USERS MANUAL

1401 OPERATING SYSTEM VERSION 2

UWMCC

COMPUTER CENTER

FORWARD

This manual is intended as a guide for programmers and users of the 1401 Operating System, Version II.

All the 47 routines documented here are available to the user as of July 1, 1968. The documents are self-sufficient as far as possible. Where several routines are derived from the same source and are identical in pattern, they have been combined in one writeup, as in the case of the Disk Input/Output macros (DSKIØ/IØGET/IØPUT/IØSK). In the interests of clarity, the basic format is identical for all writeups.

This manual is divided into two parts, for users' convenience. The first section consists of mainline programs. The second is entirely AUTØCØDER macro-instructions. Standards halts and messages appear in Appendix A.

Please direct any queries concerning the details of program operation to the

Systems Division
UWM Computer Center
Room 28
Mitchell Hall

1401 OPERATING SYSTEM

VERSION 2

Table of Contents

TITLE:	MØNITR
MACHINE:	IBM 1401
LANGUAGE:	AUTØCØDER
SUPERVISOR PROGRAM:	none
PROGRAMMER:	Systems Division
DATE COMPLETED:	July 1, 1968

PURPOSE:

MØNITR, the 1401 Operating System inter-job supervisor, provides the following services and facilities:

a) Logging of time and other parameters at the beginning and end-of-jobs.

b) Maintenance of the system file containing job numbers, man numbers, and function codes. These services include addition of new numbers and codes, and deletion of inactive numbers and codes.

c) The facility to enter an operator's man number at the beginning of his shift of duty, and to remove the "operator number" at any time.

d) The facility to command the reading of a log-start card, either to begin a new job or to "flush through" an input deck from an abnormally-ended job. Information on the card may be overridden from the console.

e) Printing of beginning-of-job and end-of-job pages containing all the information logged for a job.

f) Inquiry into the recent history of the system, including last time entered, last job performed, and activity for the day.

g) The facility to originate a job, without need for a log-start card, from the 1407 console.

h) The facility to easily call various "system support" programs, such as log-dump, copy-disk, etc.

i) Maintenance of a "master line" file containing messages to be communicated to users from the operations staff, either typed or printed at the beginning of a job. Lines may be added or removed.

j) Logging of operator comments, and of the use of facilities b), and i).

k) Punching of log-start cards.

RESTRICTIONS:

Time, Core Requirements:

No timing estimates can be given for a particular use of the inter-job supervisor, since this depends upon the operator and on the options which are used. In general, the time spent in the inter-job supervisor should not add more than .01 computer clock hours per average job.

MØNITR is effectively a mainline program, using all of core storage during its operation.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

No specialized set-up or previous processing is necessary for use of the inter-job supervisor except: 1) the Operating System pack must be resident on one of the 1311 disk drives whenever the inter-job supervisor is called or is in operation. 2) The job number, man number, and function code associated with a job to be run must either have been entered into the system previously, or must be added before the job can be run. Output from MØNITR on the 1403 printer is such that any type of form can be used.

The details of originally building a system are given in a separate document.

Processing/Method:

The inter-job supervisor is an extremely modular program, with various routines being utilized as needed due to operator entry or historical situation (e.g. a past information request or some past entry). The various routines are described separately in the paragraph on Processing Method -- Technical Description.

USAGE:

Source of Input:

Input to the inter-job supervisor consists of the log-start card, typewriter input commands, and system-requested typewriter input. Each of these sources is separately detailed below.

Calling Sequence:

The inter-job supervisor should be called on an end-of-job condition by all 1401 programs written at UWM, by means of the EXIT macro (KF08A). The operator has two equivalent alternatives if an abnormal end-of-job occurs, preventing the EXIT in the program from being executed:

a) if the job was run under the LØAD program, the operator can set 15870 as the I-address and press START.

b) the operator can load a log-start card with the user information columns blank.

These possibilities are both equivalent to the execution of an EXIT macro. The log-start card for the next job can be loaded, but this does not allow entry of the ending time of day and computer clock time.

Control Cards:

Log-start Card:

The version 1 log-start card operates correctly with the version 2 Operating System, in that a new-style card is punched and also executed in the machine.

The format of the basic Version 2 log-start card is as follows:

Columns	1-7	,047008
"	8	N
"	38-46	019798001
"	47-53	,054062
"	54-61	L%F0037R
"	62-68	,069076
"	69-75	D046057
"	76-79	B054

User-provided information is in the following format:

Columns	12-16	Job number
"	17-19	Man number
"	20-21	Function code
"	22-27	Program name
"	28-36	Program common
"	37	Drive code on which the Operating System Pack resides (0 or 2)

All of the user-provided fields are optional except the program name and drive code. The job number is usually present.

Typewriter Input:

Typewriter Entry Codes:

A typewriter entry code consists of a 2-character code which may or may not be followed by one or more operands. Some codes have different meanings depending on what has been entered previously and some others have prerequisites without which they will not be honored, but most may be entered at any time and in any order.

Some entry codes are commands in nature, others provide information to the system. A few entries must be in a fixed format, but most can have spaces imbedded in the entry at the discretion of the operator.

A complete list of entry codes follows:

AF command (add function code)

This command must be entered in a fixed format consisting of the letters AF, followed by a 2-digit function code, followed by a 13-character name corresponding to the code. The function code is added to the system file if it is not already present.

Prerequisites: The time of day and a man number must be entered (by means of the TM and MN entries) before this command can be used.

AJ command (add job number)

This command must be entered in a fixed format consisting of the letters AJ, followed by a 5-digit job number, followed by a 13-character name corresponding to the number. The job number is added to the system file if it is not already present.

Prerequisites: The time of day and a man number must be entered (by means of the TM and MN entries) before this command can be used.

AM command (add man number)

This command must be entered in one of two fixed formats consisting of the letters AM, followed by a man number, followed by a 13-character name corresponding to the number. The man number may be either 2 or 3 digits in length. The man number is added to the system file if it is not already present.

Prerequisites: The time of day and a man number must be entered (by means of the TM and MN entries) before this command can be used.

CC Entry (enter computer clock time)

This entry provides the current machine clock reading to the system. The letters CC should be followed by:

a) the four least-significant digits of the clock reading, or
b) all six digits of the clock reading, or
c) all six digits of the clock reading followed by a record mark.

Format a) should suffice in most cases. Format b) must be used when a turnover occurs every 100 machine hours. Format c) must be used when a sequence error occurs. A sequence error occurs when an error was made in the previous machine clock reading entry, causing the present reading to be lower than the last, or when more than two hours of clock time have elapsed since the previous entry.

The CC entry and the TM entry should be made immediately after an end-of-job so as to provide ending times for the job. If the entry is made after a job number or man number is entered, as in loading a log-start card, the entry counts only as a beginning reading for the next job.

If this entry is not made before the initiation of a job with LS, DØ, or loading of a job's log-start card, the computer clock time may be requested by the system.

CM entry (enter operator comment)

This entry causes an operator comment to be entered into the system log. Any string of characters can follow the letters CM. If the entry is made before a job number or man number is entered for the next job, the line is considered a comment for the previous job; if not, the comment references the next job.

DC command (execute disk-to-tape)

This command causes the disk-to-tape utility DCØPY (ZD01A) to be executed for the disk pack on the opposite drive from the Operating System pack (usually the IBM system pack). The difference between this command and a DØ DCØPY is that the operator need not enter the job number, function, and program common when using DC.

DF command (delete function code)

The letters DF are to be followed by a 2-digit function code. If the function code designated is in the system file, it is deleted from the file.

Prerequisites: The time of day and a man number must be entered (by means of the TM and MN entries) before this command can be used.

DJ command (delete job number)

The letters DJ are to be followed by a 5-digit job number. If the job number designated is in the system file, it is deleted from the file.

Prerequisites: The time of day and a man number must be entered (by means of the TM and MN ent ies) before this command can be used.

DM command (delete man number)

The letters DM are to be followed by a 2- or 3-digit man number. If the man number designated is in the system file, it is deleted from the file.

Prerequisites: The time of day and a man number must be entered (by means of the TM and MN entries) before this command can be used.

DO command (initiate)

The letters DØ are to be followed by a 1 to 6-character program name. This command enables the operator to initiate a job completely from the console, without use of a log-start card. When all information required for the running of the job is entered into the system, the program designated is called.

FC entry (enter function code)

The letters FC should be followed by a 2-digit function code. If the entry is made before a job number or man number is entered for the next job, the function code is considered as a revised function for the previous job; otherwise the function code references the next job.

If this entry is not made before the initiation of a job with LS, DØ, or loading a log-start card, the function code may be requested by the system (see the paragraph on System-Requested Typewriter Input). A function code entered from the typewriter overrides a function code read from a log-start card.

JB entry (enter job number)

The letters JB should be followed by a 5-digit job number for the next job. Ordinarily the job number is taken from a log-start card, but a job number entered from the typewriter overrides a job number read from a log-start card. This entry should not be made before the TM and CC entries for the last job are entered.

KL command (clear master lines)

This command causes all "master lines" to be deleted from the system. (See ML command.)

Prerequisites: The time of day and a man number must be entered (by means of the TM and MN entries) before this command can be used.

LG command (execute log-dump)

This command causes the log-dump program DUMP (KE02A) to be executed. The log is then dumped onto cards, printed, and set to an empty condition. The difference between this command and a DØ DUMP is that the operator need not enter the job number, function, and program common when using LG.

LL command (list log)

This command is similar to LG above, but the log is only printed; it is neither dumped onto cards nor set to an empty condition.

LS command (log-start search)

This command causes cards to be read until a log-start card is encountered. The user-provided information on the card is then entered into the log. Information on the card is overridden by information already entered from the typewriter. When all information required for the running of the job is entered into the system, the designated program is called.

This command should not be used until the TM and CC for the previous job have been entered.

ML command (enter master line)

This command causes a "master line" to be established in the system. The ML is to be followed by a string of characters which will be printed and/or typed at the beginning of each job. If the letters ML are followed directly by the characters >>or √√, the master line is both typed and printed; otherwise, the line is only printed.

The purpose of "master lines" is to communicate information from the operations staff to users and/or persons operating the system during off-hours. A maximum of 12 master lines may be entered. These remain in the system until all are deleted by a KL command.

Prerequisites: The time of day and a man number must be entered (by means of the TM and MN entries) before this command can be used.

MN entry (enter man number)

The letters MN should be followed by a 2- or 3-digit man number. A man number entered from the typewriter overrides a man number read from a log-start card. If an operator is logged in, a man number is not required to run a job, but may be entered. This entry should not be made before the TM and CC entries for the last job are entered.

ØC command (Operating System disk-to-tape)

This command causes the disk-to-tape utility DCØPY (ZD01A) to be executed for the Operating System disk pack. The difference between this command and a DØCØPY is that the operator need not enter the job number, function, and program common when using ØC.

ØI command (log in operator)

The letters ØI should be followed by a 2- or 3-digit man number. The purpose of the entry is to log an operator in, by number, until the end of his shift of duty. Thereafter during that shift, man numbers are not required. The operator is not logged in until he responds to a subsequent message as described below in the paragraph on System-Requested Typewriter Input.

ØØ command (log out operator)

This command causes the logged-in operator, if any, to be logged out immediately.

PC entry (enter program common)

The letters PC are to be followed by a 1- to 9-character program common field. If the field after PC is less than 9 characters in length, the right-end of the field is filled out with blanks. Blanks are not ignored in this entry. The program common field is generally taken from a log-start card, but a program common field entered from the typewriter overrides a program common field read from a log-start card. This entry is often used in conjunction with the DØ command.

PL command (execute program lister)

This command causes the program lister PLIST (KC08A) to be executed to produce a list of programs in the Operating System Library. The difference between this command and a DØ PLIST is that the operator need not enter the job number and function when using PL.

PN command (execute file lister)

This command causes the file lister LSNØ (KE04A) to be executed to produce lists of job numbers, man numbers, and functions in the system file. The difference between this command and DØ LSNØ is that the operator need not enter the job number and function when using PN.

PU command (punch log-start card)

This command causes a Version 2 log-start card to be punched. If the job number, man number, program common, and/or function have been entered, they are punched in the corresponding columns of the card. The letters PU may be the complete entry or may be followed by a 1- to 6-character program name. If a program name is part of the command, it is punched in the program name field; otherwise, the program name previously entered by a DØ, an LS, or a loaded log-start card (if any) is punched in the card.

TM entry (enter time of day)

This entry provides the current time of day to the system. The letters TM should be followed by:

a) the hour and minute on a 24-hour clock basis (4 digits), or
b) the hour and minute, followed by the month and date (8 digits), or
c) the hour, minute, month, and date, followed by a record mark (9 characters), or
d) the hour, minute, month, and date followed by the two digit year (10 digits), or
e) the hour, minute, month, date, and year, followed by a record mark (11 characters).

Format a) should suffice in most cases. Format b) must be used when a 24-hour clock turnover (i.e. midnight) has occurred since the last entry. Format c) must be used when a sequence error occurs. A sequence error occurs when an error was made in the previous time of day entry, causing the present reading to be lower than the last, or when more than 24 hours have elapsed since the previous entry. Formats d) and e) should only be necessary when a year change occurs.

The TM entry and the CC entry should be made immediately after an end-of-job so as to provide ending times for the job. If the entry is made after a job number or man number is entered, the entry counts only as a beginning reading for the next job.

If this entry is not made before the initiation of a job with LS, DØ, or loading of a job's log-start card, the time of day may be requested by the system.

WA command (list jobs)

This command causes a list of jobs run thus far on the date of entry to be printed on the 1403.

WJ command (list last job details)

This command causes all known information about the last job to be printed.

WT command (type last time)

This command causes the last time entered into the system to be typed. It can be useful when a sequence error occurs.

System-requested typewriter input:

In certain conditions the Operating System will request, from the operator, various parameters necessary for some activity he has initiated (typically, running a job). Only information which has not been entered previously will be requested.

The requests which may be made by the system are:

a) >> TYPE JØB NUMBER
b) >> TYPE MAN NUMBER
c) >> TYPE TIME OF DAY
d) >> TYPE CØMPUTER CLØCK TIME
e) >> TYPE FUNCTION
f) >> TYPE TIME ØUT .

a) through e) may be typed after an LS or DØ command, or after a log-start card is loaded. b) through d) may be typed after any of the commands DC, LG, LL, ØC, PL, and PN have been entered. f) can only occur after use of the ØI command.

When any of these requests are typed, the operator still has all the freedom of action as before: that is, he can still enter any monitor entry code or command. However, he has the further option of entering the information requested without any preceding 2-character code. If some other piece of information is entered by means of a monitor entry code following such a request, the

operator still has the option thereafter of entering the requested information without a preceding 2-character code. Certain commands, those mentioned above in this paragraph, can cause information to be discarded by the system and the request to be nullified. This is also caused by allowing the system to enter the "WAITING" state. However, in normal operation this is not expected to occur.

The form of response to any of the above requests is similar to that of a monitor entry code in each case, except that the two-character code need not begin the response, as follows:

The response to a) is similar in format and meaning to the JB entry.
The response to b) is similar in format and meaning to the MN entry.
The response to c) is similar in format and meaning to the TM entry.
The response to d) is similar in format and meaning to the CC entry.
The response to e) is similar in format and meaning to the FC entry.
The response to f) is similar in format to the TM entry, but the time entered should be that of the end of the operator's shift of duty.

OUTPUT:

The main output produced by the inter-job supervisor is the entries in the system log. The format and content of these entries are described in a separate document.

Other types of output produced by the inter-job supervisor are:

a) information requests, covered in the previous paragraph.
b) normal and error messages, covered below in the paragraph on Error Messages and Halts.
c) punched log-start cards, the format of which is described above in the paragraph on the Log-Start card.
d) the BØJ and EØJ pages, and similar output produced by the WA and WJ commands.

Output of type d) contains all the information known about a job at the time the output is printed. Such output is printed at every job initiation, and is printed at every job termination for which the operator enters the ending times.

ERROR MESSAGES AND HALTS:

Aside from:

a) the system standard halts and messages (note that the "WAITING..." message has an expanded meaning),

b) special messages defined as 'master lines',

c) system information requests (see System Requested Typewriter Input),

the following halts and messages can occur:

1. >> END-ØF-JØB.
This message indicates that a running program has executed an EXIT macro, or that a log-start card with the user information columns blank has been LØADED.

2. √√ DØES NØT CØMPUTE.
This message indicates that the operator has entered either a line beginning with alphabetic characters which are not a legal entry code, a line beginning with numerics when no system information request was in effect, an entry code with an operand for which one is not required, or an entry code without an operand for which an operand is required.

3. >> CARD JB xxxxx
>> CARD JB xxxxx ØVERRIDDEN
>> CARD MN xxx
>> CARD MN xxx ØVERRIDDEN
>> CARD FC xx
>> CARD FC xx ØVERRIDDEN
>> CARD PC xxxxxxxxx
>> CARD PC xxxxxxxxx ØVERRIDDEN
Any of these messages can occur following the loading or reading of a log-start card. They indicate the information punched on the log-start card and its handling in comparison with previously entered information.

4. √√ LØG-START CARD HAS NØ PRØGRAM NAME.
This message indicates that a log-start card read by means of an LS command did not contain a program name.

5. √√ NØNEXISTANT JØB NUMBER.
√√ NØNEXISTANT MAN NUMBER.
√√ NØNEXISTANT FUNCTION.
These messages indicate that a number or code entered into the system by means of a JB, DJ, MN, DM, FC, or DF entry code, from a log-start card, or as a response to a system information request, does not exist in the system file.

6. √√ WRØNG LENGTH.
This message indicates that an entry code or a response to a system information request is of a length which does not correspond to the correct format for that type of entry or response.

7. √√ IMPRØPER FØRMAT.
This message indicates that an entry code or a response to a system information request is of a format (other than its length) which is not correct for that type of entry. (Example: alphabetic information when numeric is required.)

8. √√ IMPØSSIBLE TIME
This message indicates that a time of day in an entry code or a response to a system information request is either in an incorrect format or contains a subfield having an impossible value. (Examples: month field over 12, minute field over 59.)

9. √√ SEQUENCE ERRØR.
This message indicates that an entered time is either lower than the previous time entered, or too far ahead of the previous time.

10. √√ NØT ALLØWED.
This message indicates that the prerequisites for the previously entered command (typically man number and time of day) have not been entered.

11. √√ ALREADY ØN.
This message indicates that the number or code in an AJ, AM, or AF command is already present in the system file.

12. √√ TABLE FULL.
This message should not occur, but if it does occur it will be in response to an AJ, AM, or AF command when the system file is full.

OPERATING PROCEDURES:

Various operating procedural notes have been given in previous paragraphs. Considerable flexibility and operator preference capability are inherent in the design of the inter-job supervisor. Thus only scattered recommendations can be made in the area of operating procedures:

a) if a job comes to an abnormal end-of-job, or halts, the operator can return to the Operating System by using the core-dump deck, by setting 15870 into the I-register and pressing START (if the program had been run from an object deck by means of the LØAD program), or by loading a log-start card with the user information columns blank.

b) when the end-of-job message is typed, the operator should enter a TM entry and a CC entry within the timeout period, so as to produce the EØJ page and provide finishing times for the job.

c) if there are no jobs to be run after the completion of step b), the system should be allowed to assume the 'WAITING...' state. This insures the correct starting time of day for the eventual next job.

d) operators should be aware that fields on a log-start card can be overridden from the console either before or after the LS command.

e) post-job function changes and comments should be entered before the CC and TM of step a), and must be entered before an MN or JB entry, a response to a system information request, or an LS command is done.

f) operators should be aware that the "operator in" facility removes the necessity either for their man numbers to be punched on a log-start card, or for that card field to be blank. It is perfectly acceptable, and probably desirable, for a job to be run under both a man number (that of a programmer or user) and an operator number.

g) in the case of a sequence error in the time of day, the WT inquiry can be helpful.

h) operators should be aware that there is no need for them to make up a log-start card the first time a job is run, and that by entering a JB entry and possibly other information entries, and using the PU command, they can produce a log-start card for subsequent runs of the job.

i) job numbers and man numbers should always be added with as complete and correct a name as can be contained in 13 characters.

The operator should respond as follows to the halts and messages described in the paragraph on Error Messages and Halts.

1) Type in the TM and CC entries as previously described.

2) Re-enter the line correctly.

3) Examine the message(s) for correctness. A system information request will be forthcoming.

4) Type LS again and/or examine the card which was read.

5) Either add the number to the system file, or enter a different, correct number by means of an entry code or a response to a system information request.

6) Re-enter the line correctly.

7) Re-enter the line correctly.

8) Re-enter the line correctly.

9) Reconsider the correctness of the entry and/or use an expanded format as described above in the paragraphs on Typewriter Entry Codes and System-Requested Typewriter Input.

10) Enter the prerequisite information, and then re-issue the command.

11) Reconsider the correctness of the entry and/or abandon the attempt.

12) Abandon the attempt, or delete one or more entries from the system file.

End-Of-Job, Post-Processing:

From the point of view of the inter-job supervisor, its "end-of-job" is usually the beginning of an actual job. The operator should refer to the set-up and operating instructions for the program in question.

PROCESSING METHOD -- TECHNICAL DESCRIPTION

Entry to Inter-Job Supervisor

If entry was due to a loaded log-start card, the card is moved into an input area within the program. The drive code of the Operating System is saved, and the "monitor communications sector" is read (ref: ØSINF macro, KF17A). The communications sector is initialized so that the following items of information are marked "unknown" in the communications sector:

a) the time of day
b) the computer clock time
c) the job number
d) the man number
e) the function code
f) the program name field
g) the program common field (set to blanks)

If the entry was due to a loaded log-start card on which the program name field is not blank, control is transferred to the log-start processing routine. Otherwise, the end-of-job message is typed, and control is transferred to the wait/timeout routine.

Wait/Timeout routine

This routine waits for the REQUEST/ENTER button to be depressed by the operator. If the routine waits for the number of seconds dictated by operating procedure as the "timeout factor", the communications sector is initialized, and fields marked as "unknown" as in the previous paragraph (Entry to Inter-Job Supervisor), with the exception that the computer clock time is not marked as unknown. The last information request (see System-Requested Typewriter Input), if any, is also cancelled. The systems standard message and halt for this situation then occurs, and should be started in the standard fashion.

If the operator does depress the R/E key within the time-out period, he can enter information at his own rate. However, operators should bear in mind that the computer clock runs when in this situation. When the operator depresses the RESPØND/TYPEØUT key after typing, control is transferred to the entry/scan routine.

Entry/Scan Routine

This routine first transforms the operator input by eliminating blanks from the message in all except AF, AJ, AM, CM, ML, and PC entries. A length count for the operator input line is also constructed at this time. The first two non-blank characters of the line are then examined. If the first two characters are numeric, control is transferred to an appropriate routine if a previous information request (see System-Requested Typewriter Input) had occurred; if the first two characters are alphabetic, control is transferred to an appropriate routine if they represent a valid monitor entry code (see Typewriter Entry Codes); otherwise, a diagnostic is produced.

Log-Start Card Processing Routines

On an LS command, cards are read until a log-start card is found. An old log-start card is executed when read, causing performance of the compatible log-start program (KC03A), and eventual return to MØNITR as a loaded log-start card. No check for last-card indicator is done on an LS command.

When a new log-start card has been read, if the LS command occurred after an information request (see System-Requested Typewriter Input), items c) through g) as described above in the paragraph on Entry to Inter-Job Supervisor are marked as "unknown" in the communications sector. If a log-start card with a blank program name field is read by means of an LS, a diagnostic is produced and further cards are read.

On either a loaded log-start card or a card read by means of an LS, master lines (if any) are typed. The following then occurs for the job number, man number, function, and program common fields:

a) if the field is blank on the card, no processing is done for that item.

b) if the field on the card is not blank and the field in the communications sector is marked "unknown", a message is typed and the information from the card moved to the sector.
c) if the field on the card is not blank and the corresponding field in the communications sector contains information from a previous typewriter entry, the information on the card is overridden and a message to that effect is typed.

The job number, man number, and function code fields are then checked for validity. A diagnostic is typed, and the field in the communications sector marked "unknown", for any item found invalid. If the program name is valid, control is then transferred to the request-information routine.

Request-Information Routine

This routine examines the communications sector and various switches for the presence of each of the following, in turn:

a) the job number
b) the man number
c) the time of day
d) the computer clock time
e) the function code

The first one of these it finds to be "unknown" causes the appropriate information request to be typed, and a switch to be set signifying that item to be "pending". Control is then returned to the wait/timeout routine. If all the above have been entered, control is transferred to the perform-job routine. If the man number is "unknown" but an operator is on duty, the information request for the man number is not issued.

Perform-Job Routine

This routine performs the following functions in sequence:

a) makes a job entry in the system log,
b) enters previously-made comments for this job into the system log,
c) prints master lines (if any) on the 1403,
d) prints the BØJ page on the 1403,
e) and performs the program indicated by means of a LINK macro (KF14A).

Information Entry Routines

Time of Day and Computer Clock Routines

Routines in this class are used on a numeric entry after an information request, or on a TM or CC entry code. The operation of the time and clock routines are similar, both being of the following pattern:

a) the length of the entry is checked, and a diagnostic produced if the length does not correspond to any of the proper formats. From the length the system determines what parts of the complete field are to be assumed as continuing from the previous entry.

b) the field is checked to see if it is completely numeric. If it is not, a diagnostic is produced.

c) in the case of the time of day, subfields (month, date, etc.) are examined for plausibility, and a diagnostic may be produced at this point.

d) if acceptance was not forced with a record mark, the entered information is compared with the previous entry. A diagnostic may result if the value is lower or is much higher.

e) the value is moved to the communications sector and the quantity is set as "known".

f) if the time was not known before, and the system has never been in the "WAITING" state, and neither a man number nor a job number has been entered for the next job, the entry is taken as an ending time for the previous job.

g) if all the conditions of f) are met, and the computer clock time has been entered, the end-of-job page is printed.

h) in the case of the computer clock time, the "total-cc" fields of the job number, man number, function code, operator number, and program name of the last job are incremented by the elapsed clock time.

i) in the case of the time of day, if there is an operator "in" and the time entered is after the ending time for his shift of duty, the operator-in designation is removed with no notice given.

j) if the time entered had been "pending" from a previous information request, control is transferred to the request-information routine. Otherwise, control is returned to the wait/timeout routine.

Man Number, Job Number Routines

Routines in this class are used on a numeric entry after an information request, or on a MN or JB entry code. The operation of the job and man routines are similar, both being of the following pattern:

a) the length of the entry is checked, and a diagnostic produced if the entry is not in the proper format.

b) the entered number is moved to the communications sector.

c) the system file is searched for the entered number. If the search does not produce a match, a diagnostic is produced and the field is marked as "unknown" in the communications sector.

d) if the number entered had been "pending" from a previous information request, control is transferred to the request-information routine. Otherwise, control is returned to the wait/timeout routine.

Function Routines

These routines are used on a numeric entry after an information request or on a FC entry code. The operation is of the following pattern:

a) the length of the entry is checked. If it is not of the correct format, a diagnostic is produced.

b) the system file is searched for the entered function code. If the search does not produce a match, a diagnostic is produced and the field marked as "unknown" in the communications sector.

c) if either a man number or a job number for the next job has been entered, the entered function code is moved to the communications sector and the field marked as "known". Otherwise, the entered function code is moved into the log entry for the last job as a correction.

d) if the function code entered had been "pending" from a previous information request, control is transferred to the request-information routine. Otherwise, control is returned to the wait/timeout routine.

Comment Routine

This routine is used on a CM entry code. If a job number or man number for the next job has been entered, the line is transferred to a buffer file for later recording in the system log by the perform-job routine. Otherwise, the line is recorded directly in the system log as a comment for the last job.

Program Common Routine

This routine is used on a PC entry code. A diagnostic is produced if the length of the field after the letters PC exceeds 9. Otherwise, the entered information is moved to the program common field in the communications sector, left justified, with right-end blanks supplied if necessary.

Command Routines

Program Name Routine

This routine is used on a DØ command. The length of the entry is checked: a diagnostic is produced if more than 6 characters follow the letters DØ; if not, the program name is checked for validity (presence in the Operating System Library), and a diagnostic is produced if a match is not found. Master lines (if any) are typed at this point. If the DØ command occurred after an information request (see System-Requested Typewriter Input), items c) through g), as described in the paragraph on Entry to the Inter-Job Supervisor, are marked as "unknown" in the communications sector. The program name entered is moved to the communications sector, and control is transferred to the request-information routine.

"System Support Program" Call Routines

Routines in this class are used on a PN, LL, ØC, DC, LG, PL, or PS command. These commands are similar in intent to the DØ command, but are easier for the operator to use. On any of these, a "system job number"

and a "system function code" are moved to the communications sector, and the fields marked as "known", unless a job number and/or function code was already entered. The program name and program common fields are then set up in the communications sector, as follows:

COMMAND	PROGRAM NAME	PROGRAM COMMON
PN	LSNØ	None
LL	DUMP	Blank
LG	DUMP	D
ØC	DCØPY	Operating system drive code
DC	DCØPY	Opposite drive code from Operating System
PL	PLIST	Blank

Control is then transferred to the request-information routine.

Operator In/Out Routines

This class of routines is used on a ØI or ØØ command. On an ØI, the length of the entry is checked, and a diagnostic is produced if the information entered is not in the proper format. The 2- or 3-digit man number is then moved to a temporary location and checked for validity. A diagnostic may occur at this point. If not, a system information request is issued, a switch is set indicating the "time out" to be "pending", and control is returned to the wait/timeout routine.

When the "time out" is entered, control is transferred to the time-of-day routine, which checks the time entered as previously described. That routine then transfers control to the second part of the operator-in routine, if the time is accepted. The "operator-in" indication for the system is then set.

On an ØØ, the "operator out" indication for the system is simply set, with no other investigation or processing.

Add Man-Job-Function Routines

Routines of this class are used on an AF, AJ, or AM command. Operation of any of these commands proceeds as follows:

a) if the man number or the time of day is marked "unknown" in the communications sector, a diagnostic is issued.

b) the length of the entry is checked, and a diagnostic is produced if the entry is shorter than the minimum length of the proper format.

c) the system file is searched for the number or code entered, and a diagnostic is produced if a match (duplicate) is found.

d) the number or code entered is checked for being completely numeric. If it is not, a diagnostic is produced.

e) the number and associated name are inserted in the system file.

f) a log entry for the file addition is made.

g) control is returned to the wait/timeout routine.

Delete Man-Job-Function Routines

Routines of this class are used on a DF, DJ, or DM command. Operation of any of these commands proceeds as follows:

a) if the man number or the time of day is marked "unknown" in the communications sector, a diagnostic is issued.

b) the length of the entry is checked, and a diagnostic is produced if the entry is shorter than the minimum length of the proper format.

c) the system file is searched for the number or code entered.

d) a diagnostic is produced if the number or code entered is not completely numeric.

e) the number and its associated name are replaced by the last entry in the file.

f) the end-of-file indicator is moved up one record.

g) control is returned to the wait/timeout routine.

Punch Log-Start Card Routine

This routine is used on a PU command. A basic log-start card is augmented by the fields in the communications sector which have been previously entered and are "known". If a program name follows the letters PU, it is moved to the program-name field in the card; otherwise, the program-name field in the communications sector (possibly blank) is moved to the card. The log-start card is then punched into stacker 4. An extra blank card is punched, to clear the punch, into stacker N/P. Control is returned to the wait/timeout routine.

Master-Line Routines

These routines are used on ML and KL commands. On an ML, the length of the entry is checked for being 4 or greater, and a diagnostic can be produced at this point. If the man number or the time of day is marked "unknown" in the communications sector, a diagnostic is issued. Otherwise, the line entered is added at the end of the master line file, and the "number of master lines" count is incremented by one. A log entry for the file addition is made, and control is returned to the wait/timeout routine.

On a KL, if the man number or the time of day is marked "unknown" in the communications sector, a diagnostic is issued. If the "number of master lines" count is not zero, it is set to zero and a log entry is made. Control is then returned to the wait/timeout routine.

Inquiry Routines

What-Time Routine

This routine is used on a WT inquiry. The last time entered is edited and typed out.

What-Job Routine

This routine is used on a WJ inquiry. A page similar to the BØJ and EØJ pages is printed on the 1403, containing all known information on the last job.

What-Activity Routine

This routine is used on a WA inquiry. An entry similar to the BØJ and EØJ pages is printed on the 1403 for each job run previously on the day the inquiry is entered, containing all known information about each job.

1401 O. S. II
I. D. # KE02A
I. D. # KE03A

PROGRAM:	DUMP/HELP
MACHINE:	IBM 1401
LANGUAGE:	AUTØCØDER
SUPERVISOR PROGRAM:	1401 Operating System II
PROGRAMMER:	Systems Division
DATE COMPLETED:	July 1, 1968

PURPOSE:

The programs DUMP (KE02A) and HELP (KE03A) are log-dump routines. They produce a narrative log listing and punch the 1401 log into cards. DUMP is the normal log-dump routine. HELP is the emergency log-dump routine, performing this function without the aid of the Operating System.

RESTRICTIONS:

Time, Core Requirements:

Execution time for DUMP/HELP will normally be less than 5 minutes, but may vary considerably depending on the number of entries in the log.

DUMP/HELP is a mainline program, effectively using all of core during execution.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

There should be 14 x 11 inch paper mounted on the 1403 printer. The Operating System pack must be mounted on one of the 1311 disk drives.

Processing/Method:

The log entries are checked to determine what type of information they contain. When the ending time (real) is missing from the type 1 entry (see below, Punched Output), a scan is made to determine the next time of day. The management and operator performance logs are then created and punched when the next time has been determined.

When the end-of-file condition occurs, the communications sectors are updated, the end-of-file is reset, and control is transferred to the inter-job supervisor.

USAGE:

Calling Sequence:

General Usage (DUMP):

DUMP is normally called by means of the inter-job entry codes LD and LL, but may be called by a log-start card or by execution of a LINK macro from an AUTØCØDER or CØBØL program.

Emergency Log-Dump (HELP):

HELP is executed by clearing the reader, placing the HELP object deck in the reader, and pressing Load.

Program Common Field:

The contents of the program common field used to call DUMP determine whether the machine log is to be punched and listed, or just listed. When the letter "D" is present, left justified, in the program common field, the log is punched into cards.

Control Cards:

DUMP/HELP requires no control cards.

Input:

The only input consists of the log information stored on the Operating System disk pack.

OUTPUT:

Printed Output:

A narrative log is printed containing the following information:

a) all the information contained on the E.O.J. page.
b) program library operations controlled by DUP or DLØAD.

c) system support functions caused by use of the inter-job entry codes LG, ML, or KL.
d) file changes such as addition or deletion of job numbers, man numbers, and function codes.

Punched Output:

Three types of log are punched by DUMP/HELP. They are the complete log, the operator's performance log, and the computer management log.

The complete log is subdivided into five different types of intormation:

a) The job initialization and ending times, both real and computer. The job number, the function code, program name, man number, operator number, program common, and the run count are included in the type 1 log card.
b) The information created by use of the CM code, comment entry, man name, and man number are included in the type 2 log.
c) The system support log, type 3, contains the program name, starting time, man number, and man name.
d) The log entry file change log, type 4, contains the name of the file changed, if it was an addition or deletion, and the man number and man name of the person responsible.
e) The program update log entry, type 9, contains the operation name, program name 1, program name 2, or tape name, sector address, number of sectors, starting time (real), man name, and man number.

The operator performance log contains statistics that pertain to the function of the machine operation and utilization of the machine.

The management log contains information that pertains to the job initiation and ending times.

NOTE: Type n refers to the number appearing in column 1 of the punched card.

ERROR MESSAGES AND HALTS:

Aside from the standard system halts and messages which may occur during execution of DUMP, no halts or messages should occur.

OPERATING PROCEDURES:

No special operating procedures need be specified.

End-Of-Job, Post-Processing:

DUMP returns to the inter-job supervisor. When the punch option is specified, the operator will find 3 stackers containing the log. The operator performance log should be forwarded to the 1401 operations manager; the complete log should be forwarded to the computer systems group; the management log should be forwarded to the business manager.

PROGRAM:	LSNØ
MACHINE:	IBM 1401
LANGUAGE:	AUTØCØDER
SUPERVISOR PROGRAM:	1401 Operating System II
PROGRAMMER:	Systems Division
DATE COMPLETED:	July 1, 1968

PURPOSE:

LSNØ, when executed, sorts and lists the man-job-function number file in usage order. The man-job-function numbers are then subdivided and listed in numerical order.

RESTRICTIONS:

Time, Core Requirements:

Processing time varies as the square of the number of entries in the man-job-function file. The average run is about 5 minutes.

LSNØ is a mainline program effectively using all of core during its execution.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

Paper at least 14 inches wide must be used in the 1403 printer. The Operating System pack must be mounted on one of the disk drives.

Processing/Method:

The man-job-function number file is sorted in place, and then listed in order of the number of runs. During the listing process, a sort of binary tags is performed. The file is then listed using the tags correspondence to the new sorted file.

USAGE:

Calling Sequence:

LSNØ is normally called by use of the inter-job entry code PN, but may be called by a log-start card, or by execution of a LINK macro from an AUTØCØDER or CØBØL program.

Control Cards:

LSNØ requires no control cards.

Input:

Input consists of the man-job-function file, read from disk.

OUTPUT:

LSNØ lists the man-job-function numbers in numerical order, subdivided into the three classes. The following statistics are printed with each entry:

a) date entered.
b) last date used.
c) number of runs.
d) total number of computer clock hours used.
e) name.

ERROR MESSAGES AND HALTS:

Aside from the standard system halts and message, no halts or messages should occur during execution of LSNØ.

OPERATING PROCEDURES:

No special operating procedures are required.

End-Of-Job, Post-Processing:

LSNØ returns control to the inter-job supervisor on completion of the listing.

1401 O. S. II
I. D. # KC07A

TITLE	LØAD
MACHINE:	IBM 1401
LANGUAGE:	Autocoder
SUPERVISOR PROGRAM:	none
PROGRAMMER:	Systems Division
DATE COMPLETED:	July 1, 1968

PURPOSE:

LØAD simulates the function of the 1402 LØAD button, transferring control to a card object program. Furthermore, LØAD provides the object program with the IØCS date, a program common field, and an EXIT macro in high-core.

RESTRICTIONS:

Time/Core Requirements:

Execution time for LØAD itself is less than one second. LØAD is a mainline program, effectively using all of core during its brief execution. As noted below, LØAD leaves locations 82-86 and 15861-15998 occupied when it transfers control to the object program.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

Set-up is as required by the object program to be loaded.

Processing/Method:

Cards are read in succession until one with a comma in column 1 is found. If that first card is a "C1" card, it and the following card are passed. Core is then cleared except for the following:

a) locations 82-86 contain the IØCS date, with a wordmark at 82.

b) locations 15861-15869 contain the program common field, with a wordmark in 15861.

c) locations 15870-15998 contain an EXIT macro.

The card is then branched to at location 1. (A wordmark has been set in 1 previously.)

USAGE:

Calling Sequence:

LØAD is generally called by a log-start card. It may also be called by a LINK macro from an AUTØCØDER or CØBØL program, or by means of the inter-job entry code DØ.

Control Cards:

LØAD requires no control cards.

Input:

The only input consists of the object program to be loaded.

OUTPUT:

LØAD produces no output.

ERROR MESSAGES and HALTS:

Except for the system standard disk halts and messages, no halts or messages should occur.

OPERATING PROCEDURES:

Operating procedures are as required by the program to be loaded.

End-of-Job, Post-Processing:

End-of-job and post-processing requirements are as needed by the object program to be loaded.

PROGRAM:	PLIST
MACHINE:	IBM 1401
LANGUAGE:	Autocoder
SUPERVISOR PROGRAM:	1401 Operating System II
PROGRAMMER:	Systems Division
DATE COMPLETED:	July 1, 1968

PURPOSE:

PLIST produces a list, on the 1403 printer, of all programs and files resident under the Operating System. This list includes all known information about each program.

RESTRICTIONS:

Time, Core Requirements:

Execution time for PLIST varies according to the number of programs loaded under the Operating System. PLIST is a mainline program, effectively using all of core during its execution.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

No previous processing is necessary for use of PLIST. The listing produced is 132 positions in width, and thus 14 inch wide paper should be used in the 1403 printer.

Processing/Method:

The program table is read, and lines are printed in the order in which program-names are encountered in the table (descending order of sector address). For program entries, the header and trailer sectors are read, and checked for validity, providing the information for the printed line.

USAGE:

Calling Sequence:

PLIST is usually called by a log-start card, but may be called by a LINK macro from an AUTØCØDER or CØBØL program, by use of the inter-job entry codes PL or DØ, or by a failure to locate the specified program name in a FETCH or LINK macro. If a program name is included (right-justified) in the program common field, PLIST will link to that program after its execution.

Control Cards:

PLIST requires no control card.

Input:

PLIST uses no input.

OUTPUT:

A list is produced of all programs on the Operating System file, with all information known about each program. This information includes the program parameters created at DLØAD time, and the usage parameters accumulated since. Notations are included in the right edge of the listing, giving characteristics or error messages for certain programs.

ERROR MESSAGES and HALTS:

Notations in Output:

The following messages may appear at the right end of a printed line:

NO GMWM — indicates that group-mark word-marks are not restored when the program is loaded.

FIXED LOC — indicates that the program is stored at a fixed "system disk" address and will not be moved.

TTFD XXXXXX — indicates that the block is a Tabtran file description, with Tabtran file-name XXXXXX.

HEADERLESS indicates that the block is in load mode but has no program header, loader, or trailer. (viz. compatible DLØAD, EXIT, and log-start programs).

MOVE MODE indicates that the block is a move mode file (viz, the system log).

BAD FORMAT indicates that the program's header, loader, and/or trailer have been damaged. This is a serious error condition and should be reported to a member of the Computer Systems Group.

BAD ENTRY indicates that the program table itself is damaged. This is a serious error condition and should be reported to a member of the Computer Systems Group.

Other Halts and Messages:

Aside from the system standard disk halts and messages, no other halts or messages should occur.

OPERATING PROCEDURES:

No further operating procedures need be given: PLIST uses no sense switches, tapes, or other special configuration.

End-of-Job, Post-Processing:

PLIST either performs a link to the program named in the program common field (right justified), or performs an EXIT at end-of-job.

PROGRAM: DUP
MACHINE: IBM 1401
LANGUAGE: Autocoder
SUPERVISOR PROGRAM: none
PROGRAMMER: Systems Division
DATE COMPLETED: July 1, 1968

PURPOSE:

DUP, the 1401 Operating System disk utility program, provides various services associated with maintenance of the Operating System program library. These include:

- Deletion of programs
- Renaming of programs
- Copying of programs to magnetic tape
- Loading of programs from magnetic tape
- Printing of programs
- Modification of object programs via "patch cards"
- Punching of object decks of programs

Other facilities involving the program library are: Original loading of programs by means of "psuedo-execution", provided by the DLØDR program (KC06A); listing of names and specifications of programs provided by the PLIST program (KC08A).

The details of the operation of particular services are described in separate following sections. This section deals with the control cards and structure of DUP.

RESTRICTIONS:

Time, Core Requirements:

Timing estimates for performance of various services are detailed in the following sections. The time required for DUP itself to read and analyze the control cards is negligible, except in the case of typewriter input. DUP itself occupies the area from location 335 to approximately location 7000. The subprograms DUPA, DUPB, DUPC, DUPD, and DUPE utilize the area from 7000

to approximately 11300. Locations 11300 to 15900 are used as work areas. DUP is thus largely a mainline program, occupying most of core during its execution. However, locations 15901 to 15999 are never disturbed by execution of DUP or the subprograms. Thus they may be used, by programs calling DUP with disk input and re-calling, as a communications area.

PROGRAM PROCEDURE:

Set-up, Previous Processing:

In the case of *CØPY and *LØAD processing, magnetic tapes must be mounted. Paper at least 8 1/2 inches wide is required on the 1403 printer for all processing, and, in the case of *DUMP and *PATCH processing, 14 inch wide paper is required.

For all except *LØAD processing, the program(s) to be manipulated must have been loaded onto the program library previously. In the case of *LØAD processing, the program(s) must have been copied to tape previously.

Processing/Method:

The method used to read and accumulate a control record is largely defined in the following section (USAGE). Once a complete control record is accessed, its first non-blank characters are examined to determine the type of service request being made. When this is determined, one of the subprograms DUPA, DUPB, DUPC, DUPD, or DUPE is FETCHed into upper core, and a branch to the appropriate entry point made. The subprogram examines the remainder of the control record, issues diagnostics and/or performs the services indicated, and returns to DUP at the completion of its processing. The subprogram may use data areas and subroutines in DUP.

A table of the subprograms used to perform various services follows:

Service	Subprogram
*CØPY	DUPA
*LØAD	DUPB, possibly DUPC
*DELETE	DUPC
*RENAME	DUPC
*DUMP	DUPD
*PUNCH	DUPD
*PATCH	DUPE, DUPD

USAGE:

Calling Sequence:

DUP is usually called by a log-start card, but may be called by a LINK macro from an AUTØCØDER or CØBØL program, or by use of the inter-job entry code DØ. The subprograms DUPA, DUPB, DUPC, DUPD, and DUPE should not be called directly under any circumstances.

Program Common:

The content of the program common field used to call DUP to a large extent determines its operation. Consider the positions in the field to be numbered 1-9, from left to right:

a) If position 1 contains a "-", the operation of DUP consists of deleting the program whose name is to be found in positions 2-7. No input is required; control is returned to the inter-job supervisor following the deletion.

b) If position 1 contains a "T", control record input is taken from the console typewriter.

c) If position 1 contains a "D", control record input is taken from the "auxiliary Operating System communications sectors", on the 1311 disk drive.

d) If position 1 contains a blank or any other code, control record input is taken from the card reader.

e) In cases b), c), and d), if there is a program name to be found in positions 4-9, DUP links to that program at the conclusion of its operation.

Control Cards:

Source of Input:

Input to DUP, except in the case of *PATCH processing, consists entirely of control records. Programs on disk or tape are read and handled in various ways as described in following sections, but for the purpose of this section the term "input" will refer to information found in the file determined by position 1 of the program common field.

Control Card Structure:

Card/Disk Input:

With card input, all 80 columns of a card are used. With disk input, sectors 399622 through 399639 can contain control cards. The sectors are in the move mode; only the first 80 positions of each sector are used. With either card or disk input, except in the case of *PATCH processing, a "control record" may consist of one to six cards or card images. The end of one control record is determined by any of the following:

a) End-of-file indication (last card indicator for card input, advancing to sector 399640 for disk input).

b) A card whose first character is a comma, indicating that an object deck or log-start card follows (the previously read card completes the control record).

c) A card whose first non-blank character is an asterisk (the previously read card completes the control record).

d) A card containing a period followed by a blank, anywhere (that card completes the control record).

With card or disk input, an entire control record is accumulated and is terminated, according to the above rules, before any consideration is given to the syntax or meaning of the card, and before performance of the indicated services is initiated.

Typewriter Input:

With typewriter input, an entered line may be up to 80 characters in length. A single entered line always constitutes a complete control record. The system standard entry conventions are followed. The performance of the services indicated on each line is initiated immediately after the line is entered. On completion of these services, DUP again waits for the next control record to be entered, unless the line which initiated the services contained (ended with) a period followed by a space. End-of-job is indicated by a typewriter end-of-file: that is, use of the RESPØND/TYPEØUT key immediately following use of the REQUEST/ENTER key.

Content and Format of Control Records:

One rule about DUP control records has been implied in previous paragraphs: any control record which ends in a period followed by a space forces DUP to go to end-of-job immediately following performance of the services indicated on the control record. Other than this, the meaning of a DUP control record is exactly the same whether it ends with a period or not.

A program name in the 1401 Operating System may consist of 1 to 6 of the legal 1401 characters, with the exceptions:

a) a name may not contain a record mark
b) a name may not contain a comma
c) a name may not contain a blank
d) a name may not end with a period
e) a name may not begin with an asterisk.

The format of DUP control records is free: a control record may begin in any column; as many blanks as desired (or none) may be inserted between any two entities on a control record; a control record may occupy more than one card or card image. However,

a) spaces are required in the syntax in several cases.
b) the start of a control record must be the first character on a card or card image (the asterisk beginning it must be on the first non-blank character).
c) if "continuation cards" are used to form a complete control record, column 1 of each succeeding card is considered to directly follow column 80 of the preceding card.

Control Record Syntax:

The syntax for < DUP control record > is as follows:

< DUP control record > ::= < basic control card > /
< basic control card > . < space >

< basic control card > ::= < delete card > / < rename card >
/ < copy card > / < load card >
/ < dump card > / < punch card >
/ < patch card >

```
<delete card>     ::=   *DELETE  <space>  <program list>
                      / *DELET   <program list>

<rename card>     ::=   *RENAME  <rename list>

<copy card>       ::=   *CØPY    <program list> TØ <tape name>
                      / *CØPY    <program list> TØ <tape name>
                        WITH     <tape name>

<load card>       ::=   *LØAD    <program list> FRØM <tape name>

<dump card>       ::=   *DUMP    <program list>

<punch card>      ::=   *PUNCH   <program list>

<patch card>      ::=   *PATCH   <program name>

<program list>    ::=   <program name>
                      / <program list>,  <program name>

<tape name>       ::=   <program name>

<rename list>     ::=   <rename part>
                      / <rename list>,  <rename part>

<rename part>     ::=   <program name>  <space>
                        TØ  <space>  <program name>

                      / <program name>  space  <program name>
```

Control Card Examples:

```
*DELETE PRØGA, PRØGB
*DELETGRADIT
*RENAME  BØY TØ MAN, PRØG14 TØ PRØG15, GRADIT TØ GRADE
*RENAME BØY MAN, PRØG14 PRØG15
*CØPY MØNITR, DLØDR, PLIST, SYSCL TØ SYSTAP
*CØPY LIBSER, LSUPDT TØ JUN15 WITH JUN01
*LØAD PLIST FRØM SYSTAP
*DUMP MØNITR, DLØDR, $FILE, TTF.05
*PUNCH MYPRØG
*PUNCH PLIST, SYSCL
*PATCH MØNITR
*PATCH SYSCL
*PATCH LIBSER
```

OUTPUT:

The reports produced by DUP vary according to the service(s) requested on the control record. Output associated with each service is detailed in the following

sections. However, it is important to note at this point that:

a) each complete control record is printed beginning at the top of a page.
b) the report associated with the first requested service begins immediately below the last line of the complete control record.
c) reports associated with subsequent services requested by the same control record each begin at the top of a page, except in the case of *CØPY or *LØAD processing.
d) each complete control record is also typed unless input is from the typewriter.

ERROR MESSAGES AND HALTS:

Aside from the standard system halts and messages, the following halts and messages can occur in any usage of DUP:

0. √√ xxxxxx NØT ØN DISK.
No halt accompanies this message. Reference has been made to a program not in the Operating System library.

1. √√ UNKNØWN CØNTRØL CARD.
No halt accompanies this message. It indicates that a control record did not start with one of the key words defined in the syntax. The message is both typed and printed.

2. √√ SYNTAX ERRØR IN CØNTRØL CARD.
No halt accompanies this message. It indicates that a control record did not conform to the form defined in the syntax. The message is both typed and printed.

3. √√ PRØGRAM xxxxxx IS DAMAGED.
Hard halt No. 405 accompanies this message. It indicates that the header, loader, and/or trailer do not contain the proper identifying information. The message is both typed and printed.

4. √√ MØRE THAN 6 CARDS IN A DUP CØNTRØL RECØRD.
No halt accompanies this message. It indicates that 7 consecutive cards were read which appear to be part of the same control record (see the previous section on the structure of control records). The message is both typed and printed.

Halts and messages peculiar to particular services are documented in the following sections.

OPERATING PROCEDURES:

DUP is designed to operate as automatically as possible, with a minimum of operator intervention and no operator decisions. Aside from job set-up as specified in a previous paragraph, and action on errors which indicate a failure in the operating system, no special operating procedures need be specified.

Should message 3 occur in any operation of DUP, core and disk dumps should be taken, and a member of the Computer Systems group notified, as in the case of standard system halts and messages.

Messages 0, 1, 2 and 4 are the concern of the user.

End-Of-Job, Post-Processing:

DUP may either perform an EXIT, returning to the inter-job supervisor, or LINK to some other program at the completion of its processing. The manner in which this choice is made is described in the Usage paragraph under Program Common.

*DELETE

PURPOSE:

The program deletion service of DUP provides a means of removing program blocks from the 1401 Operating System Library. This may be preparatory to reloading, or because the programs are in error, or no longer useful.

RESTRICTIONS:

As noted below (see Processing/Method), certain types of program blocks cannot be deleted.

If any part or phase of DUP, namely program blocks DUP, DUPA, DUPB, DUPC, DUPD, or DUPE, is to be deleted, all phases must be deleted. DUPC should be the last program block named, and the control record should end in a period followed by a space.

Time/Core Requirements:

The time required to delete one program varies from two to ten seconds, depending on its length and its neighboring programs.

Core requirement is as described for DUP (General).

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

Set-up is as required for DUP (General).

Processing/Method:

The following describes the deletion of one program:

The program table is searched for the program name. If the program is not present, a diagnostic is produced. If the program is present, but is

a) a move-mode file,
b) a headerless load-mode block,
or c) a fixed-location (system disk) program,

a diagnostic is produced. If the program block is not

The header and trailer of the program are checked. If either is in error, a diagnostic is produced and the system comes to a hard halt. Otherwise,

a) the program table entry is removed.
b) if there are floating-location program blocks above the deleted block on the cylinder, they are moved down.
c) an amount of disk equal to the size of the program is cleared.
d) a log entry is made.
e) the verification message is typed and printed.
f) the parameter and usage reports are printed.
g) the service is completed.

USAGE:

Calling Sequence:

The program deletion service of DUP may be called by a control record beginning with *DELETE or *DELET, by a call to DUP with position 1 of the program common field containing a minus sign, or by the *LØAD option. It can be called implicitly by the DLØDR program (KC06A) with disk input.

Control Cards:

The *DELETE or *DELET control record, which initiates this service, is described in the Control Record Syntax section of DUP (General). No other control cards are required.

Input:

*DELETE requires no input.

OUTPUT:

The output which normally results from the deletion of one program consists of:

a) a message, both typed and printed, verifying the deletion.
b) a report on the program parameters from the DLØAD process for the program.
c) a report on the usage statistics of the program.

ERROR MESSAGES and HALTS:

In addition to the standard system halts and messages, and those messages described in the section on DUP (General), the following messages may occur on the typewriter and printer:

5. >> DK DELETED xxxxxx sssss nnn
   This is the normal verification message, where xxxxxx is the program name, sssss is the sector address its header had occupied, and nnn is the number of sectors it had occupied.

6. √√ xxxxxx CANNØT BE DELETED
   This indicates that the program block is either a file, or at a fixed location, and therefore cannot be deleted.

OPERATING PROCEDURES:

Operation during a program deletion should be completely automatic. Both the messages in the above paragraph are the concern of the user. Attention should also be given the operating procedures described in DUP (General).

End-Of-Job, Post-Processing:

After each requested deletion has been attempted, control is returned to mainline DUP. See the End-Of-Job, Post-Processing paragraph of DUP (General).

*RENAME

PURPOSE:

The program renaming service of DUP provides a means of changing the names of program blocks stored in the 1401 Operating System Library. This may be done to avoid name conflicts in the system.

RESTRICTIONS:

As noted below (Processing/Method), certain types of programs cannot be renamed.

Time/Core Requirements:

The time required to rename a program is less than two seconds. Core requirements are as described for DUP (General).

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

Set-up is as required for DUP (General).

Processing/Method:

The program table is searched for the new name. If a program by that name is already present, a diagnostic is produced. If not, the program table is searched for the old name. If the program is not present, a diagnostic is produced. If the program is present, but is either

a) a move mode file,
b) a headerless load mode block,
or c) a fixed-location (system disk) program,

a diagnostic is produced. If the program block is not of the above types, it is a candidate for renaming.

The header and trailer of the program are checked. If it is in error, a diagnostic is produced and the system comes to a hard halt. Otherwise,

a) the program table is updated with the new name.

b) the header and trailer are updated with the new name.
c) a log entry is made.
d) the verification message is typed and printed.
e) the parameter and usage reports are printed.
f) the service is complete.

USAGE:

Calling Sequence:

The program renaming service of DUP is called by a control record beginning with *RENAME. It can be called implicitly by the DLØDR program (KC06A) with disk input.

Control Cards:

The *RENAME control record initiating the renaming service is described in the paragraph on Control Record Syntax of DUP (General). No other control cards are required.

Input:

*RENAME requires no input.

OUTPUT:

The output which normally results from renaming one program consists of:

a) a message, both typed and printed, verifying the renaming.
b) a report on the program parameters from the DLØAD process for the program.
c) a report on the usage statistics of the program.

ERROR MESSAGES AND HALTS:

In addition to the standard system halts and messages, and those messages described in the section on DUP (General), the following messages may occur on the typewriter and printer:

7. √√ xxxxxx CANNØT BE RENAMED
   This indicates that the program block is either a file, or at a fixed location, and therefore cannot be renamed.

8. √√ THERE ALREADY IS A xxxxxx
   This indicates that the "new name" is already used for another program on the library.

9. >> DK RENAMED xxxxxx TØ yyyyyy sssss nnn
   This is the normal verification message, where xxxxxx is the "old name", yyyyyy is the "new name", sssss is the sector address its header occupies, and nnn is the number of sectors in the program block.

OPERATING PROCEDURES:

Operation during a program renaming should be completely automatic. All the messages in the above paragraph are the concern of the user. See the Operating Procedures for DUP (General).

End-Of-Job, Post-Processing:

After each requested renaming has been attempted, control is returned to mainline DUP. See the End-Of-Job, Post-Processing paragraph of DUP (General).

*CØPY

PURPOSE:

The program copying service of DUP provides a means of copying programs stored in the 1401 Operating System library to magnetic tape, in a format suitable for input to the *LØAD service of DUP. This may be done to "back up" a program against possible removal, modification, or replacement, either by other users or by setbacks of the system due to use of the mass restoration program RESTR (ZD02A). It may also be used to provide an "auxiliary library" capability, wherein programs which have been little used and/or will not be used for some time in the future can be copied to tape, and then deleted to free disk space for more constructive uses.

RESTRICTIONS:

The following program blocks may not be copied to tape:

$PRGTB	DUPB
$LØG	DUPC
DUP	DUPD
DUPA	DUPE

Time/Core Requirements:

The time required for the program copying service varies according to the number of programs to be copied, and the number of programs on the input tape (if the 'WITH tape name' option is used). In any case, use of the program copying service should be less than ten minutes. Copying proceeds at nearly full tape speed.

Core requirement is as described for DUP (General).

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

An output tape should be mounted on tape drive 2. A mini-reel will often suffice for this purpose. If

the "WITH tape-name" option is to be used (tape update), the proper previously *CØPYed tape should be mounted on tape drive 1 as input. Also see the Set-Up, Previous Processing paragraph of DUP (General).

Processing/Method:

A full description of the *CØPY operation appears below, in the paragraph titled Processing Method -- Technical Description.

USAGE:

Calling Sequence:

The program copying service of DUP is called by a control record beginning with *CØPY.

Control Cards:

The two forms of the *CØPY control record which initiate this service are described in the Control Record Syntax section of DUP (General). No other control cards are required.

Input:

Tape input is required if the "WITH tape name" form of the control record is used. The tape, on drive 1, should have been previously produced by the *CØPY service of DUP. Its format should be identical to that described in the following paragraph on output.

OUTPUT:

Tape Output:

The main output from use of the program copying service of DUP consists of a tape containing:

a) header information consisting of an IØCS header and a DUP tape-name header.
b) a program table identifying the programs on the tape. These may be program blocks that were specified for copying, or carryover program blocks if the "WITH tape name" form of control record was used.

c) a set of records for each program block which was carried over from a previous tape due to use of the "WITH tape name" form of control card.
d) a set of records for each program block which was specified for copying.
e) a DUP trailer and IØCS trailer.

The format of the IØCS header is type "B" (80-positions), with identification DUP*CØPY and a negative retention cycle. The format of the DUP tape-name header is a 110-position BCD load-mode record with the tape name in the first 6 positions.

The format of the program table is one or more 110-position BCD load-mode records, each containing zero to seven 15-character entries. The last entry is followed by a record mark. The 15-character entry consists of the name, beginning sector address at time of copying, and number of sectors, as in the disk program table.

The set of records corresponding to a program block consists of a "program block header" having the characters $* in the first two positions, followed by an M or L denoting move- or load-mode, followed by a 15-character entry as described above. The program block header is a 110-character binary move-mode record. Following the program block header are records containing the sequential disk images of the program, one sector to a record. Records containing the header, loader, and trailer are included for programs.

For a load-mode file or program, there are two records per sector:

a) a 110-position BCD load-mode record
pos. 1-3) sequence number
pos. 4-10) blank
pos. 11-100) word mark skeleton of the sector, over blanks
pos. 101-110 blank

b) a 110-position binary move-mode record
pos. 1-3) sequence number
pos. 4-10) blank
pos. 11-100) data of the sector, without word marks
pos. 101-110) blank

At the end of a load-mode program block, one 110-position BCD load-mode record is written having $* in positions 1-2.

For a move-mode file, there is one record for each sector:

a) a 110-position binary move-mode record
pos. 1-3) sequence number
pos. 4-10) blank
pos. 11-110) data of the sector

The DUP trailer record contains $*≠ in positions 1-3. It is a 110-position binary move-mode record.

The IØCS trailer is standard "B" format, with a record-count kept, and with tape-marks before and after.

Printed Output:

In addition to the output tape, a message is typed and printed for each program written on the output tape, whether it be passed along from an input tape or copied from disk. The typed output is in the form of a "tape packing list" which should be inserted in the tape case when the output tape is dismounted.

ERROR MESSAGES AND HALTS:

Four types of halts and messages may occur during use of the program copying service:

a) the system standard halts and messages.
b) IØCS halts, described in IBM publication C24-3298 (Input/Output Control System - Disk, Operating Procedures. IBM File Number 1401/1460-30).
c) the halts and messages described in DUP (General).
d) halts and messages specific to the program copying service.

Only halts and messages of type d) will be discussed here:

10. √√ BAD RECØRD ØR MISSING RECØRD, NØ. nnn IN xxxxxx
This message indicates that the input tape mounted is probably unusable.

11. √√ TØØ MANY PRØGRAMS
This message should not occur, but if it does it indicates that the number of program names in the control record, plus those already on the tape if the "WITH tape-name" option is used, is too great for the capacity of the system.

12. √√ xxxxxx IS NØT ØN DRIVE 1
This message indicates that the input tape mounted was produced by the program copying service, but is not the one specified in the control record after the word 'WITH'.

13. √√ ERRØR IN TAPE STRUCTURE
This message indicates that the input tape mounted is probably unusable. Hard halt 222 accompanies this message.

14. >> xxxxxx CØPIED FRØM yyyyyy
This message can occur only if the 'WITH tape-name' option is used, and is the normal verification of the copying of a program block from the old tape to the new tape.

15. >> DK CØPIED xxxxxx sssss nnn
This is the normal verification message for the successful copying of one program block from disk to tape.

16. >> TAPE PACKING LIST
>> nn-nn PM, day, month/date, year
>> DUP TAPE NAME IS yyyyyy
This is the normal header of the list of programs on the output tape.

OPERATING PROCEDURES:

Set-up is as noted above. In normal operation the procedure would merely be to dismount the tape when it rewinds, and insert the tape packing list in the case with the tape. Also consult the Operating Procedures in DUP (General).

If message 10 or 13 occurs, the copying attempt should be abandoned. Message 11 is the concern of the user. Message 12 requires the mounting of the correct tape: the operator should consult the tape packing lists of the tapes available to him. Messages 14 through 16 are normal.

End-Of-Job, Post-Processing:

After copying all requested program blocks, control is returned to mainline DUP. The operator should dismount the tape and insert the tape packing list in the case with the tape. Also consult the End-Of-Job, Post-Processing paragraph of DUP (General).

PROCESSING/METHOD -- TECHNICAL DESCRIPTION

The complete control record is scanned and a table of program blocks to be copied is built. At this time, duplicate names on the control record, if any, and certain uncopyable program names, are eliminated from the table. For a list of these, see Restrictions. Program blocks specified for copying which are not on disk result in a diagnostic at this time.

If the 'WITH tape-name' option is used on the control record, the tape on drive 1 is opened and checked for identification. Either an IØCS halt or a typed diagnostic can result at this point if the identification check fails.

With the 'WITH tape-name' option, the following sequence then occurs:

a) the output tape is opened and the DUP header is written.
b) a comparison of the program names to be copied and the program table of the old tape yields a table of program blocks to be carried over. If a program name is already represented on the old tape, and is specified for copying, the old version is ignored and the present version on disk is copied onto the output tape.
c) the program table from the old tape is copied to the output tape, and augmented with program names to be newly copied.
d) program blocks on the old tape are read, and either passed or copied to the output tape as determined in step b) above, messages being produced for blocks which are copied.
e) when the DUP trailer record is encountered, one more record is read to accomplish the IØCS end-of-file processing, including record-count checking, and the old tape is closed.
f) operation proceeds as in step c) of the following description.

Without the 'WITH tape name' option,

a) the output tape is opened and the DUP header written.
b) the program table is constructed and written.
c) program blocks specified for copying are read from disk and written, messages and log entries being produced for each.
d) the DUP trailer is written and the output file is closed.
e) the typewriter sheet is spaced up to isolate the tape packing list.
f) the service is completed, and control is returned to mainline DUP.

*LØAD

PURPOSE:

The program loading service of DUP provides a means of loading programs, previously copied to magnetic tape by the program copying service, back onto the 1401 Operating System library. This service is not to be confused with the original loading service afforded by the DLØAD macro or deck and the DLØDR program (KC06A).

RESTRICTIONS:

Time, Core Requirements:

The time required for the program loading service varies according to the number of programs to be loaded and the number of programs on the input tape. In any case, use of the program loading service should be less than ten minutes. Loading proceeds at nearly full tape speed.

Core requirement is as described for DUP (General).

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

The tape named on the control record must exist, and must be a tape previously created by the program copying service of DUP. This tape should be mounted on drive 1. Also see the paragraph on Set-Up, Previous Processing for DUP (General).

Processing/Method:

A detailed description of the processing method will be found below in the paragraph on Processing Method - Technical Description.

USAGE:

Calling Sequence:

The program copying service of DUP is called by a control card beginning with *LØAD.

Control Cards:

The format of the *LØAD control record initiating the program loading service is described in the paragraph on Control Record Syntax of DUP (General). No other control cards are required.

Input:

The format of the input tape required on drive 1 is described in the paragraph on Tape Output of the DUP (*CØPY) section of this document. No other input is required.

OUTPUT:

On completion of the loading service, the format of the program is identical with the format at the time it was copied to tape. However, its location in the library may be different.

Messages are typed and printed for each program loaded from the tape. Diagnostics concerning absence of a requested program from the tape and output from the program deletion service may precede this normal *LØAD output.

ERROR MESSAGES AND HALTS:

Five types of halts and messages may occur during performance of the program loading service:

a) the system standard halts and messages,
b) IØCS halts, as described in IBM publication C24-3298 (Input-Output Control System - Disk, Operating Procedures. IBM File No. 1401/1460-30),
c) the halts and messages described in the DUP (General) section,
d) messages 10-13 described in the DUP (*CØPY) section,
e) halts and messages specific to the program loading service.

Action for messages 10-13 described in the DUP (*CØPY) section should be as described in that section.

Only halts and messages of type e) will be discussed here:

17. √√ xxxxxx NØT ØN TAPE
This message indicates that program block xxxxxx was specified for copying but is not on the input tape.

18. >> DK LØADED xxxxxx sssss nnn
This message is the verification message for the successful loading of one program block. It can be distinguished from the similar message put out by the DLØDR program (KC06A) in that the latter is followed by a period.

19. √√ xxxxxx CØNFLICTS WITH PERMANENT DISK BLØCK
This message indicates that conflict with an already-existant program block on disk has been discovered, either in name or disk location. See the paragraph on Processing Method -- Technical Description.

20. √√ INSUFFICIENT DISK FØR xxxxxx
This message indicates that a serious disk crowding problem has developed, and available space large enough for program block xxxxxx cannot be found.

OPERATING PROCEDURES:

Set-up is as noted in a previous paragraph. In normal operation the procedure would merely be to dismount the tape when it rewinds, and make sure that the tape packing list remains in the tape case. Notice should also be paid to the operating procedures in the section on DUP (General).

Action on messages 10 through 13 are as described in the section on DUP (*CØPY). Messages 17 and 19 are the problem of the user. Message 20 indicates a serious disk crowding problem, and action should be taken according to current UWMCC policy. Message 18 is normal.

End-Of-Job, Post-Processing:

When all requested program blocks have been loaded, control is returned to mainline DUP. The operator should dismount the tape, making sure that the tape packing list remains in the tape case. Also see the End-Of-Job, Post-Processing paragraph of DUP (General).

PROCESSING METHOD - TECHNICAL DESCRIPTION:

The complete control record is scanned and a table of program names to be loaded is constructed. At this time duplicate names on the control record, if any, are eliminated.

The input tape on drive 1 is opened and checked for identification. Either an IØCS halt or a typed diagnostic may result at this point if the identification check fails.

A table of the names to be deleted is constructed. This table contains all names which are:

a) specified for loading, and
b) on the tape, and
c) in the 1401 Operating System library.

If a program block is already on disk it will be replaced by the version on the tape.

In order to determine the above, the program table on the tape is scanned. When this is completed, diagnostics may be produced indicating program(s) specified for loading which are not on the input tape.

At this point, if there are programs to be deleted, the program deletion service of DUP is called. After deleting the one or more programs necessary and producing its normal output, the deletion phase returns control to the program loading service. On such a return, the input tape must be re-opened (hence rewound), and repositioned to its former status.

Each program block on the tape is then handled in the following fashion:

a) If the block has not been specified for loading, it is passed and the next block considered.

b) If a move-mode file or headerless load-mode file is duplicated, a diagnostic is produced and the block passed unless the duplicates start at the same disk location and are of the same length.
c) If a fixed location program is duplicated, a diagnostic is produced and the block passed unless the duplicates start at the same disk location and there is sufficient space for the tape version.
d) If there is no duplication, and the program is not at a fixed location, an unused disk area which is the smallest that will accomodate the program block is chosen. If there is no such area, a diagnostic is produced.
e) If there is no duplication, and the program block is at a fixed location, a diagnostic is produced unless sufficient unused area starting at that location is found.
f) In the cases with no duplication, an entry is inserted in the Operating System program table.
g) The block is copied to disk at the appropriate location, sector by sector.
h) The loading verification message is typed and printed, and a log entry is made.
i) The service of loading one program is complete.

*DUMP

PURPOSE:

The program printing service of DUP provides a means of obtaining a "core dump" of a disk-stored program block as it will appear when loaded into core but not yet executed (i.e. in "disk image"). The service prints all types of program blocks that can exist under the 1401 Operating System -- blocks with or without headers, in either mode, etc.

RESTRICTIONS:

Time, Core Requirements:

Printing proceeds at approximately 150 one-hundred character segments per minute. The time to print one program varies according to its length.

Core requirement is as described for DUP (General).

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

Paper at least 12 inches wide must be on the 1403 printer in order to use the program printing service. Also see the Set-Up, Previous Processing paragraph of DUP (General).

Processing/Method:

Each one hundred characters is read from disk by a "virtual core" routine. The annotation line for each is then constructed, and substitution characters inserted, by scanning the block read from disk. On the first line of the program block, the left end of annotation and data lines is cleared; on the last line the right end is cleared. The annotation line is printed, the data is loaded into the print area, the data and wordmark lines are printed, and the printer is spaced one line. This recurs until the last line is printed.

USAGE:

Calling Sequence:

The program printing service of DUP is called by a control record beginning with *DUMP. It is also called at completion of service by the program modification service.

Control Cards:

The *DUMP control record, which initiates this service, is described in the paragraph on Control Record Syntax of DUP (General). No other control cards are required.

Input:

*DUMP requires no input.

OUTPUT:

The output from the printing of one program consists of:

a) the program parameter information produced at DLØAD time.
b) the usage information accumulated since.
c) a number of annotated lines, each representing 100 characters of object program, sufficient to print the entire program.

For a non-program block, only lines of type c) are printed, preceded by an indication of name, location, length, and mode.

The first and last lines of type c) may be only partial lines so that the character corresponding to the load-to address is the first position represented in the print-out, and the character just before the group-mark word-mark address is the last character represented. Each such line begins at the XXX01 position and ends with the succeeding XXX00 position, and is accompanied by an annotation line giving the relative positions

in the line. In the annotation line may appear substitution characters giving notice of unprintable or ambiguous characters in the line below, as follows:

1402 Card Code	1401 Internal	1401 Name	Substitution Character
none	A	Cent sign	A
0-6-8	CA842	Backward slash	B
5-8	841	Colon	C
11-7-8	B8421	Delta	D
11-0	B82	Exclamation point	E
12-7-8	CBA8421	Group mark	G
12-5-8	BA841	Left bracket	L
0-7-8	A8421	Tape segment mark	M
12-0	CBA82	Question mark	Q
11-5-8	CB841	Right bracket	R
11-6-8	CB842	Semicolon	S
7-8	C8421	Tape mark	T
0-5-8	CA841	Word separator	W
12-6-8	BA842	Less than	Y
6-8	842	Greater than	Z

ERROR MESSAGES AND HALTS:

Aside from normal output, no halts or messages besides the system standard halts and messages and the halts and messages described in the DUP (General) section can occur.

OPERATING PROCEDURES:

Set up is as noted in a previous paragraph. Operation of the program printing service should be completely automatic. See the paragraph on Operating Procedures for DUP (General).

End-Of-Job, Post-Processing:

When all requested program blocks have been printed, control is returned to mainline DUP. See the End-Of-Job, Post-Processing paragraph of DUP (General).

*PUNCH

PURPOSE:

The program punching service of DUP provides a means of obtaining card object decks of programs stored in the 1401 Operating System Library.

RESTRICTIONS:

Move-mode files and headerless load-mode blocks cannot be punched.

Time, Core Requirements:

The time required to punch one program varies according to the length of the program. Punching proceeds at approximately 200 cards per minute unless large numbers of blank output cards are being bypassed.

Core requirement is as described for DUP (General).

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

A sufficient number of cards must be readied in the 1402 punch hopper. See the Set-Up, Previous Processing paragraph of DUP (General).

Processing/Method:

A "virtual core routine" is used to read the area of disk containing the data to be punched on a given card. Data is then moved to the card, character by character, and loading instructions are created, until one of the following occurs:

a) 39 characters have been moved to the card.
b) a word-marked character is encountered and no more set-word-mark loading instructions are available.
c) the last character before the group-mark word-mark address has been moved.

The card is then punched, unless the characters to be loaded are all blanks and the loading instructions set no word-marks.

After an entire deck is punched, a log entry is made and a verification message is typed and printed.

USAGE:

Calling Sequence:

The program punching service of DUP is called by a control card beginning with *PUNCH.

Control Cards:

The *PUNCH control record initiating the service is described in the paragraph on Control Record Syntax of DUP (General). No other control cards are required.

Input:

*PUNCH requires no input.

OUTPUT:

Decks are punched in self-loading format, with object characters in columns 1-39 and loading instructions in columns 40-71. Columns 72-75 are used for a sequence number, increasing by one for each card. Columns 76-80 contain as identification the first 5 characters of the name of the program.

The deck begins with 16K core-clear cards followed by a "bootstrap" card and the object cards. The deck ends with an execute card which transfers control to the execution address of the program block.

The deck produced is virtually a "disk image" of the program as it is stored in the Operating System library, except that cards which would load 39 blank characters without word-marks are suppressed, since the core-clear cards will perform this function. Because of this, the following differences may exist between the deck punched and a deck used to DLØAD the program:

a) only data between the load-to address and the group-mark word-mark address will be represented in the deck punched.
b) any "execute" sequences in the original deck will not be reflected in the deck punched.
c) cards produced by the ",C" option of a DA will not be reflected in the deck punched.
d) other DA options will be handled differently.

ERROR MESSAGES AND HALTS:

In addition to the standard system halts and messages, and those messages described in the section on DUP (General), the following messages may occur on the typewriter and printer during use of the program punching service:

21. >> DK PUNCHED xxxxxx sssss nnn
This is the normal verification message for the successful punching of a program. xxxxxx is the program name, sssss is its starting sector, and nnn is the number of sectors in the program.

22. √√ xxxxxx CANNØT BE PUNCHED
This message indicates that the name of a non-program block was found in a *PUNCH control record.

23. >> nnn CARDS WERE PUNCHED
This message follows message 21 on the printer.

OPERATING PROCEDURES:

Operation during the punching of one or more programs should be completely automatic. The operator should keep track of where each deck ends if more than one is punched. An extra blank card is punched after each program, so that the last card need not be run out.

All the above messages are the concern of the user. See the paragraph on Operating Procedures for DUP (General).

End-Of-Job, Post-Processing:

On completion of the processing of all programs for which the service was requested, control is returned to mainline DUP. See the End-Of-Job, Post-Processing paragraph of DUP (General).

*PATCH

PURPOSE:

The program modification service of DUP provides a means by which programs stored on the 1401 Operating System Library may be updated, changed, and corrected by means of standard-format "patch cards", without recourse to the original deck from which the program was DLØADed. Patch cards may either be in Autocoder self-loading or condensed format.

RESTRICTIONS:

Move-mode files and headerless lode-mode blocks cannot be patched. The program modification service is not designed for the complete replacement of programs by means of decks produced by the AUTØCØDER assembler. For example, cards produced by DA statements using the ",C" option cause format diagnostics. As a result, the area may not be cleared. In addition, the size of the program may be different than that of the disk-stored version. Use the program deletion service and the DLØAD macro or deck with the DLØDR program (KC06A) to perform complete replacement.

Time, Core Requirements:

The program modification service reads patch cards from the 1402 reader at approximately 100 cards per minute. The time required for use of the service varies according to the number of patch cards. See the Time, Core Requirements paragraph of the *DUMP section of this document.

Core requirements are as described for DUP (General).

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

The patch cards must be prepared by the user in the correct format. See the Set-Up, Previous Processing paragraphs in the DUP (General) and DUP (*DUMP) sections of this document.

Processing/Method:

A full description of *PATCH operation is found below, in the paragraph titled Processing Method -- Technical Description.

USAGE:

Calling Sequence:

Use of the program modification service of DUP is initiated by a control record beginning with *PATCH.

Control Cards:

The *PATCH control record initiating the program modification service is described in the Control Record Syntax section of DUP (General). Note that, unlike other DUP control records, only one program name may appear in a *PATCH control record. As with other DUP control records, if the record ends with a period followed by a space, the DUP end-of-job occurs after the service is performed (i.e., all patch cards are read).

Input:

Patch Cards:

Patch cards may be in either self-loading or condensed format, in any mixture. A card is judged self-loading if:

a) columns 68-71 contain 1040
b) columns 40, 47, 54, and 61 each contain one of these characters: M, L, □ , N, or ',' (comma).
c) if all of the address fields of the instructions having op codes as in b) are valid 1401 addresses.

A card is judged to be in condensed format if:

a) columns 1-3 contain a valid 1401 address
b) columns 4-5 contain numeric digits less than 67 in value
c) the columns beyond the number of characters indicated in 4-5 are blank.

Diagnostics are produced if neither set of conditions above is satisfied, and also if the card is "out of bounds", as described below in the paragraph on Processing Method -- Technical Description.

Trailer Card:

Following all patch cards must be a card which is completely blank in at least its first 41 columns. Other DUP control cards may follow this card directly.

OUTPUT:

Normal output from use of the program modification service includes an 80-80 listing of the patch cards (diagnostics may appear interspersed in this listing), followed by the normal output of the program printing service for the same program, namely the program parameters, usage statistics, and "core dump" of the program.

ERROR MESSAGES AND HALTS:

In addition to the standard system halts and messages and the halts and messages noted in the DUP (General) and DUP (*DUMP) sections, the following messages may occur:

24. √√ xxxxxx CANNØT BE PATCHED
This message indicates that an attempt has been made to patch a non-program entry.

25. √√ NØT IN PATCH FØRMAT
This message indicates that the patch card above it is not recognizable as in self-loading or condensed-loading format, as defined above in the paragraph on Patch Cards.

26. √√ ØUT ØF BØUNDS
This message indicates that the patch card above it does not satisfy the bounds criteria described below in the paragraph on Processing Method -- Technical Description.

27. >> DK PATCHED xxxxxx sssss nnn
This is the normal verification message at the completion of the program modification service, where xxxxxx is the program name, sssss is its first sector, and nnn is the number of sectors in the program.

OPERATING PROCEDURES:

No special operating procedures are necessary for use of the program modification service. All of the above messages are the concern of the user. If message 24 or DUP (General) messages 0 or 2 occur, the patch cards are automatically run through the reader. If DUP (General) message 1 occurs on what appears to be a *PATCH control record, the operator must remove the patch cards from the reader manually. See the Operating Procedures paragraph for DUP (General).

End-Of-Job, Post-Processing:

When the trailer card is read: a log entry is made; a verification message is typed and printed; and the program printing service is called. When that service is completed, control is returned to mainline DUP. See the End-Of-Job, Post-Processing paragraph for DUP (General).

PROCESSING METHOD -- TECHNICAL DESCRIPTION:

Input is read through the same I/O routine as control records: thus "patch cards" may be entered from the typewriter, the card reader, or disk: in any case, the same input device as control cards. It is permissible to terminate the entry of patch cards with an end-of-file on the device in question rather than a blank card as cited above. The card is first printed, then checked for self-loading validity according to the standards above. If this check fails, the card is checked for condensed-loading validity. If this check also fails, a diagnostic is produced.

For both formats a check is performed to ensure that the patch represented on the card is within the bounds of the program. A diagnostic is produced for a self-loading card if any of the following conditions fail:

a) the lowest reference in the loading instructions (excluding references below 81) must be at or above the load-to address of the program.
b) the highest reference in the loading instructions must be below the group-mark word-mark address.
c) the difference between the lowest reference (excluding references below 81) and the highest reference must be less than 90.

A diagnostic is produced for a condensed-loading card if either of the following conditions fail:

a) the address in positions 1-3 must be at or above the load-to address of the program.
b) the address formed by adding positions 4-5 to positions 1-3 (the highest reference) must be below the group-mark word-mark address of the program.

The actual patching operation is done through a "virtual core" routine which reads the appropriate part of the program from disk. The patch is made, and the segment is rewritten to disk. The next card is then read and processed in the same manner, unless it is the trailer card.

PROGRAM:	SYSCL
MACHINE:	IBM 1401
LANGUAGE:	AUTØCØDER
SUPERVISOR PROGRAM:	none
PROGRAMMER:	Systems Division
DATE COMPLETED:	July 1, 1968

PURPOSE:

SYSCL calls the IBM system and starts the operation of that system, leading to a Fortran or CØBØL compilation, AUTØCØDER assembly, or RPG generation. It duplicates the operation of the 14-card IBM system boot, with certain added capabilities.

RESTRICTIONS:

Time, Core Requirements:

Execution of SYSCL itself takes less than 3 seconds, which is followed by the processing time of the relevant IBM system.

SYSCL is a mainline program, effectively using all of core during its brief execution.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

No previous processing is necessary for use of SYSCL. The IBM system pack should be on the opposite drive from the Operating System pack, except in the case when one drive is inoperative. 14 inch wide paper is normally used in the 1403 printer for assemblies and compilations.

Processing/Method:

A detailed description of the method of operation will be found below, in the paragraph titled Processing Method -- Technical Description.

USAGE:

Calling Sequence:

Source of Call:

SYSCL is usually called by a log-start card, but may be called by means of the inter-job entry code DØ, or by a LINK macro from an AUTØCØDER or CØBØL program.

Program Common:

The program common field used to call SYSCL determines the "assumed assignments" of various files used by the IBM system. To get file assignments suitable for Fortran (or CØBØL) compilation, the word FØRTRAN or CØBØL should be contained in the program common field, left justified. If the program common field contains any other letters or is blank, the assumed file assignments will be those most suitable for Autocoder assembly.

Control Cards:

A SYSTEM ASGN card is not needed when SYSCL is used. Other than that, control cards are as described in the relevant IBM publications. The need for special ASGN cards for Fortran or CØBØL compilation is largely obviated by the program common conventions described above.

Input:

Input is as described in the relevant IBM publications.

OUTPUT:

Output is as described in the relevant IBM publications.

ERROR MESSAGES AND HALTS:

Aside from the standard system disk halts and messages, the following may be produced during use of SYSCL, all on the console typewriter:

1. TYPE WHETHER DRIVE n IS AVAILABLE
   followed by a wait-for-console loop.
   In this message, n = '0' references module 0,
   n = '2' references module 1.

2. TURN ØN DRIVE n
   followed by a halt with A-register blank.
   n as in message 1.

3. MØUNT IBM PACK ØN DRIVE n
   followed by a halt with A-register blank.
   n as in message 1.

4. IN THAT CASE ØNLY SMALL AUTØCØDER JØBS MAY BE RUN.

The halts and messages documented in the IBM system publications should also be noted.

OPERATING PROCEDURES:

If message 2 occurs, the operator should turn on the indicated drive, and press START when the drive has come to ready status. If the indicated drive is inoperative, the operator should simply press START, whereupon message 1 is typed. The operator should enter 'NO' at this point. Any other response causes the program to revert to message 2. If 'NO' is entered, message 4 is typed, and the job is aborted if the program common field contains indication of Fortran, CØBØL, or Tabtran activity.

When the drive is selected, message 3 may occur, in which case the operator should mount the IBM system pack, and press START when the drive has come to ready status.

The operating procedures documented in the IBM system publications should also be noted.

End-Of-Job, Post-Processing:

If the last card indicator is on when the IBM system completes an assembly or compilation, control is transferred back to the Operating System with an EXIT.

The IBM system recognizes the Version 2 log-start card (but does not recognize the Version 1 card), and performs a simulated LØAD upon it if encountered. No post-processing is required after use of SYSCL and the IBM system, but it is common for an object deck to be loaded immediately for testing or loading to disk.

PROCESSING METHOD -- TECHNICAL DESCRIPTION:

Choice of Drive:

The log-start card is selected to reader pocket one. The Operating System communication sector is read, and the job number and man number are saved. The IBM system drive is at first assumed to be the opposite drive from the Operating System. If that drive is not on or does not contain the IBM system, the operator is instructed to mount the proper pack. If a not-on condition persists a second time, the operator is given an option to select a one-drive configuration (other drive not operable).

Transfer to the IBM System:

Whichever drive is selected, processing continues as follows:

1. The IBM permanent file-hardware table is read, the WØRK1, 2, and 3 files and the LIBRARY, LØADER, CØRELØAD and INPUT files are given their assumed assignments, as controlled by the program common field. The temporary file-hardware table is written.

2. The IBM phase index table is searched for an entry for phase 8ØP. If an entry for 8ØP is found, and 8ØP is patched to punch a log-start card, a log-start card image with the appropriate job number, man number, and program-name LØAD (all other fields blank) is written into 8ØP.

3. The IBM I/O package is read.

4. Various parameters are moved into the first 4K to simulate the results of executing the card boot and the system boot.

5. The IBM Determiner phase is read.

6. Core is cleared in the upper 12K.

7. If the program common field indicates that SYSCL was called by the TABTRAN processor, an appropriate AUTØCØDER RUN card is passed to the Determiner.

8. Control is transferred to the Determiner.

Assumed File Assignments:

The following file assignments are set up by SYSCL. The interested user should also refer to the section IBM SYSTEM CHANGES.

a. If both drives are operative, and the program common field contains the word FØRTRAN:

```
WØRK1      ASGN  1311 UNIT y, START 000800, END 003800
WØRK2      ASGN  1311 UNIT y, START 003800, END 006800
WØRK3      ASGN  1311 UNIT y, START 006800, END 008800
LIBRARY    ASGN  1311 UNIT x, START 012000, END 013900
LØADER     ASGN  1311 UNIT x, START 010000, END 012000
CØRELØAD   ASGN  ØMIT
INPUT      ASGN  READER 1
```

b. If both drives are operative, and the program common field contains the word CØBØL:

```
WØRK1      ASGN  1311 UNIT y, START 000800, END 006000
WØRK2      ASGN  1311 UNIT x, START 010000, END 012000
WØRK3      ASGN  1311 UNIT y, START 006000, END 008800
LIBRARY    ASGN  1311 UNIT x, START 013900, END 019980
LØADER     ASGN  ØMIT
CØRELØAD   ASGN  ØMIT
INPUT      ASGN  READER 1
```

c. If both drives are operative, and the program common field indicates SYSCL was called by the TABTRAN processor:

```
WØRK1      ASGN  1311 UNIT y, START 000800, END 006000
WØRK2      ASGN  1311 UNIT y, START 006000, END 007800
WØRK3      ASGN  1311 UNIT y, START 007800, END 008800
LIBRARY    ASGN  1311 UNIT x, START 013900, END 019980
LØADER     ASGN  ØMIT
CØRELØAD   ASGN  1311 UNIT y, START 008800, END 010000
INPUT      ASGN  1311 UNIT x, START 010000, END 012000
```

Assumed File Assignments: (con't)

d. If both drives are operative, and the program common field is of any other form or is blank:

```
WØRK1      ASGN  1311 UNIT y, START 000800, END 008000
WØRK2      ASGN  1311 UNIT x, START 010000, END 012000
WØRK3      ASGN  1311 UNIT y, START 008000, END 008800
LIBRARY    ASGN  1311 UNIT x, START 013900, END 019980
LØADER     ASGN  ØMIT
CØRELØAD   ASGN  ØMIT
INPUT      ASGN  READER 1
```

e. If only one drive is available:

```
WØRK1      ASGN  1311 UNIT x, START 010000, END 011300
WØRK2      ASGN  1311 UNIT x, START 011300, END 011800
WØRK3      ASGN  1311 UNIT x, START 011800, END 012000
LIBRARY    ASGN  1311 UNIT x, START 013900, END 019980
LØADER     ASGN  ØMIT
CØRELØAD   ASGN  ØMIT
INPUT      ASGN  READER 1
```

where: x is the IBM system drive
y is the Operating System drive.

PROGRAM:	TLØAD
MACHINE:	IBM 1401
LANGUAGE:	AUTØCØDER
SUPERVISOR PROGRAM:	1401 Operating System II
PROGRAMMER:	Systems Division
DATE COMPLETED:	July 1, 1968

PURPOSE:

TLØAD initiates loading of a tape-stored object program into 1401 core storage, simulating the function of the tape-load button on the 1401 C.P.U.

RESTRICTIONS:

The first record on the input tape must be a tape boot-strap.

Time, Core Requirements:

TLØAD execution time is less than one second. It is a mainline program, effectively using all of core during execution.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

The input tape should be mounted on tape unit 1. The first record must be a tape boot-strap.

Processing/Method:

TLØAD clears high core, reads the first record from the input tape, and branches to core position 1.

USAGE:

Calling Sequence:

TLØAD is normally called by a log-start card, but may be called by use of the inter-job entry code DØ, or by execution of a LINK macro from an AUTØCØDER or CØBØL program.

Control Cards:

TLØAD requires no control cards.

Input:

The only input consists of the input tape on tape unit 1.

OUTPUT:

TLØAD produces no output.

ERROR MESSAGES AND HALTS:

Aside from the standard system halts and messages, no halts or messages should occur during execution of TLØAD.

OPERATING PROCEDURES:

Set-up is as described in a previous paragraph. Operating procedures are as required by the object program to be loaded.

End-Of-Job, Post-Processing:

Control is transferred to the user's routine, which is read into core by the tape boot-strap.

PROGRAM:	LIST
MACHINE:	IBM 1401
LANGUAGE:	Autocoder
SUPERVISOR PROGRAM:	1401 Operating System II
PROGRAMMER:	Systems Division
DATE COMPLETED:	July 1, 1968

PURPOSE:

LIST provides 80-80 card-to-printer listing, with the following options:

double spacing
page numbering
card numbering
header line
column annotation
unprintable character annotation
horizontal displacement of card image
listing of log-start cards encountered
new page on special character in column 1

RESTRICTIONS:

Time, Core Requirements:

LIST will read and print at 600 cards/lines per minute, unless the A or S option is used, in which case the rate is 300 cards per minute or slightly less.

LIST is a mainline program, effectively using all of core during its execution.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

Paper at least 8 1/2 inches wide must be used in the 1403 printer, with 10 1/2 inch wide paper required if the T, P, or N option is used (see below, Program Common).

A standard 1-12 carriage tape of the correct length should be used.

No previous processing is required for use of LIST.

Processing/Method:

A detailed description of the method of operation will be found below in the paragraph titled Processing Method -- Technical Description.

USAGE:

Calling Sequence:

Source of Call:

LIST is normally called by a log-start card, but may also be called by use of the inter-job entry code DØ or by a LINK macro from an AUTØCØDER or CØBØL program.

Program Common:

The content of the program common field used to call LIST determines its operation to a large extent. One or more non-blank characters may be entered anywhere within the field, each character indicating an option as described below. If no such characters are used (program common field blank), LIST produces an 80-80 listing, single spaced, and executes any log-start card it encounters in the input deck.

Characters indicating options are:

A causes annotation of card positions in an extra line above each card image, consisting of a line of periods with positions 10, 20, ..., 80 noted explicitly. If this option is used together with the S option, special character annotation overlays the position annotation. Due to the extra line, this option causes the card-per-minute rate to drop to 300.

B causes cards to be read "binary" -- any card code is legal. No responsibility is taken for the character printed in a non-BCD position, since binary columns are treated as the 6-bit "hash" of the holes punched.

D causes the line images to be double-spaced. This skipped line is in addition to the extra line caused by use of the A or S option.

H causes the first card in the input deck (after any log-start card) to be read and stored. Thereafter, it is used as a "header" line at the top of each page, including the first.

L causes any log-start cards in the input deck to be listed rather than "trapped" and executed.

N causes a card number to be printed to the right of each card image.

P causes a page number to be printed at the top of each page. This is on the same line as, and to the right of, the header line (if the H option is used).

S causes annotation of ambiguous and unprintable characters, in an extra line above each card image. Characters appear above the position occupied by the unprintable or ambiguous character, as follows:

Card code	1401 Name	Annotation Character
0-6-8	Backward slash	B
5-8	Colon	C
11-7-8	Delta	D
11-0	Exclamation point	E
12-7-8	Group mark	G
12-5-8	Left bracket	L
0-7-8	Tape segment mark	M
12-0	Question mark	Q
11-5-8	Right bracket	R
11-6-8	Semicolon	S
7-8	Tape mark	T
0-5-8	Word separator	W
12-6-8	Less than	Y
6-8	Greater than	Z

If the S option is used together with the A option, special character annotation overlays the position annotation. Due to the extra line, and the considerable internal processing involved, this option causes the card-per-minute rate to drop to 300 or slightly below 300.

T causes the card images to be centered on a 14-inch-wide page, occupying printer positions 21-100 rather than 1-80. This distributes wear on the printer ribbon.

Any special character entered in the program common field causes LIST to start a new page on the listing for any card having that character in column 1.

Control Cards:

LIST requires no control cards.

Input:

Input consists of any deck which is to be listed. LIST will always return to the Inter-Job Supervisor (KE01A) on a last-card condition, or when a log-start card is read without the L option.

OUTPUT:

Output consists of 80-column card images printed on the 1403 printer, with output options as described in the paragraphs on Purpose, and Program Common.

ERROR MESSAGES AND HALTS:

Aside from the system standard halts and messages, no halts or messages should occur during use of LIST.

OPERATING PROCEDURES:

No special operating procedures are required.

End-Of-Job, Post-Processing:

LIST returns to the Inter-Job Supervisor on a last-card condition, or when a log-start card is read without the L option (see the paragraph on Program Common). No post-processing is required after use of LIST.

PROCESSING METHOD -- TECHNICAL DESCRIPTION:

The program common field is first scanned to determine the options, if any, to be used. Then, if the H option is used, the first card is read and stored. An initial

"carriage-restore sequence" is done, with header line and/or page number if these options are used. Then, for each card,

a) If the special-character option is used and the first column contains this character, a carriage-restore sequence is done.
b) If the A option is used, the position-annotation line is set up.
c) If the S option is used, the unprintable character annotation is set up.
d) If either the S or A option is used, the line set up previously is printed.
e) If the D option is used, a "double space after print" instruction is executed.
f) If the N option is used, the card count field is moved into the print line and incremented.
g) The card image is moved to the print line and the line is printed.
h) If a page overflow condition was encountered anywhere in the above operations, a carriage-restore sequence is done.

The above operations occur for each card until:

a) a log-start card is read and the L option is not being used, or
b) a last-card condition is encountered.

PROGRAM: MLIST
MACHINE: IBM 1401
LANGUAGE: AUTØCØDER
SUPERVISOR PROGRAM: 1401 Operating System II
PROGRAMMER: Systems Division
DATE COMPLETED: July 1, 1968

PURPOSE:

MLIST is a card-to-printer program, producing multiple listings of card decks.

RESTRICTIONS:

The maximum number of copies is 99.
The maximum deck size is 187 cards.

Time, Core Requirements:

MLIST prints at the rate of 600 lines per minute. It is a mainline program, effectively using all of core during execution.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

Paper at least 8 1/2 inches wide must be mounted on the 1403 printer.

Processing/Method:

The program common field is examined. If the number of copies is not present, the message "HØW MANY CØPIES" is typed. After the operator has entered the number of copies, the input file is read into core, until either a last-card or end-of-file condition is sensed (see below, End-Of-File Card). The printer carriage is restored, and the card file is listed the required number of times. A carriage-restore is performed before each new copy is printed.

USAGE:

Calling Sequence:

Source of Call:

MLIST is normally called by a log-start card, but may be called by use of the inter-job entry code DØ, or by execution of a LINK macro from an AUTØCØDER or CØBØL program.

Program Common:

The program common field should contain, right-justified, the number of copies to be generated. The maximum number of copies is 99.

Control Cards:

MLIST requires no control cards, unless other programs are in the job stream (batch mode). In that case, the End-of-File card is used.

End-Of-File Card:

This card is required when the call to MLIST is followed by a stack of decks. One End-of-File card must terminate the deck to be listed. The format is:

Col.	Content
1-4	4 record marks (0-2-8 multiple-punch)

Input:

Input consists of the deck to be listed, terminated by an End-of-File card.

OUTPUT:

Output consists of the 80-80 listing(s), and possibly the message "HØW MANY CØPIES".

ERROR MESSAGES AND HALTS:

Aside from the standard system halts and messages, only the following message may occur:

HØW MANY CØPIES

This message only appears if the number of copies is not present in the program common field.

OPERATING PROCEDURES:

Set-up is as described in a previous paragraph. If the message "HØW MANY CØPIES" occurs, the operator should enter this information from the console typewriter. The operator should be aware that MLIST always returns to the inter-job supervisor on a last-card indicator, after printing the previously-read deck.

End-Of-Job, Post-Processing:

MLIST returns control to the inter-job supervisor on a last-card indicator (or after reading a deck terminated by an end-of-file card), after printing the requested number of copies.

PROGRAM: REPRØ
MACHINE: IBM 1401
LANGUAGE: Autocoder
SUPERVISOR PROGRAM: 1401 Operating System II
PROGRAMMER: Systems Division
DATE COMPLETED: July 1, 1968

PURPOSE:

REPRØ provides general purpose 80-80 reproduction of card decks on the 1403 card punch. Any combination of punches may be reproduced. There is no restriction on deck size.

RESTRICTIONS:

Time, Core Requirements:

REPRØ operates at a rate of 250 cards per minute; it is a mainline program, effectively using all of core during execution.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

There must be sufficient cards in the 1403 punch hopper.

Processing/Method:

Cards are read and punched in column binary mode. This removes any restrictions on the nature of the deck to be reproduced. Processing is terminated when a log-start card is read, or on a last-card condition. REPRØ punches one additional blank card at end-of-job to clear the punch.

USAGE:

Calling Sequence:

REPRØ is normally called by a log-start card, but may be called by use of the inter-job entry code DØ, or by performance of a LINK macro from an AUTØCØDER or CØBØL program.

Control Cards:

REPRØ requires no control cards.

Input:

The only input consists of a card deck to be reproduced.

OUTPUT:

Output consists of 80-column card images punched into a blank deck.

ERROR MESSAGES AND HALTS:

Aside from the standard system halts and messages, no halts or messages should occur.

OPERATING PROCEDURES:

REPRØ requires no special operating procedures.

End-Of-Job, Post-Processing:

REPRØ returns to the inter-job supervisor on an end-of-job condition (i.e., when a log-start card is read, or when the last-card indicator is sensed). One additional blank card is punched, eliminating the need for a non-process runout.

1401 O. S. II
I. D. # ZB02A

PROGRAM:	MREPRØ
MACHINE:	IBM 1401
LANGUAGE:	AUTØCØDER
SUPERVISOR PROGRAM:	1401 Operating System II
PROGRAMMER:	Systems Division
DATE COMPLETED:	July 1, 1968

PURPOSE:

MREPRØ is a card-to-card program, producing multiple copies of card decks on the 1403 card punch.

RESTRICTIONS:

The maximum number of copies is 99.
The maximum deck size is 187 cards.

Time, Core Requirements:

MREPRØ punches cards at the rate of 250 cards per minute. It is a mainline program, effectively using all of core during execution.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

There must be a supply of blank cards in the 1403 punch hopper.

Processing/Method:

The program common field is examined. If the number of copies is not present, the message "HØW MANY CØPIES" is typed. After the operator has entered the number of copies, the input file is read into core, until either a last-card or end-of-file condition is sensed. MREPRØ punches the first copy into the N/P pocket, and the second copy into stacker pocket 8, alternating pockets in this fashion until the required number of copies has been punched.

USAGE:

Calling Sequence:

Source of Call:

MREPRØ is normally called by a log-start card, but may be called by means of the inter-job entry code DØ, or by execution of a LINK macro from an AUTØCØDER or CØBØL program.

Program Common:

The program common field should contain, right-justified, the number of copies to be generated. The maximum number of copies is 99.

Control Cards:

MREPRØ requires no control cards, unless other programs are in the job stream (batch mode). In that case, the End-of-File card is used.

End-of-File Card:

This card is required when the call to MREPRØ is followed by a stack of decks. One End-of-File card must terminate the deck to be reproduced. The format of this card is:

Col.	Content
1-4	4 record marks (0-2-8 multiple-punch)

Input:

Input consists of the deck to be reproduced, terminated by an End-of-File card.

OUTPUT:

Output consists of the duplicate decks, and possibly the message "HØW MANY CØPIES".

ERROR MESSAGES AND HALTS:

Aside from the standard system halts and messages, only the following message may occur:

HØW MANY CØPIES

This message only appears if the number of copies is not present in the program common field.

OPERATING PROCEDURES:

Set-up is as described in a previous paragraph. If the message "HØW MANY CØPIES" occurs, the operator should enter this information from the console typewriter. The operator should be aware that MREPRØ always returns to the inter-job supervisor, after reproducing a deck the requested number of times, on a last-card indicator.

End-Of-Job, Post-Processing:

MREPRØ returns control to the inter-job supervisor on a last-card indicator (or after reading a deck terminated by an end-of-file card), on completion of the requested number of copies.

PROGRAM:	CØLLAT
MACHINE:	IBM 1401
LANGUAGE:	AUTØCØDER
SUPERVISOR PROGRAM:	1401 Operating System II
PROGRAMMER:	Systems Division
DATE COMPLETED:	July 1, 1968

PURPOSE:

CØLLAT performs most of the functions of a collator, with the additional capability of collating on large alphabetic fields. Collating is performed on the 1402 Card Read/Punch at rates of 250 or 800 cards per minute.

RESTRICTIONS:

All restrictions are fully described below in the paragraphs on Usage and Program Procedure.

Time, Core Requirements:

CØLLAT reads cards at a rate of up to 800 cards per minute. When cards are being merged from the punch side, read speed drops to 250 cards per minute.

CØLLAT is a mainline program, effectively using all of core during execution.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

The 1402 card reader and punch must be cleared. The log-start card (if any), read-side control card, and primary file are placed in the card reader hopper; the punch side control card and the secondary file are placed in the card punch hopper. Then the console sense switches are set.

Processing/Method:

A detailed description of CØLLAT processing will be found below in the paragraph on Processing Method--Technical Description.

USAGE:

Calling Sequence:

CØLLAT is normally called by a log-start card, but may be called by use of the inter-job entry code DØ, or by execution of a LINK macro from an AUTØCØDER or CØBØL program.

Control Cards:

CØLLAT requires two control cards: one for the punch side and one for the read side. They designate the control field to be used for their respective files. The control field may consist of a group of subfields, which need not be sequentially located in the format of the input files. Subfields are listed on the control cards in descending order of significance (i.e., high-order subfield to low-order subfield). Four characters are required to define each subfield: the first two digits indicate the starting column, and the last two digits indicate the ending column. The read-side control field need not be identical to the punch-side control field, but must be the same length. The format of these cards is:

Read-Side Control Card:

Col.	Content
1-72	The subfields of the control field, in 4-character blocks.
(1-4)	(the first and last columns of the high-order subfield)
(5-8)	(the first and last columns of the next-highest-order subfield)
. . .	
(xx-xx)	(the first and last columns of the low-order subfield)
(73-74)	The number of blank cards to be merged (if the Merge Blanks option is used).
(76-80)	MERGR

Punch-Side Control Card:

Col.	Content
1-72	The subfields of the control field, in 4-digit blocks (see above, Read-Side Control Card, columns 1-72; the same format is used).
73-75	Blank
76-80	MERGP

Input:

Input to CØLLAT consists of the log-start card and the Read-Side Control Card followed by the primary input file in the reader hopper, and the Punch-Side Control Card followed by the secondary input file in the punch hopper. The operation to be performed is determined by the sense-switch settings. A full explanation of the sense-switch settings and their results is found below in the paragraph on Sense-Switches.

Sense-Switches:

The operation to be performed by CØLLAT is determined by the sense switch settings, as follows:

Sense Switch	Operation	Result	
B on	Straight Merge	All cards	- pocket 8/2
B on C on	Merge Select Equals (Both Feeds)	Equals(Read) Unequals(Read) Equals(Punch) Unequals(Punch)	- pocket 1 - pocket 8/2 - pocket 4 - pocket 8/2
B on C on E on	Merge Select Equals (Read Feed)	Equals(Read) Unequals(Read) Equals(Punch) Unequals(Punch)	- pocket 1 - pocket 8/2 - pocket 8/2 - pocket 8/2
B on C on F on	Merge Select Equals (Punch Feed)	Equals(Read) Unequals(Read) Equals(Punch) Unequals(Punch)	- pocket 8/2 - pocket 8/2 - pocket 4 - pocket 8/2
B on D on	2-Pocket Merge	Equals(Read) Unequals(Read) Equals(Punch) Unequals(Punch)	- pocket 8/ - pocket 8/2 - pocket 8/2 - pocket 4

Sense Switch	Operation	Result	
B on D on E on	Merge Replace Equal:	Equals(Read) Unequals(Read) Equals(Punch) Unequals(Punch)	- pocket 1 - pocket 2 - pocket 8/2 - pocket 4
B on E on	Merge Equals Only	Equals(Read) Unequals(Read) Equals(Punch) Unequals(Punch)	- pocket 8/2 - pocket 1 - pocket 8/2 - pocket 4
C on	4-Pocket Match	Equals(Read) Unequals(Read) Equals(Punch) Unequals(Punch)	- pocket 1 - pocket NR - pocket 4 - pocket NP
D on	3-Pocket Match, Select (Read)	Equals(Read) Unequals(Read) Equals(Punch) Unequals(Punch)	- pocket 8/2 - pocket 1 - pocket 4 - pocket 4
G on	Sequence Check (step-down)	Error Others	- pocket 1 - pocket N/P
G on F on	Sequence Check (step-down and equal)	Error Others	- pocket 1 - pocket NR
B on G on	Merge Blanks (after each card)	All cards	- pocket 8/2
B on F on G on	Merge Blanks (after Control Group)	All cards	- pocket 8/2

OUTPUT:

CØLLAT produces no output.

ERROR MESSAGES AND HALTS:

Aside from the standard system halts and messages, the following halts may occur during execution of CØLLAT:

	A - Address	Result
1.	111	No Read-Side Control Card. Insert control card, press start.
2.	222	No Punch-Side Control Card. Insert control card, press start.
3.	333	Read-side control field is not the same size as the punch-side control field. Insert corrected control cards, and press start.
4.	888	No sense-switches are set. Set console sense switches and press start to begin the program again.
5.	777	Number of blank cards not specified on control card, for merge-blanks options. Press start to begin the program again.
6.	969	Sequence error in the merge-blanks-after-control-group routine.
7.	842	Sequence error in the read-side file. Press start to read the next card.
8.	124	Sequence error in the punch-side file. Press start to read the next card.
9.	999	End of job. Press start for a new job.

OPERATING PROCEDURES:

Set-up is as described in a previous paragraph. Operating procedures consist of the following steps:

Set console sense-switches.

Place the following cards in the read hopper:

a) log-start card
b) read-side control card
c) primary card file

Place the following cards in the punch hopper (if applicable):

a) punch-side control card
b) secondary card file

The switch settings and their results will be printed on the 1403 printer. Set the appropriate switches and press START. If no switches are set the machine will halt. Pressing start will return the program to the sense-switch testing routine.

Errors in sequence will cause halts. Pressing start will read another card and drop the error card in the normal read or normal punch pocket.

If the read side control field is not the same size as the punch control field, the program will halt. Pushing start will start the program over again.

Errors in sequence using the sequence check option will be selected into pocket 1.

All options, except merging blanks option, require four blank cards at the end of each file. The merge blanks option must have exactly one blank card at the end of the read-side file. At the end-of-job the program will halt. The program can then be restarted by pushing start.

End-Of-Job, Post-Processing:

CØLLAT returns to the inter-job supervisor.

PROCESSING METHOD -- TECHNICAL DESCRIPTION:

CØLLAT sets up a control field based on the size of each subfield. It compares punch-side and read-side and then stacker selects the files into appropriate pockets based on sense-switch settings. All address adjustment is done by brute force, using no indexing or store registers.

Additional options (of the match-merge type) should easily be added. It is necessary only to understand the logic of testing each feed, and to follow the instructions used for a particular option.

The program compares a card from the read-side to a card from the punch-side. If the read-side is low, the read-side card is stacker selected and another card is read. This card is then compared to the previous read-side card. If the comparison is equal, the read-side card is selected, another card is read, and the comparison repeated. If the new card is high, a new comparison is made between the punch-side and the read-side. If the new card is low, there is an error in sequence and the program halts, and reads another card when restarted.

If the punch-side is low, the punch-side card is selected, and a new card is read. If equal, the selection is performed again. If high, a new comparison is made between the punch-side and the read-side. If low, there is an error in sequence.

If the two sides are equal, first the read-side card is selected, and a new card is read and properly handled as above. Once the new read-side card is high, the punch-side card is selected, and a new read-side card is read and properly handled. Once the new punch-side card is high, a new comparison is made between the punch-side and the read-side.

Thus, there are four unique stacker selections in the routine: a selection for the read-side if it is low; a selection for the read-side and a selection for the punch-side if the sides are equal; and a selection for the punch-side if it is low.

The punch-feed does not feed cards at the same rate as the read-feed. There must be a time delay before a card from the read-side may be allowed to fall, after a card from the punch-side has fallen. This allows the card from the punch-side to stack before the card on the read-side. This applies only if the card from the punch-side and the card from the read-side are selected into the same pocket.

If no delay is provided, the read-side card will arrive before the punch-side card. The delay routine is performed immediately prior to a read instruction, if a switch is set. The switch is set after a punch command if the sequence of cards on the punch-side is high.

The instruction to set the switch is either deleted or made active in the routine defining the option.

PROGRAM: RECØDE
MACHINE: IBM 1401
LANGUAGE: AUTØCØDER
SUPERVISOR PROGRAM: 1401 Operating System II
PROGRAMMER: Systems Division
DATE COMPLETED: July 1, 1968

PURPOSE:

RECØDE is a general character-set-exchange program for card files. As many distinct characters as desired can be transformed into other characters during one use of RECØDE. Any combination of punches whatsoever can be used as an "input" or "output" character in any of the transformations: RECØDE operates on a column-binary basis. The transformations to be done are specified on control cards, with the input deck being reproduced with the indicated modifications.

RESTRICTIONS:

Care should be taken in normal RECØDE usage to avoid loss of distinction between characters distinct in the input deck. In a true character-set conversion, any character that appears as an output character in one transformation should appear as an input character in another.

Time, Core Requirements:

Processing rate decreases rapidly as the number of transformations increases. For a single transformation, the rate is approximately 200 cards/minute.

RECØDE is a mainline program, effectively using all of core during its execution.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

A sufficient number of cards to reproduce the input deck should be present in the punch hopper.

Processing/Method:

The two control cards are read, and the specified transformations are stored, until a card column is found in which both control cards contain blanks.

A data card is read, and scanned for a match with the input characters. If a match is found, that character is replaced with the corresponding output character. The card is then punched. This process continues until a log-start card is read or a last-card condition is sensed.

USAGE:

Calling Sequence:

RECØDE is normally called by a log-start card, but may be called by means of the inter-job entry code DØ, or by a LINK macro from an AUTØCØDER or CØBØL program. The program common field is not used.

Control Cards:

Two control cards are required. The first specifies the one or more characters which are to be scanned for and replaced (the "input" characters). The second specifies the characters with which the "input" characters are to be replaced in the reproduced deck (the "output" characters). The input and output characters must appear in corresponding columns of their respective control cards. The end of the transformations to be performed is identified by a column in which both cards contain blanks.

Thus the N columns encountered on the pair of control cards, before a blank-blank correspondence, define N transformations to be performed: any occurrence of any of the input characters in the input deck will be replaced by the corresponding output character.

Input:

The input deck should immediately follow the pair of control cards.

OUTPUT:

RECØDE produces a deck which is identical to the input deck except for the transformations indicated on the control cards.

ERROR MESSAGES AND HALTS:

Only the system standard halts and messages may occur during execution of RECØDE.

OPERATING PROCEDURES:

No special operating procedures need be specified.

End-Of-Job, Post-Processing:

RECØDE returns control to the Inter-Job Supervisor via an EXIT macro on a last-card condition, and executes any log-start card encountered in the input deck.

1401 O. S. II
I. D. # ZC01A

PROGRAM: TLIST
MACHINE: IBM 1401
LANGUAGE: AUTØCØDER
SUPERVISOR PROGRAM: 1401 Operating System II
PROGRAMMER: Systems Division
DATE COMPLETED: July 1, 1968

PURPOSE:

TLIST is a tape-dump program for records of 132 characters or less. Records in either binary (odd-parity) or BCD (even-parity) format may be listed, in any mixture.

RESTRICTIONS:

Time, Core Requirements:

Execution time for TLIST varies according to the number of records listed. TLIST is a mainline program, effectively using all of core during its execution.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

No previous processing is necessary for use of TLIST. The input tape is assumed to be on drive 2. 14 inch wide paper should be used in the 1403 printer, as the listing produced is 132 characters wide.

Processing/Method:

The tape on unit 2 is rewound, and the following ensues for each record:

1. The annotation line is printed.
2. The record is read in BCD mode with word marks.
   If there is no error, the data and word mark lines are printed, and the fourth line is spaced.
   If there is a tape error,
3. The record is backspaced over.

4. The record is read in binary mode without word marks.
   If there is no error, the data line and the "BIN" line are printed, and the fourth line is spaced.
   If there is a tape error,
5. A line that is blank except for "ERR" in positions 130-132 is printed, and the third and fourth lines are spaced.

Neither halts nor retries are performed on an actual tape error. On an EOF indication, either a tape mark or a reflective spot sensed, "EOF" is printed in 130-132, the carriage is restored to a new page, and the program halts. Printing of any records beyond the EOF can be accomplished by pressing START at this point.

USAGE:

Calling Sequence:

TLIST may be called by a log-start card, by a LINK macro from an AUTØCØDER or CØBØL program, or by means of the inter-job entry code DØ. The program common field is not used.

Control Cards:

TLIST requires no control cards.

Input:

The only input to TLIST consists of the tape on unit 2.

OUTPUT:

Each tape record produces output comprising four lines:

1. Guideline -- provides positioning annotation
2. Data
3. a. for BCD records -- word marks if any
   b. for binary records -- blank except for "BIN" in positions 130-132
4. Blank line

No indication is provided in the output of the length of the tape record, nor are group marks or other unprintable characters indicated in any fashion.

ERROR MESSAGES AND HALTS:

Aside from the standard system disk halts and messages, the "BIN, "ERR, and "EOF" lines described previously are the only halts and messages.

OPERATING PROCEDURES:

No further operating procedures need be given: TLIST uses no sense switches or peripherals other than the 1403 printer and tape unit 2.

End-Of-Job, Post-Processing:

End-of-job indication consists of the "EOF" line at a point where printing is not to be continued. TLIST does not EXIT or LINK.

PROGRAM:	TDUMP
MACHINE:	IBM 1401
LANGUAGE:	AUTØCØDER
SUPERVISOR PROGRAM:	1401 Operating System II
PROGRAMMER:	Systems Division
DATE COMPLETED:	July 1, 1968

PURPOSE:

TDUMP is a tape-to-printer routine, listing the contents of a BCD tape on the 1403 printer, with positional annotation.

RESTRICTIONS:

Time, Core Requirements:

TDUMP prints at the rate of 600 lines per minute; it is a mainline program, effectively using all of core during execution.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

The BCD tape to be dumped may be mounted on any of the tape drives. The drive is selected by means of sense switch settings (see below, Sense Switches).

Processing/Method:

TDUMP displays the sense switch settings and their results on the 1403 printer, and then halts. After the switches have been set and the program restarted, the contents of the requested tape are printed on the 1403 printer.

USAGE:

Calling Sequence:

TDUMP is normally called by a log-start card, but may be called by means of the inter-job entry code DØ, or by execution of a LINK macro from an AUTØCØDER or CØBØL program.

Control Cards:

TDUMP requires no control cards.

Input:

The only input to TDUMP consists of the BCD tape, mounted on the tape unit specified by the sense switch settings, as described below.

Sense Switches:

Sense switches B, C, and D are used to determine the tape input unit. Switch B = 1, C = 2, and D = 4, the tape unit being determined by addition of these values. Switch E is off to allow special format for 80-character records; switch F is on to force this special format.

OUTPUT:

Printed output for both normal format and 80-character special format consists of:

a) the sense switch settings and their results.
b) the tape unit providing input.

Then, for 80-character records,

c) two annotation lines, followed by a blank line, specifying the 80 positions of the printed record, in the form:

```
0000 ... 77778
1234 ... 67890
```

d) the 80-character records. At the extreme right end of each printed line the file/record count appears (see below, Record Count).

For other than 80-character records,

c) an annotation line, numbering the positions of the 100-character printed line, and giving the record count for that record (see below, Record Count).

d) the contents of the record, 100 characters to the line. Each line is tagged with a value indicating the location of the right most character of that line within the record. (100, 200, 300, etc.).

Record Count:

This is a value appearing at the extreme right end of the page. It is on the record print line for 80-character records, and on the annotation line for other records. It is of the form:

N-XXXXX

where N is the file number (number of tape-mark halts plus one), and XXXXX is the record number within that file.

ERROR MESSAGES AND HALTS:

All messages are self-explanatory. A halt occurs whenever a tape-mark is read (see below, Operating Procedures).

OPERATING PROCEDURES:

At the tape-mark halt, the operator has three options:

a) reading beyond the tape-mark, by pressing Start.

b) changing the input drive by changing the logical drive numbers on the tape units and pressing Start.

c) returning to the inter-job supervisor, by pressing Start/Reset and Start.

If option a) above is selected, a log-start card must be loaded to return to the inter-job supervisor.

End-Of-Job, Post-Processing:

If option c) in the preceding paragraph is selected, TDUMP returns control to the inter-job supervisor and rewinds the tape. Otherwise control must be returned by means of a loaded log-start card, and the tape rewound manually.

PROGRAM: DCØPY
MACHINE: IBM 1401
LANGUAGE: Autocoder
SUPERVISOR PROGRAM: 1401 Operating System II
PROGRAMMER: Systems Division
DATE COMPLETED: July 1, 1968

PURPOSE:

DCØPY, when executed, will copy the disk pack on a 1311 disk drive onto a tape on tape unit 2.

RESTRICTIONS:

Time, Core Requirements:

Time requirement varies with the number of mode changes and the mode, in general, of the disk. Execution is less than 15 minutes on the average.

DCØPY is a mainline program and effectively uses all of core during execution.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

No previous processing is necessary for use of DCØPY. The disk pack to be copied should be on the opposite drive from the Operating System pack.

Processing/Method:

DCØPY first determines which disk is to be copied to tape, by use of the program common field. A disk track is read. If a mode error occurs, the track is reread in the other mode. If the mode of the track was move, only the control header and move-mode record are written. Otherwise, the area is cleared to a blank word mark skeleton and a load-mode record is generated. The track is then reread and the move-mode record is generated.

USAGE:

Calling Sequence:

Source of Call:

DCOPY is normally called by a log-start card, but may be called by means of the inter-job entry codes ØC, DC, or DO, or by execution of a LINK macro from an AUTØCØDER or CØBØL program.

Program Common:

The drive code of the disk drive to be copied to tape must appear, left justified, in the program common field. (i.e. 0 or 2)

Control Cards:

DCØPY requires no control cards.

Input:

The only input consists of the disk pack to be copied.

OUTPUT:

The output from DCØPY consists of a tape containing:

a) header information consisting of an IØCS header.
b) copy control records.
c) move or load mode records of complete disk tracks.
d) move or load mode records of disk sectors.
e) an IØCS trailer.

The format of the IØCS header is type "B" (80-positions), with the identification "DISK COPY."

The copy control record is a move-mode BCD 30-character record containing the following information:

a) T--the next record is a complete track.
   S--the next record is a sector.
b) L--the next record is in the load-mode.
   M--the next record is in the move-mode.
c) The actual sector address.
d) The natural sector address.

For each load-mode track copied, there are three tape records generated:

a) the copy control record.
b) the BCD load-mode work mark skeleton.
c) the move-mode binary disk data.

When a track is in move-mode, only the BCD move-mode binary record is generated.

When a track consists of mixed mode sectors, the same type of records are written as were read. However, the length of each sector record as written is 110 positions.

ERROR MESSAGES AND HALTS:

Three types of halts and messages may occur during execution of the disk copying program:

a) the systems standard halts and messages,
b) IØCS halts, described in the IBM publication C24-3298(Input/Output Control System -- Disk, Operating Procedures. IBM File No. 1401/1460-30),
c) halts and messages specific to the disk copying program.

Only halts and messages of type c) will be discussed here:

0. TURN CØMPARE DISABLE SWITCH ØN
TURN WRITE ADDRESS KEY ON
MØUNT DISK TØ BE CØPIED ØN DRIVE n
MØUNT TAPE TØ BE CØPIED ØNTØ ØN DRIVE 2
This message is for the information of the 1401 operators.

1. DRIVE CØDE NØT SPECIFIED IN CØMMØN
This message states the cause for the return to the operating system.

2. TRACK BEGINNING AT xxxxxx UNCØPYABLE
This message indicates that a parity error existed when the track was read in both modes.

3. TURN CØMPARE DISABLE SWITCH ØFF
TURN WRITE ADDRESS KEY ØFF
This message is for the information of the 1401 operator.

4. PARITY ERRØR ØR ØTHER BAD NEWS AT SECTØR xxxxxx
NØT CØPIED
This message indicates that, at the sector level, the sector was found to be damaged.

OPERATING PROCEDURES:

Set-up is as noted in a previous paragraph. Special operating procedures are fully specified in the messages to the operator (see Error Messages and Halts, messages 0 and 3). Messages 1, 2, and 4 are the concern of the user.

End-Of-Job, Post-Processing:

Normally, the rewound tape on unit 2 would be dismounted and returned to the user. Control is returned to the inter-job supervisor on completion.

PROGRAM: RESTR
MACHINE: IBM 1401
LANGUAGE: AUTØCØDER
SUPERVISOR PROGRAM: 1401 Operating System II
PROGRAMMER: Systems Division
DATE COMPLETED: July 1, 1968

PURPOSE:

RESTR is a general tape-to-disk program, designed to restore the content of a disk pack from a tape copy previously produced by the program DCØPY (ZD01A).

RESTRICTIONS:

Time, Core Requirements:

Time requirement varies with the number of mode changes and the mode, in general, of the tape. Execution requires less than 5 minutes, on the average. RESTR is a mainline program and effectively uses all of core during execution.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

General Usage:

The tape to be copied to disk should be mounted on tape drive 2. The disk pack to be restored should be mounted on the drive opposite the Operating System Pack.

Operating System Restoration:

When the Operating System has been damaged, a special set-up is required. Set-up and procedure details will be found below in the paragraph titled Operating System Restoration Procedures.

Processing/Method:

The program common field is examined to determine which drive contains the pack to be restored. Then the tape is opened, and the copy control record is read. From this record the type (track or sector) and mode of the succeeding record(s) is (are) determined. From this point on, the process is a step-by-step inversion of that used by DCØPY (ZD01A).

USAGE:

Calling Sequence:

Source of Call:

RESTR is normally called by a log-start card, but may be called by use of the inter-job entry code DØ, or by execution of a LINK macro from an AUTØCØDER or CØBØL program.

Program Common:

The drive code of the disk drive containing the pack to be restored, either 0 or 2, must be placed in the program common field, left-justified.

Control Cards:

RESTR requires no control cards.

Input:

The only input consists of the tape to be copied.

OUTPUT:

The tape records written to disk are controlled by the copy control records. They may be in load-mode, move-mode, or mixed.

ERROR MESSAGES AND HALTS:

Three types of halts and messages may occur:

a) the system standard halts and messages,
b) IØCS halts, as described in IBM publication C24-3298 (Input/Output Control System -- Disk, Operating Procedures. IBM File No. 1401/1460-30),
c) halts and messages specific to the disk restoration service.

Only halts and messages of type c) will be discussed here:

0. MØUNT PACK TØ BE RESTØRED ØN n
MØUNT PREVIØUSLY CØPIED TAPE ØN 2
TURN CØMPARE DISABLE SWITCH ØN
TURN WRITE ADDRESS KEY ØN
This message is for the information of the 1401 operator.

1. DRIVE CØDE NØT SPECIFIED IN CØMMØN
This message states the cause for the return to the operating system.

2. THE PØSSIBILITY EXISTS THAT THE
(TRACK) (s) STARTING AT xxxxxx
(SECTOR)
HAS(HAVE) BEEN BYPASSED
Due to tape read errors or other reasons the disk track/sector may not have been restored.

3. TURN CØMPARE DISABLE SWITCH ØFF
TURN WRITE ADDRESS KEY ØFF
This message is for the information of the 1401 operator.

OPERATING PROCEDURES:

General Usage:

Set-up is as noted in a previous paragraph. Normally, the rewound tape will be dismounted and returned to the user. Messages 0 and 3 provide information to the operator; 1 and 2 are the concern of the user.

Operating System Restoration Procedures:

If the Operating System has been destroyed, the following procedure should be followed:

a) run HELP (KE03A), the emergency log-dump routine.
b) mount the Operating System backup tape on tape unit 2.
c) mount the Operating System disk pack on disk drive 2 (the left-hand module).
d) clear the card reader.
e) place the card function RESTR in the reader and press LØAD, with sense switch A on. Be sure that no decks or cards follow this object deck.

End-Of-Job, Post-Processing:

RESTR returns control to the inter-job supervisor on an end-of-job condition.

PROGRAM: CLRDSK
MACHINE: IBM 1401
LANGUAGE: AUTØCØDER
SUPERVISOR PROGRAM: 1401 Operating System II
PROGRAMMER: IBM
DATE COMPLETED: -----

PURPOSE:

CLRDSK clears disk areas as defined in area-definition control cards. The user must provide the CTL and area-definition control cards.

RESTRICTIONS:

No header operations may be performed.

USAGE:

Calling Sequence:

CLRDSK is normally called by a log-start card, but may be called by use of the inter-job entry code DØ, or by execution of a LINK macro from an AUTØCØDER or CØBØL program.

NOTE:

For further information, see the following IBM publications, where this program is described as CLEAR DISK:

Disk Utility Programs, 1401/40/60	C24-1484
Disk Utility Programs, Operating Procedures	C24-3105

PROGRAM: PRDSK
MACHINE: IBM 1401
LANGUAGE: AUTØCØDER
SUPERVISOR PROGRAM: 1401 Operating System II
PROGRAMMER: IBM
DATE COMPLETED: ---

PURPOSE:

PRDSK lists disk areas as defined in area-definition control cards. The user must provide the CTL and area-definition control cards.

RESTRICTIONS:

No header operations may be performed.

USAGE:

Calling Sequence:

PRDSK is normally called by a log-start card, but may be called by use of the inter-job entry code DØ, or by execution of a LINK macro from an AUTØCØDER or CØBØL program.

NOTE:

For further information, see the following IBM publications, where this program is described as PRINT DISK:

Disk Utility Programs, IBM 1401/40/60 C24-1484
Disk Utility Programs, Operating Procedures C24-3105

PROGRAM:	DALTR
MACHINE:	IBM 1401
LANGUAGE:	AUTØCØDER
SUPERVISOR PROGRAM:	1401 Operating System II
PROGRAMMER:	Systems Division
DATE COMPLETED:	July 1, 1968

PURPOSE:

DALTR provides the user with the ability to view and/or correct a disk sector.

RESTRICTIONS:

Time, Core Requirements:

Execution time depends directly on the number of sectors viewed and changes made. DALTR is a mainline program, effectively using all of core during execution.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

No previous processing is necessary for use of DALTR, except that the Operation System Pack must be mounted.

Processing/Method:

The DALTR program requests the drive code, sector address, and mode from the user. With this information DALTR constructs a disk control field and reads the specified sector into storage. If the mode is wrong or an address check occurs, messages to that effect are typed. If the read is successful, the sector is typed and the user may correct it. In the case of a correction the sector is written back to disk.

USAGE:

Calling Sequence:

DALTR is normally called by a log-start card, but may be called by means of the inter-job entry code DØ, or by a LINK macro from an AUTØCØDER or CØBØL program.

Control Cards:

DALTR requires no control cards.

Input:

Input consists of responses to requests by the program, via the console typewriter. Details appear below in the paragraph on Operating Procedures.

OUTPUT:

DALTR produces no output except the viewed and/or correct disk sector.

ERROR MESSAGES AND HALTS:

Two types of halts and messages may occur during the execution of DALTR.

a) the system standard halts and messages.
b) requests and the disk sector.

See the following paragraph on Operating Procedures.

OPERATING PROCEDURES:

The operating procedures consist of responding to the following typed requests:

a) The system types

TYPE DRIVE CØDE, ADDRESS

The user responds with R/E (pressing the request/enter key), then typing the one digit drive code followed by the six digit sector address. The response is completed with R/T (pressing the respond/typeout key).

b) The system then types

TYPE MØDE, M ØR L

The user responds with R/E, then types an M or an L and finishes with R/T.

c) The system then types 9 or 10 rows of 10 characters each. These rows contain the contents of the disk sector under investigation. Ten rows are typed if the sector was in the move-mode. Nine rows are typed if the sector was in the load-mode.

d) The system then types

TYPE SECTION $\begin{Bmatrix} 01\text{-}10 \\ 01\text{-}09 \end{Bmatrix}$

The user looks at the typeout and decides which section he wants to change. He then responds with R/E, types the two-digit section number, and finishes with R/T.

e) The system types

TYPE CHANGE

The user responds by R/E, typing his change, and finishing with R/T.

f) The procedure reverts to step c.

When the user has completed alteration of a sector, he presses R/E and R/T consecutively without entering a section number (see step e), in which case the operation reverts to step a. When all desired sectors have been reviewed, the user merely hits R/E and R/T and control returns to the Operating System.

Corrections may exceed the selected section. If the sector to correct is in the load-mode and the user wishes that a word mark be entered, he must depress the WØRD MARK key for each word marked character. The user may type an A-bit character when he desires to retain characters.

As in all 1401 operations, if the user makes a typing error he may correct it by depressing the CLEAR key, in which case the system returns to the beginning type-in operation.

End-Of-Job, Post-Processing:

After pressing R/E and R/T, DALTR returns control to the inter-job supervisor.

PROGRAM: ALINK
MACHINE: IBM 1401
LANGUAGE: AUTØCØDER
SUPERVISOR PROGRAM: 1401 Operating System II
PROGRAMMER: Systems Division
DATE COMPLETED: July 1, 1968

PRUPOSE:

The ALINK macro, when executed, causes a program to be loaded from the 1401 Operating System library, and control to be transferred to that routine. The purpose is similar to that of the LINK macro (KF14A), except that the disk location of the program block to be called must be known in order to use ALINK. The advantage of ALINK is that it requires far less core than the LINK fixed routines LØDER and LØADR.

RESTRICTIONS:

No check is performed to verify that the disk sector addressed by the operand actually contains the header record of a program block.

The following restrictions exist in comparison to a LINK macro:

a) the usage statistics of the called program are not updated.
b) the program common facility is not available.
c) no variability of the execution address after loading is available.
d) the point at which execution is begun after loading is the same point at which execution was begun last time the program was loaded. Thus a program to be called by an ALINK macro should not be FETCHed or otherwise loaded with any execution address other than the normal execution address of the program.

Time, Core Requirements:

If group-mark word-marks are to be restored in the called program, execution time for the ALINK macro will be between 3 and 25 seconds. If group-mark word-marks are not to be restored, execution time is less than 2 seconds. ALINK is slightly faster than the corresponding LINK.

Core requirement is 129 positions.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

The Operating System Pack should be on one of the 1311 disk drives at execution time. If it is not, the system halts, and may be started when the pack is mounted.

The programmer may write an ALINK macro at any point during his processing.

Processing/Method:

ALINK attempts to read the first sector of the loader of the specified program into the card read area, from disk module 1. If this fails, the attempt is made on module 0. If both attempts fail, the system halts. When either attempt succeeds, control is transferred to the instructions read from the disk. The definition of "failure", in this context, is simply an incompatible sector format (i.e. either move-mode or non-address-protected). Thus if the operand specifies a sector which is not the start of a program, the macro itself cannot detect this and the results are unpredictable: a process check will probably result.

USAGE:

Operands:

The ALINK macro uses only one operand: the 5-digit sector address of the header sector of the program to be called. This should be written as an actual operand (not enclosed in quotes as a literal, or given in a DCW elsewhere) and can lie in the range of 10000 to 19790. This should be the same value that was used as operand 11 of the DLØAD macro (KF06A) which loaded the program in question to the Operating System library.

Control Cards:

The ALINK macro uses no control cards.

Input:

The ALINK macro uses no input.

OUTPUT:

The ALINK macro produces no output.

ERROR MESSAGES AND HALTS:

Only the standard system halts and messages may occur during execution of the ALINK macro.

OPERATING PROCEDURES:

Aside from the fact that the correct Operating System pack must be mounted if a halt occurs, and the halt then started, no special operating procedures need be followed.

End-Of-Job, Post-Processing:

The macro transfers control to the loader of the specified program when it has successfully read the loader from disk.

PROGRAM: CDSN
MACHINE: IBM 1401
LANGUAGE: AUTØCØDER
SUPERVISOR PROGRAM: 1401 Operating System II
PROGRAMMER: Systems Division
DATE COMPLETED: July 1, 1968

PURPOSE:

The CDSN macro, when executed, will compute the check digit of a given nine digit student number, and then test the given check digit against the computed check digit.

RESTRICTIONS:

Time, Core Requirements:

Execution time for the CDSN macro is about 50 milliseconds. Each time a CDSN macro is used, 259 positions of core are generated.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

No special set-up or previous processing is required.

Processing/Method:

Digits 4-9 of the student number are added triangularly to a sum box, this process being repeated for digits 1-3. This sum, modulus 11, gives the computed check digit. The computed and given check digits are compared. If they are found to be unequal, control is transferred to a user routine. Otherwise control returns to the next instruction following the macro.

USAGE:

Operands:

Operands 1 and 2 of CDSN are required; no other operands are used.

1. Ten Digit Student Number
   This operand gives the location of the student number and check digit. It may be a 10 digit literal or an indexed/adjusted tag.

2. Failure Exit
   This operand contains the address of the user routine which will process the student record information if the given student number is found to be incorrect.

Control Cards:

The CDSN macro uses no control cards.

Input:

The CDSN macro uses no input.

OUTPUT:

The CDSN macro produces no output.

ERROR MESSAGES AND HALTS:

Only the system standard halts and messages may occur during execution of the CDSN macro.

OPERATING PROCEDURES:

No special procedures need be specified.

End-Of-Job, Post-Processing:

Control is transferred either to the next sequential instruction, or to the given user routine in the case of an incorrect student number (failure exit).

1401 O. S. II
I. D. # KF03A
I. D. # KF04A

PROGRAM:	CMADD/CMPAD
MACHINE:	IBM 1401
LANGUAGE:	AUTØCØDER
SUPERVISOR PROGRAM:	1401 Operating System II
PROGRAMMER:	Systems Division
DATE COMPLETED:	July 1, 1968

PURPOSE:

The CMPAD macro, when executed, causes two 1401 machine language addresses to be compared according to address sequence. The appropriate indicators (low, equal, high, unequal) are set, depending on the result of the comparison. Appropriate conditional branch instructions should be provided by the programmer following the CMPAD macro, according to the logic of his program. The CMPAD macro sets up linkage to the inflexible routine CMADD, which does the actual comparison and returns control to the next sequential instruction following the CMPAD macro.

RESTRICTIONS:

If an address compared by means of a CMPAD macro contains an index tag, the contents of the index register are not taken into account in the comparison. This obstacle may be avoided by use of a 2-address SBR instruction immediately preceding the CMPAD macro.

No responsibility is taken as to the results if one or both of the address fields compared is not a valid 1401 address: i.e., contains a numeric part which is blank or greater than 9 in binary value.

Time, Core Requirements:

Execution time for the CMPAD sequence, including the processing done in CMADD, is 1.69 milliseconds plus .04 milliseconds for each operand which is indexed.

The CMPAD generated code occupies 34 positions. The CMADD routine occupies 61 positions; it is generated only once, regardless of the number of CMPAD's written or of literal origins in the program.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

No special set-up or previous processing is required to use the CMPAD macro, except that the two addresses to be compared must exist at execution time. The programmer may write a CMPAD macro at any point during his processing.

Processing/Method:

A detailed description of CMPAD processing is found below in the paragraph on Processing Method -- Technical Description.

USAGE:

Operands:

The CMPAD macro requires operand 1 and operand 2. These operands are of the same form as the operands of a normal compare instruction, giving the locations of the two addresses to be compared. The way in which indicators are set, based on the address fields at the A and B addresses, is also identical to that of a normal compare instruction. The referenced address fields need not contain word-marks in the hundreds position, and should not contain word-marks in the tens or units position.

Control Cards:

The CMPAD macro requires no control cards.

Input:

The CMPAD macro uses no input.

OUTPUT:

The CMPAD macro produces no output.

ERROR MESSAGES AND HALTS:

No halts or messages may occur during execution of either CMPAD or CMADD.

OPERATING PROCEDURES:

No special operating procedures need be specified.

End-Of-Job, Post-Processing:

Control is transferred to the next sequential instruction following the CMPAD macro, where the programmer may code conditional branch instructions appropriate to the logic of his program.

PROCESSING METHOD -- TECHNICAL DESCRIPTION

The CMPAD linkage to the fixed routine CMADD consists of moving the addresses to be compared into the CMADD routine and then branching into the routine. CMADD saves the return point and then transforms the addresses into two five-character fields, referred to as $A_1A_2A_3A_4A_5$ and $B_1B_2B_3B_4B_5$, in the following fashion:

a) The numeric parts of the A-field address are moved under blank zones at $A_3A_4A_5$, in the same left-to-right order as in the address.
b) The numeric parts of the B-field address are moved under blank zones at $B_3B_4B_5$, in the same left-to-right order as in the address.
c) The hundreds-position zone of the A-field address is moved over a zero numeric at B_2.
d) The hundreds-position zone of the B-field address is moved over a zero numeric at A_2.
e) The ones-position zone of the A-field address is moved over a zero numeric at B_1.
f) The ones-position zone of the B-field address is moved over a zero numeric at A_1.

The two fields thus created are compared using a normal compare instruction, with $B_1B_2B_3B_4B_5$ as the B-field, and control is returned to the next sequential instruction following the CMPAD macro.

PROGRAM:	CØRE
MACHINE:	IBM 1401
LANGUAGE:	AUTØCØDER
SUPERVISOR PROGRAM:	1401 Operating System II
PROGRAMMER:	Systems Division
DATE COMPLETED:	July 1, 1968

PURPOSE:

The CØRE macro, when executed, produces an annotated printout of the contents of 1401 storage: a "core dump". This may be useful if a program has detected an "impossible" or otherwise troublesome condition outside the framework of its logic, as in certain types of abnormal end-of-job conditions. Execution of CØRE disturbs/alters only positions 201-332, and execution of the program proper will be resumed on completion of the printout.

RESTRICTIONS:

Non-printable characters other than group-marks are not annotated.

Time, Core Requirements:

Execution time is approximately two minutes. Core requirement is 575 positions.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

The only set-up required for CØRE is 14 inch wide paper on the 1403 printer.

No special previous processing is required for use of CØRE. The programmer may write a CØRE macro at any point in his program.

Processing/Method:

For each 100 characters:

a) the word-mark status of the XXX01 position is saved.
b) a "guardian" word-mark is set at the XXX01 position.
c) the data from XXX01 through XXY00 is loaded into the print area.
d) the word-mark status of the XXX01 position is restored in both the actual position and in the print line.
e) the character line is printed.
f) the word-mark line is printed.
g) the annotation line is set up.
h) positions XXX01 through XXY00 are scanned for group-marks, and 'G's are inserted in the annotation line as appropriate.
i) the annotation line is printed.
j) a line is spaced on the printer.

USAGE:

Operands:

CØRE requires no operands. All of core is always dumped, regardless of blank areas.

Control Cards:

CØRE requires no control cards.

Input:

CØRE uses no input.

OUTPUT:

The contents of the print area and word-marks in the print area are printed first with annotation. Positions 1-100, 101-200, 333-400, 401-500,..., 15900-15999 are then printed in succeeding sets of lines. The output for each 100 positions consists of

a) a line representing the characters
b) a line representing the word-marks

c) a line containing annotation, both horizontal position and the hundreds-address (in both 3-character and 5-digit form). Wherever a position contains a group-mark, a 'G' overlays the horizontal-position annotation.
d) a blank (spaced) line.

ERROR MESSAGES AND HALTS:

No halts or messages may occur during use of CØRE.

OPERATING PROCEDURES:

Set-up is as noted in a previous paragraph. Operation of CØRE should be completely automatic.

End-Of-Job, Post-Processing:

CØRE is an "in-line" macro -- control enters it in sequential fashion and is passed from it in sequential fashion. Thus the programmer should provide instructions following the CØRE macro to accomplish whatever purpose he desires following completion of the core dump.

PROGRAM: DLØAD
MACHINE: IBM 1401
LANGUAGE: AUTØCØDER
SUPERVISOR PROGRAM: 1401 Operating System II
PROGRAMMER: Systems Division
DATE COMPLETED: July 1, 1968

PURPOSE:

The DLØAD macro, when executed, causes a program block to be loaded onto the Operating System library. Various operands of the macro control the fashion in which the program is subsequently loaded into core. Only the program name need be known to call the program once it is loaded onto the library. This document is complemented by the description of the DLØDR program (KC06A), which is called by the macro.

RESTRICTIONS:

Time, Core Requirements:

The DLØAD macro executes in less than 3 seconds. The overall DLØAD process, including operation of the DLØDR program (KC06A), may take between 10 and 30 seconds. Core requirement is in the range of 363-375 core positions.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

The Operating System pack should be on one of the 1311 disk drives at execution time. Paper at least 8 1/2 inches in both width and height should be mounted on the 1403 printer. The DLØAD macro is often executed as the result of running an object deck under the LØAD program (KC07A), or as a result of an AUTØCØDER or CØBØL "RUN THRU EXECUTIØN" under the SYSCL program (KC10A).

Processing/Method:

A detailed description of DLØAD processing is found below in the paragraph on Processing Method -- Technical Description.

USAGE:

Operands:

Only operand 1, the Program Name operand, is required. Operands 2 through 11 are optional.

1) Program Name (required)

This operand must be present, and should be 1 to 6 characters, digits or special characters in length. The name of a program under the Operating System may not contain record marks or commas, and should not end in a period. Care must be taken to avoid name conflicts with previously loaded programs, or with programs to be loaded in the future. A name like PRØCSS would be undesirable in the latter context.

The name in the DLØAD macro is simply written as an operand, and is neither surrounded by "at" signs as a literal nor given in a DCW elsewhere.

2) Execution Address (optional)

This operand gives the location at which execution is to be begun after a LINK or ALINK macro is performed. If the operand is omitted, the program block may be accessed only by means of a FETCH macro. The operand may be symbolic or actual, indexed and/or adjusted.

3) Load-Into-Core Address (optional)

This operand gives the location at which the program block is to be read into core. If the operand is omitted, the load-into-core address is synthesized as the first word-marked or non-blank character above position 86. The operand may be symbolic or actual, adjusted but not indexed, and if given, should be in the range 81 to 15908 inclusive. Ten positions below this address must be available for use when the program block is loaded.

4) Suggested Group-Mark Word-Mark Address (optional)

This operand gives the location one above the last core position which must be loaded into core with the program block. If the operand is omitted, the

suggested group-mark word-mark address is synthesized as one above the last word-marked or non-blank character in core (not considering the EXIT macro and common field left in core by the LØAD program). The operand may be symbolic or actual, adjusted but not indexed, and if given should be greater than the load-into-core address and less than or equal to 15998. The "actual group-mark word-mark address" is constructed by the DLØDR program (KC06A), and may be equal to, or up to 89 positions higher than, the suggested GMWM address.

5) Clear-Storage-From Address (optional)

This operand gives the highest core position to be cleared to blanks prior to loading the program. If the operand is omitted, the clear-storage-from address is 15999. The operand may be symbolic or actual, adjusted but not indexed, and if given should be greater than or equal to the suggested group-mark word-mark address.

6) Clear-Storage-To Address (optional)

This operand gives the lowest core position to be cleared to blanks prior to loading the program. If the operand is omitted, the clear-storage-to address is 81. If this address is less than the load-into-core address minus 10, a word mark is associated with the position after loading. The operand may be symbolic or actual, adjusted but not indexed, and if given should be less than or equal to the load-into-core address.

7) IØCS Date Decision (optional)

If it is desired that the IØCS date be loaded into positions 82-86 as part of the loading process for the program block, include "Y" or "YES" as operand 7. If this operand is other than these or is omitted, the IØCS date is not loaded.

8) Program Common Location (optional)

If it is desired that the 9-character program common field from the log-start card or the LINK or FETCH macro be placed in a specified location during the loading process, include that location (right-end) as operand 8. If this operand is omitted, the program common field will not be passed to the program block. The operand may be symbolic or actual, indexed and/or adjusted.

9) Origin Address (optional)

If operand 9 is included, it is used as an origin address for the DLØAD macro. A branch instruction to DLØAD is included at the in-line location if operand 9 is used. The operand may be symbolic or actual, adjusted but not indexed.

10) Immediate Execution Address (optional)

If operand 10 is included, the program block is immediately LINKed to after it is loaded to disk, with execution beginning at the address specified by operand 10. This operand need not be the same as the execution address (operand 2); indeed, there need be no execution address specified at all. In all other respects (core-clearing, etc.) the "immediate execution" LINK process is exactly like that of a normal LINK to the program block. Additional information can be found in the write-up of the DLØDR program (KC06A). Segmentation of a single source (object) deck into two or more disk stored program blocks can be done through use of operand 10 in intermediate DLØAD macros, with immediate execution to a branch to the condensed loader or self-loading deck.

The operand may be symbolic or actual, adjusted but not indexed.

11) System Disk Address/"No Group-Mark Word-Mark Restoration" Specification (optional)

If this operand is included, it should be a five-digit number. If the operand is in the range 10000 to 19790, the operand specifies that the program block should be loaded to disk starting at that sector address, and should never be moved in the future. The five-digit number can then be used as the operand of an ALINK macro (KF01A) in other programs. The user considering use of such a "system disk address" should consult a recent program list produced by PLIST (KC08A) to determine a suitable address.

Whether or not the operand is in the above-mentioned range, group-mark word-marks are not restored (left as word-marked colons) during the loading process for the program block, simply because the operand is present. This non-restoration greatly increases

the speed of LINKs or FETCHes done on the program block (as much as 20 times), but has the disadvantage that the programmer must create his own group-mark word-marks. If only this latter purpose is intended, use of 00000 as operand 11 is recommended.

Control Cards:

The DLØAD macro uses no control cards.

Input:

The DLØAD macro uses no input.

OUTPUT:

The DLØAD macro itself produces no output. However, the DLØDR program (KC06A), which is called by the DLØAD macro, produces output as described in the write-up of that program.

ERROR MESSAGES AND HALTS:

If the program name (operand 1) is not included in the DLØAD macro, the message

1 DLØAD ERRØR - NAME MUST BE PRESENT

is generated in the source program, followed by a hard halt.

Aside from this, the only halts or messages possible in the DLØAD macro itself are the standard system disk halts and messages. The user should consult the write-up for the DLØDR program (KC06A) concerning the several halts and messages possible in that phase of the DLØAD process.

OPERATING PROCEDURES:

Set-up is as described in a previous paragraph. No special operating procedures need be given for the macro itself. Consult the Operating Procedures for the DLØDR program (KC06A).

End-Of-Job, Post-Processing:

See the Post-Processing paragraph for DLØDR (KC06A).

PROCESSING METHOD - TECHNICAL DESCRIPTION:

The drive on which the Operating System Pack resides is determined by reading the first sector of the loader of the DLØDR program (KC06A). Module 1 is tried first, then module 0 if the first attempt fails. A halt results if both attempts fail.

A loop is used to clear all group-mark word-marks out of core, except for one in position 15998, replacing them with word-marked colons (a colon is a 5-8 card code). All of core is then written to sectors 000000 through 00176 of the Operating System Pack.

The operands of the macro are set up in locations 15965 to 15997. The first sector of the DLØDR program (KC06A) is again read, and control is transferred to it.

1401 O. S. II
I. D. # KF07A
I. D. # KF10A
I. D. # KF11A
I. D. # KF12A

PROGRAM:	DSKIØ/IØGET/IØPUT/IØSK
MACHINE:	IBM 1401
LANGUAGE:	AUTØCØDER
SUPERVISOR PROGRAM:	1401 Operating System II
PROGRAMMER:	Systems Division
DATE COMPLETED:	July 1, 1968

PURPOSE:

The IØGET macro, when executed, causes a disk record to be read into core storage from a specified disk address.

The IØPUT macro may be utilized to write data from core storage to disk storage.

The IØSK macro, when executed, initiates a disk seek. Since direct seeks are done by IØGET and IØPUT if necessary, the IØSK macro is not required unless the programmer wishes to overlap processing time with access-movement time.

All three macros generate linkage to the DSKIØ inflexible library routine, which performs the actual read/write/seek operation and error checking.

DSKIØ offers the facility of disk error checking with 10 retries.

RESTRICTIONS:

DSKIØ requires about 548 core positions, while IØCS can take anywhere from the same number to several times that number (based on the number of DTF's) for random disk processing. Both IØCS and DSKIØ use the direct seek special feature; however, DSKIØ uses it automatically. DSKIØ will allow processing of records in either the move or load-mode, while IØCS can only handle records in the move-mode. Cylinder overflow is not handled by DSKIØ; IØCS will process a record block that overflows from one cylinder to the next.

DSKIØ provides less facilities than IØCS and thereby greater flexibility for the programmer. The programmer must provide for the following:

a) header and trailer label creation and/or checking, if necessary.
b) blocking/deblocking and padding, if necessary.
c) address calculation if the disk control field is to be changed.

Time, Core Requirements:

Execution time for DSKIØ varies from 10 to 175 milliseconds, depending on the distance to be seeked, rotational location, and the size of the data records.

Execution time for IØSK varies from 50 to 250 milliseconds, depending on the number of tracks to be seeked, but allows for processing overlap of up to 200 milliseconds.

Core requirement is between 20 and 44 core positions for each IØGET, IØPUT, or IØSK macro. The DSKIØ routine requires 548 core positions. DSKIØ is included only once in the object program.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

The referenced disk drive(s) should be operable.

Processing/Method:

A detailed description of I/Ø macro processing method and linkage appears below in the paragraph on Processing Method -- Technical Description.

USAGE:

Operands:

The IØGET and IØPUT macros require Operand 1; -- Operands 2 through 8 are optional.

The IØSK macro uses only Operands 2, 3, 6, and 8.

1. Mode

   This operand should be either an M, which will cause 100-character sectors to read/written without word-marks, or an L, which will cause 90-character sectors to read/written with word-marks.

2. Drive Code

   This operand should be either 0 or 2, depending upon which disk drive is to be referenced. If this operand is omitted, the value of 0 will be assumed.

3. Sector Address

   This operand gives the initial (or only) sector address for the first sector to be read, written, or seeked, 00000 through 99999. This sector address may, in turn, be modified by the program. See operands 6 and 8, and Processing Method -- Technical Description.

4. Sector Count

   This operand gives the initial (or only) value of the number of sectors to be read/written/seeked; it may be modified by the program.

5. Input/Output Area

   This operand specifies the left most end of the area which the disk record will be read into or written from. This area should be long enough to hold the largest record expected to be read/written in the file. If not included, the program must fill in the address. See Processing/Method.

6. Out-Of-Line Disk Control Field Address

   This operand specifies the location of the ten-digit field where the out-of-line disk control field is located. It should be the label of a 10-character DCW or area-defining literal, or X1, X2, or X3 adjust tag. The IØGET, IØPUT, or IØSK macro will move this disk-control field into the generated linkage, if the out-of-line disk-control field address is specified.

7. Group-Mark Word-Mark Load

The DSKIØ routine checks the wrong length record indicator and thus the input or output area must have a group-mark-word-mark in the proper location following the data area. Include the GMWM as operand 7 if it is desired that IØPUT or IØGET generate the instructions to load a group-mark-word-mark to the proper location. If this feature is used, operands 1, 4 and 5 must all be present. If this feature is not desired, omit operand 7 entirely.

8. Generated Disk-Control Field Label

This operand, when used, will cause the included label (distinct from any other labels in the user's program), to reference the right end of the 10-digit generated disk control field.

Control Cards:

IØGET, IØPUT, IØSK, and DSKIØ require no control cards.

Input:

The only input to these macros is the record on disk.

OUTPUT:

Aside from the core storage record, and the error message (see below), these macros produce no output.

ERROR MESSAGES AND HALTS:

When a disk error is detected, the message:

```
**ERRØR**
DISK INDIC.n
```

is typed, where n is:

a) N - access inoperable.
b) W - wrong length record check.
c) X - unequal address compare.
d) V - validity check.

and a halt occurs.

OPERATING PROCEDURES:

When a disk error halt occurs, the operator should make certain that the disk drive is operable, and that the referenced file is on the correct drive, or else abort the run.

End-Of-Job, Post-Processing:

Control is transferred to the next sequential instruction after the appearance of the IØGET, IØPUT, or IØSK macro, unless an error condition occurs.

PROCESSING METHOD -- TECHNICAL DESCRIPTION:

The linkage generated by IØGET/IØPUT/IØSK consists of:

a) a load of GMWM (if operand 7 is included) to the proper position determined by the number of sectors to be used and the mode.

b) a move of the out-of-line disk control field, if operand 6 is included.

c) a branch to the start of the DSKIØ routine.

Processing in DSKIØ normally consists of:

a) storing the return address.
b) zeroing of the attempt counter.
c) storing the disk control field.
d) execution of the read-write operation.
e) testing for an error condition.
f) exit to location following the IØGET or IØPUT generated code.

If a disk error is encountered, DSKIØ

a) adds to the attempt counter.
b) retries the operation.
c) if the attempt counter is up to 10 tries, an error zeros the attempt counter and halts.

Sample linkage:

```
      GET   IØGET L,2,400,10,AREA,ØUTDCF,GMWM,DCF

      GET   EQU   *+1
            SW    *+13              Generated if load
            LCA   *+6,AREA+900      GMWM operand is used.
            B     *+3
            DCW   @  @
            MCW   ØUTDCF,*+20       Generated if operand 6 is used.
            B     DSKIØ
            DCW   @L1R@             load mode read operation.
            DSA   AREA              input area.
      DCF   EQU   *+10              generated if operand 8 is used.
            DCW   @20000400010@     disk control field.
```

PROGRAM: EXIT
MACHINE: IBM 1401
LANGUAGE: AUTØCØDER
SUPERVISOR PROGRAM: 1401 Operating System II
PROGRAMMER: Systems Division
DATE COMPLETED: July 1, 1968

PURPOSE:

The EXIT macro, when executed, returns control from a user program to the 1401 Operating System inter-job supervisor (KE01A). A message is typed by that program, and the operator can enter the ending time of day and computer clock time for the job, as well as other information. UWMCC programming standards indicate that the EXIT macro must be included as the end-of-job action for every AUTØCØDER or CØBØL program written at UWM, unless the operational details of the program preclude it, as when the end-of-job time is uncertain.

RESTRICTIONS:

Time, Core Requirements:

Execution of an EXIT macro takes less than 2 seconds.

If a FETCH or LINK macro has preceded the EXIT macro in the source program, core requirement is 25 positions. Otherwise core requirement is 129 positions.

PROGRAM PROCEDURE:

Set-Up, Previous Processing

The Operating System Pack should be on one of the 1311 disk drives at execution time. If it is not, the system halts and should be started when the pack is mounted. No special previous processing is required.

Processing/Method:

There are two alternate forms of the generated code constituting the EXIT macro. The form generated when a LINK (KF14A) or FETCH (KF09A) macro has preceded the EXIT in the source program is simply a LINK to the inter-job supervisor, MØNITR (KE01A). The form generated if neither a FETCH nor a LINK has previously appeared in the source program is substantially an ALINK macro to address 19640. An attempt is made to read the first sector of the loader into the card read area, from module 1. If this fails, an attempt is made on module 0. If both attempts have failed the system halts. When either attempt succeeds, control is transferred to the instructions read from disk.

USAGE:

Operands:

The EXIT macro uses no operands. The macro itself investigates each drive in turn for presence of the Operating System Pack. "Log-out" protection is afforded by the operator's ability to supply ending times to the inter-job supervisor, and must be supported by policy which makes such entries mandatory.

Control Cards:

The EXIT macro requires no control cards.

Input:

The EXIT macro requires no input.

OUTPUT:

EXIT itself produces no output. However the inter-job supervisor, MØNITR (KE01A), which is called by the EXIT macro, produces the console message

>> END-ØF-JØB

and, possibly, the printed end-of-job page as well as other output as described in the write-up of that program.

ERROR MESSAGES AND HALTS:

Only the standard system halts and messages may occur during execution of an EXIT macro.

OPERATING PROCEDURES:

Aside from the fact that the correct Operating System Pack must be mounted if a halt occurs, and the halt then started, no special operating procedures need be followed as far as the macro itself is concerned. Entry of the 'TM' and 'CC' codes should be made as soon as the end-of-job message is typed.

End-Of-Job, Post-Processing:

All of the previous paragraphs deal with end-of-job processing, since that is the purpose of the macro. Further information on the subject is given in the write-up on the inter-job supervisor, MØNITR (KE01A).

1401 O. S. II
I. D. # KF09A
I. D. # KF14A

PROGRAM:	FETCH/LINK
MACHINE:	IBM 1401
LANGUAGE:	AUTØCØDER
SUPERVISOR PROGRAM:	1401 Operating System II
PROGRAMMER:	Systems Division
DATE COMPLETED:	July 1, 1968

PURPOSE:

The FETCH macro, when executed, causes a program to be loaded into core from the 1401 Operating System Library. Control is then returned to the calling program, at the next sequential instruction after the appearance of the FETCH macro.

The LINK macro, when executed, causes a program to be loaded in the same fashion as the FETCH macro. Control is then transferred to the called program block at the "execution address" specified when the program block was DLØADed.

Loading procedures for both macros are controlled by the program parameters established for the program block at DLØAD time.

This document is complemented by the descriptions of LØADR (KF15A) and LØDER (KF16A), the closed routines which are called by the LINK and FETCH macros.

RESTRICTIONS:

Time, Core Requirements:

The generated code constituting the LINK and FETCH macros requires less than one millesecond of execution time, and occupies between 11 and 26 core positions.

The routines execute in between .1 and 2.0 seconds. The loader of the called program may require between .3 and 30.0 seconds of execution time, depending on the length of the program and whether group-mark word-marks are restored. LØADR requires 865 core positions. LØDER requires 940 positions.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

The Operating System pack must be on one of the 1311 disk drives at execution time. The program to be called into core should have been previously loaded onto the Operating System Library by either the DLØDR program (KC06A) or the program loading service of DUP (KC09A).

The programmer may write a LINK or FETCH macro at any point in his processing.

Processing/Method:

A detailed description of LINK and FETCH operation is found below in the paragraph on Processing Method -- Technical Description.

USAGE:

Operands:

The FETCH and LINK macros use two operands -- Operand 1 and Operand 3. Both are optional. Operand 2 is not used, for reasons of compatibility with previous versions.

1. Program Name (optional)

   This operand, if present, gives the location of the name of the program to be called into core. It may be either a 6-character literal enclosed in quotes, containing the name left-justified, or the label of a 6-character DCW containing the name left-justified.

   If the operand is omitted, the programmer must provide code, before the macro, to:

   a) move the program name to LØADRA or LØDERA
   b) move either a dollar sign (indicating a LINK), or an address at which execution is to be begun after loading, into LØADRX or LØDERX.

   This capability provides for multiple-entry-point subroutines.

2. Not used.

3. Program Common (optional)

This operand, if present, gives the location of the program common field to be passed to the called program. It may be a 9-character literal enclosed in quotes or the right-end label of a 9-character DCW. The contents of the field are determined by the program common conventions of the called program. One would not use this operand in calling a program which does not use program common.

If the operand is omitted, the program common field used is blank.

Control Cards:

Neither LINK nor FETCH require control cards.

Input:

Neither LINK nor FETCH use input.

OUTPUT:

Neither LINK nor FETCH produce output.

ERROR MESSAGES AND HALTS:

Only the standard system halts and messages may occur during execution of LINK and FETCH.

OPERATING PROCEDURES:

No special operation procedures need be specified for LINK and FETCH. See the Operating Procedures for the closed routines LØADR (KF15A) and LØDER (KF16A).

End-Of-Job, Post-Processing:

On completion of loading:

a) FETCH returns control to the calling program.
b) LINK transfers control to the loaded program block.

See the paragraphs on Purpose and Processing Method -- Technical Description.

PROCESSING METHOD -- TECHNICAL DESCRIPTION:

Permanent macro switches are used to determine which of the two closed routines LØADR or LØDER are included in the program, that only one of these is included, and that this one occurs only once in the program regardless of literal origins. If an IØGET, IØPUT or IØSK macro has occurred in the program previous to the LINK or FETCH the LØADR routine is included and the generated code references that routine. If none of the disk I/Ø macros has occurred, the LØDER routine is included and the generated code references that routine. Thus, if the programmer is planning to use the disk I/Ø macros anyway, it is to his advantage to arrange his program with at least one IØGET, IØPUT or IØSK previous to the first appearance of a LINK or FETCH, since the LØADR routine uses less core than LØDER.

The following paragraph describes the LINK and FETCH transfer to the closed routines. Further explanation will be found in the document on LØADR (KF15A) and LØDER (KF16A).

If operand 1 is present in either macro, the program name is moved to LØADRA or LØDERA. If operand 1 is present in a LINK, a dollar sign is moved to LØADRX or LØDERX. If operand 3 is present in either macro, the program common field is moved to LØADRZ or LØDERZ; otherwise, blanks are moved. A FETCH macro branches to LØADR or LØDER, a LINK macro branches to LØADR+4 or LØDER+4.

PROGRAM: LE
MACHINE: IBM 1401
LANGUAGE: AUTØCØDER
SUPERVISOR: 1401 Operating System II
PROGRAMMER: Systems Division
DATE COMPLETED: July 1, 1968

PURPOSE:

The LE (look-up-equal) macro provides a means of searching a table arranged in a standard fashion for a table entry having a key equal in value to a specified argument. Table entries must be contiguous. They may vary in length, provided the key is at the right end and is of the same length as the argument in all entries. This macro simulates the operation of the IBM 1410 hardware instruction having the same AUTØCØDER mnemonic.

RESTRICTIONS:

The rightmost position of a key must not contain a record mark in any entry of a table scanned by an LE macro.

Time, Core Requirements:

Execution time varies according to the number of entries that must be scanned before either a match or the end of the table is encountered, and according to the length of the entries and the length of the argument.

Core requirement is 50 positions.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

A table to be searched by an LE macro must be arranged in the following fashion:

a) the order of the table is "right to left": that is, the first entry occupies the rightmost (highest addressed) positions of the table, and the end of the table is marked to the left of the last entry.

b) each complete table entry, including key and functional (return) value, must have a word-mark at its leftmost position and no other word-marks within it.
c) the rightmost (highest addressed) n positions of each table entry must contain the key (the characters to be compared to the argument of the macro), where n is the number of characters in the argument.
d) the functional (return) value of each table entry should be immediately to the left of the key for that entry. The functional (return) value may be as long as desired, but must be present in each entry (at least one character in length).
e) the end of the table must be defined by a record mark (0-2-8 card code) immediately to the left of the last entry. Thus, the rightmost position of a key may not contain a record mark in any table entry. The macro operates correctly for a null table (one consisting of only a record mark).

No special computer set-up is needed when using the LE macro.

Processing/Method:

The first operations in the generated code of the LE macro store the address of the table in the succeeding instructions. Then, for each table entry:

a) a branch to the failure exit is performed if a record mark is detected in the rightmost position of the apparent "entry".
b) the comparison of the argument to the key is made.
c) the address of the functional (return) value is put into the match address location.
d) addresses are adjusted inside the macro for the possible next comparison.
e) if the comparison was unequal, control is transferred back to step a) for the next entry.

USAGE:

Operands:

The LE macro uses four operands, all of which are required, as follows:

1. Table Location

This operand gives the location of the right end of the first entry in the table. It may be symbolic or actual, indexed and/or adjusted.

2. Argument Location

   This operand gives the location of the right end of the argument for which the table is to be searched. It may be symbolic or actual, indexed and/or adjusted.

3. Match Address Location

   This operand gives the location of the right end of a 3-character field into which the right end address of the functional (return) value is to be stored if a match occurs (if a key equal to the argument is found).

4. Failure Branch Address

   This operand gives the address to which control is to be transferred if the entire table is scanned without a match.

Control Cards:

The LE macro requires no control cards.

Input:

The LE macro requires no input.

OUTPUT:

The LE macro produces no output.

ERROR MESSAGES AND HALTS:

No halts or messages may occur during execution of LE.

OPERATING PROCEDURES:

No special operating procedures need be specified.

End-Of-Job, Post-Processing:

On completion of the execution of LE, if the argument matched the key of an entry in the table, control is transferred to the instruction following the LE, and the address of the functional (return) value in that entry is in the location specified by operand 3. It is common for operand 3 to specify an index register, to facilitate examination of the value by following instructions.

If no match occurred, control is transferred to the instruction specified by operand 4, and the location specified by operand 3 contains the address of the functional value of the last entry in the table.

1401 O. S. II
I. D. # KF15A
I. D. # KF16A

PROGRAM: LØADR/LØDER
MACHINE: IBM 1401
LANGUAGE: AUTØCØDER
SUPERVISOR PROGRAM: 1401 Operating System II
PROGRAMMER: Systems Division
DATE COMPLETED: July 1, 1968

PURPOSE:

The closed library routines LØADR and LØDER, when included in a program and executed, cause a program to be loaded into core from the 1401 Operating System library. One of these routines is automatically included in a program by use of a LINK (KF14A) or FETCH (KF09A) macro. The loading itself is accomplished through the disk-stored loader of the called program, and is controlled by the program parameters established for that program at DLØAD time.

The only difference between the two routines is the fashion in which they perform disk input/output. LØADR utilizes the DSKIØ routine, while LØDER contains its own disk I/Ø instructions. Thus LØADR is shorter if DSKIØ is used by the program in its normal processing, but longer if DSKIØ must be specially included for this one purpose. LØADR uses the direct seek capability of DSKIØ, LØDER seeks on a return-to-home basis. Hereafter, references will be given in terms of the LØADR routine, since the processing done by the two routines is identical. The user should note that labels used by each are identical in their sixth character (e.g., LØDERX has the same role as LØADRX).

RESTRICTIONS:

Time, Core Requirements:

The closed routine requires between .1 and 2.0 seconds of execution time. The loader of the called program may take between .3 and 30.0 seconds to execute, depending on the length of the program and whether group-mark word-marks are restored. LØADR requires 865 positions; LØDER requires 940 positions.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

The Operating System pack should be on one of the 1311 disk drives at execution time. The program to be called into core should have been previously loaded onto the Operating System library by means of the DLØDR program (KC06A) or the program loading service of DUP (KC09A).

Processing/Method:

A detailed description of LØADR processing is found below in the paragraph on Processing Method -- Technical Description.

USAGE:

Calling Sequence:

A program common field should be moved into LØARDZ. The name of the program to be called should be moved into LØADRA. If the program block is to be executed at the "execution address" specified at its DLØAD time, a dollar sign should be moved to LØADRX and a branch to LØADR+4 performed. If execution is to continue at the next sequential instruction following the linkage to LØADR, a branch to LØADR should be performed. If execution is to be taken up at any other location, this location should be moved into LØADRX and a branch to LØADR+4 performed. If LØDER is to be used, substitute "LØDER" for "LØADR" in the aforementioned labels.

Control Cards:

Neither LØADR nor LØDER require control cards.

Input:

Neither LØADR nor LØDER require input.

OUTPUT:

The closed routine itself produces no output. However, if the program name requested is not on disk, output from the PLIST program (KC08A) results.

ERROR MESSAGES AND HALTS:

Only the standard system halts and messages may occur during execution of LØADR or LØDER.

OPERATING PROCEDURES:

Aside from the fact that the correct Operating System pack should be mounted if a halt occurs, and the halt then started, no special operating procedures need be specified.

End-Of-Job, Post-Processing:

Control is transferred to the address described in the paragraphs on Calling Sequence and Processing Method -- Technical Description.

PROCESSING METHOD -- TECHNICAL DESCRIPTION:

The drive on which the Operating System pack resides is determined by reading the first three sectors of the program table. Module 1 is tried first; then module 0, if the first attempt fails. A halt results if both attempts fail.

The program table is searched for the specified name, three sectors (20 entries) being read from disk at a time. If the name is not found, or if it references a non-program entry, a diagnostic results and the PLIST program (KC08A) is called instead.

The Operating System communications sector is read and the IØCS date and normal date are saved. The program header of the specified program is read and its identification checked. If the check fails, a diagnostic is produced (the same as described in the preceding paragraph), and the PLIST program (KC08A) is called. If the check succeeds, the usage count and last use date are updated and the header is written back to disk. If LØADRX contains a dollar sign, the execution address in the header sector is moved into LØADRX. The last 2 sectors of the 5-sector loader are then read, updated with the program common field, IØCS date, and execution address, and written back to disk.

The first sector of the loader of the specified program is then read into the card read area, and control is transferred to it.

PROGRAM: ØSINF
MACHINE: IBM 1401
LANGUAGE: AUTØCØDER
SUPERVISOR PROGRAM: 1401 Operating System II
PROGRAMMER: Systems Division
DATE COMPLETED: July 1, 1968

PURPOSE:

The ØSINF macro, when executed, causes the Operating System communications sector to be read from disk, providing the user's program with the following information as established at the beginning of the job:

job number and name
man number and name
date
time of day
day name
IØCS date
machine clock reading
function code and name

RESTRICTIONS:

The ØSINF macro may only be executed once.

Time, Core Requirements:

Time required for execution of the ØSINF macro is negligible: less than 50 milliseconds. Core requirement is 207 positions.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

The Operating System pack must be present on one of the 1311 disk drives at execution time. No special set-up or previous processing is required. The programmer may write an ØSINF macro at any point in his processing.

Processing/Method:

When the ØSINF macro is executed, sector 399620 of the Operating System pack is read, in the move-mode. Module 1 is tried first; then module 0, if the first attempt fails. A halt results if both attempts fail. The 100-position area into which the "communications sector" is read contains the following labeled, word-marked fields, which the instructions following the ØSINF macro may reference:

Label	Length	Content
JØBSC	7	job number
JNMSC	13	job name
MANSC	3	man number
MNMSC	13	man name
DATSC	10	date, time of day, in the format: yymmddhhtt (i.e. year-month-date-hour-minute, with the hour field in 24-hour time).
DNMSC	9	day name
IØDSC	5	IØSC date
CCTSC	6	machine clock reading
FCTSC	2	function code
FCNSC	13	function name

Elaboration on the meaning and possible contents of the more obscure of these fields can be found in the document on the Inter-Job Supervisor, MØNITR (KE01A).

USAGE:

Operands:

The ØSINF macro uses no operands.

Control Cards:

The ØSINF requires no control cards.

Input:

The ØSINF macro uses no input.

OUTPUT:

The ØSINF macro produces no output.

ERROR MESSAGES AND HALTS:

Only the standard system halts and messages may occur during execution of the ØSINF macro.

OPERATING PROCEDURES:

No special operating procedures need be specified.

End-Of-Job, Post-Processing:

The instructions following the ØSINF macro can reference the labels described above in the paragraph on Processing/Method.

PROGRAM:	ØSLØC
MACHINE:	IBM 1401
LANGUAGE:	AUTØCØDER
SUPERVISOR PROGRAM:	1401 Operating System II
PROGRAMMER:	Systems Division
DATE COMPLETED:	July 1, 1968

PURPOSE:

The ØSLØC macro, when executed, causes the Operating System program table to be scanned for the presence of a particular program name. If the program block is present, the disk address of its first sector is returned as a result; if it is not, the result field is filled with nines.

RESTRICTIONS:

No indication is returned to a program using the ØSLØC macro of

a) whether the name (operand 1) referenced a non-program block, or
b) the mode of that block.

Time, Core Requirements:

Execution time required for the ØSLØC macro varies according to the length of the program table, and the position of the name within the table. In any case, execution should require less than .5 seconds.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

The Operating System pack should be on one of the 1311 disk drives at execution time. The programmer may use an ØSLØC macro at any point during his processing.

Processing/Method:

When the ØSLØC routine is first executed, the drive on which the Operating System pack resides is determined by reading the first three sectors of the program table. Module 1 is tried first; then module 0, if the first attempt fails. A halt results if both attempts fail.

The program table is searched for the specified name, three sectors (20 entries) being read from disk at a time. If the name is found, the drive code and address of the first sector of the program block are moved to the result field; if not, the field is filled with nines.

ØSLØC is an in-line macro: control should enter at its first position, and is transferred to the next sequential instruction following its appearance in the mainline.

USAGE:

Operands:

The ØSLØC macro uses operands 1 and 2; both are required.

1) Program Name

This operand gives the location of the name for which the program table is to be scanned. It may be a 6-character literal enclosed in quotes containing the name, left-justified, or the label of a 6-character DCW containing the name, left-justified.

2) Result Field

This operand gives the location of the result field: it should be the label of a 7-character DCW or area-defining literal. If the program specified by operand 1 is in the Operating System library, the Operating System drive code is inserted in the field, followed by the actual (380,000 file-protected) address of the first sector of the program block. If it is not present the field is filled with nines.

Control Cards:

The ØSLØC macro requires no control cards.

Input:

THE ØSLØC macro uses no input.

OUTPUT:

The ØSLØC macro produces no output.

ERROR MESSAGES AND HALTS:

Only the standard system halts and messages may occur during execution of the ØSLØC macro.

OPERATING PROCEDURES:

Aside from the fact that the correct Operating System pack should be mounted if a halt occurs, and the halt then started, no special operating procedures need be specified.

End-Of-Job, Post-Processing:

Control is transferred to the next sequential instruction following the ØSLØC generated code. There, the programmer can write code which tests for the presence of nines in the result field, and may wish to write code to read the header sector of a program, the actual sectors of a program, or the data sectors of a non-program block. In a program, the disk address returned is that of the header sector, and the first sector of the program itself is located at that address plus 6. In a non-program block, the disk address returned is that of the first data sector.

1401 O. S. II
I. D. # KF19A
I. D. # KF20A

PROGRAM:	RDTPM/TPERR
MACHINE:	IBM 1401
LANGUAGE:	AUTØCØDER
SUPERVISOR PROGRAM:	1401 Operating System II
PROGRAMMER:	Systems Division
DATE COMPLETED:	July 1, 1968

PURPOSE:

The RDTPM macro, when executed, causes a tape record to be read into a specified area of core storage, from a specified tape unit. It accomplishes this by setting up linkage to the TPERR inflexible library routine, which performs the actual tape operation and error checking. RDTPM and TPERR offer the following facilities:

a) tape error checking with 50 retries. Control is transferred to the user's error routine if the error persists.
b) storage of the address of the last position brought into core (the group-mark).
c) transfer to a user routine on an end-of-reel/file condition (tape mark or reflective spot sensed).

RESTRICTIONS:

RDTPM and TPERR require less core than IØCS, but provide less facilities and thereby greater flexibility for the programmer. The programmer must provide facilities for:

a) tape control (RWD, RWU, etc.), except for error-backspacing.
b) wrong-length record checking, if desired.
c) header and trailer checking and block counts, if desired.
d) deblocking and padding if necessary.
e) variable-length record group-mark.
f) distinction between end-of-reel and end-of-file conditions.

Load-mode operation can be accomplished with RDTPM and TPERR by moving an "L" into NEXT-8, and binary operation can be accomplished by moving a "B" into NEXT-6, where NEXT is the location following the RDTPM generated code.

Time, Core Requirements:

Processing rate is dependent on the speed of the tape units and the density and record-length of the file. RDTPM and TPERR have less "execution overhead" than IØCS tape routines, but this time is not significant in either case.

RDTPM generated code requires 40 positions; the TPERR routine requires 86 positions plus a 3-character literal. TPERR is included only once in the object program.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

There must be an input tape on the drive referenced by RDTPM.

Processing/Method:

A detailed description of RDTPM processing and linkage appears below in the paragraph on Processing/Method -- Technical Description.

USAGE:

Operands:

Operands 1 through 5 are required for the RDTPM macro: no other operands are used.

1) Tape Unit

This operand should be a single-digit number, 1 through 6 inclusive.

2) Input Area

This operand specifies the left end of the input area into which the next record on the drive specified by operand 1 is to be read. The area should be followed by a group-mark with a word-mark, and be long enough to hold the largest record expected on the file, plus one extra position for the end-of-record group-mark.

3) Location for Group-Mark Address

This operand specifies the location of a three-digit field into which the address of the group-mark following the record will be placed. It should be the label of a 3-character DCW, an area-defining literal, or X1, X2, or X3. This field may be checked for correct record length by instructions following the RDTPM macro.

4) End-Of-Reel/File Address

This operand specifies the label of the first instruction in the user's end-of-reel/file routine. This address is branched to when a tape-mark or a reflective spot is sensed.

5) Error Address

This operand specifies the label of the first instruction in the user's error recovery routine. This address is branched to if 50 retries to read the record all encounter tape errors. When the error routine is entered, the tape is positioned just past the error record.

Control Cards:

The RDTPM macro requires no control cards.

Input:

The only input consists of the move-mode BCD record to be read from tape.

OUTPUT:

RDTPM and TPERR produce no output.

ERROR MESSAGES AND HALTS:

No halts or messages occur during execution of RDTPM and TPERR. The user may wish to provide halts or messages in the error or end-of-file routines.

OPERATING PROCEDURES:

No special operating procedures need be specified.

End-Of-Job, Post-Processing:

With variable-length records, the programmer should insure against subsequent-record cutoff by moving a blank, a record mark, etc., into the group-mark location.

In the end-of-file routine, the programmer should rewind the tape unless it is to be read further.

PROCESSING METHOD -- TECHNICAL DESCRIPTION:

The linkage generated by RDTPM consists of:

a) movement of an appropriate read-tape instruction into TPERR.
b) storage of the address of the "group-mark address location"into TPERR.
c) storage of the tape-error and end-of-reel/file addresses into TPERR.
d) a branch to the start of the TPERR routine.

Processing in TPERR normally consists of:

a) storage of the return address.
b) zeroing of an error counter.
c) execution of the actual tape read instruction.
d) storage of the group-mark address into the specified location.
e) testing for an end-of-file condition.
f) testing for an error condition.
g) exit to the location following the RDTPM generated code.

If a tape error is encountered, TPERR:

a) adds to an error counter.
b) exits to the error address provided if the error counter has reached 50.
c) backspaces the tape.
d) proceeds to normal-processing step c) above.

PROGRAM:	VLCHK
MACHINE:	IBM 1401
LANGUAGE:	AUTØCØDER
SUPERVISOR PROGRAM:	1401 Operating System II
PROGRAMMER:	Systems Division
DATE COMPLETED:	July 1, 1968

PURPOSE:

The VLCHK macro is an error checking routine for variable length tape records. VLCHK computes the proper last character address of the given variable length tape input record, and compares this address with the IØCS last character address for that record, transferring control to the user's error routine on an unequal compare.

RESTRICTIONS:

The maximum record length which can be checked by VLCHK is 1999 characters.

Time, Core Requirements:

Execution time is less than 50 milliseconds.

Core requirement varies from 25 to 68 core positions.

PROGRAM PROCEDURE:

Set-Up, Previous Processing:

The VLCHK macro may only be used after a variable length tape record has been read.

Processing/Method:

Two forms of the generated code constituting the VLCHK macro may be produced, depending on the length of the record length field. The address of the left end of the input area is moved to a sum box. This sum box is modified by use of the MA instruction which references the right most 3 digits of the length field. If the length of the record length field is 4 digits, code is generated to modify the sum box by an additional thousand.

USAGE:

Operands:

The VLCHK macro uses operands 1 through 4; no other operands are required.

1. Input Area

   This operand specifies the left end of the input area, into which the tape record was read.

2. Record Length

   This operand specifies the core location of the tape record length field. It may be symbolic or actual, indexed and/or adjusted.

3. Length of the Record Length

   This operand specifies the number of digits contained in the record length field. It may be either 3 or 4.

4. Error Address

   This operand specifies the label of the first instruction in the user's error recovery routine. If a wrong length record is read, VLCHK branches to this instruction.

Control Cards:

The VLCHK macro requires no control cards.

Input:

A variable length tape record must have been read before the VLCHK macro may be used.

OUTPUT:

The VLCHK macro produces no output.

ERROR MESSAGES AND HALTS:

No halts or messages may occur during execution of the VLCHK macro.

OPERATING PROCEDURES:

No operating procedures need be specified.

End-Of-Job, Post-Processing:

Control is returned to the next sequential instruction following the VLCHK macro unless the error condition has been detected, in which case control is transferred to the user's error routine (see Operands).

1401 O. S. II
I. D. # KF23A
I. D. # KF21A

PROGRAM:	WRTPM/TPERW
MACHINE:	IBM 1401
LANGUAGE:	AUTØCØDER
SUPERVISOR PROGRAM:	1401 Operating System II
PROGRAMMER:	Systems Division
DATE COMPLETED:	July 1, 1968

PURPOSE:

The WRTPM macro, when executed, causes a tape record to be written onto a specified tape unit, from a specified area of core storage. It accomplishes this by setting up linkage to the TPERW inflexible library routine, which performs the actual tape operation and error checking. WRTPM and TPERW offer the following facilities:

a) tape error checking, with 3 retries per tape location and 20 skip-and-blank sequences before transfer to the user's error routine.
b) transfer to a user routine on an end-of-reel condition (reflective spot sensed).

RESTRICTIONS:

WRTPM and TPERW require less core than IØCS, but provide less facilities and thereby greater flexibility (with greater effort) for the programmer. The programmer must provide facilities for:

a) tape control (RWD, RWU, etc.) except for error-backspacing.
b) writing of tape marks (WTM) or other end-of-file indication.
c) header and trailer checking and block counts, if desired.
d) blocking and padding, if necessary.
e) variable-length record group-mark erasure, if appropriate.

Load-mode operation can be accomplished with WRTPM and TPERW by moving an "L" into NEXT-8, and binary operation can be accomplished by moving a "B" into NEXT-6, where NEXT is the location following the WRTPM generated code.

Time, Core Requirements:

Processing rate is dependent on the speed of the tape units and the density and (average) record length of the file. WRTPM and TPERW have less "execution overhead" than IØCS tape routines, but this time is not significant in either case.

WRTPM generated code requires 33 positions; the inflexible TPERW routine requires 107 positions plus a 3-character literal. TPERW is included only once in the object program.

PROGRAM PROCEDURES:

Set-Up, Previous Processing:

At execution time there must be an output tape (with ring) mounted on the tape drive referenced by the macro.

Processing/Method:

A detailed description of WRTPM and TPERW processing and linkage appears below in the paragraph on Processing Method -- Technical Description.

USAGE:

Operands:

Operands 1 through 4 are required for the WRTPM macro; no other operands are used.

1) Tape Unit

This operand should be a single-digit number, 1 through 6 inclusive.

2) Output Area

This operand specifies the left end of the output area from which the record is to be written to the drive specified by operand 1. The last position of the physical record to be written should be followed by a group-mark with word-mark.

3) End-Of-Reel Address

This operand specifies the label of the first instruction in the user's end-of-reel routine. This address is branched to if a reflective spot is sensed.

4) Error Address

This operand specifies the label of the first instruction in the user's error recovery routine. This address is branched to after 20 skip-and-blank-tape sequences have been attempted, with 3 write attempts at each position, if all 60 write attempts produced error indications. This generally indicates either a malfunction of the tape unit or controller, or a reel of tape in extremely bad condition.

Control Cards:

Neither WRTPM nor TPERW require control cards.

Input:

Neither WRTPM nor TPERW use input.

OUTPUT:

The only output consists of the move-mode BCD record to be written to tape.

ERROR MESSAGES AND HALTS:

No halts or messages should occur in either WRTPM or TPERW. The programmer may wish to provide halts or messages in the error or end-of-file routines.

OPERATING PROCEDURES:

No special operating procedures need be specified.

End-Of-Job, Post-Processing:

With variable-length records, the programmer should insure against subsequent-record cutoff by moving a blank, record mark, or data field over the group-mark with word-mark used to terminate the tape-write.

In the end-of-reel routine, the programmer should rewind the tape unless records are to be written following the reflective spot.

After all records have been written, the user's program should execute one or two write-tape-mark (WTM) instructions.

PROCESSING METHOD -- TECHNICAL DESCRIPTION:

The linkage generated by WRTPM consists of:

a) movement of an appropriate write-tape instruction into TPERW.
b) storage of the tape-error and end-of-reel addresses into TPERW.
c) a branch to the start of the TPERW routine.

Processing in TPERW normally consists of:

a) storage of the return address.
b) zeroing of the skip counter.
c) zeroing of the attempt counter.
d) execution of the actual tape-write instruction.
e) testing for an error condition.
f) testing for an end-of-reel condition.
g) exit to the location following the WRTPM generated code.

If a tape error is encountered, TPERW:

a) adds to the attempt counter.
b) backspaces the tape.
c) if the attempt counter is not up to 3, proceeds to normal-processing step d) above.
d) adds to the skip counter.
e) exits to the error address provided if the skip counter is up to 20.
f) skips and blanks six inches of tape.
g) proceeds to normal-processing step c) above.

PROGRAM: WS
MACHINE: IBM 1401
LANGUAGE: AUTØCØDER
SUPERVISOR PROGRAM: 1401 Operating System II
PROGRAMMER: Systems Division
DATE COMPLETED: July 1, 1968

PURPOSE:

The WS macro provides a means of coding write-with-space-suppression instructions. The macro may be written with or without a branch address, and is in this and all other respects, identical to the write(W) instruction, except that an S is added as the D-character.

For further information on this macro, see:

IBM publication A24-3071,

Special Features Instructions
IBM 1401 Data Processing System
IBM 1460 Data Processing System
File No. 1401/1460-13.
Page I-49.

Appendix A

Standard System

Halts and Messages

The halts and messages listed below are standard for the 1401 Operating System, Version II. Details pertaining to operating procedures will be found in the document on the program during which the halt or message occurs.

Messages:

1. WAITING

   This message is produced by the Inter-Job Supervisor, MØNITR (KE01A).

2. **ERROR**
   DISK INDIC. n

   This message is produced by the DSKIØ macro, KF07A.

   n=N, access inoperable.
   n=W, wrong length record check.
   n=X, unequal address compare.
   n=V, validity check.

Halts:

A-Address	Source of Halt
804	MØNITR (KE01A)
111	EXIT macro (KF08A)

Note:

If the system comes to a hard halt during operation of DUP (KC09A), the Operating System Pack has been damaged.