System Development Corporation

IMAGE DATA PROCESSING SYSTEM
(IDAPS)

SOFTWARE DOCUMENTATION

30 JUNE 1976

TM-HU-039/000/01

System Development Corporation

IMAGE DATA PROCESSING SYSTEM

(IDAPS)

SOFTWARE DOCUMENTATION

S-056 EXPERIMENT

SYSTEM SOFTWARE DESCRIPTION

VOLUME I

INTERDATA-70 SOFTWARE LISTINGS

VOLUME II

IBM-360 SOFTWARE LISTINGS

VOLUME III

CONTRACT NO. NAS8-31667

30 JUNE 1976

TM-HU-039/000/01

FOREWORD

The IDAPS facility is maintained by the Marshall Space Flight Center as a service to S-056 researchers for processing S-056 image data. The IDAPS facility became operational in November 1974. Since that time the facility has been in joint use by the S-056 researchers for data processing and by the system development contractor for continued facility development. As additional facility capabilities are incorporated, the IDAPS Software Document will be updated by the insertion of change pages.

System Development Corporation

IMAGE DATA PROCESSING SYSTEM

(IDAPS)

SOFTWARE DOCUMENTATION

S-056 EXPERIMENT

SYSTEM SOFTWARE DESCRIPTION
VOLUME I

CONTRACT NO. NAS8-31667

30 JUNE 1976

TM-HU-039/000/01

TABLE OF CONTENTS

SOFTWARE DESCRIPTION

SECTION   PAGE

LIST OF FIGURES

1.0 INTRODUCTION

The purpose of the Image Data Processing System (IDAPS) Software Document is
to provide the necessary information for personnel with programming back-
grounds to understand and follow the logic of the software which supports
the IDAPS system hardware configuration. The reader should have some
familiarity with image processing in general and the IDAPS system in particular.
For a description of the IDAPS system, the reader is referred to the pub-
lication Image Data Processing System (IDAPS) User Manual, System Description,
Volume I (TM-HU-037/000/00).

There are two major divisions of software in the IDAPS system; the Inter-
data-70 software and the IBM-360 software. The documentation is divided
into three volumes:

        Volume I   — System Software Description

        Volume II   — Interdata-70 Software Listings

        Volume III — IBM-360 Software Listings

In order to minimize confusion of the two systems, each section of Volume I
treats the two systems separately.

2.0 SYSTEM SOFTWARE DESCRIPTION

This section describes the major components of the Interdata-70 and the
IBM-360 software systems which comprise the IDAPS system. The interactive
relationship between the software of the two systems is illustrated in
Figure 2-1.

2.1 INTERDATA-70 MAJOR COMPONENTS

2.1.1 Real-Time Operating System-RTOS

RTOS is a fairly comprehensive multi-task operating system. A task is the
basic operating unit of RTOS. It may consist of a single program, or it may
include a main program and a number of subroutines. For the most part, the
IDAPS Terminal system has been designed to make use of the features of RTOS.
Those features which are particularly important to IDAPS are as follows:

a) Two or more tasks may be active at a given time.

b) A task may suspend itself for a particular length of time, thus a
    form of time-sharing may be accomplished.

c) There is a centralized and consistent means of accessing peripheral
    devices through the Supervisor Call (SVC) instruction.

d) Disk files may be accessed either sequentially or randomly. Disk
    file allocation may be easily changed as needed.

e) One task may activate another task through the SVC 6 facility, thus
    there is no need for an overall control program.

f) A Task Common area provides access to system-wide information.

RTOS is made up of several individual modules, each of which is a machine-
language program. The RTOS modules are assembled separately and then linked
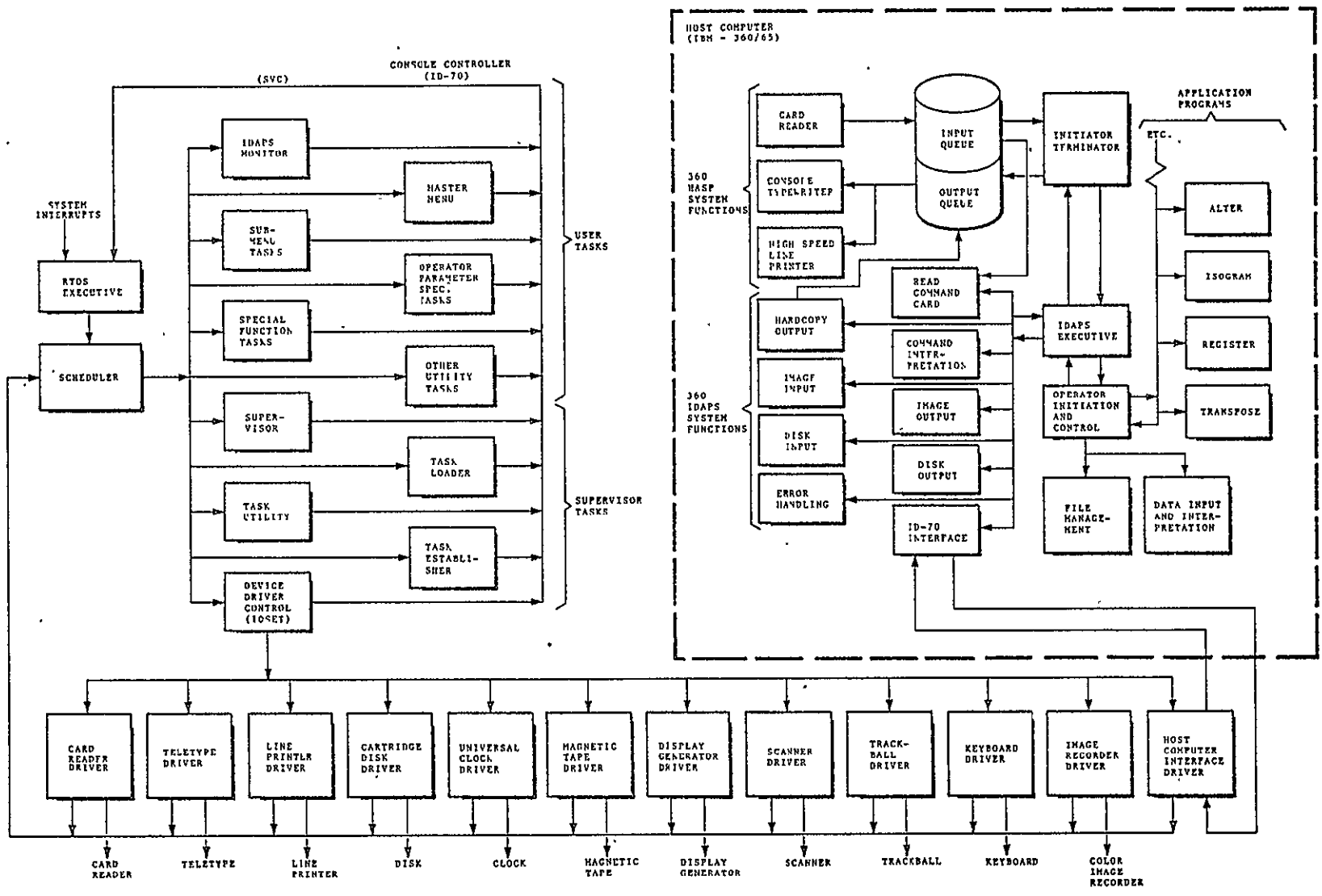
Figure 2-1.   IDAPS System Block Diagram

together in a single core-load module as described in paragraph 2.1.5.3. The
modules are listed here in the order in which they appear in core:

EXEC	–	RTOS Executive Program
RLSTAB	–	Reentrant Subroutine Library Table
LODER	–	RTOS Task Loader
IDAPS	–	IDAPS Monitor Program
TCBTAB	–	Task Control Block Table
DGDVR	–	Anagraph Display Generator Driver
DGDCB	–	Anagraph Display Generator Device Control Block (DCB)
MXDVR	–	Anagraph Keyboard/Trackball Multiplexor Driver
MXDCB	–	Anagraph Keyboard/Trackball Multiplexor DCB
DIDVR	–	DICOMED Scanner/Recorder Driver
SCDCB	–	DICOMED Scanner DCB
REDCB	–	DICOMED Recorder DCB
IFDVR	–	360 Interface Driver
IFDCB	–	360 Interface DCB
LFCDVR	–	Line Frequency Clock Driver
DCB6D	–	Line Frequency Clock DCB
PICDVR	–	Precision Interval Clock Driver
DCB6C	–	Precision Interval Clock DCB
MTDVR	–	Magnetic Tape Driver
TTYDVR	–	Teletype Driver
DCB02	–	Teletype DCB
CRDVR	–	Card Reader Driver
DCB04	–	Card Reader DCB
LPDVR	–	Line Printer Driver
DCB62	–	Line Printer DCB
ROLRTN	–	Roll-out/Roll-in Subroutine
WTDDVR	–	Dummy Watchdog Timer Driver
DSC200	–	Disk Driver

DCBC6L   —   Disk DCB

DCB85   —   Magnetic Tape '85' DCB

DCB85   —   Second copy, Magnetic Tape '95' DCB

SUPV   —   RTOS Supervisor Program

INIT   —   RTOS Initializer Program

The major modules and functional areas of RTOS are described briefly in the
paragraphs which follow, with additional references as needed.

2.1.1.1 EXEC – RTOS Executive

EXEC coordinates all activity within RTOS. It performs the task scheduling
function, that is, it activates tasks according to the established rules of
priority and status. EXEC senses and initiates action on all SVC's, and in
some cases completes the action itself. EXEC contains the code for the
various dedicated memory locations in low core. It is the first module in
core and the only one which has an absolute starting point in core.

EXEC has been modified slightly for IDAPS, as described in Section 4.0.

2.1.1.2 RLSTAB – Reentrant Subroutine Library Table

RLSTAB is designed to contain entry pointers to reentrant subroutines, of
which there are none in IDAPS. However it also contains several critical
parameters related to the system configuration. RLSTAB has been modified for
IDAPS, as described in Section 4.0.

2.1.1.3 LODER – RTOS Task Loader

LODER is a System Task whose task name is *LODER. When there is a request to
load a task into core, LODER finds space for the task and loads it into core
with the necessary relocation adjustments. LODER has been modified for IDAPS
as described in Section 4.0.

2.1.1.4 IDAPS - IDAPS System Monitor

IDAPS is a System Task written especially for the IDAPS system. It performs
certain functions which are unique to this system and which could not easily
be performed by a non-system task. IDAPS is described in Section 3.1.1.

2.1.1.5 TCBTAB - Task Control Block Table

TCBTAB is not an operating program, but rather a set of lists and tables. At
any given time, TCBTAB describes the current state of the system in regards
to which tasks are active, waiting for memory space, waiting for some event,
etc. TCBTAB has been modified for IDAPS, as described in Section 4.0.

2.1.1.6 ROLRTN - Roll-out/Roll-in Subroutine

ROLRTN is responsible for the roll-out and roll-in of "background" tasks such
as the FORTRAN Compiler when this is necessary to service the IDAPS application
tasks. Three disk files (13C6, 14C6, 15C6) are reserved for this purpose in
the IDAPS configuration.

2.1.1.7 SUPV - RTOS Supervisor

SUPV is a System Task whose task name is *SUPER. SUPV is responsible for
interpreting and processing RTOS commands entered through the Teletype
console. SUPV also contains the queue (IOLIST) in which Teletype output
messages are processed.

2.1.1.8 INIT - RTOS Initializer

INIT performs the actions necessary to start or restart the system. It has
entry points to handle the SYSG01, SYSG02, and SYSG03 startups (see RTOS
Reference Manual, Section 2.2) as well as an entry to handle the power-fail
recovery. INIT also contains the entry point (RTOP) which defines the
uppermost cell in core used by RTOS, thus INIT must always be the last
module in core.

2.1.1.9 Drivers and Device Control Blocks

The Drivers and Device Control Blocks (DCB's) provide the means of access to all of the peripheral devices. In general, a driver interprets the SVC 1 call which requests an I/O operation, then executes the device commands and I/O instructions which cause that operation to take place. A DCB contains the control information for a particular device. There is a DCB for each physical device, but a single Driver can service several similar devices. In the IDAPS configuration, this feature is used in two cases:

a)   MTDVR handles both magnetic tape drives.

b)   DIDVR handles the DICOMED Scanner and Recorder.

The following drivers and DCB's were written especially for IDAPS and are described in Section 3.1.1:

DGDVR   —   Anagraph Display Generator Driver
DGDCB   —   Anagraph Display Generator DCB
MXDVR   —   Anagraph Keyboard/Trackball Multiplexor Driver
MXDCB   —   Anagraph Keyboard/Trackball Multiplexor DCB
DIDVR   —   DICOMED Driver
SCDCB   —   DICOMED Scanner DCB
REDCB   —   DICOMED Recorder DCB
IFDVR   —   360 Interface Driver
IFDCB   —   360 Interface DCB

The following drivers and DCB's have been modified for IDAPS, as described in Section 4.0:

LFCDVR   —   Line Frequency Clock Driver
TTYDVR   —   Teletype Driver

CRDVR   —   Card Reader Driver

DSC200   —   Disk Driver

DCBC6L   —   Disk DCB

The remaining drivers and DCB's are in their standard form as supplied by INTERDATA, Inc.:

DCB6D   —   Line Frequency Clock DCB

PICDVR   —   Precision Interval Clock Driver

DCB6C   —   Precision Interval Clock DCB

MTDVR   —   Magnetic Tape Driver

DCB85   —   Magnetic Tape '85' DCB

DCB85   —   Second copy, Magnetic Tape '95' DCB

DCB02   —   Teletype DCB

DCB04   —   Card Reader DCB

LPDVR   —   Line Printer Driver

DCB62   —   Line Printer DCB

2.1.2   Disk File Allocation and Usage

The disk unit contains two physical disks whose device numbers are 'C6' and 'C7'. The disks are logically divided into several different files; and, with the RTOS disk file facilities, each file is treated as a separate device by the IDAPS and utility programs.

The basic addressable record on the disk is a sector, which contains 256 bytes. There are 24 sectors in each track. At a given position of the read/write heads, 2 tracks are accessible; and this quantity is called a cylinder. There are 408 cylinders on each of the two disks. In RTOS, the disk files are allocated in whole numbers of cylinders.

Disk file allocation is controlled by the Allocate (ALLO) and Release (RELE) commands (see INTERDATA RTOS Reference Manual, Publication No. 29-240,

Section 2.6). Three of the files (20C6, 21C6, and 22C6) have special meaning
to RTOS and cannot be changed ·by ALLO ·and RELE. The current allocation of
the disk files is as follows:

	Starting Cylinder		Ending Cylinder	
File Name	Hexadecimal	Decimal	Hexadecimal	Decimal
00C6	119·	281	158	344
01C6	159	345	168	360
02C6	169	·361	178	376
13C6	179	377	17B	379
14C6	17C	380	17E	382
15C6	17F	383	181	385
16C6	182	386	183	387
17C6·	184	388	187	391
18C6	188	392	197	407
20C6	0	0	3	3
21C6	4	4	4	4
22C6	5	5	118	280
00C7	FF	255	FF	255
01C7	100	256	197	407
03C7	60	96	D8	216
04C7	D9	217	FE	254
10C7	0	0	5F	95

The usage of the various files or groups of files is as follows:

a)   File 00C6 is used by the utility tasks as a scratch file.

b)   File 01C6 is where the assembler and compiler normally placé the
     object code which they produce.

c)   File 02C6 is normally used to contain a task generated by TETC.
      (See paragraph 2.1.5.4)

d)   Files 13C6, 14C6, and 15C6 are the roll-out/roll-in files used by
      the RTOS task loader, LODER.

e)   File 16C6 contains the current values of the IDAPS menu parameters.

f)   File 17C6 contains the HELP messages.

g)   File 18C6 contains the IDAPS Data Tables (current and default
      values).

h)   File 20C6 contains a core-image copy of RTOS for loading by the
      RTOS Disc Bootstrap Loader.

i)   File 21C6 contains the Task Library Index.

j)   File 22C6 contains the Task Library.

k)   File 00C7 is used as a scratch file by some of the IDAPS tasks.

l)   File 01C7 contains the current terminal display images.

m)   File 03C7 contains the IDAPS Subroutine Library.

n)   File 04C7 contains the FORTRAN Run-Time Library.

o)   File 10C7 contains the User Identification File, the User Utiliza-
      tion File, and a dynamic record of all 360 operations performed
      since the last LOG IN operation.

2.1.3 IDAPS Task Organization

The IDAPS tasks, of which there are approximately ninety, were written to operate under control of RTOS and to perform all the various functions needed to support the IDAPS terminal.

Most of the tasks are directly concerned with interacting with the IDAPS user to let him direct the activity of the system as he desires. The starting point for each operation is the Master Menu, which is handled by the task called MASTER. According to the option chosen, MASTER passes control to one of the Sub-menu Tasks. The Sub-menu Tasks are identifiable as those whose names end with the letters 'SM'. The Sub-menu Task, in turn, passes control to the appropriate Parameter Specification Task, whose names end with the letters 'PS'. When the Parameter Specification Task completes its operation, control is passed back to MASTER, where the process can start again. Figure 2-2 is a graphic illustration of the functional flow of IDAPS.

The mechanism for passing control from one task to another, as described above, is provided by the RTOS SVC 6 operation (RTOS Reference Manual, Section 3.10). This facility is available to FORTRAN programs through the LINK subroutine which is in the IDAPS Subroutine Library.

The other major category of IDAPS tasks are the Special Function Tasks, whose names end with the letters 'SF'. These tasks are called by the keyboard input subroutines as a result of one of the Special Function Keys being pressed.

There are several other IDAPS tasks which do not fall into one of the major categories. These tasks perform various miscellaneous functions. All of the IDAPS tasks are described in Section 3.1.2.
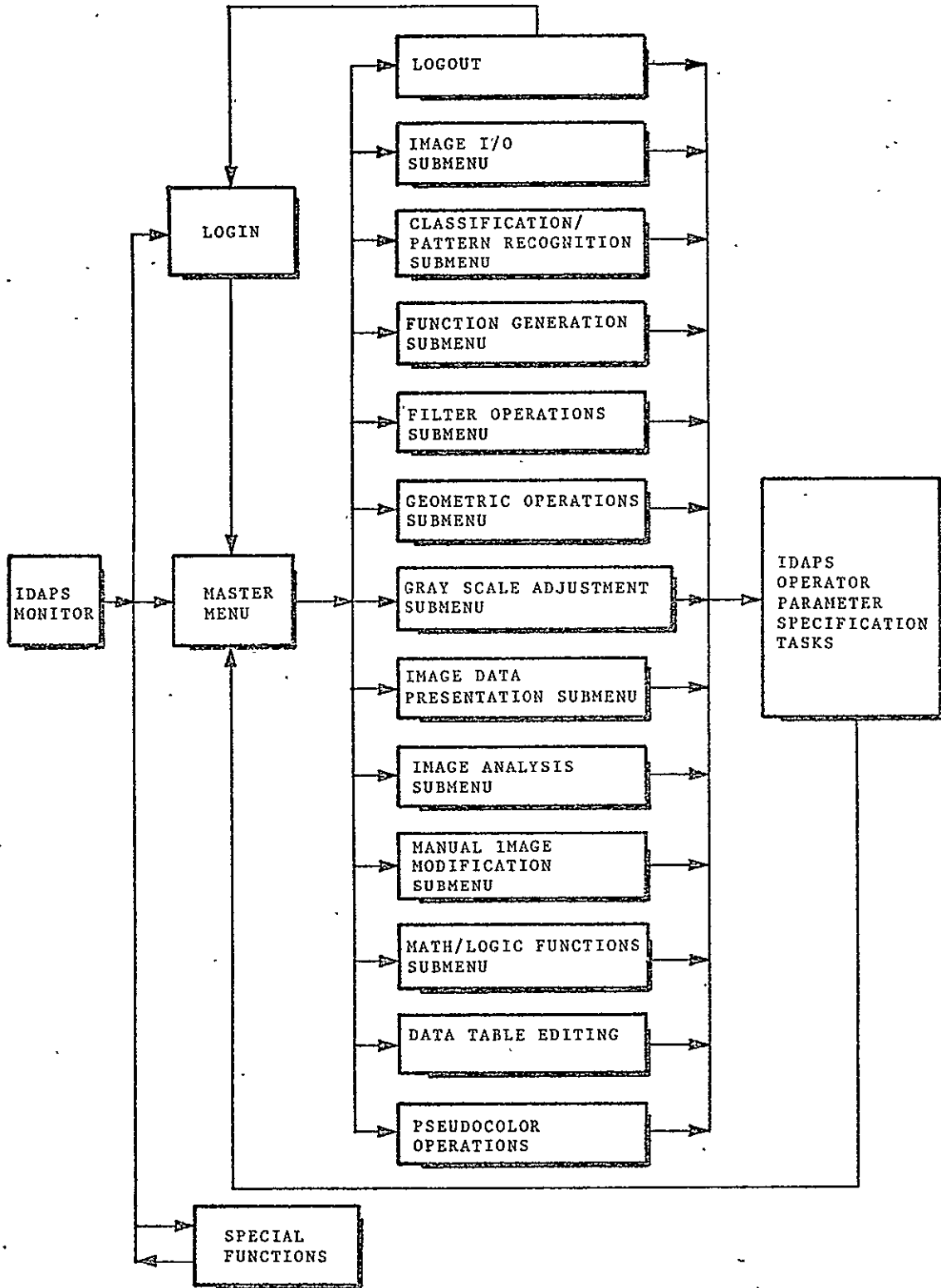
Figure 2-2.   General Functional Block Diagram of ID-70 Tasks

2.1.3.1 Task Common

The IDAPS tasks make use of the RTOS facility for defining and using a Task
Common area. The starting location of Task Common is contained in the RTOS
entry point called TSKCOM. Thus, in a machine-language program, the value of
TSKCOM is used as the base address for references to Task Common. In a
FORTRAN program, TSKCOM is used as the name of a labeled common block. The
elements within Task Common are defined in each program according to their
relative location. The IDAPS Task Common area is allocated as follows:

Relative Byte Location (Dec.)	Name	Description
0	KBCH	Keyboard input character. If non-zero, contains right-justified ASCII code.
2	TBFLAG	Trackball input flag. If non-zero, there is a trackball input.
4	TBX	Trackball X-coordinate (column).
6	TBY	Trackball Y-coordinate (line).
8	FLG360	Status of 360 interface. (0 = down, 1 = ready, 2 = busy, 3 = busy-wait).
10	FILNAM	Name of last-generated 360 file, 4 bytes ASCII.

14	UTIME	Minutes of elapsed time since last login.
16	TRSFLG	Interactive termination flag (0 = TRM360 termination, 1 = TRS360 termination).
18	LSAD	Pointer to next available sector on logout file (10C7).
20	LSTORE	Store/don't store log file (0 = store, 1 = don't store).
22	LU	Logical unit (1 or 2) for LOGOUT tape IO.
24	K1	Output option (1 or 2) for LOGOUT.
26	SCRTCH	Scratch area.
512	FILMSG	Start of File Directory Message.
516	FILTAB	Start of File Directory Data. Each entry occupies 10 bytes as follows:

Bytes	Contents
0-3	File name, ASCII
4-5	No. of lines, binary
6-7	No. of columns, binary
8-9	File type (1 = integer, 2 = floating point)

The FORTRAN statements to produce this allocation could be as follows:

```
INTEGER*2 KBCH,TBFLAG,TBX,TBY,FLG360,FILNAM,UTIME,TRSFLG,LSAD,LSTORE,
LU,K1,SCRTCH,FILMSG,FILTAB
```

COMMON /TSKCOM/ KBCH,TBFLAG,TBX,TBY,FLG360,FILNAM(2),UTIME,TRSFLG,LSAD,
                LSTORE,LU,K1,SCRTCH(243),FILMSG(2),FILTAB(254)

2.1.3.2 Logical Unit Assignments

Under RTOS, all references to peripheral devices are in terms of logical
unit numbers. The assignments used by the IDAPS tasks are as follows:

Logical Unit	Physical Unit	Device Description
0	0C7	Scratch File
1	1C7	Display Image File
2	02	Teletype
3	62	Line Printer
4	04	Card Reader
5	18C6	Data Table File
6	1F	Keyboard/Trackball Multiplexor
7	10C7	LOG File
8	85	Magnetic Tape #1
9	95	Magnetic Tape #2
A	8A	DICOMED Scanner
B	8B	DICOMED Recorder
C	36	360 Interface
D	FF	Anagraph Display Generator
E	16C6	Default Parameter File
F	17C6	HELP Message File

2.1.4 IDAPS Subroutine Library

The IDAPS Subroutine Library consists of approximately 110 subroutines which have been produced to perform various functions for the IDAPS programs. These subroutines are described in Section 3.1.3.

The library is built and modified using the OS Library Loader, LDR3 (see Section 2.1.5.3). When a new library is produced, it is first made on a magnetic tape. This tape is then duplicated to file 3C7, which is the working location of the library.

When an IDAPS task is established using TETC (see Section 2.1.5.4), the "EDIT 3C7" card instructs TETC to use the library. The EDIT operation causes those subroutines called within the task to be retrieved from the library and made a part of that task.

2.1.5 Program Development Tools

2.1.5.1 Extended FORTRAN Compiler - FTN

The Extended FORTRAN Compiler, as supplied by INTERDATA Inc., has the task name FTN in the IDAPS configuration. The nominal logical unit assignments which may be of interest to the user are as follows:

Logical Unit	Physical Unit	Description
1	04	Source deck input, card reader
2	1C6	Object code output, file 1C6
3	62	Program listing, line printer
7	62	Error listing, line printer

The language and deck structure processed by FTN are described in the INTER-DATA FORTRAN IV Reference Manual, Publication No. 29-220, and include the extended features indicated in the manual. All IDAPS FORTRAN program and subroutine decks begin with a card of the form "$LAB= name". The general

deck structure is as follows:

$LAB= name

    SUBROUTINE name(parameters) "If this is a subroutine"
    "declarative statements"
    "executable statements"
    END

The general procedure for compiling a program is as follows:

a)   Place program deck in card reader, make card
     reader ready.

b)   Make line printer ready.

c)   Type in command: REWI 1C6
     If compilations and assemblies are being stacked on 1C6,
     omit this command for subsequent compilations.

d)   Type in command: STAR FTN
     Compiler will now execute.

e)   When message "FTN.....EOJ" is received, type
     in command: DELE FTN

Many of the FORTRAN language forms generate calls to run-time subroutines.
These subroutines are contained in the FORTRAN Run-Time Library, which
resides on file 4C7. The library is as supplied by INTERDATA Inc., and
includes the RTOS extensions. The library is used in the operation of TETC,
as the result of the "EDIT 4C7" card (see Section 2.1.5.4).

2.1.5.2 Machine Language Assembler - ASM

The Assembler, as supplied by INTERDATA, Inc., has the task name ASM. The
nominal logical unit assignments which may be of interest to the user are as
follows:

Logical Unit	Physical Unit	Description
1	04	Source deck input, card reader
2	1C6	Object code output, file 1C6
3	62	Program listing, line printer

The language and deck structure processed by ASM are described in the INTER-
DATA Model 70 User's Manual, Publication No. 29-261, Section 11.3 (Section
7.5 in Publication No. 29-261R01). The general deck structure is as follows:

Column 1	Column 10	Column 16
	OPT	SCRT,GO,LAB=name
	ENTRY	name
name	EQU	*
	program body	
	END	

The general procedure for assembling a program is as follows:

a) Place program deck in card reader, make card
    reader ready.

b) Make line printer ready.

c) Type in command: REWI 1C6
    If compilations and assemblies are being stacked on 1C6,
    omit this command for subsequent assemblies.

d) Type in command: STAR ASM
    Assembler will now execute.

e)   When message "ASM.....EOJ" is received, type

in command: DELE ASM

2.1.5.3 OS Library Loader - LDR3

The OS Library Loader, as supplied by INTERDATA Inc., has the task name
LDR3. The nominal logical unit assignments which may be of interest to the
user are as follows:

Logical Unit	Physical Unit	Description
3	62	Line printer
8	85	Magnetic tape # 1
9	95	Magnetic tape # 2
A	0	Null device
B	1C6	Object code file
D	3C7	IDAPS Subroutine Library File
E	4C7	FTN Run-Time Library File

The operations of LDR3 are described in the INTERDATA Model 70 User's Manual,
Publication No. 29-261, Section 11.4.3 (Section 7.9.4 in Publication No. 29-
261R01). Care must be taken when using LDR3 to operate on disk files rather
than tapes. Several of the LDR3 operations involve reading to a particular
point in a file, then backspacing to achieve the desired positioning. These
operations will not work properly on disk files.

In regards to IDAPS, the use of LDR3 fulfills three major purposes:

a)   Produce IDAPS Subroutine Library

b)   Produce RTOS Object Code File

c)   Produce RTOS Load Module

The procedures to accomplish each of these are described in the following
paragraphs.

2.1.5.3.1 Production of IDAPS Subroutine Library

The first step in producing a new library is to compile and/or assemble the subroutines to be replaced or added (see Section 2.1.5.1 and 2.1.5.2). The object code for these subroutines should be stacked on file 1C6 in the order in which the subroutines are to appear in the library. The order of subroutines in a library must be such that a particular subroutine appears before any subroutines which it calls.

Next, mount a scratch tape (with write-ring) on tape drive #1 and the current Subroutine Library tape on tape drive #2.

At the teletype, enter command: STAR LDR3

When the message "LDR3 ENTER DATA:" is received, LDR3 is ready for commands, as follows:

a) Enter command: RE B
    This "rewinds" the object code file (1C6).

b) Enter command: EO 8
    This writes an EOF and initializes tape 1, where the new
    library is to be produced.

c) Enter command: DU 908 name
    This duplicates subroutines from the current library (tape
    2) to the new library (tape 1) up to, but not including,
    the subroutine specified by "name". The named subroutine
    will be in one of two categories:

    1) Subroutine being replaced by a new version.
    2) Subroutine before which a new subroutine is
        being added.

d)   If a subroutine is being replaced at this point,
     enter command: CO 90A
     This copies the subroutine from the current library to a null
     device which, in effect, skips past it. If a subroutine is being
     added at this point, leave out this step.

e)   Enter command: CO B08
     This copies the modified or new subroutine from the object code
     file (1C6) to the new library.

f)   Repeat steps c, d, and e for each subroutine on file 1C6.

g)   Enter command: DU 908
     This duplicates the remainder of the current library to the new
     library and completes the new library.

h)   Check to see that the line printer is ready and enter command:
     TA 803
     This "tables", i.e., lists the contents of the new library on the
     printer. Verify that all subroutines are present and in the
     proper order.

i)   Enter commands: RE 8, RE D, DU 80D
     This duplicates the new library to the Subroutine Library File
     (3C7), where it will subsequently be used in establishing tasks.

j)   Enter command: END
     This terminates the operation of LDR3.

The tape on tape drive #1 is now the current library tape and should be
saved as such (remove write-ring).

2.1.5.3.2 Production of RTOS Object Code File

When one or more of the RTOS programs are to be modified, the first step is to assemble the modified programs in the proper order, with the object code being stacked on file 1C6. The order of the RTOS programs is as listed in Section 2.1.1.

Next, mount a scratch tape (with write-ring) on tape drive #1 and the current RTOS Object Code tape on tape drive #2.

At the teletype, enter command: STAR LDR3

When the message "LDR3 ENTER DATA:" is received, LDR3 is ready for commands.

Follow the procedures described in steps "a" through "h" of Section 2.1.5.3.1.

Enter command: END

This terminates the operation of LDR3.

The tape on tape drive #1 is the new RTOS Object Code tape and should be saved as such (remove write-ring).

2.1.5.3.3 Production of RTOS Load Module

When a new RTOS Object Code file has been produced, as described in Section 2.1.5.3.2, it must be converted into a Load Module before it can be loaded into core and activated.

Mount the current RTOS Object Code tape on tape drive #1 and a scratch tape (with write-ring) on tape drive #2.

At the teletype, enter command: STAR LDR3

When the message "LDR3 ENTER DATA:" is received, LDR3 is ready for commands

as follows:

a) Enter command: EO 9

    This writes an EOF and initializes tape 2, where the new Load
    Module will be produced.

b) Enter command: OUT 9

    This prepares LDR3 to produce a Load Module on tape 2.

c) Enter command: BI 20

    This sets the core location bias to the value needed by RTOS.

d) Enter Command: LO 8

    This causes the first program on tape 1 to be loaded and processed
    as the first segment of the Load Module. This program is EXEC,
    the RTOS Executive.

e) For each remaining RTOS program, enter command: LI 8

    Each time this command is entered, the next program on tape 1 is
    linked into the Load Module, until on the last time, INIT, the
    RTOS Initializer, is linked in.

f) See that the line printer is ready, and enter command: MA 3

    This causes the RTOS Load Map to be printed. The Load Map should
    be saved, as it is necessary for tracing problems and making
    temporary modifications to RTOS.

g) Enter commands: XOUT, END

    These commands cause LDR3 to complete the Load Module tape and
    terminate operation.

The tape on tape drive #2 is the new RTOS Load Module tape and should be
saved as such (remove write-ring). The procedure for loading the new version
of RTOS into core and activating it is described in Section 5.1.4.

2.1.5.4 Task Establisher - TETC

The Task Establisher, as supplied by INTERDATA Inc., usually has the task name TET. TETC is a copy of TET with its logical unit assignments set up to be convenient for the IDAPS configuration. The differences between TETC and TET are as follows:

a)   TETC reads commands from the card reader; TET reads them from the teletype.

b)   TETC prints messages on the line printer; TET prints them on the teletype.

The operations of TET/TETC are described in the INTERDATA RTOS Reference Manual, Publication No. 29-240, Section 2.7. The primary purpose of TETC is to build a task, as defined by RTOS, from the object code of a program and any subroutines that program may call. For the IDAPS tasks, a standard TETC control deck has been developed. The cards which make up the deck are as follows:

a)   ESTA name
     The "name" on this card determines the name of the task being established. This is the only card in the deck which changes for each individual task.

b)   OPTI 0100 1000 0100 0000
     This card determines the task's option configuration.

c)   GET 168
     This card causes working space to be allocated for certain FORTRAN-generated usage.

d)   EXCL TSKCOM
     This card causes linkage to the Task Common pointer.

e)   ASSI 0,0C7,1,1C7,2,02,3,62,4,04,5,18C6,6,1F,7,10C7

    ASSI 8,85,9,95,A,8A,B,8B,C,36,D,FF,E,16C6,F,17C6

    These cards determine the Logical Unit assignments.

f)   LOAD 1C6

    This card causes the object code of the program to be loaded
as the first segment of the task.

g)   EDIT 3C7, EDIT 4C7

    These cards cause the IDAPS Subroutine Library and FORTRAN Run-
Time Library to be used as sources of subroutines to be added into
the task.

h)   MAP 62

    This card causes a load map of the task to be printed on the line
printer.

i)   TASK 2C6

    This card causes a copy of the task to be stored on file 2C6.

j)   END

    This card causes TETC to terminate its operation.

Before operating TETC to establish a task, the object code of the program
should be stored on file 1C6 by the FORTRAN Compiler or the Assembler (see
Sections 2.1.5.1, 2.1.5.2). The following operations should then be performed:

a)   Place TETC control deck, as described above, in card reader; make
card reader ready. Make line printer ready.

b)   Enter commands: REWI 1C6, REWI 2C6, STAR TETC

Operation of TETC requires one to three minutes. Completion is signalled by
the teletype message "TETC.....EOJ". At this point, the resulting task is

on file 2C6. Incorporation of the task into the Task Library is described
in Section 2.1.5.5.

2.1.5.5 Task Utility - TUT

The Task Utility program, as supplied by INTERDATA Inc., has the task name
TUT. The operations of TUT are described in the INTERDATA RTOS Reference
Manual, Publication No. 29-240, Section 2.8. TUT is used to maintain the
Task Library on which all of the IDAPS tasks and program development tasks
reside.

The most common usage of TUT is the insertion of a task into the library.
Before this can be done, the task must have been established, normally on
file 2C6, as described in Section 2.1.5.4. The following operations should
then be performed via the teletype:

```
REWI 2C6
STAR TUT
INSE 2C6
END
```

If a task of the same name already exists on the library, this operation
accomplishes a replacement of the old version with the new.

A task may be deleted from the library by the following commands:

```
STAR TUT
DELE name
END
```

A list of the current contents of the library may be produced on the line
printer by the following commands:

```
STAR TUT
INDE 62
END
```

Infrequently, a current copy of both the Task Library file and its Index
should be dumped to mag tape. These backups are the only way to regenerate
the Task Library in case of disk failure. The following operations
create a backup:

a)   Mount two scratch tapes on tape drives #1 and #2

in TUT:

b)   COPY 21C6,85

c)   COPY 22C6,95
      Note: This file is the Task Library file and takes several minutes
      to copy.

      If an IO error of the form '8485' or '8495' occurs on a copy
      operation, try switching drives or a different scratch tape.

d)   Put labels including the current date on both tapes, and
      remove their write rings.

As the process of task replacement and deletion goes on, unused space is
left in the Task Library file. If this is allowed to continue indefinitely,
the file will fill up to the point that no more insertions can be made.
A special compress operation is provided to repack the library and get
rid of the unused space. This operation must be used very carefully,
since any reference to the library by another task or any malfunction
during the compress operation may result in a total loss of the library.
The following steps should be performed:

a)   Make sure there are no tasks currently operating.

b) To ensure that no operations are initiated through the IDAPS
    keyboard, disable the IDAPS Monitor Program with the following
    command: CANC IDAPS

c) Enter commands: STAR TUT, COMP WARN
    The compress operation requires up to 10 minutes to complete.
                Be patient and leave the teletype alone. When the operation
                is complete, TUT outputs the new task index list on the
                teletype.

d) When TUT requests another command, enter: END
    This terminates the operation of TUT.

e) The IDAPS Monitor may now be restarted with the command: STAR
    IDAPS
    Normal operations may now be resumed.

2.2 IBM-360 MAJOR COMPONENTS

2.2.1 Control Program

The Control Program of the IBM-360 IDAPS system consists of all the
software necessary to provide support for the applications programs
(operators). The Control Program serves as the monitor for all processing
requests, batch or interactive. In addition to overall control of the
order of excution in the 360 IDAPS system, the Control Program performs
the following tasks:

a) Determine whether batch or interactive mode of operation.

b) Open all disk files.

c) Provide disk read/write capability.

d)   Provide tape read/write capability in batch mode.

e)   Interface with the ID-70 in interactive mode.

f)   Provide file management of disk files.

g)   Report error conditions.

The Control Program consists of several modules as illustrated in Figure
2-3 and discussed in the following sections.

2.2.1.1 Executive Module

The Executive module of the Control Program provides executive control
of the IDAPS system.   This module determines the mode of operation,
opens all disk files, reads commands and directs control to the other
modules of the Control Program.   In the interactive mode of operation,
this module also monitors the ID-70 for commands and notifies the ID-70
upon completion of a command.

2.2.1.2 ID-70 Interface Module

Interface between the IBM-360 and the ID-70 is handled on the 360 side
by the ID70 Interface module.   This module establishes and maintains
synchronous operation between the two computers, provides the capability
to transmit data and checks for and reports error conditions.

2.2.1.3 Command Interpretation Module

Applications tasks on the 360 are initiated by 80 byte card images.   In
batch mode these are read by the card reader.   In interactive mode they
are read via the interface device.   Interpretation of the command card
images is performed by the Command Interpretation module.   This module
checks the card image for syntax errors, separates the command parameters
and input file names, and interprets the command parameters.
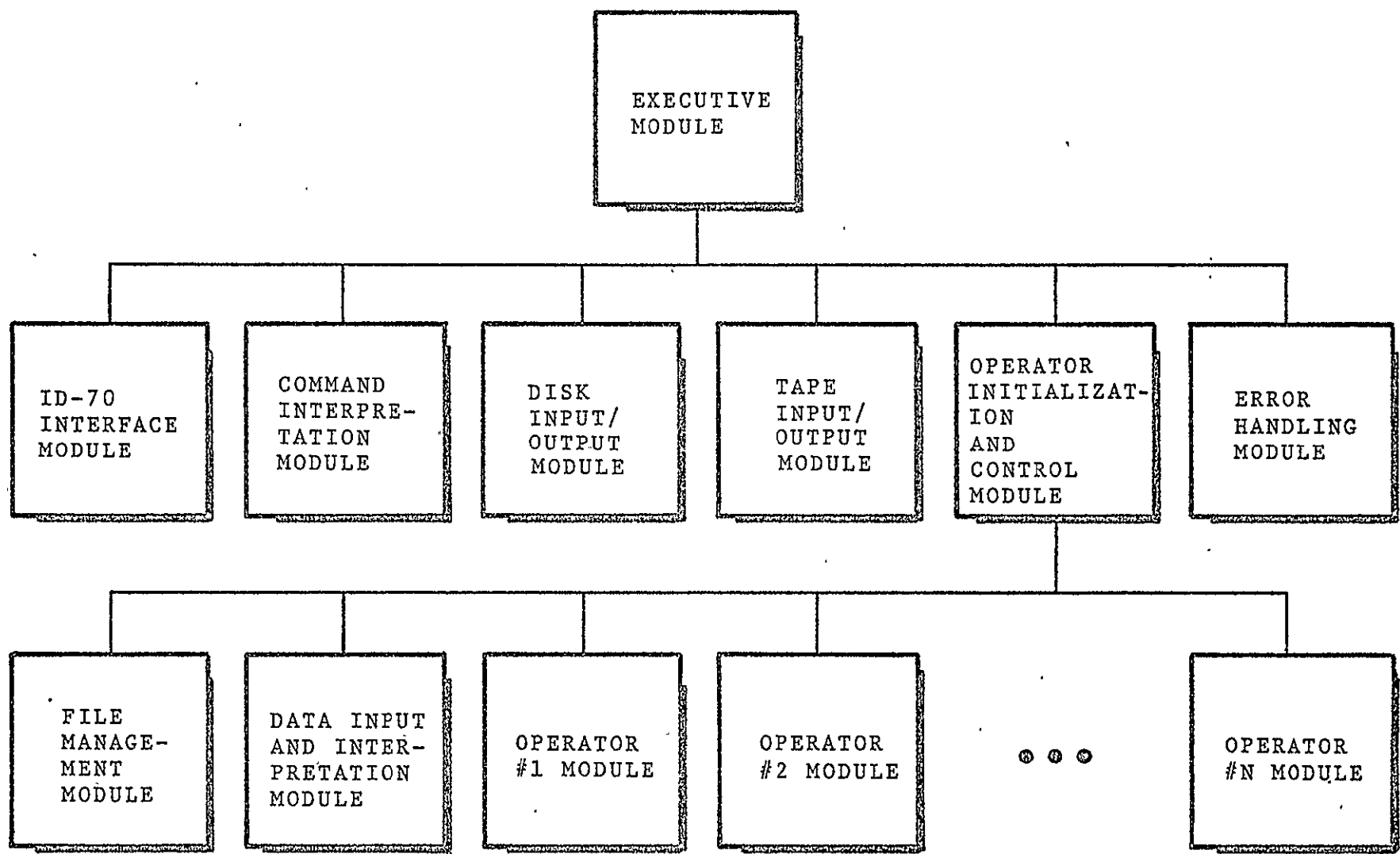
Figure 2-3.   IBM-360 Control Program Block Diagram

2.2.1.4 Operator Initialization and Control Module

The Operator Initialization and Control module is the application program executive. This module reads any additional data required by the operator, calls any preprocessing subroutine required, calls the file manager and passes input parameters to the operator through the calling sequence. This module checks for errors in the data input, file management or application programs and reports error conditions to the IDAPS Executive.

2.2.1.5 Data Input and Interpretation Module

Non-image data required by the 360 operators is read and interpreted by the Data Input and Interpretation module. This data is in the form of 80 byte card images and this module checks the card images for syntax errors, interprets the card images and stores the data values. In batch mode these are read by the card reader. In interactive mode they are read via the interface device.

2.2.1.6 File Management Module

The bookkeeping associated with naming, storing, indexing and retrieving the system image and floating point disk files is provided by the File Management module. This module maintains the file directories which contain the pertinent information relating to the active files, the scratch files, and the disk space available for use. This module also checks for and reports error conditions related to file specification, file deletion and space availability.

2.2.1.7 Disk Input/Output Module

The image and floating point data generated by the operators are stored in the system disk files. The storage and retrieval of this data are handled by the Disk Input/Output module. Any errors that may occur are reported to the module which called the Disk Input/Output module and should then be reported to the Executive module.

2.2.1.8 Tape Input/Output Module

In the batch mode input of images to be processed must be from magnetic tape. It is likely that these input tapes were generated on the ID-70 or in previous batch mode runs. It is also desirable to output processed images on magnetic tape for display on the ID-70 or subsequent batch processing. The capability to read and write magnetic tapes in a format compatible with the ID-70 tape devices is provided by the Tape Input/Output module. Any errors that may occur are reported to the module which called the Tape Input/Output module and should then be reported to the Executive module.

2.2.1.9 Error Handling Module

Errors which are detected and reported to the Executive module are handled by the Error Handling module. In the interactive mode these conditions are reported to the ID-70. In the batch mode a hardcopy error message is provided for the user and the job is terminated.

2.2.2 COMMON Storage and Initialization

Several arrays of variables are used by a number of control and application packages and are included in COMMON blocks. These variables have initial or default values which are set in the BLOCK DATA subroutine. These variables will be discussed on a COMMON block basis in the following paragraphs.

COMMON /OPER/ contains the IDAPS Operator directory IOP (3,100) and will handle up to 100 operators. There are three entries per operator: name, number of parameters and file requirement. These entries are stored as follows:

    IOP(1,I) = Name (3 character operator acronym)
    IOP(2,I) = Number of parameters on command card
    IOP(3,I) = File requirement

The file requirement is entered as a six digit decimal number where each
digit indicates how many of a particular type of file is required by the
operator. The organization of the file requirement entry is illustrated
below:

IOP(3,I) = UVWXYZ, where

        U = number of input files

        V = number of integer output files

        W = number of floating point output files

        X = number of secondary output files

        Y = number of integer scratch files

        Z = number of floating point scratch files

COMMON/FILEPR/ contains the file management control variables. The IDAPS
file directory FD(6,164) handles the file information for the
secondary output files and up to 160 active files. The file information
entries are stored as follows:

    FD(1,I) = Name (4 character file name)

    FD(2,I) = File format (1 - integer, 2 - floating point)

    FD(3,I) = Number of records (lines)

    FD(4,I) = Number of bytes (columns)

    FD(5,I) = Relative address of first line of file

    FD(6,I) = Logical unit number

The number of active integer files (PCOUNT), active floating point files
(DCOUNT) and the index of the next file directory entry (FCOUNT) are
stored in the common block. The counter used to number the files created
by each operator, FILNDX(100), and the array used to assign alternate
packs, INP(3), are stored in this common block. KDELX(9) contains the number
of files which have been deleted for each of the 9 logical units where
active files are stored.

COMMON/FILEDR/ contains the input, output and scratch file directories
which are provided for the operators. The input file directory, IFD(6,5),
contains the file management information stored in the IDAPS file directory
for up to 5 input files as required by the operator. The output file
directory, OFD(6,5), provides the file management information for the
output files required by the operator. The entries in the output file
directory are organized as follows:

     OFD(I,1) = First output file
     OFD(I,2) = Second output file
     OFD(I,3) = First secondary file·
     OFD(I,4) = Second secondary file
     OFD(I,5) = Third secondary file

COMMON/SPACE/ contains the primary space directory PSPACE(2,3,3) and the
deleted space directory DSPACE(2,3,3,50). The entries in these directories
are organized as follows:

     PSPACE(1,J,K)   = Number of records (lines) available
     DSPACE(1,J,K,L) = Number of records (lines) available
     PSPACE(2,J,K)   = Next relative block address
     DSPACE(2,J,K,L) = Next relative block address
where
     J = 1, pack number 1
       = 2, pack number 2
       = 3, pack number 3
     K = 1, 512 byte line length
       = 2, 1024 byte line length
       = 3, 2048 byte line length
     L = index of deleted space entry.

COMMON /BUF/ contains the interface message buffer. MESS is the four
byte binary message type code and LL(512) provides a 2048 byte area for
the body of the message. COMMON /TIMER/ contains the timer option
variable KTIME. For KTIME = 0, the CPU time for each operator is calculated.
For KTIME = 1, the elapsed wall clock time for each operator is calculated.
COMMON /CDBUF/ contains the computed data buffer. The variable NWORDS
determines the number of values entered in the buffer and CD(300) provides
a 1200 byte area permitting a maximum of 300 computed data values to be
stored. CD is a floating point buffer; however, using EQUIVALENCE to
set up an integer array which will share the buffer space, integer
values may be stored. These three common blocks are not initialized by
BLOCK DATA.

The common blocks and the initial values as set by BLOCK DATA are shown
in Section 3.0 of the Appendix.

2.2.3 Disk Space Allocation

The operator output in the form of image files or floating point files
is stored on disk. There are five types of data files as illustrated in
Figure 2-4. Output files come in three lengths (512, 1024 or 2048
bytes). Output files are used to provide permanent storage for image or
floating point data which can be subsequently processed by other operators.
Scratch files provide temporary storage for intermediate data, image or
floating point, which an operator may generate. Secondary output files
provide temporary storage for image data which may be processed by other
operators. Secondary output files differ from scratch files in that
they may be accessed as input files by other operators. Any use of an
operator which creates secondary output files will result in previous
secondary output files being destroyed.

Type of Data	Type of File	Length (Bytes)	No. of Records
Output (image)	1	512	5120
Output (image)	2	1024	6144
Output (image or floating point)	3	2048	30720*
Scratch (image or floating point)	4	2048	8192
Secondary (image)	5	2048	2048

*Batch pack number 1 has only 24576 records.

Figure 2-4.   Types of IBM-360 System Disk Files

The five types of data files are allocated space on three disk packs. Each pack has one of each of the five file types, and the number of records available in each file type is given in Figure 2-4. Each of these 15 files is referenced internally by a logical unit index. This index is a two-digit decimal number where the first digit references the pack number and the second type of file. For example, if the logical unit index is 22, then space is assigned in the 1024 byte output file on pack number 2. The correlation between internal pack numbers and the external pack identification for batch and interactive IDAPS systems is given below:

Disk Number	Batch Pack ID	Interactive Pack ID
1	SDC001	SDC201
2	SDC002	SDC202
3	SDC003	SDC004

2.2.4 Overlay Structure

One of the features of the IDAPS system is the facility to easily add
new applications packages. In order to maintain a stable core requirement
in a changing environment, the IDAPS system utilizes the overlay capability
provided by the IBM software. Each applications package resides in an
overlay segment and a copy is brought from disk into core at the time it
is executed.

The overlay structure used by IDAPS consists of three major divisions;
the Control Division, the Transient Division and the Region Division.
The Control Division consists of Control Program subroutine load modules
which are always core resident. The Transient Division consists of
segments containing the application subroutine load modules and the
Control Program subroutine load modules which do not have to be core
resident. The Region Division consists of segments containing subroutine
load modules which are used by the Control Division and/or Transient
Division modules but do not always need to be core resident.

The Control Division requires 70,000 bytes of core storage. The longest
segment of the Transient Division requires 164,000 bytes of core storage.
The longest segment of the Region Division requires 16,000 bytes of core
storage. Using this overlay structure the 360 IDAPS System requires a
minimum of 250,000 bytes of core storage.

2.2.5 System Libraries

The modules which comprise the IDAPS system reside in three libraries
which are stored on the SDC001 disk pack. There are two types of libraries,
load module and binary. There is one binary format library (USER) which
is made up of subroutines which have been compiled and collected in
binary format in a sequential data set. The load module libraries,
IDAPS.CONTROL and IDAPS.APPLIC, are partitioned data sets, where each

member is a subroutine which has been compiled and link edited to generate
a load module. The distinct advantage to this type of library is that
in order to make a program change only the affected subroutine(s) must
be recompiled. These libraries will be discussed separately with attention
given to their content and purpose.

2.2.5.1 USER

The binary format library USER contains the Control Program subroutines
and the overlay control statements which must be changed when adding a
new operator. The control program subroutines included in the USER
library are listed below:

    BLOCK DATA
    INIT
    INIT2
    INTPAR
    MAIN
    OPERR

Since it is necessary for the MAIN and BLOCK DATA subroutines to be
included in the primary input to the linkage editor, the USER library is
concatenated with OS linkage editor primary input data set &LOADSET. An
additonal attractive feature of this approach arises from the fact that
the overlay structure control statements must also be included in the
primary input to the linkage editor. The IDAPS system catalogued procedure
USERMOD discussed in Paragraph 6.1 generates the binary modules for the
five subroutines and merges these with the overlay structure control
statements to produce the binary format library USER.

2.2.5.2 IDAPS.CONTROL

The remaining control program subroutines reside in the load module
library named IDAPS.CONTROL. This library is concatenated with the OS
library SYS1.FORTLIB as secondary input to the linkage editor. The

IDAPS system provides three catalogued procedures (ASCLIDAP,FGCLIDAP and
FHCLIDAP) discussed in Paragraph 2.2.6.1 which compile, link edit and
add or replace members in IDAPS.CONTROL. The subroutines included in
the IDAPS.CONTROL library are listed below (entry points in parentheses):

ALPIN (FLTIN,INTIN)
ASCEBC
CIR
CORE
DATAIN
DISKIO (DIRREA,DIRWRT)
DISKSP
DISKW (DISKR)
DISKWF (DISKRF)
DISKSP (DISKRP)
EBCASC
ECHO
ELTIME
ERRMES
ETIME
FILNAM
FILNFO
FMR
IANDF
IEXEC
INSERT
INTFAC
IORFU
ITOD
ITOP
ITOT

ITRANS

JGETB

JPUTB

LOGIN (STORE)

MOVST

NEXTPK

OPENNL (READNL,ENDTP1)

PACK

PCKINT

PCK608

PL4

SCD

SIZFIL

STIME

STOI

TTOI

UNPACK

UNPINT

WRITE9 (ENDFIL)

2.2.5.3 IDAPS.APPLIC

The application package modules reside in the load module library named
IDAPS.APPLIC. This library is concatenated with the OS library SYS1.FORTLIB
(and IDAPS.CONTROL) as secondary input to the linkage editor. The
modules included in the IDAPS.APPLIC library are discussed in paragraph
3.2.2. Adding or replacing members in IDAPS.APPLIC is accomplished by
using the catalogued procedures discussed in paragraph 2.2.6.1.

2.2.6 System Maintenance

A number of tools are provided for maintaining the IDAPS system. In
order to reduce the OS Job Control Language (JCL) statements required to
perform the routine tasks of maintaining IDAPS, the most often used
procedures have been catalogued. In addition a number of utility programs
are available in deck form which provide necessary support functions for

the IDAPS system. These tools and the techniques for updating the existing libraries and adding new operators to the IDAPS system will be discussed in the following paragraphs.

2.2.6.1 System Development Tools

The catalogued procedures are collections of JCL statements which provide OS with the instructions necessary to perform desired tasks on the IBM 360. The catalogued procedures reside in SYS1.PROCLIB and are listed in Section 1.0 of the Appendix. They will be discussed on an individual basis with attention given to their purpose and how they are to be used.

2.2.6.1.1 IDAPS Catalogued Procedure

The catalogued procedure IDAPS provides the user with the capability to run IDAPS in batch mode. The JCL statements needed to invoke IDAPS are given in the publication, Image Data Processing System (IDAPS) User Manual, Batch IDAPS, Volume II (TM-HU-037/200/00).

2.2.6.1.2 IDAPSI Catalogued Procedure

The catalogued procedure IDAPSI provides the user with the capability of readying the IDAPS IBM 360 system for an interactive session. The JCL statements needed to invoke IDAPSI are given in Paragraph 5.2.

2.2.6.1.3 IDAPSB Catalogued Procedure

The catalogued procedure IDAPSB provides the user with the capability to compile new or changed subroutines and run IDAPS in batch mode for checkout purposes. The JCL statements needed to invoke IDAPSB are the same as for IDAPS with two exceptions. First, the EXEC card must reference IDAPSB instead of IDAPS and indicate the binary library to be used. A sample EXEC card using IDAPSB and the binary library BAT1 is given below:

```
//CHECKOUT EXEC IDAPSB,BIN=BAT1,TIME.GO=10
```

Second, following the EXEĆ card the user must include his FORTRAN source
modules as illustrated below:

```
//FORT.SYSIN DD *

        .

        .   (FORTRAN SOURCE MODULES)

        .

/*
```

2.2.6.1.4 ASCLIDAP Catalogued Procedure

The catalogued procedure ASCLIDAP provides the user with the capability
to assemble and link edit a machine language source module into the load
module library IDAPS.CONTROL. The JCL statements needed to invoke
ASCLIDAP for a subroutine named ASEMBL are given below:

```
//ASEMBL EXEC ASCLIDAP,MEMNAME=ASEMBL
//ASM.SYSIN DD *

        .

        .   (ASSEMBLY LANGUAGE MODULE)

        .

/*
```

2.2.6.1.5 FGCLIDAP and FHCLIDAP Catalogued Procedures

The catalogued procedures FGCLIDAP and FHCLIDAP provide the user with
the capability to compile and link edit FORTRAN source modules into the
load module libraries IDAPS.CONTROL and IDAPS.APPLIC. FGCLIDAP employs
the FORTRAN G compiler which requires 100K of core and does not optimize
code. FHCLIDAP employs the FORTRAN H compiler with optimization level 2
which requires 250K of core and does optimize code. Either procedure is
invoked in the same manner. An example of how the H compiler might be
invoked to compile a subroutine named HOPT2 and then link the load
module into the IDAPS.APPLIC library is given below:

```
//HOPT2 EXEC FHCLIDAP,PDSNAME=APPLIC,MEMNAME=HOPT2
//FORT.SYSIN DD *

        .

        .  (FORTRAN SOURCE MODULE)

        .

   /*
```

2.2.6.1.6 USERMOD Catalogued Procedure

The catalogued procedure USERMOD permits the user to recompile the subroutines
in the binary library and update the overlay structure control statements.
Use of the USERMOD procedure is discussed in detail in Paragraph 6.1.3.

2.2.6.2 Utility Programs

The utility programs provide the user with the capability to execute tasks
which are necessary to support the IDAPS system. Listings of the utility
program decks are given in Section 2.0 of the Appendix. The purpose of each
utility program will be discussed in the following paragraphs.

If a disk pack goes bad it will be necessary to replace the pack. Before
using the new pack it will be necessary to scratch and structure the disk
space on the batch or interactive disk packs. The utility program used for
the batch disk packs is listed in Section 2.1 and the program for the
interactive disk packs is listed in Section 2.2.

If using ASCLIDAP, FGCLIDAP, or FHCLIDAP results in an IBM System Completion
Code of E37, then the space on the load module library being updated must be
compressed. It is necessary to run the appropriate compress utility program
to compress the library and free all available disk space. The utility
program to compress the IDAPS.CONTROL library is listed in Section 2.3 and
the program to compress the IDAPS.APPLIC library is listed in Section 2.4.

If the load module libraries become inoperable or are destroyed, it is
necessary to run the appropriate utility deck to scratch and build a library.
The utility program to scratch and build the IDAPS.CONTROL library is listed
in Section 2.5 and the program to scratch and build the IDAPS.APPLIC library
is listed in Section 2.6.

If a listing of the data sets which reside on the IDAPS disk packs is desired,
then the utility program to list the disk packs should be run. This utility
program will also calculate the available space on each pack and is listed
in Section 2.7.

If a listing of one of the libraries is desired, the lister utility program
should be run to generate a listing of the source code of the desired library
(or portion of the library). This program is listed in Section 2.8.

2.2.6.3 Updating Load Module Libraries

An existing module in one of the load module libraries IDAPS.CONTROL or
IDAPS.APPLIC may be updated (replaced) by using one of the ASCLIDAP, FGCLIDAP
or FHCLIDAP catalogued procedures. Repeated use of these procedures to
replace existing modules may result in all the library space being exhausted.
This will result in a System Completion Code of E37 and it will be necessary
to run the compress utility program for that library.

3.0 MODULE DESCRIPTION

3.1 INTERDATA-70 MODULES

3.1.1 Real-Time Operating System-RTOS

This section describes the RTOS modules which have been produced especially
for the IDAPS configuration. For a description of the standard RTOS modules,
refer to the INTERDATA RTOS Reference Manual, Publication No. 29-240, Chapter
5. All of the RTOS modules are coded in Machine Language.

3.1.1.1 IDAPS - IDAPS System Monitor

The IDAPS Monitor operates as a System Task, as defined by RTOS conventions.
A general discussion of the characteristics of System Tasks is found in the
RTOS Reference Manual, Section 6.6.

The Protected TCB for IDAPS is defined in the RTOS Module TCBTAB, as described
in Section 4.0.

The first group of cells within IDAPS, starting at label IDAPS, make up the
Unprotected TCB. These cells all initially contain zero, except for the
Logical Unit assignments.

Labels IGO, IRE, and ID1000 share the same location at the first executable
instruction in IDAPS. IGO and IRE are the startup and restart points, as
contained in the Protected TCB. ID1000 is the internally-used label for the
start of the monitor loop, which makes up the bulk of the program.

The first instruction in the monitor loop is an SVC 2 call for an Interval
Wait of .1 second. This delay determines the basic monitoring rate of
IDAPS, which is 10 cycles per second. As long as IDAPS is allowed to remain
active, the program never terminates, but continues to execute the monitor
loop at the rate of 10 times per second.

Every time through the loop, IDAPS reads the Keyboard/Trackball Multiplexor
to sense any operator actions at the terminal. If no actions have been
taken, control goes to label ID1500. Starting at label ID1300, the Multi-
plexor words are examined for keyboard input. If the "Return to Master" key
has been pressed, subroutine RMSUB is called. RMSUB searches TCBTAB and,
using SVC 10, cancels any tasks whose names end with "PS", "SF", or "SM".
When this is complete, RMSUB starts MASTER and returns to the monitor loop.
The value of any other key is stored in KBCH, if KBCH does not already
contain a non-zero value. Before storing, lower-case letter codes are
converted to upper-case. Starting at label ID1350 the Multiplexor words are
examined for trackball input. If TBFLAG is not already non-zero and there
are both x and y values available, then TBFLAG, TBX, and TBY are set appro-
priately.

At label ID1500, IDAPS begins actions related to the 360 interface. If the
interface status, as indicated by FLG360, is not "down" or "ready", control
goes to label ID2180. TIMR1 is a counter which controls the rate of "down"
and "ready" processing and is currently set to take action at 15 second
intervals. If the interface is "down" an attempt is made to initialize it,
then send and receive a "HELLO" message. If the interface is "ready", an
attempt is made to send and receive a "HELLO" message. In either case, if
the actions are successful, FLG360 is set to "ready" and "360 STATUS: READY"
is displayed at the console. If the actions are unsuccessful, FLG360 is set
to "down" and "360 STATUS: DOWN" is displayed. Control goes to label ID3000.
At label ID2180, "busy-wait" processing is done. TIMR3 controls the rate of
this processing at three second intervals. An attempt is made to read an
"Operation Complete" message. When this is successful, the End-of-Operation
Handler, EOPS, is called with an SVC 6.

At label ID3000, the current time is sampled at .7 second intervals, using
SVC 2. If the time is not the same as the previously displayed value and
the value of seconds ends in zero or five, it is displayed at the console.

At label ID3500, if the value of seconds is zero, UTIME, the user's elapsed time in minutes, is incremented; and the current value is displayed.

At label ID4000, a branch is made back to label ID1000 to make the next pass through the monitor loop.

3.1.1.2 DGDVR, DGDCB - Display Generator Driver and DCB
DGDVR and DGDCB handle write operations to the Anagraph Display Generator. See the RTOS Reference Manual, Section 6.5, for a general discussion of drivers and DCB's.

DGDVR checks the specified buffer limits for the 2048-byte maximum that the Display Generator can accept. If this is exceeded, the Illegal Function code "C000" is returned. If the Display Generator status indicates "device unavailable", code "A000" is returned. DGDVR then checks for "buffer empty" status from the Display Generator. If this is not indicated after 65,000 tries, the "unrecoverable error" code "8400" is returned.

When all tests are passed, DGDVR commands the Display Generator to be ready for data, then writes to the buffer with a Write Block instruction.

3.1.1.3 MXDVR, MXDCB - Keyboard/Trackball Multiplexor Driver and DCB
MXDVR and MXDCB handle I/O operations to and from the Anagraph KB/TB Multiplexor. See the RTOS Reference Manual, Section 6.5, for a general discussion of drivers and DCB's.

Before any I/O operation, the Multiplexor is commanded to go into "Transfer Mode". This "shuts off" the keyboard and trackball and makes the multiplexor's buffer accessible to the program.

For the write operation, only a single halfword is sent to the Multiplexor.

For the read operation, the contents of the buffer (up to 16 halfwords) is read into core, and a count parameter is returned to the caller.

After each operation, the Multiplexor is commanded to go into "Normal Mode", so that the keyboard and trackball are enabled.

3.1.1.4 DIDVR, SCDCB, REDCB - DICOMED Driver and DCB's

DIDVR, SCDCB, and REDCB handle I/O operations to and from the DICOMED Scanner and Recorder. See the RTOS Reference Manual, Section 6.5, for a general discussion of drivers and DCB's.

Since the Scanner and Recorder have identical controllers, except for their device numbers, a single driver handles the operations of both. The two devices share a Selector Channel, through which all data transfers pass.

In response to an "8000" function code, DIDVR·senses the status of the Scanner or Recorder and returns the value to the caller. For a read or write request, DIDVR sets up the Selector Channel with the buffer limits, commands the Scanner or Recorder to prepare for a read or write, and commands the Selector Channel to start the transfer. If the Selector Channel has not terminated after an adequate length of time, a malfunction has occurred; and the "unrecoverable error" code "8400" is returned to the caller.

3.1.1.5 IFDVR, IFDCB - 360 Interface Driver and DCB

IFDVR and IFDCB handle I/O operations to and from the 360 computer. See the RTOS Reference Manual, Section 6.5, for a general discussion of drivers and DCB's.

The 360 interface involves three separate devices, each of which produces interrupts under certain conditions. These devices are:

a)   Device "3F", the Interface Controller

b)   Device "60", the 360 Device

c)   Device "F2", the Interface Selector Channel

Access to the Interface, with SVC 1 calls, is made through a "pseudo-device"
with the number "36", thus this is the device number defined in IFDCB. The
driver is functionally divided into four major sections as follows:

Device	Section Label	Description
36	IFDVR	SVC 1 handling and overall control
60	INTDEV	360 Device interrupt handling
3F	INTINF	Interface controller interrupt handling
F2	INTSC	Selector channel interrupt handling

3.1.1.5.1 IFDVR - SVC 1 Handling and Overall Control

As a result of an SVC 1 call, the driver is entered at label IFDVR. The
action taken must be one of the following:

Function Code	Description
88	Initial setup of interface
98	Send attention signal to 360
38	Write to 360
58	Read from 360
F8	Clear out interface

At label DVR100, the function code is fetched and examined. If it is not
'88', control goes to label DVR105. Commands are sent to the Interface to

set conditions to their nominal values, and control goes to label DVR265,
where an Attention signal is sent to activate the 360 program. Control then
goes to label DVR280.

At label DVR105, the function code is further examined. If the function
code is 'F8', certain flags are cleared, and IFDVR terminates. If the
function code is '98', control goes to label DVR265, which is described
above. The only remaining functions are '38' (write) and '58' (read). At
label DVR115, the buffer limits are fetched and checked for legality. If
the function is '58', the Operation Flag is set for "read header", and
control goes to label DVR200. For function '38', the Operation Flag is set
for "write header", and the header words are set up in core. The format of
the header is as follows:

Bytes	Contents
0, 1	Length of message, bytes, including header.
2	Unused.
3	Check byte, exclusive OR of all bytes in message body.

At label DVR200, certain flags are initialized for the coming operation, the
Interface status is checked to make sure it is ready, and control goes to
label DVR270. At label DVR270, a "Control Unit End" (CUE) status signal is
sent to the Interface, and the Allow IS (Initial Selection) flag is set.
The CUE status tells the 360 that the INTERDATA-70 is ready for a transfer.
When the status has been accepted by the 360, an interrupt is generated,
which is sensed and processed by the INTINF section of the driver. From
this point, the phases of the transfer operation are interrupt-driven.

At label DVR280, the program goes into a loop, waiting for a flag (DONFL) to
be set, indicating that the operation has been completed. When DONFL is
set, by one of the interrupt-driven sections, the driver terminates.

3.1.1.5.2   INTDEV - 360 Device Interrupt Handling

The driver section starting at label INTDEV is entered when device '60', the 360 device, produces an interrupt. This happends as a result of the 360 doing an Initial Selection (IS). INTDEV reads from the interface the command which was issued by the 360. If status indications appear normal, control goes to label DEV150, where the command is examined. The following commands are recognized:

Command (Hex)	Description
00	Test I/0
01	Write from 360
02	Read to 360
13	Write done
23	Read done
53	Write done, error
63	Read done, error

At label DEV200, a reply is sent to a "done" command (13, 23, 53, or 63). This reply is a status signal with "device end" and "channel end" bits set. A flag (DONREP) is set to indicate that this is a "done reply", and the Allow IS flag is cleared. When the status signal is accepted by the 360, an interrupt is generated, which is sensed and processed by the INTINF section of the driver. INTDEV now restores the processor to pre-interrupt conditions.

At label DEV440 a read or write operation is initiated. The Selector Channel is set up to read or write the message header, then commanded to start the transfer. When the header transfer is complete, an interrupt is generated, which is sensed and processed by the INTSC section of the driver. INTDEV now restores the processor to pre-interrupt conditions.

3.1.1.5.3 INTINF - Interface Controller Interrupt Handling

The driver section starting at label INTINF is entered when device '3F', the
Interface Controller, produces an interrupt. This happens as a result of
the 360 accepting a status signal which the INTERDATA-70 has sent.

At label INF200, if the Allow IS (Initial Selection) flag is set, the Interface
is commanded to allow IS. This clears the way for the 360 to issue a command,
which will be sensed and acted on by the INTDEV section of the driver.

At label INF210, if the status signal sent was a "done reply", actions are
taken to close up the completed operation. This involves setting Interface
conditions back to their nominal values and setting DONFL to indicate to the
IFDVR section that the operation is completed.

When INTINF completes its processing, it restores the processor to pre-
interrupt conditions.

3.1.1.5.4 INTSC - Selector Channel Interrupt Handling

The driver section starting at label INTSC is entered when device 'F2', the
Selector Channel, produces an interrupt. This happens as a result of a data
transfer, as defined by the Selector Channel buffer limits, going to completion.

At label SC175, if a message header was just transferred, the Selector
Channel is set up to transfer the body of the message, then commanded to
start the transfer. The processor is then restored to pre-interrupt conditions.

At label SC300, the body of a message has completed transfer. If the transfer
was a read to the INTERDATA-70, the Check Byte is checked against the Exclusive
OR of all of the bytes of the message body. An ending status signal is sent
to the 360. If the transfer was good, the status is "device end" and "channel
end". If the transfer was bad, the status is "device end", "channel end",
and "unit exception". The Allow IS flag is set. At this point, conditions

are such that the 360 can issue a "done" command, or request a repeat of the transfer if a problem was detected. In either case, the resulting interrupt will be sensed and processed by the INTDEV section of the driver. INTSC now restores the processor to pre-interrupt conditions.

3.1.2 IDAPS Tasks

Tasks are the basic operating entities of the IDAPS system. IDAPS tasks fall into four categories:

1. Operator parameter specification tasks.
2. Master menu and sub-menus.
3. Special function operations.
4. Subordinate tasks.

A task is made up of a main program and all subroutines called by that program. It is labelled, with specific information attached to it, such as load options, storage requirements, logical unit assignments, common blocks, etc.

TASKS

NAME	DESCRIPTION
ALTPS	Alter operator parameter specification
AREPS	Area operator parameter specification
AUTPS	Auto Scale operator parameter specification
AVEPS	Average operator parameter specification
CENPS	Center/Radius operator parameter specification
CONPS	Convolutional Filter operator parameter specification
COORSF	Coordinates special function
CPRSM	Classification/Pattern Recognition Sub-menu
DEPPS	Dependent Alter operator parameter specification
DIFPS	Difference operator parameter specification
DISPS	Distance operator parameter specification
DPSPS	Feature Analysis subordinate task
DT85PS	Data Table output subordinate task
DTEPS	Data Table Edit operator parameter specification
DTIKPS	Integer data table edit subordinate task
DTRDPS	Real data table edit subordinate task
EOPS	End of operation task
ERASSF	Erase special function
EXTPS	Extract operator parameter specification
FDSPS	File Delete/Save/Dump operator parameter specification
FEAPS	Feature Analysis operator parameter specification
FFTPS	Fast Fourier Transform parameter specification
FGNSM	Function Generation Sub-menu
FGNPS	Filter Generation parameter specification
FOPSM	Filter Operators Sub-menu
FOUPS	Fourier Operator parameter specification
FRAPS	Frame operator parameter specification

TASKS

NAME	DESCRIPTION
GOPSM	Geometric Operations Sub-menu
GSASM	Gray Scale Adjustment Sub-menu
HANPS	Hand-drawn shapes operator parameter specification
HDCPS	HDC operator parameter specification
HDC1PS	HDC subordinate task
HDC2PS	HDC subordinate task
HISTSF	Histogram special function
IDPSM	Image Data Presentation Sub-menu
IDT	Initialize data tables on disk
IFFPS	Inverse Fast Fourier Transform operator parameter specification
IHL	Initialize Help messages on disk
IIOSM	Image Input/Output Sub-menu
IMAPS	Image generation operator parameter specification
IMASM	Image Analysis Sub-menu
INSPS	Insert operator parameter specification
INTRPS	Interceptor data table subordinate task
INVPS	Invert operator parameter specification
IPL	Initialize default parameter list on disk
ISOPS	Isogram operator parameter specification
ITDPS	System to Display operator parameter specification
ITFPS	System to Recorder operator parameter specification
ITPPS	System to Printer operator parameter specification
ITTPS	System to Tape operator parameter specification
LABPS	Label operator parameter specification
LISTSF	List files special function
LOGIN	Login operator parameter specification
LOGPS	Logout operator parameter specification
LOGTAP	Output procedure for logout operator
MAGPS	Magnify operator parameter specification
MASTER	Master menu
MATPS	Math operator parameter specification

TASKS

NAME	DESCRIPTION
MIMSM	Manual Image Modification Sub-menu
MLFSM	Math/Logic Functions Sub-menu
OVEPS	Overlay operator parameter specification
PCTGPS	Pseudocolor color specification task
PGSVSF	Gray Scale Wedge special function
PIDPS	Pseudocolor System to Display operator parameter specification
PIFPS	Pseudocolor System to Recorder operator parameter specification
PITDPS	Pseudocolor subordinate task
PITFPS	Pseudocolor subordinate task
PSDPS	Pseudocolor Scanner to Display operator parameter specification
PSEPS	Pseudocolor Sub-menu
PSFPS	PSF Generator operator parameter specification
PSRPS	Pseudocolor subordinate task
PSTDPS	Pseudocolor subordinate task
PSTFPS	Pseudocolor Scanner to Recorder operator parameter specification
PTDPS	Pseudocolor Tape to Display operator parameter specification
PTFPS	Pseudocolor Tape to Recorder operator parameter specification
PTTDPS	Pseudocolor secondary task
PTTFPS	Pseudocolor subordinate task
QLKSF	Quick look scan special function
REGPS	Register operator parameter specification
RESTSF	Restore display special function
RIDT	Re-initialize data tables on disk
ROTPS	Rotate operator parameter specification
SIMPS	Similarity operator parameter specification
SLICSF	Slice special function
STDPS	Scanner to Display operator parameter specification
STFPS	Scanner to Recorder operator parameter specification
STIPS	Scanner to System operator parameter specification
STOPS	Stonyhurst operator parameter specification
STPPS	Scanner to Printer operator parameter specification
STTPS	Scanner to Tape operator parameter specification

TASKS

NAME	DESCRIPTION
TRAPS	Transpose operator parameter specification
TTDPS	Tape to Display operator parameter specification
TTFPS	Tape to Recorder operator parameter specification
TTIPS	Tape to System operator parameter specification
TTPPS	Tape to Printer operator parameter specification
TTTPS	Tape to Tape operator parameter specification
USEPS	User generator operator parameter specification

3.1.3 IDAPS Subroutines

The IDAPS subroutine library consists of both Fortran and assembly language programs which are used to provide support to the IDAPS tasks. The library is stored on disk file 3C7 with a backup on magnetic tape, and is built and edited by LDR3, the OS Library Loader (section 2.1.5.3.1).

SUBROUTINES

NAME	LANG.	DESCRIPTION
BINHEX	M	Convert a binary number to its hexidecimal ASCII representation
CATOF8	F	Convert 8 ASCII characters to a floating point binary number
CATOFB	F	Convert an ASCII string to a floating point binary number
CBTOA	M	Convert a binary integer to its ASCII representation
CENDIA	F	Center/radius calculator
CFBTOA	F	Convert floating binary to its ASCII representation
CHSRCH	M	Check ASCII string for a specific character
CLEAR	M	Erase master monitor
COLOR	M	Establish display color for master monitor
CONVRT	M	Convert from ASCII to integer
CRDFLD	M	Card Field fetch
CTRANS	M	Transfer string of ASCII bytes
CURSOR	M	Write· and erase cursor on master monitor
DISKIO	M	Disk I/O Control program
DISKR	M	Read record from disk
DISKW	M	Write record to disk
DISLIN	M	Display line of image data
DISTNC	F	Distance calculation
DOLCK	M	Look for dollar sign character in card image
DOT	F	Display dot and plus sign character for data table edit
DTGET	F	Get data table and store binary values
DTINC	F	Read data table from cards and store on disk
DTKBTB	F	Read either keyboard or trackball input for data table edit
DTPUT	F	Generate ASCII data table from binary array
DTRD	F	Read specified data table from disk
DTSRCH	F	Look for address specified data table name

SUBROUTINES

NAME	LANG.	DESCRIPTION
DTWR	F	Write specified data table to disk
ENDFIL	M	Write end of file on tape
ERASE	M	Blank out a display monitor
ERRMSG	M	Error message and user interface handler
FILNFO	F	Get information about specified 360 file
GEOSIZ	F	Calculate geometric information
GOPUT	M	Display generator control
GPL	M	Get parameter list from disk
GRAPH	F	Display integer data table in graphic format
GRAPHF	F	Display real data table in graphic format
GUNINT	F	Color gun intensity for color chart
HELPMS	M	Display designated Help message
HISTOW	M	Display histogram
HLINE	F	Display horizontal line on master monitor
HOLD	F	Pause until 'EXEC' or down arrow received from kb
INCHST	M	Input character string from kb
INFP	M	Input floating point number from kb
INFPT	F	Control program for INFP - checks boundaries
ININ	M	Input image name from kb
ININT	M	Input integer from kb
INMNIN	M	Input monitor number or image name from kb
INOPT	M	Input alphabetic option from kb
INPUTS	M	General purpose kb input routine
INSDP	M	Insert decimal point in ASCII string if needed
INTERP	F	Interpolation routine
IPL	F	Initialize default parameter values
ITOD	F	System to display image transfer
ITOF	F	System to recorder image transfer
ITOP	F	System to printer image transfer
ITOT	F	System to tape image transfer
LABEL	M	Write aphanumeric string on master monitor

SUBROUTINES

NAME	LANG.	DESCRIPTION
LHX	M	Hex constant function
LINK	M	Start up designated task
LND	M	Logical "AND" function
LOGRD	F	Read an 80 character record from log file
LOGWR	F	Write an 80 character record to log file
LOR	M	Logical "OR" function
LSH	M	Logical shift function
LXR	M	Logical "exclusive OR" function
MNINFO	F	Get information about image or specified monitor
MONITR	F	Change ASCII monitor designations to 1-5
MOVR	F	Move three last bytes in a 6 byte field to first three bytes
MREAD	M	Input special monitor buttons
NERASE	M	Selective monitor erase
NLABEL	M	Display ASCII string on specified monitor
NREC	M	Display rectangle on specified monitor
PCTGEN	F	Pseudocolor table input
PTTOD.	F	Pseudocolor tape to display
RCMD	M	Recorder command output subroutine
REC	M	Display rectangle on master monitor
REERR	M	Recorder error handler
REGCOM	F	Compute information for Register operator
REWIND	M	Rewinds mag tape
RFILT	M	Recorder filter selection subroutine
RIDT	F	Reinitialize data table default values
RINIT	F	Initialize recorder
RTRANS	M	Recorder data transfer routine
SCERR	M	Scanner error handler
SCLEAR	M	Clear selective area on master monitor
SCMD	M	Scanner output command subroutine
SQUASH	F	Delete internal blanks from card image
SKIPF	M	Skip files on mag tape
SKIPR	M	Skip records on mag tape

SUBROUTINES

NAME	LANG.	DESCRIPTION
SFILT	M	Scanner filter selection subroutine
SINIT	F	Initialize scanner
SLABEL	M	Display special character
SORT	F	Ascending sort
SPL	M	Store parameter list on disk
SPLINE	F	Spline interpolation routine
SQUASH	F	Delete all blanks from a string of characters
STOD	F	Scanner to display image. transfer
STOF	F	Scanner to recorder image transfer
STOI	F	Scanner to 360 system image transfer
STOP	F	Scanner to printer image transfer
STOT	F	Scanner to tape image transfer
STRANS	M	Scanner data transfer subroutine
TAPEIO	M	Tape read/write handler
TAPEP·	F	Position tape at specified file and record
TAPER	M	Tape read routine
TAPERL	F	Determine number of bytes in tape record
TAPEW	M	Tape write routine
TBKBIN	M	Input number pair from kb or trackball
TBLINK	F	Dummy routine
TPERR	M	Tape error message output
TRACE	F	Front end for trackball trace
TRACK	M	Trackball trace
TTOD	F	Tape to display image transfer
TTOF	F	Tape to recorder image transfer
TTOI	F	Tape to 360 system image transfer
TTOP	F	Tape to printer image transfer
TTOT	F	Tape to tape image transfer
UNPKLN	M	Unpack/pack line of image data
VLINE	F	Display vertical line on master monitor
W360	M	Write/Read/Terminate 360 routine
WAIT	M	Go into RTOS wait mode

3.2 IBM-360 MODULES

3.2.1 Control Program Modules

This section describes the Control Program modules of the IBM-360 IDAPS
system. The relationship between the modules is illustrated in Figure 2-3.
The subroutines which comprise the Control Program modules are given in
paragraph 3.2.3.

3.2.1.1 MAIN, BLOCK DATA, IEXEC, LOGIN - IDAPS Executive

The Executive module consists of four subroutines; MAIN, BLOCK DATA, LOGIN
and IEXEC. This module directs the flow of execution to the remaining
modules of the Control Program and continuously monitors all system and
application software for any malfunctions.

The MAIN routine serves as the entry point for the 360 IDAPS system. Sub-
routine BLOCK DATA provides initialization of the Control Program buffers
and system COMMON areas. For a description of the COMMON areas see paragraph
2.2.2.

The heart of the Executive module is the IEXEC subroutine which is called by
MAIN and then exercises overall control of the 360 IDAPS system. IEXEC
calls subroutine DISKIO to open all the system disk files. After successful
opening of the disk files, IEXEC reads the card input to determine whether
batch or interactive mode of operation is desired. If batch mode is selected,
then all commands are read in from the card reader. If interactive mode is
selected, then IEXEC calls subroutine INTFAC to monitor the ID-70 for all
commands.

When IEXEC receives a command (batch or interactive), subroutine CIR is
called to interpret the command card image. Upon successful interpretation
of the card image, IEXEC calls subroutine INIT to direct execution of the
desired operator. Upon completion of the operator, IEXEC provides hard copy
output to indicate status of the operation and, in interactive mode, notifies
the ID-70 of the completion of the operator.

If IEXEC determines that an error has occurred, subroutine ERRMES is called
to provide hardcopy error messages. In the batch mode execution is aborted,
but in the interactive mode the ID-70 is notified of the error by an error
code. The error codes are given in the publication, Image Data Processing
System (IDAPS) User Manual, Interactive IDAPS, Volume III (TM-HU-037/300/00).

LOGIN provides the capability to reinitialize IDAPS parameter to initial
values or set them to values which were stored on disk by a call to entry
point STORE after the last interactive operation.

3.2.1.2 ITRANS, INTFAC, ECHO -- ID-70 Interface
Interface between the IBM 360 and the ID-70 is handled on the 360 side by
three subroutines. ITRANS is the assembler language routine which uses the
EXCP (Execute Channel Program Access Method) to transmit and receive data
via the Interdata IBM-360 Universal Interface. INTFAC is an assembler
language routine which provides interface between the FORTRAN level routines
and the ITRANS routine. In addition INTFAC monitors the ITRANS routine and
reports error conditions. Subroutine ECHO is used to notify the ID-70 that
the last transmitted data was successfully received.

The types of data transmitted between the two computers fall into five
categories. Handshaking type of data is used to establish and maintain
synchronous operation and continuous monitoring of the status of the other
computer. Command type data is in the form of an 80 byte card image used to
initiate an operator in the 360. Non-image type data (tables, labels, etc.)·
required by operators is sent to the 360 as card images. Image data is in
the form of records containing gray values which are stored one byte per
point. Status type data consists of 80 byte operation complete messages,
file directory, and non-image computed data sent from the 360 to ID-70.

Data transmitted via the interface must be synchronized to avoid a condition where both computers are attempting to read or write, or one is trying to send more data than the other is expecting. For these reasons the types of data transmitted between the two computers are grouped into categories called message types. These message types and their attributes (code, length, initiator and function) are given in Figure 3-1. The messages consist of two parts, a 4-byte header containing the code and the body of the transmitted data.

Subroutine ITRANS opens the interface device, transmits and receives data from the ID-70 by reading and writing to the interface device and checks for and reports error conditions. Whenever either computer senses an error condition or encounters a synchronization problem, an attempt is made to return to the nominal handshaking condition of the ID-70 sending and the 360 echoing type-1 messages.

3.2.1.3 CIR, INTPAR – Command Interpretation

The 80 byte command card images are interpreted, the file name(s) separated and the parameters isolated, interpreted and stored in the parameter array by this module. Subroutine CIR checks the command card image for syntax errors, assigns the operator name an index and stores the file name(s) in the input file array. CIR isolates the command card parameters, checks for correct number of parameters and calls subroutine INTPAR to interpret the parameters. These parameters are then stored in the parameter array and any errors reported to the IDAPS Executive.

Subroutine INTPAR branches to the appropriate logic based on the operator index assigned in CIR. The parameters are then interpreted as expected by the particular operator. Subroutine INTIN is called to interpret integer format parameters and subroutine FLTIN is called to interpret floating point format parameters. Subroutine ALPIN is called to interpret and convert alphabetic parameters. The alphabetic parameters are used to designate

Code	Length	Initiated By	Type	Function
1	32	360 or ID-70	Hello	Maintain handshaking
2	84	ID-70	Start operation	Issue 360 command
3	84	ID-70	Data input	Send non-image data to 360
4	32	360 or ID-70	Image file ready	Notify other computer that image file is to be transmitted.
5	≤2052	360 or ID-70	Image file record	Transmit line of image data
6	32	360 or ID-70	End-of-file	Notify other computer that all data has been transmitted.
7	32	360	Operation complete	Notify ID-70 that the 360 operation is complete.
8	32	360	Goodbye	Currently not used.
9	1644	360	File management	Send File Directory to ID-70.
10	≤1204	360	Computed data	Send non-image data to ID-70.

Figure 3-1.   Interface Messages

options selected, and ALPIN converts an A to an integer value of one, B to
an integer value of two, etc.

3.2.1.4 INIT, INIT2 - Operator Initialization and Control
Subroutines INIT and INIT2 comprise this module which acts as the application
program (operator) executive by providing input parameters and non-image
data to the appropriate application subroutine. Subroutine INIT controls
operators whose indices are less than 35 and subroutine INIT2 controls
operators whose indices are 35 or greater. INIT (INIT2) branches to the
appropriate logic based on the operator index. If non-image data is required
by the operator, INIT (INIT2) reads the data from the card reader in batch
mode or by calling subroutine INTFAC in interactive mode. If data tables
are required by the operator, INIT (INIT2) calls subroutine DATAIN to read
and interpret the data card images. If any preprocessing subroutines are
required by the operator they are called by INIT (INIT2). Subroutine FMR is
called to provide the file management required by the operator. If no
errors are reported by the other modules, INIT (INIT2) calls the application
program for the operator being executed and passes the input parameters and
tables required through the calling sequence. All errors reported by the
Control Program modules or the application program are reported to the IDAPS
Executive.

3.2.1.5 DATAIN - Data Input and Interpretation
The 80 byte data card images are read, interpreted and stored in the data
table array by this module. DATAIN reads the card images from the card
reader in batch mode or by calling subroutine INTFAC in the interactive
mode. Subroutine INTIN is called to interpret integer format data, and
subroutine FLTIN is called to interpret floating point format data. DATAIN
checks the card images for syntax errors, stores the values in the table
array and reports any errors to the operator executive INIT (INIT2).

3.2.1.6 FMR, NEXTPK, DISKSP, FILNFO, FILNAM - File Management

The File management module consists of the five subroutines FMR, NEXTPK, DISKSP, FILNFO and FILNAM. This module provides the bookkeeping associated with naming, storing, indexing and retrieving the system image files. Subroutine FMR is the executive routine of the File Management module and checks for and reports error conditions to the operator executive INIT (INIT2).

Subroutine FMR checks the IDAPS system file directory for the input image file(s) and sets up the input file directory. FMR calls subroutine NEXTPK which determines on which disk pack(s) the input file(s) are stored and assigns an alternate pack whenever possible to the output image file. FMR also assigns secondary output image files where required to alternate packs whenever possible and sets up the output file directory for the operators. In addition FMR assigns scratch files as required to alternate packs whenever possible and sets up the scratch file directory for the operators. The IDAPS system file directory is maintained by FMR, and deletion of files is handled by FMR. FMR checks for and reports error conditions related to file sepcification, file deletion and space availability.

Subroutine NEXTPK determines the disk pack(s) on which the input file(s) are stored and provides subroutine FMR with the alternate pack assignment array to use in assigning output and scratch files in order to reduce contention resulting from accessing two files on the same pack.

Subroutine DISKSP maintains a primary space directory and a deleted space directory which reflect the current disk space utilization. When a file is assigned storage from the primary space, that directory is updated by DISKSP. When a file is deleted, and its space is made available for reuse, DISKSP enters this information in the deleted space directory. Whenever all primary space has been exhausted, DISKSP assigns space from the deleted space directory and updates that directory. Any errors relating to disk space availability are reported to FMR.

Subroutines FILNFO and FILNAM provide support to the other routines in the
File Management module. Given a file name, FILNFO searches the IDAPS system
file directory and returns the size and format parameters for that file.
Given an operator which has used all file names avialable, FILNAM searches
the IDAPS system file directory and returns the name of a file which has
been deleted.

3.2.1.7 DISKIO, DISKW, DISKWF, DISKWP – Disk Input/Output
Storage and retrieval of image data from the disk storage is handled by the
Disk Input/Output module. This module consists of an assembler language
disk subroutine (DISKIO) which opens the disk files and provides read/write
capability through DIRREA and DIRWRT entry points. There are also three
FORTRAN subroutines which provide interface between the FORTRAN level operator
subroutines and the DISKIO routine. DISKW (and its entry DISKR) permits
writing and reading of image data in integer format. DISKWF (and its entry
DISKRF) permits writing and reading of floating point format data. DISKWP
(and its entry DISKRP) permits writing and reading of image data in packed
integer format (4 pixels/word). Any input/output errors which occur while
reading or writing data on disk storage are reported to the calling routine.

3.2.1.8 OPENNL, WRITE9 – Tape Input/Output
The Tape Input/Output module provides the capability to input images from
magnetic tapes and output images to magnetic tapes. This module is designed
to provide compatibility between the ID-70 and the IBM-360 tape drives. Any
input/output errors which occur while reading or writing data on magnetic
tape are reported to the calling routine.

Subroutine OPENNL opens the magnetic tape file for input. Entry point
READNL in the OPENNL subroutine permits sequential reading of the records in
the file and indicates when an end-of-file has been sensed. Entry point
ENDTPI closes the current file and spaces to the beginning of the next file
on the magnetic tape.

Subroutine WRITE9 opens the magnetic tape for output and permits sequential writing of records on the tape. Entry point ENDFIL closes the current file, writes an end-of-file, and positions the tape for opening the·next output file.

3.2.1.9 ERRMES, OPERR - Error Handling

The Error Handling module consists of the two subroutines ERRMES and OPERR. ERRMES handles all ·errors by providing hardcopy printout of the Control Program error messages and calling OPERR to handle the application program errors. OPERR uses the operator index to branch to the appropriate logic and there the hardcopy. printout of the particular error message is provided. The IDAPS Executive notifies the ID-70 of the error code in interactive mode and terminates the run in batch mode.

3.2.2 IDAPS Operators

The IDAPS system is comprised of two types of operators: system operators and application operators.

3.2.2.1 System Operators

The system operators provide the capabilities to input image data to the 360 IDAPS system, output image data from the 360 IDAPS system, delete system files, and transmit non-image data to the ID-70. The system operators, their three-character acronyms, and the subroutines which comprise the operators are given below.

Operator	Acronym	Subroutine
Input from magnetic tape	INP	TTOI
Input from scanner	SCA	STOI
Output to magnetic tape	OUT	ITOT
Output to hardcopy printer	PRI	ITOP

Operator	Acronym	Subroutine
Output to film recorder	FIL	ITOR
Display system file	DIS	ITOD
Delete system file	DEL	FMR
Send computed data to ID-70	SCD	SCD
Login	LOG	LOGIN

3.2.2.2 Application Operators

The application operators provide the capabilities to process images and consist of the image processing application programs. The application operators, their three-character acronyms, and the subroutines which comprise the operators are given below.

Operator	Acronym	Subroutine
Alter gray scale values	ALT	ALTER
Isogram	ISO	ISOGRM
Picture difference	DIF	DIF
Automatic scaling	AUT	SCAL
Extract subframe	EXT	EXT
Insert subframe	INS	INSURT
Label image	LAB	LABL, DSPLIN
Invert image (complement)	INV	INVERT, ALTER
Mathematical operations	MAT	ARITH
Stonyhurst grid	STO	STONYH, STONY1, LLGRID, LTLNGE, PLTA, DDPLT, SLBITS, LIN2
Matrix transposition	TRA	TRNPOS, TRNPS, FLIP
Frame an image	FRA	FRAME, DSPLN, FRAMWR, HISTQW
Average images	AVE	AVERGE

Operator	Acronym	Subroutine
H and D correction	HDC	HDC
Rotate and image	ROT	GEOTRN, SIZOUT, CORAV, PIXMAN, BILAT, IN, OUT
Register an image	REG	GEOTRN, SIZOUT, CORAV, PIXMAN, IN, OUT, REGCOM
Hand drawn shapes	HAN	HANS
Fast Fourier Transform	FFT	FRX2V, FRXFM, WINDOW
Inverse Fast Fourier	IFF	FRX2V, FRXFM, WINDOW
Fourier Filter	FOU	FFTFIL, FRX2V, FRXFM, WINDOW
Filter Generation	FGN	FGN, WTGEN, PROFIL, ADJUST, BESJ, ELIPSE, FIT, LAGRAN, PINV, PROT, SET
Window an image	WIN	WINDOW
Overlay an image	OVE	MASK
Feature analysis	FEA	SEGMNT
Area	ARE	AREA
Border	BOR	BORDER, LINE
PSF Generator	PSF	PSFGEN, LAGRAN, FRX2V, FRXFM
Convolutional filter	CON	CONFIL
Similarity	SIM	SIMIL
Dependent alter	DEP	DEPALT
Test Image Generation	IMA	IMAKER

3.2.3 Control Program Subroutines

The subroutines which comprise the IDAPS 360 Control Program are listed below in alphabetical order. Subroutine entry points are listed and their parent routine given. The language in which each module is written is given as FORTRAN (F) or machine language (M).

Name	Language	Description
ALPIN	F	Convert from EBCDIC to binary
ASCEBC	M	Convert from ASCII to EBCDIC
BLOCK DATA	F	Initialize COMMON
CIR	F	Command interpretation
CORE	M	Read/Write buffer with format conversion
DATAIN	F	Data input interpretation
DIRREA (Entry in DISKIO)	M	Direct access read
DIRWRT (Entry in DISKIO)	M	Direct access write
DISKIO	M	Direct access input/output for disk files
DISKR (Entry in DISKW)	F	Read integer format disk files
DISKRF (Entry in DISKWF)	F	Read floating point format disk files
DISKRP (Entry in DISKWP)	F	Read packed integer format disk files
DISKSP	F	Allocate disk space
DISKW .	F	Write integer format disk files
DISKWF	F	Write floating point format disk files
DISKWP	F	Write packed integer format disk files
EBCASC	F	Convert from EBCDIC to ASCII
ECHO	F	Echo ID-70 messages
ELTIME	F	Determine elasped time in timer intervals
ENDFIL (Entry in WRITE9)	M	Write end-of-file on output tape
ENDTP1 (Entry in OPENNL)	M	Space to end-of-file on input tape

Name	Language	Description
ERRMES	F	Error analysis (system)
ETIME	F	Determine elapsed time in seconds
FILNAM	F	Reassign file names
FILNFO	F	Retrieve file information
FLTIN (Entry in ALPIN)	F	Convert from EBCDIC to floating point
FMR	F	File Manager
IANDF	M	Logical AND function
IEXEC	F	IDAPS Executive routine
INIT	F	Operator initialization (1-34)
INIT2	F	Operator initialization (35-up)
INSERT	F	Insert a character in a string
INTFAC	M	IBM-360/ID-70 interface control routine
INTIN (Entry in ALPIN)	F	Convert from EBCDIC-to integer
INTPAR	F	Parameter interpretation
IORFU	M	Logical OR function
ITOD	F	360 System to display
ITOP	F	360 System to printer
ITOR	F	360 System to film recorder
ITOT	F	360 System to tape
ITRANS	F	IBM-360/ID-70 data transfer
JGETB	M	Get bits
JPUTB	M	Put bits
LOGIN	F	Reinitialize IDAPS parameters
MAIN	F	Control program entry routine
MOVST	F	Move string of characters
NEXTPK	F	Assign next pack to output file

Name	Language	Description
OPENNL	M	Open non-labeled tape file
OPERR	F	Error analysis (operator)
PACK	F	Pack characters
PCKINT	M	Pack bytes
PCK608	M	Pack six bits in one byte
PL4	F	Pack line of data (4 characters/ line)
READNL (Entry in OPENNL)	M	Read non-labeled tape file
SCD	F	Send computed data to ID-70
SIZFIL	F	Determine output file size
STIME	M	Initialize timer
STOI	F	Scanner to 360 system
STORE (Entry in LOGIN)	F	Save restart parameters on disk
TTOI	F	Tape to 360 system
UNPACK	F	Unpack characters
UNPINT	M	Unpack bytes
WRITE9	M	Write 9 track non-labeled tape

4.0 OPERATING SYSTEM MODIFICATIONS

4.1 INTERDATA-70 RTOS MODIFICATIONS

This section describes the modifications which have been made to certain
standard RTOS modules to suit the requirements of IDAPS.

4.1.1 EXEC - RTOS Executive

In the standard EXEC, processing of an SVC 10 (Cancel Task) request
causes the message "name..... CANCELLED" to be output to the Teletype.
This is not desirable for IDAPS, since the frequently-used "Return to
Master" logic in the IDAPS Monitor uses SVC 10 to get rid of any IDAPS
tasks which are active.

Following label CAN221 (location 1064 Hex.) in EXEC, this instruction
causes the message to be printed:

        BAL   R1,LOGCAN

To disable the printing, the two halfwords of this instruction have been
replaced by two NO-OP (0200 Hex.) instructions. The change was made in
this way so that the subsequent code would remain in the same locations
and so that the message could be easily enabled again if desired.

4.1.2 RLSTAB - Reentrant Subroutine Library Table

The constant defined at label LIBSIZ (loc. 38 Hex.) must equal the
number of disk cylinders allocated to the Task Library File. The current
size of this file is 276 (114 Hex.) cylinders, thus the definition is as
follows:

        LIBSIZ   DC   X'114'

4.1.3 LODER - RTOS Task Loader

The standard LODER prints the message "name LOADED" on the Teletype each
time a task is successfully loaded into core. This is undesirable for

the IDAPS tasks, since in normal operations controlled from the terminal
console, a great many task loads take place.

A means of deleting the message, without doing away with it entirely,
has been added by making LODER sensitive to the "OPTIONS" bit which
deletes the initiate (RDY) and terminate (EOJ) messages (see RTOS Reference
Manual, Section 2.7). Thus, if the bit is set, as it is for all IDAPS
tasks, the "LOADED" message will not appear.

The logic added to LODER fetches the OPTIONS word from the task's Protected
TCB, tests for the "delete message" bit, and if it is set bypasses
output of the "LOADED" message. The modified area starts at label N06
(loc. A14 Hex.) and is as follows:

```
N06       EQU    *
          LH     W1,TCBSAV            GET TCB ADDR.
          LH     W1,26(W1)            GET OPTIONS WORD
          THI    W1,X'0040'           TEST BIT
          BNZ    N07                  BRANCH IF SET
          LM     W1,LOMSG             GET LOADED MSG
          STM    W1,MSG1
          BAL    RTN3,LOGMSG          PRINT IT
N07       EQU    *
```

4.1.4   TCBTAB - Task Control Block Table

The modifications of TCBTAB have been made to accommodate the System
Task definition of the IDAPS Monitor.

The first change is the addition of External Definitions of the IDAPS
entry points as follows:

```
EXTRN IDAPS,IGO,IRE
```

The TCB list has been modified to remove TCB02 from the list of available TCB's. Its modified form is:

```
TCBLST   EQU . *
         DC    X'0505'
         DC    0
         DC    TCB03
         DC    TCB04
         DC    TCB05
         DC    TCB06
         DC    TCB07
```

TCB02 has been set up to be the Protected TCB for the IDAPS Monitor, as follows:

```
TCB02    EQU    *
IDPTCB   DC     C'IDAPS '              TASK NAME
         DC     X'8000'               CURRENT STATUS
         DC     0,0,0,0               MEM. PROTECT PATTERN
         DC     IDAPS                 UNPROTECTED TCB ADDR.
         DC     X'7E00',IGO           INITIAL PSW
         DC     4                     PRIORITY
         DC     X'FF00'               POINTERS
         DC     X'3018'               OPTIONS
         DC     0                     FILLER
         DC     X'FFFF'               POINTERS
         DC     X'7E00',IGO           CURRENT PSW
         DC     X'FE00',IRE           RESTART PSW
         DC     0,0,0,0,0,0           FILLER
```

4.1.5 LFCDVR - Line Frequency ·Clock Driver

Some of the new drivers and the modified Card Reader Driver operate for
appreciable lengths of time with interrupts locked out. Under heavy
load conditions, especially when many decks of cards are being read,
this causes a loss of some of the interrupts from the Line Frequency
Clock. The result is a loss of time in the Time of Day processing in
LFCDVR. To compensate for the· loss of interrupts and make Time of Day
more accurate, LFCDVR has been modified to work as though the Line
Frequency Clock produced 119 interrupts per second rather than the usual
120. The Modified constant definition is as follows:

```
COUNT    EQU    119
```

4.1.6 TTYDVR - Teletype Driver

The modification to TTYDVR was made to incorporate a correction received
informally from INTERDATA, relating to output of a line consisting of
only a Carriage Return. The corrected area starts at label TTY220 (loc.
72 Hex.), and is as follows:

```
TTY220   EQU    *
         CLHR   R9,RA        IS IT CR ONLY
         BE     RD030        BRANCH IF YES
         SIS    R9,1         DONT INCLUDE CR
```

4.1.7 CRDVR - Card Reader Driver

The standard CRDVR reads successive bytes from the card under interrupt
control, as the Card Reader produces an interrupt when each byte of data
becomes available. However, some of the new drivers may keep interrupts
locked out for a long enough time that bytes from the card reader could
be lost and errors result. CRDVR has been modified to read the entire
card, with interrupts locked out. This modification follows label SAVE,
at location 134 Hex.:

```
     B      CRISR1          GO BACK FOR NEXT 2 BYTES
```

The standard CRDVR will loop indefinitely if the Card Reader becomes
unavailable or remains busy after a read sequence has started. It has
been modified to time-out and give an error indication for such an
occurence. The modified area starts at label BSY (loc. 112 Hex.) and is
as follows:

```
     BSY        EQU    *
                LIS    R9,0
     BSY1       SSR    RD,R8          GET STATUS
                BTC    1,ERR          BRANCH IF UNAVAILABLE
                BNLS   BSY2           BRANCH IF NOT BUSY
                SIS    R9,1
                BNZS   BSY1           TRY AGAIN
                B      ERR            TIME-OUT
     BSY2       EQU    *              OK
```

4.1.8 DSC200 - Disk Driver

The standard DSC200 produced a serious blow-up if the ending address of
a random-access read or write was not specified to be an odd number.
The situation was remedied by adding 1 to an even-numbered ending address.
The modified area follows label RW at loc. 96 Hex.:

```
     BNM    T3          BRANCH IF SEQUENTIAL
     LH     RF,6(R3)    GET ENDING ADDR.
     NHI    RF,X'FFFE'  CLEAR LAST BIT
     AIS    RF,1        ADD ONE
     STH    RF,6(R3)    STORE IT
```

4.1.9 DCBC6L - Disk DCB

DCBC6L contains the definitions of the three disk files which are dedicated
to RTOS usage:

File	Description
20C6	RTOS Core Image
21C6	Task Library Index
22C6	Task Library

The standard allocation of two cylinders for file 20C6 was too small·for
the IDAPS version of RTOS, so it was increased for four cylinders. This
made it necessary to.change the starting cylinders of files 21C6 and
22C6. The size of file 22C6 has been increased to accommodate the IDAPS
tasks. These three files are defined at labels FILE20, FILE21, and
FILE22 (loc. 148 Hex.) in DCBC6L as follows:

FILE20	DC	0	STARTING CYLINDER
	DC	3	ENDING CYLINDER
	DC	0	CURRENT POSITION
	DC	-1	KEYS
FILE21	DC	4	STARTING CYLINDER
	DC	4	ENDING CYLINDER
	DC	0	CURRENT POSITION
	DC	-1	KEYS
FILE22	DC	5	STARTING CYLINDER
	DC	X'118'	ENDING CYLINDER
	DC	0	CURRENT POSITION
	DC	-1	KEYS

4.2 IBM-360 OS MODIFICATIONS

The ID-70 and IBM-360 are connected on-line via the INTERDATA UNIVERSAL
360 INTERFACE. The interface is a non-supported device to the 360 and
several modifications to the IBM-360 Operating System (OS) must be made
to effect the communication link between the two computers. The following
paragraphs discuss these modifications.

4.2.1 Attention Service Routine

The Attention Service Routine (IGC254) is a user-written type 1 transient
SVC routine. This routine is invoked by the following call:

        SVC   254

This results in the interface (front-end) being enabled. Upon receipt
of an interrupt from the ID-70 (via the interface) OS gives control to
the appropriate interrupt handling routine as specified by the attention
table in the Input/Output (I/O) Supervisor. Entry FEOEORTN in IGC254
handles this interrupt and posts a "complete" in the event control
block for the front-end. IGC254 must be linked into SYS1.SVCLIB.

4.2.2 System Generation Inputs

At OS system generation (SYSGEN) time it is necessary to include a unit
control block for the device address of the ID-70/IBM-360 interface
device. This is done in Stage I of the SYSGEN by using the IODEVICE and
UNITNAME macro instructions. The unit address assigned for this instal-
lation is 360, and the macros are coded as follows:

        IODEVICE   UNIT=DUMMY,ADDRESS=360
        UNITNAME   NAME=360,UNIT=360

It is also necessary to incorporate the user-written SVC into OS. This
is accomplished at SYSGEN time by using the SVCTABLE and SVCLIB macro
instructions. The SVC code used for this installation is 254, and the
macros are coded as follows:

        SVCTABLE   SVC-254-TI-SO
        SVCLIB   PDS=SYS1.SVCLIB,MEMBERS=(IGC00254)

In addition the attention table of the I/O Supervisor must be modified
by entering a V-type address constant referencing the routine which is
to handle the interrupt. The attention table resides in the IECIOS
module, and the entry point FEOEORTN must be entered in the table. This
is accomplished in Stage II of the I/O generation by coding the following
instructions:

```
./    CHANGE      NAME=IECIOS,LIST=ALL
      DC      `   V(FEOEORTN)            INTERDATA TERMINAL
```

5.0   SYSTEM STARTUP AND RECOVERY

5.1   INTERDATA-70

5.1.1   INTERDATA-70 Control Console

In order to accomplish the various phases of startup and recovery of the
IDAPS Terminal, it is necessary to become familiar with the operations of the
INTERDATA-70 Control Console. A complete description of the Control Console
and its operations is contained in the INTERDATA-70 User's Manual, Publication
No. 29-261, Chapter 7 (Chapter 6 in 29-261R01). For convenience, those
operations which are generally sufficient to accomplish the procedures
described in this document are summarized here.

The Console Lock must be in the ON position before any Control Console
operations can be accomplished.

5.1.1.1   Enter HALT Mode

To enter HALT Mode:

    a)   Turn Rotary Switch to PSW.
    b)   Put RUN Switch UP.
    c)   Press EXE Switch.

Data Display Lights should change and WAIT Light should come on. If this
does not happen, processor is hung; press INT Key to free it and enter HALT
mode.

5.1.1.2   Memory Read

To read contents of memory cells, first enter HALT Mode as described above,
then:

    a)   Set Rotary Switch to ADR/MRD.

    b)   Put SGL Switch UP.

c) Enter starting memory address in Data Switches.

d) Press EXE Switch. Memory address will be echoed in lower Data
    Display Lights.

e) Put SGL Switch DOWN.

f) Press EXE switch. Data read from memory is displayed in lower Data
    Display Lights. Upper Data Display Lights show current memory
    address, plus two.

Step f may be repeated to display successive halfwords from memory. The
referenced memory address is automatically incremented by two each time EXE
is pressed.

5.1.1.3 Memory Write

To write data into memory, first enter HALT Mode as described above, then:

a) Set Rotary Switch to ADR/MRD.

b) Put SGL Switch UP.

c) Enter starting memory address in Data Switches.

d) Press EXE Switch. Memory address will be echoed in lower Data
    Display lights.

e) Set Rotary Switch to OFF/MWR.

f) Put SGL Switch DOWN.

g) Enter data to be written in Data Switches.

h) Press EXE Switch. Data stored in memory is displayed in lower Data Display Lights. Upper Data Display Lights show address stored into, plus two.

Steps g and h may be repeated to store into successive halfwords of memory. The memory address to store into is automatically incremented by two each time EXE is pressed.

5.1.1.4 Start Program Execution

To start program execution, first enter HALT Mode as described above, then:

a) Set Rotary Switch to ADR/MRD.

b) Put SGL Switch UP.

c) Enter program starting address in Data Switches.

d) Press EXE Switch. Memory address will be echoed in lower Data Display Lights.

e) Set Rotary Switch to PSW.

f) Put RUN Switch DOWN.

g) Press EXE Switch. Program execution will begin.

Occasionally, due to particular combinations of Program Status (PSW) conditions and pending interrupts, an interrupt and unpredictable results will prevent normal startup of program execution. The condition may be cleared as follows:

a)   Starting at location 32_x, store the following values:

Location(Hex.)	Value
32	0000
34	0000
36	Program starting address

b)   As described above, start execution at location 32_x.

5.1.2   50 Sequence Boostrap Loader

Most startup operations begin with the execution of a small loader program
which is referred to as the 50 Sequence because of its starting location at
50_x. The 50 Sequence for RTOS operation is as follows:

Location(Hex.)	Value(Hex.)
50	D500
52	00CF
54	4300
56	0080

In addition, the cells starting at location 78_x must contain specific values.
The usual values for RTOS operation are:

Location(Hex.)	Value(Hex.)
78	0294
7A	C600

Deviations from these values are noted under particular phases of startup and
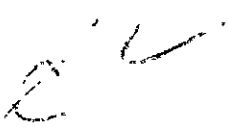recovery.

5.1.3 <u>RTOS Startup From Disk</u>

When it is desired to load RTOS into core from the disk and start its operation, the following steps should be performed:

a) Verify that the 50 Sequence is in core and that locations 78_x and $7A_x$ contain the values 0294_x and $C600_x$.

b) Put the RTOS Disk Bóotstrap paper tape in the Teletype Paper Tape Reader. Position the tape so that the blank section is over the read mechanism. · Turn the Power Switch to the LINE position. Put the reader switch at the AUTO position.

c) Start program execution at location 50_x.

d) Momentarily press the reader switch to the MANUAL START position. The tape will begin moving through the reader.

e) When RTOS begins printing on the Teletype, move the reader switch to the MANUAL STOP position.

f) Enter the current Time of Day and Date as follows:

        TIME hhmmss
        DATE mm/dd/yy

    To obtain the current Time or Date, type RDTI or RDDA.

g) Start operation of the IDAPS Monitor as follows:
        START IDAPS

h) Start normal IDAPS execution as follows:
        STAR LOGIN

Note: Infrequently, the result of the above operations is a LOAD
ERR condition upon the system's trying to load LOGIN. The following
sequence of inputs should enable the system to recover:

LOAD	ASM
LOAD	FTN
DELE	FTN
DELE	ASM
STAR	LOGIN

5.1.4 RTOS Startup From Load Module Tape

Startup from an RTOS Load Module magnetic tape, produced as described in
Section 2.1.5.3.3, is accomplished as follows:

a) Mount the Load Module tape on tape drive #1.

b) Verify that the 50 Sequence is in core and that location 78_x
    contains the value 0294_x.

c) Put the INTERDATA REL Loader paper tape in the Teletype Paper Tape
    Reader. Position the tape with the blank leader over the read
    mechanism. Turn the Power Switch to the LINE position. Put the
    reader switch at the AUTO position.

d) Start program execution at location 50_x.

e) Momentarily press the reader switch to the MANUAL START position.
    The tape will begin moving through the reader and will require
    several minutes to read. At times a teletype reader inconsistency
    causes the reader to pause at each end of record gap, in which
    case the MANUAL START switch must be pressed to continue the read.
    This does not affect the accuracy of the read.

f)   When the paper tape reading is complete, set location 78_x to the
      value $85A1_x$.

g)   Start program execution at address $FC00_x$, as described in Section
      5.1.1.4; however, before pressing the EXE switch to start execution,
      put all of the Data Switches UP. (Otherwise, REL Loader will
      attempt to skip past files on the magnetic tape.) REL Loader will
      now load RTOS into core from the magnetic tape.

h)   Start program execution at address $2D0_x$. This is the nominal
      startup point for RTOS.

i)   Using the RTOS REPL Command, restore locations 78_x and $7A_x$ to their
      normal values as follows:

          REPL 78,294,C600

j)   If there are any in-core corrections or modifications to be made to
      RTOS, enter them using the REPL Command.

      Currently the in-core modifications consist of the following commands:

      REPL 25CA,2652
      REPL 28F0,5053,2020
      REPL 2F6E,24CF
      REPL 2C60,4300,62BA
      REPL 62BA,48AA,0000,98DA,DED0,2CAE,4300,2C66

k)   Allocate the disk files, using the ALLO Command, as follows:

      ALLO 00C6,119,158
      ALLO 01C6,159,168

```
ALLO 02C6,169,178
ALLO 13C6,179,17B
ALLO 14C6,17C,17E
ALLO 15C6,17F,181
ALLO 16C6,182,183,1600
ALLO 17C6,184,187,1700
ALLO 18C6,188,197,1800
ALLO 00C7,0FF,0FF
ALLO 01C7,100,197
ALLO 03C7,060,0D8
ALLO 04C7,0D9,0FE,0400
ALLO 10C7,000,05F
```

l)   Verify proper file allocation with the following commands:

```
LIST C6
LIST C7
```

m)   Assign the Roll-out files as follows:

```
ASSI *LODER,3,13C6,4,14C6,5,15C6
```

n)   Establish the Task Common area as follows:

```
TCOM 3
```

If other than three blocks are to be allocated to Task Common, use the appropriate number.

If it is desirable at this point, the newly constructed version of RTOS in core may be written to the disk, as described in Section 5.1.5. Or, the

system may be allowed to go through a period of checkout before writing it to
the disk.

Normal operation may be initiated as follows:

    a)   Enter the current Time of Day and Date:

            TIME hhmmss
            DATE mm/dd/yy

    b)   Start operation of the IDAPS Monitor:

            STAR IDAPS

    c)   Start normal IDAPS execution:

            STAR LOGIN

        See note in Section 5.1.3, paragraph h.

5.1.5 Write RTOS to Disk

At any point, the currently operating version of RTOS in core may be written
to the disk, such that subsequent use of the RTOS Disk Bootstrap (see Section
5.1.3) will bring in this version.

In general, RTOS should be written to the disk following checkout of a new
version, or when in-core corrections have been made.

The procedure is as follows:

    a)   Disable the IDAPS Monitor:

            CANC IDAPS

Note: Do not DELE IDAPS

b), With the MAP Command, determine what tasks, if any, are in core.

c) Delete all tasks from core with the command:

    DELE name

d) As described in Section 5.1.1.4, start program execution at location
    $2D8_x$. This is the startup point from which RTOS writes itself to
    the disk.

Resume normal operation as follows:

a) Start operation of the IDAPS Monitor:

    STAR IDAPS

b) Start normal IDAPS execution:
    STAR LOGIN

    See note in Section 5.1.3, paragraph h.

5.1.6 Regeneration of Systems Task Library
The Systems Task Library and its Index may be regenerated from their backup
copies (see Section 2.1.5.5) in the following manner:

a) Start the paper tape established task version of TUT:

    STAR TUT,102

    See note in Section 5.1.4, paragraph e.

b)   Mount the Index mag tape on tape drive #1 and the Systems Task
      Library mag tape on tape drive #2.

c)   In TUT:

      COPY 85,21C6
      COPY 95,22C6
      INDE 62
      END

      Note: If an IO error of the type '8485' or '8495' occurs during a
      COPY operation, mount the tape on the other tape drive, CONT TUT,
      and try the COPY on the other tape drive.

5.2   IBM-360

In order to activate the IBM-360 system for interactive IDAPS the INTERACTIVE
control deck listed below must be submitted at the IBM-360 operations
control desk.

```
//   A11,300,10),1HNTSVSE3000USERNAME,MSGLEVEL=(1,1),CLASS=L,TYPRUN=HOLD
//ID70   EXEC IDAPSI,TIME.GO=300
//FORT.SYSIN DD *
C     MAIN ROUTINE FOR INTERACTIVE IDAPS
      CALL IEXEC
      STOP
      END
/*
```

```
GO.SYSIN DD *
/*
```

There should only be one copy of this control deck in use, since submitting a copy while another is in the system will hang up the system.

5.3   INTERACTIVE SYSTEM STARTUP, RECOVERY AND SHUTDOWN.

5.3.1   <u>System Startup</u>

In order to bring up the interactive IDAPS System the following steps are necessary:

1)   Submit the INTERACTIVE control deck described in paragraph 5.2 at the IBM-360 control desk.

2)   When the 360 operations personnel advise that the IDAPS 360 program is "UP", place the ID-70 on line to the IBM-360. This is accomplished by placing the two ID-70/IBM-360 interface switches in the "ON LINE" position. These switches are located in the lower rear of the ID-70 console cabinet.

3)   The ID-70 Monitor should report a "READY" status for the 360. If the status continues to report a "DOWN" condition, it will be necessary to bootstrap the ID-70 Real Time Operating System (RTOS) into core as discussed in paragraph 5.1.3. If the Monitor continues to report a "DOWN" condition, contact IDAPS personnel.

5.3.2   <u>System Recovery</u>

Certain conditions may require the user to invoke system recovery procedures. These conditions and the appropriate user response for each condition are discussed in this section.

Condition: ID-70 Monitor reports an error in an operation.

Response: Look up the error message in the table of message codes, correct
any input errors and repeat the operation. If the error persists, save
all applicable information such as the operator attempted, input file
name(s), options selected, etc. and notify IDAPS personnel.

Condition: ID-70 Monitor reports a "DOWN" status for the 360.

Response: Wait two minutes to permit both computers to cycle through error
checking and attempt to return to nominal status. This should result in a
"READY" status for the 360 and an error message. Look up the error message
in the table of message codes, correct any input errors and repeat the
operation. If the error persists, save all applicable information such as
the operator attempted, input file name(s), options selected, etc. and notify
IDAPS personnel.

If the "DOWN" status remains more than two minutes, check with the 360
operations personnel to determine if the IDAPS 360 program is still executing.
If the program has aborted, then the interactive IDAPS system must be restarted
as described in paragraph 5.3.1. If, however, the IBM-360 system is down,
then the ID-70/IBM-360 interface switches should be placed in the "OFF LINE"
position until the 360 operations personnel advise that the IBM-360 system is
running and the IDAPS 360 program is "UP".

Condition: ID-70 Monitor reports "BUSY" status for the 360 for an excessively
long time for the operator being executed.

Response: Check with the 360 operations personnel to determine if the IDAPS
360 program is still executing. If the program has aborted, then the inter-
active IDAPS system must be restarted as described in paragraph 5.3.1. It
will be necessary to place the ID-70/IBM-360 interface switches in the "OFF
LINE" position until a "DOWN" status is reported for the 360. This must be
done in order to reset the 360 status bit.

If, however, the IBM-360 system is down, then the ID-70/IBM-360 interface switches should be placed in the "OFF LINE" position until the 360 operations personnel advise that the IBM-360 interface system is running and the IDAPS 360 program is "UP".

5.3.3 System Shutdown

When the day's interactive operation of IDAPS is complete, the system should be shutdown by placing the ID-70/IBM-360 interface switches in the "OFF LINE" position and notifying the 360 operations personnel to take the IDAPS 360 program "DOWN".

6.0 ADDING OPERATORS

The IDAPS system is structured to permit new operators (application programs) to be added with a minimum of knowledge of the system software. However, certain restrictions and conventions must be followed when adding new operators. The purpose of this section is to acquaint someone considering adding a new operator with these restrictions and conventions. Each computer system will be discussed separately.

6.1 IBM-360 SYSTEM

6.1.1 Operator Restrictions

The restrictions placed on new operators are given below:

1) All subroutines in an operator should be coded in IBM-360 compatible FORTRAN.

2) The total storage requirement for an operator must not exceed 164,000 bytes.

3) All input and output of image data must be handled through the IDAPS Input/Output module.

4) All input, output and scratch files must reside in the IDAPS disk storage.

5) An operator is limited to five input files, two output files, three secondary output files, and three scratch files.

6) Secondary files must be integer format and are destroyed by any subsequent operator which requires secondary files.

7) . Integer data which is to be written into an integer (pixel) file must not lie outside the zero to 255 range.

8) Secondary and scratch files must be the same size as the output file(s).

9) All files are limited to a maximum of 2048 lines and 2048 columns.

10) All parameters required by an operator must be input via the command card image and, except for the input file names, passed to the operator through the calling sequence.

11) All parameters on the command card image are limited to a maximum field width of eight characters.

12) No more than two tables may be input to an operator and these must be input in IDAPS data table format via card images and passed through the calling sequence. (Reference Image Data Processing System (IDAPS) User Manual, Batch IDAPS, Volume II (TM-HU-037/200/00) Section 3.10)

13) Tables may be input in integer or floating point format, but they must be limited to a maximum of 256 values.

14) Non-image output data values may be returned to the ID-70 by storing the values in the computed data buffer (see Paragraph 2.2.2 for a description of the common block CDBUF) and calling the SCD operator to transmit the values.

15) All error conditions must be assigned a five digit error code (the first two digits must be the operator index) by the operator and returned to the IDAPS Executive module.

6.1.2 Using the Disk Input/Output Module

All image input and output must be handled through the Disk Input/Output
module (see Paragraph 2.2.1.7). The routines in this module permit random
access of the data in the system files. These routines handle two types of
data; four-byte integer format and single precision floating point format.
The calling sequence for each of these routines is the same and is illustrated
by the call to DISKR given below:

        CALL DISKR (IFD(1,1),LINE,IBUF,IERR)
        IF(IERR .NE. 0) RETURN

where:

        IFD(1,1)   =   File name
        LINE   =   Index of line
        IBUF   =   Buffer containing line of data
        IERR   =   Error condition returned by DISKR

The check on the error condition results in non-zero codes being returned to
the IDAPS Executive module.

To write integer format data, use the DISKW routine. Since integer data is
stored in packed format (four bytes or pixels per word), the calling program
must assure that the input buffer, which must be in four-byte integer format,
does not contain any values which are negative or greater than 255. To
write floating point format data, use the DISKWF routine. Since floating
point data is stored in single precision format, the calling program must
assure that the input buffer is single precision format.

To read integer files, use the DISKR routine. A four-byte integer buffer
must be provided because DISKR always returns data in this format. DISKR
may be used to read a floating point file, but it will read, truncate and

return the values in four-byte integer format. To read a floating point
file, use the DISKRF routine. A single precision floating point format
buffer must be provided because DISKRF always returns data in this format.
DISKRF may be used to read an integer file, but it will read, convert and
return the values in single precision format. Since these two routines read
a complete line of data, the buffers must provide space for 2048 values.

The file name in the calling sequence to these routines must be an integer
variable. The file management module handles all the bookkeeping associated
with naming and maintaining the system files. This module sets up the input
(IFD), output (OFD) and scratch (SFD) file directories, and these are avail-
able to the new operator through the COMMON block FILEDR. These directories
must be included in the new operator subroutine(s) by coding the following:

        COMMON/FILEDR/IFD(6,5)OFD(6,5),SFD(6,3)
        INTEGER IFD,OFD,SFD

The first index of each directory accesses the file parameters as illustrated
for IFD:

        IFD(1,I)   =   Name (4 character file name)
        IFD(2,I)   =   File format (1 = integer, 2 = floating point)
        IFD(3,I)   =   Number of records (lines)
        IFD(4,I)   =   Number of columns per record
        IFD(5,I)   =   Relative address of first line
        IFD(6,I)   =   Lo gical unit number

The second index of each directory refers to the first, second, etc. file.
The output file is organized as follows:

        OFD(I,1)   =   First output file
        OFD(I,2)   =   Second output file

OFD(I,3)   =   First secondary file
OFD(I,4)   =   Second secondary file
OFD(I,5)   =   Third secondary file

There are three scratch files and their parameters are organized as outlined above and stored in the SFD array.

6.1.3   IDAPS System Modifications

To add a new operator to the IDAPS IBM-360 system, entries must be made in five places in the source code and in the overlay control statements for the binary library. Each set of these entries will be discussed separately. For the purpose of illustration, it is assumed that the operator being added is named NEW, requires seven input parameters, one data table, two input files, creates a floating point output file and uses a floating point scratch file. Furthermore, it is assumed that the last operator entered in the directory is LAS.

The first set of entries is made in the Executive module in subroutine BLOCK DATA as illustrated below:

    0   4HLAS ,2, 110000, 4HNEW ,7, 201001, 4H   ,0,0000000,

See Paragraph 2.2.2 for a discussion of the entries above. The NEW operator entries in BLOCK DATA are entered in IOP (I,44), so the NEW operator is assigned an operator index of 44. This index is used to branch to the appropriate parameter interpretation logic in INTPAR, the appropriate operator initialization logic in INIT and the appropriate error message in OPERR.

The second set of entries is made in the Command Interpretation module in subroutine INTPAR. These entries are based on the types of input parameters which the new operator expects. For this example, it is assumed that the first two parameters are integer values; the third, sixth and seventh are

alphabetic options; and the fourth and fifth are floating point values. The
INTPAR entires for the new operator would be made as illustrated below:

```
44   CONTINUE
C---
C              NEW OPERATOR (NEW)
C---
     CALL INTIN (8,RL(1),PARM(1))
     CALL INTIN(8,RL(2),PARM(2))
     CALL ALPIN(RL(3),PARM(3))
     CALL FLTIN(9,RL(4),PARM(4))
     CALL FLTIN(8,RL(5),PARM(5))
     CALL ALPIN(RL(6),PARM(6))
     CALL ALPIN(RL(7),PARM(7))
     RETURN
```

The 44 CONTINUE statement corresponds to the address to which the computed
GO TO will branch based on the operator index (in this case 44). The first
two calls are to INTIN which interpret the first two parameters as integers
values. The three calls to ALPIN result in the third, sixth and seventh
parameters being interpreted as alphabetic options (A=1, B=2, etc.). The two
calls to FLTIN result in the fourth and fifth parameters being interpreted
as floating point values. In each of these calling sequences, RL(I) is the
Ith parameter as passed from the CIR subroutine, and PARM(I) is the Ith
parameter properly interpreted to be returned to the CIR subroutine. The
value 8 in the INTIN and FLTIN calling sequences indicates the parameter to
be interpreted is in an 8-byte field.

The third set of entries is made in the Operator Initialization and
Control module in subroutine INIT (INIT2). These entries are based on the
file and table requirements of the NEW operator and are made as illustrated
below:

```
44  CONTINUE
C---
C        NEW OPERATOR (NEW)
C---
     NL = IPARM(1)
     NC = IPARM(2)
     CALL DATAIN(X,256,1,MODE,IERR)
     IF(IERR .NE. 0) RETURN
     CALL FMR(IFILE,NUMF,NOP,NAME,NL,NC,LAST,IERR)
     IF(IERR .NE. 0) RETURN
     CALL NEW(NL,NC,IPARM(3),PARM(4),PARM(5),IPARM(6),IPARM(7),X,IERR)
     RETURN
```

In this example, it is assumed that the input data table is floating point
format and consists of 256 values. In the calling sequence to DATAIN, the
fourth variable indicates batch or interactive mode of operation and is set
by the Executive module. The first three variables must be set by the
programmer. The first parameter is the array in which DATAIN returns the
input table. INIT provides two arrays for floating point tables (X and Y)
and two arrays for integer tables (IX and IY). The second parameter indicates
the exact number of values in the table (256 in this case). The third
parameter indicates in which format the table is input (0 for integer, 1 for
floating point). If DATAIN returns a non-zero value in IERR, then the Data
Input and Interpretation module has detected an error and a return from INIT
(INIT2) will result in proper handling of the error.

In the calling sequence to FMR, all variables except the fifth and sixth (NL
and NC) which give the output file size are set by INIT (INIT2). In this
example, the first two parameters are integers (IPARM values) and represent
the number of lines (NL) and number of columns (NC) in the output file
generated by the NEW operator. The call to FMR invokes the file manager
which assigns an output file for the operator NEW, updates the file directory

with this entry, and sets up the input, output and scratch file directories
for the NEW operator. If FMR returns a non-zero value in IERR, then the
file manager has detected an error and a return from INIT (INIT2) will
result in proper handling of the error. The call to NEW passes the properly
interpreted parameters (integer values for the alphabetic and integer inputs
are stored in the IPARM array and floating point values are stored in the
PARM array) to the subroutine NEW. An additional parameter IERR is passed
to subroutine NEW to permit any error conditions which that subroutine may
detect (or any error codes returned by calls to the disk read/write subroutines)
to be flagged by a non-zero error code. The error handling logic will trap
and report any errors so flagged.

The fourth set of entries is made in OPERR. These entries result in on-line
printout of the error messages corresponding to the error codes returned by
NEW. The entries for two error codes (44001 and 44002) flagged in NEW are
given below:

```
   44 CONTINUE
C---
C         NEW OPERATOR (NEW)
C---
      IF(IERR .EQ. 44001) WRITE(6,44001)
      IF(IERR .EQ. 44002) WRITE(6,44002)
44001 FORMAT('0   TYPE-1 ERROR IN NEW')
44002 FORMAT('0   TYPE-2 ERROR IN NEW')
      RETURN
```

In this example, it is assumed that there are two types of error conditions
in the NEW operator. Whenever the first type is encountered, IERR is set to
44001. Similarly, the second type of error is denoted by setting IERR=44002.

The final set of entries is made in the overlay structure control statements.
In this example, it is assumed that the NEW operator consists of three sub-
routines. NEW is the control routine, SETUP provides initialization of
data, and CALC performs some repetitive calculations. The entries for the
NEW operator should be made after the LAST operator and before the REGION
overlay as illustrated below:

```
OVERLAY A
    INSERT LAST
OVERLAY A
    INSERT NEW
    INSERT SETUP
    INSERT CALC
OVERLAY R(REGION)
    INSERT DISKW
```

See Paragraph 2.2.4 for a complete discussion of the overlay structure for
IDAPS.

In order to incorporate into the IDAPS system the entries which have been
discussed, it is necessary to execute the USERMOD catalogued procedure. The
JCL setup necessary to invoke the USERMOD procedure is illustrated below.
The FORTRAN source modules and the overlay structure control statements are
inserted as shown:

```
//ADDOPER EXEC USERMOD,BIN=USER
//SCRATCH.SYSIN DD *
        SCRATCH VOL=3300=SDC001,DSNAME=USER
/*
//FORT.SYSIN DD *

          .

        o   .

        o

(FORTRAN source modules for MAIN, BLOCK DATA, INTPAR, INIT and OPERR)

        o

        o .

        o

/*
//OVERLAY.INPUT DD *      .

        o

        o

        o

(Overlay structure control statements)

        o

        o

        o

/*
```

In order to incorporate into the IDAPS system the new subroutines (NEW, SETUP and CALC) required by the NEW operator, it is necessary to use the appropriate cataloged procedure ASCLIDAP, FGCLIDAP or FHCLIDAP for each new subroutine. See Paragraph 2.2.6.1 for a discussion of these procedures and how they are invoked.

Caution must be exercised when adding new operators to avoid causing the interactive system to abnormally terminate. The following procedure is suggested in order to minimize the impact on the IDAPS system of adding an operator.

1. Compile and verify correct execution of the subroutine(s) included
    in the new operator on a stand-alone basis.

2. Make the necessary entries in the FORTRAN source modules and the
    overlay structure control statements as previously outlined.

3. Execute the USERMOD procedure as outlined above to incorporate the
    entries into a checkout binary library rather than the IDAPS
    system binary library USER. The two changes in the USERMOD JCL
    required to set up a checkout binary library are BIN=CHKOUT on the
    EXEC card and DSNAME=CHKOUT on the SCRATCH card.

4. Execute the ASCLIDAP, FGCLIDAP and/or FHCLIDAP procedures as
    required to incorporate the subroutines used by the new
    operator into the IDAPS.APPLIC load module library.

5. Execute the IDAPSB catalogued procedure to checkout the new operator
    under the IDAPS system.

6. Verify correct execution of the new operator. Repeat steps 3, 4
    and 5 as needed to incorporate any corrections required to assure
    correct execution.

7. Execute the USERMOD catalogued procedure to incorporate the entries
    into the IDAPS system library USER.

8. Verify correct execution of the new operator in the interactive
    mode. Repeat steps 3, 4, 5, 6 and 7 as needed to incorporate any
    corrections required to assure correct execution.

6.2 INTERDATA-70 SYSTEM

6.2.1 <u>Adding An Interactive Operator To IDAPS</u>

Once an operator has been checked out under Batch IDAPS, it may be implemented as an interactive operator by the following steps:

1) Design the parameter specification display.

2) Modify the sub-menu task for the category under which the new operator falls.

3) Code and check out the parameter specification task and store it as part of the system task library.

4) Define default values for all parameters.

5) Enter appropriate HELP messages for all input parameters.

6) If data tables are used, enter default values for them, and modify the on-line editing task.

7) If any subordinate Interdata subroutines are developed as part of the tasks, add them to the systems library after they have been checked out.

In the following sections, when the term "add the task to the system" is used, the reference is to the following sequence of operations, input from the teletype:

1) REWInd 1C6

2) STARt FTN to compile the parameter specification task

3)   If a subordinate subroutine not on the systems library
     is to be used:
     STARt FTN or STARt ASM to compile the subordinate
     subroutine

4)   REWInd 1C6
· 5)* STARt TETC
6)   REWInd 2C6
7)   STARt TUT
8)   In TUT: INSErt 2C6
9)   In TUT: END

*The TETC card deck has a leading card in the form

    ESTA TASKID

Where TASKID is the name of the parameter specification task as used in the
sub-menu task (see paragraph 6.2.3). Also, if one or more subordinate
subroutines were compiled to 1C6 in Step 3, a LINK 1C6 card should be inserted
for each subroutine after the LOAD card in the TETC deck.

6.2.2 Designing The Parameter Specification Display

The form shown in Figure 6-1 may be used to design the parameter speci-
fication display. Note that the heading, INTERACTIVE IDAPS, always appears
and need not be coded. The subheading is of the form:

    CATEGORY – OPERATOR NAME (OPERATOR CODE)

The remainder of the display is generally free form, but should follow the
general format of the sample parameter specification shown in Figure 6-2.

DISPLAY NAME _____ CODE ⬚⬚⬚⬚⬚⬚ PROGRAMMER _____

INTERACTIVE IDAPS

Figure 6-1. Interactive IDAPS Coding Form

INTERACTIVE IDAPS

GRAY SCALE ADJUSTMENT -- ALTER (ALT)

OPTION | PARAMETER DESCRIPTION

INPUT FILE NAME

DATA SOURCE
    A - SYSTEM DEFAULT (ALT)
    B - CARDS

Figure 6-2.   Sample Parameter Specification

6.2.3 Modifying A Sub-Menu Task

To add an operator name to a sub-menu display, it is necessary to modify the
appropriate sub-menu task. The sub-menu task names are as follows:

MASTER MENU CATEGORY	TASK NAME
A-IMAGE INPUT/OUTPUT	IIOSM
B-FUNCTION GENERATION	FGNSM
C-GRAY SCALE ADJUSTMENT	GSASM
D-MANUAL IMAGE MODIFICATION	MIMSM
E-FILTER OPERATIONS	FOPSM
F-GEOMETRIC OPERATIONS	GOPSM
G-CLASSIFICATION/PATTERN RECOGNITION	CPRSM
H-MATH/LOGIC FUNCTION	MLFSM
I-IMAGE ANALYSIS	IMASM
J-PSEUDOCOLOR	PSEPS
K-IMAGE DATA PRESENTATION	IDPSM

An example of a sum-menu program for category GRAY SCALE ADJUSTMENT is shown
in Figure 6-3. The sub-menu task must be modified as follows:

1.   Increment by one the second dimension bounds of the array definition
     for PNAME (INTEGER PNAME (2,8) in Figure 6-3).

2.   Add a DATA statement for the corresponding subscript of PNAME. The
     DATA statement should initialize the PNAME location to the name of the
     new paramter specification task, left justified. The name of the task
     is, by convention, the operator code followed by the letter PS.

```
GRAY SCALE ADJUSTMENT SUBMENU (GSASM)

      INTEGER PNAME(2,7)
      DATA(PNAME(I,1),I=1,2)/6HALTPS /
      DATA(PNAME(I,2),I=1,2)/6HDEPPS /
      DATA(PNAME(I,3),I=1,2)/6HAUTPS /
      DATA(PNAME(I,4),I=1,2)/6HINVPS /
      DATA(PNAME(I,5),I=1,2)/6HMATPS /
      DATA(PNAME(I,6),I=1,2)/6HHANPS /
      DATA(PNAME(I,7),I=1,2)/6HHDCPS /
      CALL CLEAR
      CALL COLOR(3)
      CALL LABEL(36,0,21,2,21HGRAY SCALE ADJUSTMENT)
      CALL COLOR(2)
      CALL LABEL(72,0,25,0,25HIMAGE PROCESSING OPERATOR)
      CALL LABEL(-2,24,15,0,15HA - ALTER (ALT))
      CALL LABEL(-2,24,25,0,25HD - DEPENDENT ALTER (DEP))
      CALL LABEL(-2,24,23,0,25HC - AUTOMATIC SCALE (AUT))
      CALL LABEL(-2,24,27,0,27HD - INVERT GRAY SCALE (INV))
      CALL LABEL(-2,24,14,0,14HE - MATH (MAT))
      CALL LABEL(-2,24,27,0,27HF - HAND DRAWN SHAPES (HAN))
      CALL LABEL(-2,24,26,0,26HG - H-D CORRECTION (HDC)          )
      CALL LABEL(-2,0,8,0,8HOPTION =)
      CALL INOPT(1HG,264,72,0,K,I,IFLAG,55)
      CALL LINK(PNAME(1,I))
      END
```

Figure 6-3 Sample Sub-menu Program

3. Add a call to Subroutine LABEL to display a description of the new
    operator and its code on the displayed menu. Be sure to use the next
    available alphabetic option (H, in Figure 6-3).

4. Change the first parameter (the legal option limit) in the call to
    INOPT to the alphabetic option used in 3 above. Also increment the
    second parameter (the line number of the echoed option) of the INOPT
    call by 24. (In Figure 6-3, the call would be charged to CALL INOPT
    (1HH, 288, 72, 0, K, I, IFLAG, 55))

When the submenu task has been changed, add the task to the system.

6.2.4. Coding The Parameter Specification Task

Figure 6-4 shows a typical parameter specification task, which can be used
to demonstrate the general format to be followed for parameter specification
tasks. The general format can be divided into six parts:

A. Default parameters. The default parameter(s) must have been
    previously stored on the default disk file as described in paragraph
    6.2.6. In the parameter specification task, any default parameter
    must have local storage defined for it of length (maximum no.
    characters + 1) (IP1 and IP2) and a DATA statement which initializes
    the storage to blanks. The default values in ASCII are picked up
    by a call to GPL and displayed by calls to LABEL. After all new
    parameter values have been input, a call to SPL must be made in
    order to store them on the disk.

B. Default file name. If it is desired for the input file name to be
    defaulted to the last generated 360 file, the name of that file is
    found by a call to FILNFO with parameter 6 set to 0. If the error

```
. ALTER        (ALT)   PARAMETER SPECIFICATION TASK(ALIPS)          PAGE 0001
       IMPLICIT INTEGER*2 (I-N)
C
C
       DIMENSION IPJ(2),KARD(42),JHRD(42)
       DIMENSION IDM(2)
C
       DATA KARD/0,2,4HALT ,2H (,210,2H) ,4HL 1 ,324X'2020'/
       DATA JARD/0,3,40+X'2020'/
C
       CALL GFL (16,1,IF2)
       CALL CLEAR
       CALL COLOR(3)
       CALL LABEL (36,0,35,3,3SHGRAY SCALE ADJUSTMENT - ALTER (ALT))
       CALL COLOR (2)
       CALL LABEL (72,0,31,0,31HOPTION    PARAMETER DESCRIPTION)
       CALL LABEL (-2,64,15,0,15HINPUT FILE NAME)
       CALL LABEL (-2,64,11,0,11HDATA SOURCE)
       CALL LABEL (-1,80,25,0,25HA - SYSTEM DEFAULT (ALT ))
       CALL LABEL (-1,80,9,0,9HB - CARDS)
       CALL COLOR (3)
       CALL HLINE (88,0,300)
       CALL VLINE (72,160,57)
       CALL COLOR (2)
       CALL FILNFO(IP1,IM1,IDM,IDM,IDM,0,JFLAG)
       IF(JFLAG .EQ. 0)GO TO 50
       CALL EPRINT(J?,12HNO FILES AVAILABLE)
       CALL LINK(CHARACTER)
       GO TO 9999
    50 CONTINUE
       CALL LABEL (96,8,4,0,IP1)
       CALL LABEL (120,8,1,0,IP2)
   100 CONTINUE
       CALL INIM (96,8,IP1,IFLAG,30)
       CALL FILNFO(IP1,IDM,IDM,IDM,IDM,1,JFLAG)
       IF(JFLAG .EQ. 0)GO TO 150
       CALL EPRINT(23,2HTHIS FILE NOT AVAILABLE)
       GO TO 100
   150 CONTINUE
       GO TO (200,100,300),IFLAG
   200 CONTINUE
       CALL INOPT(1H?,120,8,0,K2,IP2,IFLAG,43)
       GO TO (300,100,300),IFLAG
   300 CONTINUE
       CALL SPL (16,1,IP2)
       CALL CONVRT(L,IE2,K2,2)
       KARD(6) = IP1(1)
       KARD(7) = IP1(2)
       WRITE (3,2000) KARD(2),(KARD(J),J=3,23)
  2000 FORMAT (1X,I2,1X,21A2)
       CALL WRGD (84,KARD,KS)
       IF (KS EQ 0) GO TO 400
C
```

Figure 6-4 Sample Parameter Specification Program

```
 ALTER          <ALI>     PARAMETER SPECIFICATION TABK<OLTES>          PAGE 0002
C     ERROR ON START OPERATION
C
      GO TO 800
 400  CONTINUE
C
C.... DATA INPUT
C
      GO TO (1,15,2,3),K?
 1    CONTINUE
C
C.... SYSTEM DEFAULT
C
      KS = 600
      DO 650 K=1,100
      CALL DIRD(4HALT ,K,JARD(3))
 1001 FORMAT (400?)
      WRITE (3,2001) JARD(2),(JARD(J),J=3,42)
 2001 FORMAT (1X,I3,1X,400?)
      CALL U360 (S4,JARD,KS)
      IF (KS.EQ.0) GO TO 600
C
C.... ERROR ON DATA TRANSFER
C
      GO TO 800
 15   CONTINUE
C
C.... CARD INPUT
C
      KS=600
      CALL DTINC(4HALT ,0,J)
      GO TO 1
 600  CONTINUE
C
C.... CHECK FOR I
C
      CALL DOLCK (JARD(3),KPOL)
      IF (KPOL.EQ.1) GO TO 20
 650  CONTINUE
      KS = 200
      GO TO 880
      GO TO 20
 2    CONTINUE
C
C.... KEYBOARD INPUT
C
      KS = 300
      GO TO 800
 3    CONTINUE
C
C.... TRACKBALL INPUT
C
      KS = 400
```

Figure 6-4 Sample Parameter Specification Program (Cont'd)

```
ALTER       <ALT>    PARAMETER SPECIFICATION TASK<ALTPS>          PAGE 0003
          GO TO 800
  20   CONTINUE
C
C..... RETURN TO MASTER MENU WHILE 360 PERFORMS ALTER OPERATION
C
          CALL TRN360 (KS)
C
C.... PAUSE FOR CARD READER PROBLEM
C
  700  CONTINUE
          PAUSE 4
          GO TO 1
C
C..... ERROR
C
  800  CONTINUE
          WRITE (3,1000) KS
 1000 FORMAT (1X,7HERROR =,I5)
          GO TO 20
 9999 CONTINUE
       END
```

Figure 6-4 Sample Parameter Specification Program (Cont'd)

flag is returned non-zero, there are no 360 files and an appropriate
message is displayed by a call to ERRMSG. The name of the default
input file is displayed by a call to LABEL.

C.   Input parameters. A parameter value may be input from the keyboard in
      the form of an image file name, a character string, an alphabetic
      option, an integer, or a floating point number. Pairs of numbers
      representing coordinates on an image may be input from the keyboard or
      from the trackball. A file name is entered by a call to ININ and its
      validity is checked by a call to FILNFO. If the error flag is returned
      non-zero, an appropriate error message is displayed via the subroutine
      call to ERRMSG. The up arrow/down arrow/EXECUTE flag IFLAG is used to
      determine the next statement to execute. Care should be taken when
      writing a parameter specification task that the path the EXECUTE key
      takes does not leave any local parameter storage area blank.

D.   Card image generation. The card image (KARD), which is sent to the 360
      to initiate execution of the 360 operator, is dimensioned 84 bytes and
      defined in a DATA statement. The first two halfwords must contain a 0
      and a 2. The variable parameters must be stored in the card image
      separated by commas, and the input file name must be enclosed in paren-
      theses. It is good practice to code a printout of the card image for
      checkout purposes.

E.   360 Operation. To initiate 360 execution of the operator, the card
      image is sent to the 360 by a call to W360. The third parameter to
      W360 is checked for any 360 detected error and execution terminated if
      an error is detected. A call to TRM360 must be the last executed
      statement of the parameter specification task.

F.   Data table input. An operator may require one or more data tables to
      be sent to the 360. They may be input either from the default tables

stored on disk (see paragraph 6.2.8) or from punched cards. If card
input is chosen by the user, a call to DTINC is used to input values
from cards and store them on disk. Then repeated calls to DTRD will
read the card images off disk into a buffer (JARD) set up previously in
a DATA statement. These card images are sent to the 360 by calls to
W360 until a dollar sign is recognized by subroutine DOLCK. If the
default table option is chosen, the data table is read directly from
the disk by calls to DTRD.

6.2.5 Subordinate Subroutines

If a subordinate subroutine is required by the parameter specification
program and it has not been previously stored on the systems library file,
it may be advantageous to insert the subroutine on the systems library.
Care should be taken that the subroutine is completely checked out, however,
before it is permanently stored. The OS Library Loader LDR3 is used to add
subroutines to the library (see paragraph 2.1.5.3.1). Put a scratch tape on
tape drive #1 and the library tape on #2. A new subroutine is added to the
library by the following commands:

```
                STAR LDR3
    In LDR3:    RE B
                EO 8
                CO B08
                DU 908
                RE 8
                RE D
                DU 80D
                END
```

The tape on tape drive #1 should be saved as the current library tape.

6.2.6 Default Parameters

With the exception of file names, default values should be stored on the disk for all parameters to be input from the keyboard. The values are initialized by task IPL (Figure 6-5) which must be modified in the following manner:

1.   Determine the next available operator number. In the parameter specification task this will be the index used in calls to GPL and SPL (see Paragraph 6.2.4, Part 1).

2.   Change the DATA statement corresponding to the operator number to include the operator code and its default parameters in the format:

        OPC   DP1,DP2,....,DPN,

     NOTE: Several dummy data statements, indicated by the name "NON", are provided at the end of IPL for convience in adding default parameters. Note also that the last default value must be followed by a comma.

3.   Add task IPL to the system. The next time LOGIN is executed with a cold start option, the default parameters will be stored on the disk.

6.2.7   Help Messages

Help messages should be provided for all input parameters used in the parameter specification task. These messages are three-card (80 characters/card) definitions of each parameter. They must be punched and inserted in the data deck for task IHL. Their order in the deck determines the index used in the calls to the input routines in the parameter specification task (see paragraph 6.2.4, part C). After the new cards are inserted in the deck, start IHL from the teletype, with the printer on, and an up-to-date listing of all help messages will be produced.

```
IPL - INITIALIZE PARAMETER LISTS FOR IDAPS SYSTEM                : PAGE 0001

      INTEGER*2 IFUF(80)
      INTEGER*2 IPAR(40,60)
      DATA(IPAR(I,1),I=1,40)/  'S-D A.A.C.B.A.64.64.8.0.8.2.
     *                                    '/
      DATA(IPAR(I,2),I=1,40)/  'S-T A.A.C.B.A.0.0.2047.2047.8.0.0.2.0.
     *                                    '/
      DATA(IPAR(I,3),I=1,40)/  'S-C A.A.C.B.A.0.0.2047.2047.8.0.8.A.A.
     *A.0.0.
      DATA(IPAR(I,4),I=1,40)/  'S-C A.A.C.B.A.0.0.8.0.8.32.32.
     *                                    '/
      DATA(IPAR(I,5),I=1,40)/  'SCA A.A.C.B.A.0.0.2047.2047.8.0.8.
     *                                    '/
      DATA(IPAR(I,6),I=1,40)/  'T-D 1.0.1.1.0.1.
     *                                    '/
      DATA(IPAR(I,7),I=1,40)/  'T-T 1.0.1.1.0.
     *                                    '/
      DATA(IPAR(I,8),I=1,40)/  'T-F 1.0.1.1.A.A.A.C.0.0.
     *                                    '/
      DATA(IPAR(I,9),I=1,40)/  'T-P 1.0.1.1.32.32.
     *                                    '/
      DATA(IPAR(I,10),I=1,40)/ 'INP 1.0.1.1.2048.2048.
     *                                    '/
      DATA(IPAR(I,11),I=1,40)/ 'DIS 1.1.1.0.
     *                                    '/
      DATA(IPAR(I,12),I=1,40)/ 'OUT 1.1.2.0.
     *                                    '/
      DATA(IPAR(I,13),I=1,40)/ 'FIL 1.1.1.A.A.A.A.0.0.
     *                                    '/
      DATA(IPAR(I,14),I=1,40)/ 'PRI 1.1.32.32.
     *                                    '/
      DATA(IPAR(I,15),I=1,40)/ 'E-F B.8.0.
     *                                    '/
      DATA(IPAR(I,16),I=1,40)/ 'ALT B.
     *                                    '/
      DATA(IPAR(I,17),I=1,40)/ 'ISO 1.1.A.10.25.51.76.102.127.153.178.2
     *04.229.255.                         '/
      DATA(IPAR(I,18),I=1,40)/ 'FID A.1.0.1.1.1.
     *                                    '/
```



```
IPL - INITIALIZE PARAMETER LISTS FOR IDAPS SYSTEM                PAGE 0003

      DATA(IPAR(I,52),I=1,40)/ 'FGN A.A.0.0.A.0.0.1.0.A.128.A.5.
     *                                    '/
      DATA(IPAR(I,53),I=1,40)/ 'FOU B.A.
     *                                    '/
      DATA(IPAR(I,54),I=1,40)/ 'NON
     *                                    '/
      DATA(IPAR(I,55),I=1,40)/ 'NON
     *                                    '/
      DATA(IPAR(I,56),I=1,40)/ 'NON
     *                                    '/
      DATA(IPAR(I,57),I=1,40)/ 'NON
     *                                    '/
      DATA(IPAR(I,58),I=1,40)/ 'NON
     *                                    '/
      DATA(IPAR(I,59),I=1,40)/ 'NON
     *                                    '/
      DATA(IPAR(I,60),I=1,40)/ 'NON
     *                                    '/
      N=60
      DO 100 I=1,N
      DO 200 K=1,40
      IBUF(K)=IPAR(K,I)
      IBUF(K+40)=IBAS(K,I)
  200 CONTINUE
      IFLAG=1
      CALL DISK(1,IBUF,160,IFLAG)
  100 CONTINUE
      END
```

Figure 6-5 Parameter List Initialization

6.2.8 <u>Default Data Tables</u>

Any data table used by the operator must have default values stored on the disk. This is done by punching the values on 80 character cards in the format:

$$NAM = Val1,Val2,Val3,....,ValN\$$$

Floating point tables must have floating point values on the cards. They may be in E format, and the FORTRAN convention i*v, to indicate value v is to be repeated integer i times, is recognized. There is a maximum of 256 values per table. After the cards are inserted in the data deck, start IDT from the teletype.

In order to be able to use the data table editing features of IDAPS, the new table name should be entered in task DTEPS (see IDAPS TASKS listing). This is done by adding a call to LABEL to display the operator description, table name (the operator code is used by convention), and minimum and maximum number of values to be allowed in the table. When the change is in the deck, add DTEPS to the system.

APPENDIX

IBM-360 SYSTEM DEVELOPMENT TOOLS

.1.0. .CATALOGUED PROCEDURES   :

1.1 .IDAPS

```
//IDAPS     PROC BIN=USER
//LKED      EXEC PGM=IEWL,REGION=250K,PARM=(XREF,LET,LIST,OVLY)
//SYSLIB  DD DSN=SYS1.FORTLIB,DISP=SHR,UNIT=SYSDA
//        DD DSN=IDAPS.APPLIC,DISP=SHR
//        DD DSN=IDAPS.CONTROL,DISP=SHR
//SYSLMOD DD DSN=&GOSET(MAIN),UNIT=SYSDA,DISP=(NEW,PASS),
// SPACE=(1024,(30,10,1),RLSE)
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=&SYSUT1,UNIT=SYSDA,SPACE=(1024,(100,10),RLSE),
// DCB=BLKSIZE=1024
//SYSLIN DD DSN=&BIN,DISP=SHR,UNIT=SYSDA,VOL=SER=SDC001
//        DD DDNAME=SYSIN
//GO        EXEC PGM=*.LKED.SYSLMOD,COND=(4,LT,LKED),REGION=250K
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT07F001 DD SYSOUT=B
//DISKF011 DD DSN=BT11,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC001,
// SPACE=(512,(5120)),DCB=(DSORG=DA,BLKSIZE=512)
//DISKF012 DD DSN=BT12,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC001,
// SPACE=(1024,(6144)),DCB=(DSORG=DA,BLKSIZE=1024)
//DISKF013 DD DSN=BT13,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC001,
// SPACE=(2048,(24576)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF014 DD DSN=BT14,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC001,
// SPACE=(2048,(8192)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF015 DD DSN=BT15,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC001,
// SPACE=(2048,(2048)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF021 DD DSN=BT21,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC002,
// SPACE=(512,(5120)),DCB=(DSORG=DA,BLKSIZE=512)
//DISKF022 DD DSN=BT22,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC002,
// SPACE=(1024,(6144)),DCB=(DSORG=DA,BLKSIZE=1024)
//DISKF023 DD DSN=BT23,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC002,
// SPACE=(2048,(30720)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF024 DD DSN=BT24,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC002,
// SPACE=(2048,(8192)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF025 DD DSN=BT25,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC002,
// SPACE=(2048,(2048)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF031 DD DSN=BT31,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC003,
// SPACE=(512,(5120)),DCB=(DSORG=DA,BLKSIZE=512)
//DISKF032 DD DSN=BT32,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC003,
//. SPACE=(1024,(6144)),DCB=(DSORG=DA,BLKSIZE=1024)
//DISKF033 DD DSN=BT33,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC003,
// SPACE=(2048,(30720)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF034 DD DSN=BT34,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC003,
// SPACE=(2048,(8192)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF035 DD DSN=BT35,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC003,
// SPACE=(2048,(2048)),DCB=(DSORG=DA,BLKSIZE=2048)
//SYSUDUMP DD SYSOUT=A
```

1.2   IDAPSI

```
//IDAPSI    PROC BIN=USER
//FORT       EXEC PGM=IEYFORT,REGION=250K,PARM=LIST
//SYSPRINT DD SYSOUT=A
//SYSPUNCH DD SYSOUT=B
//SYSLIN   DD DSN=&LOADSET,UNIT=SYSDA,DISP=(MOD,PASS),
// SPACE=(80,(2000,100),RLSE),DCB=BLKSIZE=80
//LKED       EXEC PGM=IEWL,REGION=250K,PARM=(XREF,LET,LIST,OVLY),
// COND=(4,LT,FORT)
//SYSLIB   DD DSN=SYS1.FORTLIB,DISP=SHR,UNIT=SYSDA
//         DD DSN=IDAPS.APPLIC,DISP=SHR
//         DD DSN=IDAPS.CONTROL,DISP=SHR
//SYSLMOD DD DSN=&GOSET(MAIN),UNIT=SYSDA,DISP=(NEW,PASS),
// SPACE=(1024,(30,10,1),RLSE)
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=&SYSUT1,UNIT=SYSDA,SPACE=(1024,(100,10),RLSE),
// DCB=BLKSIZE=1024
//SYSLIN DD DSN=&LOADSET,DISP=(OLD,DELETE)
//         DD DSN=&BIN,DISP=SHR,UNIT=SYSDA,VOL=SER=SDC001
//         DD DDNAME=SYSIN
//GO         EXEC PGM=*.LKED.SYSLMOD,COND=((4,LT,FORT),(4,LT,LKED)),
// REGION=250K
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT07F001 DD SYSOUT=B
//FT04F001    DD UNIT=SYSDA,DSN=&&LABL,DISP=(NEW,DELETE),
//   SPACE=(80,(20,10)),DCB=(RECFM=F,BLKSIZE=80)
//DISKF011 DD DSN=IT11,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC201,
// SPACE=(512,(5120)),DCB=(DSORG=DA,BLKSIZE=512)
//DISKF012 DD DSN=IT12,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC201,
// SPACE=(1024,(6144)),DCB=(DSORG=DA,BLKSIZE=1024)
//DISKF013 DD DSN=IT13,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC201,
// SPACE=(2048,(30720)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF014 DD DSN=IT14,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC201,
// SPACE=(2048,(8192)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF015 DD DSN=IT15,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC201,
// SPACE=(2048,(2048)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF021 DD DSN=IT21,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC202,
// SPACE=(512,(5120)),DCB=(DSORG=DA,BLKSIZE=512)
//DISKF022 DD DSN=IT22,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC202,
// SPACE=(1024,(6144)),DCB=(DSORG=DA,BLKSIZE=1024)
//DISKF023 DD DSN=IT23,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC202,
// SPACE=(2048,(30720)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF024 DD DSN=IT24,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC202,
// SPACE=(2048,(8192)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF025 DD DSN=IT25,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC202,
// SPACE=(2048,(2048)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF031 DD DSN=IT31,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC004,
// SPACE=(512,(5120)),DCB=(DSORG=DA,BLKSIZE=512)
//DISKF032 DD DSN=IT32,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC004,
// SPACE=(1024,(6144)),DCB=(DSORG=DA,BLKSIZE=1024)
//DISKF033 DD DSN=IT33,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC004,
```

```
// SPACE=(2048,(30720)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF034 DD DSN=IT34,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC004,
// SPACE=(2048,(8192)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF035 DD DSN=IT35,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC004,
// SPACE=(2048,(2048)),DCB=(DSORG=DA,BLKSIZE=2048)
//FRONT    DD UNIT=360,DISP=OLD
//SYSSNAP   DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
```

1.3   IDAPSB

```
//IDAPSB    PROC BIN=BAT1
//FORT        EXEC PGM=IEYFORT,REGION=250K,PARM=LIST
//SYSPRINT DD SYSOUT=A       .
//SYSPUNCH DD SYSOUT=B
//SYSLIN   DD DSN=&LOADSET,UNIT=SYSDA,DISP=(MOD,PASS),
// SPACE=(80,(2000,100),RLSE),DCB=BLKSIZE=80
//LKED         EXEC PGM=IEWL,REGION=250K,PARM=(XREF,LET,LIST,OVLY),
// COND=(4,LT,FORT)
//SYSLIB   DD DSN=SYS1.FORTLIB,DISP=SHR,UNIT=SYSDA
//         DD DSN=IDAPS.APPLIC,DISP=SHR
//         DD DSN=IDAPS.CONTROL,DISP=SHR
//SYSLMOD DD DSN=&GOSET(MAIN),UNIT=SYSDA,DISP=(NEW,PASS),
// SPACE=(1024,(30,10,1),RLSE)
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=&SYSUT1,UNIT=SYSDA,SPACE=(1024,(100,10),RLSE),
// DCB=BLKSIZE=1024
//SYSLIN DD DSN=&LOADSET,DISP=(OLD,DELETE)
//         DD DSN=&BIN,DISP=SHR,UNIT=SYSDA,VOL=SER=SDC001
//         DD DDNAME=SYSIN
//GO          EXEC PGM=*.LKED.SYSLMOD,COND=((4,LT,FORT),(4,LT,LKED)),
// REGION=250K
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT07F001 DD SYSOUT=B
//DISKF011 DD DSN=BT11,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC001,
// SPACE=(512,(5120)),DCB=(DSORG=DA,BLKSIZE=512)
//DISKF012 DD DSN=BT12,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC001,
// SPACE=(1024,(6144)),DCB=(DSORG=DA,BLKSIZE=1024)
//DISKF013 DD DSN=BT13,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC001,
// SPACE=(2048,(24576)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF014 DD DSN=BT14,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC001,
// SPACE=(2048,(8192)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF015 DD DSN=BT15,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC001,
// SPACE=(2048,(2048)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF021 DD DSN=BT21,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC002,
// SPACE=(512,(5120)),DCB=(DSORG=DA,BLKSIZE=512)
//DISKF022 DD DSN=BT22,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC002,
// SPACE=(1024,(6144)),DCB=(DSORG=DA,BLKSIZE=1024)
//DISKF023 DD DSN=BT23,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC002,
// SPACE=(2048,(30720)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF024 DD DSN=BT24,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC002,
// SPACE=(2048,(8192)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF025 DD DSN=BT25,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC002,
// SPACE=(2048,(2048)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF031 DD DSN=BT31,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC003,
// SPACE=(512,(5120)),DCB=(DSORG=DA,BLKSIZE=512)
//DISKF032 DD DSN=BT32,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC003,
// SPACE=(1024,(6144)),DCB=(DSORG=DA,BLKSIZE=1024)
//DISKF033 DD DSN=BT33,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC003,
// SPACE=(2048,(30720)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF034 DD DSN=BT34,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC003,
```

```
// SPACE=(2048,(8192)),DCB=(DSORG=DA,BLKSIZE=2048)
//DISKF035 DD DSN=BT35,UNIT=SYSDA,DISP=(OLD,KEEP),VOL=SER=SDC003,
// SPACE=(2048,(2048)),DCB=(DSORG=DA,BLKSIZE=2048)
//SYSUDUMP DD SYSOUT=A
```

1.4 ASCLIDAP

```
//ASCLIDAP PROC PDSNAME=CONTROL
//ASM        EXEC PGM=IEUASM,PARM=(LOAD,NODECK),REGION=50K
//SYSPRINT DD  SYSOUT=A
//SYSPUNCH DD  DUMMY
//SYSLIB   DD  DISP=SHR,DSN=SYS1.MACLIB
//SYSUT1   DD  UNIT=SYSDA,DSN=&SYSUT1,SPACE=(1700,(400,50))
//SYSUT2   DD  UNIT=SYSDA,DSN=&SYSUT2,SPACE=(1700,(400,50))
//SYSUT3   DD  UNIT=SYSDA,DSN=&SYSUT3,SPACE=(1700,(400,50))
//SYSGO DD UNIT=SYSDA,DSN=&LOADSET,SPACE=(80,(200,50)),
//          DISP=(MOD,PASS),DCB=BLKSIZE=80
//LKED     EXEC PGM=IEWL,PARM=(XREF,LIST,NCAL),REGION=50K,
//          COND=(8,LT,ASM)
//SYSPRINT DD  SYSOUT=A
//SYSLIN   DD  DSNAME=&LOADSET,DISP=(OLD,DELETE)
//         DD  DDNAME=SYSIN
//SYSLMOD  DD  DISP=SHR,VOL=SER=SDC001,DSNAME=IDAPS.&PDSNAME(&MEMNAME),
//          UNIT=SYSDA
//SYSUT1   DD  DSNAME=&SYSUT1,UNIT=(SYSDA,SEP=(SYSLIN,SYSLMOD)),
//          SPACE=(1024,(50,20))
```

1.5   FGCLIDAP

```
//FGCLIDAP       PROC PDSNAME=APPLIC
//FORT          EXEC PGM=IEYFORT,REGION=100K,PARM=(LIST)
//SYSPRINT DD   SYSOUT=A
//SYSPUNCH DD   SYSOUT=B
//SYSLIN    DD   DSNAME=&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,            X
//               SPACE=(80,(400,100),RLSE),DCB=BLKSIZE=80
//LKED          EXEC PGM=IEWL,REGION=96K,PARM=(XREF,LET,LIST),          X
//               COND=(4,LT,FORT)
//SYSPRINT DD   SYSOUT=A
//SYSLIB    DD  DSNAME=SYS1.FORTLIB,DISP=SHR
//SYSLMOD   DD  DISP=SHR,VOL=SER=SDC001,DSNAME=IDAPS.&PDSNAME(&MEMNAME),
//               UNIT=SYSDA
//SYSUT1    DD  UNIT=SYSDA,SPACE=(1024,(100,10),RLSE),DCB=BLKSIZE=1024, X
//               DSNAME=&SYSUT1
//SYSLIN    DD  DSNAME=&LOADSET,DISP=(OLD,DELETE)
//          DD  DDNAME=SYSIN
```

1.6 FHCLIDAP

```
//FHCLIDAP      PROC PDSNAME=APPLIC
//FORT EXEC PGM=IEKAA00,REGION=250K,PARM='LIST,MAP,XREF,NOEDIT,OPT=2'
//SYSPRINT DD   SYSOUT=A
//SYSPUNCH DD   SYSOUT=B
//SYSLIN     DD   DSNAME=&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,              X
//               SPACE=(80,(400,100),RLSE),DCB=BLKSIZE=80
//SYSUT1 DD  SPACE=(CYL,2),UNIT=3330,DSNAME=&SYSUT1
//SYSUT2 DD  SPACE=(CYL,2),UNIT=3330,DSNAME=&SYSUT2
//LKED      EXEC PGM=IEWL,REGION=96K,PARM=(XREF,LET,LIST),               X
//               COND=(4,LT,FORT)
//SYSPRINT DD   SYSOUT=A
//SYSLIB     DD   DSNAME=SYS1.FORTLIB,DISP=SHR
//SYSLMOD    DD   DISP=SHR,VOL=SER=SDC001,DSNAME=IDAPS.&PDSNAME(&MEMNAME),
//               UNIT=SYSDA
//SYSUT1     DD   UNIT=SYSDA,SPACE=(1024,(100,10),RLSE),DCB=BLKSIZE=1024, X
//               DSNAME=&SYSUT1
//SYSLIN     DD   DSNAME=&LOADSET,DISP=(OLD,DELETE)
//           DD   DDNAME=SYSIN
```

1.7   USERMOD

```
//USERMOD   PROC BIN=USER
//SCRATCH   EXEC PGM=IEHPROGM
//SYSPRINT DD SYSOUT=A
//DD1    DD  UNIT=3330,VOL=SER=NASA01,DISP=OLD
//DD2    DD  UNIT=3330,VOL=SER=SDC001,DISP=OLD
//DD3    DD  UNIT=3330,VOL=SER=SDC002,DISP=OLD
//FORT EXEC PGM=IEKAA00,REGION=250K,PARM='DECK,NOLIST,NOEDIT,OPT=2'
//SYSPRINT DD   SYSOUT=A
//FORT.SYSPUNCH DD  DSN=&BIN,UNIT=SYSDA,VOL=SER=SDC001,DISP=(NEW,KEEP),
// SPACE=(80,(200,100)),DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSLIN     DD   DSNAME=&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,            ·X
//              SPACE=(80,(400,100),RLSE),DCB=BLKSIZE=80
//SYSUT1 DD  SPACE=(CYL,2),UNIT=3330,DSNAME=&SYSUT1
//SYSUT2 DD  SPACE=(CYL,2),UNIT=3330,DSNAME=&SYSUT2
//OVERLAY   EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT2 DD  VOL=REF=*.FORT.SYSPUNCH,DSN=&BIN,DISP=(MOD,PASS)
//SYSUT1 DD DDNAME=INPUT
```

2.0 UTILITY PROGRAMS

2.1 STRUCTURE INTERACTIVE PACKS

```
//SCRATCH   EXEC PGM=IEHPROGM
//SYSPRINT DD SYSOUT=A
//DD1   DD  UNIT=3330,VOL=SER=NASA01,DISP=OLD
//DD2   DD  UNIT=3330,VOL=SER=SDC201,DISP=OLD
//DD3   DD  UNIT=3330,VOL=SER=SDC202,DISP=OLD
//DD4   DD  UNIT=3330,VOL=SER=SDC004,DISP=OLD
//SYSIN DD *
      SCRATCH VOL=3330=SDC201,DSNAME=IT11
      SCRATCH VOL=3330=SDC201,DSNAME=IT12
      SCRATCH VOL=3330=SDC201,DSNAME=IT13
      SCRATCH VOL=3330=SDC201,DSNAME=IT14
      SCRATCH VOL=3330=SDC201,DSNAME=IT15
      SCRATCH VOL=3330=SDC202,DSNAME=IT21
      SCRATCH VOL=3330=SDC202,DSNAME=IT22
      SCRATCH VOL=3330=SDC202,DSNAME=IT23
      SCRATCH VOL=3330=SDC202,DSNAME=IT24
      SCRATCH VOL=3330=SDC202,DSNAME=IT25
      SCRATCH VOL=3330=SDC004,DSNAME=IT31
      SCRATCH VOL=3330=SDC004,DSNAME=IT32
      SCRATCH VOL=3330=SDC004,DSNAME=IT33
      SCRATCH VOL=3330=SDC004,DSNAME=IT34
      SCRATCH VOL=3330=SDC004,DSNAME=IT35
/*
//CONVRT EXEC FORTGCLG,REGION.GO=100K,TIME.GO=60
//FORT.SYSIN DD *
C        ROUTINE TO STRUCTURE DISK PACKS FOR DISKIO PACKAGE
C           LEN(I) - LINE LENGTH FOR THE ITH FILE
C           INL(J) - NUMBER OF RECORDS IN THE JTH FILE
C---
C        DIRACC - THIS SUBROUTINE INITIALIZES THE DISK FILES
C---
      DIMENSION  INL(5),  IBUF(2048),  LEN(5)
      DATA   INL /5120,6144,30720,8192,2048/
      DATA   IBUF /512*0/
      DATA   LEN /512,1024,3*2048/
C        ROUTINE TO STRUCTURE DISK SPACE
      CALL DIRACC(IBUF,0,11,1,IERR)
      DO 30 J=1,5
      DO 20 I=1,3
      NL = INL(J)
      LU = I*10 + J
      IBUF(1) = LU
        CALL STIME(1)
      DO 10 K=1,NL
      K1 = K-1
      IERR = 0
      CALL DIRACC(IBUF,K1,LU,0,IERR)
      IF(IERR .NE. 0) WRITE(6,40) IERR
```

```
   10 CONTINUE
      CALL ETIME(TIME)
      WRITE(6,50) LU,K,TIME
   50 FORMAT('0  DISK STRUCTURED FOR LU =',I3,' , WITH ',I6,
     A          ' RECORDS, TIME =',F10.4)
   20 CONTINUE
   30 CONTINUE
   40 FORMAT('+',T50,'** ERROR **      ',Z8)
      STOP
      END
/*
//LKED.SYSLIB DD UNIT=SYSDA
//             DD DSN=IDAPS.CONTROL,DISP=SHR
//GO.DISKF011 DD DSN=IT11,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC201,
// SPACE=(512,(5120)),DCB=(DSORG=DA,BLKSIZE=512)
//GO.DISKF012 DD DSN=IT12,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC201,
// SPACE=(1024,(6144)),DCB=(DSORG=DA,BLKSIZE=1024)
//GO.DISKF013 DD DSN=IT13,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC201,
// SPACE=(2048,(30720)),DCB=(DSORG=DA,BLKSIZE=2048)
//GO.DISKF014 DD DSN=IT14,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC201,
// SPACE=(2048,(8192)),DCB=(DSORG=DA,BLKSIZE=2048)
//GO.DISKF015 DD DSN=IT15,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC201,
// SPACE=(2048,(2048)),DCB=(DSORG=DA,BLKSIZE=2048)
//GO.DISKF021 DD DSN=IT21,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC202,
// SPACE=(512,(5120)),DCB=(DSORG=DA,BLKSIZE=512)
//GO.DISKF022 DD DSN=IT22,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC202,
// SPACE=(1024,(6144)),DCB=(DSORG=DA,BLKSIZE=1024)
//GO.DISKF023 DD DSN=IT23,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC202,
// SPACE=(2048,(30720)),DCB=(DSORG=DA,BLKSIZE=2048)
//GO.DISKF024 DD DSN=IT24,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC202,
// SPACE=(2048,(8192)),DCB=(DSORG=DA,BLKSIZE=2048)
//GO.DISKF025 DD DSN=IT25,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC202,
// SPACE=(2048,(2048)),DCB=(DSORG=DA,BLKSIZE=2048)
//GO.DISKF031 DD DSN=IT31,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC004,
// SPACE=(512,(5120)),DCB=(DSORG=DA,BLKSIZE=512)
//GO.DISKF032 DD DSN=IT32,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC004,
// SPACE=(1024,(6144)),DCB=(DSORG=DA,BLKSIZE=1024)
//GO.DISKF033 DD DSN=IT33,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC004,
// SPACE=(2048,(30720)),DCB=(DSORG=DA,BLKSIZE=2048)
//GO.DISKF034 DD DSN=IT34,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC004,
// SPACE=(2048,(8192)),DCB=(DSORG=DA,BLKSIZE=2048)
//GO.DISKF035 DD DSN=IT35,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC004,
// SPACE=(2048,(2048)),DCB=(DSORG=DA,BLKSIZE=2048)
//GO.SYSABEND DD SYSOUT=A
/*
```

2.2   STRUCTURE BATCH PACKS

```
//SCRATCH   EXEC PGM=IEHPROGM
//SYSPRINT DD SYSOUT=A
//DD1   DD  UNIT=3330,VOL=SER=NASA01,DISP=OLD
//DD2   DD  UNIT=3330,VOL=SER=SDC001,DISP=OLD
//DD3   DD  UNIT=3330,VOL=SER=SDC002,DISP=OLD
//DD4   DD  UNIT=3330,VOL=SER=SDC003,DISP=OLD
//SYSIN DD *
      SCRATCH VOL=3330=SDC001,DSNAME=BT11
      SCRATCH VOL=3330=SDC001,DSNAME=BT12
      SCRATCH VOL=3330=SDC001,DSNAME=BT13
      SCRATCH VOL=3330=SDC001,DSNAME=BT14
      SCRATCH VOL=3330=SDC001,DSNAME=BT15
      SCRATCH VOL=3330=SDC002,DSNAME=BT21
      SCRATCH VOL=3330=SDC002,DSNAME=BT22
      SCRATCH VOL=3330=SDC002,DSNAME=BT23
      SCRATCH VOL=3330=SDC002,DSNAME=BT24
      SCRATCH VOL=3330=SDC002,DSNAME=BT25
      SCRATCH VOL=3330=SDC003,DSNAME=BT31
      SCRATCH VOL=3330=SDC003,DSNAME=BT32
      SCRATCH VOL=3330=SDC003,DSNAME=BT33
      SCRATCH VOL=3330=SDC003,DSNAME=BT34
      SCRATCH VOL=3330=SDC003,DSNAME=BT35
/*
//CONVRT EXEC FORTGCLG,REGION.GO=100K,TIME.GO=60
//FORT.SYSIN DD *
C         ROUTINE TO STRUCTURE DISK PACKS FOR DISKIO PACKAGE
C            LEN(I) - LINE LENGTH FOR THE ITH FILE
C            INL(J) - NUMBER OF RECORDS IN THE JTH FILE
C---
C         DIRACC - THIS SUBROUTINE INITIALIZES THE DISK FILES
C---
      DIMENSION INL(5), IBUF(2048), LEN(5)
      DATA   INL /5120,6144,30720,8192,2048/
      DATA   IBUF /512*0/
      DATA   LEN /512,1024,3*2048/
C         ROUTINE TO STRUCTURE DISK SPACE
      CALL DIRACC(IBUF,0,11,1,IERR)
      DO 30 J=1,5
      DO 20 I=1,3
      NL = INL(J)
      LU = I*10 + J
      IF(LU .EQ 13) NL = 24576
      IBUF(1) = LU
         CALL STIME(1)
      DO 10 K=1,NL
      K1 = K-1
      IERR = 0
      CALL DIRACC(IBUF,K1,LU,0,IERR)
      IF(IERR .NE. 0) WRITE(6,40) IERR
   10 CONTINUE
      CALL ETIME(TIME)
```

```
      WRITE(6,50) LU,K,TIME
   50 FORMAT('0   DISK STRUCTURED FOR LU =',I3,' , WITH ',I6,
     A          ' RECORDS, TIME =',F10.4)
   20 CONTINUE
   30 CONTINUE
   40 FORMAT('+',T50,'** ERROR **      ',Z8)
      STOP
      END
/*
//LKED.SYSLIB DD UNIT=SYSDA
//             DD DSN=IDAPS.CONTROL,DISP=SHR
//GO.DISKF011 DD DSN=BT11,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC001,
// SPACE=(512,(5120)),DCB=(DSORG=DA,BLKSIZE=512)
//GO.DISKF012 DD DSN=BT12,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC001,
// SPACE=(1024,(6144)),DCB=(DSORG=DA,BLKSIZE=1024)
//GO.DISKF013 DD DSN=BT13,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC001,
// SPACE=(2048,(24576)),DCB=(DSORG=DA,BLKSIZE=2048)
//GO.DISKF014 DD DSN=BT14,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC001,
// SPACE=(2048,(8192)),DCB=(DSORG=DA,BLKSIZE=2048)
//GO.DISKF015 DD DSN=BT15,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC001,
// SPACE=(2048,(2048)),DCB=(DSORG=DA,BLKSIZE=2048)
//GO.DISKF021 DD DSN=BT21,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC002,
// SPACE=(512,(5120)),DCB=(DSORG=DA,BLKSIZE=512)
//GO.DISKF022 DD DSN=BT22,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC002,
// SPACE=(1024,(6144)),DCB=(DSORG=DA,BLKSIZE=1024)
//GO.DISKF023 DD DSN=BT23,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC002,
// SPACE=(2048,(30720)),DCB=(DSORG=DA,BLKSIZE=2048)
//GO.DISKF024 DD DSN=BT24,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC002,
// SPACE=(2048,(8192)),DCB=(DSORG=DA,BLKSIZE=2048)
//GO.DISKF025 DD DSN=BT25,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC002,
// SPACE=(2048,(2048)),DCB=(DSORG=DA,BLKSIZE=2048)
//GO.DISKF031 DD DSN=BT31,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC003,
// SPACE=(512,(5120)),DCB=(DSORG=DA,BLKSIZE=512)
//GO.DISKF032 DD DSN=BT32,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC003,
// SPACE=(1024,(6144)),DCB=(DSORG=DA,BLKSIZE=1024)
//GO.DISKF033 DD DSN=BT33,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC003,
// SPACE=(2048,(30720)),DCB=(DSORG=DA,BLKSIZE=2048)
//GO.DISKF034 DD DSN=BT34,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC003,
// SPACE=(2048,(8192)),DCB=(DSORG=DA,BLKSIZE=2048)
//GO.DISKF035 DD DSN=BT35,UNIT=SYSDA,DISP=(NEW,KEEP),VOL=SER=SDC003,
// SPACE=(2048,(2048)),DCB=(DSORG=DA,BLKSIZE=2048)
//GO.SYSABEND DD SYSOUT=A
/*
```

2.3 COMPRESS IDAPS.CONTROL LIBRARY

```
//COMPRSCN EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=A
//LIBR    DD DSN=IDAPS.CONTROL,UNIT=3330,VOL=SER=SDC001,DISP=(OLD,KEEP)
//SYSUT3 DD UNIT=3330,SPACE=(TRK,(1))
//SYSUT4 DD UNIT=3330,SPACE=(TRK,(1))
//SYSIN DD *
     COPY OUTDD=LIBR,INDD=LIBR
/*
```

2.4 COMPRESS IDAPS.APPLIC LIBRARY

```
//COMPRSAP EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=A
//LIBR    DD DSN=IDAPS.APPLIC,UNIT=3330,VOL=SER=SDC001,DISP=(OLD,KEEP)
//SYSUT3 DD UNIT=3330,SPACE=(TRK,(1))
//SYSUT4 DD UNIT=3330,SPACE=(TRK,(1))
//SYSIN DD *
     COPY OUTDD=LIBR,INDD=LIBR
/*
```

2.5   SCRATCH AND BUILD IDAPS.CONTROL LIBRARY

```
//SCRATCH  EXEC PGM=IEHPROGM
//SYSPRINT DD SYSOUT=A
//DD1  DD UNIT=3330,VOL=SER=NASA01,DISP=OLD
//DD2  DD UNIT=3330,VOL=SER=SDC001,DISP=OLD
//SYSIN DD *
     SCRATCH VTOC,VOL=3330=SDC001,SYS
     SCRATCH DSNAME=IDAPS.CONTROL,VOL=3330=SDC001
     UNCATLG DSNAME=IDAPS.CONTROL
/*
//STEP1     EXEC FORTGCL,PARM.FORT=LIST,REGION.FORT=100K
//FORT.SYSLIN DD UNIT=SYSDA
//FORT.SYSIN DD *

            .
            .
            .
     (SUBROUTINE ALPIN FORTRAN SOURCE MODULE)
            .
            .
            .
/*
//LKED.SYSLIB DD UNIT=SYSDA
//LKED.SYSLMOD DD UNIT=SYSDA,VOL=SER=SDC001,DSN=IDAPS.CONTROL(ALPIN),
// DISP=(NEW,KEEP),SPACE=(CYL,(20,1,40))
/*
//CATLG     EXEC PGM=IEHPROGM
//SYSPRINT DD SYSOUT=A
//DD1  DD UNIT=3330,VOL=SER=NASA01,DISP=OLD
//DD2  DD UNIT=3330,VOL=SER=SDC001,DISP=OLD
//SYSIN DD *
     CATLG DSNAME=IDAPS.CONTROL,VOL=3330=SDC001
/*
```

2.6   SCRATCH AND BUILD IDAPS.APPLIC LIBRARY

```
//SCRATCH   EXEC PGM=IEHPROGM
//SYSPRINT DD SYSOUT=A
//DD1   DD UNIT=3330,VOL=SER=NASA01,DISP=OLD
//DD2   DD UNIT=3330,VOL=SER=SDC001,DISP=OLD
//SYSIN DD *
     SCRATCH VTOC,VOL=3330=SDC001,SYS
     SCRATCH DSNAME=IDAPS.APPLIC,VOL=3330=SDC001
     UNCATLG DSNAME=IDAPS.APPLIC
/*
//STEP1     EXEC FORTGCL,PARM.FORT=LIST,REGION.FORT=100K
//FORT.SYSLIN DD UNIT=SYSDA
//FORT.SYSIN DD *

              •
              •
              •
       (SUBROUTINE ALTER FORTRAN SOURCE MODULE)
              •
              •
              •
/*
//LKED.SYSLIB DD UNIT=SYSDA
//LKED.SYSLMOD DD UNIT=SYSDA,VOL=SER=SDC001,DSN=IDAPS.APPLIC(ALTER),
// DISP=(NEW,KEEP),SPACE=(CYL,(20,1,40))
/*
//CATLG     EXEC PGM=IEHPROGM
//SYSPRINT DD SYSOUT=A
//DD1   DD UNIT=3330,VOL=SER=NASA01,DISP=OLD
//DD2   DD UNIT=3330,VOL=SER=SDC001,DISP=OLD
//SYSIN DD *
       CATLG DSNAME=IDAPS.APPLIC,VOL=3330=SDC001
/*
```

2.7   LIST DISK PACKS

```
//LISTCAT   EXEC PGM=IEHLIST
//SYSPRINT DD SYSOUT=A
//DD2 DD UNIT=3330,VOL=SER=NASA01,DISP=OLD
//SYSIN DD *
   LISTCTLG VOL=3330=NASA01
/*
//LISTCT1   EXEC PGM=IEHLIST
//SYSPRINT DD SYSOUT=A
//DD2 DD UNIT=3330,VOL=SER=SDC001,DISP=OLD
//SYSIN DD *
   LISTCTLG VOL=3330=SDC001
   LISTVTOC FORMAT,VOL=3330=SDC001
/*
//LISTCT2   EXEC PGM=IEHLIST
//SYSPRINT DD SYSOUT=A
//DD2 DD UNIT=3330,VOL=SER=SDC002,DISP=OLD
//SYSIN DD *
   LISTCTLG VOL=3330=SDC002
   LISTVTOC FORMAT,VOL=3330=SDC002
/*
//LISTCT3   EXEC PGM=IEHLIST
//SYSPRINT DD SYSOUT=A
//DD2 DD UNIT=3330,VOL=SER=SDC201,DISP=OLD
//SYSIN DD *
   LISTCTLG VOL=3330=SDC201
   LISTVTOC FORMAT,VOL=3330=SDC201
/*
//LISTCT4   EXEC PGM=IEHLIST
//SYSPRINT DD SYSOUT=A
//DD2 DD UNIT=3330,VOL=SER=SDC202,DISP=OLD
//SYSIN DD *
   LISTCTLG VOL=3330=SDC202
   LISTVTOC FORMAT,VOL=3330=SDC202
/*
//LISTCT5   EXEC PGM=IEHLIST
//SYSPRINT DD SYSOUT=A
//DD2 DD UNIT=3330,VOL=SER=SDC003,DISP=OLD
//SYSIN DD *
   LISTCTLG VOL=3330=SDC003
   LISTVTOC FORMAT,VOL=3330=SDC003
/*
//LISTCT6   EXEC PGM=IEHLIST
//SYSPRINT DD SYSOUT=A
//DD2 DD UNIT=3330,VOL=SER=SDC004,DISP=OLD
//SYSIN DD *
   LISTCTLG VOL=3330=SDC004
   LISTVTOC FORMAT,VOL=3330=SDC004
/*
```

2.8 LIST LIBRARY

```
//LISTER EXEC FORTGCLG
//FORT.SYSIN DD *
C---
C        THIS PROGRAM LISTS A DECK OF SUBROUTINES.  IT TESTS FOR AN
C        END CARD AND THEN PAGE EJECTS.
C---
      DIMENSION N(20)
      INTEGER*2 IT3,IT4,LN(2)
      EQUIVALENCE (N(1),LN(1))
      DATA IT3,IT4 /2H//,2H/*/
      DATA IT1,IT2 /4H  EN,4HD    /.
      DATA IT5/4H END/
      DATA ITAPE,IPRI  /4HTAPE,4HPRIN/
      IN = 5
      READ(5,5) INP,IOUT
    5 FORMAT(A4,4X,A4)
      IF( INP .EQ. ITAPE) IN = 3
   10 CONTINUE
      WRITE(6,12)
   12 FORMAT('1')
      DO 20 I=1,59
      READ(IN,14,END=100) N
   14 FORMAT(20A4)
      IF(LN(1) .EQ. IT3) GO TO 15
      IF(LN(1) .EQ. IT4) GO TO 15
      IF(IOUT .EQ. ITAPE) WRITE(4,18) N
      WRITE(6,16) N
   15 CONTINUE
   16 FORMAT(' ',20A4)
   18 FORMAT(20A4)
      IF(N(3) .EQ. IT5) GO TO 10
      IF(N(2) .NE. IT1) GO TO 20
      IF(N(3) .EQ. IT2) GO TO 10
   20 CONTINUE
      GO TO 10.
  100 CONTINUE
      STOP
      END
/*
//GO.SYSIN DD DATA,DLM=ZZ
CARD TO PRINT
          .
          .
          .
      (SOURCE DECK FOR LIBRARY TO BE LISTED)
          .
          o
          .
ZZ
```

3.0 COMMON INITIALIZATION

3.1 BLOCK DATA

```
      BLOCK DATA
C---
C     SUBROUTINE BLOCK DATA
C---
C        THIS ROUTINE INITIALIZES THE COMMON AREAS OF THE IDAPS
C        SYSTEM.
C---
      COMMON /OPER/  IOP(3,100)
      DIMENSION KOP(3,50)
      EQUIVALENCE (IOP(1,51),KOP(1,1))
C---
C        OPERATOR DIRECTORY
C        NAME   = IOP(1,NOP)
C        NPAR   = IOP(2,NOP)
C        FILREQ = IOP(3,NOP) = UVWXYZ
C           U = NO. INPUT FILES
C           V = NO. OUTPUT FILES (PIXEL)
C           W = NO. OUTPUT FILES (FLOATING POINT)
C           X = NO. OUTPUT FILES (SECONDARY)
C           Y = NO. SCRATCH FILES (PIXEL)
C           Z = NO. SCRATCH FILES (FLOATING POINT)
C
C---
      DATA IOP
     A /4HSCA ,2, 010000,   4HINP ,4, 010000,   4HDIS ,6, 100000,
     B  4HOUT ,4, 100000,   4HFIL ,5, 100000,   4HPRI ,4, 100000,
     C  4HALT ,0, 110000,   4HISO ,5, 110000,   4HDIF ,6, 210201,
     D  4HDEL ,0, 500000,   4HAUT ,8, 110000,   4HEXT ,4, 110000,
     E  4HINS ,11,210000,   4HIMA ,6, 010000,   4HLAB ,2, 110000,
     F  4HINV ,0, 110000,   4HMAT ,2, 201000,   4HSTO ,6, 010000,
     G  4HTRA ,5, 110000,   4HFRA ,9, 110000,   4HAVE ,0, 510000,
     H  4HHOC ,5, 110000,   4HROT ,1, 110010,   4HLOG ,1, 000000,
     I  4HREG ,8, 110010,   4HOVR ,12,210000,   4HSLR ,0, 010240,
     J  4HFEA ,4, 110020,   4HSCD ,0,0000000,   4HMAG ,1, 110010,
     K  4HGEO ,6, 110010,   4HARE ,6, 200000,   4HBOR ,2, 010000,
     L  4HCON ,2, 301000,   4HSIM ,6, 200000,   4HDEP ,3, 101000,
     M  4H    ,0, 000000,   4HFFT ,4, 102303,   4HIFF ,1, 201303,
     N  4HFOU ,2, 301303,   4HHAN ,3, 310000,   4HFGN ,13,001000,
     P  4HWIN ,1, 101000,   4HPSF ,3, 001000,   4H    ,0,0000000,
     Q  4H    ,0,0000000,   4H    ,0,0000000/
      DATA KOP
     A /4H    ,0,0000000,   4H    ,0,0000000,   4H    ,0,0000000,
     B  4H    ,0,0000000,   4H    ,0,0000000,   4H    ,0,0000000,
     C  4H    ,0,0000000,   4H    ,0,0000000,   4H    ,0,0000000,
     D  4H    ,0,0000000,   4H    ,0,0000000,   4H    ,0,0000000,
     E  4H    ,0,0000000,   4H    ,0,0000000,   4H    ,0,0000000,
     F  4H    ,0,0000000,   4H    ,0,0000000,   4H    ,0,0000000,
     G  4H    ,0,0000000,   4H    ,0,0000000,   4H    ,0,0000000,
```

```
      H   4H      ,0,0000000,   4H       ,0,0000000,   4H       ,0,0000000,
      I   4H      ,0,0000000,   4H       ,0,0000000,   4H       ,0,0000000,
      J   4H      ,0,0000000,   4H       ,0,0000000,   4H       ,0,0000000,
    . K   4H      ,0,0000000,   4H       ,0,0000000,   4H       ,0,0000000,
      L   4H      ,0,0000000,   4H       ,0,0000000,   4H       ,0,0000000,
      M   4H      ,0,0000000,·  4H       ,0,0000000,   4H       ,0,0000000,
      N   4H      ,0,0000000,   4H       ,0,0000000,   4H       ,0,0000000,
      O   4H      ,0,0000000,   4H       ,0,0000000,   4H       ,0,0000000,
      P   4H      ,0,0000000,   4H       ,0,0000000,   4H       ,0,,0000000,
      Q   4H      ,0,0000000,   4H       ,0,0000000/
C---
      COMMON /FILEDR/ IFD(6,5), OFD(6,5), SFD(6,5)
      COMMON /FILEPR/ FD(6,164), FCOUNT, PCOUNT, DCOUNT, FILNDX(100),
     A                INP(3), KDELX(9)
      INTEGER  FCOUNT, PCOUNT, DCOUNT, FILNDX
      INTEGER FD
      INTEGER SFD, OFD
C---
C
C        FILE DIRECTORIES
C        FD(FN,DF,NL,NC,RA,LU)        - IMAGE FILE DIRECTORY -
C        IFD(FN,DF,NL,NC,RA,LU)       - INPUT FILE DIRECTORY -
C        OFD(FN,DF,NL,NC,RA,LU)       - OUTPUT FILE DIRECTORY -
C        SFD(FN,DF,NL,NC,RA,LU)       - SCRATCH FILE DIRECTORY -
C
C        FN - FILE NAME
C        DF - DATA FORMAT
C             (1 - PIXEL, 2 - FLOATING POINT)
C        NL - NUMBER OF RECORDS (LINES)
C        NC - NUMBER OF PIXELS PER RECORD
C        RA - RELATIVE ADDRESS OF LINE 1
C        LU - LOGICAL UNIT NUMBER
C---
      DATA IFD /30*0/
      DATA OFD /12*0,4H1SEC,1,0,0,0,0,4H2SEC,1,0,0,0,0,
     A          4H3SEC,1,0,0,0,0/
      DATA FD /4H1SEC,1,0,0,0,0,4H2SEC,1,0,0,0,0,4H3SEC,1,0,0,0,0,
     A          966*0/
      DATA SFD /4H1SCR,4*0,14,4H2SCR,4*0,24,4H3SCR,4*0,34,12*0/
C---
C        FILE PARAMETERS
C          FCOUNT - FILE DIRECTORY INDEX
C          PCOUNT - PIXEL FILE COUNTER
C          DCOUNT - FLOATING POINT FILE COUNTER
C          FILNDX - ARRAY OF FILE NAME PREFIXES
C          INP    - ALTERNATE PACK ARRAY
C          KDELX  - DELETED SPACE INDEX ARRAY
C---
      DATA FILNDX /100*1/
      DATA FCOUNT,PCOUNT,DCOUNT /4,1,1/
      DATA INP /1,2,3/
      DATA KDELX /9*0/
C---
      COMMON /SPACE/ PSPACE(2,3,3), DSPACE(2,3,3,50)
```

```
      INTEGER  PSPACE,DSPACE
C---
C          PRIMARY SPACE
C            PSPACE = (I,J,K)
C          DELETED SPACE      .
C            DSPACE = (I,J,K,L)
C               I = 1, NUMBER OF LINES AVAILABLE
C                 = 2, RELATIVE BLOCK ADDRESS
C               J = 1, PACK NO. 1
C                 = 2, PACK NO. 2
C                 = 3, PACK NO. 3
C               K = 1, 512 BYTE LINE LENGTH
C                 = 2, 1024 BYTE LINE LENGTH
C                 = 3, 2048 BYTE LINE LENGTH
C               L = INDEX OF DELETED SPACE
C---
      DATA DSPACE /900*0/
      DATA PSPACE /5120,0,5120,0,5120,0,6144,0,6144,0,6144,0,
     A            24576,0,30720,0,30720,0/
C---
      COMMON  /BUF/  MESS,LL(512)
C---
C          INTERFACE MESSAGE BUFFER
C            MESS - 4-BYTE MESSAGE TYPE CODE
C            LL   - 2048 BYTE BUFFER AREA
C---
C---
C          TIMER CONTROL
      COMMON  /TIMER/  KTIME
C---
      COMMON  /CDBUF/  NWORDS,CD(300)
C          COMPUTED DATA BUFFER
C            NWORDS - NUMBER OF VALUES IN BUFFER
C            CD     - 1200-BYTE (300 VALUES) BUFFER AREA
      END
```

EXTERNAL DISTRIBUTION

Mr. Douglas T. Thomas   (6)
EF33
George C. Marshall Space Flight Center
National Aeronautics and Space Administration
Marshall Space Flight Center, Alabama   35812

INTERNAL DISTRIBUTION

Huntsville

C. Cooper   (6)
Library   (1)

Santa Monica

Document Distribution   (1)