IBM DOCUMENT NO.
67-M31-0100A, 0101, 0102

IBM
PROPOSAL

MOL COMPUTER

PROGRAM DEVELOPMENT

Volume One - Technical Proposal

Submitted to

General Electric Company
Valley Forge, Pennsylvania

March 24, 1967

Federal Systems Division
INTERNATIONAL BUSINESS MACHINES CORPORATION
Gaithersburg, Maryland

TABLE OF CONTENTS

FOREWORD

This Technical Proposal for the MOL Computer Program Development activity is being submitted to GE in response to their Request for Proposal No. RB-004, dated 17 March 1967. The three major sections, contained herein, and identified below, describe in detail Attachments Nos. 1, 2, and 3 referenced in our Statement of Work (Section 1 - Cost Proposal):

- Section 1 - Computer Program Functional Requirements (Attachment No. 1)
- Section 2 - Input/Output Functional and DCSG Self-Test and Diagnostics Requirements (Attachment No. 2)
- Section 3 - Software Development Schedule (Attachment No. 3)

Qualifications and experience resumes of the key scientific and software personnel proposed to implement this program are presented in the Appendix to this proposal.

INTRODUCTION

IBM's MOL Programming Organization, because of the importance of software development in achieving optimum usage of the DCSG, will report directly to the MOL Program Director. By combining the experiences gained in commercial operating software systems, NASA and SSD ground support systems, and, in preparing operational space programs on Titan III, Saturn, and Gemini, this organization will continue to bring a full spectrum of capabilities and experience to bear on the MOL Program. In addition to providing those software elements, this group will be in a position to support GE as required on the preparation and integration of the application (operational) programs.

IBM's specific experience related to the MOL programming requirements is shown in Figure 1. This spectrum of programming experience for applied programming systems supporting space applications is reflected in the description column. In addition, programming experience with System 360 job processing applications is readily available as an IBM service.

Mr. B. P. Whipple is manager of the MOL Software Development Department, reporting directly to Mr. R. T. Ellsworth, Jr., Program Director. In this capacity, he assures that software development for all MOL efforts is consistent with respect to specification detail, planning detail, program approach, validation techniques and level of documentation.

This department is organized as project groups to assure proper emphasis on each of the required programming systems. Each project manager is fully responsible for the specification, implementation, validation and

AGENCY		SATELLITE OR PROBE	ASSEMBLY PROGRAMS	PROGRAMMING EFFORTS							COMPUTER SYSTEM	DESCRIPTION
				OPERATIONAL PROGRAMS	SIMULATION PROGRAMS	PROGRAM LIBRARY	TEST DATA GENERATORS	DISPLAY PROGRAMS	REAL TIME MONITOR	PROGRAM SYSTEM MAINTENANCE		
NASA	1. AMES RESEARCH CENTER	PIONEER			X							CALIBRATION PROCEDURE, OPTIMUM INTERVAL SELECTION, DATA ORGANIZATION SCHEME
	2. GODDARD SPACEFLIGHT CENTER	PROJECT MERCURY			X	X		X	X	X	IBM 7090/94	DEVELOPED LAUNCH MONITOR SUB-SYSTEM, ANALYSES, FLOWCHARTS, PROGRAMMING, TEST, IMPLEMENTA-TION, SIMULATED FLIGHT TEST FOR SYSTEM CHECKOUT
	A. SPACE SCIENCES DIVISION	'EXPLORER' SERIES. IMP. EGO. POGO			X	X	X	X			IBM 1410, IBM 7040/7094 DIRECT COU-PLE SYSTEM	DATA SMOOTHING, CURVE FITTING, FILTERING, MERGING EPHEMERIS DATA, SORT FILES, DECOMMUTATE INTO EXPERIMENT CHANNELS
	B. DATA SYSTEMS DIVISION	TIROS SERIES. S-6, SERB. RELAY			X	X	X	X			IBM 7040/ 7094 DIRECT COU-PLE SYSTEM	ADVANCED PROGRAMMING, STATIS-TICAL AND NUMERICAL ANALYSES, ATTITUDE DETERMINATION, DIF-FERENTIAL CORRECTION TECHNIQUES
	3. JPL SPACE FLIGHT OPERATIONS FACILITY	RANGER MARINER			X	X		X	X	X	IBM 7040/ 7094	SYSTEM DESIGN AND IMPLEMENTA-TION I/O EQUIPMENT, SPECIFICA-TIONS, DESIGN OF MONITOR PRO-GRAM, DIAGNOSTIC PROGRAM FOR SYSTEM TEST, REAL-TIME DIAG-NOSTICS DURING MISSION
	4. MANNED SPACECRAFT CENTER A. REAL TIME CONTROL CENTER	GEMINI AND APOLLO			X	X		X	X	X	IBM 7040/ 7094 TRIPLEX, DIRECT COU-PLE SYSTEM	DEVELOP AND IMPLEMENT FLEXIBLE SELF-ORGANIZING REAL-TIME COM-PUTER COMPLEX, ASYNCHRONOUS DATA INPUTS, SEPARATE PROCES-SOR MODULES
	B. (SUBCONTRACT TO MCDONNELL AIRCRAFT)	GEMINI	X	X	X						GEMINI	DESIGN DEVELOP AND RADIATE FLIGHT AMPLIFIER OPERATIONAL PROGRAMS PRELAUNCH, RESEARCH, RENDEZVOUS, TELEMETRY, RE-ENTRY, AND COMPUTER RELATED POST-FLIGHT
	5. MARSHALL SPACE FLIGHT CENTER	SATURN	X	X	X						ASC-15 SATURN V	DESIGN, DEVELOP AND VALUDATE FLIGHT QUALIFIED OPERATIONAL PROGRAMS, PRE-LAUNCH, SELF CHECK, GUIDANCE, AND POST-FLIGHT
USAF	1. BALLISTICS SYSTEMS DIVISION (SUBCONTRACT TO AC ELECTRONICS)	TITAN II	X	X	X						ASC-15	DEVELOP AND VALIDATE FLIGHT QUALIFIED OPERATIONAL PROGRAMS. PRE-LAUNCH, SELF-CHECK, GUIDANCE AND POST-FLIGHT
	2. SPACE SYSTEMS DIVISION (SUBCONTRACT TO AC ELECTRONICS)	TITAN III										
	3. PROGRAM 461				X	X	X	X	X	X	IBM 7094	DESIGN AND DEVELOP APPLICATION PROGRAMS, DISPLAYS, REAL-TIME 'LIBRARY' AND MONITOR FORMAT ANALYZE AND CORRELATE DATA
	4. PROJECT ATHENA	ABRES RE-ENTRY BODIES			X	X		X		X	IBM 7040/ 7094 DIRECT COUPLE SYSTEM	DEVELOP REAL-TIME EXECUTIVE MONITOR PROGRAM, AUTOMATIC PRE-LAUNCH COUNTDOWN, IMPACT PREDICTION

Figure 1. Related Software Experience

documentation of all programs associated with his respective programming system in accordance with specified software implementation plans and schedules.

Successful implementation of the effort depends upon the application of proven programming skills and methods throughout the development and validation effort.

In accordance with our Statement of Work (Section 1 - Cost Proposal), we have organized this Technical Proposal around Attachments Nos. 1, 2, and 3--as indicated by the following Section breakdown:

o	Section 1	Computer Program Functional Requirements (Attachment No. 1)
	Part 1	ADC Executive Control System
	Part 2	Mathematical Utility Programs
	Part 3	Program Preparation Processor
	Part 4	Simulation Supervisor
	Part 5	System/360 M44 ADC Support
o	Section 2	Input/Output Functional and DCSG Self-Test and Diagnostic Requirements (Attachment No. 2)
	Part 1	I/O and DCSG Self-Test and Diagnostic Programs
o	Section 3	Software Development Schedules (Attachment No. 3)

Resumes of key scientific and programming personnel are presented as an appendix to this proposal.

Section 1

COMPUTER PROGRAM FUNCTIONAL REQUIREMENTS (ATTACHMENT NO. 1)

PART 1

ADC EXECUTIVE CONTROL SYSTEM

1.1 INTRODUCTION

The ADC Executive Control System (ECS) shall supervise operational control over the execution of programs required for support of the MOL mission.

The basic performance and design objectives of the ECS are:

- Provision for a multi-programmed environment which shall permit programs to be constructed and executed independently.
- Centralization of functions which control the flow of information between programs, the external subsystems and crew.
- Management of system resources to obtain efficient use of the hardware and to insure the response required by the application programs.
- Continuous monitoring and control of program and hardware operation.

The basic ECS package shall be designed to supervise the execution of programs in unscheduled sequence. In this environment an application program shall be initiated by the occurrence of an asynchronous event, and receive control directly from an interrupt or during the course of interrupt processing. The system structure for this basic package shall consist of five components and associated tables.

- Storage Management shall control core utilization and access to the Auxiliary Memory Units.
- Interrupt Supervision shall process all interrupts and synchronize the ECS with external subsystems.

- Input/Output Supervision shall provide access to the services associated with the use of I/O devices.
- System Services shall provide a number of services used by the ECS and application programs.
- Environmental Interface routines shall respond to and modify the conditions under which the system is to operate.

Optional packages shall be designed to expand the capability of the basic ECS package.

The first option shall provide for scheduled program operation, in which a program shall receive control under specified conditions as the result of a decision by a scheduling algorithm. Scheduled programs implement applications for which CPU time may be shared. The scheduling algorithm shall have the capability to multiplex program execution. Programs operating in the multiplexed environment shall be structured as program units. The system routines implementing this option are called Scheduled Program Control. They shall govern the initiation, termination and execution of multiplexed programs.

The second option shall provide capability for queue control. Queue control shall permit location — independent passage of data and control information between program units.

1.2 STORAGE MANAGEMENT

The Storage Manager shall control allocation of all core not permanently assigned at system generation time and control access to the Auxiliary Memory Unit (AMU).

1.2.1 BCSM

The Basic Core Storage Manager (BCSM) shall maintain a pool of unassigned core and upon request assign or release blocks of the pool. Managed core blocks can be variable in size and shall be members of one of three storage groups.

- Program Storage
- Data Storage
- Free Storage

1.2.2 Storage Macro Capability

Allocation — There shall be two macros, one which generates a request for data storage and one which generates a request for program storage. The BCSM shall respond by assigning the core and supplying the location to the requestor. If a block of the size requested is not available, the BCSM shall inform the requestor of this condition.

Retrieval — Execution of this macro shall generate a request to release a specified core block. The BCSM shall respond by returning the block to the free storage group.

1.2.3 AMU Routine

The AMU Routine shall provide access to programs and data resident on the AMU. There shall be two basic functions implemented:

- Position and read
- Directory research

The Position and Read Routine shall forward space or backspace the AMU to a specified record number and read that record into a specified core location. The Position and Read Routine shall be the sole means of accessing the AMU.

The Directory Search Routine shall accept requests for locating items on the AMU tape by symbolic name. It shall return to the requestor information which specifies the tape position and the number of words in the item requested.

1.3 INTERRUPT SUPERVISION

The ECS shall provide routines to process:

- Machine Check interrupts
- Supervisor Call (SVC) interrupts
- Program interrupts
- External interrupts
- Input/Output interrupts

System functions available to these routines shall also be available to user-provided priority interrupt routines.

The interrupt supervisor routines shall be designed to:

- Minimize the time required to save and restore machine conditions
- Run with interrupts enabled as much as possible
- Permit stacking of interrupts
- Permit the use of common system code at multiple interrupt levels by means of a gate mechanism.

In order to accomplish these objectives all interrupt routines shall follow a set of specified system conventions.

1.3.1 Interrupt Save Mechanisms

Machine conditions shall be saved for all interrupts except SVC interrupts in save areas to be provided by the system. Save areas required for processing SVC interrupts shall be provided by the user requesting the service. The code to perform save and restore machine conditions shall be provided in macros expanded into in-line code in the individual interrupt routines. The code performing save and restore shall sequence the system — provided save areas to allow interrupted routines to regain control in priority order.

1.2.3 Interrupt Processing Routines

The following interrupt routines shall be provided:

- The Machine Check Interrupt routine shall transmit a message identifying the conditions to the operators, and shall attempt recovery if specified.
- The SVC Interrupt routine shall perform initial processing of these interrupts to include saving of the Program Status Word (PSW) in the user-supplied area and transfer of control to the requested subroutine by means of the SVC Transfer Table.
- The Program Interrupt routine shall analyze the cause of the interrupt. Interrupts which indicate a possible system malfunction shall cause a message to be transmitted to the operator. Interrupts which are program specific shall be ignored.
- The External Interrupt routine shall have the ability to process multiple sources causing a given external interrupt. It shall transfer control to the processing routines indicated by the PSW interrupt codes in a pre-specified order.
- The Input/Output Interrupt routine shall analyze the Channel status Word (CSW) to determine the cause of the interrupt. If the cause is attention or error, control shall be passed to a device routine to take the appropriate action. If the cause is

I/O completion, the result shall be posed and the next operation shall be initiated on the channel.

1.3.3 Interrupt Control Gates

Interrupt control gates shall be provided to permit serially-reusable code to be accessed by more than one interrupt level.

1.4 INPUT/OUTPUT SUPERVISOR

The input/output service shall consist of a set of device independent supervisory routines and a set of device routines. The supervisory routines shall accept I/O requests and perform the requested operations. The device routines shall prepare for I/O operations and take appropriate action for the device.

1.4.1 Input/Output Macros

Requests for I/O operations shall be communicated to the system I/O routines by use of I/O macros. The macros provided shall be:

READ
WRITE
EXIO

The operands of the macros shall specify a device and the location of an I/O Request Block (IORB). The IORB shall contain space for Channel Command Word (CCW) specification to be used with the request. For the EXIO macro, the CCW specification shall be assumed to have been previously established. For the READ and WRITE macros, additional operands shall specify the number of bytes to be transmitted and the location of a user provided I/O area.

1.4.2 Input/Output Request Processor

The I/O Request Processor shall accept and interpret I/O macros. For READ and WRITE, a device routine shall be called to create appropriate CCW's. For EXIO, and after returning from the device routines for READ and WRITE, the IORB shall be queued by device. The operation shall be initiated whenever the IORB reaches the top of its queue. The IORB shall be extracted from its queue by the I/O Interrupt Routine when the operation is completed.

1.4.3 Device Routine

There shall be a device routine for each of the following:

- Command Uplink
- Command Output
- AMU
- Hz Timing
- Channel to Channel
- Internal Control Register
- CSC Data Error

These device routines shall provide the interface between the devices and the Input/Output Supervisor. A device routine shall:

- Create CCWs required for the READ or WRITE operation
- Respond to attention interrupts
- Initiate device-peculiar error recovery procedures
- Post device-peculiar completion information

1.4.4 Logical Posting

Routines shall be provided to post I/O completion information for synchronization with the I/O operation. The following functions shall be implemented:

- Coordination of the internal real time clock with the external timing subsystem.
- AMU directory search

1.5 <u>SYSTEM SERVICES</u>

The following system services shall be provided:

- Timing
- Message Handling
- Power Conservation
- Universal Common
- Table Search I/O Macro

1.5.1 <u>Timing</u>

1.5.1.1 <u>Malfunction Timer.</u> The eight-second malfunction timer shall be reset at regular intervals by ECS.

1.5.1.2 <u>Absolute Time.</u> A clock shall be kept internally by the system in the form of a 32-bit integer representing time in milliseconds. The clock shall record time in a continuous manner. Time, in milliseconds, shall be available to programs by means of a service macro and shall be synchronized with a clock external to the ADC.

1.5.1.3 <u>Interval Timer.</u> An interval timer queue shall be available to accept requests for execution of routines at a specified time of day. Provision shall be made to delete a request.

In the basic ECS the request for an interrupt shall be accompanied by the location of the routine that will perform the desired functions.

In any of the extended ECS systems there shall be provision for timing services to program units which shall implement the timed initiation and elimination of program units.

1.5.2 Message Handling

A system message handling routine shall centralize the transmission of all data between the ADC and the crew. The Message Handler shall process all output messages (via Printer, Downlink, and Display), and shall construct the parameters necessary for the I/O routine.

1.5.3 Power Conservation

ECS shall have access to the CPU time requirements of the currently executing programs. Whenever the current program environment permits, ECS shall turn off power by means of a power conservation request:

- to the logic of the CPU
- to the MDAU and the DCSG except for the CSC

1.5.4 Universal Common

ECS shall provide for Universal Common, an area of contiguous core, permanently resident in nonprotected memory, fixed in format, and accessible to both application and system routines. It may be used as a data base area for information declared at assembly time, for data provided by the system to the application programs, and for the dynamic data received and created during real time operation which is common to more than one program. Data in Universal Common shall be accessed directly or in conjunction with the gating mechanisms.

1.5.5 Table Search I/O Macro

ECS shall provide a means of extracting items of data from records on the

AMU. A list containing the identification of the items to be extracted from the data records and the locations in memory at which the information is to be placed after extraction, shall be supplied as an input parameter. This service routine shall use the AMU routines to transfer the specified data.

1.6 THE ENVIRONMENTAL INTERFACE

ECS shall provide environmental interface routines for initiation and termination of the system, for control of the system configuration, and for the recovery from specified error conditions.

1.6.1 System Initiation

The Basic Initialization Routine (BIR) shall be responsible for initial table construction and for the loading of all programs and data not loaded at IPL. The information for loading these programs shall be obtained from the AMU Directory Search routine. The AMU directory shall be segmented and only one segment shall be required in core at a given time.

The BIR shall be non-resident and shall be loaded as the result of the IPL. It shall perform a standard initialization or an alternate initialization.

- Standard Initialization A table internal to the BIR shall contain the symbolic names of the first non-resident programs to be loaded and operated. After the BIR has performed its table construction, these programs shall be loaded and prepared for operation.

- Alternate Initialization The BIR shall receive information from an input device which specifies the initial set of programs to load and shall ignore the table used by the Standard Initialization.

1.6.2 System Termination

The Basic Termination Routine (BTR) shall assure an orderly shutdown by:

- completing all active I/O requests
- giving control to a user-defined program to complete processing
- powering down upon return from the user-defined program

1.6.3 Error Recovery

The system shall provide a set of routines to be used for the purpose of error recovery. These routines shall be accessible to any program which detects a specified error condition. The following functions shall be provided.

- IORB deactivation
- Configuration change

1.6.4 Configuration Control

There shall be a machine configuration list maintained by the system which describes the current hardware configuration. This table shall contain the following:

- CPU error indicator
- Number and status of the channels attached to the machine
- Unit Control Block locations for devices implementing specified functions
- Data adapter status information
- Physical core availability
- Current setting of configuration control register

Routines shall be supplied to update the information in this table.

1.7 TABLES

The following tables and control blocks shall be employed for intra-system information transfer and for communication between programs and ECS:

- System Communication Region This table shall provide residence for universally required system information such as address constants for locating the major data structure.
- SVC Transfer Table This table shall locate programs and system subroutines accessed by the SVC.
- System Transfer Table This table shall contain address constants for locating independently assembled portions of the systems.
- Unit Control Block (UCB) There shall be a UCB for each I/O device in the system. It shall contain information regarding the physical characteristics and status of a device and other information required by the system to service the device.
- Message This table shall contain a set of messages that serve as a basis of communication between the ADC and the operator.

1.8 PROGRAM UNITS AND SCHEDULED PROGRAM OPERATION (OPTION 1)

Option 1 shall be obtained by the addition of a set of routines and tables to the basic ECS. The additional capability provided by Option 1 shall permit the operation of programs in a scheduled multiplexed manner.

1.8.1 Program Unit Definition and Structure

Scheduled programs shall be constructed from structural entities called program units.

A program unit shall consist of:

- A Program Control Block (PCB) shall be the interface between the program unit and the system.
- Text shall be relocatable code.
- Common shall be a data area private to the program unit (optional).
- A Relocation Dictionary shall provide for address modification upon relocation.

In addition, program units shall be written in segments. Each segment shall form a logical division in the program unit which provides an entry point for execution and which returns control to ECS prior to the expiration of a specified time interval. Interval timer interrupts shall be used as a guard against excessive execution time in a given segment. Operational specifications for the program unit shall be supplied during program preparation and incorporated into the PCB.

1.8.2 Scheduled Program Control

The operation of program units in a scheduled, multiplexed manner shall be implemented by a set of system routines which shall execute:

- Program Unit Initiation
- Program Unit Scheduling
- Program Unit Termination
- Intersegment Supervision

1.8.2.1 Resident Program List (RPL). Intercommunication between the Scheduled Program Control routines shall be provided by a data structure called the Resident Program List (RPL). This data structure shall provide an index to, and the means for controlling the activity of, all program units in core or in the process of being loaded into core. Program unit entries shall be grouped into five queues which correspond to the five possible states of program unit activity.

- Initiation requested and awaiting loading
- Resident in core and waiting to be assigned CPU time
- Currently receiving CPU time as the PU's priority and available CPU time permits
- Execution terminated and storage work areas awaiting release
- Resident in core but currently in an inactive state.

1.8.2.2 Program Unit Initiation. Requests to initiate the execution of program units shall be made by means of macros. The following processing shall be performed:

- Initial processing shall determine the current status of the requested program unit and select the appropriate routine to continue the initiation process.
- During the next intersegment time, a routine shall initiate the reading from the AMU of the requested program unit's PCB, if it is not in core.
- Routines which provide service to load the program unit and adjust the location dependent quantities in the text shall operate in multiplex mode under control of the PCB of the program unit being serviced.

1.8.2.3 Scheduler. Program units shall receive CPU time according to their operational requirements. These requirements shall be expressed in terms of a repetition rate which specifies the desired number of segment executions per second. The scheduler shall achieve priority multiplexing by examining the repetition rates and priorities of all active program units in order to interleave their execution.

Three classes of scheduling service shall be provided:

- <u>Cyclic</u>
  CPU time shall be assigned at equally spaced intervals according to the repetition rate specified for the program unit.

- <u>Priority</u>
  CPU time shall be assigned in proportion to the repetition rate to meet overall processing requirements.

- <u>Non-Priority</u>
  CPU time shall be assigned when available.

The Activate Routine shall place in the priority multiplexing loop as many of the program units requiring CPU time as unused time in the loop permits. Program units shall be placed in the loop in priority order, starting with cyclic program units and descending through the various priority levels to the lowest level. Selection Routines, one each for cyclic, priority and non-priority service, shall select the next program unit at that level of service to receive CPU time. An Entry Routine shall complete preparations for transfer of control to the selected program.

1.8.2.4 <u>Program Unit Termination Function.</u> A macro capability shall be provided which will permit an executing program unit to deactivate itself or to deactivate another program unit. Such macros shall be initially processed by a routine which shall perform the necessary adjustment of the program unit's entry in the RPL. This routine shall also request the initiation of a second routine, to be operated as an independent multiplexed program unit, which shall complete the deactivation process by releasing any data storage assigned to the program unit.

1.8.2.5 <u>Intersegment Supervision.</u> Routines executed during intersegment time shall be controlled by the Intersegment Supervisor. In addition to the Program Unit Initiation and Termination routines and the Scheduler, the Intersegment Supervisor shall drive a subroutine to collect and save information describing the status of Program Units.

1.8.2.6 Program Unit Gates. Program Unit Gates shall be provided to control access to serially reusable code shared by program units. The Program Unit Gates Mechanism shall place a requesting program unit in wait status if the gated code is in use. Program Units in wait status shall be bypassed by the Schedule until the gate is opened. Unstacking of program unit requests to enter the gated code shall be done in FIFO order.

1.9 QUEUE CONTROL (OPTION 2)

Option 2 shall be implemented by the addition of a queue control capability to the Option 1 package. Provision shall be made for input queues to be associated with program units. Macros shall be provided to process these queues which shall implement the following capabilities:

- Creating a permanent or temorary queue
- Adding an item either to the top or bottom of a queue
- Disconnecting either the top or bottom item from a queue
- Disconnecting an item from a user specified place in a queue
- Locating an item either forward or backward from a given starting point within a queue without disconnecting the item in the process
- Transferring an item or group of items from one queue to another
- Releasing an item or group of items from a queue directly to free storage
- Deleting a temporary queue

Queue control shall be maintained by means of queue headers which shall be contained in a system data structure called the Queue Header List.

PART 2

MATHEMATICAL UTILITY PROGRAMS

2.1 INTRODUCTION

The following mathematical functions shall be implemented:

- * Sine/Cosine Routine
- Arc Sine/Arc Cosine Routine
- Tangent Routine
- * Arc Tangent Routine
- * Square Root Routine
- Exponential Routine
- Logarithm Routine
- Matrix Operation Routine

* Assumed to be in ROS memory.

The basic design requirement shall be the use of fixed-point arithmetic. A secondary requirement shall be that computation speed be as rapid as possible. An optimal time-space combination for each routine shall be sought.

Unless specified otherwise, the error allowance for these subroutines sh shall be:

If f(x) is one of the functions and s(x) is the function generated by its subroutine, then

$$| f(x) - s(x) | < 2^{-25}$$

for all x in the domain (input range) of s(x).

Each subroutine shall assume the argument to be in a fixed General Register and deliver the function value to another fixed General Register. When other General Registers are required during execution, their contents before entry to the subroutine shall be saved and restored upon exit.

In the following descriptions, input/output formats will be specified as SA.B, where A and B are integers giving the number of bits the left and right of the binary point respectively; A + B = 31. S denotes sign.

2.2 SINE/COSINE ROUTINE

$Y = \text{Sine } X,\ Y = \text{Cosine } X$

Domain: $-\pi \leq X \leq \pi$, Format $\pm$ 3.28

Range: $-1 \leq Y \leq 1$, Format $\pm$ 1.30

Error return if $|X| > \pi$.

2.3 ARC SINE/ARC COSINE ROUTINE

$Y = \text{Arc Sine } X,\ Y = \text{Arc Cosine } X$

Domain: $-1 \leq X \leq 1$, Format $\pm$ 1.30

Range: $-\frac{\pi}{2} \leq Y \leq \frac{\pi}{2}$ (Arc Sine) Format $\pm$ 3.28

$0 \leq Y \leq \pi$ (Arc Cosine) Format + 3.28

Error return if $|X| > 1$.

2.4 TANGENT ROUTINE

$Y = \text{Tangent } X$

Domain: $-\frac{\pi}{2} < X < \frac{\pi}{2}$, Format $\pm$ 3.28

Range: $-2^8 < Y < 2^8$ Format $\pm$ 8.23

Error return if $|X| \geq \frac{\pi}{2}$

For this subroutine the error bound must be increased to 2^{-24}.

2.5 ARC TANGENT ROUTINE

$$Y = \text{Arc Tangent } X.$$

Domain:	$-2^8 < X < 2^8$,	Format $\pm$ 8.23
Range:	$-\frac{\pi}{2} < Y < \frac{\pi}{2}$,	Format $\pm$ 3.28

For this function, it shall be the programmer's responsibility to determine if $|X| \geq 2^8$, in which case $Y = \pm\frac{\pi}{2}$.

2.6 SQUARE ROOT ROUTINE

$$Y = \sqrt{X}$$

Domain:	$0 \leq X < 1$,	Format + 0.31
Range	$0 \leq Y < 1$,	Format + 0.31

Error return if $X < 0$.

The least significant 7 bits of X shall be set to zero before computation. Thus, if $X < 2^{-24}$, it is assumed to be zero.

2.7 EXPONENTIAL ROUTINE

$$Y = 2^X$$

Domain:	$-1 \leq X \leq 1$,	Format $\pm$ 1.30
Range:	$\frac{1}{2} \leq Y \leq 2$,	Format + 2.29

Error return if $|X| > 1$.

2.8 LOGARITHM ROUTINE

$$Y = \text{Log}_2 X$$

Domain:	$\frac{1}{2} \leq X \leq 2$,	Format + 2.29
Range:	$-1 \leq Y \leq 1$,	Format $\pm$ 1.30

Error return if $X < \frac{1}{2}$ or $X > 2$.

2.9 MATRIX OPERATION ROUTINE

The remarks concerning accuracy from Sec. 2.1 do not apply.

A routine for obtaining the product AB of matrices A and B shall be written. The dimensions of A and B shall be permitted to be as high as 4 X 4.

Design of the subroutine shall be directed to low execution time when the dimensions of A and B are 3 X 3 and 3 X 1, respectively.

Access to the matrix multiply subroutine shall be via a calling sequence specifying locations and dimensions of A and B and permitting the user to specify he desires B to be multiplied by the transpose of A.

2.10 APPENDIX. DEVELOPMENT OF ERROR BOUNDS

The algorithms selected to implement the basic design requirements may be affected by the error bounds.

Only a detailed error analysis of the computation done in an application can determine allowable error bounds for subroutines which are components of the computation. With the exception of the Tangent and Arctangent functions (Paragraphs 2.4 and 2.5), the error bounds given above were adopted from preliminary analysis of the applications. In the case of Tangent and Arctangent, the bounds were dictated by a desire to stay away from a multiple word or floating point representation of the numbers involved.

PART 3

PROGRAM PREPARATION PROCESSOR

3.1 INTRODUCTION

The Program Preparation Processor (PPP) shall convert program object modules into a form suitable for loading and executing in the ADC. All programs executed in the ADC shall first be processed by PPP.

This processor shall accept, as input, both the output of the System/360 Model 44 Programming System (44PS) processors, and statements by the programmer which specify program construction.

The output from PPP shall be in the form of program units. These program units shall consist of one or more object modules including those from the System Reference Library that shall require combination with new object modules. An object module may be independent or self-contained. There may be numerous cross references between object modules and between control sections within object modules.

PPP shall be constructed in four phases:

- Initialization Phase
- Card Processing Phase
- Post Processing Phase
- Library Maintenance Phase

(See 4.1 for definition of PSCS/LPSS configurations).

3.2 INITIALIZATION PHASE

This phase shall determine the location of tables and buffers, check options saved by the Job Control Processor from the // EXEC PPP job control statement, set switches, save information for subsequent phases, perform module card processing, and initiate processing of input data. It shall contain constants and subroutines common to all phases, and two computer program routines:

- Initialize for Execution Routine
- MODULE Processor Routine

3.2.1 Initialize for Execution Routine

This routine shall make necessary preparations for subsequent operation of PPP.

This shall be the first routine operated for each program unit to be constructed. It shall initialize switches and buffers, define core maps, and establish pointers to various buffers.

When all input has been processed or when the Temporary Program Unit Library is full, this routine shall call in and operate the Library Maintenance Phase.

3.2.2 MODULE Processor Routine

This routine shall check the System Input data set for object modules. If any are found, this routine shall transfer them to the Language Processor Output data set and reflect this change in the Language Processor Output Directory data set.

At the end of input on the System Input data set, this routine shall return to the Initialize for Execution routine with an indication of such. Otherwise, this routine shall call in and operate the Card Processing Phase.

3.3 CARD PROCESSING PHASE

This phase shall be a group of computer program routines whose function shall be the production of an intermediate program unit. This processing shall consist of:

- Reading the control statements which identify the output program unit and specify its construction in terms of format and content.

These control statements shall be:

1. Program Unit (PROGUNIT)
2. Symbol Dictionary (SYMBDICT)
3. Include (INCLUDE)
4. Entry (ENTRY)

- Reading the object module cards that are to be used to construct the program unit. These object module cards shall be:

  1. External Symbol Dictionary (ESD)
  2. Text (TXT)
  3. Replacement (REP)
  4. Relocation List Dictionary (RLD)
  5. End (END)

- Resolving the cross references within the program unit

- Constructing the Intermediate Relocation Dictionary

This phase shall contain the following computer program routines:

- Start Program Unit Routine
- Read Next Card Routine
- Control Card Scan Routine
- PHASE Processor Routine
- ENTRY Processor Routine
- SYMBDICT Processor Routine
- INCLUDE Processor Routine
- AUTOLINK Processor Routine
- ESD Processor Routine
- TXT Processor Routine
- REP Processor Routine
- RLD Processor Routine
- END Processor Routine

3.3.1 Start Program Unit Routine

This routine shall start construction of the program unit by indicating gross structure, identifying by name, and creating a skeleton Program Control Block (PCB) (if there is a PCB).

This shall be accomplished by analyzing the information presented on a PROGUNIT control statement, creating a PHASE control statement, and operating the PHASE Processor Routine.

3.3.2 Read Next Card Routine

This routine shall read a card from one of three data sets; System Input, Language Processor Output, System Reference Library, and operate a routine to process the card read according to type, e.g., the Control Card Scan Routine shall be operated when a control statement is read.

3.3.3 Control Card Scan Routine

This routine shall be operated when the Read Next Card Routine has read a control statement from the System Input data set. It shall determine the type of control statement and perform a sequence check. It shall transfer to the appropriate routine to process it, e.g., the ENTRY Processor Routine shall be operated when an ENTRY control statement is read.

3.3.4 PHASE Processor Routine

This routine shall analyze the PHASE control statement produced by the Start Program Unit Routine. This analysis shall set up the program unit origin in the appropriate save areas, set switches indicating whether or not "Autolinking" (See Paragraph 3.3.8 of this Specification) shall be provided, and construct the first two entries in the Control Dictionary that define the program unit.

3.3.5 ENTRY Processor Routine

This routine shall determine the Initial Segment Entry Point of the program unit from the ENTRY control statement and finalize the processing of the Card Processing Phase.

3.3.6 SYMBDICT Processor

This routine shall construct a Symbol Table indicating the symbols required in the Symbol Dictionary from an analysis of the SYMBDICT control statement(s).

3.3.7 INCLUDE Processor Routine

This routine shall locate object modules to be included in the program unit. These object modules shall be defined by INCLUDE control statements or implied via "Autolinking".

3.3.8 AUTOLINK Processor Routine

This routine shall determine which object modules are to be automatically incorporated into the program unit by assuming unresolved external references are names of object modules in the System Reference Library.

3.3.9 ESD Processor Routine

This routine shall create an entry in the Control Dictionary for each entry in the input ESD object module cards.

3.3.10 TXT Processor Routine

This routine shall extract the text from the TXT object module cards and insert it into the Text element of the program unit.

3.3.11 REP Processor Routine

This routine shall convert REP object module cards into TXT object module cards.

3.3.12 RLD Processor Routine

This routine shall build the Intermediate Relocation List Dictionary data set from the input RLD object module cards.

3.3.13 END Processor Routine

This routine shall save the transfer information on an END object module card and complete the processing of the current input object module by:

- Completing cross references within the Control Dictionary
- Obtaining the length of the last control section
- Preparing PPP to process the next input object module

3.4 POST PROCESSING PHASE

This phase shall complete the construction of the program unit, produce the Map as specified on the // EXEC PPP job control statement, and place the program unit in the Temporary Program Unit Library. This phase shall be composed of the following computer program routines:

- MAP Processor Routine
- RLD Post Processor Routine
- Save Program Unit Routine

3.4.1 MAP Processor Routine

This routine shall perform three functions:

- Print COMMON Entries

- Compute the length of Private Common
- Print a Storage Assignment Map, unless NOMAP option is specified on the / / EXEC PPP job control statement

3.4.2 RLD Post Processor Routine

This routine shall complete the construction of the program unit by:

- Allocating storage within the program unit for Private Common
- Constructing the Symbol Dictionary from the Symbol Table and the Control Dictionary
- Assigning addresses to all address constants and simultaneously creating the Relocation Dictionary portion of the program unit from the Intermediate Relocation List Dictionary data set and the Control Dictionary.

3.4.3 Save Program Unit Routine

This routine shall place the constructed program unit into Temporary Program Unit Library for storage until PPP is ready to update the Program Unit Library.

3.5 LIBRARY MAINTENANCE PHASE

This phase shall be a single computer program routine whose function shall be to combine the current Program Unit Library with the Temporary Program Unit Library creating an updated Program Unit Library reflecting the newly prepared program units.

PART 4

SIMULATION SUPERVISOR

4.1 INTRODUCTION

The Simulation Supervisor (SS) shall provide the capability to test, debug, and integrate ADC programs within the PSCS/LPSS. The Programming Support Computer System (PSCS) shall have a minimum configuration of:

- A System/360 Model 44G
- Four 2400 Series Tape Drives, two of which must have 9-track read/write heads. Any 7-track tape drives require the data conversion feature.
- A 1403 or 1443 Printer
- A 2540 or 1442 Card Read Punch
- A Single Disk Storage Drive
- A Floating Point Arithmetic feature if FORTRAN is used
- Appropriate control units and a multiplex channel
- Appropriate computer supplies

The Laboratory Programming Support System (LPSS) shall have a minimum configuration of:

- A System/360 — Model G or higher
- Four 2400 Series Tape Drives, two of which must have 9-track read/write heads. Any 7-track tape drives require the data conversion feature
- A 1403 or 1443 Printer
- A 2540 or 1442 Card Read Punch
- A Floating Point Arithmetic feature if FORTRAN is used
- A 2311 Disk Storage Drive
- Appropriate control units
- A multiplex channel and/or a selector channel
- Appropriate supplies

Facilities shall be incorporated within SS to accommodate:

- The interface between System/360 Model 44 Programming System (44PS), SS, and ADC programs
- Loading and monitoring ADC programs during execution
- Simulation of I/O devices associated with ADC programs
- Debugging information, and program flow traces

SS shall consist of two phases:

- Initialization Phase
- Interrupt Handling and Processing Phase

4.2 INITIALIZATION PHASE

44PS shall be modified to recognize an Execute Control Card and load the Initialization routines into main storage. The Initialization Phase shall contain the following routines:

- Control Routine
- Get Control Card Routine
- Card Type Determination Routine
- Build Table Routines

4.2.1 Control Routine

This routine shall control the processing flow of the Initialization Phase. It shall establish queue and table areas, transfer control to the proper processing routines, construct the communication region containing the interface between an Interrupt Handling Processing Phase and the constructed queues and tables, load an Executive Control System (ECS) and relinquish control to the Interrupt Handling and Processing Phase.

4.2.2 Get Control Card Routine

This routine shall read the SS control cards and store them in a buffer area for inspection by the Card Type Determination Routine.

4.2.3 Card Type Determination Routine

This routine shall inspect the control card read by the Get Control Card Routine.

This Routine shall:

- Determine control card type
- Edit and extract parameters
- Transfer control to the processing routines to construct the queues and tables

Control card types that shall be recognized are:

- AMU Directory Control Card
- I/O Equivalence and Data Definition Control Card
- Debugging Aids Control Card
- Replace Text Control Card
- Time Dependent Events Control Card
- Interval Timer Conversion Control Card
- ECS Queue Dump Control Card

4.2.3.1 AMU Directory Control Card. This card shall contain the names of the program units involved in the simulation. The following tables shall be constructed:

- AMU Program Directory
- Program Unit Library Directory
- AMU Directory Queue List

4.2.3.2 I/O Equivalence and Data Definition Control Card. This control card shall specify the I/O device equivalence between the simulated devices and those of the PSCS/LPSS.

4.2.3.3 Debugging Aids Control Card. This control card shall contain the parameters required by the Debugging Aids Routine (4.3.5).

4.2.3.4 Replace Text Control Card. This control card shall contain information required to alter a portion of code within a program without reassembling.

4.2.3.5 Time Dependent Event (TDE) Control Card. This control card shall contain the parameters required to build a TDE queue and table. The information contained in the table shall allow SS to initiate an I/O attention interrupt at the time specified on the control card.

4.2.3.6 Interval Timer Conversion Control Card. This control card shall contain the parameters required to compute a conversion factor to be applied to all interval timer actions. This conversion factor shall be used to correct resolution differences between the simulated timer and PSCS/LPSS time.

4.2.3.7 ECS Queue Dump Control Card. This control card shall contain information required to dump the ECS queues after completion of a specific program unit.

4.2.4 Build Table Routines

These routines shall process and construct queues and tables for each type of control card. After completion of this function, control shall be returned to the Control Routine (Paragraph 4.2.1).

4.3 INTERRUPT HANDLING AND PROCESSING PHASE

This phase shall contain the routines to interface and insure execution of ADC programs. This phase shall use the interrupt capabilities of the PSCS/LPSS.

All SS routines shall be executed in the supervisor state while the ECS and problem programs shall be executed in the problem state. SS shall gain control during the simulation by any of the following interrupts:

- Program Interrupt
- External Interrupt
- Machine Check Interrupt
- Supervisor Call (SVC) Interrupt

Before processing an interrupt, SS shall:

- Save the status of the machine
- Save the remaining time in the interval timer
- Convert the interval timer value to ADC time units
- Update the ADC programs running time

After processing the interrupt SS shall:

- Reset the saved machine status
- Reset the interval timer

4.3.1 Program Interrupt Routine

Program interruption shall occur due to error conditions and attempted execution of privileged operations in the problem state. The error conditions shall cause debugging information to be ouput by SS. Control shall then be transferred to the ECS Program Interrupt Handler with the error condition.

The privileged operation shall be examined and simulated by SS. This simulation shall be intended not to change the sequence of DCSG operations. For example, the Start I/O (SIO) instruction shall cause SS to perform the following sequences:

- Determine the simulated device being referenced
- Find the real device associated with the simulated device

- Interrogate the channel command word to determine the intent of the command
- Accomplish the I/O request
- Construct a psuedo I/O interrupt to be forced upon the ECS
- Return to the ADC program that causes the interrupt

4.3.2 External Interrupt Routine

This routine shall cause the Timer Routine to process this interrupt. The functions of this routine shall be:

- Update the total running time of the ADC programs
- Enter the Debugging Aids routine to output data concerning this interrupt
- Construct the return to the External Interrupt Handler of the ECS

4.3.3 Machine Check Interrupt Routine

Machine Check Interrupts shall be caused by machine malfunction. If the interrupt initiates within an ADC program, SS shall allow the Machine Check Handler of the DCSG to attempt recovery. If it is not initiated within an ADC program, the execution shall be terminated.

4.3.4 SVC Interrupt Routine

SS shall be concerned with two SVC's originating from ADC programs. These are the Segment End, and Program Unit End SVC's. When recognized, program unit traces and optional debugging information shall be output if requested on the Debugging Aids Control Card. Control, after all SVC's have been processed by SS, shall be transferred to the ECS SVC Handler.

4.3.5 Debugging Aids Routine

This routine shall provde the debugging support for SS. Output from this routine shall consist of such debugging information as program traces, main storage dumps, and contents of the general registers. When specified on the Debugging Aids Control Card, a trace shall be produced. Once specified, the trace shall be output at the end of each segment of the program unit, at a program interrupt error condition or at an interval timer interrupt.

Trace shall consist of:

- Program unit name
- The location of the segment end
- Starting location of the segment
- Trace label (segment end, type of error, etc.)
- Next segment entry point
- Timer information
- Contents of the general registers
- Optional main storage dumps (or a dump of the entire program unit if an error condition)

If the trace option is not specified, a main storage dump and contents of the registers shall be output if an interrupt occurs because of an execution error, or the interval timer.

4.4 APPENDIX

Insertion of Data

Data shall be inserted (input) to the ADC programs in two ways:

- Programmer generated data residing on a tape unit
- Programmer generated data residing in core within the particular program unit.

Inputs Residing on Tape

Data residing on tape shall be formatted into files for each device accessed within a program unit. If more than one program unit accesses the same device, separate files shall be maintained. Within these files a record shall be generated by the problem programmer for each time the device is accessed.

A time dependent event file shall also reside on tape. This file shall be generated by the problem programmer to introduce data required to respond to a TDE caused I/O interrupt.

Inputs Residing in Core

The problem programmer shall have the option to insert data as a separate segment within his program unit. This shall be accomplished by using the SYMBDICT control statement in the Program Preparation Processor (Paragraph 3.0)

PART 5

SYSTEM/360 M44 ADC SUPPORT

5.1 INTRODUCTION

System/360 M44 ADC Support shall contain a set of programs, routines, and macros to:

- Simulate the AMU, the Airborne Vehicle Equipment (AVE) printer, and partially simulate the Keyboard Display Unit (KDU) on the System/360 M44.

5.2 SYSTEM/360 M44 ROUTINES FOR ADC I/O SIMULATION

These routines shall provide the means to:

- Simulate the AMU
- Simulate the AVE Printer
- Partially simulate the KDU

These routines shall operate under control of 44PS. There shall be no simulation of error conditions. Each of the routines shall accept requests from and send data to the Airborne Digital Computer Adapter Subsystem (ADCAS) Interface Routines. (The ADCAS Interface Routine description is not included in this document.)

5.2.1 AMU Simulation Routine

This routine shall accept requests from the ADCAS Interface Routine to position the Simulated AMU Tape, and to supply data and sense information to the ADCAS Interface Routine for delivery to the ECS.

5.2.1.1 Normal Functions. AMU commands shall be simulated by the AMU Simulation Routine. If a sense command is received by this routine, a "no error" condition shall be presented to the ADCAS Interface Routine.

5.2.1.2 Error Detection. There are two types of errors which could arise in execution: actual errors for the 2400 tape and simulated errors for the AMU.

- Permanent 2400 tape errors shall be logged as such and the appropriate error return shall be given
- AMU errors shall not be simulated in the status or the sense bits.

5.2.2 AVE Printer Simulation Routine

This routine shall provide functional simulation of the AVE printer. Data input to the Printer Simulation Routine shall consist of 8 bit; each byte containing a 6 bit code. Each valid byte shall be converted from ADCII (6 bit) to one of the 8 bit EBCDIC codes in the 48 character print chain by the Printer Simulation Routine. Data shall be output to the 1403 printer.

5.2.2.1 Modes of Output. Output shall normally consist of 24 character data lines centered on a 132 character print line. An option shall be provided to include additional commentary pertaining to the print command received, the data received, diagnostic information, etc. Options shall be provided for assembly and execution.

5.2.2.2 Classification of Output. An indicator shall be set to specify whether or not the output is to be formatted with security classification headers. A security classification value defining classification of the job shall be provided. One classification per job shall be permitted. If no classification value is provided and classification is requested, the highest defined security classification shall be assumed.

5.2.2.3 Error Handling. The provisions of 44PS shall be used for error recovery. If a sense command is received by this routine, a "no error" condition shall be presented to the ADCAS Interface Routine

5.2.3 KDU Simulation Routine

This routine shall partially simulate the functions of the KDU by means of a 1052 console typewriter. The functional keyboard facility shall not be simulated by this routine. The simulation routine shall provide input and output of printed data only.

5.2.3.1 Information Code Conversion. ADC data input to the KDU Simulation Routine shall consist of 8 bit bytes, each byte containing a 6 bit code indicating character or control. Each byte shall be interpreted and converted as necessary from ASCII (6 bit) to 8 bit EBCDIC code or output on the 1052. Alphabetic characters (upper or lower case) from the 1052 shall be interpreted as upper case only.

5.2.3.2 Display Output Representation.

Because 1052 input and output consists of printed characters, output representing displayed information on the KDU shall be distinguished by indicators.

PART 6

SYSTEM/360 M44 ADC SUPPORT

6.1 INTRODUCTION

System/360 M44 ADC Support shall contain a set of programs, routines, and macros to:

- Generate an AMU Tape

6.2 ADC PROGRAM FOR AMU TAPE LOADING AND VERIFICATION

The AMU Tape Loading and Verficiation Program (TLV) shall load an AMU Tape and verify data transmission. Data transmission or comparison errors shall be displayed. This program shall execute on an ADC with a 2400 series tape, an AVE printer, and a KDU that interface with the ADC through a Printer/AMU Adapter (PAM). This two phase program shall be complete in itself.

6.2.1 AMU Tape Load Phase

This phase shall read data from the AMU Load Tape and write it on the AMU Tape. The input for this program shall be the second file on the AMU Load Tape. The progress of the writing process shall be displayed. Each data transmission error shall be presented with an indication of its location. The program shall log every error. Permanent errors shall be logged as such and termination shall take place.

6.2.2 AMU Tape Verification Phase

This phase shall read the second file of the AMU Load Tape and the AMU Tape synchronously and compare all the data. Indication shall be provided of the progress of the verifying process. Each data transmission or comparison error shall be presented with an indication of its location. The program shall log every error. Permanent errors shall be logged as such and termination shall take place.

Section 2

INPUT/OUTPUT FUNCTIONAL AND DCSG SELF-TEST AND DIAGNOSTIC REQUIREMENTS (ATTACHMENT NO. 2)

PART 1

I/O AND DCSG SELF-TEST AND DIAGNOSTIC PROGRAMS

1.1 PRINTER OUTPUT PROCESSOR

This program shall accept data from any on-board operational program and initiate data output to a hardcopy printer.

1.1.1 Source and Type of Inputs

The following are input to the Printer Output Processor.

a. Output data from any operation program.

b. Conversion request and format specification from any operational program requesting output to the printer.

1.1.2 Destination and Type of Outputs

Print with a request to the Executive Control System to initiate transmission of print data to the hardcopy printer is output.

1.1.3 Information Processing

This function shall be initiated upon receipt of print data from any operational program. The data to be printed shall be converted to ASCII code. Two levels of priority shall be recognized in the data to be printed.

1.2 DISPLAY OUTPUT PROCESSOR

This function shall accept display data from any operational program and initiate transmission of these data to the display unit.

1.2.1 Source and Type of Inputs

Inputs to the Display Output Processor are as follows:

a. Display data from any operational program.

b. A request for conversion of numerical data to ASCII codes, and format specifications from a program requesting a display.

1.2.2 Destination and Type of Outputs

Display data with a request to the Executive Control System to initiate transmission of display data to the display is output.

1.2.3 Information Processing

This function shall accept display data from any operational program. Data to be displayed shall be converted to ASCII code. Two levels of priority shall be recognized in the data to be displayed for at least 10 seconds before being replaced by a higher priority message.

1.3 KEYBOARD INPUT PROCESSOR

The Keyboard Input Processing Function shall accept, and translate inputs originating at the Keyboard and the Master Control Console.

1.3.1 Source and Type of Inputs

The following are inputs to the Keyboard Input Processing Function:

a. The inputs shall consist of 8-bit codes representing the following:

1. Alphabetic characters
2. Numeric characters
3. Special characters
4. Function controls
5. Data entry controls
6. Backup codes

b. A complete message consisting of a string of these codes shall be accepted as a higher-level input. These inputs shall include the following:

1. A request for the value of a specified telemetry parameter.

2. A request for the contents of a specified location in main memory to be displayed
3. An immediate or delayed command to the Command Processor
4. A request to the Executive Control System for the initiation of a specified program
5. A data input to a specified memory location
6. A message to the Downlink Processor

1.3.2 Destination and Type of Output

The following are output from the Keyboard Input Processing Function:

a. A Keyboard input message shall be output to the display.
b. A request to the Data Acquisition Function for the acquisition, conversion, limit check and display of a specified telemetry parameter.
c. A request to the Display Output Processor to display the contents of a specified memory location.
d. An immediate or delayed command to the Command Processor.
e. A request to the Executive Control System for the initiation of a specified program.
f. A data transfer to a specified memory location.
g. A message to the Downlink Processor .
h. A Time Reference Signal to the Gemini Deorbit Data Handler.

1.3.3 Information Processing

The processing of the Keyboard Input Processing Function is as follows:

a. The Keyboard Input Process shall be initiated upon the receipt of a Keyboard or Master Control Console code. Alphabetic, numeric and special characters shall be routed to the Display Output Processor. A string of these characters shall be assembled in memory as a message capable of being edited from the Keyboard. The following control codes shall be interpreted as requiring the indicated implementation:

1. CLEAR -- The entire message shall be erased from the memory and from the display.
2. BACKSPACE -- The last character input at the Keyboard shall be erased.
3. SPACE -- The next character position shall be skipped.
4. ENTER -- Processing of the completed message shall be initiated.

The capability shall be provided to substitute a backup code for a malfunction code.

b. A completed message shall be checked for syntax, validated and interpreted as follows:

1. Request for a telemetry parameter -- A request shall be sent for the Data Acquisition function to sample the specified telemetry parameter.
2. Request for memory data -- This input message shall enable smapling, and display of the contents of a specified location in memory in hexadecimal code.
3. Immediate and Delayed Commands -- The input message shall enable the immediate execution of an immediate command and shall provide the storage for later execution of a delayed command. The commands shall be sent to the command processor.
4. Request for Initiation of a Program -- A request shall be made to the Executive Control System for the initiation of the specified program.

5. Manual Input of Data to Computer -- The input of certain data into code storage shall be permitted under control of the Executive Control System.
6. Downlink Message -- The input message shall be transmitted to the Downlink Processor without modification for PCM transmission to the ground.

1.4 DCSG SELF-TEST

The DCSG Self-Test Program shall be co-resident in main storage with the Executive Control System (ECS) and the operational programs, or alternately may be stored on the AMU and called when needed. It shall be executed periodically as a nonpriority program under control of the ECS and shall not interfere with the operation of the operational programs. The DCSG Self-Test shall provide a moderate check of the ADC, LDA, and KACU. The DCSG Self-Test program shall perform the following functions:

a. ADC Test -- This function shall test the arithmetic and logical elements of the ADC.
b. LDA Test -- This function shall echo-check each interface of the LDAU.
c. KACU Test -- This function shall echo-check the KACU.

1.4.1 Source and Type of Input.

Inputs to the DCSG Self Test are:

a. ADC Test -- None
b. LDA Test -- Input to the LDA Test shall be test data read from the LDA interface shift registers.
c. KACU Test -- Input to the KACU Test shall be a data byte returned from the keyboard data register.

1.4.2 Destination and Type of Output

Output of the DCSG Self-Test Program is as follows:

a. ADC Test -- Upon detection of an ACD failure, output from the ADC test function shall be the ADC malfunction signal to the DCSG Computer Subsystem Controller (CSC).

b. LDA Test

   1. Test commands to the LDA interface shift registers.
   2. Notification to the ECS of the status of each of the LDA interfaces .
   3. The LDA malfunction signal to the CSC if any of the interfaces fail.

c. KACU Test

   1. Test data to the Display data register.
   2. If the test was unsuccessful, notification to the ECS of the failure.

1.4.3 Information Processing

The processing of the DCSG Self-Test Function is as follows:

a. The ADC test shall determine the operational capability of the major ADC data paths and functional elements, as follows:

   1. Each of the ten mover actions shall be executed using data patterns designed to detect hardware malfunctions. With the exception of I/O facilities, all sources of data to the right and left mover inputs shall be exercised. With the exception of the I/O facilities, all destinations of mover output shall be exercised.

   2. Each of the eight adder actions shall be executed using data patterns designed to detect hardware malfunctions.

3. With the exception of I/O facilities, each of the sources of data to the right - and left - adder inputs shall be exercised. With the exception of I/O facilities, all adder output destinations shall be exercised.
4. Each of the eight shift instructions shall be executed using data patterns designed to detect hardware malfunctions.
5. Each local storage register shall be addressed, and proper data transfer and retention shall be established.

b. The LDA test shall echo check the AMU Interface, Printer Interface, ACTS Interface, DASG Interface, Command Output Interface, and Command Uplink Interface. The AMU, Printer, ACTS, DASG, and CSG themselves shall not be exercised during this test. For each interface, the computer shall issue a Test Write command with bit 5 set to a 1. As a result, the command code byte shall be placed in the interface shift register. A test read command shall then cause the byte to be read back into the computer where it shall be compared with the original byte. If the bytes are not the same, the ECS shall be notified that this particular interface is malfunctioning. If any of the interfaces do not work properly, the LDA malfunction signal shall be set and the ECS notified.

c. The KACU Test shall:
1. Issue a write command to the Display with bit 0 set to 1.
2. Issue a read command to the Keyboard and compare the byte returned with the original byte.
3. If the comparison fails, notify the ECS.

1.5 DCSG MAINTENANCE DIAGNOSTIC

This function shall provide a thorough test of each DCSG device. It shall be operated during activation of the laboratory and thereafter at the crew's request. The Maintenance Diagnostic functions other than the ADC test shall be performed only on the master functions other than the ADC test shall be performed only on the master computer. The Maintenance Diagnostic shall perform the following tests:

a. ADC Test.

   The DMDP ADC test shall prove the operational capability of the following ADC functions:

   1. Instruction Execution
   2. CPU Data flow
   3. Arithmetic/Logical Processing
   4. ROS Addressing and Decoding
   5. Micro-order Execution and flow
   6. Main Storage Addressing and Data Transfer
   7. Local Storage Addressing and Data Transfer

b. LDA Test -- The LDA test shall check the LDA interfaces, the Internal Control Registers, the Channel to Channel Adapters, and the CSC channels.

c. KACU Test -- The KACU Test shall provide a thorough test of the KACU

d. Printer Test -- This function shall test each Printer Unit's ability to accept and print each of the 48 possible character codes at each of the 24 possible character positions.

e. Keyboard Test -- The Keyboard test shall check the Keyboard's ability to accept an operator input and process it properly.

f. Display Test -- The Display Assembly Test shall check the ability of the Display to accept data from the ADC and display it properly. It shall require operator verification of the displayed data.

g. AMU Test -- The AMU Test shall employ and test the operational ability of the following components of the Auxiliary Memory Unit.
    1. Command/control logic as required for on-orbit.
    2. Commands (read, read backward, sense, and control).
    3. Amplifier and detect functions.
    4. Voter matrix function.
    5. Output data register and parity bit generator.
    6. Parity and validity.
    7. End and beginning of tape sensing.

1.5.1 Source and Type of Input

Inputs to the DCSG Maintenance Diagnostic Function are as follows:

a. ADC Test -- None
b. LDA Test -- The following shall be inputs to this test:
    1. Notification from the ECS of the status of the Slave Computer.
    2. Test data returned from the LDA interface shift registers.
    3. Test data read from the LDA Internal Control Register.
    4. Test Data returned by the slave computer
    5. Telemetry sample of the CSC configuration register.
c. KACU Test -- The following shall be inputs to this test:
    1. Test data returned from the Keyboard data register.
    2. Bus-out parity errors from the Keyboard and Display sense registers.
d. Printer Test -- Input to Printer test shall be operator keyboard verification of the printed data patterns.
e. Keyboard Test -- Input to the Keyboard Test shall be bytes of data from the Keyboard Assembly.

f. Display Test -- Input to the Display test shall be operator verification of the test data via the Keyboard assembly.

g. AMU Test -- The inputs to the AMU test routine shall be test data blocks read from the AMU.

1.5.2 Destination and Type of Output

Outputs of the DCSG Maintenance Diagnostic Program are as follows:

a. ADC Test -- Upon detection of an ADC failure, output from the ADC test shall be the ADC malfunction signal to the CSC.

b. LDA Test

   1. Test Command bytes to the LDA interface shift registers.
   2. Notification to the ECS of the status of each of the LDA interfaces, of the internal control register, the Channel to Channel Adapter, and the CSC.
   3. The LDA malfunction signal to the CSC if any interfaces of an LDA fail.
   4. Test data to be transferred to the Slave Computer.
   5. Command outputs to the CSC.

c. KACU Test

   1. Test commands and data to the Display data register.
   2. Test commands to the Keyboard.
   3. Invalid commands to the Display and Keyboards.
   4. If the test was unsuccessful, notification to the ECS of the failure.

d. Printer Test

   1. A block of test data written to the DCSG Printer.
   2. Notification to ECS of each Printer!s operability.
   3. A display requesting operator verification of the test.

e. Display Test

   1. Test data displayed on the DCSG Display Unit.
   2. A printer message denoting the start of the test.
   3. Notification to the ECS of test failure.

f. Keyboard Test

   1. A message to the DCSG Display Assembly requesting the start of operator inputs.
   2. Notification to the ECS of the operability of the Keyboard.

g. AMU Test

   1. In the event of an AMU test failure, notification to the ECS.
   2. I/O commands written to the AMU.

1.5.3 Information Processing

The processing of the DCSG Maintenance Diagnostic Program is as follows:

a. ADC Test

   1. With the exception of I/O instructions, each machine instruction shall be executed using data patterns explicitly chosen to detect malfunctions in the arithmetic/logical unit and data paths.
   2. With the exception of the I/O mode microinstructions, and those which would cause permanent loss of control, each microinstruction in the ADC microprogram shall be addressed and executed.
   3. A sum check of all main storage available to the ADC test shall be executed. In addition, all available main storage shall be hard cycled as one of the functions of the Diagnose instruction.
   4. Proper local storage data retention and change of state shall be verified by cycling ones and zeros through each general register. To establish proper local

storage addressing, an array of 16 distinct data factors shall be loaded in one order, restored in a different order, and the result array analyzed for errors.

b. LDA Test -- The LDA test shall perform as follows:

1. Issue a test write command to each interface followed by read commands.
2. Upon return of the test byte ,compare it to the original byte.
3. Issue invalid commands to each interface and await a command reject response.
4. Write a comprehensive set of patterns to the Interface Control Register, read those patterns back, and compare them to the original.
5. Write patterns of test data to the slave computer, read them back and compare the patterns.
6. Read backwards the Channel to Channel Adapter test patterns and again compare results.
7. Write configuration commands to the CSC, wait approximately 400 sec, sample the CSC configuration register and verify that the commands were received properly.
8. Notify the ECS of any malfunction detected.

c. KACU Test -- The KACU Test shall perform as follows:

1. Issue test write commands to the Display which shall transfer test data bytes to the display data register.
2. Issue (test) read commands to the Keyboard data register and compare the results to the original.
3. Issue invalid commands to the Keyboard.

4. Issue invalid commands to the Display.
5. Notify the ECS of any failure of the KACU to respond properly.

d. Printer Test -- A complete set of characters from a preestablished pattern in core shall be printed. Afterwards the operator shall be asked, via the Display Assembly, to verify the printout. If the response is negative, the ECS shall be notified.

e. Keyboard Test -- The operator shall be asked, via the Display, to depress a predetermined sequence of keys. The keyed input shall be compared to a predetermined pattern. If the input does not match the pattern in storage, the ECS shall be notified of the Keyboard malfunction.

f. Display Test -- The Display Test shall perform the following:
   1. Notify the operator via the printer that the display test is to begin.
   2. Write patterns of data to the Display Assembly which will cause each of 46 characters to be displayed in each possible position.
   3. If the test fails, notify the ECS.

g. AMU Test -- The AMU test function shall perform the following:
   1. Cause the AMU to seek the test data record by execution of the forward space multiple record command.
   2. Read the test block and compare it to its known contents.

3. Cause the tape to skip backward the then skip forward.
4. Cause the AMU to execute the read backward command and, again, check the data.
5. Cause the AMU to execute the rewind command.
6. If the test fails, notify the ECS.

Section 3

SOFTWARE DEVELOPMENT SCHEDULES

(ATTACHMENT NO. 3)

Section 3

SOFTWARE DEVELOPMENT SCHEDULES (ATTACHMENT NO. 3)

	DAC I/O	*Self Test & Diag.	ECS	Math Util.	Simulation Supervisor	Program Prep. Proc.	AMU Tape Load & Ver.	System/360 M44 ADC I/O Sim.
Part I CEI			5/15/67	5/15/67	5/1/67	5/1/67	7/15/67	5/15/67
Item Test Plan			5/15/67	5/15/67	5/1/67	5/1/67	9/15/67	6/19/67
GE-PDR			6/1/67	6/1/67	5/17/67	5/17/67	8/15/67	6/1/67
Part 2 CEI	8/15/67	10/2/67	8/15/67	8/15/67	6/15/67	6/15/67	10/15/67	8/1/67
Item Test Proc.	8/15/67	10/2/67	8/15/67	8/15/67	7/15/67	7/15/67	1/15/68	9/15/67
GE-CDR	8/30/67	11/1/67	8/30/67	8/30/67	7/12/67	7/12/67	11/15/67	8/15/67
Prel. Oper. Guide	8/30/67		8/30/67	8/30/67	8/15/67	8/15/67	1/15/68	8/1/67
Prel. Delivery (Not Validated)	10/2/67		10/2/67	10/2/67	10/2/67	9/1/67		
Version Disc.	3/1/68	5/1/68	3/1/68	3/1/68	1/15/68	1/15/68	5/15/68	12/4/67
Item Test Report	3/1/68	5/1/68	3/1/68	3/1/68	1/15/68	1/15/68	5/15/68	12/4/67
Final User's Man.	3/1/68	5/1/68	3/1/68	3/1/68	1/15/68	1/15/68	5/15/68	12/4/67
G. E. Acceptance	3/1/68	5/1/68	3/1/68	3/1/68	1/15/68	1/15/68	5/15/68	12/4/67

*Based on DAC PDR 5/1/67

APPENDIX

RESUMES

APPENDIX

RESUMES

Resumes of the key scientific and programming personnel to be used on this MOL Computer Program Development Program follow:

ROBERT T. ELLSWORTH, JR., MOL Program Director

Education:	BS, Massachusetts Institute of Technology MBA, University of Buffalo
Experience:	12 years experience, 10 at IBM

Mr. Ellsworth's 12 years in administration, engineering, and marketing have provided him with a wide range of experience in all aspects of aerospace programs. His last position before this assignment was assistant to the President of the Federal Systems Division.

Mr. Ellsworth's previous assignments have provided an extensive background in the technical, fiscal, and contract requirements of both ground-based and spaceborne information handling systems for the government. His technical experience includes navigation and guidance equipment; inertial optical and radar sensors; special purpose ground data handling systems; and systems integration. Those previous assignments include:

- Assistant Programs Director of scientific and space operations. In this position, Mr. Ellsworth was concerned with NASA programs including Project Mercury range computation, Saturn computer and instrumentation unit, real time computer complex, and Gemini inertial guidance system, and with intelligence data processing projects such as 438L (IDHS), 466L, and fleet intelligence centers.
- Program Manager for aerospace systems (Air Force programs). Mr. Ellsworth's experience ranged from Air Force study

requirements in the space area through data handling support to Lockheed on 117-L.

Mr. Ellsworth's assignment as Assistant to the President of FSD, Marketing Manager for the Electronics System Center, and FSD Washington Marketing Manager, have provided him with direct contact and familiarity with the government community. His overall experience will insure effective communication and integration between IBM, Douglas Aircraft Company, and the General Electric Company, rapid response to customer direction and efficient application of IBM resources to the 632A program.

BURTON P. WHIPPLE, MOL Programming Manager

Education: BA Mathematics, University of Connecticut, 1951

Experience: 14 Years Experience, 6 at IBM.

Mr. Whipple is presently responsible for all programming projects related to MOL, reporting directly to the MOL Program Director. Prior to this assignment, he was manager of the Systems Conversion Department Group responsible for converting computer programs from IBM's 1401, 1410, 1620, 7040, 7070, 7080, and 7090 to System/360.

Previously, Mr. Whipple was a Special Representative in Programming Systems, acting as a consultant to IBM customers.

He also has six years of experience as Programming Manager, working on various scientific programming projects for the Federal Government and performing systems and analyses for Rocketdyne Division of North American Aviation.

RICHARD B. TALMADGE, PhD - Consultant: Software Problems

Time: 50%

Education BS, California Institute of Technology - 1948

PhD, California Institute of Technology - 1951

Experience: 10 Years at IBM

Dr. Talmadge has 14 years experience in computing, 6 of wich were spent in aerospace applications. His present assignment is special assistant for programming to Mr. A. E. Cooper, Vice-President and General Manager of Space Systems Center. Dr. Talmadge's experience includes a variety of applications in system programming efforts, including numerical analysis, radar tracking, compressible fluid flow, free-jet nozzle design, and missile trajectories. System work includes design and implementation of interpretative processors, compilers, assemblers and control programs.

Previous assignments with IBM include:

- Manager, Experimental Systems Group, Los Angeles Scientific Research and Development Center. Dr. Talmadge's group developed the design for a large scale multi-processing network.
- Senior Programmer, Programming Systems, Data Systems Division where he was active in the design of the IBJOB processor for the 7090/94, and had responsibility for the design, implementation, and checkout of the 7090/94 IBMAP Assembler.
- Manger, Commercial Translator Project, Applied Programming Department, Data Systems Division. During this two year period, Dr. Talmadge's group designed and produced the 709/7090 commercial translator processor, the 709/7090/7094 input/output control system, and the initial version of the 709/7090/7094 IBSYS monitor.

Experience prior to IBM includes the following:

- 3-1/2 years with Lockheed Missile Systems Division as Manager of Systems Programming and as a research scientist. System programming efforts produced a complete operating system for the UNIVAC 1103A. Prior work involved a number of applications; the most significant of which was an extensive 3D missile trajectory program with which design optimization could be performed.
- 1-1/2 years with Marquardt Aircraft Corporation as Research Scientist and Manager of Computer Programming. Work performed involved design of variable Mach number free-jet nozzle.
- 2 Years with Hughes Aircraft Company engaged in various aspects of analysis and computation.

WILLIAM F. HUBBARTH, PhD - Senior Engineer - Mission Simulator Development Manager

Education: BA, College of Wooster - 1952

MA, Ohio State University - 1953

PhD, Ohio State University - 1956

Experience: 11 years at IBM.

While employed at IBM, Dr. Hubbarth has directed and performed system analyses on advanced manned systems including a variety of air, missile, and spacecraft systems. He has been program director of human factor systems analysis, simulation studies and control studies. He has developed methods of systems analysis and simulation that extend into all aspects of subsystems implementation.

He was Director of human engineering, system analysis and procedures simulation on AN/ASQ-28(V) program; Director of human factors design activities at IBM SGC; Director of the IBM manned space guidance and control program; Experiment definition and Simulation consultant for MORL, AORL, and OSSS; MOL simulation project director; IVSS Assistant Study Director and Simulation Task Director; MOL Mission Simulator Phase O, Phase 1(b) and Phase II Program Manager. He is currently responsible for all aspects of Simulation in Program 632A as a manager in the MOL Software Function.

ANDREW H. OLSON, Technical Assistant MOL Software Development

Education: BS, University of Southern California
Advanced Graduate Studies, University of Southern California

Experience: 10 Years, 2 Years with IBM

Mr. Olson reports directly to the MOL Programming Manager and is responsible for insuring contractual compliance for MOL Programming activities.

Prior to this assignment, Mr. Olson was on the Saturn Systems Programming Staff activity at Huntsville in KSC.

Prior to joining IBM, Mr. Olson spent two years with the computer programming integration contractor for the USAF Satellite Control Network. He spent two years on the staff of the GSBA Computer Center at the University of Southern California. Prior to this, Mr. Olson was with the Rocketdyne Division of North American Aviation on the Atlas ICBM Program.

BERNARD D. RUDIN, PhD - Consultant, Computing Techniques
Time: 50%

Education: BS, California Institute - 1949

MS, University of Southern Calif. - 1951

PhD, Stanford University - 1965

Experience: Dr. Rudin's experience has all been in computing for the aerospace industry. His present assignment is that of Manager, Information Technology Development, reporting directly to the Manager MOL Software Development. Dr. Rudin's experience includes work in numerical analysis, trajectory computation, and other tasks in applied mathematics. In addition, he has participated in and directed research and development activities in linguistics, programming systems, and special purpose computer design.

Experience prior to joining IBM includes:

12 years with Lockheed Missile and Space Co., Palo Alto, California. During the last six years of this period, Dr. Rudin was a consulting scientist and Senior Member of the Research Laboratory directing a group of 20 scientists and engineers in research and development in the information sciences field. Before that, he was Manager of Programming Research and head of Scientific Computation for the company's Computating Center.

- 1-1/2 years with Marquardt Aircraft Company, Van Nuys, California, as Mathematical Engineer, Numerical Analysis group. Work included programming of missile system simulations.

- 1-1/2 years with Lockheed California Company, Burbank, California as Mathematical Analyst doing analysis and programming on problems in aircraft structures and flutter.

GEORGE T. HAZELWORTH, Manager, Laboratory Vehicle Programming Development

Education: BS, Michigan State University

Experience: 15 Years

Mr. Hazelworth is presently Manager of Laboratory Vehicle Programming Development. He is responsible for the overall design and development of the On-Board Executive, Math. Utilities, Program Preparation Processor, Simulation Supervisor, and System/360 ADC Support.

Prior to his current position, Mr. Hazelworth was a Special Representative in the Western Region Programming Systems Department for IBM.

MARIO SURDI - Development Manager

Education: BS, Math, Queens College

Experience: 10-1/2 years experience with IBM.

Mr. Surdi is presently responsible for the On-Orbit Executive. Mr. Surdi was also Manager of FORTRAN G for IBM System/360 Programming Systems.

Prior to that, he worked on the Advanced Operating System for the IBM System/360. Previously he was an Advisory Programmer on COBOL F for System/360 and Manager of COBOL for the IBM 7040/44 systems and COBOL/7070 project coordinator.

He has also worked on Language Development, 7070 Comtran, BOUARC Simulation, LOBAL System and on Software error recovery for hardware failures.

STANLEY COHN, Project Manager

Education: BS Accounting, New York University
Graduate Studies - 1 Year Towards
BS, Operation Research -- Rutgers

Experience: 7 Years — 5-1/2 with IBM

Mr. Cohn is presently a manager on the MOL Project. Previously, he was manager of FORTRAN E and COBOL E maintenance for IBM System/360.

Prior to that, he was Project Coordinator for COBOL F, Coordinator for COBOL and IBM 7040/44 systems and worked on COBOL for the IBM 7070 system.

Before joining IBM, he worked on the JOVIAL Compiler for System Development Corporation.

MONTE MINAMI - Development Programmer

Education: BA Mathematics, UCLA - 1954

Experience: 12 years, 5 with IBM

Design, programming and modifying operating systems for large scale digital computers, compilers (FORTRAN, PACT), Assemblers (SAP, FAP, SCAT, etc.) He also has 5 years experience with IBM as a systems programmer (IBM 709 - 7094).

Experience Previous to IBM:

- 3-1/2 years at North American Aviation as a systems programmer for a large scale digital computer (IBM 701/704)
- 3-1/2 years at TRW Systems (was Space Tech. Laboratories), as a systems programmer (IBM 704/709).

HAROLD W. JOHNSON, Manager of System/360 Model 44 and ADC Support

Experience: 12 Years with IBM

Mr. Johnson is presently Manager of System/360 Model 44 and ADC Support reporting directly to Mr. G. T. Hazelworth, Manager of Laboratory Vehicle Programming Development, on matters of Model 44, 44 Programming System and ADC Integration Support.

Previously he was manager of Systems Techniques in the Data Processing Division; as such, he was responsible for the design and implementation of System/360 data utilities. His other experiences included Systems Engineer for a large aerospace customer, systems programming for IBM regional support and Customer Engineer on the 700 Series equipment.