File No. 7090-36
Form C28-6383-2

IBM Systems Reference Library

IBM 7090-7040 Direct Couple Operating System

Systems Programmer's Guide

This publication contains information useful to programmers who require a thorough understanding of the IBM 7090-7040 Direct Couple Operating System (DCOS), #7090-PR-161. This system consists of the following parts:

7090/7094 Operating System (IBSYS)

7040/7044 Control Program (DCMUP)

The publication provides descriptions of general DCOS program logic, the format of the DCOS System Library, the Operating System Monitor (DC-IBSYS), and the Input/Output Executor (DC-IOEX). It also includes instructions for system library preparation and maintenance.

Separate publications contain descriptions of the components of the 7090/7094 Operating System (IBSYS). Other related publications contain information needed by applications programmers and instructions for machine operators.

PREFACE

This publication is primarily intended for systems programmers who are responsible for the maintenance of the 7090-7040 Direct Couple Operating System (DCOS) and any modifications to it at the installation. It may also be of interest to individuals who desire a more thorough understanding of the system.

The reader of this publication is assumed to be familiar with the contents of the publication IBM 7090-7040 Direct Couple Operating System: Programmer's Guide, Form C28-6382.

Information required by machine operators is contained in the publication IBM 7090-7040 Direct Couple Operating System: Operator's Guide, Form C28-6384.

For information on the IBJOB processor, the major subsystem of DCOS, the reader is referred to the following IBM publications:

IBM 7090/7094 IBSYS Operating System: IBJOB Processor, Form C28-6389
IBM 7090/7094 Programming Systems: FORTRAN IV Language, Form C28-6390
IBM 7090/7094 Programming Systems: Macro Assembly Program (MAP) Language, Form C28-6392
IBM 7090/7094 Programming Systems: COBOL Language, Form C28-6391
IBM 7090/7094 IBSYS Operating System: IBJOB Processor Debugging Package, Form C28-6393

Information about the IBM 7040/7044 Data Processing Systems, the IBM 7090/7094/7094 II Data Processing Systems, and for the Direct Couple feature are given in the following publications:

IBM 7040/7044 Principles of Operation, Form A22-6649

IBM 7090 Principles of Operation, Form A22-6528

IBM 7094 Principles of Operation, Form A22-6703

Directly Coupled Processing Units--7040 to 7090/7094; 7040 to 7094/7094 II, Form A22-6803

The Direct Couple Operating System is designed for five machine configurations:

1. 7090-DC-7040
2. 7094-DC-7040
3. 7094-DC-7044
4. 7094 II-DC-7044
5. 7094 II-DC-7040

Throughout this publication, the terms 7090 and 7040 are used to refer to the 7090, 7094, or 7094 II and to the 7040 or 7044, respectively, as permitted in the configurations above.

This publication is divided into ten sections. The first section is devoted to the logic flow of the 7040 side of the system. The next three sections are devoted to the 7090 side of the system. The fifth section explains the logic used by the 7040 when servicing 7090 input/output requirements. The sixth section describes the DCOS Distribution Tape. The seventh section describes system editing procedures. The eighth and ninth sections deal with system library preparation and maintenance. The last section discusses DCOS unit assignment philosophies.

Major Revision (March 1965)

This edition, Form C28-6383-2, is a major revision of Form C28-6383-1. This publication amplifies material previously presented, and, in some cases, provides additional material. Major changes and additions are concerned with: examples of system modification, unit control blocks (7040 tape units), the DCOS distribution tape, the system editor (IBEDT), job setup, the $JOB card, and the *REMARK card.

Additions or changes are indicated by a vertical line to the left of the text; new or revised illustrations are denoted by the symbol • to the left of the caption.

The following publications are made obsolete by this revision: C28-6383-1, C28-6383-0, and the Technical Newsletters N28-0146-0 and N28-0144-0.

CONTENTS

ILLUSTRATIONS

ILLUSTRATIONS

The IBM 7090 IBJOB Processor is the only distributed programming system operating under DCOS control. Any program that is written in the FORTRAN IV, the COBOL, or the MAP language and that uses the IBJOB Input/Output Control System will operate under DCOS with little or no modification. The 7090 Operating System Monitor (IBSYS) has been replaced by a 7090 Operating System Monitor with Direct Couple capabilities (DC-IBSYS) and the system editor (IBEDT) has been modified to operate under DC-IBSYS. The IBJOB processor has been modified to communicate, through DC-IBSYS, with the 7040 for input/output processing.

The 7040 control program (DCMUP) performs the preprocessing and postprocessing of jobs, schedules work for 7090 processing, and performs all input/output functions for the 7090.

Jobs are prepared for DCOS in much the same manner as for the IBSYS operating system. When a job is ready for entry into the system, it is placed in the IBM 1402 Card Reader. As a job is read in, a job description block is created for job control. The job deck is then written on the disk (drum) storage unit to await processing, and an entry is made in the job queue table in a position based on the job's priority. The job description block and the job queue table are described later in this publication.

When the 7090 requires input/output activity, the request and the input/output commands are sent to the 7040 for execution. In the case of input, the 7040 interprets the request and transmits the specified number of words from a 7040 core storage buffer to the 7090. When the 7090 has an output record ready, the 7040 transmits the record to a 7040 core storage buffer. As the buffers are filled, they are written on the disk (drum) or a 7040 tape unit. Communication is effected between the 7040 and the 7090 by traps.

The 7040 interprets input/output requests and transmits data between its core storage and 7090 core storage under control of 7090 data channel commands (IOCP, IORT, etc.). When transmission has been completed, the 7040 transmits flags to the 7090 for such conditions as end of file and traps the 7090 to signal that the request has been serviced. The 7090 then handles this trap in the same manner that it handles a data channel trap in a stand-alone system.

When the 7090 completes a job, it signals that it is ready for the next job. The 7040 selects the next job that is ready to be processed by the 7090 and reads its job description block into 7040 core storage. A 7090 initialization program is transmitted to the 7090. This program clears 7090 core storage. DC-IBSYS is then loaded into the 7090 from the disk (drum) and is given control. For systems that operate in the direct mode, job processing then proceeds as it would on a stand-alone 7090 except for processing of input/output requests. When a DC-IBSYS subsystem job segment that is to be run in the compatibility mode is recognized by the DC-IBSYS Supervisor (DC-IBSUP), an enter-compatibility-mode call is sent to the 7040. The 7040 then loads the desired DC-IBSYS subsystem and gives it control.

For job segments that do not operate under DC-IBSYS, the 7090 issues an enter-compatibility-mode call to the 7040. The method used to load the desired system is then determined by the parameter of the $EXECUTE card. This parameter will be the word CARDS, the word TAPE, or a system name. If the parameter is CARDS or TAPE, pressing of the appropriate 7090 LOAD button is simulated in the 7090. If the parameter is a system name, the specified system is loaded into the 7090 from the disk (drum) and given control.

When a job has been completed, the 7090 stops, trapping the 7040 so that all input and output files for that job can be closed and another job initiated for the 7090. The output of the completed job is then queued for printing and punching. When all postprocessing has been completed, the job is purged from the job queue table and all disk (drum) storage tracks associated with the job are made available to the system for use by subsequent jobs.

DCOS USE OF DISK/DRUM STORAGE UNITS

DCOS contains procedures that provide dynamic assignment of disk (drum) tracks. The tracks are allocated to the jobs as they are needed and returned to available status when the job is completed. No particular job type or data file is restricted to any specific number of tracks or modules.

MODES OF OPERATION

The Direct Couple Operating System has two modes of operation: the direct mode and the compatibility mode. The primary difference between the two modes is the method used for processing 7090 input/output requirements. The direct mode is the faster of the two in that input/output requests are serviced with a minimum amount of interpretation and a minimum number of multiprocessor traps. The 7040 contains a program that services 7090 input/output requests. The 7040 control program also schedules jobs for the 7090 and handles preprocessing and postprocessing. Control of the entire system resides in the 7040 commutator.

THE DIRECT MODE

The direct mode is the normal DCOS mode of operation. In this mode, DCOS ignores the existence of 7090 data channels. DC-IBSYS and DC-IOEX in the 7090 handle all input/output by treating the DC channel as a data channel. The input/output executor (DC-IOEX) structures an input/output request and traps the 7040, the 7040 then transmits the request to its core storage, and the 7090 continues processing. The 7040 interprets the request, processes it, and traps the 7090. DC-IOEX then processes the trap as it would a normal data channel trap. The 7090 does not execute any input/output instructions or commands.

THE COMPATIBILITY MODE

The compatibility mode is provided for the execution of systems and object programs that do not utilize the direct mode input/output conventions. When the 7090 DC-IBSYS determines from a $EXECUTE card that one of these systems is specified, the compatibility mode must be entered. The logic of the compatibility mode is shown in Figure 1. However, in the compatibility mode, each time an input/output instruction is encountered in the 7090, the 7090 halts, trapping the 7040. The 7040 then transmits the input/output instruction to its core storage and interprets it before restarting the 7090. This multiple trapping and interpretation while the 7090 is idle results in slower input/output processing when operating in the compatibility mode.

To enter the compatibility mode, the 7090 sets up two communication cells (THIS and THIS+1). An input/output call is set up with 00009 in the decrement of THIS and the name of the system to be loaded in THIS+1. The 7090 then executes an HEY (BTTE) instruction, which causes the 7040 to trap and places the 7090 DC-IBSYS in a waiting loop. The input/output call is decoded by the 7040 trap routine WHAT, which then transfers control to routine HIP. In the HIP routine, an Enter Halt on Input/Output Primary Mode (EHM) instruction is executed, putting the system in compatibility mode. The name of the system to be loaded is then examined.

If the system to be loaded resides on disk, the system name is given on the $EXECUTE card. If the system is to be loaded from cards or tape, the word CARDS or TAPE is on the $EXECUTE card instead of the system name.

If the $EXECUTE card parameter is CARDS or TAPE, the 7040 executes an FMT (Force Multiprocess Trap) instruction, causing the 7090 to trap to a routine that breaks the waiting loop. The 7090 then transfers control to a routine that determines whether the system is to be loaded from tape or cards. This routine simulates pressing of the appropriate load button and the system is loaded in the compatibility mode.

If the $EXECUTE card parameter is not CARDS or TAPE, the system must have been edited onto the disk. If the system is a DC-IBSYS subsystem, it is scatter-loaded by the 7040. At the completion of the load, the 7040 posting routine (CPOST) starts the 7090. If the system is on the disk but is not a DC-IBSYS subsystem, it is loaded by the 7040 as a data file using a normal read routine.

GENERAL LOGIC FLOW

Figure 2 shows the general logic flow of DCOS. The figure shows the sequence of operations performed while processing one job. However, the reader should keep in mind that many jobs are being processed simultaneously. A job may be staged for processing by a routine but may have to wait until the routine has completed processing a previous job.

Job flow is controlled in the 7040 by the job queue table and the DCOS commutator. When the commutator gives control to a routine, the routine searches the job queue table for jobs that it must process. If the routine does not locate any jobs, it returns control to the commutator. If a job that requires processing is located, the routine processes the specified task, performing at least one unit of work before returning control to the commutator. For example, the print routine sets up one line of print, signals the 1403 to start, and returns control to the commutator. The next time the print routine is entered, it sets another print image, signals the 1403, and exits. In this way the 7040 is not tied up waiting for input/output units to complete their operation.

The following descriptions are concerned with 7040 tables, blocks, and routines used to process jobs. A detailed figure and explanations of the contents of the tables and blocks are included in Appendix A.

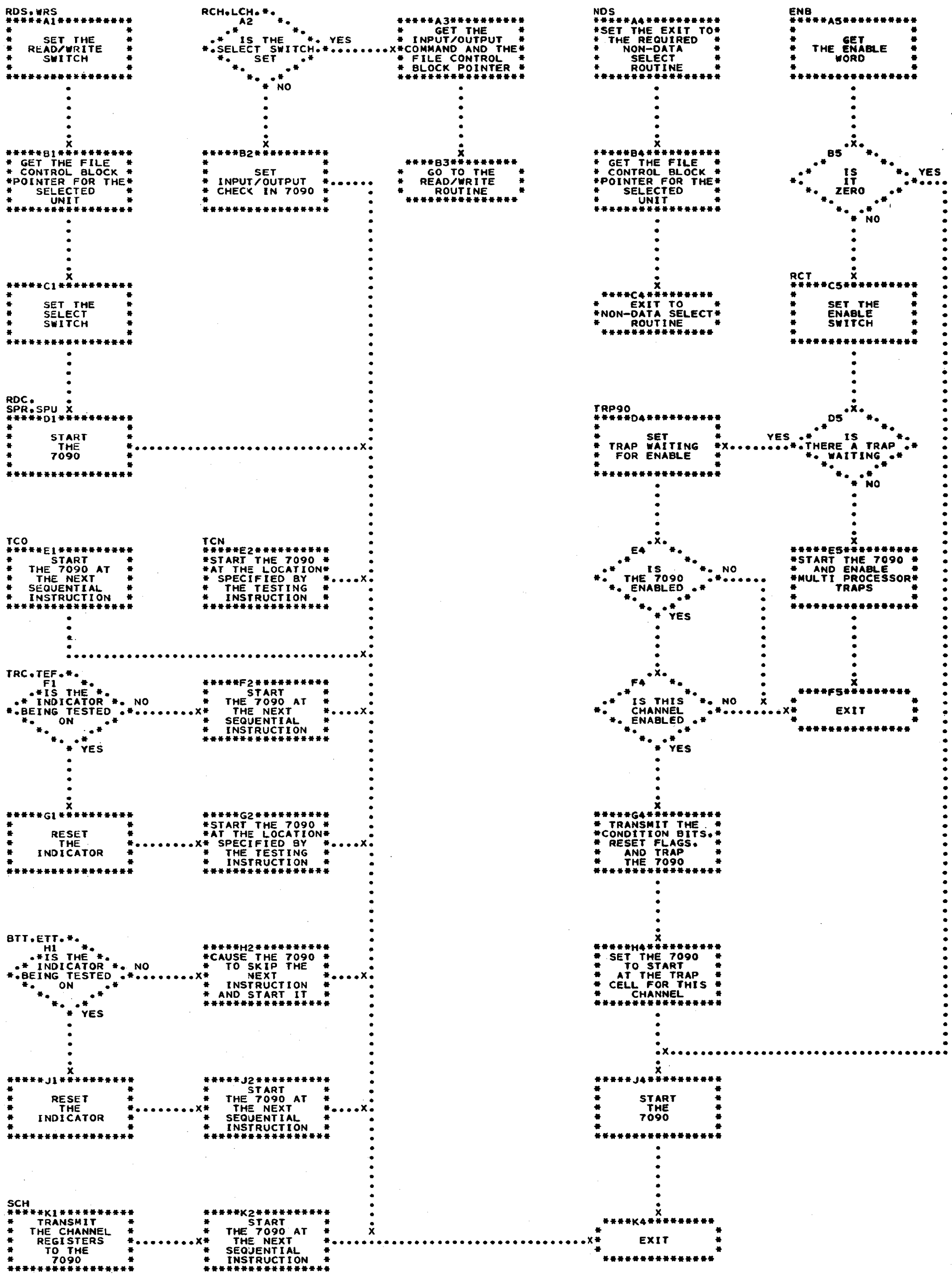

RDS.WRS
A1 SET THE READ/WRITE SWITCH
RCH.LCH.
A2 IS THE SELECT SWITCH SET
YES
NO
A3 GET THE INPUT/OUTPUT COMMAND AND THE FILE CONTROL BLOCK POINTER
NDS
A4 SET THE EXIT TO THE REQUIRED NON-DATA SELECT ROUTINE
ENB
A5 GET THE ENABLE WORD
B1 GET THE FILE CONTROL BLOCK POINTER FOR THE SELECTED UNIT
B2 SET INPUT/OUTPUT CHECK IN 7090
B3 GO TO THE READ/WRITE ROUTINE
B4 GET THE FILE CONTROL BLOCK POINTER FOR THE SELECTED UNIT
B5 IS IT ZERO
YES
NO
C1 SET THE SELECT SWITCH
C4 EXIT TO NON-DATA SELECT ROUTINE
RCT
C5 SET THE ENABLE SWITCH
RDC. SPR.SPU
D1 START THE 7090
TRP90
D4 SET TRAP WAITING FOR ENABLE
YES
D5 IS THERE A TRAP WAITING
NO
TCO
E1 START THE 7090 AT THE NEXT SEQUENTIAL INSTRUCTION
TCN
E2 START THE 7090 AT THE LOCATION SPECIFIED BY THE TESTING INSTRUCTION
E4 IS THE 7090 ENABLED
NO
YES
E5 START THE 7090 AND ENABLE MULTI PROCESSOR TRAPS
TRC.TEF.
F1 IS THE INDICATOR BEING TESTED ON
NO
YES
F2 START THE 7090 AT THE NEXT SEQUENTIAL INSTRUCTION
F4 IS THIS CHANNEL ENABLED
NO
YES
F5 EXIT
G1 RESET THE INDICATOR
G2 START THE 7090 AT THE LOCATION SPECIFIED BY THE TESTING INSTRUCTION
G4 TRANSMIT THE CONDITION BITS. RESET FLAGS. AND TRAP THE 7090
BTT.ETT.
H1 IS THE INDICATOR BEING TESTED ON
NO
YES
H2 CAUSE THE 7090 TO SKIP THE NEXT INSTRUCTION AND START IT
H4 SET THE 7090 TO START AT THE TRAP CELL FOR THIS CHANNEL
J1 RESET THE INDICATOR
J2 START THE 7090 AT THE NEXT SEQUENTIAL INSTRUCTION
J4 START THE 7090
SCH
K1 TRANSMIT THE CHANNEL REGISTERS TO THE 7090
K2 START THE 7090 AT THE NEXT SEQUENTIAL INSTRUCTION
K4 EXIT

Figure 1. Compatibility Mode

THE DCOS COMMUTATOR

In DCOS, three operations may occur at the same time: input/output functions, 7040 preprocessing and postprocessing, and data processing in the 7090. These operations are controlled by a section of DCMUP called the commutator. Figure 3 shows the logic of the commutator.

The commutator consists of a series of in-line instructions that act as gates. These gates are opened and closed to determine which set of processing routines is to be entered next. The gates are opened and closed by changing their operation codes. A gate is said to be open when its operation code is a Transfer on Index Low or Equal (TXL) and closed when it is a Transfer on Index High (TXH). Because index register zero is specified, the TXL always transfers control to the desired routine and control always falls through the TXH to the next gate.

THE JOB QUEUE TABLE

The job queue table is the DCOS work flow supervisor. As each job is read into the 7040, a one-word entry is made in the table that is used during processing to reflect the job's priority, its present status, and to provide the module and track address of the job description block for the job. The table is located and maintained in a 460-word buffer in the 7040. Each time an entry is updated, the entire table is written on track 2, module 0 (channel B) of the disk.

THE JOB DESCRIPTION BLOCK

Each job in DCOS has a job description block associated with it. These blocks are created as the jobs are read into the system. The blocks are saved on the disk and read into the 7040 each time the job enters a new phase of processing. As a job is processed, its job description block is updated to reflect the job's status and the block is rewritten on the disk (drum).

The major portion of a job description block is reserved for task descriptions. Each task that must be performed for a job during processing has a three-word task description in the job description block for that job. These task descriptions are used to indicate the type of task to be performed and the track address of the first block of data associated with the job. There is a task description for each file written on the disk. Also, at the end of each job description block there is a group of words called the peripheral track bit map, whose bits represent the disk tracks containing the input/output information for the job.

For a detailed explanation of the peripheral bit maps, the scratch bit map, and the master bit map, see the section "DCMUP Track Allocation Maps," in this manual.

UNIT CONTROL BLOCKS (7040 TAPE UNITS)

Each 729 Magnetic Tape Unit that is attached to the 7040 is represented in the 7040 by a four-word unit control block. The unit control blocks are generated by DCMUP at assembly time in accordance with assembly parameters. These blocks maintain an indication of the status of the 7040 tape units (i.e., record and file counts, unit availability, etc.).

FILE CONTROL BLOCKS

Each 7090 unit is represented in the 7040 by an eight-word file control block. The file control block links the data buffers in use by the file to a 7040 input/output device. The block is used to maintain the status of the buffers in use by the file, the location of the buffers, the location of the current logical record, and the location of the current task description in the job description block.

CHANNEL INFORMATION BLOCKS

Each channel that may be referenced by the 7090 in the compatibility mode is represented in the 7040 by a two-word channel information block. This block maintains what would be the contents of the channel registers (operation, address, and location) on a stand-alone 7090 and the channel indicators (beginning of tape, end of tape, etc.). This information is transmitted to the 7090 when the 7040 is trapped as a result of a Store Channel (SCH) instruction being executed in the 7090 or when simulating a 7090 data channel trap.

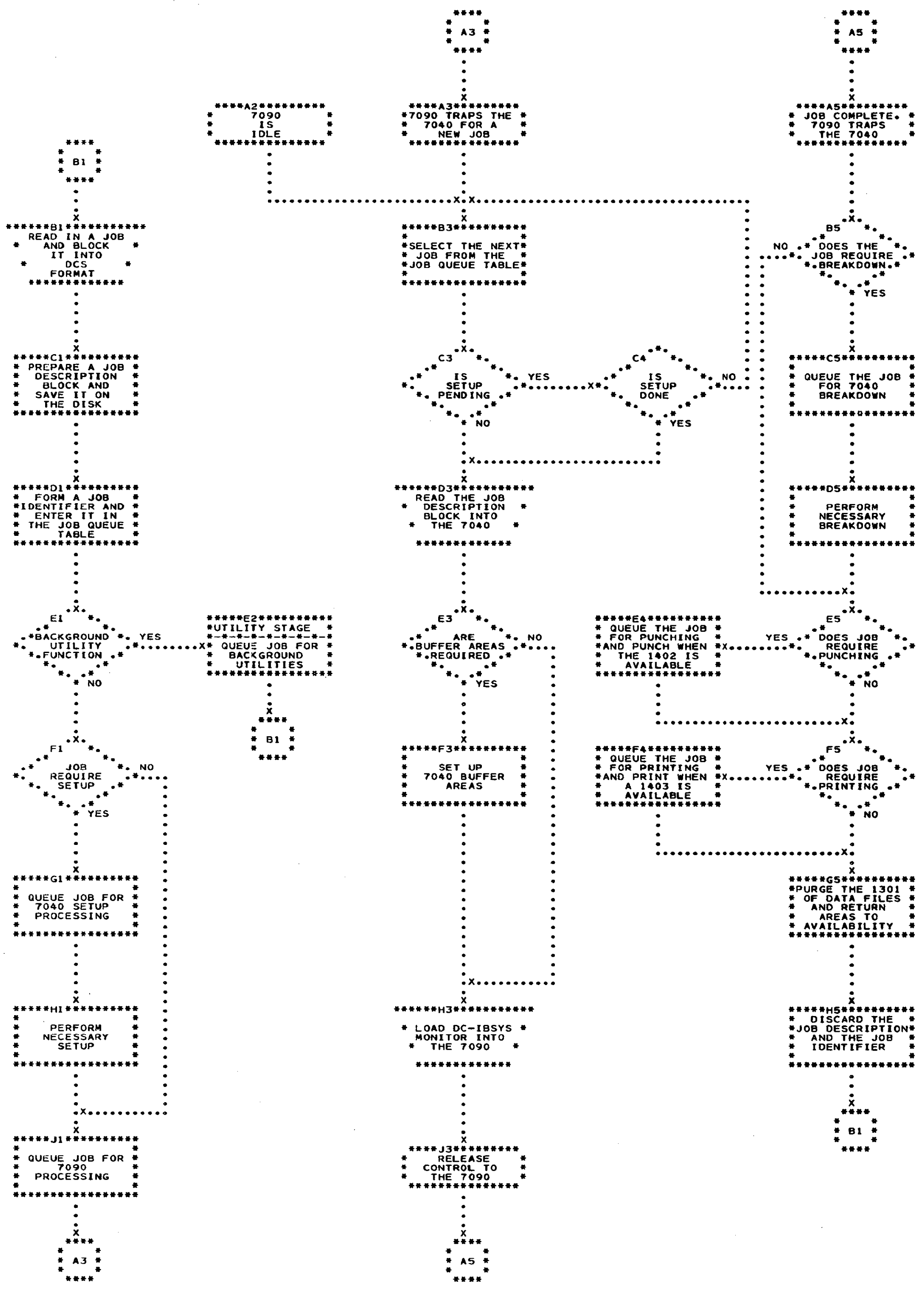

A3
A5
B1
A2 7090 IS IDLE
A3 7090 TRAPS THE 7040 FOR A NEW JOB
A5 JOB COMPLETE. 7090 TRAPS THE 7040
B1 READ IN A JOB AND BLOCK IT INTO DCS FORMAT
B3 SELECT THE NEXT JOB FROM THE JOB QUEUE TABLE
B5 DOES THE JOB REQUIRE BREAKDOWN
NO
YES
C1 PREPARE A JOB DESCRIPTION BLOCK AND SAVE IT ON THE DISK
C3 IS SETUP PENDING
C4 IS SETUP DONE
C5 QUEUE THE JOB FOR 7040 BREAKDOWN
D1 FORM A JOB IDENTIFIER AND ENTER IT IN THE JOB QUEUE TABLE
D3 READ THE JOB DESCRIPTION BLOCK INTO THE 7040
D5 PERFORM NECESSARY BREAKDOWN
E1 BACKGROUND UTILITY FUNCTION
E2 UTILITY STAGE QUEUE JOB FOR BACKGROUND UTILITIES
E3 ARE BUFFER AREAS REQUIRED
E4 QUEUE THE JOB FOR PUNCHING AND PUNCH WHEN THE 1402 IS AVAILABLE
E5 DOES JOB REQUIRE PUNCHING
F1 JOB REQUIRE SETUP
F3 SET UP 7040 BUFFER AREAS
F4 QUEUE THE JOB FOR PRINTING AND PRINT WHEN A 1403 IS AVAILABLE
F5 DOES JOB REQUIRE PRINTING
G1 QUEUE JOB FOR 7040 SETUP PROCESSING
G5 PURGE THE 1301 OF DATA FILES AND RETURN AREAS TO AVAILABILITY
H1 PERFORM NECESSARY SETUP
H3 LOAD DC-IBSYS MONITOR INTO THE 7090
H5 DISCARD THE JOB DESCRIPTION AND THE JOB IDENTIFIER
J1 QUEUE JOB FOR 7090 PROCESSING
J3 RELEASE CONTROL TO THE 7090
B1
A3
A5

Figure 2. General Logic Flow

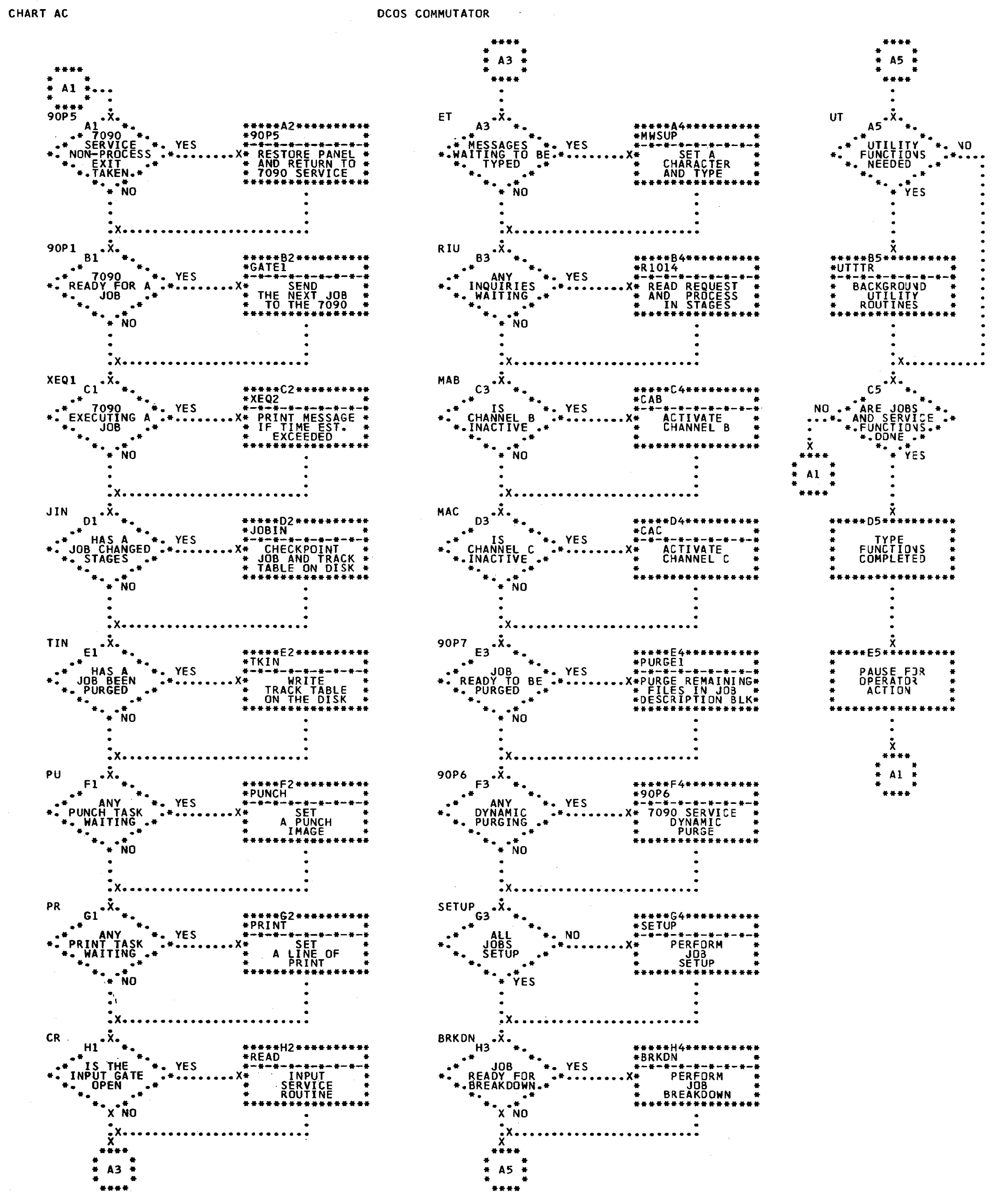


CHART AC
DCOS COMMUTATOR
A1
90P5 A1 7090 SERVICE NON-PROCESS EXIT TAKEN
YES
A2 90P5 RESTORE PANEL AND RETURN TO 7090 SERVICE
NO
90P1 B1 7090 READY FOR A JOB
B2 GATE1 SEND THE NEXT JOB TO THE 7090
XEQ1 C1 7090 EXECUTING A JOB
C2 XEQ2 PRINT MESSAGE IF TIME EST. EXCEEDED
JIN D1 HAS A JOB CHANGED STAGES
D2 JOBIN CHECKPOINT JOB AND TRACK TABLE ON DISK
TIN E1 HAS A JOB BEEN PURGED
E2 TKIN WRITE TRACK TABLE ON THE DISK
PU F1 ANY PUNCH TASK WAITING
F2 PUNCH SET A PUNCH IMAGE
PR G1 ANY PRINT TASK WAITING
G2 PRINT SET A LINE OF PRINT
CR H1 IS THE INPUT GATE OPEN
H2 READ INPUT SERVICE ROUTINE
A3
ET A3 MESSAGES WAITING TO BE TYPED
A4 MWSUP SET A CHARACTER AND TYPE
RIU B3 ANY INQUIRIES WAITING
B4 R1014 READ REQUEST AND PROCESS IN STAGES
MAB C3 IS CHANNEL B INACTIVE
C4 CAB ACTIVATE CHANNEL B
MAC D3 IS CHANNEL C INACTIVE
D4 CAC ACTIVATE CHANNEL C
90P7 E3 JOB READY TO BE PURGED
E4 PURGE1 PURGE REMAINING FILES IN JOB DESCRIPTION BLK
90P6 F3 ANY DYNAMIC PURGING
F4 90P6 7090 SERVICE DYNAMIC PURGE
SETUP G3 ALL JOBS SETUP
G4 SETUP PERFORM JOB SETUP
BRKDN H3 JOB READY FOR BREAKDOWN
H4 BRKDN PERFORM JOB BREAKDOWN
A5
UT A5 UTILITY FUNCTIONS NEEDED
B5 UTTTR BACKGROUND UTILITY ROUTINES
C5 ARE JOBS AND SERVICE FUNCTIONS DONE
D5 TYPE FUNCTIONS COMPLETED
E5 PAUSE FOR OPERATOR ACTION
A1

● Figure 3. DCOS Commutator

INPUT/OUTPUT FILE CONTROL BLOCK BASE (IOBASE)

An input/output file control block base (IOBASE) defines the relationships among simulated 7090 units, 7040 file control blocks, and 7040 physical units. An IOBASE consists of a set of file control blocks that are completely initialized. The relationship between simulated 7090 units and 7040 file control blocks is defined in an IOBASE table. Each IOBASE table is defined by a number that is specified when the

IOBASE table is defined. The IOBASE number can then be punched in a $IOBASE card, which may be included in a job to specify to the system which IOBASE table is to be used for that job. If a job does not contain a $IOBASE card, the standard DC-IBSYS IOBASE table (IOBASE 0) is used. The $IOBASE card is meaningful only for jobs that operate under a non-IBSYS system.

The standard DC-IBSYS IOBASE is defined as simulating 32 IBM 729 Magnetic Tape Units, one IBM 711 Card Reader, one IBM 716 Printer, and one IBM 721 Card Punch. The distribution of the 35 units that are simulated is:

1. Channel A--reader, printer, punch, and ten tape units
2. Channel B--ten tape units
3. Channel C--six tape units
4. Channel D--six tape units

The assignment of these units is explained in the section "PRESYS Library Preparation and Maintenance." For information on how to enter additional IOBASE definitions into the system, see Appendix B.

For a further discussion of the relationships among simulated 7090 units, 7040 file control blocks, and 7040 units, refer to the section "DCOS Unit Assignment Procedures."

DATA BUFFER FORMAT

DCOS data buffers in the 7040 are 462 words in length. A buffer includes two buffer control words and 460 words for standard DCOS physical records. Each physical record contains two physical-record control words, leaving 458 words for logical records. Preceding the first word of each logical record within a physical record is a logical-record control word. Figure 4 shows the structure of the buffers and the format of the control words.

Buffer Control Words

The first two words of a data buffer are buffer control words. The address portion of word 1 contains a buffer chain address (i.e., the location of the next buffer). The first buffer control word of the last buffer in the chain contains the location of the buffer pool control word POOL. POOL contains the location of the first available buffer in the chain. If there are no buffers available, POOL contains its own address.

When a buffer is being used for reading, the decrement portion of the first buffer control word contains the location of a read posting routine. There is a posting routine for primary buffers (PPST) and a posting routine for secondary buffers (2PST).

The second buffer control word is an input/output command that is used to read data into and write data out of the buffer. The command is of the form:

```
     IORD    *+1,,460
```

Physical-Record Control Words

The first two words in standard DCOS physical tape and disk (drum) records are control words. The contents of the two words are different for tape records than for disk (drum) records except for the decrement portion of the first word. The decrement portion of the first word always contains an internally generated job number.

For disk (drum) records, the address portion of the first physical-record control word contains the record's track address. The decrement of the second control word contains the track address of the previous physical record of the file, and its address contains the track address of the next physical record in the file.

For tape records, the address portion of the first physical-record control word contains the number of the record within the current file. The second control word contains the reel label in BCD form that was given on a $SETUP card.

Logical-Record Control Words

Preceding the first word of each logical record is the logical-record control word. The decrement of this word contains the word count of the previous logical record, and its address contains the word count of the following logical record. The remaining bits of the word, prefix and tag, are used to indicate the following.

S=0	The previous logical record is complete in this buffer.
S=1	The previous logical record is incomplete in this buffer or ends in the last word of the previous buffer.

1=0	The previous logical record is a BCD record.
1=1	The previous logical record is a binary record.
18=0	The next logical record is complete in this buffer.
18=1	The next logical record is incomplete in this buffer or starts in the first word of the next buffer.
19=0	The next logical record is a BCD record.
19=1	The next logical record is a binary record.

INPUT SERVICE ROUTINE

The input service routine enters jobs into the system, analyzes control cards, initializes job description blocks, and saves the jobs on the disk for later processing. The jobs may be entered into the system through the 1402 card reader and/or as card images from a tape unit. Tape unit C0 is used as an alternate input unit when the operator enters a code 35 in the 7040 console entry keys.

As the cards are entered into the system, they are moved from the input area (RDINA) to the input buffer (RDIN1). When the routine is ready to process another card, the next card in RDIN1 is moved to the work area (CRWKA). There it is compared to a table of control card types (i.e., $JOB, $IOBASE, etc.). When a match is found, control is transferred to the routine that processes that card type. Processing the cards consists of setting switches to indicate the type of card, saving pertinent information for other routines, and building task descriptions for the job description block. If no match is found, the card is not a 7040 control card and is passed to the 7090 when the job goes on the 7090.

JOB SETUP

Jobs are set up by a group of routines that are entered through 7040 commutator gate SETUP. At symbolic location SETUP, there is a second commutator that controls setup routines that

1. Analyze $SETUP cards and code the information for internal use.
2. Initialize 7040 unit control blocks, 7040 file control blocks, and 7040 background utilities.
3. Print tape mounting messages.
4. Execute 7040 background utilities.
5. Save information required for job execution (i.e., 7040 file control blocks, IOBASE table, 7090 unit control block information, etc.) on the disk (drum).

During the initialization phase of the setup procedure, the job queue table is searched for a job that requires setup. If there are no jobs that require setup, the 7040 setup commutator gate is closed and control is returned to the 7040 commutator at gate BRKDN. When a job requiring setup is found, its job description block is read from the disk (drum) into 7040 core storage. The address portion of the first word of the setup task description contains the location of a file that, in turn, contains the parameters of the $SETUP cards associated with the job. Each $SETUP card has a six-word entry in the file. This file, known as the setup file, is then read into 7040 core storage.

$SETUP Card Analysis and Internal Coding

The six-word entries in the setup file contain the unit in the first word and the five parameters from the $SETUP card in the remaining words. This information is coded and condensed into a four-word setup file entry. The format of the four-word entry is shown in Figure 5 and is interpreted as follows:

<u>WORD 1</u>: Field 01 (bits S-2) is a numeric code to represent option 1 and is interpreted as:

ident 1	= 0
DISK	= 1
TAPE	= 2

Field 02 (bits 3-5) is a numeric code to represent option 2 and is interpreted as:

ident 2	= 0
DISK	= 1
TAPE	= 2
NORING	= 3
null (blank)	= 4
PRINT	= 5
PUNCH	= 6
INPUT	= 7

Field 03 (bits 6-11) is a numeric code to represent option 3 and is interpreted as:

blank	= 0	
LABITS	= 1	
720	= 2	(has no function as distributed)

Unit is the unit specified, beginning in column 8 of the $SETUP card.

WORD 2: R (bits 18-20) will contain a 1 if REELS was specified in the control card. If REELS was not specified, R contains zero.

File count (bits 21-35) contains the file count parameter (FILECT) if it was specified in the control card.

WORD 3: Ident 1 is a tape reel identification if one was specified in the option 1 field of the control card.

WORD 4: Ident 2 is a tape reel identification if one was specified in the option 2 field of the control card.

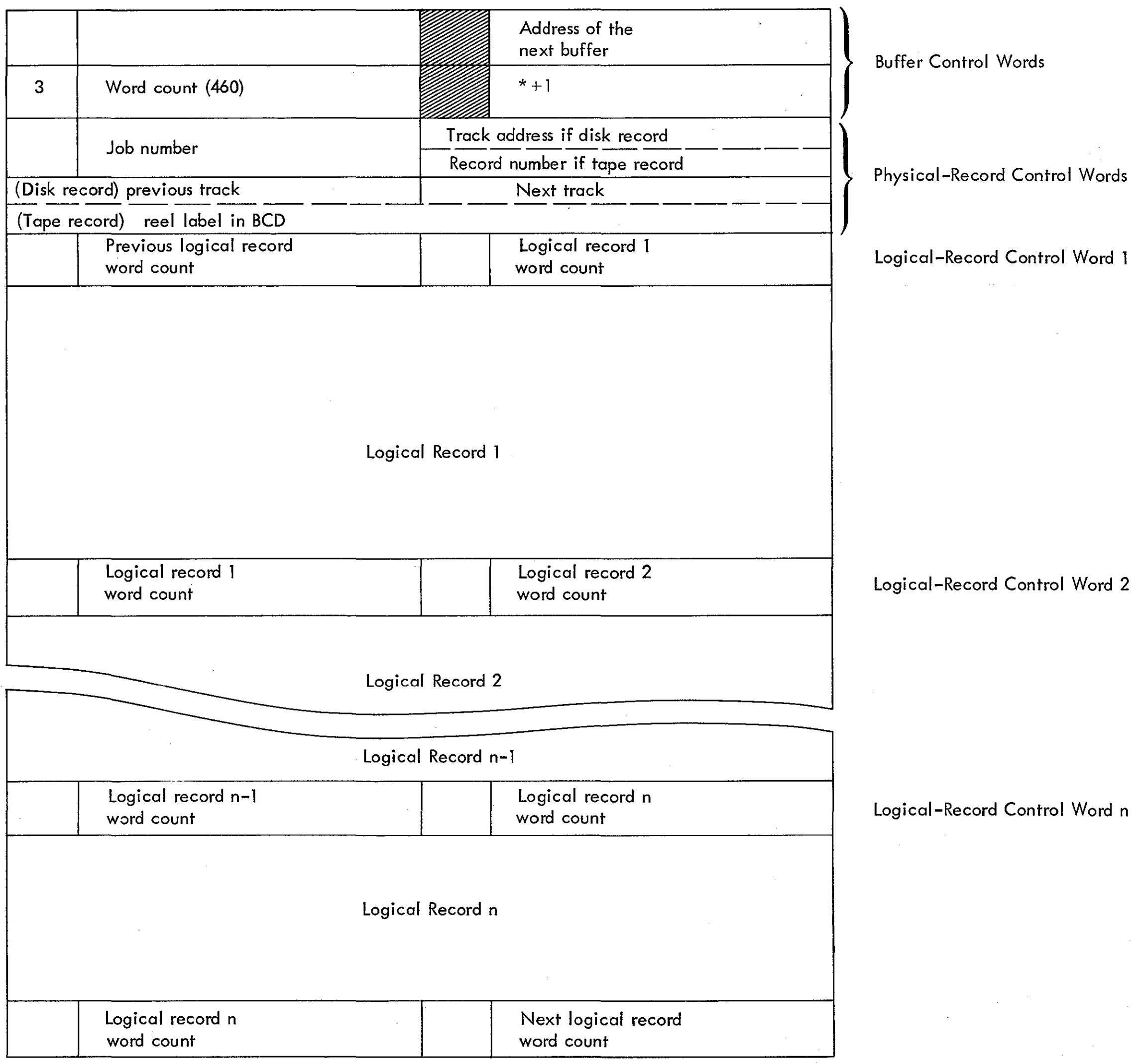
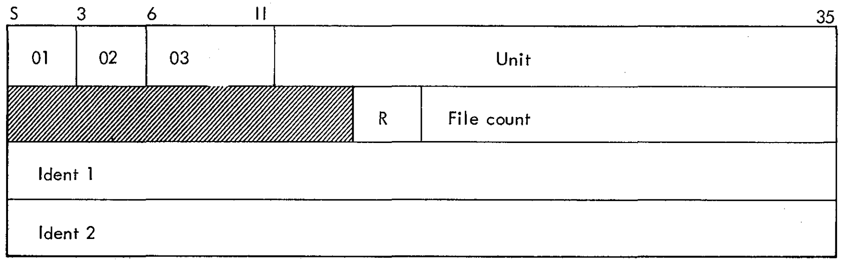

Address of the next buffer
3
Word count (460)
*+1
Buffer Control Words
Job number
Track address if disk record
Record number if tape record
(Disk record) previous track
Next track
(Tape record) reel label in BCD
Physical-Record Control Words
Previous logical record word count
Logical record 1 word count
Logical-Record Control Word 1
Logical Record 1
Logical record 1 word count
Logical record 2 word count
Logical-Record Control Word 2
Logical Record 2
Logical Record n-1
Logical record n-1 word count
Logical record n word count
Logical-Record Control Word n
Logical Record n
Logical record n word count
Next logical record word count

Figure 4. 7040 Data Buffer Format



● Figure 5. Four-Word Setup Entry

Initializing 7040 Background Utilities

The setup routine determines whether any pre-execution utility functions must be performed on the input data. If the input file is not in the DCOS standard 460-word format, it must be blocked into that format before it is processed. If a $SETUP card contains a reel identification in the option 1 field and a reel identification or the word TAPE or DISK in the option 2 field, pre-execution blocking must be performed.

If the routine determines that pre-execution blocking is required, it assigns the 7040 units that will be used by the utility routine and initializes the call to the utility.

Initialize 7040 Unit Control Blocks and File Control Blocks

The unit analysis routine, UNANYL, analyzes the unit specified on the $SETUP card. If the type of unit specified indicates that the job is to be run in the compatibility mode (i.e., a true machine unit), the IOBASE specified in the $IOBASE card is used to determine which file control block is to be used. If there was no $IOBASE card with the job, the standard DC-IBSYS IOBASE is used.

For all jobs that require setup, an available file control block is selected for each unit specified on a $SETUP card. For IOBASE 0, an available file control block is one that is not reserved for a specific library function, a peripheral function, or a utility function. For IOBASE1 and IOBASE2, an available file control block is one that is not reserved for a library function or a peripheral function. Any available file control block may be used for direct mode operation. The specified IOBASE table defines the file control blocks that are to be used for compatibility mode.

A table is then generated for DC-IBSYS. This table, the UNISYM table, contains a two-word entry for each unit that was set up. The UNISYM table is used by DC-IBSYS to initialize the 7090 unit control blocks. The first word of the two-word entry contains the logical unit number of the 7040 file control block assigned to the unit. The second word contains the BCD name of the unit.

If a tape unit is required for job execution, it is selected from the available units on the 7040. If a reel of tape must be mounted on the unit, a unit that has been rewound and unloaded is chosen. If there are none, a unit that is ready is rewound and unloaded. If neither a ready unit nor a rewound or unloaded unit is available, setup for this job cannot be completed. If the number of tape units required during the processing phase is greater than the number assembled into DCMUP, the job is deleted. However, the job normally remains in the setup stage and commutator cycles are taken until a unit becomes available. If the 7090 becomes idle while setup is pending, the following message is typed:

```
4*hhmmss    7090 IS IDLE, SETUP PENDING
```

If a work tape is required, a unit that is ready is chosen. If there are no ready units available, a unit that has been rewound and unloaded is chosen.

When a tape unit has been selected, the location of its 7040 unit control block is placed in word 1 of the 7040 file control block that corresponds to the 7090 unit control block for the unit. The job number is then placed in word 3 of the 7040 unit control block.

Print Mounting Messages

The print mounting messages routine prints all messages that inform the operator where to mount tapes and which units to make ready. This includes any units that are required for pre-execution utilities.

Execute Pre-Execution Utilities

The execute pre-execution utilities routine tests the status of all units assigned for use by the utility routine. When all units are ready, control is transferred to the appropriate utility routine. For a description of the utility routines, see the section "Background Utility Routines." When the utility routine has completed its operation and returned control to setup, all units that were assigned for input to the utility routine are rewound, unloaded, and made available for further use.

Save Pertinent Job Information on the Disk

The initialized file control blocks and the IOBASE table are written on the disk (drum) for use by the 7040 during 7090 execution time. The coded $SETUP card information (four-word entries) is saved on the disk (drum) for use by the breakdown routine. A task description entry is created in the job description block for the UNISYM table, and the table is saved on the disk (drum). The UNISYM table is called by DC-IBSYS by a special input/output call.

The job is now queued for 7090 execution, and the setup routine searches the queue table for the next job requiring setup.

INITIALIZATION FOR 7090 EXECUTION

When the 7090 has completed a job, it signals the 7040 that it is ready for the next job. The 7040 then searches the job queue table to locate the next job. When a job is located, the 7040 scans its unit control blocks to locate any units that are required for the job. The job number in the third word of the unit control block identifies the job for which the unit was set up. If no units are required, the job is passed to the 7090 for execution. If units are required, their status is tested to determine whether they are ready. If all of the units required for the job are ready, the job is passed to the 7090 for execution. If any of the units is not ready, the job is rescheduled for 7090 execution and the 7040 continues searching the job queue table. If the 7040 reaches the end of the job queue table before locating a job that is ready for 7090 execution, the message

    THE 7090 IS IDLE, SETUP PENDING

is printed on the 7040 console, and the search recycles. If there are no jobs in the job queue table awaiting execution, the message

    THE 7090 IS IDLE

is printed and the search is terminated.

JOB BREAKDOWN

The job breakdown routine is entered through 7040 commutator gate BRKDN. This routine rewinds and releases all tape units used for the job and initializes and executes any post-execution utilities that are required.

The breakdown routine rewinds and unloads all remaining tapes that were mounted for the job during job setup. All work-tape units that were assigned during job setup are rewound, and all but those required for post-execution utility processing are returned to available status. If the post-execution utilities are not required, all tape units that were assigned to the job during job setup are made available to the system.

The breakdown routine uses the coded four-word entries generated by the setup routine to determine whether any post-execution utilities are required. If any utilities are required, the units needed to execute the utility are assigned. If any tapes must be mounted, the mounting messages are printed and the breakdown routine waits until all units assigned for the utility are ready. Control is then transferred to the appropriate utility routine. When the utility routine completes its operation, control is returned to the breakdown routine, where all assigned units are returned to available status.

PUNCH STAGE

The punch routine may be thought of as two routines: the punch staging routine and the punch activating routine. The staging routine fills three card-image buffers, so that the activating routine can punch all cards from a single source. The cards to be punched may be BCD or binary cards. The activating routine also provides repunching of cards in the event of hole-count errors.

Punch Staging Routine

Entry to the punch routine is through commutator gate PU to symbolic location PUNCH. The punch routine exit is at symbolic location PUOUT, where control is returned to the commutator.

Upon initial entry to the punch routine, there is no job description block in storage to control operation. The job queue table is scanned to locate a job waiting to be punched; when found, its job description block is read in from disk (drum) storage. If there are no jobs waiting to be punched and previously initiated punching has been completed, the commutator punch gate (PU) is closed and control is returned to the commutator.

When a job description block has been read into storage, a job separator card is set up for punching. The job separator card consists of columns 31 through 60 of the $JOB card in columns 1 to 30 of the separator card and the date in columns 31-36, and nines in columns 61 through 80. Each task description is then tested to determine if it is a punch task.

When a punch task is found, the data block is read from disk storage and the card images are staged for punching. The track address of the data block is found in the address portion of word 1 of the task description. The mode that the 7090 used to write the punch card record determines whether a BCD or a binary card will be punched. The record length is checked. If the length exceeds 14 words for BCD records (28 for binary), all punching for the job is discontinued.

After all task descriptions in a job description block have been tested, the job queue table entry for that job description block is staged for printing.

Punch Activating Routine

The punch activating routine controls mechanical movement of the IBM 1402 Card Punch. It also contains the punch error-recovery routine.

A pointer is used to determine which buffer is to be punched next. When this pointer indicates that a buffer is full and ready to be punched, tests are made to establish that the punch is ready, that it is not busy, and that a punch error has not occurred. If the punch is not ready or is busy, the punch routine returns control to the commutator. If the tests indicate that punching is possible, the punch-buffer control word is decoded. The control word is used to build the select instruction and the input/output command. The select instruction and the input/output command are then executed, followed by a test to detect core-storage-to-punch-buffer transfer errors. If any errors are detected, one retry is attempted. If an error is detected on the second attempt, the card is bypassed and a message is printed on the console typewriter informing the operator of the condition.

If the sense instruction prior to the next punch select indicates that a hole-count error was detected, a message is printed on the console typewriter instructing the operator to clear the punch station. The buffer control words of the previous two buffers are reset to plus, to indicate not punched, and the punch-buffer pointer is set back two buffers. The routine then tests the punch for a not-ready condition followed by a ready condition, which indicates operator action is complete. Normal punching then continues.

All cards that are punched are placed in the 4 pocket with the exception of binary cards that contain a 7-8-9 punch in column one. These cards are placed in the 8/2 pocket.

PRINT STAGE

Entry to the print routine is through commutator gate PR to symbolic location PRINT. This single entry services all attached printers. At the beginning of the print routine, there are three instructions for each printer. These instructions set a transfer instruction and initialize index register 4. Index register 4 is used by the print routine to refer to the printers. Each set of three instructions transfers control to a common print routine. This common routine services the printer referred to by index register 4 and returns control, by means of the transfer instruction, to the next set of three instructions for the next printer. This approach allows all printers to be serviced by one commutator gate (PR) and one common routine.

The print gate (PR) is opened by the 7090 service routine and remains open until all jobs waiting to be printed have been completed. As each job is completed, its job queue table entry, bits 14-17, are set to indicate that the job is ready for the next stage.

Each printer usually requires one 460-word buffer at a time. However, at the end of each job, one printer holds (while generating the accounting information) two buffers: one for the job description block in which two print tasks for the accounting information are being created; and one for the accounting information itself. If, at the beginning of DCMUP, the assembly parameter 2BF is set to 1, the printers will be double-buffered (each printer will require two buffers most of the time). The print routine always expects print images to be in BCD character representation, with or without record marks within logical records. If editing or binary mode is desired, modifications to the print routine are required. Logical records are transmitted from the 7090 DCOS record control words.

When a job enters the print stage, a logical record is scanned (this scanning is under the control of the logical record control words) until either a record mark is found or the end of a logical record is reached. If a record mark is detected, the deblocked line is printed as one line, and, upon commutator return to the print routine, scanning of the logical record is continued. If the end of a logical record is reached (or no record mark is detected), the line is printed as one line, and, upon commutator return to the print routine, scanning begins with a new logical record at location SCANX.

There is no need to use

```
        $SETUP  OU1  DISK,PRINT,720
```

If it is used, one unnecessary SETUP stage is executed which might marginally degrade performance. There is no other effect.

BACKGROUND UTILITY ROUTINES

There are eight utility routines under DCMUP control. These utilities operate on a time-sharing basis. Four of the routines are called and used by the setup and breakdown functions and three are controlled by the DCMUP utility monitor. One routine, the 720A disk-to-printer deblocking, is not used by the system as distributed. Only one utility routine can be in 7040 core storage at a time. The required routine is read into core storage starting at location 192 (300).

Setup and Breakdown Utilities

The setup and breakdown utilities are:

1. Tape-to-tape blocking
2. Tape-to-disk blocking
3. Tape-to-tape deblocking
4. Disk-to-tape deblocking

The two blocking routines are used for job setup, and the two deblocking routines are used for job breakdown. The routines are called if the analysis of the $SETUP cards indicates a need for them.

The blocking routines block input data records into the standard DCOS 460-word format. The input data records must be no longer than 457 words. During the blocking process, the blocking routines insert a logical-record control word in front of each physical input record. The logical-record control words represent the interrecord gaps on the input tape and give the length, in words, and the mode of the record.

The deblocking routines deblock the standard DCOS 460-word output records into the lengths specified by the logical-record control words. As records are transmitted from the 7090, they are placed into 460-word buffers. Each record is preceded by a logical-record control word. When a buffer is filled, it is written on the disk or on a 7040 tape unit. If so indicated on the $SETUP card, these 460-word records are deblocked during the breakdown stage of postprocessing and written onto a tape in the lengths specified by the logical-record control words. The maximum length data record that can be deblocked is 457 words.

During execution of the tape-to-tape blocking or deblocking utility routines, an end-of-tape condition will cause an end-of-file mark to be written on the output tape. A message will be issued instructing the operator to mount a blank reel. The programmer must take into account the end-of-file mark at the end of the reel when the tape is used as input.

Monitored Utility Routines

The monitored utility routines are:

1. Card-to-tape

2. Tape-to-printer
3. Tape-to-punch

These routines are called by the utility monitor if analysis of the $UTILITY card indicates a need for them.

The card-to-tape routine blocks an input card deck into standard 460-word DCOS data records. The tape-to-printer and tape-to-punch routines write standard 460-word DCOS data records onto the disk and then make a task entry specifying either print or punch and giving the location of the transcribed file. The utility then requests staging of this job into either the print or punch stage and allows DCMUP standard procedures to complete the utility operation.

During execution of the card-to-tape utility routine, an end-of-tape condition will cause an end-of-file mark to be written on the output tape. A message will be issued instructing the operator to mount a blank reel. The programmer must take into account the end-of-file mark at the end of the reel when the tape is used as input.

7040/7044 UTILITY ROUTINES

The three 7040/7044 IBJOB tape-blocking and deblocking routines which operate under the 7040/7044 Operating System (16/32K) included on the DCOS Distribution Tape are:

1. Tape Blocking Routine: This routine blocks tapes into the standard DCOS record format.[1]
2. System Tape Blocking Routine: When a non-IBSYS system tape is to be edited into the system library, it must contain a loading program and must be blocked into the standard DCOS format. The System Tape Blocking Routine performs these functions.
3. Tape Deblocking Routine: This routine deblocks any tape records written in standard DCOS format into their original record format.[1]

These utility routines are relocatable object programs written for the 7040/7044 Operating System (16/32K) which has been assembled for 32,768 words of core storage. No other routine or system, including DCOS, can operate while the 7040/7044 System is in control.

Each reel of tape to be blocked or deblocked requires one control card. This card should be placed after the appropriate utility program deck. The format of this card is:

1-30	31-36	37-42
job ident	output reel number (right justified)	number of files to be processed (right justified)

where:

job ident is the identification assigned by the user; it may be any combination of characters and blanks.

output reel number is the numeric identification assigned to the output reel by the user. (The control card for the system tape blocking routine should have the system name (left justified) in columns 31-36.)

number of files to be processed must be a decimal integer.

[1]Records that are to be blocked may contain 3 to 17,400 words per record. Conversely, logical records that are to be deblocked may contain 3 to 17,400 words.

Multiple reels of tape may be blocked or deblocked during the same run, with each reel specified on a separate control card. Multiple input reels will yield an equal or greater number of output reels (a single input reel may yield more than one output reel).

Prior to actual job execution, the following tape-mounting message will be issued to the operator:

SET UP JOB job ident MOUNT INPUT ON xx AND SCRATCH TAPE ON yy

where xx and yy are 7040 tape units.

Should a read error occur during processing, a message identifying the record and file number of the record in error will be printed on the 1403. The record will be assumed to be valid, and processing will continue.

If an end-of-tape mark is encountered on the output tape before processing has been completed, the system will take the following action:

1. A message noting the condition and identifying the record and its reel number will be printed off-line
2. The output reel will be terminated.
3. Processing will continue on an alternate unit with the reel number increased by one.

After the specified number of files have been blocked or deblocked, the following message will be typed:

JOB job ident IS COMPLETED, TAPE ON yy IS NOW TAPE NUMBER output reel number

where yy is a 7040 tape unit.

Blocking and deblocking operations will be immediately terminated if sense switch 2 is on.

Procedures for operating the 7040/7044 IBSYS Operating System are described in the publication IBM 7040/7044 Operating System (16/32K): Operator's Guide, Form C28-6338.

The IBSYS Operating System Monitor with Direct Couple Capabilities (DC-IBSYS Monitor) controls the 7090 side of DCOS. The DC-IBSYS Monitor consists of:

1. The DC-IBSYS Supervisor, whose primary functions are to control and coordinate the processing of jobs. The supervisor performs job initialization, passes control from one IBSYS subsystem to another, and releases control to DCMUP whenever it is necessary to load a non-IBSYS system.
2. The System Core Storage Dump Program, which can facilitate the testing and analysis of any program executed under control of DC-IBSYS.
3. The System Editor, which facilitates the modification and maintenance of the entire Direct Couple Operating System and the 7090 subsystems (including non-IBSYS systems) operating under it.
4. The DC-IBSYS Nucleus, which remains in core storage during execution of all IBSYS jobs and provides facilities for communication and control among the DCOS Multiprocessor, the DC-IBSYS Monitor, and the 7090 subsystems.
5. The Direct Couple Input/Output Executor (DC-IOEX), which remains in core storage during job execution of IBSYS systems to coordinate and control input/output via the DC channel.

The DC-IBSYS Monitor may also contain an installation accounting routine tailored to the specific requirements of the installation.

The DC-IBSYS Supervisor, the DC-IBSYS Nucleus, and DC-IOEX are described in detail in the following sections. The System Editor is described in the section "System Editor." Details about the System Core Storage Dump Program may be found in the publication IBM 7090-7040 Direct Couple Operating System: Programmer's Guide, Form C28-6382.

In the discussion that follows, the terms unit, channel, tape, card reader, etc., do not refer to physical input/output devices attached to the 7090. The DC channel is the only real device attached to the 7090. Rather, above terms refer to input/output devices that are simulated by the 7040 side of DCOS.

This section concerns IBSYS subsystems and IBSYS job segments only. These may be either in direct mode (IBJOB) or compatibility mode, unless specifically noted. Non-IBSYS job segments operate only in compatibility mode.

DC-IBSYS SUPERVISOR

The DC-IBSYS Supervisor coordinates and supervises the processing of each job by initializing the job, by initializing IBSYS job segments, and by passing control from one IBSYS subsystem to another.

INITIALIZATION PROCEDURES

Jobs are initialized on the 7090 in the following manner:

1. for the first job, or
2. for the first job after a non-IBSYS job.

The CLEAR routine,

1. clears core, except for the nucleus area of IBSYS,
2. resets indicators,
3. clears traps, and
4. loads IBSYS.

If a refreshed IBSYS is in core, i.e., if:

1. a $IBSYS card had been followed by a $STOP card, and
2. no DUMP parameter was specified on a $ATEND card,

then CLEAR is not loaded.

If CLEAR is not to be loaded for the next job (a refreshed IBSYS is already in core), IBSYS execution for the next job begins immediately after processing a $STOP card for the previous job, without any intervening halt of the 7090. The 7090 will come to a halt before loading DCIDR and wait for DCMUP to restart it upon getting a new job.

When the supervisor gains control, it moves the nucleus to lower core storage and generates unit control blocks. Before each job segment (including the first) the DC-IBSYS supervisor performs unit assignment functions, as described in the section "DCOS Unit Assignment Procedures."

BASIC SUPERVISOR CONTROL CARDS

The DC-IBSYS Supervisor is directed to perform many of its functions by control cards, which it reads from a system input file, interprets, and acts upon. The most frequently used DC-IBSYS Supervisor control cards are:

$JOB
$EXECUTE
$ID
$*

Each subsystem and the System Editor, as well as DC-IBSUP, recognize the $EXECUTE and $ID cards. The $JOB card is encountered only by the DC-IBSYS Supervisor. The actions taken by the DC-IBSYS Supervisor when a $EXECUTE or $ID card is encountered are described in the following text.

$JOB CARD

The $JOB card is required for each job. It defines the beginning of a job. A job consists of all of the cards beginning with a $JOB card and ending with, but not including, the next $JOB card. A job may consist of any logical combination of job segments to be performed by the subsystems and the DC-IBSYS Monitor. The $JOB card causes control to be transferred to the installation accounting routine (if one exists at the installation). A units printout will occur which is a table giving the relationship of simulated 7090 units to 7040 FCBs and actual units.

The format of the $JOB card is:

```
1              16
$JOB           [priority], [time estimate]

                    31                 60
   [line estimate]  [job identification]
```

The parameters are:

priority
: This is the priority assigned to the job. The digits 0 through 9 may be specified. A non-numeric character may also be specified, and the low-order four digits of its BCD representation will be used as the priority number. A priority of 0 is the lowest priority that can be assigned. When this parameter is omitted, a priority of 0 is assumed.

time estimate
: This is the estimated total 7090 processing time, in minutes. A maximum specification of 32,767 minutes is allowed.

line estimate

This is the estimated line count of the printed (1403) output from the job. A maximum specification of 262,143 lines is allowed.

If either the time estimate or the line-count estimate is exceeded, the operator is notified and DCMUP initiates termination procedures. If either or both estimates are omitted, the assembled maximum values of DCMUP, symbols SDLNCT and SDTMET, are assigned.

job identification

Columns 31 through 60 are normally used to identify a job and may contain any combination of characters and blanks.

$EXECUTE CARD

The $EXECUTE card defines the beginning of a job segment. When a $EXECUTE card is read by the DC-IBSYS Supervisor, the Supervisor scans its direct-mode name table (DCTBL) to determine if the parameter refers to a direct-mode IBSYS subsystem that resides on the disk. In the distributed version, IBJOB is the only direct-mode subsystem. If a direct-mode system is to be executed, the DC-IBSYS Supervisor traps the 7040 by executing an HEY instruction with a code of 8, passing the name specified on the $EXECUTE card to DCMUP. The 7040 responds by scatter-loading the subsystem, trapping the 7090 when loading has been completed.

If the $EXECUTE card does not specify a direct-mode subsystem, the DC-IBSYS Supervisor traps the 7040 by executing an HEY instruction having a code of 9. In this case, the 7040 executes an Enter Halt on Input/Output Primary Mode (EHM) instruction, placing DCS in compatibility mode.

If the CARDS option is specified on the $EXECUTE card, the 7040 traps the 7090 into a routine that simulates pressing of the 7090 LOAD CARD button. A $ROW card must follow the $EXECUTE card, and a self-loading card loader must follow the $ROW card. The deck to be loaded must be terminated by a $ENDROW control card.

If the TAPE option is specified on the $EXECUTE card, the 7040 traps the 7090 into a tape-load-simulation routine for 7090 unit A1. Only non-IBSYS systems may be executed from tape. The tape must have been specified on a $SETUP card for unit A1 and must be in standard DCOS format.

If the DC-IBSYS Supervisor encounters a $IBJOB control card, the Supervisor will simulate a $EXECUTE card specifying IBJOB.

If a $EXECUTE card is read by a subsystem whose name is not the same as that specified on the card, control is returned to the DC-IBSYS Supervisor. The Supervisor causes loading of the new segment as it would if it had read the $EXECUTE card. When a subsystem reads a $EXECUTE card specifying itself, the subsystem retains control.

$ID CARD

The $ID card is used for intrajob accounting purposes at installations that employ an installation accounting routine. It causes a transfer to the installation accounting routine. Upon completion of the accounting routine, the next card in the system input file is read and processed.

The distributed version of DCOS does not contain an installation accounting routine. Details about designing such routines may be found in the section "Designing an Installation Accounting Routine."

The exact use and placement of the $ID card depends upon the accounting practices used at a particular installation.

UNIT ASSIGNMENT CONTROL CARDS

Unit assignment control cards are used to specify temporary reassignment of logical system unit functions. A system unit (SYSUNI) function is designated on the control card as SYSxxx or SYSyyy, where SYSxxx and SYSyyy are the symbolic names for the systems unit functions listed in Figure 7. <u>SYSLB1 may not be specified on a unit assignment control card.</u>

SYSCK1 and SYSCK2 may be used in DCOS, although not as checkpoint units. They may be validly specified on any unit assignment control card. Details about the logic of DCOS unit assignment can be found in the section "DCOS Unit Assignment Procedures."

$ASSIGN CARD

The $ASSIGN card causes the specified system unit function to be assigned to a unit. The format of the $ASSIGN card is:

```
1              16

$ASSIGN        SYSxxx
```

Selection of the unit is governed by whether a $SETUP card referring to the same system function was included among the 7040 control cards for that job.

If a $SETUP card was not used and if the specified system unit function has already been assigned, the $ASSIGN card has no effect. Otherwise, a unit is selected from one of the unit availability chains in the DC-IBSYS Nucleus and is assigned to the specified system unit function.

If a $SETUP card was used, the unit control block that was assigned a unit symbol corresponding to the symbol on the $ASSIGN card is selected and assigned to the specified system unit function. If another unit had been assigned to the specified system unit function, it is released as stipulated in the description of the $RELEASE card.

$RELEASE CARD

The $RELEASE card causes the unit assigned to the specified system unit function to be released from that function. The unit is returned to the appropriate availability chain unless it was specified on a $SETUP card or on a previous $ASSIGN card. The format of the $RELEASE card is:

```
1              16

$RELEASE       SYSxxx
```

$RESET CARD

The $RESET card causes the assignment of units to system unit functions and the release of units from system unit functions according to the following rules:

1. System unit functions that were assigned at assembly and are currently assigned remain assigned.
2. System unit functions that were not assigned at assembly and are currently unassigned remain unassigned.
3. System unit functions that were not assigned at assembly but are currently assigned are released.
4. System unit functions that were assigned at assembly and are currently unassigned are reassigned. An attempt is made to assign such functions to the units to which they were originally assigned. If, however, the original unit is reserved (bit 3 of word 1 of the unit control block is a 1) or the unit is assigned another function (word 4 of the unit control block is not zero), the system unit function is assigned to a unit selected from an availability chain.

The format of the $RESET card is:

```
1

$RESET
```

$SWITCH CARD

The $SWITCH card causes the units assigned to the two specified system unit functions to be transposed; that is, the unit assigned to SYSxxx is assigned to SYSyyy, and the unit that was assigned to SYSyyy is assigned to SYSxxx. The format of the $SWITCH card is:

```
1              16

$SWITCH        SYSxxx,SYSyyy
```

Note: The contents of the fourth word of the unit control blocks are not switched.

$CARDS CARD

The $CARDS card causes the DC-IBSYS Supervisor to read succeeding control cards from the system card reader (SYSCRD).

The format of the $CARDS card is:

```
1

$CARDS
```

Note: A $ROW-$ENDROW card group encountered in the flow of input to the 7040 defines the limits of the data file for the card reader. These data cards are converted by the 7040 to card-image form before being passed to the 7090 as input.

$TAPE CARD

The $TAPE card causes the DC-IBSYS Supervisor to read succeeding control cards from the unit assigned as the system input unit (SYSIN1). The format of the $TAPE card is:

```
1

$TAPE
```

TAPE-MANIPULATION CONTROL CARDS

Four control cards enable automatic manipulation of tape units assigned to system unit functions. If no unit is assigned to the system unit function specified on a tape-manipulation control card, or if a card or printer unit is assigned to the function, the card has no effect.

Note: SYSLB1 may not be specified on any tape-manipulation control card.

$ENDFILE CARD

The $ENDFILE card causes an end of file to be written on the unit assigned to the specified system unit function. No test is made to determine invalid operations such as writing an end of file on the system input unit.

The format of the $ENDFILE card is:

```
1              16

$ENDFILE       SYSxxx
```

$REWIND CARD

The $REWIND card causes a unit assigned to the specified system unit function to be rewound.

The format of the $REWIND card is:

```
1              16

$REWIND        SYSxxx
```

$REMOVE CARD

The format of the $REMOVE card is:

```
1              16

$REMOVE        SYSxxx
```

If the system function specified on the $REMOVE card is being simulated on the disk, the 7040 effects a simulated rewind only. If the function has been assigned to a 7040 tape unit, the tape unit is rewound and unloaded. If the unit is again referred to by a read or write instruction, the operator will be notified of a tape-not-ready condition. In this case, clarification to the operator should be provided by $PAUSE or $* cards.

$UNLOAD CARD

The $UNLOAD card has the same effect as the $REMOVE card. Its format is:

```
1              16

$UNLOAD        SYSxxx
```

MISCELLANEOUS CONTROL CARDS

$STOP CARD

This card is interpreted as meaning the end of the current job. Control of the 7090 is returned to the 7040, thus initiating termination procedures.

The format of the $STOP card is:

```
1

$STOP
```

The $STOP card is not required, since a $IBSYS and a $STOP card are generated by DCMUP and are placed at the end of the system input.

$IBSYS CARD

When the $IBSYS card is read by a subsystem or the System Editor, the DC-IBSYS Supervisor is called into core storage and control is relinquished to it. If a compatibility-mode IBSYS subsystem was in control, the return to IBSYS causes DCOS to leave compatibility mode. The Supervisor then processes succeeding control cards until control is relinquished to a subsystem because of a $EXECUTE card or to the System Editor because of a $IBEDT card.

The format of the $IBSYS card is:

```
1

$IBSYS
```

DC-IBSYS SUPERVISOR CONTROL CARDS

The DC-IBSYS Supervisor must be in control to process any of the control cards described in the remainder of this section.

$RESTORE CARD

The $RESTORE card causes the restoration of the DC-IBSYS Nucleus as specified by assembly parameters.

Form C28-6383-2
Page Revised 6/11/65
By TNL N28-0158-0

The format of the $RESTORE card is:

```
1
$RESTORE
```

The $RESTORE card causes the DC-IBSYS Monitor to be called into core storage from the disk, producing the same effect as an initial start except that the tape positions are not disturbed. The effect of all previous control cards is canceled except that the $RESTORE card does not affect the source of control card input as specified by $CARDS or $TAPE cards. However, the unit assigned as the system input unit may change as a result of the $RESTORE card if a different unit had been assigned by a $ASSIGN or $SWITCH card.

$IBEDT CARD

Upon recognizing the $IBEDT card, the DC-IBSYS Supervisor calls the System Editor into core storage from the disk and relinquishes control to it.

The format of the $IBEDT card is:

```
1
$IBEDT
```

Note: Control cards that are recognized by the System Editor are described in the section "System Editor."

$UNITS CARD

A units printout at the beginning of each job gives the relationships between simulated 7090 units and their functions and 7040 logical and physical units, along with the symbolic name for each unit. The $UNITS card may be used to get this same type of printout at other times during the processing of a job. A $UNITS card will cause a units printout whenever it is encountered by DC-IBSYS in the input stream.

The format of the $UNITS card is:

```
1
$UNITS
```

Figure 5A is an example of a units printout.

```
90 UNIT       RD     PU     PR     A1     A2     A3     A4     A5     A6     A7     A8
FUNCTION     CRD    PCH    PRT    LB1    IN1    OU1    PP1    CK1
SYMBOLIC                                                           LB2    UT3    UT1
40 LOGICAL    32     33     34     00     01     02     03     04     05     06     07
40 UNIT     DISK   DISK   DISK   DISK   DISK   DISK   DISK   DISK     C1     C2     C3

90 UNIT       A9     A0     B1     B2     B3     B4     B5     B6     B7     B8     B9
FUNCTION                   UT1    UT2    UT3    UT4    CK2
SYMBOLIC
40 LOGICAL    08     09     10     11     12     13     14     15     16     17     18
40 UNIT     DISK   DISK   DISK   DISK   DISK   DISK   DISK   DISK   DISK   DISK   DISK

90 UNIT       B0     C1     C2     C3     C4     C5     C6     D1     D2     D3     D4
FUNCTION
SYMBOLIC
40 LOGICAL    19     20     21     22     23     24     25     26     27     28     29
40 UNIT     DISK   DISK   DISK   DISK   DISK   DISK   DISK   DISK   DISK   DISK   DISK

90 UNIT       D5     D6
FUNCTION
SYMBOLIC
40 LOGICAL    30     31
40 UNIT     DISK   DISK
```

●Figure 5A. Units Printout

The DC-IBSYS Nucleus (DC-IBNUC) is a portion of the DC-IBSYS Monitor that remains in core storage at all times when DC-IBSYS is in control. It provides common facilities for communication and control among the Monitor, the 7090 subsystems operating under it, and the 7040 DCOS Multiprocessor. The DCOS Input/Output Executor and the unit control blocks for the input/output units are described separately in the section "DCOS Input/Output Executor," although they may be considered part of the System Nucleus, because they remain in core storage and provide common communication facilities.

The DC-IBSYS Nucleus consists of the following:

1. A communication region, which contains constants, control words, and transfer words required for communication among the DC-IBSYS Monitor, 7090 subsystems, and the 7040 DCOS Multiprocessor.
2. The system unit function (SYSUNI) table, which keeps account of the units assigned to system unit functions.
3. The unit control block table, which contains one word for each channel. Each word contains the address of the first unit control block for the channel and the total number of input/output units assigned to the channel.
4. The unit availability table, which contains one word for each channel. Each word serves as an entry point to a chain of available units on the channel.
5. The DC-IBSYS Loader, which is used by the DC-IBSYS Monitor and the subsystems for requesting the 7040 to load records from the system library unit.
6. A dump calling routine, which calls the 7040 whenever a dump is requested by the DC-IBSYS Monitor, by a subsystem, by an object program, or manually by the operator. The 7040 saves a portion of 7090 core storage, places the system in direct mode, and loads the DC-IBSYS Monitor to save the contents of the 7090 panel and to restore the Nucleus. When these restoration procedures are complete, the Core Storage Dump Program is loaded and executed.

The parts of the DC-IBSYS Nucleus of greatest interest to the systems programmer are the communication region, the system unit function table, and the unit availability table. These three are described in the following text, together with job control communication requirements for subsystems operating under the DC-IBSYS Monitor.

COMMUNICATION REGION

The communication region of the DC-IBSYS Nucleus consists of 30 consecutive core storage locations containing constants, control words, and transfer words that may be referred to by the 7040 DCOS Multiprocessor, the DC-IBSYS Monitor, or by any subsystem. Each entry in the communication region is listed in Figure 6 with its octal absolute address, its symbolic address, and its function. A more complete description of the function of each entry is given in Appendix C.

The entries in the communication region may be referred to by their absolute addresses, since their locations do not change. In addition, the entries in this region may be referred to by their symbolic addresses when using (1) the Macro Assembly Program (IBMAP) in relocatable mode, or (2) the FORTRAN II Assembly Program (FAP) if an SST (Save Symbol Table) pseudo-operation is included in the first card group of the FAP language source program.

Octal Address	Symbolic Address	Function
100	SYSTRA	Transfer instruction to current subsystem
101	SYSDAT	Date word
102	SYSCUR	Name of current subsystem
103	SYSRET	Location to which each subsystem returns control
104	SYSKEY	Not used in DCOS
105	SYSSWS	Not used in DCOS
106	SYSPOS	Inital position and index of current subsystem
107	SYSUNI	Location and length of system unit function table
110	SYSUBC	Location and length of table of unit control block locators by channel
111	SYSUAV	Location and length of unit availability table
112	SYSUCW	Location and length of all unit control blocks
113	SYSRPT	Not used in DCOS
114	SYSCEM	Transfer to customer engineering diagnostic routine
115	SYSDMP	Transfer to bootstrap for core storage dump
116	SYSIOX	Location and length of IOEX communication table
117	SYSIDR	Transfer to installation accounting routine, if any
120	SYSCOR	Lower limit of usable core storage in decrement, upper limit in address
121	SYSLDR	Transfer to scatter-load routine
122	SYSACC	Installation accounting routine communication word
123	SYSPID	Installation accounting routine communication word
124	SYSCYD	Not used by DCOS
125	SYSCYD+1	Not used by DCOS
126	SYSLD }	
127	SYSTCH }	Self-load sequence
130	SYSTCH+1}	
131	SYSTWT	System trap, wait, and transfer point
132	SYSGET	Subsystem communication with supervisor control word
133	SYSJOB	Job-skipping control word
134	.CHEXI	Direct Couple environment switch
135	.MODSW	Direct Couple input/ output mode switch

•Figure 6. Communication Region of DC-IBSYS Nucleus

SYSTEM UNIT FUNCTION TABLE

The system unit function table contains an entry for each of the 24 possible system unit functions. The octal location, the symbol, and the system unit function for each entry are shown in Figure 7.

Octal Address	Symbol	System Unit Function
140	SYSLB1	Library 1
141	SYSLB2	Library 2
142	SYSLB3	Library 3
143	SYSLB4	Library 4
144	SYSCRD	System Card Reader
145	SYSPRT	System Printer
146	SYSPCH	System Punch
147	SYSOU1	Output
150	SYSOU2	Alternate Output
151	SYSIN1	Input
152	SYSIN2	Alternate Input
153	SYSPP1	Peripheral Punch
154	SYSPP2	Alternate Peripheral Punch
155	SYSCK1	Checkpoint 1
156	SYSCK2	Checkpoint 2
157	SYSUT1	Utility 1
160	SYSUT2	Utility 2
161	SYSUT3	Utility 3
162	SYSUT4	Utility 4
163	SYSUT5	Utility 5
164	SYSUT6	Utility 6
165	SYSUT7	Utility 7
166	SYSUT8	Utility 8
167	SYSUT9	Utility 9

Figure 7. System Unit Function Table

If an SST (Save Symbol Table) pseudo-operation is included in the first card group of a FAP source program, the symbols listed in Figure 7 represent the symbolic addresses of the table entries. For example, SYSLB1 represents 140 and SYSUT4 represents 162 .

When using the Macro Assembly Program (MAP) language, the symbols in Figure 7 represent index numbers, beginning with 0 for SYSLB1 and ending with 27_8 for SYSUT9. To obtain the correct address of an entry in the system unit function table when programming in the MAP language, the index number expressed by the symbol must be added to the address portion of the SYSUNI entry in the communication region of the Nucleus. Since the address portion of the SYSUNI entry contains the address of the first word of the system unit function table, the addition results in the correct address. For example, adding the address portion of the SYSUNI entry (140_8) to SYSOU1 (defined as 7 by IBLDR) results in the address (147_8) of the entry in the system unit function table for the system output unit.

The address portion of each entry in the system unit function table contains the address of the first word of the unit control block for the unit assigned to the system unit function. If no unit is assigned to the system unit function, the address portion of the entry contains zeros. Normal assignments of units to system unit functions are specified when the entire Direct Couple Operating System is assembled. However, normal unit assignments may be changed temporarily by unit assignment control cards.

UNIT AVAILABILITY TABLE

The unit availability table consists of one entry per input/output channel, beginning with channel A. Each entry contains the address of the first word of the unit control block for the first unassigned (available) unit on the channel. The address portion of the first word of the unit control block for each unassigned unit on a channel contains the address of the first word of the unit control block for the next unassigned unit on the channel. In this way, a unit availability chain for each channel is formed, beginning with an entry in the unit availability table. The address portion of the first word of the unit control block for the last unassigned unit on the channel contains zeros, indicating the end of the chain.

Whenever a subsystem requires the use of an available unit, it interrogates a unit availability chain through a unit availability table entry and removes the unit from the chain.

JOB CONTROL COMMUNICATION WITH SUBSYSTEMS

Each subsystem operating under the DC-IBSYS Monitor must follow certain procedures involving communication with the Monitor to ensure proper control of jobs and job segments.

COMMUNICATION REGION LOCATIONS SYSRET, SYSGET, AND SYSJOB

Communication between the subsystems and the DC-IBSYS Supervisor is carried out chiefly through three locations in the communication region of the System Nucleus. The functions of these three locations, SYSRET, SYSGET, AND SYSJOB, are described in the following text.

Communication Location SYSRET

Whenever a subsystem must return control to the DC-IBSYS Supervisor, it transfers control to SYSRET. As a result, the DC-IBSYS Supervisor is read into core storage and control is relinquished to it.

Communication Location SYSGET

Before returning control to the DC-IBSYS Supervisor, a subsystem must ensure that a word is stored in SYSGET that indicates to the DC-IBSYS Supervisor the reason that control was returned to it. In addition, whenever a post-mortem dump is performed by the DC-IBSYS Core Storage Dump Program, the dump program places IBSXEC in SYSGET (to indicate that a post-mortem dump was performed) before it returns control to the DC-IBSYS Supervisor. When the DC-IBSYS Supervisor obtains control from the dump program or from a subsystem, it examines the word in SYSGET and takes the appropriate action.

A subsystem name in SYSGET indicates to the DC-IBSYS Supervisor that a subsystem has read a $EXECUTE card containing a subsystem name other than its own. Therefore, the DC-IBSYS Supervisor loads into core storage the first record of the subsystem whose name was in the SYSGET location and relinquishes control to this subsystem.

The word IBSYST in SYSGET indicates to the DC-IBSYS Supervisor that a $IBSYS control card was read by a subsystem. Therefore, the DC-IBSYS Supervisor starts processing control cards on the input file beginning with the next card.

The word IBSXEC in SYSGET indicates to the DC-IBSYS Supervisor either of the following conditions:

1. A post-mortem core storage dump was taken by a subsystem, an object program, or the operator; therefore, a job segment was not completed.
2. A subsystem could not complete a job segment.

In either case, the DC-IBSYS Supervisor scans the system input file until the next $IBSYS, $EXECUTE, or $STOP control card is encountered and then processes cards normally beginning with that card.

The word IBSNXT in SYSGET indicates to the DC-IBSYS Supervisor that a subsystem has determined that a job cannot be completed. The Supervisor then simulates a $STOP card. DCMUP responds by initiating job termination procedures.

The word $STOPb (where b is a blank) in SYSGET indicates to the DC-IBSYS Supervisor that a $STOP control card was read by a subsystem. Therefore, the DC-IBSYS Supervisor initiates an end-of-job sequence as though it had read the $STOP card.

Communication Location SYSJOB

The communication location SYSJOB is used by the DC-IBSYS Supervisor and the subsystems to control job processing.

Bit 17 of SYSJOB is used by the subsystems to indicate to each other whether a previous job segment could be completed. Whenever a subsystem determines that it cannot complete a job segment, it sets bit 17 of SYSJOB to a 1 and stores the word IBSXEC in SYSGET. It then returns control to the DC-IBSYS Supervisor, which skips to the next job segment.

When a subsystem gains control at the beginning of a job segment, it tests bit 17 of SYSJOB. If bit 17 is a zero, the system proceeds normally. However, if it is a 1, the subsystem determines whether the present segment should be discontinued because the previous segment has not been completed. If the current segment should be discontinued, the subsystem stores the word IBSXEC in SYSGET and returns control to the DC-IBSYS Supervisor through SYSRET. Thus SYSGET and SYSJOB are used to skip through each segment of a job, allowing each subsystem to determine its course of action, depending upon the success or failure of a previous segment. For example, one subsystem may perform compilations and assemblies but not allow execution if a previous segment failed.

At the completion of a post-mortem dump, the System Core Storage Dump Program sets bit 17 to 1 and stores the word IBSXEC in SYSGET before returning control to the DC-IBSYS Supervisor.

RECOGNITION OF DC-IBSYS SUPERVISOR CONTROL CARDS BY SUBSYSTEMS

Each IBSYS subsystem must recognize and act upon $IBSYS, $EXECUTE, $STOP, and $ID control cards. The following action must be taken by a subsystem when each card is recognized.

$IBSYS CARD

When a $IBSYS card is recognized by a subsystem, the subsystem must return control to the DC-IBSYS Supervisor. Whenever a subsystem gains control from the DC-IBSYS Supervisor, location SYSGET will contain the word IBSYST. Therefore, it is not necessary for the subsystem to load the word IBSYST into SYSGET when a $IBSYS card is recognized.

$EXECUTE CARD

When a subsystem recognizes a $EXECUTE card that contains its own name, the subsystem retains control and continues normal processing. Otherwise, the subsystem stores the name specified on the card into SYSGET and returns control, through SYSRET, to the DC-IBSYS Supervisor, which accomplishes loading of the specified system.

$STOP CARD

When a subsystem recognizes a $STOP card, it loads the word $STOPb into SYSGET and returns control to the DC-IBSYS Supervisor through SYSRET.

$ID CARD

When a subsystem recognizes a $ID card, it must call the installation accounting routine, using the sequence:

```
        TSX     SYSIDR,4
        PZE     L($ID)
        return
```

where L($ID) is the location of the first word of the buffer containing the $ID card in BCD form.

The Direct Couple Input/Output Executor (DC-IOEX) consists of a trap supervisor, a routine for requesting direct-mode input/output, and a number of utility routines that are used in common by the DC-IBSYS Monitor and the subsystems operating under its control.

A subsystem communicates with DC-IOEX and calls DC-IOEX subroutines by way of a communication table located in storage just forward of DC-IOEX. Location SYSIOX in the communication region of the DC-IBSYS Nucleus (Appendix C) contains the address and length of this table. An entry in the DC-IOEX communication table may be referred to by its symbolic address when using either the Macro Assembly Program (MAP) or the FORTRAN II Assembly Program (FAP). However, the symbols used to represent the addresses of the table entries for MAP are different from the symbols used to represent the same addresses for FAP. Both the MAP and FAP symbolic addresses, together with the function of each entry in the DC-IOEX communication table, are listed in Figure 9. When these entries are referred to symbolically using FAP, an SST (Save Symbol Table) pseudo-operation must be included in the first card group of the FAP source program.

The following description of DC-IOEX, together with the description of DC-IOEX in the publication <u>IBM 7090-7040 Direct Couple Operating System: Programmer's Guide</u>, Form C28-6382, and a symbolic listing of the DC-IOEX portion of the DC-IBSYS Monitor, should provide the systems programmer with the information required to use DC-IOEX. In both the description and the listing, the MAP symbols are used when referring to entries in the DC-IOEX Communication Table. The equivalent FAP symbols may be obtained by referring to Figure 9.

UNIT CONTROL BLOCKS

Each tape unit and each card unit is represented in DC-IOEX by a four-word unit control block having the format shown in Figure 8. The unit control blocks are generated by the DC-IBSYS Monitor at initial start in accordance with assembly parameters. In the distributed version of DCOS, DC-IBNUC contains the following 35 unit control blocks:

Channel A	card reader	1
	printer	1
	punch	1
	tapes A1-A0	10
Channel B	tapes B1-B0	10
Channel C	tapes C1-C6	6
Channel D	tapes D1-D6	6
		35

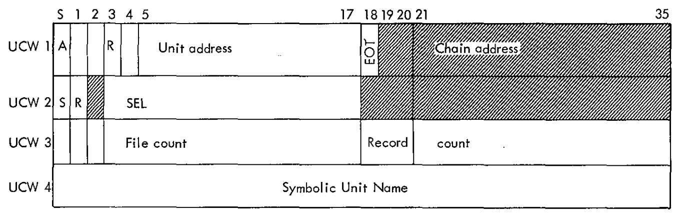


Figure 8. Unit Control Block

Should it prove necessary to expand beyond this, <u>both</u> DC-IBSYS and DCMUP must be reassembled. The copy of DCMUP contained in file 1 of DCSDT must contain the same number of file control blocks as DCSYS (file 2). Reducing the number of unit control blocks below 35 has virtually no advantage.

The contents of each unit control block, shown in Figure 8, are interpreted as follows:

WORD 1

A: Availability Flag

A=0: The unit is assigned to a particular function, not necessarily a system unit function, and the unit is not in the availability chain.

A=1: The unit is not assigned to any particular function.

R: Reserve Status Flag (Intersystem Use Only)

R=0: The unit is not reserved.

R=1: The unit is reserved. Address bits 24-35 of word 1 contain data for intersystem pickup. The unit should not be assigned a system unit function or be in the unit availability chain.

Unit Address

The BCD mode address of the tape is contained here; e.g., 1201 for 729 tape unit A1.

EOT: End-of-Tape Flag

EOT=0: No end-of-tape condition has been assigned to this unit.

MAP Symbolic Address	FAP Symbolic Address	IOEX Entry	Function
.ACTV	(ACTIV	TTR TEST	Activate Routine and Test
.ACTV+1	(ACTVX	TTR ACTIV	Activate Routine Without Test
.NDSEL	(NDATA	TTR TEST	Non-Data Select and Test
.NDSEL+1	(NDSLX	TTR NDATA	Non-Data Select Without Test
.MWR	(PROUT	TTR PROUT	Message Writer
.PUNCH	(PUNCH	TTR PUNCH	Alphameric Punch
.ENBSW	(ENBSW	PZE **	Enable Switch
.PAWS	(PAWSX	TTR PAWS	Error Pause
.PAUSE	(PAUSE	TTR PAUSE	Operator Action Pause
.STOP	(STOPX	TTR STOP	Termination Procedure
.SYMUN	(SYMUN	TTR SYUNCV	Symbolic Unit Conversion
.DECVD	(DECVD	TTR BCVDEC-1	Binary to Decimal--AC Decrement
.DECVA	(DECVA	TTR BCVDEC	Binary to Decimal--AC Address
.CKWAT	(CKWAT	TTR CKWAIT	Checkpoint Wait (not used in DCOS)
.BCD5R	(BCD5R	TTR BCD5-1	Binary to BCD Octal, Bits 3-17 of MQ
.BCD5X	(BCD5X	TTR BCD5	Binary to BCD Octal, Bits 1-14 and S of MQ
.CVPRT	(CVPRT	TTR CVPRT	Convert and Append Unit Designation to Message
.STOPD	(STOPD	TTR HSTOP	Termination Procedure
.CHXAC	(CHXAC	PZE CHXAC	Channel Activity (Indirect Reference)
.URRX	(URRXI	PZE URRX,1	Redundancy Count (Indirect Reference)
.RCTX	(RCTXI	PZE RCTX,1	Redundancy Control (Indirect Reference) *
.RCHX	(RCHXI	PZE RCHX,1	Reset Load Channel (Indirect Reference) *
.TCOX	(TCOXI	PZE TCOX,1	Channel Delay (Indirect Reference)
.TRCX	(TRCXI	PZE TRCX,1	Tape Redundancy Test (Indirect Reference) *
.ETTX	(ETTXI	PZE ETTX,1	End-Tape Test (Indirect Reference) *
.TEFX	(TEFX	PZE TEFX,1	End-File Test (Indirect Reference) *
.TRAPX	(TRAPX	PZE (TRAPS	Current Traps Enabled (Indirect Reference)
.TRAPS	(TRAPS	OCT 377	Current Traps Enabled
.COMM	(COMMM	PZE **,,**	Store Channel Results at Trap
.LTPOS	(LTPOS	PZE **,,**	Tape Position Before Last Trap
.IOXSI	(IOXSI	PZE **,,**	Sense Indicators at Trap
.CHPSW	(CHPSW	PZE **	Checkpoint Switch
.TRPSW	(TRPSW	PZE **	Trap Switch
.FDAMT	(FDAMT	TTR STOP	Termination Procedure
.SDCXI	(SDCXI	PZE SDCX	SDCX Table (Indirect Reference). Not used in DCOS
.STCXI	(STCXI	PZE STCX	STCX Table (Indirect Reference). Not used in DCOS
.COMMD	(COMMD	PZE **	Not used in DCOS
.IBCDZ	(IBCDZ	TSX STOP,4	Not used in DCOS
.CHXSP	(CHXSP	PZE CHXSP	Priority Switch (Indirect Reference)

*The DC channel entry in each of these tables is a NOP.

Figure 9. DC-IOEX Communication Table

EOT=1: An end-of-tape condition has been assigned to this unit.

Chain Address

Chain address is the address of word 1 of the next unit control block in the availability chain. The chain address of the last unit in the chain is zero. These bit positions of word 1 are available to the user when the unit is not in the availability chain or is not a reserve unit.

WORD 2

S: Select Type

S=0: Read

S=1: Write

R: Permanent Redundancy Message Control

R=0: A message is printed if a permanent read redundancy occurs.

R=1: No message is printed in the event of a permanent read redundancy.

SEL: Select Routine

SEL is the location of a user's select routine, which either sets up a data transmission request in direct-mode operation or initiates data transmission in compatibility mode. The select routine also initiates the posting of completed input/output activity.

WORD 3

File count reflects the number of file marks written on or read from this unit.

Record count reflects the number of records that have been written on or read from the current file.

WORD 4

Word 4 is used in symbolic unit assignment. The 7040 setup routine places the symbolic unit name, A(1), etc., that is on a $SETUP card into its setup table (UNISYM). At job initialization, the DC-IBSYS Supervisor requests this table from the 7040 and places the symbolic unit name into the fourth word of its corresponding 7090 unit control block. IBLDR, when selecting an available unit for a $FILE card referring to unit A(1), searches the fourth word of 7090 unit control blocks. Upon finding a matching symbol, IBLDR sets the 7090 file control block to point to the corresponding unit control block. Word 4 is also used by IOCS labeling routines for storing the tape reel serial number for multifile reels.

Notes

If the EOT bit = 1, DC-IOEX sets the EOT trigger on for every select minus routine entered until the tape is either rewound or rewound and unloaded by the .NDSEL routine. At this time the EOT bit is reset to zero.

When backspacing from an end of file, the record count is complemented. A backspace that repositions in front of a file mark results in a record count (bits 18-35 of word 3) of 777777_8. When writing begins from such a position, the two low-order tag bits (bits 19 and 20) are cleared to prevent a spurious increase in the file count when the record count is increased.

UNIT PRIORITY ON A CHANNEL

The user should keep in mind that only one input/output operation is occurring at a given time.

CHANNEL-PRIORITY LOCATION

When activity begins on a unit of a given channel, DC-IOEX places the address of the unit control block in a channel-priority location. This normally allows the unit to retain priority until all its waiting operations have been completed. DC-IOEX clears the channel-priority location when the user clears word 2 of the unit control block.

Upon return from a select minus routine, DC-IOEX selects the next unit to be activated by examining the channel-priority location. The same unit will be reselected if the user did not relinquish control by clearing word 2 of the unit control block.

USE OF CHANNEL-PRIORITY LOCATION

If the channel-priority location is not zero, its address portion is interpreted as the location of the unit control block for the unit to be activated next. Word 2 of the unit control block is then tested.

If word 2 is not zero, the select plus routine for the unit is entered. If word 2 of the unit control block is zero, the priority location is cleared and the channel is scanned for a waiting unit, that is, a unit that has a select routine specified in word 2 of its unit control block. If a waiting unit is found, the location of its unit control block is placed in the channel-priority location and the select plus routine is entered for the unit. If no waiting unit is found, the channel is allowed to become dormant.

ACTIVATING A CHANNEL AND/OR ASSIGNING PRIORITY

A channel that is dormant can be activated, or a unit can be given top priority on a channel, by means of the routine .ACTV. The calling sequence to this routine is:

```
        TSX     .ACTV,4
        pfx     a,t
        return
```

where pfx is a prefix code and the parameter a,t gives the address of a location (possibly a system unit function table entry) that contains:

S	1 17	18 20	21 35
▨	▨	▨	L(UCB)

L(UCB) is the address of the unit control block of the unit for which input/output activity is desired. Note the double indirectness to the actual unit address.

If pfx is PZE in the calling sequence, and IOEX is in direct-mode operation, controls are set up so that the user's select routine is entered whenever the channel is free to accept activity on the specified unit. No action by .ACTV is necessary if the channel is active upon entry, and control is immediately returned to the calling program. If the channel is dormant, the user's select plus routine is entered.

If IOEX is operating in the compatibility mode, or if pfx is MZE in the calling sequence, the indicated unit is given top priority on the channel (that is, the channel-priority location is set for the unit, and the select routine is entered as soon as possible). If the entry to .ACTV is made during nontrap time, control is not returned to the calling program until the select plus routine for the unit has been entered.

If the entry to .ACTV occurs during trap time, e.g., from the select minus routine, the unit is given top priority by setting the channel-priority location. However, control is returned to the calling program immediately. Hence a subsequent entry to .ACTV, requesting priority (pfx = MZE), will cancel the priority effects of the previous entry if the subsequent entry occurs before select plus is entered for the previous entry.

Any entry to .ACTV during trap time must be made using an MZE for pfx in the calling sequence. Also, the entry should be for a unit connected to the channel that caused the trap.

The .ACTV routine will always enable traps on all channels upon return unless entry was from a select minus routine.

.ACTV may not be entered from a select plus routine.

.ACTV makes a validity test on the specified input/output unit. To be valid, the location of the unit control block must fall within the area of core storage provided for all unit control blocks. In addition, the availability flag of word 1 of the unit control block must be zero. If the unit is judged to be invalid by these criteria, an automatic post-mortem dump is taken after the following message is printed

```
        ILLEGAL UNIT REQUEST AT XXXXX
```

After the dump is taken, the DC-IBSYS Supervisor skips to the next job segment.

The validity test described above may be bypassed by substituting:

```
        TSX     .ACTV+1,4
for
        TSX     .ACTV,4
```

in the calling sequence.

NON-DATA SELECTS

Non-data selects are executed by the routine .NDSEL. The calling sequence to this routine is:

```
        TSX     .NDSEL,4
        PZE     a,t,nds
        return
        return if nds = 7
```

The parameter a,t has the same meaning as in the calling sequence for .ACTV. The parameter nds is interpreted as follows:

nds=0	NOP	No Operation
nds=1	SDNL	No Operation
nds=2	SDNH	No Operation
nds=3	REW	Rewind
nds=4	RUN	Rewind and Unload
nds=5	BSR	Backspace Record
nds=6	BSF	Backspace File
nds=7	WEF	Write End of File

.NDSEL gives the specified unit top priority by using .ACTV+1 with an MZE in the calling sequence. Hence, the non-data select is executed as soon as present activity on the unit has been completed. Return is made only after the non-data select has been executed. .NDSEL may not be entered from a select plus routine.

The .NDSEL routine will always enable traps on all channels upon return unless entry was from a select minus routine. .NDSEL should be entered during trap time only for a unit that is on the channel that has trapped.

.NDSEL performs a validity test on the specified unit control block in the same manner as .ACTV. The test may be bypassed by using

```
|       TSX     .NDSEL+1,4
```

rather than

```
|       TSX     .NDSEL,4
```

Note that a NOP (nds=0) entry to .NDSEL has the effect of performing a validity test and nothing else.

STRUCTURING OF INPUT/OUTPUT REQUESTS IN DIRECT MODE

The Direct Couple Input/Output Executor is designed to operate in both direct mode and compatibility mode. This section discusses RDUN, the DC-IOEX routine whose primary purpose is to set up direct mode input/output requests.

RDUN (Figure 10) is entered by the DC-IOEX routines .ACTV and XTRAP immediately upon return from the user's select plus routine. RDUN determines the current operational mode of DCOS by testing indirectly the IOEX compatibility-mode switch, COMPSW. If the system is in compatibility mode, no input/output activity is requested by RDUN and control is immediately returned to the calling routine. (The compatibility-mode select routine will already have informed the 7040 of its input/output requirements.)

If the system is in direct mode, RDUN must request input/output activity. RDUN tests location THIS. If the decrement of THIS is nonzero, RDUN assumes that the calling program has already set up locations THIS and THIS+1 and traps the 7040 with an HEY instruction.

If the contents of THIS are zero, RDUN sets up a direct-mode input/output request in locations THIS and THIS+1 as follows:

1. RDUN places a TCH instruction which contains the address of the first input/output command in location THIS+1.
2. Using the channel-activity location RDUN computes the relative location of the unit control block of the unit for which input/output activity is desired. This number is stored in the address portion of THIS.
3. Using the sign of word 2 of the unit control block, RDUN sets the decrement of THIS to 1, signifying a read request, or to 2, signifying a write request.
4. RDUN tests the mode switch (MODSW) in the DC-IBSYS Nucleus communication region. If the contents of MODSW are zero, RDUN sets the prefix of THIS to PZE, indicating data transmission in BCD mode. If MODSW is nonzero, RDUN sets the prefix of THIS to MZE, indicating binary mode.

If the address portion of THIS is nonzero but the decrement is zero, RDUN assumes the user wants to write a block with control words. RDUN sets up locations THIS and THIS+1 as described in the preceding text except that it places a code of 14 in the decrement of THIS.

After setting up the input/output request, RDUN traps the 7040 with an HEY instruction. The request is serviced by the 7040, as described in the following section.

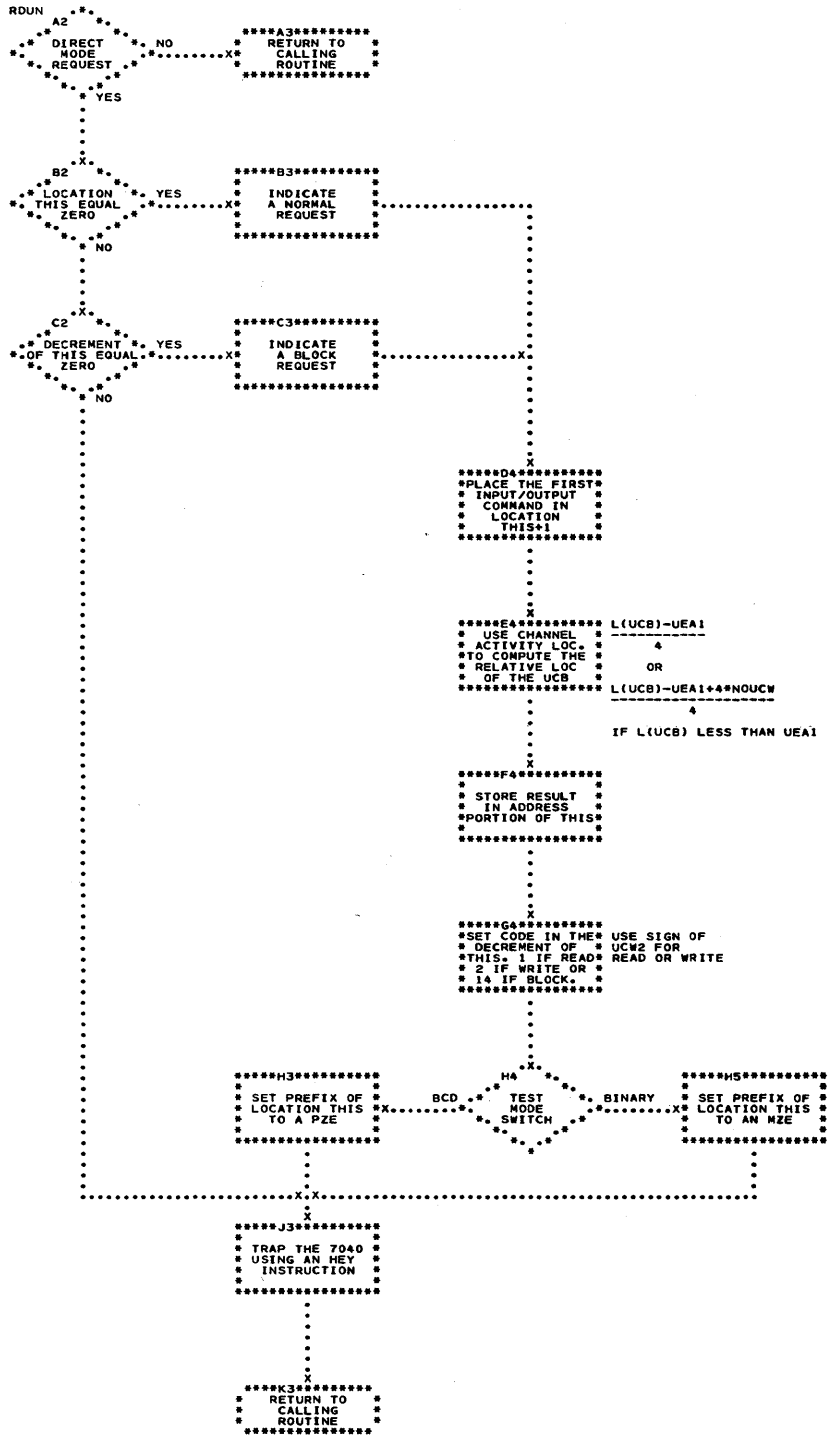

RDUN
A2 DIRECT MODE REQUEST
NO
A3 RETURN TO CALLING ROUTINE
YES
B2 LOCATION THIS EQUAL ZERO
YES
B3 INDICATE A NORMAL REQUEST
NO
C2 DECREMENT OF THIS EQUAL ZERO
YES
C3 INDICATE A BLOCK REQUEST
NO
D4 PLACE THE FIRST INPUT/OUTPUT COMMAND IN LOCATION THIS+1
E4 USE CHANNEL ACTIVITY LOC. TO COMPUTE THE RELATIVE LOC OF THE UCB
L(UCB)-UEA1 / 4
OR
L(UCB)-UEA1+4*NOUCW / 4
IF L(UCB) LESS THAN UEA1
F4 STORE RESULT IN ADDRESS PORTION OF THIS
G4 SET CODE IN THE DECREMENT OF THIS. 1 IF READ 2 IF WRITE OR 14 IF BLOCK.
USE SIGN OF UCW2 FOR READ OR WRITE
H3 SET PREFIX OF LOCATION THIS TO A PZE
BCD
H4 TEST MODE SWITCH
BINARY
H5 SET PREFIX OF LOCATION THIS TO AN MZE
J3 TRAP THE 7040 USING AN HEY INSTRUCTION
K3 RETURN TO CALLING ROUTINE

Figure 10. RDUN Routine for Requesting Direct Mode Input/Output

This section discusses the logic of the routines that service 7090 input/output requests. These routines are in the 7040 and are entered as a result of a trap from the 7090.

PROCESSING INPUT/OUTPUT REQUESTS (DIRECT MODE)

When DCOS is operating in the direct mode, words THIS and THIS+1 in the 7090 are used to specify the input/output function to be performed by the 7040. When the 7090 traps the 7040 by executing an HEY instruction, the 7040 traps to the main interrupt routine (WHAT).

WHAT analyzes the condition bits in the decrement of the input/output trap store word (ENTER) to determine what caused the trap. If bit 16 of ENTER is a 1, control is transferred to symbolic location COMP to process the input/output request in the compatibility mode. If bit 16 of ENTER is a 0, the contents of words THIS and THIS+1 in the 7090 are transmitted to words FILE and FILE+1 in the 7040. The contents of the decrement of FILE are then examined to determine which input/output routine will be used to service the request.

The format of words THIS and THIS+1 are shown in Figure 11. In Figure 11, "unit" specifies the 7090 unit control block number for which the request is being made. This number is used by the 7040 to locate the appropriate file control block for the simulated unit.

READ: The prefix and bit 18 of FILE+1 are examined to determine the type of data transfer requested (count control, record control, non-data transfer, or transfer in channel). If the command is a transfer in

Type of Operation	THIS		THIS+1
Read Tape Binary	MZE	unit,,1	First input/output command
Read Tape Decimal	PZE	unit,,1	First input/output command
Write Tape Binary	MZE	unit,,2	First input/output command
Write Tape Decimal	PZE	unit,,2	First input/output command
Rewind	PZE	unit,,3	
Rewind and Unload	PZE	unit,,4	
Backspace Record	PZE	unit,,5	
Backspace File	PZE	unit,,6	
Write End of File	PZE	unit,,7	
Load a System	PZE	0,,8	System name in BCD
Load a System Record (HIP)	PZE	0,,9	System name in BCD
Dump 7090 Core Storage	PZE	0,,10	
7090 Pause	PZE	0,,11	
Time Request	PZE	0,,12	7040 clock reading in hours, minutes, and seconds (hhmmss)
Reserved	PZE	0,,13	
Write and pass generated control words from the 7090	PZE	unit,,14	Location of input/output command (IORT)
Call to 7040 Message Writer	PZE	unit,,15	Location of input/output command (the eleventh character of the message is the class code)
Call Setup Table (UNISYM)	PZE	0,,16	Location in 7090 to store UNISYM table
Normal termination of 7090 job	PZE	0,,17	
Normal termination of IBSYS initialization	PZE	0,,18	

● Figure 11. Format of THIS and THIS+1

channel (TCH), the address portion is used to build a transmit instruction that will move an input/output command from the 7090 to the 7040. Control is then returned to symbolic location NEXT, where the new command is analyzed. The address and decrement portions of the command are saved for transmitting the specified number of words, or the next record, to the 7090. When operating in the direct mode, the Input/Output Under Count Control and Disconnect (IOCD) command causes an error message to be printed. This is an error because an IOCD will not cause a data channel trap.

If the word count specified in the decrement of FILE+1 exceeds the number of words in the primary buffer, the data in the primary buffer is moved to the 7090, the buffer is returned to the pool, and the secondary buffer is assigned primary status. If the new primary buffer is full, processing continues. If the buffer is not full, the delay 7090 interrupt gate (90P5) is opened and the nonprocessed exit is taken. When the input/output requested has been completed, routine RDONE is entered. This routine traps the 7090 and simulates a Store Channel command.

When an end of file is sensed, the secondary buffer is assigned primary status and word 1 of the file control block is set to minus to indicate that the file is closed.

<u>WRITE</u>: The procedure for servicing a write request is basically the same as for servicing a read request except that, as buffers are written, they are automatically returned to the availability pool at trap time.

When the 7040 is trapped by a write request, it takes the first input/output command from word FILE+1 and extracts from it all the needed data. If the file is not open or the request cannot be satisfied by the primary and secondary buffers, commutator gate 90P5 is opened. If the buffers cannot satisfy the request (the primary buffer is filled) it is entered into the priority write queue, and the secondary buffer is assigned primary status. In either case, the nonprocessed exit is taken.

<u>REWIND (INPUT FILES)</u>: If the file is open and either the primary buffer or the secondary buffer is not primed, commutator gate 90P5 is opened and the nonprocessed exit is taken. If the file is open and both buffers are primed, the buffers are released to the availability pool and the file is closed (word 1 of the file control block is set to minus). If the file is on tape, a rewind request is queued. If the file is on disk, its job description block is searched for the first task associated with the file. The location of the current task (address portion of word 4 of the file control block) is then updated to point to this task.

<u>REWIND (OUTPUT FILES)</u>: If the file is on tape and is open, the primary buffer is entered into the write queue and the secondary buffer is returned to the availability pool. Write tape mark and rewind requests are then queued for the file. Two tape marks are written prior to rewinding the tape.

If the file is on disk, the forward track address in the primary buffer (address portion of the second physical-record control word) is reset to zero and the buffer is entered into the write queue. The secondary buffer is returned to the availability pool, the file is closed (word 1 of the file control block is set to minus), and the current task pointer (address portion of word 4 of the file control block) is set to the location of the task entry for load point.

<u>REWIND AND UNLOAD</u>: A rewind and unload request is serviced by the same routine that services a rewind request except that the instruction at symbolic location REWT is changed to a transfer to a routine at REWT2. If the file is on tape, a rewind and unload request is queued at REWT2.

<u>BACKSPACE RECORD</u>: The current logical-record control word is analyzed to determine the status of the current record. If the record is contained completely in the primary buffer but is not the first complete record in the buffer, the logical-record control word pointer is changed to point to the previous logical-record control word. If analysis of the current logical-record control word indicates that the above conditions have not been met, the manner in which the request is processed is determined by whether the file is an input file or an output file.

If the file is an input file, a test is made to determine if the secondary buffer has been primed. If it has not, commutator gate 90P5 is opened and the non-processed exit is taken. When the secondary buffer is primed, both the primary and the secondary buffers are returned to the availability pool.

If the file is on disk, the backspace record request is serviced by, in effect, reading the disk backward down the file. This is accomplished by using the track chaining information in the logical-record control word. The decrement portion of the logical-record control word contains the number of words in the previous record. The current logical-record control word pointer is then updated to complete the request.

If the file is on tape, routine E$BSR is used to issue a Backspace Record (BSR) instruction on the tape unit.

If the file is an output file and is on disk, the manner in which the request is to be processed is determined by the status of the file (open or closed) and of the buffers (attached or released). If the file is open and the record does not start in the primary buffer, or if the file is closed and buffers are attached, processing is the same. The secondary buffer is released to the availability pool, and commutator gate 90P5 is opened to wait for the write queue on the primary buffer to be completed. When writing is complete, the disk is, in effect, read backward down the file, as described for input files on disk.

If the file is closed and no buffers are attached, the file is opened and commutator gate 90P5 is opened to attach buffers. If a secondary buffer was attached, it is released to the availability pool. The file is then closed again, and the procedure is the same as for an open file.

If the file is an output file on tape, the procedure is the same as for disk except that two tape marks are written and Backspace Record (BSR) instructions are queued to position the file.

BACKSPACE FILE: If the file is closed and is on tape, a backspace file request is queued. If the file is on disk, the current task pointer in word 4 of the file control block is set to locate the previous task in the job description block.

If the file is open, the secondary buffer is released to the availability pool. The primary buffer is truncated, the end-of-file flag (sign bit of the first physical-record control word) is set, and the buffer is entered into the write queue. The file is then marked closed, and the procedure for processing a closed file is used.

WRITE END OF FILE: If the file is open, the forward track address is set to zero, and the buffer is entered into the write queue. The file is then marked closed, and the secondary buffer is assigned primary status. If the unit is tape, a Write End of File (WEF) instruction is queued to complete the request.

If the file is closed and has no buffers attached, it is marked open, commutator gate 90P5 is opened, and the nonprocessed exit is taken. When the buffers have been attached, the procedure used for an open file is followed.

If the file is closed, is on the disk, and has one buffer attached, the address portion of word 4 of the buffer (next track address) is set to zero and routine TRACK is entered for an address. The buffer is then entered into the write queue.

SCATTER-LOAD (DC-IBSYS AND ITS SUBSYSTEMS): When the 7040 receives a scatter-load request, it tests the contents of word FILE+1. If FILE+1 contains IBNEXT, the next record on the library (SYSLB1) is scatter-loaded into the 7090. Anything else in FILE+1 causes a search of the 7040 system name table to locate the track address of the desired record. The 7040 system name table is created by the DCOS Posteditor (PEDIT) as part of the editing procedure. It contains system record names and the track addresses of the systems. The 7040 system name table also contains two-word entries that indicate logical end of file, and names and starting track addresses of data files. These data files can be read from the library (SYSLB1) by the 7090. DCMUP maintains a pointer to the system name table that indicates the current position of SYSLB1.

When the system name table entry is located, its sign is tested to determine whether the file is a data file or a system. A minus sign indicates a data file; a plus sign indicates system record. If the sign is minus, a normal read is executed. If the sign is plus, a scatter-load is executed. When the read or load has been completed, the 7090 is trapped.

A call to load DC-IBSYS causes the 7040 to leave compatibility mode and transmit the date, which was given in a $DATE card, to IBNUC.

LOAD AND ENTER COMPATIBILITY MODE: A request for a load and enter compatibility mode is serviced by the same routine that services a load request. The only difference is the execution of an Enter Halt on Input/Output Primary Mode (EHM) instruction at symbolic location HIP.

TIME REQUEST: A time request causes the contents of 7040 word 5 to be converted to the hhmmss format and to be transmitted to 7090 word THIS+1. The contents of 7040 word 5 are the time of day in seconds. When the request has been serviced, the 7090 is trapped.

PROCESSING INPUT/OUTPUT REQUESTS (COMPATIBILITY MODE)

Input/output requests are serviced in the compatibility mode by the compatibility string processor. At symbolic location COMP in the 7040, the input/output instruction that caused the trap is transmitted from the 7090 to the 7040 where it is decoded. Control is then transferred through the operation jump table (OPJTBL) to the routine that processes that type of instruction.

NONDATA SELECT REQUEST: All nondata select requests are processed in the same manner. Each type of instruction causes control to be transferred to a specific routine that initializes a transfer instruction, which, in turn, transfers control to a common nondata select processor. The transfer instruction (at OPEX) that is initialized is set to transfer control to the routine that will simulate the instruction that is being processed. The common nondata select processor, starting at symbolic location OPENT, determines the location of the file control block for the specified unit. This location is then saved in index register 1 for the instruction-simulation routine, and the transfer at OPEX is taken. For non-IBJOB DC-IBSYS subsystems all nondata select requests are handled as direct-mode input/output requests.

READ/WRITE SELECT REQUESTS: Processing of read and write select requests consists of setting switches to indicate to the data-transmission request routines that a select request has been processed. A gate is also set in the write-transmission routine according to whether a read-select or a write-select request is being processed. For read-select requests, the gate is set to minus, making it a transfer instruction to the read-transmission routine. For write-select requests, it is set to plus, effectively making it a NOP and leaving the write-transmission routine in control. The location of the file control block for the specified unit is then saved in word UINDEX for use by the data-transmission request routine.

DATA-TRANSMISSION REQUESTS: For data-transmission requests, a test is made to determine whether the select switch (IOS) has been set by the routine that processes read/write select requests. If IOS has not been set, the input/output check indicator is turned on in the 7090 and the 7090 is started. If the select switch is set to minus, routine CHCOMM is called to locate and transmit the first input/output command. The file control block pointer is then placed in index register 1, the 7090 is started, and control is transferred to READ+1, where the command is interpreted and simulated.

CHANNEL INDICATOR TESTS: The response to instructions that test the status of the channel indicators (i.e., TCOA, TEFA, etc.) is provided by the Start Remote Computer (SRC) instruction. The address portion of the SRC instruction determines where the 7090 should go for the next instruction. The specified indicator is tested by the 7040, and one of the following instructions is executed.

SRC	0	Start the 7090 (e.g., TCOA).
SRC	1	Start the 7090 and skip one instruction (e.g., BTT, ETT successful).
SRC	2	Start the 7090 and transfer to the effective address (e.g., TEF, TRC successful).
SRC	4	Start the 7090 and enable it for multiprocessor traps. Used with all ENB and RCT requests.

STORE CHANNEL REQUEST: A Store Channel request is serviced by transmitting what would normally be the contents of the channel registers to the specified 7090 location.

ENABLE AND RESTORE CHANNEL TRAPS: The enable word is transmitted to the 7040 and tested. If it is zero, the 7090 is started. If the enable word is nonzero, the enable switch is set to plus and the channel information blocks are scanned to determine if any traps are waiting (i.e., has the simulation of a 7607 operation set a trigger that corresponds to some normal 7090 data channel trap condition). If none are waiting, the 7090 is started with multiprocess trapping enabled. If a trap is waiting and the 7090 channel is enabled, the channel condition bits are transmitted to the appropriate trap word (12_8, 14_8, etc.) in the 7090, which is trapped. The data channel trap is simulated by storing the 7090 location counter in the appropriate data channel trap location (i.e., 12_8, 14_8, ..., 30_8) and putting the location of the appropriate data channel trap location+1 (i.e., 13_8, 15_8, ..., 31_8) into the 7090 location counter. The 7090 is then started.

SENSE INSTRUCTIONS: If the instruction being processed is a Sense Printer (SPR), a Sense Punch (SPU), or a Reset Data Channel (RDC), the 7090 is restarted and the normal exit is taken.

LISTINGS

One list tape for DCOS is distributed which contains a symbolic listing of DCOS and all its support programs except the 7040 standalone utilities. It also contains symbolic assembly listings of the modifications made to the IBM 7090 IBJOB Processor. The list tape is in standard blocked (five lines per block) IBM 720A format; lines are separated by record marks.

Listings may also be obtained by assembling the PREST symbolic decks on the DCOS Distribution Tape.

DCOS DISTRIBUTION TAPE

The DCOS Distribution Tape (DCSDT) contains the complete Direct Couple Operating System except listings. DCOS in symbolic (PREST) form, the maintainable binary DCOS (PRESYS Library) and the entire DCOS for disk residence are contained on this tape. In addition, the DCOS Distribution Tape contains card decks and support programs that are (or may be) required for DCOS operation and maintenance.

The format and complete contents of the DCOS Distribution Tape are described in the following text.

FILE 1

File 1 contains a control program (CTLTP), the Format Track Generator, and DCMUP with distribution tape controls. The Format Track Generator is an IBM 7040/7044 independent utility program (7040-UT-142). File 1 is in card-image form, blocked into the standard 460-word DCOS format.

CTLTP causes all of file 1 to be loaded into core storage. If sense switch 2 is on, CTLTP will punch the DCOS card decks and support programs from file 3. If sense switch 4 is on, disk format control cards will be read and the Format Track Generator will be used. Then file 2 (DCSYS) is placed on the disk, ready for disk load procedures.

FILE 2

The logical records in file 2 (DCSYS) are 461 words in length. The first word of each record is the address of the track on which the record will be placed during system initialization.

```
        ┌───────────┬───────────────────┐
Word 1  │           │ Track Address     │
        ├───────────┴───────────────────┤
        │                               │
        │     Input/Output Command      │
        │               .               │
        │               .               │
        │               .               │
        │         Instructions          │
        │               .               │
        │               .               │
        │               .               │
        │     Input/Output Command      │
        │               .               │
        │               .               │
        │               .               │
        │         Instructions          │
        │               .               │
        │               .               │
        │               .               │
   461  │                               │
        └───────────────────────────────┘
```

•Figure 12. DCSYS Format for Scatter Load Program

The track addresses are 18-bit binary integers, which, when converted to BCD integers, are of the form:

mmtttt

where

mm = module
tttt = track (0000-9999)

The section "DCOS Posteditor Output" contains additional information about the DCSYS 461-word record format and about the contents of special DCSYS records.

FILE 3

All card decks distributed with DCOS are located on file 3. Each record contains card images in blocked standard DCOS format (460 words). Each deck is prefaced by a BCD card-deck description card, as shown in

Figure 13. If columns 2-6 of the deck-description card contain 40ABS, the deck that follows may be placed in the 1402 and loaded by pressing the 7040 LOAD button.

Card-deck serialization characters (columns 73-80) are interpreted as follows.

Column 73 contains a deck identification character.

Columns 74-78 are used for serialization of cards within a deck.

Columns 79-80 may be used for subserialization.

FILES 5-15

Files 5-15 are the DCOS PRESYS library files. They may be maintained by using the procedures described in the remaining sections of this publication. File 5 contains DCMUP, DC-CLEAR, and DC-IBSYS (including DC-IOEX, DCIDR, and SYSDMP). The IBM 7090/7094 IBJOB Processor, Version 5, modified for direct-mode DCOS operation, is located in files 6-12. File 13 is the System Editor, IBEDT. File 14 contains the background utilities. File 15 is the non-IBSYS systems and the *EOT file, which indicates the end of the PRESYS library.

FILE 4

File 4 contains DCOS PREST symbolic decks.

Deck Number	Deck Description		Card Count		MOD Level	Deck Serialization
	Col. 1	8	36 39	40 70	71 72	73 80
0	*40	DISK/DRUM FORMAT CARDS	6			00000000
1	*40ABS	DCOS DISK LOAD CARD	1			10000000
2	*40ABS	CORE DUMP PROGRAM				20000000
3	*40ABS	DCOS SAVE/RESTORE				30000000
4	*40ABS	DCOS DISK LOAD AND SNAPS				40000000
5	*40REL	DCOS TAPE BLOCK ROUTINE				50000000
6	*40REL	DCOS TAPE DEBLOCK ROUTINE				60000000
7	*40REL	DCOS SYSTEM TAPE BLOCKER				70000000
8	*	SAMPLE DCOS JOB				80000000
9	*	DCOS MASTER ALTER/EDIT				A0000000
		. (contains				.
		. a one module				
		. Posteditor				.
		. deck)				.
		.				.
10	*	DCOS TAPE BLOCK SYMBOLICS				X0000000
11	*	DCOS TAPE DEBLOCK SYMBOLICS				Y0000000
12	*	DCOS SYSTEM TAPE BLOCK SYMB				Z0000000
13	*LIST DECK OF MODIFICATION 3					

● Figure 13. Card Deck Description Cards

The system editor (IBEDT) is used to maintain the DCOS PRESYS library. With the proper control cards, the user can add, delete, modify, replace, or rearrange the records and the files of the library. The records and the files that may be on the DCOS PRESYS library are shown in Figure 14.

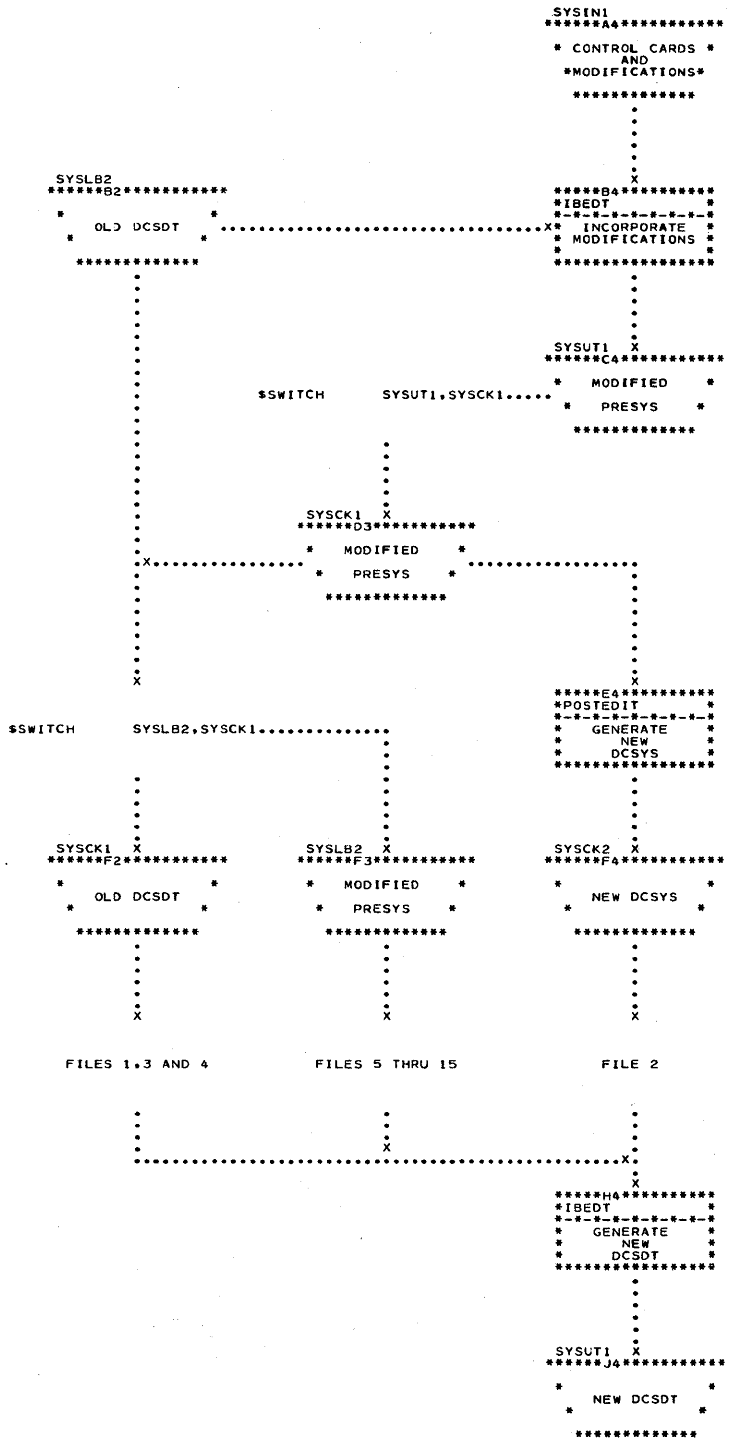
Type of Record or File	Executed In	Record Format
DCMUP	7040	scatter-load
Clear Storage Program	7090	scatter-load
DC-IBSYS, including DC-IOEX, DCIDR, and DC-SYSDMP	7090	scatter-load
IBSYS subsystems	7090	scatter-load except for enclosed data files such as IBLIB
System Editor (IBEDT)	7090	scatter-load
Utility Programs	7040	scatter-load
non-IBSYS systems	7090	

Figure 14. DCOS PRESYS Library

Figure 15 illustrates the flow of control and the system unit functions involved in an edit run. In the first phase of an edit, IBEDT incorporates the modifications defined by control cards read from either SYSIN1 or SYSUT2. The modified system (PRESYS) is written on SYSUT1. A $SWITCH card then causes the unit assigned as SYSUT1 to be reassigned as SYSCK1. This is done because the DCOS Posteditor (PEDIT), expects to find its input on SYSCK1. During the post edit phase, PEDIT generates a new DCSYS on the unit assigned as SYSCK2. The units assigned as SYSLB2, the old distributution tape, and SYSCK1, the modified DCSYS and PRESYS, are then switched by a $SWITCH card. During the final phase of the edit, SYSLB2, SYSCK1, and SYSCK2 are used as input to IBEDT to generate a new DCOS Distribution Tape. The new distribution tape is written on SYSUT1. The first file written on SYSUT1 is taken from SYSCK1, the second file from SYSCK2, the third and fourth files from SYSCK1, and the remaining files from SYSLB2.

Figures 16 and 17 are two sample editing decks. Figure 16 is the deck setup that would be used to insert modifications to DCMUP, DC-IBSYS, and IBMAP. Figure 17 is a two page listing of the DCOS master alter/edit deck. The master alter/edit deck is used to insert modifications and reassemble the DC system. Note that a system editor (IBEDT) deck precedes the assemblies. The edit copies the PREST decks onto SYSCK2 and terminates due to the insertion of a record named *EOT.

Note: A common error in editing occurs when one attempts to edit SYSLB1 rather than SYSLB2. This results in an incorrect edit terminated by a system error message or by an invalid disk channel reference which will terminate processing (requires restart).



• Figure 15. Three-Phase Editing Operation

```
$DATE           062364
$JOB            9,30,1000       PRODUCE NEW DISTRIBUTION TAPE WITH MODS.
$SETUP LB2      DCSDT,NORING
$SETUP UT1      Y00014
$ASSIGN         SYSLB2
$IBEDT
      *EDIT     SYSLB2,MAP,MODS
      *DUP      SYSLB2,SYSUT4,4
      *MODIFY   DCMUP
       .
       . DCMUP Modifications
       .
      *MODIFY   IBSYS
       .
       . DC-IBSYS Modifications
       .
      *MODIFY   IBMAPJ
       .
       . IBMAP Modifications
       .
$EOF
$IBSYS
$*    AT THIS POINT, SYSUT1 CONTAINS THE NEW PRESYS LIBRARY.
$SWITCH         SYSUT1,SYSCK1
$*    THE NEW PRESYS LIBRARY IS NOW ON SYSCK1 AS INPUT TO DCOS PEDIT.
$EXECUTE        IBJOB
$IBJOB          NOSOURCE
$IBLDR PEDIT
       .
       . DCOS Posteditor IBLDR Binary Deck.
       .
$DKEND PEDIT
$DATA
    (No data follows. PRESYS is now at load point.)
$EOF
$IBSYS
$SWITCH         SYSLB2,SYSCK1
$*    AT THIS POINT THE OLD DCSDT IS ON SYSCK1, THE MODIFIED DCSYS AND
$*    PRESYS LIBRARY ARE ON SYSLB2, AND THE NEW DCSYS IS ON SYSCK2.
$ASSIGN         SYSUT1                  CONNECT TAPE Y00014 TO SYSUT1.
$IBEDT
      *EDIT     SYSLB2                  ADVISE MAP OPTION NOT BE USED.
      *DUP      SYSCK1,SYSUT1,1         SAVE 1ST FILE FROM OLD DCSDT.
      *DUP      SYSCK2,SYSUT1,1         INSERT THE NEW DCSYS FILE.
      *DUP      SYSCK1,SYSUT4,1         DELETE THE OLD DCSYS FILE.
      *DUP      SYSCK1,SYSUT1,2         SAVE FILES 3 AND 4 FROM OLD DCSDT.
    (IBEDT will put the new PRESYS library on the new DCSDT.)
$EOF
```

• Figure 16. Sample Edit with Modifications to DCMUP, DC-IBSYS, and IBMAP

```
$DATE          043065                                                  A0000010
$JOB           0,50,60000     PRODUCE MODIFIED DCS TAPE                A0000020
$SETUP LB2     DCSDT,NORING                                            A0000030
$SETUP UT1     Y00014                                                  A0000040
$ASSIGN        SYSLB2                                                  A0000050
$IBEDT                                                                 A0000060
     *EDIT     SYSLB2                                                  A0000070
     *DUP      SYSLB2,SYSUT3,3     DUP OFF FIRST 3 FILES               A0000080
     *DUP      SYSLB2,SYSCK2,1     COPY PREST MASTER FILE ONTO CK2     A0000090
     *INSERT                       TERMINATE EDIT                      A0000100
03720  OCT     542546636060,000000000000,000000000000                  A0000110
$EOF                                                                   A0000120
$IBSYS                                                                 A0000130
$EXECUTE       IBJOB                                                   A0000140
$IBJOB PR-161  NOGO                                                    A0000150
$IEDIT DCMUP   SYSCK2,SCHF1,ALTER                                      B0000000
$IBMAP DCMUP   16000,M94,ABSMOD,()OK   7040 DCS                        B0000001
       *ENDAL                                                          B9999999
$IEDIT IBSYS   SYSCK2,SCHF1,ALTER                                      C0000000
$IBMAP IBSYS   8000,M94,ABSMOD,()OK    7090 CLEAR, IBSYS, SYSDMP       C0000001
       *ENDAL                                                          C9999999
$IEDIT DCIDR   SYSCK2,SCHF1,ALTER                                      D0000000
$IBMAP DCIDR   10,M94,ABSMOD,()OK      DUMMY ACCOUNTING ROUTINE        D0000001
     *ENDAL                                                            D9999999
$IEDIT                                                                 DA000000
$IBMAP IBJOB   10,ABSMOD,()OK          PATCHES TO ACTION               DA000010
       END     -1                                                      DA000060
$IEDIT JOBOUM  SYSCK2,SCHF1,ALTER                                      E0000000
$IBMAP JOBOUM  100,M94,ABSMOD,()OK     PATCHES TO JOBOU                E0000001
     *ENDAL                                                            E9999999
$IEDIT                                                                 EA000000
$IBMAP IBLDRO  10,ABSMOD,()OK          PATCHES TO IBLDRO               EA000010
       END     -1                                                      EA000060
$IEDIT LDRMOD  SYSCK2,SCHF1,ALTER                                      F0000000
$IBMAP LDRMOD  500,M94,ABSMOD,()OK     PATCHES TO IBLDR                F0000001
       *ENDAL                                                          F9999999
$IEDIT EDITM   SYSCK2,SCHF1,ALTER                                      G0000000
$IBMAP EDITM   30,M94,ABSMOD,()OK      PATCHES TO THE EDITOR           G0000001
       *ENDAL                                                          G9999999
$IEDIT UTILTY  SYSCK2,SCHF1,ALTER                                      H0000000
$IBMAP UTILTY  10,M94,ABSMOD,()OK      TAPE POSITION RECORD            H0000001
       *ENDAL                                                          H9999999
$IEDIT STPTP   SYSCK2,SCHF1,ALTER                                      I0000000
$IBMAP STPTP   1000,M94,ABSMOD,()OK    SETUP UNBLOCKED TAPE TO TAPE    I0000001
       *ENDAL                                                          I9999999
$IEDIT STPDK   SYSCK2,SCHF1,ALTER                                      J0000000
$IBMAP STPDK   1000,M94,ABSMOD,()OK    SETUP UNBLOCKED TAPE TO DISK    J0000001
       *ENDAL                                                          J9999999
$IEDIT BTPTP   SYSCK2,SCHF1,ALTER                                      K0000000
$IBMAP BTPTP   1000,M94,ABSMOD,()OK    BREAKDOWN TAPE TO UNBLOCKED TAPE K0000001
       *ENDAL                                                          K9999999
$IEDIT BDKTP   SYSCK2,SCHF1,ALTER                                      L0000000
$IBMAP BDKTP   1000,M94,ABSMOD,()OK    BREAKDOWN DISK TO UNBLOCKED TAPE L0000001
       *ENDAL                                                          L9999999
$IEDIT B7DBLK  SYSCK2,SCHF1,ALTER                                      M0000000
$IBMAP B7DBLK  1000,M94,ABSMOD,()OK    BREAKDOWN DISK TO PRINTER (720) M0000001
       *ENDAL                                                          M9999999
$IEDIT CDTP    SYSCK2,SCHF1,ALTER                                      N0000000
$IBMAP CDTP    1000,M94,ABSMOD,()OK    CARD TO TAPE UTILITY            N0000001
       *ENDAL                                                          N9999999
$IEDIT TPPR    SYSCK2,SCHF1,ALTER                                      O0000000
$IBMAP TPPR    1000,M94,ABSMOD,()OK    TAPE TO PRINT UTILITY           O0000001
       *ENDAL                                                          O9999999
$IEDIT TPPU    SYSCK2,SCHF1,ALTER                                      P0000000
$IBMAP TPPU    1000,M94,ABSMOD,()OK    TAPE TO PUNCH UTILITY           P0000001
       *ENDAL                                                          P9999999
$IEDIT NONSTD  SYSCK2,SCHF1,ALTER                                      Q0000000
$IBMAP NONSTD  10,M94,ABSMOD,()OK      TAPE POSITION RECORD            Q0000001
       *ENDAL                                                          Q9999999
$IEDIT CTLTP   SYSCK2,SCHF1,ALTER                                      R0000000
$IBMAP CTLTP   300,M94,ABSMOD,()OK     LOAD DCS FROM TAPE              R0000001
       *ENDAL                                                          R9999999
$IEDIT 4OLDR   SYSCK2,SCHF1,ALTER                                      S0000000
$IBMAP 4OLDR   60,M94,ABSMOD,()OK      4 CARD ABS LOADER               S0000001
       *ENDAL                                                          S9999999
$IEDIT CKPTR   SYSCK2,SCHF1,ALTER                                      T0000000
$IBMAP CKPTR   3000,M94,ABSMOD,()OK    SAVE AND RESTORE                T0000001
       *ENDAL                                                          T9999999
$IBSYS                                                                 V0000010
$REWIND        SYSLB2                                                  V0000020
```

• Figure 17. DCOS Master Alter/Edit Deck

```
$*             ASSEMBLIES ARE DONE                                      V0000030
$ENDFILE       SYSPP1                                                   V0000040
$REWIND        SYSPP1                                                   V0000050
$SWITCH        SYSUT2,SYSPP1                                            V0000060
$IBEDT                                                                  V0000070
      *EDIT    SYSLB2,MAP,MODS                                          V0000080
      *DUP     SYSLB2,SYSUT4,4                                          V0000090
TAPE  *REPLACE DCMUP                                                    V0000100
TAPE  *REPLACE CLEAR                                                    V0000110
TAPE  *REPLACE IBSYS                                                    V0000120
TAPE  *REPLACE SYSDMP                                                   V0000130
TAPE  *REPLACE DCIDR                                                    V0000140
TAPE  *MODIFY  IBJOB                                                    V0000145
TAPE  *MODIFY  IBJOBB                                                   V0000150
TAPE  *MODIFY  IBLDRO                                                   V0000155
TAPE  *MODIFY  IBLDRP                                                   V0000160
TAPE  *MODIFY  EDITOR                                                   V0000170
TAPE  *REPLACE UTILITY                                                  V0000180
TAPE  *REPLACE STPTP                                                    V0000190
TAPE  *REPLACE STPDK                                                    V0000200
TAPE  *REPLACE BTPTP                                                    V0000210
TAPE  *REPLACE BDKTP                                                    V0000220
TAPE  *REPLACE B7DBLK                                                   V0000230
TAPE  *REPLACE 00CDTP                                                   V0000240
TAPE  *REPLACE 00TPPR                                                   V0000250
TAPE  *REPLACE 00TPPU                                                   V0000260
TAPE  *REPLACE NONSTD                                                   V0000270
      *DUP     SYSUT2,SYSUT3,1     SKIP REST OF PUNCH FILE              V0000280
$EOF                               DUP REST OF SYSLB2 ONTO UT1          V0000290
$IBSYS                                                                  V0000300
$*             FIRST EDIT IS DONE                                       V0000310
$REWIND        SYSLB2                                                   V0000320
$SWITCH        SYSCK2,SYSLB3                                            V0000322
$ASSIGN        SYSCK2                                                   V0000324
$SWITCH        SYSUT1,SYSCK1       PUT NEW PRE SYS ON CK1               V0000330
$SWITCH        SYSUT2,SYSPP1       RESTORE ORIG UNITS                   V0000340
$EXECUTE       IBJOB                                                    V0000350
$IBJOB         GO,SOURCE                                                V0000360
$IEDIT PEDIT   SYSLB3,SCHF1,ALTER                                       U0000000
$IBMAP PEDIT   3000                    POST EDITOR                      U0000001
       *ENDAL                                                           U9999999
$DATA                        NO DATA FOLLOWS.PRESYS IS AT LOAD POINT    V0000370
$EOF                                                                    V0000380
$IBSYS                                                                  W0000010
$*             POST EDIT IS DONE                                        W0000020
$SWITCH        SYSLB2,SYSCK1                                            W0000030
$ASSIGN        SYSUT1                                                   W0000040
$IBEDT                                                                  W0000050
      *EDIT    SYSLB2         ADVISE MAP OPTION NOT BE USED             W0000060
      *DUP     SYSCK1,SYSUT1,1  KEEP 1ST DCSDT FILE                     W0000070
      *DUP     SYSCK2,SYSUT1,1  INSERT NEW DCSYS FILE                   W0000080
      *DUP     SYSCK1,SYSUT4,1  DELETE OLD DCSYS FILE                   W0000090
      *DUP     SYSCK1,SYSUT1,2  KEEP DCSYS 3RD AND 4TH FILES            W0000100
$EOF                                                                    W0000110
$IBSYS                                                                  W0000120
$STOP                                                                   W0000130
```

● Figure 17. DCOS Master Alter/Edit Deck (continued)

*EDIT CARD

Control information is transmitted to the System Editor by an *EDIT card. This control card is required for every edit run whether or not any options are specified, and must immediately follow the $IBEDT card. The format of the *EDIT card is:

```
7        16

*EDIT    SYSLB2[,MAP] [,MODS]
```

The parameters in this control card are interpreted as follows:

SYSLB2 specifies that SYSLB2 is to be edited.

MAP specifies that the names of the records and files on the new PRESYS library are to be listed on the system output unit.

MODS specifies that the maintenance control cards (*MODIFY, *REPLACE, etc.) or OCTAL alteration cards that affected a record be listed before the record name on the list specified by the MAP parameter. If the MAP parameter is not specified, this parameter is nullified.

Any text in columns 55 through 72 of the *EDIT card is used as the heading of all printed output from the system editor.

ARRANGEMENT OF DCMUP, DC-IBSYS, IBSYS SUBSYSTEMS AND THE SYSTEM EDITOR

The arrangement of IBSYS subsystems in the PRESYS library is indicated in the system name table of the DC-IBSYS Supervisor. This table is used by the DC-IBSYS Supervisor to determine the relative position of an IBSYS subsystem when a $EXECUTE card is read from the input file. The following two entries are made in the table for each subsystem:

```
BCI     1,sysnam
PZE     tfiles,index,nfiles
```

In the first entry, sysnam is the name of the IBSYS subsystem. It corresponds to the name on the $EXECUTE card that calls the subsystem. In the second entry, tfiles is the number of consecutive files that make up the subsystem; index is the number 1, corresponding to SYSLB1, where all IBSYS subsystems must reside; nfiles indicates that the first record of the subsystem is the first record of the nth file of the PRESYS library.

The Direct Couple Multiprocessor (DCMUP) must be the first record in the PRESYS Library. The DC-IBSYS records are always next, and the System Editor (IBEDT) is normally the last IBSYS system file. Up to fifty IBSYS subsystems may be placed in the system name table.

*PLACE CARD

The *PLACE card is used to modify the DC-IBSYS system name table. It is the only control card that can cause a change in the DC-IBSYS system name table. The *PLACE card causes the subsystem name and data concerning the location of the subsystem to be inserted into or deleted from the DC-IBSYS system name table.

Note: All *PLACE cards must immediately follow the *EDIT card.

The format of the *PLACE card is:

```
7        16

*PLACE   sysnam[,tfiles,index,order]
```

If the three parameters, tfiles, index, and order, are specified on the *PLACE card, an entry is posted in the DC-IBSYS system name table for the subsystem specified by sysnam. The parameter tfiles is the number of consecutive files that make up the subsystem; index is the number 1, and order is the order of the specified subsystem in the DCOS PRESYS Library in relation to other IBSYS subsystems. The first IBSYS subsystem (i.e., the one that immediately follows the DCMUP/DC-IBSYS file) is specified by an order of 1, the second subsystem is specified by an order of 2, and so forth.

If the parameter order is omitted and the parameters tfiles and index are included, the entry in the DC-IBSYS system name table will indicate that the specified subsystem is to be the last subsystem on the new PRESYS Library. Therefore, be sure to specify parameter order.

If the three parameters, tfiles, index, and order, are omitted from the *PLACE card, the entry in the DC-IBSYS system name table for the specified system is deleted.

MAINTENANCE OF THE SYSTEM NAME AND LOADER TABLES

The DC-IBSYS system name table is used by the DC-IBSYS Supervisor and is, therefore, part of the IBSYS record. When the DC-IBSYS Supervisor is loaded into 7090 storage, the system name table is loaded into locations SYSORG through SYSORG+99. When the system editor is subsequently loaded into core storage, it is loaded so that this table is not overlaid. The table is then updated with any *PLACE card read by the editor. When the DC-IBSYS record is edited onto SYSUT1 the system editor performs the following functions:

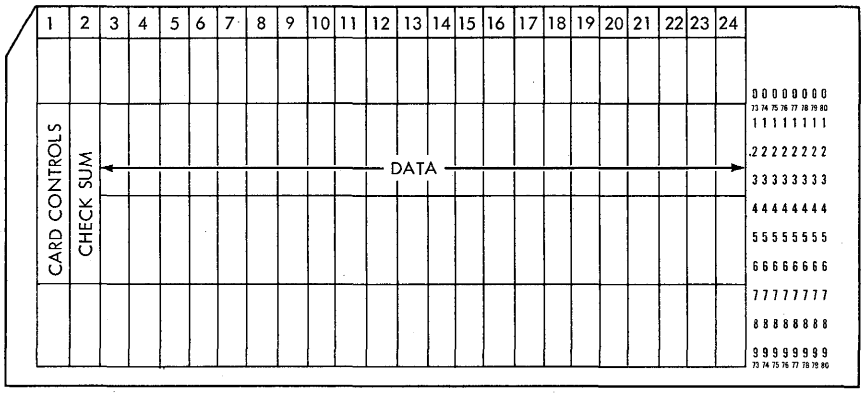
1. It examines location SYSCOR in the communication region of the system nucleus (see Appendix C) to determine the current location of the system name table to be replaced.
2. It inserts the updated system name into the DC-IBSYS record that is to be written on SYSUT1.
3. It transfers the updated table to the DC-IBSYS record that is going to be written on SYSUT1. The table is placed in the new DC-IBSYS record so that it will be located beginning at SYSORG whenever the DC-IBSYS Supervisor is loaded into core storage.

The use of *PLACE control cards does not cause rearrangement, deletion, or insertion of new system files. These operations must be performed by other system editor control cards.

Note: System name table is picked up from SYSLB1 not SYSLB2.

ALTERATION CARDS

Standard DCOS system records (i.e., DCMUP, Multiprocessor utilities, and DC-IBSYS and IBSYS subsystems) are derived from two types of alteration cards: absolute column-binary cards and octal cards. Non-IBSYS system (program) records or files are derived from one of the DCOS tape blocking programs and merged into the PRESYS Library by the system editor.



Word	Bit Positions	Contents
1	S	Must be blank.
	1	Must be blank.
	2	If punched, the System Editor does not compute a check sum from card data for comparison with card's prepunched check sum.
	9	Must be punched.
	11	Must be punched.
	12-17	Count of words in the card, excluding words 1 and 2.
	21-35	Absolute loading address -- the actual address of the core storage location where the first data or instruction word on this card is to be stored.
2	S-35	Check sum -- the logical sum (Add and Carry Logical Word) of all words in this card except word 2.
	3-24	Data or instruction words.

Figure 18. Absolute Column-Binary Card Format

ABSOLUTE COLUMN-BINARY CARDS

Absolute column-binary cards are the standard 22-word entry type cards illustrated in Figure 18. The data or instruction words, beginning with word 3 on the control card, are written on the PRESYS library in a standard PRESYS library record format. This format enables them to be loaded into sequential core storage locations, beginning at the location specified in bit positions 21-35 of word 1.

OCTAL CARDS

The format of octal cards is:

```
1       78        16                          72

octloc{*OCT  }  word 1,word 2,...,word n
      { OCTAL}
```

The use of an asterisk (*) in column 7 is optional. Each octal word must consist of 12 or fewer unsigned digits. When fewer than 12 digits are entered into a word, they are right-justified in the word. The octal words are separated by commas on the control card. The words are written on the new PRESYS library in a standard PRESYS Library record format that enables them to be loaded into sequential core-storage locations, beginning at the location specified by the octal address octloc.

STANDARD PRESYS LIBRARY RECORD FORMATS

Before records are edited onto the IBEDT output unit, they are converted, if necessary, by the system editor to a self-loading scatter-load format (shown in Figure 19). Records may be standard PRESYS library records, they may be converted from column-binary alteration cards or card images, or they may be a combination of these.

The first word in a standard record, following the initial input/output command (location LOC1 in Figure 19), must be the name of the record in BCD form without leading blanks. If a record is the first record of a file, the name of the record is also the name of the file. If the file is the first file of a subsystem, the name of the first record of the file is also the name of the subsystem (as specified in the DC-IBSYS system name table), as well as the name of the first file of the subsystem. When the DCOS PRESYS Library is being edited, the name of the first record of each subsystem in the new PRESYS library is compared with the entry for the subsystem in the DC-IBSYS system name table. If a discrepancy exists, an error message is printed.

The record name is the name printed if the MAP option on the *EDIT card is used. This name is also used by the system editor to identify a record, a file, or a subsystem. However, the DC-IBSYS Supervisor identifies the subsystem specified on a $EXECUTE card by referring to its system name table.

The last record on a DCOS PRESYS library tape has the identification *EOT and is used to inform the system editor of the logical end of tape.

First command word	IOCP LOC1,,N1
	BCI 1,NAME
N1-1 sequential words	
Second command word	IOCP LOC2,,N2
N2 sequential words	
Third command word	IOCP LOC3,,N3
N3 sequential words	
nth command word	IOCP LOCn,,Nn
Nn sequential words	
Last command word	IOCT 0,,0

● Figure 19. Standard PRESYS Library Record Format

Certain additional requirements apply to IBSYS subsystems residing in the PRESYS library. Each IBSYS subsystem consists of one or more files; each file consists of one or more records. The first record of each subsystem must load into location SYSTRA in the communication region of the nucleus, an instruction that transfers control to the first instruction in the subsystem that is to be executed. This is necessary because the DC-IBSYS Supervisor transfers control to SYSTRA after loading the first record of a subsystem specified on the $EXECUTE card.

When a subsystem or part of a subsystem is loaded into core storage, it must not overlay the DC-IBSYS Nucleus.

If the subsystem is not loaded into the area occupied by DC-IOEX, DC-IOEX is immediately available for use by the subsystem.

MAINTENANCE CONTROL CARDS

The actual addition, deletion, rearrangement, or modification of subsystems or of records and files within a subsystem is performed by using the maintenance control cards described in this section. An *PLACE card does no more than post, in the DC-IBSYS system name table, the position of an IBSYS subsystem.

Maintenance control cards must be in the same order as the records to which they refer. If a maintenance control card refers to a record out of sequence in the edit deck, the effect of the control card is nullified and a message is printed.

*MODIFY CARD

The format of the *MODIFY card is:

```
1      7        16

[TAPE] *MODIFY  recnam
```

where recnam is the name of a record in the old PRESYS Library that is to be modified.

If TAPE does not appear in columns 1-4, this control card causes the specified record to be consolidated with alteration cards that follow this control card on SYSIN1. The alteration cards on SYSIN1 may be any combination of octal and column-binary cards.

If TAPE does appear in columns 1-4, the *MODIFY card causes the specified record to be consolidated with alteration cards on SYSUT2. The alteration cards on SYSUT2 must be in the form of column-binary card images blocked in the standard DCOS format. Any card that contains the octal characters 2102 (IBM card code character, $) in the first 12 bits of word 1 is ignored.

The system editor performs the following steps after reading an *MODIFY card:

1. It transfers to SYSUT1 all files, file marks, and records on SYSLB2 up to, but not including, the record specified on the *MODIFY card.
2. It reads the specified record into core storage.
3. It deletes the IOCT command at the end of the record.
4. It appends an IOCP command of the following form to the record:

```
IOCP    loc,,n
```

where loc is the location specified in the first alteration card and n is the number of words on the first alteration card and on the alteration cards immediately following it that are to be loaded into contiguous core storage locations.
5. It places the n words from the alteration cards into the record following the newly appended command word.
6. It repeats steps 4 and 5, with appropriate values for loc and n, until, if TAPE is specified, a transfer card is encountered on SYSUT2, or, if TAPE is not specified, until a new system editor control card or end of file is encountered on SYSIN1.
7. It appends an IOCT command with a word count of zero to the end of the record.
8. It writes the expanded record onto SYSUT1.

After all actions specified by the *MODIFY card have been completed, SYSLB2 is positioned at the interrecord gap following the specified record.

*REPLACE CARD

The format of the *REPLACE card is:

```
1      7        16

[TAPE] *REPLACE {recnam} [,SYSxxx]
[FILE]          {sysnam}
```

If neither TAPE nor FILE appears in columns 1-4, the *REPLACE card causes an entire record (specified by recnam) in the old PRESYS library on SYSLB2 to be replaced on SYSUT1 by a new record formed entirely SYSIN1.

Before the specified record is replaced, all files, file marks, and records that precede the record on SYSLB2 are transferred to SYSUT1. The replacement record represented by the alteration cards is converted to the standard PRESYS Library record format shown in Figure 19.

The alteration cards on SYSUT2 must be column-binary card-image form if TAPE is specified. If neither TAPE nor FILE is specified, the alteration cards that follow the *REPLACE card on SYSIN1 may be any combination of octal and column-binary cards. As with the *MODIFY card, the end of the alteration cards is signalled by a transfer card if TAPE is specified or by a system editor control card or an end-of-file condition when TAPE and FILE are not specified.

When all operations have been completed that were called for by an *REPLACE card on which the first parameter was either TAPE or null, SYSLB2 is positioned at the interrecord gap following the record specified by recnam.

If FILE appears in columns 1-4 of the *REPLACE card and the name in columns 16-21 matches a subsystem name in the DC-IBSYS system name table, all records, files, and file marks are read in from SYSLB2 and written out on SYSUT1 until the specified subsystem is located on SYSLB2. When the subsystem is located, it is replaced on SYSUT1 by the files on SYSxxx. If the number of files in the replacement differs from the original number, an *PLACE card reflecting this must be inserted into the edit deck following the *EDIT card but preceding the maintenance control cards.

The first word of each record read from SYSxxx is checked to determine if it is an IOCP command. If it is not an IOCP command, the record is considered nonstandard and is duplicated without change on SYSUT1. If it is an IOCP command, the record is considered a standard PRESYS Library record. In this case, the record, except for the last word, is duplicated onto SYSUT1.

If the last word is an input/output command, it is changed to an

```
        IOCT    0,,0
```

command. Otherwise, this command is added to the end of the record.

If FILE appears in columns 1-4 of the *REPLACE card and if the name in columns 16-21 does not match a subsystem name in the DC-IBSYS system name table, all records, files, and file marks are read in from SYSLB2 and written out on SYSUT1 until a record with the specified name is located on SYSLB2. When the specified record is located, it and all succeeding records (if any) up to and including the next file mark are replaced on SYSUT1 by the next file and file mark on SYSxxx. The replacement records are processed by the system editor as previously described. To replace an entire file, the record name specified on the card must be the name of the first record in the file. However, if the name of the first record in the file matches a subsystem name in the system name table, all files in the subsystem are replaced, as previously described.

When all actions called for by an *REPLACE card specifying FILE have been completed, SYSLB2 is positioned at the interrecord gap following the last file (or part of a file) replaced.

*INSERT CARD

The format of the *INSERT card is:

```
 1      7        16
[TAPE] *INSERT  [FILEMK]
```

If TAPE and FILEMK are not specified, the *INSERT card causes a new record, formed from octal and/or column-binary alteration cards following the *INSERT card on SYSIN1, to be written on SYSUT1 at its current position. The end of the alteration cards on SYSIN1 is indicated by an end of file or a new system editor control card.

If TAPE appears in columns 1-4 but FILEMK is not specified, the new record is formed from alteration cards in the form of column-binary card images on SYSUT2. The end of the alteration cards on SYSUT2 is indicated by a transfer card.

The new record, represented by the alteration cards on SYSIN1 or SYSUT2, is converted to the standard PRESYS library record format shown in Figure 19.

If FILEMK is specified, a file mark is inserted on SYSUT1 at its current position. A single *INSERT card cannot be used both to insert a new record and to write a file mark.

The position of SYSLB2 is not changed by an *INSERT card.

*REMOVE CARD

The format of the *REMOVE card is:

```
1       7          16

[FILE]  *REMOVE   ⎧recnam⎫
                  ⎨sysnam⎬
                  ⎩FILEMK⎭
```

If neither FILE nor FILEMK is specified, the *REMOVE card causes the record specified in columns 16-21 to be skipped on SYSLB2 and omitted from SYSUT1. Before the specified record is removed, all files, file marks, and records that precede the record on SYSLB2 are transferred to SYSUT1. After the *REMOVE card has been processed, SYSLB2 is positioned immediately after the record that was omitted from SYSUT1.

If FILE is not specified, an *REMOVE card on which FILEMK appears causes the next file mark on SYSLB2 to be omitted from SYSUT1. Records preceding the file mark are transferred to SYSUT1. After the *REMOVE card has been processed, SYSLB2 is positioned after the file mark that was omitted from SYSUT1.

If FILE is specified and the name in columns 16-21 is a subsystem name in the DC-IBSYS system name table, all files and accompanying file marks associated with the subsystem (as defined by the DC-IBSYS system name table entry) are skipped on SYSLB2 and omitted from SYSUT1.

If FILE is specified and if the name in columns 16-21 is the name of a record but not the name of a subsystem posted in the DC-IBSYS system name table, the specified record and all succeeding records (if any) up to and including the next file mark are skipped on SYSLB2 and omitted from SYSUT1. If an entire file is to be removed, the name specified on the *REMOVE card on which FILE appears must be the name of the first record in the file. However, if the name of the first record in the file matches a subsystem name in the DC-IBSYS system name table, all files in the subsystem are removed, as previously described.

Before a specified subsystem or record is removed, all records, files, and file marks that precede it on SYSLB2 are transferred to SYSUT1.

*AFTER CARD

The format of the *AFTER card is:

```
1       7          16
[FILE]  *AFTER    ⎧recnam⎫
                  ⎨sysnam⎬
                  ⎩FILEMK⎭
```

If neither FILE nor FILEMK is specified, the *AFTER card causes the reading of control cards to be suspended until all files, file marks, and records on SYSLB2, up to and including the record specified in columns 16-21, have been transferred to SYSUT1.

If FILEMK is specified in columns 16-21, the next file mark on SYSLB2 and all records preceding it are transferred to SYSUT1.

If FILE is specified and if the name in columns 16-21 is a subsystem name in the DC-IBSYS system name table, all files, file marks, and records on SYSLB2, up to and including the files and accompanying file marks associated with the specified subsystem (as defined by the DC-IBSYS system name table entry), are transferred to SYSUT1.

If FILE is specified and if the name in columns 16-21 is the name of a record but not the name of a subsystem posted in the DC-IBSYS system name table, all files, file marks, and records on SYSLB2, up to and including the next file mark following the specified record, are transferred to SYSUT1.

*DUP CARD

The format of the *DUP card is:

```
      7          16

      *DUP       SYSxxx,SYSyyy,n
```

The *DUP control card transfers n files from SYSxxx to SYSyyy. Neither SYSxxx nor SYSyyy may be SYSLB1. The transfer includes all files through the nth file mark read on SYSxxx. If n is blank, 1, or 0, one file and the file mark following it are transferred to SYSyyy.

The *DUP card may be used to insert special files into the PRESYS library. Such special files might include IBSYS subsystem data files, such as the IBJOB Library (IBLIB), or non-IBSYS systems.

*REWIND CARD

The *REWIND control card rewinds the unit assigned to the specified system unit function. SYSLB1 may not be specified on the *REWIND card.

The format of the *REWIND card is:

```
    7        16

    *REWIND  SYSxxx
```

The *REWIND card may be used to define the starting position before using an *DUP card. The *REWIND and *DUP cards may be used to rearrange subsystems in the PRESYS library or to incorporate into the PRESYS library special-format records that have been previously edited by a subsystem.

*CHECK CARD

The *CHECK control card causes a test to be made to ensure that the correct number of editing cards were read and that the correct PRESYS library tape was processed.

The format of the *CHECK card is:

```
    7        16

    *CHECK   [count][,oldnam][,newnam]
```

The parameter count is the number of alteration cards and maintenance control cards, including the *CHECK card but not the *EDIT card. The parameter oldnam is a name that is compared with the PRESYS library name in the fourth word of the *EOT record on SYSLB2. If the two names are not the same, an error message is printed. The parameter newnam is the name assigned to the new PRESYS library on SYSUT1 (i.e., the name placed in the fourth word of the *EOT record).

The parameters must appear in the order shown in the format description. If a parameter is null, the corresponding operation is ignored.

If more than one *CHECK card is read during an edit run, only the last one is processed.

*REMARK CARD

The *REMARK control card causes the characters in columns 16-72 to be listed on the system output unit.

The format of the *REMARK card is:

```
    7        16

    *REMARK  any remark
```

The *REMARK card may not be placed between the control cards (*MODIFY, *REPLACE or *INSERT), and the alteration cards immediately following these control cards.

TERMINATION OF EDITING

The system editor completes the editing process when an end of file is encountered on SYSIN1. The system editor performs the following steps to terminate editing:

1. It transfers from SYSLB2 to SYSUT1 all remaining files, file marks, and records, up to but not including, the *EOT record on SYSLB2.
2. It reads the *EOT record from SYSLB2 and performs the tests and changes specified on the last *CHECK card read. If there is no *CHECK card, no tests or changes are made.
3. It writes the new *EOT record and a file mark on SYSUT1.
4. It rewinds SYSLB2 and SYSUT1.

The system editor then reads the cards on SYSIN1 until it finds one of the following control cards: $IBSYS, $EXECUTE, $JOB, or $STOP.

EDITING RELOCATABLE RECORDS

If a standard FORTRAN II relocatable record is located on SYSUT2 when one of the following maintenance control cards is read, the system editor converts it to a format similar to the standard 7090 System Library format.

```
1     7        16

TAPE  *REPLACE recnam
TAPE  *INSERT  recnam
```

Two IOCP commands and a record name are placed at the beginning of the record by the editor, as follows:

```
        IOCP    18944,,1
        BCI     1,name
        IOCP    18946,,n
```

where name is the record name specified on the *REPLACE or *INSERT card, and n is the number of words in the relocatable record. An IOCT command with a word count of zero is appended to the record.

After a relocatable record has been placed in the PRESYS library in the standard PRESYS system library format, all maintenance control cards except the *MODIFY card that affect nonrelocatable records in the standard PRESYS library format also affect the relocatable record.

EDITING EXAMPLES

The following are examples of typical editing jobs performed by the System editor.

EXAMPLE 1

The sample job deck in Figure 20 might be used to prepare a new PRESYS library tape with corrections to record REC1 and complete replacement of records ABC and XYZ.

EXAMPLE 2

The sample job deck in Figure 21 might be used to rearrange the subsystems on the PRESYS tape so that subsystem SYSTMA, consisting of two files located just after the IBJOB files, will be repositioned just before the System Editor, following SYSTMX.

```
$JOB           9,5,500        PRESYS EDIT
$SETUP LB2     OLDPRE,NORING           MOUNT OLD PRESYS
$SETUP UT1     NEWPRE                  MOUNT TAPE FOR NEW PRESYS
$SETUP UT2     NEWPCH,NORING
$IBSYS
$ASSIGN        SYSLB2
$ASSIGN        SYSUT1
$ASSIGN        SYSUT2
$IBEDT
      *EDIT    SYSLB2,MAP,MODS
      *MODIFY  REC1
       .
       . Octal and/or Column-binary Cards on SYSIN1
       .
TAPE  *REPLACE ABC
       .
       . Column-binary Card Images on SYSUT2 Followed by Transfer Card
       .
TAPE  *REPLACE XYZ
       .
       . Column-binary Card Images on SYSUT2
       .
$EOF
$IBSYS
$STOP
```

Figure 20. Sample Edit 1

```
$JOB           9,5,500        REARRANGE SYSTMA ON PRESYS LIBRARY TAPE
$SETUP LB2     OLDPRE,NORING           MOUNT OLD PRESYS
$SETUP UT1     NEWPRE                  MOUNT NEW PRESYS (TO BE WRITTEN)
$IBSYS
$ASSIGN        SYSLB2
$ASSIGN        SYSUT1
$IBEDT
       *EDIT   SYSLB2,MAP,MODS
       *PLACE  SYSTMA                  DELETE OLD SYSTMA REFERENCE IN SYSNAM
       *PLACE  SYSTMA,2,1,3            INSERT NEW SYSTMA REFERENCE IN SYSNAM
       *PLACE  SYSTMX
       *PLACE  SYSTMX,2,1,2
FILE   *AFTER  IBJOB                   POSITION SYSLB2 AFTER IBJOB
       *REWIND SYSUT4                  ENSURE SYSUT4 IS REWOUND
       *DUP    SYSLB2,SYSUT4,2         DUPLICATE SYSTMA (2 FILES)
FILE   *AFTER  SYSTMX                  POSITION SYSLB2 AFTER SYSTMX
       *REWIND SYSUT4
       *DUP    SYSUT4,SYSUT1,2         PUT SYSTMA ON NEW LIBRARY TAPE
$EOF                                   FINISH UP EDIT
$IBSYS
$STOP
```

• Figure 21. Sample Edit 2

The 7090-7040 Direct Couple Operating System is distributed in a binary form ready for disk residence. The system in this form, the Direct Couple Operating System Library (DCSYS), can be used without change for production job processing. However, DCSYS is not modifiable or maintainable. A second form of DCOS, the Presystem (PRESYS) Library can easily be modified.

This section contains information for performing the following modifications to the PRESYS library:

1. Preparing a backup DCOS Distribution Tape or PRESYS Library.
2. Changing the system input/output configuration.
3. Incorporating a user-designed installation accounting routine for the 7090.
4. Incorporating IBM IBSYS subsystems and user programs under DC-IBSYS.
5. Incorporating user programs to serve as DCMUP background utilities.
6. Incorporating non-IBSYS 7090 systems or programs to be run in compatibility mode.

A complete PRESYS library may be created from a 7090/7094 IBSYS Library. See Appendix D for further information about this procedure.

PREPARING A BACKUP DCOS DISTRIBUTION TAPE OR PRESYS LIBRARY TAPE

The procedure for preparing a backup DCOS Distribution Tape is optional. However, this procedure is strongly recommended, since it produces a PRESYS library map for future reference and a duplicate PRESYS library for emergency use. The procedure is as follows:

1. Place the sample job deck shown in Figure 22 in the card reader.
2. Initiate DCOS operation as described in the publication IBM 7090-7040 Direct Couple Operating System: Operator's Guide.

A duplicate of the distributed DCOS PRESYS Library will be produced on SYSUT1, and a map of the system library will be printed on the 1403 printer.

A backup copy of the entire DCOS Distribution Tape may be obtained by replacing the *DUP card shown in Figure 22 with the following card:

```
*DUP SYSLB2, SYSUT1,4
```

However, it is not advisable to use the MAP option in the *EDIT card.

The parameter Y00014 is the BCD representation of the instruction

```
TXL      68,0,0
```

This word is required as the second word of the first record of all DCOS distribution tapes.

CHANGING THE SYSTEM INPUT/OUTPUT CONFIGURATIONS

The 7090 input/output configuration shown in Figure 23 is simulated by the 7040 for IOBASE 0. The 7090 has no real

```
$JOB            ,15,500         PRESYS LIB BACKUP
$SETUP LB2      DCSDT,NORING         MOUNT DISTRIBUTION TAPE
$SETUP UT1      Y00014               MOUNT TAPE FOR BACKUP
$IBSYS                               IBSYS SEGMENT
$ASSIGN         SYSLB2               CONNECT TO
$ASSIGN         SYSUT1               MOUNTED TAPES
$IBEDT
      *EDIT     SYSLB2,MAP
      *DUP      SYSLB2,SYSUT3,4
$EOF
```

•Figure 22. Preparing a Backup PRESYS Tape input/output units. The preassembled 7090 pseudo input/output configuration should serve all DCOS users without change. If a user wishes to modify DC-IBSYS to redefine the 7090 input/output configuration, he must also change DCMUP to reflect his redefinition. For example, the one-for-one correspondence between 7090 unit control blocks and 7040 file control blocks must be maintained. Changes in the assignment of system unit functions (Figure 24) require a respecification of IOBASE 0. SYSLB1 (A1) may not be reassigned.

7090 Pseudo Channel	Unit Type	No. of Units	Attached
A	711 Card Reader	1	RDA
	721 Card Punch	1	PUA
	716 Printer	1	PRA
	729 Tapes	10	A1-A0
B	729 Tapes	10	B1-B0
C	729 Tapes	6	C1-C6
D	729 Tapes	6	D1-D6

Figure 23. Distributed 7090 Input/Output Configuration Simulated by DCMUP (IOBASE 0)

7040 INPUT/OUTPUT CONFIGURATION

The distributed binary DCOS system is assembled for the 7040 input/output configuration designated in Figure 25. Additional modules of disk or drum may be activated by using the DCOS Posteditor. This procedure is described in the section "Respecification of 7040 Disk (Drum) Environment."

The second and third printers may be activated either symbolically or through the use of alteration cards. The assembly parameters PR1, PR2, and PR3, located at the beginning of the DCOS listing, indicate the availability of printers 1, 2, and 3, respectively. In the distributed DCOS, PR1=1, PR2=0, and PR3=0. To activate printers 2 and 3, set PR2 and PR3 to 1. Printers may also be activated using octal alteration cards specifying the following modifications to DCMUP:

1. To make printer 1 available, insert
```
          AXT    1,1
```
   at location PRINT1+1. (PRINT1+1 contains this instruction in the distributed version of DCOS.)
2. To make printer 2 available, insert
```
          AXT    1,1
```
   at location PRINT2. (PRINT2 contains the instruction
```
          AXT    0,1
```
   in the distributed version.)
3. To make printer 3 available, insert
```
          AXT    1,1
```
   at location PRINT3. (PRINT3 contains the instruction
```
          AXT    0,1
```
   in the distributed version.)

UNIT	SYSTEM UNIT ASSIGNMENT
RDA	SYSCRD
PUA	SYSPCH
PRA	SYSPRT
A1	SYSLB1
A2	SYSIN1, SYSIN2
A3	SYSOU1, SYSOU2
A4	SYSPP1, SYSPP2
A5	SYSCK1
B1	SYSUT1
B2	SYSUT2
B3	SYSUT3
B4	SYSUT4
B5	SYSCK2

Figure 24. Assignment of System Unit Functions

To specify additional tape units, DCMUP may be reassembled modifying the tape parameters to reflect the configuration. The tape parameters can be found in the distributed DCOS listing. Their symbolic locations are TAPOB, TAPOC, TAPOD, and TAPOE for channels B, C, D, and E respectively.

Additional tape units may be activated using octal alteration cards. To activate a unit, an alteration card should be used to set the sign bit of the first word of the units Unit Control Block (UCB) to plus.

Following are examples of how typical system modifications are made.

As distributed, DCOS is assembled with tape drives C1 through C5 available for use. The File Control Blocks (FCBs) are all assigned to disk. In order to modify this, the following procedures should be used.

1. There are three methods for changing tape availability.
   a. By assembly. The number of tape drives available for use on the 7040 is defined by a group of values assigned through the use of the SET pseudo-operation; that is, the symbolic locations TAPOB,

TAPOC, TAPOD, and TAPOE (tape parameters) are set to the number of drives available on each of their respective channels: B, C, D, and E. Also, if tape drives are to be located on channels D and E, the symbolic location DETP must be set to 1.

b. By patching using the SNAP-PATCH Loader. The availability or non-availability of a tape drive is indicated by the sign of word 1 of the Unit Control Block (UCB). Plus indicates availability, and minus indicates non-availability. A patch, consisting of a series of MSPs and MSMs affecting the sign of the first words of the UCBs involved, is inserted at the octal location corresponding to the symbolic location TAPSE-1. Also, the word at symbolic location TOTAL in the setup area of DCOS is patched to agree with the total number of tape drives now available.

<u>Note</u>: If tape drives on channels D and E are to be made available, the symbol DETP must have been set to 1 during assembly.

c. By editing. This process is the same as the one for patching except that the changes are edited onto the tape instead of being contained in a patch deck (SNAP-PATCH Loader).

2. There are three methods for permanently assigning a File Control Block to tape.

a. By assembly. The FCBs are assumed to always be referencing files on disk. The $SETUP card is used to make temporary assignment to tape. To assign a file permanently to tape, word 1 of the FCB must be altered to contain the location of a UCB, and word 5 must be positive. To accomplish this, a series of ORG cards, each causing origin at a FCB word to be modified (and each followed by the necessary information to modify it), is assembled in just prior to the Snapshot routine. These cards should be of the following form:

```
ORG  UNITS+8n
MZE  BEGTPS+40k+4m-4
ORG  UNITS+8n+4
PZE
```

where UNITS is the beginning location of the FCBs, n is the logical FCB number, BEGTPS is the beginning location of the UCBs, k is the channel index (k=0 for channel B, k=1 for channel C, k=2 for channel D, and k=3 for channel E), and m is the tape unit number.

b. By patching using the SNAP-PATCH Loader. The procedure for patching a FCB, so that the file is permanently assigned to tape, is similar to the assembly procedure described above. The word at the octal location corresponding to symbolic location UNITS+8n is patched to contain (in the address portion) the octal location of the desired tape UCB(BEGTPS+40k+4n-4), and UNITS+8n+4 is patched to be plus. Also, the word at symbolic location TOTAL in the setup area of DCOS is patched to agree with the total number of tape drives now available. (To do this with FORTRAN II it is desirable to change, by assembly, IOBASE 2 so that it is the same as IOBASE 0. See Example 3 below.)

<u>Note</u>: k, m, and n mean the same as for assembly.

c. By editing. The process is the same as for patching except that the changes are edited onto the tape instead of being contained in a patch deck (SNAP-PATCH Loader).

Following are examples to illustrate these procedures:

<u>Example 1</u> - Assign tape drives C4 and C5 permanently to the FCBs corresponding to UT1 and UT2 in IBSYS. Assuming the minimum system, there are 5 tapes available on channel C. UT1 and UT2 are logical FCBs 10 and 11.

a. By assembly

```
             *ALTER   5,5
TAPOC        SET      3
             *ALTER   15706
             ORG      UNITS+80
             MZE      BEGTPS+40+16-4
             ORG      UNITS+80+4
             PZE
             ORG      UNITS+88
             MZE      BEGTPS+40+20-4
             ORG      UNITS+88+4
             PZE
```

b. By patch or edit (shown symbolically)

```
L(TAPSE)-1   OCT (TRA L(PATCH))
L(PATCH)     OCT (MSM L(BEGTPS)+52)
L(PATCH)+1   OCT (MSM L(BEGTPS)+56)
L(PATCH)+2   OCT (AXT L(AVTPEN)-L(BEGTPS),1)
L(PATCH)+3   OCT (TRA L(TAPSE))
L(TOTAL)     OCT (PZE 3)
L(UNITS)+80  OCT (MZE L(BEGTPS)+52)
L(UNITS)+84  OCT (PZE)
L(UNITS)+88  OCT (MZE L(BEGTPS)+56)
L(UNITS)+92  OCT (PZE)
```

<u>Note</u>: In the preceding patch or edit

example all numbers are given in base 10, all location fields would be the octal locations equivalent to what is given, and all variable fields would be octal equivalent of the instructions given.

This is how the example above would appear as an actual octal patch referring to the list of the minimum system as distributed.

```
01056      OCT      002000001240
01240      OCT      562306042060
01241      OCT      562306042064
01242      OCT      077400100240
01243      OCT      002000001057
35064      OCT      000000000003
44216      OCT      400000042060
44222      OCT      000000000000
44226      OCT      400000042064
44232      OCT      000000000000
```

Example 2 - Making available tape drives D1 through D5, assuming the minimum system with 5 tapes available on channel C.

a. By assembly

```
             *ALTER   6,6
TAPOD        SET      5
             *ALTER   40,40
DETP         SET      1
```

b. By patch or edit (Note: DETP must have been given the value 1 in the assembly)

```
L(TAPSE)-1   OCT (TRA L(PATCH))
L(PATCH)     OCT (MSP L(BEGTPS)+80)
L(PATCH)+1   OCT (MSP L(BEGTPS)+84)
L(PATCH)+2   OCT (MSP L(BEGTPS)+88)
L(PATCH)+3   OCT (MSP L(BEGTPS)+92)
L(PATCH)+4   OCT (MSP L(BEGTPS)+96)
L(PATCH)+5   OCT (AXT L(AVTPEN)-L(BEGTPS),1)
L(PATCH)+6   OCT (TRA L(TAPSE))
L(TOTAL)     OCT (PZE 10)
```

Note: In the preceding patch or edit example all numbers are given in base 10, all location fields would be the octal locations equivalent to what is given, and all variable fields would be the octal equivalent to the instructions given.

Example 3 - IOBASE 2 definition to allow the permanent assignment of utilities to tape.

```
FBPTRS    ((AVAIL,6),(A2),(A3),(B4),(A5),
          (A6),(A7),
ETC       (A8),(A9),(A0),(B1),(B2),(A4),
          (B3),(B5),
ETC       (B6),(B7),(B8),(B9),(B0),(C1),
          (C2),(C3),
ETC       (AVAIL,1),(AVAIL,2),(AVAIL,3),
          (D1),(D2),
ETC       (D3),(AVAIL,4),(AVAIL,5),(A1),
          (CRA),(PUA),(PRA))
```

Channel	Interface Number	Unit Type	Maximum Number	Distributed Configuration
A	3	1402	1	1
	3	1403	1	1
	2	1403	1	0 (1)[2]
	1	1403	1	0 (1)[2]
	3	1014	1	1
B		1301/7320	4[1]	1301-1
		729	10	0(B1-B0)[2]
C		1301/7320	4[1]	0
		729	10	C1-C5 (C1-C0)[2]
D		1301/7320	4[1]	0
		729	10	0 (D1-D0)
E		1301/7320	4[1]	0
		729	10	0 (E1-E0)

[1]Maximum Total = 10 modules.
[2]Can be activated by octal alteration cards.

Figure 25. 7040 Input/Output Configuration

INCORPORATING A USER-DESIGNED INSTALLATION ACCOUNTING ROUTINE UNDER DC-IBSYS

A user may design an installation accounting routine tailored to his requirements and incorporate it into the DCOS PRESYS Library as part of the DC-IBSYS file. The accounting routine may be designed to perform a variety of functions, such as timing jobs or producing statistical data about the processing of jobs. Since accounting practices vary considerably among installations, a specific 7090 accounting routine is not provided with DC-IBSYS.

Three one-word entries in the communication region of the DC-IBSYS Nucleus are used specifically for accounting purposes: SYSIDR, SYSACC, and SYSPID. The function of each of these entries is described in detail in Appendix C.

An installation accounting routine may be incorporated either as part of the DC-IBSYS Monitor for use by the monitor and each of the subsystems or as part of a subsystem for use by the subsystem only. Both methods of incorporating an accounting routine are described in the following text.

DESIGNING AN INSTALLATION ACCOUNTING ROUTINE

There are several considerations about the relationship of an installation accounting routine to the IBSYS operating system which should be kept in mind when designing an accounting routine. These considerations are discussed below.

Location and Size

An accounting routine must be loaded into upper core storage and must not occupy more than 500 words. There are two subsystems that will not respect the upper 500 words of core storage and may overlay all or part of an accounting routine.

1. FORTRAN II respects only the upper 64 words of core storage.
2. In IBJOB, if $DUMP or $PATCH cards are used, the JDUMP routine is loaded into upper core storage overlaying the accounting routine, and the contents of SYSIDR are replaced by TRA 2,4.

Calling Sequence

Access to an accounting routine is through location SYSIDR in the Communication Region of the System Nucleus. SYSIDR must be patched to contain a transfer to the accounting routine. Whenever the System Monitor or any of its subsystems processes a $JOB or a $ID card, a transfer is made to the accounting routine through SYSIDR (i.e., TSX SYSIDR,4). In addition, some processors under control of IBJOB, Commercial Translator, and 9PAC make a sign-on transfer to the accounting routine before they begin processing. There may be more than one sign-on for a job under IBJOB or Commercial Translator. For example, IBMAP will sign-on before loading the object program assembled by IBMAP. When all processing is complete for a job under IBJOB or Commercial Translator, there will be one (and only one) sign-off transfer to the accounting routine.

The calling sequence used in transferring to the accounting routine is:

```
        TSX     SYSIDR,4
        pfx     loc,t,n
                return
```

where:

pfx = PZE for $JOB/$ID calls
PON for sign-on calls
PTW for sign-off calls
PTH for intraprocessor calls
(prefix codes FOR, FIVE, SIX, and SVN may be used for special purposes by an installation)
loc = location of first word of an identifying message or a control card buffer (not always present)
t = identifier of sign-on and sign-off calls
1 = compiler
2 = assembler
3 = loader
4 = execution
5 = utility
(6 and 7 may be used for special purposes by an installation)

n = number of words occupied by identifying message or control card buffer (not always present)

In some cases the sign of the AC is tested upon return from the accounting routine. If the sign is found to be plus (+), processing continues. If the sign is found to be minus (-), processing is terminated and control is returned to the supervising monitor. Figure 25A gives the contents of the parameter words for the calls by various processors and indicates whether the sign of the AC is tested on return (the sign of the AC is never tested on return from a sign-off call).

$JOB and $ID Cards

The contents of the $JOB or $ID card are stored in a buffer prior to being printed on-line. The location and length of this buffer is stored in the parameter word (PZE) of the calling sequence so that the accounting routine may have access to the data in columns 16-72 of the $JOB card or columns 7-72 of the $ID card. The data on these cards may be set up in any form desired for use by the accounting routine.

SYSACC

The System Monitor or the subsystem processing a $JOB or $ID card tests location SYSACC. If the contents of SYSACC are zero, the card is printed on-line. If SYSACC contains anything other than zero, the card is not printed. In the distributed version of IBSYS, SYSACC contains:

```
PZE     0
```

If the accounting routine is to control printing of the $JOB and $ID cards, SYSACC must be patched with the following card:

```
1      8  16
-      -  --
SYSACC PZE any nonzero value
```

If both SYSACC and SYSPID are nonzero, the IBJOB Monitor prints a special page heading on each page of the job listing. The data for the special heading is contained in a 10-word buffer which the user must provide. The location of the first word of the buffer must be placed in the address portion of SYSACC. The 10-word buffer must be in the following form:

Words 1-2	Time of day
Words 3-4	Date
Words 5-7	Primary identification
Words 8-9	Secondary identification (e.g., charge number)
Word 10	Installation or run code

The routine in IBJOB that prints the special page heading is entered at symbolic location HED.

SYSPID

Location SYSPID in the System Nucleus is used only by the IBJOB Monitor. The IBJOB Monitor tests SYSPID only if SYSACC is found to be nonzero. The status of SYSPID determines whether the standard or special page heading is to be printed on the job listing. If SYSPID is zero, the standard heading is printed. If SYSPID is nonzero, the special page heading is printed. The 10-word buffer described in the "SYSACC" section must be provided and a pointer to the buffer must be placed in SYSACC if SYSPID is nonzero. If location SYSPID is to be changed, it must be patched with a card serialized IBB42030.

	$JOB/$ID		Sign-On		Sign-Off
Processor	Parameter Word[1]	Sign of AC Checked	Parameter Word	Sign of AC Checked	Parameter Word
IBSYS Monitor	PZE loc,,n	No			
Symbolic Update	PZE loc,,n	Yes			
Sort	PZE loc,,n	No			
IOCS	PZE loc,,n	No			
Utilities	PZE loc,,n	No			
IBJOB	PZE loc,,n	Yes			PTW 0,0,0
IBFTC			PON 0,1,0	[2]	
IBCBC			PON 0,1,0	No	
IBMAP			PON loc,2,n	Yes	
Loader	PZE loc,,n	Yes	PON loc,3,n	Yes	PTW loc,3,n
Execution					PTW 0,4,0
CT Monitor	PZE loc,,n	No			
Compiler			PON 0,1,0		PTW 0,1,0
Loader			PON 0,3,0	Yes	PTW 0,3,0
Postprocessor			PON loc,5,n	Yes	PTW loc,5,n
KPOUT[3]					PTH loc,,n
9PAC Monitor	PZE loc,,n	No			
Compiler			PON loc,1,n	No	
FORTRAN II	PZE loc,,n	Yes			

[1]n may be assumed to be 14 when it is omitted from the decrement field in $JOB/$ID calls.
[2]prints error messages if AC is minus but does not discontinue processing.
[3]KPOUT routine is entered when an error condition is encountered which necessitates discontinuing processing.

Figure 25A. Installation Accounting Routine Calling Sequence Parameter Words

A specific record has been reserved for incorporating the accounting routine. This record, DCIDR, is loaded by the DC-IBSYS Supervisor after the supervisor has generated the nucleus, except when IBSYS is loaded to get a dump. After the installation accounting routine has been assembled, it should be edited into the PRESYS library, replacing the dummy DCIDR record of the distributed system.

Note: The distributed DCIDR performs no accounting functions and does not affect the contents of any DC-IBSYS accounting location.

INCORPORATING AN ACCOUNTING ROUTINE INTO A SUBSYSTEM

If an accounting routine is designed for use only with a specific subsystem, the accounting routine should be assembled and then appended to the first record of the subsystem, using an *MODIFY card when editing the PRESYS library (see the section "System Editor"). This will result in the accounting routine's being loaded whenever the first record of the subsystem is loaded after a $EXECUTE card containing the name of the subsystem has been read. The first record of the subsystem should also be appended to overlay SYSIDR with an instruction that transfers control to the beginning of the installation accounting routine. If necessary, SYSCOR should be overlaid to change the definition of the end of usable core storage. SYSACC may also be overlaid if it is desired to change it to nonzero.

A $IBSYS card followed by a $RESTORE card should be placed at the end of a job segment performed by a subsystem containing an accounting routine if it is followed on the system input file by job segments to be performed by other subsystems or by the DC-IBSYS Monitor. The $RESTORE card restores locations SYSIDR, SYSCOR, and SYSACC to their original states, nullifying any changes made by the subsystem containing the accounting routine.

INCORPORATING IBSYS SUBSYSTEMS UNDER DC-IBSYS

Additional IBSYS subsystems should not be edited onto the DCOS system in a position after any DCOS utilities or nonstandard systems.

COMPATIBILITY MODE

All standard IBM IBSYS subsystems except the Restart Program may be incorporated into DCOS. The procedure for incorporating an IBSYS subsystem to be executed in compatibility mode is outlined here.

1. Using the DCOS Tape Blocking Routine, block the 7090/7094 IBSYS System Library containing the subsystem into the 460-word standard DCOS record format.
2. Edit the subsystem into the PREsys library using the procedures described in the section "System Editor." Figure 26 is an example of adding an IBSYS subsystem to the PRESYS Library.
3. Run the output resulting from step 2 through the DCOS Posteditor.

DIRECT MODE

A user may design a program to be executed in direct mode and insert it into the PRESYS library with other IBSYS subsystems. After the program has been coded and assembled, it may be inserted into the PRESYS library using a job deck similar to the sample job deck in Figure 27.

The name of the direct-mode subsystem (i.e., the name that specifies the system on the $EXECUTE card) must be inserted into the DCTBL table of the DC-IBSYS Supervisor. The program can then be called into core storage by a $EXECUTE card and can be executed. The subsystem name is inserted into the DCTBL table when the subsystem is edited into the PRESYS library.

```
$JOB             ,5,500         INSERT FORTRAN II, VER. 3 INTO DCOS
$SETUP LB2       OLDPRE,NORING       MOUNT DCOS PRESYS BACKUP,
$SETUP UT1       NEWPRE              NEW DCOS PRESYS OUTPUT
$SETUP UT4       IBSYS               BLOCKED IBSYS LIBRARY
$ASSIGN          SYSLB2              DC-IBSYS GET ASSIGNMENTS
$ASSIGN          SYSUT1              OF MOUNTED
$ASSIGN          SYSUT4              TAPE DRIVES
$IBEDT
      *EDIT      SYSLB2,MAP,MODS
      *PLACE     FORTRA,m,1, order   M FILES, ORDER=LAST
      *DUP       SYSUT4,SYSUT3,n     POSITION IBSYS TO FORTRAN
FILE  *AFTER     IBJOB               POSITION SYSLB2
      *DUP       SYSUT4,SYSUT1,m     FORTRAN TO DCOS PRESYS
$EOF
```

•Figure 26. Incorporating an IBSYS Sybsystem (Compatibility Mode)

For the sample job deck, it has been assumed that a user program named SYSTMU is located on SYSUT2 in the form of absolute column-binary card images that terminate with a transfer card. The job deck is designed to insert the program in the PRESYS library immediately after the IBJOB Processor. Before the program is inserted, it is converted by the system editor into a self-loading, scatter-load format, which is the standard format for the PRESYS library. For the sample job deck in Figure 27, it has been assumed that the user program consists of one record only. However, a program may consist of more than one record. For each additional record, another *INSERT card that specifies TAPE must be placed in the job deck, and the absolute column-binary card images from which the record is formed must follow (on SYSUT2) the card images for the previous record. The column-binary card images for each record must end with a transfer card. Additional information may be found in the section "System Editor."

In designing and coding a program for insertion as a direct-mode subsystem in the PRESYS library, the following rules must be followed to ensure proper loading of the program, coordinated control of input/output operations, and continuous job processing:

1. The program must use only the core storage between SYSORG and SYSEND (refer to SYSCOR in Appendix C).
2. The first word in the first record of the program must be a BCD name without leading blanks and may have an origin at SYSCUR (Appendix C). This name is the name of the program, and it is the name specified on the $EXECUTE card used to call the program. The name must not be the same as the name of any other record in the PRESYS library.
3. The first word of the second and succeeding records, if any, of the program must be a unique BCD name, without leading blanks, and may have an origin at SYSCUR.
4. The first record must contain a TRA instruction, which transfers control to the beginning of the program and which has an origin at SYSTRA (Appendix C).
5. The DC-IBSYS Loader (SYSLDR) in the DC-IBSYS Nucleus may be used by a program to load the second and succeeding records of the program. See Appendix C for a description of the use of the DC-IBSYS Loader. If the DC-IBSYS Loader is used to load a record, the contents of SYSTRA must be changed to transfer control to the first instruction that is to be executed after the record has been loaded. This may be accomplished by placing within the record itself a TRA instruction that has an origin at SYSTRA. The instruction will then overlay the contents of SYSTRA when it is loaded.
6. The Program must recognize and act upon the $IBSYS, $EXECUTE, $STOP, and $ID cards, as described in the section "Job Control Communication with Subsystems.'
7. If a system unit is to be used, it must be referred to through its entry in the system unit function table (refer to the section "DC-IBSYS Nucleus").
8. If a unit that is not assigned to a system unit function is to be used, it must be referred to through an availability chain (refer to the section "DC-IBSYS Nucleus").
9. The program may use IOEX, following the rules for direct-mode input/output discussed in the publication <u>IBM 7090-7040 Direct Couple Operating System: Programmer's Guide</u>, Form C28-6382. If the program does not use IOEX, it must structure its own direct-mode input/output requests in locations THIS and THIS+1, and it must trap the 7040 to service these requests.

INCORPORATING DCMUP UTILITIES

ARRANGEMENT OF DCMUP UTILITY PROGRAMS

DCMUP Utility Programs must be placed on the PRESYS library within the utility file. The first data word of each program must be a BCD name. This name is placed in word 2 of the PRESYS tape record.

The following special record must precede the first DCMUP utility edited onto a PRESYS library:

```
IOCP    address,,3
BCI     1,UTILITY
PZE     0
IOCT    0,,0
```

The address field in the first word may contain any octal address. This record is present in the distributed DCOS PRESYS Library.

Except for the additional requirement of a preface record, DCMUP utilities are maintained and modified in the same manner as DCMUP and DC-IBSYS.

```
1      78        16              31   36
$JOB             ,5,500          INSERT SYSTMU AFTER IBJOB
$SETUP LB2       PRESYS,NORING
$SETUP UT1       NEWPRE
$SETUP UT2       NEWREC,NORING
$IBSYS
$ASSIGN          SYSLB2
$ASSIGN          SYSUT1
$ASSIGN          SYSUT2
$IBEDT
       *EDIT     SYSLB2,MAP,MODS
       *PLACE    SYSTMU,1,1,2         ORDER=2nd
       *MODIFY   IBSYS           INSERT NAME IN DCTBL
FILE   *AFTER    IBJOB
TAPE   *INSERT
.
.        (column-binary card images on SYSUT2)
.
       *INSERT   FILEMK
$EOF    (end of file card)
$IBSYS
$STOP
```

• Figure 27. Sample Job Deck for Inserting a User Program as a Direct Mode IBSYS Subsystem

DESIGN OF DCMUP UTILITY PROGRAMS

To ensure proper functioning of a utility within the framework of the Direct Couple Multiprocessor, the user must adhere to the following rules when designing a DCMUP utility:

1. A utility may not occupy more than 832 core storage locations, starting at location 192_{10} (300_8). The utility is always called by the following instruction.

   ```
   TRA     192
   ```

2. A utility should perform only a small unit of work each time it is called, since it must share 7040 processing time with other DCMUP functions, such as printing, punching, and servicing 7090 requests.
3. The utility must contain the following sequence as its first three instructions:

   ```
   ORG     129
   BCI     1,utility name
   ORG     192
   ```

   (Place name in location 201_8, origin at 300_8.) The utility name must contain 6 characters and may not be blank or zero. If the name is less than 6 characters, it must be right justified and contain leading zeroes (e.g., 00NAME).
4. The utility should practice the following return conventions:
   a. Return control to location 130_{10} (202_8) where control is transferred to the beginning point of the commutator (MAIN).
   b. Return control to location 131_{10} (203_8) when the utility has completed its task.
   c. Return control to location 132_{10} (204_8) to allow normal commutator cycling to proceed.

A communication region starting at 7040 location 133_{10} (205_8) provides access to DCMUP subroutines, input/output facilities, and table-handling routines. The functions specified in this region are called by using DCOS-defined macro-operations. Details about both the contents of the DCMUP communication region and the DCOS-defined macro-operations may be found in the distributed DCOS listing of DCMUP.

INCORPORATING NON-IBSYS SYSTEMS OR PROGRAMS

Non-IBSYS systems or programs may be incorporated into a DCOS PRESYS library. The system to be incorporated must first be blocked into standard DCOS format. The system must be blocked by the DCOS System

Tape Blocker. Two cards must be placed in the card reader following the DCOS System Tape Blocker deck. The first card identifies the system to be blocked. The format of this card is

```
1                  30          31  36       37  42

job identification           sysnam         nfiles
```

where sysnam is the system name (left justified) as it will appear on the $EXECUTE card, and nfiles is the number of files (right-justified) in the system.

The second card is the start card. The start card must be a column-binary card that is identified by 2-7-9 punches in column 73. This card must contain a program that causes the desired system or program to be loaded from simulated 7090 unit A1.

After the system tape has been blocked into standard DCOS format, it must be edited onto a PRESYS library. Figures 28 and 29 show sample job decks for inserting systems that have been blocked by the DCOS System Tape Blocking Routine. Figure 28 illustrates incorporation of the first non-IBSYS system into a PRESYS library. Figure 29 illustrates incorporation of an additional non-IBSYS system.

```
 1      78        16               31
--------------------------------------------------------------------
 $JOB             ,5,500           INSERTING FIRST NON-IBSYS SYSTEM
 $SETUP LB2       PRESYS,NORING    MOUNT PRESYS TAPE
 $SETUP UT1       NEWPRE           MOUNT OUTPUT TAPE
 $SETUP UT2       SYSTMX,NORING    MOUNT BLOCKED SYSTEM X TAPE
 $ASSIGN          SYSLB2
 $ASSIGN          SYSUT1
 $ASSIGN          SYSUT2
 $IBEDT
        *EDIT     SYSLB2,MAP,MODS
        *DUP      SYSLB2,SYSUT4,4
        *AFTER    NONSTD
        *DUP      SYSUT2,SYSUT1, (number of files on system)
 $EOF   (end-of-file card)
 $IBSYS
 $STOP
```

Figure 28. Sample Job Deck for Incorporating the First Non-IBSYS System onto a PRESYS Library Tape

```
 1      78        16             31
 $JOB             ,5,500         INSERTING ANOTHER NON-IBSYS SYSTEM
 $SETUP LB2       PRESYS,NORING  MOUNT PRESYS TAPE
 $SETUP UT1       NEWPRE         MOUNT OUTPUT TAPE
 $SETUP UT2       SYSTMY,NORING  MOUNT BLOCKED SYSTEM Y TAPE
 $ASSIGN          SYSLB2
 $ASSIGN          SYSUT1
 $ASSIGN          SYSUT2
 $IBEDT
       *EDIT      SYSLB2,MAP,MODS
       *DUP       SYSLB2,SYSUT4,4
       *AFTER     NONSTD
       *AFTER     FILEMK
        .          (Include as many *AFTER FILEMK cards as there
        .         are files in the non-standard systems already
        .         on the PRESYS Library)
       *DUP       SYSUT2,SYSUT1, (number of files on system)
 $EOF    (end-of-file card)
 $IBSYS
 $STOP
```

Figure 29. Sample Job Deck for Incorporating an Additional Non-IBSYS System onto a PRESYS Tape

ROLE OF THE DCOS POSTEDITOR

The DCOS Posteditor, written in the 7090/7094 MAP language, is executed in direct mode as a relocatable object program under control of the IBJOB Processor. A DCOS Posteditor binary deck is included in the DCOS master alter/edit deck. Input to the Posteditor is a DCOS PRESYS Library, and its output is a DCOS system (on tape) ready for disk residence. In preparing the system for disk residence, the Posteditor performs the following functions:

1. It eliminates redundant words (caused by instruction overlays) in both 7040 and 7090 system records.
2. It inserts the special direct-read-to-processor bit in all 7090 system record input/output commands. This enables direct transmission to 7090 core storage from a random-access device attached to the 7040.
3. It assigns all system records and data files to specific disk (or drum) track locations.
4. It generates the following tables:
   a. The 7040 system name table, which associates each record or data file name with a specific starting track address.
   b. The utility name table, which associates each 7040 background utility name with a specific track location.
   c. The track allocation and module definition table.
5. It punches a disk load card, which loads DCMUP into 7040 core storage and initiates DCOS processing. DCOS Posteditor is organized in two passes or phases.

PHASE I

If the tape to be edited is a DCOS Distribution Tape, the Posteditor skips to the first PRESYS library file. It reads the PRESYS library, preparing a partial 7040 system name table and a complete utility name table. For each system record encountered, the Posteditor writes a record without input/output commands on the intermediate output unit (SYSUT2). This record is in blocked form, the last block being filled with zeros, and it is followed by a record containing the input/output commands. Input/output commands are analyzed to determine points of overlay.

IBSYS data files and non-IBSYS system records are written on the Posteditor output unit, SYSCK2. When the Posteditor encounters a DCMUP utility record, it enters the utility name in the utility name table and assigns a track location to it. (Track allocation for utilities begins at track 41, module 0, Channel B.) Utility records are written onto the Posteditor output unit, SYSCK2, in final form ready for transcription to disk (or drum).

PHASE II

In Phase II, the DCOS Posteditor completes the 7040 system name table by assigning track locations to each system record, data file, and non-IBSYS system. System records excluding data files do not cross module boundaries.

The Posteditor then reads each record from the intermediate output unit and merges it with the input/output command record that follows. Overlaid words are eliminated, and the input/output commands are consolidated to specify unique contiguous locations in core storage for the system-record words. The compressed system record, with input/output commands, is written on SYSCK2. When all system records have been processed, Phase II writes the following tables on the Posteditor output unit:

1. Utility name table (assigned to track 3, module 0, Channel B.)
2. 7040 system name table (assigned to track 0, module 0, Channel B).
3. Track allocation and module definition table (assigned to track 1, module 0, Channel B.)

Finally, the Posteditor writes a load disk card on SYSPP1.

RESPECIFICATION OF 7040 DISK (DRUM) ENVIRONMENT

The DCSYS file on the DCOS Distribution Tape is allocated to one module (module 0 on channel B). This allocation may be changed by reassembling the DCOS Posteditor and performing a postedit run on the DCOS PRESYS library. Modules are defined by the macro-instruction

```
8        16

MODULE   B((t1)),C((t2)),D((t3)),E((t4))
```

where B, C, D, and E are real 7040 channels and t_1, t_2, t_3, and t_4 indicate the type of modules (i.e., DISK or DRUM). As many as four module type parameters per channel may be specified. Channel B must have at least as many parameters as channel C.

Modules on each channel are numbered consecutively starting with module 0 on channel B and 4 on channel C. For example, in the macro-instruction

```
8        16

MODULE   B((DISK,,DRUM)),C((DRUM,,DISK))
```

the disk module on channel B is module 0, the drum module on channel B is module 2, the drum module on channel C is module 4, and the disk module on channel C is module 6. Drum modules must have even module numbers, and no odd-numbered following module may exist.

Null parameters should be specified only if omission of a parameter would violate a rule regarding the use of the MODULE macro-instruction.

Respecification of modules in DCOS is accomplished using the alter facilities of IBJOB by replacing the MODULE macro-instruction at alter card 100 in the Posteditor. The following is an example of the control cards needed to modify the Posteditor for 2 modules of disk on channel B:

```
1     78        16

$JOB            ,5,
$SETUP LB2      PRESYS,NORING
$SETUP UT1      NEWPRE
$ASSIGN         SYSLB2
$ASSIGN         SYSUT1
$EXECUTE        IBJOB
$IBJOB          NOGO
$IEDIT          SYSLB2,SCHF4,ALTER
$IBMAP PEDIT    3000,M90
       *ALTER   100,100
       MODULE   B((DISK,DISK))
       *ENDAL
$IBSYS
$STOP
```

PREPARATION OF A NEW DCSYS FROM A DCOS PRESYS LIBRARY

This section discusses the procedures for operating the DCOS Posteditor and for obtaining a new, modified Direct Couple Operating System.

1. Prepare a new PRESYS Library if desired.
2. Reassemble the DCOS Posteditor if a respecification of the module configuration is desired.
3. Set up a DCOS Posteditor job for IBJOB as shown in Figure 30. If any card but an end-of-file card is placed after the $DATA card in Figure 30, the Posteditor assumes that a DCOS Distribution Tape has been mounted on SYSCK1. The Posteditor will position the tape forward four files before processing. If no card is placed after the $DATA card, the Posteditor assumes the tape is a PRESYS library only and begins processing at load point of the mounted tape.
4. Run the Posteditor job (Figure 30) under DCOS. The unit assigned to SYSCK2 will be rewound at the completion of the postedit run. It will contain a new DCSYS. A map of the DCOS assignment to disk will be printed.
5. Insert the new DCSYS into a DCOS Distribution Tape using the sample job deck shown in Figure 31.

```
$JOB                            DCOS POSTEDIT RUN
$SETUP CK1      PRESYS,NORING         MOUNT PRESYS INPUT
$SETUP CK2      DCSYS                 MOUNT TAPE FOR OUTPUT
$ASSIGN         SYSCK1                CONNECT TO
$ASSIGN         SYSCK2                MOUNTED TAPES
$EXECUTE        IBJOB
$IBJOB          GO
       .        (DCOS Posteditor Program in symbolic
       .        (IBMAP) or binary (IBLDR) form.)
       .
$DATA (end-of-file card)
$EOF
```

•Figure 30. Sample Postedit Run

```
$JOB                            PLACE NEW DCSYS INTO DCSDT
$SETUP LB2      DCSDT,NORING          MOUNT DCOS DIST. TAPE
$SETUP UT1      Y00014                MOUNT TAPE FOR NEW DCOS DIST. TAPE
$SETUP CK1      DCSYS,NORING          MOUNT NEW DCSYS (POSTEDIT OUTPUT)
$ASSIGN         SYSLB2
$ASSIGN         SYSUT1
$ASSIGN         SYSCK1
       *EDIT    SYSLB2
       *DUP     SYSLB2,SYSUT1,1       KEEP FIRST DCSDT FILE
       *DUP     SYSLB2,SYSUT4,1       DISCARD OLD DCSYS
       *DUP     SYSCK1,SYSUT1,1       INSERT NEW DCSYS
       *DUP     SYSLB2,SYSUT1,2       DUP. MISC. AND PREST CARDS
       *REMARK     IBEDT WILL DUP PRESYS AUTOMATICALLY
$EOF  (end-of-file card)
```

•Figure 31. Sample Job Deck for Inserting a New DCSYS (Postedit Output) into a DCOS Distribution Tape

POSTEDITOR OUTPUT

7040 SYSTEM NAME TABLE

The 7040 system name table (Figure 32) is written on the DCSYS file and assigned to track 0, module 0 of the disk (drum). All 7040 system name table entries are two words. Most entries in the table associate a record or data file name with a specific starting track address. The format of these system name table entries is

```
BCI     1,record name
pfx     track address
```

If pfx is PZE, the record is a system record in the standard DCOS scatter-load format. If pfx is MZE, the record is a data record (or a non-IBSYS system record). In addition to the normal record name entry, two other entry types may appear in the system name table. An entry of

```
PZE     0
PZE     0
```

signifies an end of file on the PRESYS

tape. This information is used by the Multiprocessor in tape-positioning simulation. An entry of

```
        OCT     777777777777
        OCT     777777777777
```

signifies that the following entries in the 7040 system name table do not refer to IBSYS system records. These entries are made after all IBSYS system record entries are made and before each nonstandard system entry.

Word 1	AXC size of table,1
2	PZE 0,,0
	up to 230 two-word entries describing track allocation of system records
	BCI 1,*EOT
	PZE 0,,0

Figure 32. 7040 System Name Table

TRACK ALLOCATION AND MODULE DEFINITION TABLE

The track-allocation and module-definition table (Figure 33) contains information concerning the existence and availability of modules of disk (or drum). It is created by the Posteditor using information contained in the MODULE macro-instruction.

The first 16 words of the track-allocation and module-definition table are reserved for module definition entries of the form

```
        pfx     basei,,lasti
```

The prefix (pfx) of each entry indicates whether or not the module is available for dynamic track allocation. If pfx is PZE, the module exists and is available for allocation. If pfx is MZE, the module does not exist. If pfx is SIX, the module has been totally allocated for system use. The address and tag portion of each entry contains an 18-bit base track address (basei). The address of module 0 is zero, the base address of module 1 is 10000, the base address of module 2 is 20000, and so forth. The decrement portion of each entry contains the highest valid relative track address in the module (lasti). For 1301 modules, lasti is 9999; for 7320 modules, lasti is 0399.

Word 1 … 16	Module-Definition Words
17	AXT n,4
18	Track-Availability Bits

•Figure 33. Track-Allocation and Module-Definition Table

Although 16 words are reserved for module definition, the number of real entries in the table varies. One entry is made for each module (DISK, DRUM, or null) specified in the MODULE macro-instruction. The first modules specified on each channel become entries 1 through n; the second modules specified on each channel become entries n+1 through m, and so forth. Remaining words in the module-definition portion of the table are filled with zeros.

Word 17 of the track-allocation and module-definition table contains

```
        AXT     n,4
```

where n is the number of real entries in the table.

The rest of the table consists of a master bit map which contains 10000/n availabiility bits. N is a parameter in DCMUP which can be set to 1,2,4 or 8 at the user's option. Each bit represents the status of a specific group of n tracks. An availability bit equal to 1 indicates that the particular group of tracks is not in use. A bit equal to zero indicates that the parameter group of tracks is in use.

The number of words making up the master bit map depends upon n as follows:

n	Words	
1	278	(8 bits unused in first word)
2	139	(4 bits unused in first word)
4	70	(20 bits unused in first word)
8	35	(10 bits unused in first word)

For a detailed explanation of bit maps see the section "DCMUP Track Allocation Maps".

The track-allocation and module-definition table is assigned to track 1, module 0, Channel B.

UTILITY NAME TABLE

The utility name table, shown in Figure 34, associates each 7040 DCMUP utility with a specific track address. The Posteditor writes the utility name table on SYSCK2 and assigns it to track 3, module 0, Channel B, of the disk (or drum).

Word 1	AXC	size of table,2
2	PZE	0
	BCI	1,utility name 1
	PZE	track address,,0
	BCI	1,utility name 2
	PZE	track address,,0
	.	.
	.	.
	.	.
	BCI	1,utility name n
	PZE	track address,,0

•Figure 34. Utility Name Table

POSTEDITOR OUTPUT TAPE (DCSYS) FORMAT

The format of all DCMUP, DC-IBSYS, IBSYS subsystem, and DCMUP utility records written on the Posteditor output unit is shown in Figure 35. Word 1 contains an 18-bit binary track address. When this binary address is converted to BCD, it yields the six-digit track address mmtttt, where

mm = module

and

tttt = track (0000-9999)

The remaining 460 words contain several series of instructions. Each series is preceded by an input/output command of the form

```
IOCP    addr,ctl,word count
```

Addr is the address into which the following instructions are to be loaded. Word count is the number of instructions to be loaded. If ctl is 4, the record is to be loaded into 7040 core storage; if ctl is 5, the record is to be loaded into 7090 core storage.

The input/output command

```
IOCT    0,,0
```

indicates the end of a logical record. The remainder of the physical record contains unmeaningful data.

The format of data file records written on the Posteditor output unit is shown in Figure 36. Words 1 and 2 contain the binary track addresses assigned to the data record. Word 3 contains the binary track addresses of the previous record and the next data record. The remaining 458 words contain a variable number of logical data records, each preceded by a control word.

The contents of each data-record control word are interpreted in the same way as the contents of a logical-record control word. Logical-record control words are described in the section "Data Buffer Format."

```
        ┌──────────┬───────────────────┐
Word 1  │          │ Track Address     │
        ├──────────┴───────────────────┤
        │                              │
        │    Input/Output Command      │
        │              .               │
        │              .               │
        │              .               │
        │        Instructions          │
        │              .               │
        │              .               │
        │              .               │
        │    Input/Output Command      │
        │              .               │
        │              .               │
        │              .               │
        │        Instructions          │
        │              .               │
        │              .               │
        │              .               │
 461    │                              │
        └──────────────────────────────┘
```

Figure 35. Posteditor Output: Format of DCMUP, DC-IBSYS, IBSYS Subsystem, and DCMUP Utility Records

```
        ┌───────────────┬──────────────┐
Word 1  │               │Track Address │
        ├───────────────┼──────────────┤
   2    │               │Track Address │
        ├───────────────┼──────────────┤
        │   Previous    │    Next      │
   3    │ Track Address │Track address │
        ├───────────────┴──────────────┤
        │                              │
        │     Logical Data Records     │
        │            with              │
        │        Control Words         │
        │                              │
 461    │                              │
        └──────────────────────────────┘
```

Figure 36. Posteditor Output: Data Record Format

POSTEDITOR RESTRICTIONS AND ERROR MESSAGES

When using the Posteditor to prepare a system tape for disk residence, certain rules regarding record size, record contents, and number of records must be observed. The Posteditor is assembled for a maximum record size of 17,400 words. This limit may be changed by changing the address portion of the parameter word. This word is:

```
BUFSZ EQU 17400
```

If the Posteditor encounters a record of greater length than specified in BUFSZ, it prints the following message:

IN THE RECORD SEQUENCE - $name_1$ $name_2$ $name_3$ LAST RECORD TOO LARGE FOR POSTEDIT POSTEDIT TERMINATED

BUFSZ is limited, in practice, by the amount of space in 7090 core storage available to the Posteditor during the postediting process.

No record may contain more than 250 memory loading sequences (before overlays have been eliminated). The message

```
TOO MANY I/O COMMANDS IN REC.    xxxxx
POSTEDIT TERMINATED
```

is printed if too many loading sequences are encountered. The PRESYS library tape may contain no more than 223 system records, data files, and EOF marks. If additional records are present, the Posteditor notes the error condition with the following message

```
TOO MANY SYSTEM RECORDS OR DATA FILES
POSTEDIT TERMINATED
```

The Posteditor also checks that the first record of the PRESYS library is DCMUP. If it is not DCMUP, the message

```
DCMUP NOT FIRST RECORD OF PRESYS
POSTEDIT TERMINATED
```

is printed.

Two Posteditor messages indicate redundancy conditions. These are:

1. PERMANENT REDUNDANCY ON PRESYS FILE POSTEDIT TERMINATED (redundancy while reading from the Posteditor input unit, SYSCK1)
2. PERMANENT REDUNDANCY ON POSTED INTER POSTEDIT TERMINATED (redundancy while reading from the Posteditor intermediate input/output unit, SYSUT2)

Note that all Posteditor error conditions described above result in the termination of postedit processing.

The following is an in-depth discussion of the unit assignment procedures present in DCOS. Four basic precepts guide the entire philosophy.

1. Every reel of tape to be processed by a 7090 program run in the DCOS environment must have a $SETUP card in the job deck, so that the tape reel(s) may be mounted before the job reaches the 7090.
2. In the absence of a $SETUP card, all 7090 input/output units are simulated on a 7040 Disk (Drum) Storage Unit(s). The number of units that may be simultaneously simulated is limited only by the number of available buffers in 7040 core storage.
3. The assignment of 7040 tape units for 7090 input/output tapes is entirely under control of DCMUP. Within a job, an attempt is made to balance channel usage for multiple tape unit assignments. $SETUP cards specifying 7090 channels A or C will cause DCMUP to attempt tape assignments to 7040 channel B. If 7090 channels B or D are specified on $SETUP cards, 7040 tapes on channel C will be assigned if available. If there are no tapes available on the desired 7040 channel at the time of setup, no attempt will be made to wait until one is available. Instead any available tape will be assigned. Because the selection of 7040 tape units is not predictable, all tape mounting is to be performed only under direction of messages typed by DCMUP.
4. IBJOB users need never refer to any 7090 physical tape address. There is a complete symbolic unit assignment language in DCOS.

The first section of the following discussion is concerned with unit assignment for IBJOB programs. This section presents the basic relationships between 7090 unit control blocks and the 7040 input/output procedures that handle the data flow. Later sections include discussions of the unit assignment conventions used in compatibility mode, when actual 7090 physical input/output unit addresses (e.g., 1206, 1321) are obtained by the 7040 using the special DCOS machine facility for transmitting input/output instructions from the 7090 to the 7040.

7040 FILE CONTROL BLOCKS

DCMUP contains 35 file control blocks, each of which is eight words long. Each file control block is used by DCMUP to hold information required to identify distinct 7090 data files; to identify the 7040 input/output device used to read or write that data; to point to 7040 core storage buffers used to hold the data during the buffering process between 7040 input/output devices and 7090 core storage; and to indicate the state of the logical DCMUP processes that are involved (e.g., number of buffers in use, whether data is being read in, etc.). DCMUP is continuously informed of the relationship of each file control block to a specific 7090 unit control block; in this sense, a 7040 file control block may be considered an extension of a particular 7090 unit control block. In addition, DC-IOEX in the 7090 is kept informed of the relationship of 7090 unit control blocks to 7040 file control blocks. This relationship is displayed in Figure 37.

DC-IOEX uses the <u>7040 file control block number</u> as the unit specification in all direct-mode input/output calls issued by the 7090.

In the absence of $SETUP cards, DCMUP simulates each 7090 unit control block on disk (drum) storage unit(s). This is specified in the associated file control block by the absence of a reference to a 7040 tape unit control block. DCMUP is further aware of the system unit function (SYSUNI) table assignments, as indicated to the right of the 7090 unit control block number in Figure 37. These relationships are recorded in IOBASE number zero (0), which is included in DCMUP. The specified relationships cause DCMUP to:

1. Place into 7040 file control block 1 the track address of the data file that was created when the job entered DCOS.
2. Write the data directed to file control blocks 2 and 3 on the disk, and enter task descriptions into the job description block specifying printing and punching, respectively, for those data files.
3. Place into 7040 file control block 32 the track address of the data file(s), if any, created from cards read into the 1402 and enclosed by $ROW and $ENDROW control cards.

4. Effect IBM card code to BCD conversions on the data directed to 7040 file control blocks 33 and 34, and enter task descriptions into the job description block specifying punch or print, respectively.
5. Interpret all requests upon file control block 0 as requests for 7090 systems loading, and, as a result, initiate procedures to locate the desired record in the 7040 system name table, seek to its location, and initiate transmission to 7090 core storage.

7090 UCB No.		7090 Unit Address Contained	7040 FCB No.
1	(CRD)	1321 (RDA)	32
2	(PCH)	1341 (PUA)	33
3	(PRT)	1361 (PRA)	34
4	(LB1)	A1	0
5	(IN1)	A2	1
6	(OU1)	A3	2
7	(PP1)	A4	3
8	(CK1)	A5	4
9		A6	5
10		A7	6
11		A8	7
12		A9	8
13		A0	9
14	(UT1)	B1	10
15	(UT2)	B2	11
16	(UT3)	B3	12
17	(UT4)	B4	13
18	(CK2)	B5	14
19		B6	15
20		B7	16
21		B8	17
22		B9	18
23		B0	19
24		C1	20
25		C2	21
26		C3	22
27		C4	23
28		C5	24
29		C6	25
30		D1	26
31		D2	27
32		D3	28
33		D4	29
34		D5	30
35		D6	31

Figure 37. Relationship of 7090 Unit Control Blocks to 7040 File Control Blocks

ASSIGNMENT OF 7040 TAPE UNITS TO 7040 FILE CONTROL BLOCKS

If $SETUP cards are included with a job and if they specify that a reel of tape is to be processed, the setup routine in DCMUP selects a 7040 tape unit (unit control block) by the following criteria:

1. If a tape unit is needed for intermediate use (i.e., no specific tape reel need be mounted), a search is initiated for an available 7040 unit control block that specifies that the tape is mounted and ready (i.e., tape is in rewind position). If none are available, item 2 applies.
2. If a specific reel of tape is to be mounted, a search <u>for an available 7040 unit control block</u> specifying that the tape unit is rewound and unloaded is initiated.
3. In the search for an available 7040 control block, the selection of a 7040 tape unit assigned to a unit on 7090 channels A or C will be conducted in ascending order from B1-B0, C1-C0, D1-D0, E1-E0. The search in the selection of a 7040 unit assigned to a unit on 7090 channels B or D will proceed in ascending order from C1-C0, D1-D0, E1-E0, B1-B0.

An available 7040 unit control block is recognized by a plus sign in the first word of the block. When a unit control block is selected, the job number of the job undergoing setup is placed into the assigned field in word 3 and the sign of word 1 is set minus.

A complementary discussion of 7040 unit control block selection is included in the section "Job Setup."

Selection of the 7040 file control block (hence the 7090 unit control block) to be connected to the selected 7040 unit control block is under control of the DCMUP setup routines. The sign position of word seven of each file control block (see Figure 41) specifies which 7040 file control blocks are already assigned (i.e., have standard usages as defined in the 7090 SYSUNI table). Therefore, selection of a file control block, for example, S(1), is made by choosing <u>any</u> unassigned file control block and marking it assigned. The selection criteria is simply to choose the next available block starting with file control block 5 and continuing to 9, skipping 10-13 (SYSUT1-4) and continuing with 14-31. Refer to the discussion in the section "Unit Assignment in Other IBSYS Subsystems" for selection of file control blocks for physical 7090 unit addresses with IBSYS subsystems other than IBJOB. In summary, DCMUP has chosen 7040 file control blocks, committing 7090 unit assignment processes to adhere to the choices.

COMMUNICATION OF 7040 FILE CONTROL BLOCK--7090 UNIT CONTROL BLOCK RELATIONSHIP

Since a specific 7040 file control block is selected for setup by the 7040 and since the 7040 file control block to 7090 unit control block relationship is fixed, it is necessary to provide the 7090 with the file control block assignment that was made for each 7090 unit symbol specified on a $SETUP card.

For example, the $SETUP cards

```
1       8       16

$SETUP  A(2)    R671
$SETUP  S(3)    TAPE
$SETUP  J(1)    R202
```

when processed by the 7040 setup processor might have yielded the following selection:

7090 Unit Symbol	7040 FCB	7040 UCB
A(2)	5	C2
S(3)	6	D0
J(1)	7	C3

where D0 was chosen for S(3) because D0 was at rewind position and ready whereas C2 and C3 were not.

The DCMUP setup routine generates a table, known as the unit symbols table (UNISYSM), that contains the 7090 unit symbols and the 7040 file control block indexes. In our example, UNISYM would contain the following information:

7040 FCB	Unit Symbol (in BCD)
5	A(2)
6	S(3)
7	J(1)

DC-IBSUP issues an HEY instruction with a type 16 code to request transmission of the UNISYM table to the 7090 each time IBSYS gets control (i.e., at the beginning of a job and between job segments). The unit symbols are then distributed by DC-IBSUP to the fourth word of the appropriate 7090 unit control blocks, and the unit control blocks are removed from the 7090 unit availability chain. In our example, DC-IBSUP would set:

4th Word of 7090 UCB No.	Set To
A6	BCI 1,00A(2)
A7	BCI 1,00S(3)

where A6 and A7 correspond to 7040 control blocks 5 and 6, as shown in Figure 37.

The 7090 unit control block for A8 would be set up as an intersystem reserve unit, since J(1) was specified. Hence the following relationships have been established:

7090 Unit Symbol	7040 UCB	7040 UCB
A(2)	A6	C2
S(3)	A7	D0
J(1)	A8	C3

When DC-IBSUP is processing $ASSIGN cards, it need only search for a match between the fourth word of a 7090 unit control block and the system unit specification on the $ASSIGN card. When a match is found, DC-IBSUP places the location of the matching unit control block into the specified SYSUNI entry. Refer to the $ASSIGN description under "Unit Assignment Control Cards" for further information.

IBLDR UNIT ASSIGNMENT PROCESS

The IBLDR unit assignment process is replaced with a special DC process when operating IBJOB under DCOS. Basically, the new process involves symbol matching between unit assignment symbols on $FILE cards presented to IBLDR and the fourth word of 7090 unit control blocks. When a match is found, the selected 7090 unit control block is assigned to the file.

If an intersystem reserve unit is undergoing assignment, the standard IBSYS reserve unit notation is checked in the first word of all 7090 unit control blocks to find a match. If a $FILE card with a file name of UNITxx is encountered (e.g., a FORTRAN IV file), the characters UNIT are stripped off the file name, leaving xx to be used when searching the fourth word of the 7090 unit control blocks. A $SETUP card, such as

```
1       8       16

$SETUP  07      R621
```

will cause a 7040 unit control block and a 7040 file control block to be assigned. The unit symbol 07 will be passed to the 7090 by the DC-IBSUP UNISYM load call, then stored into the specified 7090 unit control block, and finally selected by IBLDR by matching the fourth word of a 7090 unit control block with the $FILE UNIT07 file card. Details of the IBLDR process are presented in Figure 38.

A $SETUP card specifying a 7090 system unit SYSxxx is valid. The processes used are illustrated in the following examples. $SETUP cards specifying SYSUNI functions apply only to object-time execution unless a $ASSIGN card is recognized by DC-IBSUP.

If DCMUP had encountered

```
1       8       16

$SETUP  UT1     R627
```

but there was no

```
1              16

$ASSIGN        SYSUT1
```

the DCMUP setup routine selects a 7040 file control block that is not assigned. Since file control block 10 is assigned as system unit UT1, it will not be selected for the UT1 specified on the $SETUP card. The setup routines do not take any special action on unit symbols except compatibility 7090 real-unit address references (e.g., A1, B6). Therefore, file control block 16 might be selected for the unit symbol UT1. When DC-IBSYS is distributing unit symbols, it places BCI 1,000UT1 in the fourth word of 7090 unit control block B7, and IBLDR selects that block for assignment to the file requesting SYSUT1. If DC-IBSUP had encountered

```
1              16

$ASSIGN        SYSUT1
```

before IBJOB received control, DC-IBSUP would have selected B7 as the unit control block for the SYSUNI entry, and IBJOB and the object program would use unit control block B7. Because of this dual-definition facility for system unit symbols, object-program tapes for files specifying SYSUNI assignments can be mounted prior to the 7090 processing phase without IBJOB using that reel of tape.

ASSIGNMENT OF SECONDARY UNITS

If the unit assigned as the primary unit is a system unit, an intersystem reserve unit, or a unique symbolic unit, and if the required secondary unit is also one of these, the secondary unit is assigned in the same manner as the primary unit. If these conditions are not met, the request for a secondary unit is ignored.

A system unit is defined as a unit with a name that matches the last three characters of an entry in the SYSUNI table (e.g., UT1, IN1, etc.). Unique symbolic units are units that are specified as

A(1),A(2),...,H(9)

and

S(1),S(2),...,Z(9)

Units referred to as other in Figure 38 are units that are specified as:

A,B,...,H

and

S,T,...,Z

If a real channel or a symbolic channel is specified with no unit number or if no unit is specified, any available unit is assigned. Intersystem reserve units (J through R) may not be specified without a unit number.

UNIT ASSIGNMENT IN OTHER IBSYS SUBSYSTEMS

Since each 7090 IBSYS Subsystem adheres to the SYSUNI and intersystem reserve unit philosophy, job segments of IBSYS subsystems other than IBJOB provide complete symbolic assignments consistent with the 7040 symbolic unit symbol process if only 7090 system unit functions (SYSxxx) or intersystem reserve units are specified on $SETUP cards. All IBSYS job segments must use IOBASE number 0.

IBSYS subsystems that perform unit assignment out of the unit availability chains may cause some inconvenience to the programmer if he is not able to predict which unit will be chosen. If a unit to be set up is not a SYSUNI or intersystem reserve unit, the 7090 physical tape address (e.g., A8) of the unit control block to be assigned must be used on a $SETUP card. For example, an IBSYS subsystem that selected the first unit in the availability chain of 7090 channel B would choose B6 (B1-B5 are SYSUT1-UT4 and SYSCK2) if no other $SETUP or $ASSIGN cards were present. Therefore, the user would request setup for his file by a

```
1      8       16

$SETUP B6      R206
```

control card.

The DCMUP setup routines process all physical reference unit types first, marking the 7040 file control block unavailable for symbolic unit symbol selection. The physical unit symbol is then placed into the UNISYM table.

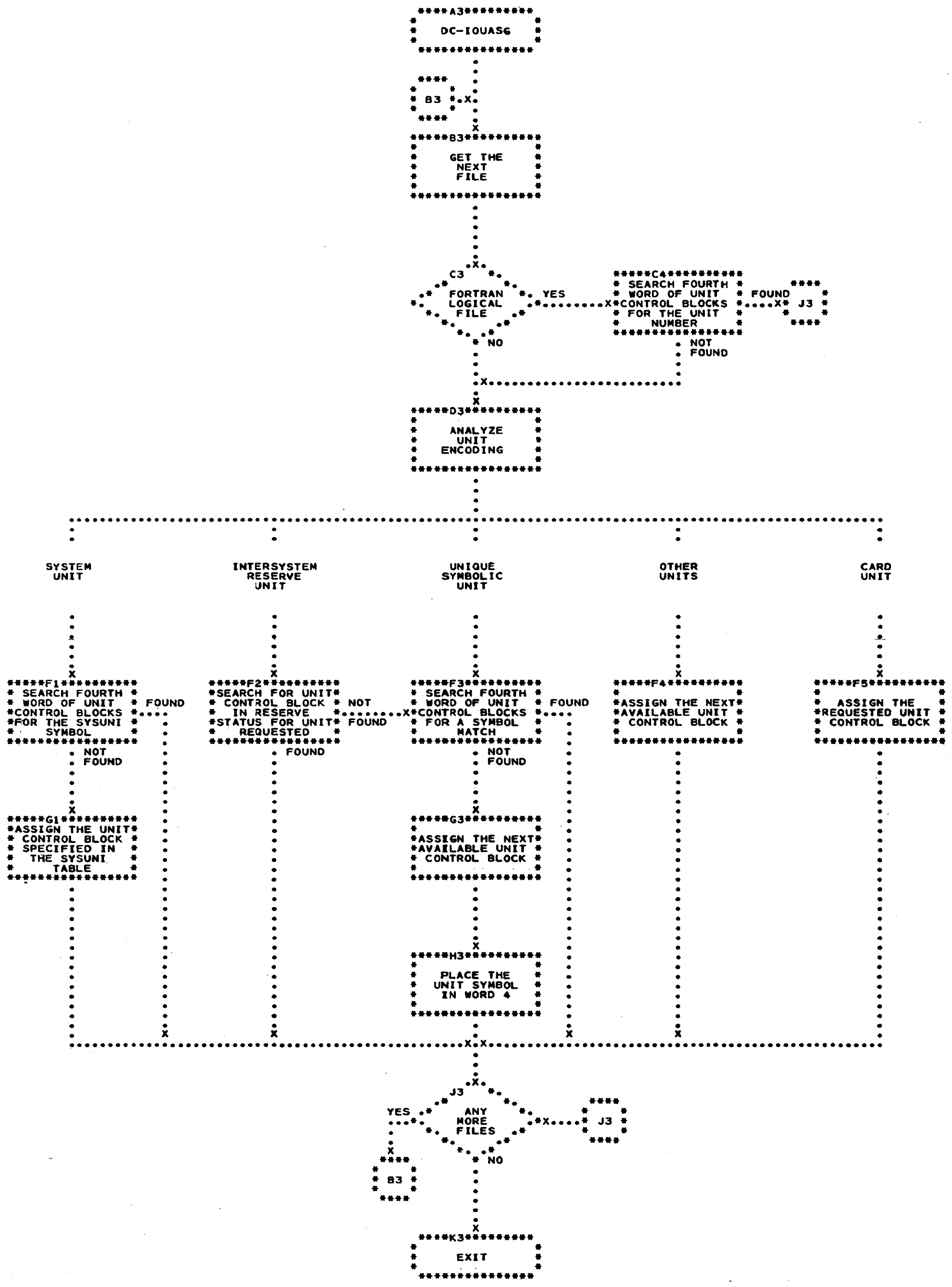

A3
DC-IOUASG
B3
B3
GET THE NEXT FILE
C3
FORTRAN LOGICAL FILE
YES
NO
C4
SEARCH FOURTH WORD OF UNIT CONTROL BLOCKS FOR THE UNIT NUMBER
FOUND
J3
NOT FOUND
D3
ANALYZE UNIT ENCODING
SYSTEM UNIT
INTERSYSTEM RESERVE UNIT
UNIQUE SYMBOLIC UNIT
OTHER UNITS
CARD UNIT
F1
SEARCH FOURTH WORD OF UNIT CONTROL BLOCKS FOR THE SYSUNI SYMBOL
FOUND
NOT FOUND
F2
SEARCH FOR UNIT CONTROL BLOCK IN RESERVE STATUS FOR UNIT REQUESTED
NOT FOUND
FOUND
F3
SEARCH FOURTH WORD OF UNIT CONTROL BLOCKS FOR A SYMBOL MATCH
FOUND
NOT FOUND
F4
ASSIGN THE NEXT AVAILABLE UNIT CONTROL BLOCK
F5
ASSIGN THE REQUESTED UNIT CONTROL BLOCK
G1
ASSIGN THE UNIT CONTROL BLOCK SPECIFIED IN THE SYSUNI TABLE
G3
ASSIGN THE NEXT AVAILABLE UNIT CONTROL BLOCK
H3
PLACE THE UNIT SYMBOL IN WORD 4
J3
ANY MORE FILES
YES
NO
J3
B3
K3
EXIT

Figure 38. IBLDR Unit Assignment Under DCOS

When DC-IBSYS is distributing symbols to unit control blocks, it inserts physical unit symbols (e.g., BCI 1,000A6) into the fourth word of the selected unit control blocks. Since each such unit control block will not be removed from the availability chain of the channel, a non-IBJOB unit-assignment process will find that unit control block available. All symbolic unit symbols cause the selected unit control block to be removed from the availability chain, making it unavailable except to DC-IBSYS or IBJOB. Further, DC-IBSUP and IBLDR will not select a unit out of the availability chain that has a nonzero fourth word. Therefore, units set up for physical address referencing cannot be assigned to any IBSYS or IBJOB symbolic reference.

UNIT ASSIGNMENT PROCEDURES FOR NON-IBSYS SYSTEMS/PROGRAMS

The major differences between assignments for non-IBSYS and IBSYS job segments are:

1. IOBASE 0 may not be used if:
   a. 7090 physical unit address A1 (1201 or 1221) is not that of system residence unit. The system must reside on the 7040 disk and have been introduced into the DCOS system via placement on the DCOS PRESYS files.
   b. 7090 units A2, A3, and A4 are not used from the disk as the system input, output, and punch functions, respectively.
2. IOBASE 1 is provided for general use. However, IOBASE 1 has no specific relationship assigned for system residence, input, output, or punch; therefore a $SETUP card set as follows must be used:

```
1      8    16

$SETUP unit DISK,INPUT  if job input is
                        placed after
                        $EXECUTE card

$SETUP unit DISK,PRINT  to get listing
                        performed on
                        1403 printer
                        of 7040

$SETUP unit DISK,PUNCH  to get cards
                        punched on 1402
                        of 7040

$SETUP A1   ident       if $EXECUTE
                        TAPE is used
```

In the above cards, unit must be a physical 7090 unit address symbol (e.g., A3, B6, etc.). IOBASE 1 specifies the relationship of the data file created by $ROW-$ENDROW card groups in the job input through the 1402 and the 7090 physical unit symbol RDA (7090 select address 1321). Similarly, the relationship of PUA (1341) and PRA (1361) to the required IBM card-code conversion routines is specified. IOBASE 1 allows a 7090 configuration as described in Figure 39.

The selection of 7040 unit control blocks for setup is the same as for IBSYS jobs. The selection of the 7040 file control block that refers to the selected 7040 unit control block is governed by the IOBASE table.

7040 FCB No.	7090 Physical Address
0-3	no correspondence
4	A1 (1201, 1221)
5	A2 (1202, 1222)
6	A3 (1203, 1223)
7	A4 (1204, 1224)
8	A5 (1205, 1225)
9	A6 (1206, 1226)
10	A7 (1207, 1227)
11	A8 (1210, 1230)
12	A9 (1211, 1231)
13	A0 (1212, 1232)
14	B1 (2201, 2221)
15	B2 (2202, 2222)
16	B3 (2203, 2223)
17	B4 (2204, 2224)
18	B5 (2205, 2225)
19	B6 (2206, 2226)
20	B7 (2207, 2227)
21	B8 (2210, 2230)
22	B9 (2211, 2231)
23	B0 (2212, 2232)
24	C1 (3201, 3221)
25	C2 (3202, 3222)
26	C3 (3203, 3223)
27	C4 (3204, 3224)
28	D1 (4201, 4221)
29	D2 (4202, 4222)
30	D3 (4203, 4223)
31	D4 (4204, 4224)
32	RDA (1321)
33	PUA (1341)
34	PRA (1361)

Figure 39. IOBASE 1 Unit Configuration

UNIT ASSIGNMENT CONSIDERATIONS

The following special considerations must be handled by installation personnel.

1. 7090 messages referencing 7090 physi-

cal unit addresses or symbols (e.g., 1201, A1) can only be correlated to 7040 tape units by knowledge of the IOBASE being used and the mounting messages typed at setup time. Further, 7090 units being simulated on disk have no corresponding 7040 tape unit.
2. Normally 7090 IBJOB messages, including deferred tape mounting messages, are not considered operator pertinent because they are not specified as such in the MWR calling sequence. 7040 sense switch 2 controls whether 7090 messages are typed on the 7040 typewriter or 1401.

Operational inconsistencies may arise out of the above two considerations due to the DCOS assumption that all tape mounting has been accomplished before the job reaches the 7090.

APPENDIX A: DCMUP TABLES AND CONTROL BLOCKS

The format and contents of the job queue table, job description blocks, file control blocks, 7040 unit control blocks, and channel information blocks are described in this appendix. The table and blocks are generated by and maintained by DCMUP. Figures depicting the formats of the control blocks follow the descriptions of the contents of the blocks.

THE JOB QUEUE TABLE

S 3	4 12	13	14 17	18 35
PRI	JOB No.	A	STG	Module and Track Address

Bits S-3 contain the low-order four bits of the six bits representing the BCD character in column 16 of the $JOB card.

Bits 4-12 contain a DCOS-assigned job sequence number to be used for job status inquiry and job logging.

Bit 13 contains a one-bit activity indicator, which is 0 if the job is not active in the specified stage and 1 if it is.

Bits 14-17 contain a numeric representation of the next or current processing stage of the job. If bit 13 is 0, bits 14-17 represent the next stage. If Bit 13 is 1, bits 14-17 represent the current stage.

Bits 18-35 contain an 18-bit binary number, which, when converted to BCD, represents the module and track address in the form mmtttt where:

mm is the logical module number (00-15)

and

tttt is track address (0000-9999)

The module range (00-15) does not mean that DCOS permits 16 modules. The maximum number for DCOS is ten modules.

JOB DESCRIPTION BLOCK

The contents of the job description block are shown in Figure 40 and are interpreted as follows.

Word 1

Job number (bits 4-12) is an internally generated number used for job control during processing.

Track (bits 18-35) is the track address of the job description block.

Word 2

Word 2 must contain all zeros.

Word 3

Time entered system (entire word) is the time that the job was read into the system, as reflected by the 7040 clock.

Word 4

Time on the 7090 (entire word) is the time that the 7090 started processing the job, as reflected by the 7040 clock.

Word 5

Time off the 7090 (entire word) is the time that the job was either completed or discontinued on the 7090, as reflected by the 7040 clock.

Word 6

Number of cards read (bits 3-17) is a count of the cards read during processing of the job.

Number of cards punched (bits 21-35) is a count of the cards punched during processing of the job.

Word 7

Number of lines printed (bits 3-17) is a count of the lines printed while processing the job.

Number of tapes used (bits 21-35) is the number of tape units assigned to process the job.

Word 8

Setup task description (bits 18-35) is the relative location in the job description block of the setup task description.

Word 9

IOBASE number (bits 3-17) is the number from column 16 of a $IOBASE card.

IOBASE task description (bits 18-35) is the relative location in the job description block of the IOBASE task description. The IOBASE task description contains the address of a track that, in turn, contains a set of file control blocks fully set up for 7090 execution.

Word 10

If a $ATEND card was processed for this job, the options specified in the card are stored in word 10.

Pfx (bits S-2) governs the dump format.

Limit 1 (bits 3-17) is the location of the first word in the 7090 to be dumped.

D (bit 18) is 0 if the DUMP option was not specified on the control card.

D is 1 if the DUMP option was specified on the control card. The portion of 7090 core storage specified by limit 1 and limit 2 is dumped regardless of what caused job termination (normal end of job or job discontinued for any reason).

P (bit 19) is 0 if the POST option was not specified on the control card.

P is 1 if the POST option was specified on the control card. The IBJOB Debugging Postprocessor will be called if the 7090 terminates the job for any reason other than normal end of job.

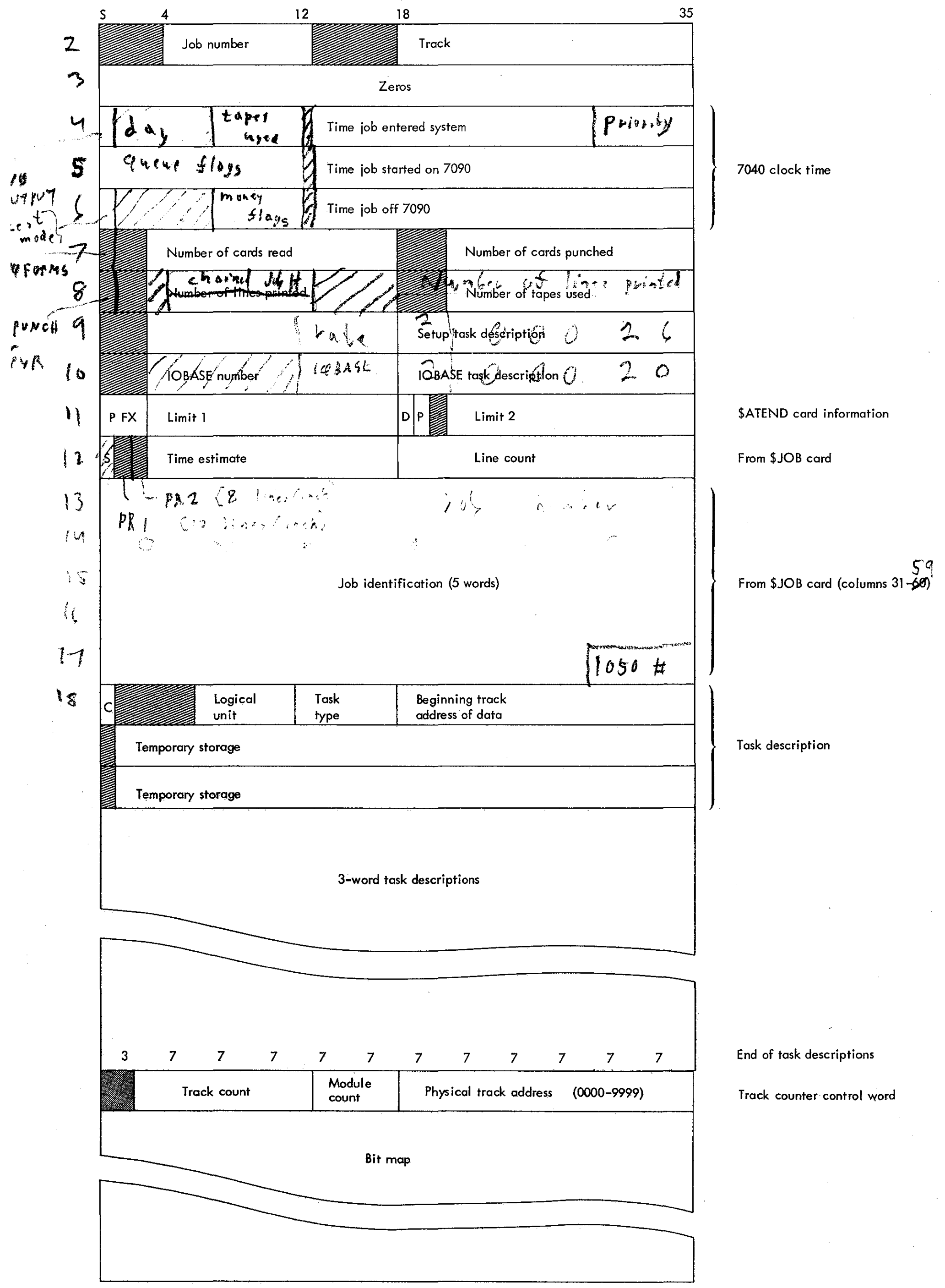
Limit 2 (bits 21-35) is the location of the last word in the 7090 to be dumped.

Word 11

Time estimate (bits 3-17) is the estimated job time in minutes, as specified in the $JOB card.

Line count (bits 18-35) is the estimated output line count, as specified in the $JOB card.

If either the time estimate or the line count is exceeded, the job will be terminated.



● Figure 40. Job Description Block

Task Description Entries

A task description entry is inserted into the job description block for every file written on the disk. Certain entries such as SETUP, IOBASE, INPUT, and accounting tasks will always be present.

The sign bit of the first word of a task description entry is used to indicate whether the task has been completed. If the sign bit is minus, the task is complete. If the sign bit is plus, the task has either been queued or is in process.

The first word of the task description also contains the logical unit (bits 6-11), the task type (bits 12-17), and the address of the first track containing data associated with the job (bits 18-35). The following is a description of the task types:

	Task Description
1	reserved
2	Input data for 7090
3	Punch (721 images)
4	Printer (716 images)
5	Breakdown disk-to-tape
6	7090 unit symbol table
7	Disk-to-printer
8	Disk-to-punch
9	Purge
10	Reserved
11	Input data for setup
12	IOBASE for this job
13	Input data for utilities

Peripheral Track Bit Map

At the end of each job description block there is a group of words called the peripheral track bit map, whose bits represent the disk tracks containing the input/output information for the job. For detailed explanation of bit maps see the following section, "DCMUP Track Allocation Maps."

DCMUP TRACK ALLOCATION MAPS

DCMUP track allocation is handled by three types of bit maps.

Peripheral Bit Map

At the end of each job description block is a peripheral bit map which may contain a record of the disk tracks holding the job's input stream (prior to 7090 execution), and output information (during and after 7090 execution) that is to be punched or printed. The first word of the peripheral bit map is a track counter control word used by the track allocation subroutine to issue tracks (within a specific group) on disk, and to issue a new group of tracks when the current one is exhausted. (Tracks are issued in order, from the track with the lowest number, across all modules, to the track with the highest number, across all modules.) The remainder of the bit map consists of a sufficient number of words to contain 10,000/n bits, where n is the number of tracks in a group and is a parameter in DCMUP that may be set to 1,2,4, or 8 at the user's option. Each bit represents a specific group of n tracks being used by the job. When a bit is one, the group of tracks associated with it is being used by the job.

The following table shows the size of a peripheral bit map and the maximum number of task entries that can appear in a job description block, at a given time, for permissible values of n.

n	Size of Map	Number of Tasks
1	279 words	53
2	140 words	100
4	71 words	123
8	36 words	135

Scratch Bit Map

There is a scratch bit map for the job currently being run on the 7090. When a job goes into 7090 execution the input information in the peripheral bit map of the job description block is transferred to the scratch bit map. During 7090 execution, tracks containing intermediate files will be recorded in the scratch bit map. When 7090 execution is complete, all tracks recorded in the scratch bit map are purged (returned to unused status and made available for use by other jobs).

Master Bit Map

The master bit map controls the use by the peripheral and scratch bit maps of the specific groups of track used for input/output purposes. If a bit is one, the group of tracks associated with it is available for use by the peripheral or scratch bit maps. If a bit is zero, the tracks are already in use (the corresponding bit in the scratch bit map <u>or</u> in only <u>one</u> of the peripheral bit maps will be one, indicating issuance of the tracks to that bit map.)

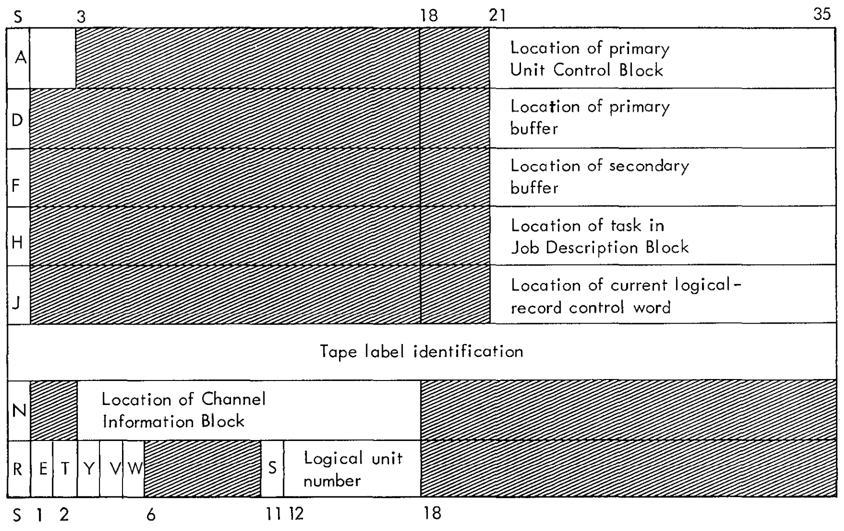

S
3
18
21
35
A
Location of primary Unit Control Block
D
Location of primary buffer
F
Location of secondary buffer
H
Location of task in Job Description Block
J
Location of current logical-record control word
Tape label identification
N
Location of Channel Information Block
R
E
T
Y
V
W
S
Logical unit number
S
1
2
6
11
12
18

Figure 41. File Control Block

FILE CONTROL BLOCK

The contents of the 7040 file control block are shown in Figure 41 and are interpreted as follows.

Word 1

If A (bit S) is 0, the file is open.

If A is 1, the file is closed.

Location of unit control block (bits 21-35) is the address of the unit control block that is associated with the unit attached to the file.

Word 2

If D (bit S) is 0, a primary buffer has been assigned to the file.

If D is 1, a primary buffer has not been assigned to the file.

Location of the primary buffer (bits 21-35) is the address of the primary buffer assigned to the file.

Word 3

If F (bit S) is 0, a secondary buffer has been assigned to the file.

If F is 1, a secondary buffer has not been assigned to the file.

Location of the secondary buffer (bits 21-35) is the address of the secondary buffer that has been assigned to the file.

Word 4

If H (bit S) is 0, the input/output unit assigned to the file is disk and the file is associated with the 1402 reader, the 1402 punch, or a 1403 printer.

If H is 1, the input/output unit assigned to the file is either the disk or a tape unit.

Location of task in job description block (bits 21-35) is the address of the current or next task to be performed on the file. If a task is in process, the location is that of the current task description.

Word 5

If J (bit S) is 0, the input/output unit assigned to the file is a 7040 tape unit.

If J is 1, the input/output unit assigned to the file is the disk.

Location of the current logical-record control word (bits 21-35) is the address, in the buffer, of the logical-record control word associated with the next logical record to be transmitted to or from the 7090.

Word 6

Word 6 contains the BCD representation of the six-character reel identification as it was specified in one of the option fields of a $SETUP card.

Word 7

If N (bit S) is 0, the file control block is available for use by the setup routine.

If N is 1, the file control block is not available for use by the setup routine.

Location of the channel information block (bits 3-17) is used for compatibility-mode operation. Each channel referred to by the 7090 in compatibility mode has a two-word channel information block in the 7040. The address of the block is stored here.

Word 8

If R (bit S) is 0, the assigned input/output unit is not the card reader or the card punch.

If R is 1, the assigned input/output unit is either the card reader or the card punch.

If E (bit 1) is 0, the unit assigned is an input unit.

If E is 1, the unit assigned is an output unit.

If T (bit 2) is 0, data is not currently being read into the primary buffer.

If T is 1, data is currently being read into the current buffer.

If Y (bit 3) is 0, data is not currently being read into the secondary buffer.

If Y is 1, data is currently being read into the secondary buffer.

If V (bit 4) is 0, the primary buffer does not contain a physical record.

If V is 1, the primary buffer contains a physical record.

If W (bit 5) is 0, the secondary buffer does not contain a physical record.

If W is 1, the secondary buffer contains a physical record.

If S (bit 11) is 0, the assigned input/output unit has not been set up.

If S is 1, the assigned input/output unit has been set up.

Logical unit number (bits 12-17) is the relative position (00-34) of this file control block within the set of all file control blocks.

UNIT CONTROL BLOCKS (7040 TAPE UNITS)

The contents of the 7040 unit control block are shown in Figure 42 and interpreted as follows.

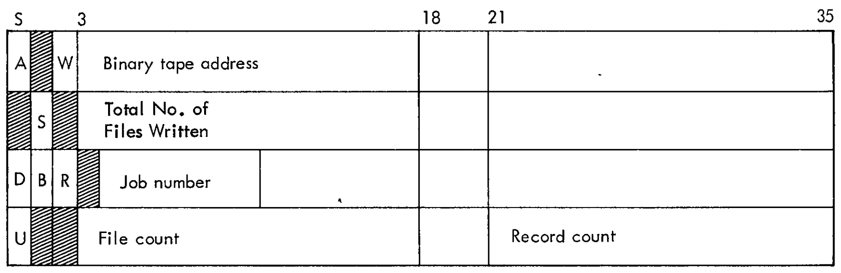


Figure 42. 7040 Unit Control Block (Tape Interface)

Word 1

If A (bit S) is 0, the unit is available for use by the system.

If A is 1, the unit is not available (i.e., it is currently in use for some job).

If W (bit 2) is 0, there are no operations waiting to be performed on the unit.

If W is 1, there is an operation waiting to be performed on the unit.

Binary tape address (bits 3-17) is the binary mode address of the unit (e.g., 2221 for 729 tape unit B1).

Word 2

If S (bit 1) is 0, the unit is currently being used within an active setup, breakdown, or utility routine.

Total number of files written gives the total number of files written on an output tape.

Word 3

If D (Bit S) is 0, the unit is to be rewound and unloaded when processing is complete.

If D is 1, the unit is to be rewound when processing is complete.

If B (Bit 1) is 0, the unit does not contain input to BTPTP.

If B is 1, the unit contains input to the BTPTP utility.

If R (Bit 2) is 0, no rewind-unload operation is waiting execution.

If R is 1, a rewind-unload operation is waiting for execution.

Job number (bits 4-12) is an internally generated number used to determine to which job the unit has been assigned.

Word 4

If U (bit S) is 0, the unit is ready for input/output operations.

If U is 1, the unit is rewound and unloaded.

File count (bits 3-17) is the number of file marks written on or read from the tape.

Record count (bits 21-35) is the number of records that have been written on or read from the tape.

CHANNEL INFORMATION BLOCK

The contents of the channel information block are shown in Figure 43 and are interpreted as follows.

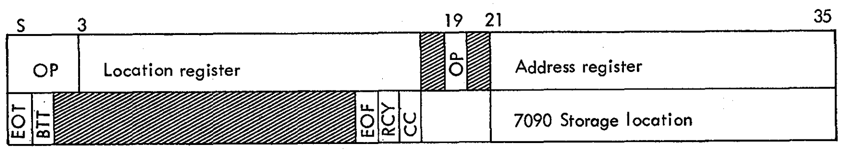


Figure 43. Channel Information Block

Word 1

Op (bits S-2 and 19) is the operation code of the last input/output command simulated for the channel.

Location register (bits 3-17) is the location in the 7090 following the last input/output command that was simulated for the channel.

Address register (bits 21-35) is the location in the 7090 following the last data word referred to by the last input/output command simulated for the channel.

Word 2

If EOT (bit S) is 0, no end-of-tape condition has occurred on the channel.

If EOT is 1, an end-of-tape condition has occurred while writing on the channel.

If BTT (bit 1) is 0, the beginning-of-tape reflective spot has not been sensed or simulated on the channel.

If BTT is 1, the beginning-of-tape reflective spot has either been sensed or simulated during the simulation of a BSR, BSF, or REW operation on the channel.

If EOF (bit 15) is 0, end of file has not occurred on the channel.

If EOF is 1, a data channel trap has been requested as a result of an end-of-file condition on this channel.

If RCY (bit 16) is 0, a read/write redundancy has not occurred on the channel.

If RCY is 1, a data channel trap has been requested as a result of a read/write redundancy on the channel.

If CC (bit 17) is 0, a channel command has not requested a trap on the channel.

If CC is 1, a data channel trap has been requested on the channel as a result of an IOxT instruction being simulated without an LCH waiting.

7090 storage location (bits 21-35) is the location in the 7090 that is to be used for storing the channel condition bits when trapping.

APPENDIX B: PROCEDURE FOR DEFINING AN IOBASE TABLE

An IOBASE table may be defined using the macro-instructions discussed below. These macro-instructions allow the user to define an IOBASE table to DCOS for systems that operate in the compatibility mode and have unit assignments that differ from DC-IBSYS. The skeletons of these macro-operations are shown in the Macro Definitions section of the DCOS program listing.

ZERO POINTERS (ZERPTR) MACRO-INSTRUCTION

The zero pointers macro-instruction must be used when defining an IOBASE. It resets to zero the pointers specified in its argument field. The general form of the instruction is:

```
ZERPTR (ch1,ch2,...,AV,AV1)
```

where ch1 and ch2 are simulated 7090 channels (A, B, etc.) and Av and Av1 are the two words in the IOBASE table that indicate which file control blocks are available. For example, the macro-instruction:

```
ZERPTR (A,B,C,AV,AV1)
```

resets to zero the pointers for simulated 7090 channels A, B, and C and both available file-control-blocks words.

FILE CONTROL BLOCK POINTERS (FBPTRS) MACRO-INSTRUCTION

The file control block pointers macro-instruction assigns file control blocks to simulated 7090 units. This macro-instruction must contain one parameter for each file control block. If a file control block is to be assigned to a simulated 7090 unit, the parameter is the unit designation, e.g., (A3), (B4), (C6). If the file control block is to be set up as unassigned, the parameter contains (AVAIL,i) where i is the desired position of the file control block in relation to the other unassigned file control blocks, e.g., (AVAIL,1) makes the next file control block the first unassigned one, (AVAIL,2) makes the next file control block the second unassigned one, etc. A maximum of 12 file control blocks may be unassigned.

For example, the macro-instruction:

```
FBPTRS  ((A1),(A2),(A3),(B4),
        (AVAIL,1),(AVAIL,2),(B1),
        (B2),(B3),(A4))
```

would assign the first four file control blocks to units (A1), (A2), (A3), and (B4), make the next two file control blocks the first two unassigned ones, and assign the next four file control blocks to units (B1), (B2), (B3), and (A4).

The parameters may be specified in any sequence; but it must be kept in mind that the first parameter is assigned to the first file control block, the second parameter to the second file control block, etc.

INPUT/OUTPUT UNIT CONFIGURATION (IOBASE) MACRO-INSTRUCTION

The purpose of this macro is to define the 7090 input/output unit configuration that is to be simulated for the compatibility-mode system involved. This macro-instruction is also used to assign the IOBASE its number. Up to six IOBASE configurations may be defined. All six may be retained in core storage for use by systems or programs operating in the compatibility mode.

The general form of the IOBASE macro-instruction is:

```
IOBASE n,((channel,number))
```

where n is the IOBASE number (0, 1, 2, ..., 5), channel is the simulated 7090 channel (A, B, C, etc.), and number is the number of input/output units simulated on the channel. If the units to be simulated are the on-line reader, printer, and punch, the channel designation is AUR for channel A, BUR for channel B, etc. For example:

```
IOBASE 3,((AUR,3),(A,10),
          (B,10),(C,6))
```

defines IOBASE 3 as simulating a card reader, a card punch, a printer, and ten tape units on channel A, ten tape units on channel B, and six tape units on channel C.

These macro-instructions are entered into the DC system by the IBJOB alter technique.

$IOBASE CARD

The $IOBASE card specifies which IOBASE is to be used for processing the current job. The format of the card is:

```
1              16
$IOBASE        n
```

where n is the IOBASE number assigned when the IOBASE configuration was defined.

APPENDIX C: DC-IBNUC COMMUNICATION REGION ENTRIES

The functions of the entries in the communication region of the DC-IBSYS Nucleus are described in this appendix.

SYSTRA is loaded with a transfer instruction when the first record of a subsystem is scatter-loaded into core storage following the reading of a $EXECUTE card that specified an IBSYS subsystem. After loading the first record of a subsystem, the DC-IBSYS Loader transfers control to this entry, which, in turn, transfers control to the beginning of the subsystem. A transfer instruction may also be loaded when succeeding records of a subsystem are loaded.

SYSDAT contains a six-character word, containing the data specified on the last $DATE card processed by DCMUP. The entry is initialized at the beginning of each job. The date word is provided for use in headings and labels by subsystems and object programs.

SYSCUR contains the BCD name of the subsystem or subsystem record currently in core storage. The DC-IBSYS Supervisor places the subsystem name from the $EXECUTE card in this location before loading the first record of the subsystem. When succeeding records of a subsystem are loaded, the name of the record may be loaded into this location.

SYSRET is the location to which each subsystem transfers control to call in the DC-IBSYS Supervisor. After the Supervisor is in core storage, the instruction EMTM is executed. Therefore, the machine is always in multiple-tag mode when control is passed to a subsystem from the DC-IBSYS Supervisor.

SYSKEY is not used in DCOS.

SYSSWS is not used in DCOS.

SYSPOS contains the position on SYSLB1 of the subsystem currently in core storage. This information is entered into SYSPOS by the DC-IBSYS Supervisor after it determines from its system name table the position of a subsystem specified on the $EXECUTE card. The format of the entry is:

```
        PZE     index,,nfiles
```

where index is always 1, corresponding to SYSLB1, and nfiles is the number of files preceding the system called.

SYSUNI contains the first location (in the address portion) and length (in the decrement portion) of the system unit function table.

SYSUBC contains the first location (in the address portion) and length (in the decrement portion) of the unit control block table. This table consists of one word for each channel, and the word contains the following information:

Prefix	Number of card units assigned to the channel.
Decrement	Total number of units assigned to the channel.
Address	Address of the first unit control block for the channel.

SYSUAV contains the first location (in the address portion) and length (in the decrement portion) of the unit availability table. This table is described in the section "Unit Availability Table."

SYSUCW contains the first location (in the address portion) and the combined length (in the decrement portion) of all unit control blocks. The unit control blocks are generated in contiguous locations.

SYSRPT contains the instruction

```
        TRA     1,4
```

SYSDMP contains an instruction that transfers control to a bootstrap routine for calling DCMUP to save 7090 core storage and to load the System Core Storage Dump Program. A transfer to SYSDMP initiates job termination procedures in accordance with the options specified on the $ATEND card. See the publication IBM 7090-7040 Direct Couple Operating System: Programmer's Guide for additional information on the dump program. The 7040 FCB 13 (7090 unit B4 in IOBASE 0) is used for the dump spill.

SYSIOX contains the first location (in the address portion) and length (in the decrement portion) of the DC-IOEX communication table (Figure 9). This table contains entries for transferring control to DC-IOEX subroutines.

SYSIDR is provided for transferring control to an installation accounting routine. Whenever a $ID or $JOB card is processed by a subsystem or the DC-IBSYS Monitor, control is transferred to SYSIDR by the sequence

```
        TSX     SYSIDR,4
        PZE     L($ID)
        return
```

where L($ID) is the location of the first word of the buffer containing the $ID or $JOB card in BCD form. The distributed version of DCOS does not contain an installation accounting routine.

SYSCOR contains the limits of the core storage area available for use by subsystems operating under control of the DC-IBSYS Monitor. The format of SYSCOR is:

```
        PZE     SYSEND,,SYSORG
```

In the distributed version of the DC-IBSYS Monitor, SYSEND and SYSORG are defined as follows:

SYSORG = 2652_8 or 1450_{10}

SYSEND = 77777_8 or 32767_{10}

The FORTRAN II Assembly Program (FAP) does not refer to the SYSCOR entry when defining the two symbols SYSORG and SYSEND. These symbols are defined by FORTRAN II as follows:

SYSORG = 3720_8 or 2000_{10}

SYSEND = 77777 or 32767

The user may redefine SYSEND to allow space for an installation accounting routine in upper core storage. In this case, the following limits apply: for FORTRAN II, SYSEND may not be lower than 77677 or 32703 ; for all other subsystems, SYSEND may not be lower than 77013 or 32267 .

SYSLDR contains an instruction that transfers control to the DC-IBSYS Loader. The DC-IBSYS Loader may be used to scatter-load subsystem records that have the same standard PRESYS library record format as the first record of the subsystem. When loading has been completed, the DC-IBSYS Loader transfers control to SYSTRA. Therefore, the contents of SYSTRA must be modified during or before loading.

Whenever the first record of a subsystem is loaded into core storage after reading a $EXECUTE card, the DC-IBSYS Supervisor places in the decrement portion of SYSLDR the location of the unit control block for the unit from which the subsystem is being loaded. Subsequent transfers to SYSLDR by the subsystem will result in the next sequential records being loaded from the unit indicated in the decrement portion of SYSLDR.

A subsystem may not specify in the decrement of SYSLDR the unit from which the next record is to be loaded, since all system records are on SYSLB1. The DC-IBSYS Loader always uses direct-mode input/output conventions.

The DC-IBSYS Loader may be entered using the instruction

```
     TSX      SYSLDR,4
```

The user must position SYSLB1 in front of the record that he wishes to load.

SYSACC is used for communication between the installation accounting routine, and the subsystems and the DC-IBSYS Monitor. Whenever a $ID or $JOB card is processed by a subsystem or by the DC-IBSYS Monitor, control is transferred to SYSIDR, and SYSACC is tested upon return. If SYSACC contains all zeros, the subsystem or the DC-IBSYS Monitor lists the $ID or $JOB card on the system output unit before transferring control to SYSIDR. In the case of a $JOB card, a page is ejected before the card is listed. If the contents of SYSACC are nonzero, the $ID or $JOB card is not listed by the subsystem or DC-IBSYS Monitor. The installation accounting routine is provided with the location of the first word of the buffer containing the $ID or $JOB card (as described in the section "SYSIDR") and must list the card if SYSACC is set to nonzero at an installation.

SYSPID is reserved for use in communication between an installation accounting routine, when one is incorporated into DC-IBSYS, and the subsystems and the DC-IBSYS Monitor. The exact use of this location depends on the design of the accounting routine.

SYSCYD and SYSCYD+1 are not used in DCOS.

SYSSLD, SYSTCH, and SYSTCH+1 contain the following input/output command sequence.

```
     SYSSLD   **+1,,1 (IOCP)
     SYSTCH   *+1
              *-2
```

The sequence may not be used to read SYSLB1.

SYSTWT is not used with DCOS.

SYSGET contains a word that indicates to the DC-IBSYS Supervisor the reason that control was returned to it by a subsystem. Additional information on SYSGET appears in the section "DC-IBSYS Nucleus."

SYSJOB contains a control word that is used by the DC-IBSYS Supervisor and subsystems in controlling the skipping of jobs and job segments. The section "DC-IBSYS Nucleus" contains additional information on SYSJOB.

| .CHEXI is the direct-couple environment location. An indirect zero test of this location determines whether the system is operating under direct couple. A successful zero test indicates direct couple.

| .MODSW indicates to IOEX whether the next record is in BCD or binary mode. If the contents are zero, binary mode is indicated; if the contents are nonzero, BCD mode is indicated. (Operational in direct mode only.)

APPENDIX D: CREATION OF A DCOS PRESYS LIBRARY FROM A 7090/7094 IBSYS OPERATING SYSTEM LIBRARY

A DCOS PRESYS Library may be created from the latest 7090/7094 IBSYS Operating System tape and the distributed DCOS symbolic tape, but this procedure should never be necessary. It is described here in general terms to provide an approach to be taken if special requirements arise and to exemplify the close relationship between a DCOS PRESYS Library and a 7090/7094 IBSYS

Operating System tape. The following procedure is not necessarily applicable to every IBSYS system tape and is meant only to serve as a guide.

CASE 1: PREPARATION ON A 7090/7094 SYSTEM

1. Assemble DCOS symbolic deck.
2. Prepare an IBSYS system editor deck to do the following:
   a. Edit SYSLB2, using *PLACE cards to specify IBSYS subsystem order.
   b. Delete the IBSYS LOAD button record (label=5U0004).
   c. Insert in its place the 7040 DCMUP binary deck.
   d. Replace 7090/7094 IBSYS with DC-IBSYS.
   e. Modify IBJOB with DCOS IBJOB patch decks.
   f. Modify IBEDT with DCOS system editor patch decks.
   g. Delete any IBSYS subsystems not desired.
   h. Insert binary decks of DCMUP utilities <u>after</u> the last IBSYS subsystem and IBEDT.
   i. Insert a 3 word record named NONSTD.
3. Perform an IBSYS edit run of the 7090/7094 IBSYS Operating System tape on SYSLB2, obtaining new output on SYSUT1.
4. Run the prepared output from step 3 through the 7040/44 IBJOB Tape Blocking Utility Routine on the 7040. Output from the blocking routine will be a DCOS PRESYS Library.

CASE 2: PREPARATION ON A 7090-7040 DIRECT COUPLE SYSTEM

1. Assemble DCOS symbolic deck.
2. Run the 7090/7094 IBSYS Operating System tape through the DC support package tape blocker.
3. Same as case 1, step 2, except for the insertion of appropriate $SETUP and $ASSIGN cards for SYSLB2 and SYSUT1.
4. Same as case 1, step 3. Output on SYSUT1 will be the DCOS PRESYS Library.

<u>APPENDIX E: REDUNDANCY CONDITIONS, TAPE DENSITY, MODE, AND PARITY IN DCOS</u>

The following seven points are presented to clarify the methods used by DCOS for handling tape density, input/output mode, and input/output and core storage error conditions.

1. Tape density is controlled only through the 7040 console density switches. The 7090 instructions SDNH and SDNL are ignored.
2. Input to the 7090 is in mixed mode and contains look-ahead bits supplied by the 7040.
3. The mode of all 7090 output is indicated in the logical-record control words. If a 7090 job attempts to read, in one mode, a record it has written in the other, a read error condition will be sent to the 7090. Attempts by the 7090 to reread the record in the same mode will be ineffective.
4. If the 7040 Setup routine encounters a redundancy while reading an input tape, it will attempt to read the record in the opposite mode. Ten reads are attempted in each mode. After 20 unsuccessful reads, the Setup routine will inhibit the processing of the job.
5. The breakdown utility routines write mixed-mode records with no look-ahead bits unless the look-ahead bits are in the record created by the 7090. The mode of each record is dictated by the mode bit in the logical-record control word.
6. A 7040 core storage parity error causes a full DCOS stop. No recovery procedures are attempted. The system should be restarted according to the procedure discussed in the publication <u>IBM 7090-7040 Direct Couple Operating System: Operator's Guide</u>, Form C28-6384.

<u>APPENDIX F: DCOS DEBUGGING AIDS PROGRAM</u>

The DCOS debugging aids program consists of a routine for inserting patches to DCMUP, a 7040 core storage snapshot routine (SNAP), and a DC-IBSYS patch routine. The snapshot routine gives the systems programmer a tool to assist him in debugging program failures resulting from a problem in DCMUP. The DC-IBSYS patch routine gives the systems programmer a tool for inserting temporary patches into DC-IBSYS. The entire debugging aids program (Snap-Patch deck) is contained in a binary card deck distributed with DCOS. Figure 44 shows the card deck setup that must be used to load and to use the debugging aids program.

DCMUP PATCHES

Patches to DCMUP may be inserted after DCMUP has been loaded. The patches are inserted by placing patch cards in front of the TRA 77 card (Figure 44). The TRA 77 card is a binary transfer card to location 77_8. The format of the patch cards is the same as the editor octal alteration cards with the exception that only one patch is permitted per card.

The format of the card is:

```
1       8    16-27

octloc  OCT  patch
```

DC-IBSYS PATCH ROUTINE

The DC-IBSYS patch routine inserts temporary patches into the 7090 after each loading of the DC-IBSYS Monitor. The patches will be effective until DCMUP is reloaded into the 7040 without the patch routine and the patches.

The format of the patch cards is basically the same as that of the system editor octal alteration cards. The difference is that each instruction in the patch cards must contain 12 octal digits. The format of the patch card is

```
1       8      16      29      42      55

octloc  OCT    word 1,word 2,word 3,word 4
```

The octal words must be separated by commas. A blank terminates the variable field. The octal words are loaded into sequential 7090 core storage locations beginning with the location specified by the octal address octloc.

The DC-IBSYS patch cards must be placed between the TRA 77 card and the 00300 card as shown in Figure 44. The 00300 card is an H-code card with 00300 in columns 1-5 and columns 6-80 blank.

A card reader error (e.g., reader check), encountered while reading patches to the system or to the snapshot debugging request loader, will cause a system stop, probably at 7040 core location 270 . The user must then reload the start deck.

7040 CORE STORAGE SNAPSHOT ROUTINE (SNAP)

The SNAP routine gives the systems programmer a method of dumping selected sections of 7040 core storage. When the SNAP routine is used, the dumps are printed on the 1403 attached to interface 3 (printer 1).

The section of 7040 core storage that is to be dumped is specified in the SNAP card. The format of the SNAP card is

```
1       11

octloc  snap limits
```

A dump of the section of 7040 core storage specified by snap limits is printed prior to executing the instruction at octal location octloc. The number of dumps printed is determined by the number of snap limits specified. The snap limits are specified by two five-character octal locations. The first five characters specify the first location of the limit; the second five characters specify the last location of the limit. Four limits can be specified on one card in columns 11-50. Columns 11-20 specify the limits of the first dump; columns 21-30 specify the limits of the second dump; etc.

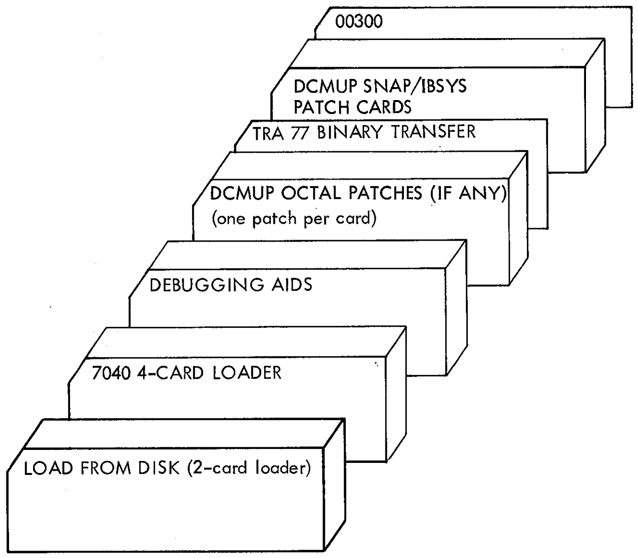
If the snap limit is to be specified by a word in the 7040, an indirect snap can be specified. The indirect snap is specified by punching an asterisk (*) in the first column of the snap limits field (11, 21, 31, and 41). The next four columns of the field must contain the number of locations (in octal) to be dumped. The last five columns of the field must contain the octal location of a word in the 7040 that contains in its address the location of the first word to be dumped.

The SNAP cards must be placed between the TRA 77 card and the 00300 card as shown in Figure 44. To temporarily suspend SNAP dumping, put sense switch 4 on. To resume, put sense switch 4 off.

APPENDIX G: CONVERSION OF IBSYS SYSTEMS TO DIRECT MODE

The following rules and conventions must be observed in converting IBSYS systems and programs to direct mode.

1. No input/output instructions may be executed in direct mode. For systems or programs that use DC-IOEX, changing all RDS and WRS Instructions to NOP instructions will ordinarily satisfy


00300
DCMUP SNAP/IBSYS PATCH CARDS
TRA 77 BINARY TRANSFER
DCMUP OCTAL PATCHES (IF ANY)
(one patch per card)
DEBUGGING AIDS
7040 4-CARD LOADER
LOAD FROM DISK (2-card loader)

● Figure 44. Deck Setup For Using Debugging Aids Program

this requirement. Modification of the select routine instruction

```
XEC*    .RCHX
```

is not required, since the instruction executed will be NOP.

2. Binary is the ordinary input/output mode of direct-mode DCOS. The contents of location MODSW must be changed to zero if BCD input/output is desired. A read in the wrong mode is considered as a permanent redundancy; a record on a mixed-mode file should be read in the proper mode.
3. All writing on SYSOU must be unblocked print lines. Blocked printing may be sent to the 7040 with a type 14 code if the blocked data is separated by logical-record control words, as discussed in the section "Data Buffer Format."
4. All writing on SYSPP must be unblocked card images to be punched.
5. Sense switch tests and enter-keys instructions should be preceded by messages to the operator used in conjunction with 7090 pause calls (HEY instructions with an 11 code) to ensure operator response.
6. Existing Message-Writer calls result in off-line printing only. If an on-line message is desired, the prefix of word 2 in the calling sequence to .PROUT must be changed to one of the following.

   PTW
   MTW
   PTH
   MTH

   See the publication <u>IBM 7090-7094 Direct Couple Operating System: Programmer's Guide</u> for a complete description of Message Writer.
7. Care must be taken to ensure the compatibility of unit assignment provisions in the system/program with the unit assignment conventions of DCOS.
8. The DCOS operator cannot communicate with a system/program via RDA (the 711 Card Reader). However, if control card requirements are known, the control cards may be identified by $ROW and $ENDROW cards and placed in the 1402 with the rest of the job.

APPENDIX H: GETBUF AND PUTBUF AS SYSTEM MACROS OR SUBROUTINES

GETBUF and PUTBUF are macros in DCMUP whose function is to get and release buffers. In the system as presently distributed these macros generate calls to subroutines. They may be reassembled to use the subroutines in-line by making the SET pseudo-operation BFSUB = 0. It is presently set to 1. Keeping GETBUF and PUTBUF in subroutine form, however, saves about one buffer of core storage.

This is a master index for all three Direct Couple Operating System guides. References are keyed to the individual publications by the following codes:

PG = Programmer's Guide, Form C28-6382

SPG = Systems Programmer's Guide, Form C28-6383

OG = Operator's Guide, Form C28-6384

191)

IBM Technical Newsletter

File Number	7090-36
Re: Form No.	C28-6383-2
This Newsletter No.	N28-0158-0
Date	June 11, 1965
Previous Newsletter Nos.	None

IBM 7090-7040 Direct Couple Operating System: Systems Programmer's Guide

This Technical Newsletter amends the publication IBM 7090-7040 Direct Couple Operating System: Systems Programmer's Guide, Form C28-6383-2. The pages attached to this newsletter document the use of the IBJOB Processor and the System Editor of the Version 13 IBSYS Operating System under the Direct Couple system.

In the referenced publication, replace the pages listed below with the corresponding pages attached to this newsletter.

Cover and Preface
Pages 1 through 7
Pages 9 through 28
Pages 37 and 38
Pages 41 and 42
Pages 45 and 48
Pages 53 and 54
Pages 61 and 62
Pages 69 through 76
Pages 79 through 84
Pages 87 through 90

A vertical line to the left of the column shows where text has been changed; new or revised illustrations are denoted by the symbol to the left of the caption.

File this Newsletter at the back of the manual. It will provide a reference to changes, a method of determining that all amendments have been received, and a check for determining if the manual contains the proper pages.

IBM Corporation, Programming Systems Publications, Dept. D39, 1271 Avenue of the Americas, New York, N.Y. 10020

PRINTED IN U.S.A.

IBM Technical Newsletter

File Number	7090-36
Re: Form No.	C28-6383-2
This Newsletter No.	N28-0175-0
Date	December 7, 1965
Previous Newsletter Nos.	N28-0158-0

IBM 7090-7040 Direct Couple Operating System: Systems Programmer's Guide

This technical newsletter amends the publication IBM 7090-7040 Direct Couple Operating System: Systems Programmer's Guide, Form C28-6383-2.

In the publication, replace the pages listed below with the corresponding pages attached to this newsletter.

Pages 5 and 6	Pages 57 through 60
Pages 9 and 10	Pages 61.2 and 62
Pages 19 through 20	Pages 69 through 74.1
Pages 27 and 28	Pages 85 and 86
Pages 43 through 54	Pages 89 through 100

A vertical line to the left of a column shows where text has been changed. A large dot to the left of a figure caption shows that the figure has been changed.

File this newsletter at the back of the manual. It will provide a reference to changes, a method of determining that all amendments have been received, and a check for determining if the manual contains the proper pages.

IBM Corporation, Programming Systems Publications, Dept. D39, 1271 Avenue of the Americas. New York, N.Y. 10020

PRINTED IN U.S.A.

CUT ALONG LINE

COMMENT SHEET

IBM 7090–7040 DIRECT COUPLE OPERATING SYSTEM
SYSTEMS PROGRAMMER'S GUIDE

FORM C28-6383-2

FROM

NAME ______________________________

OFFICE NO. ______________________________

FOLD FOLD

CHECK ONE OF THE COMMENTS AND EXPLAIN IN THE SPACE PROVIDED

☐ SUGGESTED ADDITION (PAGE)

☐ SUGGESTED DELETION (PAGE)

☐ ERROR (PAGE)

EXPLANATION

FOLD FOLD

NO POSTAGE NECESSARY IF MAILED IN U.S.A.
FOLD ON TWO LINES, STAPLE, AND MAIL

STAPLE

FOLD FOLD

FIRST CLASS
PERMIT NO. 81

POUGHKEEPSIE, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY

IBM CORPORATION
P.O. BOX 390
POUGHKEEPSIE, N.Y.

ATTN: PROGRAMMING SYSTEMS PUBLICATIONS,
DEPARTMENT D39

FOLD FOLD

CUT ALONG LINE

STAPLE