NASA CR-
141709

DETAILED REQUIREMENTS DOCUMENT

FOR

COMMON SOFTWARE

Of

Shuttle Program Information Management System
Job Order 88-019

Prepared By

Lockheed Electronics Company, Inc.
Aerospace Systems Division
Houston, Texas

Contract NAS 9-12200

For

INSTITUTIONAL DATA SYSTEMS DIVISION

National Aeronautics and Space Administration

LYNDON B. JOHNSON SPACE CENTER

Houston, Texas

February 1975

LEC-5479

DETAILED REQUIREMENTS DOCUMENT
FOR
COMMON SOFTWARE
Of
Shuttle Program Information Management System
Job Order 88-019

PREPARED BY

J. M. Everette, Scientific Programming Specialist

L. D. Bradfield, Data System Programming Analyst

C. L. Horton, Data System Programming Analyst

APPROVED BY

W. B. Hopkins, Supervisor
SPIMS, System Design and
Integration Section

B. L. Brady, IDSD Project Manager
Shuttle Program Information
Management System
Institutional Data System Division

R. L. Ready, Manager
Shuttle Information Management
Department

R. R. Regelbrugge, Chief/FD6
Data Systems Development Branch

H. F. Thompson, Director
Applications Branch

for LaRue W. Burbank, Project Manager
Shuttle Program Information
Management System
Data Systems & Analysis Directorate

C. R. Huss, Chief
Institutional Data Systems Division

Produced By:

LOCKHEED ELECTRONICS COMPANY, INC.
For
Institutional Data Systems Division

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
LYNDON B. JOHNSON SPACE CENTER
HOUSTON, TEXAS

FOREWORD

Common software was conceived as a method for minimizing development and maintenance cost of the Shuttle Program Information Management System (SPIMS) applications while reducing the time-frame of their development. This document identifies those requirements satisfying these criteria and also describes those stand-alone modules which may be used directly by applications.

The SPIMS applications operating on the CYBER 74 computer, are specialized information management systems which use System 2000 as a data base manager. Common software will provide the features to support user interactions on a CRT terminal using form input and command response capabilities. These features will be available as subroutines to the applications.

CONTENTS

FIGURES

ABBREVIATIONS AND ACRONYMS

ASCII	American Standard Code for Information Interchange
CDC	Control Data Corporation
CMA	Configuration Management Accounting
COSMOS	Communications-Oriented Switching Monitor and Operating System
CPU	Central Processing Unit
CRT	Cathode Ray Tube
CS	Common Software
CTM	Communication Terminal Module
CTMC	Communication Terminal Modular Controller
DBMS	Data Base Management System
DMS	Data Management System
DRD	Detailed Requirements Document
ECS	Extended Core Storage
FDS	Functional Design Specification
GDSD	Ground Data Systems Division
HS	High Speed
H2K	Hazeltine 2000 Terminal
H4K	Hazeltine 4000 Terminal
ICE	Interface Control Executive
IDSD	Institutional Data Systems Division
I/O	Input/Output
JSC	Johnson Space Center
Kbs	Kilobits per second
LDP	Logical Data Path
LEC	Lockheed Electronics Corporation
LEC/ASD	Lockheed Electronics Corporation/Aerospace Systems Division
LS	Low Speed
MML	Master Measurement List

MOPS	Mission Operational Planning System
MRI	MRI Systems Incorporated
NASA	National Aeronautics and Space Administration
PDS	Problem Data System
PICRS	Program Information Coordination and Review Service
PLI	Programming Language Interface
PPU	Peripheral Processing Unit
RG	Repeating Group
RSPI	Resources, Scheduling and Procurement Integration
SIS	Shuttle Information Service
SLAHTS	Stowage List and Hardware Tracking System
SPIMS	Shuttle Program Information Management System
SSPO	Space Shuttle Program Office
S2K	System 2000
TCS	Terminal Control System
TTY	Teletype

LIST OF APPLICABLE DOCUMENTS

1. Implementation Feasibility Study for Space Shuttle Program Management Systems Application Final Report, LEC-1787, Lockheed Electronics Company, Inc., March 1974.

2. Terminal Control System (TCS) Functional Specifications, Revision 2, SP-378-77, Sperry UNIVAC Systems Programming, November 1, 1973.

3. Terminal Control System (TCS) Sequence of Events Specifications, Revision 2, SP-380-78, Sperry UNIVAC Systems Programming, November 1, 1973.

4. Control Data CYBER 70, Model 74, Computer System, System Description and Programming Information Reference Manual, Vol. 1-3, Revision F, Publication No. 60347400, 1974.

5. Worker Interface with COSMOS, UM-338-65, Sperry UNIVAC Systems Support Division, Revision 2, June, 1972.

6. System 2000 Reference Manual, MRI Systems Corporation, Revision A, July, 1973.

7. "Minutes of CYBER/494 I/F Software Meeting of 18 December 1974", Memorandum 74CLKA-668, Control Data Corporation, December 30. 1974.

8. ICE Functional Specification, Control Data Corporation, To Be Published.

9. ICE Functional Design Specification, 90CLKA0020-1, Control Data Corporation, To Be Published.

10. CRT Terminal System Software Interface Specification, SI-09637, EO 030E, Philco-Ford Corporation, October 29, 1971.

11. Hazeltine 2000 Operating Manual, HI1004, Hazeltine Corporation, August 1973.

12. Detailed Requirements Document for Stowage List and Hardware Tracking System, LEC-5362, Lockheed Electronics Company, January 1975.

13. Detailed Requirements Document for Problem Data System, Lockheed Electronics Company, Inc., To Be Published.

14. Detailed Requirements Document for Shuttle Information Service, Lockheed Electronics Company, Inc., To Be Published.

15. Detailed Requirements Document for Configuration Management Accounting, Lockheed Electronics Company, Inc., To Be Published.

16. SPIMS/CYBER Interface Control Document, Control Data Corporation, To Be Published.

1.0 SYSTEM OVERVIEW

1.1 IDENTIFICATION

The Common Software for the Shuttle Program Information Management System (SPIMS) is being developed in response to Job Order 88-019, initiated by the Data Systems Development Branch of the Institutional Data Systems Division, Johnson Space Center. Mr. B. L. Brady is the Job Order Technical Monitor. Project number 6400 has been assigned to the task for the fiscal year 1975.

1.2 BACKGROUND

The long-term, multi-organizational nature of the Space Shuttle Program has presented the Space Shuttle Program Office (SSPO) with a monumental information management problem. Numerous areas of responsibility have been defined by SSPO for the management of information. To date, JSC has been assigned responsibility to support the following application areas: loose equipment tracking, reliability, quality assurance, problem tracking, configuration accounting, documentation accession, and requirements traceability. The SPIMS project was established in order to automate the storage and retrieval of the massive volumes of data needed to fulfill these management responsibilities.

SPIMS is the outgrowth of a feasibility study performed during the second quarter of fiscal year 1974. The results of the study can be found in Implementation Feasibility Study for Space Shuttle Program Management Systems Application Final Report (see Reference 1). That study

concluded that an automated data management system could be implemented at JSC. SPIMS was formally inaugurated in July of 1974.

Six (6) specific applications are being addressed by SPIMS to date:

- Problem Data System (PDS)
- Configuration Management Accounting (CMA)
- Shuttle Information Service (SIS)
- Stowage List and Hardware Tracking System (SLAHTS)
- Master Measurements Data Base (MMDB)
- Resources, Scheduling, and Procurement Integration (RSPI)

Preliminary requirements surveys of four (4) of these applications (PDS, CMA, SIS, SLAHTS) revealed that a number of functional capabilities were required by two or more of the applications. The aggregate of these shared functional requirements came to be known as the Common Software for SPIMS applications. The premise on which the Common Software concept was based was that a single development of software to fulfill the common requirements would permit its cost to be amortized over all of the utilizing applications, yielding a significant overall cost-savings for the SPIMS application software. Further, the development cost of additional applications that are anticipated in the future would be greatly reduced.

This Common Software Requirements Document identifies those applications requirements which have been firmly established as being common to multiple applications.

1.3 GENERAL DESCRIPTION

The principal source of all SPIMS requirements is the applications: PDS, CMA, SIS and SLAHTS. The PICRS feasibility study synthesized a system, both hardware and software, to satisfy the different applications. Different functions were allocated to the system components. One of these components is common software.

The current SPIMS development effort consists of only four of the six original applications. These are PDS, CMA, SLAHTS and SIS. This requirements development effort further refines the PICRS feasibility study definition of common software. The study indicates that the primary need for common software is to facilitate terminal communications.

When using forms (see section 2.2 for a description of forms) it is necessary to transmit large quantities of data to and from the terminal in a single transmission. This capability is not supported by KRONOS/TELEX; therefore, it will be accomplished via common software. Functions common to several applications will be supplied by common software.

The PICRS feasiblity study rated System 2000 weak in such areas as input validation and input/output translation. Common software will provide these and other input/output processing capabilities.

1.4 ASSUMPTIONS AND CONSTRAINTS

The following list enumerates those assumptions and constraints that currently guide the implementation of common software as it is described in this document:

- It is assumed that a multi-thread version of System 2000 will be delivered to the JSC CYBER 74 installation by June 15, 1975.
- It is assumed that the CYBER 74/TCS interface will be operational by August 19, 1975 and that testing of that interface will begin approximately August 1, 1975.
- The implementation of common software is constrained to the development of a sub-program library from which each application can extract those functional units that are applicable to its own needs.
- The implementation of common software input and output processing functions; validation, translation, editing, etc. is constrained to be subroutines which operate on only one parameter and perform one function.
- The implementation of common software is constrained from providing responses to the terminal when an error is recognized except as indicated in paragraph 3.2.10. Error indicators will be returned to the applications program.
- Certain of the functional capabilities of common software are required earlier than others. The development is constrained to deliver these items in accordance with the need dates negotiated with the applications developers. (Section 5 outlines the current priorities for the common software development.)

- The requirements and priorities contained in this document are based on current knowledge of the applications listed in paragraph 1.2. Addition of new applications or major changes in the requirements of the initital applications may cause impact to be experienced in the common software development effort.
- It is assumed that the communications mode described in references 7 and 8 exists in that form. (This mode prescribes the use of mass storage files for passing terminal input/output messages to applications programs.)
- Subroutine calls, common software table formats, and other interface procedures between applications programs and common software will be defined in the Common Software Design Document.

1.5 SPIMS SYSTEM DESCRIPTION

SPIMS is a multi-dimensional development effort involving many facets of classic system development:

- Hardware procurement
- Special hardware design and fabrication
- Communications and message switching
- Operating system modification and extension
- Applications development

In addition, SPIMS encompasses two additional software items: a data base management system, and common software. The sections that follow present an overview of the nature

and function of all of the components with special emphasis on the common software.

1.5.1 Hardware Configuration

Figure 1-1 illustrates the hardware components required to support SPIMS. SPIMS is primarily a terminal oriented system. The end-user will normally have access to one or both of two types of terminal devices: a Hazeltine 2000 CRT or a Hazeltine 4000G CRT. Both types are capable of buffered operation, that is, local screen editing can be performed without computer intervention. When a complete display has been prepared, the user may transmit the buffered data to the computer. Some users may access SPIMS via TTY compatible terminals operating in line mode. TCS (described below) will be configured to handle such terminals.

Communications and message switching capabilities are provided through the facilities of the Terminal Control System (TCS). Terminals may be located at JSC or at sites remote from the Center. Communications are transmitted over standard telephone lines in some cases and over dedicated communications circuits in others. TCS comprises a UNIVAC 494 computer and all of the communications lines and associated equipment required to effect terminal input and output. As currently configured, TCS will support up to 64 SPIMS interactive terminals logically connected to the SPIMS application computer in addition to providing computer to computer, computer to terminal, and terminal to terminal communication functions. (TCS is capable of supporting more

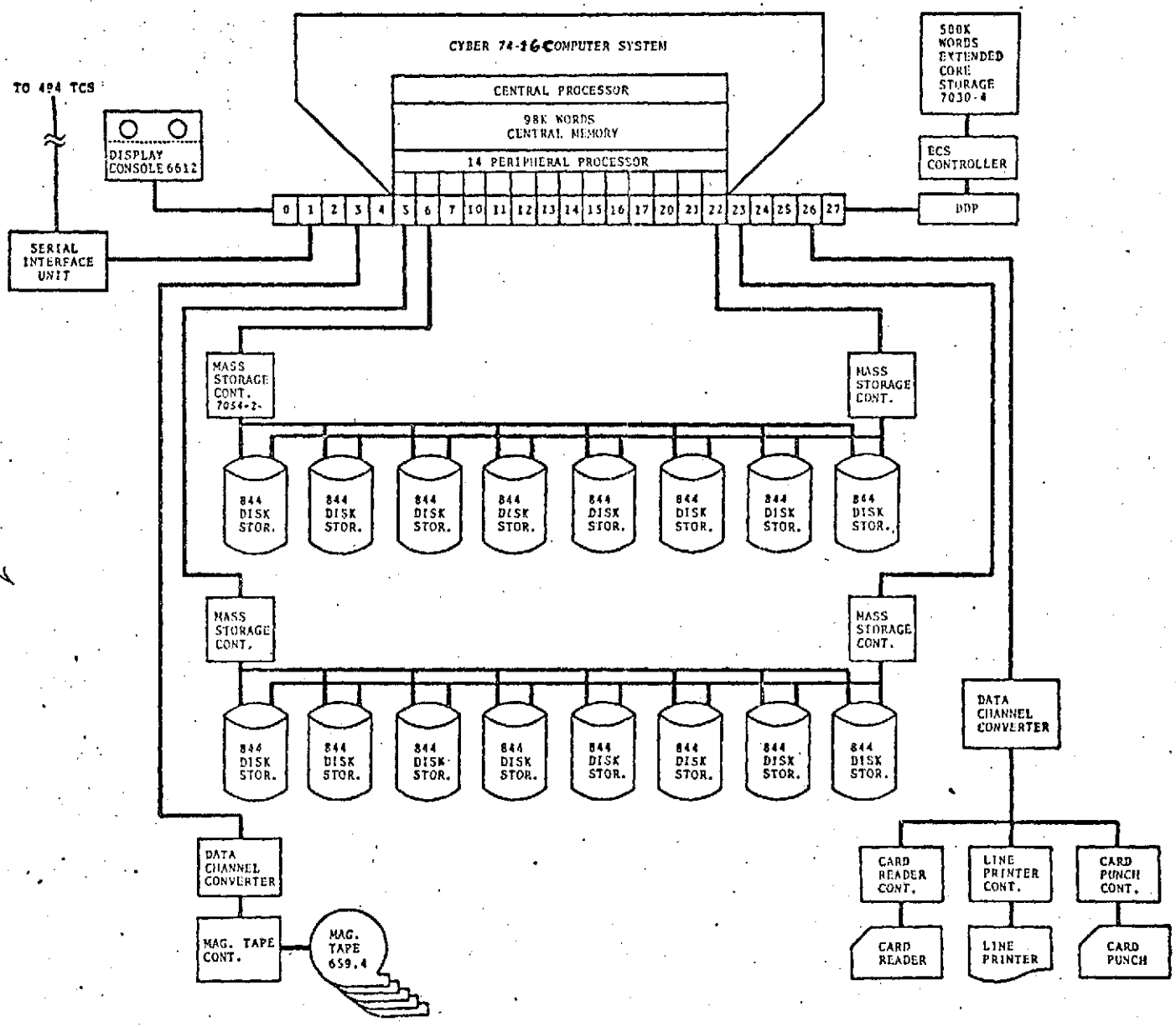


Figure 1-1 Cont.
SPIMS Hardware Configuration

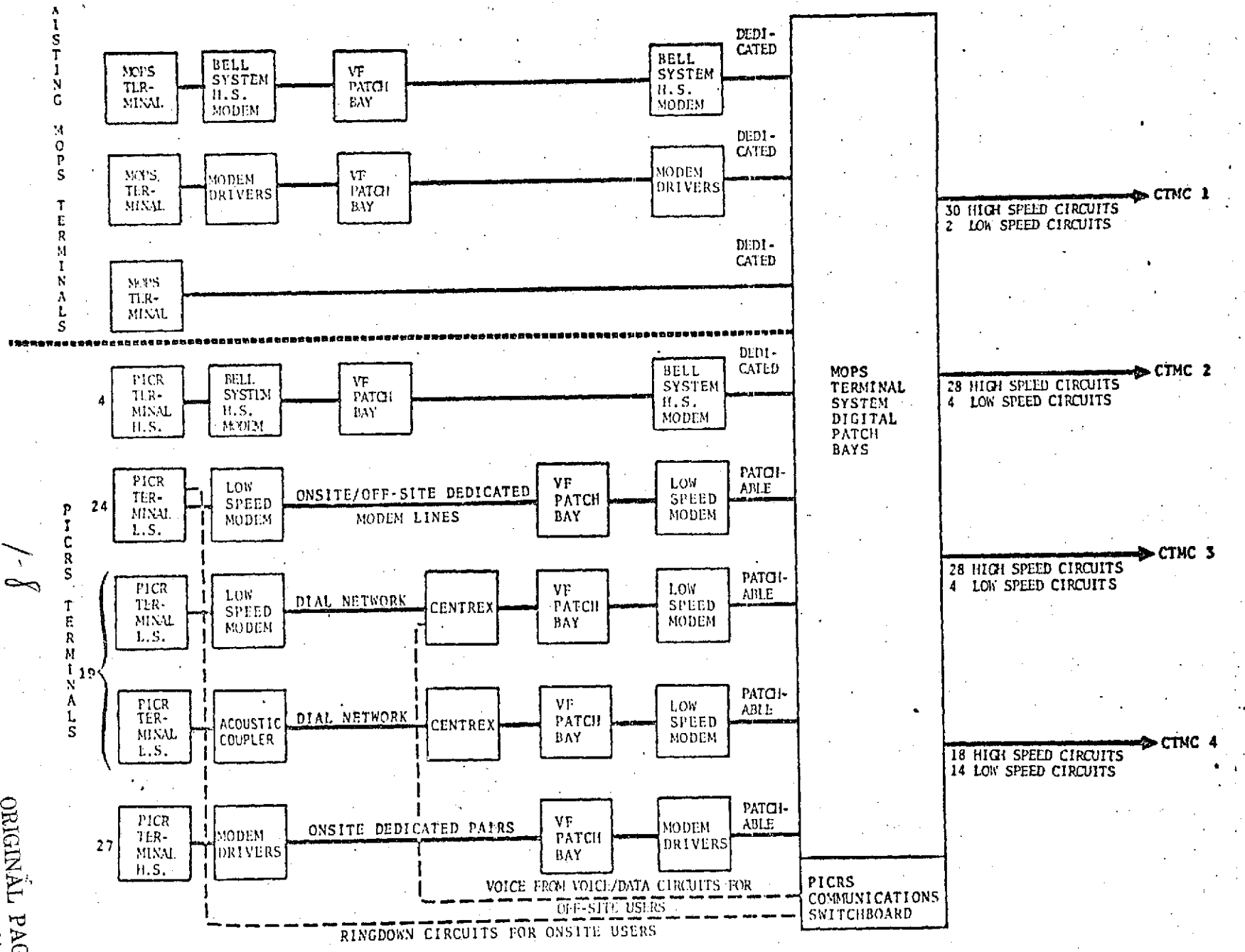
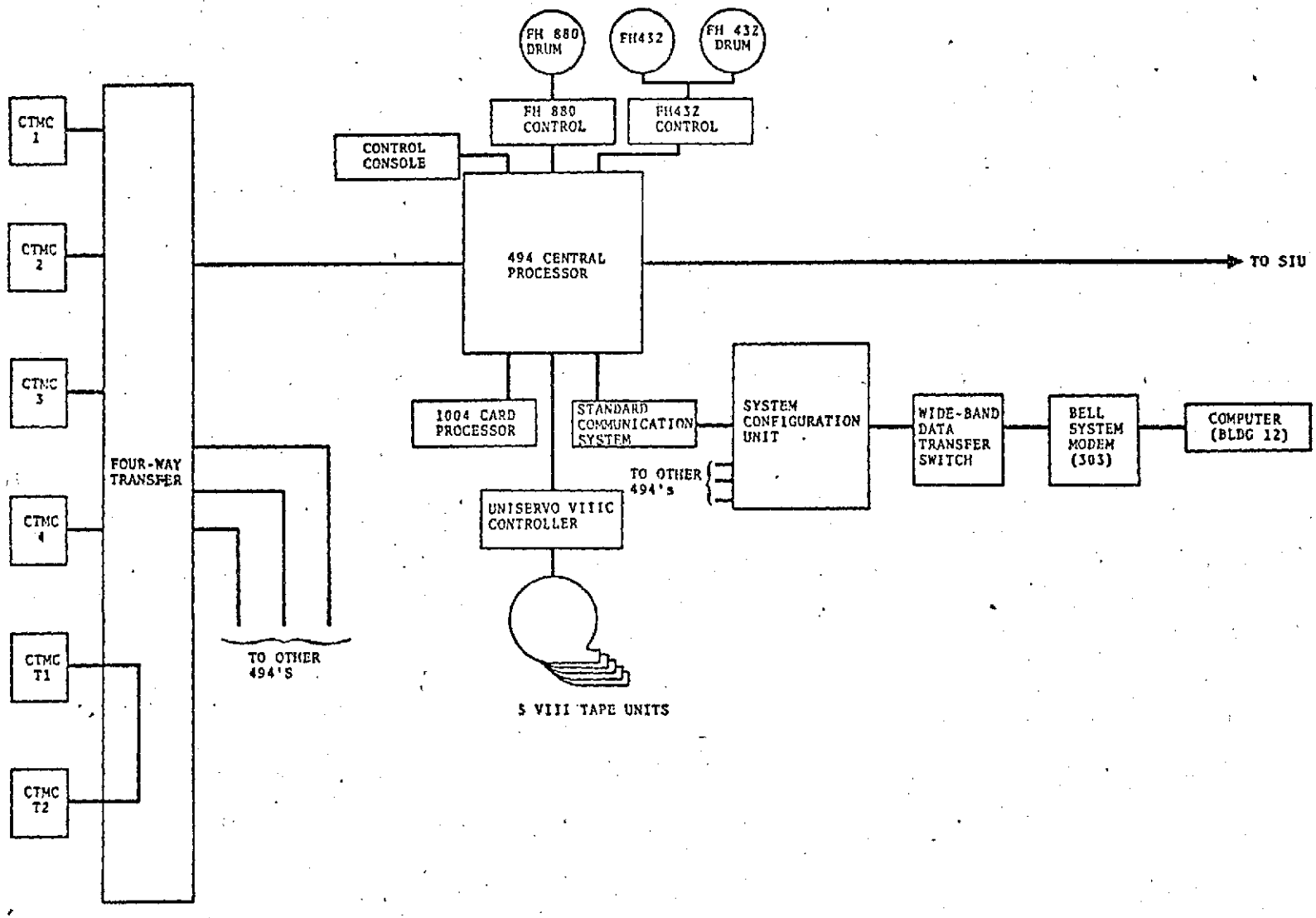


Figure 1-1
SPIMS Hardware Configuration



1-9

ORIGINAL PAGE IS
OF POOR QUALITY

Figure 1-1 Cont.
SPIMS Hardware Configuration

than 64 terminals but the communications software in the CYBER 74 is being developed with that limit.)

The TCS message switching computer is linked to the SPIMS application computer, a Control Data Corporation, CYBER 74, by means of an 81.6 Kbs bi-directional, full-duplex, synchronous line. The CYBER 74 has an adapter which converts the TCS transmissions into 12-bit parallel signals that may be transmitted to its Peripheral Processing Unit (PPU). Each of these 12-bit PPU bytes will contain 1-1/2 8-bit TCS characters. Each character consists of 7 data bits and a parity bit.

The CYBER 74 is the host computer for most of the SPIMS applications. It is a distributive processing machine in which user programs perform their processing in mainframe memory. This particular CYBER 74 is a unit-processor, i.e., it is configured with only one Central Processing Unit. However, it contains 14 Peripheral Processing Units (PPU's) which perform a myriad of functions such as disc and tape access, communications interface and unit record device interface. Each of the PPU's is in essence a mini-computer slaved to the user's mainframe program, performing such peripheral service functions as it might be called upon to do.

The peripheral equipment on the CYBER 74 includes:

- Card reader/punch
- Line printer
- Mass storage - 16 disc units (110 million characters each)

- Extended core storage (503,000 words)
- Magnetic tape drives (four 9-track, two 7-track)

A complete equipment list is contained in section 1.5.2 of this document.

For additional detail regarding the Terminal Control System, References 2 and 3 should be consulted. Reference 4 provides a more detailed description of the CDC CYBER 74 computer system.

1.5.2 CYBER 74 Equipment List

The following list identifies the hardware components designated for SPIMS use of the CYBER 74. Starred (*) items are currently leased. Purchase of these may occur at some future date.

Description	Model	Quantity
Central Processor	74-16	1
Memory Option	10265-3	1
PPU and I/O Extension	10269-1	1
ECS (503K)	7030-4	1
Data Channel Converter	6681	1
Line Printer Controller	3555-1	1
Card Reader Controller	3447-1	1
Magnetic Tape Controller	3528-3	1*
Line Printer	512-1	1
Card Reader	405	1
Magnetic Tape Unit - 9 Track	659-4	4*
Magnetic Tape Unit - 7 Track	657-4	2*
Disk Controller	7054-2	4

Disk Unit	844-2	16
Card Punch Controller	3446	1
Card Punch	415	1
Interface Control Unit	10276	5
Print Train Cartridge	595=1	1*

1.5.3 Software Configuration

The software configuration of concern is the CYBER 74 resident software. The Terminal Control System computer operates under the COSMOS operating system and is mentioned here only for completeness. (See Reference 5 for a description of COSMOS.) Figure 1-2 is a diagram of the software configuration for SPIMS.

1.5.3.1 Operating System. The CYBER 74 operating system to be used for SPIMS is KRONOS. KRONOS is a product of the Control Data Corporation. It is a time-sharing system and is oriented toward interactive terminal operations. However, it also supports local and remote batch processing concurrent with terminal users. It provides facilities for high-level programming language programs to be created as well as an assembly language. Languages include FORTRAN, COBOL, BASIC, ALGOL, and COMPASS (assembly). (Not all of these are currently installed at the JSC.)

1.5.3.2 Communications System. Control Data Corporation is developing a communications processor for the CYBER 74/TCS interface. This processor consists of a PPU program called 1CE and a mainframe program called the Interface Control Executive (ICE). ICE and 1CE working in

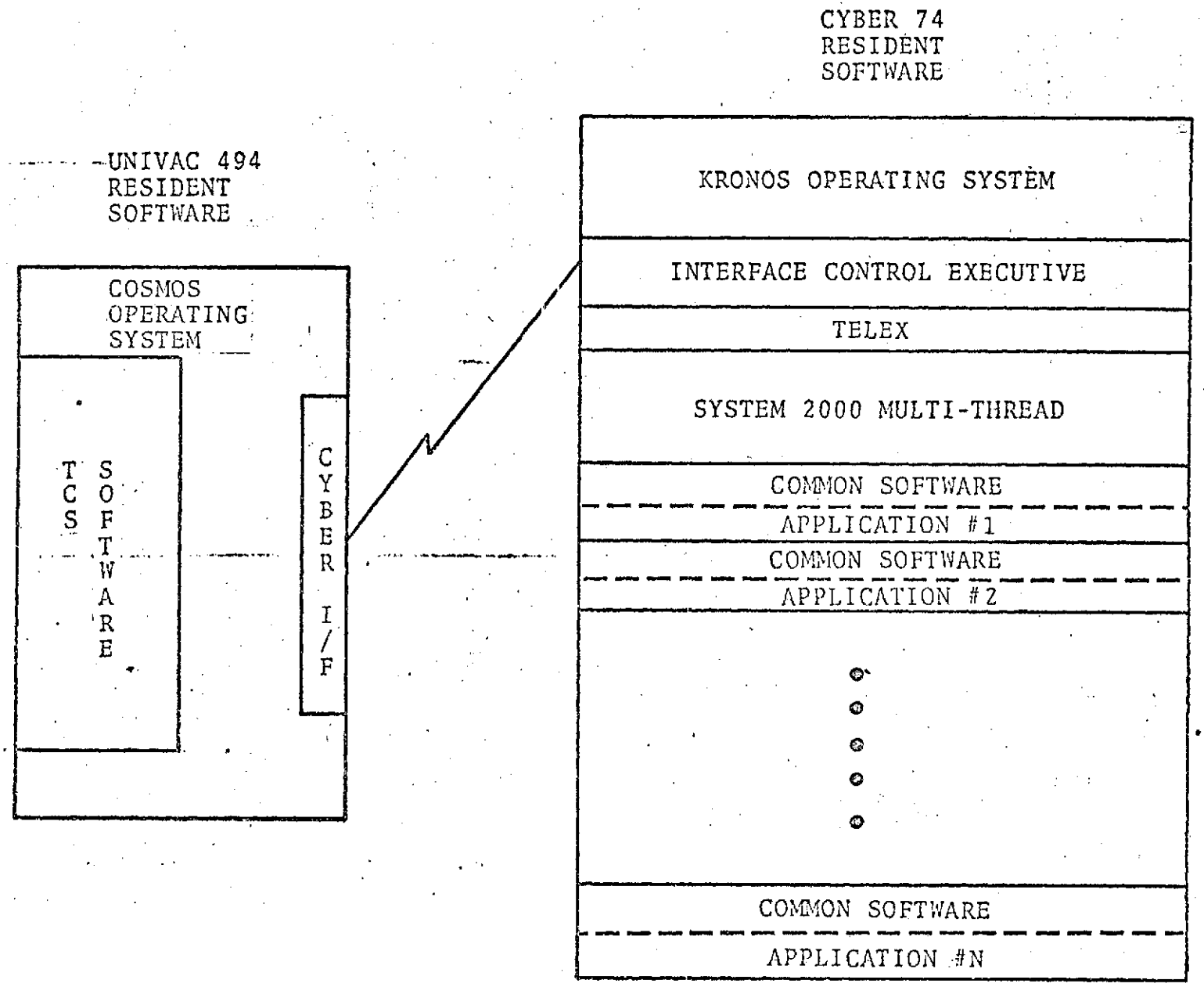


Figure 1-2. — SPIMS software configuration.

concert will support all of the Terminal Control System protocol required to accomplish the two-way communications between terminals and the CYBER 74. 1CE is responsible for interfacing with the CYBER Adaptor in order to read and write TCS transmissions over the 81.6 Kbs line. ICE is responsible for accepting data from (or sending data to) 1CE. ICE must pass input data to the requesting application and obtain output data from an application. ICE also has responsibilities in the area of message logging. ICE must interface with the data base management system and those applications running under control of the DBMS and it must interface with normal time-sharing runs. The latter may be KRONOS products such as compilers, or a user program that does not operate under the control of the DBMS.

1.5.3.3 Data Base Management System. System 2000, a product of MRI Systems Corporation, will function as the Data Base Management System for SPIMS. System 2000 was recommended as the DBMS in the feasibility study final report. A development effort is in progress by MRI Systems Corporation to provide extensions to the functional capabilities of the system. Also, a multi-thread version, a single copy of which will support concurrent updates on a data base, of System 2000 is scheduled to be delivered to the JSC CYBER 74 facility.

System 2000 will be responsible for all data base activity for SPIMS. It provides a self-contained Natural Language that permits a terminal or batch user to interact with a data base without the necessity for generating source language programs. It also provides a FORTRAN or COBOL Programming Language Interface which permits a user program

to manipulate the data in a more conventional manner. (Actual data base references are made via calls to System 2000 entry points which permit it to exercise the required controls on the data base.)

Reference 6 provides a detailed description of System 2000 and its functional capabilities.

1.5.3.4 Applications Software. Applications may be configured in several different ways:

- System 2000 Natural Language with no user code invoked
- System 2000 Programming Language Interface with user code
- System 2000 Natural Language and/or PLI with user code and Common Software

Note that a requested extension to System 2000 is an interface between a user program and the Natural Language capability of System 2000. It is currently anticipated that no application will operate in a pure Natural Language mode although that option remains open should it be required by some future application.

An application may have need for multiple users to interact with the same data base concurrently. Two considerations supporting this likelihood are: 1) an extension to System 2000 has been identified (called the multi-user capability); 2) multiple copies of an application may be resident and processing in the CYBER 74 concurrently. System 2000 has the responsibility to resolve any data base

deadlocks that might result from multi-user access (in particular update access) to a data base. Each application can function as though it were the only user of that data base.

1.5.3.5 Common Software. Common software is the name applied to application software used by more than one application system. For example, all of the SPIMS applications must have the ability to read and write a user terminal. Since the nature of the terminals to be used by SPIMS requires the introduction of a variety of control sequences into a data transmission, a simple FORTRAN or COBOL read function is not sufficient to interpret the data stream. Hence, it falls to the common software to perform the interface to terminal input and output. In this manner, the software to perform a given function can be generated once, tested, and then integrated into the various applications requiring the capability.

Just as use of a data base management system reduces the development costs of the applications, common software reduces costs. Instead of designing, coding, and testing a piece of software several times for several applications, those functions are performed only once and the cost is amortized over the several applications.

Common software requirements are presented in detail in sections 2.0 and 3.0 of this document.

2.0 FUNCTIONAL REQUIREMENTS

2.1 REQUIREMENT SOURCES

The SPIMS applications PDS (see reference 13), SLAHTS (see reference 12), CMA (see reference 15), and SIS (see reference 14) were the principal sources for the derivation of common software requirements. An analysis of the applications led to the identification of certain common functions. A system was synthesized to identify the components of the different applications which would be common.

This system included the Terminal Control System, the CYBER using KRONOS as the operating system and System 2000 as the DMS, a set of common software, and the SPIMS applications. The Terminal Control System, the operating system, and the DMS satisfied many common application requirements. Common software was identified to satisfy common requirements not handled by other systems software.

This section defines the common functional requirements which were deemed to be more economically satisfied with the development of common software rather than by independent development of each application.

Common software not being an operational system, has no operational requirements itself; but it must be constructed in such a manner that the application's operational requirements can be satisfied. Section 5.3 discusses operational requirements considerations.

2.2 COMMON SOFTWARE FUNCTIONAL REQUIREMENTS

The four currently identified SPIMS applications to be run on the CYBER are oriented toward large data bases using System 2000 as a data base manager. All systems applications are required to provide a simple user interface geared to their specific users. The interface selected was that of form mode input.

A form will consist of headings, titles, and other labeling and tutorial data, and of parameter areas into which the user enters those data needed to describe his query or update. Use of the foreground/background intensity capabilities of the Hazeltine 2000 and 4000G terminals will enhance the usability of form mode input. The parametric areas of the form can be displayed in foreground while the tutorial data are displayed in background. Under this configuration, the TAB key on the terminal keyboard can be used to position the cursor to the beginning of the next parametric area on the form, thus eliminating the necessity for manually moving the cursor character by character until the desired position is achieved.

Many of the common software requirements are either indirectly or directly introduced by the need of forms. TELEX/ICE, the KRONOS Communications Software Subsystem, suffices for transmitting one line of data at a time to the computer, but additional software is needed to support full page transmissions and allow for the use of specific capabilities of the MOPS and Hazeltine 2000 terminals which are needed for forms.

Common software requirements have been grouped for discussion in the following categories:

- Communications interface
- Input/Output processing
- Terminal Interface Support
- Programmer support programs for constructing forms
- Utility routines

The following subsections discuss these requirements.

2.2.1 Communications Interface Requirements

The common software communications interface requirements are to:

- simplify I/O for applications, reduce it to a form comparable to FORTRAN or COBOL I/O;
- allow utilization of specific terminal capabilities;
- provide terminal independence for applications.

2.2.1.1 Simplification of I/O. In order to simplify I/O for applications, common software must make transparent the following:

- TCS conventions - TCS blocks all data into 360 character or less blocks which contain control data. This interface is described in detail in references 2 and 3.
- TELEX/ICE System 2000 interface conventions - Terminal data communications between ICE and applications, when using System 2000, are via an

interface file. The Normal INPUT and OUTPUT files are used to communicate information with ICE/TELEX concerning when and how much data is being transmitted. The details of TELEX/ICE interface requirements are described in the SPIMS/CYBER Interface Control Document (reference 16), to be published.

- Terminal control information - Both the MOPS and Hazeltine 2000 terminals have control characters to indicate the beginning and ending of information, headers, foreground indicators, and background indicators as well as normal text data which is in a modified ASCII character set. In addition, the MOPS terminal distinguishes between a normal and selective transmit by control indicator. Position data exists to indicate locations for character sequences. Function keys may also be transmitted from MOPS terminals.

Common software must process the control information on both input and output, making them transparent to applications, except as noted in the remainder of this section. References 10 and 11 describe the detailed characteristics of the MOPS and Hazeltine 2000 terminals.

2.2.1.2 Utilization of Terminal Capabilities. In order to allow utilization of specific MOPS and Hazeltine 2000 terminals capabilities, common software is required to provide for an application to distinguish whether data is foreground or background. It also is required to provide for transmitting selected data to and from a terminal.

Also, common software must identify function key data on a MOPS terminal.

2.2.1.3 Terminal Independence. Some applications require the ability to interface with more than one type of terminal. Common software will be required to support terminal independence by obtaining the type of terminal from ICE and processing control information in the form necessary for that terminal. In addition, a blocking convention will be defined for multiple page displays.

This blocking convention will be defined to allow optimal breaking of display data depending on the size of the terminal. Applications will identify to common software the specific lines in a block and the organization of blocks on specific pages.

Current requirements specify only the use of the MOPS and Hazeltine 2000 terminals. But, common software must be be constructed to incorporate other types of terminals with minimal effort at a later date.

2.2.2 Input/Output Processing

This section identifies specific input and output processing requirements which are needed to enhance the data management capabilities of System 2000.

2.2.2.1 I/O Translation. This is the requirement to provide the capability to replace on input and/or output a corresponding table value for the specified input or output value.

2.2.2.2 I/O Conversion. This requirement is similar to the translation process except that the replacement is to be from a specific row in an N dimension table. An example of usage would be to convert from different center terminology to that used at JSC on input and to perform the converse operation on output.

2.2.2.3 Input Validation. System 2000 performs certain types of input validation. Common software is required to perform these validations to provide better control of error processing and provide additional input validation. The requirements are:

- Limits validation - The determination if value is between a pair of specified values.
- Table validation - The determination if a value is in a specific table.
- Date validation - The determination if a value is a legal date.
- Integer validation - The determination if a value is an integer.
- Alphabetic validation - The determination if a value contains only alphabetic or blank characters.
- Real/Exponential validation - The determination if the value is a legal floating point arithmetic value.

2.2.2.4 Parameter Editing. FORTRAN performs input and output parameter editing via format statements in conjunction with Read and Write statements. FORTRAN also has Encode and Decode statements which function like core to core Read and Write statements. The parameter editing

requirements could be satisfied with the Encode and Decode statement except their use requires considerable core overhead.

In order to eliminate this core overhead, it is required that parameter edit routines be developed; they will perform the following conversions for parameters:

- Display Code to real
- Display Code to integer
- Display Code in scientific notation to real
- Real to Display Code
- Integer to Display Code
- Real to Display Code via scientific notation

2.2.2.5 Limited Text Editing of Input Data. The ability to perform a specific text editing operation on input data is required. This operation requires elimination of more than one contiguous blank except at the end of lines. Sufficient blanks would be inserted at the end of the line to cause the next line to start with a complete word rather than splitting a word between lines.

2.2.3 Terminal Interface Support Routines

The SPIMS applications share requirements for several functional capabilities in the area of terminal interface. Common software will provide support routines to assist in the fulfillment of these requirements. The functional areas are listed below and are followed by a brief discussion of each:

- Preparation for form processing
- Page storage and retrieval
- Block storage and retrieval
- Mapping of external and internal formats to one another
- Advisory message handling
- Command interpretation

When a user references a given form for the first time, the application may not be prepared to process the data contained on the form. In particular, the various tables describing the form and its contents are not likely to be loaded into core. The Initialize Form routine will permit the application to load those data required to recognize and properly process the form.

Terminal users will be page-oriented when using a SPIMS application. Their operations will in general be restricted to a single displayed page on the terminal. The applications, however, will orient their processing to a subset of a display page (that subset known as a block). Display blocks have been identified to simplify the requirement for interacting with two types of terminals (Hazeltine 2000 and 4000G). A page displayed at the user terminal might consist of several blocks; for example, a heading block, followed by one or more user parameter blocks. When considering input from the terminal, the user transmission might include all or parts of several blocks, although the user thinks in terms of transmitting a "page" to the application. Support routines will exist to aid the application in manipulating terminal data in its two formats. Routines to store and retrieve blocks and pages

will be provided. In addition, the capability to combine blocks together to form a page will be provided.

Parametric data associated with a form will normally be transmitted to the application in the form of a string of characters. In general, this string will not be directly usable by a PLI application program because parameters sent to System 2000 PLI must be properly oriented on word boundaries. Conversely, data to be displayed to the user terminal needs to be restructured as a character string. Routines will be provided as part of common software to facilitate these mapping processes.

Support routines to aid in the construction and display of advisory messages will be included in the common software library. One such routine would cause an advisory to be displayed from a table of messages. It would also permit object-time parameters to be inserted into an advisory prior to display. Another routine would provide an interface to permit a message formatted independent of the advisory tables to be displayed. (Such a message would already conform to all of the communications standards and conventions.)

Finally, common software support in the area of command interpretation will be provided.

2.2.4 Support Programs for Generating Forms

In order to facilitate development of applications, a program is required to be developed to store new blank forms (a form with tutorial data but no parameter data) on

permanent file for later access. This program will fill out certain table data to describe where parameters exist on the form. This table data is used by common software in the processing of the form.

All tables used by common software may be maintained by the KRONOS Text Editor.

2.2.5 Common Software Utility Routines

All common software are required to be developed in a modular method. All levels of routines will be made available to applications programs for their use. Lower level routines will include a string package for manipulating character strings, a table handling package for manipulating data tables, a character set conversion routine, file I/O routines, and error display routines. The availability of these modules will facilitate application development. The need for file I/O routines arises because of the overhead resulting from FORTRAN and/or COBOL I/O.

3.0 DESIGN REQUIREMENTS

This section identifies specific, required common software subroutines. The identification of these subroutines was performed in conjunction with SPIMS applications representatives from both IDSD and LEC.

The intent of the subroutine definitions is to allow application development to proceed concurrently with the development of common software.

The routines in this section are grouped into the following logical groups:

- Communications Interface Routines
- Terminal Interface Support Routines
- Input/Output Processing Routines
- Utility Routines

All subroutines will be required to return error indicator. Appendix A contains a description of the "tables" referenced in this section.

3.1 COMMUNICATIONS INTERFACE ROUTINES

There will be two sets of communications interface routines. These routines perform the following functions:

- Receive data from a terminal.
- Transmit data to a terminal.

All terminal I/O will conform to the interface conventions described in reference 16.

3.1.1 Input Processor

The Input Processor will cause transfer of an entire terminal transmission into the applications buffer. It is analogous to a FORTRAN binary read.

Inputs

Terminal input:
(1) Communications data from TELEX/ICE.
(2) Terminal data from ICE.

Program input:
(1) Form Description Table (description of foreground and background usage).
(2) Terminal Description Table (type of terminal, length of command line, length of display area).
(3) Command Interpretation Table.

Processing. In order to input terminal data, a read request will be sent to TELEX. When a response is received from TELEX, data will be transferred from a mass storage file. Then this routine will process the input data according to the MOPS or Hazeltine 2000 terminal conventions depending on the type of terminal in use. TCS, Terminal, and TELEX/ICE control information will be eliminated. The terminal characters will be converted to CYBER 74 Display Code and stored in a specified buffer.

Data will be placed in the command buffer or the data buffer corresponding to the location of the information on the terminal screen. The locations of the first and last display characters transmitted are stored. A function key command transmitted from a MOPS terminal is placed in a function key buffer. If a command has been transmitted, command interpretation will take place using the Command Interpretation Table. An advisory will be transmitted to the terminal for invalid commands. Control then will be passed to the application.

Outputs

(1) Character matrix corresponding to the display terminal.
(2) Begin and End Transmit locations.
(3) Command/Data Indicator
(4) Command Type.
(5) Command line buffer.
(6) Invalid command Advisory (to terminal).
(7) Error Status Indicator.

3.1.2 Output Processor

The Output Processor moves data to a display terminal in foreground or background mode as specified.

Inputs.

(1) Character matrix corresponding to display terminal.
(2) Form Description Table (description of foreground and background usage).

(3) Form Description Override table.

(4) Begin and end location of characters to be transferred from character matrix.

(5) Transfer location to terminal.

(6) Terminal Description Table (Type of terminal, command line length, display area length, advisory area length).

(7) Type of Transfer (foreground only, background only, both foreground and background, or no character conversion).

Processing. Information to be transferred is selected from the data buffer and converted to the terminal display code. Control information conforming to terminal and TCS conventions is supplied. Data is transferred to the terminal according to TELEX/ICE and System 2000 interface conventions. The program will use either MOPS or Hazeltine 2000 terminal conventions depending on the type of receiving terminal.

During the transfer process characters are converted from 6 to 8 bits code, skipped, or moved directly as 8 bit characters depending on the type of transfer code and the description of foreground information which is contained in the Form Description Table. The items indicated in the Form Description Override Table are treated as if they were indicated as background data in the Form Description Table.

Data is sent to ICE by first writing the data to a mass storage file, then writing a message to the Output File (to TELEX) indicating to ICE that the data file is ready.

Output.

(1) Terminal display characters to terminal.

(2) Error status indicator.

3.2 TERMINAL INTERFACE SUPPORT ROUTINES

This section describes those routines currently defined to support terminal input and output. Availability of these routines allow for most application code to be developed independent of the terminal type. For example, application code may be block oriented despite that the blocking of a terminal is terminal type dependent. The first six of these routines facilitate translating from blocks to pages and pages to blocks.

3.2.1 Page

This function displays a specified page on the terminal. It may be used for responding to an application pageing command and/or displaying the results of a query type command.

Input. Paging instruction, number forward, number backward or specific page number, Block Description Table.

Processing. The page number is determined and the specific blocks comprising the page are retrieved using the Retrieve Page routine, then it is displayed using the Output Processor.

Output. Specific page to display area on the terminal.

3.2.2 Retrieve Page

This function retrieves the blocks for a specified page. This routine is used by the Page routine.

Input. Page Identification, Block Description Table (table describing blocking for page, blocks, and address of blocks for form), and buffer to hold page.

Processing. This program determines the blocks used for a specified page and retrieves them using the GET Block Routine.

Output. Page data in buffer.

3.2.3 Store Page

This function stores the blocks comprising a page of data on mass storage.

Inputs. Block Description Table (table describing blocking for page, describing blocks, and identifying address for storing blocks on mass storage), buffer containing page of data.

Processing. This program determines the data for each specific block of the page; then writes the block on mass storage using the Put Block routine.

Output. Specific page blocks to mass storage.

3.2.4 GET Block

This function retrieves a specified block. This routine will be used by the retrieve page and application routines.

Input. Block ID, Block Description Table, location for block.

Processing. The address of the specified block is determined, the block is read into core.

Output. Contents of block.

3.2.5 STORE Block

This function stores a specified block on mass storage. This routine provides for applications to store blocks of data which may later be retrieved by the retrieve page routine or application routines.

Input. Block identification, Block Description Table, block of data.

Processing. If a mass storage location exists for the specified block, the block is stored at that location. Otherwise, a block mass storage address is determined, the block is stored at that address, and the Block Description Table is updated to reflect the address.

Output. Content of block is stored on mass storage.

3.2.6 Initialize Form

This function establishes the specified form as the current active form and displays the first page of the blank form.

Input. Form identification from application, table containing mass storage address for forms and Form Description Table.

Processing. The specified form is copied from permanent files to temporary files using the Initialize Form Function in order that the form may be updated. The first page of the form is displayed using the Page routine. Also, the Form Description and Block Description Tables are retrieved.

Output. Display of first page of form, Block Description Tables, and Form Description Table.

3.2.7 Initialize Form Function

This function copies a blank form and makes it available for processing. This routine is a subroutine for Initialize Form.

Input. Form identification, table containing mass storage address for forms and Form Description Table.

Processing. The specified form is copied from permanent storage to temporary storage. The Form Description tables are retrieved. The Block Description tables are updated to reflect the block addresses.

Output. Form Description Table, Block Description Table.

3.2.8 Convert to Word-Boundary Format

It is the purpose of this routine to facilitate the transformation of an input character string from the display matrix format to a word-boundary format. It can be called to convert only one or a series of contiguous parameter fields during an entry into the routine.

Inputs.

1. Input matrix containing terminal display character string.
2. Item number of first parameter to be converted.
3. Item number of last parameter to be converted.
4. Address (core) of first word in the word-boundary list to be generated.
5. Form Description Table for the form.
6. Block Description Table.

Processing. The Form Description Table is used to extract the location (in the display matrix) and length of the first and each succeeding parameter (through the last item specified in (3)). Each parameter in turn is extracted from the display matrix and placed in the next available word in the word-boundary list. The transfer is treated as a character transfer with left justification resulting in the output list. Cognizance is maintained of the position of the next available word in the output list and termination of a parameter field causes a pointer into that list to be incremented unless the transfer ended at the

right hand boundary of a word. (In that case, the pointer would already have been incremented.)

Outputs.

1. Word-boundary list with the converted input parameters stored.

3.2.9 Convert to Matrix Format

This routine is required to move character strings from a list into the proper parametric display areas of a form.

Inputs.

1. Core address of a list of character strings to be converted.
2. Core address of the display matrix to receive converted strings.
3. Item number of first parameter to receive a string.
4. Item number of last parameter to receive a string.
5. Form Description Table for the form.

Processing. The length of each parameter along with its location in the destination form is extracted from the Form Description Table for the form. The length of the parameter field is used initially to determine how many characters (and by implication how many words) in the input list are to be used while moving that parameter. Each change of parameter field will cause the routine to proceed to the next word-boundary in the list. Any excess of character space in the preceding word will be ignored. All

succeeding parameters will be handled in an identical fashion until the last parameter (specified in (4) above) has been converted into the matrix format.

Outputs.

1. Display matrix with the requested parameter fields entered.

3.2.10 Advisory Message Handler

The Advisory Message Handler routine displays the indicated error message.

Input. Error number, Advisory Message Table.

Processing. This routine retrieves the indicated error message from the Advisory Message Table, then using the output processor sends it to the advisory area of the user's terminal.

Output. Advisory message to the terminal.

3.2.11 Special Message Routine

This routine provides for "bit strings" to be transmitted to the user's terminal. Its purpose is to allow special cursor positioning and ringing the terminal bell.

Inputs.

1. Core address of the "bit string".
2. Length of the "bit string".

Processing. Control information for TCS, TELEX/ICE, and the terminal (message header,STX, and ETX) are generated for the "bit string" via a call to the Output Processor, which also routes the message on to the desired terminal.

Output. Transmission of the data to the terminal.

3.3 INPUT/OUTPUT PROCESSING ROUTINES

This section defines subroutines which perform input and output processing functions on the Display Area.

3.3.1 Translation

The Translation Routine performs translation of input and output parameters using a table.

Input. Value, translation table, flag indicating whether input or output translation is to be performed.

Processing. The routine performs a table look-up on the translation table using the input value, translates the input value, if it is in the table, and sets the error indicator if the input value is not found.

Output. Translated value, error indicator.

3.3.2 Conversion

The Conversion Routine performs input and output conversion for parametric values using a two dimensional conversion table.

Input. Value, column indices for conversion table.

Processing. The routine performs a table "look-up" on the conversion table for the given parameter using the column indices specified. The error indicator is set if the input value is not found.

Output. Converted value, error indicator.

3.3.3 Input Validation Routines

The following set of validation routines will be developed.

3.3.3.1 Table Validation. The Table Validation Routine determines if the input value matches any of the values in the table.

Input. Value, Validation Table.

Processing. The routine matches the input value against the values in the table and sets an indicator showing the results.

Output. Success/failure indicator.

3.3.3.2 Limits Validation. The Limits Validation Routine determines if the input value is within the limits.

Input. Value, Limits.

Processing. The routine determines if the input value is equal to or greater than the lower limit and equal to or less than the upper limit.

Output. Success/failure indicator.

3.3.3.3 S2K Table Look Up. The S2K Table Look Up Routine determines if an input value is defined in an S2K table.

Input. Value, S2K File.

Processing. The routine determines the input value is defined in the S2K file and sets a success/ failure indicator.

Output. Success/failure indicator.

3.3.3.4 Integer Validation. The Integer Validation Routine examines a string of characters to determine if they represent an integer.

Input. Character array.

Processing. The routine checks the array for numeric characters; any alphabetic character or special character except a leading minus sign is considered an error.

Output. Success/failure indicator.

3.3.3.5 Exponential Validation. The Exponential Validation Routine examines an array of characters to determine if they represent a valid exponential expression.

Input. Character array.

Processing. The routine checks the array for a valid exponential expression as defined by ASCII FORTRAN; any other combination of characters is considered an error.

Output. Success/failure indicator.

3.3.3.6 Real Number Validation. The Real Number Validation Routine examines an array of characters to determine if they represent a valid real number expression.

Input. Character array.

Processing. The routine checks the array for a valid real number expression as defined by ASCII FORTRAN; any other combination of characters is considered an error.

Output. Success/failure indicator.

3.3.3.7 Alphabetic Validation. The alphabetic validation routine examines an array of characters to determine if they are all alphabetic characters.

Input. Character array.

Processing. The routine checks each character in the array; any non-alphabetic character is considered an error.

Output. Success/failure indicator.

3.3.3.8 Date Validation. The Date Validation Routine examines an array of characters to determine if they form a valid date.

Input. Character array.

Processing. The routine checks the characters of the array for a one or two digit month not greater than twelve, a one or two digit day of maximum magnitude determined by the month and a two or four digit year. Any other combination of characters is considered an error.

Output. Success/failure indicator.

3.3.4 Table Processing Routines

The following routines will be available for initializing ECS tables and retrieving tables from ECS. The ECS tables will include such tables as validation tables which are maintained as part of the Applications System 2000 data base.

3.3.4.1 Table Retrieval. The Table Retrieval Routine moves a specified table from ECS files to core.

Input. File name, Table name, core address.

Processing. The routine locates the table in the file and moves it to core.

Output. Table.

3.3.4.2 Data Base Load. The Data Base Load Routine moves tables from S2K Files and permanent mass storage files to temporary files (on ECS) to be accessed by the applications.

Input. S2K Files and permanent file.

Processing. The routine moves the tables in the S2K files and permanent files to ECS files. A table, used for locating the particular tables, is constructed in the process. Note: Application must supply subroutines or S2K commands for retrieving the application unique tables.

Output. Table location of the various tables, all files on temporary files (on ECS).

3.3.5 Input/Output Editing

This section defines functions for editing parameter data. One of the functions performs limited text editing on input. The other two capabilities are similar to those of the FORTRAN Format statements for input and output.

3.3.5.1 Limited Editing Function. This function performs a text edit on the specified text string.

Input. Text string, display line length.

Processing. The text string is processed one character at a time from the first character to the last. Multiple

contiguous blanks are eliminated except when the last word on a display line would be split between two lines. When a word would be split between two lines during processing, enough blanks are inserted to cause the entire word to be moved to the next line. A word is defined here as any string of non-blank characters preceded and followed by blank characters. (Note that the end character on a line is defined to be the last character of a word if it is non-blank.)

3.3.5.2 Input Editing Routine. This routine provides editing capabilities similar to those of a FORTRAN formatted read statement.

Input. (1) character data to be edited, (2) location for edited data, (3) type of edit (no editing, convert to integer form, convert to real or exponential form).

Processing. The specified conversion is performed. If an error is found in conversion, the error flag is set. The no editing function may be used to place data on word boundaries.

Output. (1) converted data item, (2) error status word.

3.3.5.3 Output Editing Routine. This routine provides editing capabilities similar to a FORTRAN formatted write statement.

Input. (1) data item to be edited, (2) type of edit (no edit, convert from integer, convert from real), (3) location for resulting characters.

Processing. The specified conversion is performed. If an error occurs, a blank is substituted and an error flag is set.

Output. Edited data characters.

3.3.6 Construct String Arguments

This function constructs arguments for a System 2000 string reference.

Input. Argument value, System 2000 parameter number, System 2000 String.

Processing. The System 2000 string is searched for the location where the argument is to be inserted. The location in the string is identified by a parameter identifier. The parameter identifier is replaced by the input argument value.

Output. Updated System 2000 string.

3.4 UTILITY PROCESSING

A requirement of common software is to provide the applications developers with a package of subroutines that perform utility functions. The components of this utility package are described below.

3.4.1 String Manipulation Functions.

A large portion of the data to be processed by the SPIMS applications and common software can be represented by character strings. A family of string handling routines will be provided to accomplish the following:

- Locate the first occurrence of a specified string within another string.
- Insert one string into another string.
- Delete one string from within another string.
- Replace one string with another string.
- Convert a string from one character set to another character set.
- Slide a subset of a string to the right or left.
- Extract one or more characters from a string.

Each of these functions must be capable of operating on characters of varying size with the size to be specified by the routine requesting the function.

3.4.2 File Input/Output

It is expected that normal FORTRAN input and output commands will prove to be too expensive in core utilization to be used in the SPIMS applications. (Such statements call a large Control Data Product, the Record Manager, into play.) Instead, file input/output operations will be performed in the COMPASS assembly language using the KRONOS RA+1 mechanism. A utility routine will be provided to the applications to accomplish such I/O needs as:

- Sequential reads and writes
- Random reads and writes
- SUBMITting of batch runs
- SAVE, GET, APPEND, ATTACH, etc. (permanent files)

3.4.3 Forms Construction Utility

An independent common software program will be provided to assist the applications developer in the construction of the forms required for his application. This program will aid in the construction of the tables needed to describe the form and will provide for permanent storage of the form.

The programmer will construct a new form on a MOPS or Hazeltine 2000 terminal. All headings, titles, and other tutorial information are positioned in the desired locations. Parametric fields are delimited (by special characters on Hazeltine 2000, by "selective transmits" on MOPS). The form would then be transmitted to the program in block-sized segments, each of which requires a block-identification. The program will reassemble these blocks into the form, construct the Form Description and Blocking Tables, and provide permanent storage for the blocks and tables. (Note that the terminal dependent aspects, i.e., which blocks are combined to display the form to a particular type of terminal, must be supplied in a separate operation.)

3.4.4 Table Handling Utility

This feature of the common software will provide the application developer with tools to manipulate the various

tables required. These tools may be called directly by the application program. They will provide the capability to search the N x M dimensional matrices that comprise the tables.

3.4.5 Character Set Translation Routine

This routine converts a string of data in a given character set to another specified character set. This requirement will be satisfied with "off the shelf" CDC supplied software if possible.

Input. Translation table, character string, translation type indicator.

Processing. Depending on the type of translation to be performed, each character is translated from display code or from the specified code to display code.

Output. Translated character string.

4.0 TESTING

Individual subroutines will be developed and tested as a unit. When a subroutine has been tested as a unit, it will be made available to applications for development.

4.1 TEST DATA SOURCES

Subroutines will be tested using 1) trivial input (counts of 0, etc.), 2) input known to be in error (negative counts, counts less than specified minimums, counts greater than specified maximum, invalid formats where applicable, etc.) to verify that such errors are detected and 3) input known to be valid. The results will be verified to show that the unit performs as projected.

4.2 GENERAL TEST APPROACH

Drivers will be written to exercise each subroutine and print out the results of the exercise. Memory dumps may be required in some cases.

4.3 ACCEPTANCE CRITERIA

The developed drivers and their test results shall be made available to IDSD for inspection. Successful exercise of subroutines on an individual basis shall constitute criteria for acceptance of the individual subroutine.

A Type 1 test report document will be published for TELEX/ICE and common software interface testing.

5.0 PRODUCTION IMPLEMENTATION

This section identifies considerations for implementing an operational set of common software. First, priorities for individual routines are defined, then the different stages of implementation are discussed, and finally operational requirements are discussed.

5.1 SUBROUTINE PRIORITIES

Routines defined in Section 3 have been grouped in terms of priority for implementation. There are two groups, where group 1 is the highest priority. The groups and the routines comprising the groups are as follows:

PRIORITY GROUP 1 - This priority group includes those routines necessary to perform terminal input and output, and to manipulate ECS tables. The existence of these routines are essential to checkout of any application. They are as follows:

- Input processor
- Output processor
- Table Retrieval Routine
- Data Base Load Program

PRIORITY GROUP 2 - This priority group contains the remainder of common software.

The group priority reflects the importance of a specific routine in terms of early SPIMS application development. The order of implementation should consider the priorities.

5.2 OPERATIONAL STAGES

There will be only one official release of common software. It will contain all of the subroutines identified in Section 3. It is scheduled to be released on September 19, 1975, contingent upon assumptions identified in Section 1.

As unit testing is completed on individual subroutines, they will be made available to applications for developmental purposes. During the period prior to the official release, error correction will be made on an informal basis without the necessity of formal Discrepancy Reports.

5.3 OPERATIONAL REQUIREMENTS

5.3.1 General Operational Aspects

Common software will be used with System 2000 in both batch and online modes to maintain data bases. The frequency of utilization of common software is expected to be very high because all SPIMS CYBER applications are currently expected to use common software. Applications will require tape usage. Because of System 2000 multi-thread limitations for tape usage, they may require staging to disc.

Because applications govern the use of common software, common software has no real operational requirements. Common software must not prevent applications from satisfying their own requirements. The following sections

discuss performance considerations and resource utilization assumptions.

5.3.2 Performance Considerations

The following performance information should be considered during the development of common software:

- Response time - applications will require very rapid response, generally from 5 to 30 seconds. Common Software should be developed in such a manner as to have minimal impact on applications response time.

- Accuracy - common software should provide exact answers, except for conversions involving real numbers where common software is restricted to machine accuracy.

- Reliability - in order to maximize system reliability common software will use disc files to back up ECS files.

- File security - security is a function of the application. System 2000 security features will also be available to the application.

5.3.3 Resource Utilization Assumptions

A crude system modeling effort was performed as part of the PICRS's feasibility study. This effort indicated system utilization as follows:

- Core - very high
- Disk Accesses - high
- Storage (on M.S. devices) - high
- Computation - very low

The order of criticality of the above resources are assumed to be: (1) core, (2) disc accessability, (3) mass storage, (4) CPU computation. This order of criticality should be assumed in design trade-offs. But the system should also be constructed in a manner that optimization based on real data may be easily accomplished.

APPENDIX A

This appendix describes the tables identified during the development of the SPIMS common software requirements.

Terminal_Description_Table. This table identifies the types of terminals which may interface the application defining the table. It describes the standards for dividing the viewing area of the terminal into the Command Area, Display Area, and Advisory Area. At present, two types of terminals have been defined for SPIMS support: Hazeltine 2000, and Hazeltine 4000G (also known as MOPS). The data in these tables are used by common software to determine the type of data included in a transmission from a user.

Command_Interpretation_Table. This table contains a list of all legal commands that may be executed by a user of the application. When command data are received from the terminal, this table is referenced to determine the nature of the command. In addition, MOPS terminals may utilize function keys to identify a command and this table would contain entries to establish the type of command based on the function key value.

Forms. A form represents a pseudo-table for the system; that is, forms are treated as a table for the purposes of storage and retrieval, but they do not contain the normal tabular data. Rather, a form contains the skeleton of a user display. Conventionally, a form will consist of a set of structured tutorial information such as titles, headings, and commentary, and of "fill-in-the-blank" fields for the entry of parametric data. There may be as

many forms defined for the application as required to support all of the application functions. Forms may be defined to be more than one display (page) long. (Note that page size is partially dependent on the type of terminal being used.) Finally, forms are used for output as well as input. The response to a user query may be displayed on a form. That same form might then be modified and retransmitted as an update request.

Form_Description_Table. This table describes the structure of a given form the system to facilitate processing of the form. There must exist a form description table for every form known to the system. Such a table defines the location of parametric data fields on the form.

Blocking_Tables. Because forms may be multi-paged and because at least two different types of terminals must be interfaced, a given form may be divided into small units called blocks. Upon output of the form, these blocks must be combined in a predefined sequence to create a page for display. This table describes the make-up of a form in terms of the size and composition of individual blocks, the order in which the blocks are to be displayed, and the combination of blocks comprising a display on a particular terminal type.

Advisory_Table. This table contains a list of application user advisory messages that might be issued during the course of a run. These are messages that are normally displayed in the Advisory Area of a user's terminal.

Table_Location_Table. This table serves as a master directory for all of the tables known to the system. It will be used to locate a specific table when required so that the table can be retrieved. There will be a single copy of this table for each application.