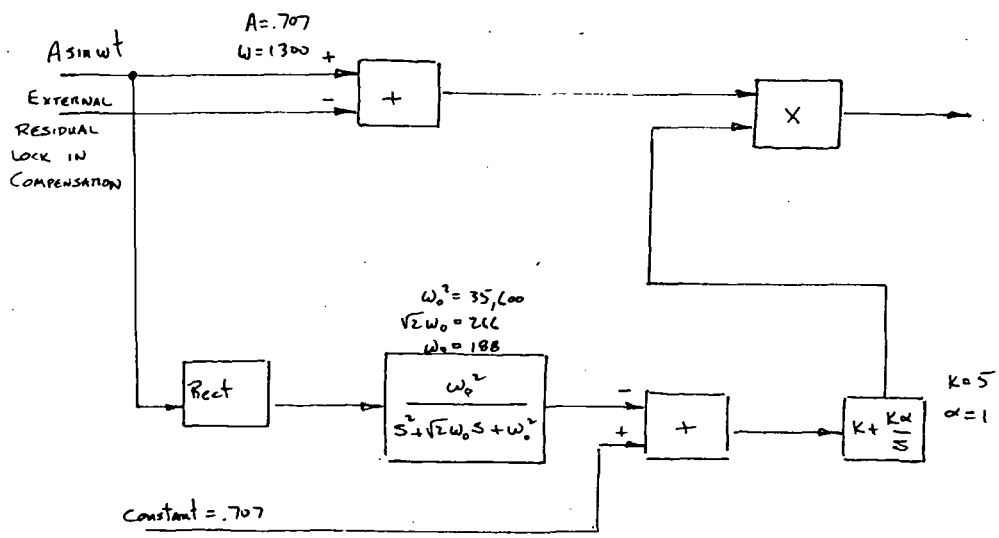
Dither Servo Functional Diagram

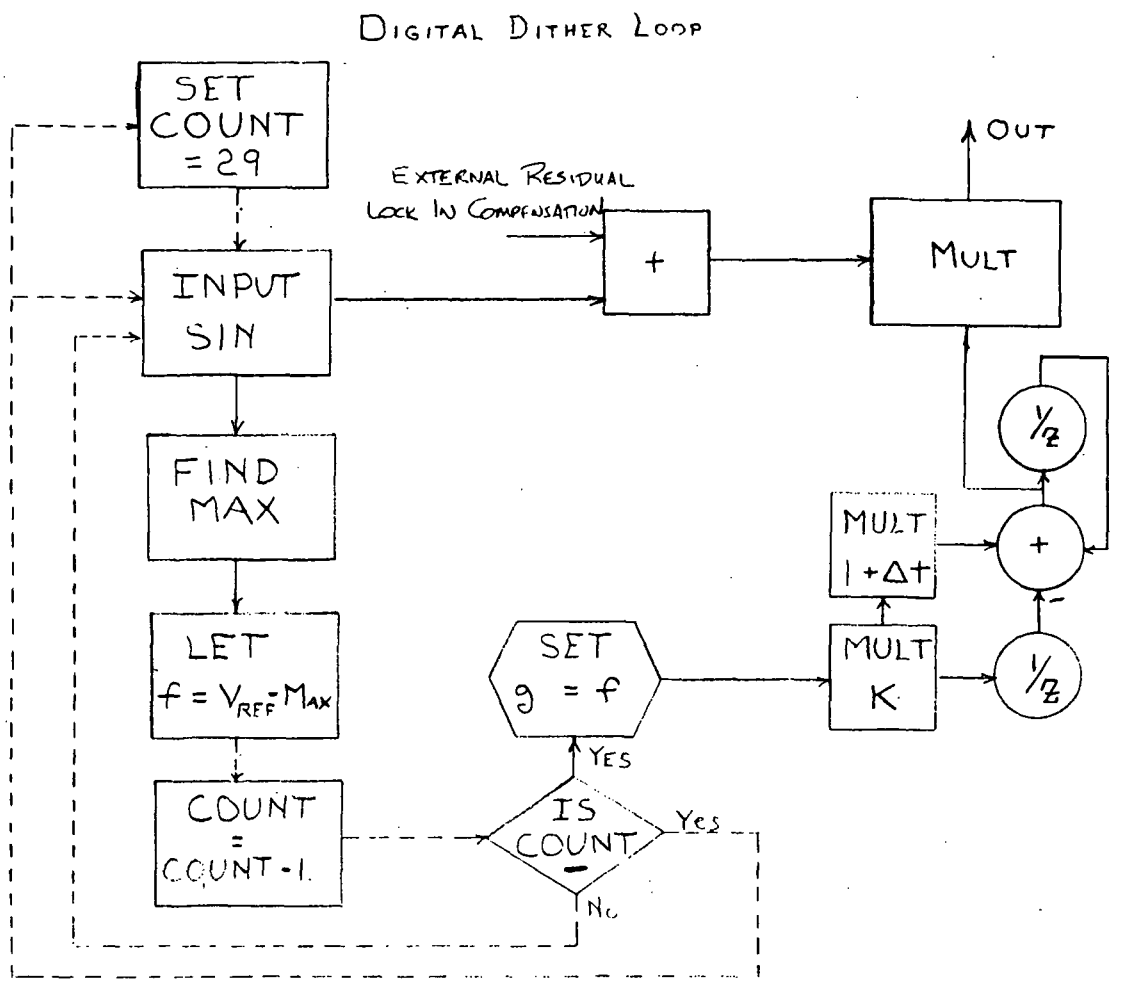


PATH LENGTH CONTROL LOOP
FUNCTIONAL DIAGRAM



Dither Control Loop (μ processor section)





REDUCTION IN SYSTEM COMPLEXITY AND COST SAVINGS

Electronics Circuit Board Area Required for this Function	Analog	Microprocessor (Digital)
	* 250 in^2	50 in^2
Number of parts	1400 discrete parts	10 chips
Cost	$4000	$1000
Reliability (Predicted) failure rate λ = failures per 10^6 hrs	200	30

* With hybridization approximately same board area could be achieved but cost factor then becomes about 8:1.

DEVICES BEING CONSIDERED FOR LOOP IMPLEMENTATION

DEVICE	ON CHIP	A/D ON CHIP	D/A ON CHIP SAMPLE & HOLD ETC.	OTHER EQUIPMENT REQUIRED	SCRATCH PAD MEMORY	PERMANENT MEMORY	LOAD STORE ADDITION TIME	MULTIPLY TIME	I/O INTERFACE	COMMENTS
INTEL 2920 25 BIT MICROCOMPUTER	YES	YES 9 BITS A/D ANALOG INPUT ONLY	YES 9 BIT D/A ANALOG & DIGITAL OUTPUT	±5V POWER SUPPLY CLOCK	RAM 25 BIT DATA WORD BY 40 WORDS	EPROM 24 BIT INSTRUCTION WORD BY 192 WORDS	ALL INSTRUCTIONS 400 nSEC	MUST BE DONE IN SOFTWARE 34 uSEC	4 INPUT CHANNELS 8 OUTPUT CHANNELS	28 PIN DIP NO INTERRUPTS NO JUMPS 192 INSTRUCTIONS/ PROGRAM MAX
INTEL 8022 8 BIT MICRO-COMPUTER 8048 FAMILY	YES	YES 8 BIT A/D ANALOG OR DIGITAL INPUTS ALLOWED	NO	+5V POWER SUPPLY *D/A - SAMPLE & HOLD ETC.	RAM 8 BIT DATA WORD 64 WORDS	ROM 8 BIT ADDRESS WORD 2K WORDS	ALL INSTRUCTIONS 8.5 USEC OR 17 USEC	MUST BE DONE IN SOFTWARE 90 USEC	2 INPUT CHANNELS	40 PIN DIP 2 INTERRUPTS JUMPS ETC
INTEL 8751 8 BIT MICROCOMPUTER	YES	*NO	NO	+5V POWER SUPPLY *A/D & D/A ETC	RAM 8 BIT DATA WORD 128 WORDS	EPROM 8 BIT WORD 4K WORDS	ALL INSTRUCTIONS 1 USEC OR 2 USEC	4 USEC MULT & DIV	4 I/O PORTS	40 PIN DIP 2 LEVEL INTERRUPTS JUMPS ETC
AMI 2400 4 BIT MICROCOMPUTER	YES	YES 8 BIT A/D ANALOG & DIGITAL INPUT	YES 8 BIT D/A ANALOG & DIGITAL OUTPUT	+5V POWER SUPPLY	RAM 4 BIT DATA WORD 128 WORDS 16 BIT WORDS	ROM 8 BIT ADDRESS WORD 4K	ALMOST ALL 53 INSTRUCTIONS 4.5 USEC (4 BIT)	NO MUST BE DOUBLE PRECISION AT LEAST	8 BIT	40 PIN DIP 2 LEVEL INTERRUPTS JUMPS ETC
Z-8000 16 BIT	NO ON BOARD	NO	NO	A/D-D/A BOARD	16 BIT WORDS	16 BIT WORDS	1-2 USEC	12 USEC		
INTEL 8086 16 BIT ISBC 86/12A	NO ON BOARD	NO	NO	*+5V POWER SUPPLY A/D-D/A BOARD	32-64 K OF 16 BIT WORDS	16-32K OF 16 BIT WORDS	1-2 USEC	12 USEC	24 LINES	9-65 INTERRUPT LEVELS

THE SPACELAB EXPERIMENT INTERFACE DEVICE (SEID)

Ron Kallus and Saul Stantent
Intermetrics
Cambridge, Massachusettes

The Spacelab Experiment Interface Device (SEID) has been designed and built to simulate the spacelab CDMS/RAU Interface for spacelab experiments. Its purpose is to provide a low cost method to aid in experiment hardware/software verification and spacelab/experiment interface verification.

Until recently a spacelab experimenter would have to set aside time and resources for the design and construction of a suitable interface simulator. Any incompatibilities discovered during spacelab integration could result in the removal of the experiment from the flight. The consequences of integration failure may substantially increase project cost and create unacceptable project delay.

The SEID simulates the electrical, and logical connections of the Spacelab Remote Acquisition Unit (RAU), the interface functions of the spacelab experiment computer software, and the electrical aspects of the High Rate Multiplexer (HRM).

The SEID meets ESA and NASA electrical, level, timing, drive, and loading requirements for the RAU and HRM interface. Simulated RAU interfaces include PCM serial channels (up to four), User Time Clock (UTC), flexible inputs (up to 128) and descrete outputs (up to 64). Connectors are logically compatible with the RAU.

The SEID accepts commands from any ASCII source, to execute RAU functions which are normally driven from the software resident in the central experiment computer. The commands are entered in a symbolic or a compressed format and can be executed singly or in user defined groups.

The experiment can thus be connected to the SEID, subjected to various sequences of operations and checked for proper system interface characteristics.

The SEID can be instructed to execute sequences of input/output commands to the RAU, similar to those performed during flight. This approximation of the experiment computer software is adequate to detect interface problems and prevent these from occurring during Spacelab--Payload integration.

The SEID hardware is a microprocessor based system, utilizing an 8085 microprocessor with 6K of PROM and 4K of static RAM.

SUMMARY

The Spacelab Experiment Interface Device (SEID) has been designed and built to simulate the spacelab CDMS/RAU Interface for spacelab experiments. Its purpose is to provide a low cost method to aid in experiment hardware/software verification and spacelab/experiment interface verification.

Until recently a spacelab experimenter would have to set aside time and resources for the design and construction of a suitable interface simulator. Any incompatibilities discovered during spacelab integration could result in the removal of the experiment from the flight. The consequences of integration failure may substantially increase project cost and create unacceptable project delay.

The SEID simulates the electrical, and logical connections of the Spacelab Remote Acquisition Unit (RAU), the interface functions of the spacelab experiment computer software, and the electrical aspects of the High Rate Multiplexer (HRM).

The SEID meets ESA and NASA electrical, level, timing, drive, and loading requirements for the RAU and HRM interface. Simulated RAU interfaces include PCM serial channels (up to four), User Time Clock (UTC), flexible inputs (up to 128) and discrete outputs (up to 64). Connectors are logically compatible with the RAU.

The SEID accepts commands from any ASCII source, to execute RAU functions which are normally driven from the software resident in the central experiment computer. The commands are entered in a symbolic or a compressed format and can be executed singly or in user defined groups.

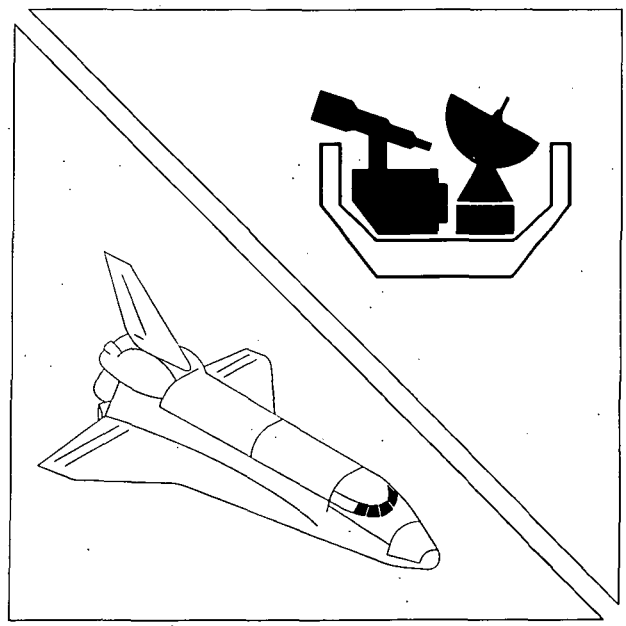
The experiment can thus be connected to the SEID, subjected to various sequences of operations and checked for proper system interface characteristics.

The SEID can be instructed to execute sequences of input/output commands to the RAU, similar to those performed during flight. This approximation of the experiment computer software is adequate to detect interface problems and prevent these from occurring during Spacelab--Payload integration.

The SEID hardware is a microprocessor based system, utilizing an 8085 microprocessor with 6K of PROM and 4K of static RAM. The basic unit has been augmented with an LSI-11 microcomputer to provide disk storage and a more dynamic environment for generation of control data. A simulation of the General Monitor Loop (GML) is provided to emulate spacelab system timing.

SPACELAB
EXPERIMENT
INTERFACE
DEVICE

FOR PAYLOAD DEVELOPMENT



SEID

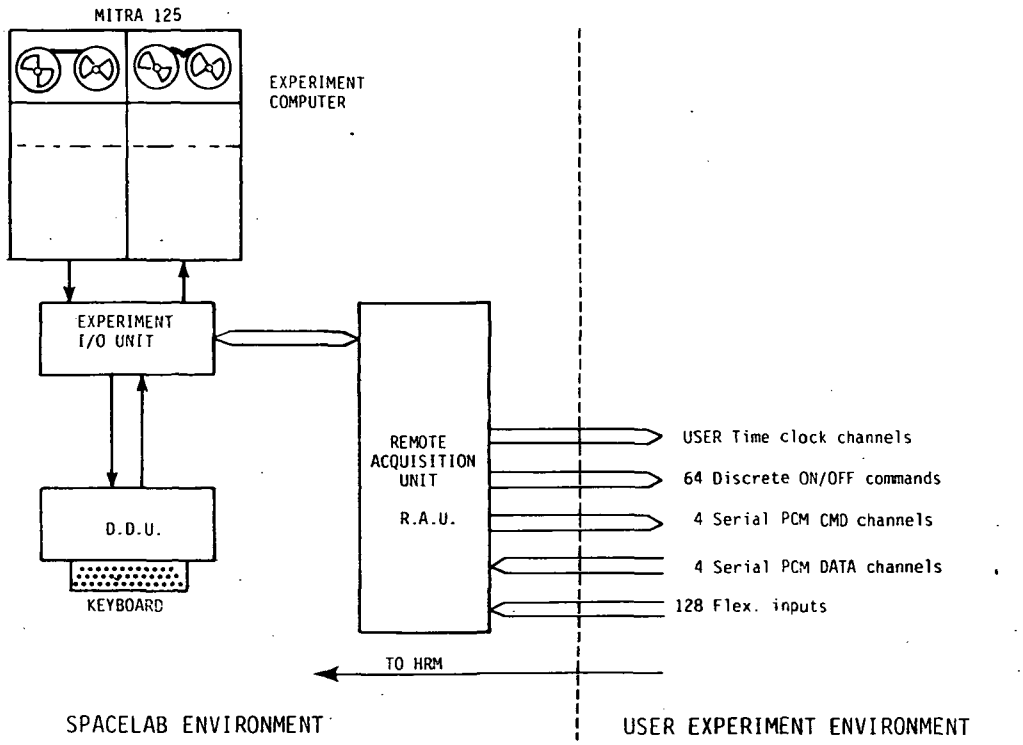
MOTIVATION

- THE PRINCIPAL INVESTIGATOR IS TOTALLY RESPONSIBLE FOR THE CORRECT OPERATION OF HIS EXPERIMENT.
- THE EXPERIMENT MUST HAVE BEEN TESTED PRIOR TO LEVEL IV INTEGRATION.
- HARDWARE/SOFTWARE ERRORS OCCURRING DURING PAYLOAD INTEGRATION MAY RESULT IN REMOVAL OF EXPERIMENT.
- LEVEL IV INTEGRATION FACILITY (OR SIMILAR FACILITY) WILL NOT BE AVAILABLE FOR INDIVIDUAL EXPERIMENT TESTING/VERIFICATION.

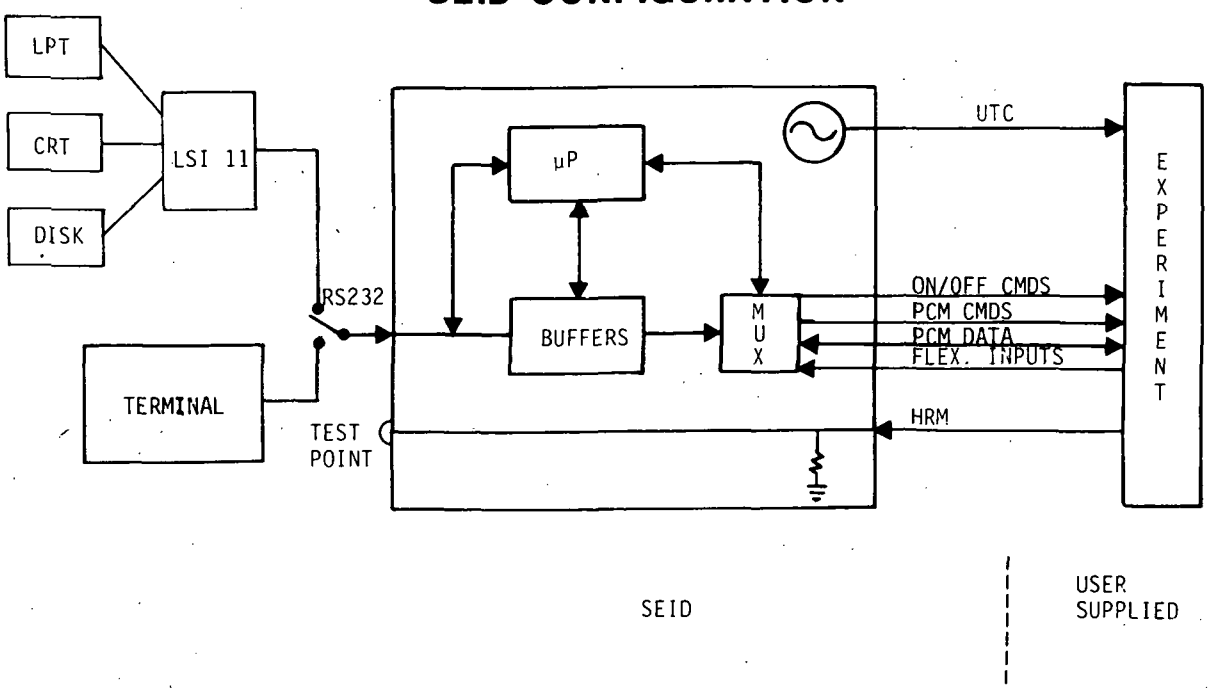
SEID OBJECTIVES

- TESTING OF EXPERIMENT HARDWARE.
- TESTING OF EXPERIMENT PROCESSOR SOFTWARE.
- TESTING OF SPACELAB/EXPERIMENT INTERFACE.
- OFF-LINE TROUBLE-SHOOTING DURING PAYLOAD INTEGRATION.
- INTERFACE CIRCUITS IDENTICAL (OR FUNCTIONALLY IDENTICAL) TO SPACELAB SPECIFICATIONS.
- ALL INTERFACE CONNECTORS FUNCTIONALLY COMPATIBLE TO SPACELAB HARDWARE.
- ABILITY TO APPROXIMATE ECOS/EAS SEQUENCES OF I/O OPERATIONS.

SPACELAB CONFIGURATION



SEID CONFIGURATION



SEID OPERATION MODES

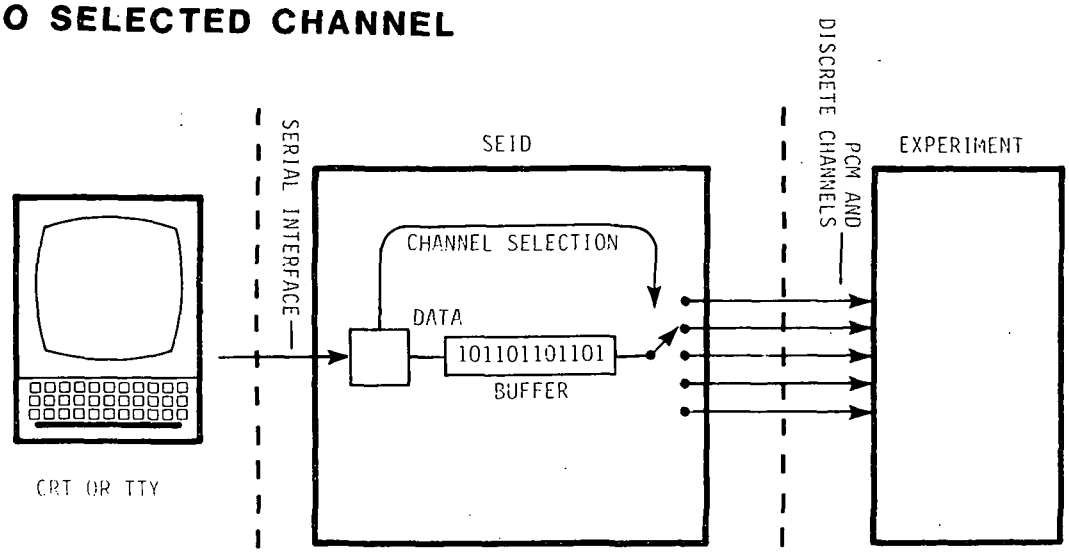
EQUIPMENT INTERFACE TESTING

SEQUENCING

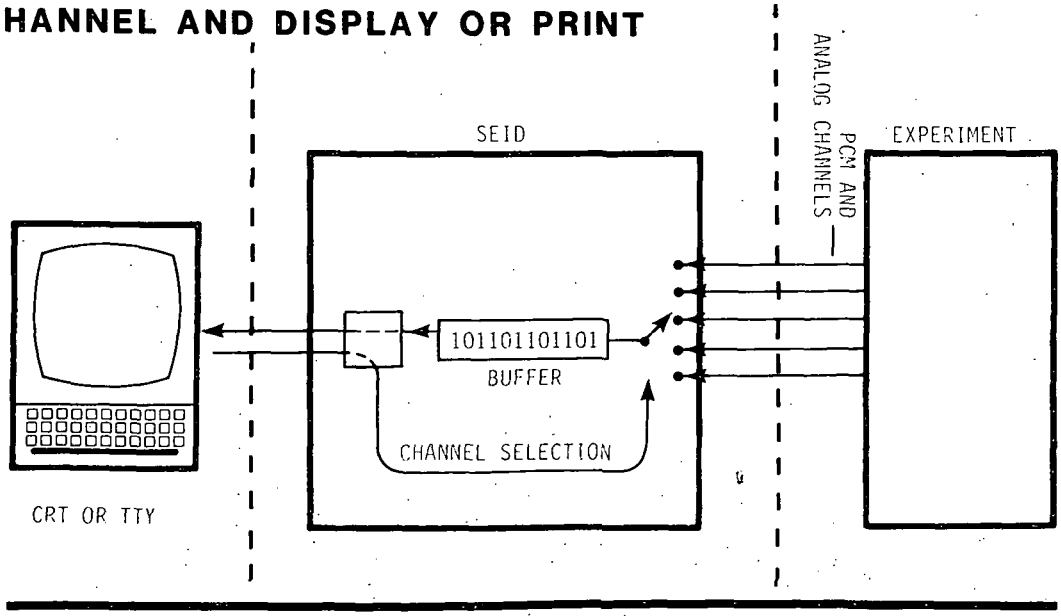
DYNAMIC REAL TIME CONTROL

EQUIPMENT INTERFACE TESTING

STIMULUS: SEND DIGITAL PATTERN TO SELECTED CHANNEL



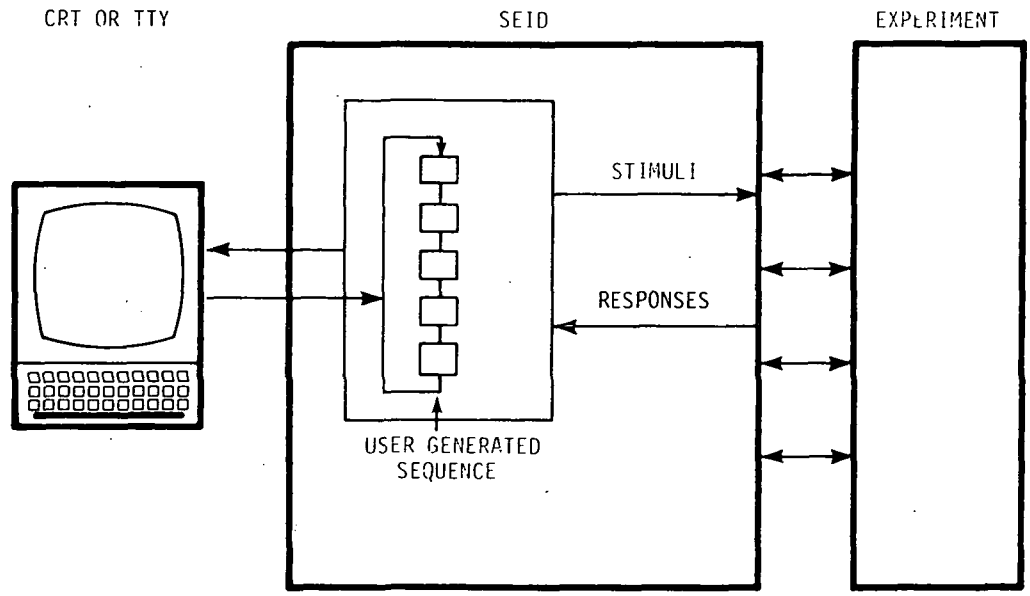
RESPONSE: READ DATA FROM SELECTED CHANNEL AND DISPLAY OR PRINT



Examples of Commands:

```
>ISSUEØ12, ON
>SENSEØ08
>READØ2
>WRITE 0, 2, 1A3F, BØØ3
```

SEQUENCING

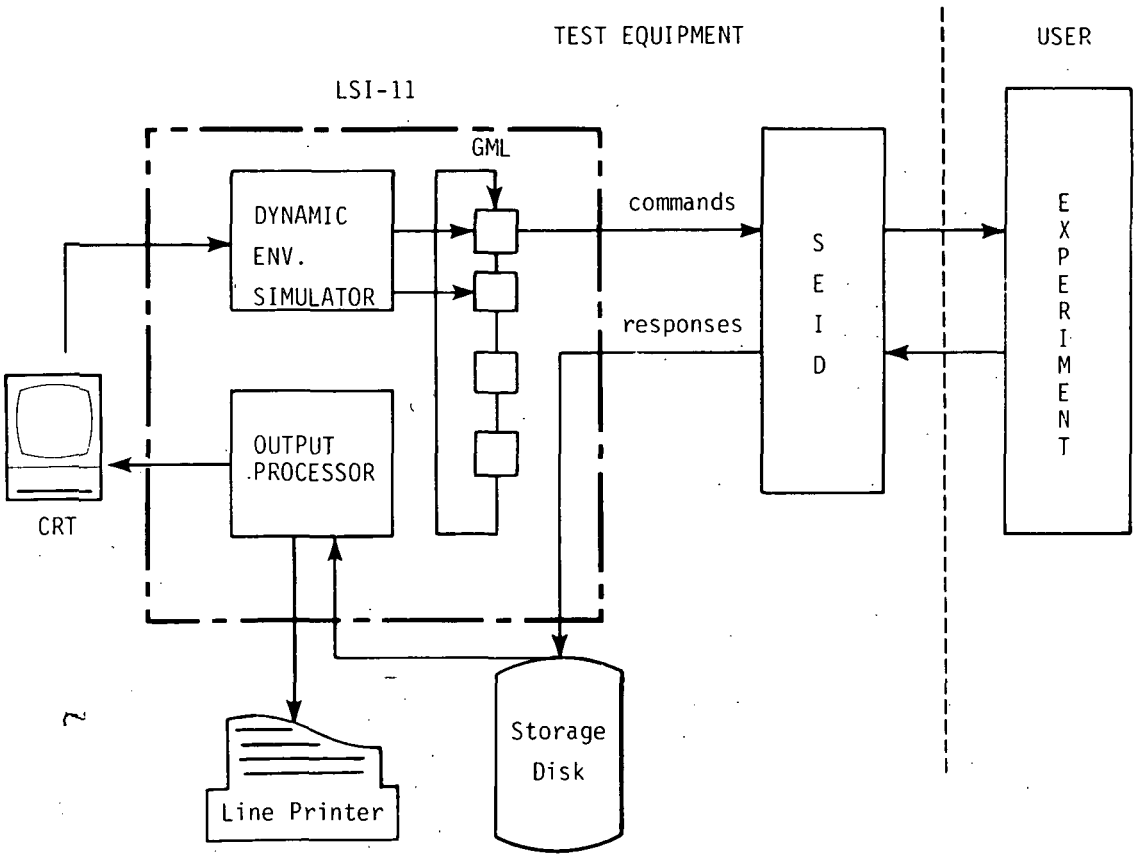


- User may specify sequences via keyboard
- The sequence can form a loop and generate stimuli and buffer responses for CRT or TTY presentation
- No external support software required

Example: ENTERING A SEQUENCE

```
> DEFINE
  seq 10 begun (echoed from SEID)
= ISSUEØ7,ON
= WAIT 3,0
= READØ2
= SENSEØ15
= STARTØ10
= ENDDEF
> STARTØ10
> STOPØ10
```

DYNAMIC REAL-TIME CONTROL



In those situations where even more computer power may be necessary the SEID can be interfaced to a user (or I^2-) supplied computer. This configuration enables the use of a GML simulation.

Example:

```
PRINT    TEST
MONITOR  ON
LOAD     TEST
PERFORM  TEST
```

SEID COMMANDS

RAU

ISSUE NN,ON (OFF)
PULSE NN,ON (OFF)
SENSE NNN
SAMPLE NNN
WRITE N,CC,HHHH,HHHH,H----
DBL-WRITE N,C1,C2,HHHH,HHHH----
READ N
SET-GMT DDD,MMMMMMMM
SET-MET DDD,MMMMMMMM
WRITE N GMT
TIME

SEQUENCE

DEFINE
ENDDEF
START NNNN
STOP NNNN
WAIT SS,MM
TYPE AAA---

DEP

LINK N
DEP-RESP N,CC,HHHHHHHH---
DEP-WRITE SS,MM
POLL-RATE SS,MM
POLL-N

SPSME

SPSME N
SISSUE NN,ON (OFF)
SSENSE NN,NN,---
SSAM-BLK NN,NN,---
SSAMPLE NN

LSI-11 COMMANDS

I/O MONITOR

MONITOR ON (OFF)
ASSIGN II,NN
END
REMOVE II,NN

SEQUENCE MANIPULATION

DEVELOP
END
SAVE NAME
DESPLAY NN,MM

SEQUENCE EDITING

INSERT NN
END
DELETE NN
CHANGE NN

SEQUENCE EXECUTION

LOAD NAME

PERFORM NAME

CONFIGURATION OPTIONS

	OPTIONS	BASIC UNIT	INCREMENT	MAXIMUM
HARDWARE	PCM Command/Data Channels and User Time Clock	-	1	4
	Discrete Outputs	16	16	64
	Flexible Inputs	-	16	128
	HRM Channels	-	1	2
	300 Baud Serial Interface	X	-	X
	110-19.2K Selectable Baud	-	X	X
SOFTWARE	SEID-RAU Software	X	-	X
	SPSME Software	-	X	X
	DEP Software	-	X	X
LSI 11/2	Front-End Controller Hardware	-	X	X
	GML Simulation Software	-	X	X

FUTURE USES OF THE SEID CONCEPT

- SHUTTLE PAYLOAD INTERFACE DEVICE:
  - PAYLOAD MDM
  - PCMMU
  - MTU

- MORE GENERALIZED TEST EQUIPMENT:
  - ANALOG, SERIAL, DISCRETE INTERFACE CIRCUITS AS APPROPRIATE

G-CUEING MICROCONTROLLER
(A MICROPROCESSOR APPLICATION IN SIMULATORS)

Chris G. Horattas
Goodyear Aerospace Corporation
Akron, Ohio

Digital Simulation of aircraft flight requires the iterative solution of a time and event dependent mathematical model. Simulation realism is enhanced by high rate of solution. A simulation whose solution rate produces cues which are perceived to be the same as real-world cues, is considered a real-time simulation. Achieving real time simulation is a prime consideration in simulator design.

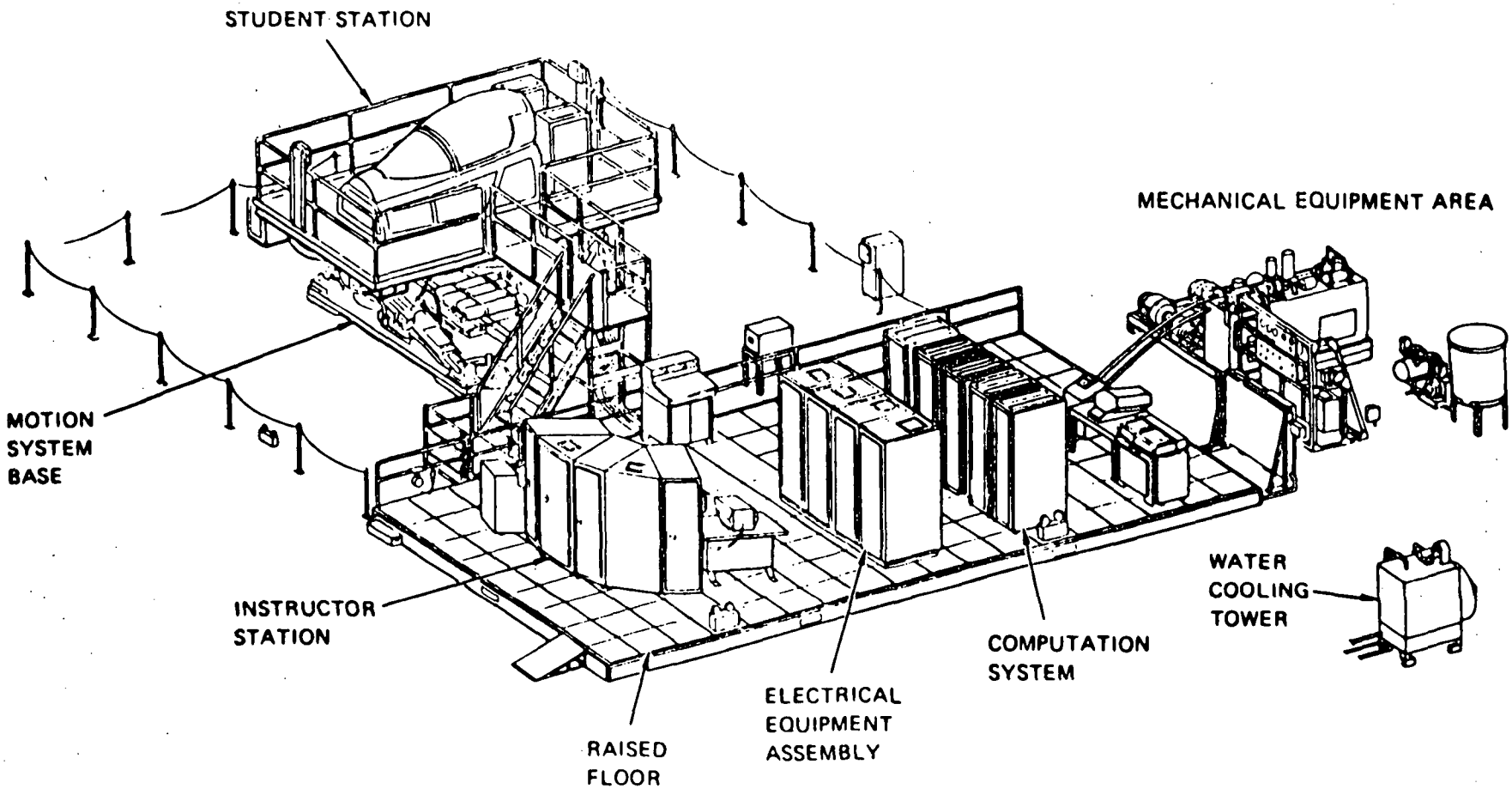
The computation system required to produce real-time simulation is either a single, high cost, extremely high speed processor, or an array of less powerful processors which share the computation task. When multiple processors are used in such array, each is usually dedicated to a simulation subtask and must operate synchronously with the other processors in the array.

One such dedicated processor is the G-Cueing Microcontroller (G-CM). The G-CM consists of a tandem pair of microprocessors, dedicated to the task of simulating pilot sensed cues caused by gravity effects. This task includes execution of a g-cueing model which drives actuators that alter the configuration of the pilot's seat.

The G-Cueing Microcontroller receives acceleration commands from the aerodynamics model in the main computer and creates the stimuli that produce physical acceleration effects of the aircraft seat on the pilots anatomy. One of the two microprocessors is a fixed instruction processor that performs all control and interface functions. The other, a specially designed bipolar bit-slice microprocessor with on-board hardware multiply and firmware implemented divide square root and sine functions, is a microprogrammable processor dedicated to all arithmetic operations. The two processors communicate with each other by a shared memory.

The G-Cueing Microcontroller contains its own dedicated I/O conversion modules (analog-to-digital, and digital-to-analog) for interface with the seat actuators and controls, and a DMA controller for interfacing with the simulation computer.

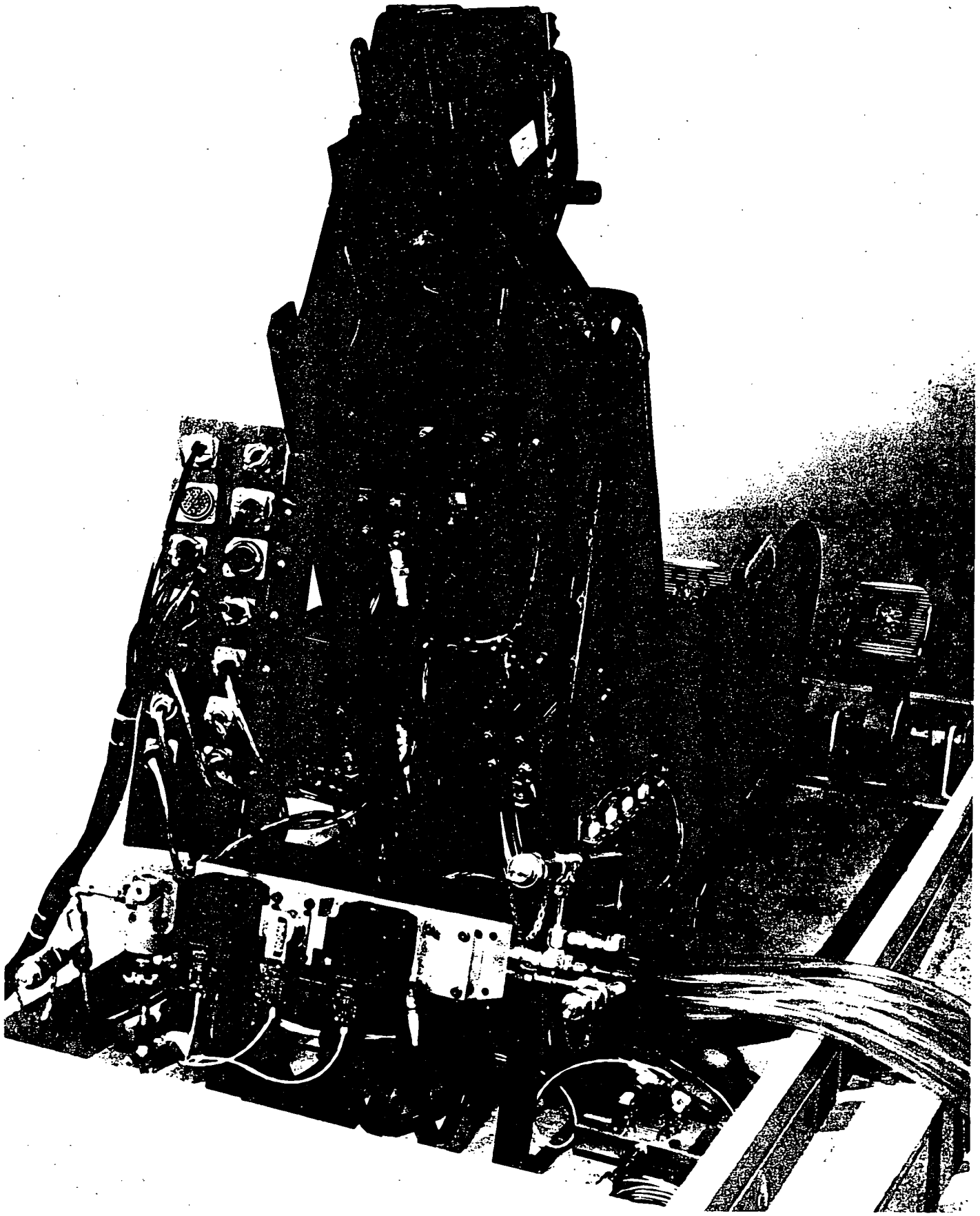
Even though the microcontroller is programmed to perform the g-cueing model, it is not limited to this specific application. Any application which can be microcoded within the available memory, the available real time and the available I/O channels, could be implemented in the same controller. Furthermore, the microcontroller capacity can be expanded by the addition of memory and I/O modules.

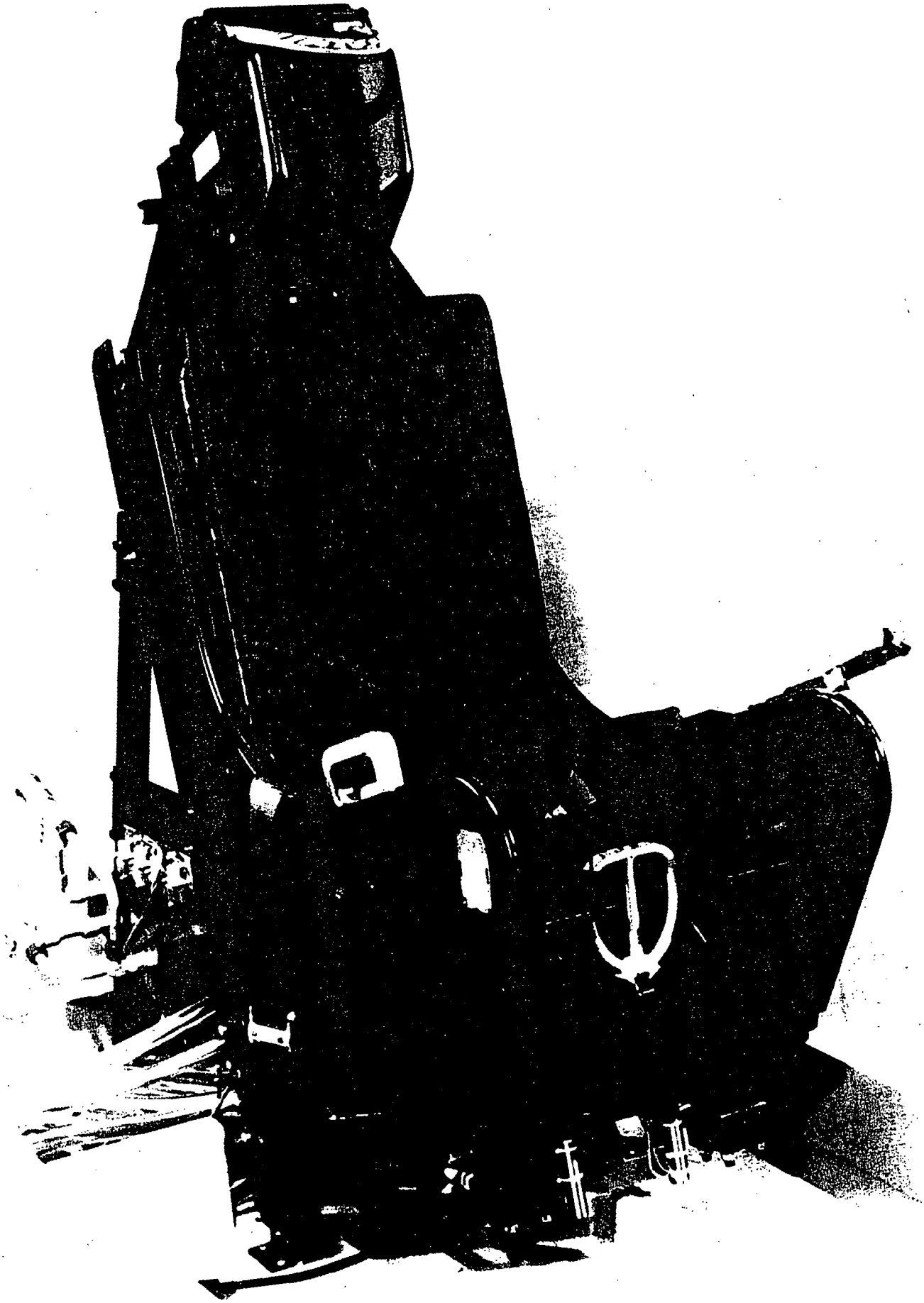


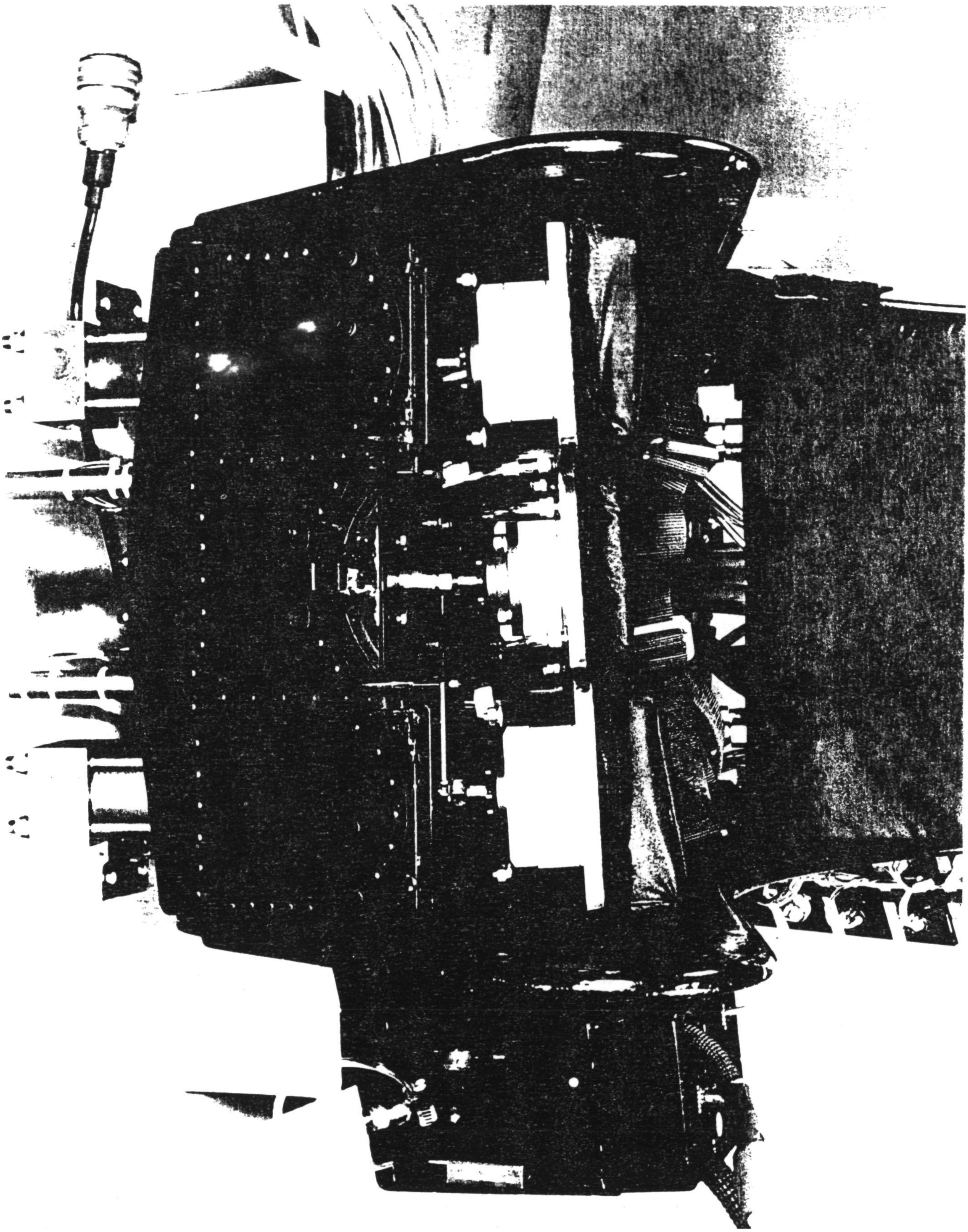
ISOMETRIC VIEW

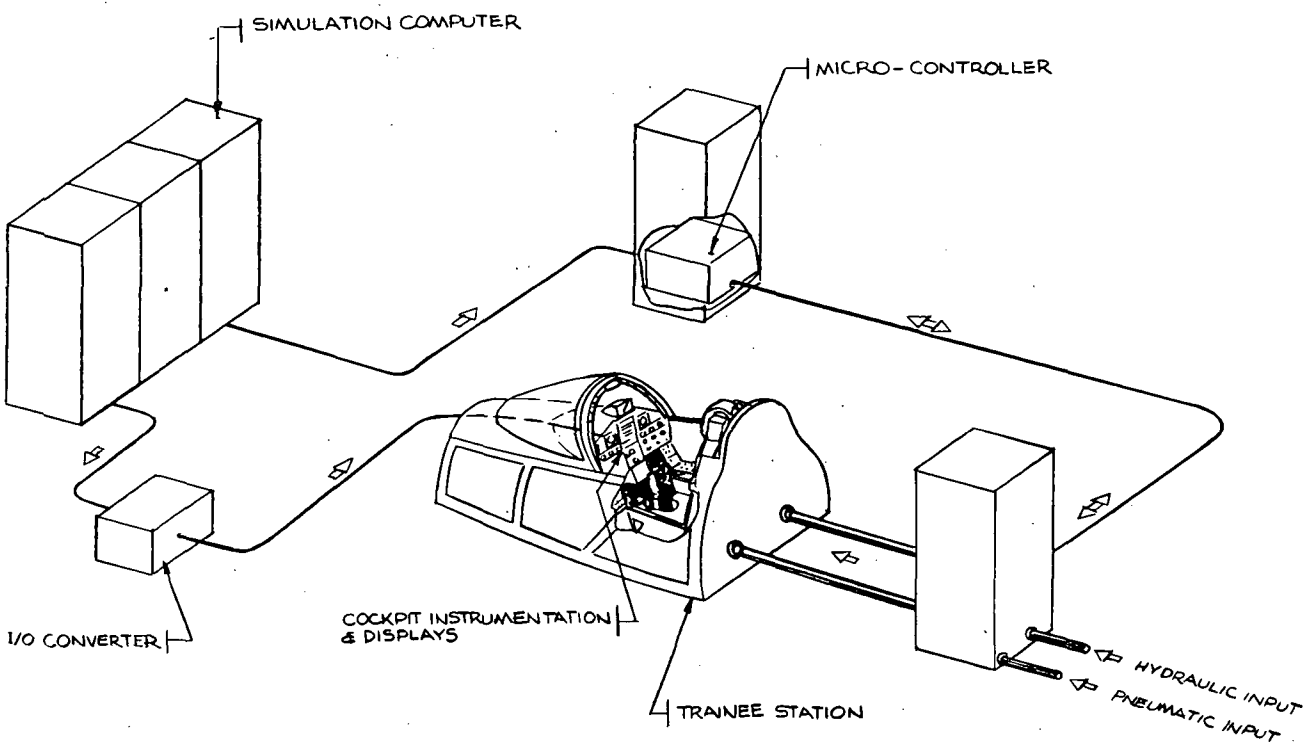
F-15

MISSION IN
PROGRESS

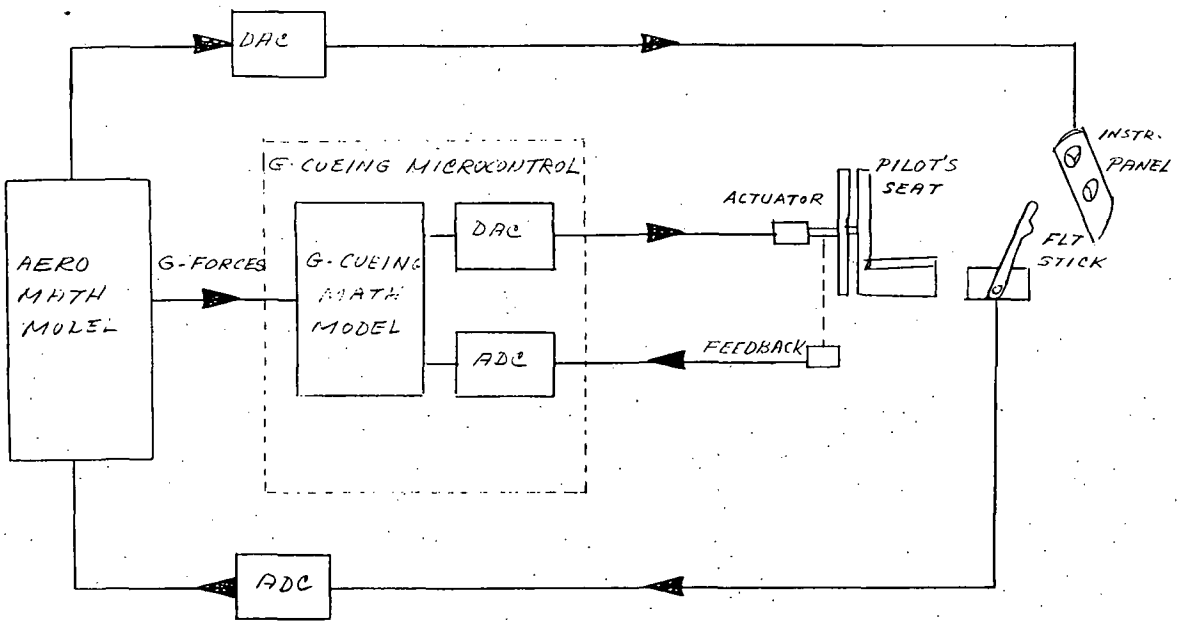




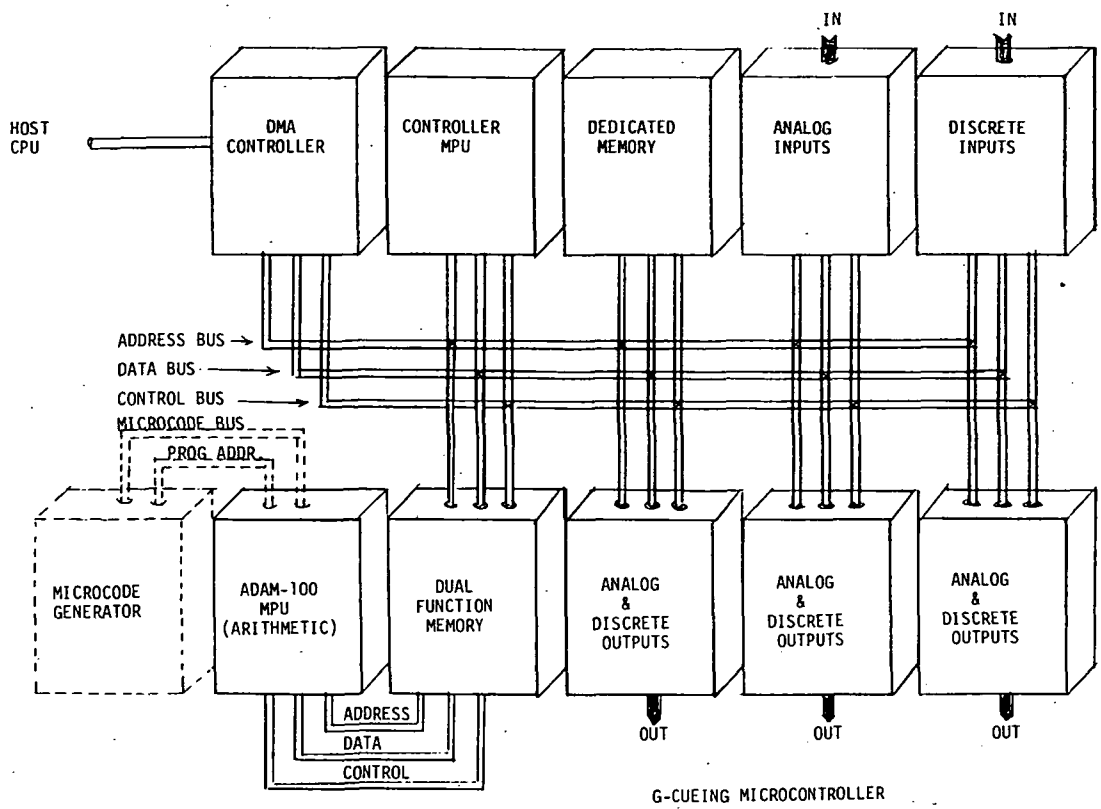




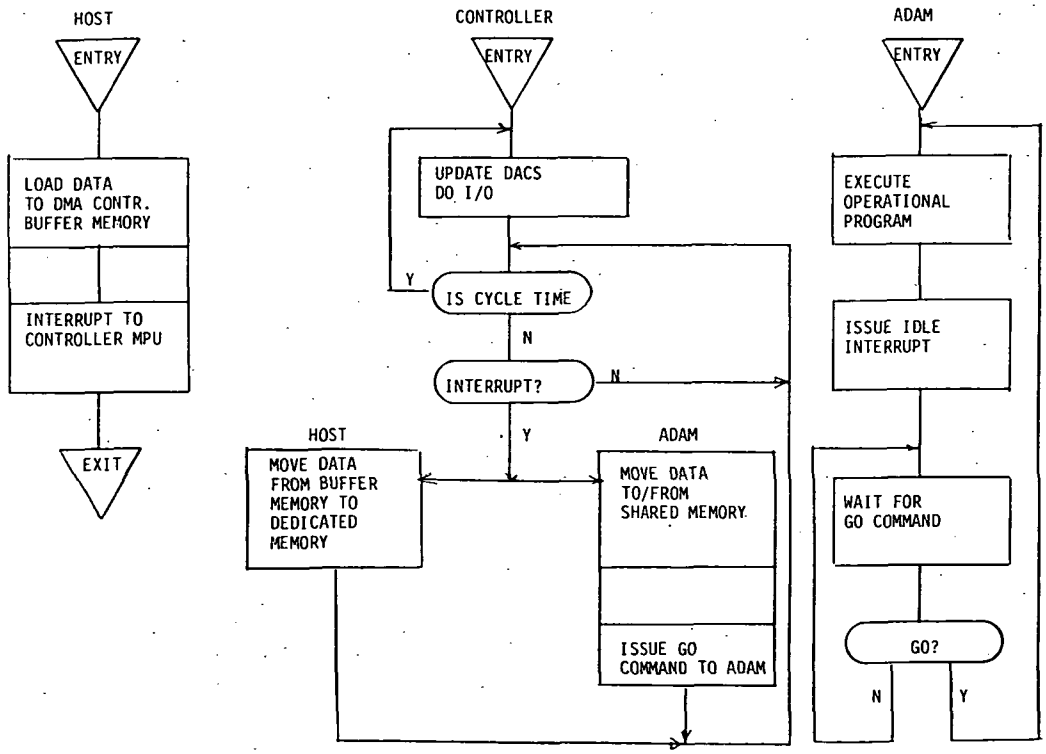
G-CUEING SIMPLIFIED DIAGRAM



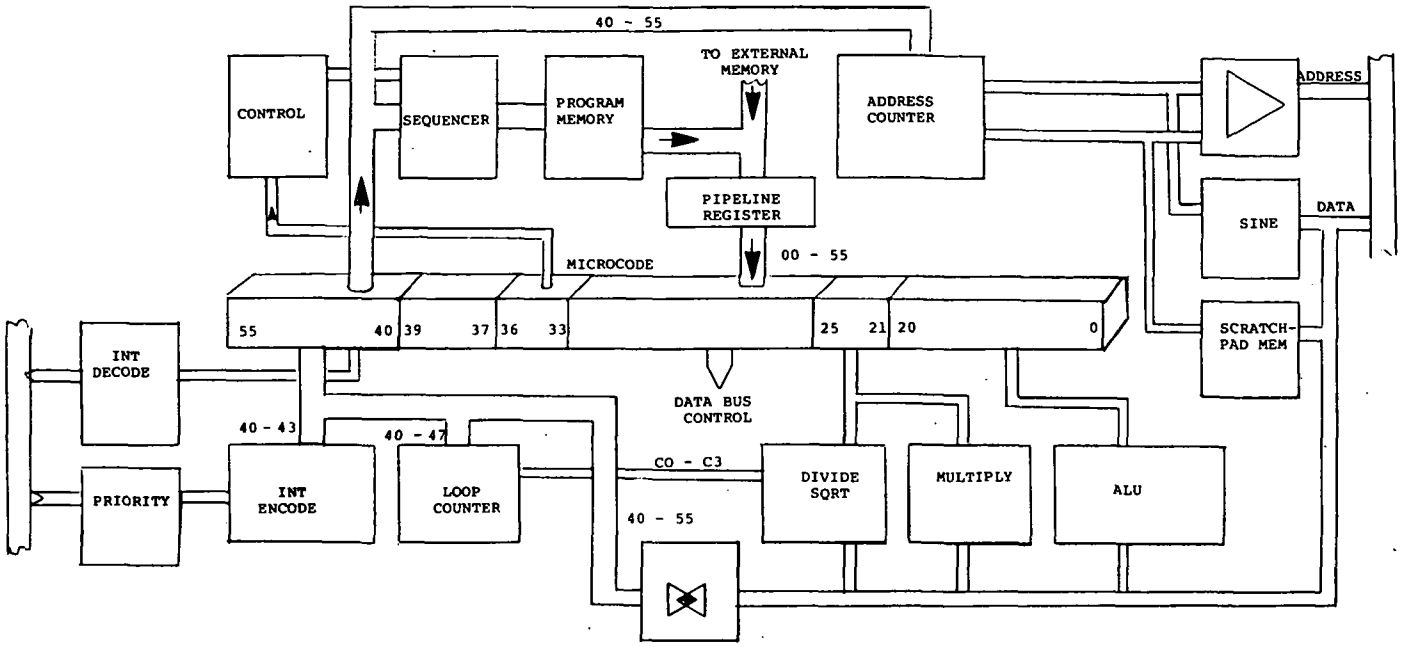
G-CUEING DIAGRAM



G-CUEING MICROCONTROLLER



SYNCHRONIZATION-G-CUEING MICROCONTROLLER



ADAM - FUNCTIONAL BLOCK DIAGRAM

(INSTRUCTION WORD FORMAT)

55 40	37	33	30	26	21	19	17	13	9	6	3	0
BRANCH ADDRESS	TEST	BRANCH CTL	MEMORY CTL	DATA BUS ENABLES	DIV/SQRT MPY/SINE	ALU CARRY	ALU SHIFT	R_B REG ADR	R_A REG ADR	ALU DEST	ALU FUNCT.	ALU SOURCE
1	2	3	4	5	6	7	8	9	10	11	12	13

FIELD	FUNCTION	BITS	
1	BRANCH ADDRESS	55-40	
2	CONDITION CODE TEST	39-37	
3	BRANCH CONTROL	36-33	
4	MEMORY CONTROL	32-30	
5	DATA BUS ENABLES	29-26	
6	DIV/SQRT/MPY/SIN	25-21	
7	ALU CARRY SELECT	20-19	ALU
8	ALU SHIFT SELECT	18-17	
9	R_B SELECT	16-13	
10	R_A SELECT	12-9	
11	ALU DESTINATION	8-6	
12	ALU FUNCTION	5-3	
13	ALU SOURCE	2-0	

1. [ ] INDICATE OPTIONAL ARGUMENT IN INSTRUCTION
2. NOTE THAT ALL CONTROL INSTRUCTIONS WILL BE ENCODED WITH FIELD 4 SET TO 111, FIELD 11 SET TO 001, AND FIELD 12 SET TO 100 ANY OF WHICH MAY BE OVERRIDDEN DURING A MERGE
3. REMAINDER OF FIELDS DEFAULT TO $\emptyset$ AND MAY BE OVERRIDEN

MICROCONTROLLER MEMORY MAP
ON CONTROLLER MPU DEBUG, MONITOR
EPROM
MEMORY EXPANSION (PROGRAM)
EPROM + RAM
MICROPROGRAM GENERATION (RAM)
RAM
DUAL FUNCTION MEMORY (RAM)
RAM
I/O MAP
REGISTERS
TOTAL MEMORY MAP
0
4K
8K
12K
16K
20K
24K
28K
32K

MICROPROCESSOR SOFTWARE APPLICATIONS FOR FLIGHT TRAINING SIMULATORS

Wayne P. Leavy
Goodyear Aerospace Corporation
Akron, Ohio

Microcomputer distributed processing may be the answer to modifications and improvements for overloaded computer systems of training simulators. Top down functional design is a very useful tool for software implementation, and the same concept can be applied to a multiple processor computer system. The grouping of many independent functions into one large, software module leads to confusion, overhead, and inefficiency; the same is true when many large mainframe computers are grouped into one system.

G-cueing is one of the many functions of a pilot training simulator system and a unique candidate for the application of distributed computation techniques. The G-cueing system must respond to the aircraft six-degree-of-freedom (DOF) equations of motion to provide static and dynamic stimuli to the student pilot's proprioceptive, tactile, and visual (with respect to eyepoint) sensors. The cues provided by the system must include onset cues as well as sustained acceleration cues.

A G-cueing system has been developed as a strap-on attachment to the F-15 operational flight training simulator, utilizing a microprocessor computation system. Hydraulic actuators produce the onset cues while pressure control of AIRMAT pneumatic cushion cells produces seat hardness, softness, and contouring to provide for sustained cues. An active lap belt and G-suit is also provided to reproduce those sensations experienced in the actual aircraft.

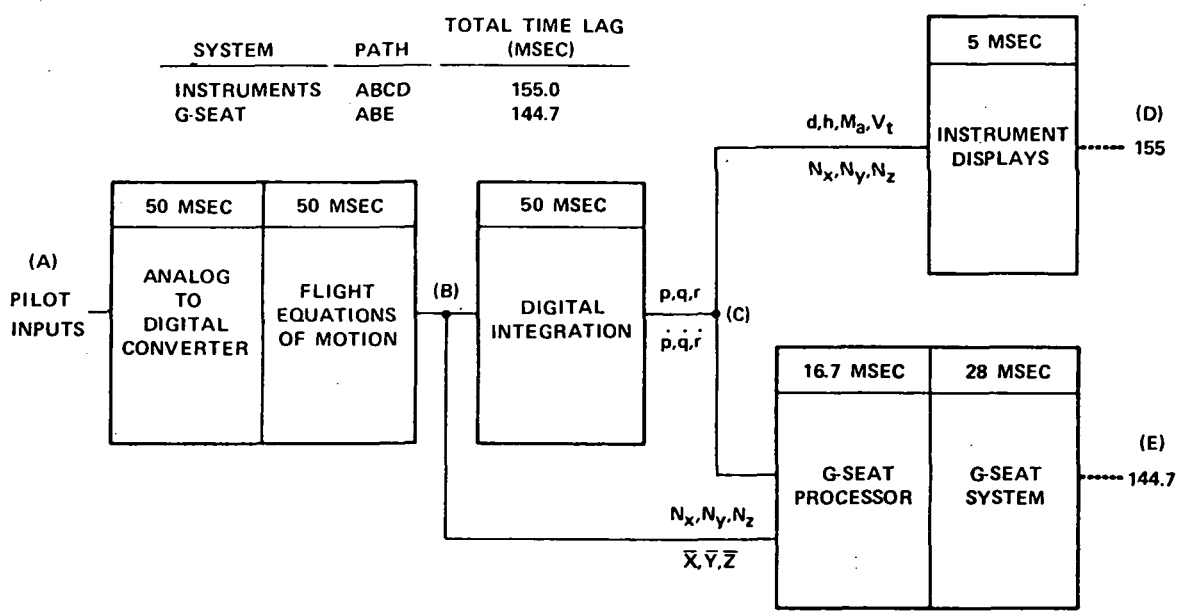
The interface between the G-cueing system software and the software of other simulator systems is very simple: minimal amount of data exchange. However, in itself the G-cueing system is a complex system which requires a significant amount of computer power. This paper presents the G-cueing system software design ann implementation in the dual microprocessor system of the F-15 operational flight training simulator G-cueing system. The software is structured in the two microcomputers such that one serves as a controller performing all logical functions and interface with the host computer system while the other serves as an arithmetic unit performing all mathematical functions.

WHY DISTRIBUTED COMPUTATION ?

- OFFLOAD THE HOST COMPUTER SYSTEM
- REAL-TIME CONSIDERATIONS
- COST ADVANTAGES
- MODULAR HARDWARE (PLUG-IN UNIT)
- INDEPENDENT DEVELOPMENT (SCHEDULE ADVANTAGES)

SYSTEM TIME LAG DIAGRAM

SYSTEM	PATH	TOTAL TIME LAG (MSEC)
INSTRUMENTS	ABCD	155.0
G-SEAT	ABE	144.7



G-CUING A SELF CONTAINED SYSTEM

HOST PROVIDES:

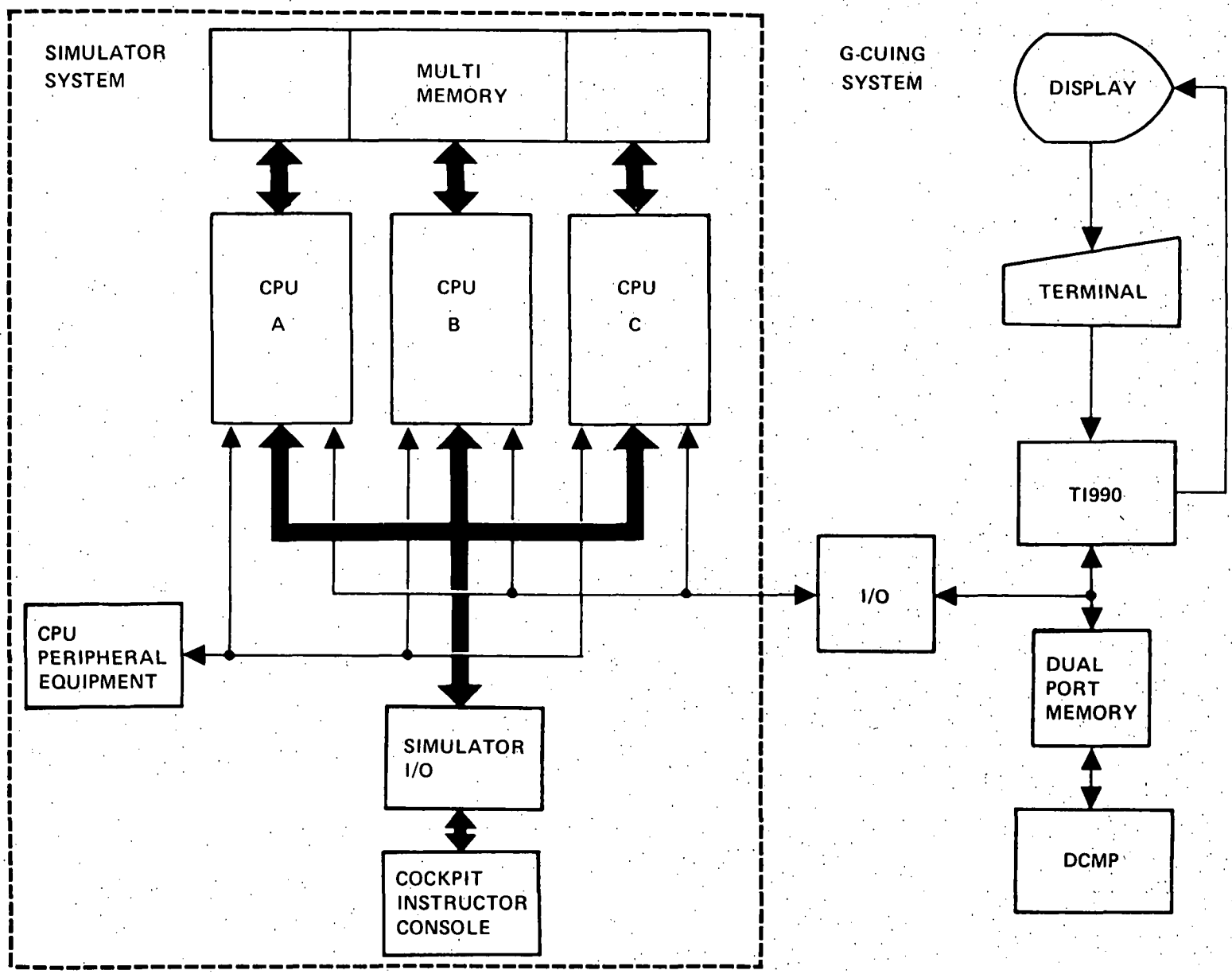
- FRAME SYNCHRONIZATION
- AERODYNAMIC STIMULI
- TRAINER MODE

G-CUING A SELF CONTAINED SYSTEM (CONTD)

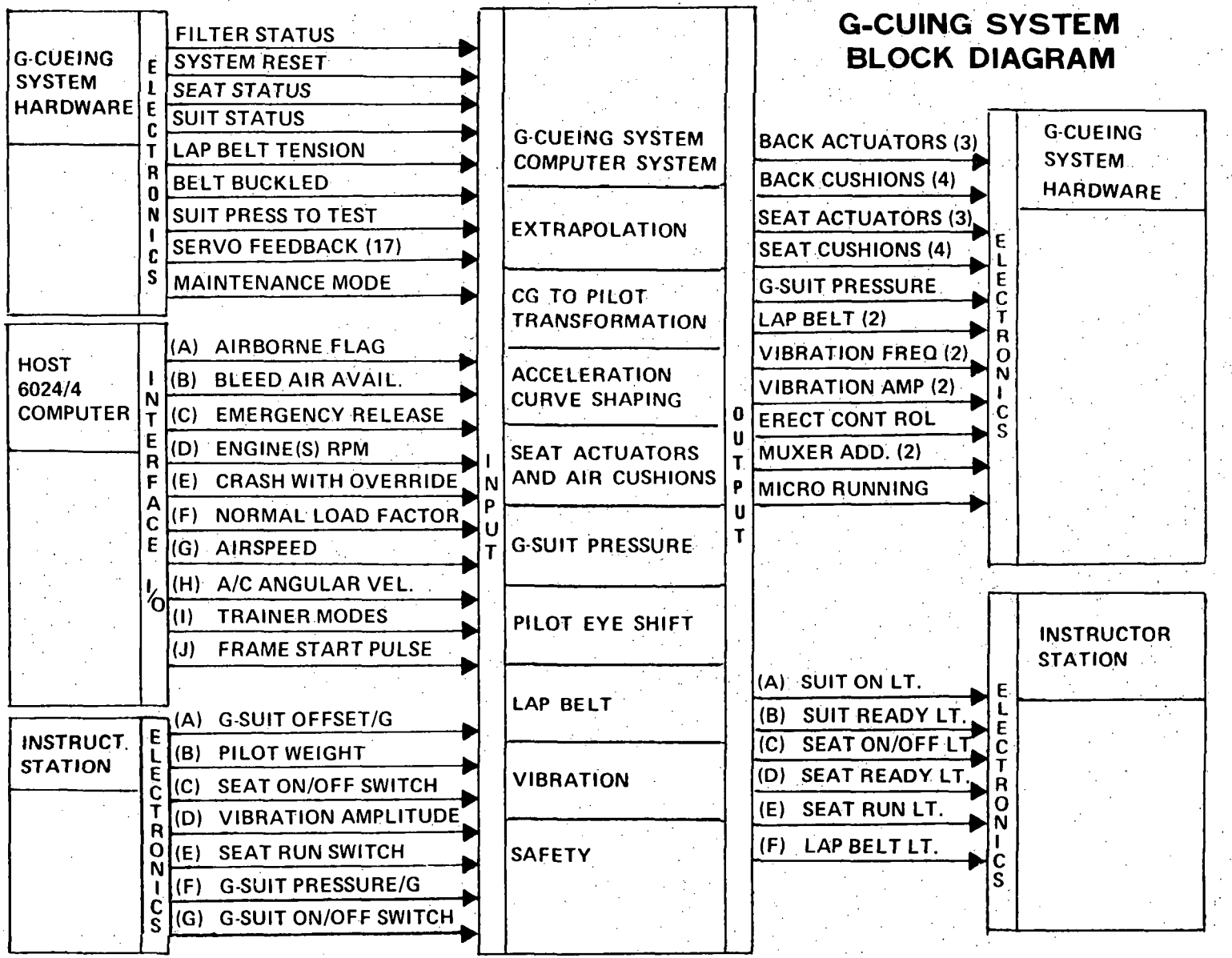
G-CUING SYSTEM PROVIDES:

- ONSET CUES
- SUSTAINED CUES
- BUFFET/VIBRATION CUES
- G-SUIT CUES
- SAFETY MONITORING
- CUE SYNCHRONIZATION
- SELF TEST

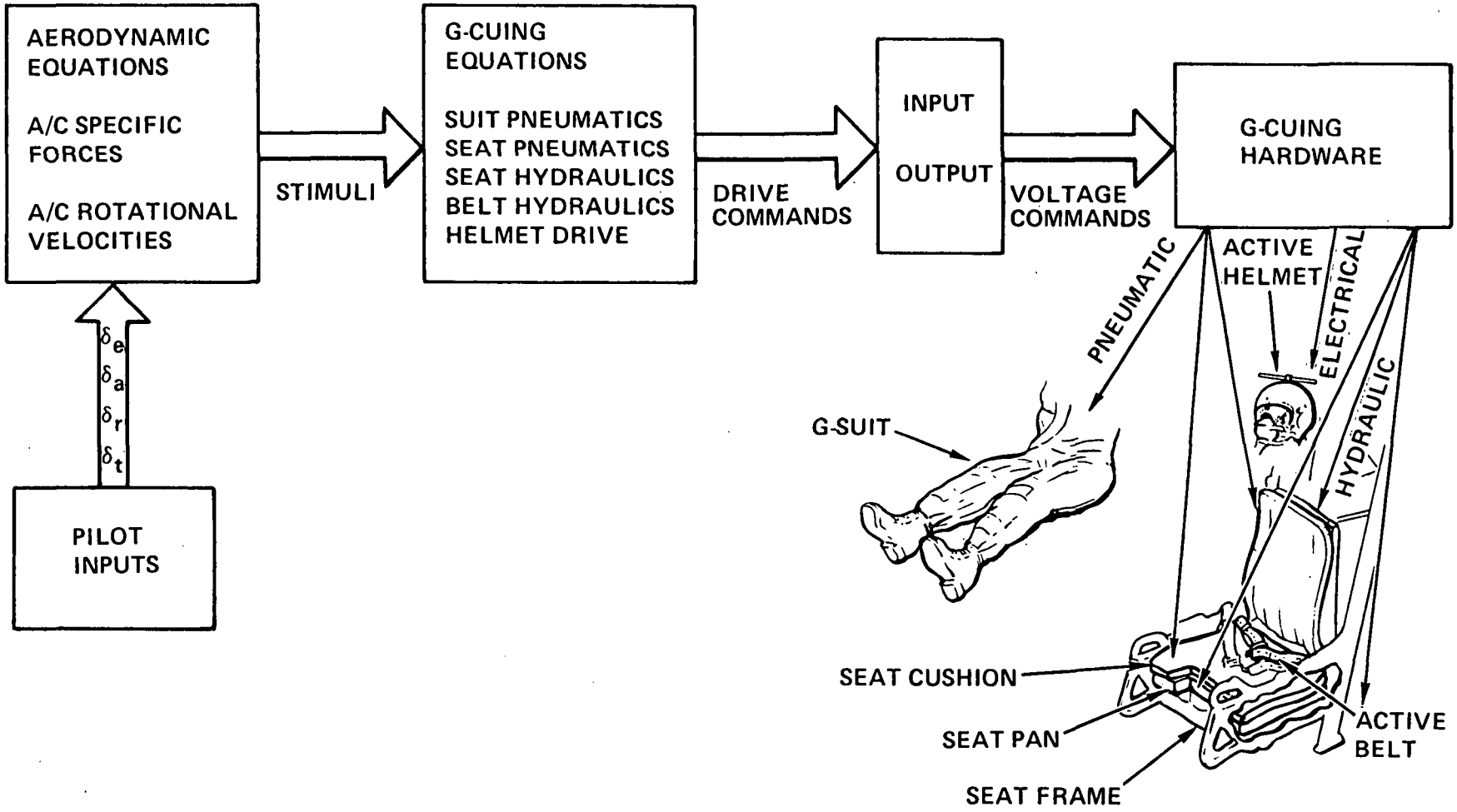
COMPUTATIONAL SYSTEM



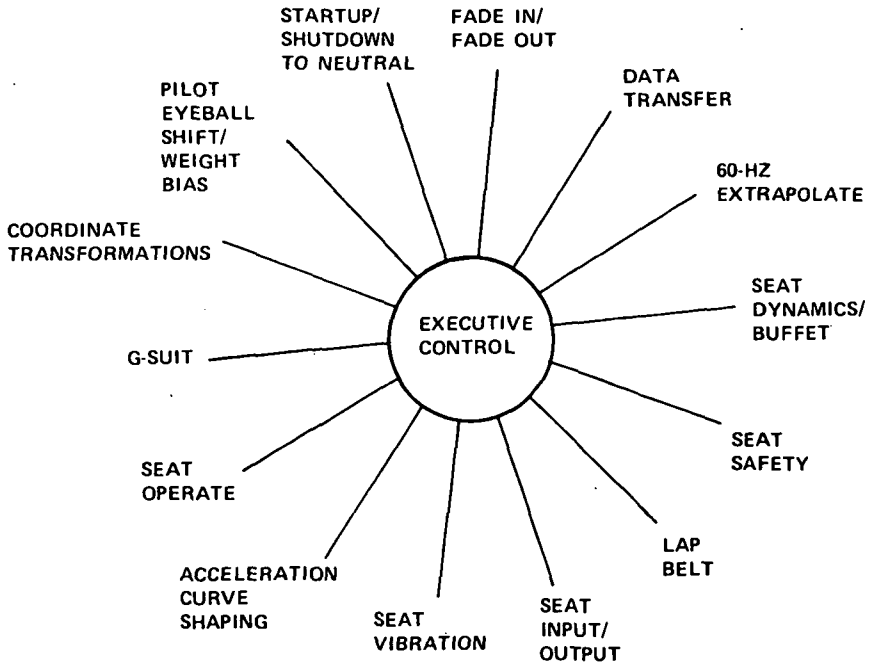
G-CUING SYSTEM BLOCK DIAGRAM



G-CUING SYSTEM DIAGRAM



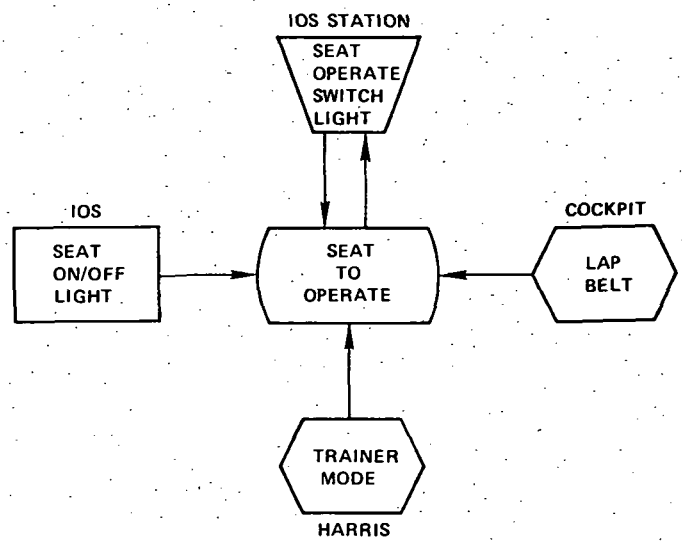
FUNCTIONAL DESCRIPTION OF SOFTWARE



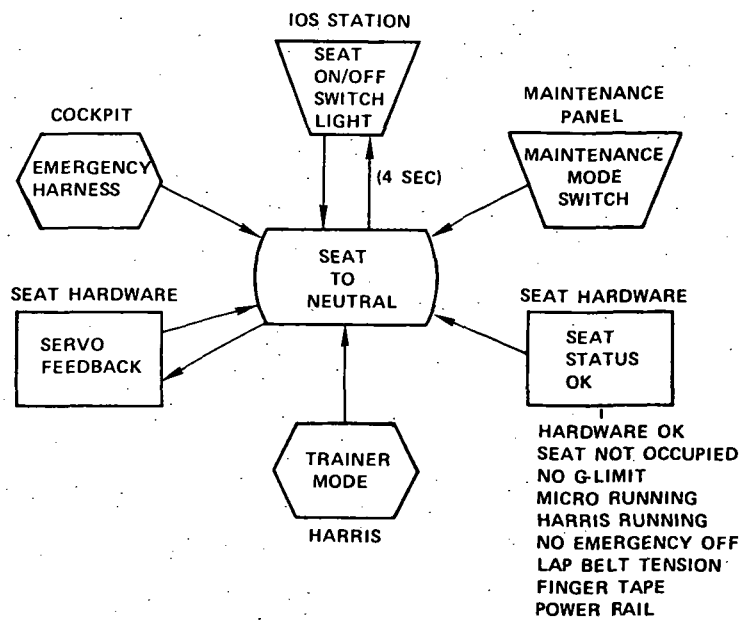
G-CUING SYSTEM
SOFTWARE APPORTIONMENT

T I 9900 PROCESSOR	SCRATCH PAD MEMORY	D C M P PROCESSOR
8 K ROM MEMORY	4 K RAM MEMORY	2 K ROM MEMORY
1) SELF TEST	1) TEMPORARY VARIABLES	1) 60 HZ EXTRAPOLATOR
1) EXECUTIVE	2) DMCP MODULE ADDRESSES	2) SEAT DYNAMICS
2) FADE IN/OUT	3) VARIABLE GAIN ADDRESSES	3) LAP BELT
3) DATA TRANSFER		4) ACCELERATION CURVE SHAPING
4) SEAT SAFETY		5) G-SEAT
5) SEAT I/O		6) COORDINATE TRANSFORMATION
6) SEAT OPERATE		7) PILOT WEIGHT BIAS
7) START UP/SHUT DOWN		8) SEAT VIBRATION
4 K RAM MEMORY		
SCRATCH PAD		
3 K MEMORY USED	1 K MEMORY USED	0.9 K MEMORY USED

FUNCTIONAL DESCRIPTION OF G-SEAT TO OPERATE



FUNCTIONAL DESCRIPTION OF G-SEAT STARTUP/SHUTDOWN



SEAT EQUATIONS OF MOTION

- PILOT COORDINATES

$$\ddot{X}_P = N_X + PQY_{CG} + P\gamma Z_{CG} - Q^2X_{CG} - \gamma^2X_{CG} + \dot{Q}Z_{CG} - \dot{\gamma}Y_{CG}$$

$$\ddot{Y}_P = N_Y + QPX_{CG} + Q\gamma Z_{CG} - P^2Y_{CG} - \gamma^2Y_{CG} + \dot{\gamma}X_{CG} - \dot{P}Z_{CG}$$

$$\ddot{Z}_P = N_Z + \gamma PX_{CG} + \gamma QY_{CG} - P^2Z_{CG} - Q^2Z_{CG} - \dot{P}Y_{CG} - \dot{Q}X_{CG}$$

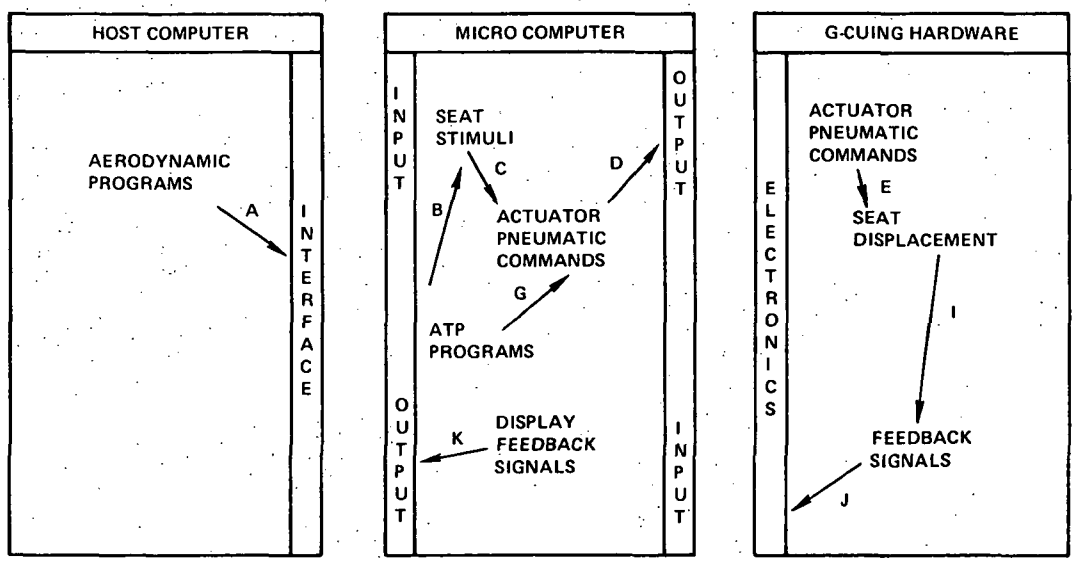
- SEAT ANGLE EFFECTS

$$\ddot{X}_{SP} = \ddot{X}_P \text{ COS } a - Z_P \text{ SIN } a$$

- SEAT ELEMENT COORDINATES

$$ELE_{C,A} = S\left(-W_1\ddot{X}_{SP} - W_2\ddot{Z}_{SP} - W_3\ddot{Y}_{SP}\right) - PWB$$

AUTOMATED ATP OPERATION



OPERATIONAL TRAINING MODE – ABCDE

INTEGRATED ATP MODE – BCDE

HARDWARE ATP MODE -- GDE

ATP SIGNAL RETURN PATH – IJK

Page Intentionally Left Blank

AN EXPERIMENTAL DISTRIBUTED MICROPROCESSOR IMPLEMENTATION WITH A SHARED MEMORY COMMUNICATIONS AND CONTROL MEDIUM

Richard S. Mejzak
Naval Air Development Center
Warminster, Pennsylvania

An experimental distributed microprocessor subsystem is currently under development at the Naval Air Development Center as a vehicle to investigate distributed processing concepts with respect to replacing larger computers with networks of microprocessors at the subsystem or node level. Major benefits being exploited include increased performance, flexibility, system availability, and survivability by use of multiple processing elements with reduced cost, size, weight and power consumption.

This paper concentrates on defining the distributed processing concept in terms of control primitives, variables, and structures and their use in performing a decomposed DFT (Discrete Fourier Transform) application function. The DFT was chosen as an experimental application to investigate distributed processing concepts because of its highly regular and decomposable structure for concurrent execution. The design assumes interprocessor communications to be anonymous. In this scheme, all processors can access an entire common database by employing control primitives. Access to selected areas within the common database is random, enforced by a hardware lock, and determined by task and subtask pointers. This enables the number of processors to be varied in the configuration without any modifications to the control structure. Decompositional elements of the DFT application function in terms of tasks and subtasks are also described.

The experimental hardware configuration consists of IMSAI 8080 chassis which are independent, 8-bit microcomputer units. These chassis are linked together to form a multiple processing system by means of a shared memory facility. This facility consists of hardware which provides a bus structure to enable up to six microcomputers to be interconnected. It provides polling and arbitration logic so that only one processor has access to shared memory at any one time. For discussion purposes, five of the processors are designated as slaves and one as a master where each slave contains an identical copy of a control executive and application program tasks. In actual operation, the slave processors cooperate to compute the DFT where the master provides external input, output, and control functions. With this implementation, commands to perform a DFT iteration are provided through the master.

It is expected that this concept will be tested and demonstrated on a laboratory model by the end of 1980. Evaluations will concentrate on areas such as performance comparisons based on varying the number of processors and bus contention factors as a function of local processing and common data base access times. Future work will focus on fault tolerant techniques that can be directly implemented and evaluated on the baseline laboratory model.

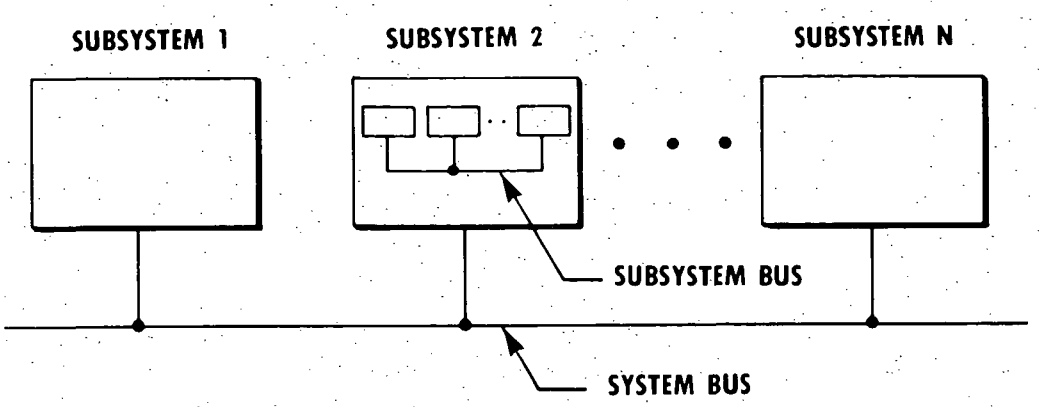
MOTIVATION

- AVIONIC PROCESSING SYSTEMS ARE BECOMING MORE DISTRIBUTED IN ORDER TO EXPLOIT THE FOLLOWING MAJOR BENEFITS:
  - INCREASED SYSTEM-WIDE REAL TIME PERFORMANCE
  - EASE OF ADAPTABILITY TO INTEGRATION AND CHANGE
  - HIGH SYSTEM AVAILABILITY
  - DECREASED SYSTEM VULNERABILITY
- BECAUSE OF REDUCED SIZE, WEIGHT, POWER CONSUMPTION AND COST ADVANTAGES, MICROPROCESSOR TECHNOLOGY WILL IMPACT AVIONIC PROCESSING SYSTEMS IN THE FOLLOWING AREAS:
  - INTERFACE AND HARDWIRED LOGIC REPLACEMENT APPLICATIONS
  - ➡ REPLACING LARGER COMPUTERS WITH NETWORKS OF SMALLER COMPUTERS

MICROPROCESSOR TECHNOLOGY AND DISTRIBUTED PROCESSING

- REASONABLE COST-PERMITS EXPERIMENTING WITH CONCEPTS WHICH WOULD OTHERWISE BE PAPER STUDIES
- REDUCED SIZE, POWER, AND WEIGHT PERMITS APPLICATIONS THAT WOULD OTHERWISE NOT BE FEASIBLE
- LIFE CYCLE COSTS OFTEN MUCH LOWER THAN FORMER SOLUTIONS TO SAME PROBLEM

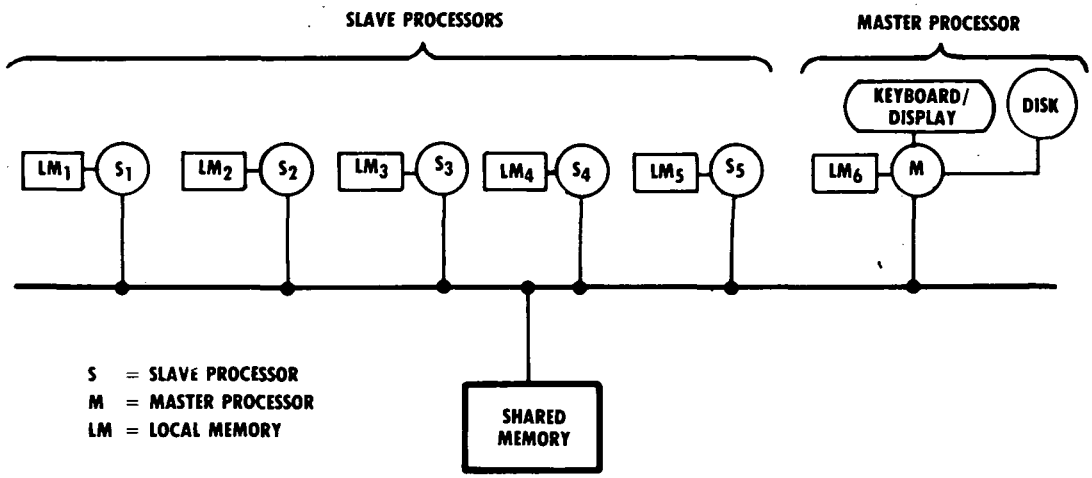
GLOBAL/LOCAL DISTRIBUTION



APPROACH

- EXPERIMENTAL INVESTIGATION
- LABORATORY MODEL
- OFF-THE-SHELF HARDWARE (MICROPROCESSORS ARE INEXPENSIVE)
  - MULTIPLE PROCESSORS
  - SHARED MEMORY FACILITY INTERCONNECT
- EXPERIMENTAL CONTROL STRUCTURE
  - LOCAL KNOWLEDGE OF EXISTANCE OF OTHER PROCESSORS NOT REQUIRED
  - GLOBAL CONTROL AND TASK SCHEDULING VIA HIGHLY RELIABLE SHARED MEMORY
- EXPERIMENTAL WELL-KNOWN APPLICATION-DFT
- DEMONSTRATE CONCEPT FEASIBILITY
- PERFORM TRADE-OFF ANALYSES
- IDENTIFY AND IMPLEMENT FAULT-TOLERANT CONCEPTS

EXPERIMENTAL HARDWARE CONFIGURATION



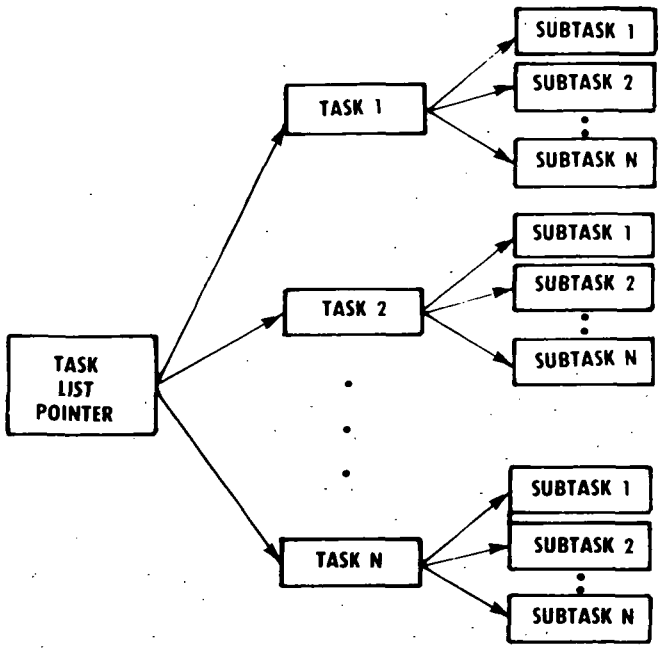
SHARED MEMORY FACILITY CONSTRAINTS

- SIX PROCESSORS MAXIMUM
- ROUND-ROBIN POLLING SCHEME
- ONE BYTE ACCESSED PER POLL
- FIXED LOCK-OUT TIME IN FLAG BLOCK

ASSUMPTIONS

- MASTER PROCESSOR PERFORMS INTERFACE AND DISPLAY FUNCTIONS
- SLAVE PROCESSORS PERFORM APPLICATION FUNCTION CONCURRENTLY AS DIRECTED BY MASTER PROCESSOR
- LOCAL MEMORY
  - EACH SLAVE PROCESSOR CONTAINS IDENTICAL COPY OF PROGRAMS
  - CONTROL EXECUTIVE
  - APPLICATION TASKS
- SHARED MEMORY
  - COMMON TO ALL PROCESSORS
  - CONTROL VARIABLES
  - APPLICATION DATA
  - ACCESSED BY CONTROL PRIMITIVES
  - ACCESS RIGHTS ENFORCED BY SEMAPHORES
- VARYING NUMBER OF PROCESSORS DOES NOT AFFECT CONTROL STRUCTURE

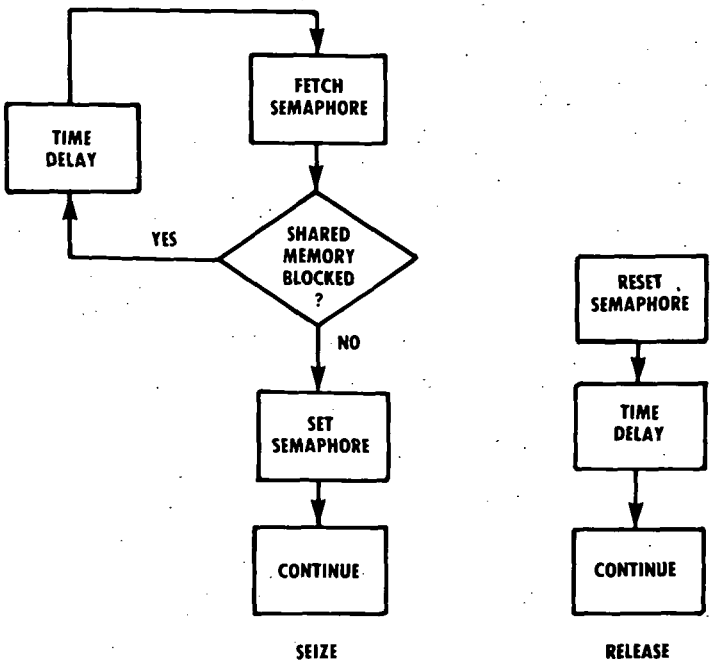
TASK STRUCTURE



SYSTEM CONTROL: SEMAPHORES

- ENFORCES ACCESS RIGHTS TO SHARED MEMORY
- USED TO INDICATE CONDITIONS
  - SHARED MEMORY BLOCKED
  - SHARED MEMORY AVAILABLE
  - ITERATION IN PROGRESS
  - ITERATION COMPLETED

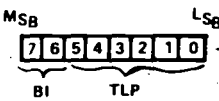
CONTROL PRIMITIVES



CONTROL VARIABLES

BI/TLPt

FORMAT:



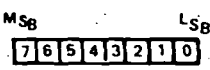
– BI IS A 2-BIT SEMAPHORE AND INDICATES THE FOLLOWING CONDITIONS:

SEMAPHORE B	SEMAPHORE I	CONDITION
0	0	SHARED MEMORY BLOCKED AND ITERATION COMPLETED
0	1	SHARED MEMORY BLOCKED AND ITERATION IN PROGRESS
1	0	SHARED MEMORY AVAILABLE AND ITERATION COMPLETED
1	1	SHARED MEMORY AVAILABLE AND ITERATION IN PROGRESS

– TLP IS A 6-BIT TASK LIST POINTER THAT CAN POINT TO ANY ONE OF 64 TASKS

• TSt

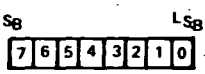
– FORMAT:



–TSt IS AN 8-BIT WORD USED TO ASSOCIATE CORRESPONDING DATA WITH A TASK AND CAN TAKE ON 256 VALUES

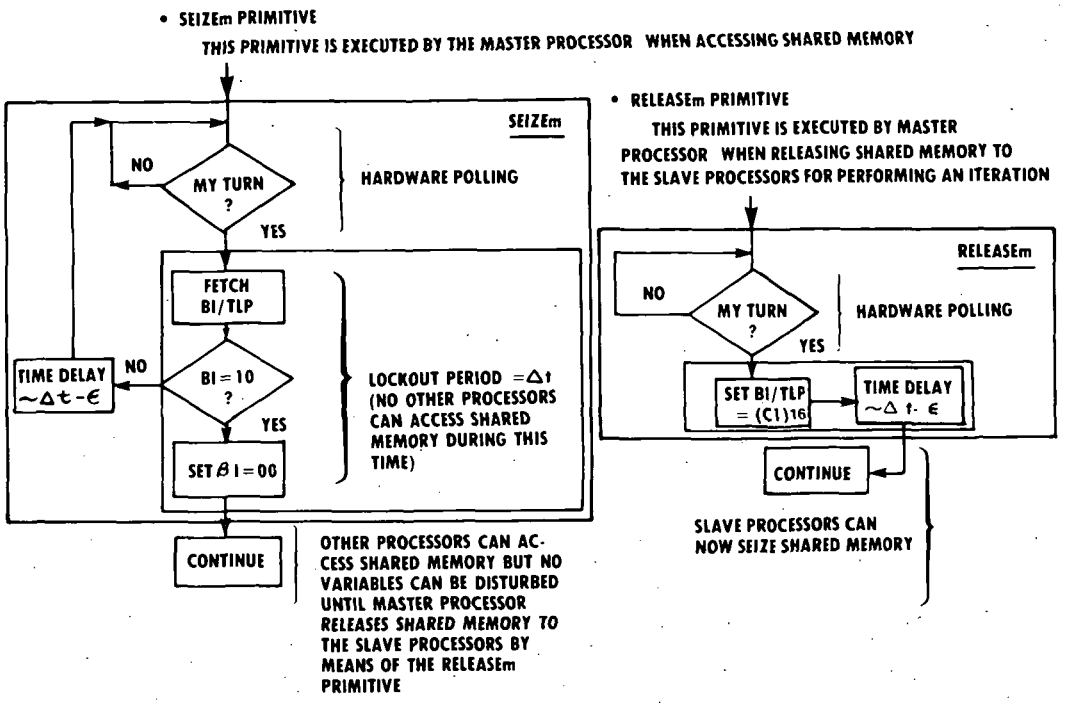
• CTCt

– FORMAT:

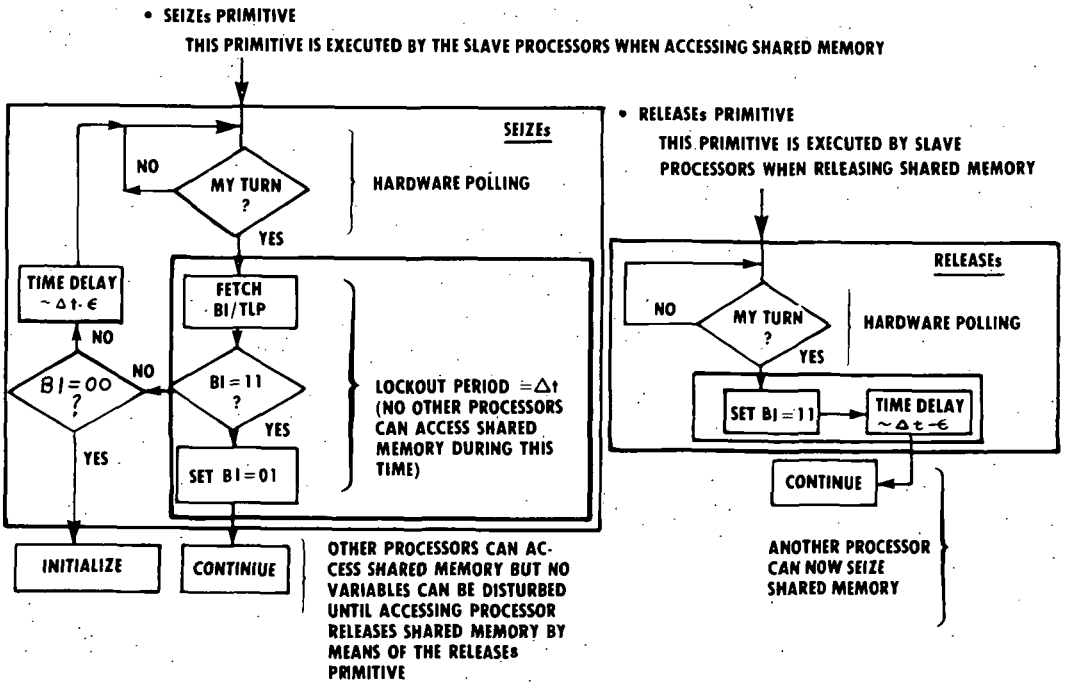


– CTC IS AN 8-BIT CUMULATIVE TASK COUNTER. ONE CTC IS REQUIRED FOR EACH TYPE OF TASK BEING PERFORMED, i. e., THE NUMBER OF CTC s ARE EQUAL TO THE NUMBER OF TASKS POINTED TO BY TLP.

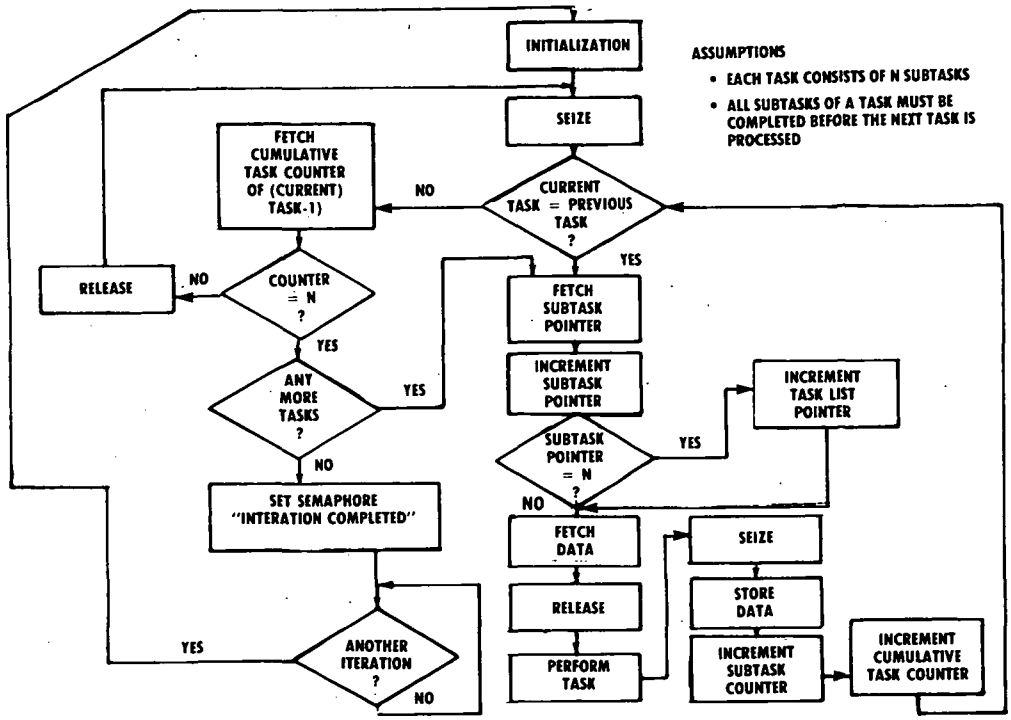
MASTER PROCESSOR CONTROL PRIMITIVES



SLAVE PROCESSOR CENTROL PRIMITIVES



TASK EXECUTION CONTROL



DFT APPLICATION

A DFT CAN BE DEFINED IN THE FOLLOWING MATRIX FORM:

$G = WF$

IF WE LET:

- $n = 0,1,2, \ldots, N-1$ = MATRIX ROW NUMBER AND FREQUENCY STEP
- $k = 0,1,2, \ldots, k-1$ = MATRIX COLUMN NUMBER AND TIME STEP
- $N = k$ BUT MAINTAINING n AND k NOTATIONS TO DISTINGUISH ROWS FROM COLUMNS

THEN:

- W IS AN $N \times k$ MATRIX CONSISTING OF THE TERMS

$$W^{n,k} = e^{(-2\pi j/N)(nk \text{ MOD } N)}$$

$$= \cos\left[\left(\frac{2\pi}{N}\right)(nk \text{ MOD } N)\right] - j \sin\left[\left(\frac{2\pi}{N}\right)(nk \text{ MOD } N)\right]$$

- F IS A $k \times 1$ MATRIX REPRESENTING THE FUNCTION $F(tk)T/2\pi k$ OVER THE TIME SPAN T
- G IS AN $N \times 1$ MATRIX WHERE $Gn = T/2\pi k \sum_{k=0}^{K-1} W^{n,k} F(tk)$

IN EXPANDED FORM, $G = WF$ CAN BE WRITTEN AS:

$$\begin{pmatrix} G0 \\ G1 \\ G2 \\ \vdots \\ GN-1 \end{pmatrix} = \begin{pmatrix} W^{0,0} & W^{0,K-1} & W^{0,2} & W^{0,K-1} \\ W^{1,0} & W^{1,K-1} & W^{1,2} & W^{1,K-1} \\ W^{2,0} & W^{2,K-1} & W^{2,2} & W^{2,K-1} \\ \vdots & & & \vdots \\ W^{N-1,0} & W^{N-1,K-1} & W^{N-1,2} & W^{N-1,K-1} \end{pmatrix} \begin{pmatrix} Fto\ T/2\ \pi\ K \\ Ft1\ T/2\ \pi\ K \\ Ft2\ T/2\ \pi\ K \\ \vdots \\ FtK-1\ T/2\ \pi\ K \end{pmatrix}$$

SINCE:

$$Gn,k\ (REAL) = \cos\left[\left(\frac{2\pi}{N}\right)(nk \text{ MOD } N)\right] F(tk)\ T/2\ \pi k$$

$$Gn,k\ (IMAGINARY) = -j\left\{\sin\left[\left(\frac{2\pi}{N}\right)(nk \text{ MOD } N)\right\}F(tk)\ T/2\ \pi k\right.$$

$\omega n = n\Delta\omega$ WHERE $\Delta\omega = \frac{2\pi K}{NT}$

$$|Gn| = \sqrt{\left[\sum_{k=0}^{K-1} Gn,k\ (REAL)\right]^2 + \left[\sum_{k=0}^{K-1} Gn,k\ (IMAGINARY)\right]^2}$$

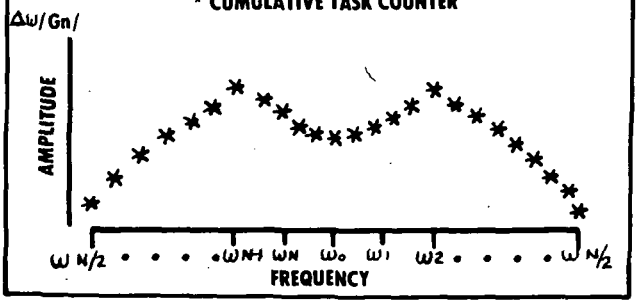
THE AMPLITUDE/FREQUENCY VALUES CAN BE OBTAINED AS FOLLOWS:

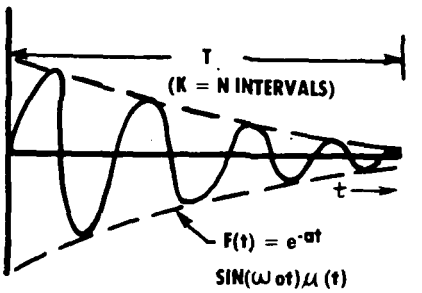
$$AMP(\omega_n) = \Delta\omega |Gn|$$

INPUT/OUTPUT

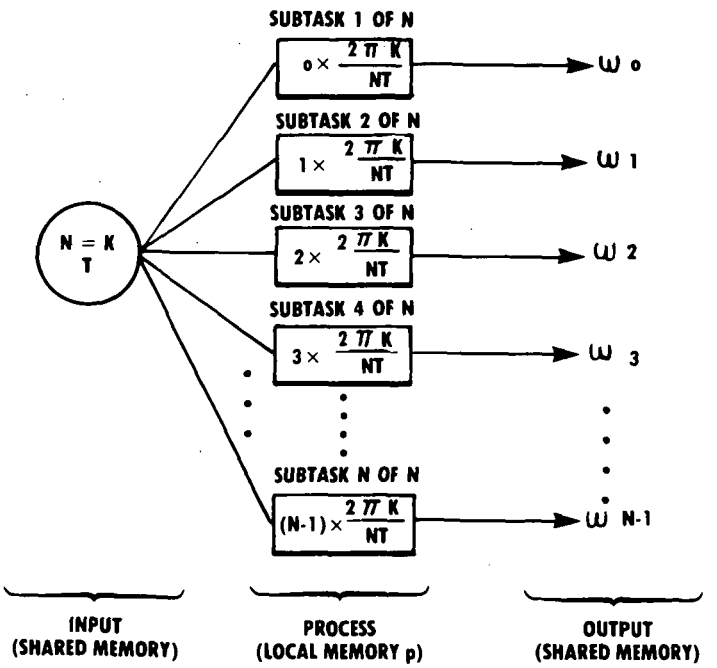
SLAVE PROCESSOR	TASK/SUBTASK COMPLETIONS		
	TASK 1	TASK 2	TASK 3
1	X	X	X
2	X	X	X
3	X	X	X
4	X	X	X
5	X	X	X
CTC*	X	X	X

* CUMULATIVE TASK COUNTER

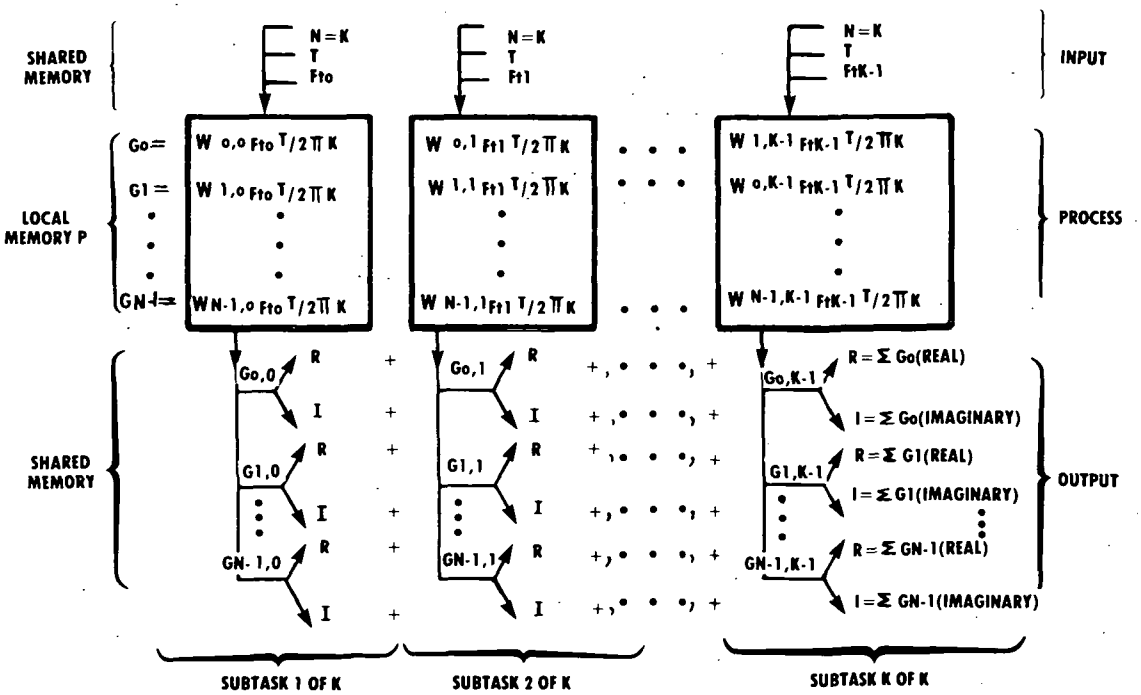




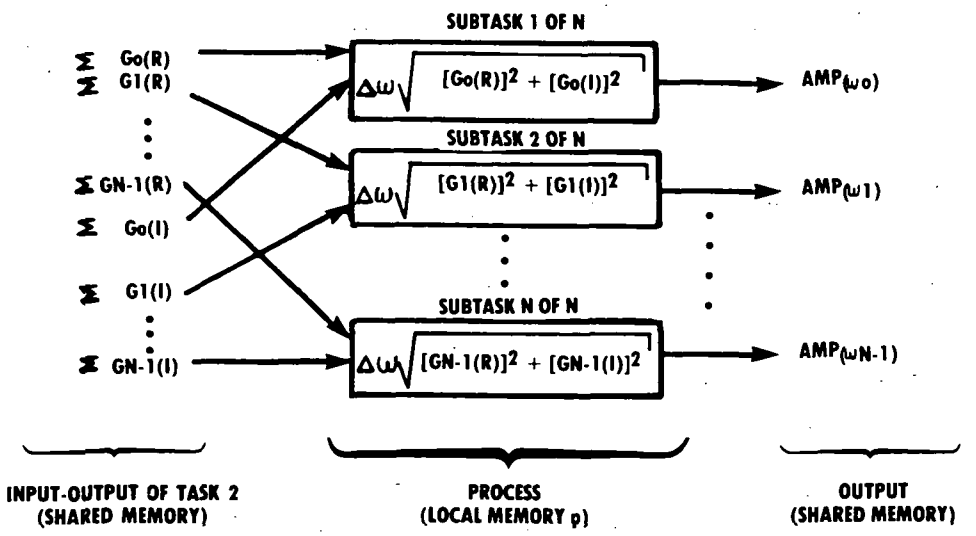
DFT DECOMPOSITION FOR TASK 1



DFT DECOMPOSITION FOR TASK 2



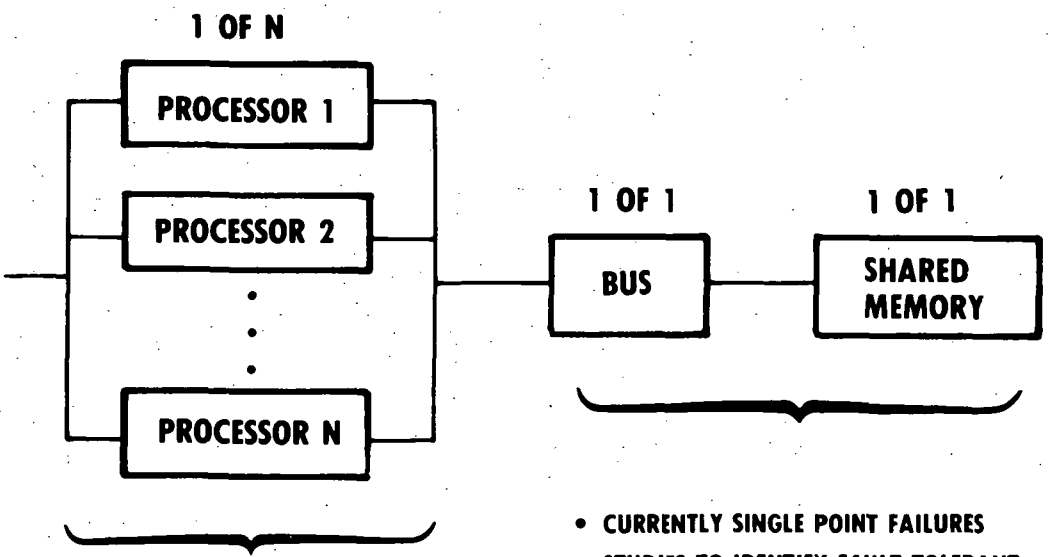
DFT DECOMPOSITION FOR TASK 3



STATUS

- IMPLEMENTATION
- GCSS SIMULATION
- LABORATORY EVALUATION
- FAULT TOLERANT STUDIES
  - PROCESSOR
  - SHARED MEMORY
  - BUS

RELIABILITY MODEL



- TAKE ADVANTAGE OF MULTIPLE PROCESSORS
- OPTIMIZE EXISTING CONTROL STRUCTURE FOR FAULT-TOLERANCE PURPOSES

- CURRENTLY SINGLE POINT FAILURES
- STUDIES TO IDENTIFY FAULT TOLERANT SCHEMES
- POSSIBLE IMPLEMENTATION OF HIGHLY RELIABLE SHARED MEMORY WOULD BE DUPLEXED CONFIGURATION EACH WITH SINGLE ERROR CORRECTION AND DOUBLE ERROR DETECTION

Page Intentionally Left Blank

DISTRIBUTED MICROPROCESSORS IN A TACTICAL UNIVERSAL MODEM

D. M. Gray, J. B. Malnar, and H. Vickers
Harris Corporation
Melbourne, Florida

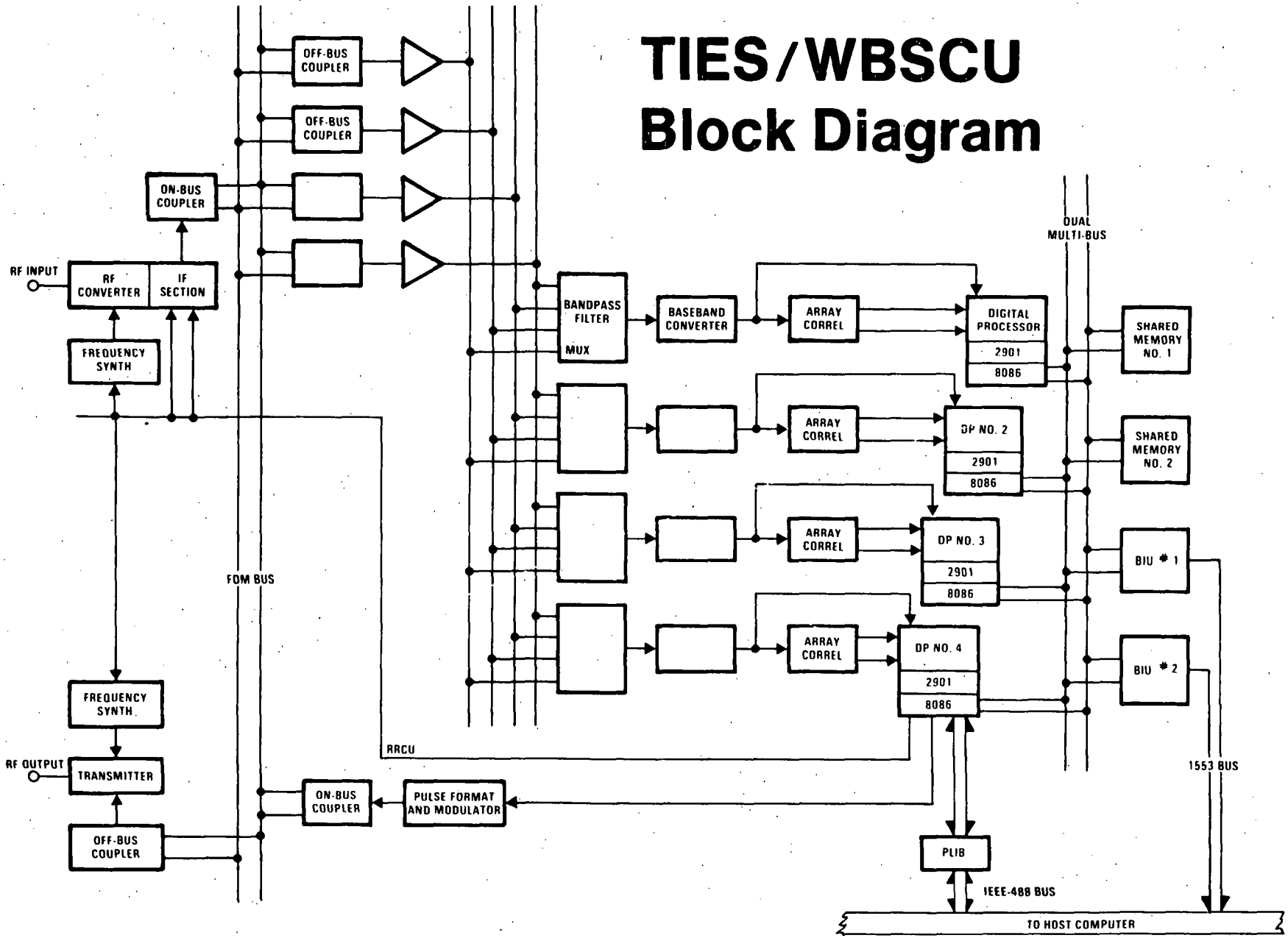
The Wideband Signal Conversion Unit (WBSCU) for the Tactical Information Exchange System (TIES) is a four-channel software reprogrammable modem. System goals are high resource availability, growth potential, reliability, and graceful degradation. The WBSCU processes JTIDS, GPS, IFF/DABS, and TACAN waveforms simultaneously under direction of a host computer. For both complex spread spectrum and pulse-based waveforms, the WBSCU provides matched filtering, signal processing, error detection/correction and encoding/decoding, and data communication via IEEE 488 and MIL-1553 buses.

Multiple embedded 8086 and 2901 microprocessors, supported by dedicated hardware modules, perform the required real-time operations for both transmit and receive functions. Commands from a host computer determine the configuration of the WBSCU via the IEEE 488 bus. Each of the four WBSCU channels is assigned to process a specified IF waveform; each channel configures its own resources and, in some cases, borrows resources from other channels. The processed waveform data is communicated from individual channels to redundant global memories. Data flow between the user community and global memories occurs via redundant 1553 buses through intelligent Bus Interface Units.

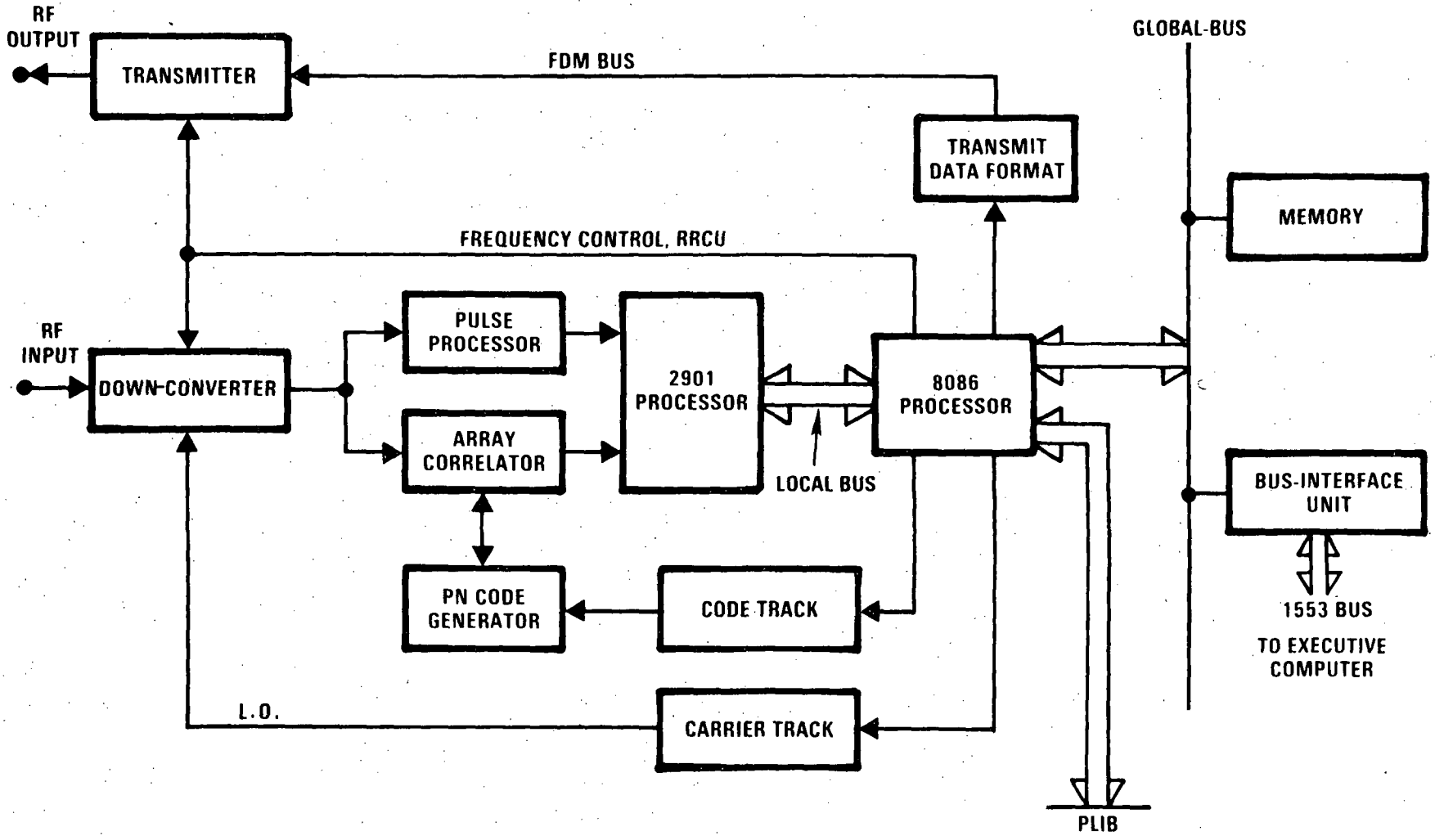
Each WBSCU channel contains one 2901 bit-slice machine and one 8086 microprocessor. The 2901 provides high-speed processing capability for the most time-critical operations. Features include a 16-bit word size, 64-bit microcoded instruction, and 10 MHz instruction throughput. The 8086 is used for lower speed processing tasks where its high level language capability can be better exploited. Each 8086 has a global bus for wideband interprocessor communication, and a local bus for 8086/2901, master/slave communication. Software architecture consists of a control and communications structure governing mode-dependent signal processing tasks.

In GPS data acquisition, the 2901 processes array correlator outputs and generates code and carrier error signals. The 8086 implements the loop filters required to achieve code lock and carrier synchronization, performs error detection and extracts message bits at a 50 b/s data rate. This data is output via the global memory. During JTIDS signal processing, the 2901 controls hardware modules to generate the acquisition strobe and time refine signals, readying the system to receive the data message. The 8086 performs message level data processing for both transmit and receive functions, data routing, and interchannel communications.

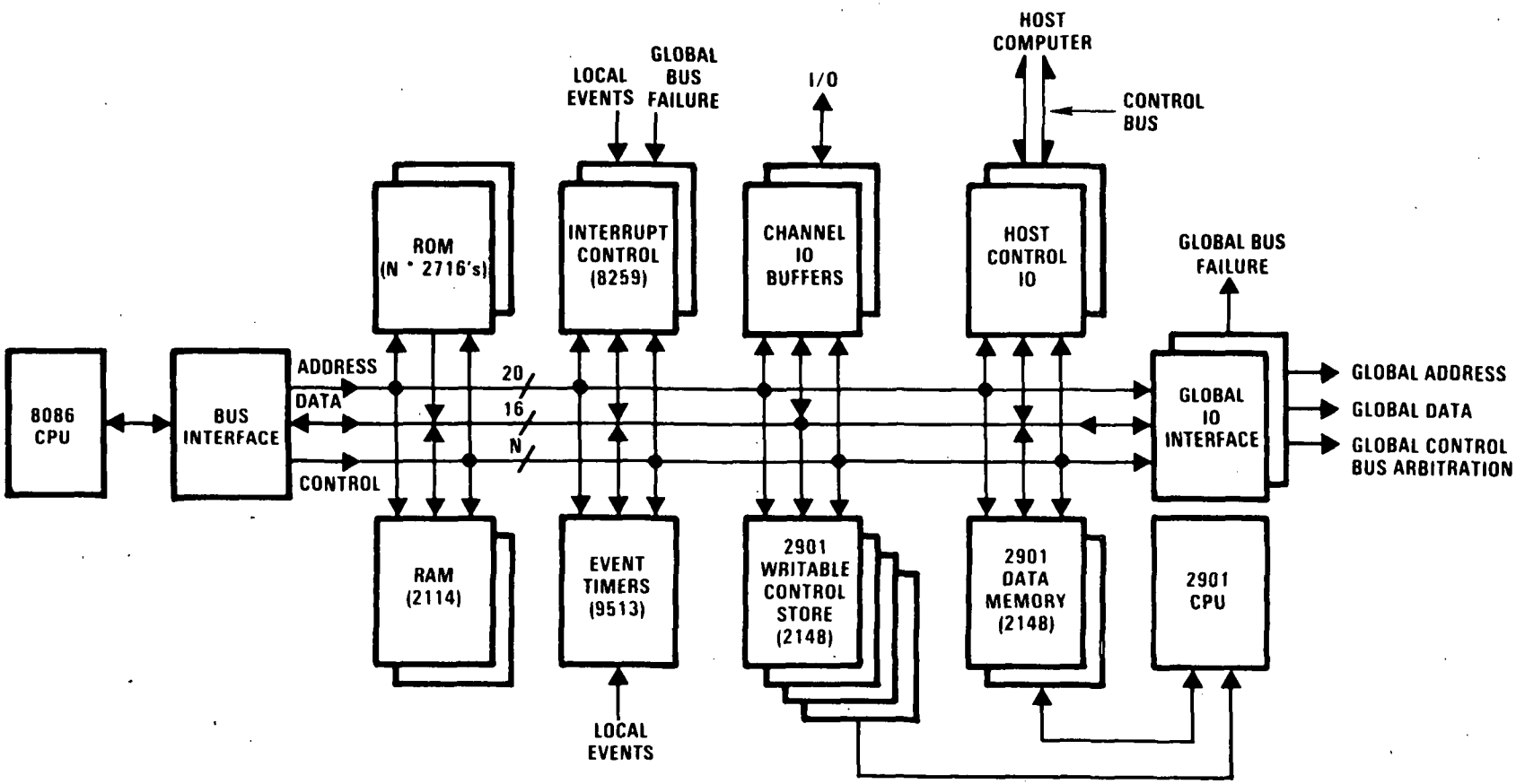
TIES/WBSCU Block Diagram



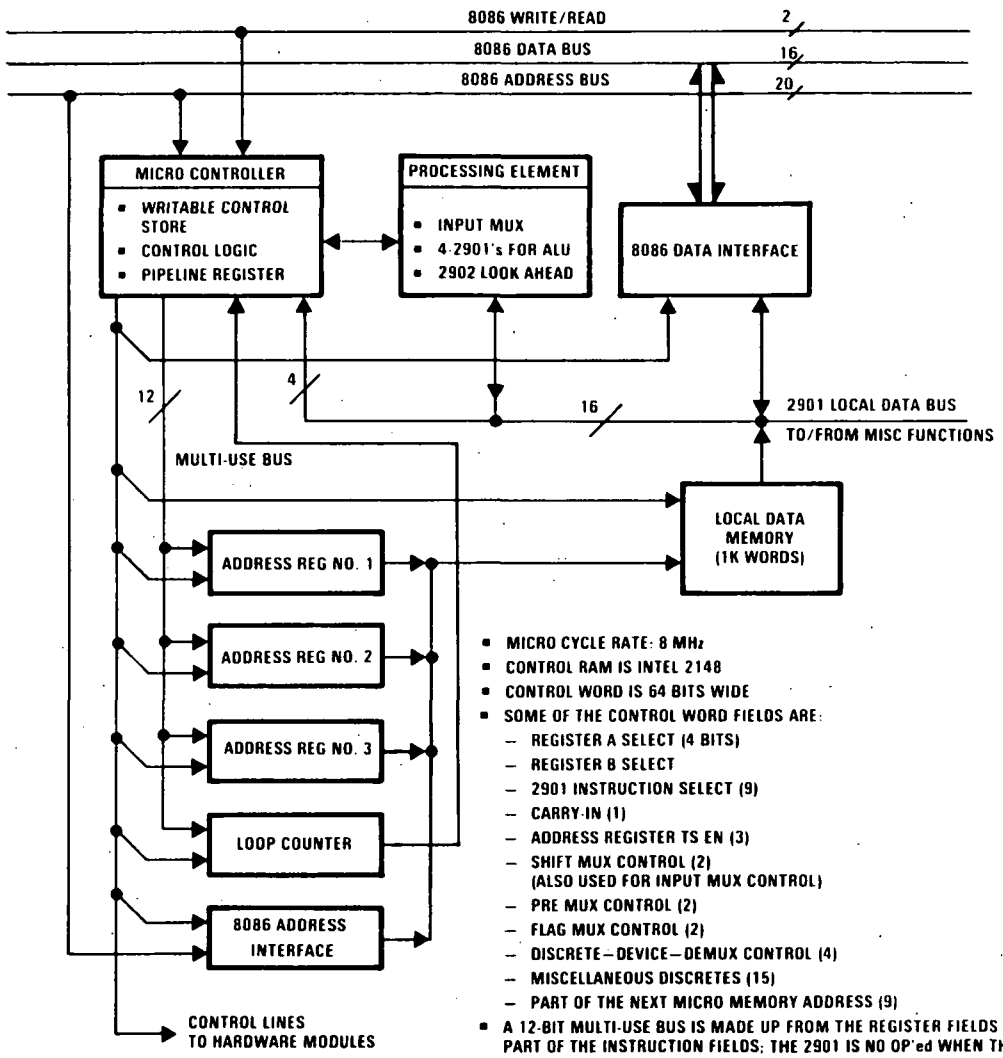
Single Channel WBSCU Configuration



Local 8086 Architecture

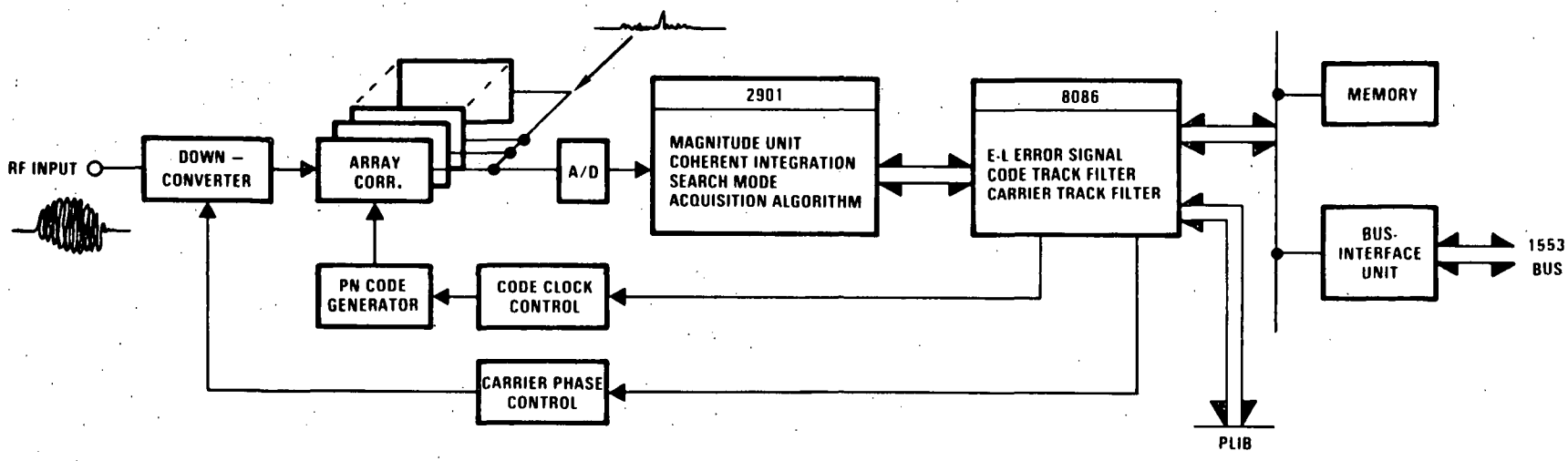


2901 Bit-Slice Microprocessor

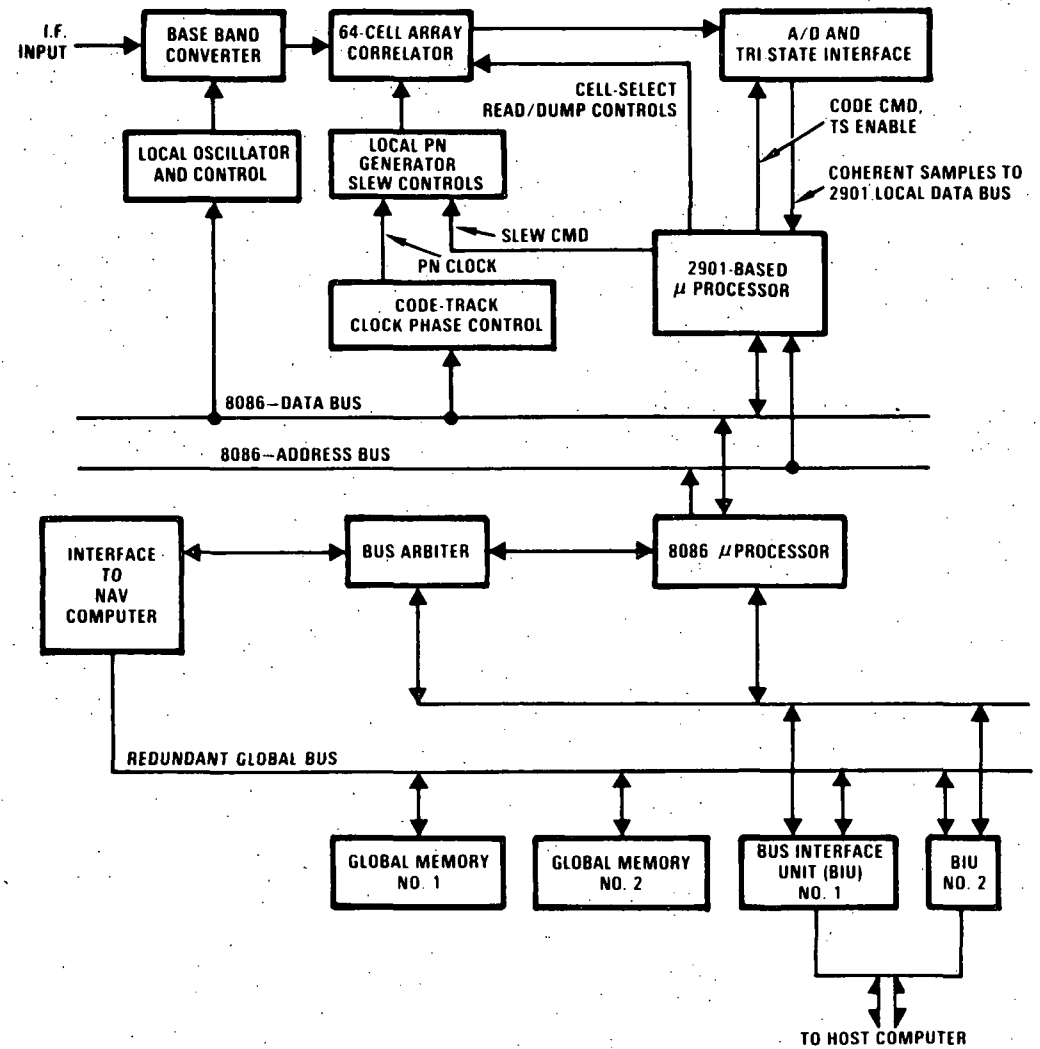


- MICRO CYCLE RATE: 8 MHz
- CONTROL RAM IS INTEL 2148
- CONTROL WORD IS 64 BITS WIDE
- SOME OF THE CONTROL WORD FIELDS ARE:
  - REGISTER A SELECT (4 BITS)
  - REGISTER B SELECT
  - 2901 INSTRUCTION SELECT (9)
  - CARRY-IN (1)
  - ADDRESS REGISTER TS EN (3)
  - SHIFT MUX CONTROL (2) (ALSO USED FOR INPUT MUX CONTROL)
  - PRE MUX CONTROL (2)
  - FLAG MUX CONTROL (2)
  - DISCRETE–DEVICE–DEMUX CONTROL (4)
  - MISCELLANEOUS DISCRETES (15)
  - PART OF THE NEXT MICRO MEMORY ADDRESS (9)
- A 12-BIT MULTI-USE BUS IS MADE UP FROM THE REGISTER FIELDS AND PART OF THE INSTRUCTION FIELDS; THE 2901 IS NO OP'ed WHEN THE MULTI-USE FIELD IS USED

Single Channel GPS Configuration

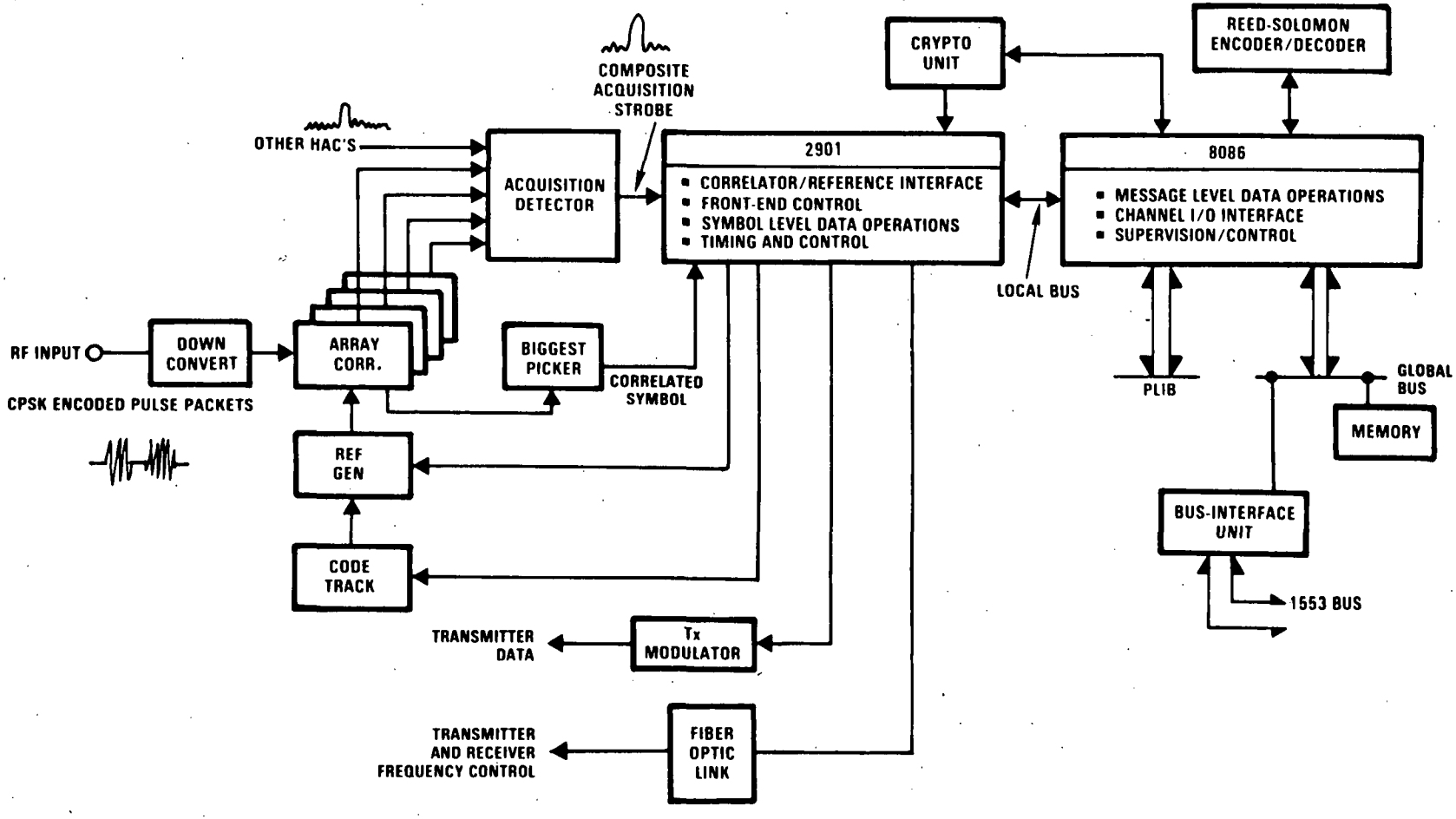


GPS SYSTEM USES 2901 AND 8086 MICROPROCESSORS



- 8086 TASKS
  - LOAD 2901 WRITABLE CONTROL STORE
  - LOAD CONSTANTS IN 2901 LOCAL DATA RAM
  - SERVICE 2901 INTERRUPTS
  - USE INTEGRATED DATA FROM ON-TIME CELL TO LOCATE GPS DATA BIT BOUNDARIES
  - FILTER INTEGRATED DATA FROM EARLY AND LATE CELLS TO CONTROL CODE-TRACK CIRCUIT
  - FILTER INTEGRATED DATA FROM ON-TIME CELL TO CONTROL LOCAL OSCILLATOR CIRCUIT
  - PASS GPS DATA AND WBSCU CHANNEL STATUS TO GLOBAL MEMORIES
- 2901 TASKS
  - COHERENT INTEGRATION OF CORRELATOR SAMPLES; THIS EXTENDS 64-CELL ARRAY INTEGRATION TIME
  - NON COHERENT INTEGRATION OR VECTOR SUMATION OF SAMPLES PREVIOUSLY INTEGRATED COHERENTLY
  - THRESHOLD DETECTION AND ACQUISITION—TO-TRACK DECISION-MAKING
  - COHERENT INTEGRATION OF DATA SAMPLES FROM THE ON-TIME CELL
  - COHERENT INTEGRATION OF DATA SAMPLES FROM THE EARLY AND LATE CELLS

Single Channel JTIDS Configuration



SESSION III

MICROPROCESSOR SOFTWARE TECHNOLOGY

MICROCOMPUTER SOFTWARE DEVELOPMENT FACILITIES

J. S. Gorman and C. Mathiasen
Ford Aerospace
Houston, Texas

Historically, microcomputer software/firmware has been developed on a system designed to perform software development and emulation functions for a specific processor. This method proved successful during the early years of microcomputer software development. However, both the number of microcomputers used in system design, and the complexity of the microcomputer software, have increased. These changes dictate that more efficient and cost effective methods be found for developing microcomputer software.

One approach is to utilize a host computer with high-speed peripheral support. Application programs such as cross assemblers, loaders, and simulators are implemented in the host computer for each of the microcomputers for which software development is a requirement. The host computer is configured to operate in a time-share mode for multi-users. The provided remote terminals, printers, and down loading capabilities are based on user requirements. With this configuration a user, either local or remote, can use the host computer for microcomputer software development. Once the software is developed (through the code and modular debug stage) it can be downloaded to the development system or emulator in a test area where hardware/software integration functions can proceed. The microcomputer software program sources reside in the host computer and can be edited, assembled, loaded, and then downloaded as required until the software development project has been completed.

The use of a host computer for microcomputer software development allows greater expansion and versatility and has resulted in increased performance and improved programmer productivity.

Preceding Page Blank

Microcomputer Software Development History

- Traditionally accomplished on dedicated development systems
- Development systems supported one specific processor or processor sets
- Minimal operating systems or monitor programs supplied for run time and development support
- Standard utilities provided: assemblers and editors
- Development method initially adequate

The Need for Change

- Increasing software complexity surfaced many problems
- Single user environment time consuming, inefficient
- Difficult to support more than one type of target processor
- Development systems provided minimum peripheral support
- Development capability relatively slow
- Non-availability of development system became serious problem
- Need for change eminent

Host Computer System Philosophy

- Use of host computer system solves many problems
- Host computer can service multi-user environment
- Via cross-product software, one host computer can support any number of target processors
- Host computer peripheral support can be wide and varied
- Host computer provides substantial, resident development tools
- Host computer can support remote development facilities
- Once software developed, can be downloaded to target processor
- Debug and integration can continue via in-circuit emulators

Implementation: Hardware Implications

- Terminal, printer, and download line selection can be determined from user requirements
- Communication equipment selection can be determined by data rates, line protocol, transmission distances, and signal multiplexing
- Computer interfacing equipment selection can be determined by I/O availability and user equipment interface characteristics
- Remote printers and host computers must support busy/ready indication using modem circuits
- Using standard host computer interface protocol in user equipment can minimize computer interface driver modifications

Implementation: Software Implications

- First priority: selection of cross-product software
- If local talent and experience available, software products can be developed instead of purchased
- Off-the-shelf assemblers and high level language compilers will support most processors
- Most vendors supply executable format software only. Some vendors provide source code
- Software for simulation capability should also be purchased
- Via total cross-product software package and host system development facilities, software can be checked out through software-to-software integration phase

Possible Obstacle: Transferring Software to Target Processor

- Use like peripherals on both host system and target processor
- Host computer flexible disk system can simplify software transfer
- Host computer flexible disk system can also provide source code archival
- Flexible disk formats must be hardware and software compatible with target processor format
- If host computer is remote from target processor, shuttling diskettes back and forth can lead to additional software transfer method analysis

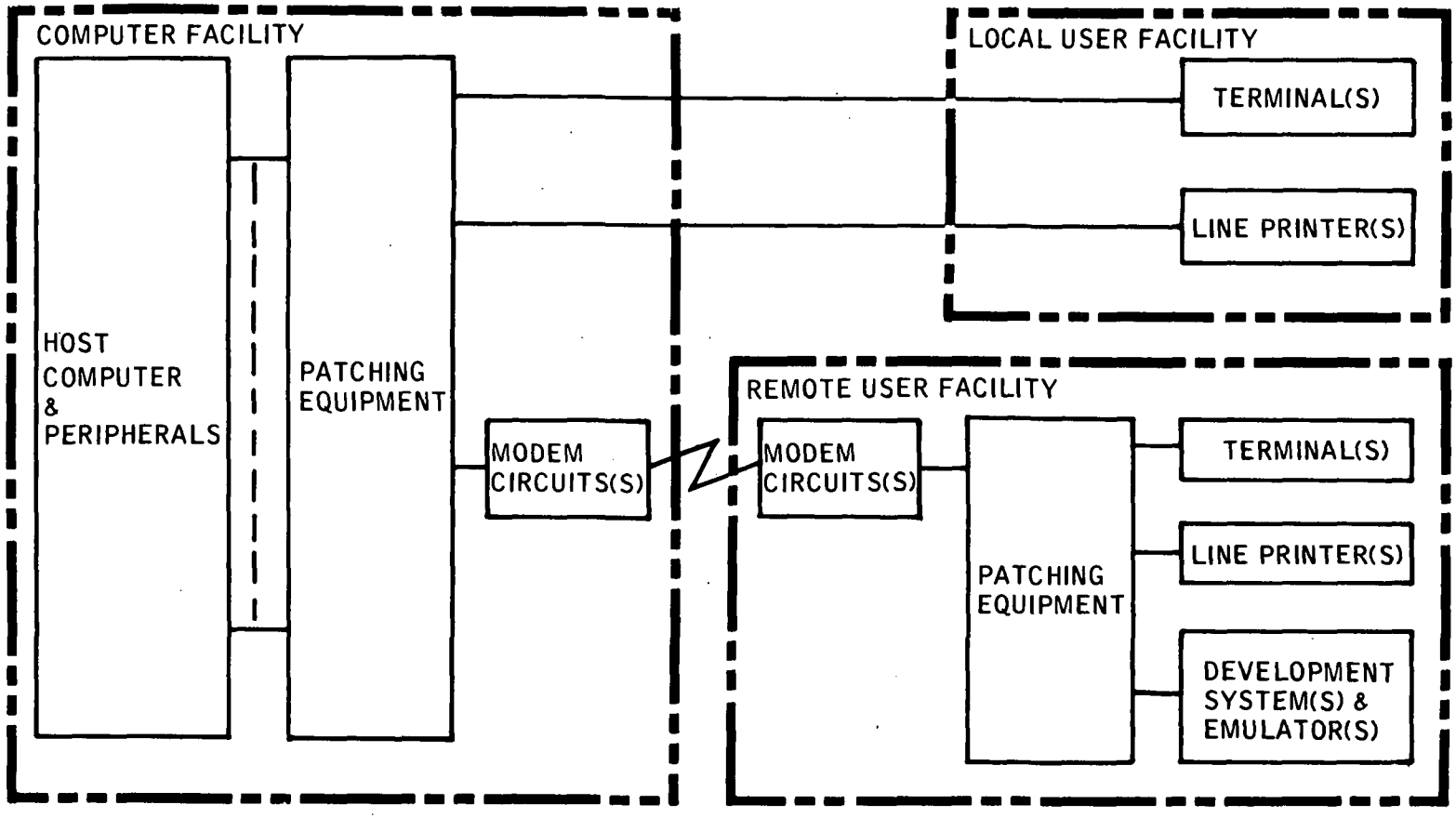
Developing A Downloader Capability

- Some vendors supply downloader software
- Development involves minimal software effort
- Downloader approach involves use of development system or emulation device
- Can use dedicated development system or emulator port for host computer interface
- Via download lines, can use I/O device on development system or emulator to communicate with host computer
- Two downloading philosophies
  - -- Controlled method
  - -- Pseudo terminal method
- Same environment will also support upload capability

Avoiding Pitfalls

- Consider many issues during planning stages
- Host computer memory limitations can also limit performance and efficiency
- Host computer task size limits can increase overlay needs
- Cross-product software idiosyncracies can limit software transportability
- Investigate symbol table and macro limitations before purchasing cross-product software
- If source code purchased, additional cost factors can surface
- Development facility personnel must be adequately trained in host computer use and cross-product software

Typical Microcomputer Software Development Facility Configuration



MICROPROCESSOR USER SUPPORT AT LANGLEY RESEARCH CENTER

Jerry H. Tucker
Langley Research Center
Hampton, Virginia

The microprocessor is the most significant advance in electronics since the invention of the transistor. We are building systems today with microprocessors that a few years ago would have been completely impractical or extremely expensive. Microprocessors provide numerous advantages to the system designer such as lower cost, reduced development time, increased flexibility, and increased reliability. Despite these advantages, the use of microprocessors pose significant problems. These include:

(1) A long learning process for proficient use of microprocessors.

(2) The requirement for extensive support in both hardware and software.

(3) The need for coordination and sharing of the creative effort to avoid unnecessary duplication.

The following steps have been implemented at Langley Research Center to address these problems.

(1) The establishment of a microprocessor users committee to provide an advisory interface for management and users.

(2) The training of microprocessor users.

(3) The publication of a newsletter to dissiminate information among microprocessor users.

(4) The use of both cross-software on the central computer complex and microprocessor development systems to support the design of microprocessor based systems.

This presentation provides a general overview of microprocessor user support at Langley Research Center, including a detailed review of each of the above items. Special emphasis is given to the microprocessor support available from the central computer complex. An assessment of the effectiveness of the approach being taken at Langley is given. In addition, specific hardware and software development efforts that are targeted toward enhancing the existing microprocessor support is discussed.

MICROPROCESSOR SUPPORT AT LANGLEY RESEARCH CENTER

- MICROPROCESSOR USER'S COMMITTEE
- CENTRAL COMPUTER COMPLEX SUPPORT
- LOCAL MICROPROCESSOR SYSTEM DEVELOPMENT LABS

MICROPROCESSOR USER'S COMMITTEE

- CONSISTS OF KEY MICROPROCESSOR USERS
- ADVISES MANAGEMENT AND USERS
- ASSISTS IN PROVIDING USER TRAINING
- ASSISTS IN DEFINING REQUIRED HARDWARE AND SOFTWARE SUPPORT
- COMMUNICATES TO USER COMMUNITY VIA A "MICRO-NEWSLETTER" (sent to 200 people)

CENTRAL COMPUTER COMPLEX SUPPORT

- MAINTAINS MICROPROCESSOR SUPPORT SOFTWARE
  - ASSEMBLERS (14)
  - COMPILERS (4)
  - SIMULATORS (12)
  - UTILITY PROGRAMS

- IN-HOUSE DEVELOPMENT
  - ASSEMBLERS, DISASSEMBLERS
  - 8748 PASCAL COMPILER
  - 68000 HALS COMPILER
  - CENTRAL COMPUTER COMPLEX LINKS TO MICROPROCESSOR LABS
  - HARDWARE AIDS FOR USERS

MICROPROCESSOR SUPPORT MAINTAINED ON LRC's CENTRAL COMPUTER COMPLEX

MICROPROCESSOR	DESCRIPTION	ASSEMBLER	SIMULATOR	HOL
8086	Intel 16-Bit	✓	✓	
68000	Motorola 16-Bit	✓	✓	PASCAL (HALS)
9900	Old 16-Bit	✓		
8080/8085	8-Bit	✓	✓	PLM
6800/6802 . . .	8-Bit	✓	✓	MPL
Z-80	8-Bit, Super 8080	✓	✓	8080 PLM
6809	8-Bit, Super 6800	✓		
8748/8741	8-Bit, Single Chip	✓	✓	PASCAL
1802	8-Bit	✓	✓	
6502	8-Bit	✓		
4040	4-Bit	✓	✓	

LOCAL MICROPROCESSOR SYSTEM DEVELOPMENT LABS

- Typical Lab (ACD's)
  - Intel Development System linked to Central Computer Complex
  - Over a dozen users with standard procedures and documentation
  - Assemblers for 8080/8085, 8748/8741, 8086/8088
  - PLM compilers for 8080/8085, 8086/8088
  - FORTRAN for 8080/8085
  - IN-CIRCUIT EMULATORS (ICE) for 8080, 8085, 8748, 8086, 3000
  - EPROM PROGRAMMER
- Local Microprocessor Development Systems at Langley
  - Intel (7)
  - Motorola (5)
  - Tektronix (2)
  - Texas Instruments (2)

CHARACTERISTICS OF MICROPROCESSOR SOFTWARE ON CENTRAL COMPUTER COMPLEX

- STRENGTHS
  - UNIVERSAL
  - SIMULTANEOUS USERS
  - FAST
  - FLEXIBLE
  - TAKES ADVANTAGE OF EXISTING RESOURCES
- WEAKNESSES
  - RELATIVELY DIFFICULT TO LEARN (COMPLEX OPERATING SYSTEM AND EDITOR)
  - LACK STATE OF THE ART SOFTWARE
  - CANNOT SIMULATE ENTIRE SYSTEM
  - FRAGMENTS DEVELOPMENT PROCESS

DEBUGGING EMBEDDED COMPUTER PROGRAMS

Gilbert H. Kemp
General Dynamics/Western Data Systems Center
Pamona, California

The debugging of embedded digital computer programs is a function that can use a wide range of support tools from essentially none to sophisticated systems. Every embedded computer program must complete its debugging cycle using some system that will allow real time debugging.

A listing of many of the common items addressed during debugging is given. Several approaches to debugging are analyzed to evaluate how well they treat those items. Cost evaluations are also included in the comparison.

The approaches compared are: (1) no software support, (2) embedded computer augmented with additional software for debugging purposes, (3) microprocessor development systems, (4) an environment simulation and interpretive computer simulation combination run on a large scale computer, (5) an environment simulation on a hybrid computer coupled with the embedded computer, (6) an environment simulation on a midi-computer coupled with an emulation of the embedded computer, and (7) an environment simulation on a midi-computer coupled with a slightly modified embedded computer.

The results of the comparison indicates that the best collection of capabilities to cover the common items present in the debugging task occurs in the approach where a midi-computer handles the environment simulation with an emulation of some kind representing the embedded computer. This approach can be taken at a reasonable cost.

The case study chosen is an embedded computer in a tactical missile. Several choices of computer for the environment simulation are discussed as well as different approaches to the embedded computer emulator.

The selected choice for computer the environment simulation is a special-purpose computer designed to rapidly solve differential equations. The Applied Dynamics International AD-10 computer is an example of this type. This appears to be capable of solving a full 6 Degree of Freedom missile simulation in real time.

The proposed choice for emulating the embedded computer is to use a modified version of the embedded computer itself. It is called an "Extended" computer. The "extension" amounts to adding extra bits to the program memory. When an instruction is loaded for execution, an interrupt will occur if one of these extra bits is up. Software interrupt service routines will determine what debugging function is to occur, e.g., start a trace.

The conclusion is that the use of the AD-10 and "Extended" embedded computer will show a debugging cost savings approaching 44 percent over the current commonly used team of Interpretive Computer Simulation used in conjunction with a hybrid/embedded computer pair.

Reference

Glass, Robert L., "Real-Time: The 'Lost World' Of Software Debugging and Testing", Comm. ACM 23, 5 (May 1980), Pages 264-271.

DEBUGGING EMBEDDED COMPUTER PROGRAMS

FOR ANY TASK:

- o IMPROVED PRODUCTIVITY IS DESIRABLE
- o BETTER TOOLS CAN IMPROVE PRODUCTIVITY
- o THE TOOLS MUST BE COST EFFECTIVE

APPLYING THESE THOUGHTS TO THE TASK OF DEBUGGING EMBEDDED DIGITAL COMPUTER PROGRAMS WILL BE DISCUSSED.

POMONA DIVISION DESIGNED DIGITAL COMPUTERS

COMPUTER	YEAR DESIGNED	NUMBER OF INSTRUCT.	NUMBER OF GEN.PURPOSE REGISTERS	INTER-RUPTS	SHORTEST INSTRUCTION TIME(μSEC.)	RELATIVE TIME OF EXECUTION	TECH-NOLOGY	MICRO-CODED
A	1971	24	2	NONE	0.4	1.0	HARD WIRED	NO
B	1975	74	4	ONE	0.9	3.1	BIT SLICE-INTEL 3000	YES
C	1976	102	8(16)	VECTORED	0.4	1.2	BIT SLICE-AMD 2900	YES
D	1978	158	9(17)	VECTORED	0.3	0.8	BIT SLICE-AMD 2900	YES

ITEMS IN DEBUGGING

A. INPUT -- HARDWARE AND CONVERSION
B. OUTPUT -- HARDWARE AND CONVERSION
C. MATHEMATICAL CALCULATIONS
D. LOGIC DECISIONS
E. CHECK ALL PATHS OUT OF DECISIONS
F. OVERFLOWS
G. MAXIMUM USE OF PRECISION WITHOUT OVERFLOW
H. LACK OF SIGNIFICANCE
I. INITIALIZATION OF VARIABLES
J. TIMING OF THE PROGRAM
K. PROCESS SWITCHING

DEBUG TECHNIQUES

A. TRACING SELECTED SECTIONS OF THE PROGRAM AT APPROPRIATE TIMES
B. ANALOG RECORDING OF SELECTED VARIABLES OF THE PROGRAM
C. PRINTING SELECTED PROGRAM VARIABLES AT APPROPRIATE TIMES
D. STOPPING ON BREAKPOINTS AND EXAMINING CONTENTS OF REGISTERS AND MEMORY

COMPARISON OF DEBUG TECHNIQUES

TYPE OF SIMULATION	COST OF DEVELOPMENT OF SIMULATION (4)	COST OF USE OF SIMULATION (4)	CAPABILITIES (4) (VERSATILITY)	USEFULNESS (4)	TYPE OF INPUTS (5)	SPEED (7)	USEFULNESS IN TESTING (8)										TYPE OF OUTPUT (9)
							A	B	C	D	E	F	G	H	I	J	K	
NONE--MFR. SOFTWARE	N	N	S	S	L/S	RT	-	-	X	-			-	-				E
NONE--ADD DEBUG SOFTWARE	S	N	S	S	L/S	RT	-	-	X	-			-	-			-	E
MICROPROCESSOR DEVELOPMENT SYSTEM (MDS)	S	S-	S	S	L,R	RT	-	-	X	-			-	-		-		P,T,E (6)
SIMULATION/I.C.S. (1) --CLOSED LOOP	H	H	H	H-	L	S	-	-	X	X	X	X	X	X	X	X	X	P,T,A,E
HYBRID (3)--CLOSED LOOP	H	H	M (10)	M	L,R	RT	X	X	-	X	-						-	A,E
MIDI/EMULATION (2) --CLOSED LOOP	M	S	H	H	L,R(?)	RT(?)/S	X	X	X	X	X	X	X	X	X	X	X	P,T,A,E

NOTES

(1) THE FLIGHT COMPUTER IS SIMULATION ON A LARGE SCIENTIFIC COMPUTER. THIS IS AN INTERPRETIVE COMPUTER SIMULATION (I.C.S.). IT IS MATED TO A FORTRAN MISSILE SIMULATION, NORMALLY IS 5 DEGREE OF FREEDOM (D.O.F.) BUT MAY BE 6 D.O.F.

(2) A 32-BIT MIDI-COMPUTER, HOOKED TO AN EMULATION OF THE FLIGHT COMPUTER WHOSE PROGRAM IS UNDER TEST, RUNS THE MISSILE SIMULATION OF (1). HARDWARE RECEIVER SIMULATOR.

(3) A FULL 6 D.O.F. MISSILE SIMULATION PUT TOGETHER FOR THIS TESTING, PLUS DESIGN AND EVALUATION OF EQUATION IS USED. HARDWARE RECEIVER SIMULATION OFTEN USED.

(4) N = NONE, S = SMALL, M = MEDIUM, H = HIGH.

(5) L = SIMULATED LIVE, S = STATIC VALUES, R = SIMULATED INPUTS TO REAL HARDWARE.

(6) TIMING INVOLVES THE USE OF A LOGIC ANALYZER.

(7) RT = REAL TIME, S = $\measuredangle$RT.

(8) LETTERS ARE FROM THE TYPES OF CODE LISTED IN TABLE 2. X INDICATES GOOD CAPABILITY, (-) INDICATES THE CAPABILITY IS NOT AS GOOD AS OTHER APPROACHES AND A BLANK INDICATES NO CAPABILITY.

(9) E = EXAMINE ONE VARIABLE AT A TIME, A = ANALOG TYPE OUTPUT, P = PRINTED OUTPUT, T = TRACING OUTPUT.

(10) WHEN USING HYBRID, SCALING OF VARIABLES IS NOT EASY TO CHECK. IF IT WERE, THEN WOULD RATE H.

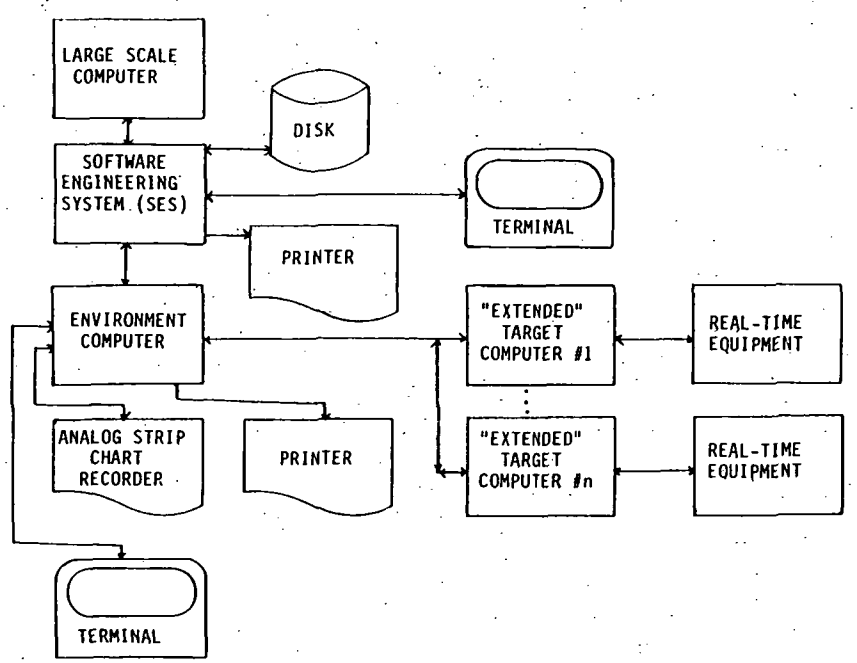
EMULATION APPROACHES

ENVIRONMENT COMPUTER	EMULATION COMPUTER
VAX	QM-1
VAX	"EXTENDED" TARGET
AD-10	"EXTENDED" TARGET

FUNCTIONS OF THE ADDED BITS

BITS	FUNCTION
0001X	IGNORE AN OVERFLOW ON THIS INSTRUCTION IF IT OCCURS, OTHERWISE PRINT AN ERROR MESSAGE IF AN OVERFLOW OCCURS.
0101X	CAUSE PROGRAM TO BECOME SYNCHRONIZED WITH ENVIRONMENT COMPUTER PROGRAM.
0100X	BREAKPOINT.
1000X	START A TRACE--IF TIME HAS REACHED A SPECIFIED VALUE. ALSO SET A FLIP-FLOP TO COMMAND AN INTERRUPT TO PRINTOUT TRACE DATA ON EACH SUCCEEDING INSTRUCTION.
1010X	STOP TRACING, I.E., RESET THE TRACE FLIP-FLOP.
1100X	START TIMING FROM POINT A TO POINT B IF RUN TIME HAS REACHED A SPECIFIED VALUE, I.E., READ THE CLOCK OF TARGET OR ENVIRONMENT COMPUTER.
1110X	STOP TIMING--READ THE CLOCK OF TARGET OR ENVIRONMENT COMPUTER AND PRINT THE DELTA TIME SINCE TIMING STARTED.

ENVIRONMENT COMPUTER/"EXTENDED" TARGET COMPUTER BLOCK DIAGRAM



OVERALL COMPARISON OF DEBUG TECHNIQUES

APPROACH	ADVANTAGES	DISADVANTAGES
ACTUAL TARGET COMPUTER.	LEAST EXPENSIVE RE HARDWARE.	NOT ENOUGH TOOLS. TOO PRIMITIVE.
MDS	INEXPENSIVE RE HARDWARE.	NOT ENOUGH TOOLS. TOO PRIMITIVE.
INTERPRETIVE COMPUTER SIMULATION (I.C.S.) ON A LARGE COMPUTER PLUS HYBRID PAIR--USED SEPARATELY.	GOOD SET OF TOOLS. HYBRID IS REAL-TIME. HYBRID USES REAL TARGET COMPUTER. CURRENT METHOD IN USE.	VERY EXPENSIVE RE HARDWARE AND OPERATING COSTS. I.C.S. TURN AROUND IS SLOW.
VAX FOR MISSILE SIMULATION PLUS QM-1 FOR TARGET EMULATION.	GOOD SET OF TOOLS. EMULATE NON-EXISTENT TARGET COMPUTER. LESS EXPENSIVE THAN I.C.S. PLUS HYBRID.	VERY SLOW. MULTI-CPU TARGET EVEN SLOWER.
VAX FOR MISSILE SIMULATION PLUS "EXTENDED" TARGET COMPUTER.	GOOD SET OF TOOLS. USES REAL TARGET COMPUTER. ALLOWS MULTI-CPU TARGET. LESS EXPENSIVE THAN VAX PLUS QM-1.	NOT REAL-TIME. REQUIRES VERIFICATION OF "EXTENDED" TARGET COMPUTER CONCEPT.
AD10 FOR MISSILE SIMULATION PLUS "EXTENDED" TARGET COMPUTER.	GOOD SET OF TOOLS. USES REAL TARGET COMPUTER. ALLOWS MULTI-CPU TARGET. REAL-TIME! LESS EXPENSIVE THAN VAX PLUS "EXTENDED" TARGET. BEST OVERALL APPROACH.	REQUIRES VERIFICATION OF AD10 SPEED. REQUIRES VERIFICATION OF "EXTENDED" TARGET COMPUTER CONCEPT.

EMULATION METHODS SUMMARY

ITEM	APPROACH		
	VAX/QM-1	VAX/EXT.	AD10/EXT.
FULL DEBUGGING TOOL SET	YES	YES	YES
MULTIPLE CPU'S	YES-SLOW	YES	YES
"ACTUAL" HARDWARE	NO	ALMOST	ALMOST
REAL-TIME	NO	NO	YES
FULLY REPLACE HYBRID?	NO	NO	YES
EASE OF ENVIRONMENT COMPUTER PROGRAMMING	EASY	EASY	HARDER
"RUN" A NON-EXISTENT COMPUTER	YES	NO	NO
NET SAVINGS OVER PRESENT METHOD (HYBRID/ICS) OF FEATURES THAT DIFFER BETWEEN THE FOUR APPROACHES (2 MAJOR WEAPONS PROGRAMS)	17%	32%	44%

REAL-TIME OPERATING SYSTEM FOR SELECTED INTEL PROCESSORS

W. R. Pool
Ford Aerospace
Houston, Texas

This paper addresses development of a real-time operating system for selected Intel processors, in terms of four project phases.

The first phase develops the system development rationale. Included are reasons for not using vendor supplied operating systems. The second phase deals with the system design and performance goals. While many of these goals are dictated by problems with vendor supplied systems, other goals surfaced as a result of a design for a custom system able to span multiple projects. The third phase deals with system implementation. Discussed are system development and management problems and areas that required redesign or major code changes. The final phase deals with project results. First, the relative successes of the initial projects are developed. Then, a generic description of the actual project is provided. Finally, the ongoing support requirements and future plans are discussed.

Preceding Page Blank

Make/Buy: Requirements Make the Decision

- First, determine what system must do
  - -- Real-time - quick response
  - -- Multi-tasking - priority scheduling
  - -- High performance, especially for I/O
  - -- Disk file system compatible with ISIS-II
  - -- Block mode CRT support
- Selection: what will do the job ?
  - -- Does system exist that meets requirements ?
  - -- Can a system be modified at less expense/time ?
- Developing your own
  - -- Does talent exist ?
  - -- Is there time/money ?
  - -- Are development facilities available ?

System Design: A Real Experience

Design Goals

- Modification ease: add foreign devices
- Modular: use only necessary parts
- Standard device interfaces to applications
- Support any board/device configuration

System Design: A Real Experience (Cont'd)

Control Element Components

Control Program	Control Block
• Task Management	Task Control Block (TCB)
• Event Signaling	Event Control Block (ECB)
• I/O Control	File Control Block (FCB)
• Interrupt and Device Handling	Device Control Block (DCB)

System Design: A Real Experience (Cont'd)

Utility Functions

- Display support
- Disk file management
- Loader
- Operator interface
- Advisory facilities
- Debug monitor

Additional Services

- Timer services
- Dynamic memory allocation

Implementation: No Easy Victories

The Team

- Designer: control block contents, program functions
- Programmer: program design, development

The Plan

- Develop task management, interrupt service, and I/O initiation first
- Validate control block structure
- Correct any control block and design errors
- Rewrite as necessary
- Develop device support and utilities

Implementation: No Easy Victories (Cont'd)

The Problems

- Development systems slow and unreliable
- Processor poorly suited for multi-tasking and re-entrant code
- Early delivery resulted in system not fully checked out
- Tight bindings must be replaced with loose binding to separate OS from applications
- User documentation slow/non-existent
- Disk support control block formats must be changed to support different densities
- Dynamic memory support should have been included in original design
- Needed to fund support group as user group increased
- Configuration control problems due to different releases on projects and inadequate library facilities
- Should have had cross software and timesharing network at beginning

Results

Met All Requirements and Design Goals

- Currently used in three systems; being implemented in four others
- Being used in different hardware configurations
- Interrupt processing capabilities faster than other available systems
- Achieved true multi-tasking capabilities

Cost Effective

- Effort and budget savings via use on multiple projects
- No license fees

Results (Cont'd)

System Characteristics

- 6 months to produce first version; 1 year to produce debugged version in macro form
- 4K PROM for basic system with task management, I/O support, and loader
- 8K PROM adds display support, debug monitor, advisory, and full disk support
- 4-12K of RAM necessary for buffers and control blocks
- Normally will support CRT, line printer, and two flexible disk drives
- Systems usually embedded in larger equipment systems

Future Plans

- Recode for different processor
- Change user interface to loose binding type
- Augment dynamic memory support
- Convert to higher level language
- Provide support for users/projects
- Redesign disk support/control block
- Provide software library on host development system

A FOURIER TRANSFORM WITH SPEED IMPROVEMENTS FOR MICROPROCESSOR APPLICATIONS

Donald C. Lokerson
Goddard Space Flight Center
Greenbelt, Maryland

and

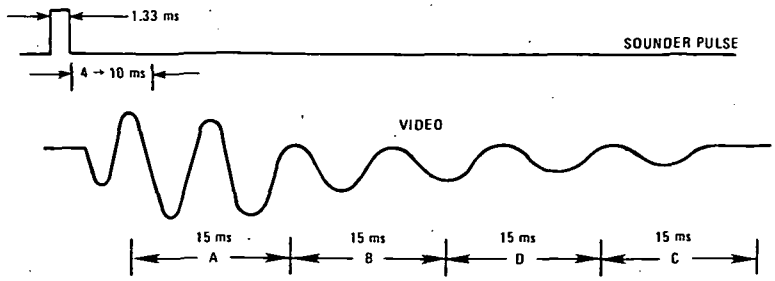
Dr. Robert Rochelle
University of Tennessee
Knoxville, Tennessee

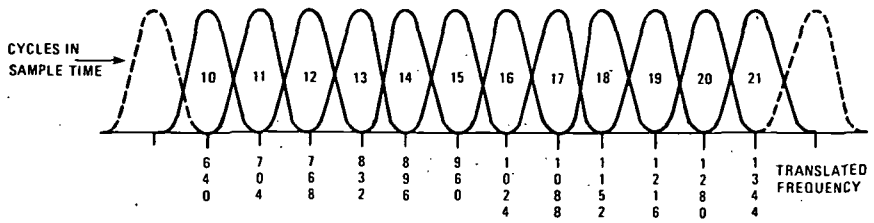
A fast Fourier transform algorithm for the RCA 1802 microprocessor has been developed for spacecraft instrument applications. The computations have been tailored for the restrictions an eight bit machine imposes.

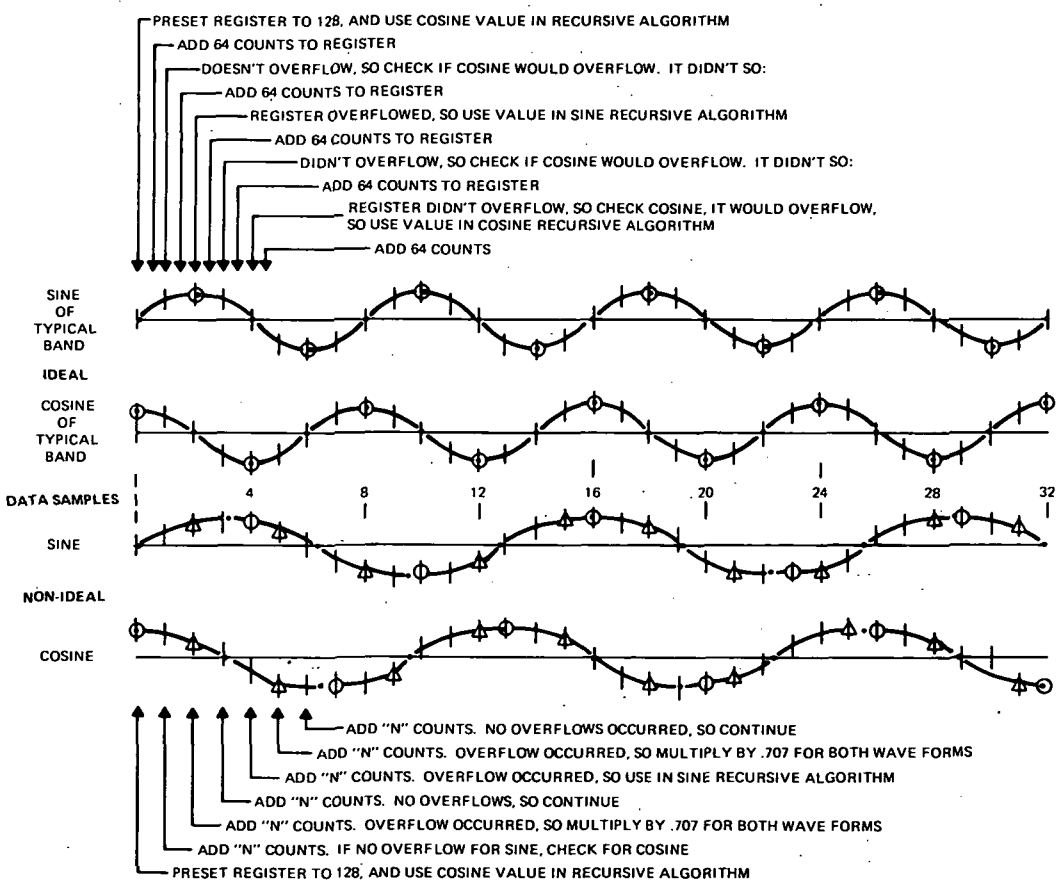
The algorithm incorporates some aspects of Walsh function sequency to improve operational speed. This method uses a register to add a value proportional to the period of the band being processed before each computation is to be considered. If the result overflows into the "DF" register, the data sample is used in computation; otherwise computation is skipped. This operation is repeated for each of the 64 data samples. This technique is used for both sine and cosine portions of the computation.

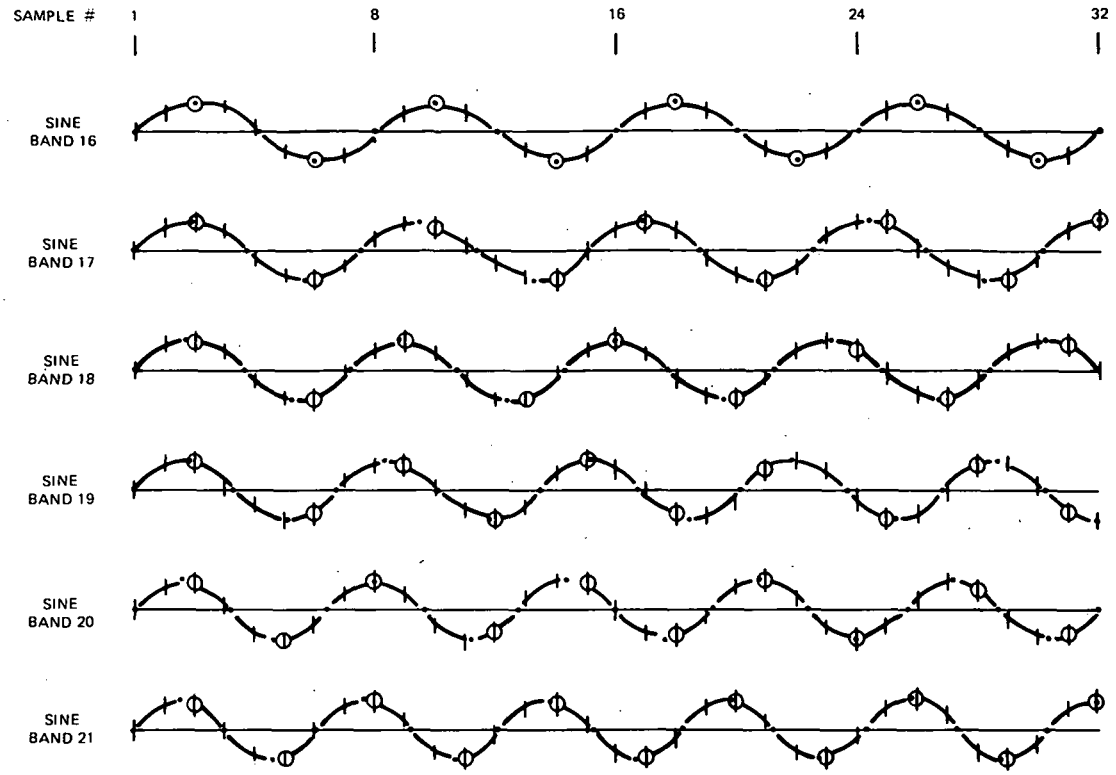
The processing uses eight bit data but, because of the many computations that can increase the size of the coefficient, floating point form is used.

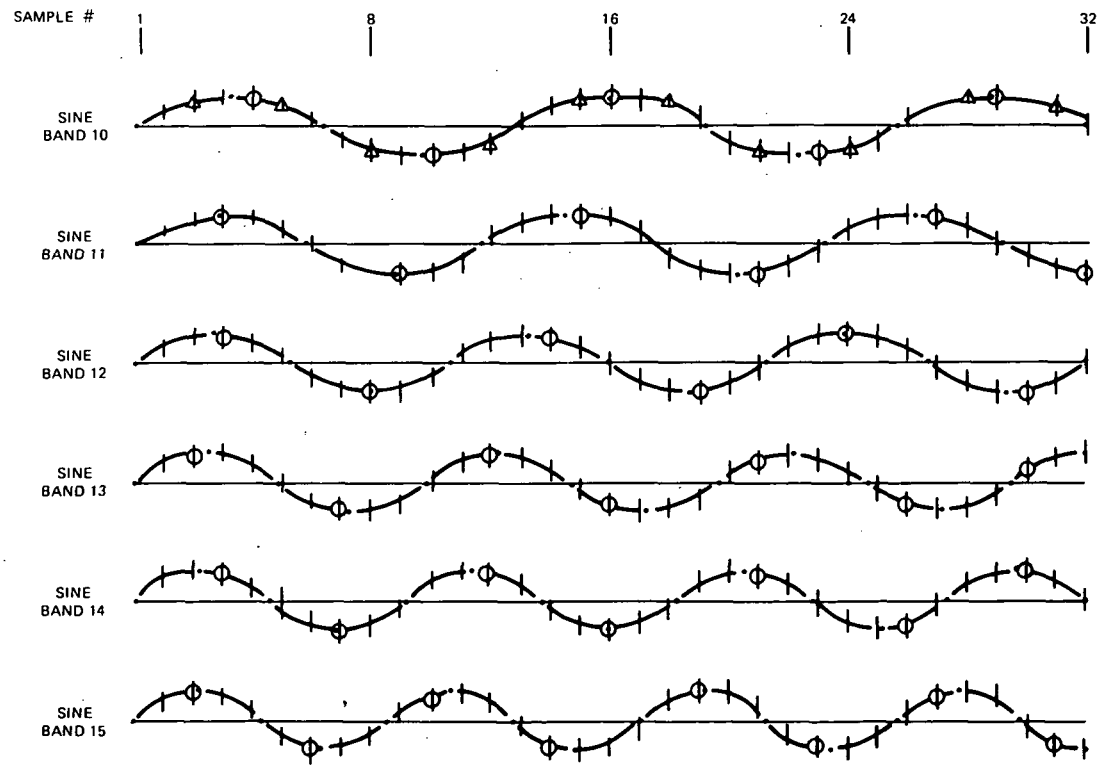
A method to reduce the alias problem in the lower bands will also be described.


1.33 ms
SOUNDER PULSE
4 → 10 ms
VIDEO
15 ms
A
15 ms
B
15 ms
D
15 ms
C


CYCLES IN SAMPLE TIME
10
11
12
13
14
15
16
17
18
19
20
21
640
704
768
832
896
960
1024
1088
1152
1216
1280
1344
TRANSLATED FREQUENCY


PRESET REGISTER TO 128, AND USE COSINE VALUE IN RECURSIVE ALGORITHM
ADD 64 COUNTS TO REGISTER
DOESN'T OVERFLOW, SO CHECK IF COSINE WOULD OVERFLOW. IT DIDN'T SO:
ADD 64 COUNTS TO REGISTER
REGISTER OVERFLOWED, SO USE VALUE IN SINE RECURSIVE ALGORITHM
ADD 64 COUNTS TO REGISTER
DIDN'T OVERFLOW, SO CHECK IF COSINE WOULD OVERFLOW. IT DIDN'T SO:
ADD 64 COUNTS TO REGISTER
REGISTER DIDN'T OVERFLOW, SO CHECK COSINE, IT WOULD OVERFLOW, SO USE VALUE IN COSINE RECURSIVE ALGORITHM
ADD 64 COUNTS
SINE OF TYPICAL BAND
IDEAL
COSINE OF TYPICAL BAND
DATA SAMPLES
4
8
12
16
20
24
28
32
SINE
NON-IDEAL
COSINE
ADD "N" COUNTS. NO OVERFLOWS OCCURRED, SO CONTINUE
ADD "N" COUNTS. OVERFLOW OCCURRED, SO MULTIPLY BY .707 FOR BOTH WAVE FORMS
ADD "N" COUNTS. OVERFLOW OCCURRED, SO USE IN SINE RECURSIVE ALGORITHM
ADD "N" COUNTS. NO OVERFLOWS, SO CONTINUE
ADD "N" COUNTS. OVERFLOW OCCURRED, SO MULTIPLY BY .707 FOR BOTH WAVE FORMS
ADD "N" COUNTS. IF NO OVERFLOW FOR SINE, CHECK FOR COSINE
PRESET REGISTER TO 128, AND USE COSINE VALUE IN RECURSIVE ALGORITHM


SAMPLE #
1
8
16
24
32
SINE BAND 16
SINE BAND 17
SINE BAND 18
SINE BAND 19
SINE BAND 20
SINE BAND 21


SAMPLE #
1
8
16
24
32
SINE BAND 10
SINE BAND 11
SINE BAND 12
SINE BAND 13
SINE BAND 14
SINE BAND 15

FAME—A MICROPROCESSOR BASED FRONT-END ANALYSIS AND MODELING ENVIRONMENT

Jacob D. Rosenbaum and Edward B. Kutin
Higher Order Software
Jericho, New York

FAME, (Front-End Analysis and Modeling Environment) is a microprocessor based interactive computer aided design aid, for designers and developers of computer-based systems. FAME is especially useful in microprocessor applications where systems complexity has increased the need for more rigorous approaches to the development process.

A variety of techniques and methods are being proposed by government and industry to meet the extensive need for better approaches to development, documentation and verification of systems. Some of the more widely used include; formal specification of system requirements and designs, static analysis of related models, and simulation to insure that the system to be developed meets the intended objective. To date, support of these activities involve large, costly computerized systems or extensive manual procedures.

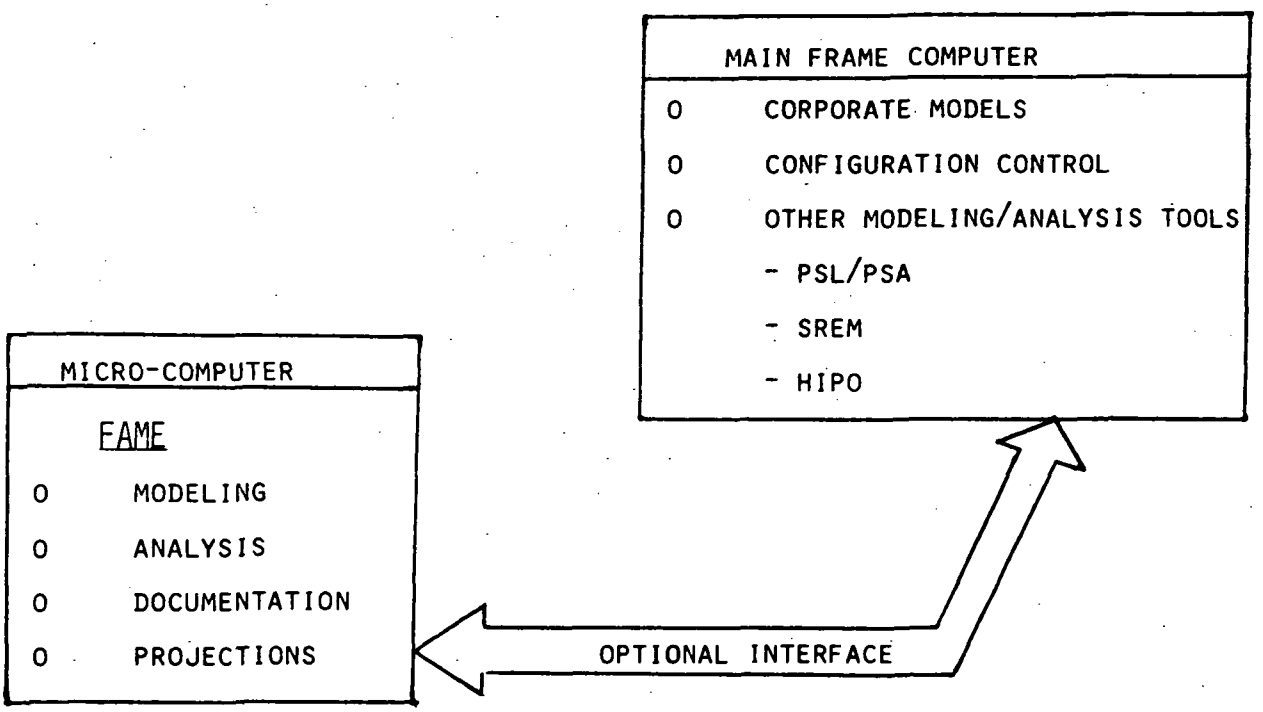
Higher Order Software (HOS) is a methodology for the specification and verification of large scale, complex, real-time systems. An emphasis in its development has been the aerospace environment. Typical systems applications include guidance and control, navigation, communications, radar imaging, satellite tracking and many others. The methodology integrates Functional Decomposition, Abstract Data Types, and Control Structures in accordance with a set of axioms that describe decomposition rules, nodal relationships and responsibilities. The systems models produced can be verified statically using small amounts of computer resources and time.

The HOS methodology has now been implemented as FAME (Front-End Analysis and Modeling Environment), a microprocessor based system for interactively developing, analyzing and displaying system models in a low cost user-friendly environment. The nature of the model is such that when completed it can be the basis for projection to a variety of forms such as Structured Design Diagrams, Petri-Nets, Data Flow Diagrams, PSL/PSA Source Code etc. The user's interface with the analyzer is easily recognized by any current user of a structured modeling approach; therefore extensive training is unnecessary. Furthermore, when all the system capabilities are used one can check on proper usage of Data Types, Functions and Control Structures and thereby add a new dimension to the design process that will lead to better, and more easily verified software designs.

FAME is now available on a range of computer systems as well as on any Microprocessor with 64K of memory and dual floppy disks that can use the CP/M operating system. It is especially useful on Microprocessor Development Systems where one would now be able to combine the requirements, documentation analysis, verification and code development activities on a single device.

Preceding Page Blank

FAME CAPABILITIES



BENEFITS OF HOS

- O MODELS ARE VERIFIABLE STATICALLY
- O MODELS INTEGRATE
  - FUNCTIONS
  - DATA TYPES
  - LIBRARIES OF OPERATIONS & STRUCTURES
  - RULES FOR MODEL CREATION
- O SUPPORTS ALL PHASES OF DEVELOPMENT
- O MODELS CAN SUPPORT DIRECT SIMULATION
- O MODELS CAN PROVIDE EXTRACTS TO SATISFY OTHER METHODS
- O AN HOS ANALYSIS REQUIRES KNOWLEDGE OF ONLY A PARENT & ITS OFFSPRING

COST BENEFITS OF FAME ON MICRO

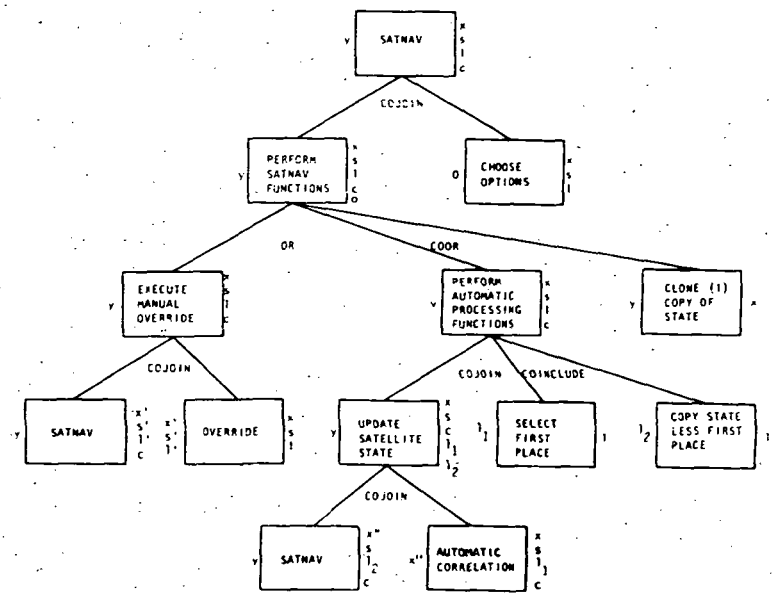
- OFFLOADS MAINFRAME T/S SYSTEMS
- PROVIDES GOOD RESPONSE @ <$4/HOUR
- MINIMUM LINE CHARGES
- HOS MODELING & ANALYSIS COSTS ESTIMATES (ON 8080 MICROCOMPUTER)

MODELING PHASE	CLOCK MINUTES/NODE	NODES INVOLVED
DECOMPOSITION	15 MIN	PARENTS
ANALYSIS	1.5 MIN	PARENTS
DISPLAY	1.0 MIN	PARENTS

CURRENT STATUS

- PROTOTYPE (C BASIC) ON 8080 MICRO-COMPUTERS (CPM)
- PROTOTYPE (C BASIC) ON INTEL MDS
- PASCAL VERSION ON MICRO (IN DEVELOPMENT)
- PASCAL VERSION ON VAX 11/780
- CURRENTLY REHOSTING TO:
  - HARRIS
  - CONTROL DATA (CYBERNET)
  - MULTICS

MODEL OF SATELLITE NAVIGATION SYSTEM



PARENT/OFFSPRING DIAGRAM

HOS/ FRONT-END ANALYSIS AND MODELING ENVIRONMENT

SATELLITE NAVIGATION SYSTEM

MODEL NAME: SATNAV
AUTHOR: EK
DATE/TIME: 7/14

OUTPUTS		INPUTS
	ABB	
YSATST	PERFORM AUTOMATIC PROCESSING FUNCTIONS	XSATST SSETIM LSETPL CSTSIM

0=PROC

ABBC	COJOIN	ABBB	COINCLUD	ABBA
UPDATE SATELLITE STATE		SELECT FIRST PLACE		COPY SET LESS FIRST PLACE
YSATST	XSATST SSETIM CSTSIM LSETPL 1 LSETPL 2	LSETPL 1 LSETPL		LSETPL 2 LSETPL

DATA			DATA TYPE--PART OF
XSATST	---> STATE OF	SATELLITES	STATE
YSATST	---> STATE OF	SATELLITES	STATE
SSETIM	---> ORDERED	SETS OF IMAGES	ORDSET
LSETPL	---> ORDERED	SETS OF PLACES	ORDSET
CSTSIM	---> ORDERED SETS OF (ORDERED SETS OF IMAGES)		ORDSET
LSETPL 2	---> ORDERED	SET OF PLACES 2	ORDSET
LSETPL 1	---> ORDERED	SET OF PLACES 1	ORDSET

CONDITION

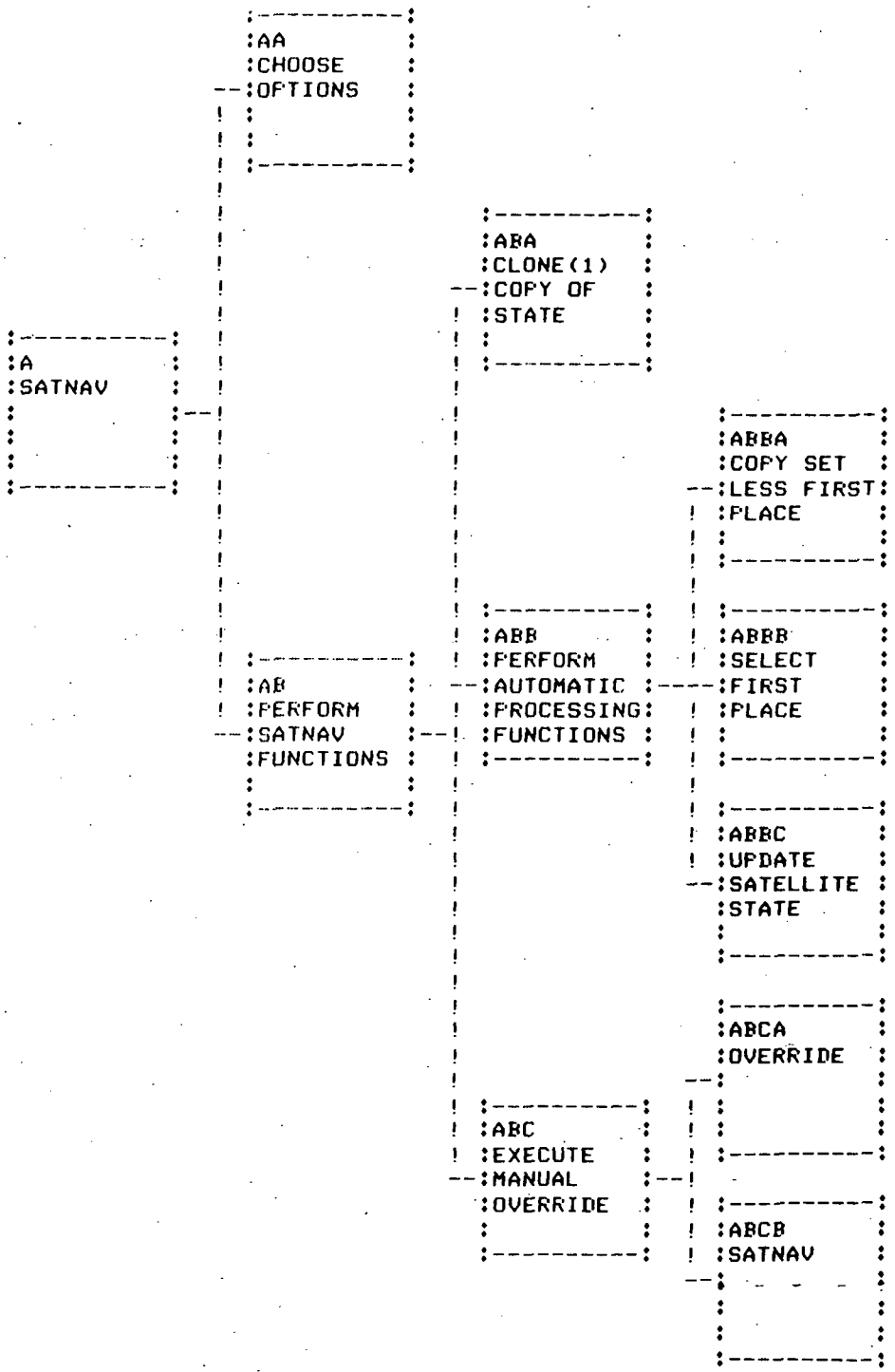
0=PROC ---> EQUAL(0,PROCEED)

A>

TYPICAL PROMPTED SCENARIO

```
HOW MANY OFFSPRING DO YOU WANT TO ADD?  2
FOR NODE  ABBCA
FN NAME (8 ): aUIOCOB_
OF NAME (8 ): ________
LONG NAME (10 ): aUIOMaIIC_
LONG NAME (10 ): COBBELaIIO
LONG NAME (10 ): *_________
CONNECTOR (8 ): ________
INPUT(8): 2_______
INPUT(8): 4_______
INPUT(8): 8_______
INPUT(8): 6_______
INPUT(8): *_______
OUTPUT(8): XSaISI''


FOR OUTPUT XSATST''
DATA TYPE(8): SIaIE___
LONG NAME(10): SIaIE_OE__
LONG NAME(10): SaIELLIIES
LONG NAME(10): *_________

OUTPUT(8): *_______
CONDITION (8 ): ________
FOR NODE  ABBCB
FN NAME (8 ): SaINaCCV
OF NAME (8 ): ________
LONG NAME (10 ): SaINaV____
LONG NAME (10 ): *_________
CONNECTOR (8 ): COJOIN__
INPUT(8): XSaISI''

INPUT(8): 4_______
INPUT(8): 10______
INPUT(8): 6_______
INPUT(8): *_______
OUTPUT(8): 1_______
OUTPUT(8): *_______
DO YOU WANT TO MAKE ANY CHANGES TO INPUTTED INFORMATION?N
FILES CLOSED---RUNNING UPDATE
PASSWORD ****

UPDATE COMPLETED
```

TREE DIAGRAM



APPLICATION OF SOFTWARE TECHNOLOGY TO A FUTURE SPACECRAFT COMPUTER DESIGN

Robert J. LaBaugh
Martin Marietta Corporation
Denver, Colorado

An Independent Research and Development task* at Martin Marietta has been investigating advanced spacecraft computer systems for the past couple of years. The task objectives are to demonstrate how major improvements in spacecraft computer systems can be obtained from recent advances in hardware** and software technology. This presentation covers the major software topics which have been addressed during the task.

Investigations into integrated circuit technology performed at the beginning of the task indicated that the CMOS/SOS chip set being developed for the Air Force Avionics Laboratory at Wright Patterson had the best potential for improving the performance of spaceborne computer systems. An integral part of the chip set is the bit slice arithmetic and logic unit (ALU). The flexibility allowed by microprogramming, combined with the software investigations described below, led to the specification of a baseline architecture and instruction set.

*This work was conducted by the Denver Division of Martin Marietta Corporation under Independent Research and Development Project Authorization D-80D.

**Related paper, "Application of Advanced Electronics to a Future Spacecraft Computer Design", in Microprocessor Hardware Technology Session.

One of the goals was to design an instruction set similar to modern minicomputer instruction sets, with multiple user registers. Another goal was to provide the throughput and precision required for flight applications, and at the same time provide an instruction set which would ease the programming of such applications. Several assembly language application programs, along with the necessary microcode, were written to help define the features to be included in the processor design. One of the areas this was used for was determining the number of registers in the system. An increase in the number of floating point registers from four to eight provided around a 10% improvement in execution time for one of the application programs. The number of registers was limited, however, by a desire to store them in the 16 physical registers internal to the ALUs. This would avoid delays in accessing the registers and help keep the parts count down. The need for scratch registers by some of the microprograms was another limiting factor. As a compromise between having as many registers as possible and the limits imposed by the ALUs, it was decided there should be seven floating point registers and eight general purpose registers. Partly to accommodate all the user registers desired, the system was designed to have two arithmetic processing units: one to handle floating point operations and store the floating point registers; and the other to handle general purpose registers, and system registers such as stack pointers and the program counter. Only one of the two processors is active at any given time. Which processor is active during a cycle is determined by means of a bit in the microword. This was done so that the processors could share the 26 bits in the microword needed for ALU control.

The precision and format of floating point operands was derived in part from the results of a study on high precision attitude computations. The main goal of the study was to characterize the drift rate of the integrator as a function of operand precision and rate sampling interval. One of the conclusions was that 32 bit floating point operands, with 24 bit mantissas, were adequate for currently envisioned projects. The study was based on data using operands with binary normalization. To remain consistent with this, we decided to use binary rather than hex normalization which can result in only 21 bits of significance in a 24 bit mantissa.

Because of the characteristics of the chip set and a desire for fairly high performance, a horizontal, rather than vertical, microword was used. There are 34 fields in the microword, which is 80 bits wide. As wide as the microword is, a fair number of fields still have to be decoded. Encoded fields are primarily used for things like selection of operand source and destination, the source for register specifications, and condition code selection. A wide microword, however, puts constraints on the number of words of control store because of the high non-recurring cost of ROMs. In an effort to keep the size of the control store within limits, and to assure adequate system performance, the microcode was developed concurrently with the circuit design. This allowed us to make various hardware versus software trades at a time when changes to the hardware design could be accomplished without too much difficulty.

Fairly early in the task an absolute assembler and an instruction set simulator were developed. These allowed the software development to proceed while the hardware design was being completed, and the hardware was being built. Langley Research Center has recently provided Pascal and HAL compiler frontends. The Pascal system includes a compiler which produces P-code, and a P-code interpreter. The HAL system consists of a compiler which produces HALMAT, a program which translates HALMAT into H-code, and an H-code interpreter. H-code is P-code with a few extra instructions and an expanded run time library. A program to translate from P-code/H-code to assembly language has been developed and Pascal and HAL routines have been executed on the instruction set simulator. The translator has undergone several refinements to improve the code generated. The initial version mimicked P-code fairly closely by keeping the expression evaluation stack in memory. By redefining register usage so that the top of the stack was kept in registers wherever possible, a 50% improvement in memory usage and execution time was achieved. Floating point push and pop instructions were also added. An investigation of a one instruction lookahead in the translation process indicated a further 12 to 14 percent improvement in memory usage and 7 to 30 percent improvement in execution time was possible. Future plans include continued investigation of P-code instruction lookahead in the translation process and further examination of the impact of high order languages on the instruction set.

BACKGROUND

OBJECTIVES:*

QUANTITATIVELY DETERMINE HOW RECENT ADVANCEMENTS IN HARDWARE** AND SOFTWARE TECHNOLOGY CAN BE USED TO OBTAIN IMPROVEMENTS IN SPACECRAFT COMPUTER CAPABILITIES.

CMOS/SOS INTEGRATED CIRCUITS
SEMI-CUSTOM LSI DEVICES
LEADLESS CARRIER PACKAGING
MICROPROGRAMMING
PASCAL, HAL, ADA, HIGHER ORDER LANGUAGE

*THIS WORK WAS CONDUCTED BY THE DENVER DIVISION UNDER INDEPENDENT RESEARCH AND DEVELOPMENT PROJECT AUTHORIZATION D-80D

**RELATED PRESENTATION, "APPLICATION OF ADVANCED ELECTRONICS TO A FUTURE SPACECRAFT COMPUTER DESIGN", IN SESSION IV: MICROPROCESSOR HARDWARE TECHNOLOGY

APPROACH

PRELIMINARY REQUIREMENTS AND IMPACT

S/W TO ASSIST IN ARCHITECTURE DESIGN

- ABSOLUTE ASSEMBLER
- INSTRUCTION SET SIMULATOR

MICROPROGRAM DESIGN

S/W DEVELOPMENT TOOLS

FEATURES REQUIRED IN PROCESSOR

MINICOMPUTER LIKE INSTRUCTION SET

MULTIPLE FLOATING POINT AND GENERAL PURPOSE REGISTERS

FLIGHT APPLICATIONS

- SUFFICIENT THROUGHPUT
- SUFFICIENT PRECISION
- EASE OF PROGRAMMING

REGISTER CONSIDERAIONS

TYPES:

- GENERAL FOR USER
- FLOATING POINT FOR USER
- SYSTEM FOR USER
- SYSTEM FOR MICROPROGRAMS

CONSTRAINTS:

- 16 PHYSICAL REGISTERS INTERNAL TO ARITHMETIC AND LOGIC UNIT DEVICES
- SEPERATE ALU DEVICES FOR FLOATING POINT

TRADES:

- PERFORMANCE SENSITIVITY TO SIZE OF REGISTER FILE
- EXTERNAL REGISTER FILE - DEGRADES PERFORMANCE

REGISTER CONSIDERATIONS

OPERAND PRECISION:

- LARGER OPERANDS REQUIRE MORE PARTS, LONGER CYCLE TIMES
- MICROCODE VS HARDWARE - SLOWER, LARGER CONTROL STORE NEEDED
- FLOATING POINT PRECISION - 32 BITS WITH BINARY NORMALIZATION SUPPORTED BY SPECIALIZED HARDWARE
- INTEGER PRECISION - 16 BIT WITH 32 BIT PERFORMED BY MICROCODE

FUNCTIONAL REGISTER UTILIZATION

16 BITS

PROGRAM COUNTER
USER STACK POINTER
PRIV STACK POINTER

GEN REG 0
GEN REG 1
GEN REG 2
GEN REG 3
GEN REG 4
GEN REG 5
GEN REG 6
GEN REG 7

32 BITS

FLOATING POINT REGISTER 1
FLOATING POINT REGISTER 2
FLOATING POINT REGISTER 3
FLOATING POINT REGISTER 4
FLOATING POINT REGISTER 5
FLOATING POINT REGISTER 6
FLOATING POINT REGISTER 7

NOTES:

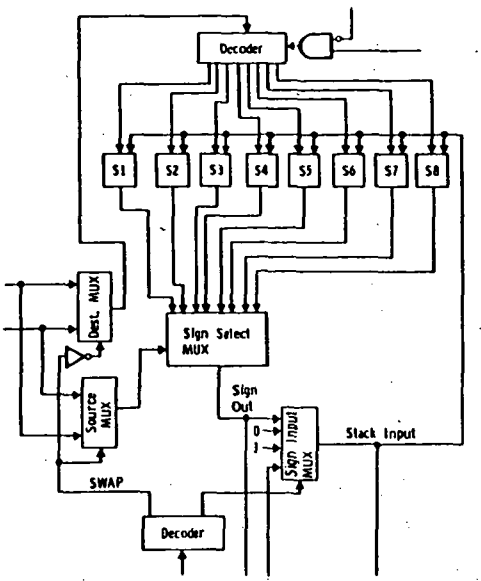
- FLOATING POINT SIGN BIT KEPT IN UNIQUE REGISTER FILE
- 5 OTHER 16-BIT REGISTERS RESERVED FOR MICROPROGRAMMER
- 1 OTHER 32-BIT REGISTER RESERVED FOR MICROPROGRAMMER
- GENERAL REGISTERS 1 - 7 CAN BE USED AS INDEX REGISTERS

EXAMPLE OF PHYSICAL REGISTER UTILIZATION

FLOATING POINT PROCESSOR

0	SCRATCH
1	FLT PT REG 1 MANTISSA
2	FLT PT REG 2 MANTISSA
3	FLT PT REG 3 MANTISSA
4	FLT PT REG 4 MANTISSA
5	FLT PT REG 5 MANTISSA
6	FLT PT REG 6 MANTISSA
7	FLT PT REG 7 MANTISSA
8	SCRATCH
9	FLT PT REG 1 EXPONENT
A	FLT PT REG 2 EXPONENT
B	FLT PT REG 3 EXPONENT
C	FLT PT REG 4 EXPONENT
D	FLT PT REG 5 EXPONENT
E	FLT PT REG 6 EXPONENT
F	FLT PT REG 7 EXPONENT

24 BITS



THE FLOATING POINT SIGN BIT FILE IS EMBEDDED IN CUSTOMIZED LOGIC

MACRO LEVEL INSTRUCTION SET

106 INSTRUCTIONS

8 CATAGORIES

- FIXED POINT
- INDEX/COUNTER REGISTER
- FLOATING POINT
- LOGICAL
- BRANCH
- STACK AND REGISTER SAVE AND RESTORE
- EXECUTIVE FUNCTIONS
- MISCELLANEOUS

10 FORMATS

- REGISTER-REGISTER
- REGISTER
- REGISTER-ADDRESS
- REGISTER-IMMEDIATE
- INDEX-REGISTER
- INDEX EXTENDED
- ADDRESS
- INDEX-ADDRESS
- SPECIAL
- SPECIAL EXTENDED

MICROPROGRAM DESIGN

HORIZONTAL RATHER THAN VERTICAL

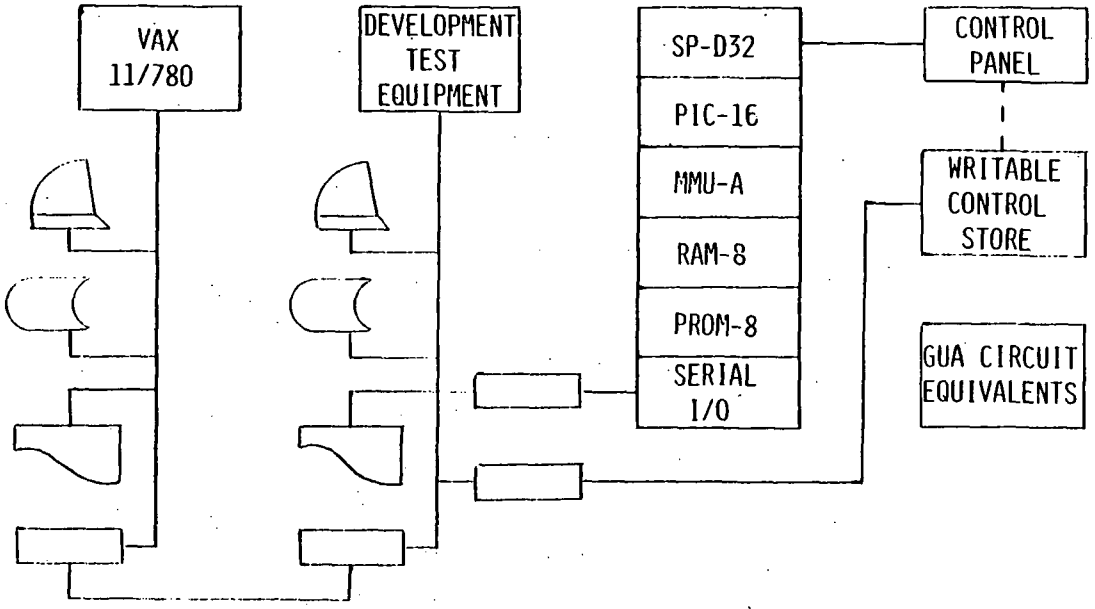
- DECODING FIELDS DEGRADES PERFORMANCE
- FORCE LOGIC TO QUIESCENT STATE WHEN NOT BEING USED
- BECAUSE OF WIDE MICROWORD, NEED TO KEEP NUMBER OF WORDS OF CONTROL STORE AS SMALL AS POSSIBLE

MICROPROGRAMS DEVELOPED CONCURRENTLY WITH HARDWARE DESIGN

- ASSURE ADEQUATE PERFORMANCE
- CONTROL STORE LIMITED BECAUSE OF HIGH NON-RECURRING COST

PRIMARY SOFTWARE MODULES & STATUS		REQMTS	DESIGN	IMPLEMENTATION
ASMA-D32:	ABSOLUTE ASSEMBLER	COMPLETE	COMPLETE	COMPLETE
ASMR-D32:	RELOCATABLE ASSEMBLER	COMPLETE	COMPLETE	COMPLETE
LNK-D32:	LINK EDITOR	COMPLETE	COMPLETE	COMPLETE
SIM-D32:	INSTRUCTION SET SIMULATOR	COMPLETE	COMPLETE	COMPLETE
PAS-LC1:	PASCAL COMPILER	COMPLETE	COMPLETE	COMPLETE
HAL-LC1:	HAL COMPILER	COMPLETE	COMPLETE	COMPLETE
RTEX:	REAL TIME EXECUTIVE	IN PROGRESS	1981	1981
SSP-D32:	SCIENTIFIC SUBROUTINE PACKAGE	IN PROGRESS	IN PROGRESS	IN PROGRESS
STD-2:	SELF TEST/DIAGNOSTIC ROUTINES	IN PROGRESS	IN PROGRESS	1981

COMPUTER DEMONSTRATION UNIT SET UP



HIGH ORDER LANGUAGE CAPABILITIES

PASCAL AND HAL COMPILERS

- FROM LANGLEY RESEARCH CENTER
- WRITTEN IN PASCAL

PATH PASCAL COMPILATION PROCESS

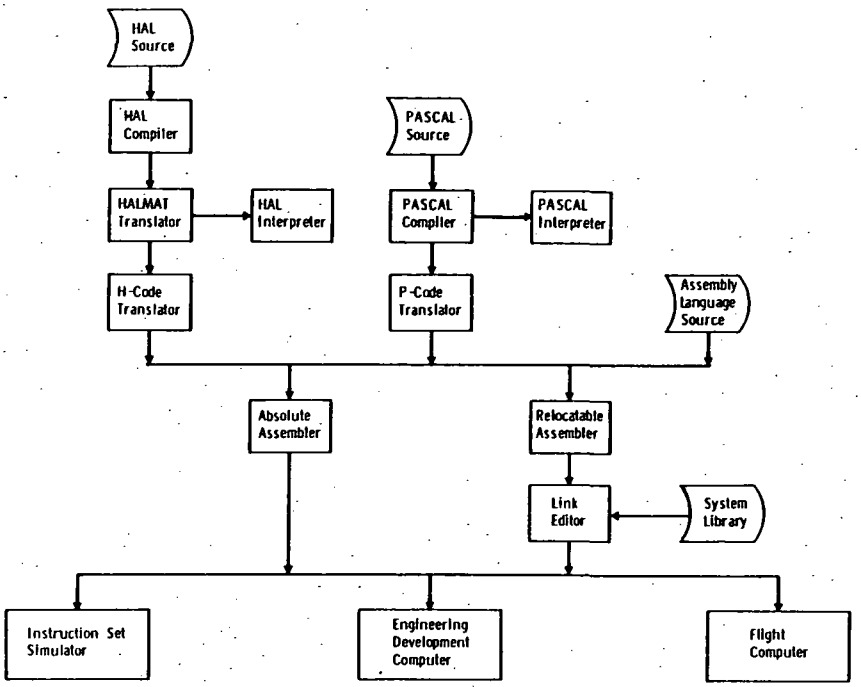
- PRODUCES P-CODE
- INTERPRETER FOR P-CODE

HAL COMPILATION PROCESS

- PHASE 1 PRODUCES HALMAT
- HALMAT TO H-CODE
- INTERPRETER FOR H-CODE

TRANSLATOR FROM P-CODE/H-CODE TO ASSEMBLY LANGUAGE

SOFTWARE PRODUCTS



TRANSLATOR REFINEMENTS

SAMPLE PROGRAMS

- CALCULATE PI TO SIX DIGITS
- BINARY SEARCH

PRELIMINARY DESIGN:

- STACK IN MEMORY

FIRST REVISION:

- TOP OF STACK KEPT IN REGISTERS
- FLOATING POINT PUSH AND POP ADDED

SECOND REVISION:

- LOOK AT TWO P-CODE INSTRUCTIONS BEFORE GENERATING CODE

PROCESSOR SOFTWARE COMPARISON

NUMERIC TEST PROGRAM - PI APPROXIMATION

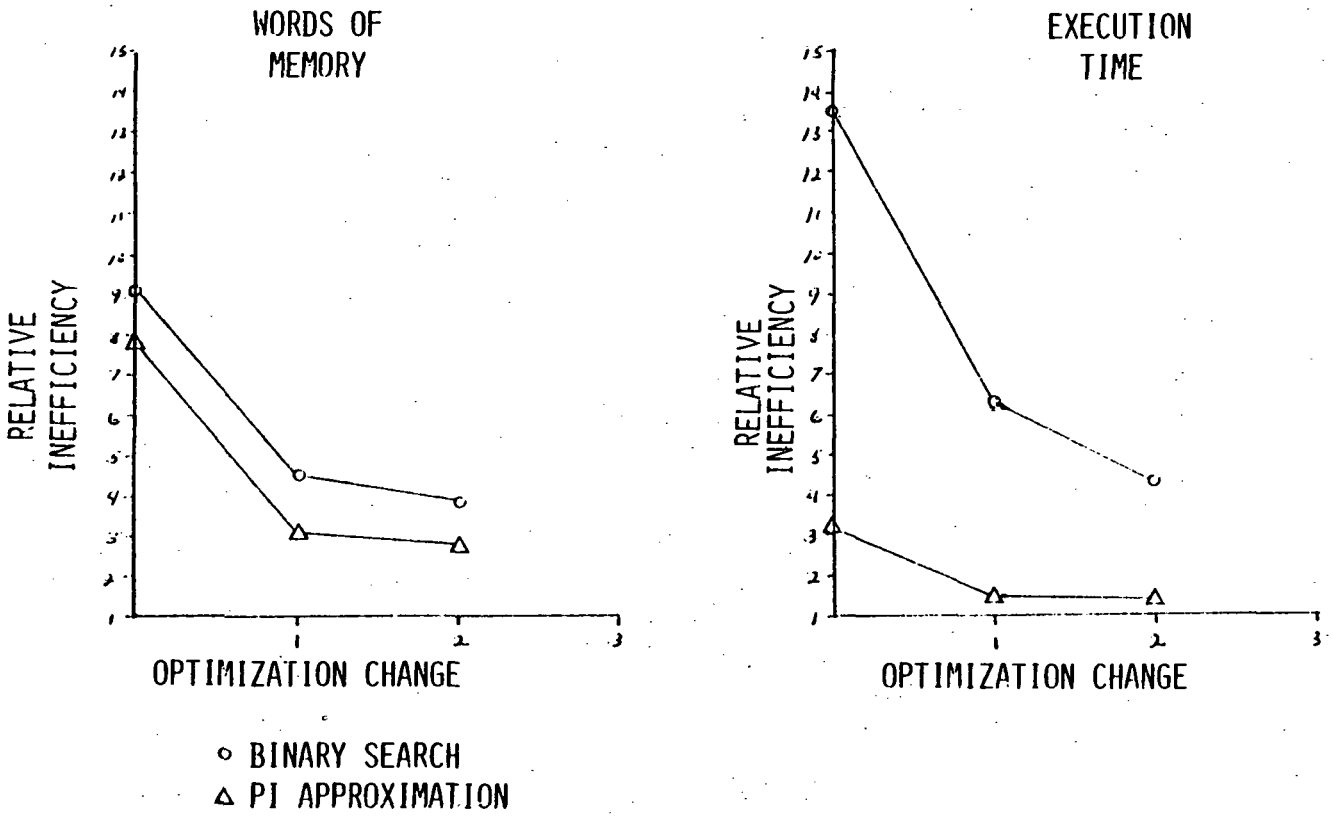
	MAC-16	PDP 11/34M	ATAC-16
ASSEMBLY LANGUAGE			
LINES OF CODE	40	37	50
WORDS OF MEMORY	74	68	79
EXECUTION TIME	11969	14909	12412
PASCAL LANGUAGE			
LINES OF CODE	28	28	-
WORDS OF MEMORY	243	347	-
EXECUTION TIME	16691	-	-
HAL LANGUAGE			
LINES OF CODE	28	-	28
WORDS OF MEMORY	254	-	157
EXECUTION TIME	16885	-	13722

PROCESSOR SOFTWARE COMPARISON

NON-NUMERIC TEST PROGRAM - BINARY SEARCH

	MAC-16	PDP 11/34M	ATAC-16
ASSEMBLY LANGUAGE			
LINES OF CODE	25	26	24
WORDS OF MEMORY	31	31	24
EXECUTION TIME	143	170	127
PASCAL LANGUAGE			
LINES OF CODE	17	17	-
WORDS OF MEMORY	138	107	-
EXECUTION TIME	886		-
HAL LANGUAGE			
LINES OF CODE	17	-	17
WORDS OF MEMORY	142	-	65
EXECUTION TIME	937	-	665

CODE GENERATOR IMPROVEMENT HISTORY



FUTURE PLANS

TRANSLATOR:

- MULTI P-CODE INSTRUCTION LOOKAHEAD
- MULTI PASS - OPTIMIZE REGISTER USAGE

DIRECT HALMAT TO ASSEMBLY LANGUAGE CONVERSION
BASED ON HALMAT TO H-CODE PROGRAM

CONTINUE EXAMINING HOL IMPACT ON INSTRUCTION SET

A TRANSLATOR WRITING SYSTEM FOR MICROCOMPUTER HIGH-LEVEL LANGUAGES AND ASSEMBLERS

W. Robert Collins*
Computer Sciences Corporation
Hampton, Virginia

John C. Knight
Langley Research Center
Hampton, Virginia

and

Robert E. Noonan**
College of William and Mary
Williamsburg, Virginia

NASA LaRC uses many dedicated microprocessors in aerospace research. Few software tools are available for these machines, and in particular, very few have any form of high-level language facility. Since the Langley environment involves considerable experimentation, a great deal of software is experimental and may change frequently. It has to be prepared relatively quickly and at low cost.

In order to implement high-level languages whenever possible, a Translator Writing System of advanced design has been developed. It is intended for routine production use by many programmers working on different projects. As well as a fairly conventional parser generator, it includes a system for the rapid generation of table driven code generators. This code generation system is the result of research performed at the College of William and Mary under NASA sponsorship. The parser generator was developed from a prototype version written at the College of William and Mary.

The Translator Writing System includes various tools for the management of the source text of a compiler under construction. In addition, it supplies various "default" source code sections so that its output is always compilable and executable. The system thereby encourages iterative enhancement as a development methodology by ensuring an executable program from the earliest stages of a compiler development project.

This presentation will describe the Translator Writing System and some of its applications. These include the PASCAL/48 compiler, three assemblers, and two compilers for a subset of HAL/S. PASCAL/48 is a Pascal-like language for the Intel-8748 microcomputer. The assemblers which have been built are for assembly language subsets for the Intel-8080, the Motorola M68000, and the NSSC-II. The HAL/S subset was implemented for the Intel-8080 and the GE 703. Detailed measurements of the use of the system to build the code generators for the HAL/S compilers will be given.

*Work performed under NASA contract numbers NAS1-14900 and NAS1-16078.
**Work performed under NASA grant number NSG-1435.

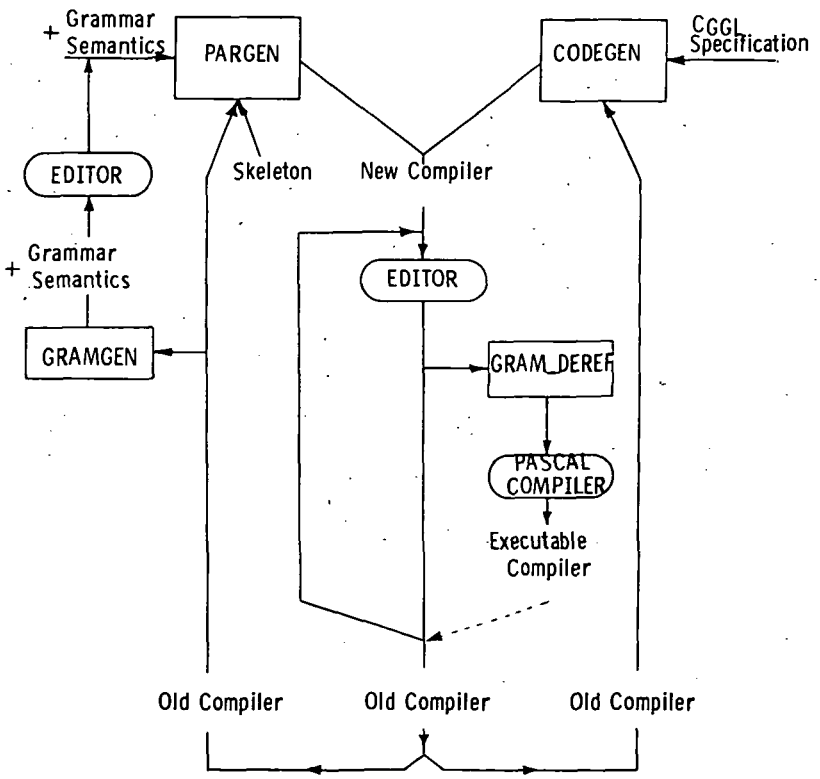
THE PROBLEM

- NEED HIGH-LEVEL LANGUAGES, HENCE COMPILERS
- NEED ASSEMBLERS
- ONE SOLUTION IS A TWS

TWS CRITERIA

- ENCOURAGE ITERATIVE ENHANCEMENT
  - EARLIEST POSSIBLE EXECUTION
  - TEXT MANAGEMENT TO RELIEVE TEDIUM
- FLEXIBILITY IN ITS USE
- TRANSPORTABLE IMPLEMENTATION

TRANSLATOR WRITING SYSTEM



USE OF TWS

1. IF PARSER NEEDED, RUN PARGEN, EXECUTE RESULTING COMPILER TO TEST.
2. CHANGE GRAMMAR AS NECESSARY, RERUN PARGEN.
3. ADD SEMANTICS USING EDITOR.
4. RECOVER GRAMMAR AND SEMANTICS WITH GRAMGEN IF NECESSARY TO RERUN PARGEN.
5. IF CODE GENERATION NEEDED, PREPARE CGGL SPECIFICATION AND RUN CODGEN.
6. MODIFY CGGL AS NEEDED.
7. ITERATE THROUGH ABOVE STEPS ADDING LANGUAGE FEATURES AS DESIRED.

PARGEN

- INPUTS
  - GRAMMAR IN STANDARD BNF
  - SEMANTICS IN PASCAL
  - SKELETON OR OLD COMPILER
- OUTPUT IS AN EXECUTABLE COMPILER INCLUDING
  - SCANNER
  - LALR (1) PARSER
  - SEMANTICS ROUTINE
- TEXT MANAGER PRESERVES PROGRAMMER'S CONTRIBUTION TO COMPILER E.G., SYMBOL TABLE ROUTINES

CODGEN

- INPUTS
  - CGGL SPECIFICATION
  - SKELETON OR OLD COMPILER
- OUTPUT IS AN EXECUTABLE COMPILER INCLUDING A CODE GENERATOR.
- CGGL IS A NON-PROCEDURAL LANGUAGE FOR DESCRIBING THE CODE-GENERATION PROCESS.
- TEXT MANAGER PRESERVES PROGRAMMER'S CONTRIBUTION TO COMPILER E.G., MACHINE LANGUAGE FORMATTER.

PASCAL/48

- INTEL-8748
  - MICROCOMPUTER
  - 8-BIT CPU
  - 64 WORD RAM
  - 1024 WORD ROM
  - 27 I/O LINES
- PASCAL/48
  - PASCAL DERIVATIVE FOR 8748
  - EXTENSIONS TO ALLOW CONTROL OVER GENERATED CODE
  - RESTRICTIONS TO PROHIBIT INEFFICIENT FEATURES
  - COMPILER AVAILABLE ON CDC CYBERS

ASSEMBLERS

- CUSTOMIZED SKELETON FOR ASSEMBLERS
  - TWO PASSES
  - STANDARD LISTING BY DEFAULT
  - FLEXIBLE INPUT FORMAT CONVENTIONS
  - HANDLES MACROS WITHOUT PARAMETERS
- COMPARED TO META-ASSEMBLER, ASSEMBLER BUILT FOR NSSC-II
  - WAS PRODUCED MORE QUICKLY
  - EXECUTES 5 TIMES FASTER
  - USES ONE FOURTH THE SPACE

EXAMPLE PASCAL/48 PROGRAM

```
NASA, LANGLEY RESEARCH CENTER      80/08/26.      08.51.12.              PAGE  1
PASCAL 8748C    VERSION 1.0.0    PROGRAM   MAIN                        CSC/NASA

  1 PROGRAM FOR_YOU;
  2
  3    VAR      I[2] : INTEGER;
  4             A[16_300, ROM] : ARRAY [100] OF INTEGER;
  5
  6    VALUE    A = (99 OF 0, 1);
  7
  8
  9    PROCEDURE GET_INPUT;
 10    BEGIN
 11       REPEAT
 12       UNTIL PORT1 BIT 3
 13    END;  (* GET_INPUT *)
 14
 15
 16 BEGIN  (* PROGRAM FOR_YOU *)
 17    FOR I := 100 DOWNTO 1 DO
 18       BEGIN
 19       GET_INPUT;
 20       PORT1 := PORT1 AND 2_11100011;
 21       PORT2 := A[I] + PORT1 XOR I
 22       END  (* FOR I := 100 DOWNTO 1 DO BEGIN *)
 23 END.  (* PROGRAM FOR_YOU *)
```

GENERATED CODE FOR EXAMPLE PROGRAM

```
        JMP     L009
        NOP
L003:   JMP     L003
        NOP
        NOP
L007:   JMP     L007
L009:   CLR     A
        MOV     PSW,A
        JMP     L012
L00D:   IN      A,P1            ; LINE   9
        CPL     A               ; LINE  13
        JB3     L00D            ; LINE  13
        RET                     ; LINE  13
L012:   MOV     R2,#99          ; LINE  14
L014:   CALL    L00D            ; LINE  18
        ANL     P1,#227         ; LINE  21
        IN      A,P1            ; LINE  22
        MOV     R1,A            ; LINE  22
        MOV     A,P2            ; LINE  22
        MOVP3   A,@A            ; LINE  22
        ADD     A,R1            ; LINE  22
        XRL     A,R2            ; LINE  22
        OUTL    P2,A            ; LINE  22
        DJNZ    R2,L014         ; LINE  22
```

SEPARATE CODE GENERATION USING CGGL

LANGUAGE: HAL/S

INTERMEDIATE CODE LANGUAGE: HALMAT

- 178 OPERATORS TOTAL
- 30 OPERATORS IMPLEMENTED
- 25 GENERATE CODE
- BASICALLY AN INTEGER SUBSET WITH SIMPLE CONTROL STRUCTURES

CODE GENERATORS

- ONE PASS
- NO PRE-OPTIMIZATION PASS
- NO PEEPHOLE OPTIMIZATION
- INTEL 8080, GE 703

Implementations

Intel 8080

- 8 bit machine
- single accumulator
- no index register
- 1, 2, 3 byte instructions
- hardware stack
- only integer add, subtract
- 16 bit addresses

GE 703

- 16 bit machine
- single accumulation
- index register
- one word instructions
- no hardware stack
- integer add, subtract, multiply, divide
- only address current page, page zero
- page: 256 words

703 Code Generator

Task	Time (days)
Reading manual	.5
CGGL program	1.5
Writing Pascal routines	1.5
Debugging	1.0
	4.5 days

Notes:

1. All programs were coded and keyed by Noonan.
2. Some of debugging time was used in cleanup.
3. One debugging run was used to fix a bug introduced by cleanup.
4. A total of 6 runs (execution) were used.
5. One CGGL bug.

703 Implementation

Source of Code	No. Procedures	% Lines	% Instr. Storage
8080 Imple.	46	58%	58%
Modified 8080	4	8%	6%
Noonan	9	10%	10%
CGGL	1	24%	26%

Notes:

1. CGGL program: 292 lines
2. Pascal program: 890 lines
3. For an earlier non-table-driven implementation, CGGL accounted for 83% of lines and 77% of storage.

SESSION IV

MICROPROCESSOR HARDWARE TECHNOLOGY

A HIGH PERFORMANCE MULTIPLIER PROCESSOR FOR USE WITH AEROSPACE MICROCOMPUTERS

P. E. Pierce
Sandia National Laboratories
Albuquerque, New Mexico

An MC68000-based microcomputer including a hardware multiplier processor has been designed and prototyped for a re-entry vehicle navigation and control application. In this paper, the microcomputer is discussed with emphasis on the multiplier processor architecture, software control and theory of operation.

The MC68000 CPU of the microcomputer cannot satisfy the real-time multiply processing requirements of a high accuracy RV navigator. The standalone CPU thru-put for multiply intensive applications is increased approximately seven times by the addition of a board level Hardware Multiplier Processor (HMP). Although the HMP was designed for the MC68000 microcomputer, it can be used with any 16 or 32 bit CPU with minimal modifications.

The memory mapped HMP performs 16 and 32 bit multiplications and can optionally add or subtract the full product to previous accumulator contents. The circuitry is sufficiently fast to allow the MC68000 running at 8 MHz to write single or double precision variables to the HMP using memory to memory transfers and perform an operation with no wait states introduced or overhead time for command passing.

The result of multiply and accumulate operations may be transferred in its entirety or scaled by $2^{\pm 30}$ and rounded automatically prior to transfer to the destination location specified by the CPU. Worst case CPU wait times introduced are: 3.3 μsec for double precision scale by 2^{-14} and round to single precision; and 6.3 μsec for quadruple precision scale by 2^{-30} and round to double precision.

The Hardware Multiplier Processor incorporates Serial/Parallel Hardware Multiplier ICs, a translation PROM and address controlled logic to implement previously mentioned arithmetic functions. The use of serial arithmetic circuitry yields a processor of small physical size, low power and significant flexibility. The computation time of the HMP is shorter than most of the general memory addressing modes of the host CPU. The nine least significant CPU address bits in conjunction with the translation PROM control all HMP functions. The translation PROM provides the function related serial clock count to the clock control logic which in turn controls all HMP timing.

Preceding Page Blank

SANDIA AEROSPACE COMPUTER VERSION 4
(SANDAC IV)

ARCHITECTURE

MC68000 CPU
- 32 BIT DATA AND ADDRESS REGISTERS
- 56 INSTRUCTIONS
- 14 ADDRESSING MODES
- MEMORY MAPPED I/O

16 BIT DATA BUS
16 M BYTE ADDRESS SPACE
HARDWARE MULTIPLIER PROCESSOR (HMP)
VECTORED INTERRUPTS

POWER REQUIREMENTS

+5V @ 3A TYPICAL

PHYSICAL

EXPANDABLE MODULAR CONSTRUCTION
STACKABLE PIN-SOCKET INTERMODULE BUS
17.8 CM x 15.9 CM x 1.27 CM MODULES

SANDAC IV CPU MODULE

MC68000 CPU
16K BYTE EPROM MEMORY
16K BYTE NON-VOLATILE CMOS RAM
POWER MONITOR & RESET CIRCUIT

SANDAC IV I/O MODULE

4 CHANNEL OPTO-ISOLATED USART SERIAL I/O
8 CHANNEL PRIORITY INTERRUPT CONTROLLER
5 CHANNEL PROGRAMMABLE 16 BIT TIMER/COUNTER
16 BIT MEMORY MAPPED (4K WORD) I/O

SANDAC IV HMP MODULE

MEMORY MAPPED REGISTERS AND FUNCTIONS

SINGLE PRECISION (16 BIT) AND DOUBLE PRECISION (32 BIT) FUNCTIONS

MULTIPLY WITH OPTIONAL ADD OR SUBTRACT TO PREVIOUS ACCUMULATOR CONTENTS

SCALE $2^{\pm N}$ AND ROUND

OVERFLOW DETECTION RELATING TO ACCUMULATION

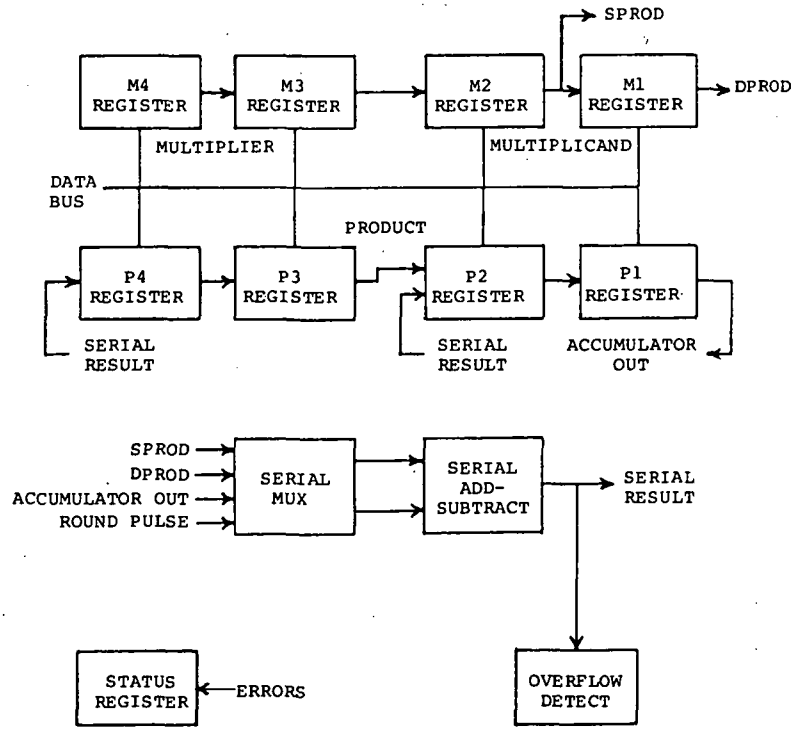
ADDRESSING ERROR DETECTION

CONTROL FUNCTIONS DERIVED FROM LATCHED ADDRESS BITS

HOST CPU ALLOWED TO PROCEED IN PARALLEL WITH HMP

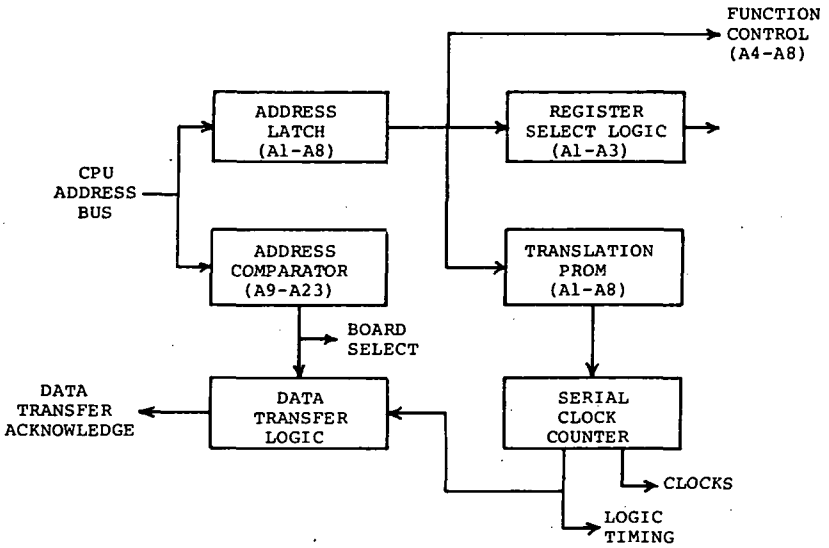
AUTOMATIC HOLD-OFF OF HOST CPU IF HMP BUSY

HARDWARE MULTIPLIER PROCESSOR
BLOCK DIAGRAM



HARDWARE MULTIPLIER PROCESSOR

CONTROL BLOCK DIAGRAM



EXAMPLE HMP ADDRESS MAPPED FUNCTIONS

ADDRESS (HEX)	FUNCTION
FE00	READ-CLEAR STATUS REGISTER
FE02	READ/WRITE P4
FE04	READ/WRITE P3
FE06	READ/WRITE P2
FE08	READ/WRITE P1
FE0A	READ/WRITE M4
FE0C	READ/WRITE M3
FE0E	WRITE M2
FE0E	READ STATUS REGISTER
FE1E	WRITE M2, S.P. MULTIPLY & ADD
FE3E	WRITE M2, CLEAR ACCUM., S.P. MULT. & ADD
FE5E	WRITE M2, S.P. MULTIPLY & SUBTRACT
FE7E	WRITE M2, CLEAR ACCUM., S.P. MULT. & SUB.
FE8E	WRITE M2
FE90	WRITE M1, D.P. MULTIPLY & ADD
⋮	⋮

EXAMPLE HMP ADDRESS MAPPED FUNCTIONS

ADDRESS (HEX)	FUNCTION
FF00	D.P. P REG x 2^{-14} & S.P. ROUND
⋮	⋮
FF1A	D.P. P REG x 2^{-1} & S.P. ROUND
FF1C	D.P. P REG x 2^{0} & S.P. ROUND
FF1E	D.P. P REG x 2^{1} & S.P. ROUND
⋮	⋮
FF38	D.P. P REG x 2^{14} & S.P. ROUND
FF80	Q.P. P REG x 2^{-30} & D.P. ROUND
⋮	⋮
FFBA	Q.P. P REG x 2^{-1} & D.P. ROUND
FFBC	Q.P. P REG x 2^{0} & D.P. ROUND
FFBE	Q.P. P REG x 2^{1} & D.P. ROUND
⋮	⋮
FFF8	Q.P. P REG x 2^{30} & D.P. ROUND

FUNCTION EXECUTION TIME

FUNCTION	EXECUTION TIME
S.P. MULTIPLY & ACCUMULATE	2.38 μs
D.P. MULTIPLY & ACCUMULATE	4.38 μs
D.P. SCALE 2^{-14} & S.P. ROUND	3.31 μs
D.P. SCALE 2^{0} & S.P. ROUND	2.44 μs
D.P. SCALE 2^{14} & S.P. ROUND	1.56 μs
Q.P. SCALE 2^{-30} & D.P. ROUND	6.31 μs
Q.P. SCALE 2^{0} & D.P. ROUND	4.44 μs
Q.P. SCALE 2^{30} & D.P. ROUND	2.56 μs

SANDAC IV BENCHMARK EQUATION

$$A_{11} = B_{11}C_{11} + B_{12}C_{21} + B_{13}C_{31} + K$$

NOTE: ALL TERMS ARE 32 BIT FIXED POINT.

CONFIGURATION	EXECUTION TIME
MC68000 CPU @ 8 MHZ (SUBROUTINE SOLUTION)	235 μs
MC68000 CPU @ 8 MHZ + HMP (HMP SOLUTION)	31 μs

BENCHMARK EQUATION MACRO INSTRUCTION SOLUTION

$$A_{11} = B_{11}C_{11} + B_{12}C_{21} + B_{13}C_{31} + K$$

SOURCE CODE:

```
LQPP K              /LOAD Q.P. CONSTANT
DPMA B11, C11       /D.P. MULTIPLY & ADD
DPMA B12, C21       /D.P. MULTIPLY & ADD
DPMA B13, C31       /D.P. MULTIPLY & ADD
DPSRM 0, A11        /QUAD P. SCALE, ROUND & MOVE
```

BENCHMARK EQUATION MACRO EXPANSION

$$A_{11} = B_{11}C_{11} + B_{12}C_{21} + B_{13}C_{31} + K$$

ASSEMBLER EXPANSION:

MACRO	MC68000 MNEMONICS	COMMENT
LQPP #0, K		/LOAD Q.P. CONSTANT
	MOVE.L 0, FE02	
	MOVE.L K, FE06	
DPMA B_{11}, C_{11}		/D.P. MULTIPLY & ADD
	MOVE.L B_{11}, FE0A	
	MOVE.L C_{11}, FE8E	
DPMA B_{12}, C_{21}		/D.P. MULTIPLY & ADD
	MOVE.L B_{12}, FE0A	
	MOVE.L C_{21}, FE8E	
DPMA B_{13}, C_{31}		/D.P. MULTIPLY & ADD
	MOVE.L B_{13}, FE0A	
	MOVE.L C_{31}, FE8E	
DPSRM 0, A_{11}		/QUAD P. SCALE, ROUND & MOVE
	MOVE.L FFBC, A_{11}	

SUMMARY

EFFECTIVELY EXPANDS HOST CPU INSTRUCTION SET

EASY INCORPORATION INTO ANY 16 BIT SYSTEM

HIGH PERFORMANCE DUE TO SIMULTANEOUS DATA & COMMAND TRANSFER BY HOST CPU

SERIAL ARITHMETIC APPROACH REDUCES COMPONENT COUNT

EQUATION EXECUTION TIME PRIMARILY DEPENDENT ON CPU MEMORY ACCESS TIME

STRAIGHT FORWARD SOFTWARE CONTROL

SINGLE μP CPU PLUS HMP PROVIDES PERFORMANCE COMPARABLE TO BIPOLAR BIT-SLICE DESIGNS

APPLICATION OF ADVANCED ELECTRONICS TO A FUTURE SPACECRAFT COMPUTER DESIGN

Philip C. Carney
Martin Marietta Corporation
Denver, Colorado

Over the past two and one half years at Martin Marietta, an Independent Research and Development task* has been conducting an extensive investigation on advanced spacecraft computer systems. The task objectives are to quantitatively determine how recent advancements in hardware and software** technology can be used to obtain major improvements in spacecraft computer system capabilities. This paper describes the major hardware aspects which have been investigated and what results have been obtained.

---

* This work was conducted by the Denver Division of Martin Marietta Corporation under Independent Research and Development Project Authorization D-80D.

** Related paper, "Application of Software Technology to a Future Spacecraft Computer Design, in Microprocessor Software Technology Session.

During 1978 several architecture trade studies were conducted to arrive at a decision on what characteristics a future spacecraft computer should have. The architecture trade studies went through two phases; the implementation independent phase and the chip set dependent phase. Some of the major factors which were included during the first phase were:

- Data types and precision,
- Processing throughput,
- Input and output,
- Address space,
- Instruction types,
- Multiprogramming features, and
- Test support equipment.

In general terms, a modern minicomputer-like architecture with a 250,000 operations per second performance and the lowest possible power consumption was desired. It was noticed almost immediately that in some cases it was difficult to define certain architecture features because they appeared to be application dependent. For example, input and output (I/O) appears to vary depending on the spacecraft data bus. To allow for this variation but not impede progress it was decided to use memory mapped I/O. This approach, insures that sufficient flexibility is maintained in the architecture for most applications. In a related area a slightly different approach was taken I/O is normally handled in one of two ways, register transfers or direct memory access (DMA). If the direct memory access was imbedded in each I/O controller than each application, each would be burdened with the nonrecurring cost of DMA. To avoid this situation we incorporated a generalized Direct Memory Processor which can buffer and transfer data between a memory mapped I/O controller and main memory, Hence there is only a one time nonrecurring design cost associated with this feature of the computer.

In 1978 integrated circuits technology was reviewed. Of particular interest in this area was the feasibility of using Complementary Metal Oxide Semiconductor/Silicon on Saphire (CMOS/SOS) parts. The primary reason for interest in CMOS/SOS was the favorable speed/power ratio which is important in spaceborne applications. The candidate CMOS/SOS chip sets which were

investigated include: custom devices developed by NASA Marshall Space Flight Center and Air Force Space Division (formerly SAMSO); parts developed under the Space Division Advanced Computer Technology program (ACT-I); 290X parts under development at Raytheon; and a microprocessor chip set being developed at RCA under contract to the Air Force Avionics Laboratory at Wright Patterson (AFWAL). We elimated the use of custom parts early in our selection process because of their high cost. The ACT and Raytheon parts were eliminated because a comprehensive family of support components was not available. It was felt, therefore, that the best potential CMOS/SOS technology for use in spaceborne applications was the AFWAL/RCA microprocessor chip set. In addition to being a comprehensive family of LSI devices, the units are being produced in a radiation hardened process with high reliability screening. Recently the Global Positioning System (GPS) program has selected this chip set for use giving further evidence that our decision to base our design on the AFWAL/RCA chip set was appropriate.

The LSI devices available in the RCA chip set are: the TCS 129 General Processing Unit (GPU), the TCS 196 Multiplier, the TCS 09X Gate Universal Array (GUA), the TCS 150 Random Access Memory (RAM), the TCS 075 Read Only Memory, and the TCS 158 Microprogram Controller Unit. The GPU forms the foundation for the entire chip set. It is an eight bit wide arithmetic and logic slice that can be cascaded to form an arbitrarily wide data word. The TCS 196, unlike other multiplier chips presently available, is also a cascadable device. All of the partial product logic is included on the TCS 196 to form an N x M multiplier without the need for supporting hardware logic. The purpose of the TCS 09X GUA is to provide the logic and circuitry which in most other chip set families is found in small and medium scale integrated circuits. These arrays are fixed regular patterns of transistors and routing paths. By defining transistor interconnection, the GUA is customized with logic in much the same way that Read Only Memory is customized with data. Some of the benefits associated with GUAs are improved speed and real estate requirements; disadvantages include higher nonrecurring cost and greater design risk. Martin Marietta has thus far designed three Gate Universal Arrays and testing has shown that the devices are functionally correct and exceed performance expections.

Once the first phase of the architecture analysis was completed, the factors associated with use of the CMOS/SOS microprocessor chip set had to be taken into account. Some of these factors were:

- Number of user available registers,
- High speed arithmetic support hardware,
- Instruction decoding,
- Data formatting and deformatting,
- Memory volatility, and
- Interrupt handling.

It is important to mention at this time that software* analysis as well as circuit analysis played a significant part at this time in determining what the final characteristics of the computer would be. For example the fact that we desired a modern minicomputer-like architecture implied that multiple general purpose registers were required. The number of registers however could only be determined by coding application programs, measuring the resulting throughput, and performing the physical circuit design to determine what the impact was in terms of hardware cost and complexity.

Throughout the first half of 1979 we performed many paper emulations in which we wrote application programs, wrote microprograms, and prepared circuit layouts. These paper emulations allowed us to determine specific advantages and disadvantages inherent in different architectures implemented with the CMOS/SOS microprocessor chip set. In midyear we chose the "best" architecture for implementation. Best is a subjective term which in some cases such as performance and power consumption can be determined quantitatively, but in other cases such as flexibility and risk it can only be measured qualitatively.

After the selection process was completed, detailed design began with the goal of having an operational demonstration unit in 1980. The most challenging part of the detailed design was the many unknowns associated with using a microprocessor chip set which was still under development at that time. Our three primary concerns were internal device performance, off-chip drive

---

* Op. Cit.

capability, and Gate Universal Array layout. The first concern was handled by derating devices a minimum of 100% using the limited test data available. Testing at Martin Marietta has since shown that all devices are much better than were originally anticipated. Our second concern, off-chip drive capability, was caused by the capacitance loading problems associated with CMOS technology. This problem becomes particularly severe in the area of the main memory interface. To eliminate the concern we incorporated a bulk hardened signal level converter to drive the memory bus at TTL levels. The addition of the signal level converter causes an added latency along the bus but this is a much better situation than that which would have occurred if an attempt had been made to fanout the CMOS signals. Our remaining concern, GUA design, was handled very conservatively. TTL circuit equivalents were built for each array, extensive software logic simulations were conducted, and the entire data submittal package was independently verified before shipment to RCA. As a result of this effort, no Martin Marietta caused problems have been found in any of the three designs which we have completed. There were some issues in the testing area which arose because of Martin Marietta's and RCA's lack of experience with the chip set. These have since been resolved by modifying our software so that test vector data is automatically generated during the logic simulation.

Throughout 1980 efforts have been directed toward fabrication of a breadboard unit which demonstrates the primary computer modules: the central processor module, the priority interrupt controller, the memory bus driver/receiver, and an 8K RAM memory bank. Additionally an operator control panel, a writable control store, a programmable read only memory bank, and a serial I/O port were implemented to facilitate development. The resulting computer has been operational for several months and many of the major design goals have already been demonstrated. Functionality, performance and power consumption meet our expectations.

In 1980 concurrent with our breadboard fabrication we took the first step towards producing a qualified flight unit. This step was the fabrication and testing of mockup printed circuit boards using leadless carrier packaging technology. Although the Orlando Division of Martin Marietta has over three years experience in the use of leadless carriers, we felt it was necessary to

perform some specific tests to eliminate the controversy surrounding the use of this technology in spaceborne applications. As a result of qualification level thermal and vibration tests, we have found that leadless carriers mounted on polyimide daughter boards which are in turn properly mounted on larger mother boards is an appropriate packaging approach.

In 1981 we intend to extend our work by taking the primary hardware modules now in wire wrap form and converting these to printed circuit boards. This effort will have two major benefits. First, it will allow us to show what performance margins exist at the system level. Performance margin cannot be demonstrated on the breadboard because of the high capacitance associated with the wire wrap technique. The second major benefit to design and fabricating the printed circuit boards is that we will be able to perform thermal and vibration qualification level testing on a unit which much more closely resembles the final flight unit.

Although the AFWAL/RCA CMOS/SOS microprocessor chip set was originally considered to be a high risk item, the fact that parts have been produced, tested and used in our computer system and the GPS program indicate that these parts will have a favorable future. The results we have obtained show that a CMOS/SOS computer can obtain the same performance levels as presently available bipolar spacecraft computer but at approximately 15% of their power consumption.

BACKGROUND

OBJECTIVES:*

QUANTITATIVELY DETERMINE HOW RECENT ADVANCEMENTS IN HARDWARE AND SOFTWARE** TECHNOLOGY CAN BE USED TO OBTAIN IMPROVEMENTS IN SPACECRAFT COMPUTER CAPABILITIES.

CMOS/SOS INTEGRATED CIRCUITS
SEMI-CUSTOM LSI DEVICES
LEADLESS CARRIER PACKAGING
MICROPROGRAMMING
PASCAL, HAL, ADA, HIGER ORDER LANGUAGE

*THIS WORK WAS CONDUCTED BY THE DENVER DIVISION UNDER INDEPENDENT RESEARCH AND DEVELOPMENT PROJECT AUTHORIZATION D-80D

**RELATED PRESENTATION, "APPLICATION OF SOFTWARE TECHNOLOGY TO A FUTURE SPACECRAFT COMPUTER DESIGN", IN SESSION III: MICROPROCESSOR SOFTWARE TECHNOLOGY

BACKGROUND

APPROACH:

1978 REVIEW AVAILABLE STATE OF THE ART TECHNOLOGY
DEFINE CANDIDATE ARCHITECTURES

1979 PERFORM DESIGN OF ARCHITECTURES AND USE "PAPER EMULATION" TO OBTAIN QUANTITATIVE RESULTS
SELECT "BEST" ARCHITECTURE

1980 DEMONSTRATE SYSTEM USING BREADBOARD

1981 BUILD AND TEST BRASSBOARD

AVAILABLE CMOS/SOS INTEGRATED CIRCUITS

MSFC AND SAMAO CUSTOM DEVICES

SAMSO ADVANCED COMPUTER TECHNOLOGY PROGRAM

RAYTHEON 290X RESEARCH PROGRAM

AIR FORCE MATERIALS LAB/RCA MICROPROCESSOR CHIP SET

CONCLUSION:

OF AVAILABLE CMOS/SOS TECHNOLOGY AFML/RCA MICROPROCESSOR CHIP SET HAS BEST POTENTIAL FOR USE IN SPACEBORNE APPLICATIONS

AFML/RCA CMOS/SOS MICROPROCESSOR CHIP SET

GPU TCS 129
- General Processing Unit
- 8-bit Parallel Slice
- Concatenatable
- Fully Static
- <125-ns Register-to-Register Add

RAM TCS 150
- Random-Access Memory
- 256x4-bit Organization
- <125-ns Access Time

ROM TCS 075
- Read-Only Memory
- Fully Static 1024 bits
- Mask-Programmable
- <100-ns Cycle Time

MUL TCS 196
- 8x8-bit Multiplier
- Expandable
- Completely Asynchronous
- Latched Input Operands

GUA TCS 093
- Gate Universal Array
- Customized Logic
- 632 Gate-Level Complexity
- 64 Pads
- Proven Cell Library
- 100-Mhz High-Speed Divider
- 452, 300 and 182 GUAs Also Available

"2910" Controller TCS 158
- Microprogram Controller
- Functional Equivalent to Am2910

GATE UNIVERSAL ARRAYS

FIXED REGULAR PATTERN OF TRANSISTORS AND ROUTING PATHS. BY DEFINING INTERCONNECTIONS AMONG DEVICES, GUAs MAY BE CUSTOMIZED WITH LOGIC VERY SIMILAR TO THE WAY READ ONLY MEMORY IS CUSTOMIZED WITH DATA.

MARTIN MARIETTA HAS DESIGNED THREE GUAs:

TCS 092-843, GPU CONTROLLER

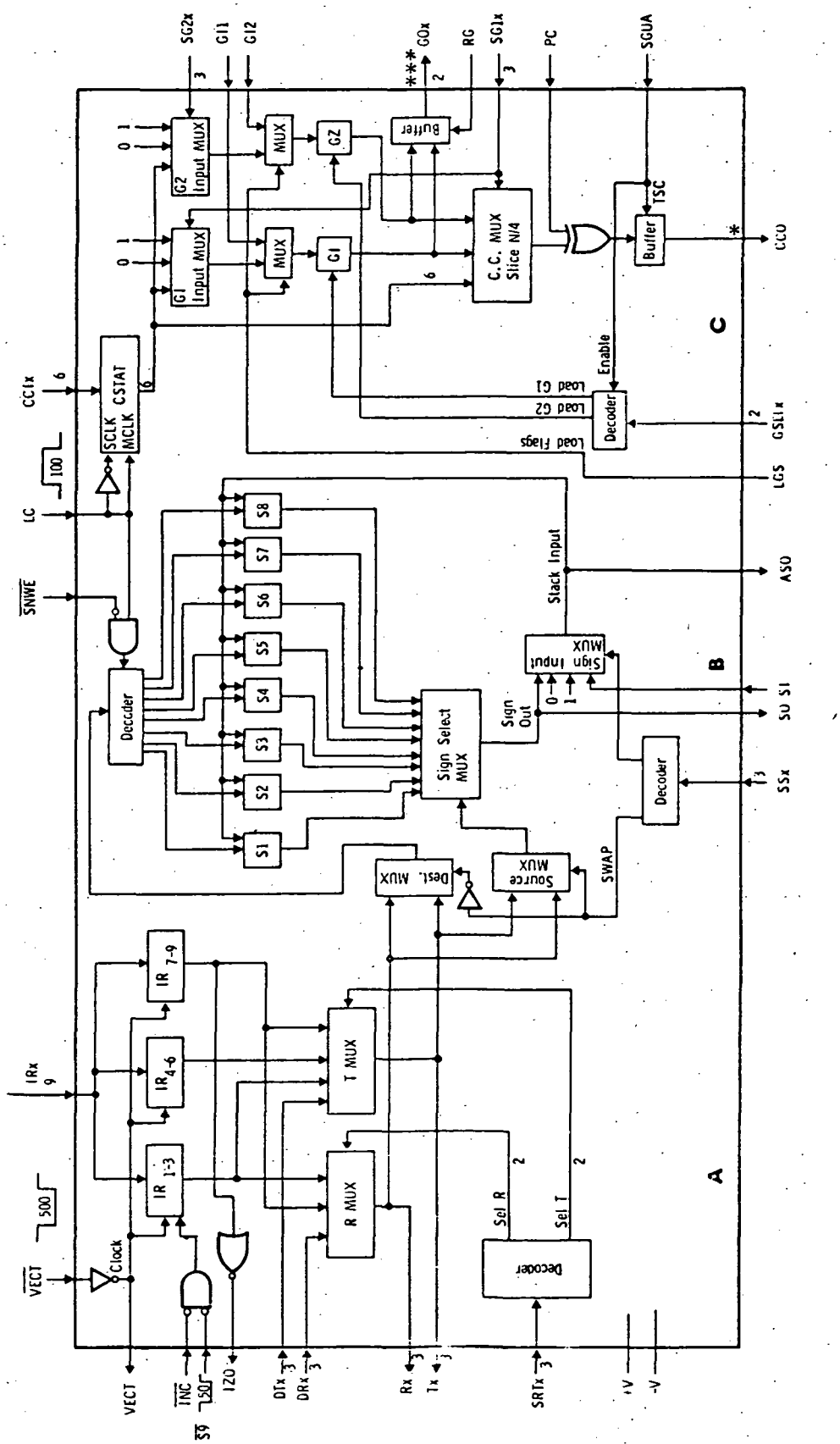
TCS 092-844, MEMORY CONTROLLER

TCS 093-845, SSI FUNCTIONS

TCS 092-843

FUNCTION:	GPU CONTROLLER
UTILIZATION:	368 INTERNAL CELLS
	62 I/O CELLS
	2 LOW Z CELLS
CIRCUITS:	A - REGISTER ADDRESS DECODE/ENCODE
	B - SIGN BIT FILE
	C - CONDITION CODE MUX

TCS 092-843 BLOCK DIAGRAM



MAC-16 COMPUTER DESIGN GOALS

LOW POWER (< 20 WATTS)

HIGH PERFORMANCE (> 250 KOPS)

MINICOMPUTER-LIKE INSTRUCTION SET ARCHITECTURE

FIXED POINT AND FLOATING POINT ARITHMETIC

PRIVILEGED AND USER MODE EXECUTION

MULTIPLE LEVEL INTERRUPT HANDLING

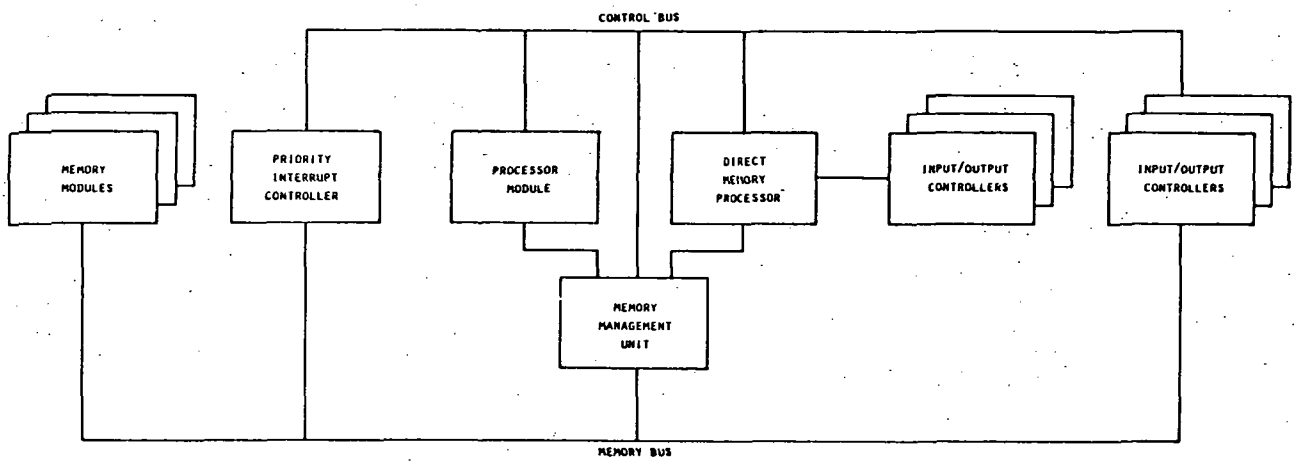
MEMORY MAPPED INPUT AND OUTPUT

DIRECT MEMORY ACCESS

MAC-16 COMPUTER PRIMARY HARDWARE MODULES

SP-D32:	CENTRAL PROCESSOR MODULE
PIC-16:	PRIORITY INTERRUPT CONTROLLER
MMU-A:	MEMORY BUS DRIVER/RECEIVER
MMU-B:	BASIC MEMORY MANAGEMENT UNIT
MMU-C:	EXTENDED MEMORY MANAGEMENT UNIT
RAM-8E:	HIGH SPEED 8K RANDOM ACCESS MEMORY
RAM-16E:	16K RANDOM ACCESS MEMORY
PROM-8:	8K PROGRAMMABLE READ ONLY MEMORY
DMP-8:	DIRECT MEMORY PROCESSOR

MAC-16 COMPUTER ARCHITECTURE



SP-D32 CENTRAL PROCESSOR PRINCIPLE ELEMENTS

FIXED POINT PROCESSOR

FLOATING POINT PROCESSOR

MICROPROGRAM CONTROL LOGIC

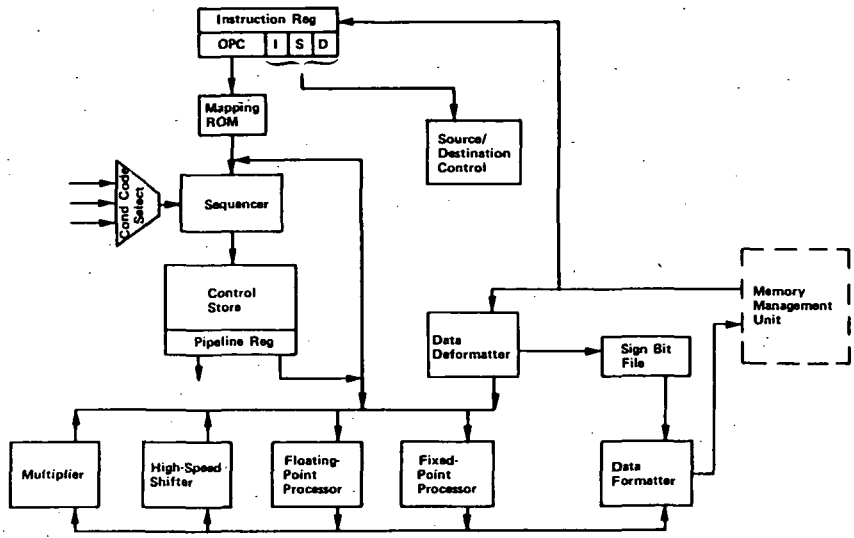
MULTIPLIER CIRCUIT

HIGH SPEED SHIFTER

DATA FORMAT AND DEFORMAT LOGIC

MEMORY INTERFACE

SP-D32 CENTRAL PROCESSOR MODULE OVERVIEW



SPACECRAFT CENTRAL PROCESSOR COMPARISON

	AUTONETICS DF224	LITTON 4516E	TELEDYNE MECA-43	IBM NSSC-II	ITEK ATAC-16	MARTIN MAC-16
FIXED POINT ADD (uS)	.8	2.0	1.6	1.7	1.25	2.0
FIXED POINT MUL (uS)	6.4	5.4	4.6	7.8	5.5	3.5
FLOATING POINT ADD (uS)	-	12.4	9.9	20.8	6.75	16.5
FLOATING POINT MUL (uS)	-	23.0	18.4	33.8	17.0	11.5
TECHNOLOGY	PMOS	TTL	TTL	TTL	TTL	CMOS/SOS
POWER (WATTS)	15.5	22	25.5	100	21	2
IC COMPONENT COUNT	50	107	5	84	?	89
REMARKS	24 BIT FIXED POINT		HYBRID PCKG		JPL MEMORY	

MAC-16 HARDWARE STATUS		REQMTS	DESIGN	BREADBOARD
SP-D32:	CENTRAL PROCESSOR MODULE	COMPLETE	COMPLETE	IN PROGRESS
PIC-16:	PRIORITY INTERRUPT CONTROLLER	COMPLETE	IN PROGRESS	1981
MMU-A:	MEMORY BUS DRIVER/RECEIVER	COMPLETE	COMPLETE	IN PROGRESS
MMU-B:	BASIC MEMORY MANAGEMENT UNIT	COMPLETE	IN PROGRESS	NOT SCHEDULED
MMU-C:	EXTENDED MEMORY MANAGEMENT UNIT	COMPLETE	NOT SCHEDULED	NOT SCHEDULED
RAM-8E:	HIGH SPEED 8K RANDOM ACCESS MEMORY	COMPLETE	COMPLETE	IN PROGRESS
RAM-16E:	16K RANDOM ACCESS MEMORY	COMPLETE	IN PROGRESS	1981
PROM-8:	8K PROGRAMMABLE READ ONLY MEMORY	COMPLETE	COMPLETE	IN PROGRESS
DMP-8:	DIRECT MEMORY PROCESSOR	COMPLETE	1981	NOT SCHEDULED

DEMONSTRATION UNIT PRIMARY ELEMENTS

MAC-16 COMPUTER ENGINEERING DEVELOPMENT UNIT

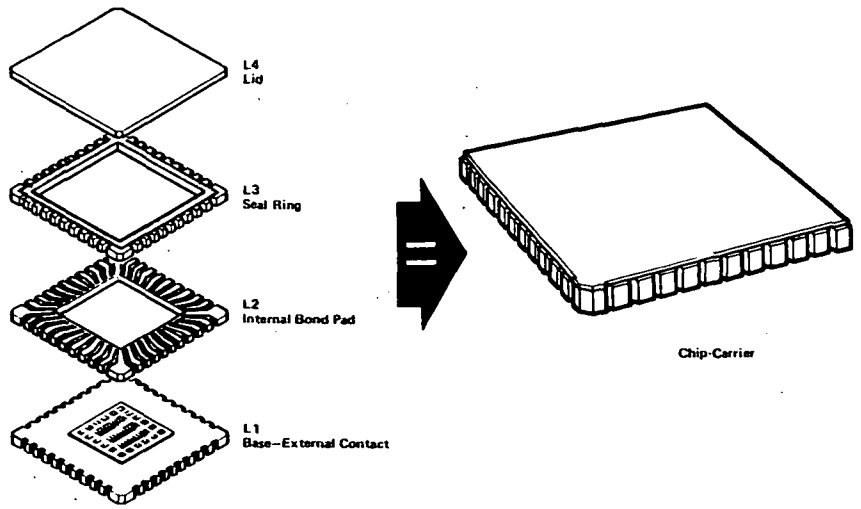
SP-D32	CENTRAL PROCESSOR MODULE
PIC-16	PRIORITY INTERRUPT CONTROLLER
MMU-A	MEMORY BUS DRIVER/RECEIVER
RAM-8E	RANDOM ACCESS MEMORY
PROM-8	PROGRAMMABLE READ ONLY MEMORY

SUPPORT EQUIPMENT

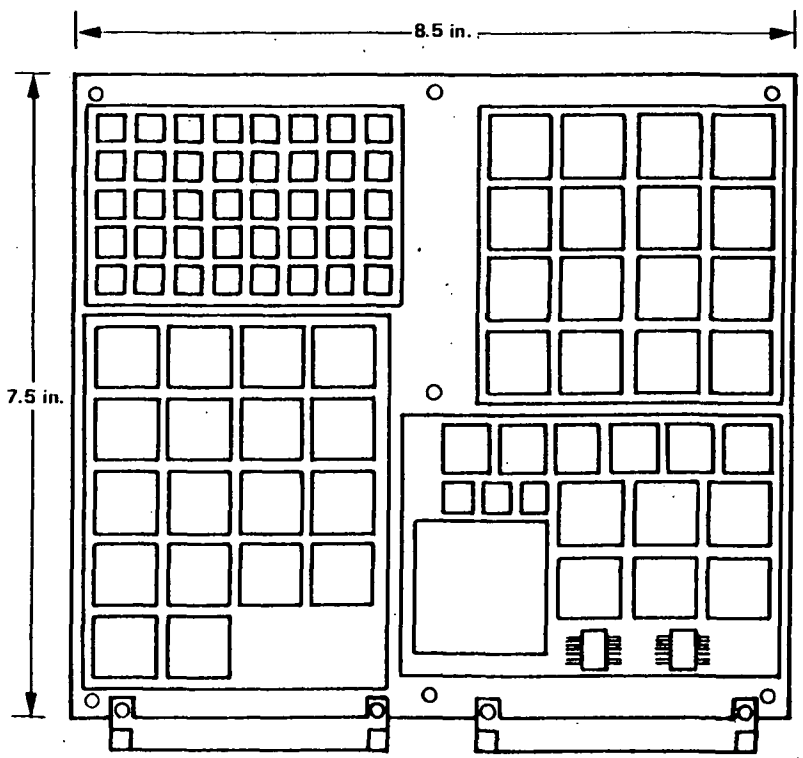
- SERIAL I/O MODULE
- OPERATOR CONTROL PANEL
- WRITABLE CONTROL STORE
- GUA TTL CIRCUIT EQUIVALENTS

MICROCOMPUTER - HARDWARE INTERFACE

VAX 11/780 - SHOFTWARE HOST



SP-D32 CENTRAL PROCESSOR MODULE FLIGHT UNIT



Quantity	Part	Package
43	TCS 075	24 HCC
1	TCS 158	48 HCC
5	TCS 129	48 HCC
16	TCS 196	64 HCC
4	TCS 092-843	64 HCC
1	TCS 092-844	64 HCC
19	TCS 093-845	64 HCC
2	Resistor Pack	16 FP
1	E34 TCXO	Hybrid
2	110 Lead Connectors	–
–	Capacitors	–

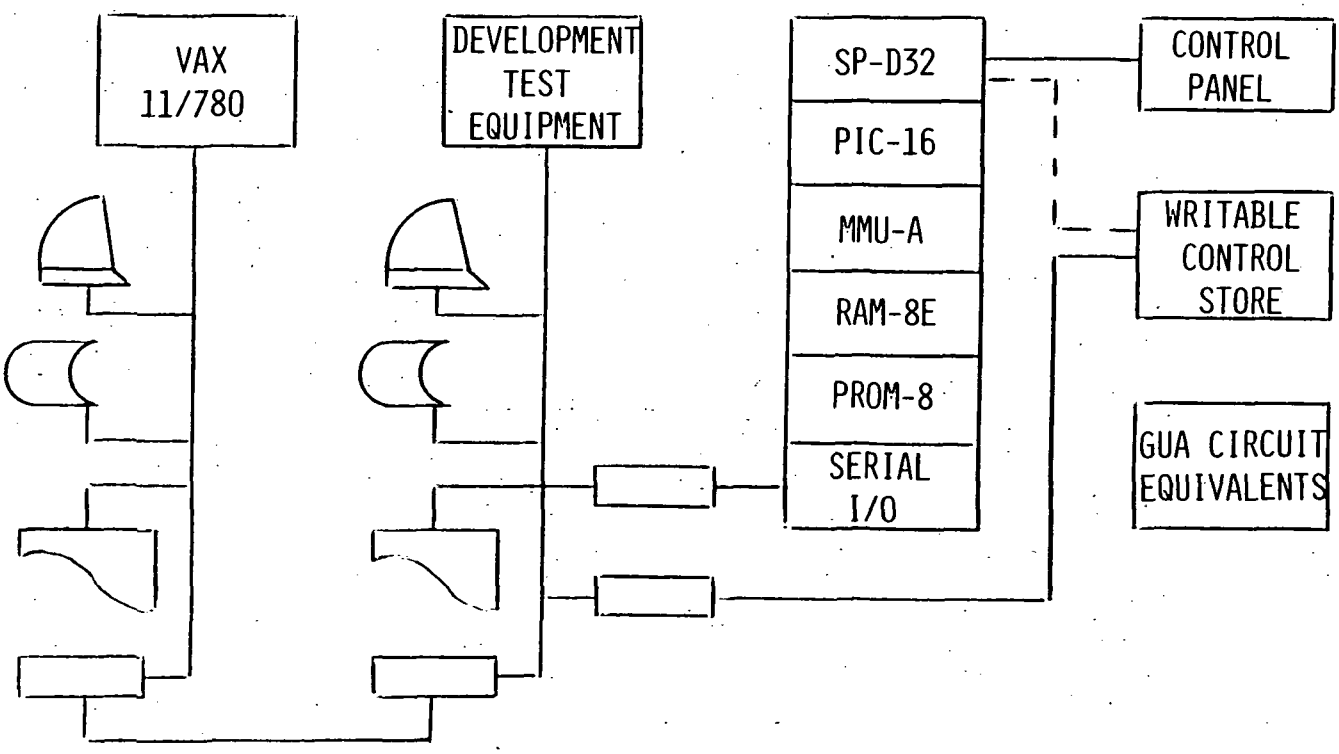
1981 HARDWARE OBJECTIVES

BUILD AND FUNCTIONAL TEST OF PRINTED CIRCUIT BOARDS FOR PROCESSOR MODULE, MEMORY MANAGEMENT UNIT, PRIORITY INTERRUPT CONTROLLER, AND 8K MEMORY MODULE

FUNCTIONAL TEST OF 4K RAMs (TCS 146) FOR AIR FORCE MATERIALS LAB

PERFORM PACKAGING MINI-QUALIFICATION TEST

MAC-16 COMPUTER TEST SET UP



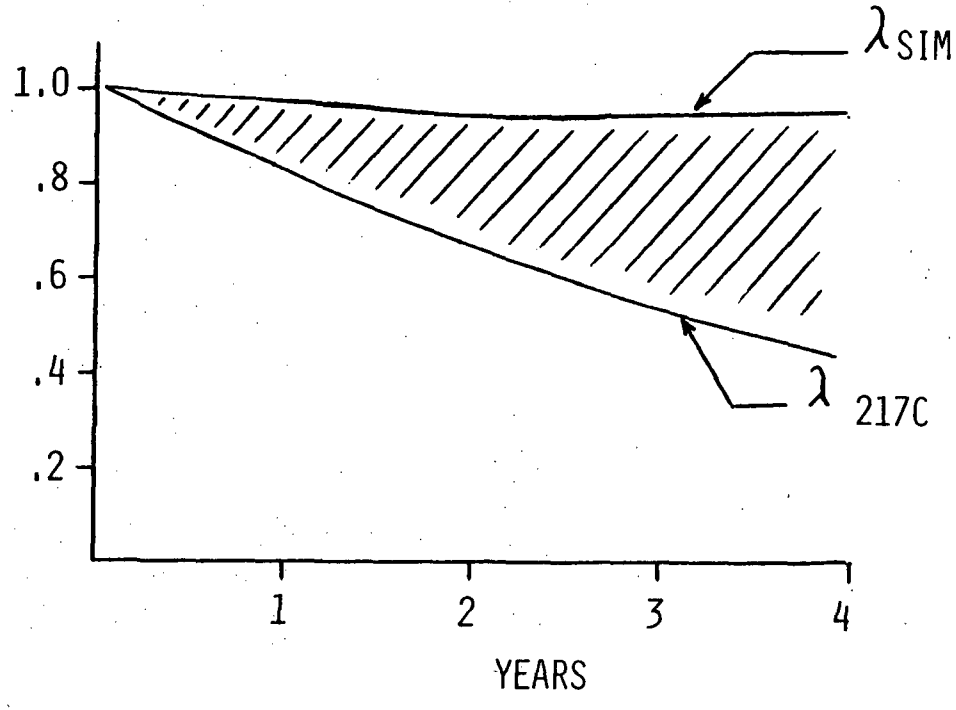
MAC-16 PROJECTED FLIGHT UNIT DATA

SYSTEM CONFIGURATION

- 1 SP-D32
- 1 MMU-B
- 4 RAM-16E (64K WORDS)
- 1 DMP-8
- 1 PIC-16

WEIGHT ESTIMATE = 5.9 LBS

POWER ESTIMATE 10 WATTS



Page Intentionally Left Blank

EVOLUTION OF A STANDARD MICROPROCESSOR-BASED SPACE COMPUTER

Manuel Fernandez*
Litton Systems, Inc.
Woodland Hills, California

Starting in 1976, an existing in-inventory computer hardware/software package (B-1 RFS/ECM) was repackaged and applied to multiple missile/space programs. Concurrent with the application efforts, low-risk modifications were made to the computer from program to program to take advantage of newer, advanced technology and to meet increasingly more demanding requirements (computational and memory capabilities, longer life, and fault tolerant autonomy).

In 1978, the 2901 microprocessor chip was incorporated; and since that time advances in this mature, multi-sourced, qualified chip (specifically the 2901B) have been used to improve computational capability.

This development establishes a base to explore the use of newer microprocessors and to discuss current trends from centralized to distributed processors. Key differences in computational and memory capabilities, orbital life, and autonomous fault-tolerant provisions are compared.

In summary, it is concluded that microprocessors hold promise in a number of critical areas for future space computer applications. However, the benefits of the DoD VHSIC Program are required and the old proliferation problem must be revised.

*Member AIAA

OUTLINE

1. UNIQUE REQUIREMENTS OF SPACE COMPUTERS
2. STATE-OF-THE-ART SPACE COMPUTER (BASELINE)
3. STATISTICS OF BASELINE COMPUTER
4. DESIRED IMPROVEMENTS IN BASELINE COMPUTER
5. CURRENT TRENDS AND TRADEOFFS IN COMPUTERS
6. USE OF NEWER MICROPROCESSORS
7. SUMMARY

UNIQUE REQUIREMENTS OF SPACE COMPUTERS

LOW POWER

POWER IMPACTS

- THERMAL BALANCE OF SATELLITE
- RELIABILITY
- POWER SOURCE CAPABILITY

ENVIRONMENTAL DESIGN

- VIBRATION (20 g RMS)
- PYROTECHNIC SHOCK (3,000 g)
- SPACE RADIATION (INCLUDING COSMIC RAYS)
- EMI (MIL-STD-1541)
- OUTGASSING (SP-R-0022A AND JSC-08962)
- POWER SOURCE (21V TO 35V DC)
- THERMAL (-34C TO +71C "COLD PLATE")

(CONT)

UNIQUE REQUIREMENTS OF SPACE COMPUTERS (CONT)

HIGH RELIABILITY (LONG ORBIT LIFE WITH HIGH PROBABILITY)

- MATURE TECHNOLOGY
- SIMPLICITY
- WORST-CASE DESIGN
- EXTRA-RELIABILITY PARTS
- EXTRA-QUALITY WORKMANSHIP/INSPECTION
- EXTENSIVE UNIT TESTING
  - RANDOM VIBRATION
  - THERMAL CYCLING
  - THERMAL VACUUM OPERATION
  - BURN-IN
- ELIMINATION OF SINGLE-POINT FAILURE MODES
  - *REDUNDANCY*
  - FAULT-TOLERANT AUTONOMY

SATELLITE THERMAL BALANCE IN ORBIT *

MAX INTERNAL POWER DISSIPATION	AVG. SURFACE TEMPERATURE
8,200 WATTS	60°F
13,000 WATTS	115°F

ASSUMPTIONS

SATELLITE - SPHERE, WHITE SURFACE, 10 FT DIAMETER
SIMPLE THERMAL MODEL UTILIZED
HEAT SOURCES - SUN, EARTH, INTERNAL POWER DISSIPATION
HEAT SINK - SPACE

*THERMAL BALANCE EFFECTS

- MAX INTERNAL POWER DISSIPATION
- POWER SOURCE CAPABILITY
- RELIABILITY

UNIT ACCEPTANCE TESTING

1. INSPECTION
2. PERFORMANCE TEST
3. RANDOM VIBRATION TEST
   - DURATION 60 SEC/AXIS (ALL AXES)
   - OVERALL 9.2 gRMS
4. FUNCTIONAL TEST
5. THERMAL CYCLING (THIS TEST TAKES APPROXIMATELY 160 HRS, OR 6.67 DAYS)
   - EIGHT CYCLES TOTAL
   - -11C TO +61C
   - CONTINUOUS UNIT OPERATION
6. FUNCTIONAL TEST

(CONT)

UNIT ACCEPTANCE TESTING (CONT)

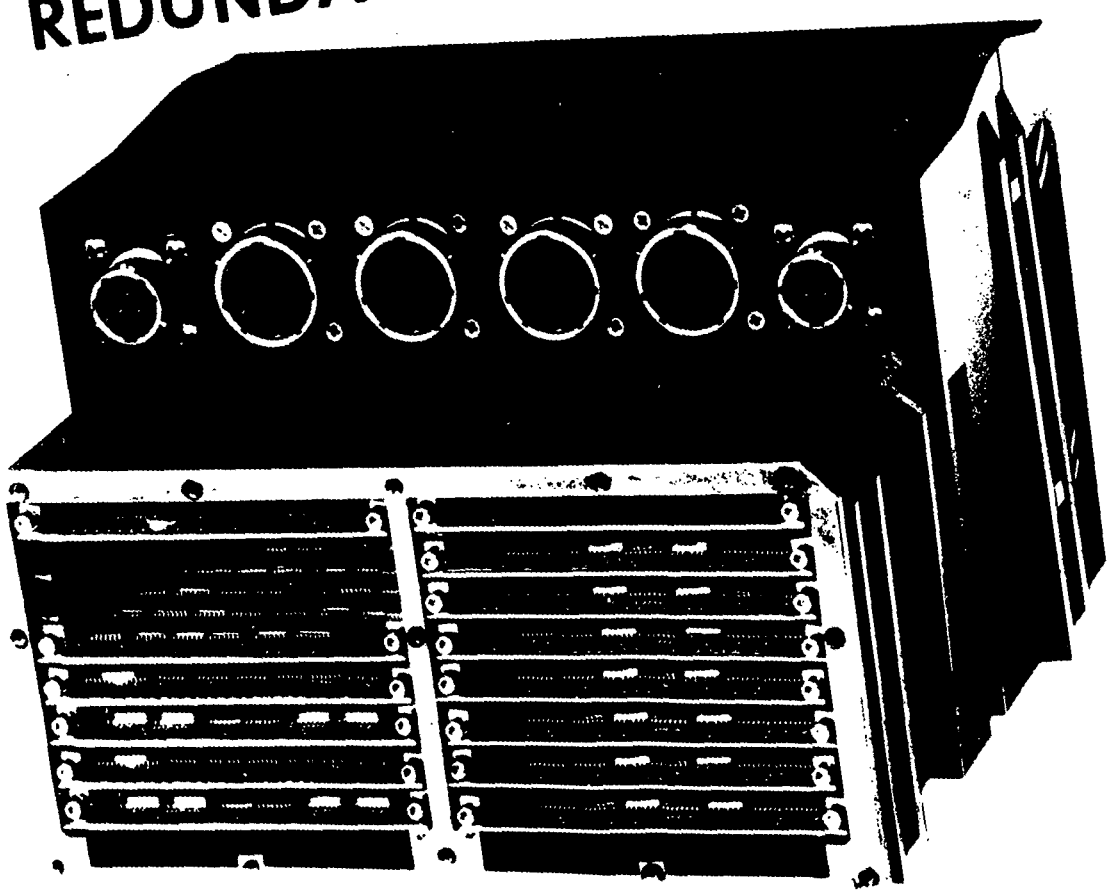
7. OPERATING ORBIT THERMAL VACUUM TEST
   - 12-HOUR SOAKS AT -11C AND +61C, AFTER STABILIZATION, AND BEFORE FUNCTIONAL TESTS AT -11C AND +61C
   - UNIT OPERATING CONTINUOUSLY
8. FUNCTIONAL TEST
9. PIN-RETENTION TEST (MIL-STD-1344A, METHOD 2014)
10. FUNCTIONAL TEST
11. BURN-IN TEST
    - +61C CONTINUOUS
    - UNIT OPERATING CONTINUOUSLY
    - 300 HR DURATION (12.5 DAYS)
    - DIAGNOSTICS TEST
    - GALWREC MEMORY TEST
12. PERFORMANCE TEST
13. POST TEST INSPECTION

A STATE-OF-THE-ART SPACE COMPUTER

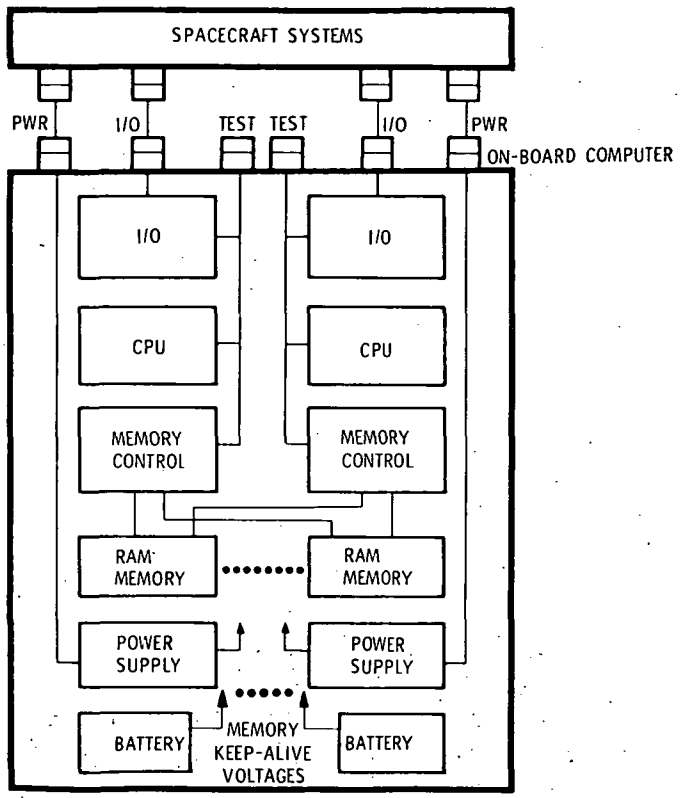
(Baseline)

Based on 2901 (Now 2901B) Microprocessor Chip

REDUNDANT COMPUTER



REDUNDANT COMPUTER



REDUNDANT COMPUTER

VOLUME	466 CU IN.	
WEIGHT	26.4 LBS	
POWER	77.3 WATTS (96K-WORDS ACTIVE)	
THROUGHPUT	539 KOPS (GIBSON MIX)	
MEMORY	128K-WORDS (ACTIVE/STANDBY) (ADDRESSING TO 256K-WORDS)	
RELIABILITY	0.966 (5 YRS)	(WITH 64K-WORDS ACTIVE, 16K-WORDS STANDBY)

STATISTICS OF BASELINE COMPUTER

POWER BREAKDOWN

CPU	26%
MEMORY	19
I/O	17
S/T	62
POWER SUPPLY	38*
TOTAL	100%

*62% EFFICIENCY, WORST CASE DUE TO POWER SOURCE CHARACTERISTICS

FLIGHT UNIT COST BREAKDOWN

MATERIAL	64%
ASSEMBLY	16
TEST/INSPECTION	20
TOTAL	100%

FLIGHT UNIT COST BREAKDOWN

CPU	11%
MEMORY	59*
I/O	6
POWER SUPPLY	6
OTHER	18
	100%

*128K-WORDS TOTAL (ACTIVE/STANDBY)

FAILURE RATE BREAKDOWN

CPU	38%
MEMORY	23*
I/O	29
POWER SUPPLY	8
OTHER	2
TOTAL	100%

*REFLECTS FAULT-TOLERANT MEMORY EFFECTS
(THIS IS RESIDUAL)

SPACE COMPUTER PROJECT COST BREAKDOWN

FLIGHT UNIT	18%
SUPPORT	40*
SOFTWARE	42
TOTAL	100%

*INCLUDES SPARE FLIGHT UNIT

SUMMARY

MICROPROCESSORS COULD IMPACT BASELINE COMPUTER, WHOSE STATISTICS YIELD:

- CPU 26% OF POWER
  (EXCLUSIVE OF P.S. DISSIPATION PENALTY)
- CPU 11% OF FLIGHT UNIT COST
- CPU 38% OF FAILURE RATE
- SOFTWARE 42% OF PROJECT COST

DESIRED IMPROVEMENTS (IN BASELINE)

- FAULT-TOLERANT AUTONOMY
- HARDER RADIATION TOLERANCE
  - TOTAL IONIZING DOSE
  - SINGLE EVENT UPSETS
- LOWER POWER
- IMPROVED COMPUTATIONAL CAPABILITY
- LONGER ORBIT LIFE WITH HIGH PROBABILITY
- LOWER SOFTWARE COSTS

CURRENT TRENDS

- CENTRALIZED SIMPLEX COMPUTER ARCHITECTURES DEMANDING HIGHER WORKLOAD CAPABILITIES
- DISTRIBUTED SYSTEMS WITH FOLLOWING INTERPRETATIONS:
  (FALL OUT OF POINT-OF-USE SYSTEMS, RESOURCE-SHARING NETWORKS, MULTIPLE PROCESSOR SYSTEMS ARCHITECTURES)
  - MULTIPROCESSORS (TO HANDLE LOAD), STILL CENTRALIZED
  - DEDICATED COMPUTERS (PER FUNCTION), LOOSELY FEDERATED
  - FEDERATED COMPUTER ARCHITECTURE WITH SYSTEM MGR AND DEDICATED SUBSYSTEM COMPUTERS
  - DISTRIBUTION OF TASKS/WORKLOADS AMONG NONDEDICATED COMPUTERS
  - COMBINATIONS OF ABOVE

TRADEOFFS

CENTRALIZED SYSTEM ADVANTAGES*

- MORE EFFICIENT LOAD SHARING AND LOWER RESPONSE TIME
- GREATER FLEXIBILITY
- MORE EFFICIENT COMMUNICATION
- LESS REDUNDANCY OF STORAGE
- GREATER TOTAL COMPUTATIONAL CAPABILITY
- HIGHER TOTAL SYSTEM RELIABILITY
- MORE EFFECTIVE USE OF REDUNDANCY

DEDICATED SYSTEM ADVANTAGES

- LESS COMPLEX SOFTWARE
- HIGHER RELIABILITY FOR INDIVIDUAL FUNCTIONS

*INCLUDES INTEGRATED, MULTIPROCESSOR, CENTRALIZED SYSTEMS

SUMMARY

MICROPROCESSORS COULD IMPACT BASELINE COMPUTER, WHOSE CENTRALIZED COMPUTER ARCHITECTURE YIELDS:

- MORE COMPLEX SOFTWARE
- LESS RELIABILITY FOR INDIVIDUAL FUNCTIONS

UTILIZATION OF NEWER MICROPROCESSORS

MICROPROCESSOR POTENTIAL IMPACT (ON BASELINE)

ON BASELINE STATISTICS OF:	
● CPU 26% OF POWER	● PROMISING
● CPU 11% OF FLIGHT UNIT COST	● PROMISING
● CPU 38% OF FAILURE RATE	● PROMISING (BUT NOT YET MATURE)
● SOFTWARE 42% OF PROJ. COST	● PROMISING (BUT NOT ASSURED)
ON DESIRED BASELINE IMPROVEMENTS OF:	
● LONGER ORBIT LIFE	● PROMISING
● FAULT-TOLERANT AUTONOMY	● POTENTIAL NOT CLEAR
● IMPROVED COMPUTATIONAL CAPABILITY	● PROMISING
● LOWER SOFTWARE COSTS	● DEPENDENT ON REUSE OF SOFTWARE
● HARDER RADIATION TOLERANCE	● MICROPROCESSOR TECHNOLOGIES AND HIGHER LEVELS OF INTEGRATION OF FUNCTIONS ON A CHIP ARE NOT CONDUSIVE TO RADIATION HARDENING TOLERANCE (DoD VHSIC PROGRAM WILL HELP)
ON CENTRALIZED SYSTEM ARCHITECTURE DISADVANTAGES	
● MORE COMPLEX SOFTWARE	● PROMISING
● LESS RELIABILITY PER FUNCTION	● PROMISING

SUMMARY

- MICROPROCESSORS BEING DRIVEN BY LARGE COMMERCIAL MARKETPLACE
- MICROPROCESSORS LOOK PROMISING IN A NUMBER OF CRITICAL AREAS FOR FUTURE SPACE COMPUTER APPLICATIONS
- DoD VHSIC PROGRAM COULD HELP SOLVE CRITICAL RADIATION HARDENING PROBLEMS
- REUSE OF SOFTWARE A CRITICAL ITEM
- USE OF LARGE NUMBER OF MICROPROCESSOR TYPES FOR FUTURE SPACE APPLICATIONS NOT WISE, AND CHOICE WILL BE DIFFICULT
- FAULT-TOLERANT AUTONOMY NEEDS ATTENTION

KEY AREAS NEEDING STRESS

REDUNDANCY MANAGEMENT

FAULT TOLERANCE

SOFTWARE DEVELOPMENT

MICROPROCESSORS FOR IMAGING SEEKERS

R. G. Hix and D. W. Smith
General Dynamics
Pamona, California

Operational imaging seekers, such as AIRIS and FAIRS (air-to-air IR seekers) require high-speed, high performance, low-power, multi-processors (system throughput in excess of eighteen million operations per second with CMOS power consumption). This paper addresses the three areas which must be investigated to achieve this combination of performance, namely, device technology, microprocessor architecture, and multi-processor architecture. The results of a comprehensive microprocessor survey are presented, which identified the ATMAC microprocessor (a high-speed, low-power CMOS/SOS microprocessor produced by RCA's Advanced Technology Laboratory) as the best candidate for imaging seeker applications. Capabilities of the CMOS/SOS technology are discussed along with problems which had to be overcome to successfully apply this technology to imaging seekers. Capabilities, problems encountered, and their solutions in the application of ATMAC microprocessors to imaging seekers are discussed. The results of a multi-processor architecture selection trade study are presented. Also, the capabilities and operating characteristics of ideal microprocessor and multi-processor architectures for use in imaging seekers are detailed, as well as their implications to multi-programming. The implemented multi-processor system is compared with the desired system, and desirable advances in device technology, microprocessors, and multi-processor architectures are highlighted.

Page Intentionally Left Blank

MODULAR MISSILE BORNE COMPUTERS

R. Ramseyer, R. Arnold, H. Applewhite and R. Berg
Honeywell Systems and Research Center
Minneapolis, Minnesota

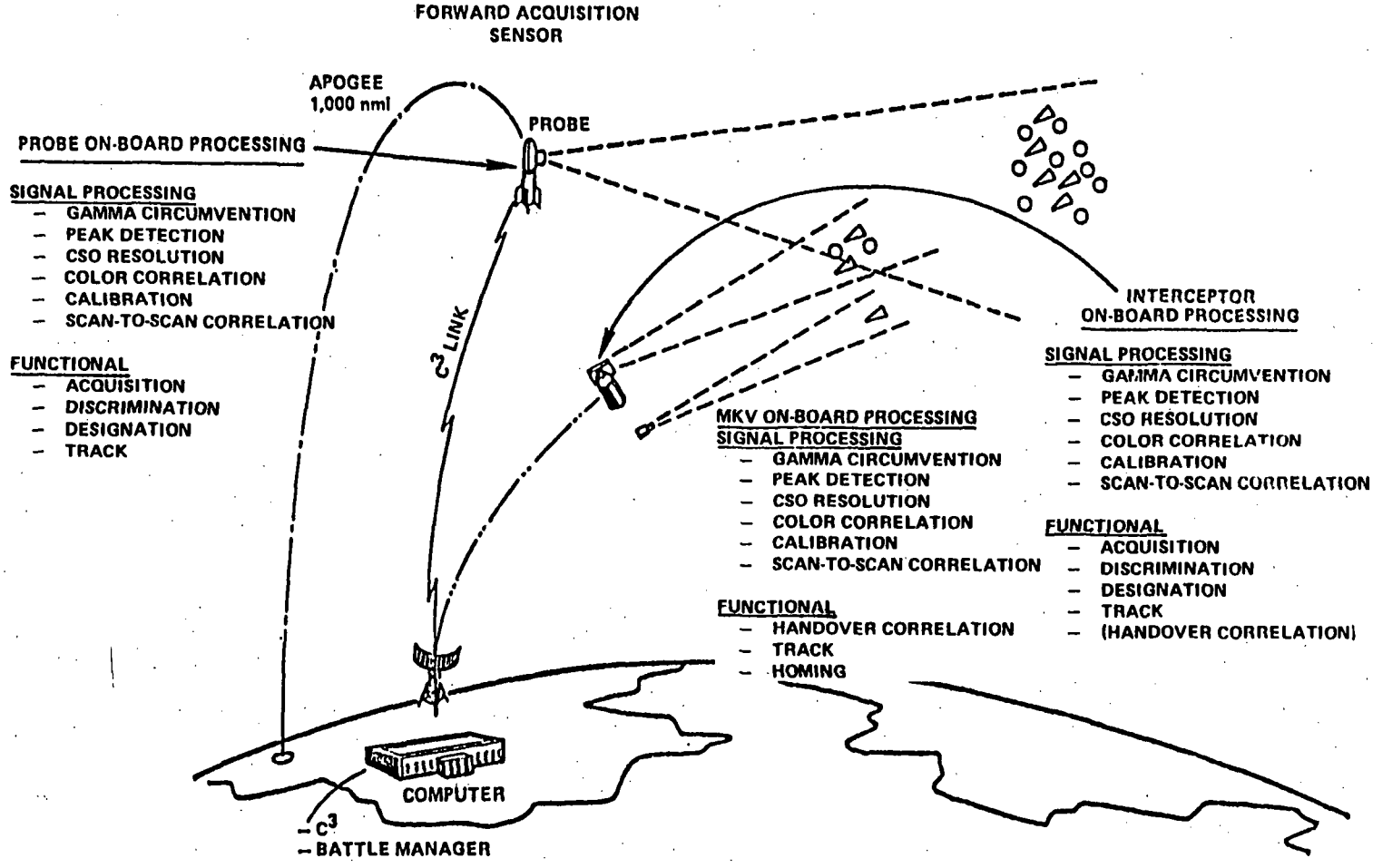
The increasing real time signal and data processing loads on-board BMD interceptors cannot be met with currently available and flyable processors. The Modular Missile Borne Computer is being developed to provide a solution to that problem through the use of a collection of microprocessors in a distributed processing system.

This paper discusses the Modular Missile Borne Computer's architecture with emphasis on how that architecture evolved from a careful analysis of both the physical constraints and the processing requirements. The development techniques used are generally applicable to real-time data processing systems and have resulted in the achievement of one of our most significant design goals. This goal is a modular, flexible, extensible system capable of adapting to evolving BMD problems as well as others where an ultra-high performance distributed processor is desirable.

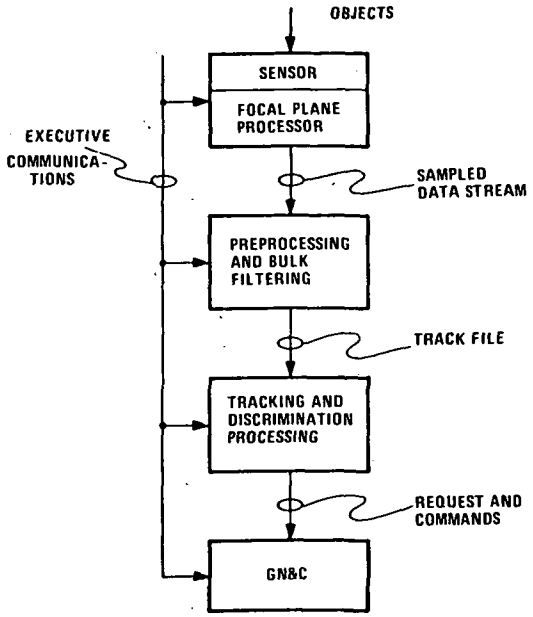
The general objective for the MMBC program is the development of a data processing system which lends itself readily to system growth, reconfiguration and changes in application or environment. Given the constraints and requirements imposed by the BMD threat, scenarios and environmental considerations, four driving architectural considerations result:

- The required processing is real time.
- There is a massive quantity of data and it is received at a rapid rate.
- A high degree of modularity, flexibility and potential for growth is desired in MMBC.
- MMBC must be capable of performing in an extremely hostile operating environment (e.g. shock, vibration, temperature, and nuclear).

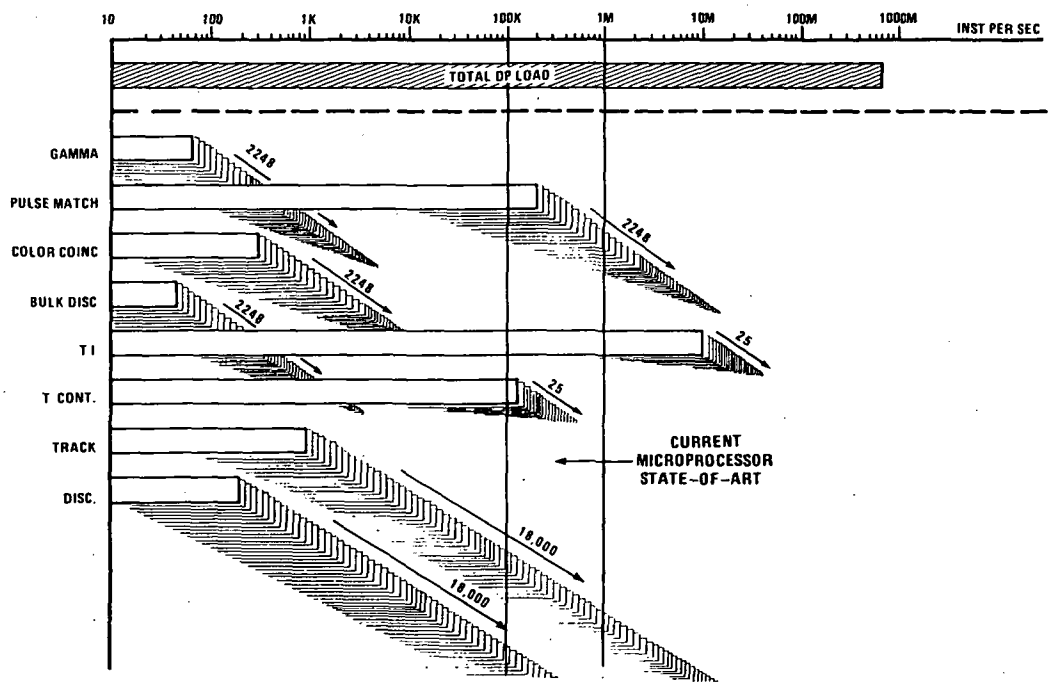
DATA PROCESSING IN A MIDCOURSE SYSTEM



HIGH LEVEL PROCESSOR STRUCTURE

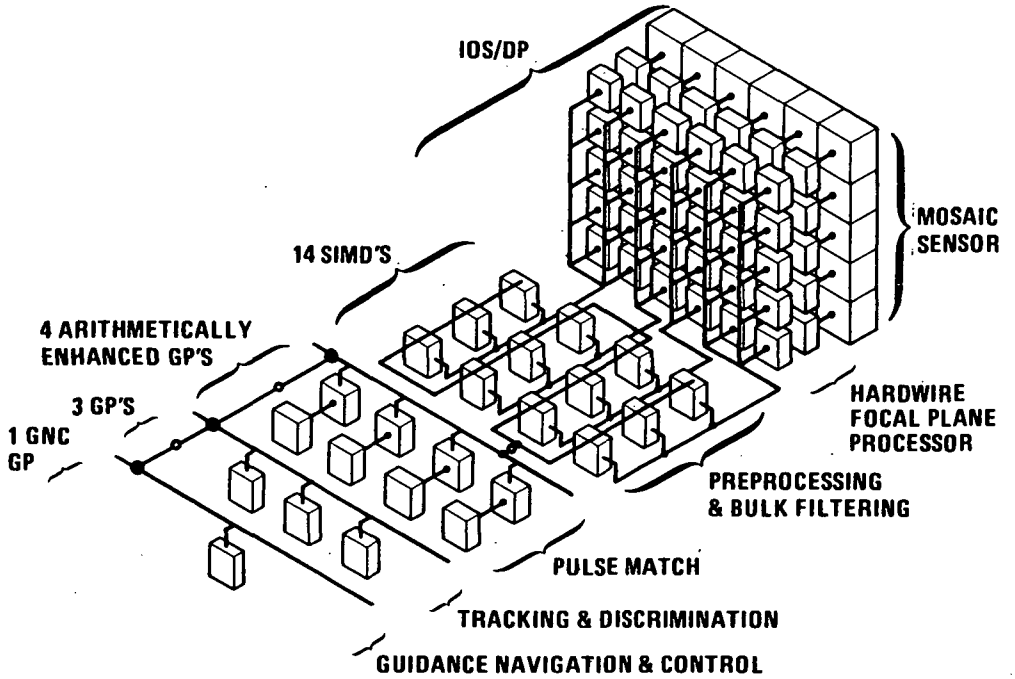


TOTAL DP LOAD IS COMPOSED OF MANY SMALL, INDEPENDENT LOADS



MODULAR MISSILE-BORNE COMPUTERS

OBJECTIVE: INVESTIGATE ADVANCED PREPROCESSING TECHNOLOGY & THE APPLICATION OF MODULAR, FLEXIBLE, EXTENDABLE MICROCOMPUTER ARRAYS TO PERFORM THE REAL TIME DATA PROCESSING NECESSARY ON BOARD A BMD INTERCEPTOR.



RATIONALE: ADVANCES IN INTERCEPTOR & TECHNOLOGY AS WELL AS INCREASING COMPUTATION OF BMD FUNCTIONS ON BOARD IMPOSE REQUIREMENTS ON THE DATA PROCESSOR THAT CANNOT BE MET BY CONVENTIONAL COMPUTER ARCHITECTURES. MODULAR, FLEXIBLE, AND EXTENDABLE COMPUTER STRUCTURES ARE REQUIRED TO MEET THE NEEDS OF THE FLUID AND RAPIDLY EVOLVING BMD SYSTEMS.

BENEFITS OF MICROPROCESSORS

- ALLOW MAXIMUM USE OF LSIC HARDWARE
  - MINIMIZES SIZE/WEIGHT/POWER/INTERCONNECTS
- SPECIAL PURPOSE OPERATION CAN BE ACHIEVED THROUGH MICROPROGRAMMING
- SIMULTANEOUS OPERATION OF MANY REAL TIME HARDWARE UNITS DRASTICALLY REDUCES NEED FOR MULTIPROGRAMMING
  - REDUCED SOFTWARE COST
  - REDUCED OVERHEAD TIME
- ALLOWS MAXIMUM MODULARITY TO BE ACHIEVED WHICH IMPROVES
  - FAULT TOLERANCE
  - FLEXIBILITY/ALTERABILITY

APPROXIMATE CAPABILITIES AND REQUIREMENTS

PROCESSING SECTION	PROCESSOR CONFIGURATION	REQUIREMENT	CAPABILITY
PREPROCESSING AND BULK FILTERING			
DMX TO PULSE MATCH	14 SIMD (1 TO 3 MIPS EA)	22 MIPS	~40 MIPS*
PULSE MATCH	4 GP W/SPECIAL ARITHMETIC (15 MIPS FOR PULSE MATCH; 1 MIPS OTHERWISE)	43 MIPS	45 MIPS
TRACKING AND DISCRIMINATION	3 GP (1 MIPS EACH)	151 KIPS	3 MIPS
GUIDANCE NAVIGATION AND CONTROL	1 GP (500 KIPS)	185 KIPS	500 KIPS
TOTAL CAPACITY			88.5 MIPS

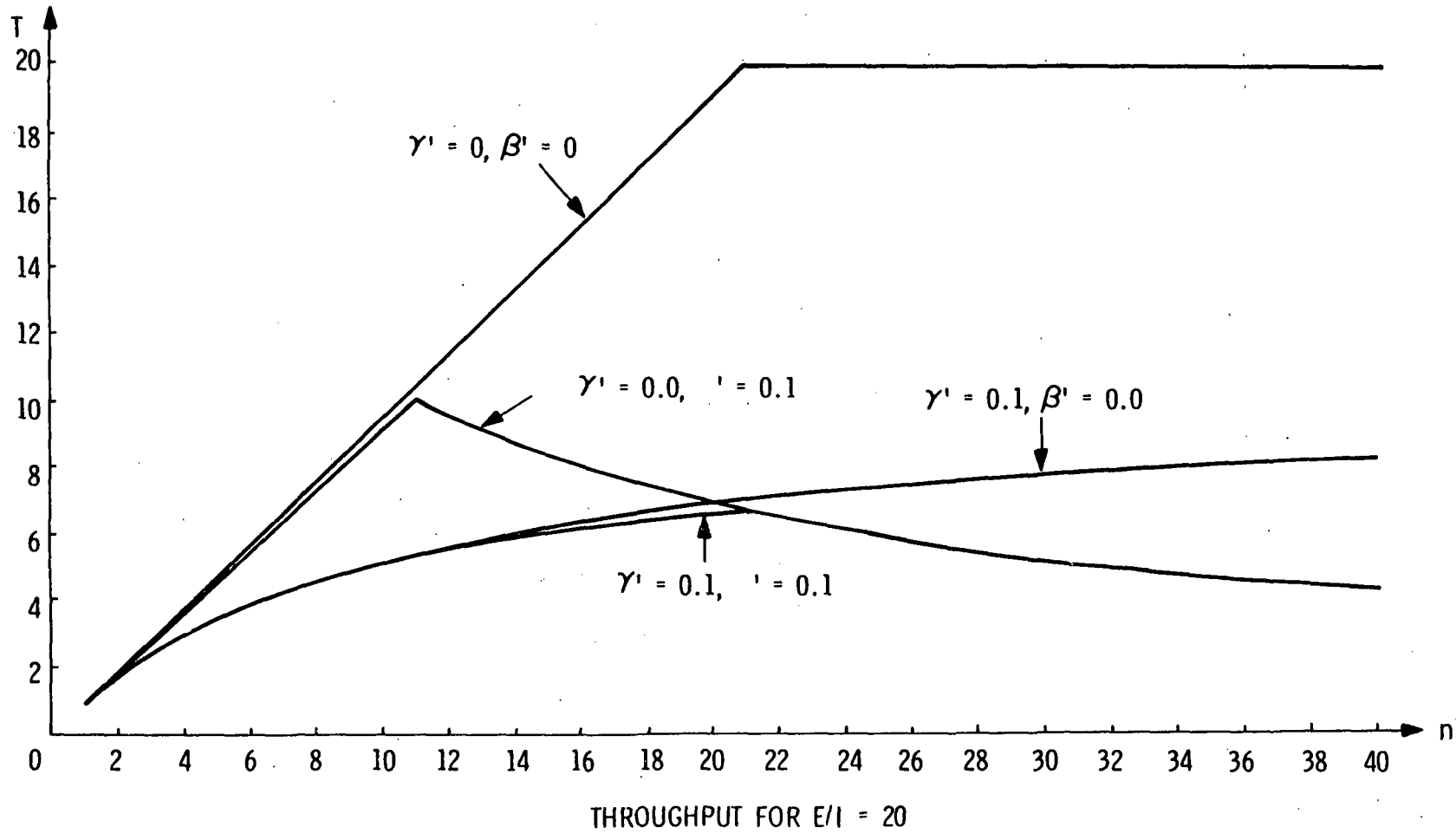
*HIGH THROUGHOUT REQUIRED TO ABSORB OVERHEAD AND SATISFY REAL-TIME RESPONSE REQUIREMENT.

EACH COMPUTER IN MMBC IS "TUNED" TO PERFORM WELL IN ITS AREA OF APPLICATION. E.G., MULTIPLE DATA STREAM PROCESSORS IN BULK FILTERING; PIPELINED, FAST ARITHMETIC UNIT IN PULSE MATCH.

CHARACTERISTICS OF HARDWARE MODULES

GENERAL PROCESSING ELEMENT (16 BITS)	1 MIPS
SIMD PE (3 ALU'S)	3 MIPS
VOA PROCESSOR	15 MIPS/5 MIPS
LOCAL MEMORY (3 PORT RAM)	
READ ACCESS TIME	270 ns
WRITE ACCESS TIME	75 ns
COMMON BULK MEMORY	
READ	375 ns
WRITE	140 ns
CYCLE	425 ns
GLOBAL BUS (3 BUSES)	
TRANSFER RATE	1 M WORDS/S
	1 M WORDS/S
CAN HANDLE 40 K TRACKS/SEC/GLOBAL BUS	
UTILIZATION	60%

IDEALIZED SHARING-THROUGHPUT VERSUS NUMBER OF PROCESSORS

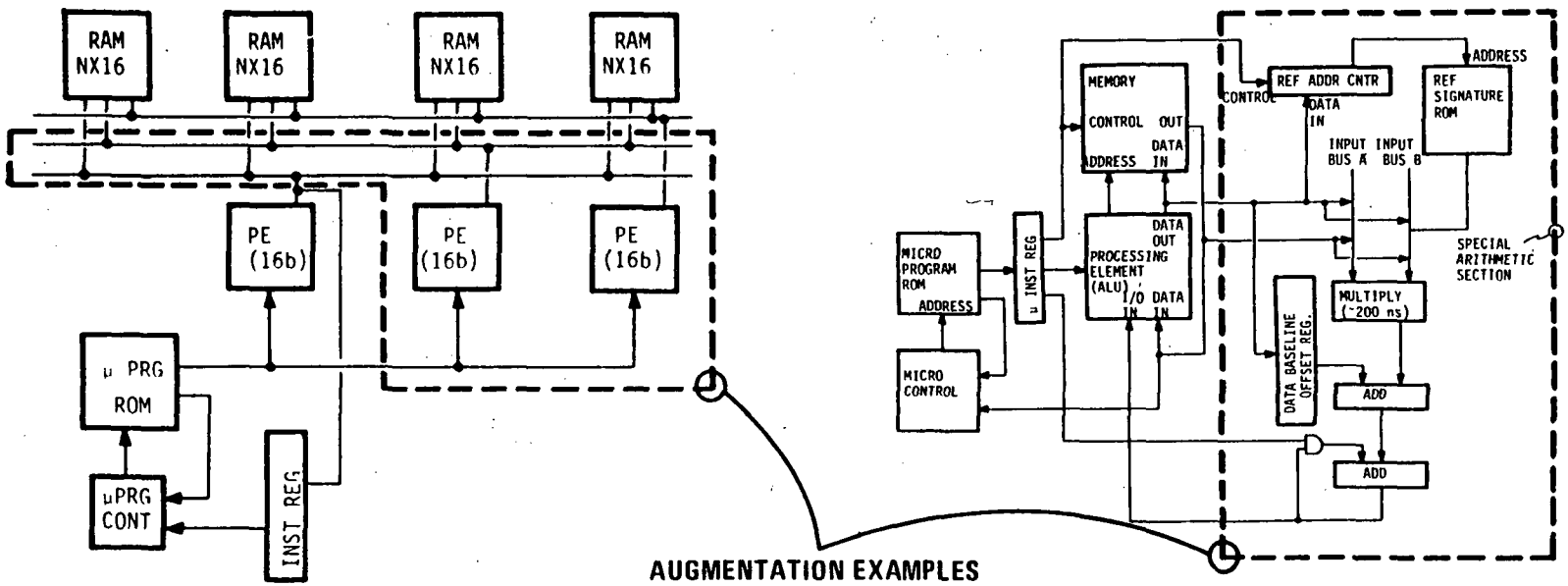


HARDWARE ARCHITECTURE

- REQUIRES MULTICOMPUTER
- MODULARITY, FLEXIBILITY, EXTENDABILITY REQUIRED
- OVERHEAD RESULTING FROM DISTRIBUTION MUST BE CONTROLLED
- REQUIRES SPECIAL PURPOSE HARDWARE
- SYSTEM MUST BE MANAGEABLE
- NO. OF COMPUTERS LIMITED BY SIZE, WT., POWER, PROBLEM STRUCTURE

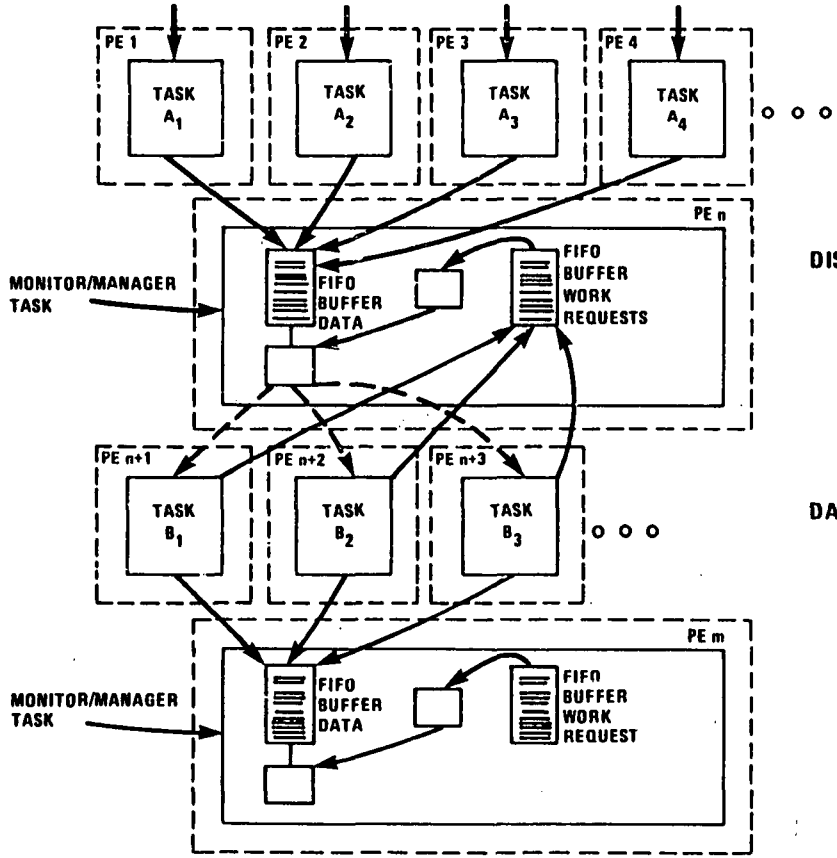
THEREFORE:

- EACH COMPUTER SHOULD BE AS POWERFUL AS POSSIBLE
- EACH COMPUTER CAN BE AUGMENTED TO PERFORM A CLASS OF ALGORITHMS WELL
- HARDWARE MUST ADAPT EASILY TO ALGORITHM STURCTURES



AUGMENTATION EXAMPLES

SOFTWARE ARCHITECTURE



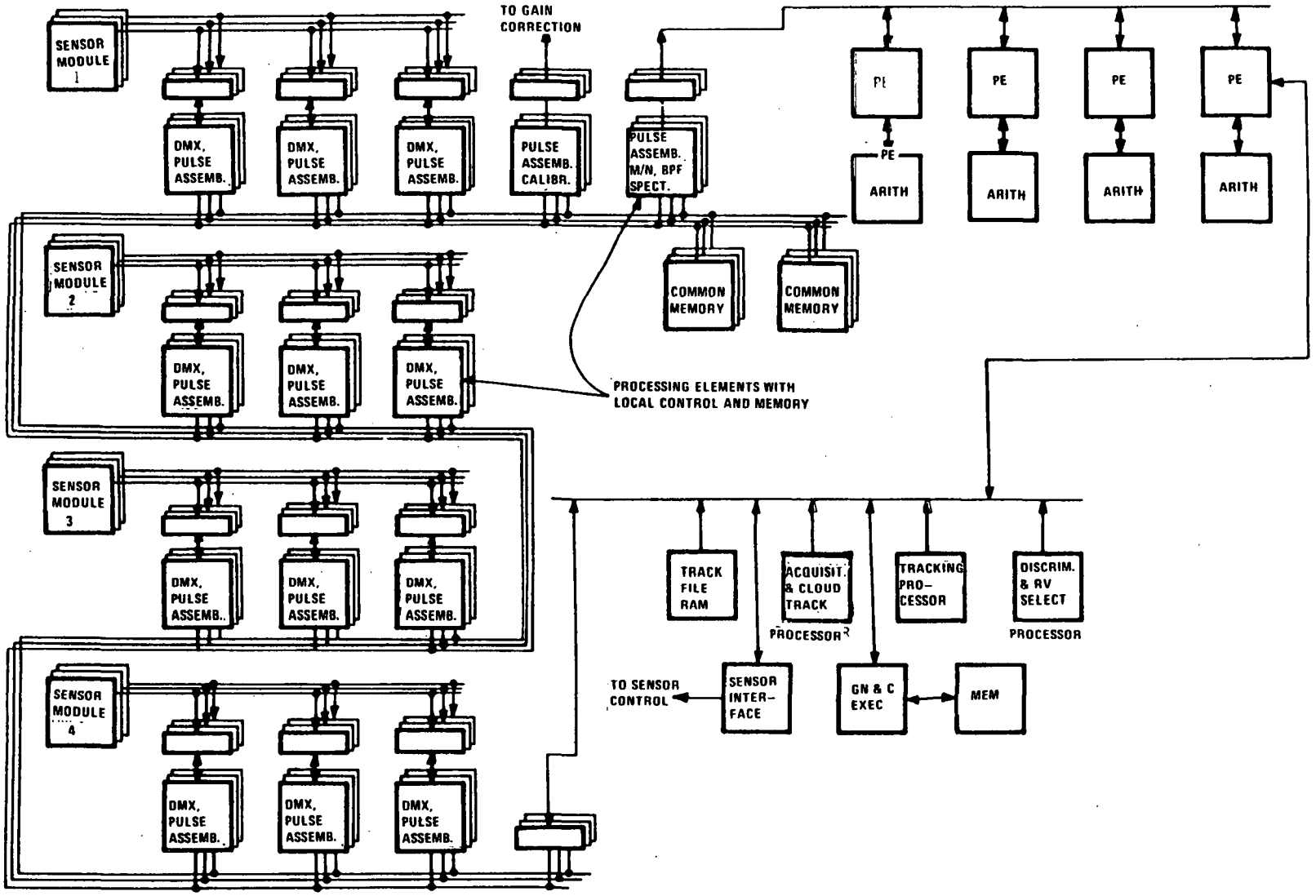
DISTRIBUTED HARDWARE:

- NATURALLY DISCIPLINES SOFTWARE DESIGN AND IMPLEMENTATION
- MAKES STRUCTURED SOFTWARE DESIGN THE EASIEST APPROACH
- PROVIDES TIGHTLY DISCIPLINED INTERFACES
- LIMITS EFFECTS OF SOFTWARE FAULTS/BUGS/CHANGES

DATA DRIVEN SOFTWARE STRUCTURE:

- ALLOWS GRACEFUL EXPANSION/CONTRACTION (REQUIRES NO SOFTWARE CHANGES)
- PROVIDES INHERENT FAULT TOLERENCE

PRELIMINARY ARCHITECTURAL CONFIGURATION, ALTERNATIVE A



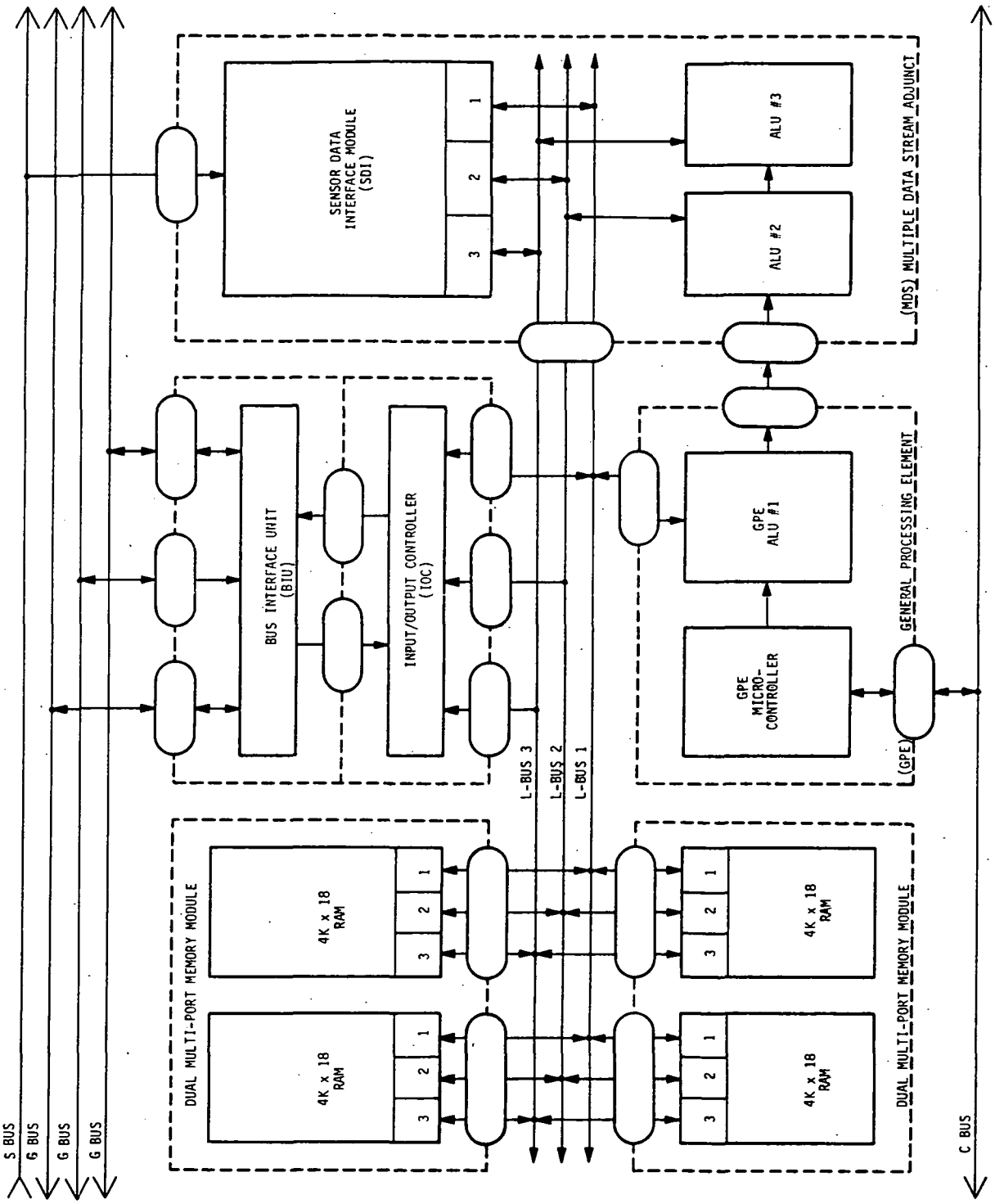
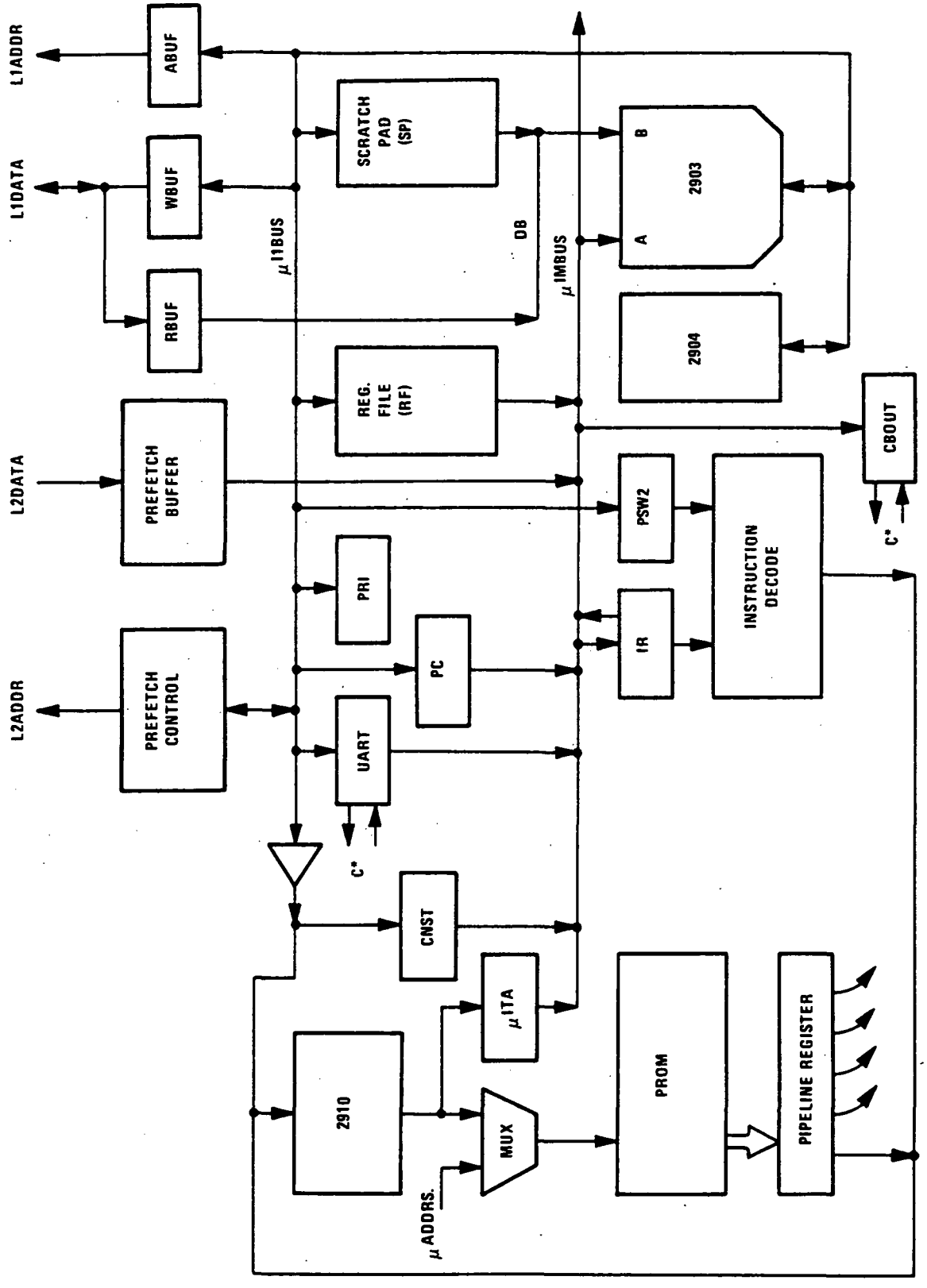


Figure 3.2. SIMD Functional Block Diagram

GPE BLOCK DIAGRAM

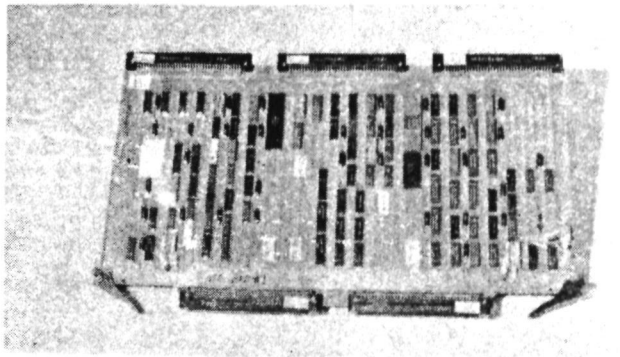


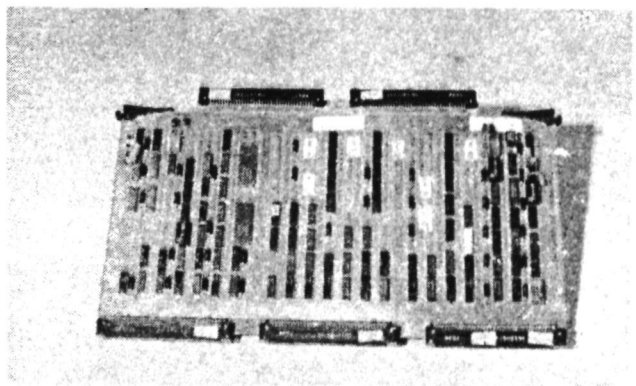
PROCESSING ELEMENT CPU

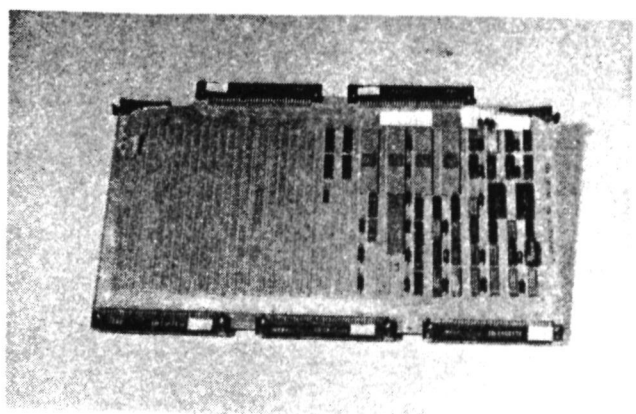
- SINGLE CPU MODULE USED THROUGHOUT SYSTEM

CHARACTERISTICS

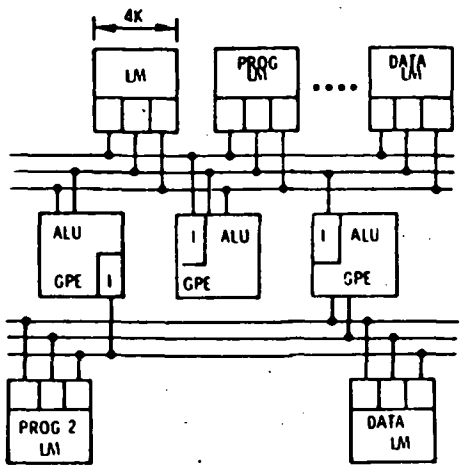
- 1 MIPS PERFORMANCE (~ 1 μs ADD)
- 16 BIT PARALLEL ORG.
- 65K WORD ADDRESS SPACE
- ASYNCHRONOUS INTERFACES
- MEMORY MAPPED I/O
- LARGE INSTRUCTION SET WITH OPERATING SYSTEMS PRIMATIVES AND FLOATING POINT
- CAPABLE OF AUGMENTATION (I.E., MORE CAPABILITY FOR PULSE MATCH, ETC.)
- 220 LOW POWER SCHOTTKY AND SCHOTTKY TTL IC's
- 96 BIT MICROINSTRUCTION WORD
- MAJOR SECTIONS: ALU, MICROCONTROLLER, CONTROL BUS INTERFACE, REAL-TIME CLOCK



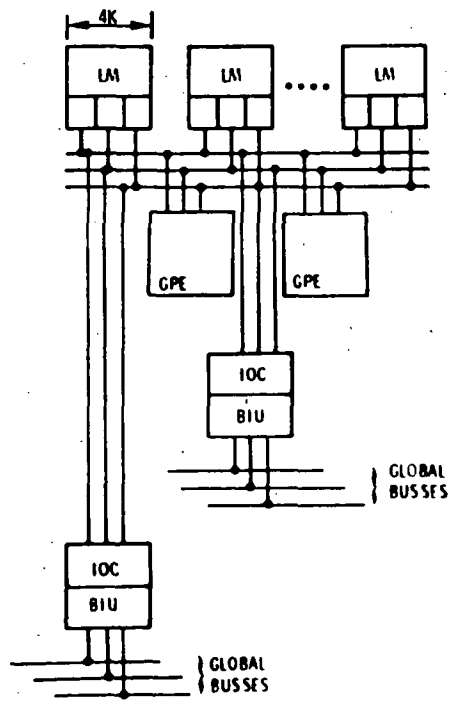




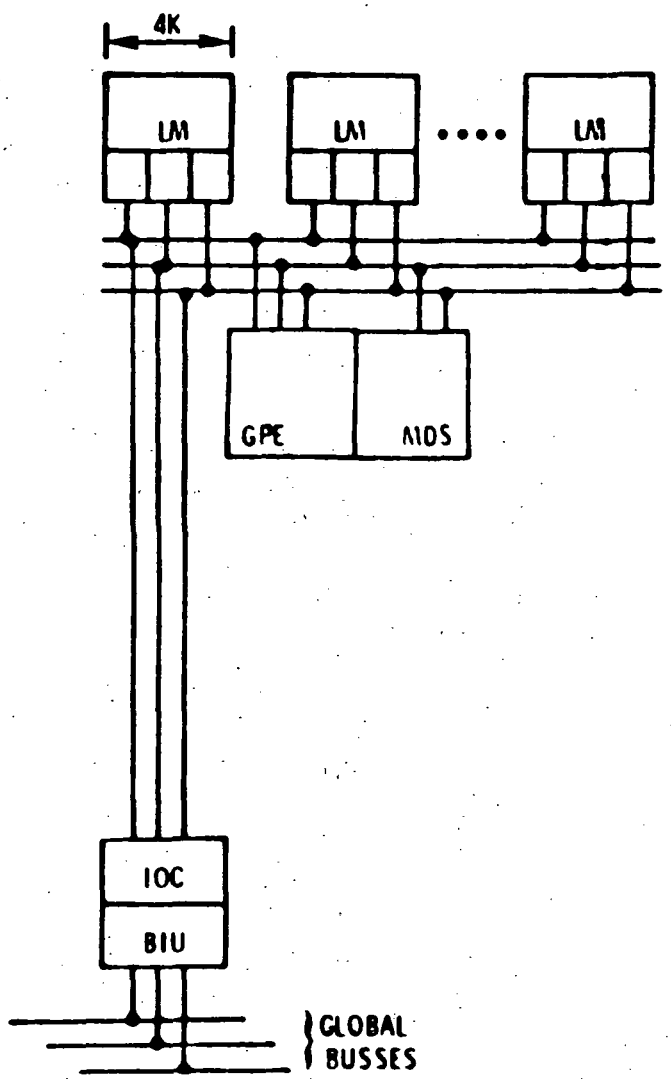
ALTERNATIVE CONFIGURATIONS OF MMBC MODULES



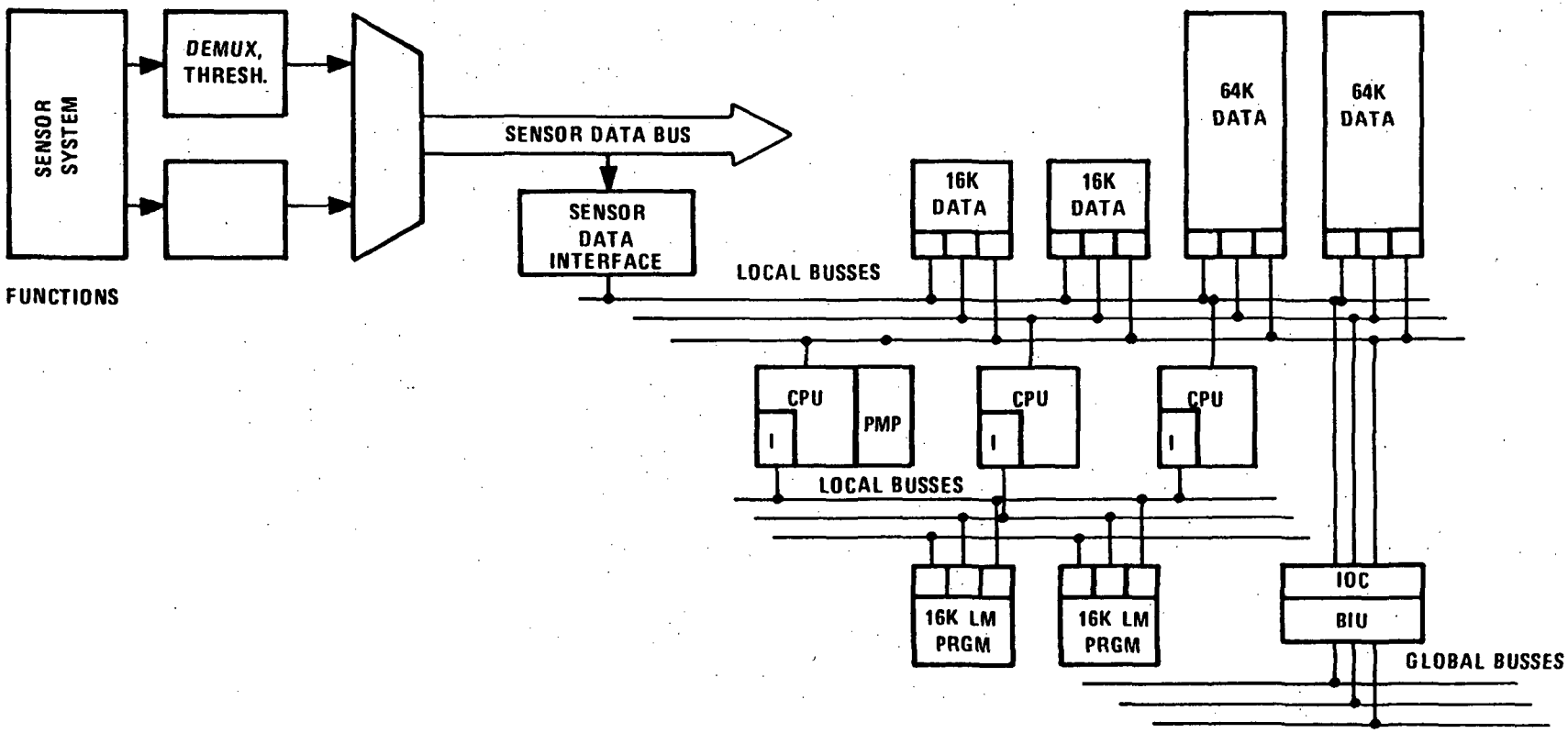
ALTERNATIVE CONFIGURATIONS OF MMBC MODULES



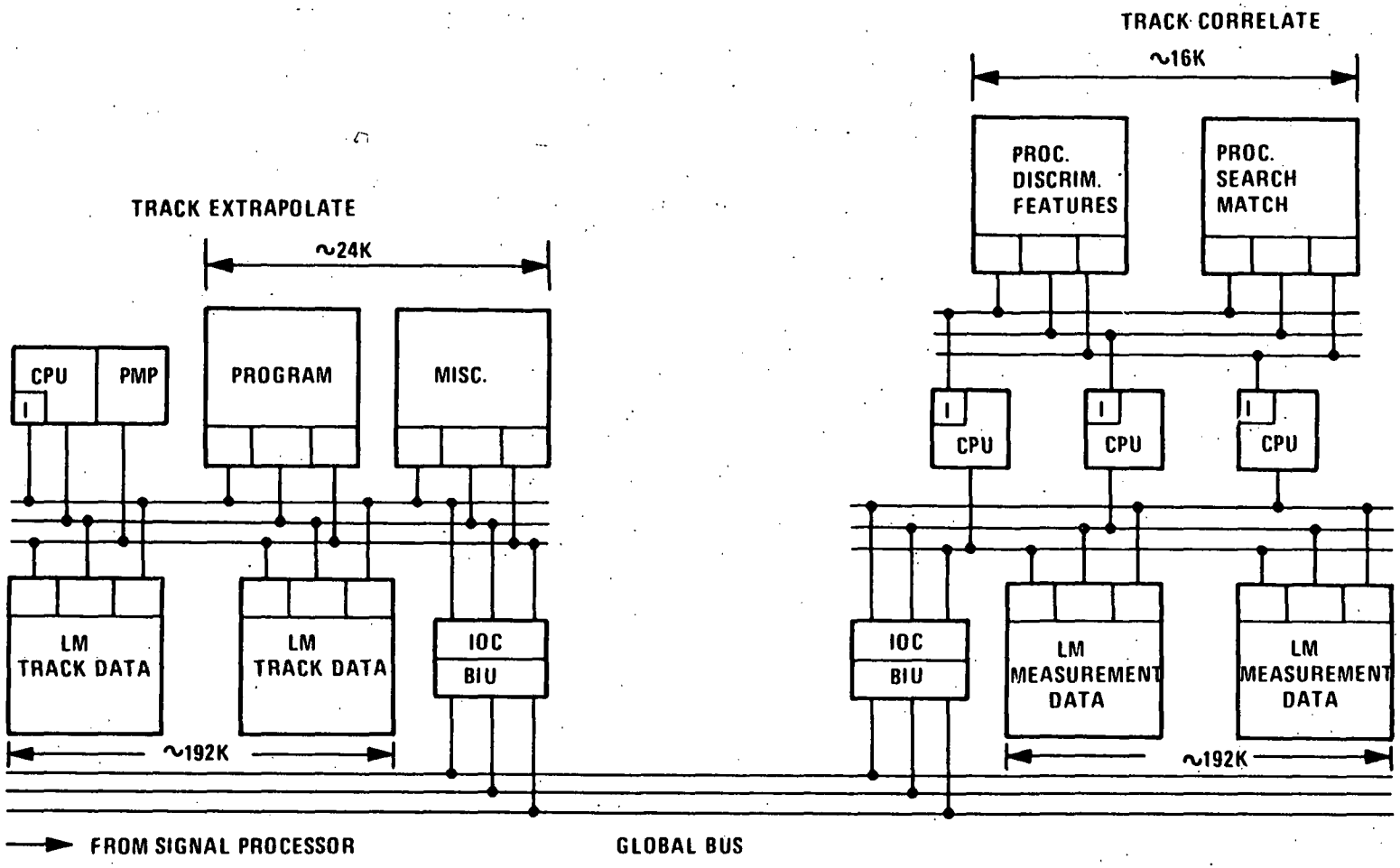
ALTERNATIVE CONFIGURATIONS OF MMBC MODULES



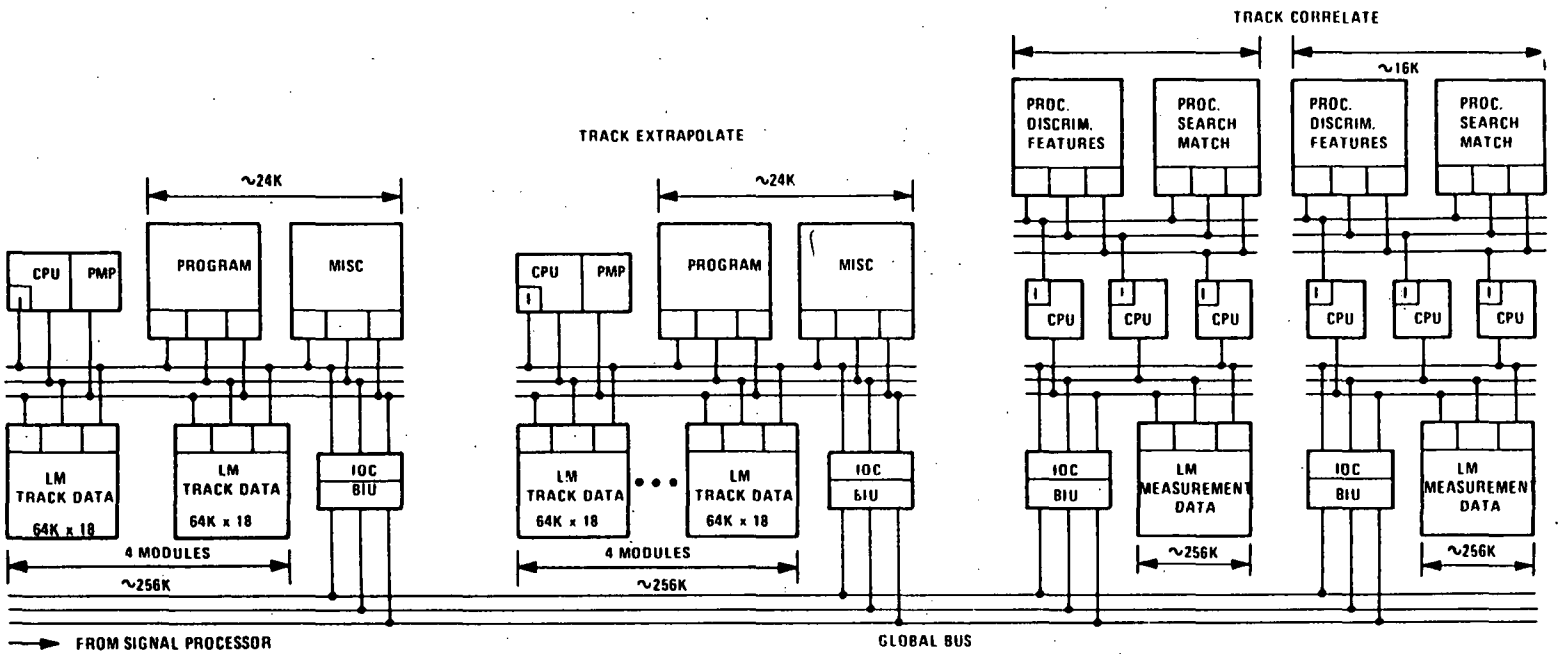
EWA PROGRAMMABLE SIGNAL PROCESSOR



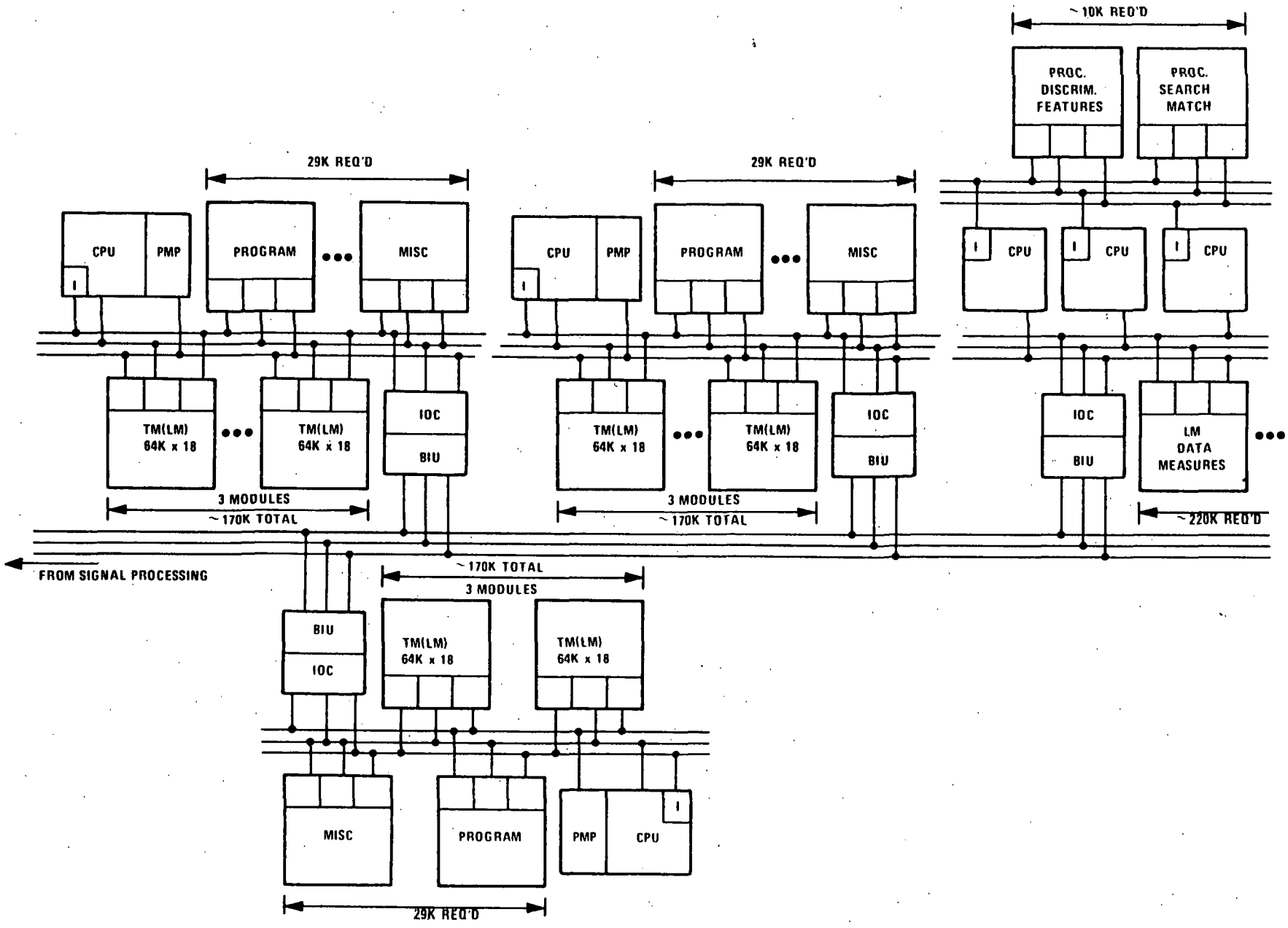
EWA DATA PROCESSING SYSTEM



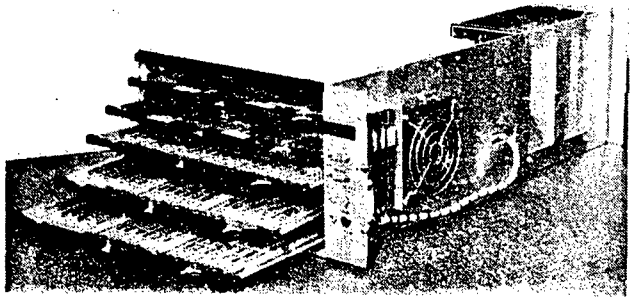
NEAR-TERM BMD DATA PROCESSING



ABMD DATA PROCESSING

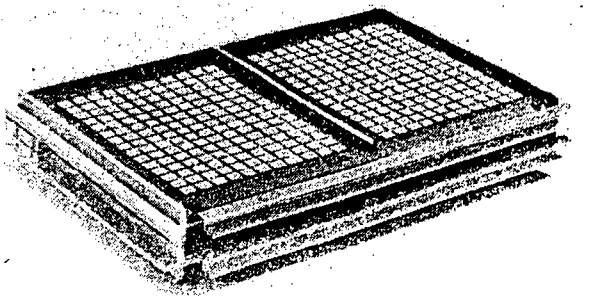


HARDWARE PACKAGING OPTIONS



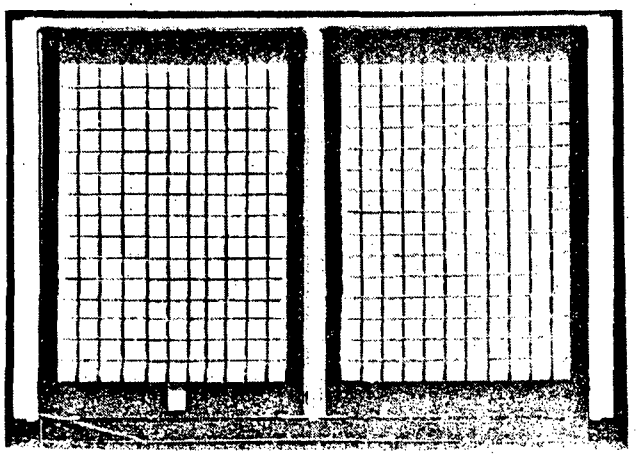
LAB PROTOTYPE IMPLEMENTATION -
CPU ON 2 15"x15" BOARDS

OPTION I



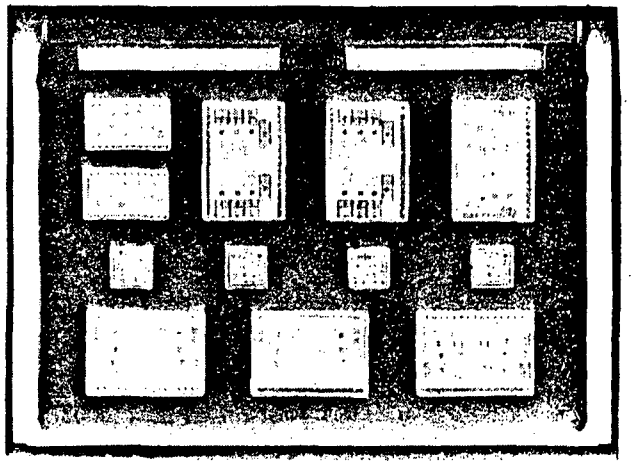
CPU ON DOUBLE MODULE -
FLAT PAKS AND CHIP CARRIERS

OPTION II



CPU ON SINGLE MODULE -
LSICs, FLAT PAKS AND
CHIP CARRIERS

OPTION III

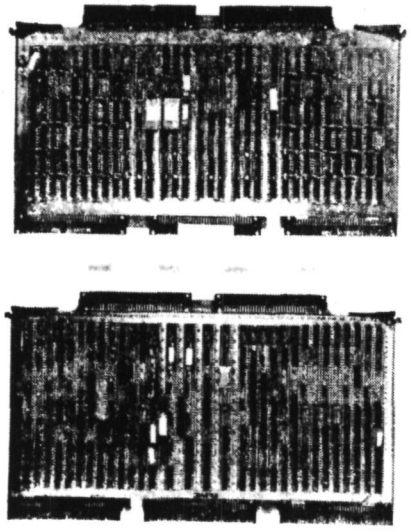


CPU ON HALF MODULE -
HYBRIDS AND LSICs

EXAMPLE OF SIZE REDUCTION FOR PACKAGING OPTIONS

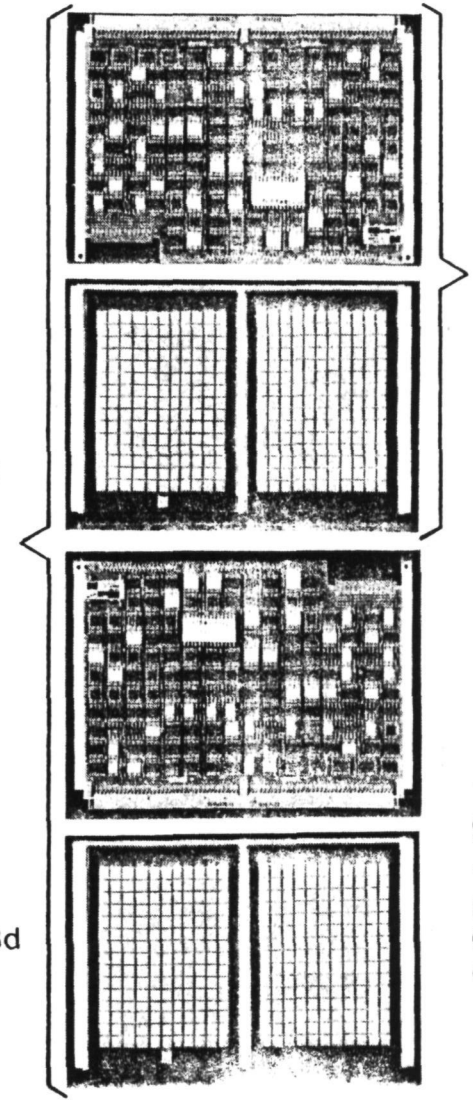
BUS INTERFACE UNIT

LAB PROTOTYPE



380 ICs
270 Sq Inches
56 Watts
DPs and WR Bds

OPTION 1
DOUBLE
MODULE
(BOTH
SIDES)

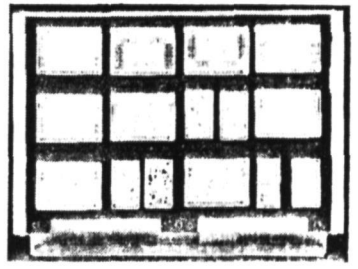


380 ICs
180 Sq Inches
56 Watts
FPs and Chip
Carrier on PC Bd

OPTION 2
SINGLE
MODULE
(BOTH
SIDES)

190 ICs
90 Sq Inches
≈20 Watts
LSICs, FPs
Chip Carriers
on PC Bd

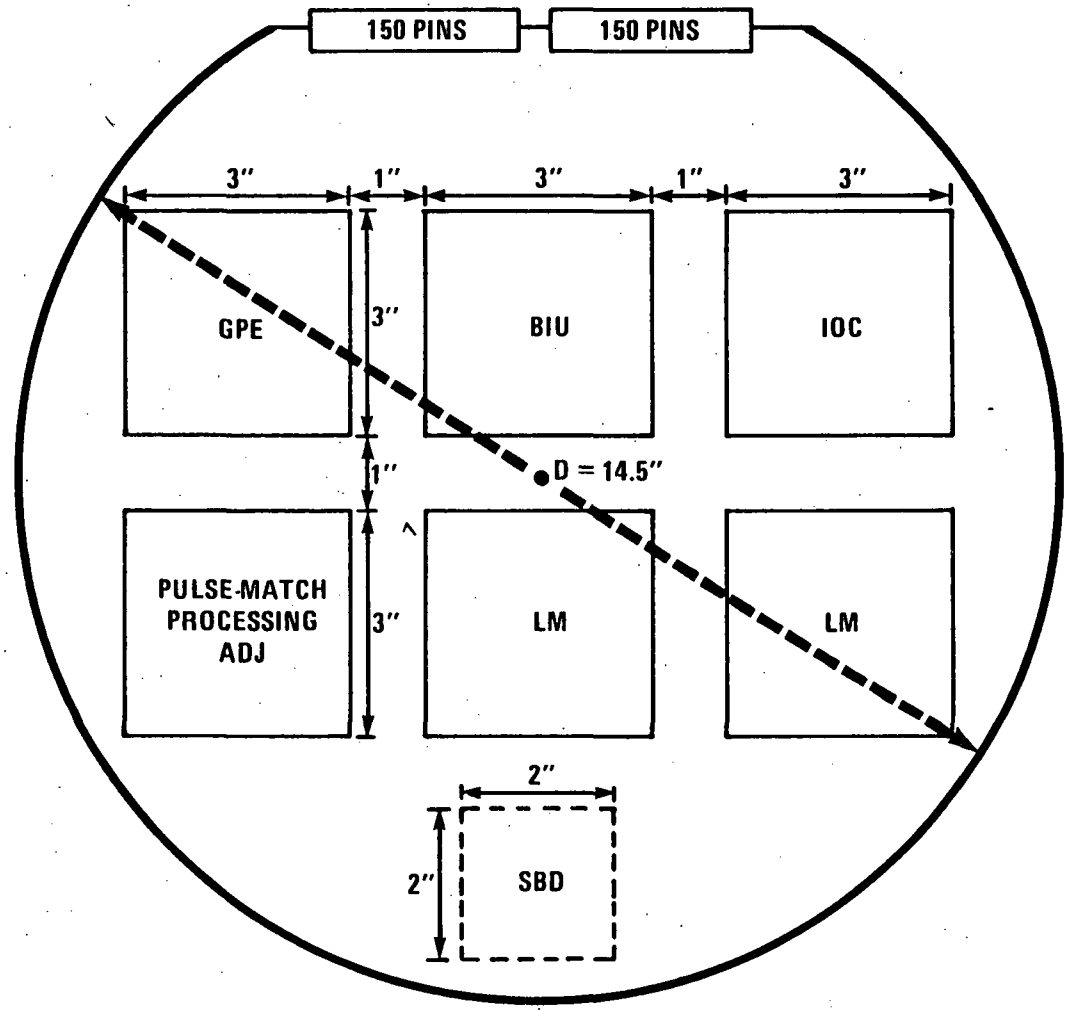
OPTION 3



½ MODULE
(1 SIDE)

190 ICs
45 Sq Inches
≈20 Watts
LSICs and Hybrid
on PC Bd

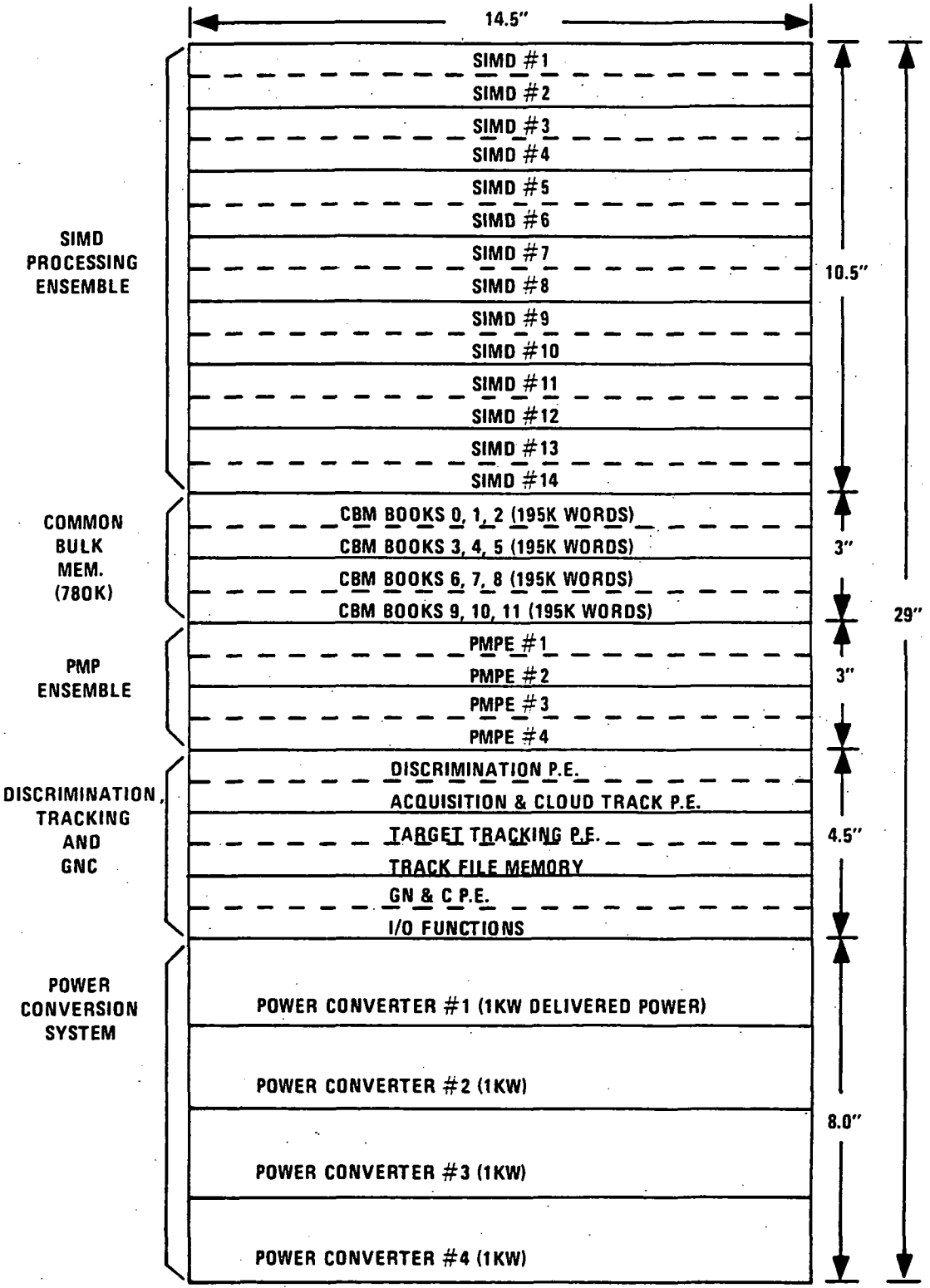
PACKAGING EXAMPLE
(PULSE-MATCH PROCESSOR MODULE)



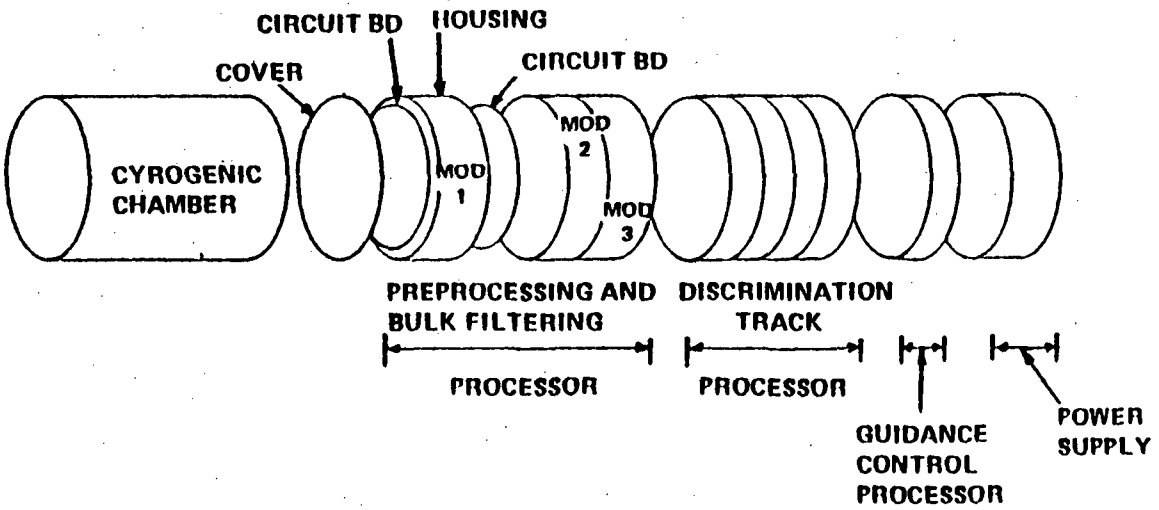
CHARACTERISTICS

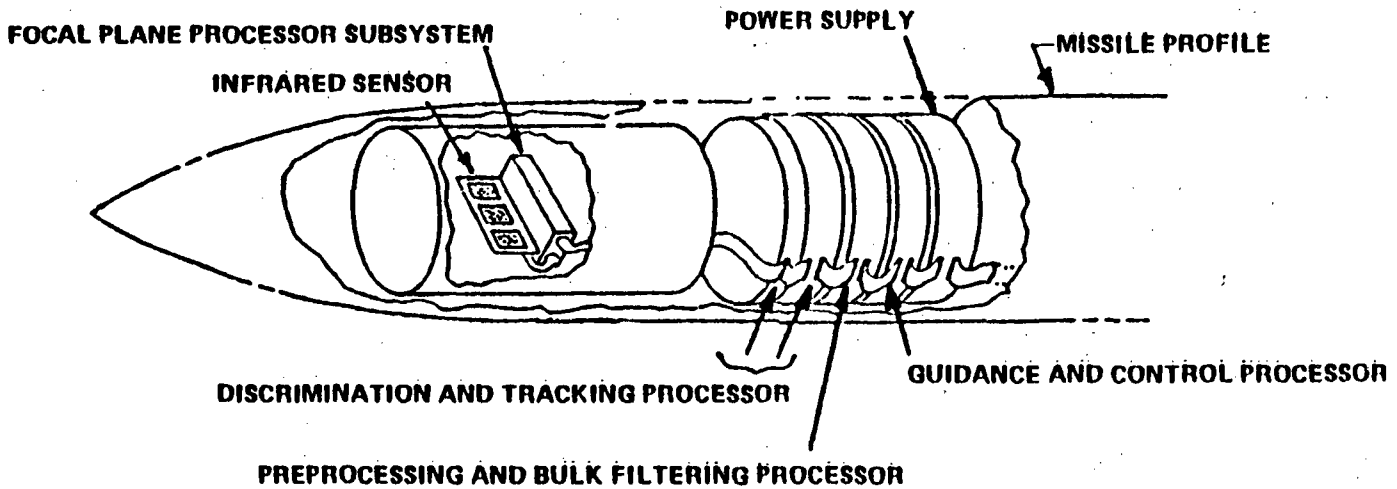
- 200 WATTS
- 14.5" DIAMETER
- TWO 150-PIN I/O CONN.
- OPTIONAL 2" X 2" (SBD) (SENSOR BUS DRIVER)
- 165 in^2
- 3/4" CENTERS BOARD SPACING

EXAMPLE OF HYBRID MMBC SYSTEM



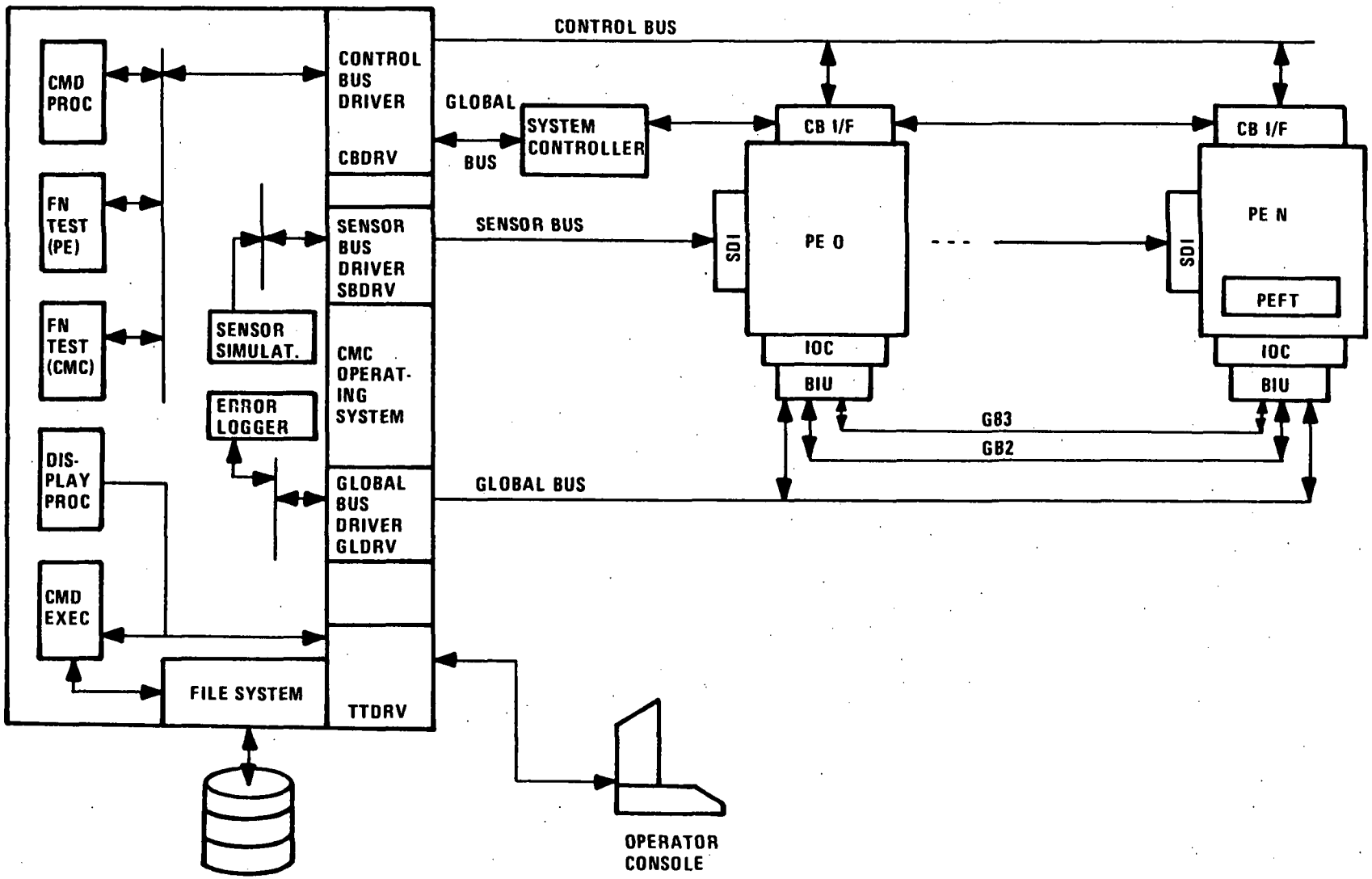
MODULAR MISSILE BORNE COMPUTERS





- ONBOARD BUDGETS FOR LDS PROBE APPEAR FEASIBLE

TEST BED

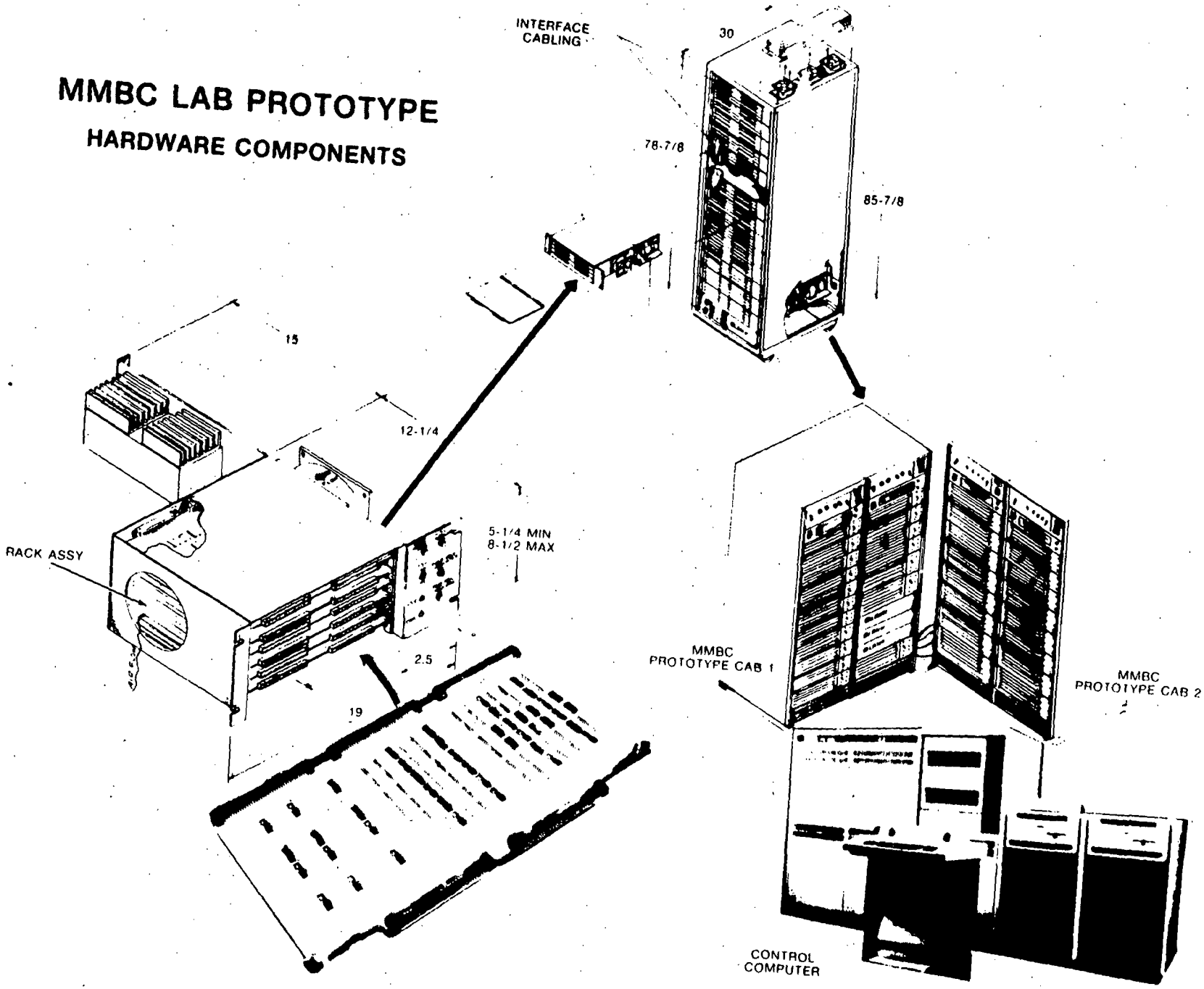

CONTROL BUS
CMD PROC
CONTROL BUS DRIVER
CBDRV
GLOBAL BUS
SYSTEM CONTROLLER
CB I/F
CB I/F
FN TEST (PE)
SENSOR BUS DRIVER SBDRV
SENSOR BUS
SDI
PE 0
SDI
PE N
PEFT
FN TEST (CMC)
SENSOR SIMULAT.
CMC OPERAT-ING SYSTEM
IOC
BIU
IOC
BIU
ERROR LOGGER
G83
GB2
DIS-PLAY PROC
GLOBAL BUS DRIVER GLDRV
GLOBAL BUS
CMD EXEC
FILE SYSTEM
TTDRV
OPERATOR CONSOLE

DIFFERENCES BETWEEN CURRENT SYSTEM & MMBC

CURRENT	MMBC
• SINGLE (OR SMALL NO.) OF PE'S	• MANY PE'S
• THRUPUT FIXED, ~10-20 MIPS MAX., 1-2 MIPS FLYABLE NOT EXPANDABLE	• EXPANDABLE, THRUPUT VARIABLE FROM ~1 TO 100 + MIPS; MAY BE TUNED TO APPLICATION REQUIREMENTS (FLYABLE)
• THRUPUT INVARIANT WITH APPLICATION	• THRUPUT DEPENDENT ON SW, SW STRUCTURE
• SOFTWARE STRUCTURE/ COMPLEXITY UNCONTROLLED	• STRUCTURED APPROACH EASIEST SW MODULARITY INHERENT, COMPLEXITY CONTROLLED; REQUIRES CAREFUL PARTITIONING
• GENERALLY SUBJECT TO SINGLE POINT FAILURES	• EASILY AVOIDS SINGLE POINT FAILURES; GRACEFULLY DEGRADES WITH FAULTS
• OVERHEAD FUNCTIONS GENERALLY IN SW	• HARDWARE IMPLEMENTS MOST OVERHEAD FUNCTIONS (ESPECIALLY THOSE DUE TO MULTI COMPUTER CONFIGURATION

MMBC LAB PROTOTYPE

HARDWARE COMPONENTS



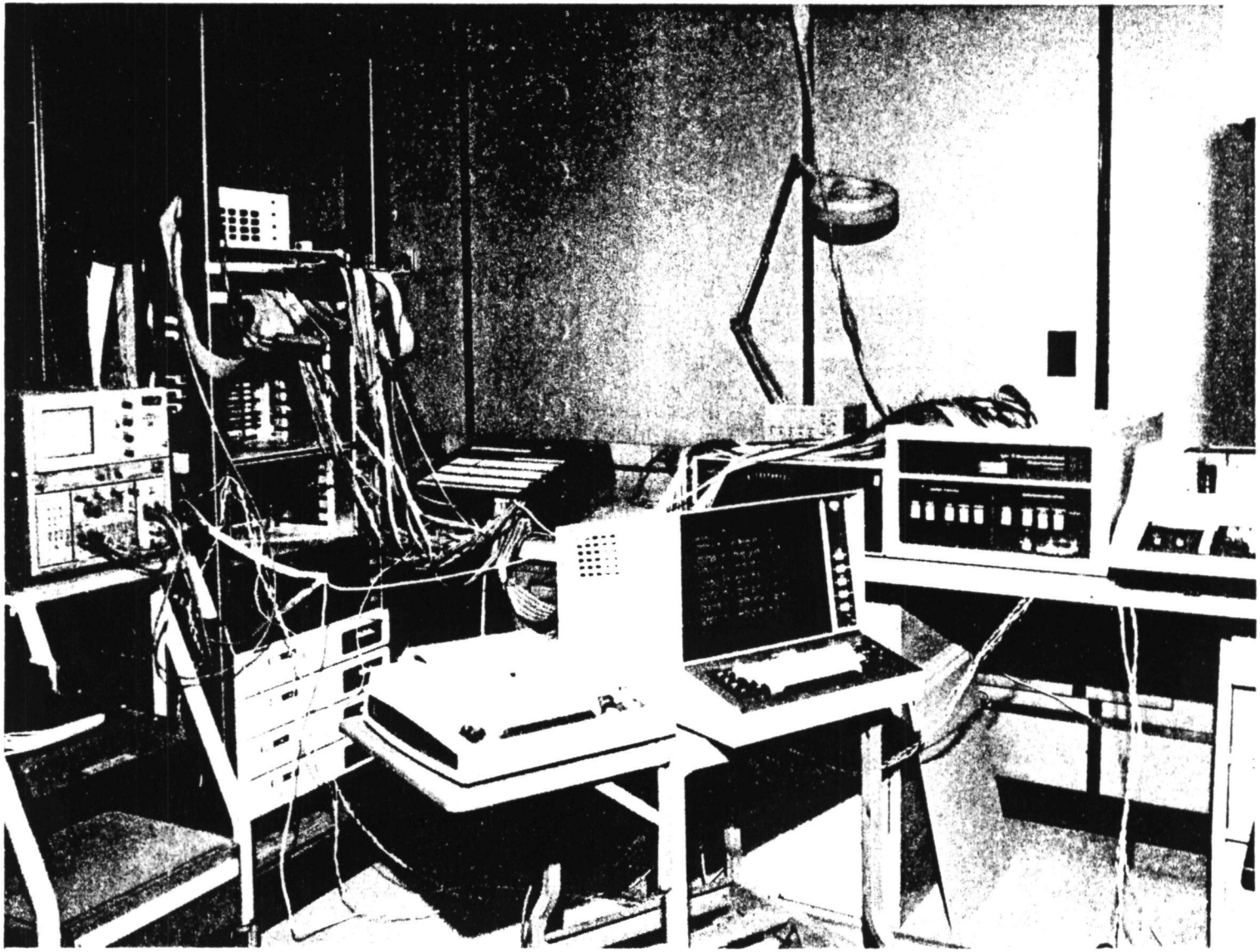


Figure 16. CPU Bus Display Box

Page intentionally left blank

Page intentionally left blank

MICROPROCESSOR-CONTROLLED TELEMETRY SYSTEM

Paul Holtzman and Stephen D. Hawkins
RCA
Princeton, New Jersey

A spacecraft telemetry unit completely controlled by a microprocessor was developed at RCA Astro-Electronics. This unit, the Programmable Information Processor (PIP), must sample 650 sources of analog and discrete housekeeping information plus digital data generated by spacecraft computers. The data must be formatted for serial transmission to spacecraft data storage devices or direct transmission to the ground via a telemetry beacon link. The choice of microprocessor to accomplish these tasks was influenced by requirements for 1) low power, 2) survival in radiation environment, and 3) instruction execution time of 6 microseconds. The RCA CDP1802 CMOS microprocessor was ultimately selected. The entire hardware system including ROM and RAM memories utilizes CMOS technology and dissipates only 5 watts of power. Many commandable modes of operation are possible with the PIP. Four major modes have differing format structure, sampling rates and output data rates. Submodes enable selected data channels to be uniformly sampled at higher sampling rates for diagnostic purposes and also enable a dump of memory data from various spacecraft computers and of the PIP itself. The sampling sequences for housekeeping telemetry points are determined by preassigned tables within the PIP software. An additional feature is the capability to restructure these tables by a memory load. Complete dual redundancy is employed in the PIP such that no single failure may compromise the mission requirements.

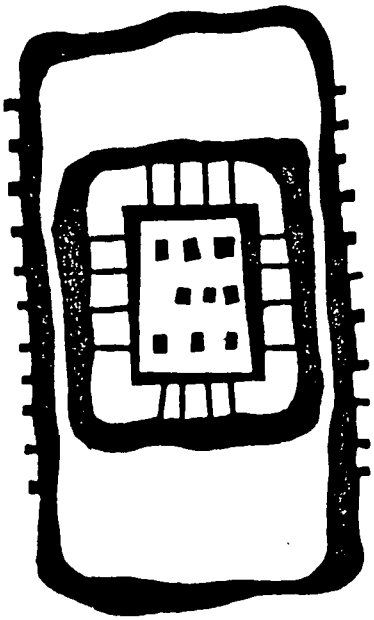
Page intentionally left blank

Page intentionally left blank

MICROCOMPUTER ARRAY PROCESSOR SYSTEM

Kenneth D. Slezak
Goodyear Aerospace Corporation
Akron, Ohio

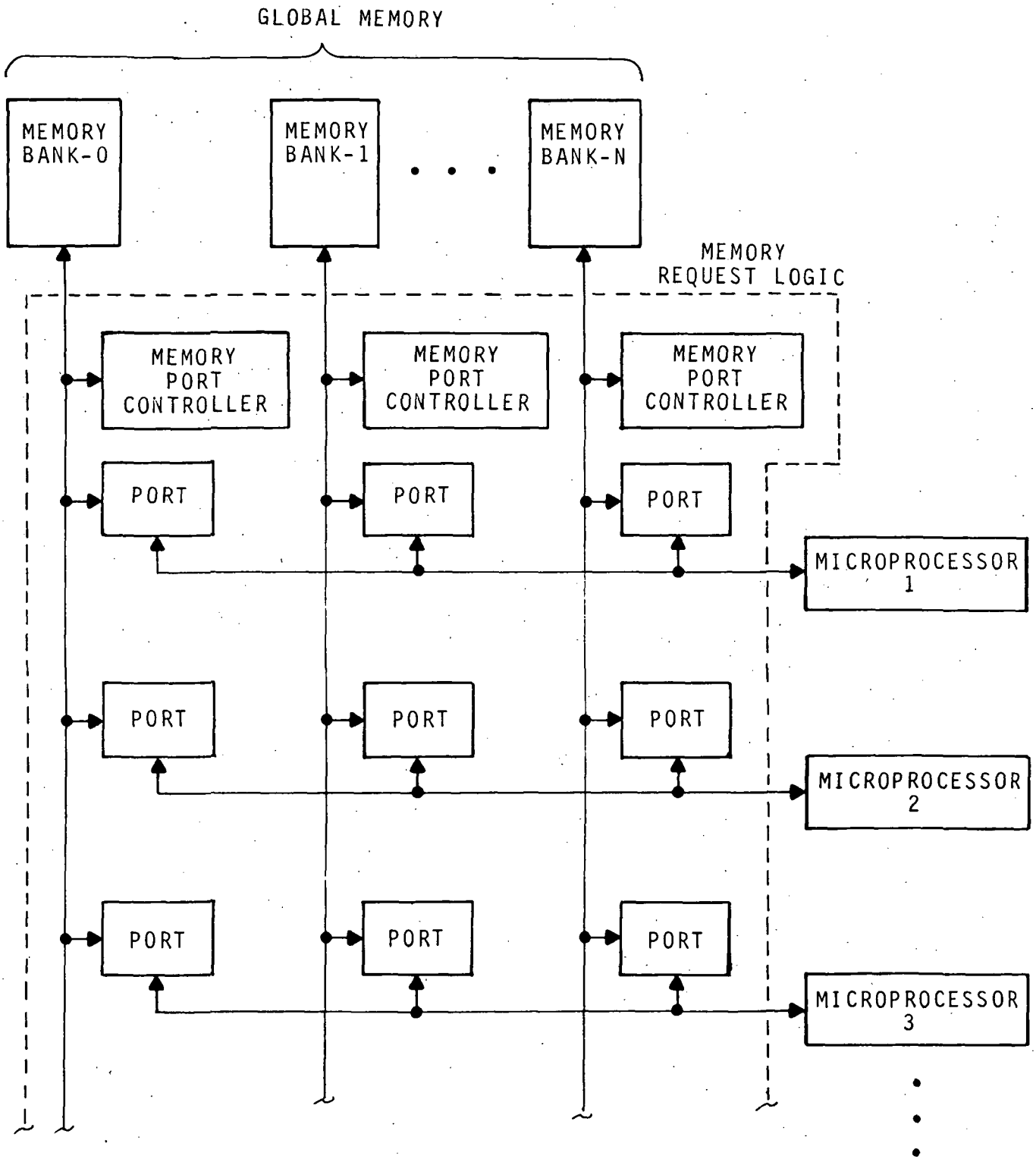
The Microcomputer Array Processor System (MAPS) is a programmable multiprocessor computer system designed for Electronic Warfare applications for the Air Force Avionics Laboratory (AFAL). The system architecture retains many of the classic multiprocessor design concepts including a master-slave relationship among its microprocessors under the control of a single operating system in a tightly coupled structure. Each processor is a 32-bit programmable computer with its own dedicated memory and a capability to execute approximately 4 million instructions a second. In addition to the dedicated memory, each processor can communicate with numerous banks of common memory (referred to as global memory). The various global memory modules and their communication structure serve to tie the individual processors together in a symmetrical multiprocessor computer architecture. The multiprocessor system is modular and can contain as few as 2 and as many as 8 processors coupled with from 1 to 16 banks of global memory and executes 32 million instructions per second. Expansions beyond these limits are possible if every processor does not have to have access to every global memory module. Currently, a 4 processor system (with 3 banks of global memory) is installed at Wright Patterson Air Force Base for use by AFAL. This system will be expanded to 6 processors during 1980. This multiprocessor subsystem is approximately 1.6 cubic feet and consumes under 400 watts of power.

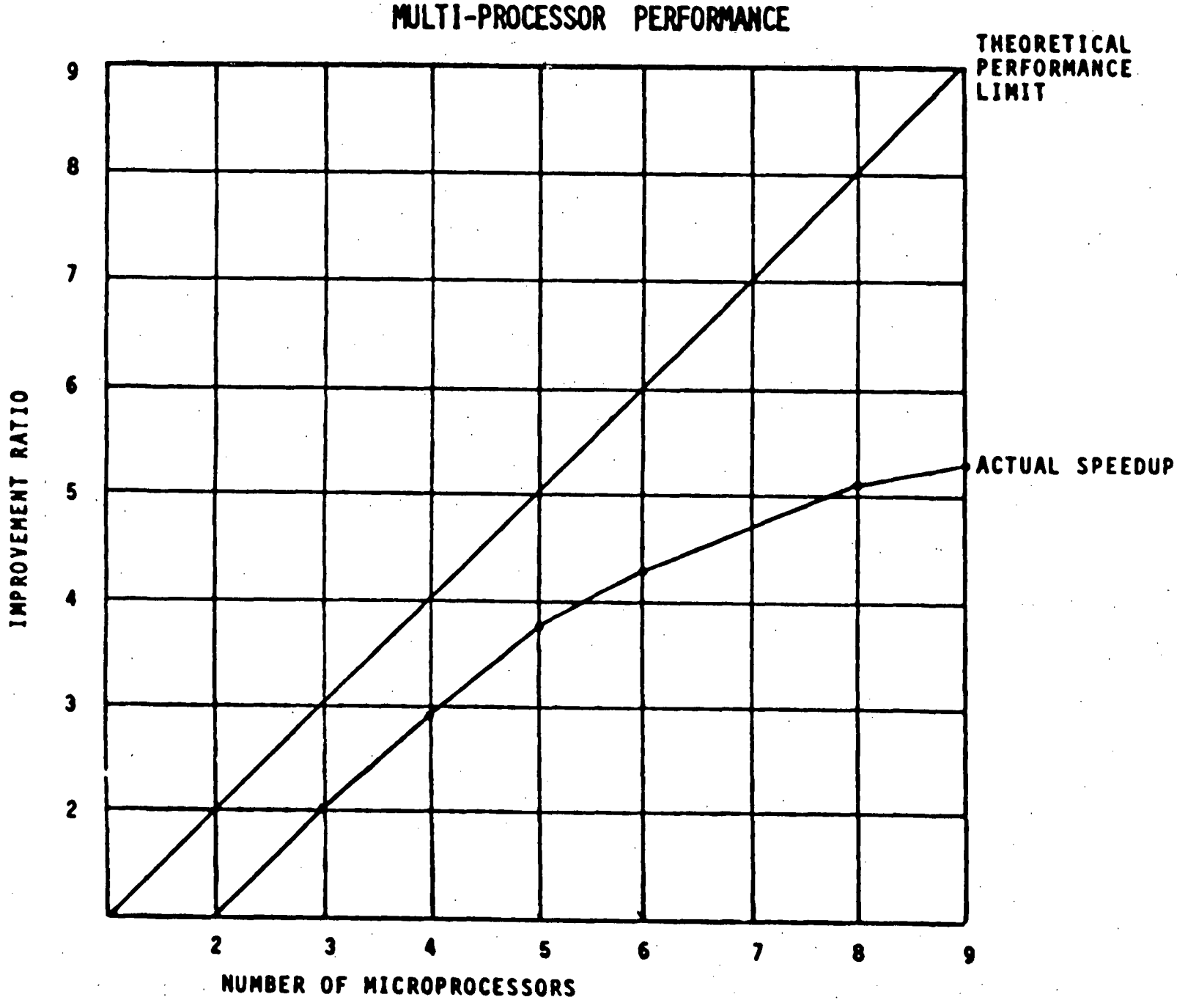


MICROCOMPUTER ARRAY PROCESSOR

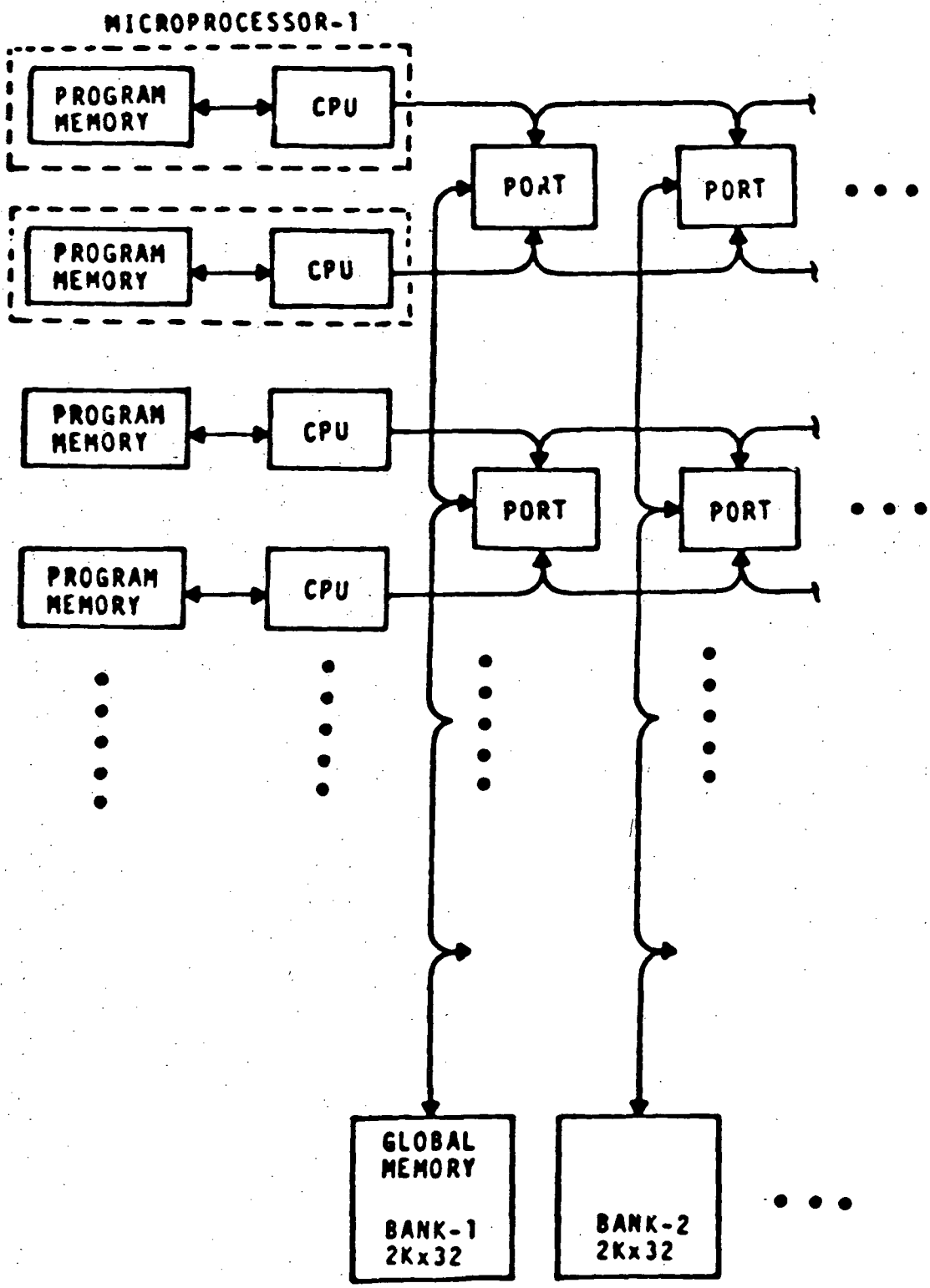
MULTIPROCESSOR SYSTEM ATTRIBUTES

- TASK
- SYMMETRY
- COMMUNICATION
- PROCESSOR INTELLIGENCE

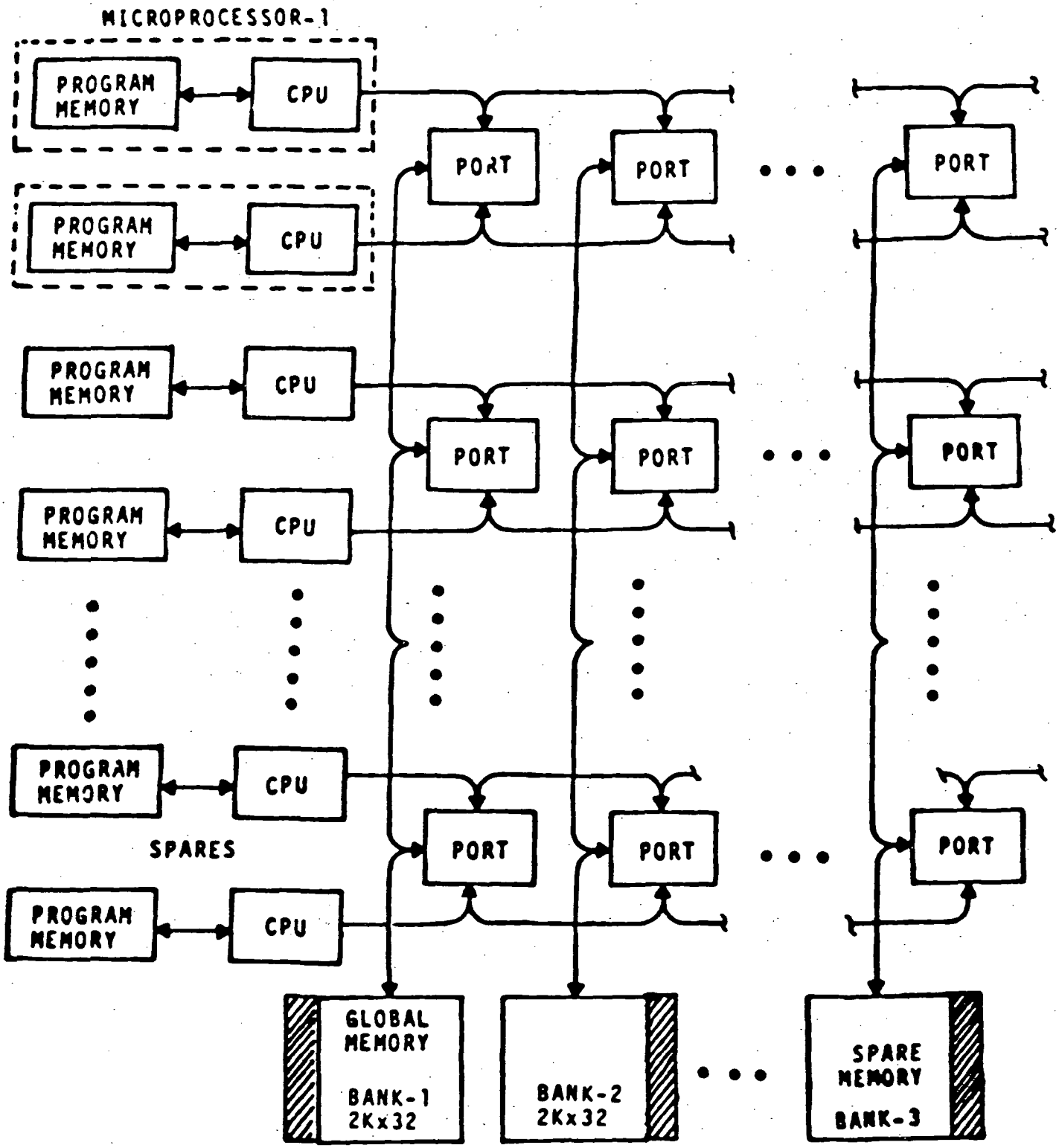

GLOBAL MEMORY
MEMORY BANK-0
MEMORY BANK-1
MEMORY BANK-N
MEMORY REQUEST LOGIC
MEMORY PORT CONTROLLER
MEMORY PORT CONTROLLER
MEMORY PORT CONTROLLER
PORT
PORT
PORT
PORT
PORT
PORT
PORT
PORT
PORT
MICROPROCESSOR 1
MICROPROCESSOR 2
MICROPROCESSOR 3


MULTI-PROCESSOR PERFORMANCE
THEORETICAL PERFORMANCE LIMIT
ACTUAL SPEEDUP
IMPROVEMENT RATIO
NUMBER OF MICROPROCESSORS
2
3
4
5
6
7
8
9

FAULT TOLERANT MAP ARCHITECTURE

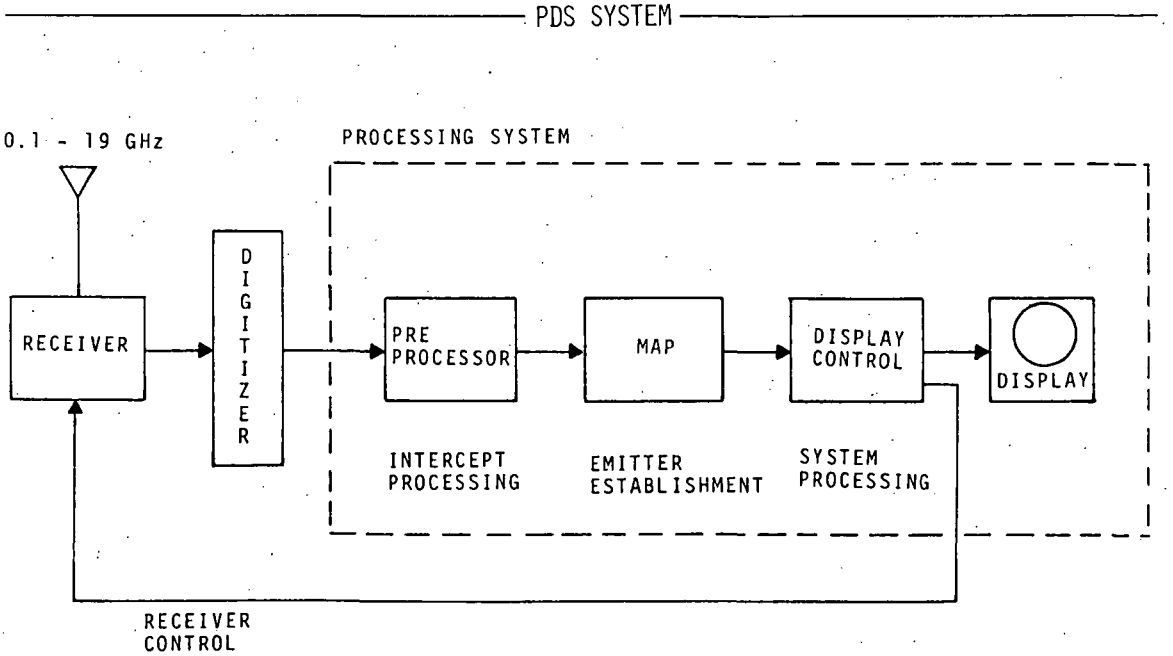


FAULT TOLERANT MAP ARCHITECTURE

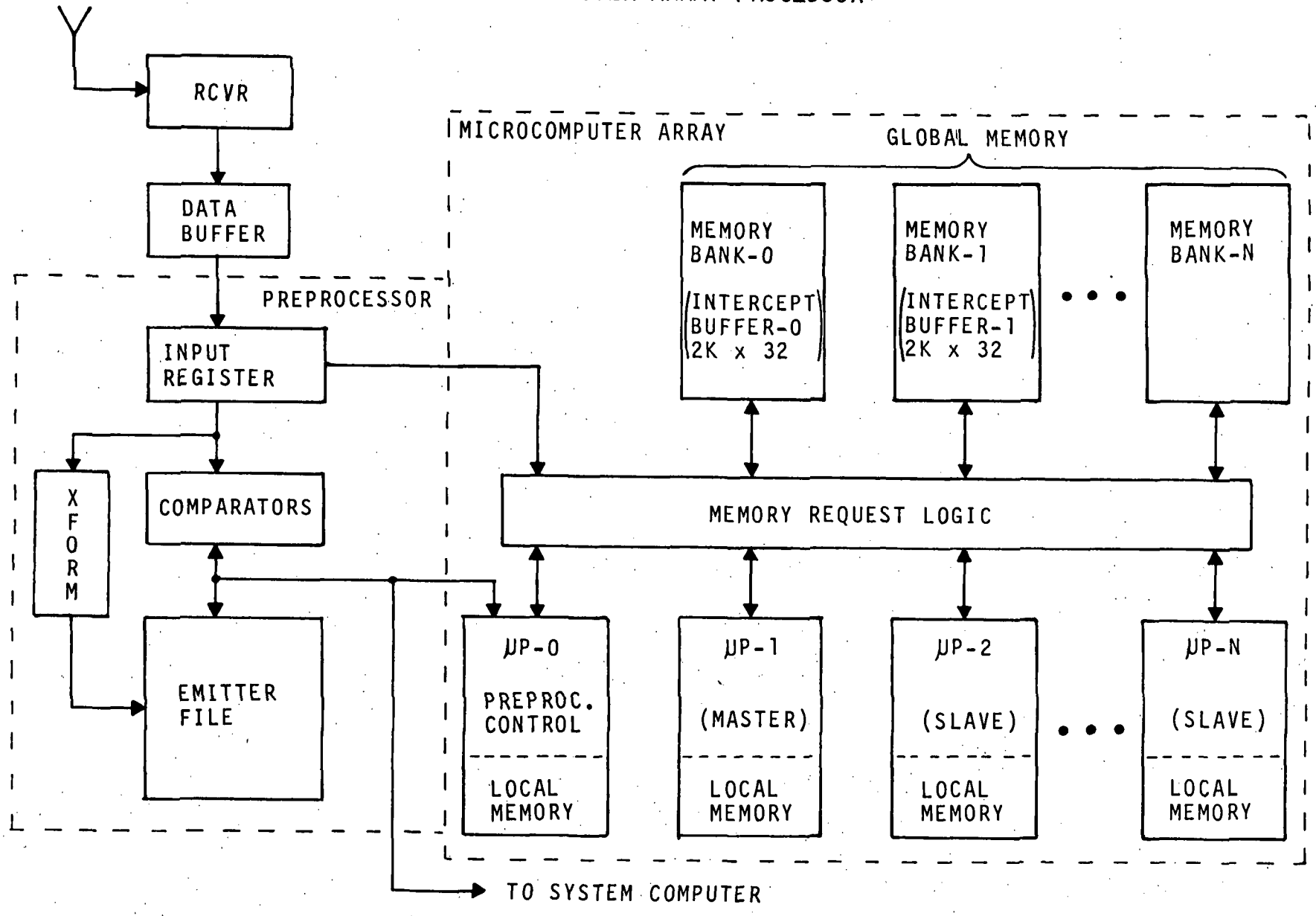


MAP PROCESSING FUNCTIONS

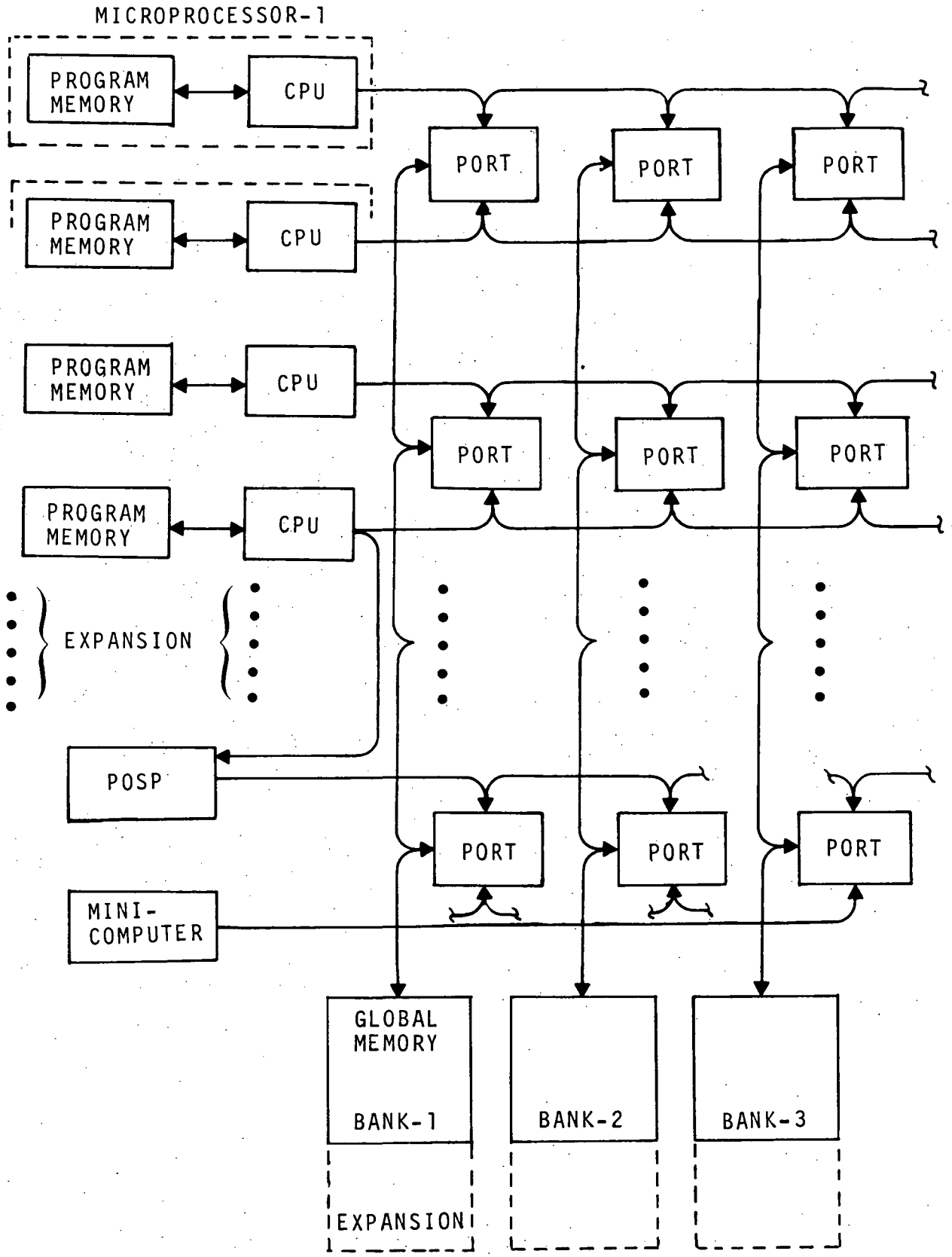
- ESTABLISHES FILE OF ACTIVE EMITTERS
  - . DETERMINES PRI
  - . REPORTS PRESENCE OF NEW EMITTERS.
- TRACKS ESTABLISHED EMITTERS
  - . TRACKS IN TIME AND ANGLE
- DELETES INACTIVE EMITTERS
- CAPABILITY FOR
  - . SCAN RATE DETERMINATION
  - . EMITTER TYPE IDENTIFICATION
  - . RECEIVER CONTROL
  - . POWER MANAGEMENT



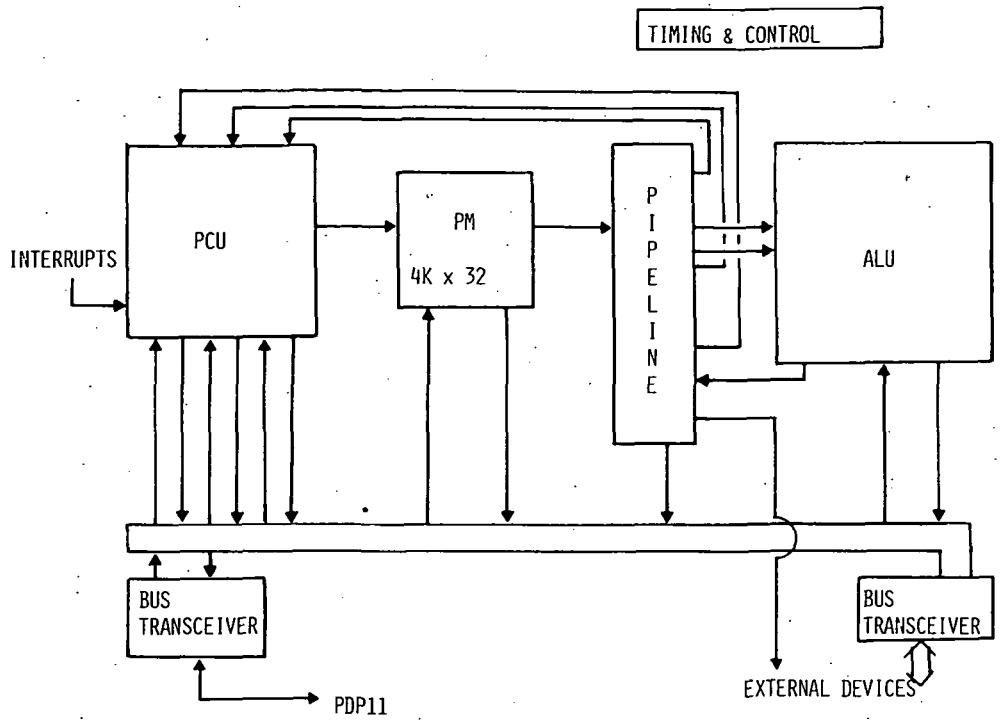
MICROCOMPUTER ARRAY PROCESSOR



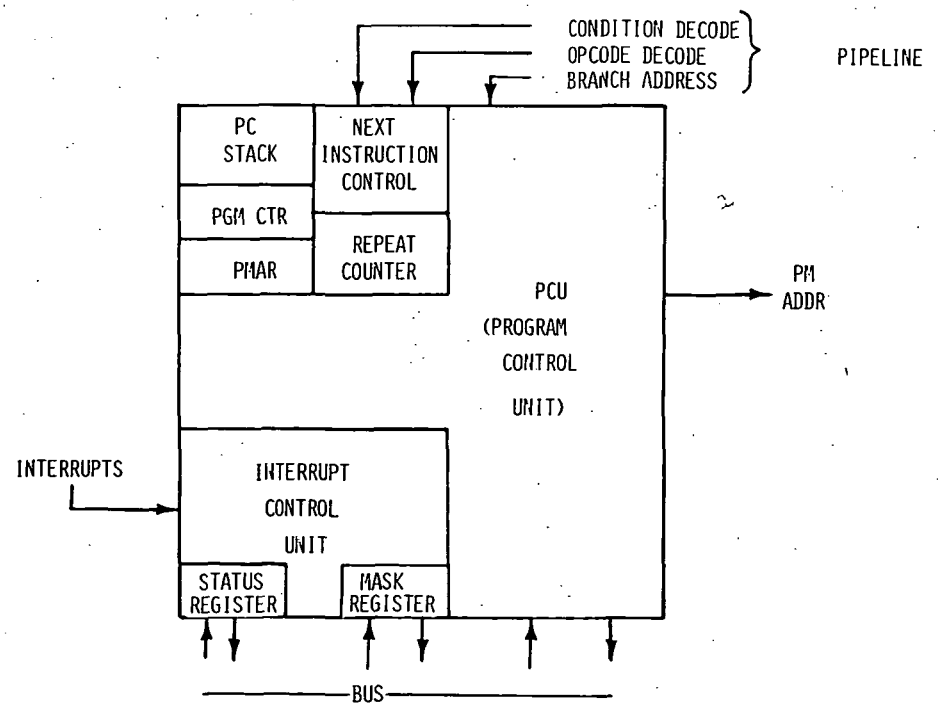
MAP ARCHITECTURE



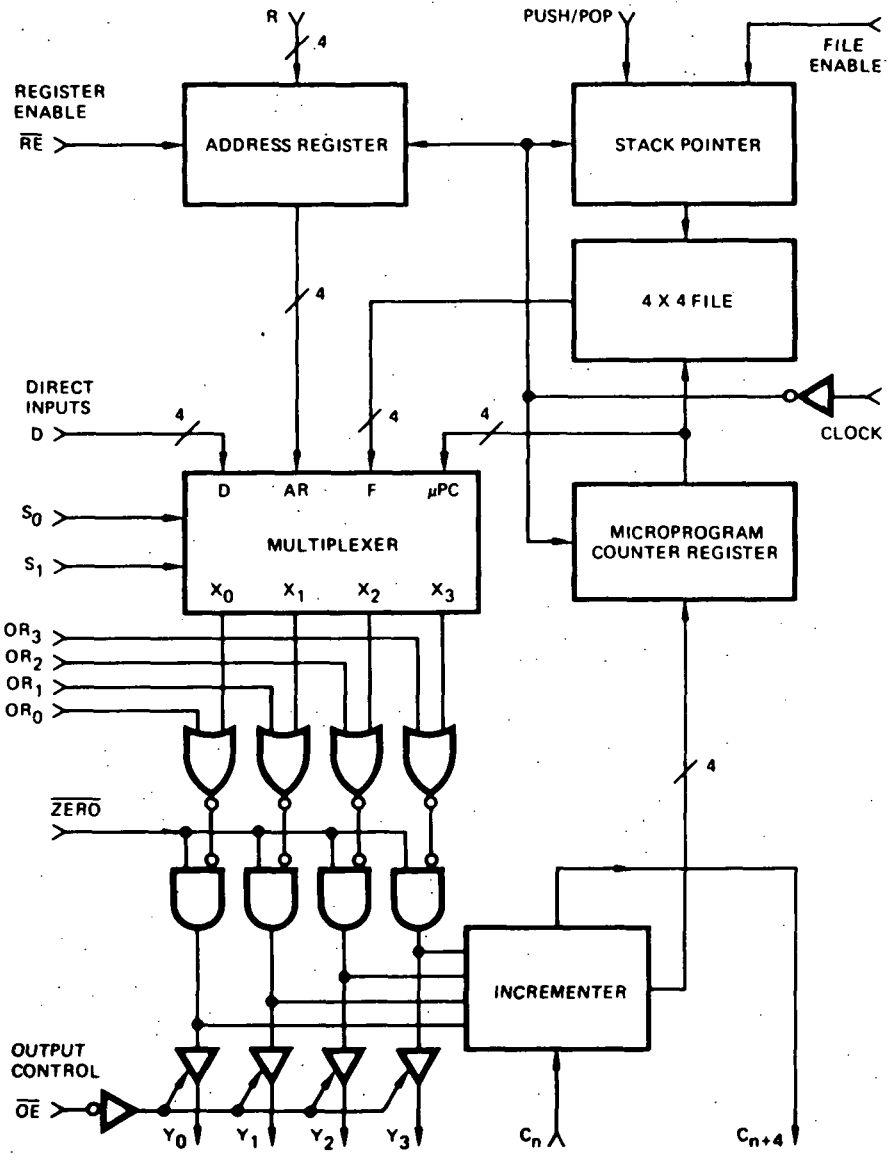
PROCESSOR



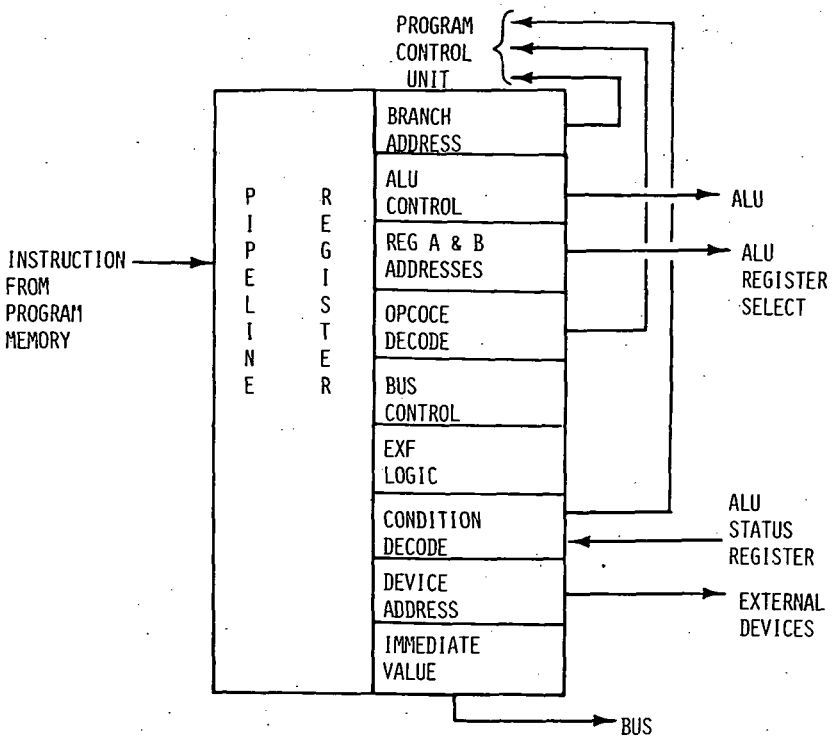
PCU



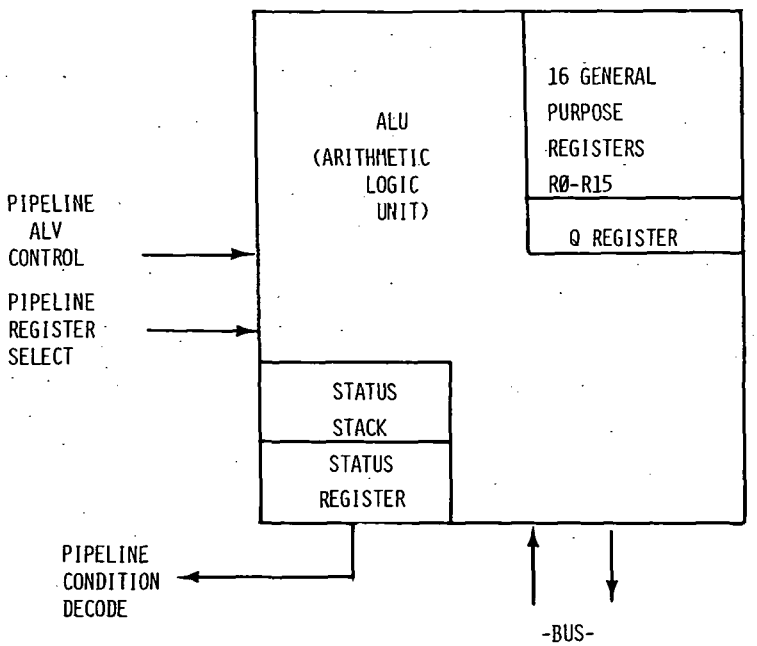
MICROPROGRAM SEQUENCER
BLOCK DIAGRAM



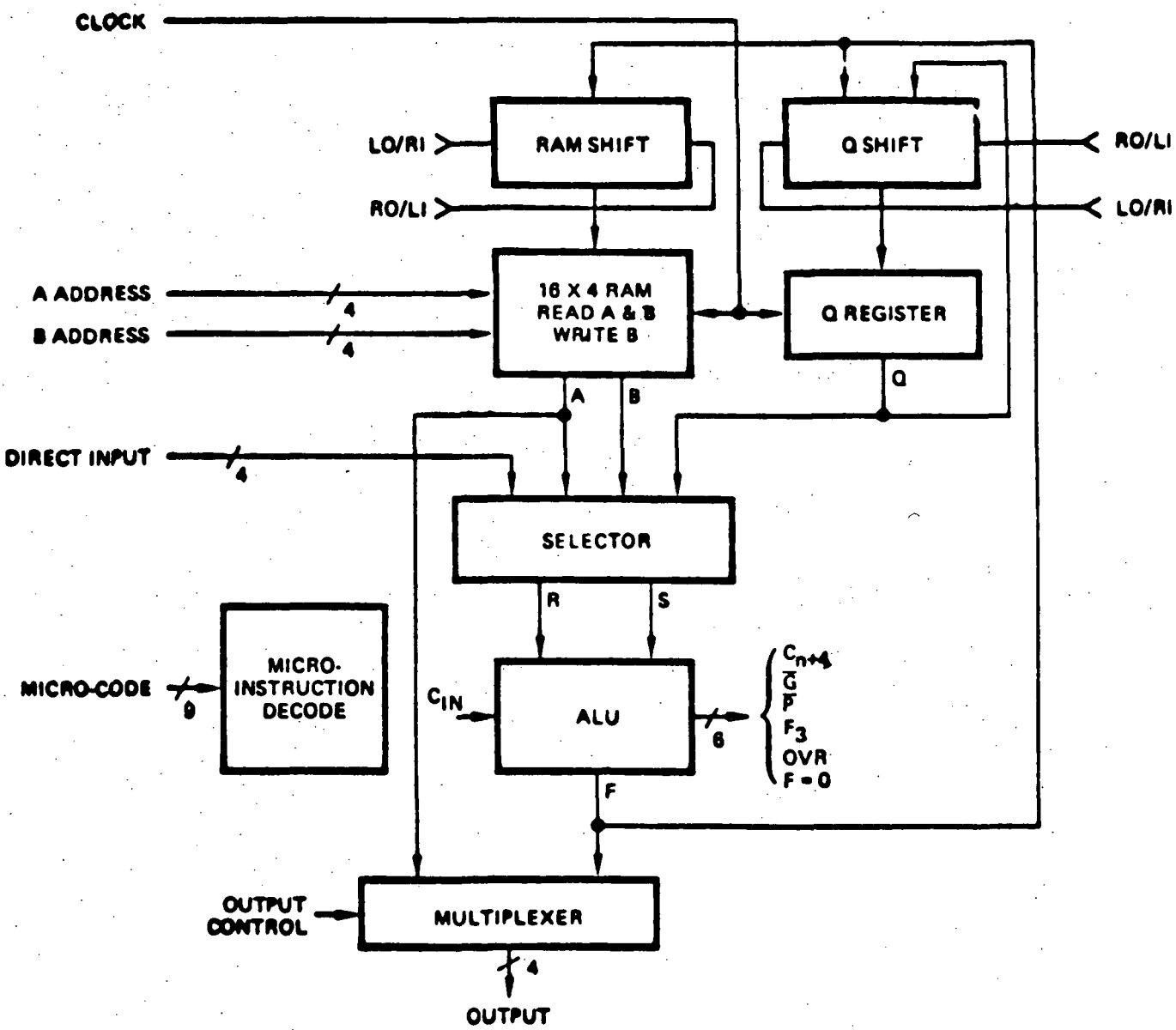
PIPELINE



ALU



MICROPROCESSOR SLICE BLOCK DIAGRAM



INSTRUCTION EXECUTION SPEED

TYPE	INSTRUCTION	EXECUTION (nSec)
0	REGISTER/REGISTER	250 or 325
1	INPUT/OUTPUT	350 OUTPUT 400 INPUT
2	REGISTER/IMMEDIATE	350
3	READ/WRITE PROGRAM MEMORY	525 READ 650 WRITE
4	EXTERNAL FUNCTION CONTROL	350
5	INTERRUPT CONTROL	400
6	PC STACK CONTROL	300
7	CONDITIONAL BRANCH	200 NO BRANCH 300 BRANCH

COMPARISON OF μPROCESSORS

	AMD AM2901	MMI MM6701	Intel 3002	TI SBP0400	Motorola M10800
Slice Width	4 bits	4-bits	2-bits	4-bits	4-bits
Cycle Time (Register to register; Read, Modify, Write)	100ns	200ns	150ns	1000ns	55ns
Power Dissipation (4 bits)	0.92W	1.12W	1.45W (2 x 0.73)	0.13W	1.3W
Addressable Registers	16	16	11	8	1 (External 4-256)
Register Addressing Mode	Two-Address	Two-Address	Single-Address	Single-Address	Single-Address
Number of Microcode Control Inputs	9	8	7	9	16
Primary Arithmetic Functions	R + S R - S S - R	R + S R - S S - R	R + S	R + S R - S S - R	R + S R - S S - R
Primary Logic Functions	5	3	3	8	6 - BCD 8 - Binary
Possible Source operand Combination to ALU	203	203*	24*	33*	6 - 262
Possible ALU Destination Registers	17	17	12	10	2 - 258
Flags	Carry Overflow Zero Negative	Carry Overflow Zero F=1111	Carry	Carry	Carry Overflow Zero

*Not all functions can be performed on all operand pairs.

DISTINCTIVE CHARACTERISTICS

- Two-address architecture –
  Independent simultaneous access to two working registers saves machine cycles.
- Eight-function ALU –
  Performs addition, two subtraction operations, and five logic functions on two source operands.
- Flexible data source selection –
  ALU data is selected from five source ports for a total of 203 source operand pairs for every ALU function.
- Left/right shift independent of ALU –
  Add and shift operations take only one cycle.
- Four status flags –
  Carry, overflow, zero, and negative.
- Expandable –
  Connect any number of Am2901's together for longer word lengths.
- Microprogrammable –
  Three groups of three bits each for source operand, ALU function, and destination control.